Lookit

May 10, 2021

Overview

Here you will find detailed information about how to use Lookit to conduct developmental research online, as well as how to contribute to the codebase.

If you're new to Lookit, check out the *Getting started* guide. The best way to get up to speed on setting up your study is to work through the *tutorial*.

CHAPTER 1

What is Lookit?

Lookit is an online platform for developmental research. Families can sign up or log in, provide some basic information about their children, and then take part in studies from a variety of labs when it's convenient for them.

There's no videoconferencing or scheduling: parents and children simply do activities in the web browser, and information about the session (e.g., survey responses, timing, condition assignment) along with webcam video of the session is sent to the Lookit server.

Lookit can be used to collect looking measures from preverbal children as well as verbal responses, pointing, etc. from older children. Researchers can design, test, and manage their studies on the platform, including contacting participants; common tasks such as checking for informed verbal consent before accessing any data from a session are built into the Lookit research workflow.

Families may take part in studies from multiple labs over time. Having one central platform allows families to access many interesting studies for all the children in their family in one place, and researchers benefit from economies of scale in software development and recruitment.

1.1 Getting started

If you're a researcher interested in using Lookit to run your own developmental studies, here's how you get started!

For a 20-minute overview of how Lookit works, you can also read Jenna Croteau's excellent Introduction to Lookit document.

Do I have to do these in order?

No! These steps can all be completed in parallel. Except for joining the Slack workspace, please do that first in case you have questions.

1.1.1 A. Join the Slack workspace!

Please fill out this form for an invite.

Slack is our primary means of communication about new features, best practices, etc. It's also where you should go with any questions or technical support requests. There are a lot of participating researchers (over 400 at last count!) who may be able to help!

If you also want to receive updates about Lookit via email, please join the lookit-research list as well.

1.1.2 B. Legal/logistical steps

You can go ahead and create an account on Lookit to start preparing your studies right away, but to actually collect data from families on Lookit, your lab will need to review the Terms of Use and sign a one-time institutional agreement with MIT. Any studies you run will need to be approved by your own IRB, just like studies you run in person. For more detail, see *IRB and legal information*. Note that you only need **one agreement** per PI (not separate agreements for each student).

1. Review the Terms of Use and complete a short quiz.
2. Get an institutional agreement signed by an authorized signer for your institution.
3. Edit your IRB protocol to include online testing, or submit a new protocol for your proposed online study.

1.1.3 C. Familiarize yourself with the Lookit Working Groups

Although a Core Team at MIT provides a foundation for Lookit to function, a lot of great additional work is done by the community of researchers working together on Lookit to run studies. For example, one Working Group is focused on increasing our sample size and diversity, and another is focused on supporting new researchers as they learn how to use Lookit.

Please review the *descriptions of the current working groups*. At least one person from your lab (the more the merrier!) should fill out the survey to express preferences for working on particular topics; we'll then be in touch to help you join a group!

1.1.4 D. Create a lab on Lookit

You can go to https://lookit.mit.edu/exp/labs to create your own "lab" on Lookit. After creating your lab, you can go ahead and put studies in it and use it to manage access to your group's studies. However, you won't be able to submit these studies to run on Lookit until the lab is approved to test. This happens once

(a) you have a signed institutional agreement for the lab PI, and

(b) someone in the group has taken the terms of use quiz

1.1.5 E. Study setup steps

1. Complete the *Lookit tutorial* to get familiar with how the platform works and how to implement a study on it.
2. Figure out the details of how your study will work - counterbalancing, practice trials, etc. Draft the parent-facing instructions, record any audio and video needed (e.g., verbal instructions, voiceover/questions, demos), and collect your stimuli. See *advice here*.
4. Set up your study on Lookit and get it working just how you want it to! You'll need to set each of these fields and write and test your study protocol.
5. Gather *peer feedback* on your study to improve it; edit and iterate.

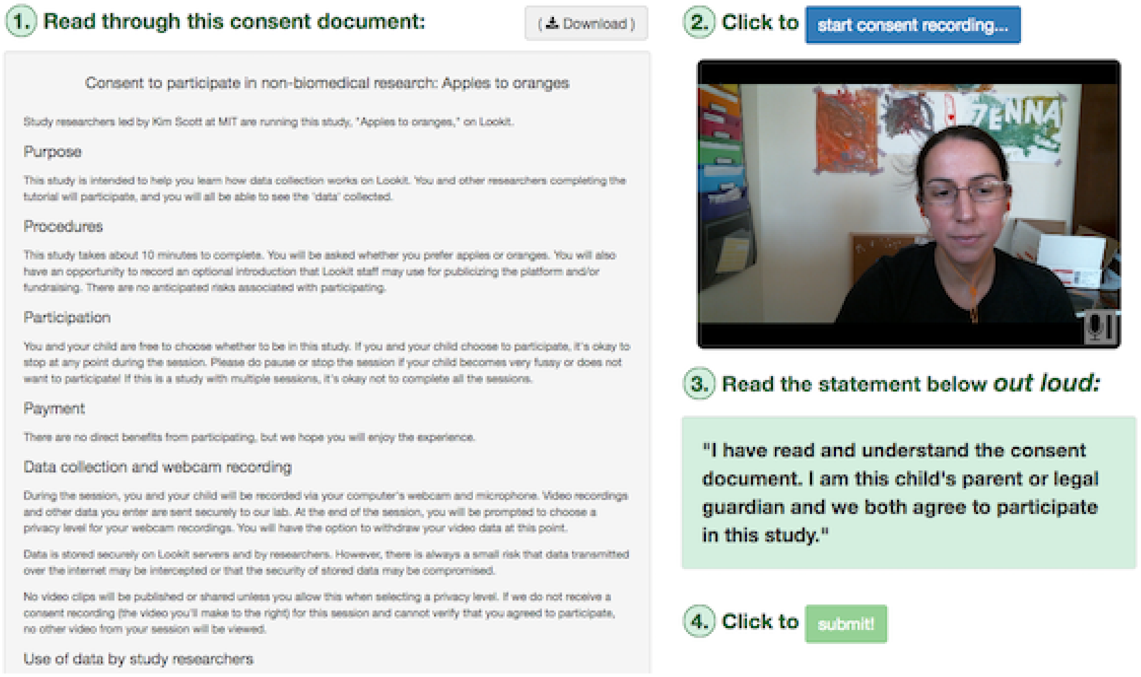
6. Submit your study for *internal review*. This process, like peer review at a journal, can take a bit of time to complete, and you might need to go through more than one revise-and-resubmit to get approval to run your study. To minimize the chances of repeated revise-and-resubmits, researchers are strongly encouraged to spend time polishing their studies as much as possible before submitting, including asking others to go through the study and provide feedback.
7. Once your study has successfully passed internal review, start data collection!
8. Put a link to Lookit on a parent-facing part of your lab or personal researcher website. This might include something like this: "Did you know that you can participate in our research from the comfort of your own home? Click here to check out Lookit, an online platform for developmental research. Families can participate in our studies on their home computers, any time they want!"

1.2 Features

Here are some of the features and advantages researchers using Lookit benefit from:

1.2.1 Common resources and processes

- A time-tested approach to **consenting families online** that has been approved by review boards at 10+ institutions.



1. Read through this consent document:

Consent to participate in non-biomedical research: Apples to oranges

Study researchers led by Kim Scott at MIT are running this study, "Apples to oranges," on Lookit.

Purpose

This study is intended to help you learn how data collection works on Lookit. You and other researchers completing the tutorial will participate, and you will all be able to see the 'data' collected.

Procedures

This study takes about 10 minutes to complete. You will be asked whether you prefer apples or oranges. You will also have an opportunity to record an optional introduction that Lookit staff may use for publicizing the platform and/or fundraising. There are no anticipated risks associated with participating.

Participation

You and your child are free to choose whether to be in this study. If you and your child choose to participate, it's okay to stop at any point during the session. Please do pause or stop the session if your child becomes very fussy or does not want to participate! If this is a study with multiple sessions, it's okay not to complete all the sessions.

Payment

There are no direct benefits from participating, but we hope you will enjoy the experience.

Data collection and webcam recording

During the session, you and your child will be recorded via your computer's webcam and microphone. Video recordings and other data you enter are sent securely to our lab. At the end of the session, you will be prompted to choose a privacy level for your webcam recordings. You will have the option to withdraw your video data at this point.

Data is stored securely on Lookit servers and by researchers. However, there is always a small risk that data transmitted over the internet may be intercepted or that the security of stored data may be compromised.

No video clips will be published or shared unless you allow this when selecting a privacy level. If we do not receive a consent recording (the video you'll make to the right) for this session and cannot verify that you agreed to participate, no other video from your session will be viewed.

Use of data by study researchers

2. Click to start consent recording...

3. Read the statement below *out loud:*

"I have read and understand the consent document. I am this child's parent or legal guardian and we both agree to participate in this study."

4. Click to submit!

- A **standard family interface** that minimizes friction when participating in studies from a variety of research groups.
- Access to a **shared userbase** of families with young children. Stored child and demographic data become available to you when a child participates in your study.

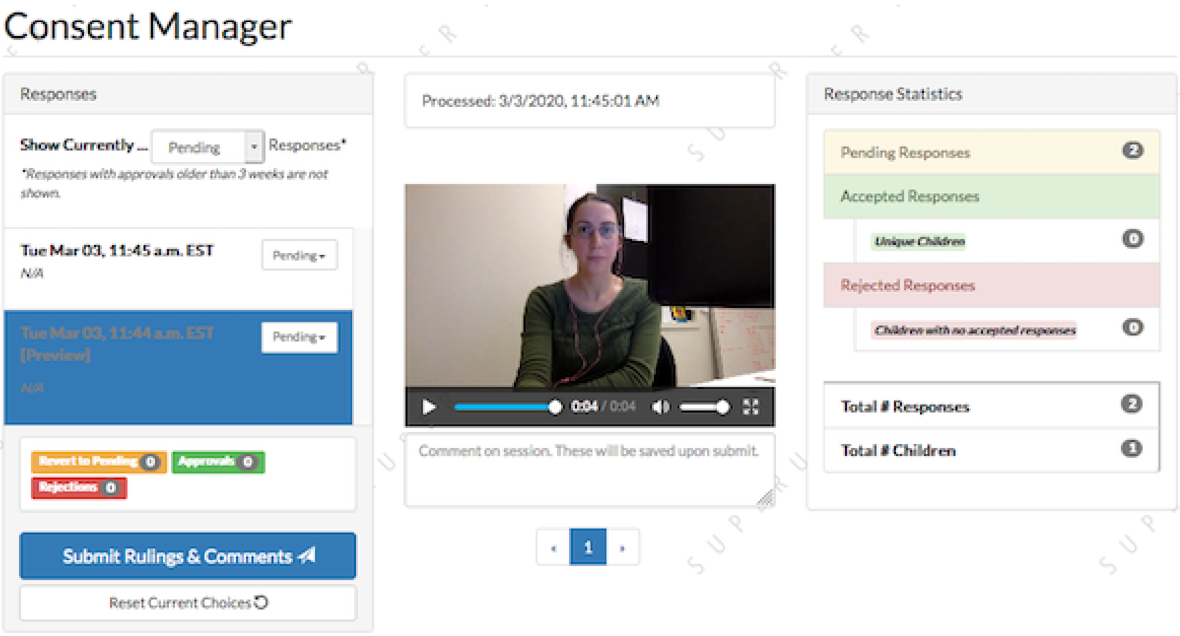
1.2.2 Community and documentation

- Active research **user community** and responsive **technical support**.

- 100% **open source**, forever. That means all of the source code that Lookit uses is publicly available, and you are free to extend and use it for your own purposes.
- **Free** to use.
- Extensive and actively maintained **documentation** (you're looking at it!), including a step-by-step tutorial to get started.
- *Community review and feedback* on studies prior to data collection, to improve data quality and ensure a high standard of family-friendly design that benefits all users.

1.2.3 Study management tools designed for developmental researchers

- Built-in **consent video management**: review consent videos in the web browser and confirm that the parent provided informed consent to participate. Information about this confirmation is stored on the Lookit server and only data from sessions with confirmed consent are available for viewing or download.



Consent Manager

Show Currently ... Pending Responses*

*Responses with approvals older than 3 weeks are not shown.

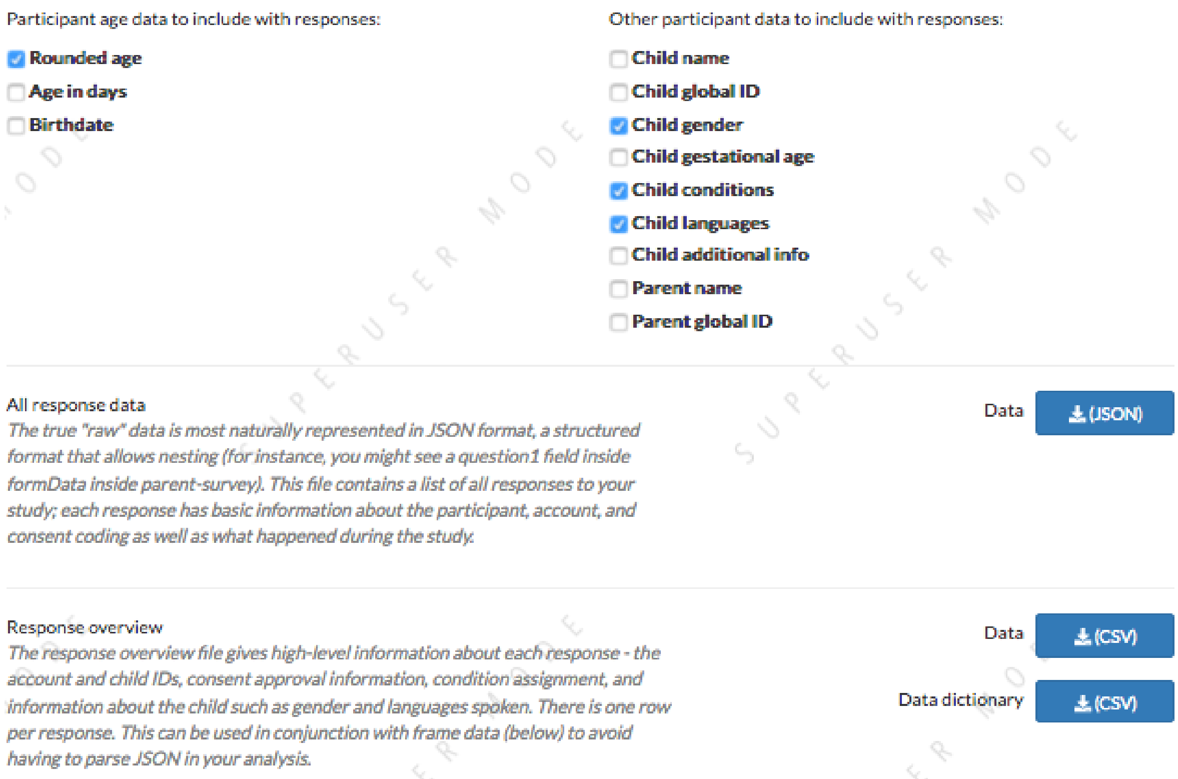
Tue Mar 03, 11:45 a.m. EST
N/A

Tue Mar 03, 11:44 a.m. EST [Preview]
N/A

Processed: 3/3/2020, 11:45:01 AM

Comment on session. These will be saved upon submit.

- Convenient **data downloads**, thoughtfully designed to protect participant data while maximizing potential for reuse and sharing.
  - Data dictionaries are provided for all CSV downloads to make it easy to share anonymized raw data upon publication.
  - Collect **video sharing preferences** in a standard format
  - Data are only available once consent is confirmed!



Participant age data to include with responses:

☑ Rounded age

☐ Age in days

☐ Birthdate

Other participant data to include with responses:

☐ Child name

☐ Child global ID

☑ Child gender

☐ Child gestational age

☑ Child conditions

☑ Child languages

☐ Child additional info

☐ Parent name

☐ Parent global ID

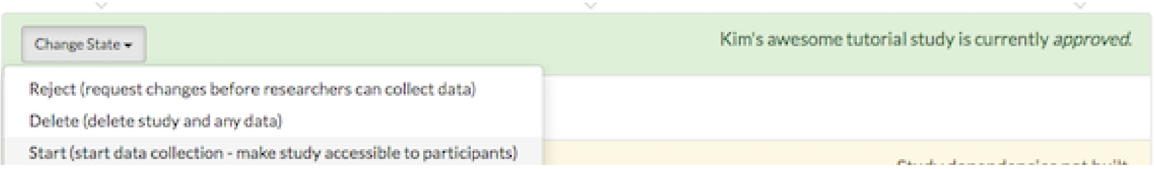
All response data

The true "raw" data is most naturally represented in JSON format, a structured format that allows nesting (for instance, you might see a question1 field inside formData inside parent-survey). This file contains a list of all responses to your study; each response has basic information about the participant, account, and consent coding as well as what happened during the study.

Response overview

The response overview file gives high-level information about each response - the account and child IDs, consent approval information, condition assignment, and information about the child such as gender and languages spoken. There is one row per response. This can be used in conjunction with frame data (below) to avoid having to parse JSON in your analysis.

- Set *study eligibility criteria* based on age, gestational age, gender, languages spoken, and other characteristics.
- **Contact participants** to invite them to another session of a longitudinal study, provide compensation, or ask clarifying questions. Email records for each study are accessible on Lookit and you can download them for your records.
- For **longitudinal studies**, determine which tasks and stimuli to use based on prior session data and child age if needed. We have run studies with as many as 12 longitudinal sessions within a few months!
- **Start and stop** data collection at any time.



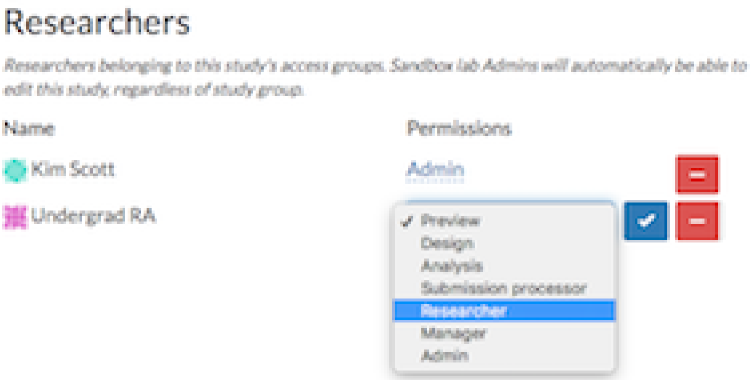
Kim's awesome tutorial study is currently *approved.*

Reject (request changes before researchers can collect data)

Delete (delete study and any data)

Start (start data collection - make study accessible to participants)

- Provide *feedback* to parents on study sessions.
- **Preview** exactly how your study will work ahead of starting data collection, including collecting sample data so you can set up analysis workflows ahead of time. You control which researchers can preview your study, and can share it by link with all Lookit researchers to get feedback if you choose.
- Create your own lab on Lookit to share studies among your lab members. **Easily collaborate** on studies with fine-grained permission roles for individual studies.



Researchers

Researchers belonging to this study's access groups. Sandbox lab Admins will automatically be able to edit this study, regardless of study group.

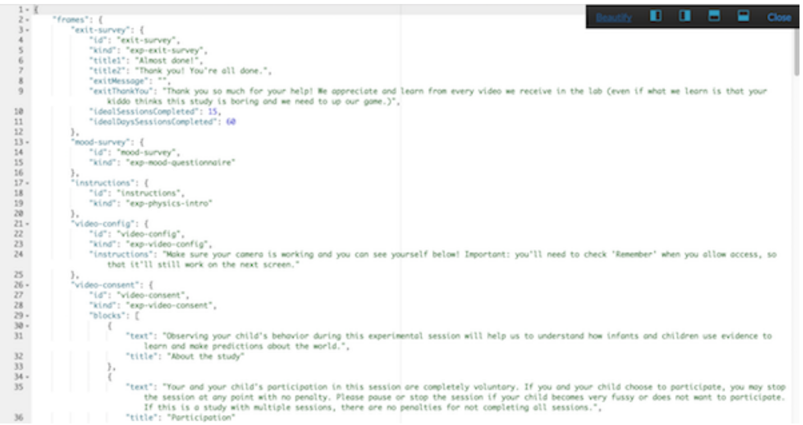
Name	Permissions
Kim Scott	Admin
Undergrad RA	Researcher

- *Unlisted (non-discoverable) studies* allow you to **invite only specific families** (e.g., in-lab participants or members of a registry) to participate based on a unique study link.
- Studies are deployed in their **own containers**, using a snapshot of the experiment runner codebase. Updates to the code never affect how your existing study works unless you choose to update!

1.2.4 An experiment runner custom-designed for developmental paradigms

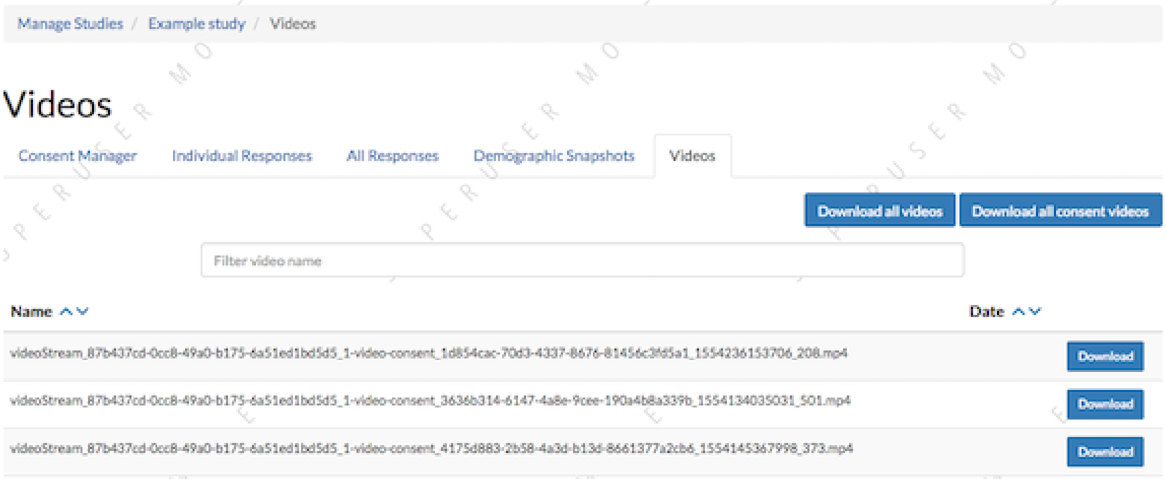
(although we're also interested in supporting other experiment building systems you may be familiar with!)

- Use a growing library of built-in components for common developmental measures and pages - e.g., looking time and preferential looking trials, change detection trials, parent-controlled observation videos, surveys, video consent and assent
- *Specify your protocol* using a readable JSON text format



```
{
    "frames": {
        "exit-survey": {
            "id": "exit-survey",
            "kind": "exp-exit-survey",
            "title1": "Almost done!",
            "title2": "Thank you! You're all done.",
            "exitMessage": "",
            "exitThankYou": "Thank you so much for your help! We appreciate and learn from every video we receive in the lab (even if what we learn is that your
                kiddo thinks this study is boring and we need to up our game.)",
            "idealSessionsCompleted": 15,
            "idealDaysSessionsCompleted": 60
        },
        "mood-survey": {
            "id": "mood-survey",
            "kind": "exp-mood-questionnaire"
        },
        "instructions": {
            "id": "instructions",
            "kind": "exp-physics-intro"
        },
        "video-config": {
            "id": "video-config",
            "kind": "exp-video-config",
            "instructions": "Make sure your camera is working and you can see yourself below! Important: you'll need to check 'Remember' when you allow access, so
                that it'll still work on the next screen."
        },
        "video-consent": {
            "id": "video-consent",
            "kind": "exp-video-consent",
            "blocks": [
                {
                    "text": "Observing your child's behavior during this experimental session will help us to understand how infants and children use evidence to
                        learn and make predictions about the world.",
                    "title": "About the study"
                },
                {
                    "text": "Your and your child's participation in this session are completely voluntary. If you and your child choose to participate, you may stop
                        the session at any point with no penalty. Please pause or stop the session if your child becomes very fussy or does not want to participate.
                        If this is a study with multiple sessions, there are no penalties for not completing all sessions.",
                    "title": "Participation"
```

- Flexibly **collect video** during studies, as well as event timing data relative to the video stream. Researchers have coded gaze measures (looking time, preferential looking) as well as verbal responses and parent interaction from webcam video.



Name	Date	
videoStream_87b437cd-0cc8-49a0-b175-6a51ed1bd5d5_1-video-consent_1d854cac-70d3-4337-8676-81456c3fd5a1_1554236153706_208.mp4		Download
videoStream_87b437cd-0cc8-49a0-b175-6a51ed1bd5d5_1-video-consent_3636b314-6147-4a8e-9cee-190a4b8a339b_1554134035031_501.mp4		Download
videoStream_87b437cd-0cc8-49a0-b175-6a51ed1bd5d5_1-video-consent_4175d883-2b58-4a3d-b13d-8661377a2cb6_1554145367998_373.mp4		Download

- Randomize condition assignment and counterbalancing, shuffle trial or task order, personalize text or stimuli based on child characteristics, add conditional logic, and more.
- Extendable for your custom games and measures; just fork the repo on GitHub, add your own frames, and tell Lookit to use your version of the code to run your study

1.2.5 Limitations

Lookit won't be the right approach for all online developmental research! Here are some cases where other tools will likely be a better fit:

- Interactive, synchronous studies where an experimenter talks with the family over audio or video chat, or where participants interact with each other. Lookit does not yet have these capabilities.
- Medical research conducted by HIPAA-covered entities. Lookit is not yet HIPAA compliant. (Note that HIPAA applies based on the status of the researcher - not just on the type of information collected. Academic researchers can generally collect health information without being covered by HIPAA.)
- Short one-off survey studies where you want to minimize time required to participate, and even asking families to create a login would be excessive
- Studies you have already implemented for adults and want to run with older children and teens as well. (You *could* run a study like this on Lookit, but if you already have a robust setup for collecting data from adults, it will probably be easier to stick with that!)
- Studies that pose appreciable risks or harm to participants or the world (e.g., you'd like to try teaching children about "the controversy" regarding climate change)

1.3 Project overview

Lookit is a platform for doing developmental research run by a small academic group at MIT. Our vision is of **a collaborative online lab**:

- Large collaborative "online lab" run by group at MIT
- Single participant interface; shared participant pool, servers; central & shared recruitment in addition to individual labs' efforts
- Lookit team provides training, IRB coordination, study implementation, design support, ongoing development, documentation

- Researchers independent, but with centralized approval of studies (covering technical problems, basic compliance with ethical guidelines, and clarity for parents)
- Support/incentives for best practices (e.g. preregistration, publishing materials and data, clearly demarcating pilot data)
- Can be funded (at steady state) primarily by participating labs' usage, although open to alternate models # Mission statement Lower barriers to conducting and participating in rigorous, reproducible developmental research that advances the understanding of development and its implications for education, parenting, policy, and medicine.

We are committed to

- Open source development
- Encouraging data and protocol sharing
- Encouraging best practices in experimental design
- Advancing our understanding of methods
- Recruiting a representative participant pool
- Respecting participants' time and parents as partners in discovery
- Enabling non-traditional developmental researchers and supporting work that benefits children or families

1.4 Progress updates

Progress updates are sent to the lookit-research list; please see the archives for a full list of past announcements. You can subscribe to this list here.

1.5 FAQ

1.5.1 What is the name of this platform? LoOkIt, Look!t, LooKit?

It is called `Lookit`. Like the exclamation. In particular it is not called `LookIt` or `Lookit!`, although the latter does sound extra exciting!

We'll respond to whatever you call us*, this is just for folks writing up results or designing recruitment materials, to avoid confusion.

*Maybe not LooKit, that sounds like a different thing.

1.5.2 Does my study need to be approved by MIT's IRB?

Nope! You need ethical approval only from your own institution. You're responsible for getting that, although there is example language here for common elements of Lookit studies.

Separately, your lab and institution will need to sign a one-time institutional agreement with MIT.

1.5.3 Who is the institutional agreement with?

MIT

1.5.4 Who should I list as a contact in case our contracts/sponsored programs/legal/etc. office has questions?

Kim Scott, lookit@mit.edu

1.5.5 Is there an IRB protocol at MIT that covers Lookit staff's use of the data on the platform?

No, because we only use the data for a limited set of purposes that aren't considered human subjects research by MIT. See the determination letter and corresponding protocol.

1.5.6 Most of my participants don't speak English. Can I test in another language?

Short answer: Yes, but.

Longer answer: Yes, and we want to support this much more fully in the future. For now, there will be some complications, but if you're up for dealing with them, you're welcome to design your study in any language you want:

- The sitewide registration and demographic forms are currently in English only, although setup for translation of these common elements is underway via ManyBabies-AtHome! If you're interested in testing in a particular language, please reach out to their translation group to see if you can contribute. This will help other people who want to test in your language, too!
- You'll likely need to write a translation file for your language for the components in the studies - e.g., consent, setup. Once that's available, you can specify a language for your study and any hard-coded text will be translated. (See elf-translations.)
- We haven't currently set up to flag which language a study is in and let participants filter (or get the right version of a study) by language. So non-English studies in general will usually to be set up as "non-discoverable" - not listed on lookit.mit.edu/studies, just accessible by direct link - to avoid confusion.
- You will need to handle any tech support for your participants directly, or translate for us.
- You will still need to get peer feedback from someone outside your research group on your study setup - this is especially important as we may not be able to tell much about the tone and clarity of your instructions.

We're thinking about this long-term but it hasn't been an immediate priority, especially as we're largely funded by the US government. Ideas/comments on how it'd ideally be set up are very welcome here.

1.5.7 Where are the data stored?

A1: In the USA.

A2: Videos are stored primarily on Amazon S3. Lookit (including databases with participant and session data) is hosted on Google Cloud Platform.

1.5.8 How is participant data secured?

Non-video data, including PI/SPI such as children's birthdates and nicknames, are stored using the Google Cloud SQL service of the Google Cloud Platform; the data security measures implemented by the Google Cloud Platform transitively apply to these data. All data are encrypted at rest using AES-256, and encrypted in transit when moving outside Google infrastructure. For an extensive treatment of the security-related provisions of this cloud infrastructure, please see Google's white paper. Service Accounts are used for all services provided by the Google Cloud Platform project instance. Permissions for access to data via the Lookit interface and API are handled using Django Guardian according to best practices.

User passwords on Lookit are required to be 16+ characters. Participants can access only their own video data; however, as researchers may be able to access video and other data from many participants in studies they have run, access to the Experimenter section of Lookit requires two-factor authentication. Researchers are responsible for the security of their Lookit credentials and for the security of data that they download using Lookit, although the platform is engineered with a focus on making it more difficult to accidentally disclose sensitive information. (For instance, consent must be confirmed before session data or video are accessible; child names and birthdates are by default omitted from session data downloads. If a participant withdraws video at the end of a study, that video is automatically made inaccessible to the researcher and deleted.)

Video data are routed securely from the web client to Amazon S3 through the Pipe media player (https://www.addpipe.com/), but not stored by Pipe. The requisite credentials for both accounts are encrypted and stored in the etcd database of a Google Kubernetes Engine instance, per the defaults provided by the platform. Video data are encrypted at rest on S3 using AES-256.

All dependencies for the Lookit-api and Ember-lookit-frameplayer repositories are continuously scanned by Github for security vulnerabilities, and the unit tests conducted as part of our CI/CD pipeline whenever code is updated cover many of the platform-specific security considerations (e.g. regarding appropriate access to specific data for specific roles).

Researchers can only access data from studies where they have permission to view data. Session, child, and family data are only possible to view after consent has been confirmed. The permissions granted to researchers are quite granular, so that researchers can be granted access to only the minimum functions needed for their role, without any unnecessary risk to participant personal data. Researchers can contact participants through the platform, but do not receive access to participant email addresses (unless participants email them).

We hired an external security consulting firm to conduct detailed manual penetration and a security risk assessment prior to launch, in spring 2020. Results of this assessment are available upon request (email lookit@mit.edu).

1.5.9 Does Lookit collect IP addresses?

No.

1.5.10 How is re-identification prevented?

There are several measures in place to *discourage* re-identification, including:

- Researchers using the Lookit platform do not receive direct access to participant email addresses. They can contact participants using the Lookit interface based on the participant's random ID, but see an email address only if a participant contacts them.
- Although each child and each family registered on Lookit is associated with a global unique random identifier, they are also associated with a different random identifier specific to each study, and the latter is the primary ID used by researchers. Per the Terms of Use, researchers may not publish the global identifiers, as these could link data across studies in ways that could lead to unanticipated re-identification potential.
- Default data downloads minimize the amount of personally identifiable information included: e.g., researchers have to specifically request columns for the child name, birthdate, parent name, etc. By default, neither the child's birthdate nor exact age at time of participation (which could be combined with timestamps to produce a birthdate) are included, and a rounded age is provided for ease of responsibly publishing raw data.
- Per the Terms of Use, no participant demographic information may be published in such a way that individual responses can be linked to participant video.

However, we cannot fully *prevent* re-identification using solely technical means, as some of the data collected on Lookit is by nature potentially identifying (e.g., video of faces). All researchers using the platform must have IRB approval for data collection, which includes assurances that they will not attempt to re-identify participants. The Terms of Use also require approval for any integration of outside information about participants (e.g., if participants are recruited from a specific registry that already has data about the families).

1.6 Resources

1.6.1 Learning materials

Here are some slides and videos you may want to check out while getting started or when training other lab members!

[PDF] Introduction to Lookit: a 20-minute overview (Jenna Croteau)

[Video] CBMM Tutorial: Using Lookit to run developmental studies online (Maddie Pelz)

[Slides], [Video] Using the MIT Lookit webcam platform for home-based studies (Caspar Addyman) - presentation at ICIS 2020

[Slides] Lookit parent perspective (Nicole Cuneo)

[Slides] Lookit video data (what it looks like) (Nicole Cuneo)

[Slides] Stimuli preparation and hosting for Lookit (Nicole Cuneo)

[Slides] FFMPEG starter powerpoint (Nicole Cuneo)

[Code] Some example FFMPEG commands (Kim Scott)

If you give a presentation about using Lookit or prepare training materials for your lab, please share them here! You can *propose that change directly* or email lookit@mit.edu.

1.6.2 Other helpful resources

- Most communication among Lookit researchers happens via a Slack workspace. Fill out this form to receive an invite. This is the best place to ask general questions or get tech support!
- The documentation for individual experiment 'frames' lives here in the "Experiment runner" tab.
- Running into a problem and want to check if it's a known issue, or have an idea for a handy new feature? Check out and/or add to the issues listed for the Lookit platform and for the experiment components/player. Or check out projects to take a look at what's coming up in terms of development!
- Join the Lookit-research email list for occasional (2-4x/year) progress updates.
- Feel free to fill out the survey collecting info about what people would do with Lookit
- Our March 2020 APS Observer article which describes the current status of Lookit and how you can get your study ready!
- 3-year plan
- Lookit papers (1, 2)
- Lookit overview video
- Video data from test studies (email Kim for access)
- Photos and video that can be used for publicity
- Recruitment materials (flyers, brochures, etc.)

1.6.3 Codebase

All Lookit code is open-source (MIT License - this is a liberal open-source license, not related to us being at MIT) and publicly available. It will stay that way.

- https://github.com/lookit/ember-lookit-frameplayer
- https://github.com/lookit/lookit-api
- https://github.com/lookit/lookit-docs
- https://github.com/orgs/lookit/projects - software development planning: known issues, planned features, scheduling.

1.7 Publications

1.7.1 Publications using Lookit data

Here's some of the research that's come out of Lookit! There's a lot more in prep, so if you're interested in questions like whether someone's tried X task on Lookit and how it went, we recommend asking in the #researchers channel on Slack for more up-to-date info.

- Beckner, A.G., Voss, A.T., Oakes, L.M., Casasola, M. (2021, April). Assessing the robustness of the mental rotation change detection procedure: the importance of task and context. In symposium: Infants' learning about object properties and categories in diverse environments. Symposium conducted at the biennial Society for Research in Child Development (Virtual).
- Bochynska, A., Scott, K., and Dillon, M. (2021, April). Bringing home Baby Euclid: Evaluating infants' basic shape discrimination using the online platform Lookit. In symposium: Infants' learning about object properties and categories in diverse environments. Symposium conducted at the biennial meeting of the Society for Research in Child Development (Virtual).
- Casey, K., Scott, K., Ashton, K., Gill, J., Simpson, E., and Bayet, L. (2021, April). Neonatal imitation of caregivers at home: Pre-registered analyses. Poster at the biennial meeting of the Society for Research in Child Development (Virtual).
- Cassamajor, K., Chu, J., Scott, K., and Schulz, L. (2021, April). A large-scale study of infant intuitive physics. Poster at the biennial meeting of the Society for Research in Child Development (Virtual).
- Chalik (2021, April). Moral behavior within and across social groups. In J. Dunlea (Chair), The role of social relationships in shaping children's socio-moral reasoning. Symposium conducted at the biennial meeting of the Society for Research in Child Development (Virtual).
- Casey K., Scott K., Ashton K., Gill J., Simpson E., & Bayet L. (2020) Neonatal imitation of caregivers: A feasibility pilot. The CogSci 2020 Virtual Conference.
- Yoon, E. J., Frank, M. C. (2019). Preschool children's understanding of polite requests. Proceedings of the 41st Annual Conference of the Cognitive Science Society. [pdf], [repository].

If you give a presentation or publish a paper using Lookit data, please share it here! You can *propose that change directly* or email lookit@mit.edu with the citation (and a PDF or video link if you want to share it here).

1.7.2 Original Lookit papers

These papers introduced a prototype of Lookit and reported on the initial test studies we ran. They may have helpful information about general considerations for unmoderated online research - e.g., do we basically see similar looking times even though the environment is noisier (spoilers: yes) and can you see where kids are looking from webcam (also yes)?

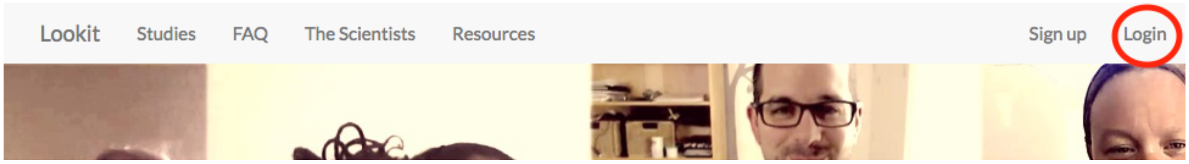
However, the platform itself has been completely re-engineered since these studies were conducted starting in 2014, so some of the details are no longer relevant - in particular, video quality is much better and we're no longer collecting variable-framerate .flv files.

- Scott, K. M. and Schulz, L. E. (2017). Lookit (part 1): a new online platform for developmental research. Open Mind 1(1):4-14. doi:10.1162/opmi_a_00002
- Scott, K. M., Chu, J., and Schulz, L. E. (2017). Lookit (part 2): Assessing the viability of online developmental research, results from three case studies. Open Mind 1(1):15-29. doi:10.1162/opmi_a_00001

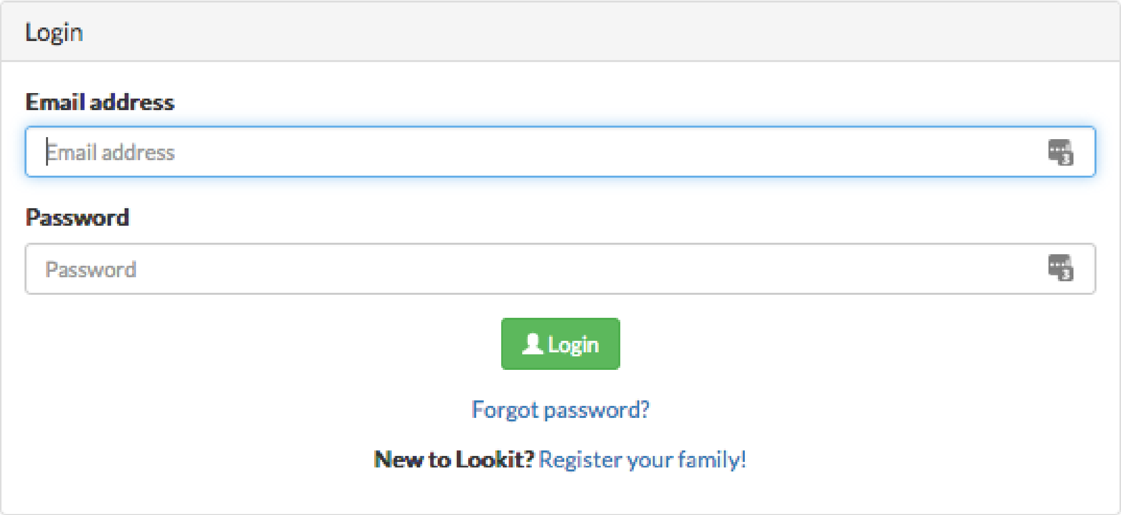
1.8 Registration and login

1.8.1 Logging in

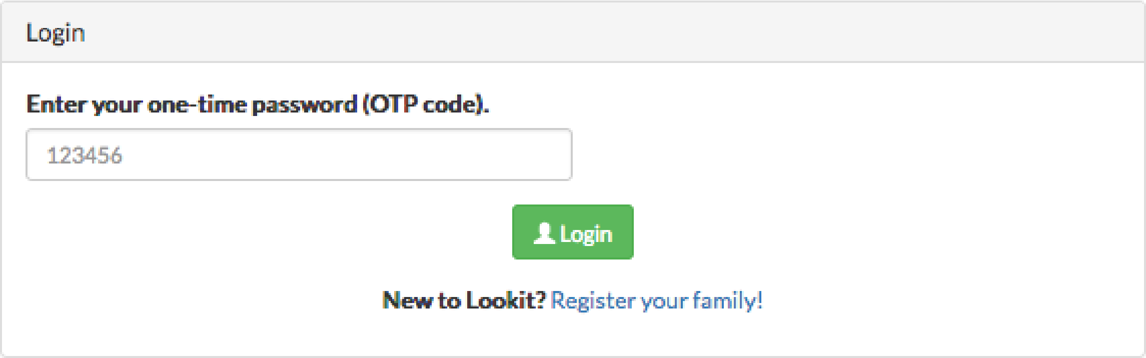
1. Click "Login" from the Lookit home page. Both researchers and participants log in from this form.



2. Enter your email address and password. After logging in, you can participate in studies, view past sessions, and manage your account or registered children.



3. If you are logging in to a researcher account, you will additionally be prompted for a one-time password from Google Authenticator so that you can access the researcher section.



Login

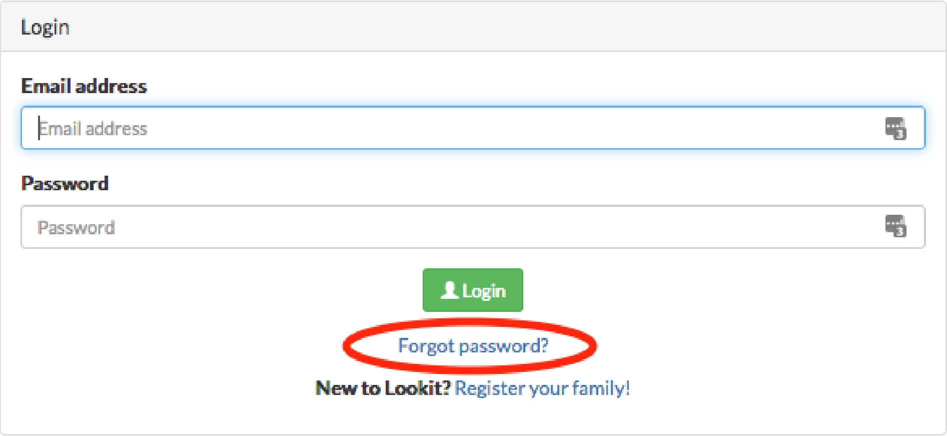
Enter your one-time password (OTP code).

123456

New to Lookit? Register your family!

If you created your account via OSF (i.e. before August 2020)

The first time you log in to Lookit after OSF login is unavailable, you will need to click "Forgot password?" to request a password reset link.



Login

Email address

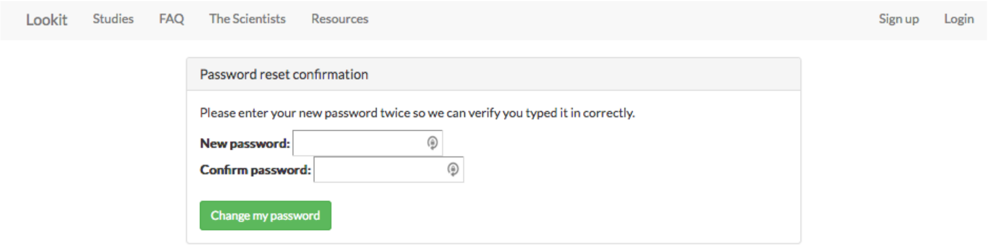
Email address

Password

Password

New to Lookit? Register your family!

Follow the link you receive by email to set a new Lookit password for your account:



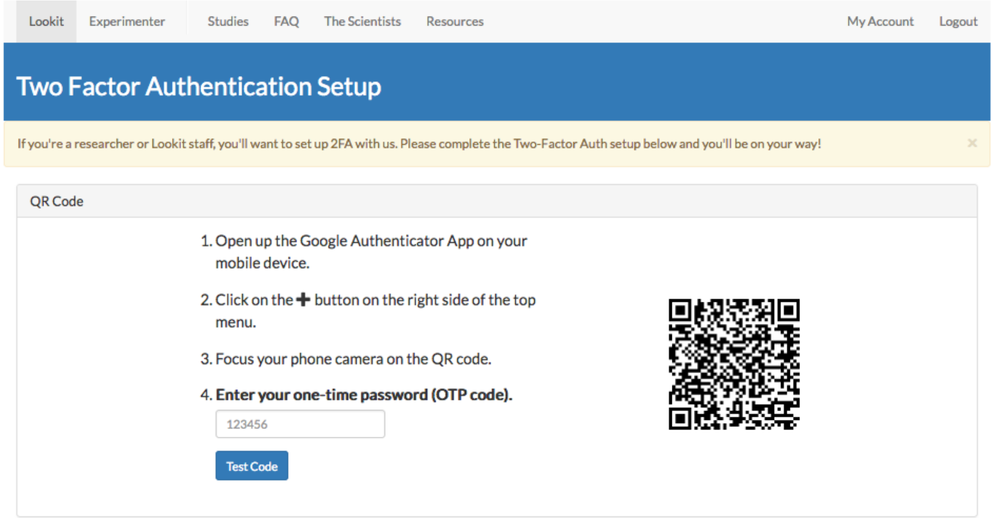
Password reset confirmation

Please enter your new password twice so we can verify you typed it in correctly.

New password:

Confirm password:

When you log in again, you will be redirected to two-factor authentication setup so you can use the researcher section of Lookit!



Two Factor Authentication Setup

If you're a researcher or Lookit staff, you'll want to set up 2FA with us. Please complete the Two-Factor Auth setup below and you'll be on your way!

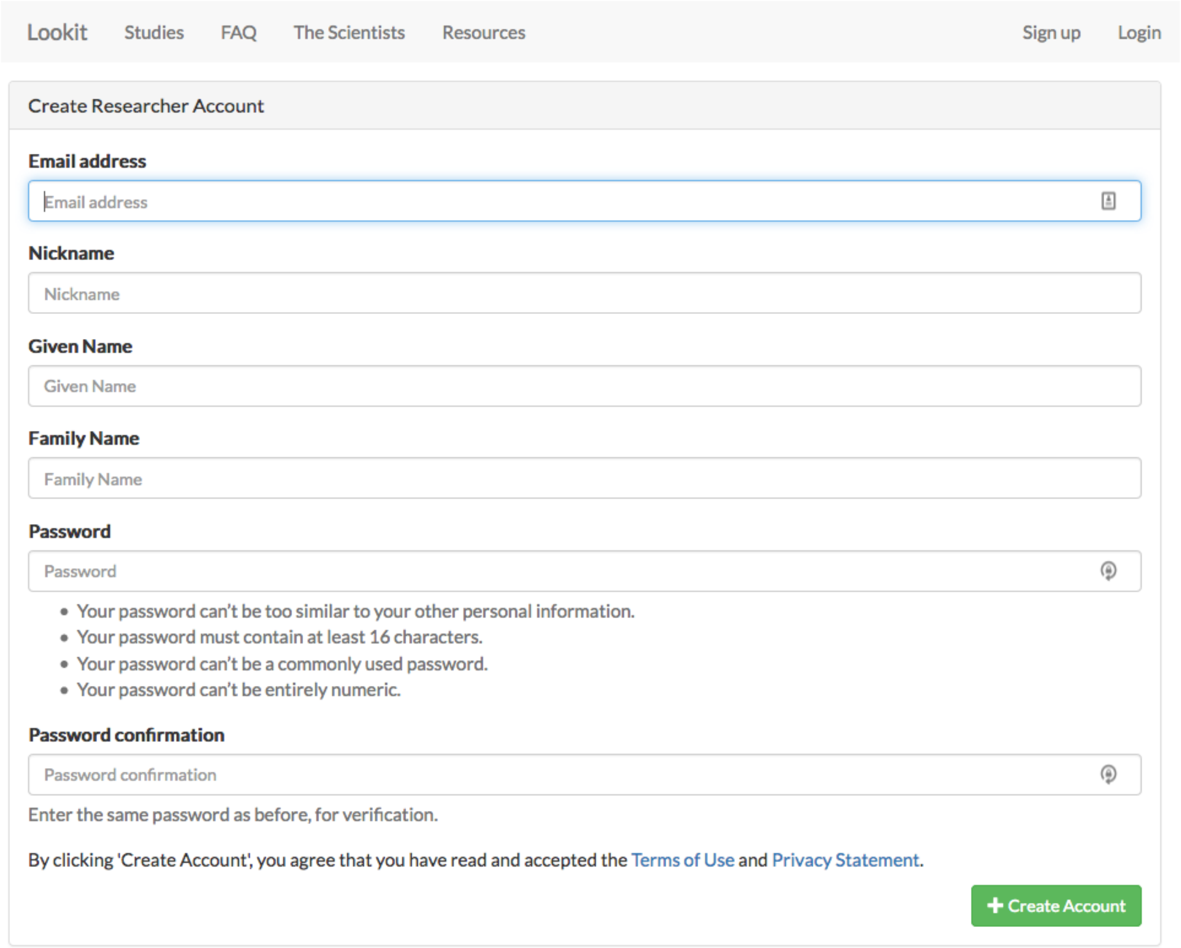
QR Code

1. Open up the Google Authenticator App on your mobile device.
2. Click on the ✚ button on the right side of the top menu.
3. Focus your phone camera on the QR code.
4. **Enter your one-time password (OTP code).**

123456

1.8.2 Creating a researcher account

1. Go to the researcher registration form. (You can also get here from the Lookit home page by clicking "Sign up" and then following the link to register as a researcher instead of a participant.)



Create Researcher Account

Email address

Email address

Nickname

Nickname

Given Name

Given Name

Family Name

Family Name

Password

Password

- Your password can't be too similar to your other personal information.
- Your password must contain at least 16 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

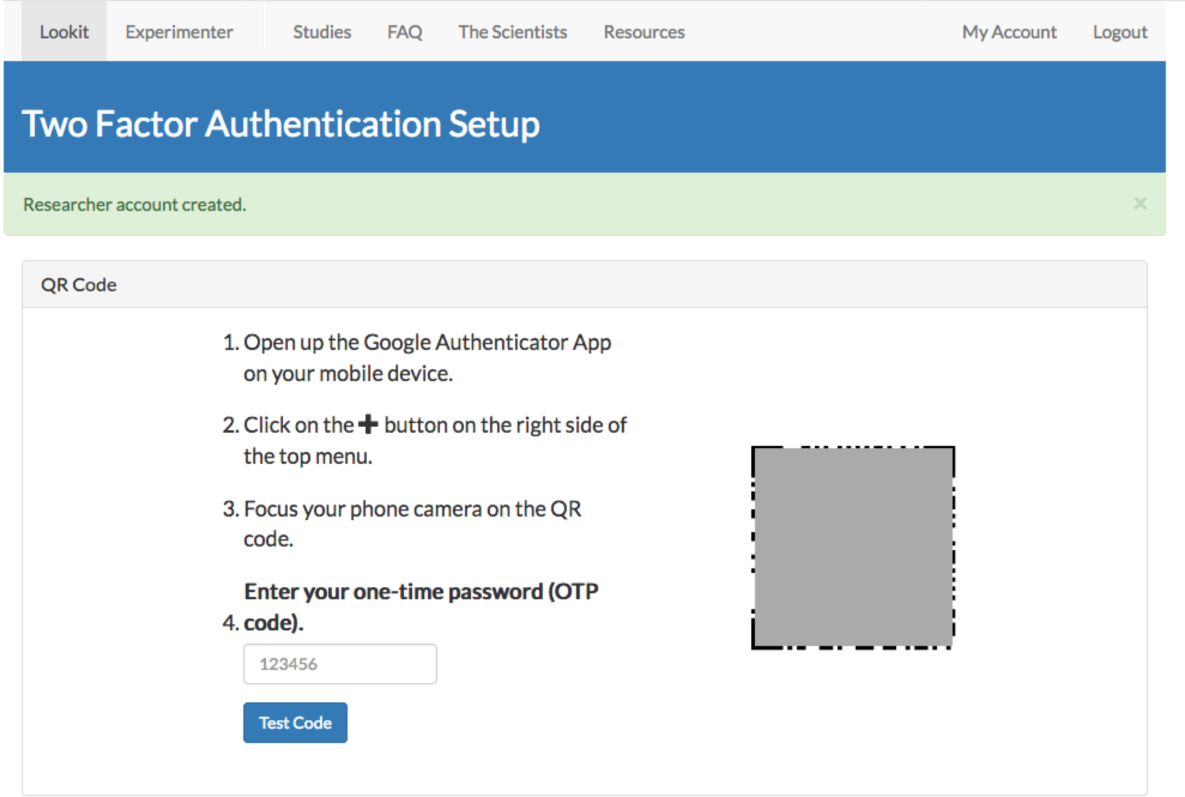
Password confirmation

Password confirmation

Enter the same password as before, for verification.

By clicking 'Create Account', you agree that you have read and accepted the Terms of Use and Privacy Statement.

2. Fill out the form and click “create account.”
3. You’ll be taken to a page like this to set up two-factor authentication (2FA), which you’ll need in order to access the researcher section of Lookit. If you haven’t already, download the Google Authenticator app on your phone. Then follow the directions on this page to activate 2FA for your account.



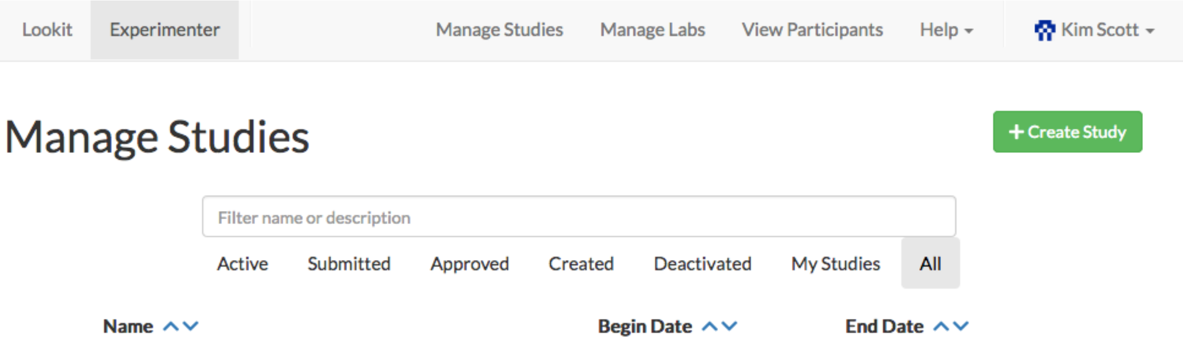
Two Factor Authentication Setup

Researcher account created.

QR Code

1. Open up the Google Authenticator App on your mobile device.
2. Click on the ✚ button on the right side of the top menu.
3. Focus your phone camera on the QR code.
4. **Enter your one-time password (OTP code).**

4. You’re logged in! You should be redirected to a page of studies like below. You will likely want to create a new lab or request to join a lab (see *Labs on Lookit*) so you can share your studies within a lab, but you can go ahead and get started by creating studies in the “Sandbox lab.”

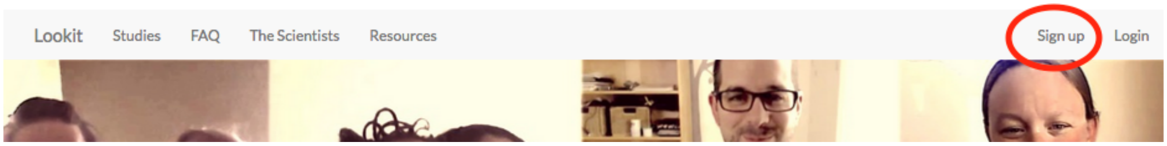


Manage Studies

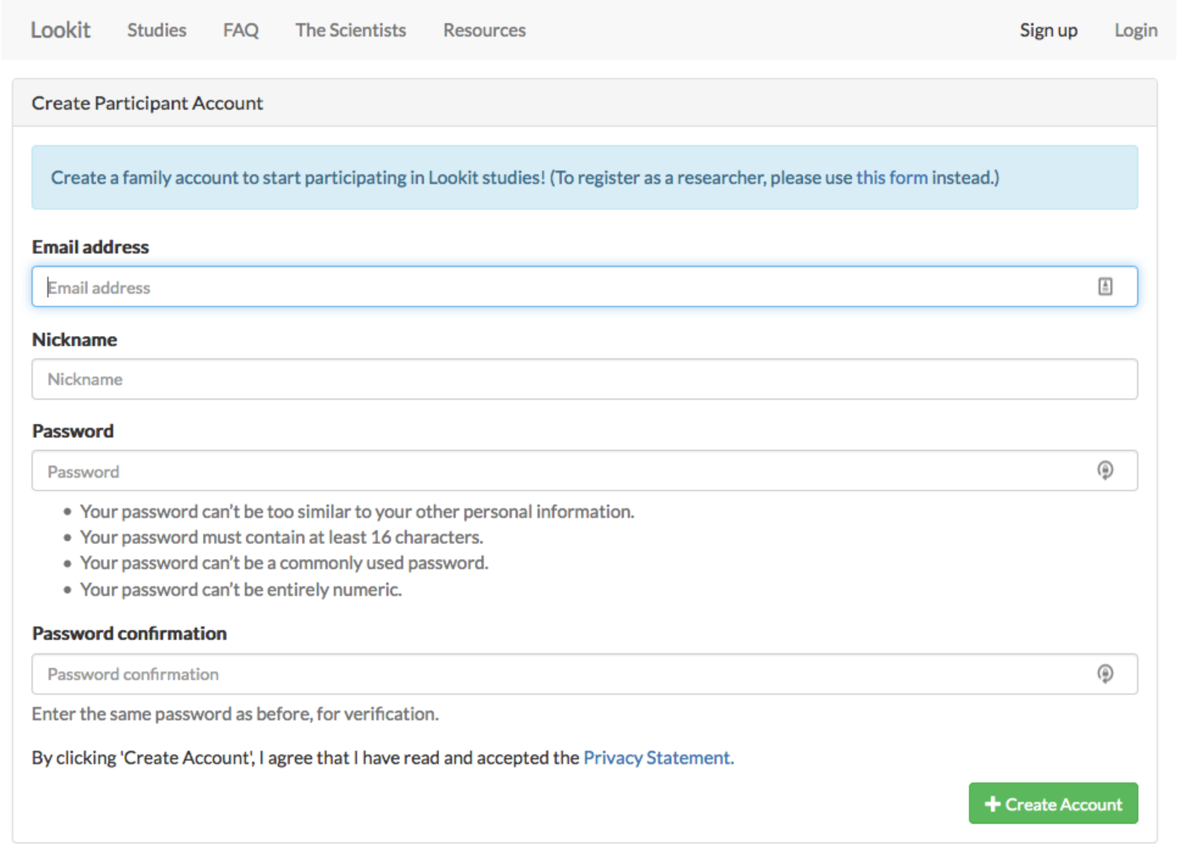
1.8.3 Creating a participant account

You do not need to make a separate participant account in order to participate with your own kids: you can log in as a researcher, then go back to Lookit and you will still be logged in. However, if you prefer to have a separate participant account, you can do so by logging out and clicking the “Sign up” button. You’ll need to use a different email address from the one your researcher account uses.

1. Click "Sign up" from the Lookit home page.



2. Fill out the form and click "create account."



Create Participant Account

Create a family account to start participating in Lookit studies! (To register as a researcher, please use this form instead.)

Email address

Email address

Nickname

Nickname

Password

Password

- Your password can't be too similar to your other personal information.
- Your password must contain at least 16 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation

Password confirmation

Enter the same password as before, for verification.

By clicking 'Create Account', I agree that I have read and accepted the Privacy Statement.

1.8.4 The Lookit staging server

The Lookit "staging server" is a separate instance of the platform from the "production server" where actual studies take place. This is a sandbox environment where we try out new features, without having to worry about messing with actual data or inconveniencing users if we break something.

The staging server is located at https://lookit-staging.mit.edu/.

You can register as a researcher at https://lookit-staging.mit.edu/registration/ or as a participant by clicking "Sign up". Logging in works just the same way as at lookit.mit.edu, but accounts are separate. (Your staging login won't work on regular or "production" Lookit, and vice versa.)

There is **absolutely no actual data collection allowed** on the staging server for security reasons.

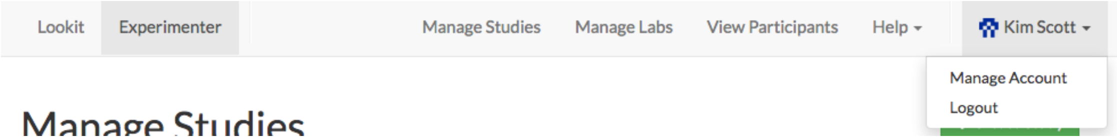
You might use the staging server if:

- You want to have a totally separate "sandbox" version of your lab on Lookit to use for training new lab members, study development, demoing something, etc. without any concern that you could possibly disclose participant information

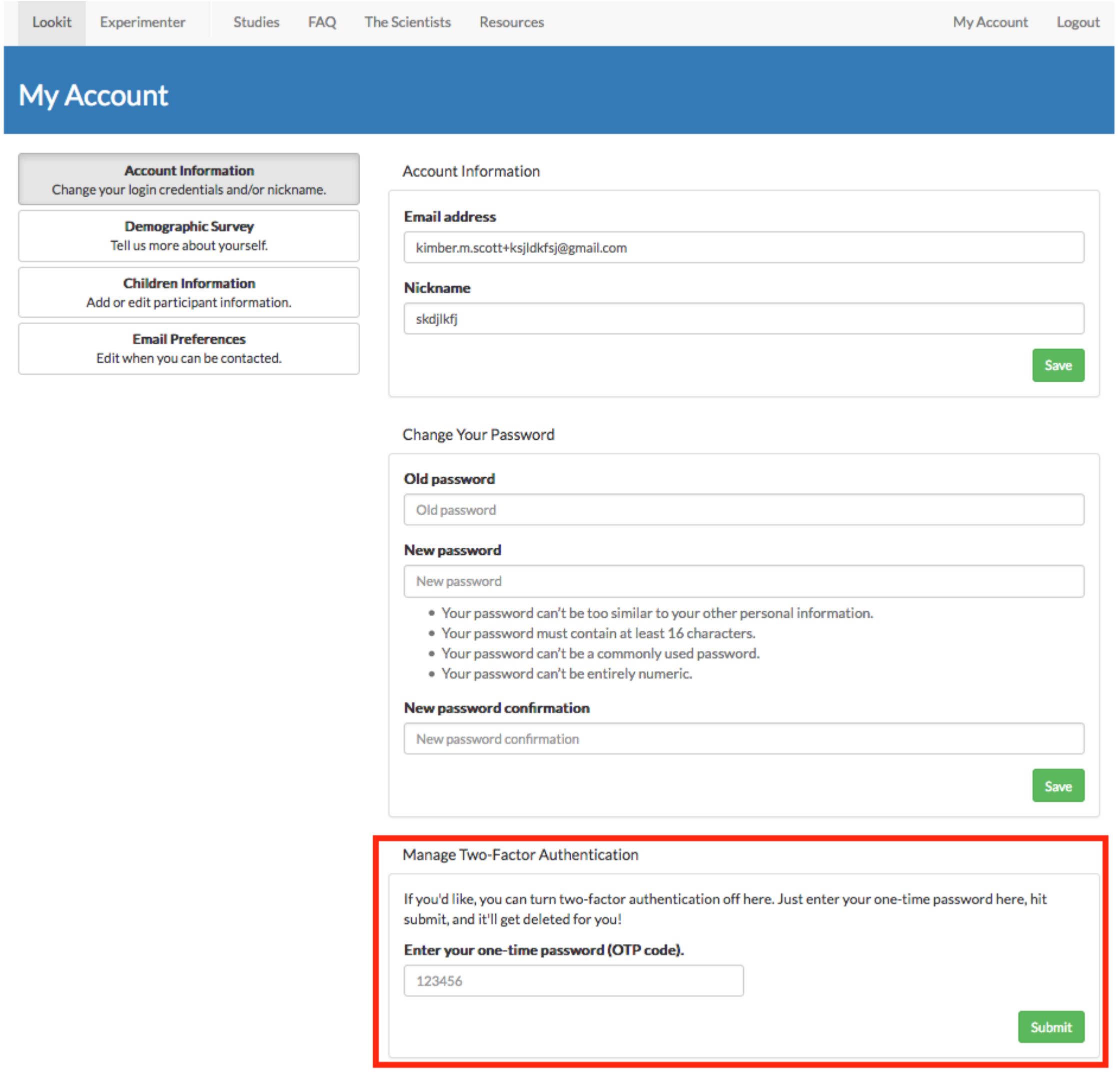
- You're helping to try out a new feature before it's added to production
- You're making especially heavy use of Lookit while developing your own studies, working on an adapter so you can hook up another type of experiment runner, etc. - to avoid unusual load on the production server we might ask you to do your development using staging.

1.8.5 Managing 2FA credentials

If needed, you can disable two-factor authentication on your account (for example, to switch to using a new phone). Click "My account" from the experimenter interface:



From the account page, you can change your email address or password and enable or disable 2FA. Note that you cannot use the Experimenter section of Lookit without 2FA enabled.



My Account

Account Information
Change your login credentials and/or nickname.

Demographic Survey
Tell us more about yourself.

Children Information
Add or edit participant information.

Email Preferences
Edit when you can be contacted.

Account Information

Email address
kimber.m.scott+ksjldkfsj@gmail.com

Nickname
skdjlkfj

Save

Change Your Password

Old password
Old password

New password
New password

- Your password can't be too similar to your other personal information.
- Your password must contain at least 16 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

New password confirmation
New password confirmation

Save

Manage Two-Factor Authentication

If you'd like, you can turn two-factor authentication off here. Just enter your one-time password here, hit submit, and it'll get deleted for you!

Enter your one-time password (OTP code).
123456

Submit

1.8.6 Logging in when running Lookit locally for development

If running Lookit locally, you can log in or create accounts as described above, substituting `https://localhost:8000` for `https://lookit.mit.edu`. (You no longer need to authenticate through the admin interface with any experimenter accounts.)

1.8.7 Troubleshooting

I'm trying to log in but it says my account is "inactive"

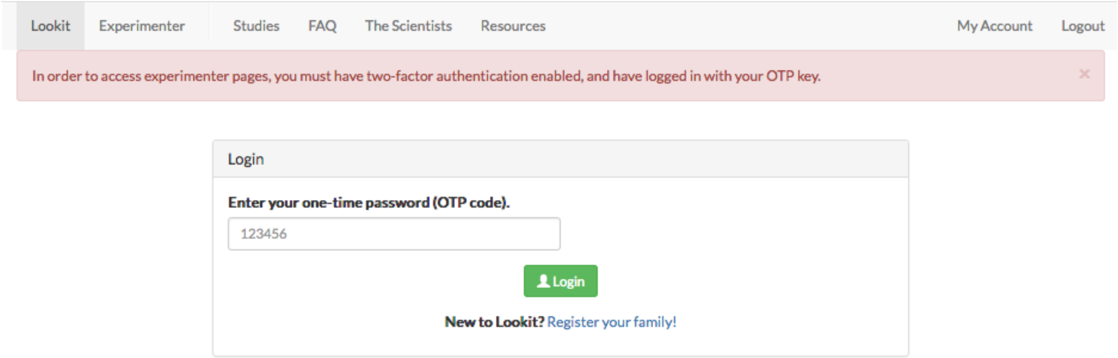
Most likely you accidentally tried to create a researcher account on Lookit before we launched, and it was inactivated. Please reach out in the #tech-support channel on Slack and we can fix it.

I can't register as a researcher because I already have a participant account

You have two options:

- make a researcher account with a different email address
- log in to your participant account and change the email address associated with it

I'm being prompted for a one-time password (OTP) but I don't have that set up



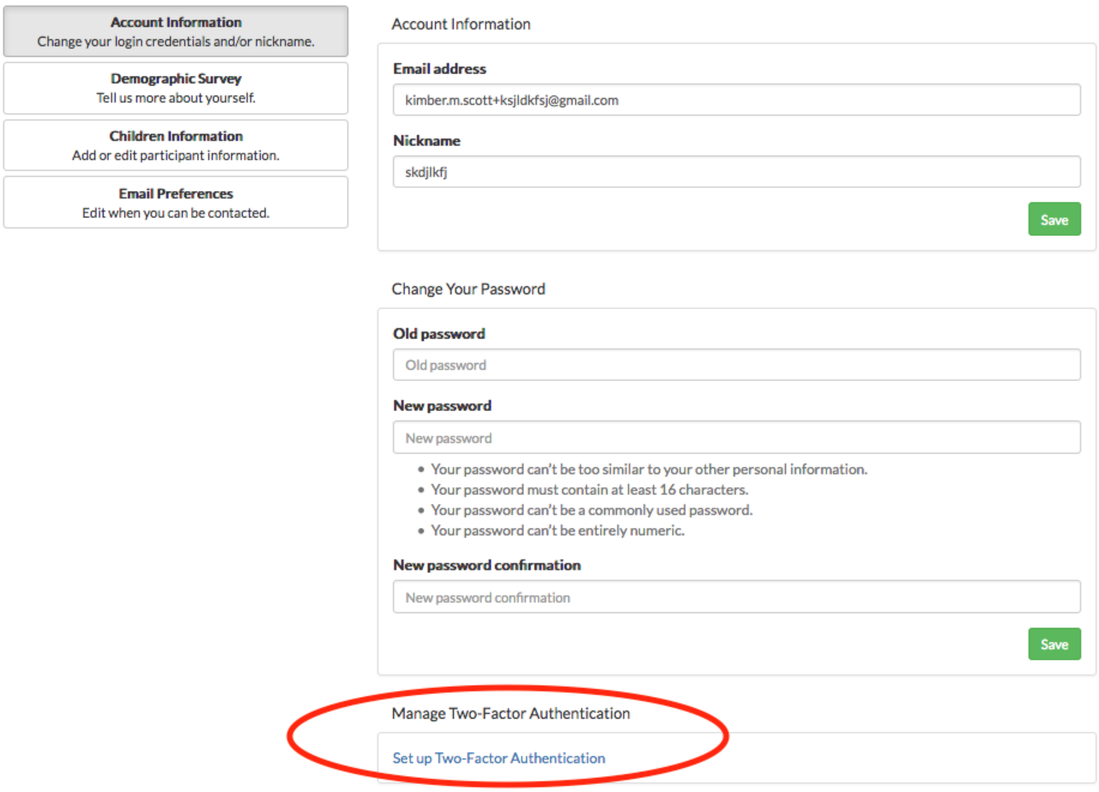
In order to access experimenter pages, you must have two-factor authentication enabled, and have logged in with your OTP key.

Enter your one-time password (OTP code).

123456

New to Lookit? Register your family!

If you are seeing a message like this when trying to access the researcher side of Lookit, click "My Account" at the top right. From there you should be able to enable 2FA or complete setup.



Account Information
Change your login credentials and/or nickname.

Demographic Survey
Tell us more about yourself.

Children Information
Add or edit participant information.

Email Preferences
Edit when you can be contacted.

Account Information

Email address
kimber.m.scott+ksjldkfsj@gmail.com

Nickname
skdjlkfj

Save

Change Your Password

Old password
Old password

New password
New password

- Your password can't be too similar to your other personal information.
- Your password must contain at least 16 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

New password confirmation
New password confirmation

Save

Manage Two-Factor Authentication

Set up Two-Factor Authentication

I'm not receiving any OTP codes on my phone when I try to log in

The OTP codes are not sent to you via push notification or text message - they are just available in your Google Authenticator app. Please see if you have Google Authenticator installed on your phone and if you see a "Lookit-production" entry there (or "Lookit-staging" for the staging server). The OTP code shown there changes every 30 seconds.

We recommend using Google Authenticator to get your OTP codes, but you may have set up via Duo - check your phone for a Lookit entry under either app.

My OTP codes don't work

First check that:

- The email address shown in your authenticator app matches the email address you're trying to log in as. If you created multiple accounts on Lookit, you may have replaced the OTP entry for one with the other. If that's the case, contact Lookit staff (#tech-support channel on Slack) for assistance.
- You are entering the code within the 30-second window. If using Duo, some users have found the timing is more finicky and they need to enter the code in the first 10 seconds or so.
- The OTP entry says "Lookit-production" if you're using lookit.mit.edu, and "Lookit-staging" if you're using lookit-staging.mit.edu.
- The timing of your Authenticator app is accurate and your phone's time is accurate. If you're using Google Authenticator, you can go to the three dots in the top right corner -> Settings -> Time correction for codes -> Sync now to ensure the timing is correct.

I switched to a new phone and can't get my OTP code

If you still have access to your old phone:

1. Log in to your account and enter the OTP code using your old phone.
2. Click "My account" or go to **'https://lookit.mit.edu/account/manage/'__** and scroll down to "Manage Two-Factor Authentication."
3. Enter your OTP from the old phone to disable 2FA temporarily.
4. From "Manage Two-Factor Authentication," turn 2FA back on using your new phone.

If you do not have access to your old phone (e.g., it was destroyed and that's why you're switching): Please contact Lookit staff (#tech-support channel on Slack) for assistance. For security reasons, there isn't a way to disable or reset your two-factor authentication unless you are already logged in using a one-time password. Depending on whether you already have access to participant data, we will reset it for you or ask for verification of your identity first.

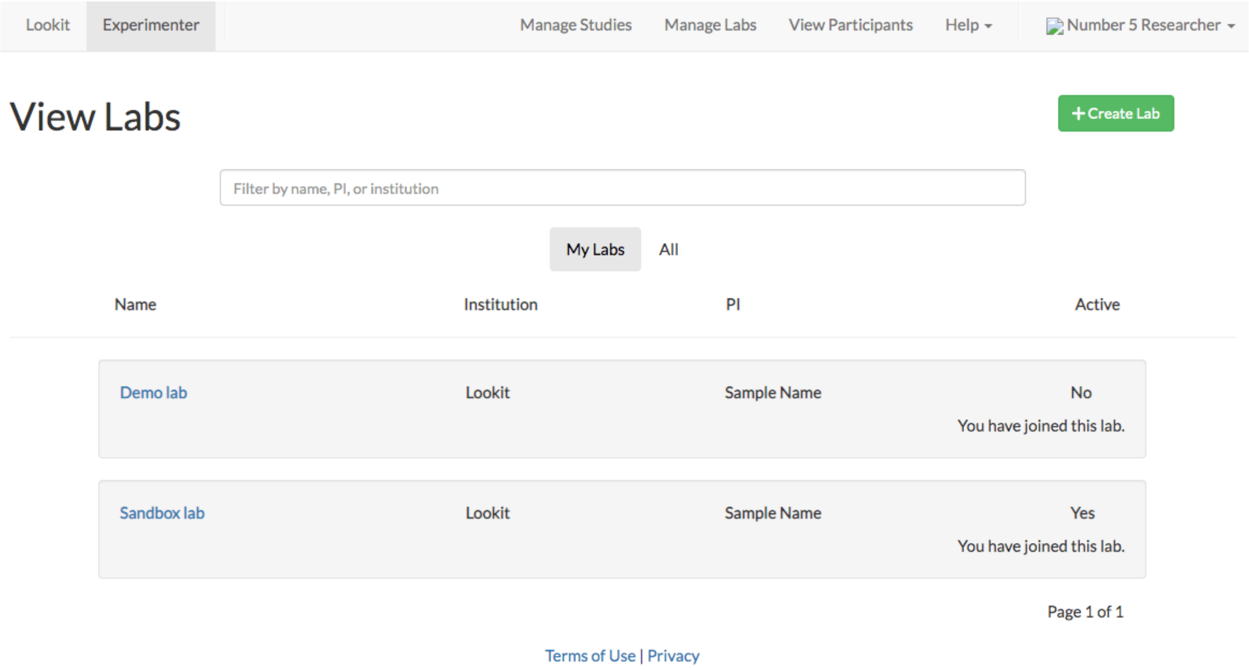
1.9 Labs on Lookit

Each study on Lookit is associated with a lab. Some basic information about each lab is stored on Lookit, like the lab's name, a contact email, and whether the lab has been approved to use Lookit to collect data. In the future, this information will be used to create a display page where participants can learn more about your lab, but for now, it is only visible to researchers.

Researcher accounts on Lookit can be affiliated with any number of labs. Only a researcher who is part of a lab can be granted permissions to studies that lab is running. Researchers can also be granted some permissions for the lab as a whole: for instance, to edit the lab description, manage membership, or see all studies the lab is running.

1.9.1 The Sandbox and Demo labs

There are two "special" labs that everyone on Lookit is added to automatically. When you log into Lookit for the first time, if you click on "Manage Labs," you should see something like this:



View Labs

Name	Institution	PI	Active
Demo lab	Lookit	Sample Name	No You have joined this lab.
Sandbox lab	Lookit	Sample Name	Yes You have joined this lab.

Page 1 of 1

You are added as a “Guest” in the **Sandbox lab.** This means you can create new studies associated with the Sandbox lab, and you can be given permissions to other studies associated with the Sandbox lab. However, you can’t change the lab or see all the studies it is running. If you’re working through the tutorial or trying out Lookit to see how it works, you can work on studies in the Sandbox lab and not have to create your own lab.

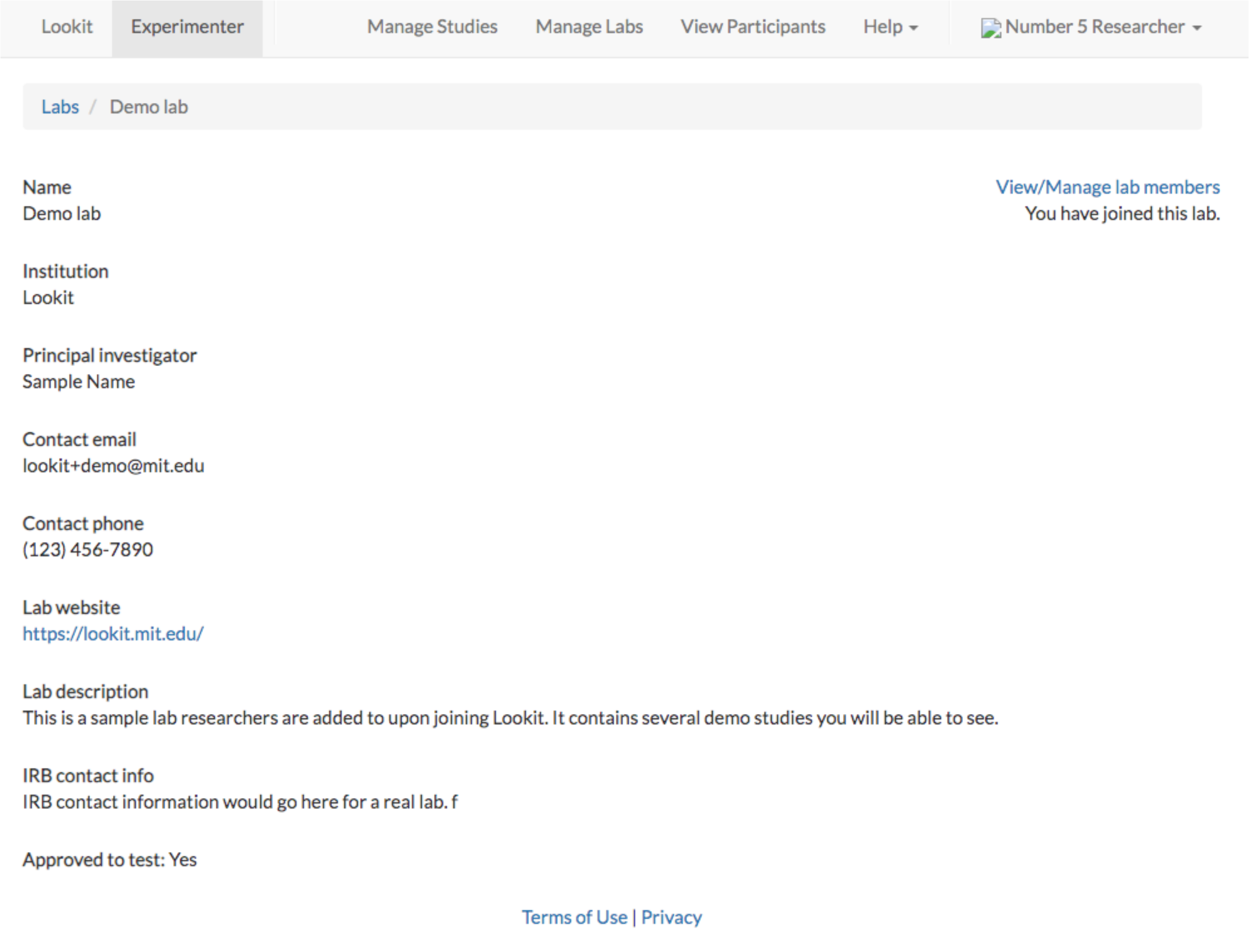
You are added with “View” permissions in the **Demo lab.** This means you can see and preview all of the studies in the Demo lab, but not create any new studies in this lab. We keep some example studies in the Demo lab so that everyone can see them. You can clone any of the Demo lab studies into the Sandbox lab or your own lab so you can edit a copy.

1.9.2 Creating a new lab

From the “Manage Labs” page, click the green “Create Lab” button. This will take you to a form where you can enter the information about your lab. When you click “Submit,” the lab will be created, with you as an admin, and Lookit staff will be notified of the new lab. You will be able to see your new lab in your list of labs.

1.9.3 Approval to test

Once Lookit staff receive a signed institutional agreement for your lab to use Lookit, and at least one of your lab members has completed the Terms of Use quiz, your lab will be changed to “approved to test.” Lab admins will be notified by email when the lab is approved to test. You can also see whether a lab is currently approved to test by clicking on it in the list of labs to view more details:



You have joined this lab.

Name
Demo lab

Institution
Lookit

Principal investigator
Sample Name

Contact email
lookit+demo@mit.edu

Contact phone
(123) 456-7890

Lab website
https://lookit.mit.edu/

Lab description
This is a sample lab researchers are added to upon joining Lookit. It contains several demo studies you will be able to see.

IRB contact info
IRB contact information would go here for a real lab. f

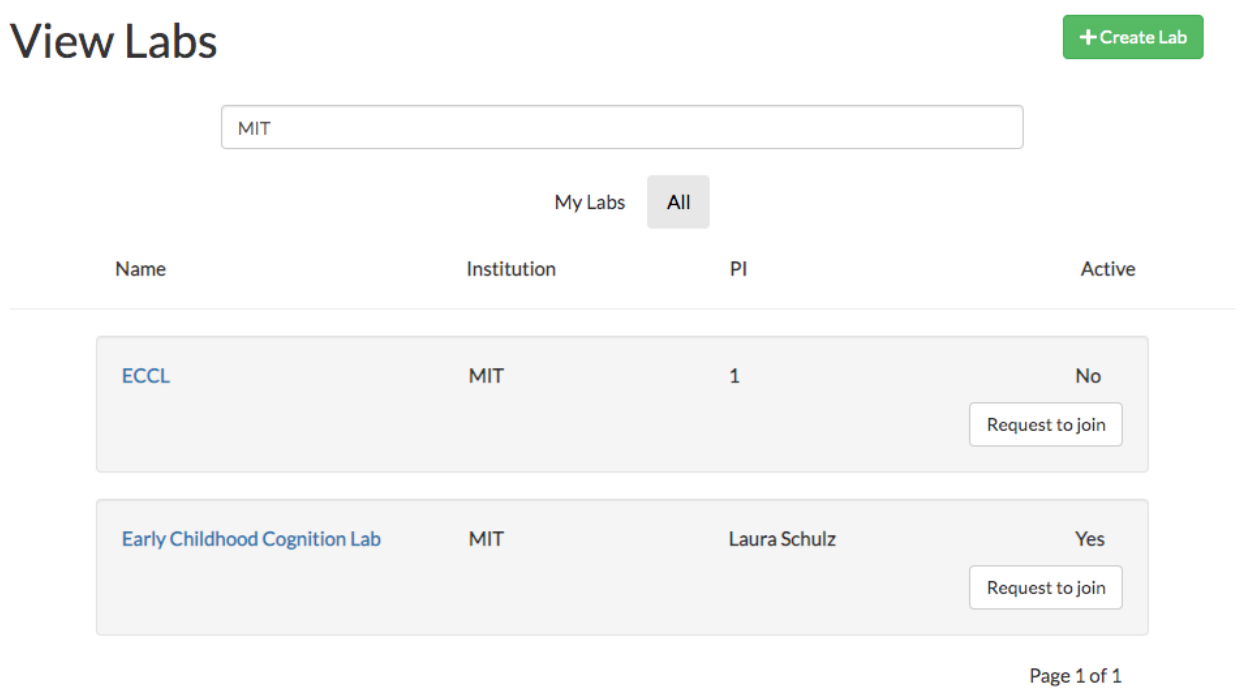
Approved to test: Yes

Before a lab is approved to test, you can add and manage lab members and create studies associated with the lab; you just won't be able to submit or start those studies yet.

Individual studies will still require review - the lab being approved to test is not blanket approval for all studies you might run.

1.9.4 Joining an existing lab

If your lab is already on Lookit, you can find it and join it. Go to "Manage Labs" and click the "All" tab. Then you can search by name, PI name, or institution to find the lab. Click "request to join" next to the lab you would like to join. (You can also click on the lab to get more details first!)



View Labs

MIT

Name	Institution	PI	Active
ECCL	MIT	1	No
Early Childhood Cognition Lab	MIT	Laura Schulz	Yes

The lab admins will receive an email notifying them of your request, and you will be emailed when they add you.

1.9.5 Adding lab members and managing permissions

Researchers have to request to join your lab; you don't add them directly. As a lab admin, you will receive emails when new researchers request to join your lab. You can also set researchers' permission levels. There are three roles you can assign researchers to:

Guest Can create new studies associated with this lab, and can be given permissions to specific studies associated with this lab

Lab member Guest-level permissions, plus: can *READ_STUDY_DETAILS*, *READ_STUDY_PREVIEW_DATA*, and *CODE_STUDY_PREVIEW_CONSENT* for every study associated with this lab. For details of these permissions, see *the section on study permissions*; essentially this lets a lab member view all the studies associated with a lab, but not edit those studies, change their status, or view any human subjects data unless they have study-specific permissions.

Admin Lab member permissions, plus: can edit lab metadata (name, contact email, etc.), can manage lab researchers (add/remove people, change perms among these three groups); and *WRITE_STUDY_DETAILS*, *CHANGE_STUDY_STATUS*, and *MANAGE_STUDY_RESEARCHERS* for every study associated with this lab. Again, this does not grant access to any human subjects data without study-specific permissions, but it does allow a lab admin to manage who has what access to which studies, edit study protocols and descriptions, and start and stop data collection.

Whether to make most lab members "guests" vs. "full members" is just a matter of whether you want a more communal workspace where everyone can see all the studies, or a more private workspace where people have to be invited to collaborate on particular studies. You might choose to make everyone in your lab a "lab member," but occasionally add an outside collaborator who's just helping with a single study as a guest.

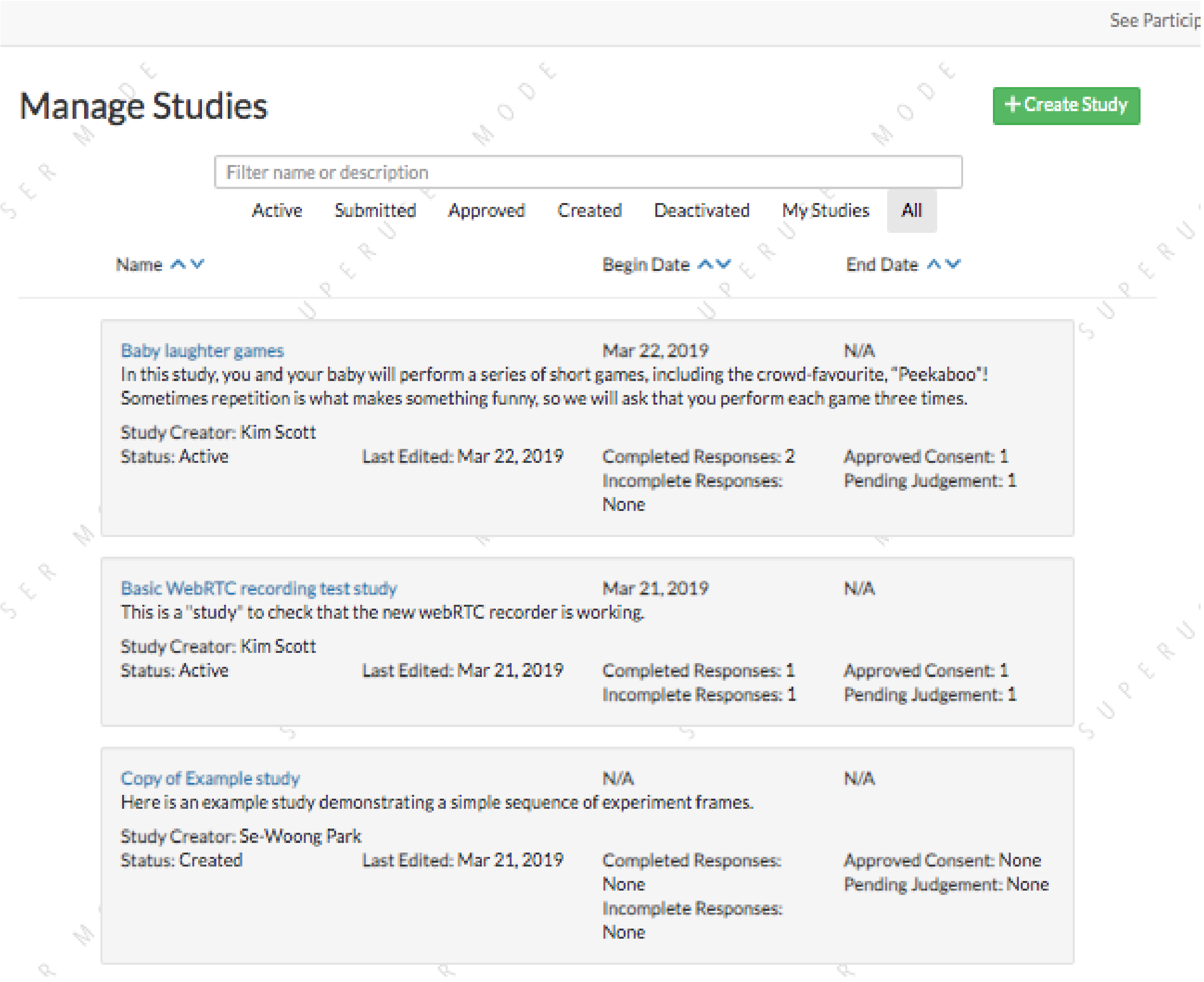
By design, there is no way to give anyone access to participant data for all studies in your lab. You have to give them permission to individual studies.

1.10 Managing studies

1.10.1 Viewing study list

To view all studies, navigate to /exp/studies/. (We will use this short format to indicate relative paths starting with the Lookit site you are using - e.g., https://lookit.mit.edu/exp/studies/ or https://lookit-staging.mit.edu/exp/studies/). From there, the researcher can only see studies they have permission to view. Lab members and admins can see all studies that belong to their lab. Otherwise, researchers can only view studies which they have created or to which they have been explicitly added.

You can filter studies by name or by keywords in the description. Additionally, you can sort on various study states like "Created" or "Submitted", or filter on your own studies by selecting "My Studies". You can also sort on study name, study end date, and study begin date.



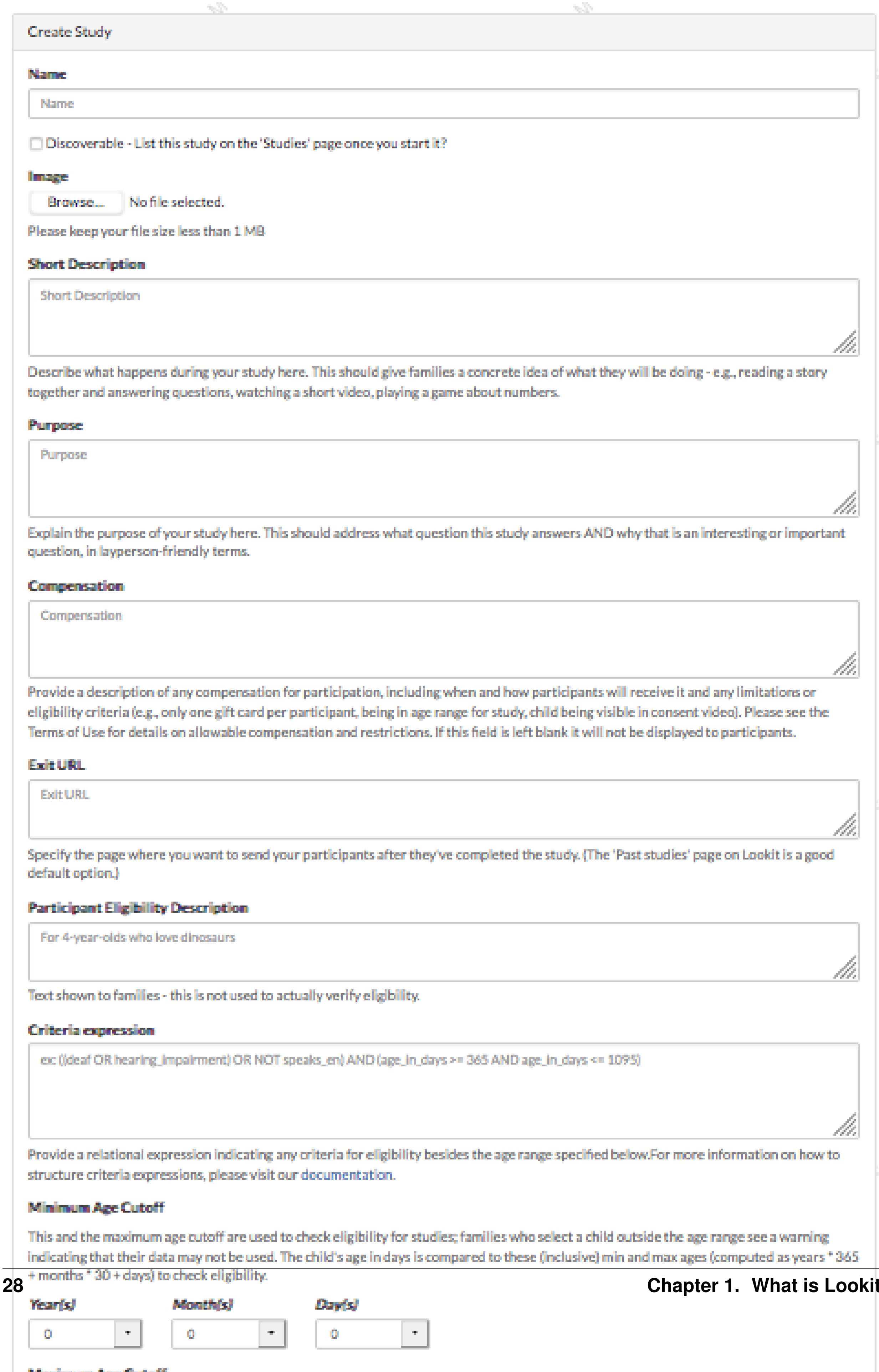
Baby laughter games Mar 22, 2019 N/A
In this study, you and your baby will perform a series of short games, including the crowd-favourite, "Peekaboo"! Sometimes repetition is what makes something funny, so we will ask that you perform each game three times.
Study Creator: Kim Scott
Status: Active Last Edited: Mar 22, 2019 Completed Responses: 2 Incomplete Responses: None Approved Consent: 1 Pending Judgement: 1

Basic WebRTC recording test study Mar 21, 2019 N/A
This is a "study" to check that the new webRTC recorder is working.
Study Creator: Kim Scott
Status: Active Last Edited: Mar 21, 2019 Completed Responses: 1 Incomplete Responses: 1 Approved Consent: 1 Pending Judgement: 1

Copy of Example study N/A N/A
Here is an example study demonstrating a simple sequence of experiment frames.
Study Creator: Se-Woong Park
Status: Created Last Edited: Mar 21, 2019 Completed Responses: None Incomplete Responses: None Approved Consent: None Pending Judgement: None

1.10.2 Creating a study

To create a study, click the green "Create Study" button on the study list page or navigate to /exp/studies/create/. You'll need to provide values for the fields as described in Setting study fields.



Create Study

Name

Name

□ Discoverable - List this study on the 'Studies' page once you start it?

Image

Browse... No file selected.

Please keep your file size less than 1 MB

Short Description

Short Description

Describe what happens during your study here. This should give families a concrete idea of what they will be doing - e.g., reading a story together and answering questions, watching a short video, playing a game about numbers.

Purpose

Purpose

Explain the purpose of your study here. This should address what question this study answers AND why that is an interesting or important question, in layperson-friendly terms.

Compensation

Compensation

Provide a description of any compensation for participation, including when and how participants will receive it and any limitations or eligibility criteria (e.g., only one gift card per participant, being in age range for study, child being visible in consent video). Please see the Terms of Use for details on allowable compensation and restrictions. If this field is left blank it will not be displayed to participants.

Exit URL

Exit URL

Specify the page where you want to send your participants after they've completed the study. (The 'Past studies' page on Lookit is a good default option.)

Participant Eligibility Description

For 4-year-olds who love dinosaurs

Text shown to families - this is not used to actually verify eligibility.

Criteria expression

ex: ((deaf OR hearing_impairment) OR NOT speaks_en) AND (age_in_days >= 365 AND age_in_days <= 1095)

Provide a relational expression indicating any criteria for eligibility besides the age range specified below.For more information on how to structure criteria expressions, please visit our documentation.

Minimum Age Cutoff

This and the maximum age cutoff are used to check eligibility for studies; families who select a child outside the age range see a warning indicating that their data may not be used. The child's age in days is compared to these (inclusive) min and max ages (computed as years * 365 + months * 30 + days) to check eligibility.

Year(s)	*Month(s)*	*Day(s)*
0	0	0

1.10.3 Cloning a study

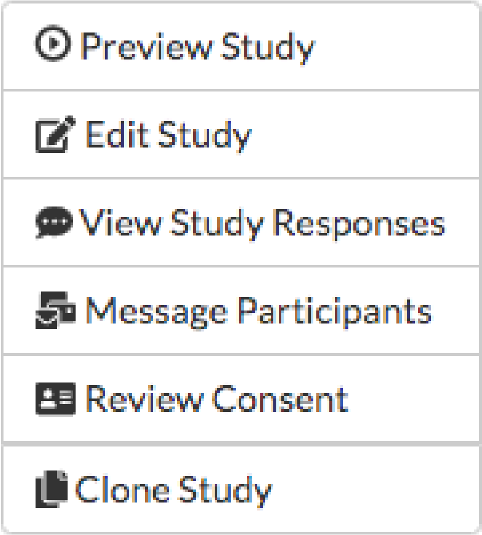
To clone a study, click the "Clone Study" button on an existing study detail page. In order to see this button, you will need to have permission to create studies in at least one lab - in general everyone has been added to the Sandbox lab, so even if you are not part of another lab, you will be able to clone studies you can see.

Cloning creates a copy of the study but adds the logged in user as the creator and puts it in a lab where the user is able to create studies. No data is copied along with the study. The clone will be moved back into "Created" status (e.g., if the current study is actively collecting data, the cloned study will not be - it will need to be approved before it can be started).

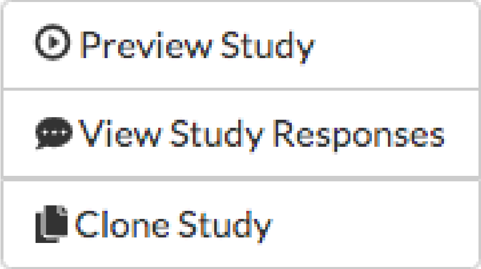
1.10.4 Study detail page

To view a single study, click on it from the study list. A researcher must have permission to view this study specifically. Lab admins and lab members can view all studies in their lab. Other researchers can only view this study if they have been explicitly added to the study. At the top, you see many of the study details that you entered when you created the study. The UUID is also displayed; this is your study's unique identifier and is used in the direct link to the study.

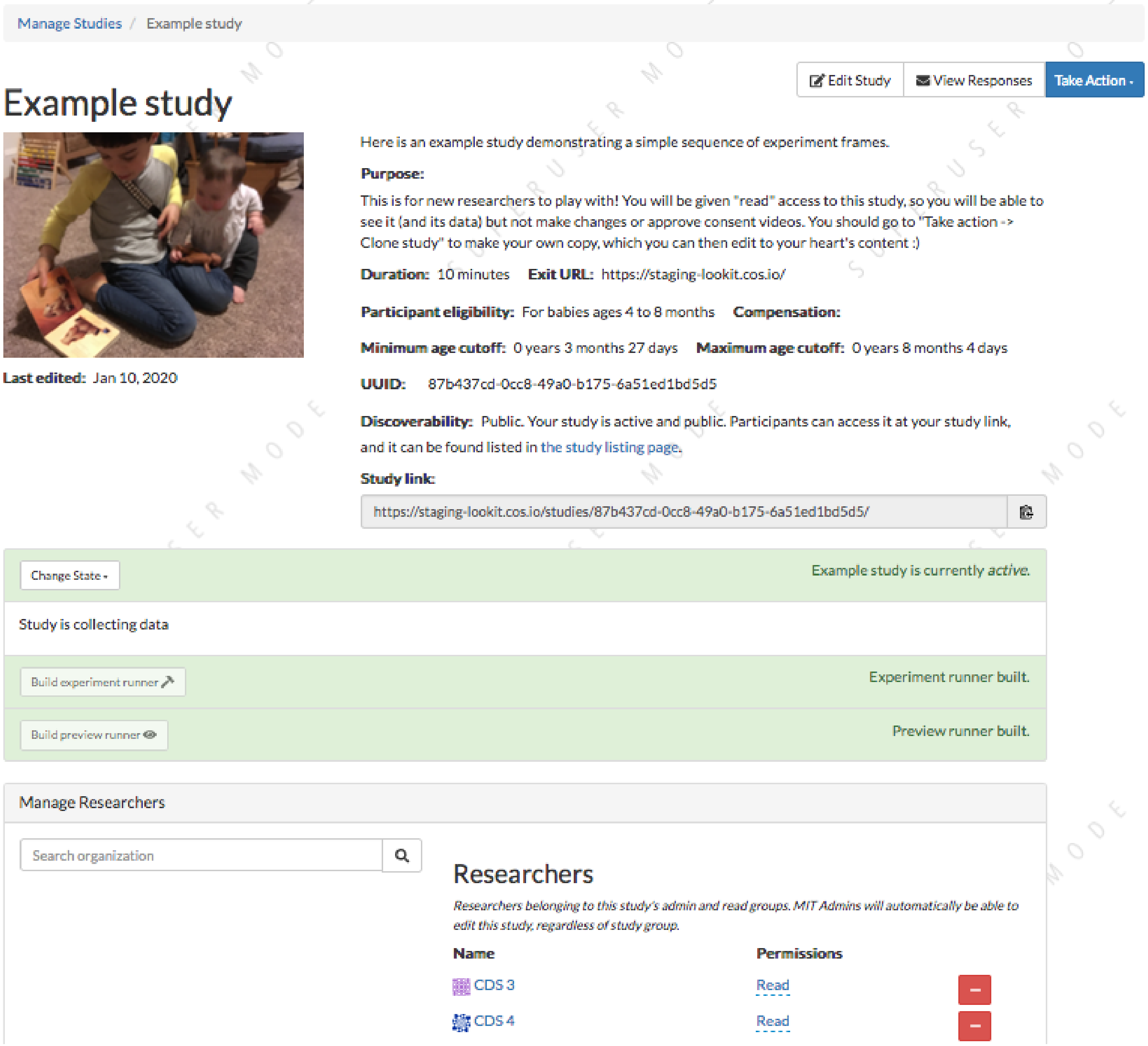
At the top right, you will see a set of buttons. Which actions are available depends on your permissions for this specific study. Here is the full set of options:



But if you have a study preview role, for example, you will only see some of these:



On this page, you can also view and edit study status, view and manage researchers who have access to the study. Study logs of when the study changed state are at the bottom of the page.



Manage Studies / Example study

Example study

Last edited: Jan 10, 2020

Here is an example study demonstrating a simple sequence of experiment frames.

Purpose:

This is for new researchers to play with! You will be given "read" access to this study, so you will be able to see it (and its data) but not make changes or approve consent videos. You should go to "Take action -> Clone study" to make your own copy, which you can then edit to your heart's content :)

Duration: 10 minutes **Exit URL:** https://staging-lookit.cos.io/

Participant eligibility: For babies ages 4 to 8 months **Compensation:**

Minimum age cutoff: 0 years 3 months 27 days **Maximum age cutoff:** 0 years 8 months 4 days

UUID: 87b437cd-0cc8-49a0-b175-6a51ed1bd5d5

Discoverability: Public. Your study is active and public. Participants can access it at your study link, and it can be found listed in the study listing page.

Study link:

https://staging-lookit.cos.io/studies/87b437cd-0cc8-49a0-b175-6a51ed1bd5d5/

Example study is currently *active.*

Study is collecting data

Experiment runner built.

Preview runner built.

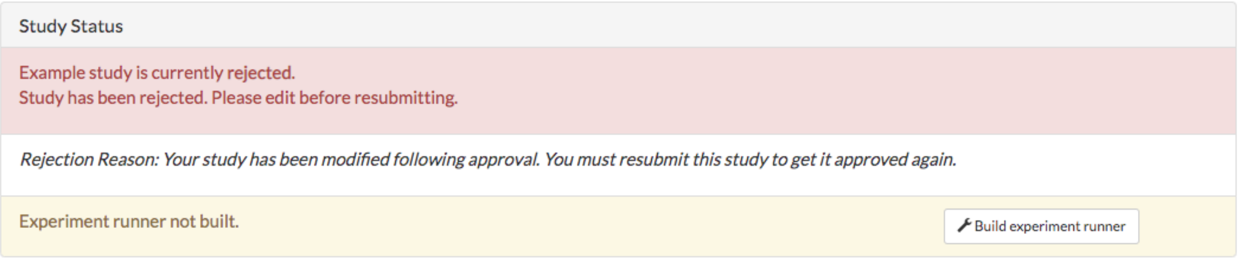
Manage Researchers

Researchers

Researchers belonging to this study's admin and read groups. MIT Admins will automatically be able to edit this study, regardless of study group.

Name	Permissions
CDS 3	Read
CDS 4	Read

1.10.5 Study status: submitting your study and managing data collection



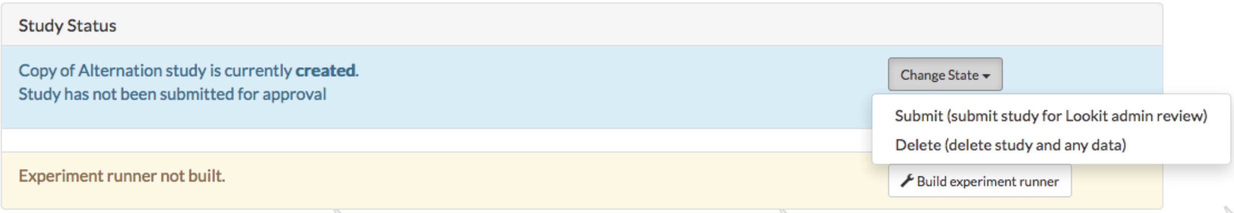
Study Status

Example study is currently rejected.
Study has been rejected. Please edit before resubmitting.

Rejection Reason: Your study has been modified following approval. You must resubmit this study to get it approved again.

Experiment runner not built.

On the study detail page, you can see the current status of a study. For users with permissions to change the study state there is also a dropdown with the available actions (start/pause data collection, submit for approval, etc.) as shown below. The available states where you can move the study depend on what state is next in the sequence, as well as your current level of permissions. For example, if a study's current state is "Created", that study can only be "Submitted" for review, or "Archived", which removes the study from display. Only Lookit admins can approve or reject a study.

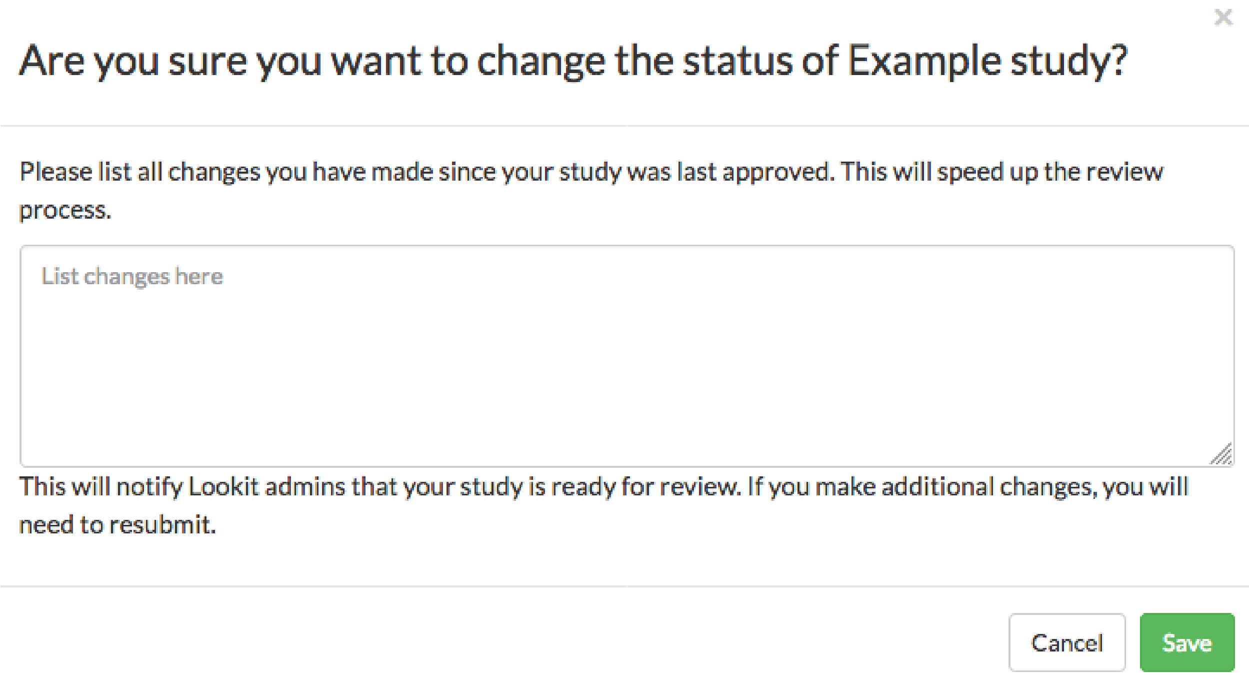

Study Status

Copy of Alternation study is currently **created**.
Study has not been submitted for approval

Experiment runner not built.
Change State ▾
Submit (submit study for Lookit admin review)
Delete (delete study and any data)
Build experiment runner

New studies must be submitted and approved by Lookit before they can be started. The *study approval process* is intended to give Lookit staff an opportunity to check that studies comply with the Terms of Use and to provide support if necessary.

When you submit a study for approval, you will be asked to provide some information to help with the review process - indicating any changes since last approval if you are re-submitting a previously approved study, and indicating anything that requires special review if you are submitting for the first time.


Are you sure you want to change the status of Example study?

Please list all changes you have made since your study was last approved. This will speed up the review process.

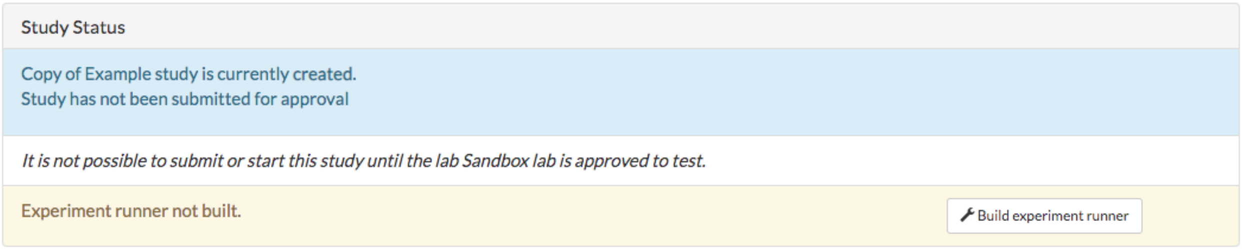
List changes here

This will notify Lookit admins that your study is ready for review. If you make additional changes, you will need to resubmit.

Researchers will receive email notifications when their study is approved or when changes are requested.

Once your study is approved, it is not automatically live and collecting data. Researchers with appropriate permissions can independently start/pause data collection at will; however, if any changes are made to the study it will be automatically rejected and will require re-approval.

Studies can only be submitted for approval once the associated Lab is approved to test on Lookit. If your Lab is not yet approved, for instance if you are using the Sandbox lab to try Lookit out, you will see a message like this:


Study Status

Copy of Example study is currently created.
Study has not been submitted for approval

It is not possible to submit or start this study until the lab Sandbox lab is approved to test.

Experiment runner not built.

The Lab is approved to test once it has a signed access agreement and someone from the group has completed the terms of use quiz. This is update manually but generally within a day of completing those steps; please contact us if you think your lab is incorrectly not approved to test yet.

Study state reference list

The possible study states are:

created Study has been initially created, but has not been submitted for approval

submitted Study is submitted and awaiting approval by a Lookit admin

approved Study has been approved by a Lookit admin to run on Lookit, but is not yet active

rejected Changes have been requested by a Lookit admin before the study can be approved. The study should be edited before resubmitting.

active Study is active and collecting data; participants can access it at the study link. If the study is also marked "Discoverable", the study will show up on Lookit's study list.

paused Study is not actively collecting data or visible on Lookit; participants cannot access it.

deactivated Study is done collecting data

archived Study has been archived and removed from search

1.10.6 Starting and stopping data collection

Starting and pausing data collection can be done instantly at any time after your study is approved, using the same dropdown menu as for submitting your study.

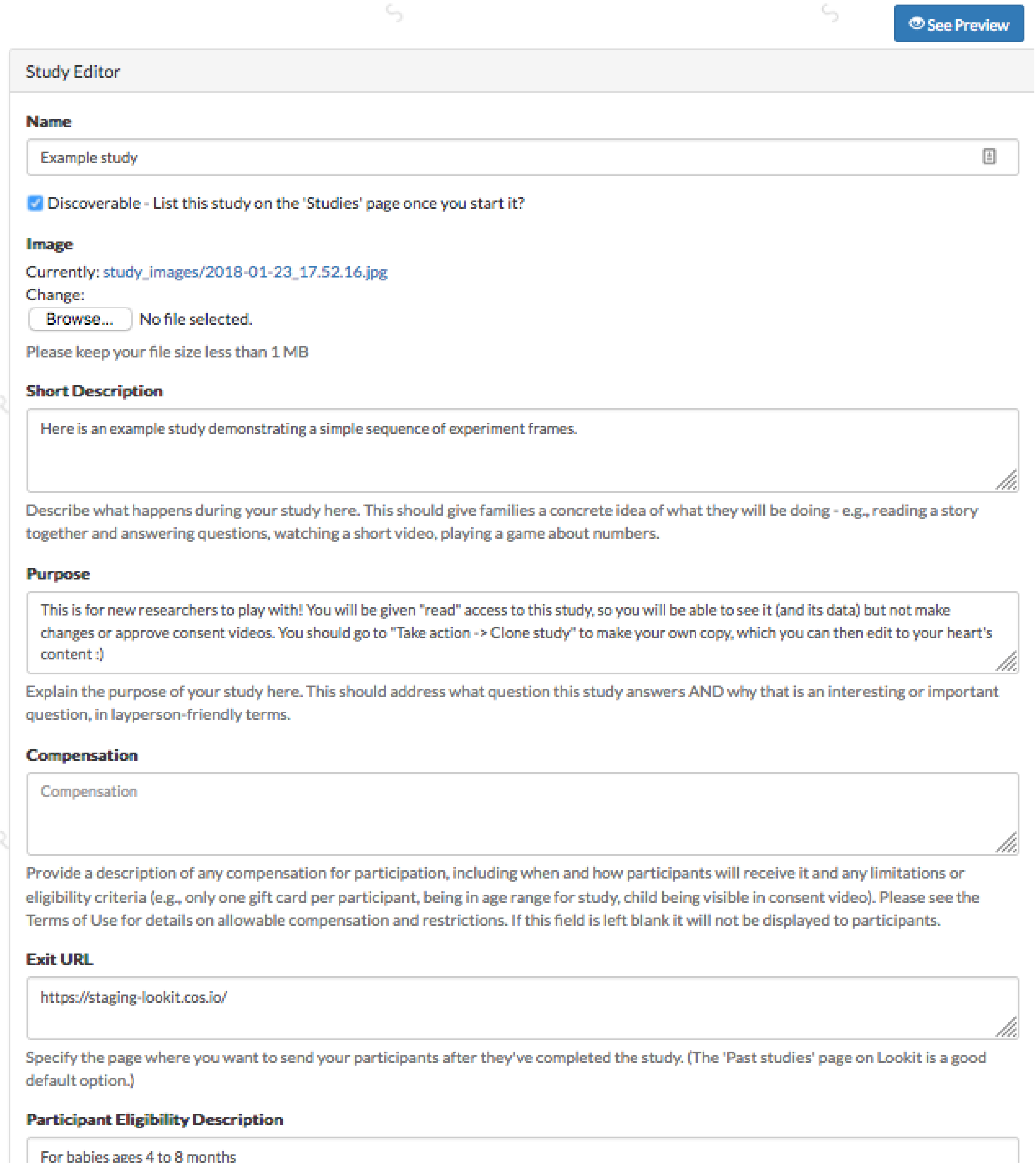
What does "starting" your study do? If your study is set as "discoverable" (one of the checkboxes under "edit study"), starting will add your study to the set of studies displayed at https://lookit.mit.edu/studies/, and anyone (including you) will be able to participate in it from there. If your study is set as non-discoverable, anyone will be able to participate via a direct link (shown on your study page in the experimenter interface). This is useful for studies intended for a very specific population, for instance if you're doing an online follow-up to an in-person study: you can email the direct link to families, without worrying about screening out other families on Lookit.

1.10.7 Study edit page

On the study edit page, you can update much of the metadata about the study. You can only view this page if you have permission to view the study details, meaning that you have been given a role specifically on this study OR you are a lab member. If you do not have permission to write study details, you will not be able to make any changes from this page. For more detail about the fields you can view and edit on this page, see "Setting study details."

To edit fields, change the information and click Save Changes in the middle of the page. If your study has already been approved, then the save button will be red. Otherwise it will be green. If your study has already been approved, then editing key details will automatically put the study in a rejected state. You must resubmit your study and get it approved again by a Lookit admin to run the study on the Lookit platform.

At the bottom of the edit study page, you can make edits to your study's structure (the frames, or pages, in your experiment), and the sequence of those frames. You can also make advanced edits to the commits we are using to build your study.



Study Editor

Name

Example study

☑ Discoverable - List this study on the 'Studies' page once you start it?

Image

Currently: study_images/2018-01-23_17.52.16.jpg

Change:

Browse... No file selected.

Please keep your file size less than 1 MB

Short Description

Here is an example study demonstrating a simple sequence of experiment frames.

Describe what happens during your study here. This should give families a concrete idea of what they will be doing - e.g., reading a story together and answering questions, watching a short video, playing a game about numbers.

Purpose

This is for new researchers to play with! You will be given "read" access to this study, so you will be able to see it (and its data) but not make changes or approve consent videos. You should go to "Take action -> Clone study" to make your own copy, which you can then edit to your heart's content :)

Explain the purpose of your study here. This should address what question this study answers AND why that is an interesting or important question, in layperson-friendly terms.

Compensation

Compensation

Provide a description of any compensation for participation, including when and how participants will receive it and any limitations or eligibility criteria (e.g., only one gift card per participant, being in age range for study, child being visible in consent video). Please see the Terms of Use for details on allowable compensation and restrictions. If this field is left blank it will not be displayed to participants.

Exit URL

https://staging-lookit.cos.io/

Specify the page where you want to send your participants after they've completed the study. (The 'Past studies' page on Lookit is a good default option.)

Participant Eligibility Description

For babies ages 4 to 8 months

1.10.8 Editing study protocol configuration

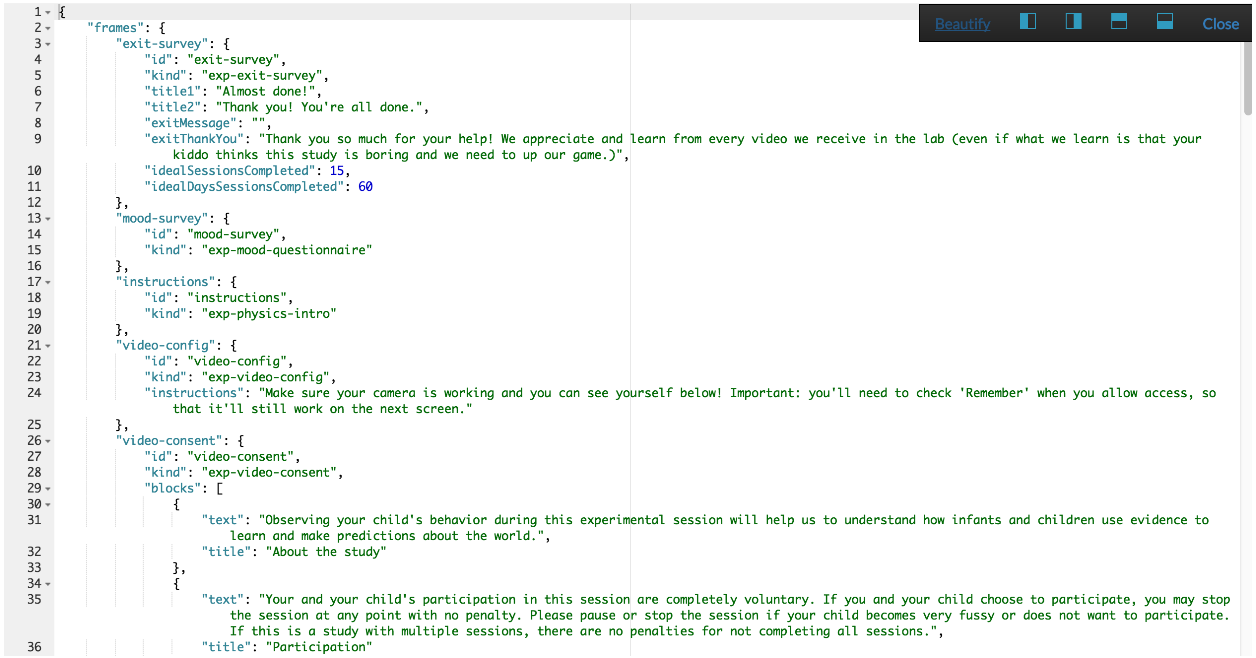
For more information about how to specify what happens during your study, see Building an Experiment. The study protocol configuration specifies the frames (or pages) of your experiment, and also specifies the order they go in.

To edit a study's protocol, click 'Edit study' from the study detail page. You will only be able to make actual changes from this page if you are a lab admin, or have a study admin, design, or manager role.

Click on the JSON block. A JSON editor will appear. Click on "Beautify" in the top right corner for better readability.

Note that any invalid JSON will be shown via a little red X at the left of the relevant line!

Once you are happy with your changes click ‘Close’. Then hit “Save Changes” in the bottom right corner. If your study has already been approved, then clicking “Save Changes” will automatically reject the study. You will have to resubmit it for a Lookit admin to reapprove.



```
{
    "frames": {
        "exit-survey": {
            "id": "exit-survey",
            "kind": "exp-exit-survey",
            "title1": "Almost done!",
            "title2": "Thank you! You're all done.",
            "exitMessage": "",
            "exitThankYou": "Thank you so much for your help! We appreciate and learn from every video we receive in the lab (even if what we learn is that your
                kiddo thinks this study is boring and we need to up our game.)",
            "idealSessionsCompleted": 15,
            "idealDaysSessionsCompleted": 60
        },
        "mood-survey": {
            "id": "mood-survey",
            "kind": "exp-mood-questionnaire"
        },
        "instructions": {
            "id": "instructions",
            "kind": "exp-physics-intro"
        },
        "video-config": {
            "id": "video-config",
            "kind": "exp-video-config",
            "instructions": "Make sure your camera is working and you can see yourself below! Important: you'll need to check 'Remember' when you allow access, so
                that it'll still work on the next screen."
        },
        "video-consent": {
            "id": "video-consent",
            "kind": "exp-video-consent",
            "blocks": [
                {
                    "text": "Observing your child's behavior during this experimental session will help us to understand how infants and children use evidence to
                        learn and make predictions about the world.",
                    "title": "About the study"
                },
                {
                    "text": "Your and your child's participation in this session are completely voluntary. If you and your child choose to participate, you may stop
                        the session at any point with no penalty. Please pause or stop the session if your child becomes very fussy or does not want to participate.
                        If this is a study with multiple sessions, there are no penalties for not completing all sessions.",
                    "title": "Participation"
```

To preview your study, click “See Preview”. (You will need to build an experiment runner first if you haven’t yet, or if you’ve changed the version you’re using.)

1.10.9 Editing experiment runner type

To edit the type of experiment runner used by your study, click ‘Edit study’ from the study detail page and scroll down to the bottom of the page.

The experiment runner is the application you’re using to enable participants to take a study. Right now, we just have one option, the Ember Frame Player. It’s an ember app that can talk to our API. All the frames in the experiment are defined in ember-lookit-frameplayer, and the exp-player component can cycle through these frames.

If you don’t want any customization and want to use the existing player and frames, just select the defaults. These are advanced options!

What does each field mean?

- The `Experiment runner code URL` is the GitHub repository where the frames and the player are stored. This is the default `player_repo_url`: https://github.com/lookit/ember-lookit-frameplayer. Advanced users may want to define their own custom frames for use with Lookit studies beyond those provided in the core library. (For more information about how to do this, see https://lookit.readthedocs.io/en/develop/developing-frames.html.) To use your own frame definitions, set `Experiment runner code URL` to your own fork of the ember-lookit-frameplayer repo (e.g., https://github.com/yourname/ember-lookit-frameplayer instead of https://github.com/lookit/ember-lookit-frameplayer).
- The `Experiment runner version (commit SHA)` is the specific version, or commit, of the experiment runner repository to use. Every time a change is made to the GitHub repository, it is assigned a unique identifier or “commit SHA.” If you don’t specify a version, then when you go to build your experiment runner, it will be use the most recent commit in the master branch and this field will get filled in. This way, your study

will continue to use exactly the same experiment player unless you deliberately choose to update - just in case any changes affect how your study works. If you do specify a version, some information about that version will be displayed to confirm, and you can click “Check for updates” at any time to view what has changed.

Important: Whenever you update the code versions you are using, you will need to re-build your re-build your experiment runner before you can preview or run your study. This build process creates your very own runner application using exactly the code you selected, so that your study will continue to run as you designed it. You only need to rebuild these when you have changed the code URL or version - not when you update your study protocol configuration or other data like the age range.

1.11 Study permissions

The permissions granted to researchers are quite granular, so that you have fine control over who can do/see what. This is so that you can grant access to just the minimum functions someone needs in their role, without any unnecessary risk to participant personal data.

If you have permissions to manage study researchers, you will be able to add/remove researchers to your study and change their roles on the Study Detail page.

1.11.1 Study roles

Researchers can be assigned any of the following roles relative to a study. Here are the permissions that each denotes (defined in more detail under “Study permission definitions” below).

	Preview	Design	Analysis	Submission processor	Researcher	Manager	Admin
READ_STUDY_DETAILS	x	x	x	x	x	x	x
READ_STUDY_PREVIEW_DATA	x	x	x	x	x	x	x
CODE_STUDY_PREVIEW_CONSENT	x	x	x	x	x	x	x
DELETE_ALL_PREVIEW_DATA		x	x	x	x	x	x
WRITE_STUDY_DETAILS		x				x	x
READ_STUDY_RESPONSE_DATA			x		x		x
CHANGE_STUDY_STATUS				x	x	x	x
CODE_STUDY_CONSENT				x	x		x
EDIT_STUDY_FEEDBACK				x	x		x
CONTACT_STUDY_PARTICIPANTS				x	x		x
CHANGE_STUDY_LAB							x
MANAGE_STUDY_RESEARCHERS						x	x

Note that only the Admin, Researcher, and Analysis roles allow access to download the study response data.

Here are some examples of cases where it would make sense to use each role:

Preview Someone in your lab that you might want to show the study design to for advice or feedback, or so they can clone it, but who is not involved otherwise.

Design The RA or programmer you’ve brought in to help you implement a study and make adjustments as needed, but not to handle actual data collection or analysis. They might not be on the IRB, and don’t need to be since they have no access to participant data.

Analysis Someone working on video coding as a study progresses (but not coding consent) or doing analysis of the dataset afterwards - e.g. someone who asks to conduct some secondary analysis after getting on your IRB.

Submission processor An RA who's handling day-to-day data collection: checking consent, writing to families to confirm consent if needed, sending gift cards, sending feedback, etc. They might start/pause the study based on how many kids are needed, if you run out of gift cards, etc. But they shouldn't be making changes to the protocol or description.

Researcher An RA who's handling day-to-day data collection AND possibly video coding or data analysis. They are still not responsible for making changes to the study or managing other people's access.

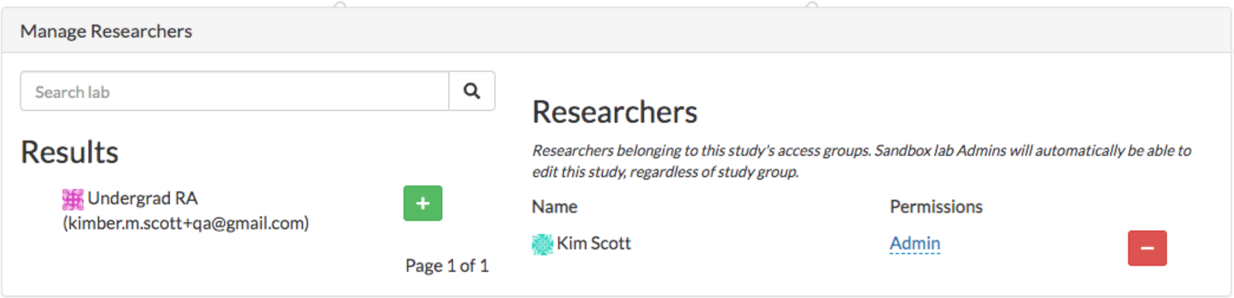
Manager A lab admin automatically gets this level of access for every lab study: they can do anything except access any participant data.

Admin A person primarily responsible for the study, who is trusted to manage access for others (e.g. RAs) and to make changes to the study protocol. Multiple students might be admins, or a student and PI. There always has to be at least one study admin.

1.11.2 Adding researchers to your study

You can only give permissions to people in the Lab associated with this study. (If you're creating a study in the Sandbox Lab, you will be able to share with anyone on Lookit.)

Halfway down the study detail page, you can see the researchers that have been added to your study. On the left, you can see researchers in this study's lab and search for a specific researcher.



Manage Researchers

Search lab

Results

Undergrad RA
(kimber.m.scott+qa@gmail.com)

Page 1 of 1

Researchers

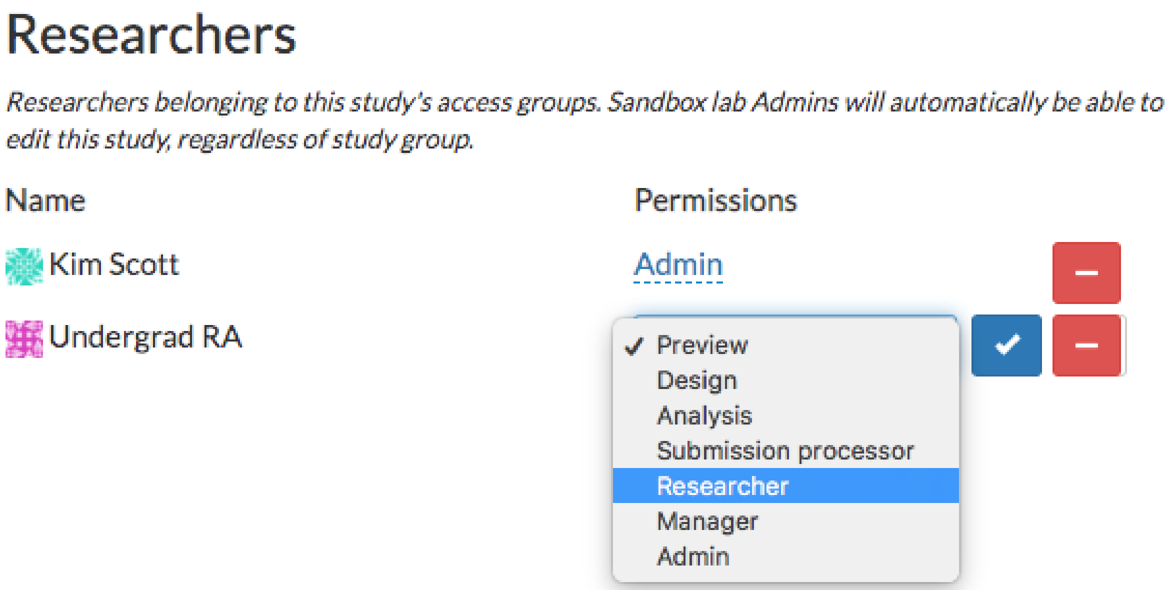
Researchers belonging to this study's access groups. Sandbox lab Admins will automatically be able to edit this study, regardless of study group.

Name	Permissions
Kim Scott	Admin

Click the green plus to add them to your study. They are given study preview permissions by default; this allows them to see all study details and preview the study, but not change anything about the study or view any participant data.

1.11.3 Editing researcher permissions on a study

To edit a researcher, select read or admin permissions in the dropdown beside the researcher name and click the checkmark. This will automatically give the researcher read or admin permissions. There must be at least one study admin at all times.



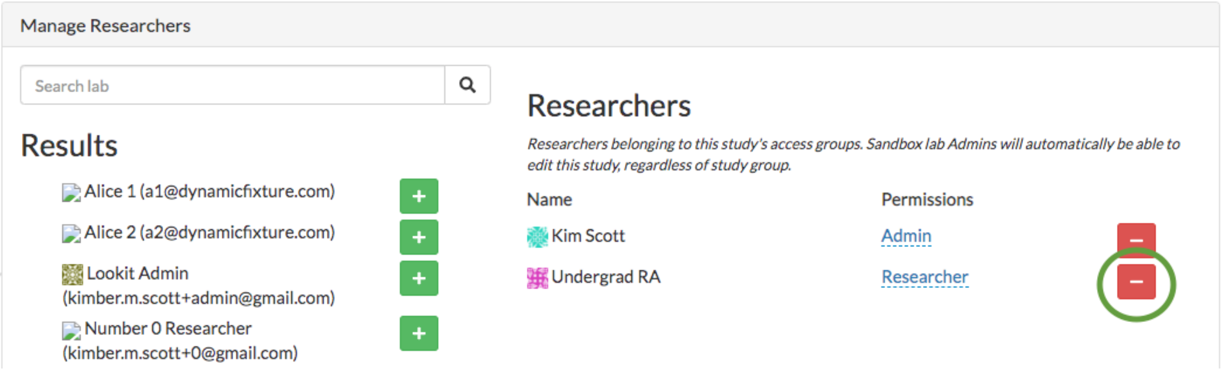
Researchers

Researchers belonging to this study's access groups. Sandbox lab Admins will automatically be able to edit this study, regardless of study group.

Name	Permissions
Kim Scott	Admin
Undergrad RA	Researcher

1.11.4 Deleting researcher permissions

To remove a researcher from a study, click the red minus button beside the researcher's name. This will automatically remove the user's study admin or study read permissions. There must be at least one study admin at all times, so it's possible that you won't be able to remove a researcher.



Manage Researchers

Search lab

Results

- Alice 1 (a1@dynamicfixture.com) +
- Alice 2 (a2@dynamicfixture.com) +
- Lookit Admin (kimber.m.scott+admin@gmail.com) +
- Number 0 Researcher (kimber.m.scott+0@gmail.com) +

Researchers

Researchers belonging to this study's access groups. Sandbox lab Admins will automatically be able to edit this study, regardless of study group.

Name	Permissions
Kim Scott	Admin
Undergrad RA	Researcher

Remember that lab members and lab admins will still have some permissions to the study (see *Lab permissions*), although they are not able to access participant data without being explicitly added.

1.11.5 Study permission definitions

Here are the specific permissions that govern what a researcher can see or do to study data. You will grant or restrict these permissions to other lab members by giving them roles on your study.

READ_STUDY_DETAILS Can preview the study; can see the study overview page where study researchers, logs, and status are shown; can see the study edit form where the protocol is (but not necessarily submit it!)

WRITE_STUDY_DETAILS Can make changes to the study using the edit study form - e.g., changing the protocol or age range. Note that if the study is already approved, making changes would automatically reject it; in this case, someone with WRITE_STUDY_DETAILS but not CHANGE_STUDY_STATUS would not be able to make changes.

CHANGE_STUDY_STATUS Can change the status of the study - starting or stopping data collection, submitting/retracting it for Lookit review, etc.

MANAGE_STUDY_RESEARCHERS Can add and remove other researchers in this lab to the various study groups, editing their permissions.

READ_STUDY_RESPONSE_DATA Can see and download actual participant data from this study.

READ_STUDY_PREVIEW_DATA Can see and download data from researchers who previewed this study. This is a separate permission because previewing the study and then seeing what data is generated is critical for study design & analysis pipeline planning (e.g. "ok, so I did this, do the videos I expected to make show up? how will our script tell what color the dax is on trial 4?"). It may also be a means of collecting feedback on the study design from other researchers (they preview a study, you see their responses).

CODE_STUDY_CONSENT Can submit rulings about whether consent videos demonstrate valid informed consent, which determines which data is then available for download

CODE_STUDY_PREVIEW_CONSENT Can submit rulings about whether consent videos from preview data demonstrate valid informed consent. (They don't really, but you might mock up the entire workflow from start to finish with preview data while planning your analyses and testing out your study.)

CONTACT_STUDY_PARTICIPANTS Can send email to participants via the Lookit platform. (This still does not grant direct access to participant email addresses.)

EDIT_STUDY_FEEDBACK Can create and edit feedback associated with participant responses, which is displayed to participants on the study history page.

CHANGE_STUDY_LAB Can change which lab this study is associated with. (Researchers not affiliated with the new lab are removed from the study upon changing.)

DELETE_ALL_PREVIEW_DATA Can use a button on the all responses page to delete existing preview data (generally useful in the process of developing a new study and testing it out repeatedly).

1.12 Setting study details

When creating or editing a study, you can set the value of the following fields. Below is more information about each:

1.12.1 Name

Participant-facing title of your study; must be <255 characters. Shoot for a short, catchy title; depending on how you advertise your study, you may want participants to be able to recognize and select it from the studies page. If you plan on running similar follow-up studies and want them to be easily distinguishable, avoid titles that encompass your entire research program like "Infant Language Study."

1.12.2 Discoverable

Do you want this study to be listed on the Lookit studies page when it's active, and eligible participants in the Lookit database to receive email invitations to participate? If so, check this box to make the study discoverable. Email invitations will go out to families with eligible children at a rate of up to 50 emails per day as long as the study is active and discoverable.

If the box is unchecked, the study will be 'non-discoverable' and participants will only be able to get to it by following a direct link with your study ID. This may be helpful if, for instance, you want to run a follow-up study (with in-lab on online participants) and want to send the link to a limited number of people, or if your inclusion criteria are very limited (e.g., a rare genetic disorder) and you want to recruit specifically without getting any random curious families stopping by.

You can switch the study back and forth from discoverable to non-discoverable any time after it's approved, without triggering re-review.

We recommend **starting** studies as non-discoverable, so that you can pilot with participants you recruit before inviting everyone who's eligible to participate!

1.12.3 Share preview

Do you want other researchers to be able to preview your study? Check this box to make it possible for any logged-in Lookit researcher to try out your study. If you check the box, you will be able to share your preview link - e.g. on the Slack channel - to ask for feedback on your study from other researchers. This is generally a good idea as we could all use another pair of eyes to check on directions, stimuli, debriefing text, etc. Getting peer feedback ahead of time will generally substantially speed up the Lookit review process too. You can leave this unchecked if you're very concerned about being scooped. (My personal feeling is that no one has the time or energy to scoop you. See also: every line of our code is publicly available and has been for years. . .)

1.12.4 Image

Thumbnail image that will be displayed to participants on Lookit's studies page. File must be an image-type, and please keep the file size reasonable (<1 MB). If you submit too large an image file you may see an error "413 Request Entity Too Large."

Sometimes your stimuli are a good basis for creating this image, or it can be something that conceptually represents your study or shows what it looks like to participate.

As noted in the self-review checklist, if you decide to include an image of a child/family participating, please don't use pictures of white people if you have flexibility not to. (It's as good a choice as any for a single study, but the problem is that especially US researchers will "default" to white people as examples, and if everyone does that we end up with a sea of pictures of white kids on the studies page. It's a small thing, but it stinks to only see pictures of families that look like yours in cases where the researchers are studying something related to race!)

1.12.5 Short description

Describe what happens during your study here (1-3 sentences). This should give families a concrete idea of what they will be doing - e.g., reading a story together and answering questions, watching a short video, playing a game about numbers.

1.12.6 Purpose

Explain the purpose of your study here (1-3 sentences). This should address what question this study answers AND why that is an interesting or important question, in layperson-friendly terms. Note: this tends to be harder than you'd think - it's not just you! Imagine all the time you spend getting comfortable explaining the point of a study in the lab (or training RAs on the same), distilled into this task. Plus you don't get to interact with the parent to gauge their interest level or familiarity first. Take your time and read this out loud as you work. Some things to check: Is it too specific - is a reasonable response "okay, you will find out whether X is true, but why does that matter?" Is it too general - could you write the same thing about a follow-up study you're planning or another study going on in your lab?

1.12.7 Compensation

Provide a description of any compensation for participation, including when and how participants will receive it and any limitations or eligibility criteria (e.g., only one gift card per participant, being in age range for study, child being visible in consent video). Please see the Terms of Use for details on allowable compensation and restrictions. If this field is left blank (which is okay if you're not providing compensation beyond the joy of participation) it will not be displayed to participants.

1.12.8 Exit URL

Must enter a URL. After the participant has completed the study, they will be automatically redirected to the exit URL. Typically this is just *https://lookit.mit.edu/*

1.12.9 Participant eligibility description

Freeform participant-facing eligibility string, of the form 'For…' (e.g., 'For babies under 1 year old'). Make this readable so participants understand if their child can take part in the study.

This is **not** directly used to automatically check eligibility, so you can include criteria that you can't check for automatically - e.g., this study is only for kids whose favorite color is orange.

Age limits specified here should be carefully considered with respect to the *minimum and maximum age cutoffs* which **are** used for automatic verification of eligibility.

How does eligibility work?

There are two separate ways you specify eligibility criteria for your study: the "automatically checkable" parts (criteria expression and min/max ages, discussed below), and the "parent-facing description" part (above).

The "automatically checkable" parts are used for several things:

- Showing parents a warning if they try to participate with a child who's not eligible
- Determining which registered families to email - announcement emails are sent out to families about discoverable studies their children are eligible for
- [Coming soon] Letting parents filter the list of active studies by which ones their kids are eligible for

For now, though, because the criteria expressions aren't guaranteed to be easy to read/interpret - and because you might have additional criteria that aren't in the database anywhere - these are separate from the description displayed to parents, which you have to provide manually.

1.12.10 Criteria expression

Providing this expression allows you to specify more detailed eligibility criteria for your study than a single age range. When a parent selects a child to participate in a study, he or she will see a warning under any of the following conditions:

- The child is under the minimum age specified (see *minimum and maximum age cutoffs*)
- The child is over the maximum age specified (see *minimum and maximum age cutoffs*)
- The child is within the specified age range, but doesn't meet the eligibility criteria defined in this expression

Note that while a warning is displayed, ineligible participants are not actually prevented from participating; this is deliberate, to remove any motivation for a curious parent to fudge the details to see what the study is like.

You may want to use the criteria expression to specify additional eligibility criteria beyond an age range - for instance, if your study is for a special population like kids with ASD or bilingual kids. In general, do **not** specify your age range here; participant eligibility checks will require the child meet the *minimum and maximum age cutoffs* AND these critera.

Every child in the Lookit database has a number of fields associated with it, ranging from gestational age to languages spoken in the home, which can be used in determining eligibility. In the study edit and create views, you can formulate your criteria expression as a boolean expression with embedded relational expressions, using a domain specific query language.

You can put together your expressions using the query fields below; the operators *AND*, *OR*, *NOT*, <, <=, =, >, and >=; and parentheses. If your expression is invalid you will see an error when you try to save your study.

Query fields

Query Handle	Value Type	Examples	Notes
[*CONDITIONS*]	N/A	deaf, hearing_impairment, NOT multiple_birth	See below for full list of available options.
speaks_[*LANGCODE*]	N/A	speaks_en, NOT speaks_ja, speaks_ru	See below for full list of available options.
n_languages	integer	0, 1, 2	Number of languages child is exposed to
gestational_age_in_weeks	integer or string	gestational_age_in_weeks <= 40, gestational_age_in_weeks = na	Values are 23 through 40 and na
gender	string	gender = f, gender !=o	Male (m), Female (f), Other (o), or Not Available (na).
age_in_days	integer	age_in_days <= 1095, age_in_days > 365	

Criteria expression examples

Deaf children only `deaf`

Multiple-birth children who are either under 1 year old or over 3 years old `multiple_birth AND (age_in_days >= 1095 OR age_in_days <= 365)`

Girls who are exposed to both English and Spanish `gender = f AND speaks_en AND speaks_es`

Children born late preterm whose adjusted age is about 6 weeks `(gestational_age_in_weeks = 34 AND (age_in_days >= 72 AND age_in_days < 102)) OR (gestational_age_in_weeks = 35 AND (age_in_days >= 65 AND age_in_days < 95)) OR (gestational_age_in_weeks = 36 AND (age_in_days >= 58 AND age_in_days < 88))`

Characteristics and conditions

Query Handle	Condition/Characteristic
autism_spectrum_disorder	Autism Spectrum Disorder
deaf	Deaf
hearing_impairment	Hearing Impairment
dyslexia	Dyslexia
multiple_birth	Multiple Birth (twin, triplet, or higher order)

Language codes

Code	Language
en	English
am	Amharic
bn	Bengali
bho	Bhojpuri
my	Burmese
ceb	Cebuano
hne	Chhattisgarhi
nl	Dutch
egy	Egyptian Spoken Arabic
fr	French
gan	Gan
de	German
gu	Gujarati
hak	Hakka
ha	Hausa
hi	Hindi
ig	Igbo
id	Indonesian
pes	Iranian Persian
it	Italian
ja	Japanese
jv	Javanese
cjy	Jinyu
kn	Kannada
km	Khmer
ko	Korean
mag	Magahi
mai	Maithili
ms	Malay
ml	Malayalam
cmn	Mandarin
mr	Marathi
nan	Min Nan
mor	Moroccan Spoken Arabic
pbu	Northern Pashto
uzn	Northern Uzbek
or	Odia

Continued on next page

Table 1 – continued from previous page

Code	Language
pl	Polish
pt	Portuguese
ro	Romanian
ru	Russian
skr	Saraiki
sd	Sindhi
so	Somali
es	Spanish
su	Sunda
tl	Tagalog
ta	Tamil
te	Telugu
th	Thai
tr	Turkish
uk	Ukrainian
ur	Urdu
vi	Vietnamese
lah	Western Punjabi
wuu	Wu
hsn	Xiang Chinese
yo	Yoruba
yue	Yue

1.12.11 Minimum and maximum age cutoffs

Integer fields specifying minimum/maximum ages of participants (inclusive). Eligibility is calculated based on the child's current age in days; this is compared to the minimum/maximum ages in days, calculated as 365*years + 30*months + days. Participants under the age range see a warning indicating that their data may not be used, and suggesting that they wait until they're in the age range. Participants over the age range just see a warning indicating that their data may not be used. Participants are never actually prevented from starting the study, to remove motivation for a curious parent to fudge the child's age.

Note that these ages do **not** in all cases correspond exactly to the child's age in 'calendar months' or 'calendar years' (e.g., 'one month' if that month is February). In general, you want to avoid a situation where the parent thinks their child should be eligible based on the participant eligibility string (e.g., "my child is one month old, she was born February 3rd and it's March 4th!") but sees a warning when trying to participate. You can do this by narrowing the eligibility criteria in the freeform string and/or by expanding them in the cutoffs here. If one has to align better with your actual inclusion criteria, in general you want that to be the minimum/maximum age cutoffs.

Please see this spreadsheet for a table translating "calendar ages" (how a parent would describe their child's age) to days.

Example: study for 5- and 6-year-olds

These kids will have lived through 1 or 2 leap years (at both ends of the age range), so the range you likely want is 5 * 365 + 1 days up to 6 * 365 + 2 days. Set the age range as 5 years, 1 day to 6 years, 2 days.

Example: study for 6-month-olds (i.e., between 6 and 7 months)

A child turns 6 months old, by the calendar, between 181 (e.g. born in September in a non leap year) and 184 (e.g. born in March) days of age. She turns 7 months old, by the calendar, between 212 days (e.g., born in August in a non leap year) and 216 days (e.g., born in July preceding a leap year). If you really want to include anyone who's "six months old" you could set the age range to 181 to 216 days by selecting 6 months 1 day 7 months 6 days. This way no one who thinks, quite reasonably, that their baby is 6 months old will see a warning that they're not eligible.

If you have theoretical reasons for wanting a particular exact age range in days, you could set that instead, and then communicate it to parents: e.g. "for babies around 6 months old (26 to 30 weeks)".

Example: study for 6-month-olds (i.e., between 5.5 and 6.5 months)

Another common standard in the literature is to report a finding in "N-month-olds," meaning babies between (N-1).5 and N.5 months of age. Actual implementations of this in terms of recruitment from databases vary, and historically we suspect in most cases researchers got what they got and then reported the range of kids they actually tested, rather than having an actual age range set in stone.

Here you might focus on how old babies are when they "turn" six months and then frame the age range in terms of that: e.g., go from 181 - 14 to 184 + 14 days, or 167 to 198 days, and describe this as being "within two weeks before or after their six-month 'birthday'."

1.12.12 Duration

Approximately how long does it take to do your study, start to finish? (Try it if you're not sure; include time to read the instructions.) You can give an estimate or range.

1.12.13 Researcher contact information

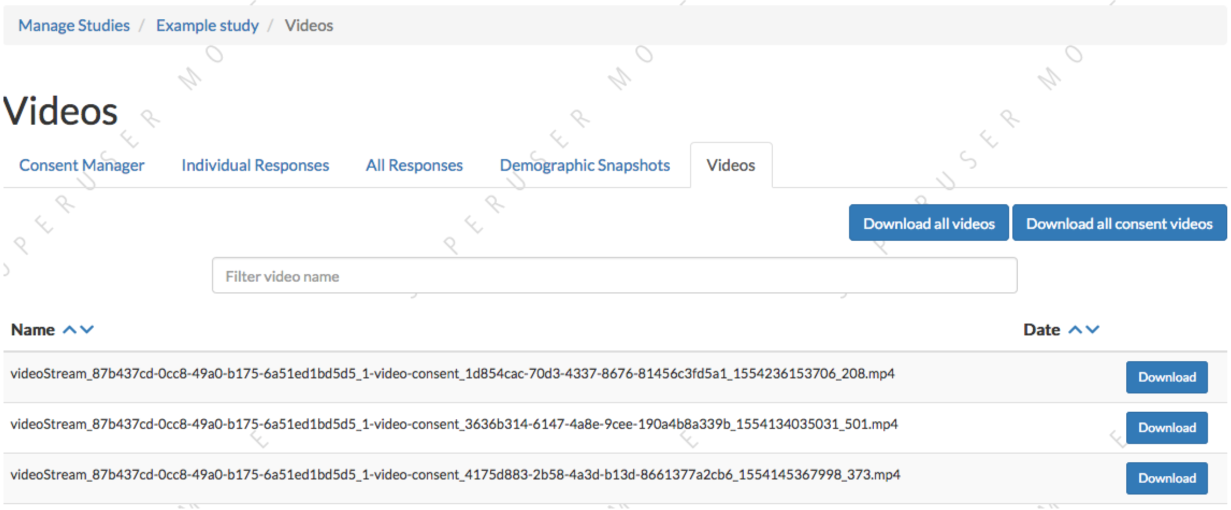
This should give the name of the PI for your study, and an email address where the PI or study staff can be reached with questions. Format: PIs Name (contact: youremail@lab.edu). This is displayed to participants on the study detail page before they choose to participate, as well as substituted into your consent form and exit survey, so in general the name needs to be the person who's listed as PI on your IRB protocol (although it may not need to be their personal email address).

1.12.14 Study protocol configuration

This needs to be a valid JSON block describing the different frames (pages) of your study, and the sequence. This can be left blank at the time you initially create your study. For detailed information about specifying your study protocol, see Building an Experiment.

1.12.15 Experiment runner type

The study type is the application you're using to enable participants to take a study. Right now, we just have one option, the Ember Frame Player. It's an ember app that can talk to our API. All the frames in the experiment are defined in Ember and there is an exp-player component that can cycle through these frames. For details, see Editing study type



Videos

Name ^v	Date ^v	
videoStream_87b437cd-0cc8-49a0-b175-6a51ed1bd5d5_1-video-consent_1d854cac-70d3-4337-8676-81456c3fd5a1_1554236153706_208.mp4		Download
videoStream_87b437cd-0cc8-49a0-b175-6a51ed1bd5d5_1-video-consent_3636b314-6147-4a8e-9cee-190a4b8a339b_1554134035031_501.mp4		Download
videoStream_87b437cd-0cc8-49a0-b175-6a51ed1bd5d5_1-video-consent_4175d883-2b58-4a3d-b13d-8661377a2cb6_1554145367998_373.mp4		Download

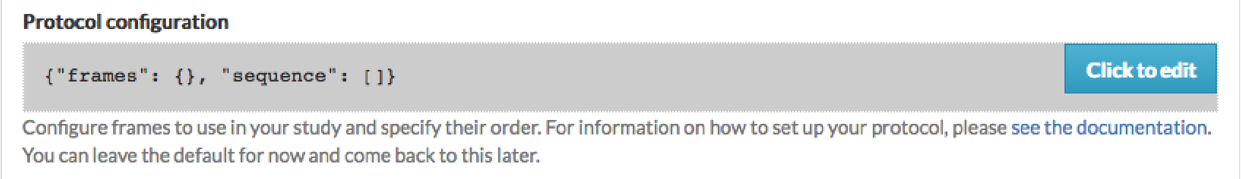
1.13 Protocol specification

Researchers specify how their Lookit study works by writing a "protocol configuration" for their study. This configuration is written in JSON, which stands for JavaScript Object Notation - this is just a special text format, not code.

In the configuration, you essentially tell Lookit what sequence of "frames" to use in your study, and set all the options for those frames like what pictures or videos to show and for how long. You can see the available frames in the experiment runner docs.

1.13.1 Experiment structure

To define what actually happens in your study, click 'Edit study' from your study detail page, and scroll down to the 'Protocol configuration' field:


Protocol configuration

```
{"frames": {}, "sequence": []}
```

Configure frames to use in your study and specify their order. For information on how to set up your protocol, please see the documentation. You can leave the default for now and come back to this later.

Click on this field to bring up the experiment editor view. Here is where you define the structure of your experiment using a JSON document. (Advanced users can choose to instead provide a 'protocol generator function', written in Javascript, which *returns* a JSON document to use as the study protocol.)

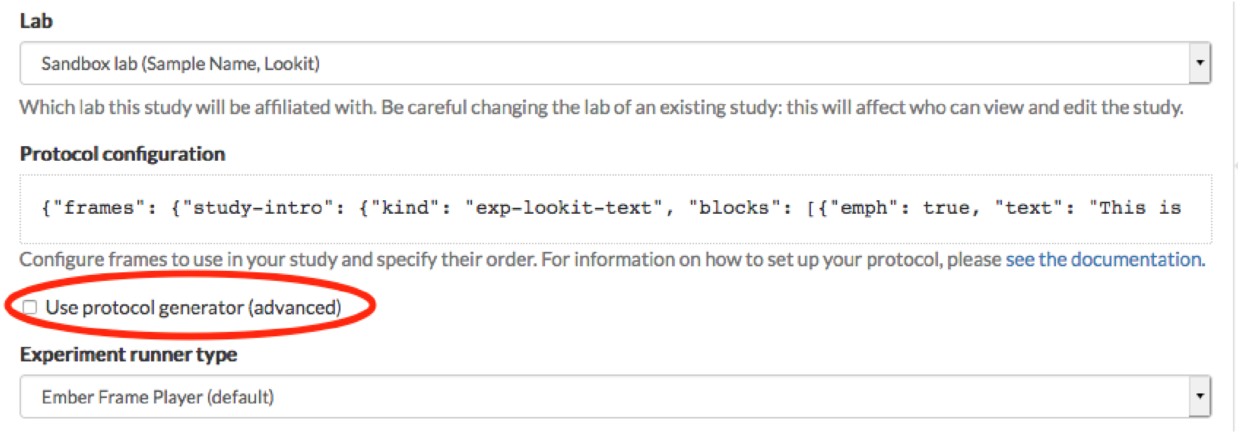
Studies on Lookit are broken into a set of fundamental units called **frames**, which can also be thought of as "pages" of the study. A single experimental trial (e.g. looking time measurement) would generally be one frame, as are the video consent procedure and exit survey. Your study protocol will define a set of `frames` and also a `sequence` saying the order in which to use those frames.

For detailed information about how to specify your study protocol, see the experiment runner documentation.

How to add a protocol generator function

Although your study protocol JSON can be configured to handle a wide range of common condition assignment, counterbalancing, and conditional logic schemes, in some cases it may still be more natural to programmatically generate the protocol. For examples and details about how to write a study protocol generator function, see the experiment runner documentation.

On your study edit form, check the "Use protocol generator (advanced)" box to use a protocol generator function in place of your study protocol:



Lab

Sandbox lab (Sample Name, Lookit)

Which lab this study will be affiliated with. Be careful changing the lab of an existing study: this will affect who can view and edit the study.

Protocol configuration

```
{"frames": {"study-intro": {"kind": "exp-lookit-text", "blocks": [{"emph": true, "text": "This is
```

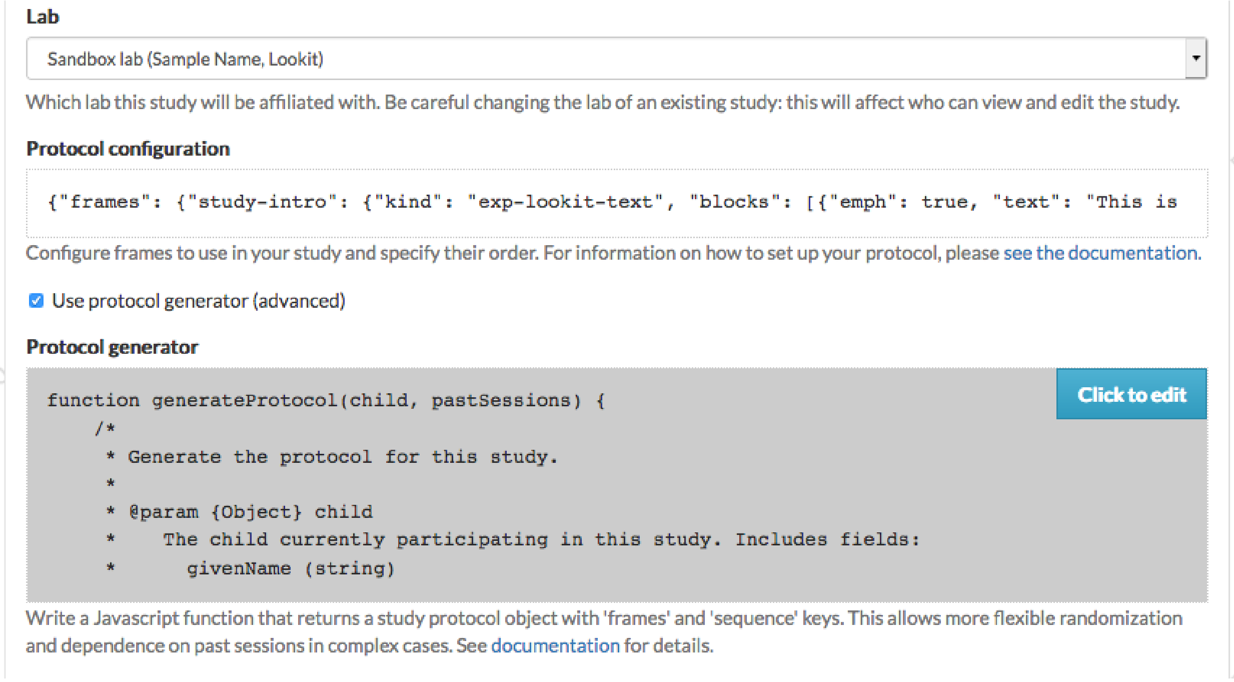
Configure frames to use in your study and specify their order. For information on how to set up your protocol, please see the documentation.

□ Use protocol generator (advanced)

Experiment runner type

Ember Frame Player (default)

This displays a field where you can edit a protocol generator function. The default generator returns an empty protocol. When you click on this field, you're taken to an editor like the one for the study protocol.



Lab

Sandbox lab (Sample Name, Lookit)

Which lab this study will be affiliated with. Be careful changing the lab of an existing study: this will affect who can view and edit the study.

Protocol configuration

```
{"frames": {"study-intro": {"kind": "exp-lookit-text", "blocks": [{"emph": true, "text": "This is
```

Configure frames to use in your study and specify their order. For information on how to set up your protocol, please see the documentation.

☑ Use protocol generator (advanced)

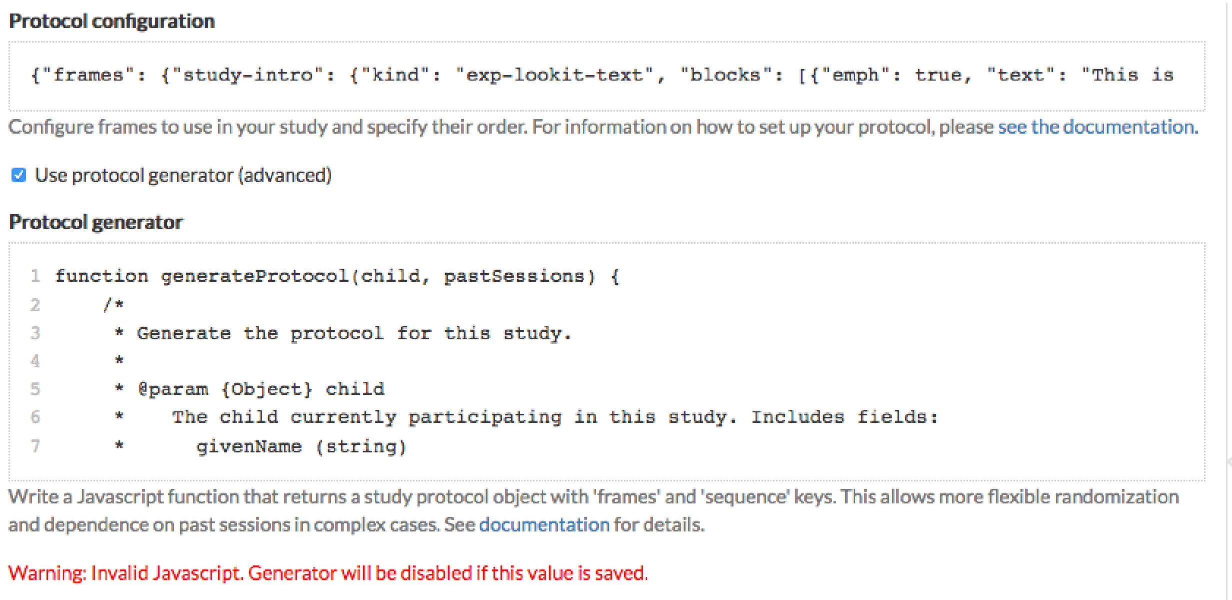
Protocol generator

```
function generateProtocol(child, pastSessions) {
    /*
     * Generate the protocol for this study.
     *
     * @param {Object} child
     *    The child currently participating in this study. Includes fields:
     *      givenName (string)
```

Write a Javascript function that returns a study protocol object with 'frames' and 'sequence' keys. This allows more flexible randomization and dependence on past sessions in complex cases. See documentation for details.

The code you write should define a single function - put any helper functions you need inside.

If your function isn't valid Javascript or doesn't evaluate to a function, you'll see a message that it's invalid, like this:



Protocol configuration

```
{"frames": {"study-intro": {"kind": "exp-lookit-text", "blocks": [{"emph": true, "text": "This is
```

Configure frames to use in your study and specify their order. For information on how to set up your protocol, please see the documentation.

☑ Use protocol generator (advanced)

Protocol generator

```
function generateProtocol(child, pastSessions) {
    /*
     * Generate the protocol for this study.
     *
     * @param {Object} child
     *    The child currently participating in this study. Includes fields:
     *      givenName (string)
```

Write a Javascript function that returns a study protocol object with 'frames' and 'sequence' keys. This allows more flexible randomization and dependence on past sessions in complex cases. See documentation for details.

Warning: Invalid Javascript. Generator will be disabled if this value is saved.

Your protocol generator will still be saved when you save the study, but the "use protocol generator" box will automatically be unchecked so that your study goes back to relying on the study protocol JSON. If there is an error when the protocol generator actually runs at the start of a session, the experiment runner also falls back to using the study protocol JSON.

1.13.2 How to try out your study as you write it

When you first create your study, you'll need to click 'Build experiment runner' on your study page and wait 5-10 minutes for your own personal experiment runner to be created. This will "freeze" the version of the experiment runner code used for your study, so that updates to the Lookit experiment runner won't affect how your study works. (You can always update if you want to - see Updating the frameplayer code). You do not need to build the experiment runner again unless you want to update the code it uses.

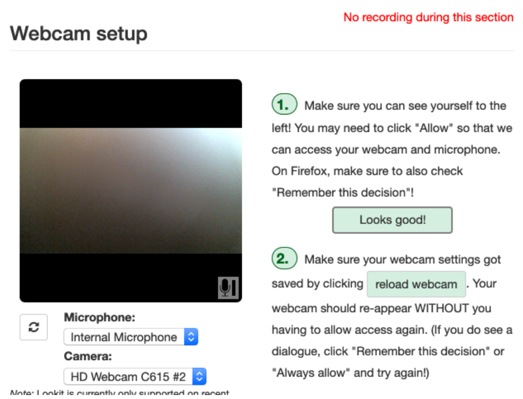
Once you've built an experiment runner, you can click 'See preview' after saving your study protocol and you will be able to preview your study, exactly as if you were participating with your child. As you write the protocol configuration for your study, you can click 'See preview' again or just refresh the preview window to see how the changes look.

As you work on a particular frame like a survey, you probably don't want to click through every bit of your study to get to it each time you make a change! You can put the frame of interest at the very start of your study by inserting it at the very start of the 'sequence' you've defined in your protocol. Then when you're satisfied with that frame, just put it back in order.

1.13.3 Keep your developer tools open!

Helpful warning and error messages will show up in the web console of your browser's Developer Tools. It's important to have this open while you test things out so you see anything going wrong, and it'll be helpful if you need to figure out why something isn't working as expected! For example, maybe there's a frame type that isn't defined, or it's trying to load an image that you haven't uploaded yet.

In **Firefox**, open up the web console by clicking the hamburger menu in the top right corner, Web Developer, then Web Console. In **Chrome**, open up the web console by clicking the three vertical dots in the top right corner, More tools, then Developer Tools. You should see something like this:



No recording during this section

Webcam setup

1. Make sure you can see yourself to the left! You may need to click "Allow" so that we can access your webcam and microphone. On Firefox, make sure to also check "Remember this decision"!

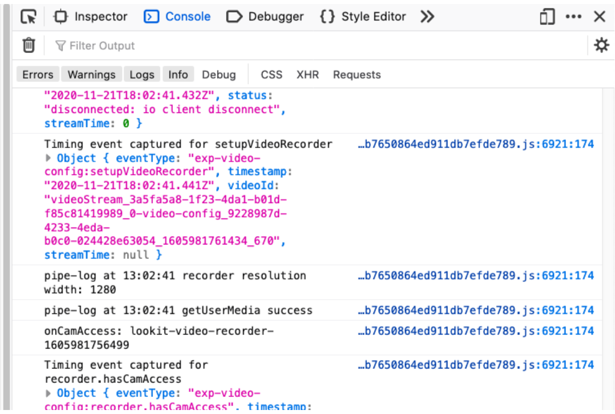
Looks good!

2. Make sure your webcam settings got saved by clicking reload webcam. Your webcam should re-appear WITHOUT you having to allow access again. (If you do see a dialogue, click "Remember this decision" or "Always allow" and try again!)

Microphone: Internal Microphone

Camera: HD Webcam C615 #2

Note: Lookit is currently only supported on recent



```
"2020-11-21T18:02:41.432Z", status:
"disconnected: io client disconnect",
streamTime: 0 }
Timing event captured for setupVideoRecorder     …b7650864ed911db7efde789.js:6921:174
▸ Object { eventType: "exp-video-
config:setupVideoRecorder", timestamp:
"2020-11-21T18:02:41.441Z", videoId:
"videoStream_3a5fa5a8-1f23-4da1-b01d-
f85c81419989_0-video-config_9228987d-
4233-4eda-
b0c0-024428e63054_1605981761434_670",
streamTime: null }
pipe-log at 13:02:41 recorder resolution         …b7650864ed911db7efde789.js:6921:174
width: 1280
pipe-log at 13:02:41 getUserMedia success        …b7650864ed911db7efde789.js:6921:174
onCamAccess: lookit-video-recorder-              …b7650864ed911db7efde789.js:6921:174
1605981756499
Timing event captured for                        …b7650864ed911db7efde789.js:6921:174
recorder.hasCamAccess
▸ Object { eventType: "exp-video-
config:recorder.hasCamAccess", timestamp:
```

1.13.4 Finding and using specific frames

For the most current documentation of individual frames available to use, please see the experiment runner documentation.

For each frame, you will find an **example** of using it in a JSON schema; documentation of the **properties** which can be defined in the schema; a description of the **data** this frame records; and any frame-specific **events** that are recorded and may be included in the eventTimings object sent with the data.

1.13.5 Recording webcam video

Some frames include functionality to record video from the participant's webcam during some or all of the frame. This will be described in the frame's documentation, including any parameters you can set to turn on/off or otherwise change the behavior of the recording. Recording may start/stop automatically in the background, or the participant may click to start and stop recording or even immediately view their recording. For test trials, the webcam is generally not displayed to the participant while recording, as it would be more interesting than almost all stimuli we could create.

You also have the option to create multi-frame recordings by starting and stopping recording using the exp-lookit-start-recording and exp-lookit-stop-recording frames. In between, recording will continue, and all events captured will include the approximate time relative to the start of that video in a *sessionStreamTime*.

1.13.6 Example Lookit study outline

A typical Lookit study might contain the following frame types:

1. exp-video-config - This is a standard frame type that almost everyone should just stick at the very start of their study. It requires no customization; we'll maintain troubleshooting directions everyone can share.
2. exp-lookit-video-consent - A video consent frame. Your study needs to use this frame and it should come before starting the study or doing any other video recording. You need to specify some text fields to use this, regarding study-specific procedures, compensation, etc. These will be inserted into the consent document. If you need to show your IRB exactly what your consent document will look like, enter your text snippets, preview your study, and copy the document (or use the download button to get a PDF).
3. exp-lookit-text or exp-lookit-instruction-video. Now we're into optional frames that will vary by study. Most existing studies have started off with either video instructions or a text 'overview' of the study. The shorter this can be, the better - it's the equivalent of "okay, we're ready to get started, we're going to do X, Y, Z!" in the lab. Writing this text, and any instructions, tends to be more time-consuming than researchers expect: in contrast to an in-lab study, you can't easily tune what you say to the individual parent and answer just the questions they bring up. And you don't want to overwhelm them with a wall of text while they try to hold a squirmy baby! **We**

strongly recommend treating this as a serious writing/design exercise, and going through a few rounds of 'play-testing' with colleagues/family to make sure everything is as clear and concise as possible.

4. exp-lookit-stimuli-preview If you are showing children images/videos and you are going to ask the parents **not** to look at those stimuli, we strongly advise that you provide parents an opportunity to preview all of the stimuli that might be shown so they can decide if they're okay with that. This is both a reasonable courtesy (who knows what unusual phobia a child has, or what image you think is totally innocuous but turns out to offend a particular family for an unanticipated reason) and practical for data quality (parents will be less inclined to peek if they know roughly what's going on).

5. exp-lookit-survey Perhaps you want to collect some information (here or later on) from the parent that isn't included in the child or demographic data you'll have automatic access to - how much of which languages they speak in the home, motor milestones, whether their child likes Kermit or Oscar better, etc. You can use a survey frame to do that!

6. exp-lookit-instructions You may want a frame like this to give some final instructions to the parent before your 'test' procedures start! You can show text, videos, audio, show the user's webcam, etc. Make sure you have indicated here or earlier that the family is free to leave at any point and how they can do that. (Ctrl-X, F1, or closing the tab/window but then staying on the page will all bring up a "really exit?" dialog - you don't need to note all methods.)

7. exp-video-config-quality Once you're almost ready to start your actual 'test' procedures, you may want to guide the parent through webcam setup optimization, especially if you need the parent and child in a particular position. We provide some default instructions intended for preferential looking but would recommend making your own images/instructions if you can! You can also use the *exp-lookit-webcam-display* frame for lighter-weight display of the family's webcam so they can check positioning.

8. [Study-specific frames, e.g. exp-lookit-video, exp-lookit-images-audio; generally, a sequence of these frames would be put together with a randomizer. Make sure that if you have the parent turn around during the study, you let them know when to turn back around at the end! Also consider adding a friendly wrap-up "trial" at the end to give parents a chance to see the stimuli with a voiceover walkthrough, actually talk with their child about the story, etc.]

9. exp-lookit-exit-survey This is a required frame and should be the last thing in your study. This is where participants will select a privacy level for their video and indicate whether data can be shared on Databrary. (If you don't have IRB/institutional approval to share on Databrary yet, it's still fine to ask this; worst case you don't share data you had permission to share. Best case it'll smooth the process of asking your IRB retroactively if you want to!) Your participants will also have the option to withdraw video beyond the consent video entirely - this is rare (<1 percent of responses). These video settings are provided at the end, rather than the start, of the study so that parents already know roughly what happened and can better judge how comfortable they are with the video being shared. (E.g., "did my child pick his nose the whole time?")

   The 'debriefing' field of this frame is **very important**! This is a chance to explain the purpose of your study and how the family helped; at this point it's more obvious to the participant that skimming the info is fine if they're not super-interested, so you can elaborate in ways you might have avoided ahead of time in the interest of keeping instructions short. You may want to mention the various conditions kids were assigned to if you didn't before, and try to head off any concerns parents might have about how their child 'did' on the study, especially if there are 'correct' answers that will have been obvious to a parent. It's great if you can link people to a layperson-accessible article on a related topic - e.g., media coverage of one of your previous studies in this research program, a talk on Youtube, a parenting resource.

   If you are compensating participants, restate what the compensation is (and any conditions), and let them know when to expect their payment! E.g.: "To thank you for your participation, we'll be emailing you a $4 Amazon gift card - this should arrive in your inbox within the next week after we confirm your consent video and check that your child is in the age range for this study. (If you don't hear from us by then, feel free to reach out!) If you participate again with another child in the age range, you'll receive one gift card per child."

1.14 Updating the experiment runner

In the future, there may be changes in the Lookit experiment runner that you want your study to use - for instance, a bug fix for an issue your participants are encountering or a new frame you want to use. (By default, your study keeps chugging along using exactly the same code, so that updates can't change how your study works without your knowledge.)

1.14.1 Checking what's changed

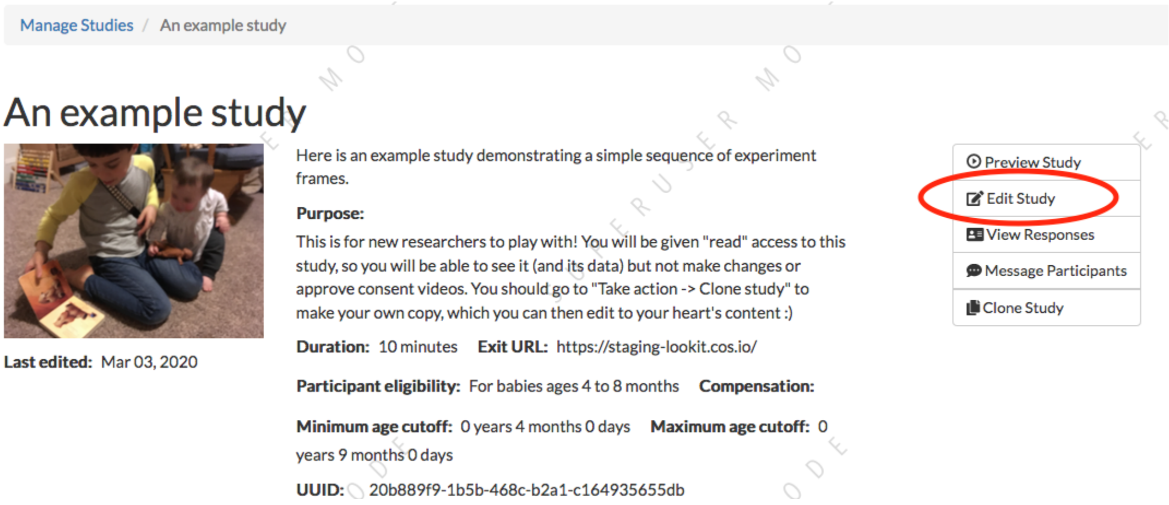
The most straightforward way to view changes to the Lookit code is to review the list of releases. If you're planning to update to the latest version, you should read through the release notes for each version between the one you're using and the new one.

The releases are numbered v<MAJOR>.<MINOR>.<PATCH> – e.g. v1.5.2. We adhere to semantic versioning, so the MAJOR version changes when there are backwards-incompatible changes - e.g., you need to change the name of a frame you're using for it to keep working. The MINOR version changes when features are added but are backwards-compatible. The PATCH version changes when there are backwards-compatible bug fixes.

1.14.2 Update steps

Here's how to update the code used:

1. Click "Edit study" on the study you want to update.



An example study

Here is an example study demonstrating a simple sequence of experiment frames.

Purpose:

This is for new researchers to play with! You will be given "read" access to this study, so you will be able to see it (and its data) but not make changes or approve consent videos. You should go to "Take action -> Clone study" to make your own copy, which you can then edit to your heart's content :)

Duration: 10 minutes **Exit URL:** https://staging-lookit.cos.io/

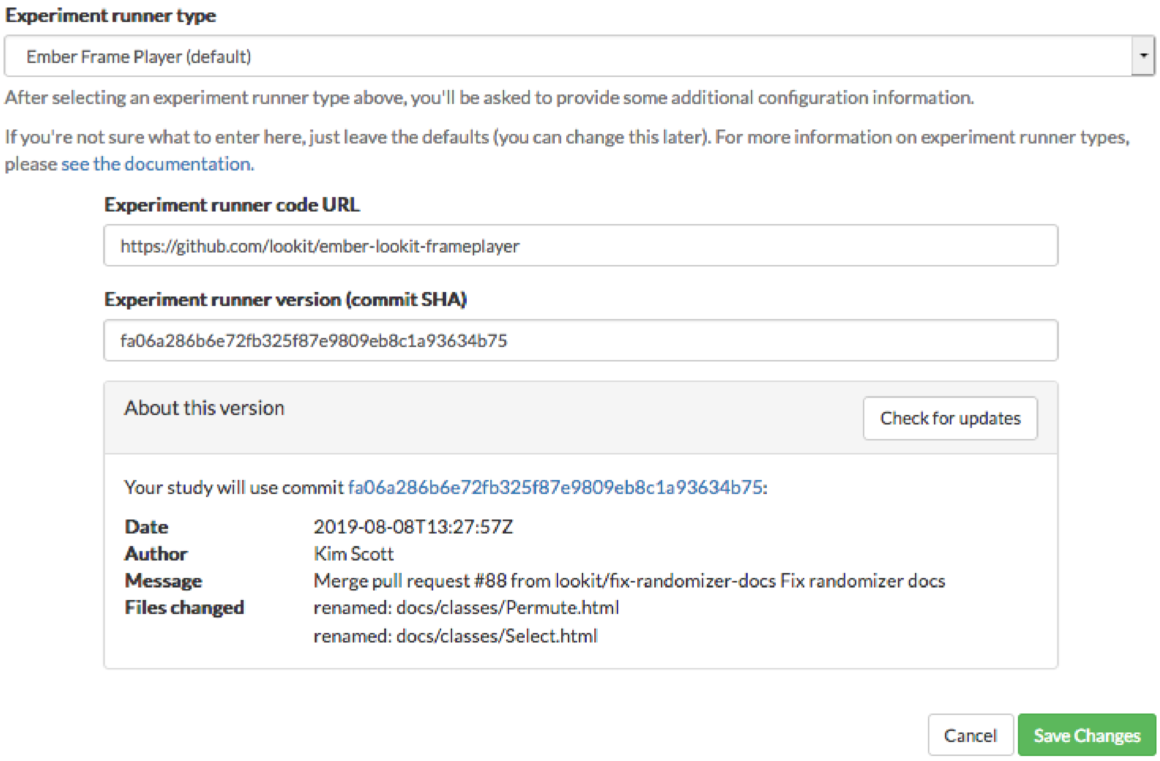
Participant eligibility: For babies ages 4 to 8 months **Compensation:**

Minimum age cutoff: 0 years 4 months 0 days **Maximum age cutoff:** 0 years 9 months 0 days

UUID: 20b889f9-1b5b-468c-b2a1-c164935655db

Last edited: Mar 03, 2020

2. Scroll down to the "Experiment runner type" section. You should see a value in the *Experiment runner version (commit SHA)*' field if you have built a preview or experiment runner before, like this:



Experiment runner type

Ember Frame Player (default)

After selecting an experiment runner type above, you'll be asked to provide some additional configuration information.

If you're not sure what to enter here, just leave the defaults (you can change this later). For more information on experiment runner types, please see the documentation.

Experiment runner code URL

https://github.com/lookit/ember-lookit-frameplayer

Experiment runner version (commit SHA)

fa06a286b6e72fb325f87e9809eb8c1a93634b75

About this version

Your study will use commit fa06a286b6e72fb325f87e9809eb8c1a93634b75:

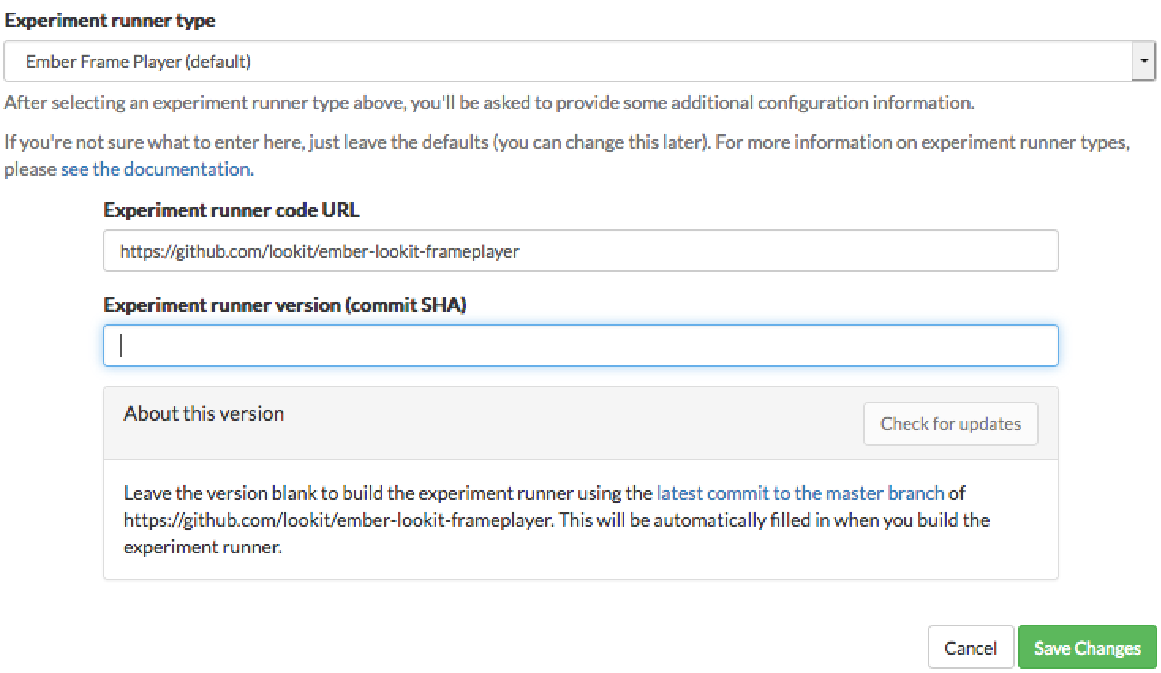
Date	2019-08-08T13:27:57Z
Author	Kim Scott
Message	Merge pull request #88 from lookit/fix-randomizer-docs Fix randomizer docs
Files changed	renamed: docs/classes/Permute.html renamed: docs/classes/Select.html

Make a note of this value, just in case the update makes something work differently in your study and you want to return to the current version.

You should see a bit of information about this version - the date, the message associated with the most recent change, and a list of files that were updated.

3. We'll focus here on just updating the version of the central Lookit code you're using. You can also edit the `Experiment runner code URL` to use a different repository entirely, like your own fork. The steps are the same regardless of which repo you're using, but pointing to your own code is more advanced.

   Option 1: latest version: You can delete the value in `Experiment runner version (commit SHA)` and leave it blank to use the default value, which is the most recent version of the Lookit frameplayer code.



Experiment runner type

Ember Frame Player (default)

After selecting an experiment runner type above, you'll be asked to provide some additional configuration information.

If you're not sure what to enter here, just leave the defaults (you can change this later). For more information on experiment runner types, please see the documentation.

Experiment runner code URL

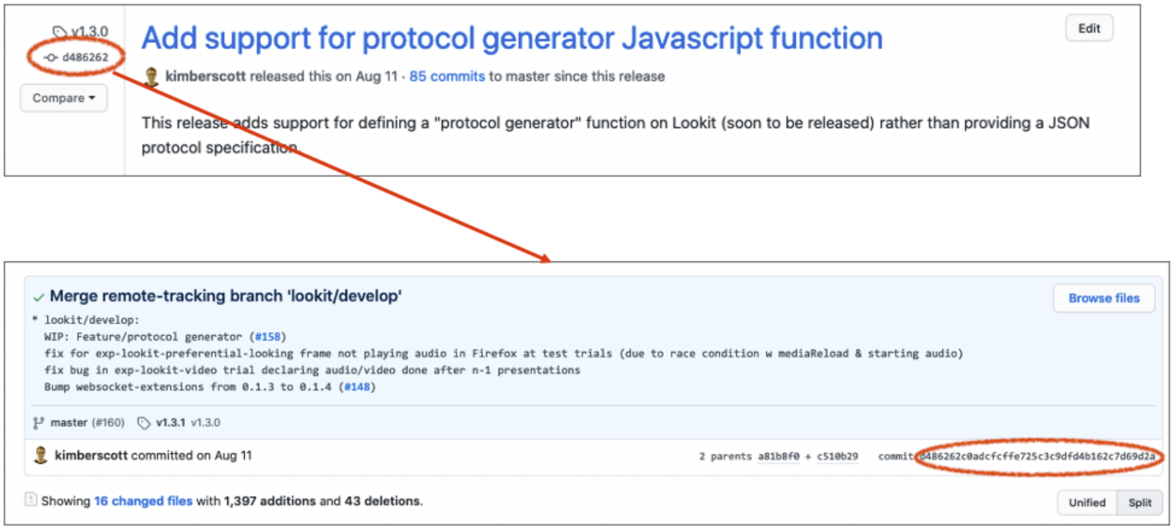
https://github.com/lookit/ember-lookit-frameplayer

Experiment runner version (commit SHA)

About this version

Leave the version blank to build the experiment runner using the latest commit to the master branch of https://github.com/lookit/ember-lookit-frameplayer. This will be automatically filled in when you build the experiment runner.

Option 2: specific version: You can click "Check for updates" and then paste in the ID of the code version you want to use ("commit sha"). You can also copy a version from the release list. If you do this, click the button circled below to go to a page that shows the commit sha:



v1.3.0

d486262

Add support for protocol generator Javascript function

kimberscott released this on Aug 11 · 85 commits to master since this release

This release adds support for defining a "protocol generator" function on Lookit (soon to be released) rather than providing a JSON protocol specification.

Merge remote-tracking branch 'lookit/develop'

```
* lookit/develop:
  WIP: Feature/protocol generator (#158)
  fix for exp-lookit-preferential-looking frame not playing audio in Firefox at test trials (due to race condition w mediaReload & starting audio)
  fix bug in exp-lookit-video trial declaring audio/video done after n-1 presentations
  Bump websocket-extensions from 0.1.3 to 0.1.4 (#148)
```

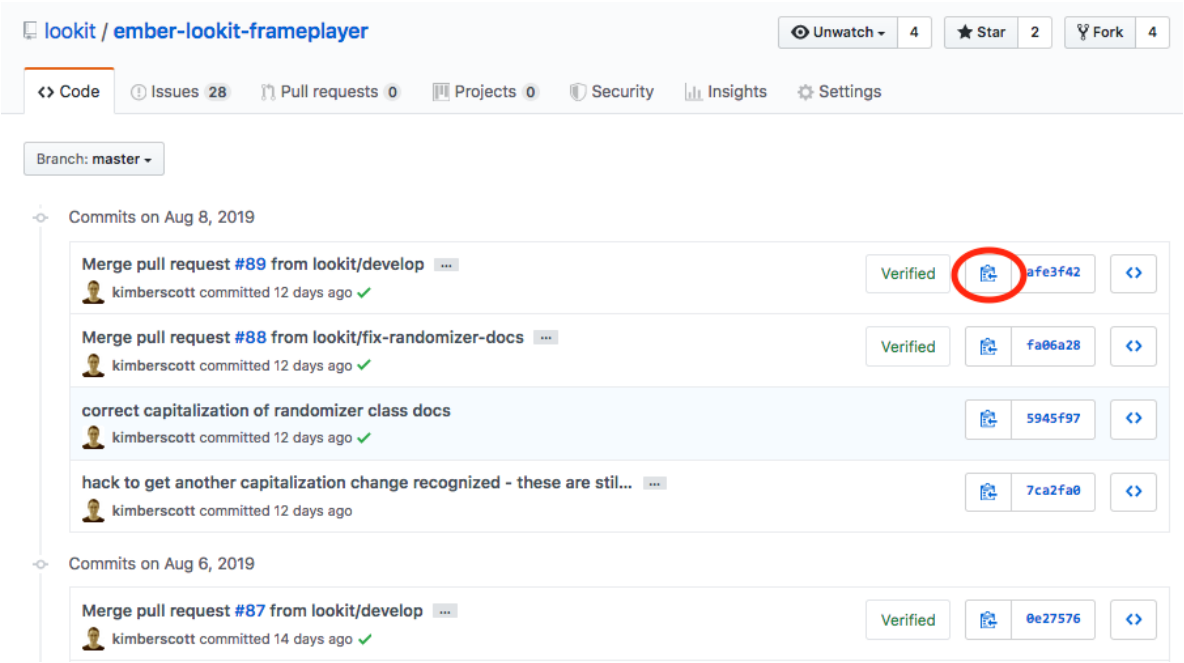
master (#160) v1.3.1 v1.3.0

kimberscott committed on Aug 11

2 parents a81b8f0 + c510b29 commit d486262c0adcfcffe725c3c9dfd4b162c7d69d2a

Showing 16 changed files with 1,397 additions and 43 deletions.

You can browse all commits by going to https://github.com/lookit/ember-lookit-frameplayer/commits/master. If you do this, click the clipboard button (circled) to copy the commit sha.



Commits on Aug 8, 2019

Merge pull request #89 from lookit/develop …
kimberscott committed 12 days ago ✓ — Verified — afe3f42

Merge pull request #88 from lookit/fix-randomizer-docs …
kimberscott committed 12 days ago ✓ — Verified — fa06a28

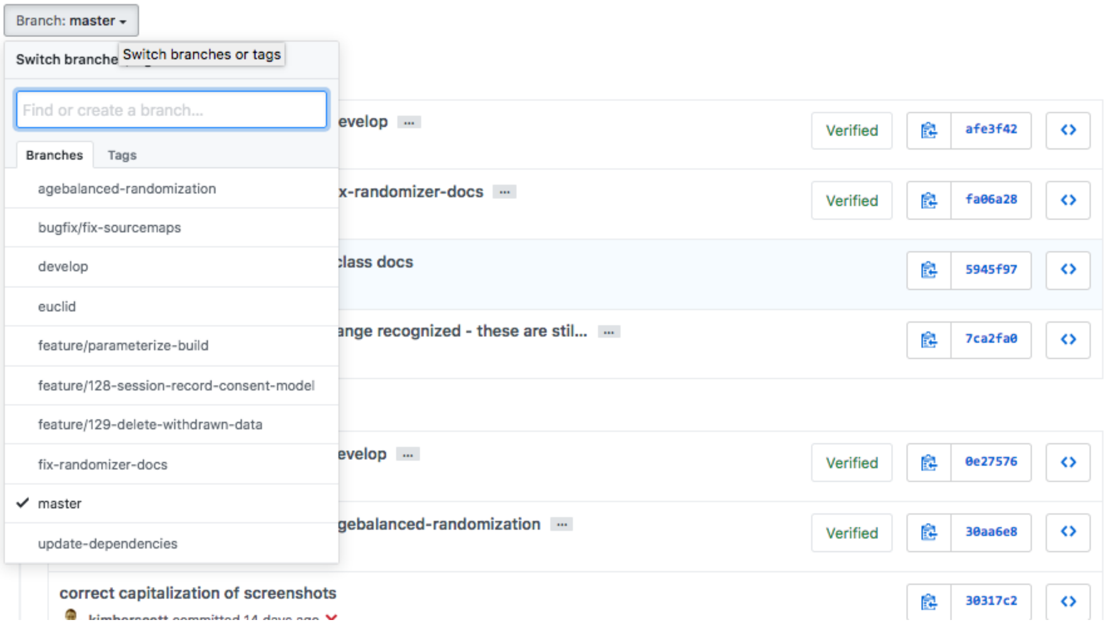
correct capitalization of randomizer class docs
kimberscott committed 12 days ago ✓ — 5945f97

hack to get another capitalization change recognized - these are stil… …
kimberscott committed 12 days ago — 7ca2fa0

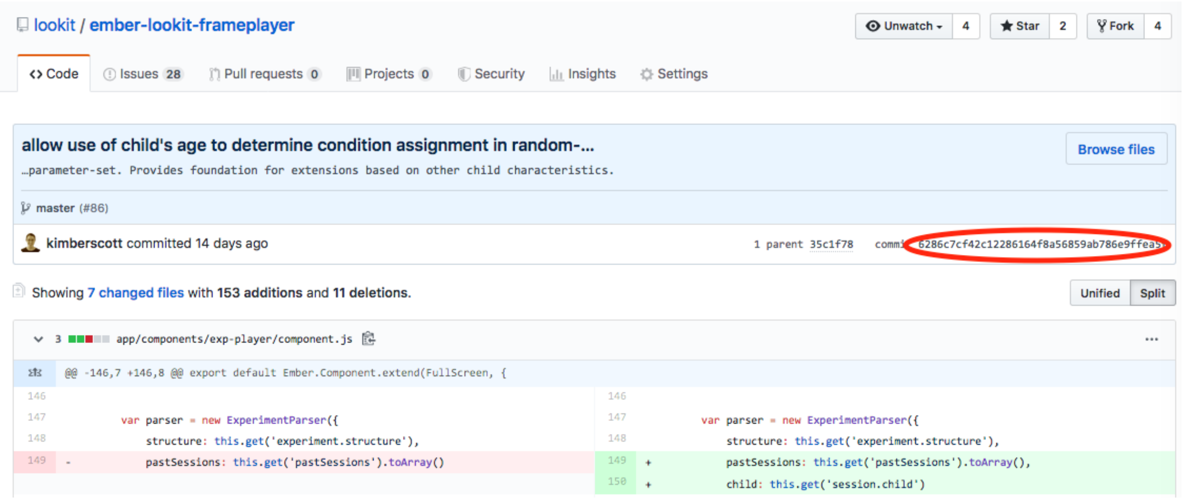
Commits on Aug 6, 2019

Merge pull request #87 from lookit/develop …
kimberscott committed 14 days ago ✓ — Verified — 0e27576

By default you will see only commits to the "master" branch of the frameplayer. If you want to use the "develop" branch, where new changes are tested out initially, you can select it instead:



You can click on any commit for more detail about what was changed. From there you can also see the commit sha (circled).



allow use of child's age to determine condition assignment in random-...

…parameter-set. Provides foundation for extensions based on other child characteristics.

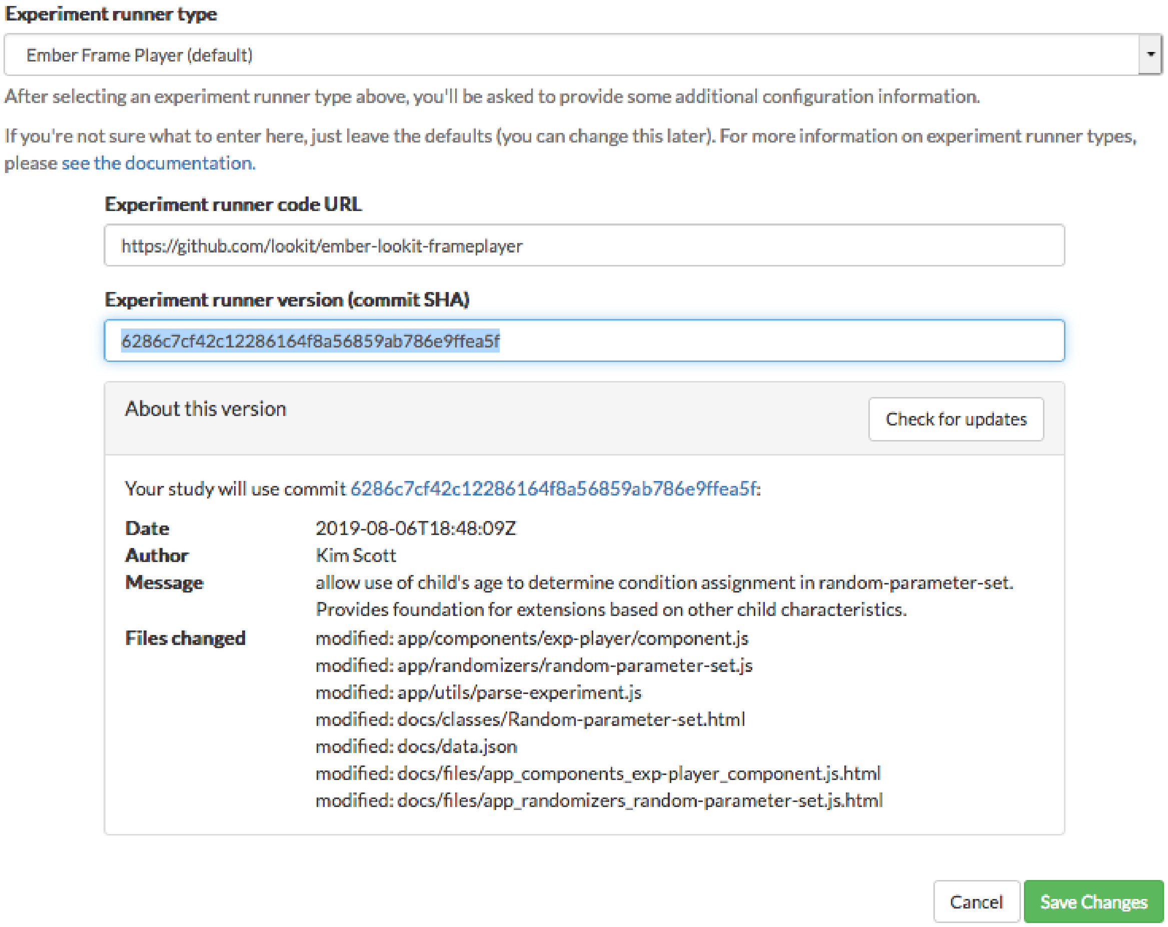
master (#86)

kimberscott committed 14 days ago   1 parent 35c1f78   commit 6286c7cf42c12286164f8a56859ab786e9ffea5

Showing 7 changed files with 153 additions and 11 deletions.

```
3 app/components/exp-player/component.js

@@ -146,7 +146,8 @@ export default Ember.Component.extend(FullScreen, {

        var parser = new ExperimentParser({          |          var parser = new ExperimentParser({
            structure: this.get('experiment.structure'),  |      structure: this.get('experiment.structure'),
-           pastSessions: this.get('pastSessions').toArray()  |  +   pastSessions: this.get('pastSessions').toArray(),
                                                       |  +   child: this.get('session.child')
```

Paste the commit sha you want to use into the study edit view, and you should see some information confirming it's the one you wanted:



Experiment runner type

Ember Frame Player (default)

After selecting an experiment runner type above, you'll be asked to provide some additional configuration information.

If you're not sure what to enter here, just leave the defaults (you can change this later). For more information on experiment runner types, please see the documentation.

Experiment runner code URL

https://github.com/lookit/ember-lookit-frameplayer

Experiment runner version (commit SHA)

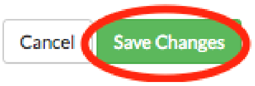
6286c7cf42c12286164f8a56859ab786e9ffea5f

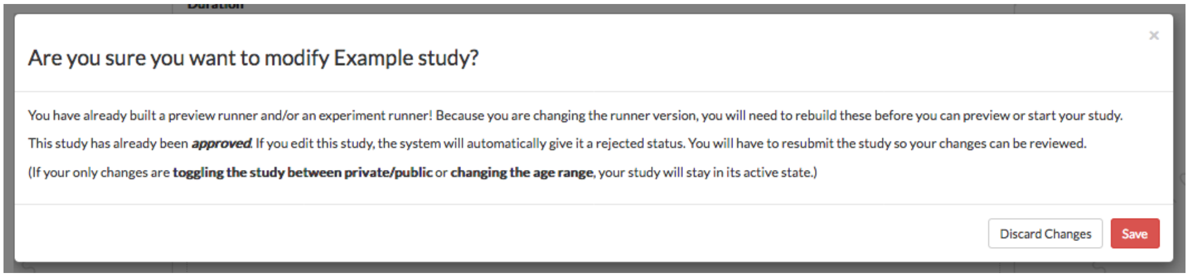
About this version

Your study will use commit 6286c7cf42c12286164f8a56859ab786e9ffea5f:

Date	2019-08-06T18:48:09Z
Author	Kim Scott
Message	allow use of child's age to determine condition assignment in random-parameter-set. Provides foundation for extensions based on other child characteristics.
Files changed	modified: app/components/exp-player/component.js modified: app/randomizers/random-parameter-set.js modified: app/utils/parse-experiment.js modified: docs/classes/Random-parameter-set.html modified: docs/data.json modified: docs/files/app_components_exp-player_component.js.html modified: docs/files/app_randomizers_random-parameter-set.js.html

4. Click "Save Changes". You will see a warning pop up if your study has already been approved, telling you it will be rejected automatically and require re-review. This is so that Lookit staff can review any new code you're using (in particular if you're using your own repo).



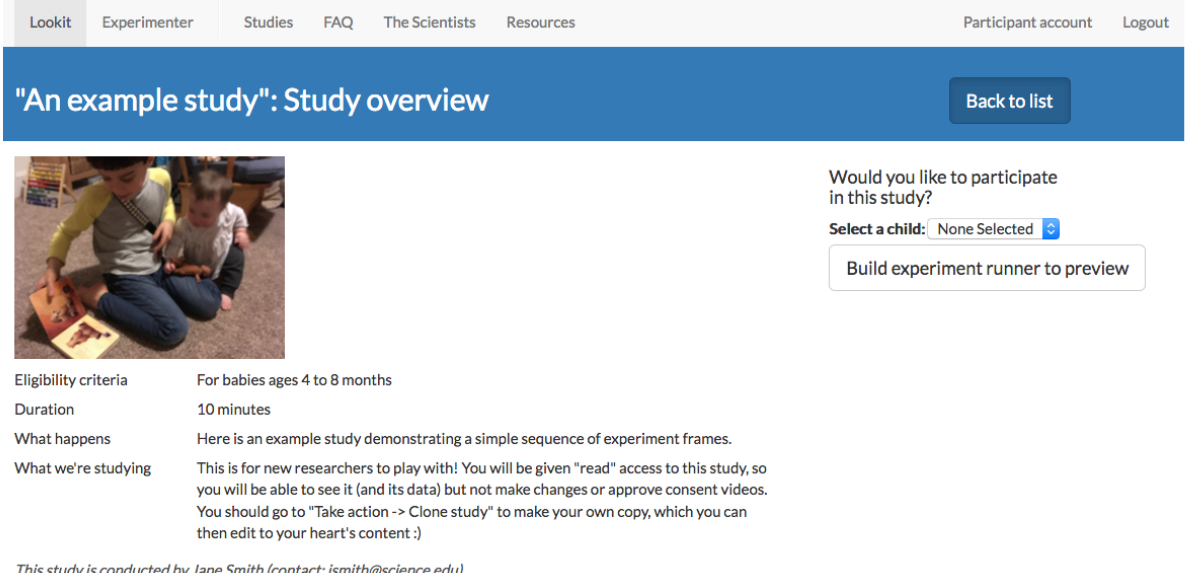

Are you sure you want to modify Example study?

You have already built a preview runner and/or an experiment runner! Because you are changing the runner version, you will need to rebuild these before you can preview or start your study.

This study has already been ***approved***. If you edit this study, the system will automatically give it a rejected status. You will have to resubmit the study so your changes can be reviewed.

(If your only changes are **toggling the study between private/public** or **changing the age range**, your study will stay in its active state.)

5. Because the code you're using is different now, you will need to build a (new) experiment runner before you can start your study again. If you click the "Preview study" button on your main study page or edit study page, you will be taken to a preview of the "study detail" page participants see before deciding whether to start the study. But you won't have the option to actually preview the study yet:



"An example study": Study overview

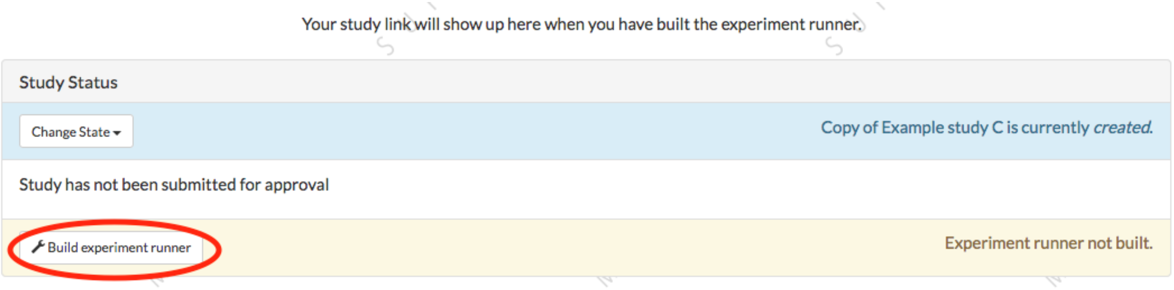
Would you like to participate in this study?

Select a child: None Selected

Eligibility criteria	For babies ages 4 to 8 months
Duration	10 minutes
What happens	Here is an example study demonstrating a simple sequence of experiment frames.
What we're studying	This is for new researchers to play with! You will be given "read" access to this study, so you will be able to see it (and its data) but not make changes or approve consent videos. You should go to "Take action -> Clone study" to make your own copy, which you can then edit to your heart's content :)

This study is conducted by Jane Smith (contact: jsmith@science.edu)

Even if you are just updating to the latest version of the master branch, you should preview your study and make sure everything still works just how you want it to! Click "Build experiment runner" on your main study page:


Your study link will show up here when you have built the experiment runner.

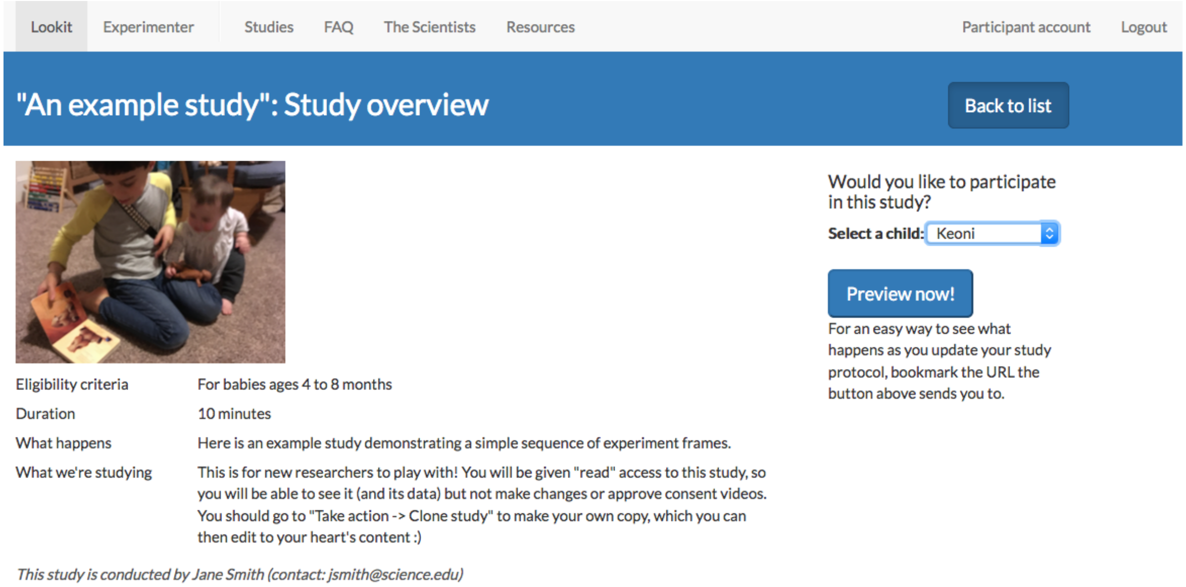
Study Status

Copy of Example study C is currently *created.*

Study has not been submitted for approval

Experiment runner not built.

Once you get an email notification that the experiment runner has been built, you will be able to try out your study. The preview detail page will now have a button to preview the study:



"An example study": Study overview

Would you like to participate in this study?

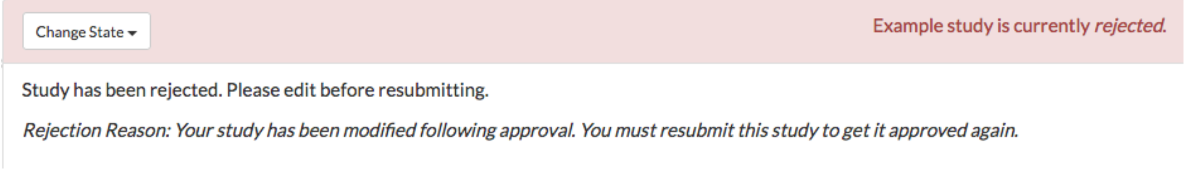
Select a child: Keoni

For an easy way to see what happens as you update your study protocol, bookmark the URL the button above sends you to.

Eligibility criteria	For babies ages 4 to 8 months
Duration	10 minutes
What happens	Here is an example study demonstrating a simple sequence of experiment frames.
What we're studying	This is for new researchers to play with! You will be given "read" access to this study, so you will be able to see it (and its data) but not make changes or approve consent videos. You should go to "Take action -> Clone study" to make your own copy, which you can then edit to your heart's content :)

This study is conducted by Jane Smith (contact: jsmith@science.edu)

6. If you were already collecting data, return to your main study page where you will probably see that its state is "rejected." Click "change state" and select "submit" to submit your study for re-approval. Once it is approved, you will be able to start data collection again. You will need to click "Build experiment runner" and wait for that process to complete (about ten minutes) before you can start the study.



Example study is currently *rejected.*

Study has been rejected. Please edit before resubmitting.

Rejection Reason: Your study has been modified following approval. You must resubmit this study to get it approved again.

1.15 Preparing stimuli

1.15.1 Creating audio and video files

Most experiments will involve using audio and/or video files! You may find that there are more 'stimuli' to create than you'd have in the lab, because in the lab you have the luxury of being present to explain what's going on. When you design an online study, you may need to record many of the things you'd say in the lab, or create pictures or demos to show how things work. This includes not just storybook audio, but little clips like "We're almost done!" or "Okay, go ahead and turn around now."

Creating video instructions

Several labs have been having success filming video instructions for families, which can be easier to process for many families (and friendlier) than text instructions. This doesn't need to look "professionally produced" to be great; the important pieces are pretty low-tech!

Here are some tips from the community:

- Smile and try to channel the same excitement and energy you'd have welcoming a family to the lab. There is no scientific need to speak in a dry, measured tone here. (Or… in most cases, really.) And remember, you think your research question is interesting enough to think about for months and months!
- Keep a simple, homogeneous background (e.g. a blank wall or curtain).

- Outline or script what you're going to say before filming the video. (You can even use a free web-browser-based teleprompted to show you your notes!)
- Add video- or image-in-video examples to make the video more engaging and make it easy to follow when you explain what's going to happen. (This can make a HUGE difference and in most cases avoid the need to give parents the option to preview the stimuli separately).
- It's normal not to be able to do 2 minutes straight of talking without making mistakes! Don't worry about trying to get a continuous "good shot," just edit the video to cut pauses or errors.
- Make sure to edit the audio track the same way you would for audio stimuli to reduce background noise.
- Provide closed captioning for accessibility (even many hearing parents will find it easier to process this way), and/or provide "alt text" for the video as shown in the exp-lookit-instruction-video frame.

Basic audio editing

For basic editing of audio files, if you don't already have a system in place, we highly recommend Audacity. You can create many "tracks" or select portions of a longer recording using labels, and export them all at once; you can easily adjust volume so it's similar across your stimuli; and the simple "noise reduction" filter works well. At a minimum, even if these are not 'stimuli' per se (e.g., verbal instructions), we recommend

1. Using **noise reduction** to make speech clearer and remove any background 'buzz' during pauses. First select a segment of silence to analyze, then apply noise reduction across the whole audio recording; you may need to play around with the defaults to get excellent noise reduction without distortion, but it does a pretty good job out of the box.
2. Using the **'amplify'** filter to make all stimuli and instructions approximately equally loud (by default, it makes a segment of audio as loud as possible without clipping).
3. **Trimming** ALL of the silence from the beginning and end of the audio clip. This silence may not be especially noticable when you simply play the file, but it translates into an unnecessary delay between whenever you trigger the audio file to play in your study and when the relevant sound actually starts.

For editing of video files, we recommend getting comfortable with the command-line tool ffmpeg. It's a bit of a pain to get used to, but then you'll be able to do almost anything you can imagine with audio and video files.

File formats

To have your media play properly in a web browser, you need to choose file format(s) that are supported by that browser. Note that AVI and MOV files generally cannot be played in a web browser without installing special plugins.

We used to recommend creating both mp4 (H.264 video codec + AAC audio codec) and webm versions of video, and mp3 and ogg versions of audio. Now mp4 and mp3 files are supported by almost all modern browsers (and we're only officially trying to support Firefox and Chrome), so you should be fine making just mp4 for video and mp3 for audio. You can look up browser support for specific formats at CanIUse; see MDN for a comprehensive overview of this topic.

If you don't already have your files in an appropriate format, the easiest way to convert them (especially if you have a lot to convert) is to use the command line tool ffmpeg.

Here's an example command to convert a video file INPUTPATH to mp4 with reasonable quality/filesize and using H.264 & AAC codecs:

```
ffmpeg -i INPUTPATH -c:v libx264 -preset slow -b:v 1500k -maxrate 1000k
-bufsize 2000k -c:a libfdk_aac -b:a 128k OUTPUTPATH
```

And to make a webm file:

```
ffmpeg -i INPUTPATH -c:v libvpx -b:v 1000k -maxrate 1500k -bufsize 2000k -c:a
libvorbis -b:a 128k -speed 2 OUTPUTPATH
```

Converting all your audio and video files can be easily automated in Python. Here's an example script that uses ffmpeg to convert all the m4a and wav files in a directory to mp3 and ogg files:

```
import os
import subprocess as sp
import sys

audioPath = '/Users/kms/Dropbox (MIT)/round 2/ingroupobligations/lookit stimuli/audio␣
↪clips/'

audioFiles = os.listdir(audioPath)

for audio in audioFiles:
    (shortname, ext) = os.path.splitext(audio)
    print shortname
    if not(os.path.isdir(os.path.join(audioPath, audio))) and ext in ['.m4a', '.wav']:
        sp.call(['ffmpeg', '-i', os.path.join(audioPath, audio), \
                 os.path.join(audioPath, 'mp3', shortname + '.mp3')])
        sp.call(['ffmpeg', '-i', os.path.join(audioPath, audio), \
                 os.path.join(audioPath, 'ogg', shortname + '.ogg')])
```

1.15.2 Putting your stimuli files online

HTTPS vs HTTP

Wherever you put your stimuli, you need to serve them using HTTPS, meaning your URLs should start with https:// - **not** http://. If you use HTTP (not secure) your stimuli may not display at all in modern browsers, and they introduce security risks. You can learn more here.

You are responsible for hosting your study stimuli online somewhere. You have a variety of options, including:

- Most universities offer some form of free static web hosting associated with your university account. This might be a nice option because (a) it's free and (b) it's actually kind of your IT department's job to help you with it. Here are some examples:
  - MIT
  - Pittsburgh
  - Michigan
  - Cornell

  The process for accessing your university storage, and for setting up a lab-wide account, will vary by institution. You can ask your IT department for instructions - what you want to ask about is "static web hosting" for your stimuli or "online file storage."

  You do **not** need to "set up a web server" (they will assume you want to do something more complicated and run backend code).

- GitHub repo: This is also free! And it offers the advantage that you can keep track of any changes to your stimuli over time in a very robust, transparent way. This may be especially handy when you go to publish your work - all your stimuli are already publicly available, with changes logged. If you choose to use GitHub to host images/video/audio, it needs to be set up as a public repo - if the files are in a private repo, Lookit won't have the access credentials needed to display them during the study. You may be most familiar with Github as a place to store and collaborate on code, but it can be used for any files. There are detailed directions available in the Lookit stimuli template repo for putting your own stimuli on GitHub - no experience required!

- Google Cloud Storage: This is free or very cheap and again fairly straightforward to use. We haven't used it personally, so if you do, please consider adding to these instructions!
- Amazon S3 storage: This is very cheap (likely a few cents per month) and fairly straightforward to use. You will need to create an Amazon Web Services account and create a "bucket" where your stimuli will live. You will also need to make that bucket's files public, which is not the default. You can follow steps 3 and 4 of this walkthrough to do so. Then you can use the web interface to create folders and upload your files. They will be accessible at URLs like `https://BUCKETNAME.s3.amazonaws.com/STUDYNAME/img/cats.jpg`.

What about Google Drive or Dropbox?

You may already be accustomed to sharing files using services like Google Drive or Dropbox, and be wondering why you can't just make your files public there. Technically, you can. However, you will run into a number of annoying practical issues: for instance, your file links will be incomprehensible random strings, which will make it difficult to interpret, debug, or change your Lookit study protocol, especially for anyone who wants to understand what you did in the future. You will not be able to use relative file paths in Lookit as described below, since your files' organization in folders isn't reflected in the URLs. Also, if you or your collaborators change a file, the URL may change in ways you didn't predict, breaking something in your study.

In short, we really don't recommend it, even though these tools are great for file sharing in other circumstances.

1.15.3 Directory structure

For convenience, many Lookit experiment frames use an expand-assets mixin that allows you to define a base directory (`baseDir`) as part of the frame definition, so that instead of providing full paths to your stimuli (including multiple file formats) you can give relative paths and specify the audio and/or video formats to expect (`audioTypes` and `videoTypes`). Please see the linked documentation for details on how to specify your base directory and how to structure your files in it!

1.15.4 Helpful resources

- [Slides] Stimuli preparation and hosting for Lookit (Nicole Cuneo)
- [Slides] FFMPEG starter powerpoint (Nicole Cuneo)
- [Code] Some example FFMPEG commands (Kim Scott)

1.15.5 Tips and tricks (advanced)

Setting up a CDN (optional)

If you are very concerned with optimizing speed of delivery of your stimuli for users worldwide, best practice is to use a Content Delivery Network (CDN). You can read a description of what this is and when it might be helpful here. This is unlikely to be necessary for most Lookit researchers, but if you do choose to set one up, it's cheap and reasonably straightforward. One option we have used successfully is Amazon CloudFront.

Making dummy stimuli

Sometimes you may not have your stimuli actually ready yet, but you want to make sure your experiment will work as intended once they're ready. Here's an example of using ffmpeg to make some "dummy" images of text to represent distinct exemplars of various categories. You could also create videos by setting the duration in seconds (here d=0.01) to something longer and using an mp4 or webm extension for output instead of jpg.

```
import os
import subprocess as sp
import sys

baseDir = '/Users/kms/Desktop/labelsconcepts/img/'

for catDir in ['nov1', 'nov2', 'nov3', 'cats', 'dogs', 'iguanas', 'manatees',
↪'squirrels']:
    os.mkdir(os.path.join(baseDir, catDir));
    for iIm in range(1, 12):
        text = catDir + '.' + str(iIm)
        output = os.path.join(baseDir, catDir, str(iIm) + '.jpg')
        sp.call(['ffmpeg', '-f', 'lavfi', '-i', 'color=c=gray:s=640x480:d=0.01', '-vf
↪',
            "drawtext=fontfile=drawtext='fontfile=/Library/Fonts/Arial Black.ttf
↪':text='" + text + "':fontsize=64:fontcolor=black:x=10:y=10",
            output])
```

1.16 Style guide

We try to conform to some common stylistic, writing, and study design guidelines to provide a familiar and friendly experience across Lookit studies.

1.16.1 General language usage

1. Use casual language and terms where possible

    - "Baby" rather than "infant" (they mean the same thing, but "baby" is far more common in everyday speech; "infant" is more common in clinical/academic settings)
    - "Newborn" rather than "neonate"
    - Referring to the parent/guardian can be tricky since they may or may not be a "Mom" or "Dad" - "grown-up" works well in most cases

2. Use singular "they" rather than "he/she" or "(s)he" or "the child." (E.g., say "Hold your child so they can see the screen" rather than "Hold your child so he/she can see the screen.") It's common enough now that it's more readable, plus it encompasses families who use pronouns other than he/she.

3. A girl is a female child. If you're talking about a female grown-up–e.g., a video is going to show your RA demonstrating a toy–that's a woman. (You can consider words like "person," "somebody", or "friend" if that sounds awkward!)

4. When asking questions about the child's family or caregivers, be explicit about who you mean. It may feel more "inclusive" simply to say "family," but for families dealing with multiple definitions it's confusing to try to figure out what you really care about - especially since research is weird and who knows how something that feels irrelevant might not be!

1.16.2 Music selections

When choosing music or nursery rhymes for stimuli or background music, please confirm that it's something that you would be comfortable sharing in a preschool classroom! In particular, a lot of US "children's songs" turn out to have disturbingly racist histories. Even if the current lyrics are inoffensive, please don't use music that is known for past use in minstrel shows, for instance, or where there are alternate historical lyrics that are offensive. Examples of songs **not** to use include "Oh Susanna," "Camptown Races," and "I've Been Working on the Railroad." Here's a good introduction to this topic if you're interested in learning more!

1.16.3 Eligibility criteria

In general, keep eligibility criteria as broad as possible for a positive family experience. The "hard" eligibility criteria you list on Lookit will include your age range and any characteristics that are critical to the study design or that are necessary for the study to make any sense to the child/family. For instance, if you have a question specifically about bilingual vs. trilingual kids, go ahead and require n_languages >= 2. But if you just routinely exclude premature babies or kids with trisomies from analysis, you can do that after the fact if needed since the study experience won't be affected.

Your eligibility criteria description should "translate" the minimum / maximum ages and any eligibility criteria expression into regular language, starting with "For…" - for example, "for 3-year-olds" or "for 8- through 12-month-olds who are exposed to least two languages at home." You can also include hard criteria that are not possible to check for programmatically, for instance if your IRB requires that participants live in a certain country.

Specific guidance:

1. If recorded audio in a particular language is a critical part of your study, note that the study is for speakers of that language. (You don't need to separately exclude deaf children!)
2. If you need kids to have typical hearing to participate, and that's not otherwise obvious from the design or a language requirement, list that as "with typical hearing" or "without hearing loss" rather than "normal hearing." Only do this if hearing is critical to the design and/or to family experience.
3. Don't require babies to be born at full term in your hard criteria unless the age range extends under 6 months, your study is specifically about effects of prematurity, or the study requires unusual time and effort from the family. Otherwise MANY studies exclude preemies and it's frustrating for parents. Plus this pushes all of us towards putting some upper bounds on how much of a difference prematurity actually makes. Some enterprising student should ask everyone for data excluded due to prematurity eventually…

1.16.4 Study purpose

The *study purpose* is the most challenging piece of the study to write for most researchers (yes, including all that JSON!). The big questions to consider are:

1. Is this an ACCURATE description of the question the study will answer? I.e. when we get the data, is it at least possible that we will be able to answer it? Often we see descriptions of a broad research program, instead of the particular question THIS study will address.

   Examples of questions your study does not answer because they are way too broad: "How do children learn language?," "How do babies think about morality?," "How does numerical cognition develop?"

2. Does it explain why this question matters or why this question is interesting? This doesn't need to be a "practical" reason like informing parenting or education practices - basic research matters! Here are a few common framings that do NOT explain why a question matters:

   - Not much is known about X.

- X is an important skill for doing Y and Z, so we're interested in whether babies can do X / how babies learn to do X.

However, you don't have to justify the entire field of cognitive science/psychology. E.g., if your question would distinguish between there being two separate systems for representing number that kids have to link, and there only being one, you're set. Distinguishing between competing theories and determining whether things are innate are generally ok places to stop.

Examples coming soon of good "purpose" sections and before/after revisions!

1.17 Coding consent

1.17.1 Overview

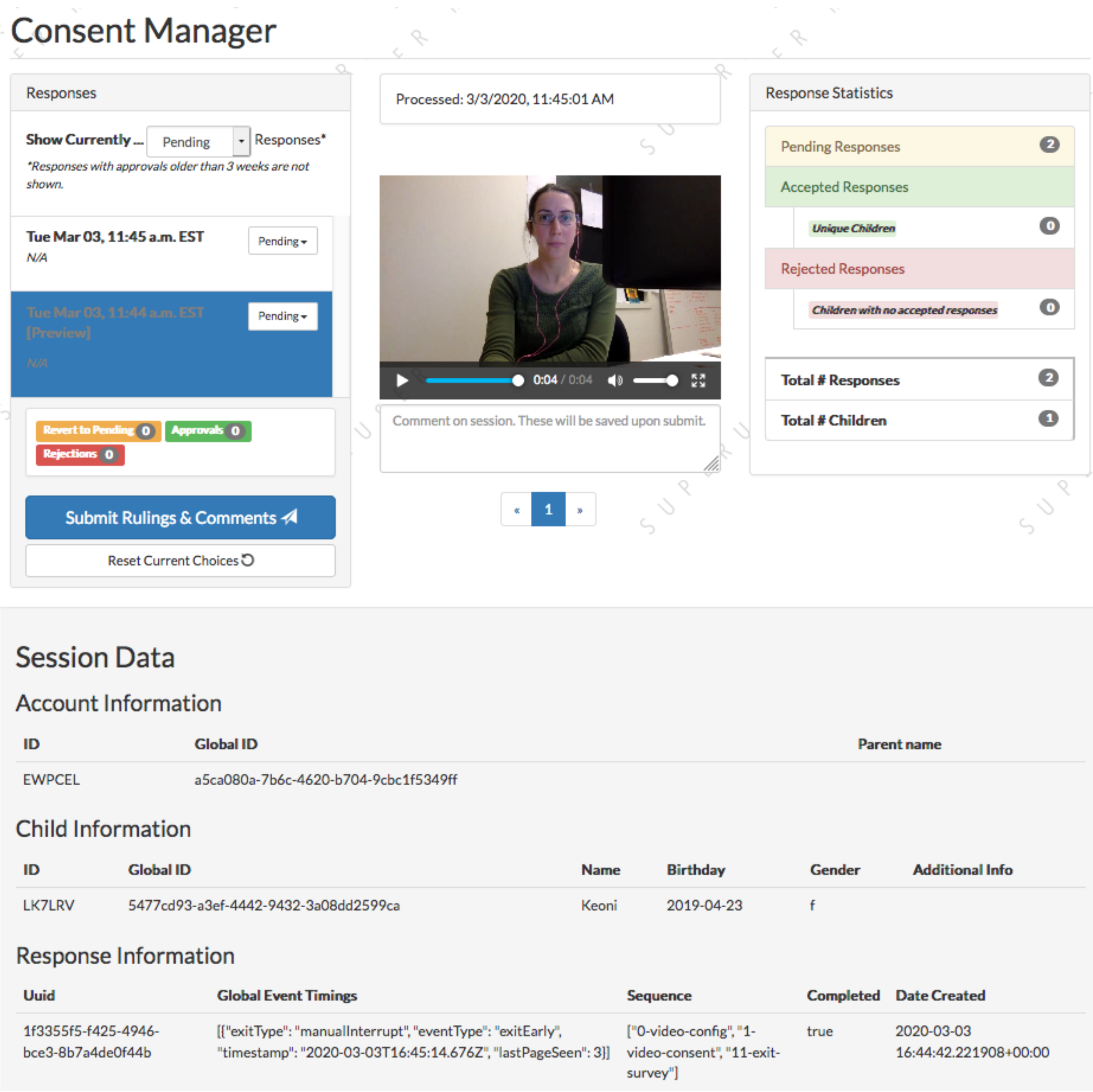
At the start of a Lookit study, the parent is asked to provide a verbal statement of informed consent. Unlike in the lab (or at least to a greater extent), it is technically possible for you to end up collecting data from a parent who did NOT consent to participate - e.g., someone idly clicking through who may not understand that this is a research study to do with a child.

For this reason it is critical that you confirm informed consent before using any data from a response! This is baked into the Lookit experimenter interface: you actually do not receive access to responses, or to the associated child, account, or demographic data, until you confirm consent using the consent manager.

Responses submitted on Lookit start out with a consent status of 'Pending.' Then a researcher working on this study can either 'approve' or 'reject' the consent video.

1.17.2 Managing consent rulings

From your study detail page, click 'Review Consent' and you will be taken to the 'Consent manager' view.


Consent Manager
Responses
Show Currently ... Pending Responses*
*Responses with approvals older than 3 weeks are not shown.
Tue Mar 03, 11:45 a.m. EST
N/A
Pending
Tue Mar 03, 11:44 a.m. EST [Preview]
N/A
Pending
Revert to Pending 0
Approvals 0
Rejections 0
Submit Rulings & Comments
Reset Current Choices
Processed: 3/3/2020, 11:45:01 AM
0:04 / 0:04
Comment on session. These will be saved upon submit.
« 1 »
Response Statistics
Pending Responses 2
Accepted Responses
Unique Children 0
Rejected Responses
Children with no accepted responses 0
Total # Responses 2
Total # Children 1

Session Data

Account Information

ID	Global ID	Parent name
EWPCEL	a5ca080a-7b6c-4620-b704-9cbc1f5349ff	

Child Information

ID	Global ID	Name	Birthday	Gender	Additional Info
LK7LRV	5477cd93-a3ef-4442-9432-3a08dd2599ca	Keoni	2019-04-23	f	

Response Information

Uuid	Global Event Timings	Sequence	Completed	Date Created
1f3355f5-f425-4946-bce3-8b7a4de0f44b	[{"exitType": "manualInterrupt", "eventType": "exitEarly", "timestamp": "2020-03-03T16:45:14.676Z", "lastPageSeen": 3}]	["0-video-config", "1-video-consent", "11-exit-survey"]	true	2020-03-03 16:44:42.221908+00:00

At the left, you will see a list of responses. By default the responses with 'Pending' consent status are displayed; you can use the dropdown menu to show 'Accepted' or 'Rejected' consent videos instead.

Data from previewing your study is displayed here too, but any preview responses are in gray and say "[PREVIEW]".

Making consent rulings

When you click on a response, any consent videos from that response are shown to the right. (It is possible, although rare, for there to be multiple consent videos associated with a single response; this will become more common when some researchers are collecting both parental consent and child assent, which would be judged together.) A minimal summary of the data is shown below so that you can see whether the child is in the age range for the study and how far the family got.

Watch the video, and decide whether it shows informed consent. You can choose to 'Accept' or 'Reject' a response, and can enter a comment if desired to keep track of any additional information. You can enter a comment without changing the consent ruling (e.g., to say “Emailed this family to confirm consent”). In general, you should ‘accept’ consent only when the consent video shows an adult reading the consent statement audibly (or signing in ASL), but see the Terms of Use for details (for instance, you may be able to contact a family to confirm consent by email in some cases).

Repeat for each consent video. When you are done for now, click ‘Submit Rulings and Comments’ to save your judgments. These changes will immediately be reflected in the number of responses available in the ‘individual responses’ and ‘all responses’ views, as well as with respect to demographic and participant data you have access to.

Consent rulings can be changed after an initial ruling is made; for instance, you can use the dropdown menu to display ‘Accepted’ responses and either ‘Reject’ or ‘Revert to pending.’

The most recent consent ruling, the time of that ruling, any comment, and the name of the researcher who made the ruling, will be included in the JSON/CSV data for this response.

Response statistics

A summary of responses is shown to the right of the consent manager, providing some very basic information about the non-consented responses that may be useful for publication of results. You can see how many responses are still pending consent confirmation; how many accepted responses there are (from how many unique children); and how many responses were rejected (from how many unique children who did not also have some response accepted).

Withdrawn responses

If a parent chooses to withdraw video data at the end of the study, that will be noted in the list item for the response (before the comment it will say ‘Withdrawn’ and the response will be crossed out). All video data beyond consent will be inaccessible to researchers, and it will be deleted automatically from Lookit servers after seven days.

However, you are still able to make a consent ruling about the consent video; this will still impact access to the remaining non-video response data as well as associated child, demographic, and account data.

Where are my preview responses?

When you preview a study, data is saved to the server the same way as when you participate. However, this data is only available for you to see if you complete at least a consent frame.

When you’re working on a study, you may often be trying out pieces of the study without going through the consent process every time. Once you want to take a look at the data collected, just make sure you include a consent frame.

1.18 Downloading data

1.18.1 What data can I access?

You can access:

- response data from responses for which you have confirmed consent in the Consent Manager
- account, demographic, and child data associated with those responses. (You will see these accounts under ‘Manage Participants’ as well; if some siblings but not others have participated in one of your studies and you have confirmed consent, you will only see the siblings who have participated.)
- video clips associated with those responses (consent videos plus any video collected during the session), unless the participant withdrew video during the exit survey.

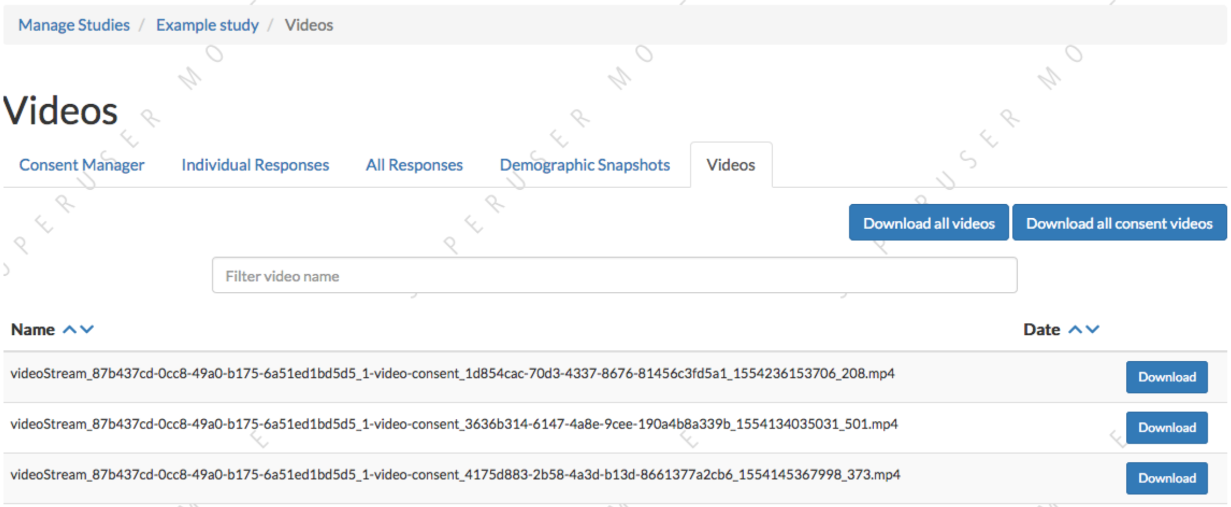
How does it work when participants withdraw video?

If the participant selected the 'withdraw video' option in an exit-survey frame at the end of the study, all video except for the consent video is unavailable (and will be deleted from Lookit servers as well in 7 days). You will still be able to see the consent video in the consent manager. The fact that video has been withdrawn is included in the response data.

There is a potential rare edge case where you access video while the participant is still doing the study, and then they withdraw, so you should still verify that none of your participants have withdrawn video.

1.18.2 Viewing all study videos

To view all video responses to your study from sessions with confirmed consent, click 'View Responses' from the study detail page and then click 'Videos'. You can filter on video name.



Name	Date	
videoStream_87b437cd-0cc8-49a0-b175-6a51ed1bd5d5_1-video-consent_1d854cac-70d3-4337-8676-81456c3fd5a1_1554236153706_208.mp4		Download
videoStream_87b437cd-0cc8-49a0-b175-6a51ed1bd5d5_1-video-consent_3636b314-6147-4a8e-9cee-190a4b8a339b_1554134035031_501.mp4		Download
videoStream_87b437cd-0cc8-49a0-b175-6a51ed1bd5d5_1-video-consent_4175d883-2b58-4a3d-b13d-8661377a2cb6_1554145367998_373.mp4		Download

The format of the video names is *videoStream_{study_uuid}_{order-frame_name}_{response_uuid}_{timestamp}_{randomDigits}.mp4*

You can match up the videos with their corresponding sessions using the *response_uuid* segment, and determine which frame each came from using the *order-frame_name* segment (e.g., *0-video-consent* or *5-alternation-trial*).

Videos can be downloaded individually here or from the 'Individual Responses' view. You also have the option of bulk downloading all consent videos for your study, or bulk downloading all responses. The bulk download will take place asynchronously, so once the videos have been downloaded and put in a zip file, you will get an email telling you this is done.

1.18.3 What sorts of data (besides video) are collected during a study?

A response record is created each time a participant starts the study. It includes a timestamp, condition assignment, the sequence of frames the participant actually saw, and frame-specific information for each frame.

Each frame type may save different data, e.g. form responses or videos played; frames that record webcam video include the video filename(s). The data captured by a particular frame are listed in the frame documentation under "Data collected."

Additionally, event data is captured for each frame and included under an eventTimings key within the frame data JSON, minimally including a timestamped event when the user proceeds to the next frame. These events are listed under "Events" in the frame documentation.

1.18.4 Hashed and global IDs

Overview

There are “global IDs” you can download for **account**, **child**, and **demographic snapshot** data. You only need them if you need to link participants across studies, either for longitudinal analysis or financial accounting. You may not publish them. Usually, you can just use the regular hashed ID fields which are six-character strings.

Why two different IDs?

Lookit participants may take part in studies from a variety of labs. This means that if researchers directly use the unique database identifiers (“global IDs”) for accounts, children, and demographic data snapshots, different labs will be using the same identifiers for the same children. That’s important to allow collaboration in cases where you have IRB approval to combine data from different studies, but it also means that if you and another lab both published those global IDs, someone else could come along and link data from a participant who did both studies. Usually this would be ok, but in some cases information that wasn’t sensitive on its own in either study could be combined to produce more sensitive or identifying information. So we provide “hashed IDs” which are study-specific and can be published.

What if we do need to link participants across studies and want to publish the data?

There are two options:

1) You can generate your own random unique IDs and replace the global IDs with those in your published data files.
2) You can publish a table of mappings from the hashed ID in one study to the hashed ID in the other, e.g. a CSV file with Study_1_ID, Study_2_ID, etc. headers.

If you do this and come up with scripts that help with the workflow, please share them to help your fellow researchers! You can also file a Github feature request on the lookit-api repo; we will likely offer a convenient download for option 2 once there is demand.

What are the regular IDs?

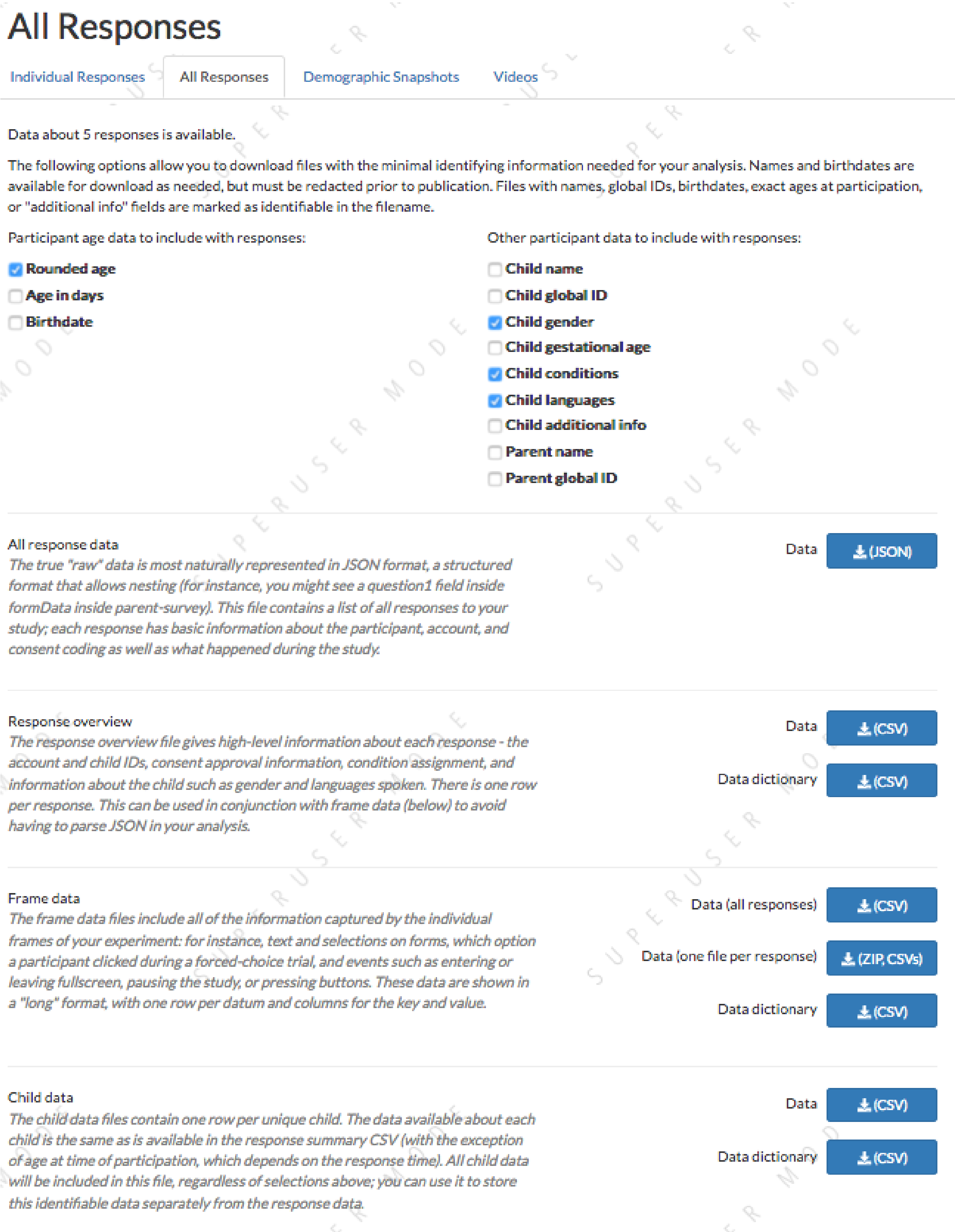
The regular hashed IDs are six-character strings (like *6RYE3U*) that uniquely identify an account, child, or demographic snapshot **within a particular study**. The same child will always have the same hashed ID within a particular Lookit study, but that child would have a different hashed ID in a different study, even one you were running.

These hashed IDs are shown in the consent manager, individual responses, all responses, and email participant views, and may be called simply IDs.

Technical details: The hashed IDs are created by combining the global ID, your study UUID, and a random “salt” associated with your study, running that through a SHA256 hash algorithm, and converting the first 30 bits to a base-32 string representation. Based on “birthday paradox” calculations, we would expect about a 0.1% chance of collisions with 1500 records. A salt is used in addition to the study UUID because the study UUID is easily publicly accessible (e.g. in your study URL) to make it harder to use lookup tables. This procedure would make it quite hard to match IDs across studies. It is meant as a strong deterrent but, given the consequences and the existence of other unavoidable methods for linking responses across studies (e.g., looking at video), it is not a level of security that would be appropriate for, say, hashing passwords.

1.18.5 Viewing all study responses

To view all of the responses to a study with confirmed consent, click ‘View Responses’ from the study detail page and then click ‘All Responses.’ You must have permission to view this study’s responses, which means you must have a Study admin, researcher, or analysis role. (If you can view preview responses, you will be able to access this same page, but only preview data will be included.)



All Responses

Data about 5 responses is available.

The following options allow you to download files with the minimal identifying information needed for your analysis. Names and birthdates are available for download as needed, but must be redacted prior to publication. Files with names, global IDs, birthdates, exact ages at participation, or "additional info" fields are marked as identifiable in the filename.

Participant age data to include with responses:

- ☑ **Rounded age**
- ☐ **Age in days**
- ☐ **Birthdate**

Other participant data to include with responses:

- ☐ **Child name**
- ☐ **Child global ID**
- ☑ **Child gender**
- ☐ **Child gestational age**
- ☑ **Child conditions**
- ☑ **Child languages**
- ☐ **Child additional info**
- ☐ **Parent name**
- ☐ **Parent global ID**

All response data

The true "raw" data is most naturally represented in JSON format, a structured format that allows nesting (for instance, you might see a question1 field inside formData inside parent-survey). This file contains a list of all responses to your study; each response has basic information about the participant, account, and consent coding as well as what happened during the study.

Data (JSON)

Response overview

The response overview file gives high-level information about each response - the account and child IDs, consent approval information, condition assignment, and information about the child such as gender and languages spoken. There is one row per response. This can be used in conjunction with frame data (below) to avoid having to parse JSON in your analysis.

Data (CSV)

Data dictionary (CSV)

Frame data

The frame data files include all of the information captured by the individual frames of your experiment: for instance, text and selections on forms, which option a participant clicked during a forced-choice trial, and events such as entering or leaving fullscreen, pausing the study, or pressing buttons. These data are shown in a "long" format, with one row per datum and columns for the key and value.

Data (all responses) (CSV)

Data (one file per response) (ZIP, CSVs)

Data dictionary (CSV)

Child data

The child data files contain one row per unique child. The data available about each child is the same as is available in the response summary CSV (with the exception of age at time of participation, which depends on the response time). All child data will be included in this file, regardless of selections above; you can use it to store this identifiable data separately from the response data.

Data (CSV)

Data dictionary (CSV)

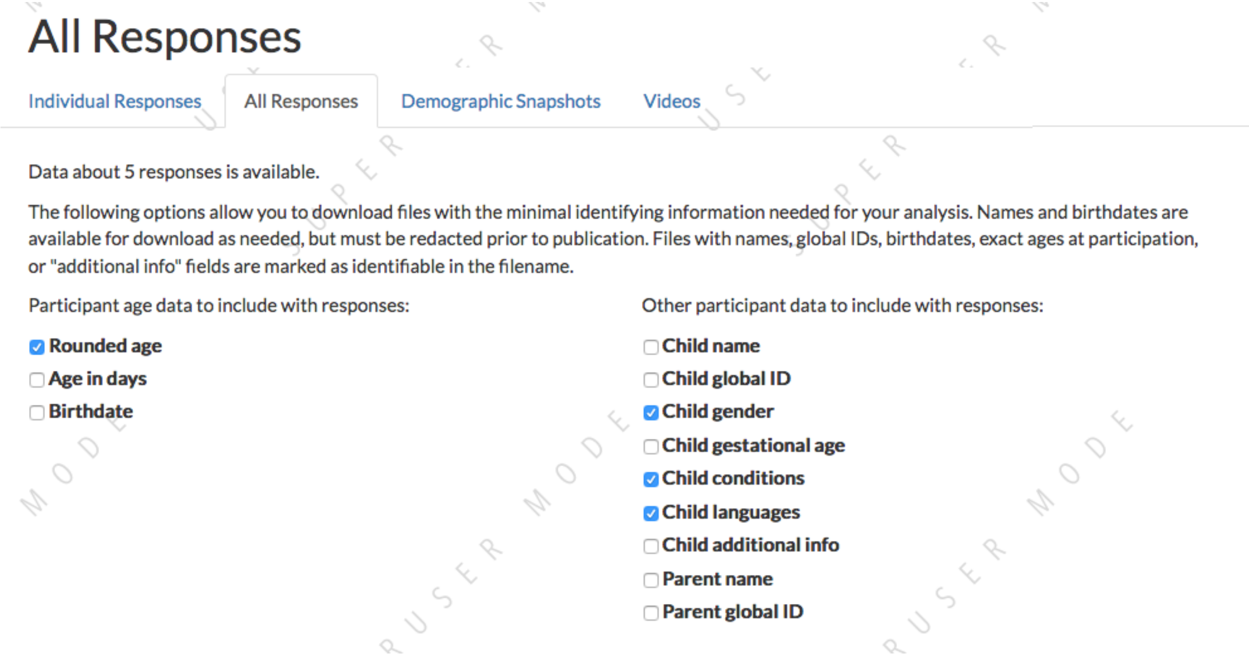
Responses only show up in this view once you have confirmed that the participant provided informed consent to participate using the Consent Manager.

Preview data is included in all responses!

Previewing a study is designed to work *exactly* the same way as participating, including saving data that you can see with other responses. This is intended to support researchers in preparing data analysis workflows and ensuring that data are formatted as expected before starting data collection. But you'll need to either (a) filter out responses where is_preview is True, or (b) use the "Delete all preview data" button to remove preview responses prior to data download.

Choosing what participant information to include

To limit the potential for accidental disclosure of identifying information about your participants, it is best to limit what you even download to what you actually need. The checkboxes at the top allow you to decide what potentially-identifying information to include in the response data files. This also aids in developing a straightforward workflow for publishing your raw data, since you will need to avoid publishing names, birthdates (or information that can be used to calculate a birthdate), and global IDs.



Data about 5 responses is available.

The following options allow you to download files with the minimal identifying information needed for your analysis. Names and birthdates are available for download as needed, but must be redacted prior to publication. Files with names, global IDs, birthdates, exact ages at participation, or "additional info" fields are marked as identifiable in the filename.

Participant age data to include with responses:

☑ Rounded age
□ Age in days
□ Birthdate

Other participant data to include with responses:

□ Child name
□ Child global ID
☑ Child gender
□ Child gestational age
☑ Child conditions
☑ Child languages
□ Child additional info
□ Parent name
□ Parent global ID

What can and can't I publish?

The main items you need to avoid publishing are global IDs, birthdates, names, and demographic survey responses if they can be linked to video also published. For convenience, we note specific fields that must be redacted for publication in the CSV data dictionaries. However, if any of this is unfamiliar, please review the Lookit Terms and Conditions!

On the left are options for downloading information about the age of the participant. You can choose to download actual birthdates, exact ages in days, and/or a rounded age. The rounded age is rounded to the nearest 10 days for children under 365 days and to the nearest 30 days after that.

On the right are options for other participant data to download - name, gender, etc. Again, it's best to only download what you actually need! You can also choose to download a separate child data file so that your response data has only the child ID in it.

The response overview data dictionary has detailed explanations of each of these optional fields.

Data download formats

There are several formats available to download your data:

The raw data is available in **JSON** format; this is a structured, human-readable text format where you will be able to see how data is nested (e.g., a form response within a form within a frame). However, it may require more processing to use in your data analysis workflow (for instance to load it into R).

For convenience, several options are provided for downloading data in CSV (comma separated value) format. CSV data can be easily examined in your spreadsheet editor of choice (like Excel) and loaded into programs like R for analysis.

The **response overview** file provides high-level information about each response and the participating child, with one row per response (a "wide" format). Not everything is included here, because there can be a lot of data per response (e.g., events collected each time the participant clicks something). You can download a data dictionary along with the response overview; this file provides information about how to interpret each column of the data file. When you publish your data, it is always a good idea to include a data dictionary, so this gives you a head start!

The **response frame data** file(s) provide all the data that was collected throughout the session. This is provided as a ZIP archive with one file per response. This data is in a "long" format, where there are few columns and each row represents a single piece of information. So each response will be associated with many rows. You can download a data dictionary for the frame data, too! Because the exact types of data collected will vary across studies based on what frames you use, what questions are in your forms, and so on, you will need to fill in some of the data dictionary to explain what the various fields mean. Some of the data dictionary is filled out for you, and there are placeholders for the study-specific explanations you'll need to add.

The **child data** files provide information about each child associated with at least one study response. There is one row per child, and all of the data from that child's sign-up is available: birthdate, gender, gestational age at birth, languages, conditions, etc. A data dictionary is available. This file is the only one not affected by the selections you make about which potentially-identifying information to include: it will always have names, birthdates, etc. The idea is that if you need that information, you can keep it separate from the response data which you might share more broadly.

Frame IDs

Each frame in a response is identified by an ID, which you will see in both the video filenames and the data downloads. This is generally based on what you called the frame in your protocol's "sequence." When the study starts, your protocol is parsed to create an ordered list of frames, for instance expanding out any groups and randomizers. The frame IDs you will see in the data (response__sequence.0, response__sequence.1, etc. in the response overview CSV; "sequence" in the JSON; `frame_id` in the frame data CSV) start with the index of the frame, starting from 0. For instance, if you started off with frames

```
video-config
video-consent
test-trial
exit-survey
```

then you would see frame IDs

```
0-video-config
1-video-consent
2-test-trial
3-exit-survey
```

These numbers are preserved if someone skips around in the study (due to e.g. selectNextFrame sending to a different frame than the next one, or skipping to the exit survey). E.g. if the participant skipped to the exit survey right after consent, you might see a sequence

```
0-video-config
1-video-consent
3-exit-survey
```

If the participant repeats a frame (e.g. by navigating using a "previous" button, selectNextFrame sending them back, or repeating a frame after pausing) then the sequence will show the actual order they saw frames in, with `-repeat-N` added after repeated frames as needed. For example, if the participant paused the test trial twice and started from the beginning each time, you would see:

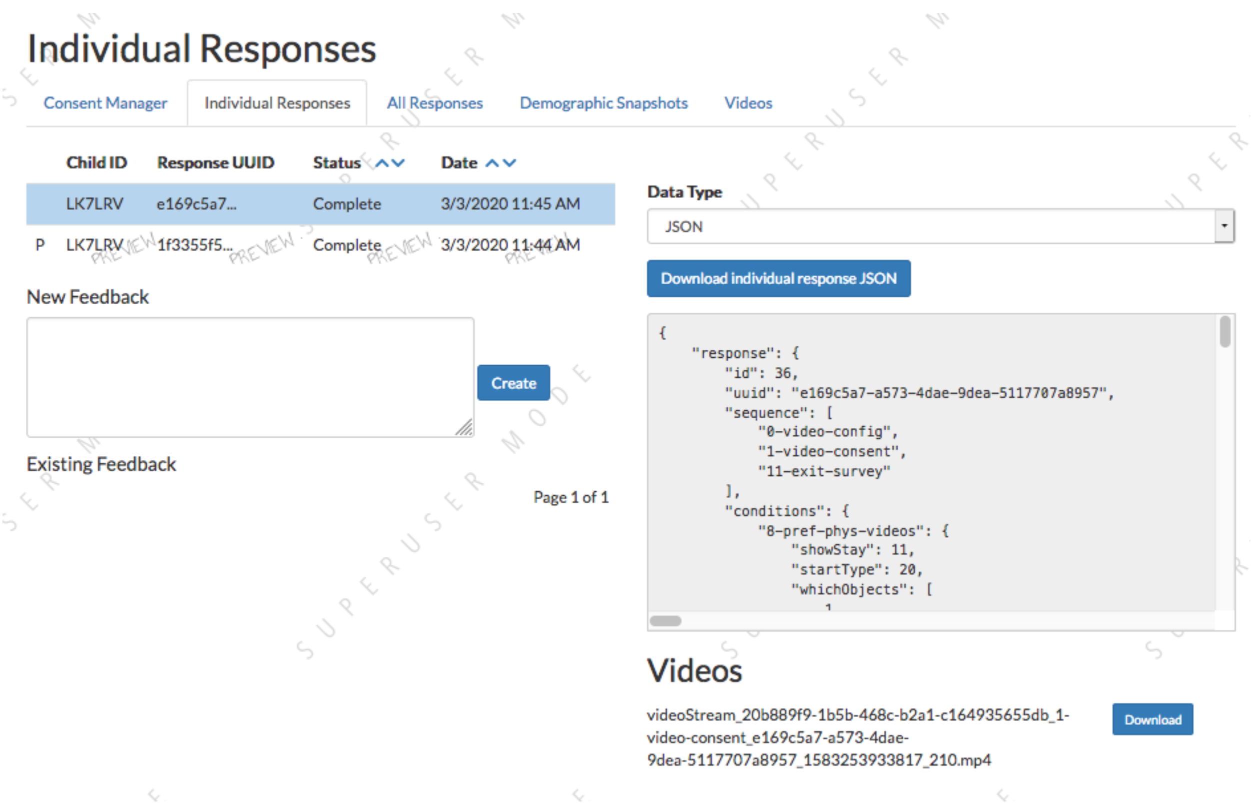
```
0-video-config
1-video-consent
2-test-trial
2-test-trial-repeat-1
2-test-trial-repeat-2
3-exit-survey
```

1.18.6 Viewing individual study responses

To inspect single responses to your study, navigate to your study and click 'View Responses,' then 'Individual responses'. You must have permission to view this study's responses, which means you must have a Study admin, researcher, or analysis role. (If you can view preview responses, you will be able to access this same page, but only preview data will be included.)

Responses only show up in this view once you have confirmed that the participant provided informed consent to participate using the Consent Manager. Both preview and real responses will show up here (depending on your permissions), but preview responses are marked with a "P" and say "PREVIEW" in the background of the row.

On the left, you have a list responses to your study, with the child ID, response ID, the study's completion status, and the date they started the study. When you click on a response, the data from that response is shown on the right. You can download the data from that response in one of several formats: JSON (JavaScript Object Notation, a structured text format); a CSV summary (a "wide format" overview with basic information about the participant and response, such as condition assignment); or CSV frame data (a "long format" detailed list of data collected in each frame during this response, complementary to the CSV summary).



Individual Responses

	Child ID	Response UUID	Status	Date
	LK7LRV	e169c5a7...	Complete	3/3/2020 11:45 AM
P	LK7LRV	1f3355f5...	Complete	3/3/2020 11:44 AM

New Feedback

Existing Feedback

Page 1 of 1

Data Type: JSON

Download individual response JSON

```
{
    "response": {
        "id": 36,
        "uuid": "e169c5a7-a573-4dae-9dea-5117707a8957",
        "sequence": [
            "0-video-config",
            "1-video-consent",
            "11-exit-survey"
        ],
        "conditions": {
            "8-pref-phys-videos": {
                "showStay": 11,
                "startType": 20,
                "whichObjects": [
                    1
```

Videos

videoStream_20b889f9-1b5b-468c-b2a1-c164935655db_1-video-consent_e169c5a7-a573-4dae-9dea-5117707a8957_1583253933817_210.mp4

Beneath the CSV/JSON response data are any individual videos that are linked to that participant's response.

Leaving feedback

On the "Individual Responses" page, you can leave feedback to participants. A lot of the motivation and reward families get from participating in research in person is the social interaction and knowledge that a real human appreciates their time and thinks their kid is super interesting. Feedback is essentially meant to approximate that from an online lab! Typically you might include a quick thanks-again, confirmation that everything worked ok (e.g., everything worked great, we can clearly see him looking right and left), some friendly personalized comment about the child/parent, and a response to any questions parents left in the exit survey. Families can see their feedback by going to Studies -> Past studies, but it is not emailed to them, so don't use this for anything where you really need to reach them (e.g., this is not a good way to send a gift card code!).

1.18.7 Structure of JSON response data

The data saved when a subject participates in a study varies based on how that experiment is defined. Let's start by looking at an example of the data you can download about a single response. (The `eventTimings` objects have been shortened to show just a single event.)

```
{
    "response": {
        "id": 1190,
        "uuid": "d96b3ba5-6806-4c09-86e2-77456163eb5a",
        "is_preview": false,
        "sequence": [
            "0-video-config",
            "1-video-consent",
```

(continues on next page)

(continued from previous page)

```
            "2-instructions",
            "3-mood-survey",
            "4-pref-phys-videos",
            "5-exit-survey"
        ],
        "conditions": {
            "4-pref-phys-videos": {
                "showStay": 18,
                "startType": 21
            }
        },
        "exp_data": {
            "3-mood-survey": {
                "active": "4",
                "rested": "1",
                "healthy": "2",
                "eventTimings": [
                    {
                        "eventType": "exp-mood-questionnaire:nextFrame",
                        "timestamp": "2018-07-06T23:56:06.459Z"
                    }
                ]
            },
            "0-video-config": {
                "eventTimings": [
                    {
                        "eventType": "exp-video-config:recorderReady",
                        "timestamp": "2018-07-06T23:54:59.548Z",
                        "streamTime": null
                    }
                ]
            },
            "2-instructions": {
                "eventTimings": [
                    {
                        "eventType": "exp-physics-intro:nextFrame",
                        "timestamp": "2018-07-06T23:55:53.530Z"
                    }
                ]
            },
            "1-video-consent": {
                "videoId": "videoStream_0f620873-2847-4eeb-9854-df7898934c17_1-video-
↪consent_d96b3ba5-6806-4c09-86e2-77456163eb5a_1530921346557_292",
                "videoList": [
                    "videoStream_0f620873-2847-4eeb-9854-df7898934c17_1-video-consent_
↪d96b3ba5-6806-4c09-86e2-77456163eb5a_1530921346557_292"
                ],
                "eventTimings": [
                    {
                        "eventType": "exp-video-consent:recorderReady",
                        "timestamp": "2018-07-06T23:55:46.558Z",
                        "streamTime": 0
                    }
                ]
            },
            "5-exit-survey": {
                "feedback": "",
```

(continues on next page)

(continued from previous page)

```
                "birthDate": "2018-07-03T04:00:00.000Z",
                "useOfMedia": "private",
                "withdrawal": false,
                "eventTimings": [
                    {
                        "eventType": "exp-exit-survey:nextFrame",
                        "timestamp": "2018-07-06T23:57:02.201Z"
                    }
                ],
                "databraryShare": "no"
            },
            "4-pref-phys-videos": {
                "videoId": "videoStream_0f620873-2847-4eeb-9854-df7898934c17_4-pref-
↪phys-videos_d96b3ba5-6806-4c09-86e2-77456163eb5a_1530921371545_923",
                "videoList": [
                    "videoStream_0f620873-2847-4eeb-9854-df7898934c17_4-pref-phys-
↪videos_d96b3ba5-6806-4c09-86e2-77456163eb5a_1530921371545_923"
                ],
                "videosShown": [
                    "https://s3.amazonaws.com/lookitcontents/exp-physics-final/
↪stimuli/stay/webm/sbs_stay_near_mostly-on_book_c2_green_NN.webm",
                    "https://s3.amazonaws.com/lookitcontents/exp-physics-final/
↪stimuli/stay/webm/sbs_stay_mostly-on_near_book_c2_green_NN.webm"
                ],
                "eventTimings": [
                    {
                        "eventType": "exp-video-physics:recorderReady",
                        "timestamp": "2018-07-06T23:56:11.549Z",
                        "streamTime": 0
                    }
                ]
            }
        },
        "global_event_timings": [],
        "completed": true
    },
    "study": {
        "uuid": "0f620873-2847-4eeb-9854-df7898934c17"
    },
    "participant": {
         "global_id": "",
         "hashed_id": "6RYEUF",
         "nickname": ""
    },
    "child": {
         "global_id": "",
         "hashed_id": "ccNdL6",
         "name": "",
         "birthday": "",
         "age_in_days": "",
         "age_rounded": "960",
         "gender": "f",
         "language_list": "en egy",
         "condition_list": "autism_spectrum_disorder multiple_birth",
         "age_at_birth": "40 or more weeks",
         "additional_information": ""
    }
```

(continues on next page)

(continued from previous page)

```
}
```

There are four top-level keys in this data: `response`, `study`, `participant`, and `child`. Study, participant, and child information should be fairly self-explanatory: which study does this response pertain to, which family account created the response, and which child was participating. (The child key `age_at_birth` refers to gestational age in weeks at birth.) You can find more detail about all of these fields by looking at the CSV data dictionaries available under All Responses; they are "flattened" for the CSV so that the "global_id" field under "child" becomes "child_global_id", for instance.

The `response` data contains information concerning this particular session: when it happened, what condition the child was assigned to, events that happened as the family proceeded through the study, etc. The response properties are described below:

- *id*: short unique ID for the response
- *uuid*: long unique ID for the response (should be used as primary identifier)
- *sequence*: The sequence of **frames** the subject actually saw (after running randomization, etc.). Does not include frames skipped if they left early. The frame names follow the pattern `<order>-<frame.id>`, where `<order>` is the order in the overall sequence where this **frame** appeared, and `<frame.id>` is the identifier of the frame as defined in the 'frames' property of the experiment structure.
- *conditions*: An object containing information about conditions to which the subject was assigned in any frames that do randomization (choice frames). Keys are in the format `<order>-<frame.id>` corresponds with the `<order>` from the 'sequence' of the *original* experiment structure, and the `<frame.id>` again corresponds with the identifier of the frame as defined in the 'frames' property of the experiment structure. Data will be stored in conditions for the *first* frame created by a randomizer (top-level only for now, i.e. not from nested randomizers). Values are objects containing mappings from condition names to their values for this session. The data stored by a particular randomizer can be found under `data collected` in the experiment runner docs
- *global_event_timings*: A list of events recorded during the study, not tied to a particular frame. Currently used for recording early exit from the study; an example value is:

```
[
        {
            "exitType": "manualInterrupt",
            "eventType": "exitEarly",
            "timestamp": "2018-07-06T23:56:55.282Z",
            "lastPageSeen": 10
        }
]
```

- *completed*: A true/false flag indicating whether or not the participant submitted the last frame of the study. Note that this may not line up with your notion of whether the participant completed the study, in two ways: first, `completed` will be true even if the participant leaves early, as long as they submit the exit survey which they skip to when pressing F1. Second, `completed` will be false if they don't submit that exit survey, even if they completed all of the important experimental parts of the study.
- *exp_data*: A JSON object containing the data collected by each **frame** in the study. More on this below…

1.18.8 Interpreting `exp_data`

Here's an example of data collected during a session (note: not all fields are shown):

```
{
    "sequence": [
        "0-intro-video",
        "1-survey",
        "2-exit-survey"
    ],
    "conditions": {
        "1-survey": {
            "parameterSet": {
                "QUESTION1": "What is your favorite color?",
                "QUESTION2": "What is your favorite number?"
            },
            "conditionNum": 0
        }
    },
    "exp_data": {
        "0-intro-video": {
            "eventTimings": [{
                "eventType": "nextFrame",
                "timestamp": "2016-03-23T16:28:20.753Z"
            }]
        },
        "1-survey": {
            "formData": {
                "name": "Sam",
                "favPie": "pecan"
            },
            "eventTimings": [{
                "eventType": "nextFrame",
                "timestamp": "2016-03-23T16:28:26.925Z"
            }]
        },
        "2-exit-survey": {
            "formData": {
                "thoughts": "Great!",
                "wouldParticipateAgain": "Yes"
            },
            "eventTimings": [{
                "eventType": "nextFrame",
                "timestamp": "2016-03-23T16:28:32.339Z"
            }]
        }
    }
}
```

`exp_data` is an object with three keys that correspond with the frame names from 'sequence'. Each of the associated values has an `eventTimings` property. This is a place to collect user-interaction events during an experiment, and by default contains the 'nextFrame' event which records when the user progressed to the next **frame** in the 'sequence'. You can see which events a particular frame records by looking at the "Events" section in its frame documentation. Events recorded by a frame that does video recording will include additional information, for instance to indicate when relative to the video stream this event happened.

The other properties besides 'eventTimings' are dependent on the **frame** type. You can see which other properties a particular frame type records by looking at the "Data collected" section of its documentation.

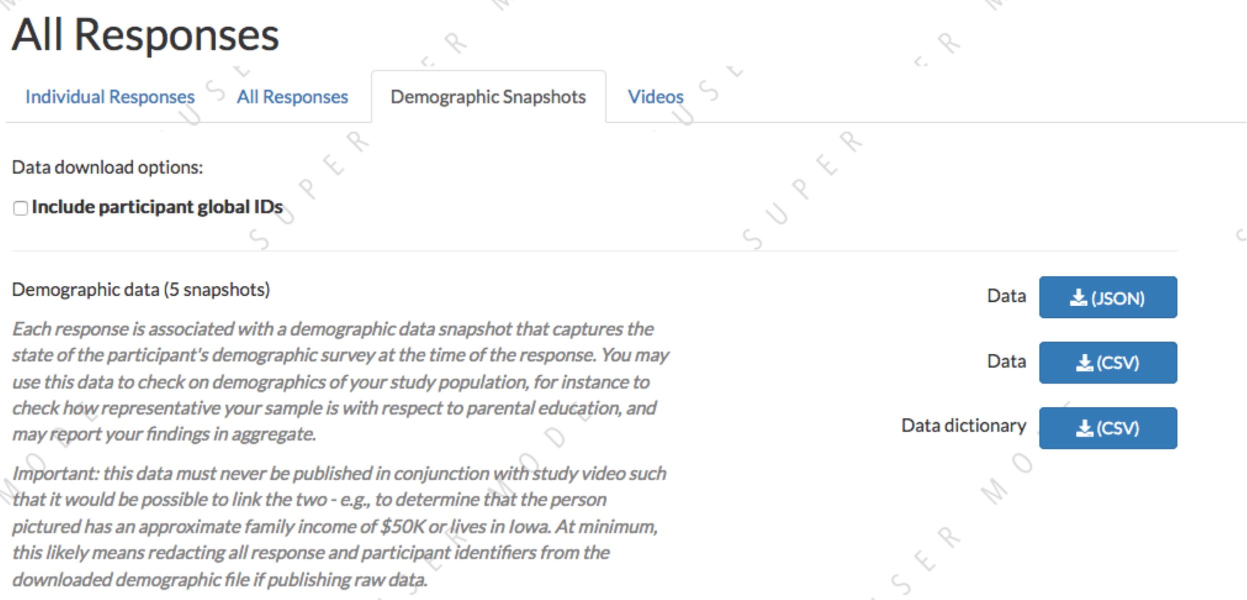
1.18.9 Viewing demographics of study participants

To view the demographics of participants that have responded to your study and have confirmed consent, click 'View Responses' from the study detail page and then click 'Demographic Snapshots.' You must have permission to view this study's responses, which means you must have a Study admin, researcher, or analysis role. (If you can view preview responses, you will be able to access this same page, but only preview data will be included.)

This list of demographic snapshots is generated by looping through all the responses to your study, and displaying the demographics of the associated participant. If a participant has responded multiple times, the demographics will appear multiple times. Demographic data is versioned, so the demographics associated with each response will be the demographics that were current at the time the participant responded to the study.

Similar to the "All responses" download options, you can choose whether to include participant global IDs in the data download. If you don't need them, we recommend omitting them to avoid potential for accidental disclosure.

You can download the demographics in JSON or CSV format. A CSV data dictionary is available for interpretation of the headers in the CSV file.



All Responses

Data download options:

□ **Include participant global IDs**

Demographic data (5 snapshots)

Each response is associated with a demographic data snapshot that captures the state of the participant's demographic survey at the time of the response. You may use this data to check on demographics of your study population, for instance to check how representative your sample is with respect to parental education, and may report your findings in aggregate.

Important: this data must never be published in conjunction with study video such that it would be possible to link the two - e.g., to determine that the person pictured has an approximate family income of $50K or lives in Iowa. At minimum, this likely means redacting all response and participant identifiers from the downloaded demographic file if publishing raw data.

1.19 Day-to-day study operation

1.19.1 Starting and stopping data collection and advertising

At any time, you can change either whether your study is active and whether it is discoverable.

Active vs. paused

You can start and stop data collection independently and whenever you want. On your study detail page, go to "change state" -> "Start" or "Pause."

When your study is active, participants can access it at the direct link shown on your study page. If the study is also discoverable, it will be listed on the Lookit "studies" page and advertised by email to eligible families.

When your study is paused, participants can't access it even if they have the direct link. If participants follow a direct link to your study and it is paused, they will see a message like this.

Discoverable vs. non-discoverable

You can also independently switch your study between discoverable

When your study is **discoverable** and active, it is:

- Listed at lookit.mit.edu/studies
- Advertised to eligible families via announcement emails that include a direct link to the study, sent each day at 4am
- Accessible at the direct link

When your study is **non-discoverable** and active, it is:

- Not listed at lookit.mit.edu/studies
- Not advertised by announcement emails
- Still accessible at the direct link!

You would make your study non-discoverable if you wanted to recruit your own specific participants - e.g., only people who previously participated in a study in your lab, or families in a database for research about a rare disorder.

We also recommend making studies non-discoverable to do some initial piloting - once your study is approved, you can make it active but non-discoverable, and recruit a few participants yourself to try it out. (This is also a good way to get a feel for your recruitment options and how effective they are!)

Announcement emails

Announcement emails are sent out to families with eligible children to let them know about new studies on Lookit:

- Emails are only sent about studies that are **discoverable** and **active**.
- Emails are only sent to families whose email preference specifies that they want to hear about new studies.
- Up to 50 families with at least one eligible child are notified about each study each day. Eligibility is determined each day, so depending on your eligibility criteria you will have a day to a few weeks of sending out 50 emails/day, then a trickle of kids aging in or registering.
- Families are only notified about a study one time per child. (Emails are bundled together if they have multiple eligible children - “There’s a new study for John and Jane!”, but they might get a second email if another child ages in.)
- Emails are currently sent at 4am Eastern Time.

The limit of 50 is currently hard-coded. If you want to limit how many announcement emails are sent further, you can send out just one day of announcements at a time by making your study discoverable overnight, then making it non-discoverable again. Families who got a direct link in an announcement email will be able to participate in your study, but no more announcements will be sent unless you make the study discoverable again.

1.19.2 Making changes to your study

If you make changes to your study - updating the fields in the study details, the experiment runner version, etc. - your study will be automatically “rejected” and will require review by Lookit staff again before you can make it active. This is quick but does still require manual action on our part. In general these are reviewed the same business day as submitted, often within an hour, but this is subject to whether Kim is working / in a meeting / etc. Feel free to contact us ahead of time to coordinate a time if you need immediate turnaround.

There are a few exceptions to changes triggering review, including changes to the age range and formatting of the study protocol configuration, which are listed when you save changes to a study.

1.19.3 Monitoring data as it comes in

We strongly recommend reviewing consent videos and looking through data at least twice a week to make sure you become aware of any issues participants are having in a timely manner. (E.g., you want to know if people are confused by your directions, if there's any weird bug with a new version of Firefox, etc.!)

1.19.4 Missing consent videos

If you come across a record that's missing a consent video, please (a) try refreshing the page to make sure it doesn't show up and then (b) notify Lookit staff, including the response UUID. If you're using a version of the experiment runner prior to 2.2.2 / 1.4.1 (see *releases <https://github.com/lookit/ember-lookit-frameplayer/releases>*) then this is probably an instance of this bug. We can fix particular instances for you but recommend updating following the instructions in this Slack post.

1.19.5 Compensating participants

If you are compensating participants, in most cases you will be messaging them through the Lookit email interface to give them gift cards, using the user IDs you can see in the consent manager and/or the response data. If your institution requires direct compensation or requires the collection of email addresses for other compliance reasons, you can add an additional survey page to your study to ask for the participant's email, as long as it is clear that that information will only be used to send payment.

Participant compensation should never depend on the child's behavior - *even if the child fusses out and/or the data is unusable*. In general, this means we try to pay anyone who submits a valid consent video. Payment for the parent's/child's time is ethical; accidentally paying the parent to take extraordinary measures to get their child to sit through the study because they think that's necessary to get paid is not :)

The exception to the above guideline is that if someone tries the study but only gets a very short way in (maybe only consent), then comes back and does the study again, you would generally pay them just once - even if you might have erred on the side of caution and paid them even for the first attempt.

You are free to put limits on how many times / how often people can participate and be compensated, and to require that the child be, say, in the age range for the study in order to participate. Basically, stuff the parent can know before they get started is fair game. (But be careful and err on the side of payment if there is any discrepancy between your listed age range - e.g. "for three-year-olds" - and the min/max used for automatic warnings; see the docs.) If a parent participates with a child well outside the age range, you might want to email them to thank them for participating, let them know it's fine to check out the study and you hope they found it interesting but since this is for x-month-olds you won't be able to use their data or provide compensation.

Rarely, adults without children may check out a study and even make a consent recording. We tell our students not to do this but you never know :) To avoid feeling obligated to pay them (which would probably be surprising to them too) you're welcome to state in your compensation info that the child needs to be visible in the consent video. (You don't actually have to enforce that for people who get the kid later, which is reasonable - but this way if someone ONLY submits the consent video and doesn't have a child present, you don't have to pay them.)

1.19.6 Parents who ran into a technical problem and want to try again

Sometimes parents may contact you to see if they can try your study again because they had a technical problem or their child wasn't interested the first time. Whether you can use the data may depend on the particular circumstances and your study design, but on a technical level it's fine - you can let the parent know they may see a warning about having already participated but that they can safely ignore it.

1.19.7 Confirming consent

You will need to review consent videos using the Consent Manager tool and determine whether each one represents clear informed consent. (See the docs.) Only after confirming consent do you receive full access to the data collected during the session.

If you come across a video where you think a parent meant to consent to participate, but you do not have an adequate recording, you can email the participant to ask for confirmation. See the 'informed consent guidelines' in the Terms of Use for guidance. Here is an example of an email we have sent to confirm consent:

> Thanks so much for participating in the Lookit study "Your baby the physicist" with your child! We really appreciate your time - and you're one of our first participants, so we're extra excited :)
>
> Unfortunately, we don't have a video recording of you saying you agree to participate - we suspect it may not have been clear that you needed to read that out loud. If it's okay for us to view your videos and use the data, could you respond with "Yes, I am this child's parent or legal guardian and we both agreed to participate in this study"? Thanks again, and I'm very sorry for the extra hassle!
>
> If you did NOT mean to consent to participate in the study, no action is required. You can ignore this email and we will not use your data.

1.19.8 Sending child-related data to families

Parents are able to review their study video in the Lookit interface. If you would like to send them additional information related to their participation, please try to do so using the "Message Participants" interface. That interface supports html but does not allow attachments. If you need to share files with the families, please share a link to the file. For example, Dropbox Business allows file-sharing links to be password-protected with an expiration date.

1.19.9 If a parent requests video deleted, or you need to delete video for any other reason

Please contact Lookit staff and we will delete the video(s). You'll need to provide the response UUID.

If a parent invokes GDPR specifically in their request, again please contact Lookit (complying is straightforward but we'll notify OGC).

1.20 Participant recruitment

1.20.1 Current efforts & past results

The Recruitment *Working Group* is actively exploring and testing out recruitment approaches!

See this Google doc for recent observations about approaches and success.

Here are some general observations:

- Substantial carryover effects to other studies (in different age range) when one group advertises a study! We're getting benefits of cooperation, and really not seeing "competition" between Lookit studies at this point.
- Paying participants makes a huge difference in ability to recruit, as well as diversity of participants. Separately, we believe it's ethical to pay families for their time & contribution. We currently aim for ~$15/hour for the expected duration of the study. Currently done by researchers manually sending gift cards post-study; eventual plans to provide centralized compensation functionality.

1.20.2 Existing avenues for online outreach:

- Facebook page
  - Posting to AcademicMamas w/ babies born 2017 page (and tagging specific moms with kids in age range) has had the most success (but still very small-scale - a total of maybe 10 kids)
  - Occasional attempts at 'boosting' posts advertising studies (most recently, ~6000 views from new parents) has never yielded participants for unpaid studies
  - Interested labs may advertise via the Lookit page - we send an invite to be an editor, you may need to double check at https://www.facebook.com/pages/?category=invites to find it. There are known problems about the invites being hard to find: https://www.facebook.com/business/help/community/question/?id=1249301125083191
  - Experimenting with interest targeting and recording results would be a helpful contribution!

1.20.3 Potential approaches:

- Ask participants to tell their friends
  - Include in study debriefing
  - Provide video for download/sharing
  - Email after study
  - Provide printable flyers on website
- Option to become a parent ambassador/advisor, or just be more available to parents
  - Advance testing of new studies
  - Providing more detailed feedback on studies & parent-facing text
  - Occasional joint meetings, or involvement in other Lookit meetings
  - Share with parent groups online
  - "Office hours" - easy to join group video chat
  - Schedule participants (even though it's unnecessary) to provide "accountability" when people intend to participate
- Cultivate a more active social media presence
  - Regularly posted content - e.g.
    - Info about cognitive development or recent studies, interesting articles
    - Cute videos of participants
    - Status updates
    - Intros to lab members
    - Q&A about cognitive development
    - Encourage sharing cute pictures/videos on particular topics
    - "Ask your kid" feature - get funny answers to questions
  - Read about how to do this and/or get help from people who know what they're doing

- In-person local advertising (see Rianna's Cambridge/Boston map). Put up flyers and/or talk to local institutions/people that work with kids to get them excited about Lookit, ask them to mention it to families, maintain relationships. Examples of places we've looked when advertising locally in Cambridge:
  - Look at lists like Boston Coop, Mommy Poppins drop-in indoor playspaces, City Moms Blog to gather more ideas
  - Activities: Cambridge Public Library lapsit, other activities like Music Together
  - Parent groups: look for new parent support groups, new parent education, baby playgroups.
  - Toy stores; Baby stuff secondhand shops, e.g. Two Little Monkeys
  - Pediatricians' offices
  - Parks
  - Daycares, preschools, elementary schools, afterschool programs
  - MIT museum (not because they get a ton of kids but because we already have a connection via MIT, and in the future maybe they'd like to host an interactive Lookit display. . .)
  - Cambridge Family Resource Center
  - Baby U, Center for Families playgroups, other parent ed
- Go to events with kids and talk with families directly
  - Cambridge Science Festival
  - Keep an eye out on various calendars for events (e.g., Cambridge DHSP)
  - Boston Children's Museum
- Possible features to support recruitment
  - Point system
  - Let parents comment on studies publicly?
- Media/online outreach
  - Talk to groups that do online research with adults (e.g. LabInTheWild, TestMyBrain) to see if they'd be up for linking to Lookit
  - Talk to groups that do online research with kids - e.g. https://www.babysleepstudy.org/studysignup,
  - Ask institutions that might be interested about featuring/linking to Lookit, or publishing content we provide - examples:
    - Boston Children's Museum
    - https://www.parentingscience.com/online-parenting-studies.html
    - http://www.scholastic.com/parents/blogs
    - https://www.sciencedaily.com/news/mind_brain/child_psychology/
    - https://theconversation.com/us (write an article?)
    - http://www.parenting.com/
    - https://www.babycenter.com/
    - https://mommypoppins.com/kids/drop-in-indoor-play-spaces-for-boston-babies-toddlers-and-preschoolers (can request listing)
    - https://boston.citymomsblog.com/guide/8-rainy-day-infant-toddler-friendly-activities-in-boston/ (guest post submission)

  - Work w/ parenting-focused bloggers (esp. who don't focus exclusively on paid review type posts)
  - Ask for help from MIT media folks
  - Posting in parent groups - BabyCenter, Facebook birth clubs, etc.
- Better organize efforts across labs
- Piggyback on local efforts to send out mailings
- Actual advertising - e.g. Facebook, Google AdWords, magazine or public transit ad space

1.21 Using the API

1.21.1 What is the API for?

Using the Lookit API allows you to programatically retrieve or update data (other than video data), rather than manually downloading JSON or CSV files from the Experimenter site.

Researchers do not need to use the API, but it is available if preferred to downloading data via the experimenter app.

1.21.2 API Tips

General

Most endpoints in this API are just meant for retrieving data. Typically, you can retrieve data associated with studies you have permission to view, or view any data that belongs to you. You can only create *responses* and *feedback* through the API. You can only update *responses* and *feedback* through the API. There is *nothing* that is permitted to be deleted through the API. For a set of sample functions using the API from Python, please see https://github.com/kimberscott/lookit-data-processing/blob/coding-workflow-multilab/scripts/experimenter.py

API Formatting

This API generally conforms to the JSON-API 1.0 spec . Top-level keys are underscored, while nested key formatting will be the casing that is stored in the db. For example, in the study response below, top-level attributes like *exp_data* and *global_event_timings* are underscored. However, nested keys like *5-5-mood-survey* and *napWakeUp* retain the casing given to them by the *exp-player*.

```
{
    "type": "responses",
    "id": "bdebd15b-adc7-4377-b2f6-e9f3de70dd19",
    "attributes": {
        "conditions": {
            "8-pref-phys-videos": {
                "showStay": 8,
                "startType": 5,
                "whichObjects": [
                    8,
                    5,
                    6,
                    8
                ]
            }
        },
```

(continues on next page)

(continued from previous page)

```
        "global_event_timings": [
            {
                "exitType": "browserNavigationAttempt",
                "eventType": "exitEarly",
                "timestamp": "2017-10-31T20:30:38.514Z",
                "lastPageSeen": 6
            }
        ],
        "exp_data": {
            "5-5-mood-survey": {
                "active": "1",
                "rested": "1",
                "healthy": "1",
                "lastEat": "6:00",
                "energetic": "1",
                "napWakeUp": "11:00",
                "childHappy": "1",
                "doingBefore": "s",
                "parentHappy": "1",
                "eventTimings": [
                    {
                        "eventType": "nextFrame",
                        "timestamp": "2017-10-31T20:10:17.269Z"
                    }
                ],
                "ontopofstuff": "1",
                "usualNapSchedule": "no"
            }
        },
        "sequence": [
            "5-5-mood-survey"
        ],
        "completed": false
    },
    "relationships": {
        "child": {
            "links": {
                "related": "http://localhost:8000/api/v1/children/da27faf2-c3d2-4701-
↪b3bb-dd865f89c1a1/"
            }
        },
        "study": {
            "links": {
                "related": "http://localhost:8000/api/v1/studies/e729321f-418f-4728-
↪992c-9364623dbe9b/"
            }
        },
        "demographic_snapshot": {
            "links": {
                "related": "http://localhost:8000/api/v1/demographics/341ea7c7-f657-
↪4ab2-a530-21ac293e7d6f/"
            }
        }
    },
    "links": {
        "self": "http://localhost:8000/api/v1/responses/bdebd15b-adc7-4377-b2f6-
↪e9f3de70dd19/"
```

(continues on next page)

(continued from previous page)

```
    }
}
```

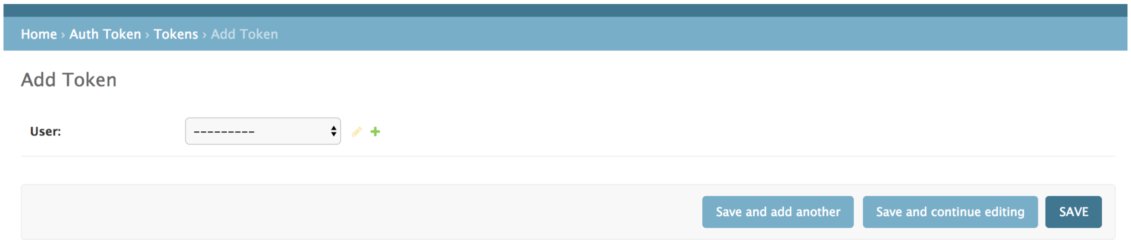
Content-Type

The following Content-Type must be in the header of the request: *application/vnd.api+json*.

Authentication

We are using a token-based HTTP Authentication scheme.

- Ask Lookit staff for a token.
  - Tokens are currently created via the Admin app accessible to staff only. Go to `/__CTRL__/authtoken/token/add/`:

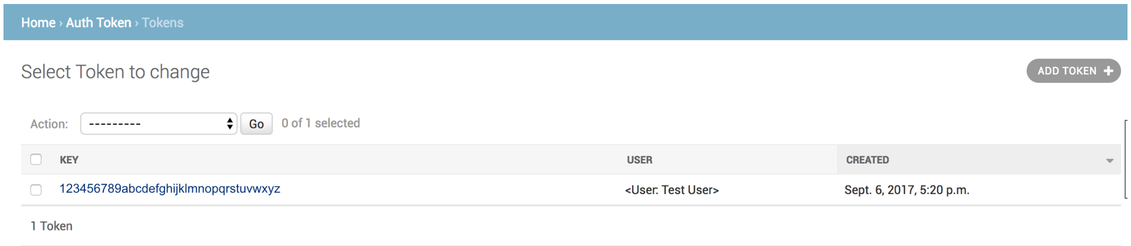


Home › Auth Token › Tokens › Add Token

Add Token

User: ---------

Select the user from the dropdown and hit 'Save'. Copy the token.



Home › Auth Token › Tokens

Select Token to change

Action: --------- Go 0 of 1 selected

□ KEY	USER	CREATED
□ 123456789abcdefghijklmnopqrstuvwxyz	<User: Test User>	Sept. 6, 2017, 5:20 p.m.

1 Token

- Include this token in your Authorization HTTP header. The word "Token" should come before it.

```
curl -X GET <API_URL_HERE> -H 'Authorization: Token <paste_token_here>'
```

- For example, here's how you would access users using curl:

```
curl -X GET https://lookit.mit.edu/api/v1/users/ -H 'Authorization: Token␣
↪123456789abcdefghijklmnopqrstuvwxyz'
```

- Here is an example of a POST request using curl, note the presence of the content-type header as well as the authorization header:

```
curl -X POST  http://lookit.mit.edu/api/v1/feedback/ -H "Content-Type: application/
↪vnd.api+json" -H 'Authorization: Token abcdefghijklmnopqrstuvwxyzyour-token-here' -
↪d '{"data": {"attributes": {"comment": "Test comment"}, "relationships": {"response
↪": {"data": {"type": "responses","id": "91c15b81-bb25-437a-8299-13cf4c83fed6"}}},
↪"type": "feedback"}}'
```

Pagination

- This API is paginated, so results are returned in batches of 10. Follow the pagination links in the API response to fetch the subsequent pages of data. In the example below, the "links" section of the API response has the first, last, next, and previous links.

Sample Response:

```
{
    "links": {
        "first": "http://localhost:8000/api/v1/responses/?page=1",
        "last": "http://localhost:8000/api/v1/responses/?page=5",
        "next": "http://localhost:8000/api/v1/responses/?page=2",
        "prev": null,
        "meta": {
            "page": 1,
            "pages": 5,
            "count": 50
        }
    }
}
```

1.21.3 Available Endpoints

Children

Viewing the list of children

GET /api/v1/children/

Permissions: Must be authenticated. You can only view children that have responded to studies you have permission to view, or your own children. Users with *can_read_all_user_data* permissions can view all children of active users in the database via this endpoint.

Ordering: Children can be sorted by birthday using the *ordering* query parameter. For example, to sort oldest to youngest:

GET http://lookit.mit.edu/api/v1/children/?ordering=birthday

Add a '-' before birthday to sort youngest to oldest:

GET http://lookit.mit.edu/api/v1/children/?ordering=-birthday

Sample Response:

```
{
    "links": {
        "first": "http://localhost:8000/api/v1/children/?page=1",
        "last": "http://localhost:8000/api/v1/children/?page=1",
        "next": null,
        "prev": null,
        "meta": {
            "page": 1,
            "pages": 1,
            "count": 1
        }
    },
```

(continues on next page)

(continued from previous page)

```
    "data": [
        {
            "type": "children",
            "id": "0b380366-31b9-45c1-86ef-0fd9ea238ff4",
            "attributes": {
                "given_name": "Ashley",
                "birthday": "2015-01-01",
                "gender": "f",
                "age_at_birth": "36",
                "additional_information": "",
                "deleted": false
            },
            "relationships": {
                "user": {
                    "links": {
                        "related": "http://localhost:8000/api/v1/users/834bbf33-b249-
↪4737-a041-43574cd137a7/"
                    }
                }
            },
            "links": {
                "self": "http://localhost:8000/api/v1/children/0b380366-31b9-45c1-
↪86ef-0fd9ea238ff4/"
            }
        }
    ]
}
```

Retrieving a single child

GET /api/v1/children/<child_id>/

Permissions: Must be authenticated. You can only view a child if he or she has responded to a study you have permission to view. You can additionally view your own child via the API.

Sample Response:

```
{
    "data": {
        "type": "children",
        "id": "0b380366-31b9-45c1-86ef-0fd9ea238ff4",
        "attributes": {
            "given_name": "Ashley",
            "birthday": "2015-01-01",
            "gender": "f",
            "age_at_birth": "36",
            "additional_information": "",
            "deleted": false
        },
        "relationships": {
            "user": {
                "links": {
                    "related": "http://localhost:8000/api/v1/users/834bbf33-b249-4737-
↪a041-43574cd137a7/"
                }
```

(continues on next page)

(continued from previous page)

```
            }
        },
        "links": {
            "self": "http://localhost:8000/api/v1/children/0b380366-31b9-45c1-86ef-
↪0fd9ea238ff4/"
        }
    }
}
```

Creating a Child

POST /api/v1/children/

METHOD NOT ALLOWED. Not permitted via the API.

Updating a Child.

PUT /api/v1/children/<child_id>/

METHOD NOT ALLOWED. Not permitted via the API.

Deleting a Child

DELETE /api/v1/children/<child_id>/

METHOD NOT ALLOWED. Not permitted via the API.

Demographic Data

Viewing the list of demographic data

GET /api/v1/demographics/

Permissions: Must be authenticated. You can only view demographics of participants whose children have responded to studies you can view. You can additionally view your own demographic data via the API. Users with *can_read_all_user_data* permissions can view all demographics of active users in the database via this endpoint.

Sample Response:

```
{
    "links": {
        "first": "http://localhost:8000/api/v1/demographics/?page=1",
        "last": "http://localhost:8000/api/v1/demographics/?page=1",
        "next": null,
        "prev": null,
        "meta": {
            "page": 1,
            "pages": 1,
            "count": 1
        }
    },
    "data": [
```

(continues on next page)

(continued from previous page)

```
        {
            "type": "demographics",
            "id": "f5fa60ca-d428-46cd-9820-846492dd9900",
            "attributes": {
                "number_of_children": "1",
                "child_birthdays": [
                    "2015-01-01"
                ],
                "languages_spoken_at_home": "English and French",
                "number_of_guardians": "2",
                "number_of_guardians_explanation": "",
                "race_identification": [
                    "white"
                ],
                "age": "30-34",
                "gender": "f",
                "education_level": "grad",
                "spouse_education_level": "bach",
                "annual_income": "30000",
                "number_of_books": 100,
                "additional_comments": "",
                "country": "US",
                "state": "AZ",
                "density": "urban",
                "extra": {
                    "no": "extra"
                }
            },
            "links": {
                "self": "http://localhost:8000/api/v1/demographics/f5fa60ca-d428-46cd-
↪9820-846492dd9900/"
            }
        }
    ]
}
```

Retrieving a single piece of demographic data

GET /api/v1/demographics/<demographic_data_id>/

Permissions: Must be authenticated. You can only view demographics of participants whose children have responded to studies you can view. You can additionally view your own demographic data via the API.

Sample Response:

```
{
    "data": {
        "type": "demographics",
        "id": "f5fa60ca-d428-46cd-9820-846492dd9900",
        "attributes": {
            "number_of_children": "1",
            "child_birthdays": [
                "2015-01-01"
            ],
            "languages_spoken_at_home": "English and French",
```

(continues on next page)

(continued from previous page)

```
            "number_of_guardians": "2",
            "number_of_guardians_explanation": "",
            "race_identification": [
                "white"
            ],
            "age": "30-34",
            "gender": "f",
            "education_level": "grad",
            "spouse_education_level": "bach",
            "annual_income": "30000",
            "number_of_books": 100,
            "additional_comments": "",
            "country": "US",
            "state": "AZ",
            "density": "urban",
            "extra": {
                "no": "extra"
            }
        },
        "links": {
            "self": "http://localhost:8000/api/v1/demographics/f5fa60ca-d428-46cd-
↪9820-846492dd9900/"
        }
    }
}
```

Creating Demographics

POST /api/v1/demographics/

METHOD NOT ALLOWED. Not permitted via the API.

Updating Demographics

PUT /api/v1/demographics/<demographic_data_id>/

METHOD NOT ALLOWED. Not permitted via the API.

Deleting Demographics

DELETE /api/v1/demographics/<demographic_data_id>/

METHOD NOT ALLOWED. Not permitted via the API.

Feedback

Viewing the list of feedback

GET /api/v1/feedback/

Permissions: Must be authenticated. You can only view feedback on study responses you have permission to view. Additionally, you can view feedback left on your own responses.

Sample Response:

```
{
    "links": {
        "first": "http://localhost:8000/api/v1/feedback/?page=1",
        "last": "http://localhost:8000/api/v1/feedback/?page=1",
        "next": null,
        "prev": null,
        "meta": {
            "page": 1,
            "pages": 1,
            "count": 1
        }
    },
    "data": [
        {
            "type": "feedback",
            "id": "cbfc64ee-30a3-491e-bd0e-1bef81540ea5",
            "attributes": {
                "comment": "Thanks for participating!  Next time, please center the␣
↪webcam; you were off-center in many of the video clips."
            },
            "relationships": {
                "response": {
                    "links": {
                        "related": "http://localhost:8000/api/v1/responses/841c8a77-
↪b322-4e25-8e03-47a83fa326ff/"
                    }
                },
                "researcher": {
                    "links": {
                        "related": "http://localhost:8000/api/v1/users/834bbf33-b249-
↪4737-a041-43574cd137a7/"
                    }
                }
            },
            "links": {
                "self": "http://localhost:8000/api/v1/feedback/cbfc64ee-30a3-491e-
↪bd0e-1bef81540ea5/"
            }
        }
    ]
}
```

Retrieving a single piece of feedback

GET /api/v1/feedback/<feedback_id>/

Permissions: Must be authenticated. You can only retrieve feedback attached to a study response you have permission to view. Additionally, you can retrieve feedback attached to one of your own responses.

Sample Response:

```
{
    "data": {
        "type": "feedback",
        "id": "cbfc64ee-30a3-491e-bd0e-1bef81540ea5",
```

(continues on next page)

(continued from previous page)

```
        "attributes": {
            "comment": "Thanks for participating!  Next time, please center the␣
↪webcam; you were off-center in many of the video clips."
        },
        "relationships": {
            "response": {
                "links": {
                    "related": "http://localhost:8000/api/v1/responses/841c8a77-b322-
↪4e25-8e03-47a83fa326ff/"
                }
            },
            "researcher": {
                "links": {
                    "related": "http://localhost:8000/api/v1/users/834bbf33-b249-4737-
↪a041-43574cd137a7/"
                }
            }
        },
        "links": {
            "self": "http://localhost:8000/api/v1/feedback/cbfc64ee-30a3-491e-bd0e-
↪1bef81540ea5/"
        }
    }
}
```

Creating Feedback

POST /api/v1/feedback/

Permissions: Must be authenticated. Must have permission to edit the study response where you are leaving feedback (which only admins have).

Sample Request body:

```
{
 "data": {
       "attributes": {
         "comment": "Thank you so much for participating in round one! Please try to␣
↪respond to the second round some time in the next three weeks!"
       },
       "relationships": {
         "response": {
           "data": {
             "type": "responses",
             "id": "841c8a77-b322-4e25-8e03-47a83fa326ff"
           }
         }
       },
       "type": "feedback"
     }
}
```

Sample Response

```
{
    "data": {
        "type": "feedback",
        "id": "aabf86c7-3dc0-4284-844c-89e04a1f154f",
        "attributes": {
            "comment": "Thank you so much for participating in round one! Please try␣
↪to respond to the second round some time in the next three weeks!"
        },
        "relationships": {
            "response": {
                "links": {
                    "related": "http://localhost:8000/api/v1/responses/841c8a77-b322-
↪4e25-8e03-47a83fa326ff/"
                }
            },
            "researcher": {
                "links": {
                    "related": "http://localhost:8000/api/v1/users/834bbf33-b249-4737-
↪a041-43574cd137a7/"
                }
            }
        },
        "links": {
            "self": "http://localhost:8000/api/v1/feedback/aabf86c7-3dc0-4284-844c-
↪89e04a1f154f/"
        }
    }
}
```

Updating Feedback

PATCH /api/v1/feedback/<feedback_id>/

Permissions: Must be authenticated. Must have permission to edit the study response where you are changing feedback (which only admins have).

Sample Request body:

```
{
    "data": {
        "attributes": {
         "comment": "Changed comment"
        },
        "type": "feedback",
        "id": "ebf41029-02d7-49f5-8adb-1e32d4ac22a5"
    }
}
```

Deleting Feedback

DELETE /api/v1/feedback/<feedback_id>/

METHOD NOT ALLOWED. Not permitted via the API.

Labs

Viewing the list of labs

GET /api/v1/labs/

Permissions: Must be authenticated.

Sample Response:

```
{
    "links": {
        "first": "https://lookit-staging.mit.edu/api/v1/labs/?page=1",
        "last": "https://lookit-staging.mit.edu/api/v1/labs/?page=1",
        "next": null,
        "prev": null,
        "meta": {
            "page": 1,
            "pages": 1,
            "count": 2
        }
    },
    "data": [
        {
            "type": "labs",
            "id": "a2a7383c-cb58-4d78-ac00-23283a762dec",
            "attributes": {
                "name": "Demo lab",
                "institution": "Lookit",
                "principal_investigator_name": "Sample Name",
                "lab_website": "https://lookit.mit.edu/",
                "description": "This is a sample lab researchers are added to upon␣
↪joining Lookit. It contains several demo\r\n                studies you will be␣
↪able to see.",
                "approved_to_test": true,
                "pk": 2
            },
            "links": {
                "self": "https://lookit-staging.mit.edu/api/v1/labs/a2a7383c-cb58-
↪4d78-ac00-23283a762dec/"
            }
        },
        {
            "type": "labs",
            "id": "1c54bccd-5f3d-4dd1-967a-0f7c0565d76d",
            "attributes": {
                "name": "Early Childhood Cognition Lab",
                "institution": "MIT",
                "principal_investigator_name": "Laura Schulz",
                "lab_website": "http://eccl.mit.edu/",
                "description": "We study how children construct a commonsense␣
↪understanding of the physical and social world. \n                Current lab␣
↪members are especially interested in how children generate new ideas and choose␣
↪which problems \n                  are worth working on.\n                 Research in␣
↪the lab often addresses 1) how children figure out cause-and-effect relations so␣
↪that they can \n                  predict, explain, and themselves cause things to␣
↪happen; 2) influences on curiosity and exploration; and 3) \n                 how␣
↪these abilities interact with social cognition to help children understand␣
↪themselves and other people. \n                 ",
```

(continues on next page)

(continued from previous page)

```
                "approved_to_test": true,
                "pk": 3
            },
            "links": {
                "self": "https://lookit-staging.mit.edu/api/v1/labs/1c54bccd-5f3d-
↪4dd1-967a-0f7c0565d76d/"
            }
        }
    ]
}
```

Retrieving a single lab

GET /api/v1/labs/<lab_id>/

Permissions: Must be authenticated.

Sample Response:

```
{
    "data": {
        "type": "labs",
        "id": "a2a7383c-cb58-4d78-ac00-23283a762dec",
        "attributes": {
            "name": "Demo lab",
            "institution": "Lookit",
            "principal_investigator_name": "Sample Name",
            "lab_website": "https://lookit.mit.edu/",
            "description": "This is a sample lab researchers are added to upon␣
↪joining Lookit. It contains several demo\r\n                studies you will be␣
↪able to see.",
            "approved_to_test": true,
            "pk": 2
        },
        "links": {
            "self": "https://lookit-staging.mit.edu/api/v1/labs/a2a7383c-cb58-4d78-
↪ac00-23283a762dec/"
        }
    }
}
```

Creating a lab

POST /api/v1/labs/

METHOD NOT ALLOWED. Not permitted via the API.

Updating a lab

PUT /api/v1/lab/<lab_id>/

METHOD NOT ALLOWED. Not permitted via the API.

Deleting a Lab

DELETE /api/v1/labs/<lab_id>/

METHOD NOT ALLOWED. Not permitted via the API.

Responses

Viewing the list of responses

GET /api/v1/responses/

Permissions: Must be authenticated. You can only view responses to studies you have permission to view. Additionally, you can view your own responses through the API.

Sort Order: By default, responses are sorted reverse date_modified, meaning the most recently modified responses appear first.

Sample Response:

```
{
    "links": {
        "first": "http://localhost:8000/api/v1/feedback/?page=1",
        "last": "http://localhost:8000/api/v1/feedback/?page=1",
        "next": null,
        "prev": null,
        "meta": {
            "page": 1,
            "pages": 1,
            "count": 1
        }
    },
    "data": [
      {
         "type":"responses",
         "id":"8260ca67-6ec0-4749-ba11-fa35612ea030",
         "attributes":{
            "conditions":{

            },
            "global_event_timings":[
               {
                  "exit_type":"browserNavigationAttempt",
                  "timestamp":"2017-09-05T14:33:41.322Z",
                  "event_type":"exitEarly",
                  "last_page_seen":0
               }
            ],
            "exp_data":{

            },
            "sequence":[

            ],
            "completed":false
         },
         "relationships":{
```

(continues on next page)

(continued from previous page)

```
            "child":{
               "links":{
                  "related":"http://localhost:8000/api/v1/children/0b380366-31b9-45c1-
↪86ef-0fd9ea238ff4/"
               }
            },
            "study":{
               "links":{
                  "related":"http://localhost:8000/api/v1/studies/a8a80880-5539-4650-
↪9387-c62afa202d43/"
               }
            },
            "demographic_snapshot":{
               "links":{
                  "related":"http://localhost:8000/api/v1/demographics/f5fa60ca-d428-
↪46cd-9820-846492dd9900/"
               }
            }
         },
         "links":{
            "self":"http://localhost:8000/api/v1/responses/8260ca67-6ec0-4749-ba11-
↪fa35612ea030/"
         }
      }
    ]
}
```

Retrieving a single response

GET /api/v1/responses/<response_id>/

Permissions: Must be authenticated. You can only view responses to studies you have permission to view as well as your own responses.

Sample Response:

```
{
    "data": {
        "type": "responses",
        "id": "8260ca67-6ec0-4749-ba11-fa35612ea030",
        "attributes": {
            "conditions": {},
            "global_event_timings": [
                {
                    "exit_type": "browserNavigationAttempt",
                    "timestamp": "2017-09-05T14:33:41.322Z",
                    "event_type": "exitEarly",
                    "last_page_seen": 0
                }
            ],
            "exp_data": {},
            "sequence": [],
            "completed": false
        },
        "relationships": {
```

(continues on next page)

(continued from previous page)

```
            "child": {
                "links": {
                    "related": "http://localhost:8000/api/v1/children/0b380366-31b9-
↪45c1-86ef-0fd9ea238ff4/"
                }
            },
            "study": {
                "links": {
                    "related": "http://localhost:8000/api/v1/studies/a8a80880-5539-
↪4650-9387-c62afa202d43/"
                }
            },
            "demographic_snapshot": {
                "links": {
                    "related": "http://localhost:8000/api/v1/demographics/f5fa60ca-
↪d428-46cd-9820-846492dd9900/"
                }
            }
        },
        "links": {
            "self": "http://localhost:8000/api/v1/responses/8260ca67-6ec0-4749-ba11-
↪fa35612ea030/"
        }
    }
}
```

Creating a Response

POST /api/v1/responses/. Possible to do programmatically, but really intended to be used by ember-lookit-frameplayer app.

Permissions: Must be authenticated. Child in response must be your child.

Sample Request body:

```
{
    "data": {
        "attributes": {},
        "relationships": {
          "child": {
            "data": {
              "type": "children",
              "id": "0b380366-31b9-45c1-86ef-0fd9ea238ff4"
            }
          },
          "study": {
            "data": {
              "type": "studies",
              "id": "a8a80880-5539-4650-9387-c62afa202d43"
            }
          }
        },
    "type": "responses"
    }
}
```

Updating a Response

PATCH /api/v1/responses/<response_id>/ Possible to do programmatically, but really intended for the ember-lookit-frameplayer to update as it moves through each frame of the study.

Sample Request body:

```
{
 "data": {
   "attributes": {
       "conditions": {"cloudy": "skies"}
   },
   "type": "responses",
   "id": "51c0a355-375d-481f-a3d0-6471db8f9f14"
 }
}
```

Deleting a Response

DELETE /api/v1/responses/<response_id>/

METHOD NOT ALLOWED. Not permitted via the API.

Studies

Viewing the list of studies

GET /api/v1/studies/

Permissions: Must be authenticated. You can view studies that are active/public as well as studies you have permission to edit.

Sort Order: By default, studies are sorted reverse date_modified, meaning the most recently modified studies appear first.

Sample Response:

```
{
    "links": {
        "first": "http://localhost:8000/api/v1/studies/?page=1",
        "last": "http://localhost:8000/api/v1/studies/?page=1",
        "next": null,
        "prev": null,
        "meta": {
            "page": 1,
            "pages": 1,
            "count": 1
        }
    },
    "data": [
        {
            "type": "studies",
            "id": "65680ade-510c-4437-a58a-e41d4b94d8ed",
            "attributes": {
                "name": "Sample Study",
```

(continues on next page)

(continued from previous page)

```
                "date_modified": "2017-09-06T19:33:24.826892Z",
                "short_description": "A short description of your study would go here.
↪",
                "long_description": "A longer purpose of your study would be here.",
                "criteria": "Children should be around five.",
                "duration": "20 minutes",
                "contact_info": "Contact Sally",
                "image": "http://localhost:8000/media/study_images/download.jpeg",
                "structure": {
                    "frames": {},
                    "sequence": []
                },
                "display_full_screen": true,
                "exit_url": "http://www.cos.io",
                "state": "created",
                "public": true
            },
            "relationships": {
                "creator": {
                    "links": {
                        "related": "http://localhost:8000/api/v1/users/834bbf33-b249-
↪4737-a041-43574cd137a7/"
                    }
                },
                "responses": {
                    "links": {
                        "related": "http://localhost:8000/api/v1/studies/65680ade-
↪510c-4437-a58a-e41d4b94d8ed/responses/"
                    }
                }
            },
            "links": {
                "self": "http://localhost:8000/api/v1/studies/65680ade-510c-4437-a58a-
↪e41d4b94d8ed/"
            }
        }
    ]
}
```

Retrieving a single study

GET /api/v1/studies/<study_id>/

Permissions: Must be authenticated. You can fetch an active study or a study you have permission to edit.

Sample Response:

```
{
    "data": {
        "type": "studies",
        "id": "65680ade-510c-4437-a58a-e41d4b94d8ed",
        "attributes": {
            "name": "Sample Study",
            "date_modified": "2017-09-06T19:33:24.826892Z",
            "short_description": "A short description of your study would go here.",
```

(continues on next page)

(continued from previous page)

```
            "long_description": "A longer purpose of your study would be here.",
            "criteria": "Children should be around five.",
            "duration": "20 minutes",
            "contact_info": "Contact Sally",
            "image": "http://localhost:8000/media/study_images/download.jpeg",
            "structure": {
                "frames": {},
                "sequence": []
            },
            "display_full_screen": true,
            "exit_url": "http://www.cos.io",
            "state": "created",
            "public": true
        },
        "relationships": {
            "creator": {
                "links": {
                    "related": "http://localhost:8000/api/v1/users/834bbf33-b249-4737-
↪a041-43574cd137a7/"
                }
            },
            "responses": {
                "links": {
                    "related": "http://localhost:8000/api/v1/studies/65680ade-510c-
↪4437-a58a-e41d4b94d8ed/responses/"
                }
            }
        },
        "links": {
            "self": "http://localhost:8000/api/v1/studies/65680ade-510c-4437-a58a-
↪e41d4b94d8ed/"
        }
    }
}
```

Retrieving a Study's responses

GET /api/v1/studies/<study_id>/responses/

Permissions: Must be authenticated. Must have permission to view the responses to the particular study.

Creating a Study

POST /api/v1/studies/

METHOD NOT ALLOWED. Not permitted via the API.

Updating a Study

PUT /api/v1/studies/<study_id>/

METHOD NOT ALLOWED. Not permitted via the API.

Deleting a Study

DELETE /api/v1/studies/<study_id>/

METHOD NOT ALLOWED. Not permitted via the API.

Users

Viewing the list of users

GET /api/v1/users/

Permissions: Must be authenticated. You can view participants that have responded to studies you have permission to view, as well as own user information. Endpoint can return both participants and researchers, if you have permission to view them. Users with *can_read_all_user_data* permissions can view all active users in the database via this endpoint. Usernames are only shown if user has *can_read_usernames* permissions.

Sample Response:

```
{
    "links": {
        "first": "http://localhost:8000/api/v1/users/?page=1",
        "last": "http://localhost:8000/api/v1/users/?page=1",
        "next": null,
        "prev": null,
        "meta": {
            "page": 1,
            "pages": 1,
            "count": 1
        }
    },
    "data": [
        {
            "type": "users",
            "id": "834bbf33-b249-4737-a041-43574cd137a7",
            "attributes": {
                "given_name": "Test",
                "middle_name": "",
                "family_name": "User",
                "identicon": "data:image/png;base64,aaaabbbbccccddddeeeefffffgggg",
                "is_active": true,
                "is_staff": true
            },
            "relationships": {
                "demographics": {
                    "links": {
                        "related": "http://localhost:8000/api/v1/users/834bbf33-b249-
↪4737-a041-43574cd137a7/demographics/"
                    }
                },
                "children": {
                    "links": {
                        "related": "http://localhost:8000/api/v1/users/834bbf33-b249-
↪4737-a041-43574cd137a7/children/"
                    }
                }
            },
```

(continues on next page)

(continued from previous page)

```
            "links": {
                "self": "http://localhost:8000/api/v1/users/834bbf33-b249-4737-a041-
↪43574cd137a7/"
            }
        }
    ]
}
```

Retrieving a single user

GET /api/v1/users/<user_id>/

Permissions: Must be authenticated. You can view participants that have responded to studies you have permission to view, as well as own user information.

Sample Response:

```
{
    "data": {
        "type": "users",
        "id": "834bbf33-b249-4737-a041-43574cd137a7",
        "attributes": {
            "given_name": "Test",
            "middle_name": "",
            "family_name": "User",
            "identicon": "data:image/png;base64,aaaabbbbccccddddeeeeffffgggg",
            "is_active": true,
            "is_staff": true
        },
        "relationships": {
            "demographics": {
                "links": {
                    "related": "http://localhost:8000/api/v1/users/834bbf33-b249-4737-
↪a041-43574cd137a7/demographics/"
                }
            },
            "children": {
                "links": {
                    "related": "http://localhost:8000/api/v1/users/834bbf33-b249-4737-
↪a041-43574cd137a7/children/"
                }
            }
        },
        "links": {
            "self": "http://localhost:8000/api/v1/users/834bbf33-b249-4737-a041-
↪43574cd137a7/"
        }
    }
}
```

Creating a User

POST /api/v1/users/

METHOD NOT ALLOWED. Not permitted via the API.

Updating a User

PUT /api/v1/users/<user_id>/

METHOD NOT ALLOWED. Not permitted via the API.

Deleting a User

DELETE /api/v1/users/<user_id>/

METHOD NOT ALLOWED. Not permitted via the API.

1.22 1. Accessing Lookit and community

1.22.1 Welcome

Welcome to the Lookit tutorial! In this tutorial you will work through a series of exercises intended to build all the skills you'll need to run your own studies on Lookit. By the end of the tutorial, you will be able to create a participant-friendly study on Lookit that includes a survey and test trials, assign children to different experimental conditions, test and troubleshoot your study, download study data, contact participants, use and even edit this documentation!

This tutorial should take between 5 and 10 hours to complete, and does not assume any familiarity with programming. There is a mix of step-by-step directions and exercises to complete. It's the best way to get started using Lookit, and a good first task to give to an RA who will be designing studies. Note that it does **not** cover issues like participant recruitment and IRB approval, just the technical side of things.

Throughout the tutorial, you may want to have the tutorial open in one browser window while you complete the tasks in another window.

To get started, you need to get on Lookit!

1.22.2 Step 1: Join the Lookit Slack workspace

Fill out this form to join the Lookit Slack workspace. If you haven't used Slack before, you'll need to create an account; otherwise it will be added to your other workspaces.

Slack is a messaging platform that serves as the primary tool for communication about development and use of the Lookit platform. The Lookit Slack workspace is essentially a collection of "chat rooms" for various topics. You can use Slack via a web interface, a desktop app, and/or on your phone.

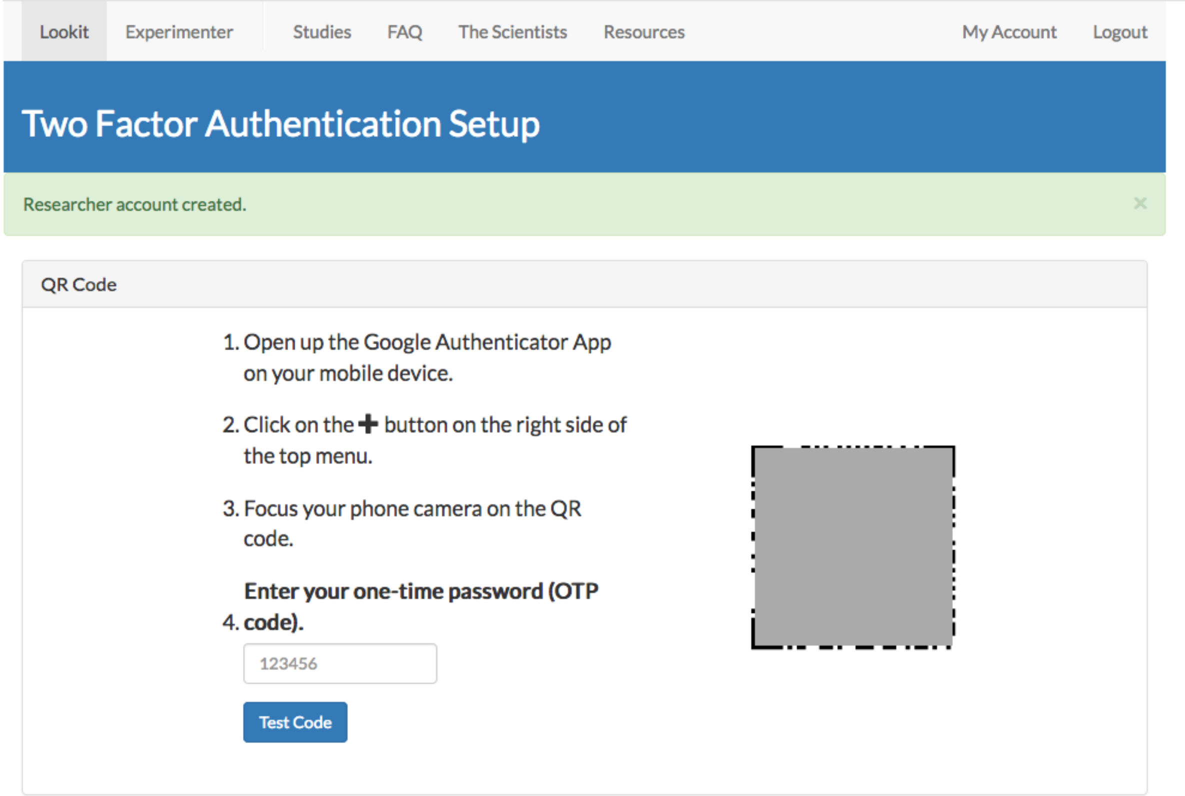
1.22.3 Step 2 (optional): Sign up for the lookit-research mailing list

We periodically (about once a month) send out updates about Lookit. If you'd prefer to get these updates via email instead of only on Slack, please sign up for this list.

1.22.4 Step 3: Create your Lookit account

1. Go to the researcher registration form. Fill out the form and click "create account."

2. You’ll be taken to a page like this to set up two-factor authentication (2FA), which you’ll need in order to access the researcher section of Lookit. If you haven’t already, download the Google Authenticator app on your phone. Then follow the directions on this page to activate 2FA for your account.

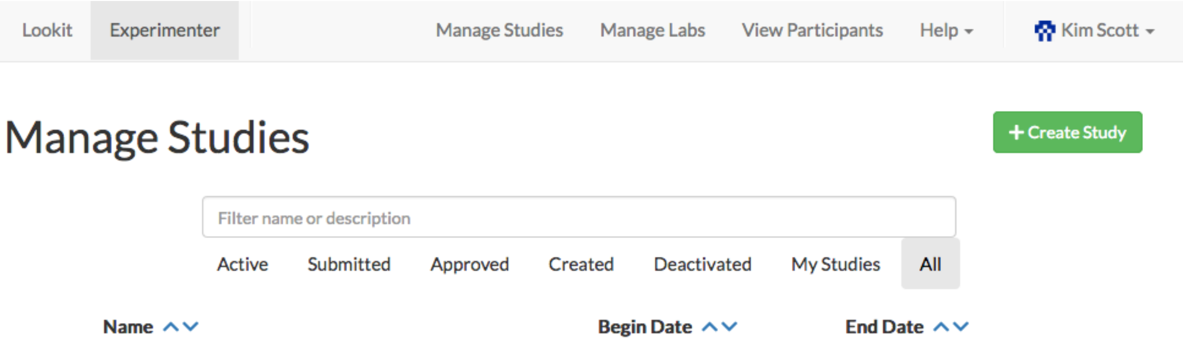


Two Factor Authentication Setup

Researcher account created.

1. Open up the Google Authenticator App on your mobile device.
2. Click on the ✚ button on the right side of the top menu.
3. Focus your phone camera on the QR code.
4. **Enter your one-time password (OTP code).**

3. You’re logged in! You should be redirected to a page of studies like this:



Manage Studies

(Having any trouble? Check the *login/registration documentation* for more information.)

1.22.5 Where to go for help

- If you need help as you’re completing this tutorial - or later when you’re creating and running studies on Lookit - the best place to go is the Lookit Slack workspace. Check the ‘researchers’ or ‘tech support’ channels to see if someone has asked the same question before! If not, post in one of these channels to get support from Lookit staff and/or other researchers who may have dealt with the same issue. (If you need help specifically from Lookit staff, feel free to add *@Kim* to your post to tag Kim, but under almost all circumstances the question should still go in one of the public channels rather than a private chat, so that other people can see the answers.)

Even if you're not totally stumped, it's worth checking in on Slack to get ideas and resources! For example, you might ask if anyone has an R script for analyzing their data that you could use as a starting point, or run a draft debriefing by other researchers to get advice on wording.

- There is a growing collection of training materials contributed by researchers that you can check out under the *"Other learning materials" section*!

1.22.6 Exercises

1. Say hello in the "tutorial" channel of the Lookit Slack workspace.

2. Get a feel for the participant experience by trying out one of the "template" studies! At https://lookit.mit.edu/exp/studies/, find a study labeled TEMPLATE from the lab "Demo lab." Click on it and then click "Preview study" to see how it would work if you were a participant! You may need to add a child profile and complete your demographic survey, which you can do from https://lookit.mit.edu/account/manage/.

3. [Extra credit] If you have young kids, take part in a currently-running study on the production server (https://lookit.mit.edu) - the experience is different with a squirming kid in your lap! Please enter only actual, not made-up, data on the production server - you are participating in real research here. (If you're curious but don't have a child in the age range, feel free to contact the lab and ask if it's ok to check it out–generally they can deal with that easily, but a heads up is helpful.)

1.23 2. Contributing to the codebase

Lookit is an open-source project, meaning that the code is freely available and anyone can contribute. This includes the actual Lookit platform, the components used in studies, AND the documentation you're reading now. Community contributions and input are critical to the project!

In this section, you'll make your first "Pull request" (PR) to propose a change to the documentation[1]. Your change will just be to add yourself to a list of tutorial participants. This way, you'll be comfortable making substantive changes whenever you're ready!

Later on in the tutorial, if you come across something confusing or you have an idea to improve the instructions or exercises, you can follow these same instructions to fix it. Although we recognize that most Lookit users will not be ready to contribute to the code that makes Lookit run, editing the documentation is something everyone can do! We need your help to keep it up-to-date and constantly improving!

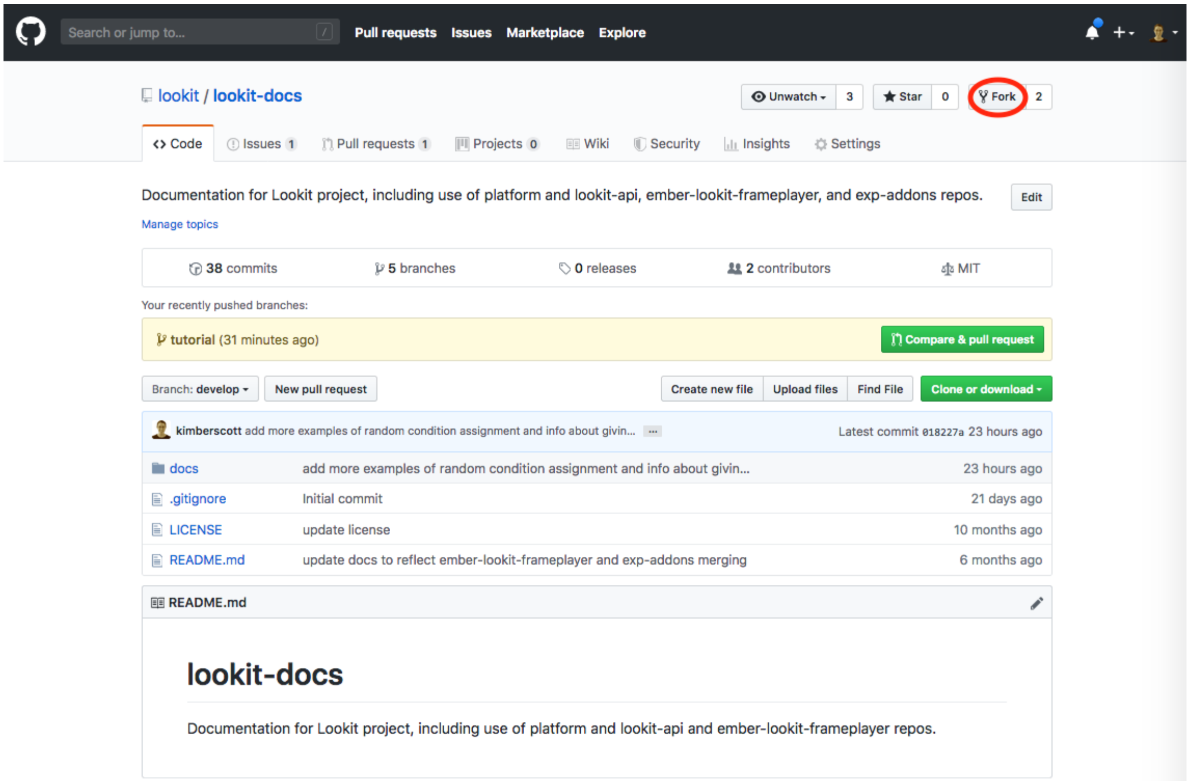
1.23.1 Making your first pull request (PR)

All of the code that makes Lookit run is stored on GitHub, which makes it easy to keep track of changes over time and merge changes from many collaborators. There are several code repositories or "repos" that house different pieces of the project. Here we are going to walk through making a change to the documentation repo, lookit-docs. Making a "pull request" or PR is a way to ask that your edits be incorporated into the main codebase. It's actually not too hard, and it's a great way to contribute!

Don't worry about making a mistake and somehow messing up the documentation - you can't directly edit the "official" version of the files! There is always a review process before your changes are merged in.

1. In a separate tab, go to https://github.com/lookit/lookit-docs and click "Fork" in the top right corner. You'll need to make an account on GitHub if you don't have one already. You only need to do this once - for future changes you can skip this step.

[1] This section, and the excellent idea to make "your first PR" an early and required step, is based on the OpenAPS documentation. Go help with their docs too. What? You didn't realize this tutorial was secretly just a way to get developmental psychologists working on open-source artificial pancreas systems?



Documentation for Lookit project, including use of platform and lookit-api, ember-lookit-frameplayer, and exp-addons repos.

Your recently pushed branches:

tutorial (31 minutes ago)

kimberscott add more examples of random condition assignment and info about givin... Latest commit 018227a 23 hours ago

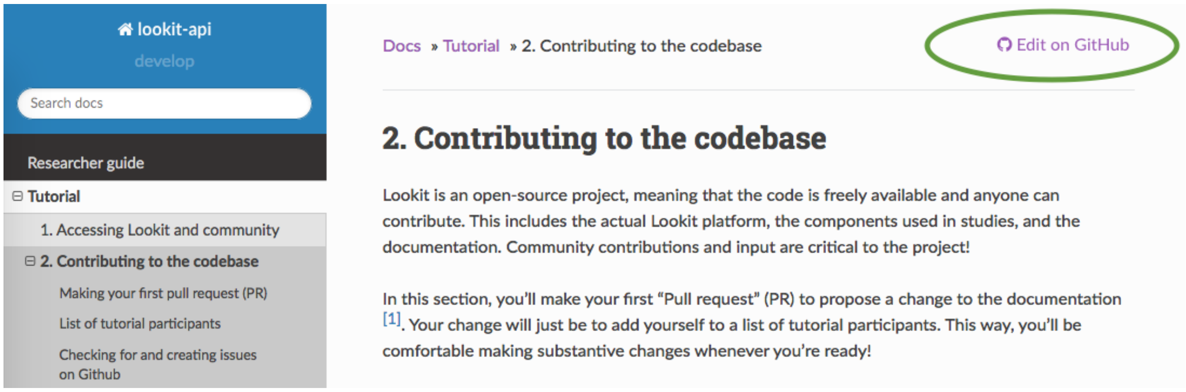
docs	add more examples of random condition assignment and info about givin...	23 hours ago
.gitignore	Initial commit	21 days ago
LICENSE	update license	10 months ago
README.md	update docs to reflect ember-lookit-frameplayer and exp-addons merging	6 months ago

README.md

lookit-docs

Documentation for Lookit project, including use of platform and lookit-api and ember-lookit-frameplayer repos.

2. From the page you are reading right now (or in the future, from whatever docs page you want to edit – *not* from GitHub!), scroll up and click the "Edit on GitHub" button at the top right. For this first PR, you'll be editing the page you're reading right now (https://lookit.readthedocs.io/en/develop/tutorial-contributing.html)!

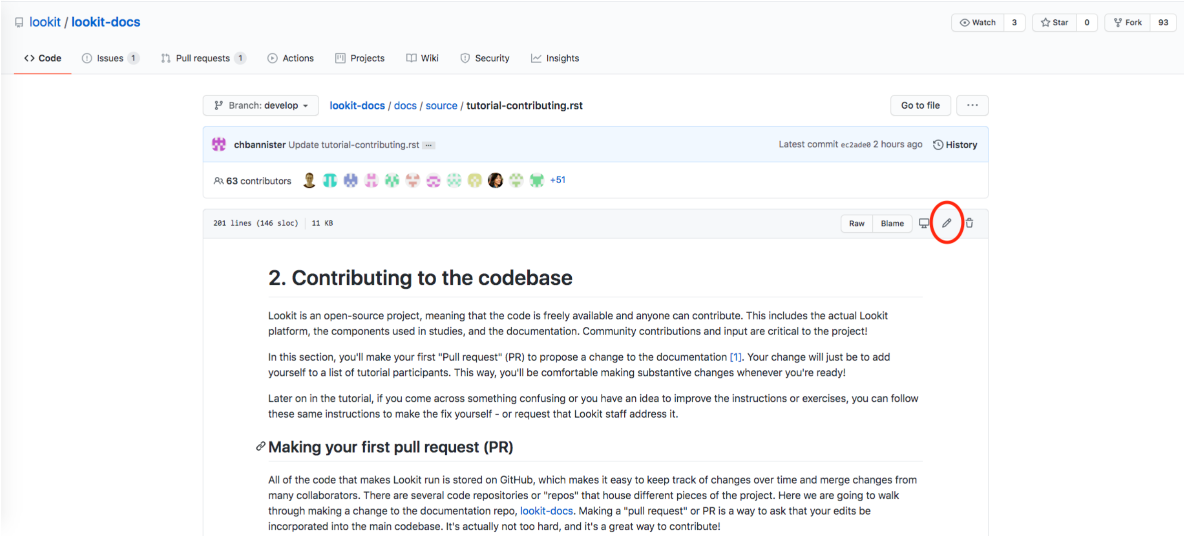

lookit-api
develop
Search docs
Researcher guide
⊟ Tutorial
1. Accessing Lookit and community
⊟ 2. Contributing to the codebase
Making your first pull request (PR)
List of tutorial participants
Checking for and creating issues on Github
Docs » Tutorial » 2. Contributing to the codebase
Edit on GitHub

2. Contributing to the codebase

Lookit is an open-source project, meaning that the code is freely available and anyone can contribute. This includes the actual Lookit platform, the components used in studies, and the documentation. Community contributions and input are critical to the project!

In this section, you'll make your first "Pull request" (PR) to propose a change to the documentation [1]. Your change will just be to add yourself to a list of tutorial participants. This way, you'll be comfortable making substantive changes whenever you're ready!

3. Clicking "Edit on GitHub" will bring you to a page like the one shown below. Click the pencil icon on this page to start editing the file.


lookit / lookit-docs
Watch 3
Star 0
Fork 93
Code
Issues 1
Pull requests 1
Actions
Projects
Wiki
Security
Insights
Branch: develop
lookit-docs / docs / source / tutorial-contributing.rst
Go to file
chbannister Update tutorial-contributing.rst
Latest commit ec2ade0 2 hours ago
History
63 contributors
+51
201 lines (146 sloc) | 11 KB
Raw
Blame

2. Contributing to the codebase

Lookit is an open-source project, meaning that the code is freely available and anyone can contribute. This includes the actual Lookit platform, the components used in studies, and the documentation. Community contributions and input are critical to the project!

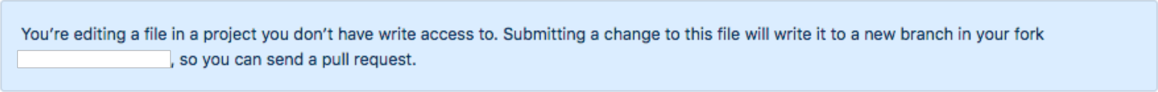
In this section, you'll make your first "Pull request" (PR) to propose a change to the documentation [1]. Your change will just be to add yourself to a list of tutorial participants. This way, you'll be comfortable making substantive changes whenever you're ready!

Later on in the tutorial, if you come across something confusing or you have an idea to improve the instructions or exercises, you can follow these same instructions to make the fix yourself - or request that Lookit staff address it.

Making your first pull request (PR)

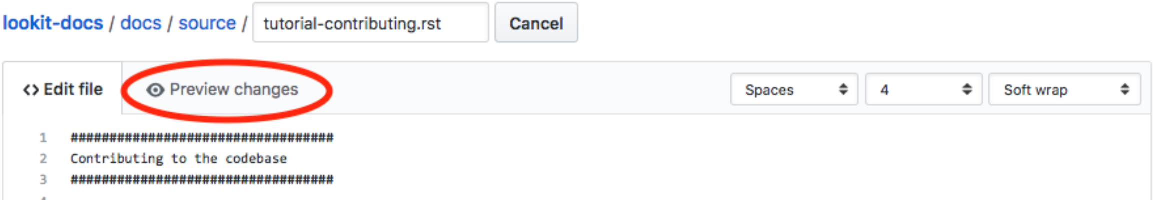
All of the code that makes Lookit run is stored on GitHub, which makes it easy to keep track of changes over time and merge changes from many collaborators. There are several code repositories or "repos" that house different pieces of the project. Here we are going to walk through making a change to the documentation repo, lookit-docs. Making a "pull request" or PR is a way to ask that your edits be incorporated into the main codebase. It's actually not too hard, and it's a great way to contribute!

You may see a message that submitting a change will write it to a new branch in your fork. That's ok!



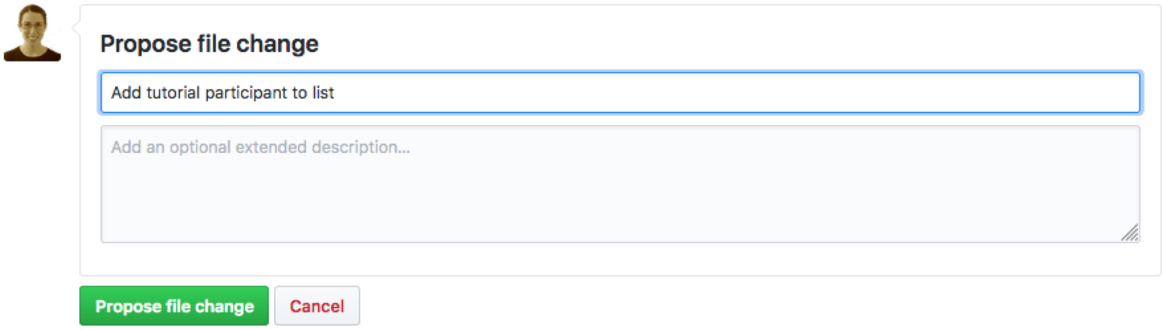
You're editing a file in a project you don't have write access to. Submitting a change to this file will write it to a new branch in your fork , so you can send a pull request.

4. Make your changes! For this PR, just add your name and institution to the list of tutorial participants. Click the "Preview changes" tab to make sure everything looks the way you want it to. You can go back and forth between the "Edit file" and "Preview changes" tabs as you make more involved changes.



```
###################################
Contributing to the codebase
###################################
```

5. Scroll down to "Propose file change." Enter a short description of your change, and then click the green "Propose file change" button:

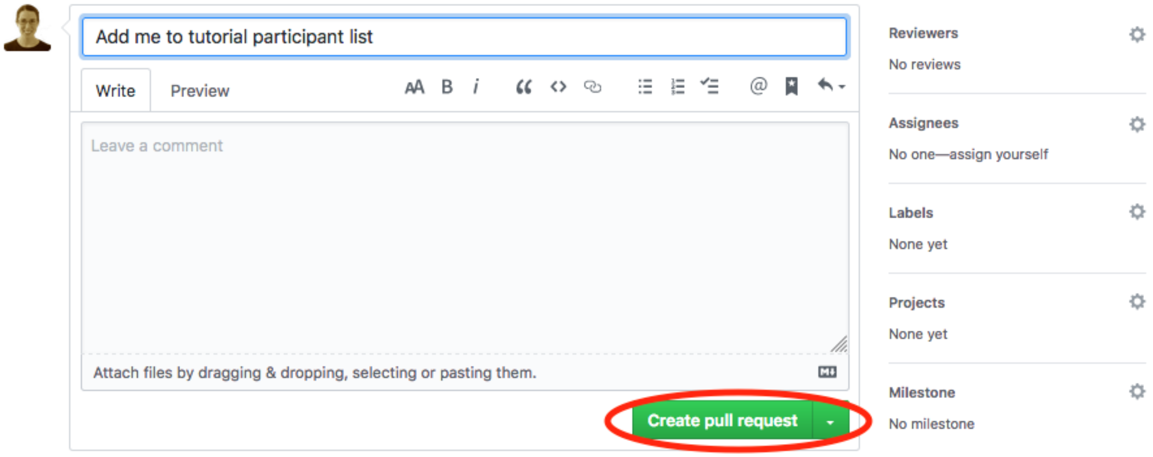


Propose file change

Add tutorial participant to list

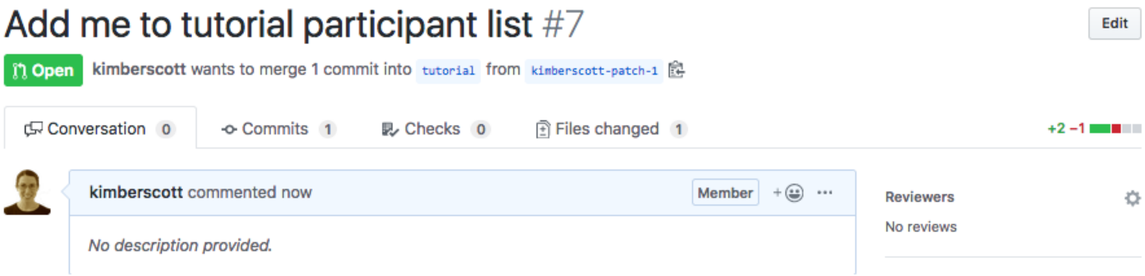
Add an optional extended description...

6. Click the green "Create pull request" button on the page that appears next:



Add me to tutorial participant list

7. Hooray! You've created your first PR. You should now be at a page where it's been given a number:



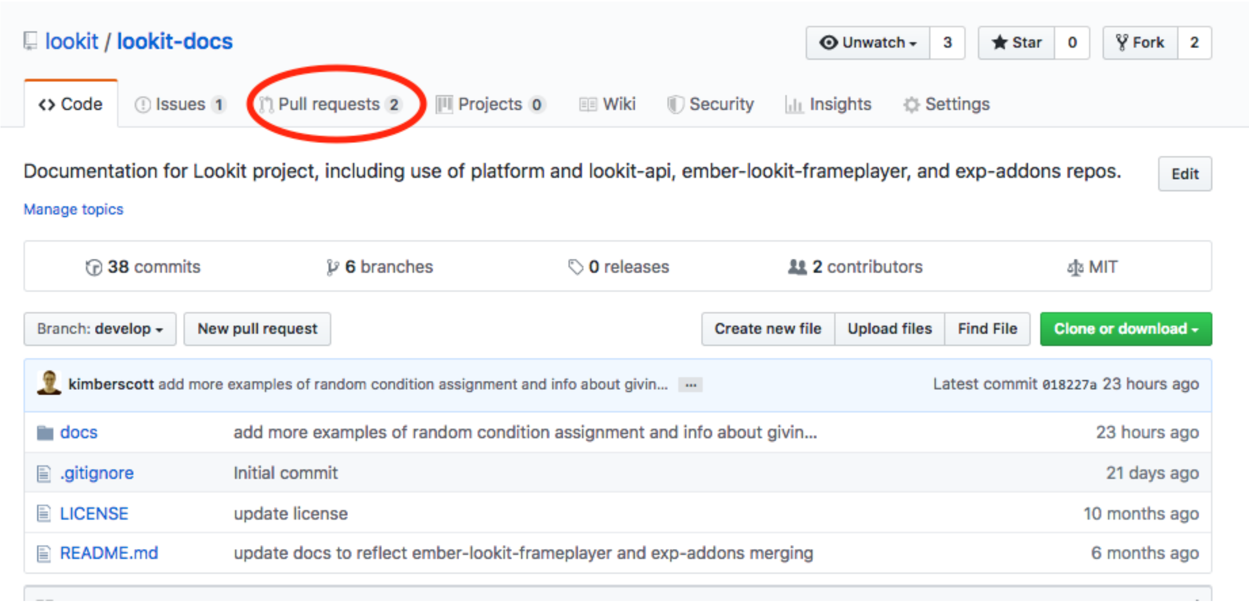
Add me to tutorial participant list #7

Open kimberscott wants to merge 1 commit into `tutorial` from `kimberscott-patch-1`

kimberscott commented now

No description provided.

The PR will now be in a list for Lookit staff to review. You can return to this page to check on it; if you have allowed Github to send notifications via email (the default), you will also get an email about any activity. You can also see your PR in the list by going to https://github.com/lookit/lookit-docs and clicking on "Pull requests":



lookit / lookit-docs

Documentation for Lookit project, including use of platform and lookit-api, ember-lookit-frameplayer, and exp-addons repos.

38 commits · 6 branches · 0 releases · 2 contributors · MIT

Branch: develop

kimberscott add more examples of random condition assignment and info about givin… Latest commit 018227a 23 hours ago

docs	add more examples of random condition assignment and info about givin…	23 hours ago
.gitignore	Initial commit	21 days ago
LICENSE	update license	10 months ago
README.md	update docs to reflect ember-lookit-frameplayer and exp-addons merging	6 months ago

This process is the easiest way to make a change to a particular file in the documentation. If you want to edit multiple files, add new files, and/or reorganize things, you will probably want to try out your changes locally before submitting your PR. You can learn more in the *section on Contributor Guidelines*. You don't need to know how to do that for the purposes of this tutorial, though - just know it's *possible* to deal with multiple files at once.

1.23.2 List of tutorial participants

- Kim Scott (MIT)
- Ashley Thomas (MIT)
- James Dunlea (Columbia University)
- Katharine Scott (University of Wisconsin-Madison)
- Meltem Yucel (University of Virginia)
- Francis Yuen (University of British Columbia)
- Rachel Dudley (Central European University)
- Lisa Renaud (Northwestern University)
- Hannah Cho (University of Toronto)
- Trisha Katz (University of California-San Diego)
- Caren Walker (University of California-San Diego)
- Elizabeth Lapidow (UCSD)
- Hyesung Grace Hwang (University of Chicago)
- Marc Colomer (University of Chicago)
- Junyi Chu (MIT)
- Maddie Pelz (MIT)
- Heather Kosakowski (MIT)
- Emily Neer (UCLA)
- Amalia Ionescu (UCLA)
- Katlyn Newman (UCSD)
- Virginia Rosenberger (MIT)
- Katarina Begus (Rutgers-Newark)
- Clement Choi (University of British Columbia)
- William Adams (University of Bath)
- Théo Morfoisse (NYU)
- Noa (UCSB)
- Kelly Kendro (UC San Diego)
- Katherine Casey (American University)
- Eren Fukuda (University of Wisconsin-Madison)
- Bailey Immel (UW-Madison)
- Annie Harris (Harvard University Extension School)
- Shari Liu (Harvard)
- Brandon Woo (Harvard)
- Caitlin Fausey (University of Oregon)
- Kate Bee (University of Oregon)

- Jasmine Engen (University of Oregon)
- Ellie McLoughlin (University of Oregon)
- Allyson Kuznia (University of Oregon)
- Haley Weaver (University of Wisconsin-Madison)
- Annika Voss (UC Davis)
- Aaron Beckner (University of California, Davis)
- Amanda Rose Yuile (UIUC)
- Lisa Oakes (UC Davis)
- Gerwin Legaspi (University of British Columbia)
- Amanda Cramer (University of Texas at Austin)
- Rebecca Houston-Read (Harvard University)
- Nicki Zieber (University of Kansas)
- Brooke Diviak (New York University)
- Charles Murray (Stanford University)
- Jing Shen (University of Wisconsin-Madison)
- Stephanie Chang (Stanford University)
- Daniela Bencid (Colby College)
- Adena Schachner (University of California, San Diego)
- Samia Razvi (UT Dallas)
- Nicoke Cuneo (Haskins Laboratories)
- Kristine Hocker (MIT)
- Estelle Hervé (AMU)
- Isabel Nichoson (Wellesley College)
- Ginni Strehle (UT Dallas)
- Gala Stojnić (NYU)
- Mark Sheskin (Minerva Schools at KGI)
- Eylem Altuntas (MARCS at WSU)
- Catherine T Best (MARCS Institute, Western Sydney University, Australia)
- Xi Jia Zhou (Stanford)
- Brendan Hancock (Queen’s University)
- Carlin Bannister (University of British Columbia)
- Allena McComas (University of California, San Diego)
- Talia Papa (University of British Columbia)
- Emily Marks (Uniersity of British Columbia)
- Toby Mintz (University of Southern California)
- Jazlyn Armendariz (California State University, Northridge)

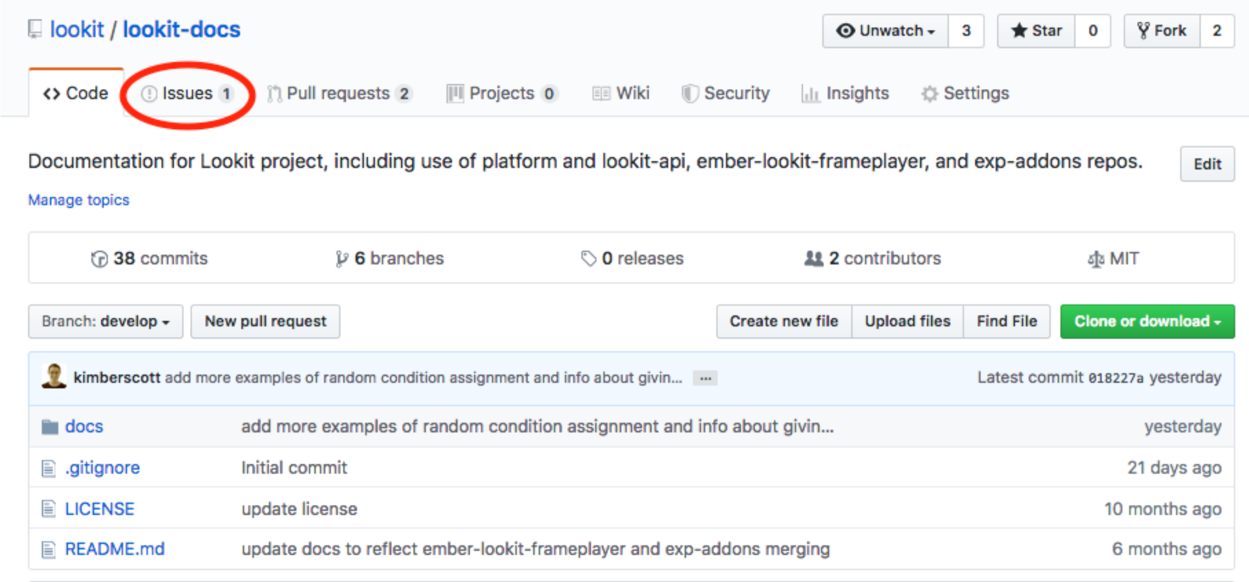
- Melissa Santos (Stanford University)
- Karen Smith (University of Wisconsin-Madison)
- Lillian Xu (University of Wisconsin-Madison)
- Ariel Starr (University of Washington)
- Taylor Petersen (University of Washington)
- Isabella Duan (Stanford University)
- Aarthi Popat (Stanford University)
- Jamie Jirout (University of Virginia)
- Cynthia Lukyanenko (George Mason University)
- Franchesca Quintero (University of California, Davis)
- Madison Buntrock (University of Maryland, College Park)
- Katie Schuler (University of Pennsylvania)
- Emily Fourie (University of California, Davis)
- Sarra Al-Zayer (Cornell University)
- Mary Eng (Cornell University)
- Elizabeth Swanson (Stanford University)
- Peppy Winchel (University of Virginia)
- Kaitlin Lawler (University of Texas at Dallas)
- Shoronda Matthews (University of Virginia)
- Yi Lin (New York University)
- Ariel Mathis (University of Pennsylvania)
- Sav Nijeboer (University of British Columbia
- Michelle Miller (University of Virginia)
- Iris Zhong (Smith College)
- Vanessa Mak (University of British Columbia)
- Sarvenaz Oloomi (University of British Columbia)
- Ania Alberski (University of Pennsylvania)
- Jenna Croteau (Smith College)
- Madison Chew (University of California, San Diego)
- Claudia Lam (University of British Columbia)
- Stephanie De Anda (University of Oregon)
- Mariam Habib (Rutgers University)
- Rosalva Mejia (University of California, Los Angeles)
- Jinyoung Jo (University of California, Los Angeles)
- Anika Brahmbhatt (Boston University)
- Hironori Katsuda (University of California, Los Angeles)

- Canaan Breiss (University of California, Los Angeles)
- Sarah Kang (University of California, San Diego)
- Megan Hoffman (University of California, San Diego)
- Sivan Barashy (University of California, San Diego)
- Stacee Santos (Boston College)
- Chippy Banarjee (Yale University)
- Alyssa Nguyen (University of Oregon)
- Erika Parisien (University of Oregon)
- Zoya Egiazaryan (University of California, Los Angeles)
- Sara Marshall (McMaster University)
- Sho Tsuji (The University of Tokyo)
- Catherine Bianco (Columbia University)
- Simran Mahajan (New York University)
- Valeria Hernández (New York University)
- Amanda Maniscalco (New York University)
- Alice Wang (Haskins Laboratories)
- Jamie Kang (University of Virginia)
- Andrea Stein (University of Wisconsin-Madison)
- Haykaz Mangardich (University of British Columbia)
- Justine Wang (University of California, San Diego)
- Kayla Good (Stanford University)
- Dimitri Prica (University of Barcelona)
- Victor Manea (University of California, San Diego)
- Candice Rubie (University of Waterloo)
- Abbey Ward (University of Oregon)
- Connor Cook (Wingate University)
- Erica Verde (University of California, Davis)
- Joseph Lang (Wingate University)
- Heather Morse (Wingate University)
- Carrie Watson (University of Southern California)
- Yiran Chen (University of Pennsylvania)
- Erica Wojcik (Skidmore College)
- Stacy Wang (University of British Columbia)
- Victor Antoine (École Normale Supérieure, Paris)
- Tiffany Widjaja (University of California, San Diego)
- Christopher J. Green (MIT)

- Jacob Guerrero (University of California, San Diego)
- Gal Raz (MIT)
- Angela Oku (University of California, San Diego)
- Hannah Ruebeck (MIT)

1.23.3 Checking for and creating issues on Github

What if you notice a problem while using Lookit, or something unclear in the documentation, but it's not something you know how to fix? Or what if you find yourself wishing there were a particular feature that would make your research easier?



Documentation for Lookit project, including use of platform and lookit-api, ember-lookit-frameplayer, and exp-addons repos.

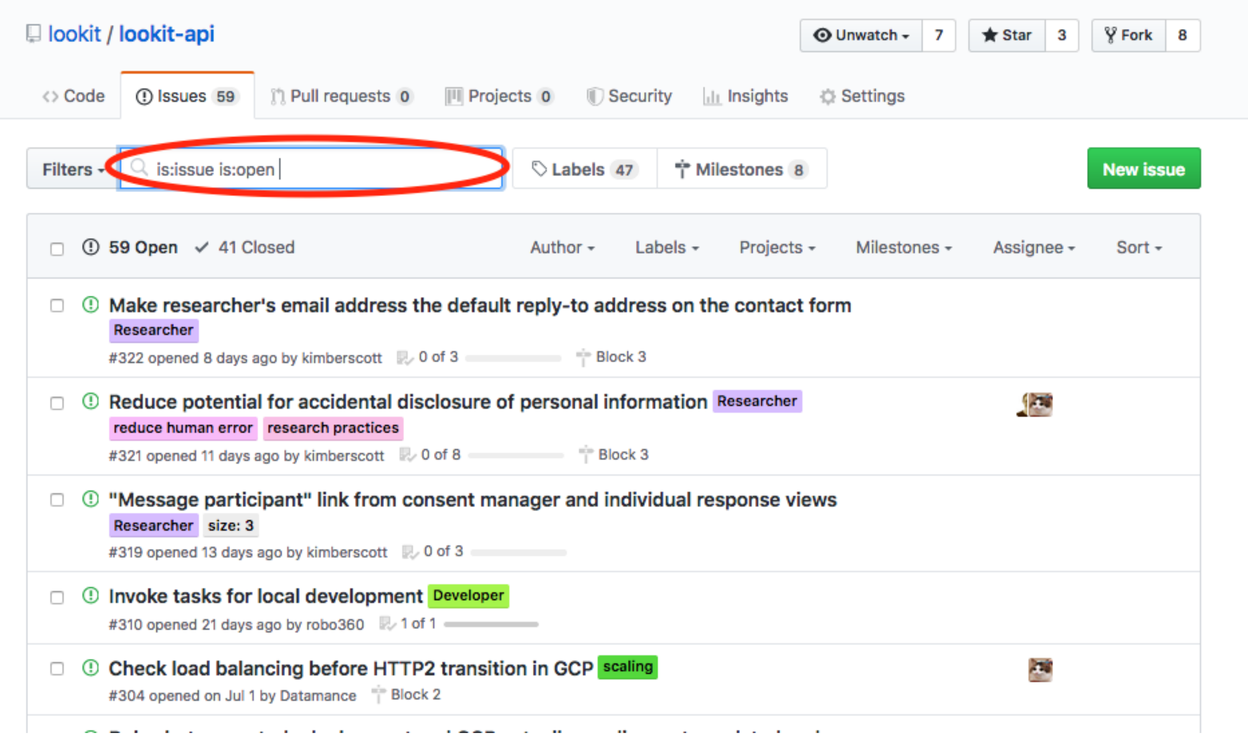
kimberscott add more examples of random condition assignment and info about givin... Latest commit 018227a yesterday

docs	add more examples of random condition assignment and info about givin...	yesterday
.gitignore	Initial commit	21 days ago
LICENSE	update license	10 months ago
README.md	update docs to reflect ember-lookit-frameplayer and exp-addons merging	6 months ago

To track bug reports and feature requests, we use GitHub **issues**. You can see issues by clicking on the "Issues" tab in the appropriate repository or "repo":

- lookit-api is the repo for the Lookit site: issues with anything to do with participant login or data, how current and past studies are displayed to participants, how you view data and manage your studies
- ember-lookit-frameplayer is the repo for the experiment components themselves: issues with how particular frames behave, frames you'd find useful, counterbalancing/condition assignment, etc.
- lookit-docs is the repo for the documentation: anything about the docs you're reading now!

To request a feature or report a bug, first search the existing issues to see if your idea is already there.



lookit / lookit-api

Unwatch 7 | Star 3 | Fork 8

Code | Issues 59 | Pull requests 0 | Projects 0 | Security | Insights | Settings

Filters | is:issue is:open | Labels 47 | Milestones 8 | New issue

□ 59 Open ✓ 41 Closed | Author | Labels | Projects | Milestones | Assignee | Sort

□ Make researcher's email address the default reply-to address on the contact form Researcher
#322 opened 8 days ago by kimberscott 0 of 3 Block 3

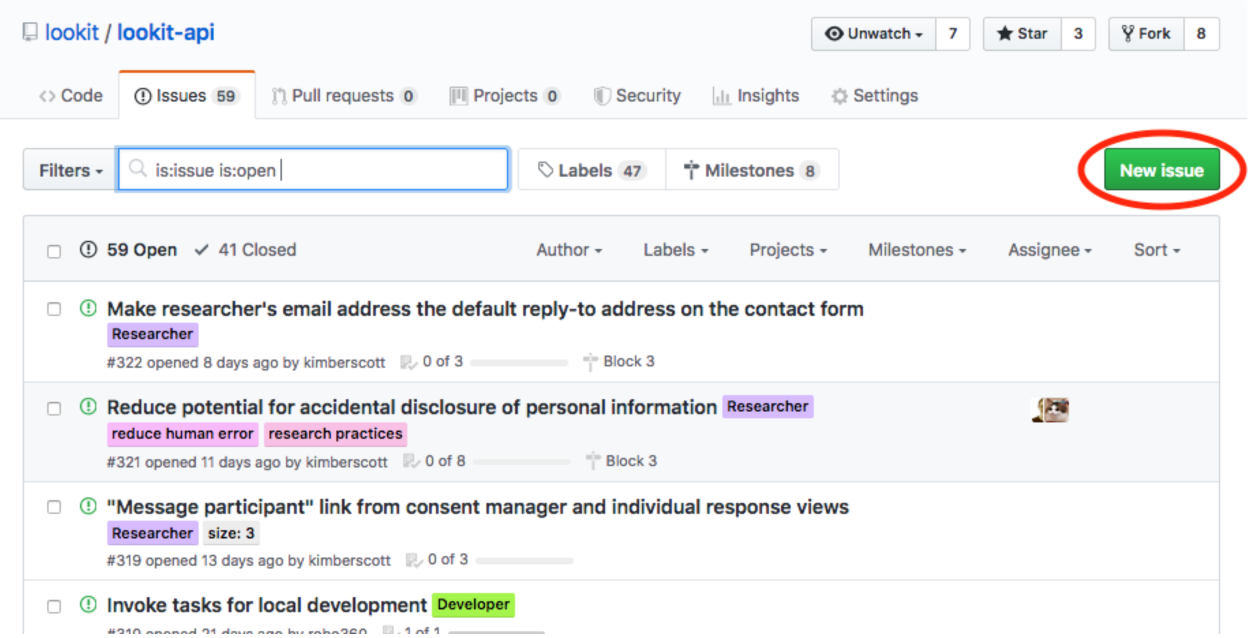
□ Reduce potential for accidental disclosure of personal information Researcher reduce human error research practices
#321 opened 11 days ago by kimberscott 0 of 8 Block 3

□ "Message participant" link from consent manager and individual response views Researcher size: 3
#319 opened 13 days ago by kimberscott 0 of 3

□ Invoke tasks for local development Developer
#310 opened 21 days ago by robo360 1 of 1

□ Check load balancing before HTTP2 transition in GCP scaling
#304 opened on Jul 1 by Datamance Block 2

If so, comment on it or add a thumbs-up reaction so Lookit staff know there's more interest! If not, click the green "New issue" button at the top right.



lookit / lookit-api

Unwatch 7 | Star 3 | Fork 8

Code | Issues 59 | Pull requests 0 | Projects 0 | Security | Insights | Settings

Filters | is:issue is:open | Labels 47 | Milestones 8 | New issue

□ 59 Open ✓ 41 Closed | Author | Labels | Projects | Milestones | Assignee | Sort

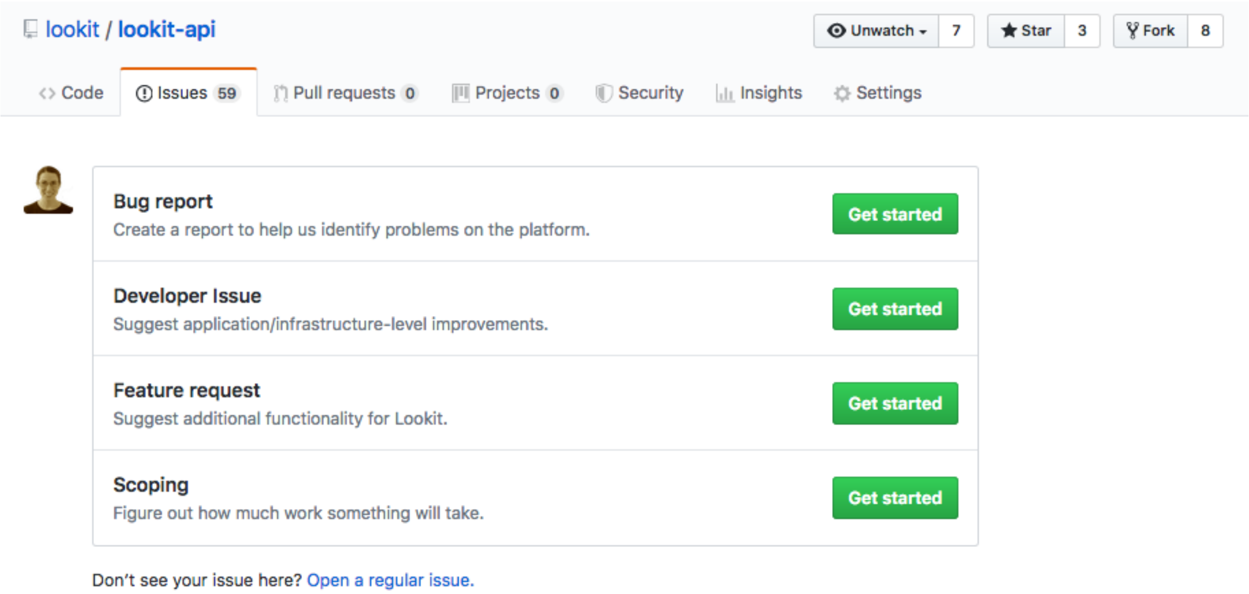
□ Make researcher's email address the default reply-to address on the contact form Researcher
#322 opened 8 days ago by kimberscott 0 of 3 Block 3

□ Reduce potential for accidental disclosure of personal information Researcher reduce human error research practices
#321 opened 11 days ago by kimberscott 0 of 8 Block 3

□ "Message participant" link from consent manager and individual response views Researcher size: 3
#319 opened 13 days ago by kimberscott 0 of 3

□ Invoke tasks for local development Developer

You will need to select an issue type. Choose the type that's closest to what you want to describe - probably "bug report" or "feature request":



Bug report
Create a report to help us identify problems on the platform.

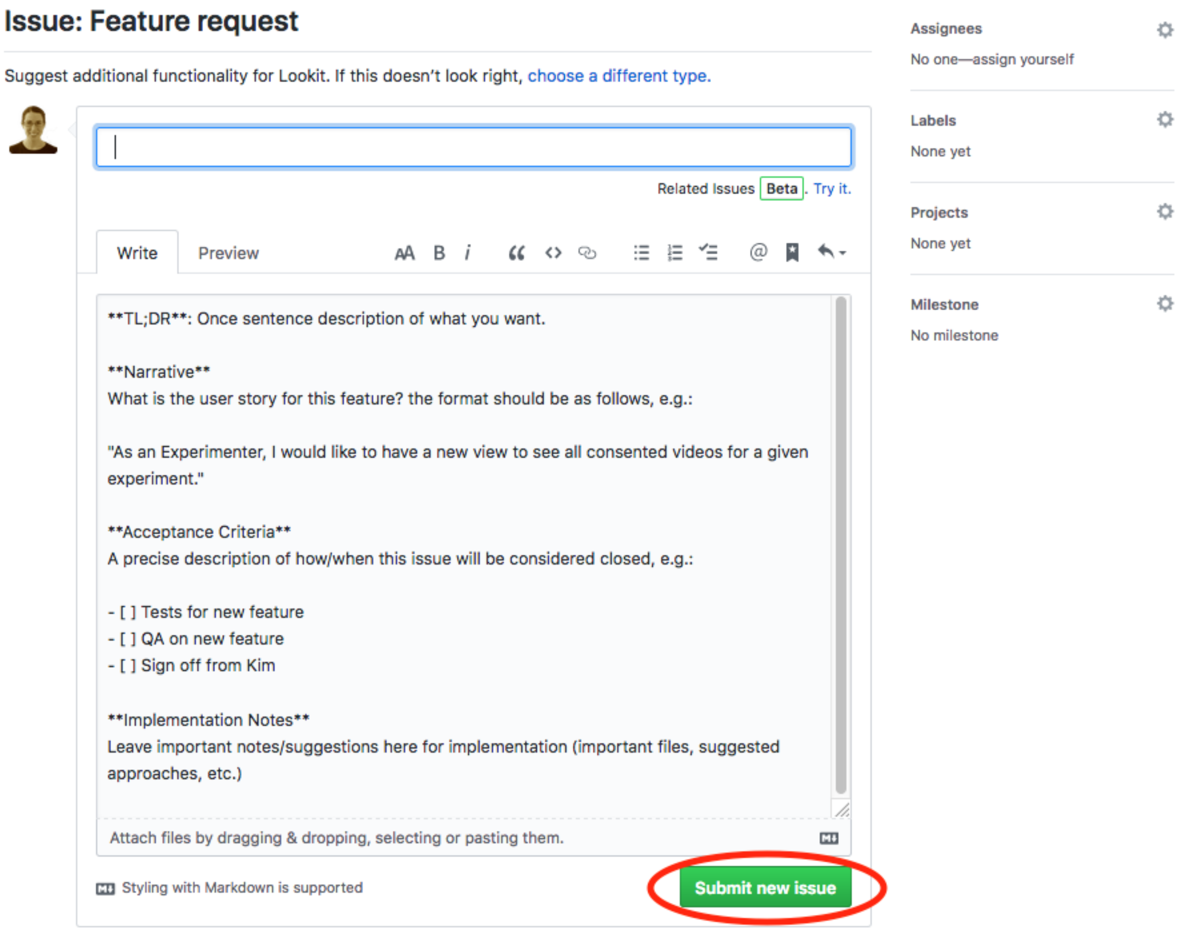
Developer Issue
Suggest application/infrastructure-level improvements.

Feature request
Suggest additional functionality for Lookit.

Scoping
Figure out how much work something will take.

Don't see your issue here? Open a regular issue.

If you had to select an issue type, you'll now have a template to fill in with information. If you're not using a template, try to give a clear one-sentence summary of the problem or requested feature/change, followed by any details needed to reproduce the problem or understand the proposed change. Then click the green "Submit new issue" button to create your issue.

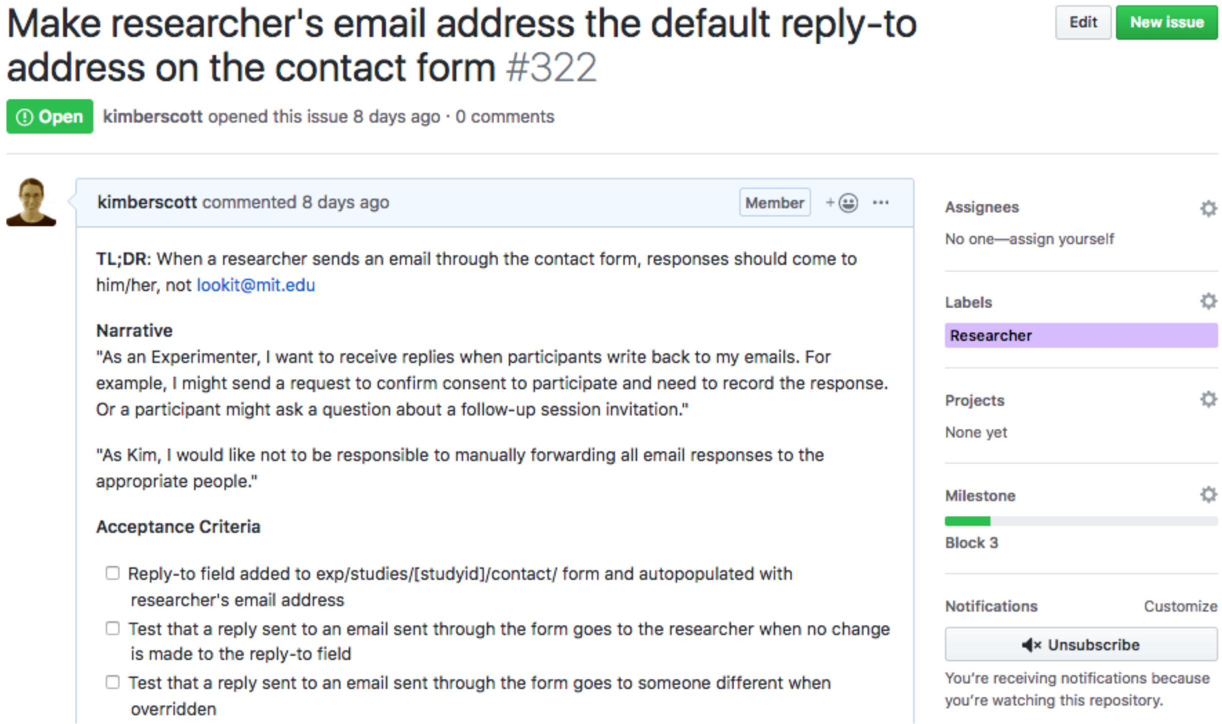


Issue: Feature request

Suggest additional functionality for Lookit. If this doesn't look right, choose a different type.

```
**TL;DR**: Once sentence description of what you want.

**Narrative**
What is the user story for this feature? the format should be as follows, e.g.:

"As an Experimenter, I would like to have a new view to see all consented videos for a given
experiment."

**Acceptance Criteria**
A precise description of how/when this issue will be considered closed, e.g.:

- [ ] Tests for new feature
- [ ] QA on new feature
- [ ] Sign off from Kim

**Implementation Notes**
Leave important notes/suggestions here for implementation (important files, suggested
approaches, etc.)
```

Your issue will now have a number assigned to it and will be listed in the issue list you looked at earlier:



Make researcher's email address the default reply-to address on the contact form #322

Open kimberscott opened this issue 8 days ago · 0 comments

kimberscott commented 8 days ago

TL;DR: When a researcher sends an email through the contact form, responses should come to him/her, not lookit@mit.edu

Narrative

"As an Experimenter, I want to receive replies when participants write back to my emails. For example, I might send a request to confirm consent to participate and need to record the response. Or a participant might ask a question about a follow-up session invitation."

"As Kim, I would like not to be responsible to manually forwarding all email responses to the appropriate people."

Acceptance Criteria

- □ Reply-to field added to exp/studies/[studyid]/contact/ form and autopopulated with researcher's email address
- □ Test that a reply sent to an email sent through the form goes to the researcher when no change is made to the reply-to field
- □ Test that a reply sent to an email sent through the form goes to someone different when overridden

Assignees: No one—assign yourself

Labels: Researcher

Projects: None yet

Milestone: Block 3

Notifications: Customize

You're receiving notifications because you're watching this repository.

Lookit staff may respond to ask for further information, schedule it for future development, and/or wait for community feedback about the idea to gauge demand.

1.23.4 Exercises

1. Suppose you would like to be able to download a file with scrambled or random data of the same form as your actual data, so that you could get your analysis scripts working without contaminating your real dataset. Which GitHub repo should you create an issue in?
2. Suppose you would like to be able to provide a study in the appropriate language for a given participant. Is there a Github issue in the lookit-api repo that addresses this?

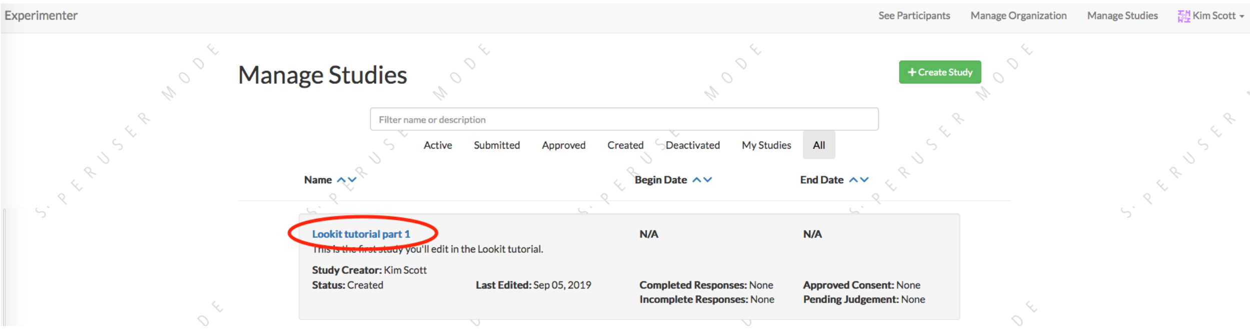
1.24 3. Setting up your first study

In this section, you will be creating your first study on Lookit. You will learn how to find and use experiment components, specify your protocol, and test out and troubleshoot your study.

1.24.1 Step 1: Clone the 'Lookit tutorial part 1' study

To get started, log in to Lookit as an experimenter. (Go to https://lookit.mit.edu/login/ to log in.)

You should see a few studies you automatically have access to, including one called "Lookit tutorial part 1". Click on that to open up the study detail page:

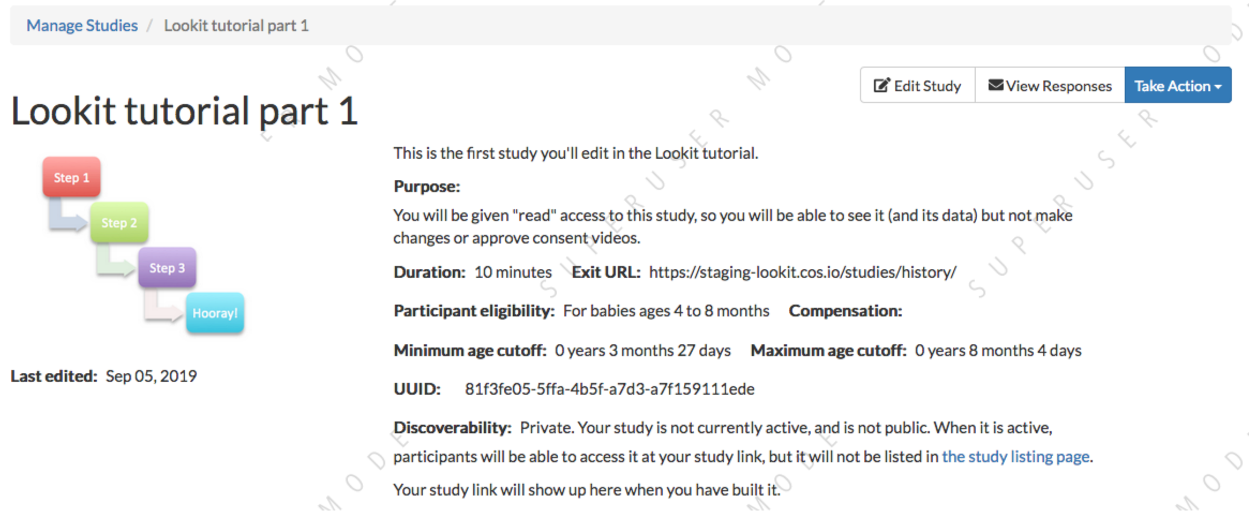


Manage Studies

Name	Begin Date	End Date
Lookit tutorial part 1 This is the first study you'll edit in the Lookit tutorial.	N/A	N/A

Study Creator: Kim Scott
Status: Created
Last Edited: Sep 05, 2019
Completed Responses: None
Incomplete Responses: None
Approved Consent: None
Pending Judgement: None

You should see something like this:



Lookit tutorial part 1

Last edited: Sep 05, 2019

This is the first study you'll edit in the Lookit tutorial.

Purpose:

You will be given "read" access to this study, so you will be able to see it (and its data) but not make changes or approve consent videos.

Duration: 10 minutes **Exit URL:** https://staging-lookit.cos.io/studies/history/

Participant eligibility: For babies ages 4 to 8 months **Compensation:**

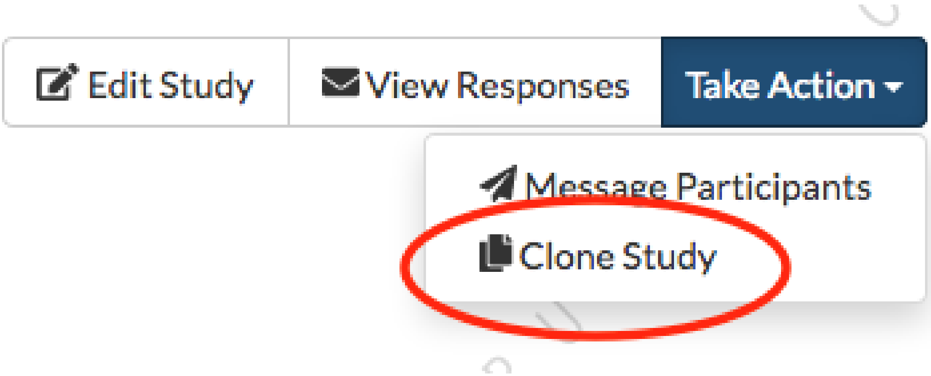
Minimum age cutoff: 0 years 3 months 27 days **Maximum age cutoff:** 0 years 8 months 4 days

UUID: 81f3fe05-5ffa-4b5f-a7d3-a7f159111ede

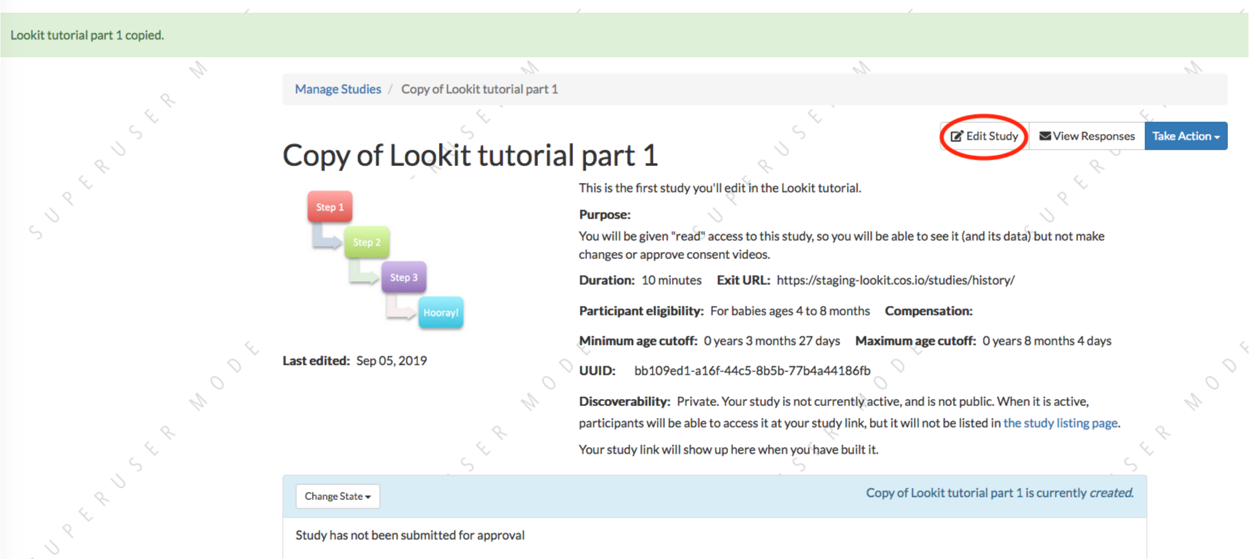
Discoverability: Private. Your study is not currently active, and is not public. When it is active, participants will be able to access it at your study link, but it will not be listed in the study listing page.

Your study link will show up here when you have built it.

At the top right, go to "Take Action" and click "Clone Study":



You will be taken directly to the "clone" or "copy" of your study, which will be named something like "Lookit tutorial part 1 copy." You should see something like this:



Lookit tutorial part 1 copied.

Manage Studies / Copy of Lookit tutorial part 1

Copy of Lookit tutorial part 1

Last edited: Sep 05, 2019

This is the first study you'll edit in the Lookit tutorial.

Purpose:

You will be given "read" access to this study, so you will be able to see it (and its data) but not make changes or approve consent videos.

Duration: 10 minutes **Exit URL:** https://staging-lookit.cos.io/studies/history/

Participant eligibility: For babies ages 4 to 8 months **Compensation:**

Minimum age cutoff: 0 years 3 months 27 days **Maximum age cutoff:** 0 years 8 months 4 days

UUID: bb109ed1-a16f-44c5-8b5b-77b4a44186fb

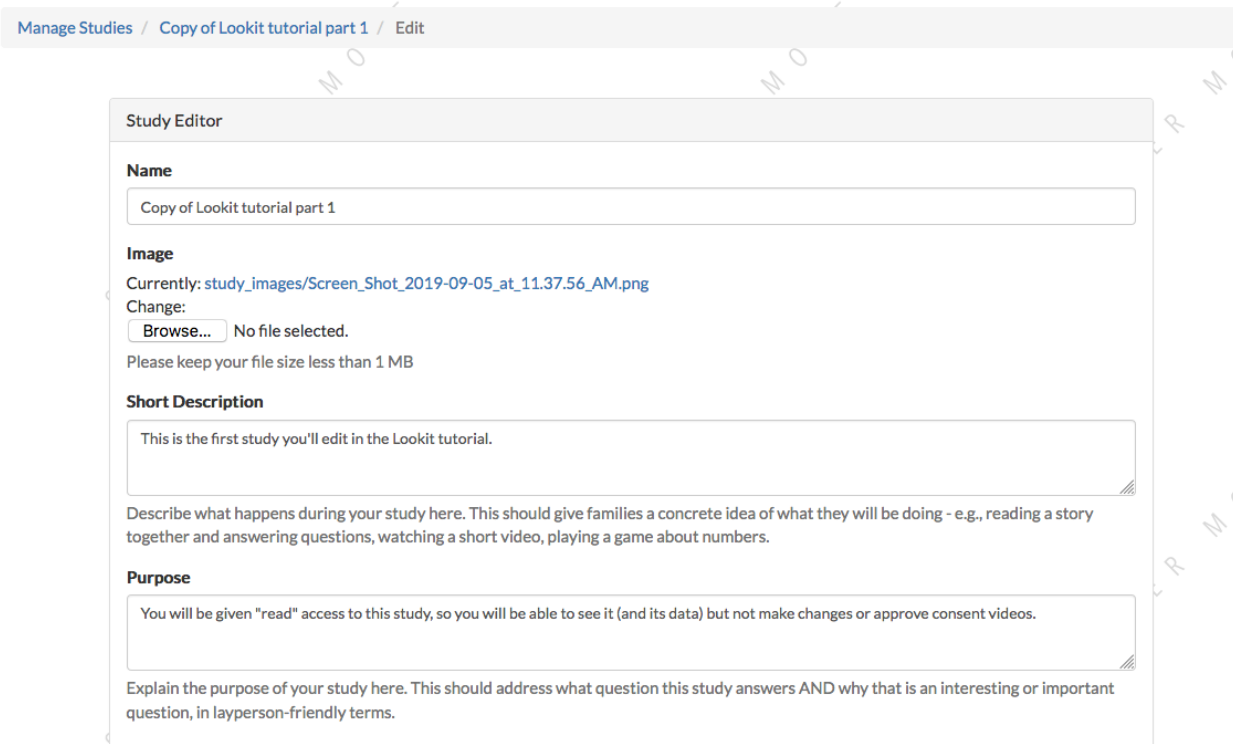
Discoverability: Private. Your study is not currently active, and is not public. When it is active, participants will be able to access it at your study link, but it will not be listed in the study listing page.

Your study link will show up here when you have built it.

Copy of Lookit tutorial part 1 is currently *created.*

Study has not been submitted for approval

Click "edit study" in the top right (circled above) and you'll see the following:



Manage Studies / Copy of Lookit tutorial part 1 / Edit

Study Editor

Name

Copy of Lookit tutorial part 1

Image

Currently: study_images/Screen_Shot_2019-09-05_at_11.37.56_AM.png

Change:

Browse... No file selected.

Please keep your file size less than 1 MB

Short Description

This is the first study you'll edit in the Lookit tutorial.

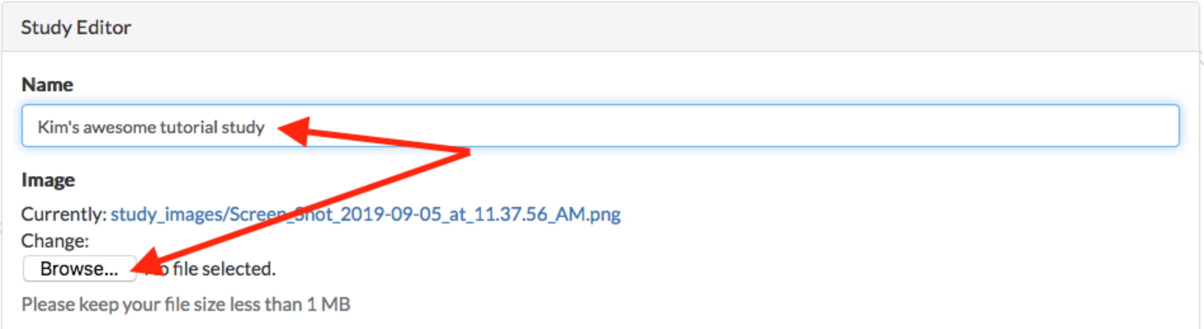
Describe what happens during your study here. This should give families a concrete idea of what they will be doing - e.g., reading a story together and answering questions, watching a short video, playing a game about numbers.

Purpose

You will be given "read" access to this study, so you will be able to see it (and its data) but not make changes or approve consent videos.

Explain the purpose of your study here. This should address what question this study answers AND why that is an interesting or important question, in layperson-friendly terms.

Each of these fields is described *here*. For now, we'll just change the name of the study and the thumbnail image that's displayed to participants. Rename your study to "[Your name]'s awesome tutorial study" and upload a different thumbnail image:



Study Editor

Name

Kim's awesome tutorial study

Image

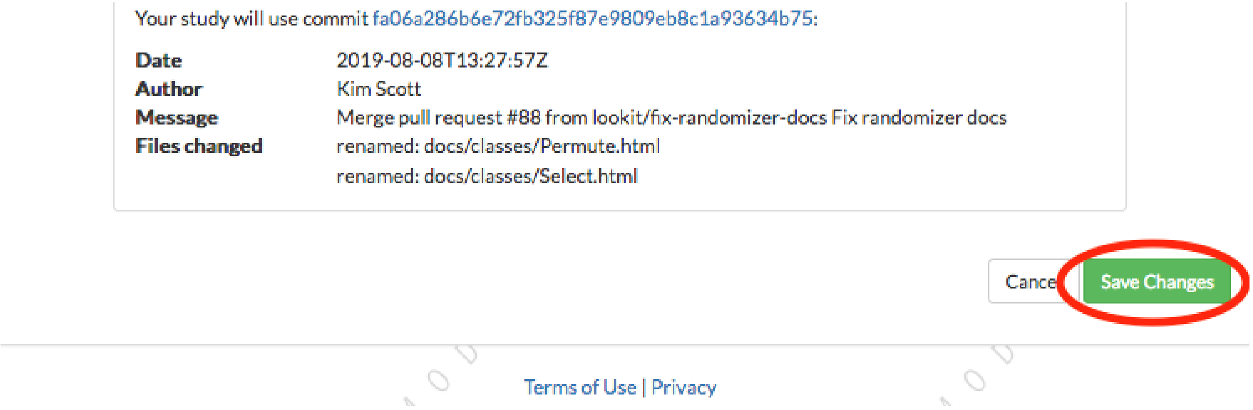
Currently: study_images/Screen_Shot_2019-09-05_at_11.37.56_AM.png

Change:

No file selected.

Please keep your file size less than 1 MB

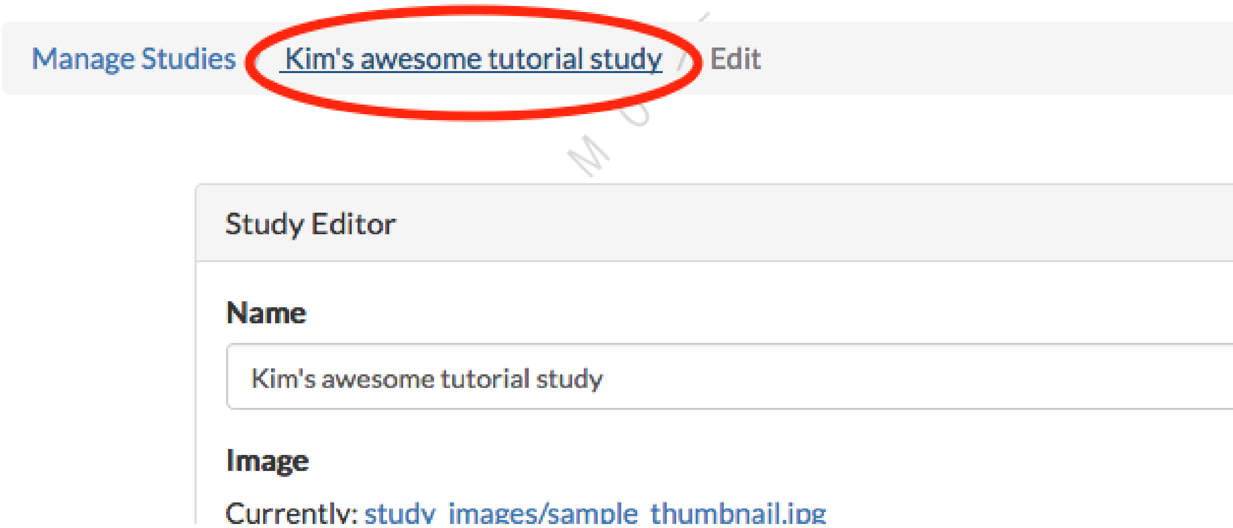
Then click "Save Changes" down below:



Your study will use commit fa06a286b6e72fb325f87e9809eb8c1a93634b75:

Date	2019-08-08T13:27:57Z
Author	Kim Scott
Message	Merge pull request #88 from lookit/fix-randomizer-docs Fix randomizer docs
Files changed	renamed: docs/classes/Permute.html renamed: docs/classes/Select.html

After saving, click the name of your study to return to the study detail view:



Study Editor

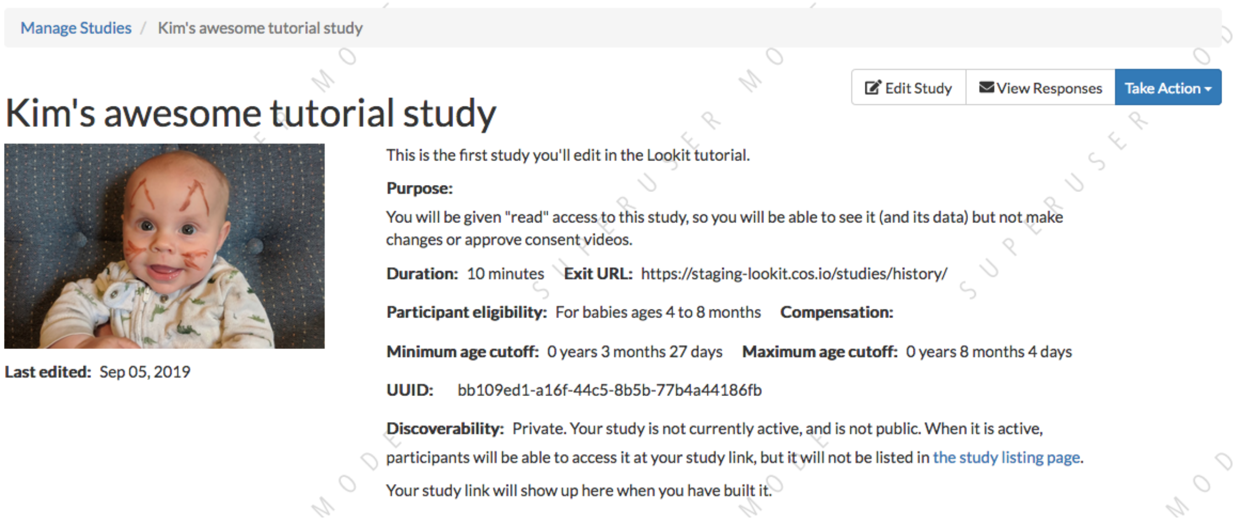
Name

Kim's awesome tutorial study

Image

Currently: study_images/sample_thumbnail.jpg

You should see your changes reflected, like this:



Kim's awesome tutorial study

This is the first study you'll edit in the Lookit tutorial.

Purpose:
You will be given "read" access to this study, so you will be able to see it (and its data) but not make changes or approve consent videos.

Duration: 10 minutes Exit URL: https://staging-lookit.cos.io/studies/history/

Participant eligibility: For babies ages 4 to 8 months Compensation:

Minimum age cutoff: 0 years 3 months 27 days Maximum age cutoff: 0 years 8 months 4 days

UUID: bb109ed1-a16f-44c5-8b5b-77b4a44186fb

Discoverability: Private. Your study is not currently active, and is not public. When it is active, participants will be able to access it at your study link, but it will not be listed in the study listing page.

Your study link will show up here when you have built it.

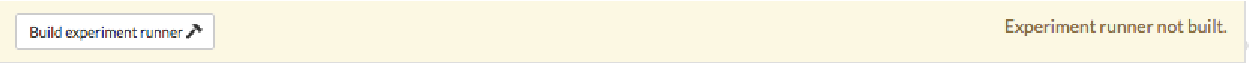
Last edited: Sep 05, 2019

Congratulations! You've created and edited your first study.

1.24.2 Step 2: Preview your study (and learn a bit about JSON on the way)

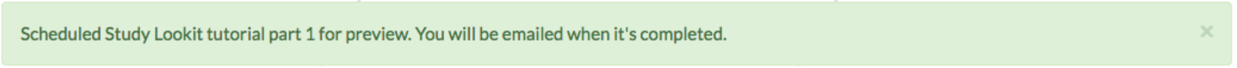
You may have noticed that below your thumbnail and basic study info, there's a section about the "status" of your study. This section is where you will submit your study for approval by Lookit staff when it's ready, and start and stop data collection.

This section also shows whether your "experiment runner" is "built" yet. You should see a bar like this:



Experiment runner not built.

Click the "Build experiment runner" button. You should see a notification at the top of the screen, something like this:


Scheduled Study Lookit tutorial part 1 for preview. You will be emailed when it's completed.

What is this "experiment runner"? When you create a study on Lookit, you specify what types of pages or "frames" to use, and provide parameters for each - for example you supply the text for an instructions page, videos or images to show in a preferential looking trial, audio and images for a storybook page, and so on. The Lookit frame player interprets this information and turns it into an interactive study families can participate in. There's code behind the scenes, which you don't have to deal with, to handle that interpretation and to make each page "go" (saying what each button should do and what data to collect, arranging and starting/stopping video, etc.) Rather than all studies sharing that code, each study gets its own siloed little environment called a Docker image where it will run.

When you click "Build experiment runner," you are creating that Docker image and installing all the necessary code on it - the Lookit frameplayer and the other libraries it depends on. This way, as we continue expanding the Lookit frameplayer code, your study will continue to run exactly as you initially designed and tested it, unless you choose to update what code your study uses and build dependencies again - for instance to take advantage of a new feature or a bug fix. You also have the advanced option of telling Lookit to use your own code instead of the standard Lookit code - for instance if your work needs a very specialized type of test trial that you want to write your own frame for.

It will probably take about 10 minutes to build the experiment runner (you can wait for the email or refresh the page to see if it's done yet).

While you're waiting, go read section on the JSON format, which you will need for the next step.

Exercises

Here are several things that are almost, but not quite, valid JSON objects. Copy and paste each of them into jsonlint, then fix the problem until you see a "valid JSON" message when you click "Validate."

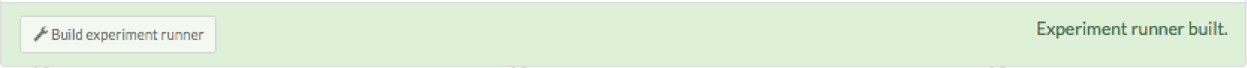
1.
   ```
   {
       "species": "cat",
       "lives_left": 7
       "enemies": ["dog", "laser", "spider"]
   }
   ```

2.
   ```
   {
       "species": "human",
       "age_class": "toddler",
       "favorites": {
            "words": ["uh-oh," "doggie," "ball," "hi"],
            "foods": ["cumin seeds, but not in a food, only plain", "bananas, but only in the grocery store, not after"]
       }
   }
   ```

3.
   ```
   {
       "species": "human",
       "role": "parent",
       "mood": "loving",
       "mood": "exhausted"
   }
   ```

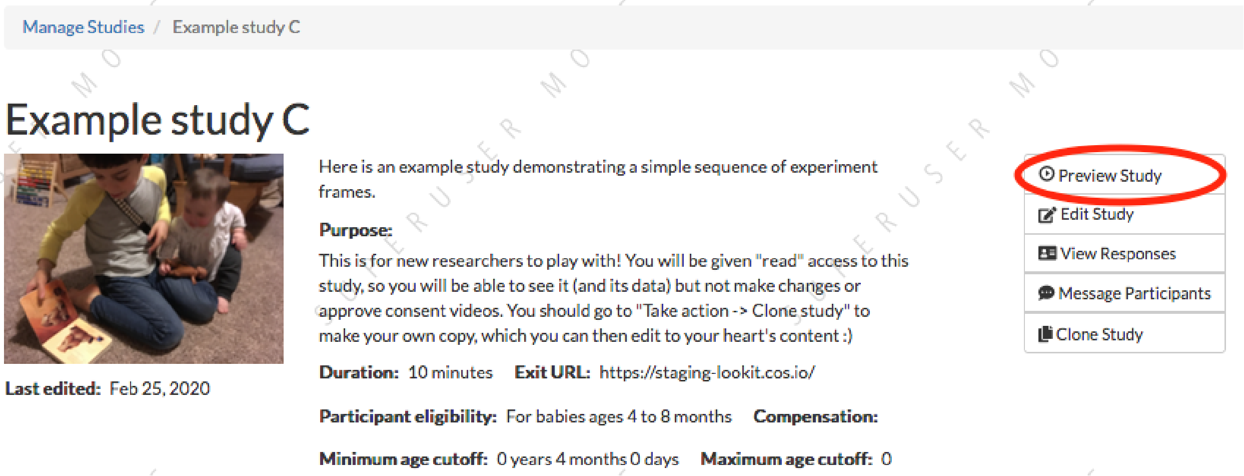
4.
   ```
   {
       "species": "human",
       "age_class"; 'child',
       "is_adorable": True
   }
   ```

OK, congrats on learning all about JSON! Your study should be ready to preview by now. You should have an email in your inbox from Lookit about this, and if you refresh the page you're on, you should see something like this:



Experiment runner built.

Click on "Preview study" near the top of the page:



Example study C

Here is an example study demonstrating a simple sequence of experiment frames.

Purpose:
This is for new researchers to play with! You will be given "read" access to this study, so you will be able to see it (and its data) but not make changes or approve consent videos. You should go to "Take action -> Clone study" to make your own copy, which you can then edit to your heart's content :)

Duration: 10 minutes **Exit URL:** https://staging-lookit.cos.io/

Participant eligibility: For babies ages 4 to 8 months **Compensation:**

Minimum age cutoff: 0 years 4 months 0 days **Maximum age cutoff:** 0

Last edited: Feb 25, 2020

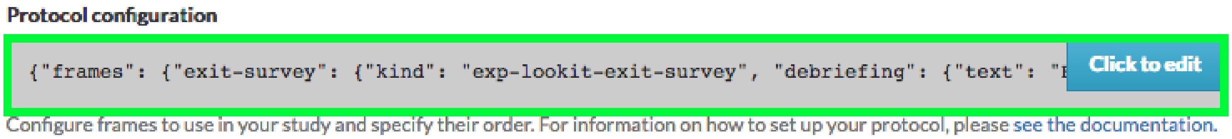
This will take you to a "study detail" page just like the one participants see when they click on a study at lookit.mit.edu/studies. You will need to register at least one child and respond to the demographic survey (you don't need to respond to all questions, or use real information). Then you will be able to click "Preview now" to proceed through the study as a participant. It's a rough, abbreviated implementation of one condition from Schulz, Bonawitz, and Griffiths (2007) - you'll read through a storybook about Bunny, who sometimes gets a tummyache, and eventually answer a question about what makes her tummy hurt.

Note

Video is collected during previews! Like other data, it's only accessible to people who have appropriate permissions. That does include a few Lookit staff in addition to researchers working on your study. We don't do anything with the video and are very unlikely to even see it, but it *is* in principle possible - so please wear clothes while testing, don't sit in front of your really cool poster of your social security number, etc. Or cover your webcam.

1.24.3 Step 3: Get comfortable making changes to how your study works

The "meat" of your study is in the "Study protocol configuration", which you can change from the Edit Study view. Scroll down and click on the text here:

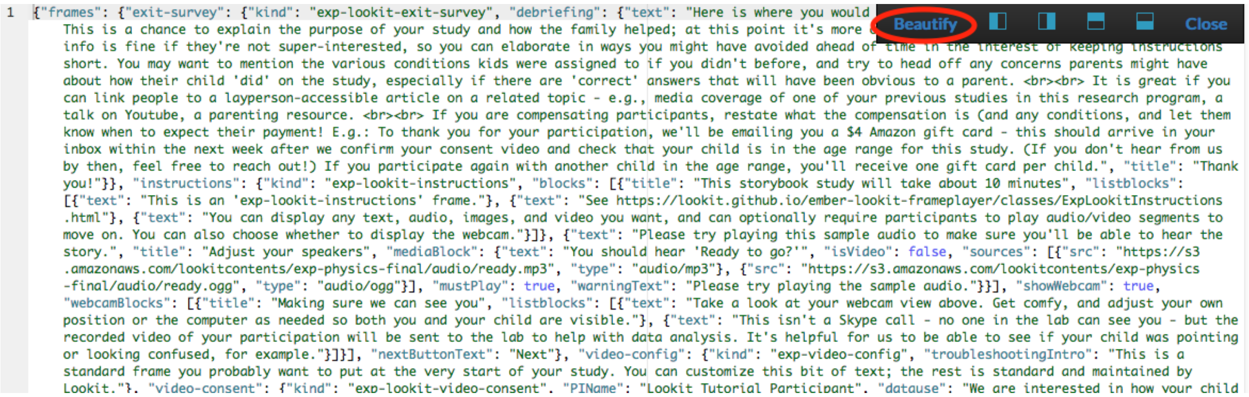


Protocol configuration

```
{"frames": {"exit-survey": {"kind": "exp-lookit-exit-survey", "debriefing": {"text": "I
```

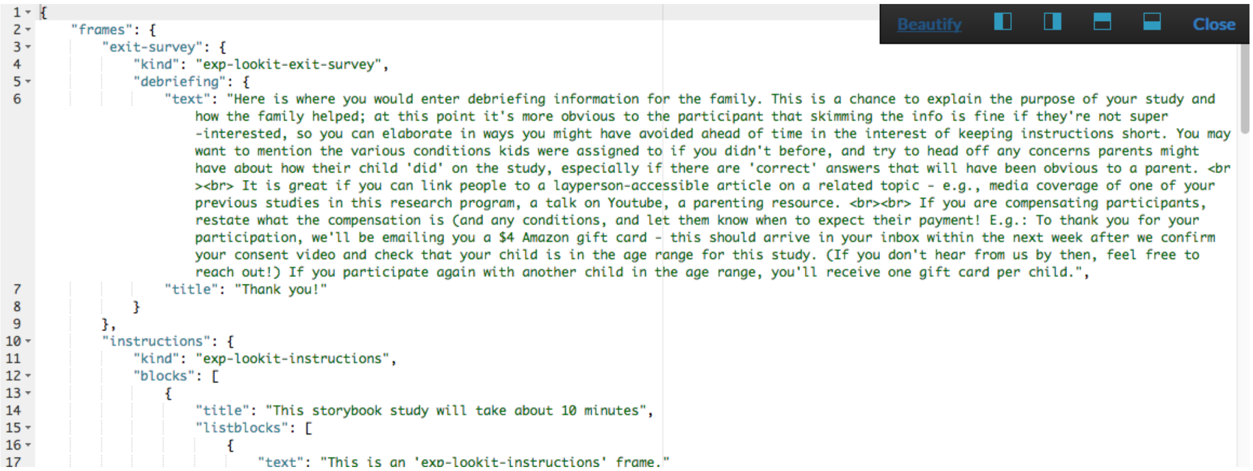
Configure frames to use in your study and specify their order. For information on how to set up your protocol, please see the documentation.

This opens up an editor. It will show a scary amount of text like this:



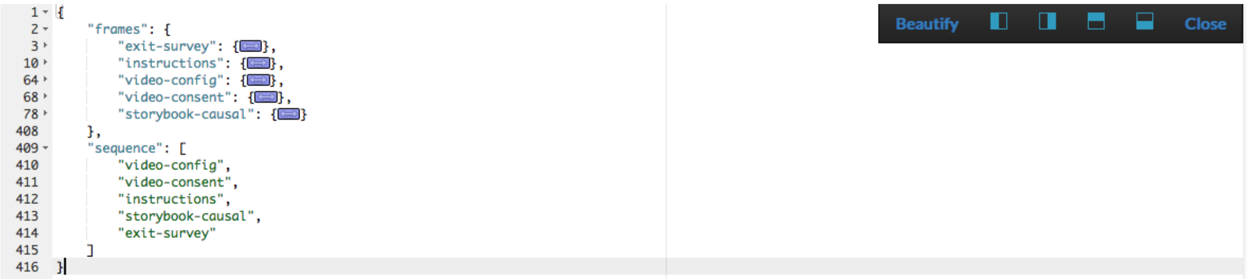
```
{"frames": {"exit-survey": {"kind": "exp-lookit-exit-survey", "debriefing": {"text": "Here is where you would
    This is a chance to explain the purpose of your study and how the family helped; at this point it's more
    info is fine if they're not super-interested, so you can elaborate in ways you might have avoided ahead of time in the interest of keeping instructions
    short. You may want to mention the various conditions kids were assigned to if you didn't before, and try to head off any concerns parents might have
    about how their child 'did' on the study, especially if there are 'correct' answers that will have been obvious to a parent. <br><br> It is great if you
    can link people to a layperson-accessible article on a related topic - e.g., media coverage of one of your previous studies in this research program, a
    talk on Youtube, a parenting resource. <br><br> If you are compensating participants, restate what the compensation is (and any conditions, and let them
    know when to expect their payment! E.g.: To thank you for your participation, we'll be emailing you a $4 Amazon gift card - this should arrive in your
    inbox within the next week after we confirm your consent video and check that your child is in the age range for this study. (If you don't hear from us
    by then, feel free to reach out!) If you participate again with another child in the age range, you'll receive one gift card per child.", "title": "Thank
    you!"}}, "instructions": {"kind": "exp-lookit-instructions", "blocks": [{"title": "This storybook study will take about 10 minutes", "listblocks":
    [{"text": "This is an 'exp-lookit-instructions' frame."}, {"text": "See https://lookit.github.io/ember-lookit-frameplayer/classes/ExpLookitInstructions
    .html"}, {"text": "You can display any text, audio, images, and video you want, and can optionally require participants to play audio/video segments to
    move on. You can also choose whether to display the webcam."}]}, {"text": "Please try playing this sample audio to make sure you'll be able to hear the
    story.", "title": "Adjust your speakers", "mediaBlock": {"text": "You should hear 'Ready to go?'", "isVideo": false, "sources": [{"src": "https://s3
    .amazonaws.com/lookitcontents/exp-physics-final/audio/ready.mp3", "type": "audio/mp3"}, {"src": "https://s3.amazonaws.com/lookitcontents/exp-physics
    -final/audio/ready.ogg", "type": "audio/ogg"}], "mustPlay": true, "warningText": "Please try playing the sample audio."}}], "showWebcam": true,
    "webcamBlocks": [{"title": "Making sure we can see you", "listblocks": [{"text": "Take a look at your webcam view above. Get comfy, and adjust your own
    position or the computer as needed so both you and your child are visible."}, {"text": "This isn't a Skype call - no one in the lab can see you - but the
    recorded video of your participation will be sent to the lab to help with data analysis. It's helpful for us to be able to see if your child was pointing
    or looking confused, for example."}]}], "nextButtonText": "Next"}, "video-config": {"kind": "exp-video-config", "troubleshootingIntro": "This is a
    standard frame you probably want to put at the very start of your study. You can customize this bit of text; the rest is standard and maintained by
    Lookit."}, "video-consent": {"kind": "exp-lookit-video-consent", "PIName": "Lookit Tutorial Participant", "datause": "We are interested in how your child
```

Click the “Beautify” button to format it nicely:



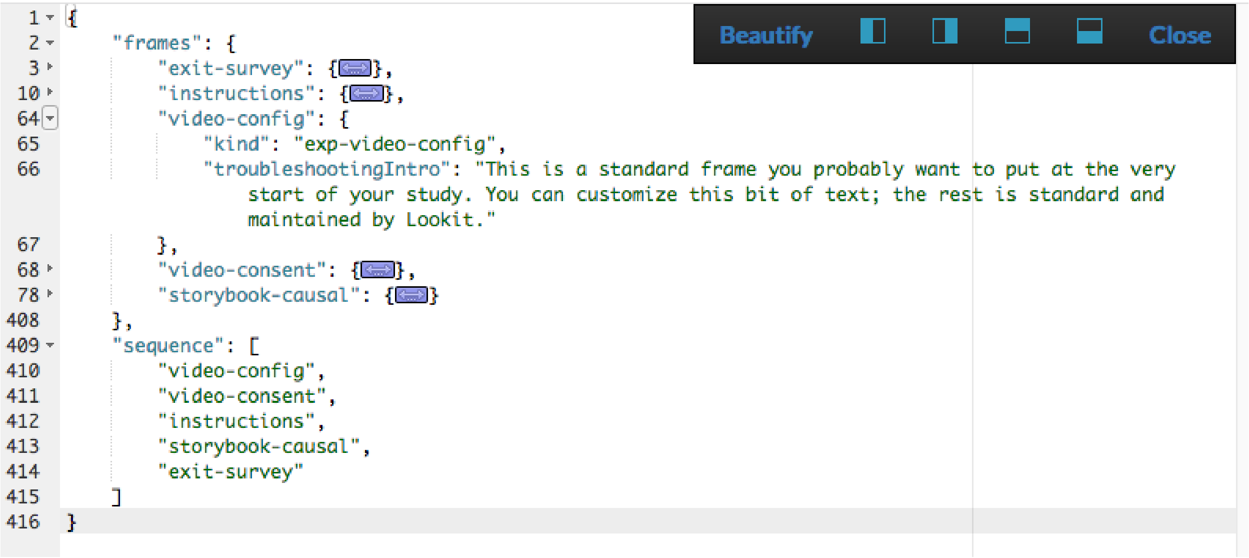
```
{
    "frames": {
        "exit-survey": {
            "kind": "exp-lookit-exit-survey",
            "debriefing": {
                "text": "Here is where you would enter debriefing information for the family. This is a chance to explain the purpose of your study and
                    how the family helped; at this point it's more obvious to the participant that skimming the info is fine if they're not super
                    -interested, so you can elaborate in ways you might have avoided ahead of time in the interest of keeping instructions short. You may
                    want to mention the various conditions kids were assigned to if you didn't before, and try to head off any concerns parents might
                    have about how their child 'did' on the study, especially if there are 'correct' answers that will have been obvious to a parent. <br
                    ><br> It is great if you can link people to a layperson-accessible article on a related topic - e.g., media coverage of one of your
                    previous studies in this research program, a talk on Youtube, a parenting resource. <br><br> If you are compensating participants,
                    restate what the compensation is (and any conditions, and let them know when to expect their payment! E.g.: To thank you for your
                    participation, we'll be emailing you a $4 Amazon gift card - this should arrive in your inbox within the next week after we confirm
                    your consent video and check that your child is in the age range for this study. (If you don't hear from us by then, feel free to
                    reach out!) If you participate again with another child in the age range, you'll receive one gift card per child.",
                "title": "Thank you!"
            }
        },
        "instructions": {
            "kind": "exp-lookit-instructions",
            "blocks": [
                {
                    "title": "This storybook study will take about 10 minutes",
                    "listblocks": [
                        {
                            "text": "This is an 'exp-lookit-instructions' frame."
```

This whole “protocol” is a JSON document, like we learned about while you were waiting for your preview dependencies to build. Using the triangles on the left may help you to explore and understand its structure better. Try collapsing headers by clicking those triangles until you can see this overall structure:



```
{
    "frames": {
        "exit-survey": {↔},
        "instructions": {↔},
        "video-config": {↔},
        "video-consent": {↔},
        "storybook-causal": {↔}
    },
    "sequence": [
        "video-config",
        "video-consent",
        "instructions",
        "storybook-causal",
        "exit-survey"
    ]
}
```

There are two high-level keys: `frames` and `sequence`. The `sequence` list says what frames should be in your study, in what order. The `frames` object is like a dictionary where the Lookit frameplayer will go to understand what each frame in the sequence should be like. Notice that each of the strings in the `sequence` is itself a key in `frames`. For instance, the sequence starts with `video-config`. We can expand the `video-config` key in `frames` to see more about that frame:



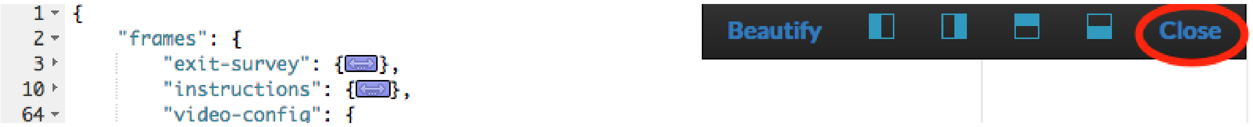
```
{
    "frames": {
        "exit-survey": {↔},
        "instructions": {↔},
        "video-config": {
            "kind": "exp-video-config",
            "troubleshootingIntro": "This is a standard frame you probably want to put at the very start of your study. You can customize this bit of text; the rest is standard and maintained by Lookit."
        },
        "video-consent": {↔},
        "storybook-causal": {↔}
    },
    "sequence": [
        "video-config",
        "video-consent",
        "instructions",
        "storybook-causal",
        "exit-survey"
    ]
}
```

Making a change to the sequence

Let's change the `sequence`` to see how it affects the study. In the editor, move `"instructions"` to the start of the `sequence` list. It should end up looking like this:

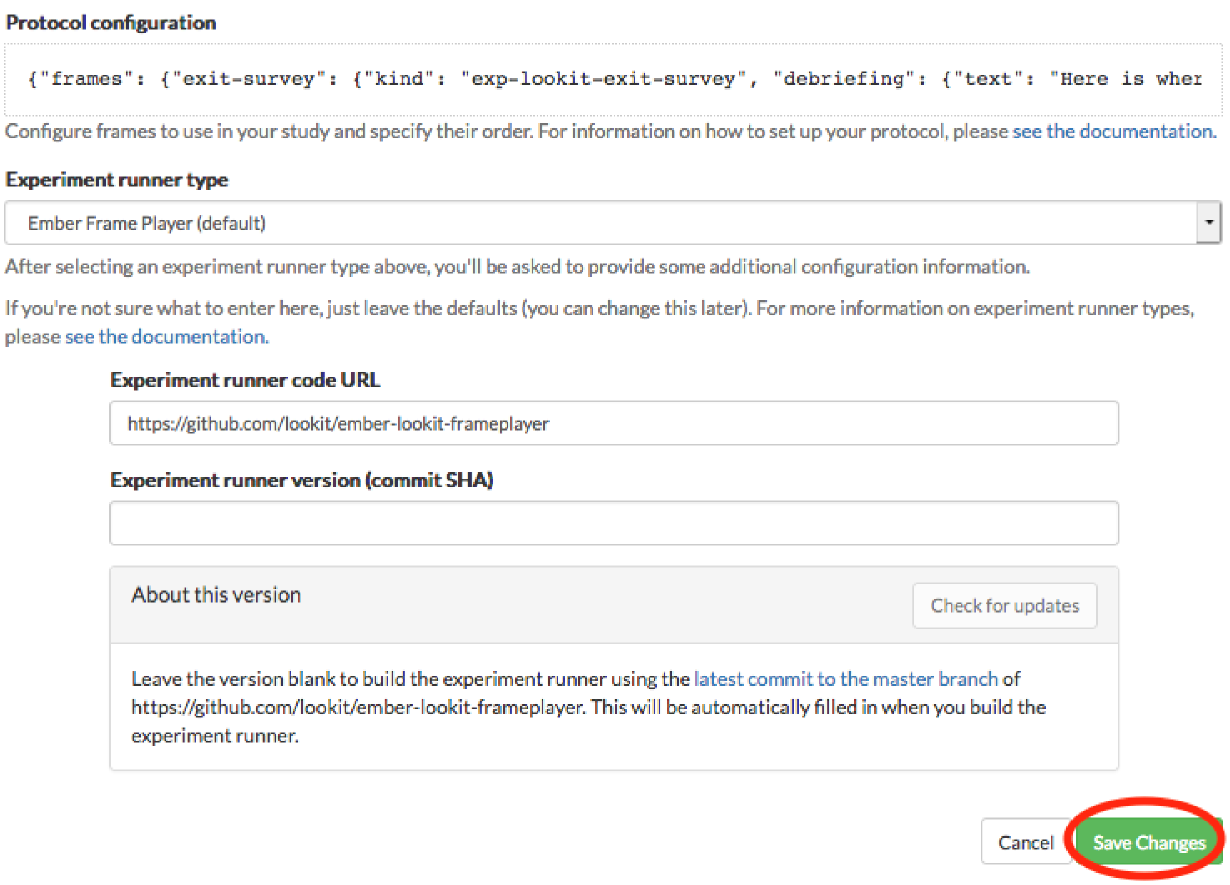
```
"sequence": [
   "instructions",
   "video-config",
   "video-consent",
   "storybook-causal",
   "exit-survey"
]
```

Now click "Close" at the top right to exit the editor:



```
{
    "frames": {
        "exit-survey": {↔},
        "instructions": {↔},
        "video-config": {
```

This will return you to the Edit Study page, but your changes aren't saved yet. Scroll down and click "Save Changes":



Protocol configuration

```
{"frames": {"exit-survey": {"kind": "exp-lookit-exit-survey", "debriefing": {"text": "Here is wher
```

Configure frames to use in your study and specify their order. For information on how to set up your protocol, please see the documentation.

Experiment runner type

Ember Frame Player (default)

After selecting an experiment runner type above, you'll be asked to provide some additional configuration information.

If you're not sure what to enter here, just leave the defaults (you can change this later). For more information on experiment runner types, please see the documentation.

Experiment runner code URL

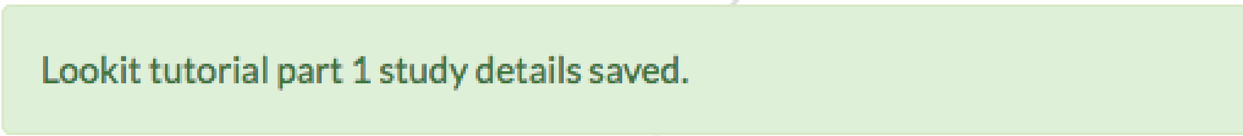
https://github.com/lookit/ember-lookit-frameplayer

Experiment runner version (commit SHA)

About this version

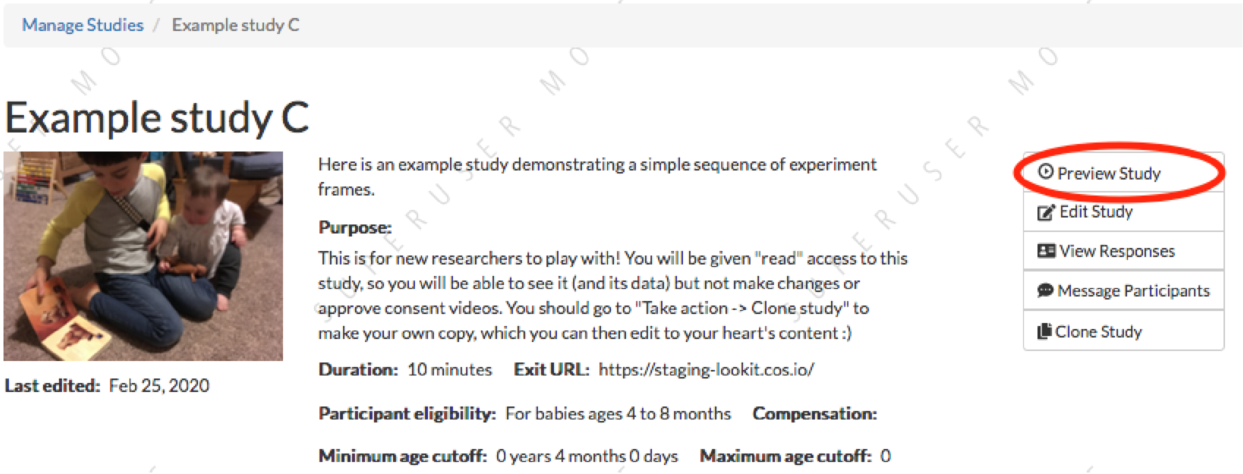
Leave the version blank to build the experiment runner using the latest commit to the master branch of https://github.com/lookit/ember-lookit-frameplayer. This will be automatically filled in when you build the experiment runner.

You should see a message at the top like the following. (If not, click on your protocol again and resolve any problems that are preventing it from saving.)



Lookit tutorial part 1 study details saved.

Now you can preview your edited study by clicking on the blue "Preview" button again:



Manage Studies / Example study C

Example study C

Here is an example study demonstrating a simple sequence of experiment frames.

Purpose:

This is for new researchers to play with! You will be given "read" access to this study, so you will be able to see it (and its data) but not make changes or approve consent videos. You should go to "Take action -> Clone study" to make your own copy, which you can then edit to your heart's content :)

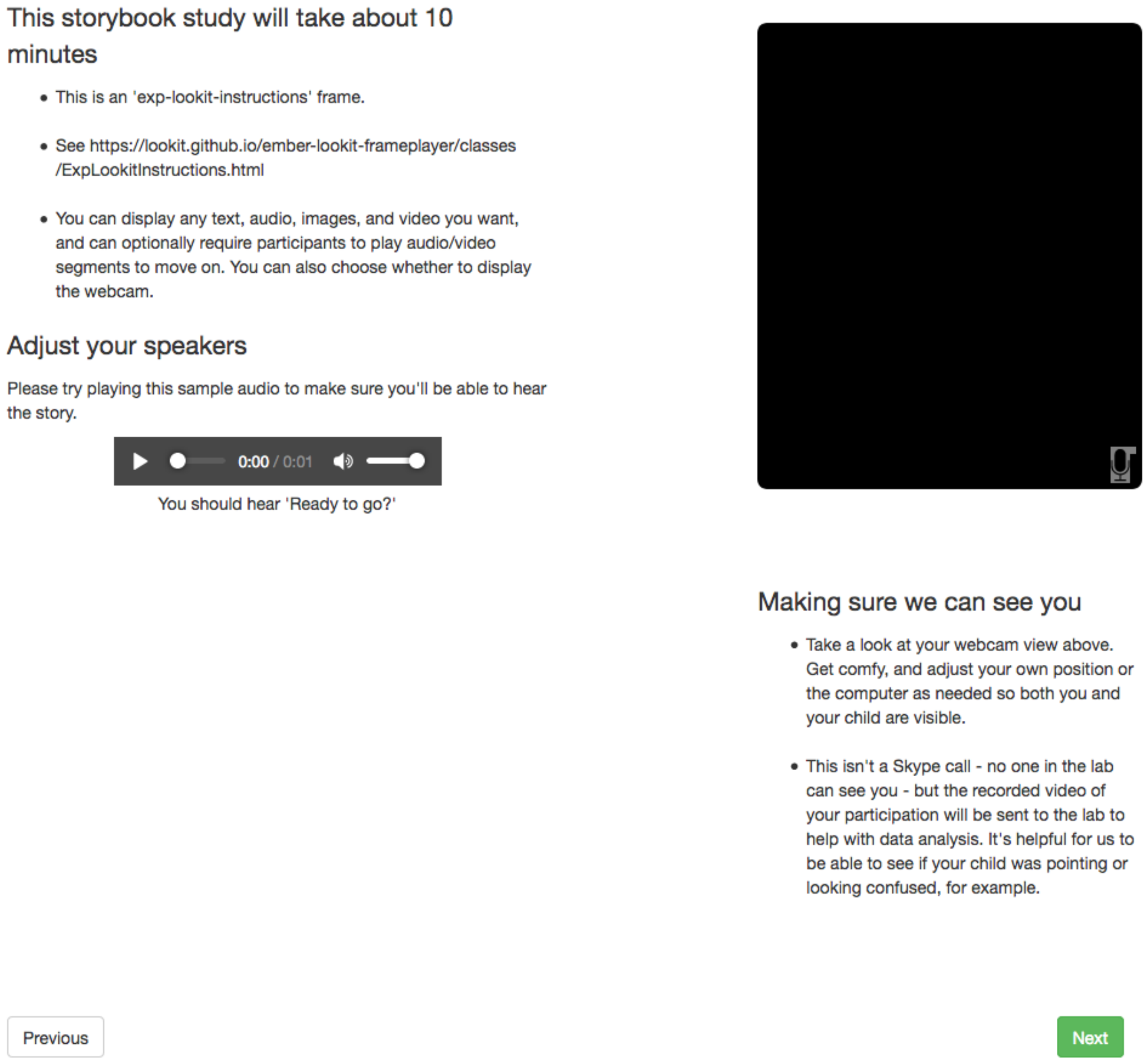
Duration: 10 minutes **Exit URL:** https://staging-lookit.cos.io/

Participant eligibility: For babies ages 4 to 8 months **Compensation:**

Minimum age cutoff: 0 years 4 months 0 days **Maximum age cutoff:** 0

Last edited: Feb 25, 2020

This time, you should be right at the instructions, instead of starting with the video configuration frame! If you back and click "Preview study" and then "Preview now" again, you should go right away to a page like this:



This storybook study will take about 10 minutes

- This is an 'exp-lookit-instructions' frame.
- See https://lookit.github.io/ember-lookit-frameplayer/classes/ExpLookitInstructions.html
- You can display any text, audio, images, and video you want, and can optionally require participants to play audio/video segments to move on. You can also choose whether to display the webcam.

Adjust your speakers

Please try playing this sample audio to make sure you'll be able to hear the story.

You should hear 'Ready to go?'

Making sure we can see you

- Take a look at your webcam view above. Get comfy, and adjust your own position or the computer as needed so both you and your child are visible.
- This isn't a Skype call - no one in the lab can see you - but the recorded video of your participation will be sent to the lab to help with data analysis. It's helpful for us to be able to see if your child was pointing or looking confused, for example.

That's because we moved the "instructions" frame to the start of our sequence. If you want to make changes to a particular frame, sticking it at the beginning of your sequence can make it easier to rapidly view your changes as you make them.

Speed up the process a bit

You may want to bookmark the URL you're at when you see that instructions page. That's the URL to preview this study with the child you selected. You can refresh this page to see your updated preview right away, without having to click through the study detail page and select a child.

Making a change to an individual frame

While we have that instructions page “front and center,” let’s edit the text so it looks more like real instructions for the study!

From the Edit Study page, click on your study protocol to open up the editor again. Click “Beautify” to make it easier to read. Find the section that defines the “instructions” frame (starting on line 10).

Tip

In addition to using the triangles at the right to expand/collapse sections of your protocol, you can double-click on a bracket or curly brace to highlight everything up until the matching one.

Here’s what it looks like now. You don’t need to understand everything going on here - just note that the text you saw in the preview is defined here! The “webcamBlocks” value at the bottom has the text you see under the webcam. The “blocks” value is a list of two sections. The first one is (or should be) a little overview of instructions for the study. The second just has participants check that their speakers are on and volume is ok.

```
"instructions": {
     "kind": "exp-lookit-instructions",
     "blocks": [
         {
             "title": "Overview of how to participate in this study",
             "listblocks": [
                 {
                     "text": "This is an 'exp-lookit-instructions' frame."
                 },
                 {
                     "text": "See https://lookit.readthedocs.io/projects/frameplayer/
↪components/exp-lookit-instructions/doc.html"
                 },
                 {
                     "text": "You can display any text, audio, images, and video you␣
↪want, and can optionally require participants to play audio/video segments to move␣
↪on. You can also choose whether to display the webcam."
                 }
             ]
         },
         {
             "text": "Please try playing this sample audio to make sure you'll be␣
↪able to hear the story.",
             "title": "Adjust your speakers",
             "mediaBlock": {
                 "text": "You should hear 'Ready to go?'",
                 "isVideo": false,
                 "sources": [
                     {
                         "src": "https://s3.amazonaws.com/lookitcontents/exp-physics-
↪final/audio/ready.mp3",
                         "type": "audio/mp3"
                     },
                     {
                         "src": "https://s3.amazonaws.com/lookitcontents/exp-physics-
↪final/audio/ready.ogg",
                         "type": "audio/ogg"
                     }
                 ],
```

(continues on next page)

(continued from previous page)

```
                 "mustPlay": true,
                 "warningText": "Please try playing the sample audio."
             }
         }
     ],
     "showWebcam": true,
     "webcamBlocks": [
         {
             "title": "Make sure we can see you",
             "listblocks": [
                 {
                     "text": "Take a look at your webcam view above. Get comfy, and␣
↪adjust your own position or the computer as needed so both you and your child are␣
↪visible."
                 },
                 {
                     "text": "This isn't a Skype call - no one in the lab can see you␣
↪- but the recorded video of your participation will be sent to the lab to help with␣
↪data analysis. It's helpful for us to be able to see if your child was pointing or␣
↪looking confused, for example."
                 }
             ]
         }
     ],
     "nextButtonText": "Next"
 },
```

First, let's flesh out the "overview of how to participate" section by replacing the text with more appropriate instruction text like this:

```
{
    "title": "Overview of how to participate in this study",
    "listblocks": [
        {
            "text": "You and your child will listen to a simple illustrated audiobook␣
↪together."
        },
        {
            "text": "There are 16 pages altogether, and one question at the end for␣
↪your child."
        },
        {
            "text": "While you listen to the story together, you can help talk to␣
↪your child to keep him or her engaged - but please don't talk about WHY you think␣
↪Bunny has a tummyache! We're interested in how your child figures that out on his␣
↪or her own, and won't be able to use data from children if their parents influenced␣
↪their answers. (But there are really, truly no wrong answers!)"
        }
    ]
},
```

Next, let's help guide families through this frame by adding numbers to the sections. (In a real study you might also consider breaking up a page like this into three shorter pages!)

- Find the line `"title": "Overview of how to participate in this study"`, and change that to `"title": "1. Overview of how to participate in this study",`
- Find the line `"title": "Adjust your speakers"`, and change that to `"title": "2. Adjust`

`your speakers",`

- Find the line `"title": "Make sure we can see you",`, and change that to `"title": "3. Make sure we can see you",`

Click "Close" in the top right corner of the editor, and then scroll down and click "Save Changes." (Make sure you see the message at the top that changes were saved successfully - fix any problems with the protocol not being valid JSON if not!) Now click "Preview study" again to see your new and improved instructions page!

Put the instructions back in order

Now that we've made our changes to the instructions frame, let's put it back where it belongs, after the video configuration and consent frames.

Open the protocol editor and find the `sequence` way at the end. Right not it should still look like this:

```
"sequence": [
   "instructions",
   "video-config",
   "video-consent",
   "storybook-causal",
   "exit-survey"
]
```

Move `"instructions"` back so it looks like:

```
"sequence": [
   "video-config",
   "video-consent",
   "instructions",
   "storybook-causal",
   "exit-survey"
]
```

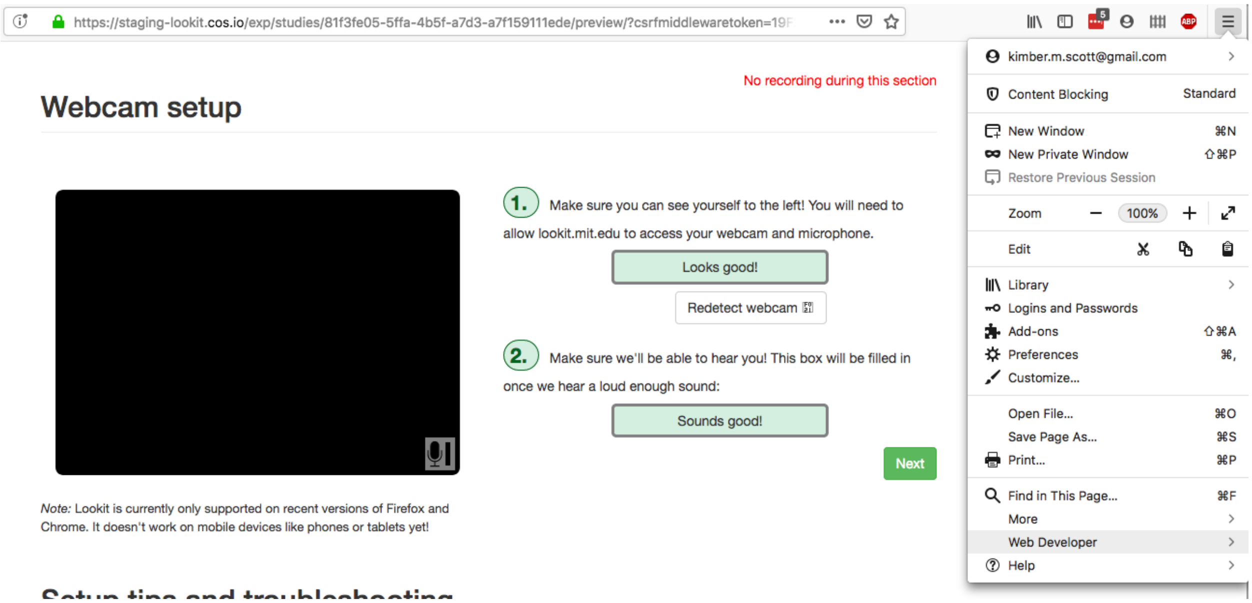
Close, save changes, and preview it again. After proceeding through video configuration and video consent, you should see your new and improved instructions.

Using the Javascript console in your browser to learn more about any problems

One of the most powerful tools you have available to troubleshoot any problems as you set up your study is called the "web console" or "Javascript console" in your web browser.

Click to preview your study, and from that browser window/tab, let's get your web console open so we can see what's going on.

If you're using Firefox: Click the "hamburger menu" (three horizontal lines) in the top right corner of your browser and click "Web Developer" (yep, that's you now!):



https://staging-lookit.cos.io/exp/studies/81f3fe05-5ffa-4b5f-a7d3-a7f159111ede/preview/?csrfmiddlewaretoken=19F

No recording during this section

Webcam setup

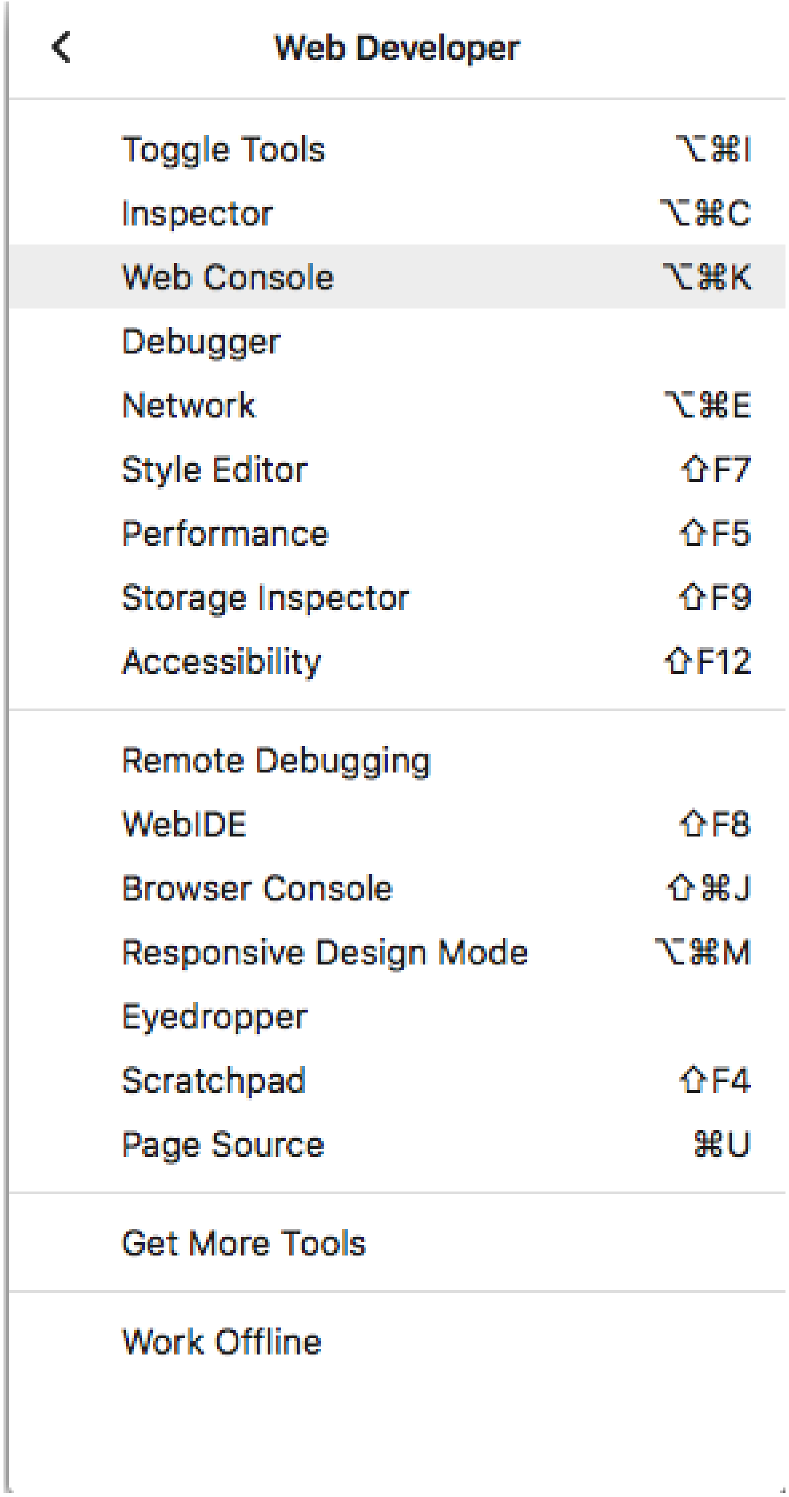
1. Make sure you can see yourself to the left! You will need to allow lookit.mit.edu to access your webcam and microphone.

2. Make sure we'll be able to hear you! This box will be filled in once we hear a loud enough sound:

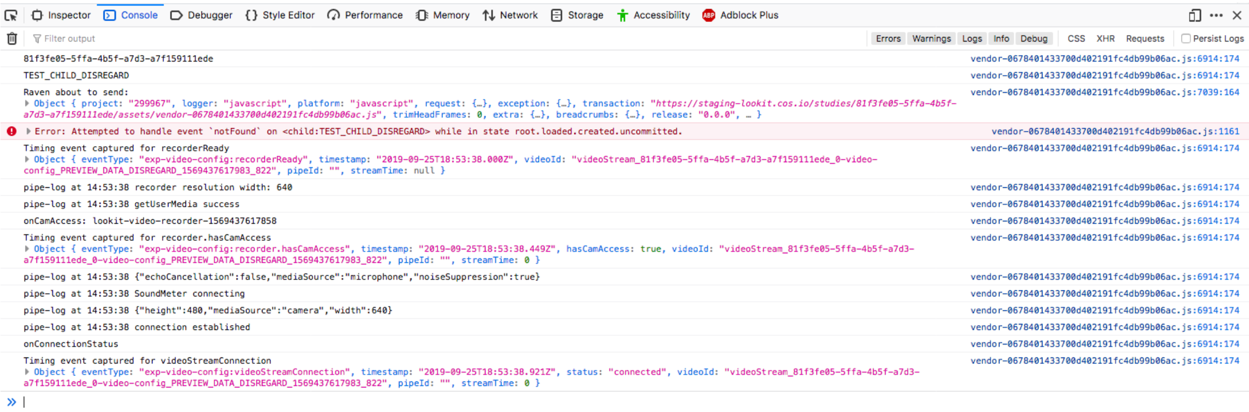
Note: Lookit is currently only supported on recent versions of Firefox and Chrome. It doesn't work on mobile devices like phones or tablets yet!

Setup tips and troubleshooting

Click “Web Console”:

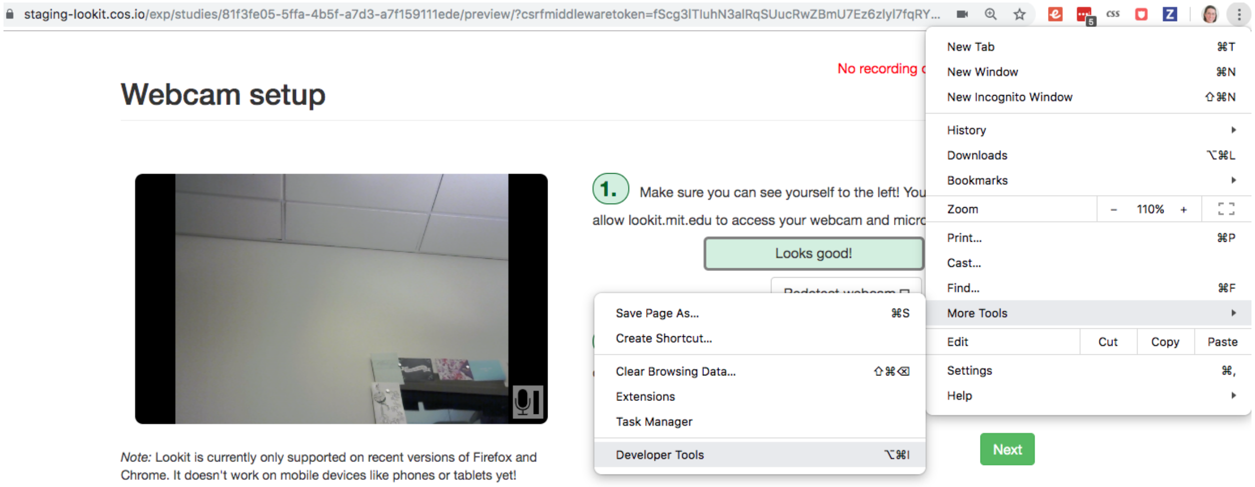


And you should see something like this:

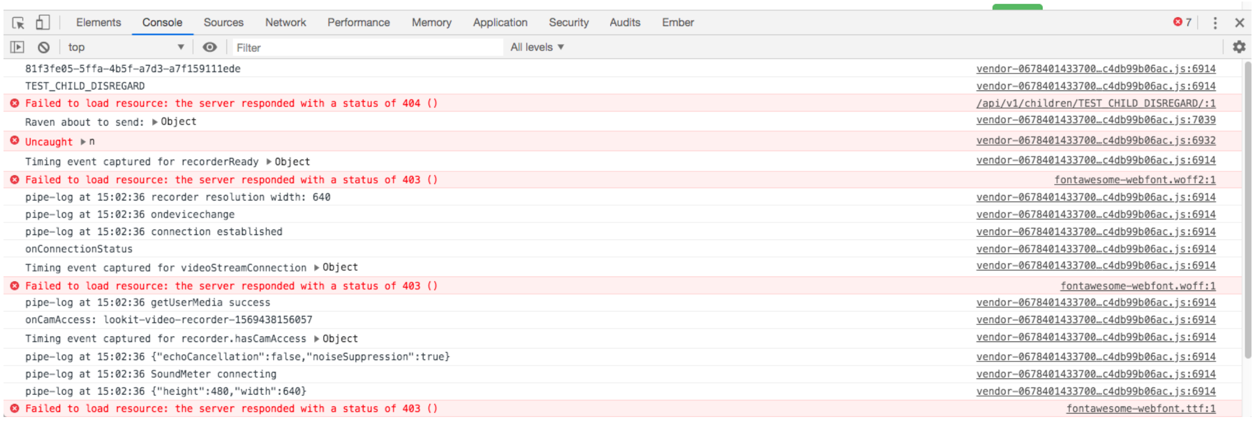


```
81f3fe05-5ffa-4b5f-a7d3-a7f159111ede                                                     vendor-0678401433700d402191fc4db99b06ac.js:6914:174
TEST_CHILD_DISREGARD                                                                     vendor-0678401433700d402191fc4db99b06ac.js:6914:174
Raven about to send:                                                                     vendor-0678401433700d402191fc4db99b06ac.js:7039:164
▶ Object { project: "299967", logger: "javascript", platform: "javascript", request: {…}, exception: {…}, transaction: "https://staging-lookit.cos.io/studies/81f3fe05-5ffa-4b5f-
a7d3-a7f159111ede/assets/vendor-0678401433700d402191fc4db99b06ac.js", trimHeadFrames: 0, extra: {…}, breadcrumbs: {…}, release: "0.0.0", … }
❗ ▶ Error: Attempted to handle event `notFound` on <child:TEST_CHILD_DISREGARD> while in state root.loaded.created.uncommitted.   vendor-0678401433700d402191fc4db99b06ac.js:1161
Timing event captured for recorderReady                                                  vendor-0678401433700d402191fc4db99b06ac.js:6914:174
▶ Object { eventType: "exp-video-config:recorderReady", timestamp: "2019-09-25T18:53:38.000Z", videoId: "videoStream_81f3fe05-5ffa-4b5f-a7d3-a7f159111ede_0-video-
config_PREVIEW_DATA_DISREGARD_1569437617983_822", pipeId: "", streamTime: null }
pipe-log at 14:53:38 recorder resolution width: 640                                      vendor-0678401433700d402191fc4db99b06ac.js:6914:174
pipe-log at 14:53:38 getUserMedia success                                                vendor-0678401433700d402191fc4db99b06ac.js:6914:174
onCamAccess: lookit-video-recorder-1569437617858                                         vendor-0678401433700d402191fc4db99b06ac.js:6914:174
Timing event captured for recorder.hasCamAccess                                          vendor-0678401433700d402191fc4db99b06ac.js:6914:174
▶ Object { eventType: "exp-video-config:recorder.hasCamAccess", timestamp: "2019-09-25T18:53:38.449Z", hasCamAccess: true, videoId: "videoStream_81f3fe05-5ffa-4b5f-a7d3-
a7f159111ede_0-video-config_PREVIEW_DATA_DISREGARD_1569437617983_822", pipeId: "", streamTime: 0 }
pipe-log at 14:53:38 {"echoCancellation":false,"mediaSource":"microphone","noiseSuppression":true}   vendor-0678401433700d402191fc4db99b06ac.js:6914:174
pipe-log at 14:53:38 SoundMeter connecting                                               vendor-0678401433700d402191fc4db99b06ac.js:6914:174
pipe-log at 14:53:38 {"height":480,"mediaSource":"camera","width":640}                   vendor-0678401433700d402191fc4db99b06ac.js:6914:174
pipe-log at 14:53:38 connection established                                              vendor-0678401433700d402191fc4db99b06ac.js:6914:174
onConnectionStatus                                                                       vendor-0678401433700d402191fc4db99b06ac.js:6914:174
Timing event captured for videoStreamConnection                                          vendor-0678401433700d402191fc4db99b06ac.js:6914:174
▶ Object { eventType: "exp-video-config:videoStreamConnection", timestamp: "2019-09-25T18:53:38.921Z", status: "connected", videoId: "videoStream_81f3fe05-5ffa-4b5f-a7d3-
a7f159111ede_0-video-config_PREVIEW_DATA_DISREGARD_1569437617983_822", pipeId: "", streamTime: 0 }
»
```

If you're using Chrome: Click the three dots in the upper right corner, then "More Tools," then "Developer Tools":


staging-lookit.cos.io/exp/studies/81f3fe05-5ffa-4b5f-a7d3-a7f159111ede/preview/?csrfmiddlewaretoken=fScg3lTluhN3alRqSUucRwZBmU7Ez6zlyl7fqRY...
No recording
Webcam setup
1.
Make sure you can see yourself to the left! You
allow lookit.mit.edu to access your webcam and micro
Looks good!
Note: Lookit is currently only supported on recent versions of Firefox and Chrome. It doesn't work on mobile devices like phones or tablets yet!
Next
New Tab ⌘T
New Window ⌘N
New Incognito Window ⇧⌘N
History
Downloads ⌥⌘L
Bookmarks
Zoom – 110% +
Print... ⌘P
Cast...
Find... ⌘F
More Tools
Edit Cut Copy Paste
Settings ⌘,
Help
Save Page As... ⌘S
Create Shortcut...
Clear Browsing Data... ⇧⌘⌫
Extensions
Task Manager
Developer Tools ⌥⌘I

You should see something like this:



```
81f3fe05-5ffa-4b5f-a7d3-a7f159111ede                                  vendor-0678401433700…c4db99b06ac.js:6914
TEST_CHILD_DISREGARD                                                  vendor-0678401433700…c4db99b06ac.js:6914
⊗ Failed to load resource: the server responded with a status of 404 ()   /api/v1/children/TEST_CHILD_DISREGARD/:1
Raven about to send: ▶Object                                          vendor-0678401433700…c4db99b06ac.js:7039
⊗ Uncaught ▶n                                                         vendor-0678401433700…c4db99b06ac.js:6932
Timing event captured for recorderReady ▶Object                       vendor-0678401433700…c4db99b06ac.js:6914
⊗ Failed to load resource: the server responded with a status of 403 ()          fontawesome-webfont.woff2:1
pipe-log at 15:02:36 recorder resolution width: 640                   vendor-0678401433700…c4db99b06ac.js:6914
pipe-log at 15:02:36 ondevicechange                                   vendor-0678401433700…c4db99b06ac.js:6914
pipe-log at 15:02:36 connection established                           vendor-0678401433700…c4db99b06ac.js:6914
onConnectionStatus                                                    vendor-0678401433700…c4db99b06ac.js:6914
Timing event captured for videoStreamConnection ▶Object               vendor-0678401433700…c4db99b06ac.js:6914
⊗ Failed to load resource: the server responded with a status of 403 ()           fontawesome-webfont.woff:1
pipe-log at 15:02:36 getUserMedia success                             vendor-0678401433700…c4db99b06ac.js:6914
onCamAccess: lookit-video-recorder-1569438156057                      vendor-0678401433700…c4db99b06ac.js:6914
Timing event captured for recorder.hasCamAccess ▶Object               vendor-0678401433700…c4db99b06ac.js:6914
pipe-log at 15:02:36 {"echoCancellation":false,"noiseSuppression":true}   vendor-0678401433700…c4db99b06ac.js:6914
pipe-log at 15:02:36 SoundMeter connecting                            vendor-0678401433700…c4db99b06ac.js:6914
pipe-log at 15:02:36 {"height":480,"width":640}                       vendor-0678401433700…c4db99b06ac.js:6914
⊗ Failed to load resource: the server responded with a status of 403 ()            fontawesome-webfont.ttf:1
```

Browser-dependent behavior

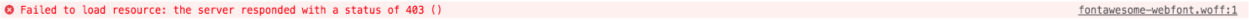
Webcam access functionality, external resource loading, or other features may work slightly differently across web browsers, especially as they are updated over time. It is always worth previewing your studies in both Firefox and Chrome, which are the browsers Lookit currently officially supports. For your own privacy, we strongly recommend not using Chrome more than you have to. (We also recommend flossing and having Easter egg hunts as a year-round activity, but these are getting further from our domain.)

Advanced developer tool features

In both Firefox and Chrome, you have access to a bunch of different tools beyond this basic web console, and you have lots of options for filtering out certain events, where to display the console (e.g. separate window vs. bottom vs. side), etc. - we're just going to cover the basics here!

Now that you've gotten your web console open, you'll see a bunch of information in it. This is generally of most interest if something is going wrong and you're not sure what. You can see events that are being logged as you proceed through the study as well as any errors. Some of these errors are ok to ignore - e.g. here are a few current ones due to known but harmless bugs:

- On the staging server there is a known bug that the fontawesome library doesn't load properly (but it does on "production" - the real Lookit site) - so you may see some errors that a resource failed to load properly, like this:

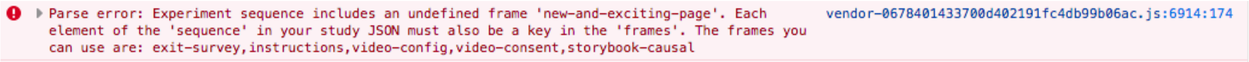


```
Failed to load resource: the server responded with a status of 403 ()          fontawesome-webfont.woff:1
```

Leave your preview tab open, and return to the browser tab where you have the "Edit Study " page open. Let's deliberately introduce a problem in our study JSON and see what we can learn from the preview. Try adding something to the "sequence" without defining it in "frames," like this:

```
"sequence": [
   "video-config",
   "new-and-exciting-page",
   "video-consent",
   "instructions",
   "storybook-causal",
   "exit-survey"
]
```

Close, save changes, and then return to your preview tab and refresh it. You'll see a totally blank page, which would be very confusing if you didn't know what had gone wrong! But if you look down at the web console, you should see an error like this:



```
▶ Parse error: Experiment sequence includes an undefined frame 'new-and-exciting-page'. Each     vendor-0678401433700d402191fc4db99b06ac.js:6914:174
  element of the 'sequence' in your study JSON must also be a key in the 'frames'. The frames you
  can use are: exit-survey,instructions,video-config,video-consent,storybook-causal
```

This explains that the problem is that the Lookit frameplayer can't make sense of your study JSON, because it doesn't have a "definition" available in the "frames" value for the frame "new-and-exciting-page" that you added to your sequence.

Return to the study edit page and open up the JSON editor again. Remove that "new-and-exciting-page" from your "sequence" and let's cause another problem instead. Scroll to the section of the `frames` object where we give parameters for the consent frame:

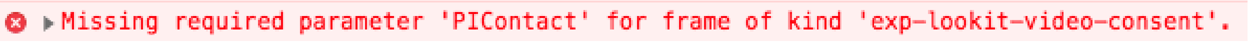
```
"video-consent": {
     "kind": "exp-lookit-video-consent",
     "template": "consent_005",
     "PIName": "Lookit Tutorial Participant",
     "PIContact": "Jane Smith at (123) 456-7890",
     "datause": "We are interested in how your child uses statistical evidence to␣
↪figure out the cause of an event. A research assistant will watch your video and␣
↪mark down your child's answer to the question at the end of the story, and as well␣
↪as other information such as interactions between you and your child during the␣
↪story.",
     "include_databrary": true,
```

(continues on next page)

(continued from previous page)

```
    "risk_statement": "There are no expected risks to participation."
    "payment": "After you finish the study, we will email you a $5 BabyStore gift␣
↪card within approximately three days. To be eligible for the gift card your child␣
↪must be in the age range for this study, you need to submit a valid consent␣
↪statement, and we need to see that there is a child with you. But we will send a␣
↪gift card even if you do not finish the whole study or we are not able to use your␣
↪child's data! There are no other direct benefits to you or your child from␣
↪participating, but we hope you will enjoy the experience.",
    "purpose": "This study is about how children use statistical information to␣
↪adjust their beliefs about cause and effect.",
    "procedures": "In this study you child will view a digital 'storybook' about␣
↪Bunny, who sometimes gets a tummyache. Each day Bunny eats different foods and does␣
↪different activities, and we hear whether she gets a tummyache. Sometimes, Bunny␣
↪feels scared because of show-and-tell. We are interested in how the pattern of␣
↪evidence influences your child's beliefs about what causes Bunny's tummyache. We␣
↪will ask you (the parent) to avoid discussing why Bunny has a tummyache until the␣
↪end of the study.",
    "institution": "Science University"
},
```

Try deleting one of these lines, like `"PIContact": "Jane Smith at (123) 456-7890",`. Close, save, and refresh your preview. Once you get to the consent page, you should see an error like this complaining about the missing parameter:



```
Missing required parameter 'PIContact' for frame of kind 'exp-lookit-video-consent'.
```

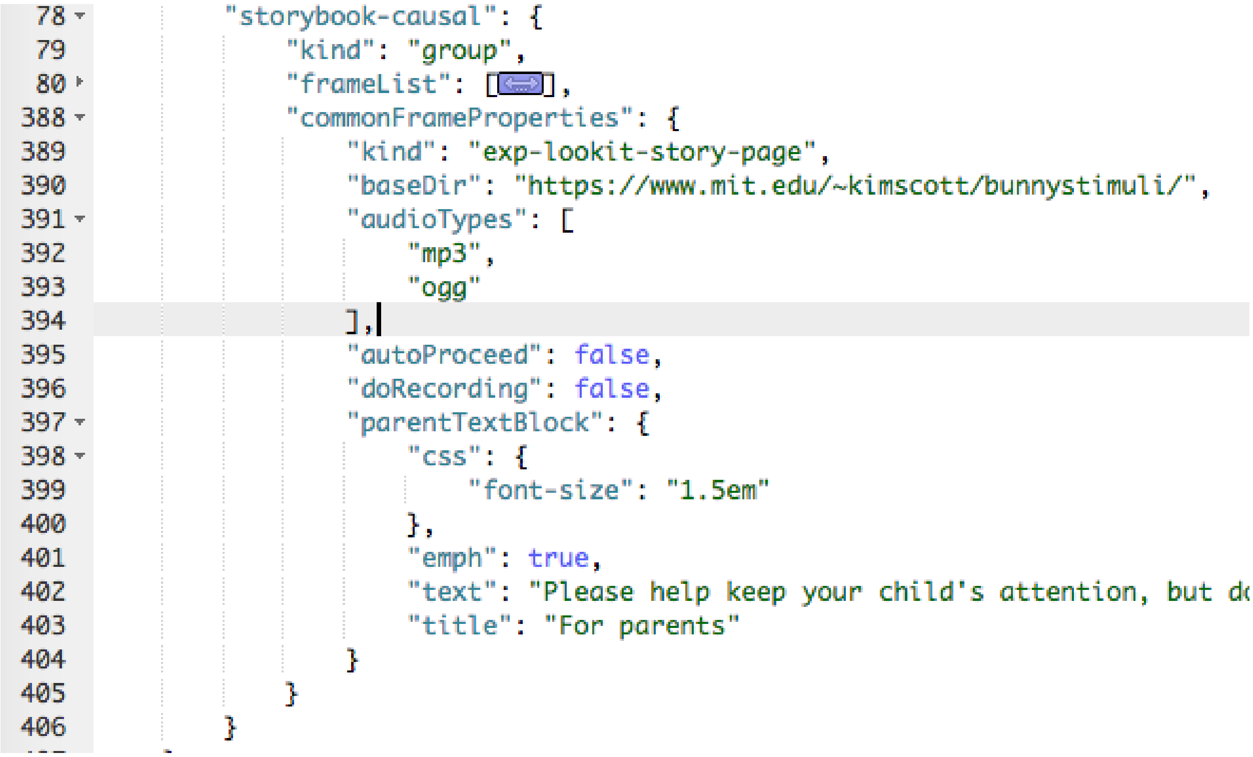
Note that this doesn't stop the frame from working at all (that bit of text is just missing from the consent form if you look carefully) - but this sort of error can be a useful clue in more complicated situations!

Go ahead and put back that "PIContact" field, and let's move on to adding some finishing touches to our study.

Adding another storybook page

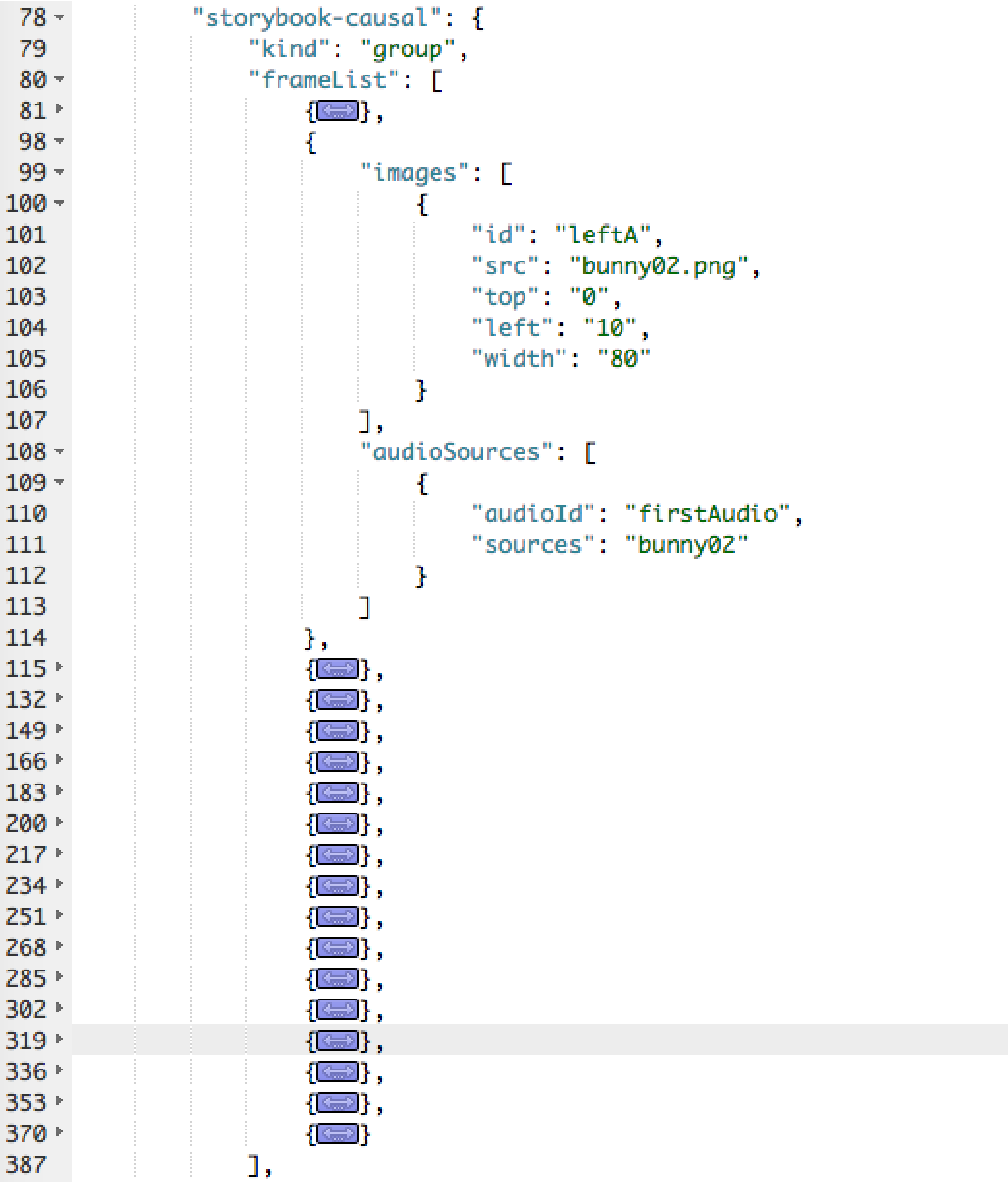
You may have noticed when you tried out the study that the ending was a little abrupt: a question for the child, and then boom! we're out in the exit survey. Let's add one last storybook frame to wrap things up - and reassure kids that Bunny ends up doing just fine at show-and-tell!

Open the protocol JSON editor again. Inside the `frames` object, find the `storybook-causal` frame definiton. It should look something like this (with the long `frameList` collapsed):



```
"storybook-causal": {
    "kind": "group",
    "frameList": [↔],
    "commonFrameProperties": {
        "kind": "exp-lookit-story-page",
        "baseDir": "https://www.mit.edu/~kimscott/bunnystimuli/",
        "audioTypes": [
            "mp3",
            "ogg"
        ],
        "autoProceed": false,
        "doRecording": false,
        "parentTextBlock": {
            "css": {
                "font-size": "1.5em"
            },
            "emph": true,
            "text": "Please help keep your child's attention, but d
            "title": "For parents"
        }
    }
}
```

This is a frame "group" that actually bundles together a list of frames, adding some `commonFrameProperties` to each one. You don't have to understand that yet! For now, let's take a look inside the `frameList` where the list of storybook pages is. Each element of this list is an object with `images` and `audioSources` - here's what it looks like collapsing most of those list elements:



```
    "storybook-causal": {
        "kind": "group",
        "frameList": [
            {↔},
            {
                "images": [
                    {
                        "id": "leftA",
                        "src": "bunny02.png",
                        "top": "0",
                        "left": "10",
                        "width": "80"
                    }
                ],
                "audioSources": [
                    {
                        "audioId": "firstAudio",
                        "sources": "bunny02"
                    }
                ]
            },
            {↔},
            {↔},
            {↔},
            {↔},
            {↔},
            {↔},
            {↔},
            {↔},
            {↔},
            {↔},
            {↔},
            {↔},
            {↔},
            {↔},
            {↔},
            {↔}
        ],
```

We're just going to add one more page to the end. Within the `frameList` list, after the last object, add a comma and then the following:

```
{
    "images": [
        {
            "id": "storybookIllustration",
```

(continues on next page)

(continued from previous page)

```
                "src": "bunnyend01.png",
                "top": "0",
                "left": "10",
                "width": "80"
            }
        ],
        "audioSources": [
            {
                "audioId": "voiceover",
                "sources": "bunnyend01"
            }
        ]
}
```

Here we're providing the name of an image to use ("bunnyend01.png") and audio to use ("bunnyend01") - if you're curious, the absolute paths to these resources are built using the `baseDir` provided to all frames in the list.

Close, save, and refresh your preview. Now after the question, you should see and hear a friendly wrap-up to this thrilling story.

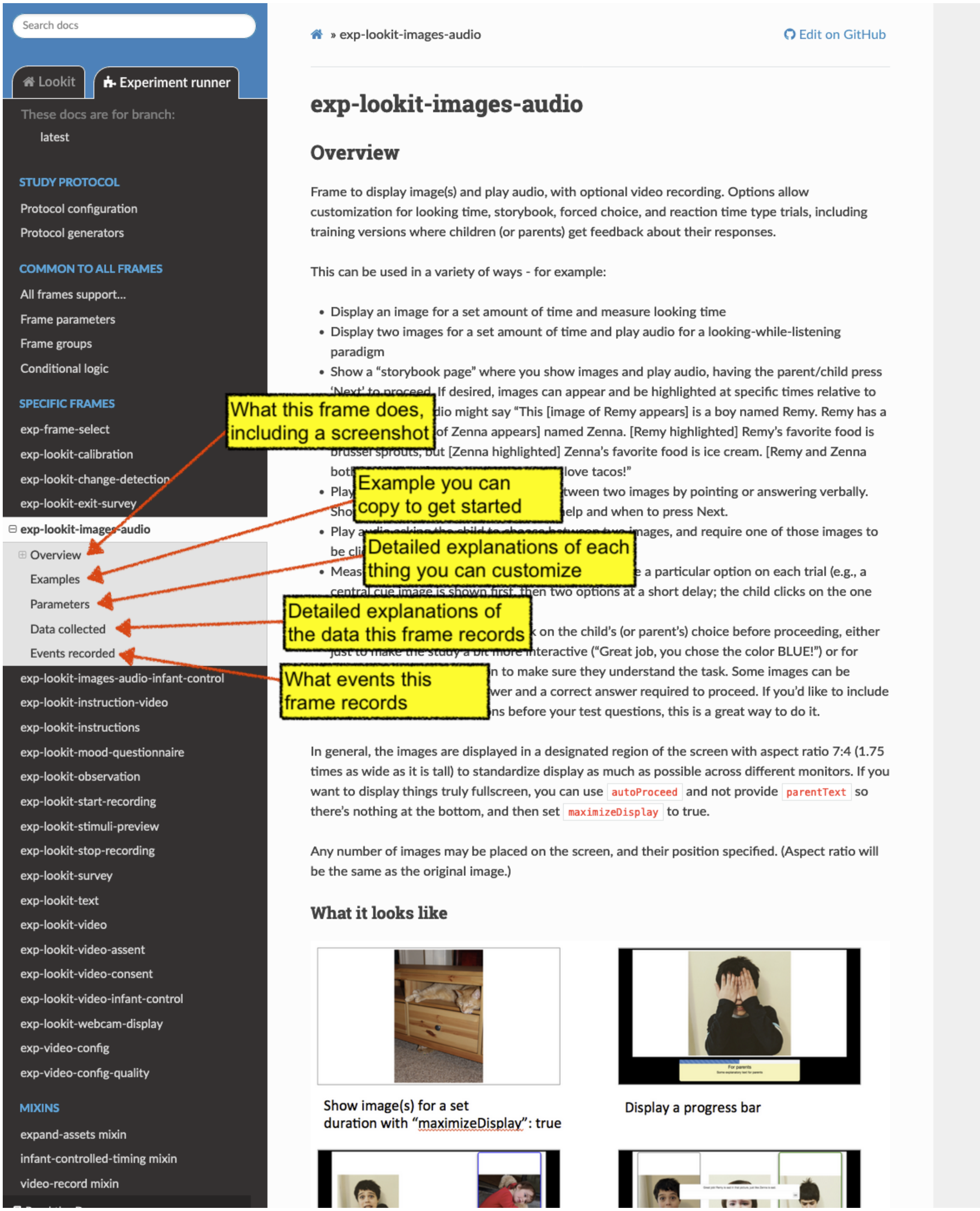
Using the frame documentation to learn more about frame-specific options

Each frame you define on Lookit has to have a property called `kind` which says what kind of frame it is. If you look through your study protocol, you'll see that

- the `exit-survey` frame has kind `exp-lookit-exit-survey`
- the `instructions` frame has kind `exp-lookit-instructions`
- the `video-config` frame has kind `exp-video-config`
- the `video-consent` frame has kind `exp-lookit-video-consent`
- the `storybook-causal` frame has kind `group` (this is a special kind of frame, documented here)
- the frames **within** the `storybook-causal` frame have kind `exp-lookit-images-audio` (this is added to each frame in the `frameList` as part of the `commonFrameProperties`)

In addition to this tutorial and the information in this documentation about how to set up a study on Lookit, there is detailed information available about each of the "frames" you can use in your Lookit study. In the experiment runner docs you can browse the options and learn about options for customizing each type of frame.

Let's take a look at the `exp-lookit-images-audio` documentation to see what options we have. Find it on the left sidebar and click on it. Here's what you'll see:



exp-lookit-images-audio

Overview

Frame to display image(s) and play audio, with optional video recording. Options allow customization for looking time, storybook, forced choice, and reaction time type trials, including training versions where children (or parents) get feedback about their responses.

This can be used in a variety of ways - for example:

- Display an image for a set amount of time and measure looking time
- Display two images for a set amount of time and play audio for a looking-while-listening paradigm
- Show a "storybook page" where you show images and play audio, having the parent/child press 'Next' to proceed. If desired, images can appear and be highlighted at specific times relative to … dio might say "This [image of Remy appears] is a boy named Remy. Remy has a … of Zenna appears] named Zenna. [Remy highlighted] Remy's favorite food is brussel sprouts, but [Zenna highlighted] Zenna's favorite food is ice cream. [Remy and Zenna bot… love tacos!"
- Play … tween two images by pointing or answering verbally. Sho… elp and when to press Next.
- Play audio asking the child to choose between two images, and require one of those images to be cli…
- Meas… e a particular option on each trial (e.g., a central cue image is shown first, then two options at a short delay; the child clicks on the one … k on the child's (or parent's) choice before proceeding, either just to make the study a bit more interactive ("Great job, you chose the color BLUE!") or for … n to make sure they understand the task. Some images can be … wer and a correct answer required to proceed. If you'd like to include … ns before your test questions, this is a great way to do it.

In general, the images are displayed in a designated region of the screen with aspect ratio 7:4 (1.75 times as wide as it is tall) to standardize display as much as possible across different monitors. If you want to display things truly fullscreen, you can use `autoProceed` and not provide `parentText` so there's nothing at the bottom, and then set `maximizeDisplay` to true.

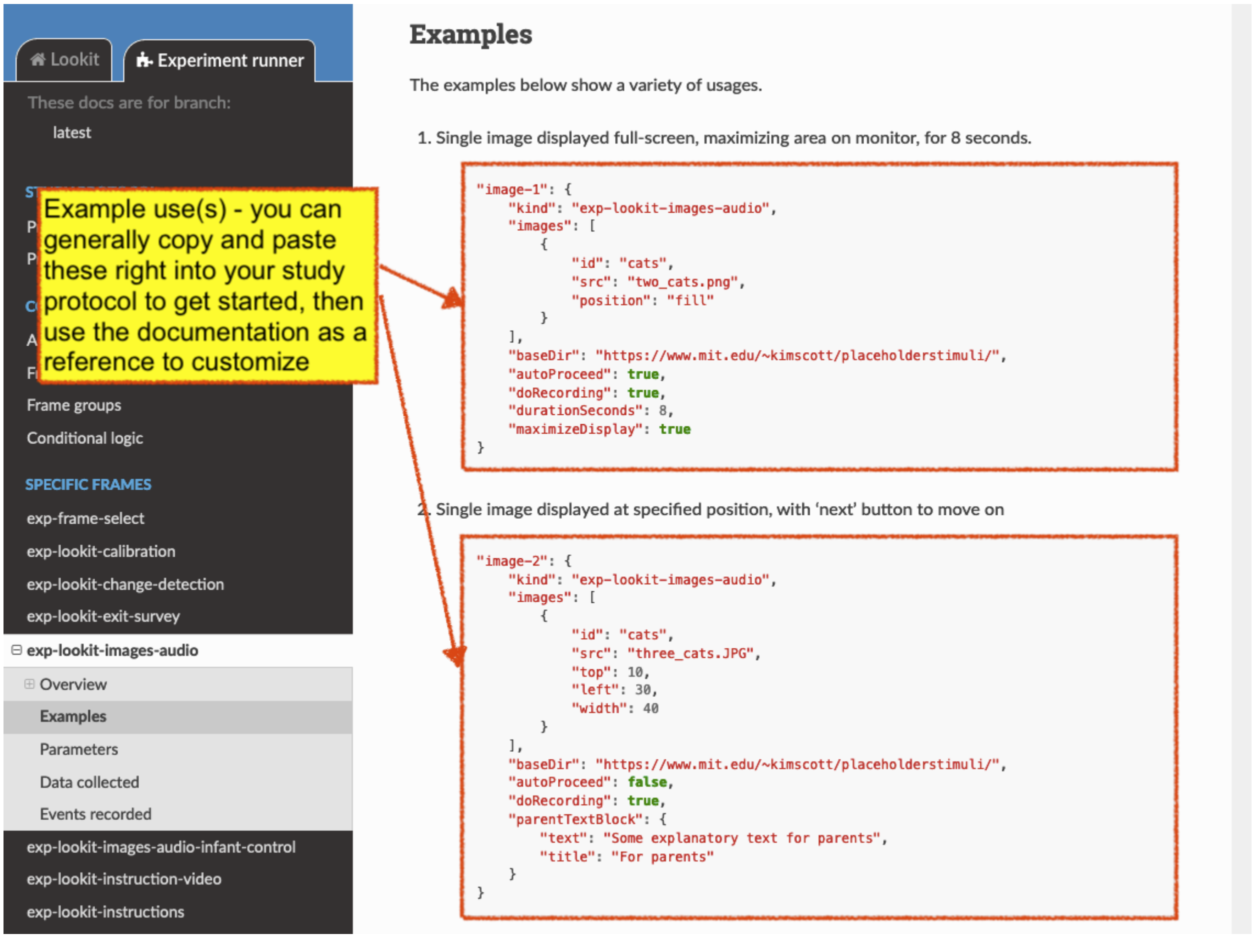
Any number of images may be placed on the screen, and their position specified. (Aspect ratio will be the same as the original image.)

What it looks like

Show image(s) for a set duration with "maximizeDisplay": true

Display a progress bar

Each frame documentation page has the same sections you can use to learn more about how to customize it, what data it collects, and so on. Near the top under "What it looks like", you can see a screenshot of the frame (or a collection of example screenshots). Under "Example" you'll find an example that you can generally copy and paste to get started:



Example use(s) - you can generally copy and paste these right into your study protocol to get started, then use the documentation as a reference to customize

Examples

The examples below show a variety of usages.

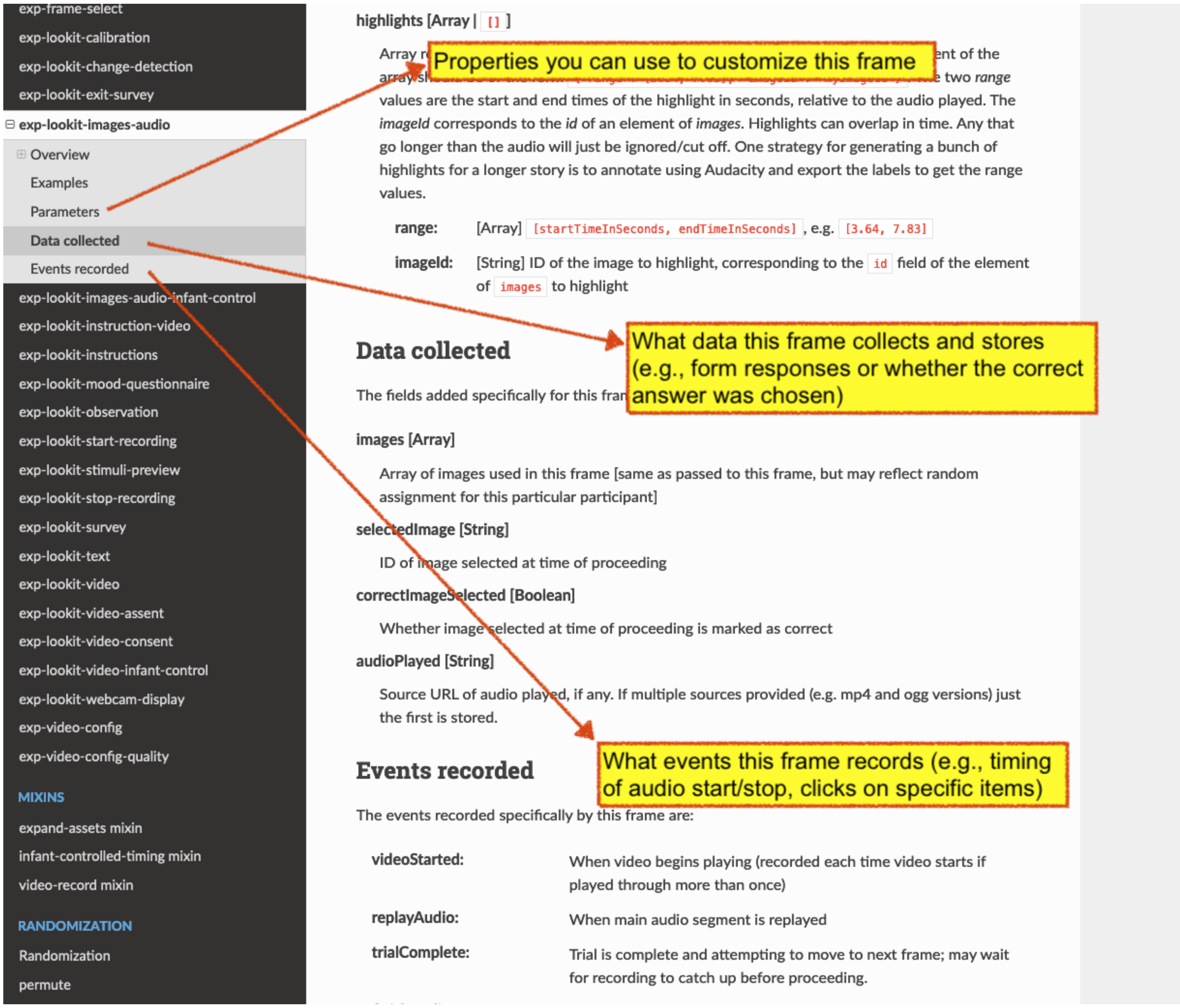
1. Single image displayed full-screen, maximizing area on monitor, for 8 seconds.

```
"image-1": {
    "kind": "exp-lookit-images-audio",
    "images": [
        {
            "id": "cats",
            "src": "two_cats.png",
            "position": "fill"
        }
    ],
    "baseDir": "https://www.mit.edu/~kimscott/placeholderstimuli/",
    "autoProceed": true,
    "doRecording": true,
    "durationSeconds": 8,
    "maximizeDisplay": true
}
```

2. Single image displayed at specified position, with 'next' button to move on

```
"image-2": {
    "kind": "exp-lookit-images-audio",
    "images": [
        {
            "id": "cats",
            "src": "three_cats.JPG",
            "top": 10,
            "left": 30,
            "width": 40
        }
    ],
    "baseDir": "https://www.mit.edu/~kimscott/placeholderstimuli/",
    "autoProceed": false,
    "doRecording": true,
    "parentTextBlock": {
        "text": "Some explanatory text for parents",
        "title": "For parents"
    }
}
```

The final three sections are “Parameters,” “Data collected,” and “Events recorded.”



highlights [Array | `[]`]

Array r ent of the array sh e two *range* values are the start and end times of the highlight in seconds, relative to the audio played. The *imageId* corresponds to the *id* of an element of *images*. Highlights can overlap in time. Any that go longer than the audio will just be ignored/cut off. One strategy for generating a bunch of highlights for a longer story is to annotate using Audacity and export the labels to get the range values.

range: [Array] `[startTimeInSeconds, endTimeInSeconds]`, e.g. `[3.64, 7.83]`

imageId: [String] ID of the image to highlight, corresponding to the `id` field of the element of `images` to highlight

Properties you can use to customize this frame

Data collected

The fields added specifically for this fran

What data this frame collects and stores (e.g., form responses or whether the correct answer was chosen)

images [Array]

Array of images used in this frame [same as passed to this frame, but may reflect random assignment for this particular participant]

selectedImage [String]

ID of image selected at time of proceeding

correctImageSelected [Boolean]

Whether image selected at time of proceeding is marked as correct

audioPlayed [String]

Source URL of audio played, if any. If multiple sources provided (e.g. mp4 and ogg versions) just the first is stored.

Events recorded

What events this frame records (e.g., timing of audio start/stop, clicks on specific items)

The events recorded specifically by this frame are:

videoStarted: When video begins playing (recorded each time video starts if played through more than once)

replayAudio: When main audio segment is replayed

trialComplete: Trial is complete and attempting to move to next frame; may wait for recording to catch up before proceeding.

Click on "Parameters" to see all the properties we can add to the frame definition in our protocol. You'll see that some of the things we can set are "audio," "autoProceed," "doRecording," "durationSeconds," "images," "parentTextBlock," and "showProgressBar." Each one includes an explanation of what it does and what format its value needs to be in.

Let's try changing the value of "autoProceed" on all our storybook pages. To do that we can change it within the "commonFrameProperties" in our study protocol:

```
"commonFrameProperties": {
     "kind": "exp-lookit-images-audio",
     "baseDir": "https://www.mit.edu/~kimscott/bunnystimuli/",
     "audioTypes": [
         "mp3",
         "ogg"
     ],
     "autoProceed": true, <-- change this from false to true!
     "doRecording": false,
     "parentTextBlock": {
         "css": {
             "font-size": "1.5em"
         },
         "text": "Please help keep your child's attention, but don't talk with him or␣
↪her about WHY Bunny might be getting a tummyache yet! Feel free to replay the audio␣
↪if your child was distracted.",
```

(continues on next page)

(continued from previous page)

```
        "title": "For parents"
    }
}
```

Save and refresh your preview, and see how the study works now. Instead of clicking on "next" to proceed after each storybook page, the study should automatically proceed to the next page! That's probably not what we actually want, so we can change it back after trying it out.

Counterbalance the test question

You may have noticed that we're asking children why Bunny has a tummyache - because of X or because of Y? But if kids tend to say X, we won't know if that's because they believe it's X or because they tend to go with the first option mentioned.

Let's set up to counterbalance the question that's asked! Again, at this point you don't need to understand all the details, let's just walk through what we'd do.

We're going to change our "storybook-causal" frame into what's called a randomizer frame, instead of just a group of frames. Find this section and make the changes indicated below:

```
"storybook-causal": {
    "kind": "group", <-- change this to "choice"
    "sampler": "random-parameter-set", <-- add this line!
    "frameList": [
        ... <--  almost everything in here can stay the same
        {
            "images": [
                {
                    "id": "storybookIllustration",
                    "src": "bunnya01.png", <-- but change this to "QUESTION_IMAGE"
                    "top": "0",
                    "left": "10",
                    "width": "80"
                }
            ],
            "audio": "bunnya01" <-- and change this to "QUESTION_AUDIO"
        },
        ...
    ],
    "commonFrameProperties": { <-- everything in here can stay the same
        ...
    },
    "parameterSets": [ <-- add this section!
       {
           "QUESTION_IMAGE": "bunnya01.png",
           "QUESTION_AUDIO": "bunnya01"
       },
       {
           "QUESTION_IMAGE": "bunnyb01.png",
           "QUESTION_AUDIO": "bunnyb01"
       }
   ]
}
```

Now when you try out the study, about half the time you'll hear "was it because of eating a sandwich, or feeling scared?" and the other half of the time you'll hear "was it because of feeling scared, or eating a sandwich?" Don't worry about the details yet - the important thing is just to understand that this is a sort of thing you can do relatively easily.

Congratulations! You've just finished setting up your first study. You've made lots of small changes to the study protocol and looked at how they affect what happens, and by now you're probably comfortable making a change, saving it, and previewing the study again.

1.25 4. A study from the ground up

Now that you've gotten your feet wet and are comfortable using Lookit's experimenter interface to modify your study protocol configuration, it's time to take a closer look at how to build your own study.

In this section, you'll build an example infant study "from the ground up," adding frames one at a time.

There's a fair amount of copying, pasting, and looking at the result in this section. Please bear with us - once you complete the tutorial, you'll be ready to set up your own study!

Piece-by-piece vs. starting from a template

When you go to create your actual studies, you will likely prefer to start by cloning one of the "templates" visible to you, rather than by building the study up piece-by-piece. But here we'll go through piece-by-piece so you have a solid understanding of how to add components as needed.

1.25.1 Introduction: intermodal matching study

Imagine you're looking to replicate the finding that infants can detect which moving face "goes with" a speech stream, an ability known as intermodal matching. In your study, babies will watch several short videos of two women's faces - one face on the left and one on the right. Both women are talking, but babies only hear the audio from one of them in each clip. (There are four trials, and who's talking is counterbalanced - babies either hear left, right, right, left audio or right, left, left, right audio.)

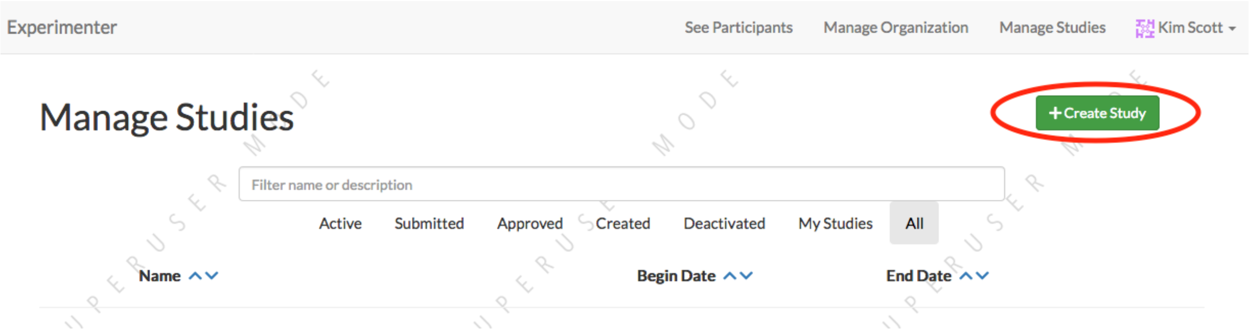
You plan to code the video collected on Lookit for preferential looking - whether the child is looking to the left of the screen, right of the screen, or away. Because you are eventually hoping to develop a measure that can be used to detect individual differences linked to social development, you are also including a short survey on parenting beliefs.

This is a real study that was run on Lookit!

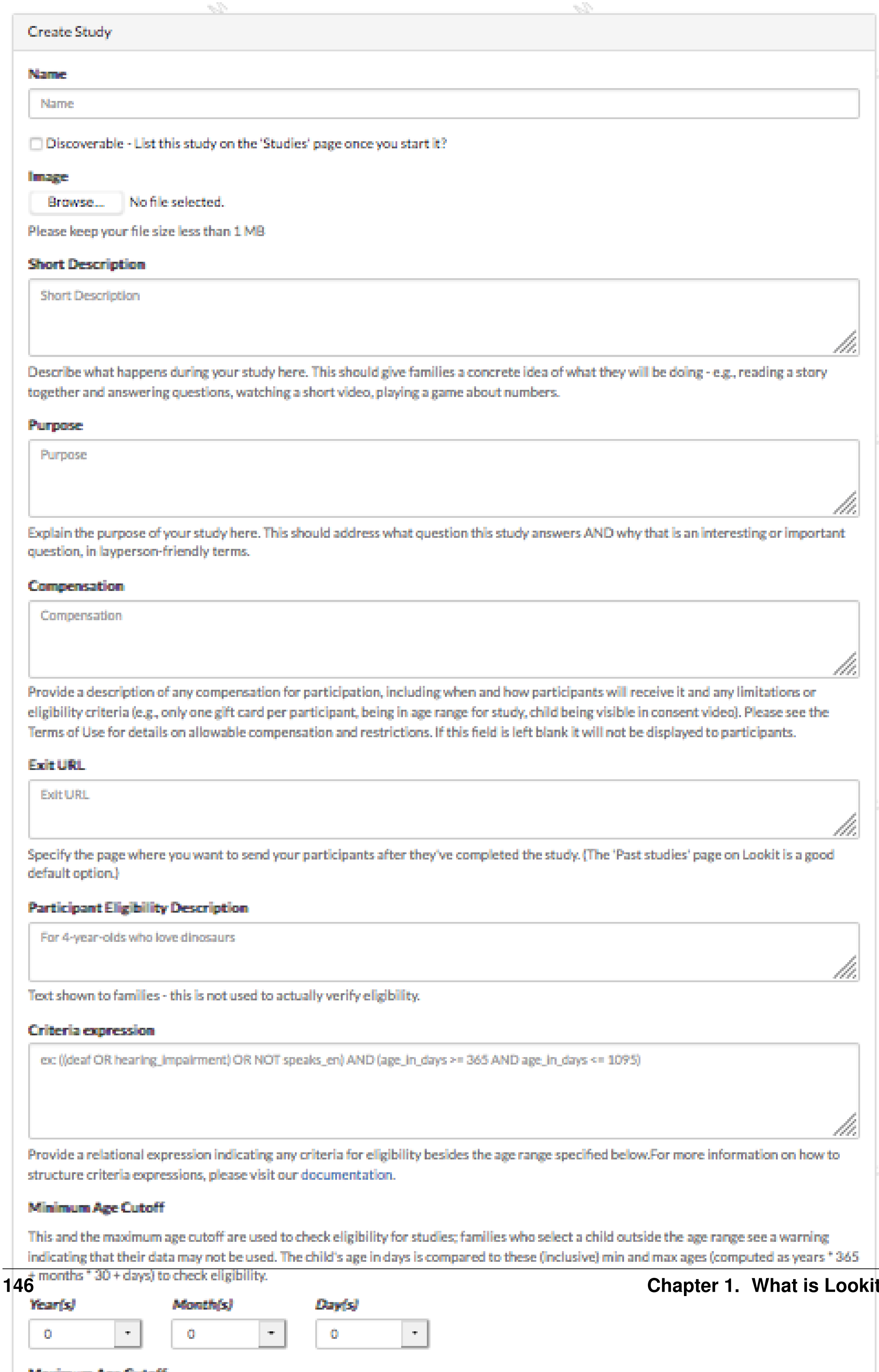
The study stimuli and protocol have been generously shared by Halie Olson and Rebecca Saxe for use as an example. Slight modifications have been made for teaching purposes.

1.25.2 Creating the study and filling out study fields

This time, instead of copying an existing study, we're going to create our own from scratch so that we see every field. Go to https://lookit.mit.edu/exp/studies/ and click the green "Create Study" button at top right:



This will bring you to a screen with a bunch of fields to fill out:



Create Study

Name

Name

□ Discoverable - List this study on the 'Studies' page once you start it?

Image

No file selected.

Please keep your file size less than 1 MB

Short Description

Short Description

Describe what happens during your study here. This should give families a concrete idea of what they will be doing - e.g., reading a story together and answering questions, watching a short video, playing a game about numbers.

Purpose

Purpose

Explain the purpose of your study here. This should address what question this study answers AND why that is an interesting or important question, in layperson-friendly terms.

Compensation

Compensation

Provide a description of any compensation for participation, including when and how participants will receive it and any limitations or eligibility criteria (e.g., only one gift card per participant, being in age range for study, child being visible in consent video). Please see the Terms of Use for details on allowable compensation and restrictions. If this field is left blank it will not be displayed to participants.

Exit URL

Exit URL

Specify the page where you want to send your participants after they've completed the study. (The 'Past studies' page on Lookit is a good default option.)

Participant Eligibility Description

For 4-year-olds who love dinosaurs

Text shown to families - this is not used to actually verify eligibility.

Criteria expression

ex: ((deaf OR hearing_impairment) OR NOT speaks_en) AND (age_in_days >= 365 AND age_in_days <= 1095)

Provide a relational expression indicating any criteria for eligibility besides the age range specified below.For more information on how to structure criteria expressions, please visit our documentation.

Minimum Age Cutoff

This and the maximum age cutoff are used to check eligibility for studies; families who select a child outside the age range see a warning indicating that their data may not be used. The child's age in days is compared to these (inclusive) min and max ages (computed as years * 365 + months * 30 + days) to check eligibility.

Year(s)	Month(s)	Day(s)
0	0	0

In a separate tab, open up the documentation about these fields for fuller explanations of what each one should contain: *Setting study details*. Below is the study-specific information you'll need to fill out each field.

Name Enter "Look and Listen" here - or another name for the study if you have a cute or catchy idea!

Discoverable Check the box so that the study would show up to participants on the Studies page, and families with eligible children would get emails inviting them to participate

Share preview Check the box so other researchers can preview your study without being added as collaborators

Image You can download a screenshot of the stimuli to use here.

Short Description Here's an example description of what happens during this study - you can copy or edit it. "Your child will watch several short videos of two people saying nonsense syllables. The sound will match just one of the people. We want to see which face your baby chooses to look at!"

Purpose Here's a draft of a description of the purpose of the study, but it's very general - it could apply to a lot of related studies. Edit it so that it explains the specific question this study asks and why the answer matters - without getting too technical. "Babies can learn language by seeing, hearing, and feeling. We want to better understand how babies pay attention to what they see and hear when people are speaking to them."

Compensation Here's the actual compensation description that was used - you can copy it: "After you participate, we'll email you a $4 Amazon gift card as a thank-you. (One gift card per child; child must be in the age range for the study.)"

Exit URL After the study let's send families to their study history, where they can see their videos right away! Use "https://lookit.mit.edu/studies/history/"

Participant Eligibility Description "For babies ages 4-18 months"

Criteria expression This study focuses on children born at or near full term: require that the child was born at 37+ weeks gestation. (Refer to the documentation to see how, and to learn about what other types of eligibility criteria you can set up here!)

Minimum Age Cutoff 3 months, 29 days (this equals 119 days, which is the youngest that a "4 month old" by the calendar can be. This avoids confusing parents who see a warning when trying to participate with their just-turned-four-month-old born in February)

Maximum Age Cutoff 1 year, 7 months, 4 days (this is a generous cutoff, again to avoid confusion from parents of kids who aren't quite 19 months old yet)

Duration "5 minutes" (you can also test this)

Researcher Contact Information Enter "<Your Name> (contact: <your email and/or phone number>)"

Protocol configuration Leave this blank for now

Experiment runner type Choose the default and leave `Experiment runner version (commit SHA)` blank.

Click the green "Create study" button at the bottom of the form to save all your work! You've got all the study metadata set… now all that's left is to write your study protocol.

1.25.3 Adding each frame

Now we're going to build out the study protocol configuration, one piece at a time. Here's the basic outline of this study. It follows a basic pattern you can also see described here: *Example Lookit study outline*.

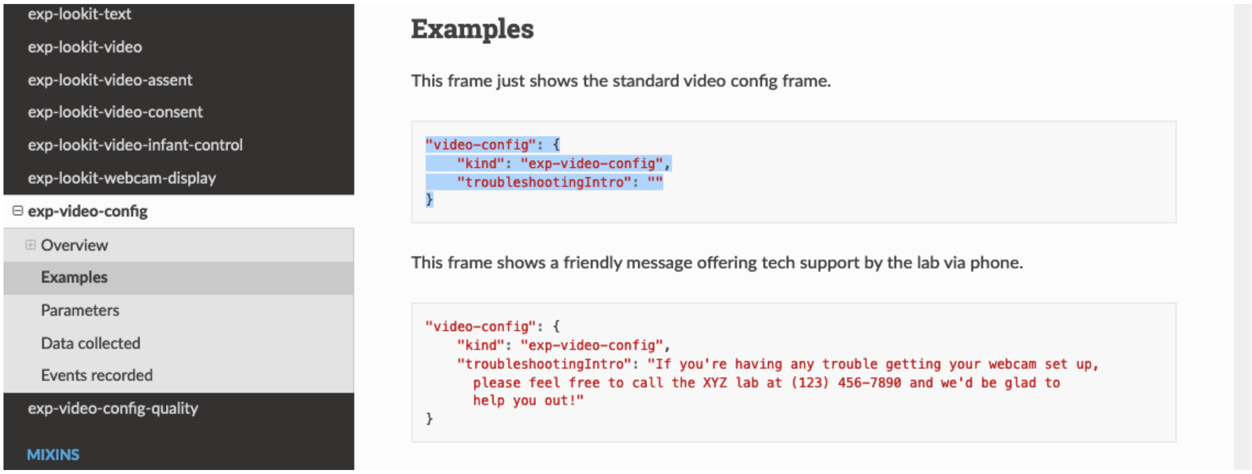
1. A "setup" frame to guide the family through getting their webcam set up
2. A video consent frame where the parent makes a verbal statement of informed consent
3. An intro frame giving the parent an overview of what will happen during the study

4. A stimuli preview frame, giving parents the option to review stimuli ahead of time
5. Some instructions about what to do during the study
6. Test trials where babies will see videos that show two women talking (one on either side of the screen) but only the audio from one speaker
7. A short survey about parenting beliefs
8. A standard "exit survey" where parents select a video privacy level

1. Setup

We'll start with a standard setup frame called "exp-video-config".

Take a look at the documentation for this frame here. You'll see a screenshot of what it looks like, and under "Examples" you'll see examples of how to define this frame in your study protocol:



Examples

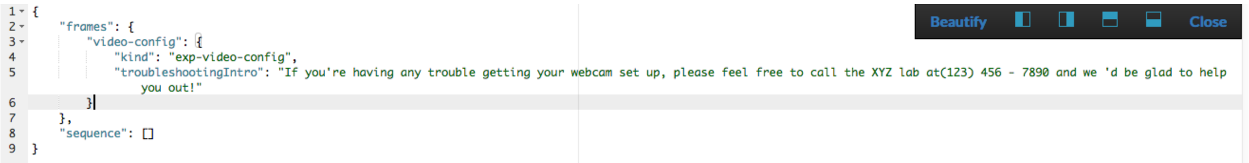
This frame just shows the standard video config frame.

```
"video-config": {
    "kind": "exp-video-config",
    "troubleshootingIntro": ""
}
```

This frame shows a friendly message offering tech support by the lab via phone.

```
"video-config": {
    "kind": "exp-video-config",
    "troubleshootingIntro": "If you're having any trouble getting your webcam set up,
      please feel free to call the XYZ lab at (123) 456-7890 and we'd be glad to
      help you out!"
}
```

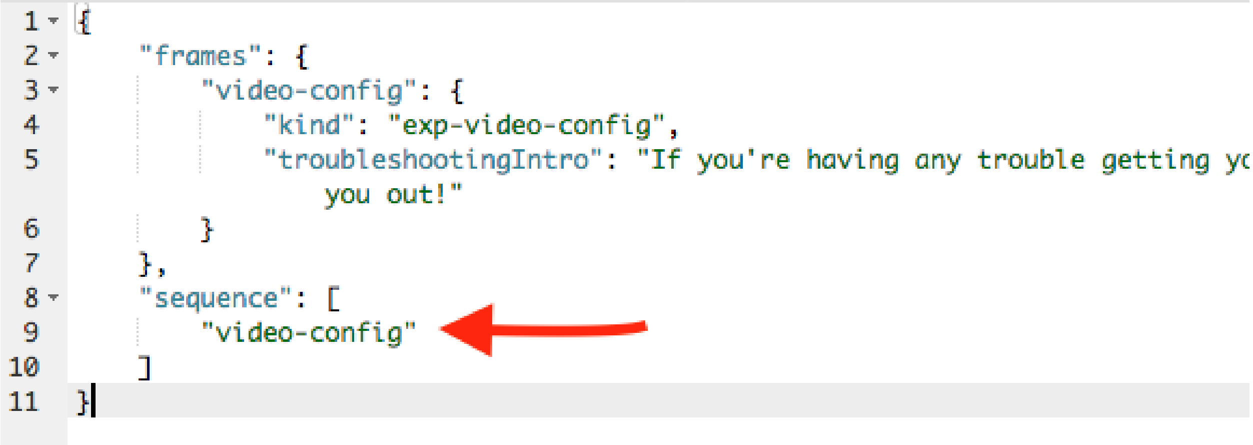
Copy one of the definitions of the "video-config" frame (`"video-config":  { ... }`, as shown highlighted above), and open up your study's protocol editor. Paste this into the "frames" value, like this:



```
{
    "frames": {
        "video-config": {
            "kind": "exp-video-config",
            "troubleshootingIntro": "If you're having any trouble getting your webcam set up, please feel free to call the XYZ lab at(123) 456 - 7890 and we 'd be glad to help
                you out!"
        }
    },
    "sequence": []
}
```

Edit the text if you'd like, so it references your own lab and an appropriate contact method!

That defines a frame that's now available for us to use. In order to actually use it, add it to your "sequence" as well:



```
{
    "frames": {
        "video-config": {
            "kind": "exp-video-config",
            "troubleshootingIntro": "If you're having any trouble getting yo
                you out!"
        }
    },
    "sequence": [
        "video-config"
    ]
}
```

The key for your frame can be whatever you want

There's nothing magical about the "video-config" key given to this frame - you can change it to whatever you want, as long as the key in `frames` matches what you call it in `sequence`. Just don't put an underscore in it (see the experiment runner documentation).

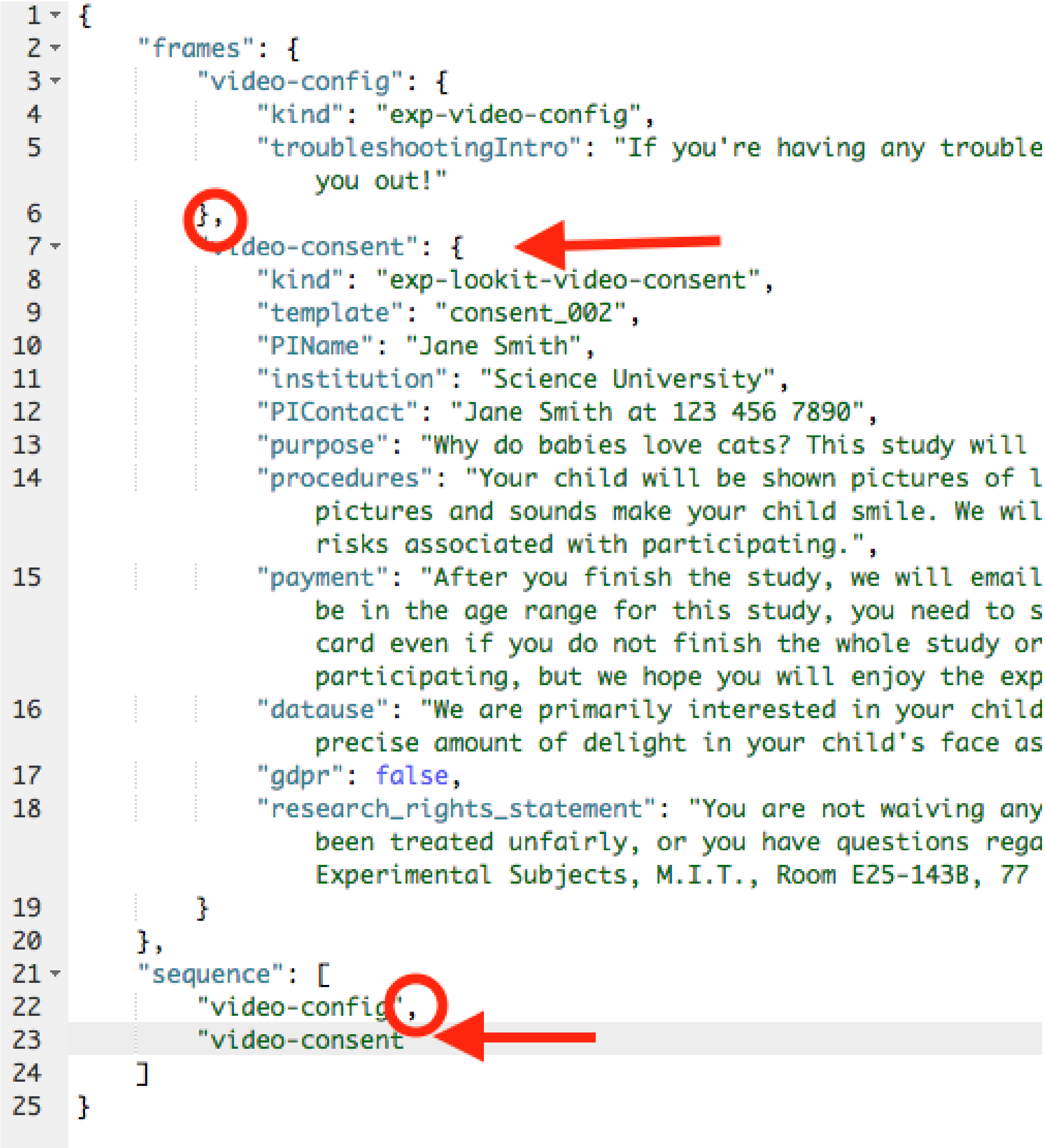
Close the editor, save your protocol, and preview your study. Make sure to click "build experimenter runner" if you have not already done so; you will be emailed when it is built and then you can preview your study. You should see the setup frame, looking just like the screenshot in the docs.

2. Consent

Now that your participants have their webcam set up, the very first thing you need to do - before starting any study procedures - is collect informed consent. Consent frames are treated somewhat specially: you will only see any data from participants who get through your consent page, and videos collected on the consent page will be available for you to review and confirm before you can access the remaining data from the corresponding sessions.

Unless you receive specific permission from Lookit, you'll be asked to use the standard video consent (and/or assent) frames to keep the experience for participants consistent.

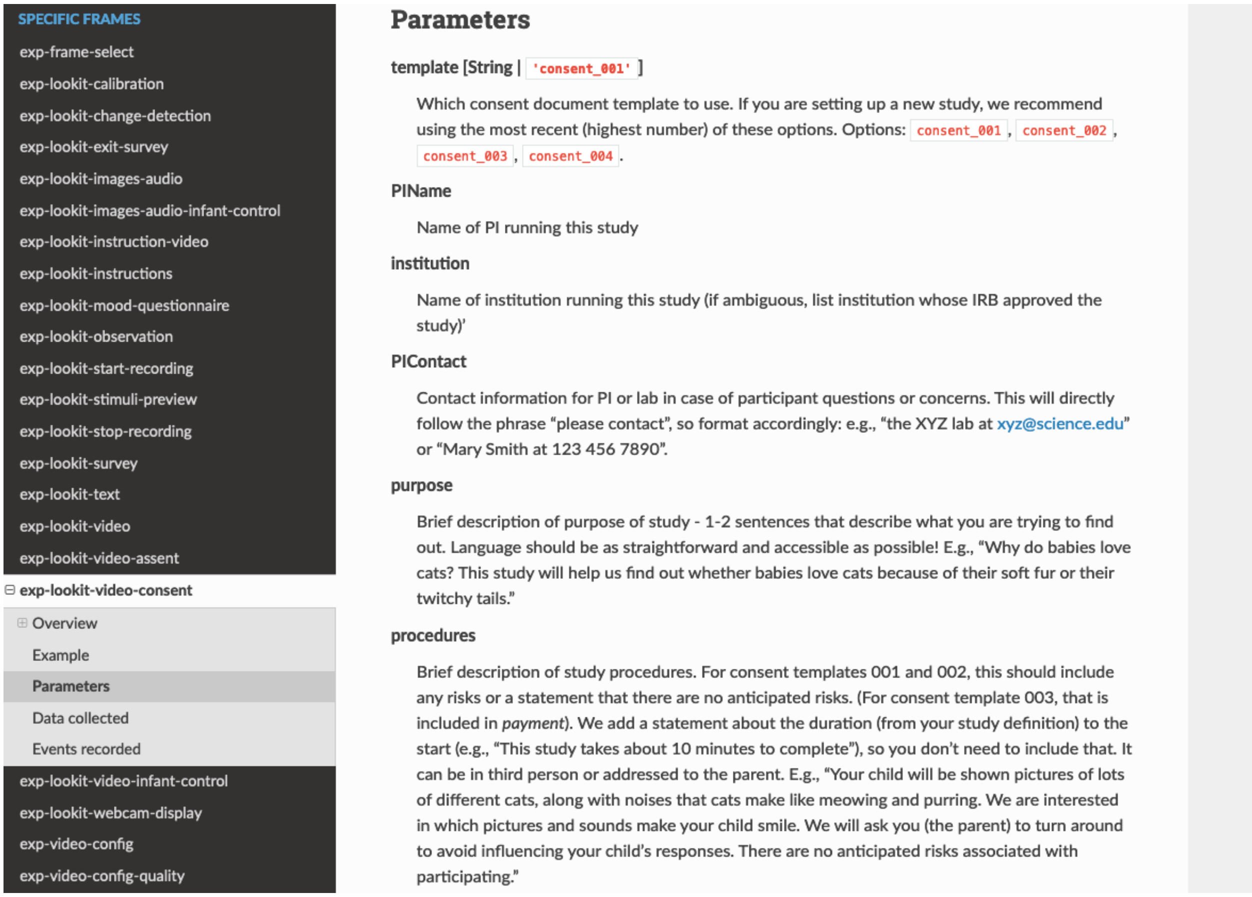
This study is for babies, so we don't need to collect child assent, just parental consent. Go to the frame docs and select 'exp-lookit-video-consent' on the sidebar. Just like you did for the setup frame, copy the sample frame definition (`"video-consent": {...}`) and add it to your study protocol frames and sequence, like this:



```
{
    "frames": {
        "video-config": {
            "kind": "exp-video-config",
            "troubleshootingIntro": "If you're having any trouble
                you out!"
        },
        "video-consent": {
            "kind": "exp-lookit-video-consent",
            "template": "consent_002",
            "PIName": "Jane Smith",
            "institution": "Science University",
            "PIContact": "Jane Smith at 123 456 7890",
            "purpose": "Why do babies love cats? This study will
            "procedures": "Your child will be shown pictures of l
                pictures and sounds make your child smile. We wil
                risks associated with participating.",
            "payment": "After you finish the study, we will email
                be in the age range for this study, you need to s
                card even if you do not finish the whole study or
                participating, but we hope you will enjoy the exp
            "datause": "We are primarily interested in your child
                precise amount of delight in your child's face as
            "gdpr": false,
            "research_rights_statement": "You are not waiving any
                been treated unfairly, or you have questions rega
                Experimental Subjects, M.I.T., Room E25-143B, 77
        }
    },
    "sequence": [
        "video-config",
        "video-consent"
    ]
}
```

Note that you will need to add commas between the previous and new items in both “frames” and “sequence”, as circled above.

Save and preview again. Now when you click “Next” from the video config page, you’ll see a consent page. The sample text is pretty silly, though! Change each of the following fields to more appropriate text for this study, substituting in your own information for the things shown in `<brackets like this>`. For more information on what each of these fields is, click on “Parameters” in the frame documentation:



Parameters

template [String | `'consent_001'`]

Which consent document template to use. If you are setting up a new study, we recommend using the most recent (highest number) of these options. Options: `consent_001`, `consent_002`, `consent_003`, `consent_004`.

PIName

Name of PI running this study

institution

Name of institution running this study (if ambiguous, list institution whose IRB approved the study)'

PIContact

Contact information for PI or lab in case of participant questions or concerns. This will directly follow the phrase "please contact", so format accordingly: e.g., "the XYZ lab at xyz@science.edu" or "Mary Smith at 123 456 7890".

purpose

Brief description of purpose of study - 1-2 sentences that describe what you are trying to find out. Language should be as straightforward and accessible as possible! E.g., "Why do babies love cats? This study will help us find out whether babies love cats because of their soft fur or their twitchy tails."

procedures

Brief description of study procedures. For consent templates 001 and 002, this should include any risks or a statement that there are no anticipated risks. (For consent template 003, that is included in *payment*). We add a statement about the duration (from your study definition) to the start (e.g., "This study takes about 10 minutes to complete"), so you don't need to include that. It can be in third person or addressed to the parent. E.g., "Your child will be shown pictures of lots of different cats, along with noises that cats make like meowing and purring. We are interested in which pictures and sounds make your child smile. We will ask you (the parent) to turn around to avoid influencing your child's responses. There are no anticipated risks associated with participating."

PIName "<Your Name>"

datause "We are primarily interested in your child's looking behavior. A research assistant will watch your video to measure the precise amount of time looking at the screen."

payment "You will be emailed a $4 Amazon gift card for participating in this study, no matter what your child does during the experiment, as long as your child is in the age range of our study and has not participated in our study in the past 30 days."

purpose "The purpose of this study is to better understand how much infants at different ages prefer to look at talking faces that are synchronized with what they hear compared to talking faces that are not synchronized with what they hear."

PIContact "<Your Name> at <your email> or <your phone number>"

procedures "For this study, your child will watch short videos (about 20 seconds long). For each video, there will be two faces on the screen that may be speaking nonsense syllables – something like "La mu ba." The audio of the nonsense syllables matching only one of the two faces will be played. We are curious which face your baby prefers to look at – the one matching the audio or the one that doesn't match the audio. We ask that you close your eyes or hold your baby over your shoulder during this experiment so that your behavior doesn't influence where your baby looks. Before each video, you will hear a chime while an image moves on the screen to get your baby's attention. You may then hear about 20 seconds of the nonsense syllables while the faces appear on the screen. The entire experiment should take less than 5 minutes. You may be asked to fill out a short survey at the end of the experiment."

risk_statement "We do not expect any risks to participating in this study."

include_databrary true

template “consent_002”

institution “<your institution>”

gdpr false

research_rights_statement “You are not waiving any legal claims, rights or remedies because of your participation in this research study. If you feel you have been treated unfairly, or you have questions regarding your rights as a research subject, you may contact <your IRB information>.”

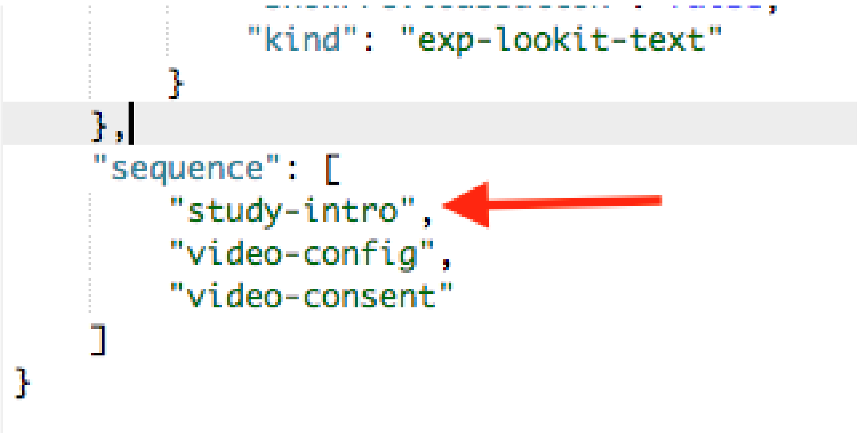
Save your protocol and take another look at the preview. Congratulations! You’ve got the start of your study set up, with a valid consent form that lets the family record a statement of informed consent.

3. Intro

Here we’ll use a simple text frame just to give parents an overview about what’s going to be happening in the study.

Go to the frame documentation, and select the “exp-lookit-text” frame. Just like before, add the example to your study protocol, putting the frame definition for “study-intro” in your “frames” object and adding “study-intro” to your “sequence” list.

For convenience, this time, let’s put “study-intro” FIRST in the sequence, so that when we preview our study it’s easy for us to see the changes we make to customize the text on this frame:



```
            "kind": "exp-lookit-text"
        }
    },
    "sequence": [
        "study-intro",
        "video-config",
        "video-consent"
    ]
}
```

Save your protocol and go ahead and preview your study. You should see a simple text frame first. Let’s change the `blocks` value to show an appropriate overview for this study: copy and paste the section below to replace the existing `"blocks": [...]` piece:

```
"blocks": [
     {
         "emph": true,
         "text": "Your child does not need to be with you until the videos begin.␣
↪First, let's go over what will happen!",
         "title": "Overview of the 'Look and Listen' study"
     },
     {
         "text": "During this study, your baby will watch videos of talking faces␣
↪while we record where he or she chooses to look."
     },
     {
         "text": "You’ll have a chance to preview the videos ahead of time. After␣
↪reading the instructions you’ll start the experiment when you and your baby are␣
↪ready."
```

(continues on next page)

(continued from previous page)

```
        },
        {
            "text": "The video section will take about 3 minutes."
        },
        {
            "text": "After the videos, you will answer a few final questions. Then you
↪'re all done!"
        }
  ]
```

Save and preview again to see your changes.

4. Stimulus preview

Especially if you need parents blind to stimuli and so you ask them to turn around or close their eyes, it's generally best practice to offer them an opportunity to preview any images, audio, or video that their child will be shown during the study. This lets them check that they don't think anything is objectionable or inappropriate for their child - e.g., interactions they find to be violent, or images of something that might interact with a child's phobia. From a practical standpoint, it also greatly decreases the temptation to "peek" at the stimuli during the study out of curiosity or concern.

We'll use the frame type "exp-lookit-stimuli-preview" here to offer parents the opportunity to preview stimuli, and record while they preview if so. You can look up the properties they accept in the frame documentation, but since you're already getting the hang of using the frame documentation to start from an example, this time you can just copy and paste the following definition into `frames`:

```
"video-preview": {
     "kind": "exp-lookit-stimuli-preview",
     "stimuli": [
         {
             "caption": "For each trial, there will be two women on the screen␣
↪speaking nonsense syllables. Only the audio for one of the videos will be played at␣
↪a time. Here's an example.",
             "video": "INSERT_EXAMPLE_VIDEONAME_HERE"
         }
     ],
     "baseDir": "https://www.mit.edu/~kimscott/intermodal/",
     "videoTypes": [
         "webm",
         "mp4"
     ],
     "blocks": [
         {
             "text": "During the videos, we'll ask that you hold your child over your␣
↪shoulder like this, so that you're facing away from the screen.",
             "image": {
                 "alt": "Father holding child looking over his shoulder",
                 "src": "INSERT_SRC_URL_HERE"
             }
         },
         {
             "text": "The reason we ask this is that your child is learning from you␣
↪all the time. Even if he or she can't see where you're looking, you may␣
↪unconsciously shift towards one side or the other and influence your child's␣
↪attention. We want to make sure we're measuring your child's preferences, not yours!
↪"
```

(continues on next page)

(continued from previous page)

```
        },
        {
            "text": "If you'd like to see an example of a video your child will be␣
↪shown, you can take a look ahead of time now. It's important that you watch the␣
↪video without your child, so that the videos will still be new to them."
        }
    ],
    "skipButtonText": "Skip preview",
    "previewButtonText": "Preview a video (my child can't see the screen)",
    "showPreviousButton": true
}
```

There are a few stimuli above that you'll need to insert. You can see all the stimuli you might need for this study at https://www.mit.edu/~kimscott/intermodal/.

- For the example video, where it says `"INSERT_EXAMPLE_VIDEONAME_HERE"`, take a look in the mp4 directory to find an example video (any example with sound is fine). You only need to give the filename without extension, like "abba1", because we're already telling the exp-lookit-stimuli-preview frame to use a "base directory" for this study and expect certain video types. You can learn more here: *Directory structure*.
- For the image of the father holding his child over his shoulder, take a look in the img directory, and insert the full path ("https://www.mit.edu/~kimscott/. . . ") to the file you want to use.

Then make sure to also add "video-preview" to your `sequence`. You can put this at the start of the sequence to make it easy to see right away. Save and take a look at the preview!

Warning about putting frames at the start to preview them quickly

Putting a frame at the start of the `sequence` is a good way to quickly keep previewing it, but it won't work if the frame is displayed full-screen. That's because web browsers won't let websites make themselves fullscreen without a "user interaction event," like clicking on a button. Whenever you switch into full-screen mode, the frame beforehand needs to have a "next" button or similar.

To rapidly preview a full-screen frame, just put it second in your `sequence`, after e.g. a text frame that doesn't require you do do anything but click Next.

5. Instructions

Almost done with the preparations! We're just going to give particpants one more frame with directions so these are fresh in their minds. This time we'll use an exp-lookit-instructions frame, which allows showing a fairly flexible combination of text, audio, video, and the user's own webcam. Here's a starting point for the frame to add:

```
"final-instructions": {
    "kind": "exp-lookit-instructions",
    "blocks": [
        {
            "text": "The video section will take about 3 minutes to complete. After␣
↪that, you will be able to select a level of privacy for your data."
        },
        {
            "title": "Study overview",
            "listblocks": [
                {
                    "text": "To get your baby's attention, first they will see a␣
↪moving shape and hear a chime. "
```

(continues on next page)

(continued from previous page)

```
                },
                {
                    "text": "Then your baby will watch four videos, each about 20␣
↪seconds long."
                }
            ]
        },
        {
            "title": "During the videos",
            "listblocks": [
                {
                    "text": "Please face away from the screen, holding your infant␣
↪so they can look over your shoulder. Please don't look at the videos yourself--we␣
↪may not be able to use your infant’s data in that case.",
                    "image": {
                        "alt": "Father holding child looking over his shoulder",
                        "src": "https://s3.amazonaws.com/lookitcontents/exp-physics/
↪OverShoulder.jpg"
                    }
                },
                {
                    "text": "Don’t worry if your baby isn’t looking at the screen␣
↪the entire time! Please just try to keep them facing the screen so they can look if␣
↪they want to."
                }
            ]
        },
        {
            "title": "Pausing and stopping",
            "listblocks": [
                {
                    "text": "If your child gets fussy or distracted, or you need to␣
↪attend to something else for a moment, you can pause the study by pressing the␣
↪space bar."
                },
                {
                    "text": "If you need to end the study early, try closing the␣
↪window or tab and you should see an 'exit' option pop up. You’ll be prompted to␣
↪note any technical problems you might be experiencing and to select a privacy level␣
↪for your videos."
                }
            ]
        },
        {
            "text": "Please turn the volume up so it's easy to hear but still␣
↪comfortable.",
            "title": "Test your audio",
            "mediaBlock": {
                "text": "You should hear 'Ready to go?'",
                "isVideo": false,
                "sources": [
                    {
                        "src": "MP3_SOURCE_HERE",
                        "type": "audio/mp3"
                    },
                    {
                        "src": "OGG_SOURCE_HERE",
```

(continues on next page)

(continued from previous page)

```
                         "type": "audio/ogg"
                     }
                 ],
                 "mustPlay": true,
                 "warningText": "Please try playing the sample audio."
             }
         }
     ],
     "nextButtonText": "Start the videos! \n (You'll have a moment to turn around.)"
 }
```

The snippet above sets up several sections ("blocks") with bulleted lists of information. (For a real study you might also consider splitting this frame into several frames - a study overview, "during the videos" directions, pausing and stopping, and the audio test. More things to click through, but less text on the page.)

As in the preview, there are some stimuli you need to add! Browse the audio files here to find an mp3 and ogg version of a "ready to go!" audio clip that you can use to have parents test their audio. Insert the full paths where it says "MP3_SOURCE_HERE" and "OGG_SOURCE_HERE". Why multiple versions of the same files? This helps make sure that the media will work across various computer setups.

Once you've added this frame to your `frames` and to your `sequence`, check out how it looks. Note that because you've set `mustPlay` to `true` in the block about testing your audio, you can't proceed to the next frame until you've played it! This is to make sure that participants don't start the video section without their sound on. If they do, (a) the study won't work because the baby needs to be able to hear the sound, and (b) they're going to be very confused because they won't hear the audio instructions that tell them what's going on, when it's time to turn back around, etc.

6. Test trial(s)

Finally, the meat of the study! Right now, we're just going to set up a single test trial to see how it works. Once we have a complete mockup of the study, we'll add the counterbalancing and the rest of the trials.

For this study, we're going to use the fairly flexible "exp-lookit-video" frame, which lets us play a video. Please skim the frame documentation now for an overview of how it works.

Copy and paste the following frame to your `frames` (removing the comments that look like `<-- TEXT HERE ``) and then add "example-test-trial" to your ``sequence`. Because this frame is shown full-screen, you should put it after at least one other frame to test it out (e.g., after your instructions frame) rather than making it the first frame. This is because your web browser won't let something go full-screen unless you take an action to trigger that (like pressing the "next" button).

```
"example-test-trial":
   {
         "kind": "exp-lookit-video",

         "video": {
             "loop": false,
             "position": "fill",
             "source": "abba1" <-- TEST VIDEO OF TWO WOMEN TALKING
         },
         "backgroundColor": "white",
         "autoProceed": true,

         "requireVideoCount": 1,  <-- PLAY THROUGH THE TEST VIDEO ONE TIME
         "doRecording": true,

```

(continues on next page)

(continued from previous page)

```
        "frameOffsetAfterPause": 0,
        "pauseAudio": "<INSERT HERE>", <-- INSERT THE NAME (NO EXTENSION) OF AUDIO␣
↪TO PLAY UPON PAUSING THE STUDY HERE
        "pauseVideo": "<INSERT HERE>", <-- INSERT THE NAME OF THE VIDEO TO SHOW␣
↪WHILE THE STUDY IS PAUSED HERE
        "unpauseAudio": "<INSERT HERE>", <-- INSERT THE NAME OF AUDIO TO PLAY WHEN␣
↪THE STUDY IS UN-PAUSED

        "baseDir": "https://www.mit.edu/~kimscott/intermodal/",
        "audioTypes": [
            "ogg",
            "mp3"
        ],
        "videoTypes": [
            "webm",
            "mp4"
        ]
  }
```

Again, you will need to browse the available audio and video files to select appropriate stimuli to insert where indicated above.

Save your protocol and take a look at what happens. You should see two women talking, and be able to tell that the audio matches just one of them!

7. Survey

After the test trials, you plan to include the Early Parenting Attitudes Questionairre (See Hembacher & Frank, https://psyarxiv.com/hxk3d/). It's a bit long, so for the purposes of this tutorial we're just going to include a few questions from it. Copy and paste the following frame into `frames`, and add "epaq-survey" to your `sequence` - you know the drill. This uses the "exp-lookit-survey" frame type.

```
"epaq-survey": {
    "kind": "exp-lookit-survey",
    "formSchema": {
        "schema": {
            "type": "object",
            "title": "This is an optional survey that will take a few minutes to␣
↪complete. Please indicate how much you agree with the following statements using a␣
↪0-6 scale with 0 being 'I do not agree' and 6 being 'strongly agree.'",
            "properties": {
                "Q1": {
                    "enum": [
                        "0 (Do not agree)",
                        "1",
                        "2",
                        "3",
                        "4",
                        "5",
                        "6 (Strongly agree)"
                    ],
                    "title": "Children should be comforted when they are scared or␣
↪unhappy.",
                    "required": false
                },
```

(continues on next page)

(continued from previous page)

```
                "Q2": {
                    "enum": [
                        "0 (Do not agree)",
                        "1",
                        "2",
                        "3",
                        "4",
                        "5",
                        "6 (Strongly agree)"
                    ],
                    "title": "It's important for parents to help children learn to␣
↪deal with their emotions.",
                    "required": false
                }

            }
        },
        "options": {
            "fields": {
                "Q1": {
                    "type": "radio",
                    "removeDefaultNone": true
                },
                "Q2": {
                    "type": "radio",
                    "removeDefaultNone": true
                }
            }
        }
    }
  }
```

Save your protocol and take a look at the preview. You should see a simple form with two questions and some intro text, and (since nothing's required) you should be able to proceed even if you don't answer the questions.

You don't need to understand all the syntax above - but even if it looks pretty opaque, you can probably see the basic structure. There are two questions Q1 and Q2 defined in "properties," with some corresponding additional information under "options." Each one has some actual question text (the "title"), some options from 0 to 6, and will be shown as radio buttons.

Go ahead and try adding the next question (call it "Q3"):

"Parents should pay attention to what their child likes and dislikes."

It will have the same format and possible answers as the others. You can copy and paste the information about "Q2" under both "properties" and "options" and just edit it!

8. Exit survey

Finally, to wrap up our study we need to include an "exp-lookit-exit-survey" frame. (This is required of all Lookit studies to keep the experience for parents fairly consistent.) This is where parents have an option to choose how you may share their video, if at all, and to give you some feedback if they want to. It's also where you'll provide some "debriefing" information, just like you might when chatting with the family after they came into the lab. There are more guidelines about what your debriefing should contain under *the sample study outline*.

You guessed it - copy and paste the frame below into `frames` in your protocol, and add "exit-survey" to your `sequence`. Put the frames in your `sequence` in order and try out the entire study!

```
"exit-survey": {
         "kind": "exp-lookit-exit-survey",
         "debriefing": {
             "text": "You and your baby are helping us to better understand how the␣
↪preference for visual/auditory synchrony in speech develops over the first 18␣
↪months of life. Babies vary in the amount of time they choose to look at the
↪'synchronized' speaker compared to the 'unsynchronized' speaker - there's no right␣
↪or wrong preference! We are interested in how much babies' preferences differ at␣
↪various ages. If you'd like, you can even participate with your baby again next␣
↪month!\n\nTo thank you for your participation, we'll be emailing you a $4 Amazon␣
↪gift card - this should arrive in your inbox within the next week after we confirm␣
↪your consent video and check that your child is in the age range for this study.␣
↪(If you don't hear from us by then, feel free to reach out!) If you participate␣
↪again with another child in the age range, you'll receive one gift card per child.␣
↪You will also receive another gift card if you participate again with this child if␣
↪it has been at least one month since the last time this child participated.",
             "title": "Thank you for participating in our study!"
         }
     }
```

Finally, pretend that your baby has fussed out partway through, and try pressing ctrl-X or F1 during the study. You should see a dialogue appear and if you choose to leave the study, you'll be taken to the last frame - which is now, appropriately, your exit survey. Hooray!

1.25.4 Add some initial audio instructions

You may have noticed that the test trial starts right away, without giving the parent much of a chance to get ready! Let's add some friendly audio instructions for that transition. We'll use another `exp-lookit-video` frame. It'll be similar to the test trial, except we'll show the attentiongrabber video (looping) while we play a separate audio file:

```
"announce-trial-1":
   {
         "kind": "exp-lookit-video",

         "video": {
             "loop": true, <-- HAVE THIS VIDEO LOOP
             "top": 40, <-- INSTEAD OF "position": "fill" we specify this one should␣
↪be smaller and centered!
             "left": 45,
             "width": 10,
             "source": "attentiongrabber"
         },
         "audio": {
             "loop": false,
             "source": "video_1_HO_intro" <-- THE AUDIO FILE TO PLAY
         },
         "backgroundColor": "white",
         "autoProceed": true,

         "requireVideoCount": 0,
         "requireAudioCount": 1, <-- PLAY THROUGH THE AUDIO ONCE, DON'T WORRY ABOUT␣
↪VIDEO
         "doRecording": false, <-- WE DON'T REALLY NEED A RECORDING OF THIS

         "frameOffsetAfterPause": 0,
         "pauseAudio": "<INSERT HERE>", <-- INSERT THE NAME (NO EXTENSION) OF AUDIO␣
↪TO PLAY UPON PAUSING THE STUDY HERE
```

(continues on next page)

(continued from previous page)

```
         "pauseVideo": "<INSERT HERE>", <-- INSERT THE NAME OF THE VIDEO TO SHOW␣
↪WHILE THE STUDY IS PAUSED HERE
         "unpauseAudio": "<INSERT HERE>", <-- INSERT THE NAME OF AUDIO TO PLAY WHEN␣
↪THE STUDY IS UN-PAUSED

         "baseDir": "https://www.mit.edu/~kimscott/intermodal/",
         "audioTypes": [
             "ogg",
             "mp3"
         ],
         "videoTypes": [
             "webm",
             "mp4"
         ]
  }
```

Add this to your list of frames and insert it in the sequence just before the first test trial.

Planning your audio instructions

You want your audio instructions to be as concise as possible, but still friendly and complete. Figuring out all the different audio

“audioSources”: “video_1_HO_intro”, <– WHAT AUDIO TO PLAY AS AN ANNOUNCEMENT

1.25.5 Add a calibration trial

We also want to add a quick calibration section where an attention-grabber pops back and forth on the screen (so that your coders will be able to verify they can see the child looking back and forth). Let’s add that after the “announce-trial-1” frame and before the test trial.

Lookit provides a custom calibration frame exp-lookit-calibration that you can use for this purpose:

```
"calibration-with-video": {
    "kind": "exp-lookit-calibration",

    "baseDir": "https://www.mit.edu/~kimscott/intermodal/",
    "audioTypes": [
        "ogg",
        "mp3"
    ],
    "videoTypes": [
        "webm",
        "mp4"
    ],

    "calibrationLength": 2000, <-- MAKE EACH CALIBRATION SEGMENT 2 S LONG
    "calibrationPositions": [
        "center",
        "left",
        "right"
    ],
    "calibrationAudio": "<INSERT HERE>", <-- CHOOSE AUDIO TO PLAY EACH TIME THE␣
↪CALIBRATION VIDEO MOVES
    "calibrationVideo": "attentiongrabber"
}
```

Add this to your list of frames and insert it in the sequence just before the first test trial. You can play around with `calibrationPositions` to see how you can show the spinning ball in a different sequence of locations.

1.25.6 Set up counterbalancing

Your plan for this study is actually to have four test trials. Either the audio will come from the left speaker, right speaker, right speaker, left speaker; or it will come from right speaker, left speaker, left speaker, right speaker. Before each test trial there will be a short "announcement" letting the parent know which trial number it is, also set up with an exp-lookit-video frame.

To do this sort of counterbalancing, the simplest approach is to use a special class of frame called a "randomizer." At the time your study protocol is interpreted in order to display the study to your participant, the randomizer frame will make some (random) selections. There are a variety of randomization options available on Lookit, which you can browse here. For our study, we will use the fairly general-purpose "random-parameter-set" randomizer, which you can read more about in those frame docs if you're curious.

We will be providing the randomizer with three main things: a list of frames (`frameList`), a set of properties all the frames should share, just for convenience (`commonFrameProperties`), and a list of sets of parameters to substitute in (`parameterSets`)- the randomizer will choose one of these at the start of the study and do the substitution.

Let's start with just a skeleton of our test trials frame:

```
"test-trials": {
     "kind": "choice",
     "sampler": "random-parameter-set",
     "frameList": [],
     "parameterSets": [],
     "commonFrameProperties": {}
 }
```

For each of the four test trials, we're going to want to use an exp-lookit-video frame with some of the same basic properties, so let's put those in `commonFrameProperties`:

```
"commonFrameProperties": {
     "kind": "exp-lookit-video",

     "baseDir": "https://www.mit.edu/~kimscott/intermodal/",
     "audioTypes": [
         "ogg",
         "mp3"
     ],
     "videoTypes": [
         "webm",
         "mp4"
     ],

     "backgroundColor": "white",
     "autoProceed": true,

     "pauseAudio": "pause_HO",
     "pauseVideo": "attentiongrabber",
     "unpauseAudio": "return_after_pause_HO"
 }
```

Now let's expand that `frameList`.

We'll do the first announcement and calibration trial separately. Then we'll have:

- trial 1,
- announcement 2 (just the attention-getter while someone says “Video 2”)
- trial 2
- announcement 3
- trial 3
- announcement 4
- trial 4
- a final announcement where we tell parents they can turn back around.

The things that will vary each frame are:

- the actual test videos
- the audio for the announcements
- whether to do recording
- whether to require the video or audio to play through

```
"frameList": [
     {
         "video": {
             "loop": false,
             "position": "fill",
             "source": "abba1"
         },
         "doRecording": true,
         "requireVideoCount": 1,
         "requireAudioCount": 0
     },
     {
         "video": {
             "loop": true,
             "top": 40,
             "left": 45,
             "width": 10,
             "source": "attentiongrabber"
         },
         "audio": {
             "loop": false,
             "source": "video_02_HO"
         },
         "doRecording": false,
         "requireAudioCount": 1,
         "requireVideoCount": 0
     },
     {
         "video": {
             "loop": false,
             "position": "fill",
             "source": "abba2"
         },
         "doRecording": true,
         "requireVideoCount": 1,
         "requireAudioCount": 0
```

(continues on next page)

(continued from previous page)

```
    },
    {
        "video": {
            "loop": true,
            "top": 40,
            "left": 45,
            "width": 10,
            "source": "attentiongrabber"
        },
        "audio": {
            "loop": false,
            "source": "video_03_HO"
        },
        "doRecording": false,
        "requireAudioCount": 1,
        "requireVideoCount": 0
    },
    {
        "video": {
            "loop": false,
            "position": "fill",
            "source": "abba3"
        },
        "doRecording": true,
        "requireVideoCount": 1,
        "requireAudioCount": 0
    },
    {
        "video": {
            "loop": true,
            "top": 40,
            "left": 45,
            "width": 10,
            "source": "attentiongrabber"
        },
        "audio": {
            "loop": false,
            "source": "video_04_HO"
        },
        "doRecording": false,
        "requireAudioCount": 1,
        "requireVideoCount": 0
    },
    {
        "video": {
            "loop": false,
            "position": "fill",
            "source": "abba4"
        },
        "doRecording": true,
        "requireVideoCount": 1,
        "requireAudioCount": 0
    },
    {
        "video": {
            "loop": true,
            "top": 40,
```

(continues on next page)

(continued from previous page)

```
             "left": 45,
             "width": 10,
             "source": "attentiongrabber"
         },
         "audio": {
             "loop": false,
             "source": "all_done_HO"
         },
         "doRecording": false,
         "requireAudioCount": 1,
         "requireVideoCount": 0
     }
 ]
```

You can go ahead and try this out with the empty parameterSets and see the whole study!

That's great, but it hard-codes in the stimuli for this counterbalancing condition. Actually, sometimes we want to use "abba[N]", and other times we want to use "baab[N]". That's just what this randomizer is for! We'll stick in placeholders for the video sources like this:

```
"frameList": [
     {
         "video": {
             "loop": false,
             "position": "fill",
             "source": "VIDEO1" <-- THIS IS THE PLACEHOLDER FOR THE VIDEO FILE WE'LL␣
↪USE
         },
         "doRecording": true,
         "requireVideoCount": 1,
         "requireAudioCount": 0
     },
     {
         "video": {
             "loop": true,
             "top": 40,
             "left": 45,
             "width": 10,
             "source": "attentiongrabber"
         },
         "audio": {
             "loop": false,
             "source": "video_02_HO"
         },
         "doRecording": false,
         "requireAudioCount": 1,
         "requireVideoCount": 0
     },
     {
         "video": {
             "loop": false,
             "position": "fill",
             "source": "VIDEO2"
         },
         "doRecording": true,
         "requireVideoCount": 1,
         "requireAudioCount": 0
```

(continues on next page)

(continued from previous page)

```
    },
    {
        "video": {
            "loop": true,
            "top": 40,
            "left": 45,
            "width": 10,
            "source": "attentiongrabber"
        },
        "audio": {
            "loop": false,
            "source": "video_03_HO"
        },
        "doRecording": false,
        "requireAudioCount": 1,
        "requireVideoCount": 0
    },
    {
        "video": {
            "loop": false,
            "position": "fill",
            "source": "VIDEO3"
        },
        "doRecording": true,
        "requireVideoCount": 1,
        "requireAudioCount": 0
    },
    {
        "video": {
            "loop": true,
            "top": 40,
            "left": 45,
            "width": 10,
            "source": "attentiongrabber"
        },
        "audio": {
            "loop": false,
            "source": "video_04_HO"
        },
        "doRecording": false,
        "requireAudioCount": 1,
        "requireVideoCount": 0
    },
    {
        "video": {
            "loop": false,
            "position": "fill",
            "source": "VIDEO4"
        },
        "doRecording": true,
        "requireVideoCount": 1,
        "requireAudioCount": 0
    },
    {
        "video": {
            "loop": true,
            "top": 40,
```

(continues on next page)

(continued from previous page)

```
             "left": 45,
             "width": 10,
             "source": "attentiongrabber"
         },
         "audio": {
             "loop": false,
             "source": "all_done_HO"
         },
         "doRecording": false,
         "requireAudioCount": 1,
         "requireVideoCount": 0
     }
 ]
```

Then we also need to define the `parameterSets`, which will let us define values for `VIDEO1`, `VIDEO2`, etc. The `parameterSets` value is a list of sets; each set should define all the values we need for one condition:

```
"parameterSets": [
     {
         "VIDEO1": "abba1",
         "VIDEO2": "abba2",
         "VIDEO3": "abba3",
         "VIDEO4": "abba4"
     },
     {
         "VIDEO1": "baab1",
         "VIDEO2": "baab2",
         "VIDEO3": "baab3",
         "VIDEO4": "baab4"
     }
 ]
```

By default, half of kids will be assigned to the first set, and half to the second. That's what we want here, so we don't need to do anything more. But if you wanted to assign more kids to one condition (for instance, because you had enough data from one condition) or assign kids to conditions based on their ages, you could also provide a `parameterSetWeights` property for this randomizer.

Putting it all together, you should now have a test-trials randomizer frame with `frameList`, `parameterSets`, and `commonFrameProperties` defined. Give it a try - a few times! Sometimes you should see one condition, and sometimes the other. (If you really want to see how a particular parameterSet works, that's another reason to provide the `parameterSetWeights` - e.g., you could set that to `[1, 0]` to only use the first set.)

1.25.7 About creating and hosting your stimuli

In this example, you used stimuli already posted for you at <*www.mit.edu/~kimscott/intermodal/*>. When you create your own studies, note that you'll in general need to create and host your own stimuli. Because researchers' needs here will vary substantially, stimulus creation and hosting is outside the scope of this tutorial. However, resources are available under *Preparing stimuli*.

1.25.8 About communicating with parents

One of the biggest challenges we have observed for researchers transitioning to running studies online isn't technical: it's the difference in communication medium. Instead of talking with parents face-to-face–answering the questions they bring up and tuning your explanations based on how they respond–you now have to anticipate the wide variety of ways people might be confused or concerned. And you're communicating, generally using text, with sleep-deprived parents at home who are holding squirming infants on their laps (and perhaps trying to keep siblings occupied too).

It is HARD, for instance, to write a few-sentence "elevator pitch" for your study that really explains - in an accessible way! - what your question is and why it's interesting. For most scientists, this is substantially harder than regular scientific writing.

It's also very hard to condense text instructions into something concise, non-condescending, and complete. (The examples above aren't perfect!) You may realize there's more than you thought to explain about how to do your study (e.g. how to avoid biasing the child), and that you want to add some training trials with feedback, video instructions, or more detailed audio instructions.

So this is a general note of caution: yes, in some respects it's easy to "throw a study up on Lookit." (Or at least we're trying to make it easy!) But it will likely take you longer than you expect to go from "We know exactly how we want our study to work" to "We're up and running," in large part because of these sorts of details. And it is absolutely worth putting in the time to come up with a study protocol that doesn't just "work" but is clear and easy to follow for parents - not least because we're all sharing the same subject pool and reputation as a fun place to do studies.

1.25.9 Using the documentation to learn about more advanced features

We hope that working through some examples has been helpful, but the Lookit documentation goes beyond just the tutorial! You can explore using the sidebar on the left to view detailed guides to preparing your study (including advanced topics not covered in this tutorial), managing your data, and developing your own custom frames. We recommend using the search function within the documentation, which ensures your results come only from the current, up-to-date version of the docs, rather than any archived older versions that might pop up on Google.

1.26 5. Study protocol exercises

Now that you've walked through the process of creating your own studies, it's time to try out what you've learned! At the end of this section, you will:

- be confident in your ability to independently set up and troubleshoot your study
- be comfortable finding information about advanced features to answer the questions "How do I do X" or "Is it possible to do X"
- have a feel for the challenges to expect in writing your instructions and preparing your stimuli

Remember, feel free to reach out in the Slack workspace if you have any trouble with these exercises.

1.26.1 1. Add another test trial

The intermodal matching study ("Look and Listen") currently includes four test trials; in each, two women are seen speaking, but only the audio from one speech stream is included. You would like to include a silent test trial as well, where the two women are seen speaking but there is no sound, so that you can measure children's baseline visual preferences for the two speakers. (You plan to use this in some exploratory work, looking at whether it might be possible to adjust for individual baseline preferences for a more sensitive measurement of audio-based shifts in attention.)

Add a silent test trial before the other four test trials.

- You can use the video at https://www.mit.edu/~kimscott/intermodal/mp4/silent.mp4 and https://www.mit.edu/~kimscott/intermodal/webm/silent.webm. Use the same silent video for both counterbalancing conditions.
- Put the calibration segment at the start of this trial (instead of keeping it at the start of the original first trial).

- Make sure the intro audio makes sense - this is now trial 1 and should include the "getting started" audio. There are now a total of five trials; you can use the "video_05_HO" audio files for the last trial intro.
- Adjust the overview and instructions to reflect the new protocol.

Note

Watch out - this is probably going to be confusing to parents, because they'll turn around and the video will "start" but they won't hear anything! Try to make sure they'll expect the first video to be silent.

1.26.2 2. Split an instructions page into more manageable pieces

The *"final-instructions"* frame of your intermodal matching study is quite a bit for a participant to read for the first time while holding a baby.

First, pare it down as much as you can. Then, split it across 3-4 frames (each with type *"exp-lookit-instructions"*). Make sure the "next" button on each frame says something appropriate (they shouldn't all say "start the videos").

Extra credit

Do you think it's easier or harder to understand the instructions this way? What else might you do to make the study more family-friendly? Share your thoughts on the #researchers Slack channel.

1.26.3 3. Add another experimental condition

In your first study ("My Awesome Tutorial Study"), children hear a story containing statistical evidence about a psychosomatic tummyache. At the end of the story, they're asked why Bunny's tummy hurts - because of eating a sandwich, or because of feeling scared? We made sure to counterbalance the order of those options, but if children do answer that the tummyache is because of feeling scared, we won't know whether they used the evidence in the story or just thought that was more likely all along.

Add a baseline condition, where instead of the whole story, children will only hear about the last day. Right now you have a sequence of test trials with the following images (and audio): *bunny01, bunny02, bunny11, bunny12, bunny13, bunny14, bunny15, bunny16, QUESTION_IMAGE/QUESTION_AUDIO, bunnyend01*.

First, take a look at the randomization options listed here and think about how you might set this up.

1.26.4 4. Make the study debriefing condition-dependent

Now that there are two pretty different conditions in your causal inference study, you may want to give personalized debriefing information to families based on which condition they were in!

Move the "exit-survey" into the "select" randomizer you're using for the storybook pages, and change the debriefing text to say "THIS WAS THE CAUSAL EVIDENCE CONDITION." Add a second version to the list of frameOptions with the debriefing text "THIS WAS THE BASELINE CONDITION." (Please note that this would NOT actually be good debriefing text!) Edit the frames used in each condition so that participants will see only the correct exit-survey for their condition. Try it out and make sure everything lines up. Make sure to remove the original "exit-survey" frame from the "sequence" in your study JSON!

1.26.5 5. Troubleshoot a frame that doesn't work as expected

In either of your studies, try adding the following frame to the "frames" in your study protocol, and add *"video-assent"* right after the consent frame in your *"sequence"*. This frame is intended to collect child assent (in addition to parental consent in the consent frame) for older children. There should be three pages to look through, but there are some errors in the frame specification and it will not work as written.

Preview the study to see what's wrong and edit the configuration for this frame until it works as intended. You will need to reference the documentation for the exp-lookit-video-assent frame.

Reminder

You can use the Javascript console in your web browser to look for error messages that might be relevant!

```
"video-assent": {
    "kind": "exp-lookit-video-assent",
    "pages": [
        {
            "text": "In this study, you'll see a lot of pictures of shapes.",
            "audio": "sample_01",
            "imgSrc": "square.png",
            "altText": "Some shapes"
        },
        {
            "audio": "We will have some questions for you about what shapes you see.",
            "imgSrc": "tall.png"
        },
        {
            "showWebcam": true,
            "textBlocks": [
                {
                    "text": "During the study, your webcam will record a video of you.
↪ We will watch this video later so we can write down your answers to the questions."
                }
            ]
        }
    ],
    "baseDir": "https://www.mit.edu/~kimscott/placeholderstimuli/",
    "videoTypes": [
        "webm",
        "mp4"
    ],
    "minimumAgeToAssent": 7,
    "participationQuestion": "Do you want to participate in this study?"
}
```

1.26.6 6. Find guidance on an advanced feature

You'd like to include a training section where parents are asked a multiple-choice question about how the experiment works before beginning, and if they get it wrong, they're directed to an additional video overview before getting started. Find the section of the documentation you would refer to for details about how to do this.

1.26.7 7. Draft a parent-facing study description

Write a *Purpose* field for a Lookit study. The study can be:

- Something you're planning to run on Lookit
- Something you've run in your lab before, or a favorite study from another lab that you're familiar with
- Or if you don't have anything in mind: try writing a purpose field for a study about infant-directed speech preference.

Post your draft description of the study purpose in the Lookit Slack workspace (#researchers) for feedback (yes, really!).

Look for other tutorial participants' drafts to give feedback on too! As you read, consider:

- Is it clear what the research question is?
- If the lab ran a follow-up study, would the same description probably cover it?
- Is it clear why the research question matters? ("X has not been studied before" is not a reason something matters.)

1.27 6. Managing study data

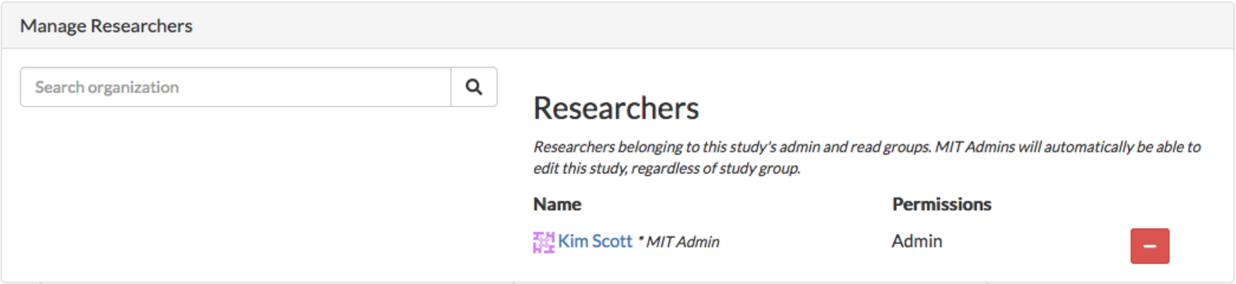
Now that you have a good handle on how to set up the study protocol you want, it's time to look at how to manage your study: controlling who has access to what, starting and stopping data collection, reviewing and downloading your data, and contacting your participants.

1.27.1 Managing access to your study: add a collaborator

Rarely will you be working completely alone! Usually you will want multiple people to have access to any particular study: you may have a few people working together to get the protocol just right, as well as several RAs checking consent and sending feedback to participants.

We very strongly recommend this model rather than sharing credentials for a lab-wide account. This way, each individual can get permissions on just the studies they actually need access to - not everything your lab has ever done. When temporary staff like undergrad RAs move on, you can just remove them from the study, instead of distributing a new password to everyone remaining in the lab. Plus, there is logging built into Lookit that keeps track of things like who did what when - including who approved consent recordings - that may be useful to you.

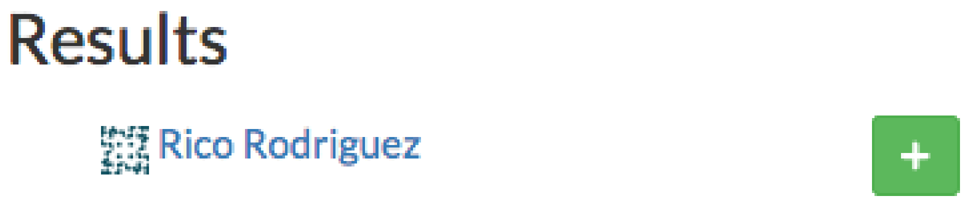
Try it out! Add another person to your tutorial study. On the study page in the Experimenter interface, scroll down to "Manage Researchers":



Researchers belonging to this study's admin and read groups. MIT Admins will automatically be able to edit this study, regardless of study group.

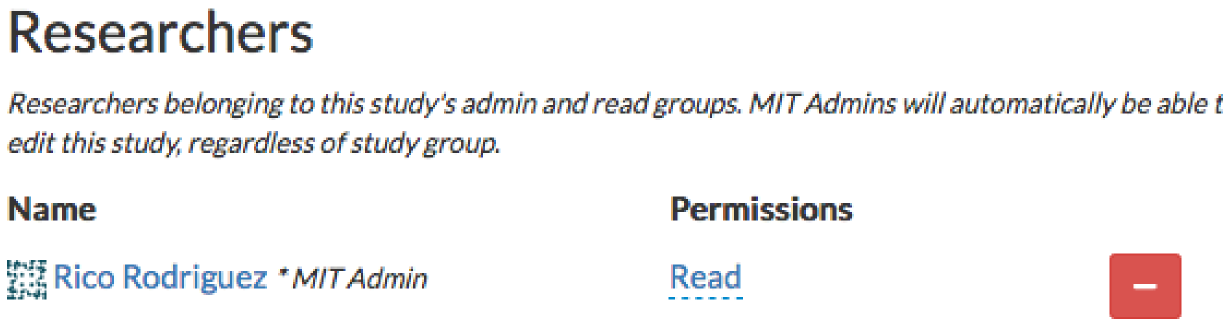
Name	Permissions
Kim Scott *MIT Admin*	Admin

In the search box, type in the first few letters of someone you want to add, and press Enter. (If you don't know anyone else using Lookit, feel free to add Lookit staff Kim Scott or Rico Rodriguez to your study!) Click the green "+" button to add them to your study:


Results

Rico Rodriguez

They will show up on the right with “Preview” permissions initially:


Researchers

Researchers belonging to this study's admin and read groups. MIT Admins will automatically be able t edit this study, regardless of study group.

Name	Permissions
Rico Rodriguez **MIT Admin*	Read

This means they can see your study in the Experimenter interface, but not see any participant data or make any changes. You can click on “preview” for a drop-down menu to give them different permissions if you want. (*See this section* for much more detailed information about the different roles available.) Or you can click the red “-” button to remove them again.

Great! Now you know how to give someone else access to your study so you can collaborate.

1.27.2 Updating the code your study uses

Another thing you’ll probably need to do eventually is set your study to use an updated version of the underlying Lookit frameplayer code.

Remember when we had to “build an experiment runner” so we could preview the study? (*You can review that here.*) That build process took the version of the frameplayer code we specified and bundled it up into a little container for our study to run in. That container includes all the information Lookit needs about what frames are available to use and how they work.

As you fine-tune your study, you will be making lots of edits to your study protocol, saying exactly what stimuli each frame should use, in what order, etc. But the study protocol is still interpreted by that same application. If at some point you want to take advantage of bug fixes, video recording improvements, new frames that have been added to the standard Lookit code, etc., you’ll need to tell Lookit to use the new version and build a fresh experiment runner.

One way to think about your current experiment runner is as a Lego set; it has certain types of building blocks that allow you to customize your project with the pieces you have available. But Lego is always making new blocks with interesting shapes and new affordances. If you want access to building blocks beyond what you had in your original set, you can get access to the new and improved set of blocks by rebuilding your experiment runner.

Try it out now! Follow the directions in *Updating the frameplayer code for your study* to update your tutorial study to use the most recent version of the Lookit frameplayer.

By design, updating the code shouldn’t break anything that currently works - you shouldn’t need to change your study protocol! However, it is important to always preview your study after any update to double check, and report any problems you run into.

1.27.3 Understanding previewing vs. participating in a study

So far, we have tried out our studies via the "preview study" button on the study edit page. There are only a few differences between previewing and actually participating in a study:

- When you preview a study, there is an "is_preview" field of the data collected that's set to true - otherwise it's false. Data collected from previewing is marked when you view consent videos or individual responses, and this field is available in the all-response downloads.
- Only Lookit researchers with appropriate permissions can preview the study. (Either the researcher needs to have read permissions for the study, or the study needs to be set to have a shared preview - then any researcher can access it.) Anyone with a child registered on Lookit can participate in a study.

Other than that, the experience is exactly the same, by design - so that you know exactly how your study will work. You see the same messages about whether your child is eligible, customization based on the child or past responses works the same way, and you use the same experiment runner.

1.27.4 Going live!: the study approval process and starting data collection

If you just want other researchers to be able to preview your study to give feedback, you can set "shared preview" to true and then share the preview link on Slack.

But what about when you actually want to start data collection?

At that point, you will "submit" your study for approval by Lookit staff. We won't practice this piece, but so you know what to expect, you can look through the *information about submitting your study*.

Why the manual approval process?

From a participant's standpoint, Lookit is a unified platform, even though there are studies from a variety of research labs. This is great for participant recruitment! But it also means we're all sharing a reputation. Someone else's study that upsets or (without adequate precautions) deceives children, that baffles parents, or that just doesn't work will affect how interested families are in your study, too. Based on our early experience with researchers using Lookit, we strongly expect that a quick review will catch substantive issues often enough to be worth putting everyone through. If you are making changes to an existing study, review is either not required (if only changing certain fields like the age range/eligibility criteria) or is very quick.

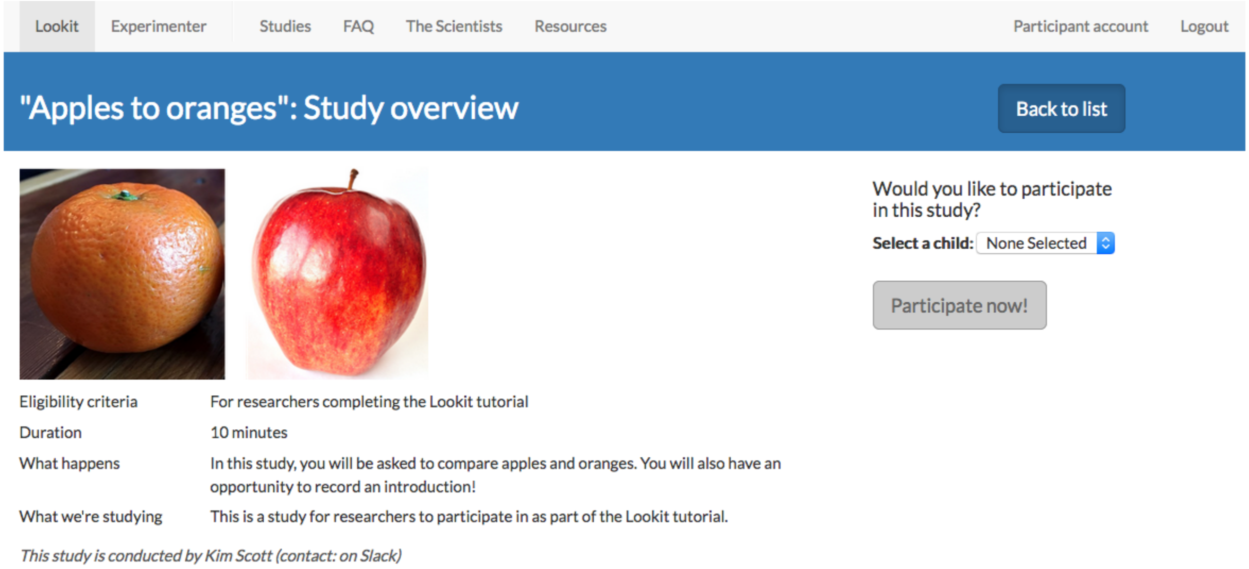
1.27.5 Create some data to play with

Because we don't want to clog up the production server with fake responses from researchers trying out Lookit, we'll do this section on the staging server, which is a separate sandbox environment that looks a lot like Lookit but doesn't have any real participant data. This is also where we try out new features before deploying them to production.

Go ahead and create an experimenter account on the staging server following the *login directions*. If you did the first part of the tutorial, you'll already have a participant account on the staging server - use a different email address for your staging experimenter account.

First, let's actually participate in another study! Go to the **staging** server studies page, https://lookit-staging.mit.edu/studies/, and select the study "Apples to oranges." This is a short study just to demo the data collection process. You can participate using your experimenter account; you may need to make a child profile and/or fill out a demographic survey before participating. Proceed all the way through this study!

Now switch back to the Experimenter interface. Note: you can toggle between Lookit (the participant-facing section) and Experimenter (the researcher-facing section) at any time via the top navbar if you are logged in as an experimenter:



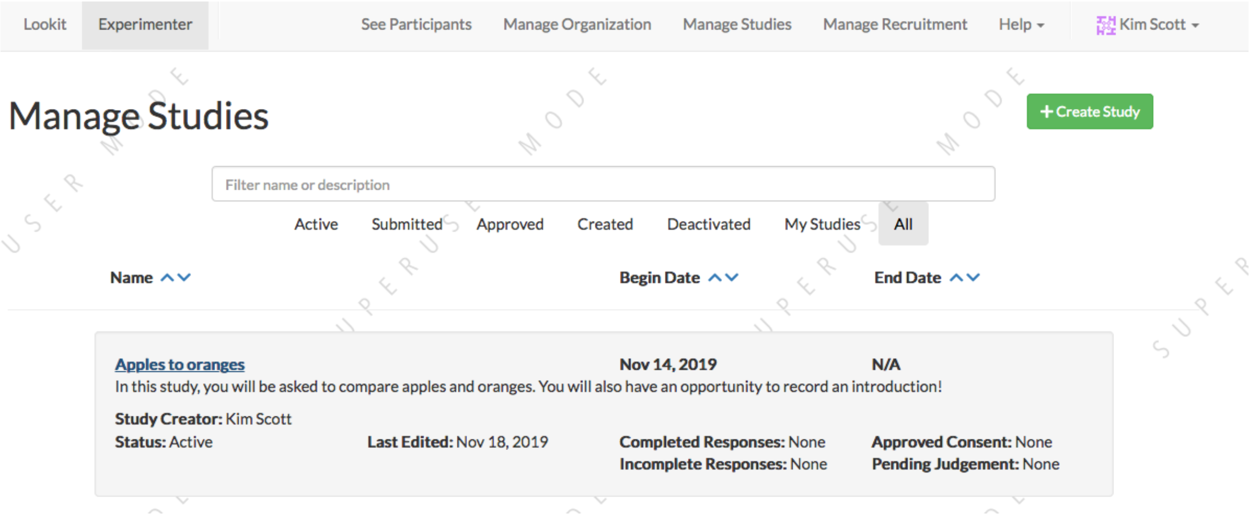
"Apples to oranges": Study overview

Would you like to participate in this study?

Select a child: None Selected

Eligibility criteria	For researchers completing the Lookit tutorial
Duration	10 minutes
What happens	In this study, you will be asked to compare apples and oranges. You will also have an opportunity to record an introduction!
What we're studying	This is a study for researchers to participate in as part of the Lookit tutorial.

This study is conducted by Kim Scott (contact: on Slack)



Manage Studies

Name	Begin Date	End Date
Apples to oranges	Nov 14, 2019	N/A

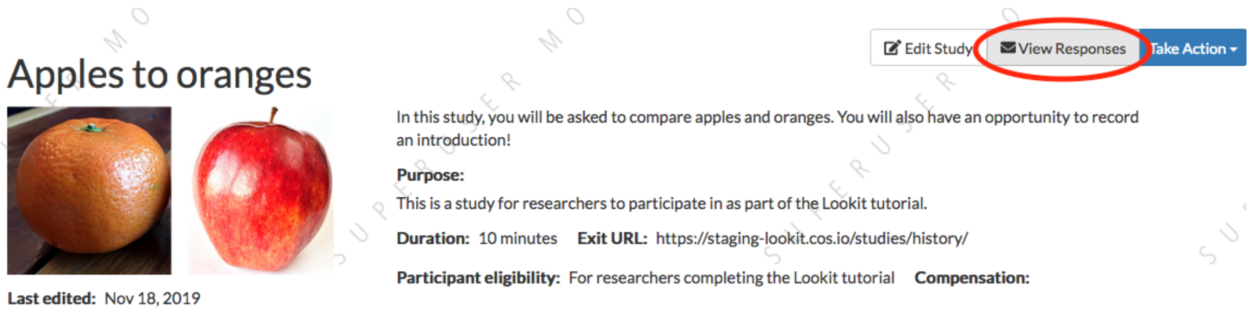
In this study, you will be asked to compare apples and oranges. You will also have an opportunity to record an introduction!

Study Creator: Kim Scott
Status: Active **Last Edited:** Nov 18, 2019 **Completed Responses:** None **Approved Consent:** None
Incomplete Responses: None **Pending Judgement:** None

1.27.6 Get access to the "Apples to Oranges" study

You are able to **see** the "Apples to Oranges" study listed on the Experimenter site on lookit-staging.mit.edu because you automatically get read-only permissions for studies within the Demo lab. However, you can't automatically see any participant data! (This is on purpose - it's not possible to grant lab-wide permissions to actual data, you have to actively add people to individual studies.)

Post in the Slack #tutorial channel and we'll add you as a researcher so you can see everything! Then, at the top of the "Apples to oranges" page, click on "View responses":



Apples to oranges

In this study, you will be asked to compare apples and oranges. You will also have an opportunity to record an introduction!

Purpose:
This is a study for researchers to participate in as part of the Lookit tutorial.

Duration: 10 minutes **Exit URL:** https://staging-lookit.cos.io/studies/history/

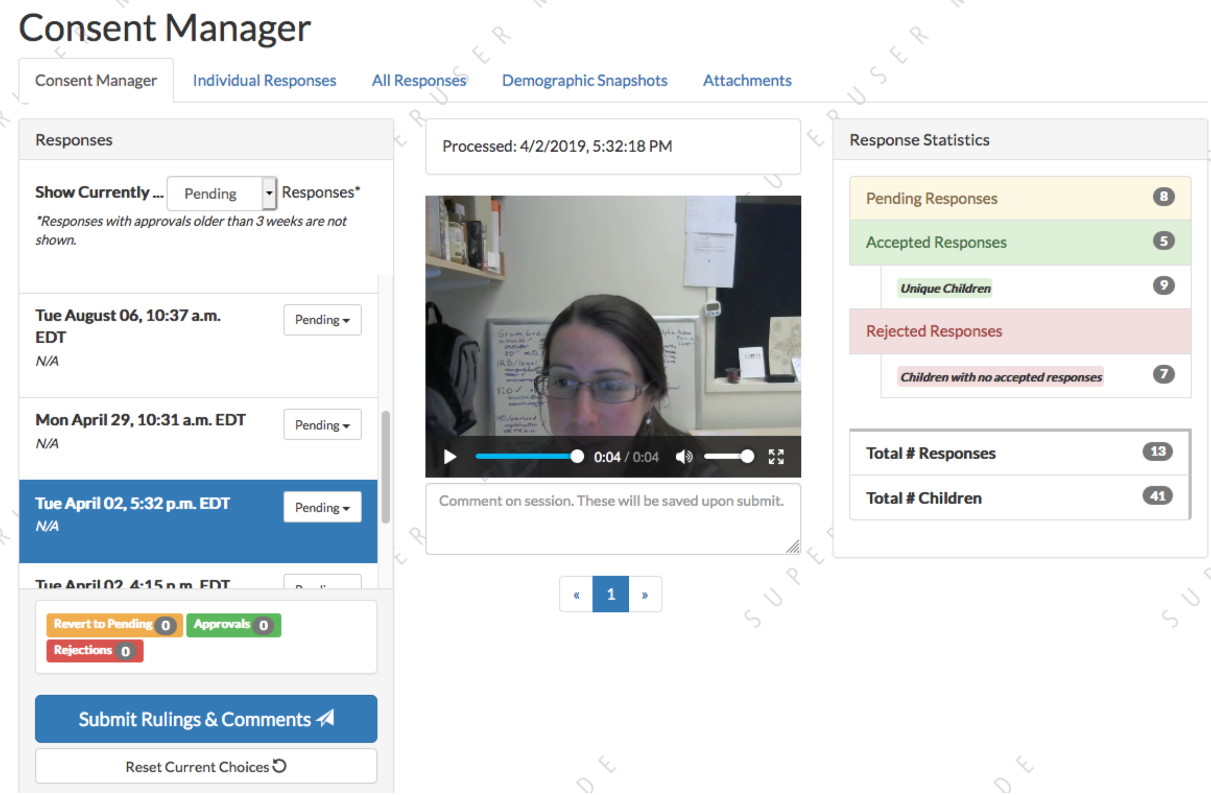
Participant eligibility: For researchers completing the Lookit tutorial **Compensation:**

Last edited: Nov 18, 2019

This will take you to a view where you can code for informed consent, view individual responses, or download response data, demographic data, and videos.

1.27.7 Checking for informed consent and giving feedback

The first page you see when you click “View Responses” is called the Consent Manager, and it should look something like this - with your own consent video (and maybe some others) displayed.



Show Currently ... Pending Responses*

*Responses with approvals older than 3 weeks are not shown.

Tue August 06, 10:37 a.m. EDT
N/A
Pending

Mon April 29, 10:31 a.m. EDT
N/A
Pending

Tue April 02, 5:32 p.m. EDT
N/A
Pending

Tue April 02, 4:15 p.m. EDT

Processed: 4/2/2019, 5:32:18 PM

Comment on session. These will be saved upon submit.

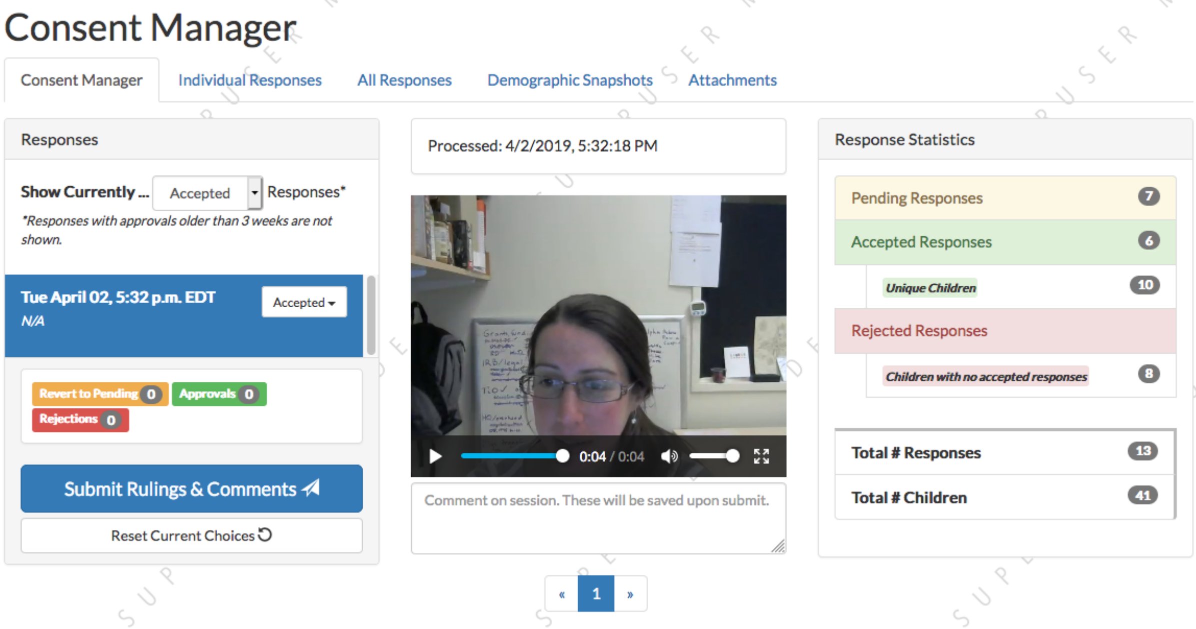
Response Statistics

Pending Responses	8
Accepted Responses	5
Unique Children	9
Rejected Responses	
Children with no accepted responses	7
Total # Responses	13
Total # Children	41

As data comes in, your first step will always be to check whether the parent provided informed consent. You do that here in the consent manager, which by default shows you the “pending” consent videos for review. In the left column, you’ll click on each session to bring up the associated consent video at the center. You can use the dropdown menu to decide whether to “accept” (mark this as valid consent) or “reject” (mark as invalid consent) each video. If you want to add any notes about the consent video, you can record comments in the text box beneath the video - for instance, you might note that there was a technical problem with the video, but you contacted the parent to confirm consent.

For now, just mark your own video that you just made as “Accepted.” Then click “Submit Rulings & Comments”. This saves your consent coding to the Lookit server. (In case it matters to your IRB: A record of which logged-in user made each consent determination and when is also stored.)

In the Consent Manager, you can now use the top drop-down menu to view currently “accepted” responses, and you should be able to see your own video there:



Consent Manager

Responses

Show Currently ... Accepted Responses*

*Responses with approvals older than 3 weeks are not shown.

Tue April 02, 5:32 p.m. EDT
N/A
Accepted

Processed: 4/2/2019, 5:32:18 PM

Comment on session. These will be saved upon submit.

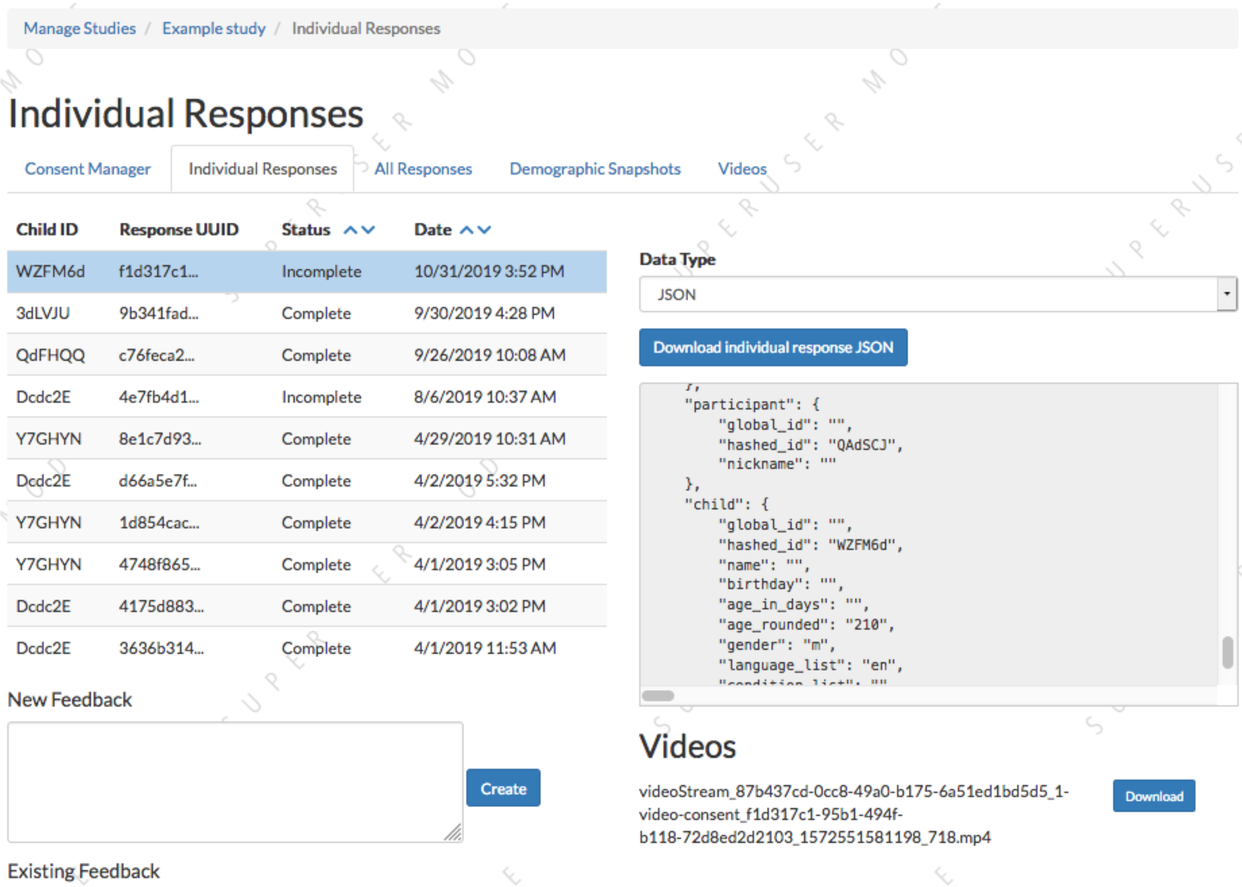
Response Statistics

Pending Responses	7
Accepted Responses	6
Unique Children	10
Rejected Responses	
Children with no accepted responses	8
Total # Responses	13
Total # Children	41

If you needed to, you could still change the ruling about this consent video, in case you made a mistake.

You may notice that, compared with before, there’s now more information displayed beneath your video when you select your consent video and scroll down! That’s because, once you mark it as having valid consent, all the session data becomes available to you.

Click the “individual responses” tab to take a look at the data that’s been collected on this study in some more detail:



Individual Responses

Child ID	Response UUID	Status	Date
WZFM6d	f1d317c1...	Incomplete	10/31/2019 3:52 PM
3dLVJU	9b341fad...	Complete	9/30/2019 4:28 PM
QdFHQQ	c76feca2...	Complete	9/26/2019 10:08 AM
Dcdc2E	4e7fb4d1...	Incomplete	8/6/2019 10:37 AM
Y7GHYN	8e1c7d93...	Complete	4/29/2019 10:31 AM
Dcdc2E	d66a5e7f...	Complete	4/2/2019 5:32 PM
Y7GHYN	1d854cac...	Complete	4/2/2019 4:15 PM
Y7GHYN	4748f865...	Complete	4/1/2019 3:05 PM
Dcdc2E	4175d883...	Complete	4/1/2019 3:02 PM
Dcdc2E	3636b314...	Complete	4/1/2019 11:53 AM

New Feedback

Existing Feedback

Data Type: JSON

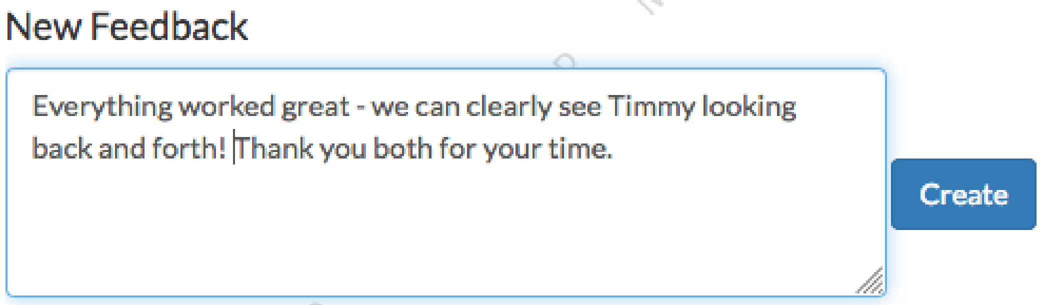
```
    },
    "participant": {
        "global_id": "",
        "hashed_id": "QAdSCJ",
        "nickname": ""
    },
    "child": {
        "global_id": "",
        "hashed_id": "WZFM6d",
        "name": "",
        "birthday": "",
        "age_in_days": "",
        "age_rounded": "210",
        "gender": "m",
        "language_list": "en",
        "condition_list": ""
```

Videos

videoStream_87b437cd-0cc8-49a0-b175-6a51ed1bd5d5_1-video-consent_f1d317c1-95b1-494f-b118-72d8ed2d2103_1572551581198_718.mp4

The top response in the table will probably be your own response that you accepted just now. With that row selected, you'll see a JSON version of data collected during the session displayed and a list of videos collected during the session. If you scroll to the bottom of the JSON data, you'll see information about the most recent consent ruling and the child who participated, so you can check who this is.

There's also a box where you can provide feedback to the participant. This feedback gets displayed on the participant's "past studies" page and is a good place to leave a short but personal thank-you message that shows a human has seen and appreciates their videos. Try it out! Leave a feedback message on your own video.



New Feedback

Everything worked great - we can clearly see Timmy looking back and forth! Thank you both for your time.

Then go back to the participant-facing site, and find that feedback under "Studies" -> "Your past studies."

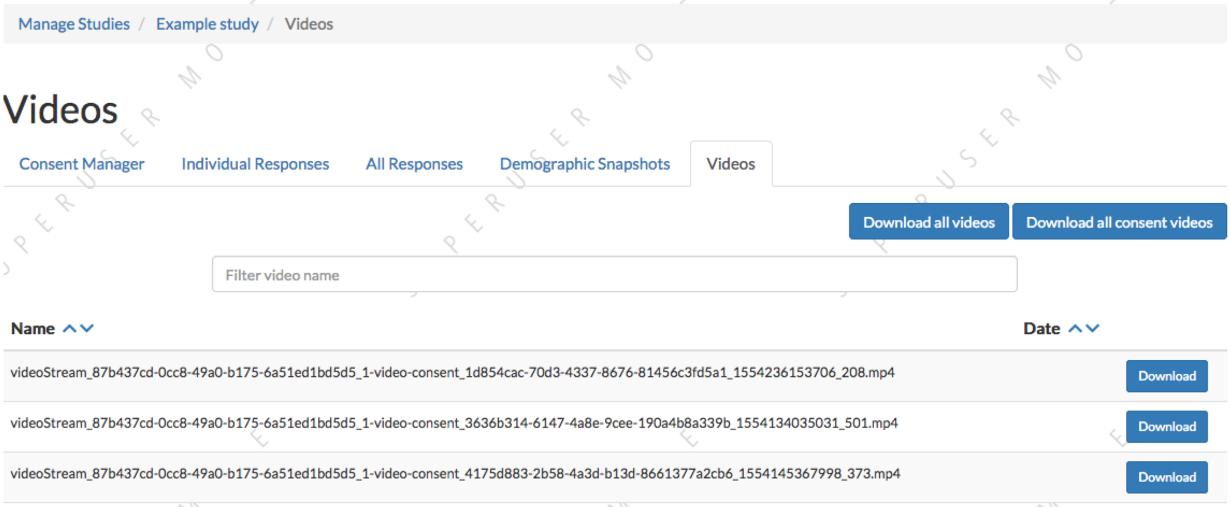
For more practice

Want to play around with this a little more? See what happens if you go back and reject your consent video. Go the consent manager, display accepted consent videos, and reject yours. Now go back to individual responses. Your response is gone! Why is that, and how would you get it back?

1.27.8 Downloading response data & videos

The consent manager and "individual responses" views can be helpful to get an idea of how data collection is going, but to code your videos and analyze your data you will want to download files that you can work with using your software of choice.

To download all videos, you can go to the "videos" tab and click "download all videos." A zip file will be bundled up for you to download, and you will receive a link by email in a few minutes. Try it out, and take a look at some of the video collected!

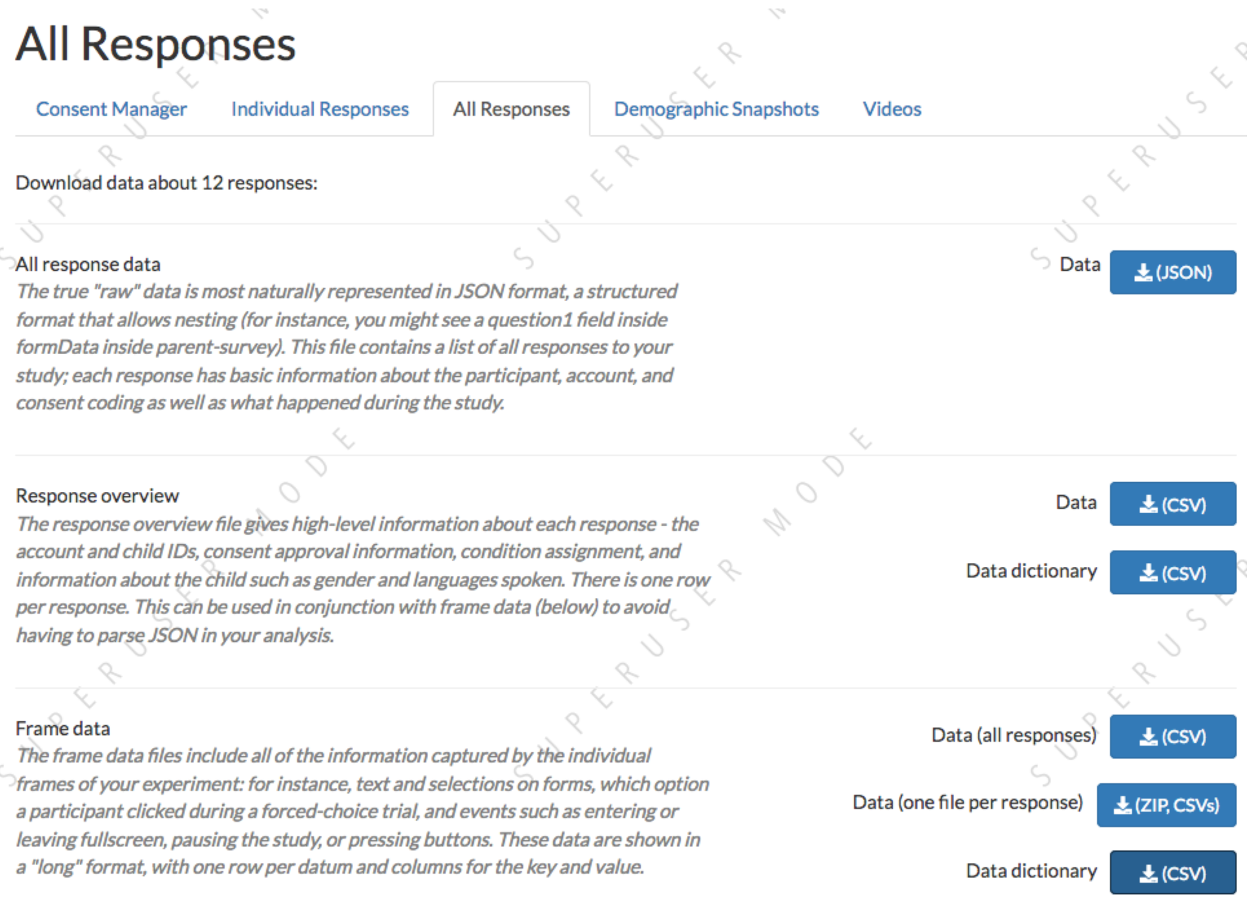


Name	Date	
videoStream_87b437cd-0cc8-49a0-b175-6a51ed1bd5d5_1-video-consent_1d854cac-70d3-4337-8676-81456c3fd5a1_1554236153706_208.mp4		Download
videoStream_87b437cd-0cc8-49a0-b175-6a51ed1bd5d5_1-video-consent_3636b314-6147-4a8e-9cee-190a4b8a339b_1554134035031_501.mp4		Download
videoStream_87b437cd-0cc8-49a0-b175-6a51ed1bd5d5_1-video-consent_4175d883-2b58-4a3d-b13d-8661377a2cb6_1554145367998_373.mp4		Download

Note that on this page you can also filter for specific parts of the filename, including the frame name and response ID.

Videos are named `videoStream_<study ID>_<frameIndex>-<frame ID>_<response ID>_<timestamp>_<random digits>.mp4`, so you can use the response ID to match videos to other response data even if you only have the filename. The response data will also contain video IDs in the `expData` for any frames that recorded video.

Under "All responses," you can download JSON or CSV files with data about all responses from this study. *You can learn more about these options here.*



All Responses

Download data about 12 responses:

All response data

The true "raw" data is most naturally represented in JSON format, a structured format that allows nesting (for instance, you might see a question1 field inside formData inside parent-survey). This file contains a list of all responses to your study; each response has basic information about the participant, account, and consent coding as well as what happened during the study.

Data (JSON)

Response overview

The response overview file gives high-level information about each response - the account and child IDs, consent approval information, condition assignment, and information about the child such as gender and languages spoken. There is one row per response. This can be used in conjunction with frame data (below) to avoid having to parse JSON in your analysis.

Data (CSV)

Data dictionary (CSV)

Frame data

The frame data files include all of the information captured by the individual frames of your experiment: for instance, text and selections on forms, which option a participant clicked during a forced-choice trial, and events such as entering or leaving fullscreen, pausing the study, or pressing buttons. These data are shown in a "long" format, with one row per datum and columns for the key and value.

Data (all responses) (CSV)

Data (one file per response) (ZIP, CSVs)

Data dictionary (CSV)

Analyzing the data collected is, in general, outside the scope of this tutorial as it will vary substantially by lab/project - although we hope that you will share your scripts and processes for analyzing Lookit data to help other researchers! The exercises below can be solved by manual inspection of the CSV (or JSON) data, although you are also welcome to set up a script in your language of choice to get a head start on real data processing.

Exercises

1. How many researchers said they preferred oranges? How many said they preferred apples?
2. What fraction of researchers gave different answers on the actual test question vs. the survey?

1.27.9 Downloading demographic data

Under ‘demographic snapshots’, you can also download demographic survey responses from the accounts associated with children who participated in your study (once consent is approved). For each response, you will see demographic survey data for that participant at the time of participation.

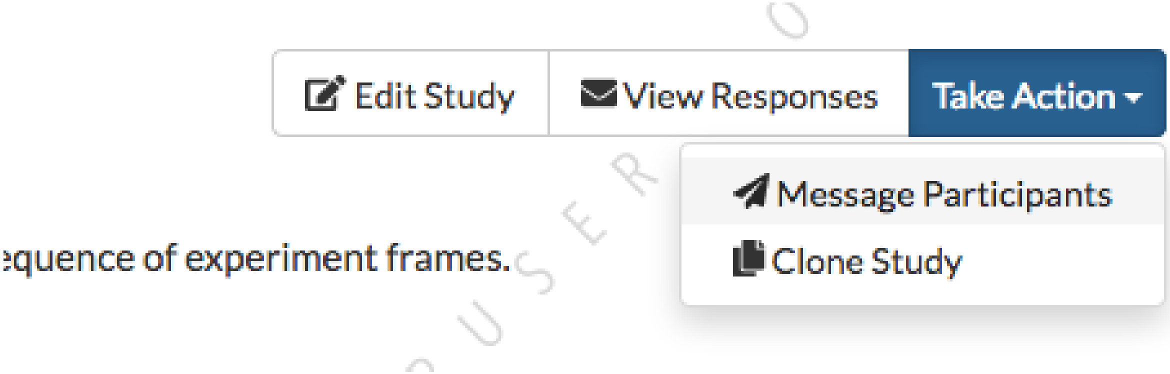
Exercises

1. What fraction of responses are from researchers in urban locations?
2. What fraction of children who responded at least once live in homes with at least 10 books?

1.27.10 Contacting participants

You may need to contact participants for a variety of reasons: for instance, to let them know it's time to complete another session of a longitudinal study, to ask for clarification about a problem they reported, or to announce that the results of your study have been published!

You can contact participants in a particular study using the "Message Participants" link at the top of your study, found here under "Take Action":



That will take you to a page link this where you can see and download previous emails (left side) or compose new emails (right side). This interface is in progress with work planned to make it easier to use, but it's functional!

Where are the email addresses?

You may notice that although you can message participants, you're not being provided with their actual email addresses. We apologize for the inconvenience this causes in implementing some custom workflows, and can discuss providing email permissions with individual labs if necessary. However, obscuring email addresses is deliberate: it allows us to programmatically enforce participants' email selections (so that they don't receive email types they don't want), protects against accidental disclosure, and ensures you have a central record of all communication. Again, this is a matter of sharing a reputation!

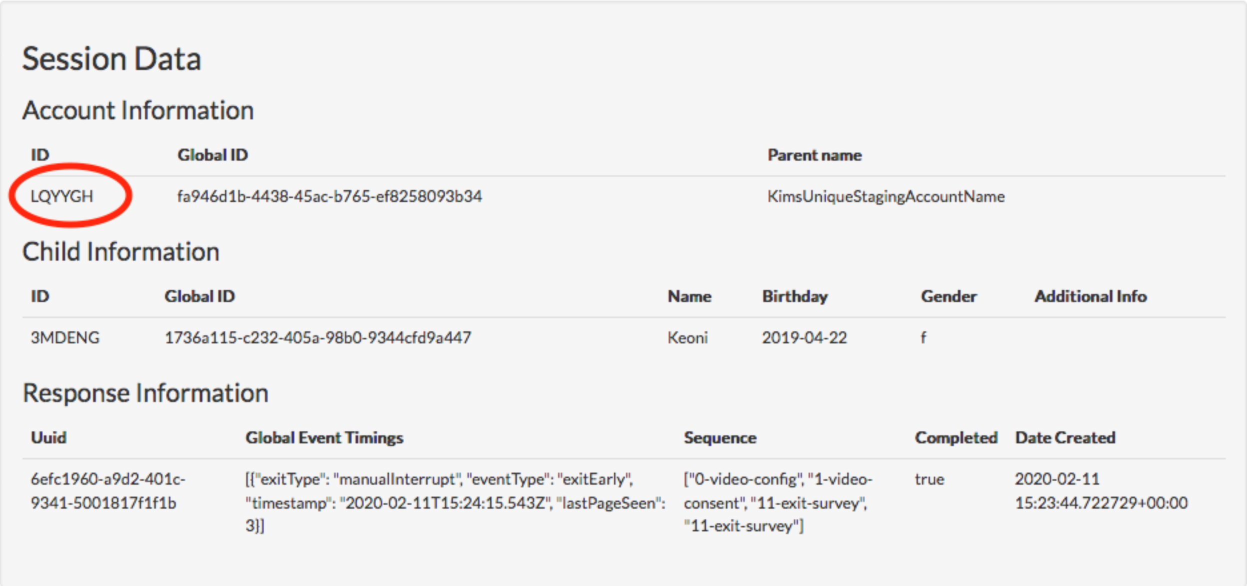
The first thing you will do when you send an email is select the "Message Type". These line up with the email types participants can opt to receive: notifications that it's time for another session of a longitudinal study; notifications that a new study is available for them to participate in; updates about this study (like that results are available); and clarifying questions about their responses.

Next, you specify the recipient(s). You can do this by searching for the appropriate **account** ID. Finally, you write your message subject and body, and hit send! Let's try it out with a few example scenarios.

Contact a participant about a consent video issue

First, let's imagine that there was an issue with your consent video and you needed to confirm that it was ok to use data from the session.

In one browser tab, open up the consent manager view for the "Apples to Oranges" study, and find your consent video. Scroll down to the information about the session. You should see a "Participant information" section, separate from "Child information." Copy the (hashed) ID for the participant.



Session Data

Account Information

ID	Global ID	Parent name
LQYYGH	fa946d1b-4438-45ac-b765-ef8258093b34	KimsUniqueStagingAccountName

Child Information

ID	Global ID	Name	Birthday	Gender	Additional Info
3MDENG	1736a115-c232-405a-98b0-9344cfd9a447	Keoni	2019-04-22	f	

Response Information

Uuid	Global Event Timings	Sequence	Completed	Date Created
6efc1960-a9d2-401c-9341-5001817f1f1b	[{"exitType": "manualInterrupt", "eventType": "exitEarly", "timestamp": "2020-02-11T15:24:15.543Z", "lastPageSeen": 3}]	["0-video-config", "1-video-consent", "11-exit-survey", "11-exit-survey"]	true	2020-02-11 15:23:44.722729+00:00

In another browser tab, open up the "Message participants" view for the same study. Choose the message type "response questions" since this is a clarifying question about the response. Under "recipients," deselect all and then paste the participant ID into the box. That should bring up exactly one potential recipient (which is you!) - click to add it.

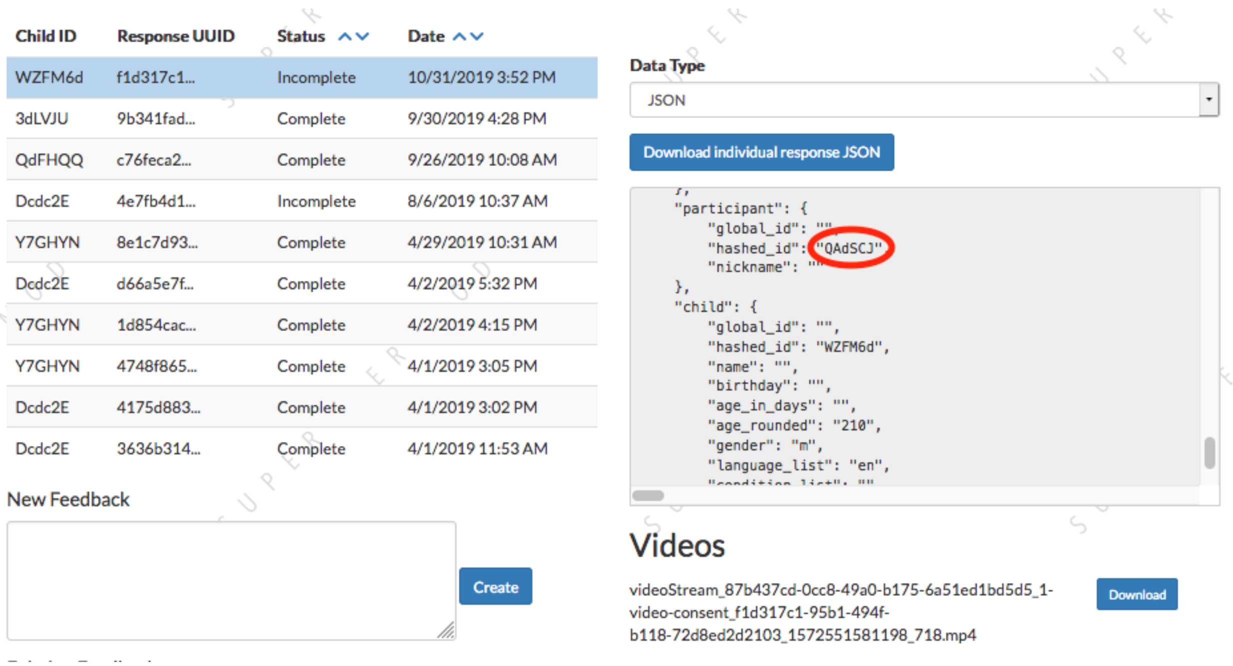
Write a subject and body for your email explaining the problem and asking whether it's ok to use data from this session. (See *day-to-day study operation* for details about what you might say!)

Go ahead and send your email, and make sure you receive it!

Contact a participant with a gift card code

Second, let's imagine that you're compensating participants with gift cards. (You'll want to take a look at the Terms of Use and *compensation info here* as you make more detailed plans, but essentially, for now researchers are responsible for handling any compensation by messaging participants.)

Instead of the consent manager, switch over to "individual responses" and find your response again. Copy the participant ID from the response JSON:



Child ID	Response UUID	Status	Date
WZFM6d	f1d317c1...	Incomplete	10/31/2019 3:52 PM
3dLVJU	9b341fad...	Complete	9/30/2019 4:28 PM
QdFHQQ	c76feca2...	Complete	9/26/2019 10:08 AM
Dcdc2E	4e7fb4d1...	Incomplete	8/6/2019 10:37 AM
Y7GHYN	8e1c7d93...	Complete	4/29/2019 10:31 AM
Dcdc2E	d66a5e7f...	Complete	4/2/2019 5:32 PM
Y7GHYN	1d854cac...	Complete	4/2/2019 4:15 PM
Y7GHYN	4748f865...	Complete	4/1/2019 3:05 PM
Dcdc2E	4175d883...	Complete	4/1/2019 3:02 PM
Dcdc2E	3636b314...	Complete	4/1/2019 11:53 AM

New Feedback

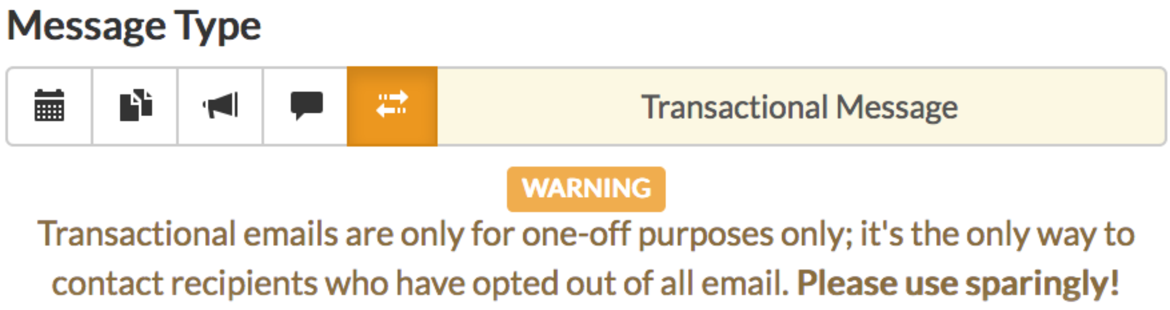
Data Type: JSON

```
    "participant": {
        "global_id": "",
        "hashed_id": "QAdSCJ",
        "nickname": ""
    },
    "child": {
        "global_id": "",
        "hashed_id": "WZFM6d",
        "name": "",
        "birthday": "",
        "age_in_days": "",
        "age_rounded": "210",
        "gender": "m",
        "language_list": "en",
```

Videos

videoStream_87b437cd-0cc8-49a0-b175-6a51ed1bd5d5_1-video-consent_f1d317c1-95b1-494f-b118-72d8ed2d2103_1572551581198_718.mp4

Returning to your "message participants" tab, let's create another email. This time, you can actually select the "transactional email" option, which allows you to reach even people who have opted out of email; this is because you sending the compensation is the completion of a "transaction" they agreed to. You will see a warning which is ok:



Message Type

Transactional Message

WARNING

Transactional emails are only for one-off purposes only; it's the only way to contact recipients who have opted out of all email. **Please use sparingly!**

Like before, paste in your ID, write your message, send it, and make sure you receive it. (Don't actually send yourself a gift card. Unless you really want to.)

Congratulations! We've covered all the basic functionality you'll need to manage your studies. Finally, we'll wrap up by briefly noting some of the advanced features you might want to use later and revisiting Github issues now that you may have some feature requests or bug reports.

1.28 7. Finishing steps

You're in the home stretch! Thanks for sticking with it. We'll wrap up by revisiting Github issues and briefly mentioning some of the advanced features you might use in the future.

1.28.1 1. Making your first Github issue or comment

Now that you've created your first two studies and practiced some of the day-to-day study management tasks you'll do on Lookit, you've almost certainly thought of something you wish you could do easily on Lookit - or just something you found confusing.

As we discussed at the start of the tutorial, Lookit is an open-source project meant to serve the research community, and we need everyone to contribute to keep making it better! One of the best ways to contribute is to document the problems you encounter and the ideas you have for improvement.

Let's practice that now. First, think of either a problem you encountered or something you'd like to see, in any of these categories:

- The documentation, including the tutorial you're reading now
- The Lookit platform, including login/account system and the experimenter inferface (study management, data viewing/download, consent coding, participant messaging, etc.)
- The frameplayer, including what frames are available to you to use in designing your studies and how they function

Review the section on Github issues and click the appropriate link above to view the current issue list. Either create a new issue if your idea/problem isn't already described, or comment on an existing issue!

1.28.2 2. Some advanced options to be aware of

Although we hope this tutorial has provided a solid introduction to designing and managing your Lookit study, it doesn't cover everything it's possible to do! The goal is that upon completion, you are ready to learn about more advanced options using the rest of the documentation. Here are some examples of topics you may be interested in exploring as you continue:

- Writing a Javascript function to generate your study protocol
- *Creating your own custom frames to use on Lookit*
- *Use the API to programmatically download study data*
- Set up conditional logic in your studies
- *Contribute a new feature to the codebase*

1.28.3 3. So what comes next?

Now that you have a basic understanding of how to use Lookit, you're ready to start setting up the study you want to run! One frequent question is how long this will take. Once you have your study completely planned out–the design, instructions, and stimuli all ready–it's likely to be very quick, perhaps a few days to implement. But the (unanimous) experience of researchers who have beta-tested Lookit is that it takes much longer than expected for purely non-technical reasons: the process of fleshing out stimuli and instructions often brings up substantive issues about experimental design that might not have been caught as early when piloting in the lab.

In addition to the technical preparations you're making to be able to implement your study, you'll want to make sure you're ready to go from a legal and IRB standpoint! See *IRB and legal info* for details. Basically, though, you'll need to get an institutional agreement signed; complete a short quiz about the Terms of Use because otherwise no one reads them; and include data collection on Lookit in your IRB protocol.

You can go ahead and:ref:*create a lab<labs>* at any point, and it'll be approved once you've done the steps above.

1.28.4 4. The last step: please give feedback on the tutorial!

One last request: please fill out the Google form to give feedback that will be used to help improve this tutorial. Thank you!

1.29 Community involvement

Currently Lookit is run by three people working at home during a pandemic. We had a description here of all the things we do, but really, enough said.

1.29.1 Ways *all* Lookit researchers contribute

1. Peer-to-peer support

We ask that everyone who goes through the tutorial helps to answer questions from other researchers, and works on improving any sections that were confusing!

2. Provide peer feedback on studies

Before submitting a study to Lookit, researchers gather peer feedback (see *Study approval process*). This is a quick and lightweight process, not scientific peer review, but it does a lot to improve the participant experience! As a researcher on Lookit, you will help to review other studies as well as posting your own for feedback.

3. Recruitment

Once you start running your study, you will need to actively recruit participants. There's a shared Lookit participant pool, but we need everyone's combined efforts to build it.

1.29.2 If you have programming experience (or employ someone who does)

Lookit is an open-source project, but struggles with the problem that relatively few of its primary stakeholders (developmental researchers) feel equipped to contribute to feature development. If you have experience programming in Python, the most helpful way to contribute is to *Set up for local development* on the Lookit codebase, take ownership of an issue in the lookit-api repo and make a PR with your implementation (and your unit tests ;)).

1.30 Working Groups

In addition to the actions described above, individual researchers are organized into a variety of Working Groups that focus on specific challenges to build resources for the community.

The existence of the Working Groups has many advantages, including:

- Formally recognizing the work that many people volunteer to do in the background, that might otherwise be invisible
- Making sure we have a wide range of voices involved in generating ideas and making decisions about our shared platform, beyond just the small number of people on the Core Team

- Providing a way for people to opt into the discussions they most want to be a part of so that (e.g.,) if you want to be involved with decisions about recruitment strategies you can be part of a group that does so, rather than decisions being made at unpredictable times and only by those who (e.g.,) happen to be paying attention to a general email listserv at that time.

Anyone involved with a current study on Lookit, or who is developing one, is very welcome to join a Working Group! We also have an expectation that **any research team hosting a study on Lookit will have at least one member who is a member of at least one Working Group**, with a flexible time commitment of about 2 hours/week. The person meeting this requirement for their research group should be a "main" member of the group, which will typically mean a lab manager, graduate student, or PI (not, e.g., an undergrad doing a small number of hours of lab work each week).

Here are the current working groups:

1.30.1 Recruitment (social media, etc.)

The goal of this Working Group is to increase the size and diversity of the participant pool. Examples of work might include producing posts for social media, contacting parenting blogs, and coordinating efforts from individual labs. In addition to balancing the goals of size and diversity, another pair of goals to balance will be exploitation vs. exploration (i.e., small tweaks to known recruitment methods vs. moonshots for much better solutions).

1.30.2 Eliciting and organizing feature feedback from researchers

The goal of this Working Group is to gather information from researchers in multiple ways (e.g., surveys, focus groups, monitoring of questions on Slack and GitHub issues) about potential new features for Lookit, and work towards prioritizing them. This requires synthesizing information about what would best benefit researchers and what is easiest for the programmers to implement, and so this group will work with the programmers as well.

1.30.3 Advise and support new Lookit researchers ("Researcher onboarding")

The goal of this Working Group is to help new researchers who are designing their first Lookit study. This can involve creating additional tutorials/samples/documentation, providing 1-on-1 mentoring, and monitoring a Slack channel where new researchers ask questions.

1.30.4 Parent-facing "first 10 minutes"

The goal of this Working Group is to maximize the parent-facing experience, from finding Lookit to finishing the consent for the child's first study. Example work might include revising the homepage and doing focus group (user experience) studies of naive parents navigating it, creating informational videos about what participation entails and instructions for participating (perhaps with help from the "assets" working group), and streamlining the informed consent process. An MIT usability & accessibility group may be available for support.

1.30.5 Producing creative assets (e.g., recruitment videos)

The goal of this Working Group is to produce a library of available pictures, videos, and audio. A primary use for these would be the "Recruitment" group, though it is also possible that creative assets might be used for other purposes (e.g., a library of stimuli available for studies, such as infant attention-getters and royalty-free cartoon pictures of many characters).

1.30.6 Researcher-facing "first 10 minutes"

The goal of this Working Group is to maximize the researcher-facing experience, from discovering Lookit to deciding whether or not it might be a good fit for their research. For example, this group might create a webpage that is directed towards PIs, to determine in 3-5 minutes whether joining Lookit might be a good use of a graduate student's time. Another webpage might be directed towards graduate students, to get a sense of the types of research Lookit is a good fit for, and what types of skills would be necessary to conduct that research via Lookit. Much of this information already exists, but optimizing it for different audiences could make a big difference in the uptake of Lookit by research teams.

1.30.7 The A-Team

Named after a TV show about mercenaries for hire: "If you have a problem, if no one else can help, and if you can find them, maybe you can hire… the A-Team." For Lookit, this Working Group will work on all of the miscellaneous tasks that are shorter term (so they don't have their own working group) but totally essential for the development of Lookit. Maybe a couple of weeks working on legal and ethical issues to build a knowledge base for getting Lookit access agreement signed. And then the next project might be exploring non-monetary compensation for families (certificates? personalized information?). All skill sets welcome for a well-balanced group of problem solvers.

1.31 IRB and legal info

1.31.1 Overview

Before you can run a study on Lookit, there are three things you'll need to do:

1) Sign and return the access agreement (details below). This covers your entire lab. You only have to do this once.
2) Get IRB approval at your own institution (details below).
3) *Get peer feedback on the study, and then have it approved by Lookit.*

1.31.2 Access agreement

What it is

In order to use Lookit for data collection, the principal investigator (PI) of a research group must first sign an institutional agreement with MIT that lays out each institution's rights and responsibilities. This may require review by a legal or contracts office or an office of sponsored programs depending on the institution.

The access agreement with a PI at a given institution covers **all** studies conducted on Lookit where the approved IRB protocol at that institution names that person as the PI. (If that institution cedes to another institution for approval, that's still okay.)

The agreement itself is short and closely based on Databrary's agreement (with permission); the meat is in the Terms of Use you are agreeing to. **Researchers should actually read the terms carefully:** these are real content written by a real person and contains non-obvious information about ethical/privacy issues you may not have encountered in the lab. Note that we do impose standards that may in some cases be more stringent than your own IRB's.

To make sure we're all on the same page, we require that at least one researcher associated with each research group using Lookit complete this quiz about the Terms of Use.

How to get it signed

If you are planning a Lookit study, getting the user agreement signed should be your first step because it takes a highly variable amount of time (record so far is under two weeks, but budgeting six months is sensible). It does not require any customization on your part - you just need to send it off to someone at your institution who's authorized to sign on behalf of the institution or who can figure out who is.

Unfortunately the appropriate office varies by institution. If you've previously set up to use Databrary, try contacting the office that signed off on that. As examples, the following offices have approved the access agreement:

- Contracts - NYU, Wisconsin, Stanford
- VP Finance - Haskins
- Partnerships - Toronto
- University/Industry Liason - UBC
- Department head - Goldsmiths London
- Provost - Yeshiva
- Sponsored programs - American University

In general, you do NOT need to have an IRB protocol to use Lookit in progress or approved first. In some cases, though, the office that signs the agreement may ask that you obtain ethical approval first.

The one exception to this process is that researchers at MIT, completing studies approved by MIT's IRB, do not require the signature of an MIT authorized signer. However, the MIT PI still needs to sign and return the agreement.

Multiple PIs at one institution

The access agreement signed by one PI only covers research where they're named as the PI on the corresponding IRB protocol. Other PIs at the same institution do need their own access agreement on file.

If you know of another lab at your institution using Lookit, please reach out to them or to Lookit for a copy to check (a) who the institutional signer should be and (b) whether any changes were required (you can use the same modified version).

Correspondence between access agreements and Lookit Labs

In general, each access agreement should correspond to exactly one Lab on Lookit, which makes it far more straightforward to ensure that everyone's actually covered legally.

Each Lab must be associated with at least one agreement. (E.g., suppose Jill is a PI at Science University and has signed an access agreement, and Joe is another PI at Science University who does not. We cannot approve Joe's Lab on Lookit to test - even if he tells us that his work will always actually be covered under Jill's IRB protocol.)

However, if multiple PIs at your institution are closely affiliated (e.g. a lab jointly headed by two professors, or students all serve as PIs on IRB protocols in your department) please go ahead and create a lab listing multiple PIs and just let us know. We'd need an access agreement for each PI, and would expect all studies conducted by this lab to be covered by an IRB protocol listing at least one of those PIs.

If your institution requires changes

We have generally been able to accommodate requests for minor changes to the agreement as required by universities' legal departments (e.g. when they are *legally unable* to agree to MA governing law, or to allow UK ethics committee approval in addition to "US-equivalent" IRBs). If it looks like some clause would be a deal-breaker for your institution, please get in touch with Kim Scott (kimscott) and general counsel lawyer Jason Baletsa (jbaletsa) at mit.edu for advice.

Please note that approval of changes happens on the MIT General Counsel office's timeline, which is not under Lookit's control.

Sending it to Lookit

Once you have the user agreement signed by the PI and institution's authorized signer, please:

- Make sure you or someone in your group has completed this quick quiz about the Terms of Use
- Send the PDF of the signed agreement to lookit@mit.edu, noting that you have also completed the quiz

1.31.3 Ethics approval for Lookit studies

A researcher using Lookit applies to their own institution's IRB for approval. Except for our involvement in a small set of initial studies, MIT is not considered to be engaged in human subjects research by running Lookit.

You can find resources for writing your IRB protocol here. You will likely need to give a brief overview of how data collection on Lookit works and what protections are included for participant data.

Lookit *may* request a copy of the approved protocol at the time a study is submitted to run on Lookit. (In general we don't - we aren't looking to get into that much paperwork. This is mostly if we have any concerns or you're getting permission to do something unusual.)

How consent and assent work

The consent process on Lookit is fairly standardized by design, to reduce the mental load on families. Parents consent by recording a verbal statement of consent after reviewing a document.

Information about what the consent page looks like to parents, and how to customize it, is available here in the frame documentation. You can see the available templates for the consent document here. Essentially you will supply pieces of text that will be substituted into the template.

If you are testing older children and need to collect assent in addition, you will use the assent page, which is less standardized at this point.

When applying for IRB approval, you may need to present the exact text of the consent document that will be presented, which you can do by substituting values into the templates linked above. You can even start setting up your study (copy one of the tutorial/example studies and customize the consent text) and preview it to get screenshots or the PDF download.

If you anticipate running follow-up studies (which will have at least different titles and durations), see if you can show your IRB the template and note that certain pieces may vary, giving them examples. They may be ok with you saying, for instance, that it's going to look exactly like this except that (a) your first study is called "Do Babies Love Cats?" but other studies may be called "Do Babies Love Dogs?", "Do Babies Love Rabbits?", etc. and (b) your first study is 15 minutes long so that's what the consent form says, but ohter studies may be 5 - 25 minutes long, and their durations will be accurately listed in the consent form. Keeping the procedures and purpose information relatively general to accommodate a broad protocol is ok if you need to!

Please keep the text you insert as easy to read as you possibly can. Do not include boilerplate text that's in your five-page in-person consent just because it's there and you're not sure if your IRB will let you remove it - ask them! Do not include information that is covered elsewhere in the template just because you usually word it slightly differently; that's confusing.

Researchers must use these standard consent pages, barring extraordinary legal barriers at their institutions (we have not encountered anyone not able to approve a protocol). You may NOT use your existing consent form instead or in addition just because you don't want to submit an amendment to your protocol (sorry).

You will likely need to apply for a waiver of the usual requirement to collect written consent from participants so that you can rely on videorecorded statements. (The statements parents make on Lookit are better evidence of true informed consent than commonly-approved checkboxes for online studies, and so far this has not been an issue.) You can see an example here.

1.31.4 Responsibilities of researchers

- Everyone:
  - Protect your Lookit account credentials. **You are responsible for any access to participant data on Lookit via your account.** You should choose a strong password, change it regularly, and never share it. (Share access by adding another researcher to your Lab - not by sharing a lab account password.)
  - Promptly report any breach or potential breach of participant information that you become aware of, both to Lookit and to your local institution.
- PIs:
  - You are responsible for the actions of anyone who's conducting research on Lookit under your access agreement. Ensure that anyone added to your Lab on Lookit understands any rules in the Terms of Use that apply to what they'll be doing. (E.g., make sure an RA tasked with approving consent and contacting families with compensation understands when it's ok to withhold compensation, when to approve/reject consent, etc. Make sure someone helping with analysis understands which fields must be omitted from published data.) You may choose to create your own training materials based on the tasks they'll actually be doing - please share these if so!
  - Ensure that people who are no longer in your lab are removed from your Lab promptly.
- Study admins:
  - Ensure you have current IRB approval for your study before beginning ANY data collection on Lookit. Promptly pause the study if approval lapses.
  - Ensure that anyone with access to participant data for this study is listed on the appropriate IRB protocol. Promptly remove permissions for anyone who is removed from the IRB.
  - Assign *minimal appropriate roles* for each person who needs access to your study.

1.31.5 Privacy policy

Both researchers and participants are covered by the privacy policy.

1.31.6 Sub-processors and information about GDPR compliance/DPAs

AWS S3 (data storage - participant video): https://aws.amazon.com/service-terms/

GCP (data storage, databases; temporary data storage, video (.zip downloads); platform and study hosting/deployment): https://cloud.google.com/security/gdpr/resource-center/contracts-and-terms

Pipe (video streaming; no copies stored): https://addpipe.com/gdpr

Sentry (error reporting; no identifiable information): https://sentry.io/security/#hipaa-and-hitech

1.32 Study approval process

Remember, before your study can go live on Lookit, there are three things you'll need to do:

1) *Sign and return the access agreement.* This covers your entire lab. You only have to do this once.
2) *Get IRB approval at your own institution.*
3) Get peer feedback on the study, and then submit it to Lookit for approval. This is the part we'll discuss here!

1.32.1 Why is there a study review process?

Because Lookit is a shared platform with a shared reputation, it benefits all of us to uphold high standards for all Lookit studies. Families who have a good experience - the study is fun, it runs smoothly, the instructions are easy to follow, they get a gift card promptly or get quick friendly answers to their questions, their child isn't upset by the stimuli - will come back. Positive media coverage likewise benefits all of us.

To make sure all studies meet community standards, we require two forms of review before studies are posted: at least one other researcher outside your lab will provide peer feedback, and Lookit staff will review as well. While this may feel like an "extra step" compared with in-lab research, it's an opportunity to improve both the participant experience and the science - saving you time later. So far, researchers who were frustrated by the extra steps have been convinced once they see how much their studies improve in the process.

1.32.2 If you don't have IRB approval or an access agreement yet

You can still go ahead with steps 1 and 2 below so you're ready to go as soon as possible! The "Lab" your study is associated with on Lookit will have to be approved to test before you'll be able to submit in Step 3, which means you'll need the access agreement set at that point.

1.32.3 Step 1: study preparation and self-review

First, get your study working smoothly, the way you want it to, and make any revisions from within-lab feedback. Please work through the *self-review checklist* before requesting peer review!

1.32.4 Step 2: peer review

Next, post a preview link for your study on the Lookit Slack #researchers channel to gather some feedback from researchers *outside your group*. This is like getting a bunch of fresh eyes at a lab meeting made up of researchers who work online!

Peer reviewers can reference the *peer review checklist* and will be looking for things like:

- Typos
- Overall flow of the study - are there any abrupt transitions? do they feel like they know what to expect?
- Places where your instructions are confusing, overwhelming, condescending, etc.
- Cases where your stimuli or prompts might be unintentionally confusing, upsetting, or unnecessarily non-inclusive (e.g., you've assumed that each child has a mother and father at home, that a child who's experienced the birth of a sibling still has that sibling, etc.; you've prompted children to describe something bad that happened to them assuming they're going to think about stuff like their popsicle falling)
- Whether your study description accurately reflects their experience

- Experimental design issues that happen to occur to them (this isn't primarily a scientific review)

With this feedback you can revise and improve your study, often the instructions and parent-facing text in particular.

You should also expect to return the favor and review other labs' studies, as part of participating in the platform.

1.32.5 Step 3: Lookit submission and approval

When your study is ready and you've responded to the peer feedback, you can *submit your study* for Lookit approval so it can go live. This is the point where you'll need to have your access agreement set up.

When you submit the study, you will be prompted to note any non-standard elements that require specific approval per the Terms of Use (e.g., integration of additional about participants from another source) as well as what changes you made based on peer feedback.

At this point, Lookit staff will review your study, focusing primarily on the participant experience. This is also when any custom code you're using will be reviewed for security or functionality issues.

Initial internal review can generally be completed within a week. Revisions may be requested before the study can be approved to run. To minimize the number of rounds of review needed, researchers are strongly encouraged to polish their studies as much as possible before submitting - please don't use us as a proofreading service!

Outcomes of Lookit review

In general, although we technically reserve the right not to host work at our discretion, the outcomes of the Lookit admin review process are "accept" and "revise and resubmit." We'll work with you to get your study ready to go.

In rare cases, a study may be fine to run on Lookit, but in the judgment of Lookit admin staff, not a good idea to advertise publicly via Lookit. (E.g., a study that is ethically designed, but stands an unusual chance of putting parents off coming back for other studies.) In this case a "partial approval" - to collect data, but not make discoverable - may be the final outcome. We consult with working group members before making such a decision to ensure it reflects community norms, and will devise a more formal process and expanded guidelines if this occurs more often.

1.32.6 Reapproval after changes

Most changes to studies require re-approval. If your study is active, paused, or approved, and you make changes to it, it will be automatically rejected and you'll need to resubmit. If your study has already been approved, you'll see a warning to this effect when you click "save" letting you know which fields will require re-approval to change.

For minor changes ("we fixed a typo", "we clarified instructions", "we're stopping data collection for some conditions", etc.) approval is quick - you do not go back into the same queue as for initial submissions. We approve studies the same day (often within 10 minutes during working hours, but no promises).

If you want, you can let us know you'd like the study restarted in addition to reapproved. By default we just approve it and you can restart it when you're ready.

1.33 Peer review checklist

If you're previewing a study someone else has requested feedback on, here are some things to think about as you look through it and make notes.

1.33.1 Title

- Is it short and memorable?

  Note: it's fine for a title to be cute but not super informative, as long as it has something to do with the content or purpose of the study and could be used to distinguish it from other similar studies. One of its most important functions is to be memorable enough that parents can say "hey I did the XYZ study and [you should too! / I have a question about when my gift card will arrive / etc.]

  Feel free to suggest any clever ideas you have!

- Would a follow-up study, another study from the same group, or another study using the same method fall naturally under the same title? If so, the name should be more specific.
- Does it include information that's covered elsewhere, e.g. eligibility criteria or duration? Don't duplicate this here. (E.g., "XYZ for 5-month-olds" or "XYZ: a 10-minute study about cats" should just be XYZ.)

Examples

Good examples:

- "A fish, THE fish, MY fish" (babies' understanding of articles)
- "Baby see, baby do" (neonatal imitation)
- "Cue the music!" (singing in a language cueing attention to that language)
- "Is two (voices) a crowd?" (kids' ability to understand speech with one/many background voices)
- "Little drummers" (toddlers' production of rhythm)
- "Does your baby know what you are thinking?" (infant theory of mind)

Examples of titles we asked for more specific versions of:

- "Thinking about actions"
- "Look at the pictures"
- "Emotion learning study"

1.33.2 Thumbnail image

- If people are pictured, and they're white - do they need to be? (It's easy for too many researchers to "default to" pictures of white kids.)
- Is this image relevant to the study? (e.g., does it have a child pictured that's in the age range, does it include potential images that the child may see during the study, etc.)

1.33.3 Short description / "What happens"

- Does this clearly explain to parents what they/their child will be doing during this experiment in a few sentences?
- Does it contain unnecessary detail (e.g. exact numbers or timing of trials) or information covered in other fields (e.g. duration)? These should be removed and covered only in the other field to avoid duplication and make it easier to keep information consistent if it changes.
- Does it adhere to the *style guide*?

1.33.4 Purpose / "What we're studying"

- Is it accurate given the study design? (Really think this through - this has been a common challenge for researchers.)
- Does it explain what this particular study addresses (not a broader research program)?

  Note: it's ok if the study is developing/testing out a measure, trying to get a baseline measurement for future work on X, etc. It's better to actually say this than to say that the study is doing something it isn't.
- Does it explain why this question matters? Note that neither "this hasn't been studied before" nor "this ability is important in adults, so we're finding out when it emerges in kids" explains why the question matters. Think about the different answers there could be - what is different about those worlds?
- Is it readable by a bright highschooler without being condescending? Check for unnecessarily complex wording or sentence structure.
- Does it adhere to the *style guide*?

1.33.5 Compensation

- If providing compensation, have they included any conditions for payment? (e.g., child needs to be in age range, child needs to be visible at some point, only one per child)
- Is compensation dependent on the child completing the study, or on the child's behavior in any way? (This is generally not allowed.)
- If providing compensation, have they included information about how long it will take to receive? (Make sure this is consistently stated throughout the study!)

1.33.6 Participant eligibility

- Is the participant eligibility description easy to understand? (E.g., translate ages into commonly-used terms; don't say your study is for children between 56 and 70 weeks old.)
- Are any eligibility criteria beyond age either language-based (e.g., speaking English or being bilingual) or rare (e.g., ASD)? We *generally ask that other criteria be implemented as part of analysis, rather than preventing families from participating.*.

1.33.7 Duration

- Measure how long the study takes you to preview and let the study authors know. Is the duration listed accurate?

1.33.8 Initial setup

- Are webcam setup & consent steps included? Does the information in the consent form make sense and avoid repetition?
- Are these at the start of the study, or if they are later is there a good reason (and are they still before any data collection, including video recording)?

1.33.9 Instructions

- If children need to be visible or arranged a particular way, do you get a chance to look at the webcam setup right before the study starts?
- If parents are facing away or have their eyes closed, is it clear when they need to do that and when they can stop? Are there any points where it might seem like there's a problem with the study if they can't see what's going on?
- Is it clear what you as a parent should be doing during the study?
- Are the directions friendly? (i.e. don't want to sound demanding/condescending)
- Do things "flow"? Are there abrupt transitions?
- Are the instructions clear and straightforward (to the point you could read them while also supervising/holding a few children)? Is there ever an overwhelming amount of info on the screen at once?

1.33.10 Test trials

- Is there an indication to the parent of progress through the study during test trials if possible, especially if the parent needs to be quiet or keep their eyes closed?
- Is audio clear enough to understand & reasonably well-balanced for volume throughout?
- Do you have any concerns about how data collection will work (e.g. whether children will be familiar with the 'familiar objects', how stimuli will look to a colorblind child, etc.) or suggestions?

1.33.11 Debriefing (after exit survey)

- Did they clearly explain the point of the study again (as in the purpose field, this needs to actually get at why the question matters)?
- Did they concretely walk through the study design and explain HOW the study will answer the question? This is the heart of the debriefing. Generally this will entail briefly explaining what happened during the study, what the dependent measure is and what it indexes if that's not obvious, and an if-then prediction: e.g., if babies realize that she doesn't know where the ball is, we expect them to look longer when she finds it right away, because that's surprising!
- Did they explain the multiple conditions if there was randomization?
- Did they head off likely potential parental concerns/objections? e.g.,
  - there are many reasons a child might answer a particular way on any given trial (e.g., first/last option, favorite objects), that's why we average over lots of kids/trial types
  - make sure parents know their child may not have answered a particular way/ looked more or less on a given trial/ or successfully performed some action and that's OK
- Did they restate information about compensation and when to expect it? (make sure this is the same throughout the study)
- Did they link to someplace to learn more about this general topic if possible? (e.g. TED talk, popular science article, website with more games, journal paper, other educational video, etc.) Feel free to share ideas!

1.33.12 General things to think about

- Are any questions/tasks ambiguous or inappropriate for…
  - A single parent (due to choice, breakup/divorce, or death), an unmarried but partnered parent, a parent with a same-sex partner, a divorced parent who shares custody, a parent with more than one partner
  - A family that lost a child in infancy (e.g. "how many siblings" type questions)
  - Multiracial families (e.g. questions about race where it's ambiguous whether you care about child, parent, or both)
  - Adoptive parents (e.g. questions about prenatal history)
  - A parent under 20 (e.g. educational background qs may be less informative measures)
  - A family of a child born very prematurely and whose adjusted age does not match her chronological age, or who has developmental delays
  - A transgender parent or parent of a gender-nonconforming child
  - You / someone you know! (This is not meant as an exhaustive list, just some examples of places where questions sometimes reveal hidden assumptions.)
- Are tasks/questions appropriate for the age range?
- Is the study aesthetically pleasing to look at? (remember parents and children need to be able to stay engaged and we don't want things to come off too "sterile")
- Is all audio clear and easy to understand? Is it as engaging as possible (intonation, pauses, etc.) given the constraints of the study? (Sometimes we default to an unnecessarily flat tone.)
- Are there any typos?
- Are there enough signposts to clearly direct you on what will be happening next?

1.34 Self review checklist

Here is a checklist to work through as you prepare to submit your study. Lookit staff will look through these categories as well during internal review, so you can avoid unnecessary rounds of review by thinking through each item before submission.

1.34.1 Title

- Is it short and memorable?

  Note: it's fine for a title to be cute but not super informative, as long as it has something to do with the content or purpose of the study and could be used to distinguish it from other similar studies. One of its most important functions is to be memorable enough that parents can say "hey I did the XYZ study and [you should too! / I have a question about when my gift card will arrive / etc.]

  Feel free to suggest any clever ideas you have!
- Would a follow-up study, another study from the same group, or another study using the same method fall naturally under the same title? If so, the name should be more specific.
- Does it include information that's covered elsewhere, e.g. eligibility criteria or duration? Don't duplicate this here. (E.g., "XYZ for 5-month-olds" or "XYZ: a 10-minute study about cats" should just be XYZ.)

Examples

Good examples:

- "A fish, THE fish, MY fish" (babies' understanding of articles)
- "Baby see, baby do" (neonatal imitation)
- "Cue the music!" (singing in a language cueing attention to that language)
- "Is two (voices) a crowd?" (kids' ability to understand speech with one/many background voices)
- "Little drummers" (toddlers' production of rhythm)
- "Does your baby know what you are thinking?" (infant theory of mind)

Examples of titles we asked for more specific versions of:

- "Thinking about actions"
- "Look at the pictures"
- "Emotion learning study"

1.34.2 Thumbnail image

- If people are pictured, and they're white - do they need to be? (It's easy for too many researchers to "default to" pictures of white kids.)
- Is this image relevant to the study? (e.g., does it have a child pictured that's in the age range, does it include potential images that the child may see during the study, etc.)
- Do you have rights to use this image royalty-free, either based on the license or because you created or bought rights to the image?

1.34.3 Short description / "What happens"

- Does this clearly explain to parents what they/their child will be doing during this experiment in a few sentences?
- Does it contain unnecessary detail (e.g. exact numbers or timing of trials) or information covered in other fields (e.g. duration)? These should be removed and covered only in the other field to avoid duplication and make it easier to keep information consistent if it changes.
- Does it adhere to the *style guide*?

1.34.4 Discoverable

When you start a new study, we recommend making it non-discoverable while you pilot. That way you can take your time fixing any issues that come up during piloting. Once it's discoverable, it will be listed on the Lookit "studies" page and email announcements will be sent out to all families with a child who's eligible.

- If you want your study posted on the Lookit studies page once you start collecting data, is the "discoverable" box checked? If you don't (because you want to pilot or recruit only from your own database), is it unchecked?

1.34.5 Researcher contact information

- Is this in the format "PIs Name (contact: youremail@lab.edu)" ?

1.34.6 Exit URL

- Is this someplace reasonable (e.g. https://lookit.mit.edu/studies/history)? Make sure it's not the staging server/localhost/etc.

1.34.7 Purpose / "What we're studying"

No seriously, please take a hard look at this section.

This is the single biggest challenge for researchers. Writing a clear and accurate description of the point of your study is HARD but worthwhile, and because this is a critical element of parent communication we're sticklers for all of the below.

- Is it truly accurate given the study design?
- Does it explain what this particular study addresses (not a broader research program)? Example: the purpose of your study is probably not to understand how babies learn new words.

  Note: it's completely ok if your study is not testing some novel question. You might be developing or testing out a measure to see if it works at all (e.g., "can we get parents to live-code infant looking?"), confirming that you can replicate a lab-based finding, norming stimuli for future work on X, trying to pin down a more precise estimate of an effect size for a model, collecting a dataset that will be used for a variety of exploratory work, etc. In that case, say that, and include a brief explanation as appropriate about the point of the related work (e.g., the work you're replicating).
- Does it explain why this matters? (This is not a grant agency - don't stretch it - but "this hasn't been studied before" isn't a reason something matters. Neither is "ability Y is used in skill X which is important" a reason to find out whether infants can do Y, on its own.
- Is it readable by a bright highschooler without being condescending?
- Does it adhere to the *style guide*?

1.34.8 Compensation

- If providing compensation, have you included any conditions for payment (e.g., child needs to be in age range, child needs to be visible at some point, only one per child)
- If providing compensation, have you included information about how long it will take to receive? (Make sure this is consistently stated throughout the study!)
- Is compensation dependent on the child completing the study, or on the child's behavior in any way? (This is generally not allowed per terms of use - check with us if you have questions.)
- Be prepared to really compensate people in that timeframe! If you've said three days, that means that you have through Monday for participants from Friday. They may be counting on the money.

1.34.9 Eligibility description, min/max ages, eligibility criteria expression (all self-review & Lookit review only)

- Are any eligibility criteria beyond age either language-based (e.g., speaking English or being bilingual) or rare (e.g., ASD)? We *generally ask that other criteria be implemented as part of analysis, rather than preventing families from participating.*.
- Don't specify the age range in the criteria expression in addition to the min/max ages (it just introduces some potential for confusion if you later change one).
- Is the participant eligibility description easy to understand? (E.g., translate ages into commonly-used terms; don't say your study is for children between 56 and 70 weeks old.)
- If participants can do the study more than once is that clearly stated?
- Sometimes it can be mildly complex to translate between age range and description. Please review *guidance on aligning ages* to make sure your parent-facing description (e.g., "for 8-month-olds") lines up with your min/max ages.
- Are any additional criteria in the eligibility criteria expression noted in the freeform description?

1.34.10 Duration

- Have you made a realistic estimate of the duration of the study, including setup/consent and children's responses, and confirmed during peer review?

1.34.11 Protocol configuration

- Is your study being randomized correctly? (e.g., you have the right audio and videos for the conditions they're intended to be for) Note: this is NOT something Lookit staff will confirm for you during review; we will generally run through one random condition focusing on communication and any technical issues.
- Are the audio/videos running the way you want them to? (e.g. video is located in the right place on the screen) Again, this is NOT something Lookit staff will confirm for you as we don't know how you wanted them to look!
- Are all stimuli hosted at URLs starting with https://, not http://? (Insecure hosts won't be allowed for both security and performance reasons.)

1.34.12 Version of experiment runner

- Are you using a recent version of the experiment runner? (If not why?)

1.34.13 Initial setup

- Are webcam setup & consent steps included? Does the information in the consent form make sense and avoid repetition?
- Are these at the start of the study, or if they are later is there a good reason (and are they still before any data collection, including video recording)?

1.34.14 Instructions

- If children need to be visible or arranged a particular way, do you give the parent a chance to look at the webcam setup right before the study starts?
- If parents are facing away or have their eyes closed, is it clear when they need to do that and when they can stop? Are there any points where it might seem like there's a problem with the study if they can't see what's going on? Please ACTUALLY TRY your study following the directions given to parents.
- Is it clear what you as a parent should be doing during the study?
- Are the directions friendly? (i.e. don't want to sound demanding/condescending)
- Do things "flow"? Are there abrupt transitions?
- Are the instructions clear and straightforward (to the point you could read them while also supervising/holding a few children)? Is there ever an overwhelming amount of info on the screen at once?

1.34.15 Test trials

- Is there an indication to the parent of progress through the study during test trials if possible, especially if the parent needs to be quiet or keep their eyes closed?
- Have you run all your stimuli through a simulator like https://www.color-blindness.com/coblis-color-blindness-simulator/ to check whether kids with common forms of colorblindness will be able to see them? (Note that few parents of preschoolers and younger will know yet if their kids are colorblind. Even some adults find out by surprise!)
- Is audio clear enough to understand & reasonably well-balanced for volume throughout (e.g., not super-loud music with very quiet speech, can use software like Audacity to normalize your audio)

1.34.16 Debriefing (after exit survey)

- Did you clearly explain the point of the study again (as in the purpose field, this needs to actually get at why the question matters)?
- Did you concretely walk through the study design and explain HOW the study will answer the question? This is the heart of the debriefing. Generally this will entail briefly explaining what happened during the study, what the dependent measure is and what it indexes if that's not obvious, and an if-then prediction: e.g., if babies realize that she doesn't know where the ball is, we expect them to look longer when she finds it right away, because that's surprising!
- Did you explain the multiple conditions if there was randomization?
- Did you head off likely potential parental concerns/objections? e.g.
  - there are many reasons a child might answer a particular way on any given trial (e.g., first/last option, favorite objects), that's why we average over lots of kids/trial types
  - make sure parents know their child may not have answered a particular way/ looked more or less on a given trial/ or successfully performed some action and that's OK
- Did you restate information about compensation and when to expect it (make sure this is the same throughout the study)
- Did you link to someplace to learn more about this general topic if possible? (e.g. ted talk, popular science article, website with more games, journal paper, other educational video, etc.)
- You can use /n/n to add line breaks for readability and can insert links as <a href="https://. . . " target="_blank" rel="noopener">Cool Website</a>

1.34.17 General things to think about

- Are any questions/tasks ambiguous or inappropriate for…
  - A single parent (due to choice, breakup/divorce, or death), an unmarried but partnered parent, a parent with a same-sex partner, a divorced parent who shares custody, a parent with more than one partner
  - A family that lost a child in infancy (e.g. "how many siblings" type questions)
  - Multiracial families (e.g. questions about race where it's ambiguous whether you care about child, parent, or both)
  - Adoptive parents (e.g. questions about prenatal history)
  - A parent under 20 (e.g. educational background qs may be less informative measures)
  - A family of a child born very prematurely and whose adjusted age does not match her chronological age, or who has developmental delays
  - A transgender parent or parent of a gender-nonconforming child
  - You / someone you know! (This is not meant as an exhaustive list, just some examples of places where questions sometimes reveal hidden assumptions.)

  In general think about what information you actually need and ask for that specifically.
- Are tasks/questions appropriate for the age range?
- Is the study aesthetically pleasing to look at? (remember parents and children need to be able to stay engaged and we don't want things to come off too "sterile")
- Is all audio clear and easy to understand? Is it as engaging as possible (intonation, pauses, etc.) given the constraints of the study? (Sometimes we default to an unnecessarily flat tone.)

1.35 Editing this documentation

1.35.1 Basic instructions

If there are changes you'd like to make to a single page - for instance, correcting a typo or adding an explanation of something that confused you - you can follow *the step-by-step directions in the tutorial* to make your changes right on Github.

1.35.2 Advanced instructions

If you want to make bigger changes, such as reorganizing content or adding new pages, it'll be easiest to work with a local copy of the documentation.

Documentation for use of the Lookit platform (what you're reading now!) lives in the lookit-docs repo. You can fork this repository to create your own copy of it on Github, and then clone that fork so you have a local copy of the docs to edit in a familiar text editor.

The file `index.rst` contains the table of contents (look for `toctree`). Documentation is written using ReStructured Text (RST) markup. For consistency we are trying to keep all documentation in .rst format. If you are more familiar with Markdown, you can convert between formats using Pandoc, e.g.:

```
pandoc -o outputfile.rst inputfile.md
```

If you are making substantial changes, you will want to take a look at how those changes look locally by using Sphinx to build your own local copy of the documentation. To do this, first create another virtual environment and install the requirements for Sphinx there:

```
/lookit-docs $ virtualenv -p python3 denv
/lookit-docs $ source denv/bin/activate
(denv) /lookit-docs $ pip install -r docs/requirements.txt
```

You can then build the docs from within the `docs` directory:

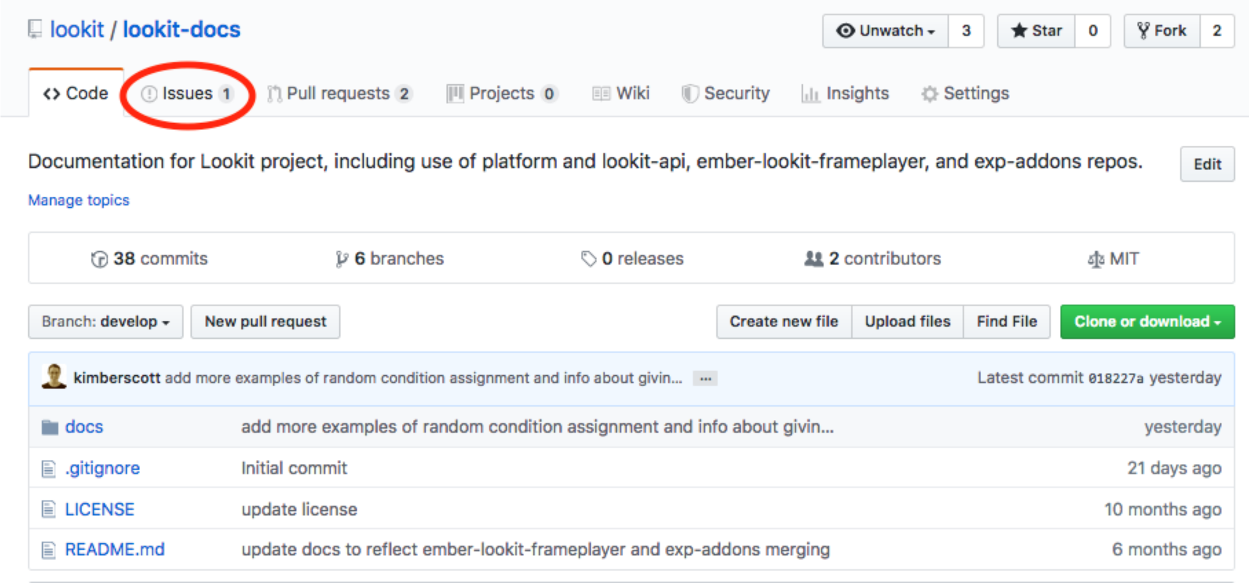
```
(denv) /lookit-docs/docs $ make html
```

Navigate to `docs/build/html/index.html` from your favorite web browser to inspect the docs.

Once you are happy with your changes, commit them using git and push the changes back up to your fork of the docs on Github. Then create a PR to the `lookit-docs/develop` branch; when it's merged, the docs served by ReadTheDocs at https://lookit.readthedocs.io will be automatically updated! (Note that it is easy to have ReadTheDocs serve multiple versions of the documentation, from different branches; we just haven't reached the point of that being more useful than confusing yet.)

1.36 Using Github issues

We use Github Issues to plan Lookit development. This is a central tool where we keep information about known bugs, planned new features, and what's going to be addressed when.



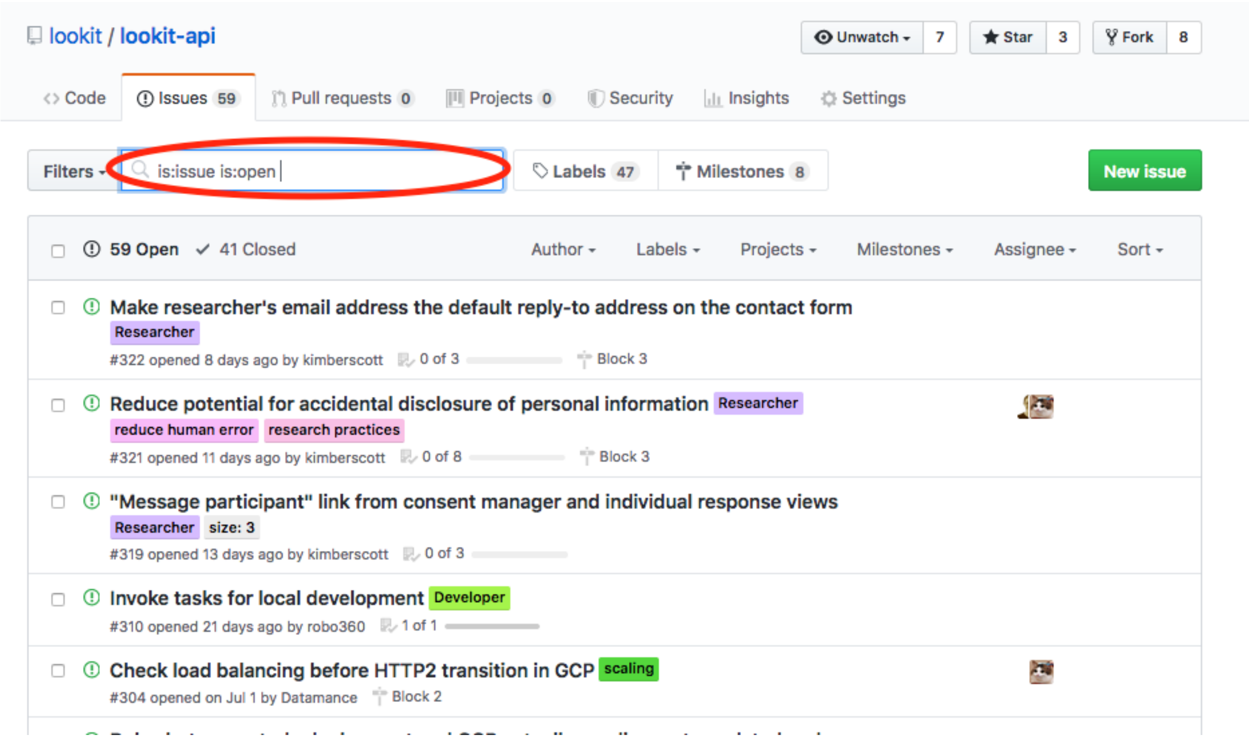
Documentation for Lookit project, including use of platform and lookit-api, ember-lookit-frameplayer, and exp-addons repos.

kimberscott add more examples of random condition assignment and info about givin... … Latest commit 018227a yesterday

docs	add more examples of random condition assignment and info about givin...	yesterday
.gitignore	Initial commit	21 days ago
LICENSE	update license	10 months ago
README.md	update docs to reflect ember-lookit-frameplayer and exp-addons merging	6 months ago

If you report a problem or mention an idea you have to improve Lookit through some other channel, we will probably ask that you make a Github issue so we can keep track of your idea and get feedback from any other researchers affected.

To request a feature or report a bug, first search the existing issues to see if your idea is already there.



is:issue is:open

□ 59 Open ✓ 41 Closed

□ Make researcher's email address the default reply-to address on the contact form Researcher
#322 opened 8 days ago by kimberscott 0 of 3 Block 3

□ Reduce potential for accidental disclosure of personal information Researcher reduce human error research practices
#321 opened 11 days ago by kimberscott 0 of 8 Block 3

□ "Message participant" link from consent manager and individual response views Researcher size: 3
#319 opened 13 days ago by kimberscott 0 of 3

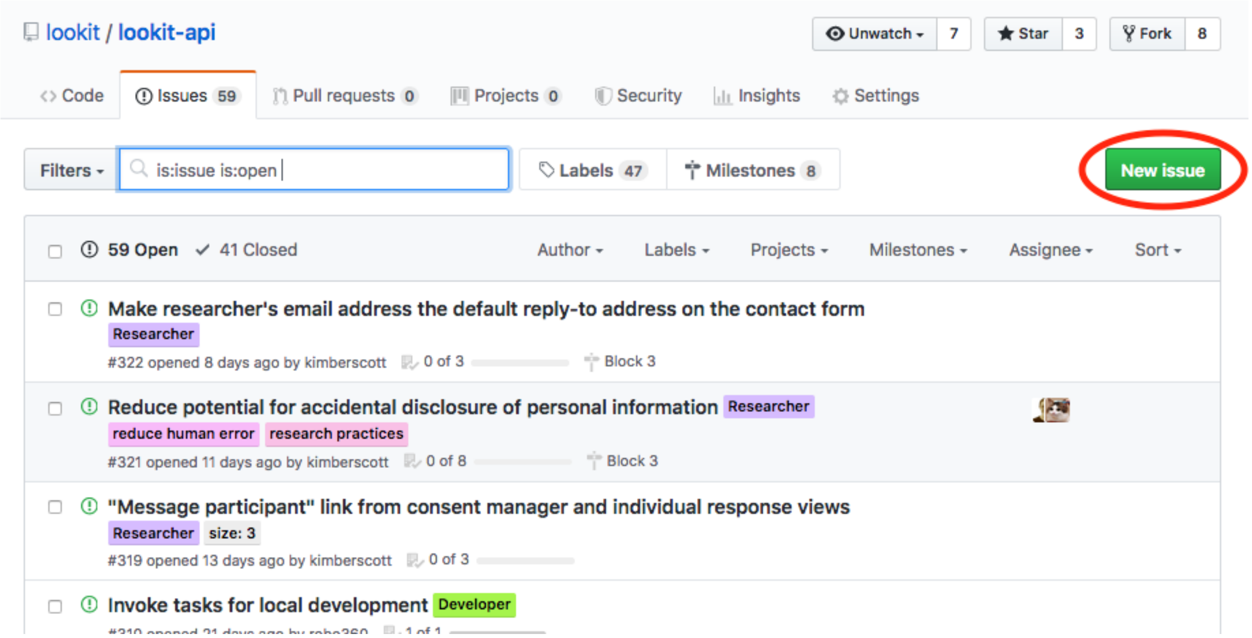
□ Invoke tasks for local development Developer
#310 opened 21 days ago by robo360 1 of 1

□ Check load balancing before HTTP2 transition in GCP scaling
#304 opened on Jul 1 by Datamance Block 2

Depending on the type of problem you are encountering or idea you have, you will want to check in one of the following repositories or "repos":

- lookit-api is the repo for the Lookit site: issues with anything to do with participant login or data, how current and past studies are displayed to participants, how you view data and manage your studies
- ember-lookit-frameplayer is the repo for the experiment components themselves: issues with how particular frames behave, frames you'd find useful, counterbalancing/condition assignment, etc.
- lookit-docs is the repo for the documentation: anything about the docs you're reading now!

If you find a relevant issue already exists, please comment on it or add a thumbs-up reaction so Lookit staff know there's more interest! If not, click the green "New issue" button at the top right.



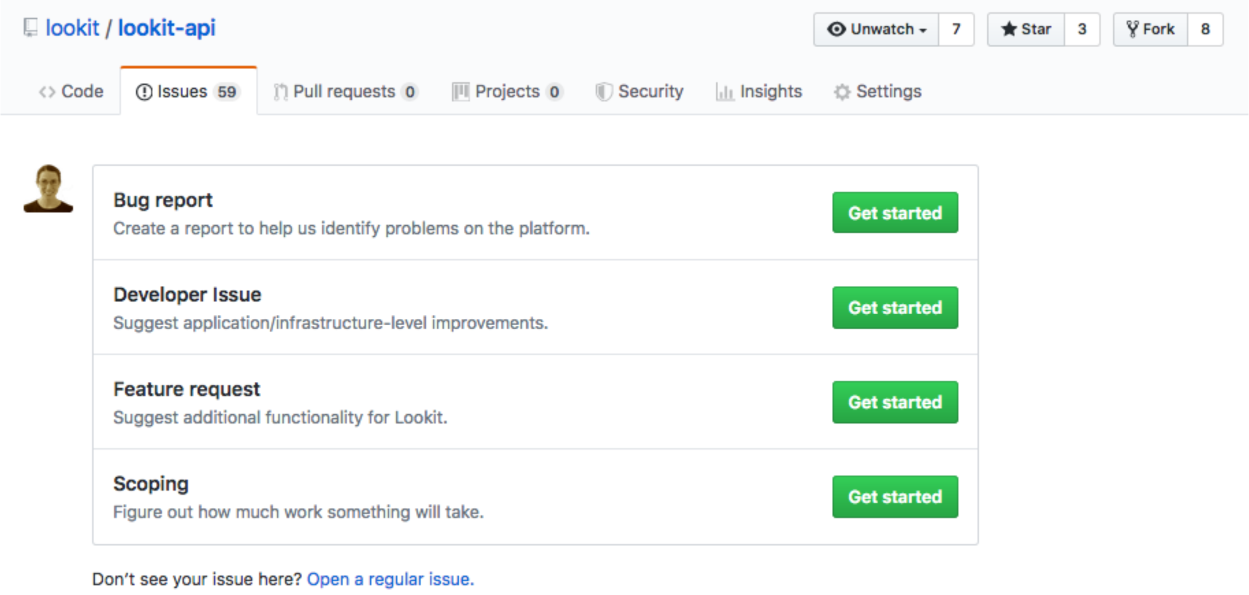
Make researcher's email address the default reply-to address on the contact form Researcher
#322 opened 8 days ago by kimberscott 0 of 3 Block 3

Reduce potential for accidental disclosure of personal information Researcher reduce human error research practices
#321 opened 11 days ago by kimberscott 0 of 8 Block 3

"Message participant" link from consent manager and individual response views Researcher size: 3
#319 opened 13 days ago by kimberscott 0 of 3

Invoke tasks for local development Developer

You may need to select an issue type. Choose the type that's closest to what you want to describe - probably "bug report" or "feature request":



Bug report
Create a report to help us identify problems on the platform. Get started

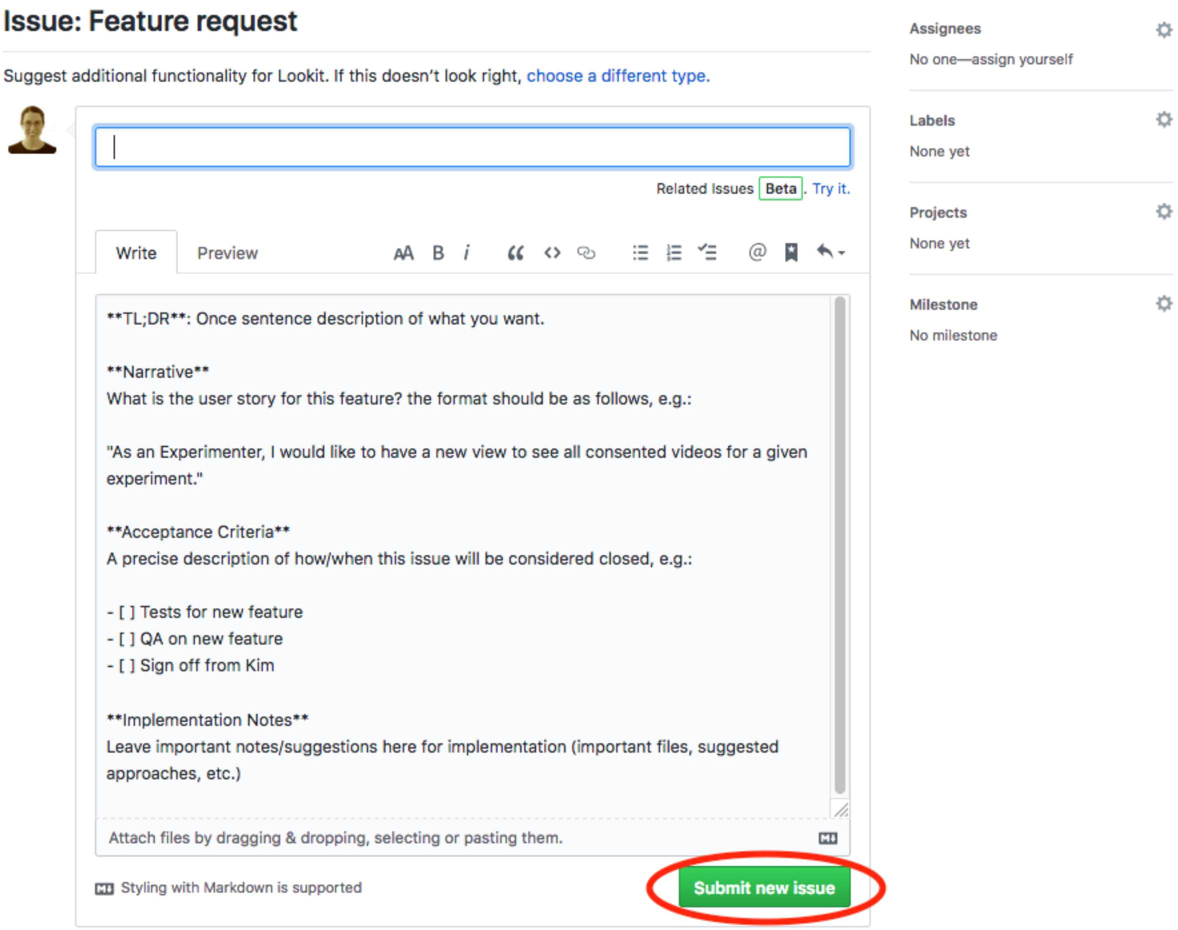
Developer Issue
Suggest application/infrastructure-level improvements. Get started

Feature request
Suggest additional functionality for Lookit. Get started

Scoping
Figure out how much work something will take. Get started

Don't see your issue here? Open a regular issue.

If you had to select an issue type, you'll now have a template to fill in with information. If you're not using a template, try to give a clear one-sentence summary of the problem or requested feature/change, followed by any details needed to reproduce the problem or understand the proposed change. Then click the green "Submit new issue" button to create your issue.



Issue: Feature request

Suggest additional functionality for Lookit. If this doesn't look right, choose a different type.

TL;DR: Once sentence description of what you want.

Narrative
What is the user story for this feature? the format should be as follows, e.g.:

"As an Experimenter, I would like to have a new view to see all consented videos for a given experiment."

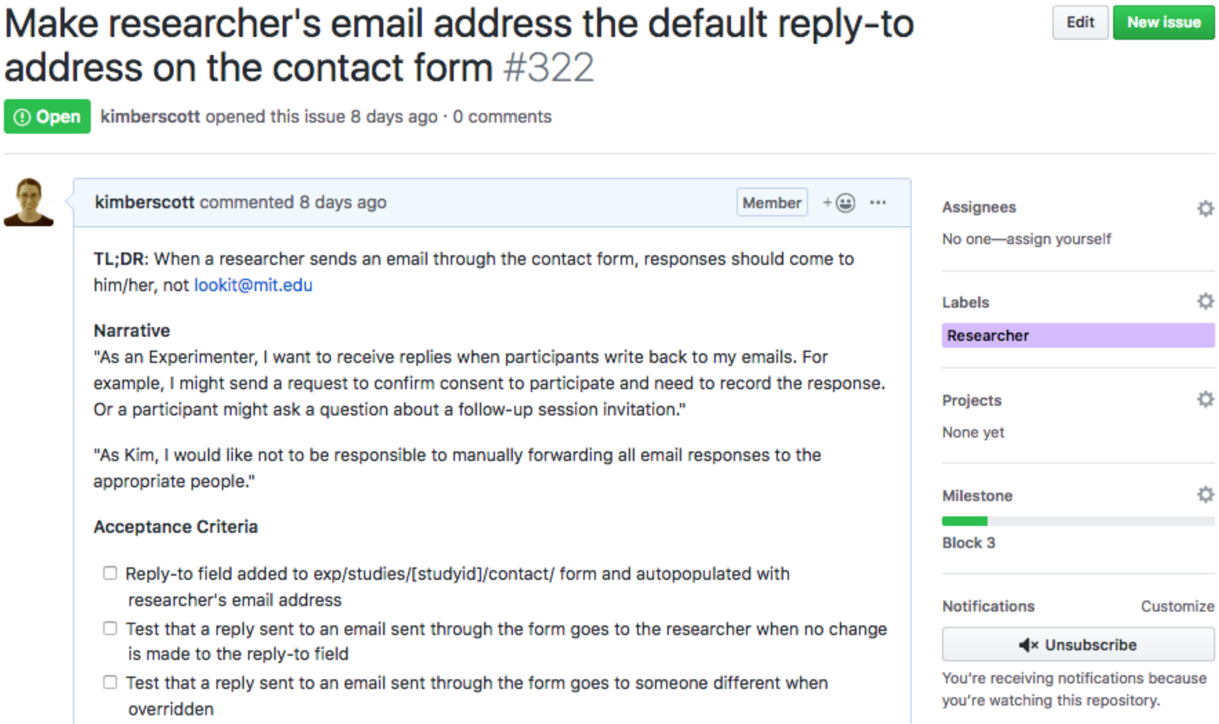
Acceptance Criteria
A precise description of how/when this issue will be considered closed, e.g.:

- [ ] Tests for new feature
- [ ] QA on new feature
- [ ] Sign off from Kim

Implementation Notes
Leave important notes/suggestions here for implementation (important files, suggested approaches, etc.)

Related Issues Beta. Try it.
Write
Preview
Attach files by dragging & dropping, selecting or pasting them.
Styling with Markdown is supported
Submit new issue
Assignees
No one—assign yourself
Labels
None yet
Projects
None yet
Milestone
No milestone

Your issue will now have a number assigned to it and will be listed in the issue list you looked at earlier:



Make researcher's email address the default reply-to address on the contact form #322

Open kimberscott opened this issue 8 days ago · 0 comments

kimberscott commented 8 days ago

TL;DR: When a researcher sends an email through the contact form, responses should come to him/her, not lookit@mit.edu

Narrative

"As an Experimenter, I want to receive replies when participants write back to my emails. For example, I might send a request to confirm consent to participate and need to record the response. Or a participant might ask a question about a follow-up session invitation."

"As Kim, I would like not to be responsible to manually forwarding all email responses to the appropriate people."

Acceptance Criteria

- □ Reply-to field added to exp/studies/[studyid]/contact/ form and autopopulated with researcher's email address
- □ Test that a reply sent to an email sent through the form goes to the researcher when no change is made to the reply-to field
- □ Test that a reply sent to an email sent through the form goes to someone different when overridden

Lookit staff may respond to ask for further information, schedule it for future development, and/or wait for community feedback about the idea to gauge demand.

1.37 Overview of Lookit architecture

1.37.1 Codebase and stack

Although we present participants and researchers with an integrated experience that has an intuitive user interface, behind-the-scenes the Lookit codebase is three independent projects:

- lookit-api is the primary application that supports study management and participation. It contains a Django (Python) application, which provides an HTTP/JSON API consumed by built experiments as well as web UIs for researchers to design and manage experiments, for participants to discover experiments, and for administrators to manage accounts and static website content. It also contains offline Celery processes (connected by a RabbitMQ message queue) to build and deploy experiments using Docker, and to send email. Django provides a solid framework and abstractions for development of views for both participants and experimenters.
- ember-lookit-frameplayer is the "experiment runner" that parses a JSON study protocol and presents a sequence of interactive pages to the participant. It is implemented in the Javascript framework Ember, which is well-suited to the structure of providing a "library" of reusable, customizable frame components. New components are written in Javascript, HTML (Handlebars), and SCSS. Parameters and outputs of these components are automatically documented using Yuidoc, along with example screenshots and videos. This experiment runner is entirely separable from the other elements of Lookit; the code is already well structured to accommodate alternate experiment types.
- lookit-orchestrator contains the continuous integration and continuous deployment (CI/CD) tools for Lookit - in particular, tools for automatically testing, building, and deploying code as changes are pushed to GitHub. This encourages the frequent incorporation of small changes to make development more robust. We use a Kustomize-based workflow with a standard containerized GitOps execution pathway.

There are three major out-of-network service dependencies:

- Amazon S3 for primary storage of participant video,
- Google Cloud Storage for storing experiment web archives and other static files, and
- Pipe to relay video streamed from experiments to Amazon S3 upon verification.

We also use Sendgrid for sending email to researchers and participants from the lookit-api web UI, and Sentry for frontend and backend error logging.

1.37.2 How it all works together

The Lookit ecosystem can be described as a sort of "restaurant" architecture, if the restaurant had a prix fixe menu. It's a cheeky and imperfect analogy, but it gets the general idea across. In this analogy there are a couple of main players you need to know about:

1. **experiments** are *meals*

- packaged/built web archives loaded into Google Cloud Storage (GCS) and proxied/served by the web application (see point 4 below)
- Right now, the only experiment type we support is ember-lookit-frameplayer; theoretically we could create adapters for lookit for different experiment front-ends.

2. **researchers** are *head chefs*

- they design experiments and then build their dependencies with a process that uploads them to GCS.

3. **participants** are *customers*

- they use the wait staff to order meals that are on the menu.

4. The **web application** is your *wait staff*

- They show off the menu to customers
- They also take orders from the head chef to add, remove, or change items on the menu, and relay those to the kitchen (see point 5 below)

5. The **builder** is your *kitchen*

- receives orders from the head chef by way of the wait staff.
- actually cooks "meals" (builds experiments)
- RabbitMQ kind of acts like a stack of order slips, in this system.

6. The **worker** is kind of like a *general manager*

- sends out mail (email) to previous customers
- occasionally, you'll see him sweeping the kitchen (cleaning up junk left over from docker builds)

The "Frameplayer" and video data

[Ember-lookit-frameplayer](https://github.com/lookit/ember-lookit-frameplayer) is an Ember application that can be "built" into a web archive (WAR) with bundled/minimized Javascript and CSS. It's the default experiment type, and currently the only one available.

To capture video data, we use [Pipe](https://addpipe.com/). The JS client is basically a plugin that is parameterized during the WAR build process; when properly configured it will stream video data to Pipe's servers. This data is then uploaded to Amazon S3, which, upon completion, triggers a webhook that fires a POST payload to a special [handler in our API](https://github.com/lookit/lookit-api/blob/master/exp/views/video.py#L17) that finds the video in S3 and renames it to something more searchable.

1.37.3 Cluster layout

Lookit is organized as a collection of Kubernetes services, backed by deployments and statefulsets:

:: code:

```
 kubectl get services
NAME                               TYPE           CLUSTER-IP   EXTERNAL-IP           ␣
↪   PORT(S)                                          AGE
kubernetes                         ClusterIP      10.x.x.x     <none>                ␣
↪   443/TCP                                          89d
staging-gcloud-sqlproxy            ClusterIP      10.x.x.x     <none>                ␣
↪   5432/TCP                                         89d
staging-google-storage             ExternalName   <none>       storage.googleapis.com␣
↪   <none>                                           89d
staging-lookit-rabbitmq            ClusterIP      10.x.x.x     <none>                ␣
↪   4369/TCP,5672/TCP,25672/TCP,15672/TCP,9419/TCP   89d
staging-lookit-rabbitmq-headless   ClusterIP      None         <none>                ␣
↪   4369/TCP,5672/TCP,25672/TCP,15672/TCP            89d
staging-web                        NodePort       10.x.x.x     <none>                ␣
↪   8080:32403/TCP                                   89d

 kubectl get deployment
NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
staging-beat              1/1     1            1           89d
staging-gcloud-sqlproxy   1/1     1            1           89d
staging-web               1/1     1            1           89d
staging-worker            1/1     1            1           89d

 kubectl get statefulset
NAME                      READY   AGE
staging-builder           1/1     89d
staging-lookit-rabbitmq   1/1     89d
```

- The backing monorepo for the `-web` (web application), `-builder` (experiment builder), `-worker` (celery tasks), and `-beat` (celery crons) resources is [lookit-api](https://github.com/lookit/lookit-api), which is a django application.
- The `-gcloud-sqlproxy` resources define as a single point of egress out to a Google Cloud SQL instance, designated by the configuration file for a given environment (production example)
- The `-lookit-rabbitmq` resources define a rabbitmq message queue that serves as a conduit between the web application and the task runner and builder.
- *-google-storage* is basically just an external service that we set up to allow nginx ingress to reroute requests for static assets to GCS.

1.37.4 Setup for a separate instance of Lookit

Please contact us if you are looking to run your own instance; it will be a good idea to work together closely to both get you up and running, which will also provide critical feedback in making this pipeline more adaptable.

A good place to start if you are interested in running your own separate instance of Lookit is the lookit-orchestrator README. To fill in a bit more:

Google Cloud Platform (GCP)

We rely almost exclusively on GCP components to orchestrate the app. A Cloud Builder Github integration trigger is tripped on deployments to either the "develop" or "master" branches of *lookit-api*, executing the "CI" piece of the pipeline ([testing and containerization](https://github.com/lookit/lookit-api/blob/master/cloudbuild.yaml)). You can see in the *deploy-to-cluster* step that the "CD" ([deployment](https://github.com/lookit/lookit-api/blob/master/cloudbuild.yaml#L68)) piece is executed near the very end. It leverages the contents using the contents of _this_ repo, which are similarly containerized (using the GitHub integration for build triggers) and loaded into GCR for use as a [custom builder](https://cloud.google.com/cloud-build/docs/configuring-builds/use-community-and-custom-builders).

The CI pipeline is not completely generalized/parameterized, so to run your own Lookit CI pipeline, you'll want to set up your own brand new environment apart from the one that is used by the MIT instance of Lookit. To accomplish this, you'll need to set up your own Google Cloud Platform project. You'll need a few things turned on: - Kubernetes Engine - Cloud Builder - Container Registry - Key Management Service

Once those services are turned on, you'll want to turn your focus to the GKE setup that is tuned by the lookit-orchestrator repo.

Kubernetes (*lookit-orchestrator* configuration)

So far, we see quite a number of "in-network" players (webapp, builder, worker, etc.) and "out-of-network" services (Pipe, S3, Google Cloud Storage, Google Cloud SQL). While Kustomize and Kubernetes work tightly together to connect "in-network" players, "out-of-network" services all need login credentials, which are safely and securely built into deployed k8s pods based on the setup described here. To configure your Kubernetes setup, we recommend following these steps:

1. Fork the lookit-orchestrator repo, as well as lookit-api
2. Change the configs to match your new environment (in fact, you will probably need to make changes to all of the files in that environment directory to suit your particular environment).
3. Create a secrets file for your new environment ignored such that it is never checked in. You'll author this in basically the same way as the config env file; to see which secrets you'll need you can take a look at the `secretKeyRef`'d env vars listed in the env var patch.
4. Run one of the *make encrypt* hooks here to encrypt your plaintext secrets into *.enc* versions that can be checked into source control (which facilitates GitOps deployment - you can see in the deploy script where the secrets are actually decrypted using GKMS.)

The pipeline is not fully parameterized to target arbitrary clusters, so you'll also need to edit the actual deployment line of the script to target whatever zone/region you're deploying to in Europe. **There are probably other things that we're missing at the moment** (Sentry setup, for instance, is baked into the app and CI/CD pipeline - you'll need to figure out if you want alerting and modify accordingly - but this should give you a decent start.)

1.38 Developing new frames

Suppose that for your study, you need a frame that's not part of the standard ember-lookit-frameplayer library. Maybe you want to use a particular game you've already implemented in Javascript, or you want to slightly change how one of the existing frames works, or you want to hard-code a particular complicated counterbalancing scheme. That's okay! You can add a new frame to your own version of the ember-lookit-frameplayer repository, and tell Experimenter to use your Github fork of ember-lookit-frameplayer (rather than the standard Lookit-maintained one) when building your study.

The Lookit codebase is composed of two main modules:

- lookit-api: The repo containing the Lookit Django app - database structure for studies, responses, children, users, etc; login and account management; study management and data download tools for researchers; and an API to allow fetching and updating data (e.g. from an experiment)
- ember-lookit-frameplayer: A small Ember app that provides the rendering engine and experiment frames for Lookit studies, talking to the Lookit API to fetch and update data. We use the term 'frame' to describe the combination of JavaScript file and Handlebars HTML template that compose a particular component of an experiment.

Generally, all frame development will happen in ember-lookit-frameplayer. If you want to develop a new frame, or make improvements to existing frames, you will likely want to run ember-lookit-frameplayer locally so that you can see and test out your changes instantly. This way, you don't have to repeatedly push your changes to GitHub and re-build your study to see how it works.

To start developing your own frames, first follow the "Setup for local development" steps.

1.38.1 Setup for local development

These instructions will walk you through setting up to run Lookit locally.

Overview

Option 1:

For a full local development setup, we will need to install *both* the the Django app (`lookit-api`) and the Ember app (`ember-lookit-frameplayer`), tell them how to talk to each other, and run both of those servers locally.

- On Lookit, we will add some basic information to our superuser, and then add a child and demographic data.
- We then create a study locally.
- In ember-lookit-frameplayer, we'll add a token which gets added to the headers of the API requests so that Lookit knows about the logged-in user making the request.
- We can then navigate directly to the study from the Ember app to bypass the build process locally.

Option 2:

If you are only making changes to the experiment runner (ember-lookit-frameplayer), you have the option of running only the Ember app locally, and having it talk to the Lookit staging server instead of a local Lookit server.

Either way, you will be able to make changes to frames locally and immediately see the results of those changes, participating in a study just as if you were a participant on the Lookit website. You will edit the study definitions, and see the collected data, on your local instance or on the staging server, depending on the option you choose.

Django App steps

Note: this is optional if you are only making changes to ember-lookit-frameplayer.

1. Follow the instructions to install the django app locally. Run the server.
2. Navigate to https://localhost:8000/login/ to log in to Lookit. Use the superuser credentials created in the django installation steps, and set up 2-factor authentication so you'll be able to access the researcher and admin interfaces

3. Navigate to https://localhost:8000/__CTRL__ to access the Admin app. Navigate to users, and then select your superuser. If you just created your django app, there should be two users to pick from, your superuser, and an anonymous user. In that case, your superuser information is here https://localhost:8000/__CTRL__/accounts/user/2/change/.

4. Update your superuser information through the admin app. Fill out the bold fields:

   - Family Name: *Your last name*
   - Identicon: *If no identicon, just type random text here*
   - Locale: *en_US, as an example*
   - Place a check in the checkbox by "Is Researcher"

   Click "Save".

5. Create a token to allow the Ember app to access the API by navigating to https://localhost:8000/__CTRL__/authtoken/token/. Click "Add Token", find your superuser in the dropdown, and then click "Save". You will need this token later.

6. Create a study by navigating to https://localhost:8000/exp/studies/create/. Fill out all the fields. The most important field is the `structure`, where you define the frames and the sequence of the frames. Be sure the frame and the details for the frame you are testing are listed in the structure.

7. Add demographic information to your superuser (just for testing purposes), so your superuser can participate in studies. Navigate to https://localhost:8000/account/demographics/. Scroll down to the bottom and hit "Save". You're not required to answer any questions, but hitting save will save a blank demographic data version for your superuser.

8. Create a child by navigating to https://localhost:8000/account/children/, and clicking "Add Child". Fill out all the information with test data and click "Add child".

Now we have a superuser with attached demographic data and a child. We've created a study, as well as a token for accessing the API. Leave the django server running and switch to a new tab in your console.

> Remember: The OAuth authentication used for access to Experimenter does not work when running locally. You can access Experimenter by first logging in as your superuser, or by giving another local user researcher permissions using the Admin app.

Ember App steps

1. Follow the instructions to install the ember app locally.

2. If you make changes to the frames, you should see notifications that files have changed in the console where your ember server is running, like this:

   ```
   file changed components/exp-video-config/template.hbs
   ```

3. Add your token and lookit-api local host address to the ember-lookit-frameplayer/.env file. This will allow your Ember app to talk to your local API or to the Lookit staging server, depending on the option you chose above. If you are using a local installation of lookit-api, insert the token you saved earlier. If you are using the Lookit staging server, please contact the admins for an API token for your account. Your .env file will now look like this:

   ```
   PIPE_ACCOUNT_HASH='<account hash here>'
   PIPE_ENVIRONMENT=<environment here>
   LOOKIT_API_KEY='Token <token here>'
   LOOKIT_API_HOST='https://localhost:8000'
   ```

(continues on next page)

(continued from previous page)

```
If you are using the Lookit staging server, this will be identical except that the
last line should be ``LOOKIT_API_HOST='https://lookit-staging.mit.edu'``.
```

4. In order to the HTML5 video recorder, you'll need to set up to use https locally. Open `ember-lookit-frameplayer/.ember-cli` and make sure it includes `ssl: true`:

   ```
   "disableAnalytics": false,
   "ssl": true
   ```

   Create `server.key` and `server.crt` files in the root `ember-lookit-frameplayer` directory as follows:

   ```
   openssl genrsa -des3 -passout pass:x -out server.pass.key 2048
   openssl rsa -passin pass:x -in server.pass.key -out server.key
   rm server.pass.key
   openssl req -new -key server.key -out server.csr
   openssl x509 -req -sha256 -days 365 -in server.csr -signkey server.key -out␣
   ↪server.crt
   ```

   Leave the challenge password blank and enter `localhost` as the Common Name.

5. Run the ember server: `ember serve`

Starting up once initial setup is completed

This is much quicker! Once you have gotten through the initial setup steps, you don't need to go through them every time you want to work on something.

1. Start the Django app:

   ```
   $ cd lookit-api
   $ pipenv shell
   $ invoke server
   ```

2. Start the Ember app:

   ```
   $ cd ember-lookit-frameplayer
   $ ember serve
   ```

3. Log in as your local superuser at http://localhost:8000/login/

Previewing or participating in a study

To participate in a study locally, you need demographic data and a child attached to the logged in user, as well as a study. To fetch studies, navigate to https://localhost:8000/api/v1/studies/. Copy the id of the study you created earlier. To fetch children, navigate to https://localhost:8000/api/v1/children/. Copy the id of your child.

Both previewing and participating will save data to your local server; there's no difference in the experience. Preview responses simply have an "is_preview" field set to True, and are displayed differently on the consent manager and individual responses views.

To preview a study, you need to either have read permissions for the study or the study needs to have "shared preview" set to true. To participate, you do not need any particular permissions.

To participate in a study, navigate to https://localhost:4200/studies/study_id/child_id, replacing study_id and child_id with the ids you obtained from the API. (For simplicity, bookmark this link while you're working!)

To preview, you can instead navigate to https://localhost:4200/exp/studies/study_id/child_id/preview/, replacing study_id and child_id with the ids you obtained from the API.

Where does my video go?

If you have set up the Pipe recorder environment variables as described in the installation instructions, video recorded during your local testing will go to Pipe and then to an S3 bucket for Lookit development video. Please get in touch if you need access to this video. Depending on the project you are working on, we may provide credentials for accessing the dev S3 bucket, or may ask that you set up your own free Pipe account and have it forward data to you own S3 bucket, which will allow you to test more of the process. (In this case you will use ngrok to send a Pipe webhook to your own local instance.)

Further Reading / Useful Links

- https://emberjs.com/
- https://ember-cli.com/
- Development Browser Extensions - https://chrome.google.com/webstore/detail/ember-inspector/bmdblncegkenkacieihfhpjfppoconhi - https://addons.mozilla.org/en-US/firefox/addon/ember-inspector/

1.38.2 Creating custom frames

Getting Started

One of the features of Ember CLI is the ability to provide 'blueprints' for code. These are basically just templates of all of the basic boilerplate needed to create a certain piece of code. To begin developing your own frame:

```
cd ember-lookit-frameplayer
ember generate exp-frame exp-<your_name>
```

Where `<your_name>` corresponds with the frame name of your choice.

A Simple Example

Let's walk though a basic example of 'exp-cat-test':

```
$ ember generate exp-frame
installing exp-frame
   create app/components/exp-cat-test/component.js
   create app/components/exp-cat-test/template.hbs
   create app/components/exp-cat-test/doc.rst
   create app/styles/components/exp-cat-test.scss
```

Notice this created three new files:

- `app/components/exp-cat-test/component.js`: the JS file for your frame
- `app/components/exp-cat-test/template.hbs`: the Handlebars template for your frame
- `app/components/exp-cat-test/doc.rst`: a documentation page for your frame
- `app/styles/components/exp-cat-test.scss`: a boilerplate file that exposes the new frame to the Ember app- you will almost never need to modify this file.

Let’s take a deeper look at the `component.js` file:

```
import ExpFrameBaseComponent from '../exp-frame-base/component';
import layout from './template';

export default ExpFrameBaseComponent.extend({
    type: 'exp-cat-test',
    layout: layout,
    frameSchemaProperties: {
         // define configurable parameters of your frame here in valid JSON Schema␣
↪format.
         // See http://json-schema.org/latest/json-schema-validation.html#rfc.section.
↪5
         // for what forms of validation are available. The frame configuration - i.e.
↪ the
         // object in the study JSON that defines a frame of this type - will be␣
↪validated
         // against a JSON Schema {type: 'object', properties: frameSchemaProperties}.
         // Each property should have a YUIdoc comment as shown in the example below.

         /**
          * Whether to show a picture of a cat.
          *
          * @property {Boolean} showCatPicture
          * @default false
          */
         showCatPicture: {
             type: 'boolean',
             default: false,
             description: 'Whether to show a picture of a cat.'
         }
    },
    meta: {
        name: 'ExpCatTest',
        description: 'TODO: a description of this frame goes here.',
        data: {
            type: 'object',
            properties: {
                // define data to be sent to the server here
            }
        }
    }
});
```

The first section:

```
import ExpFrameBaseComponent from '../exp-frame-base/component';
import layout from './template';

export default ExpFrameBaseComponent.extend({
    type: 'exp-cat-test',
    layout: layout,
...
})
```

does several things:

- imports the `ExpFrameBaseComponent`: this is the superclass that all ‘frames’ must extend

- imports the `layout`: this tells Ember what template to use
- extends `ExpFrameBaseComponent` and specifies `layout:  layout`

Next are the parameters and 'meta' section:

```
...

frameSchemaProperties: {
    showCatPicture: {
        type: 'boolean',
        default: false,
        description: 'Whether to show a picture of a cat.'
    }
},

frameSchemaRequired: ['showCatPicture'],

meta: {
    name: 'ExpCatTest',
    description: 'TODO: a description of this frame goes here.',
    data: {
         /**
         * Parameters captured and sent to the server
         *
         * @method serializeContent
         * @param {String} whatTheChildThoughtAboutTheCat Child response to cat
         */
        type: 'object',
        properties: {
            // define data to be sent to the server here
            whatTheChildThoughtAboutTheCat: {
                type: 'string'
            }
        }
    }
},
...
```

The `frameSchemaProperties` field should be the JSON Schema defining what configuration parameters this 'frame' accepts. When you define an experiment that uses the frame, you will be able to specify configuration as part of the experiment definition. Any parameters in this section will be automatically added as properties of the component, and directly accessible as `propertyName` from templates or component logic.

The `frameSchemaRequired` field is a list of any values in `frameSchemaProperties` that should be required to be defined by the user of the frame.

The 'meta' field is composed of:

- name (optional): A human readable name for this 'frame'
- description (optional): A human readable description for this 'frame'.
- data: JSON Schema defining what data this 'frame' outputs. Properties defined in this section represent properties of the component that will get serialized and sent to the server as part of the payload for this experiment. You can get these values by binding a value to an input box, for example, or you can define a custom computed property by that name to have more control over how a value is sent to the server.

If you want to save the value of a configuration variables, you can reference it in both parameters *and* data. For example, this can be useful if your experiment randomly chooses some frame behavior when it loads for the user, and you want to save and track what value was chosen.

It is important that any fields you define in data be named in camelCase: they can be all lowercase or they can be writtenLikeThis, but they should not start with capital letters or include underscores. This is because the fields from the Ember app will be converted to snake_case for storage in the Postgres database, and may be converted back if another frame in Ember uses values from past sessions. We are fine if we go fieldName -> field_name -> fieldName, but anything else gets dicey! (Note to future developers: some conversations about this decision are available if this becomes a point of concern.)

Building out the Example

Let's add some basic functionality to this 'frame'. First define some of the expected parameters:

```
...
    meta: {
        ...,
        parameters: {
            type: 'object',
            properties: {
                title: {
                    type: 'string',
                    default: 'An adorable cat'
                },
                question: {
                    type: 'string',
                    default: 'Check here if you think this is an excellent cat'
                }
            }
        }
    },
...
```

And also the output data:

```
...,
    data: {
        type: 'object',
            properties: {
                answer: {
                    type: 'boolean',
                    default: false
                }
            }
        }
    }
...
```

Since we indicated above that this 'frame' has an answer property, let's add it to the 'frame' definition:

```
export default ExpFrameBaseComponent.extend({
    ...,
    answer: null,
    meta: {
    ...
    }
...
```

Next let's update template.hbs to look more like a test trial:

```
<div class="well">
  <h1>{{ title }}</h1>
  <hr>
  <p> {{ body }}</p>
  <hr >
  <div class="input-group">
    <span>
      {{ question }}
    </span>
    {{input type="checkbox" checked=answer}}
  </div>
</div>
<div class="row exp-controls">
  <!-- Next/Last/Previous controls. Modify as appropriate -->
  <div class="btn-group">
    <button class="btn btn-default pull-right" {{ action 'next' }} > Next </button>
  </div>
</div>
```

In this silly example we don't want to let the participant continue unless they've checked the box, so let's change the footer to:

```
<div class="row exp-controls">
  <div class="btn-group">
    <button class="btn btn-default pull-right" disabled={{ excellentNotChecked }} {{ action 'next' }} > Next </button>
  </div>
</div>
```

Notice the new property `excellentNotChecked`; this will require a new computed field in our JS file:

```
    meta: {
        ...
    },
    excellentNotChecked: Ember.computed.not('answer')
});
```

Adding CSS styling

You will probably want to add custom styles to your frame, in order to control the size, placement, and color of elements. Experimenter uses a common web standard called CSS for styles.*

To add custom styles for a pre-existing component, you will need to create a file `<component-name.scss>` in the `styles/components` directory of `ember-lookit-frameplayer`. Then add a line to the top of `styles/app.scss`, telling it to use that style. For example,

`@import "components/exp-video-physics";`

Remember that anything in ember-lookit-frameplayer is shared code. Below are a few good tips to help your new frame stay isolated and distinct, so that it does not affect other projects.

- To protect other frames from being affected by your new styles, add a class of the same name as your frame (e.g., `exp-myframe`) to the div enclosing your component. Then prefix *every* rule in your .scss file with `.exp-myframe` to ensure that only your own frame is affected. Until we have a better solution, this practice will be enforced if you submit a pull request to add your frames to the common Lookit ember-lookit-frameplayer repo.

- To help protect your *own* frame's styling from possible future style changes (improperly) added by other people, you can give new classes and IDs in your component a unique prefix, so that they don't inadvertently overlap with styles for other things. For example, instead of `video-widget` and `should-be-centered`, use names like `exp-myframe-video-widget` and `exp-myframe-should-be-centered`.

Researchers using your frame can force it to be shown fullscreen (even if that is not the typical intended use) by passing the parameter `displayFullscreenOverride`. If you have not also set the `displayFullscreen` property of your frame to `true`, then the `#experiment-player` element will have class `player-fullscreen-override` but not `player-fullscreen`, to allow display to more closely mimic what it would be in non-fullscreen mode for things like forms and text pages.

If you create an (intentionally) fullscreen frame, then the element you make fullscreen will have class `player-fullscreen` while it is fullscreen, which you can use for styling.

* You may notice that style files have a special extension `.scss`. That is because styles in experimenter are actually written in SASS. You can still write normal CSS just fine, but SASS provides additional syntax on top of that and can be helpful for power users who want complex things (like variables).

Using mixins

Sometimes, you will wish to add a preset bundle of functionality to any arbitrary experiment frame. The Experimenter platform provides support for this via *mixins*.

To use a mixin for video recording, fullscreen, etc., simply have your frame "extend" the mixin. For instance, to use the VideoRecord mixin, your component.js file would define:

```
import ExpFrameBaseComponent from '../exp-frame-base/component';
import layout from './template';
import VideoRecord from '../../mixins/video-record';

export default ExpFrameBaseComponent.extend(VideoRecord, {
    type: 'exp-consent-form',
    layout: layout,
    meta: {
        ...
    }
});
```

Your frame can extend any number of mixins. For now, be careful to check, when you use a mixin, that your frame does not define any properties or functions that will conflict with the mixin's properties or functions. If the mixin has a function `doFoo`, you can use that from your frame simply by calling `this.doFoo()`.

Below is a brief introduction to each of the common mixins, which are also each documented in the frameplayer docs.

FullScreen

This mixin is helpful when you want to show something (like a video) in fullscreen mode without distractions. You will need to specify the part of the page that will become full screen. By design, most browsers require that you interact with the page to trigger fullscreen mode.

MediaReload

This attempts to work around a quirk of how ember renders the page; see stackoverflow post for more information. We recommend implementing new frames to work with Ember's intended patterns instead.

VideoRecord

Functionality related to video capture, in conjunction with the Pipe system, for which MIT has a license.

Documenting your frame

We use Sphinx to generate documentation of ember-lookit-frameplayer frames from documentation files directly in the repository. You can see the hosted documentation here.

To include documentation for your new frame, add a doc.rst file in its directory along with component.js and template.hbs. You can pattern this after existing frames' documentation. It should include:

- A general description of your frame
- A screenshot of the frame, or a diagram outlining various phases/options
- An example of using it (the relevant JSON for a study)
- All parameters and their types
- All data saved
- Any events recorded

To check how your documentation will appear, run `make html` from the `ember-lookit-frameplayer` directory.

Ember debugging

Values of variables used in your frame are tricky to access directly from the Javascript console in your browser during testing.

There's an Ember Inspector browser plugin you can use to help debug the Lookit components. Once you've installed it, you'll find it along with other developer tools.

Here's how to find relevant data for a particular frame. Screenshots below are for Google Chrome.

This lets you right away change any of the data you sent to the frame in the JSON document. E.g., on the consent page, try changing the "prompt" to something else. If something is going wrong, hopefully this information will be helpful.

You can send the entire component (or anything else) to the console using the little >$E button:

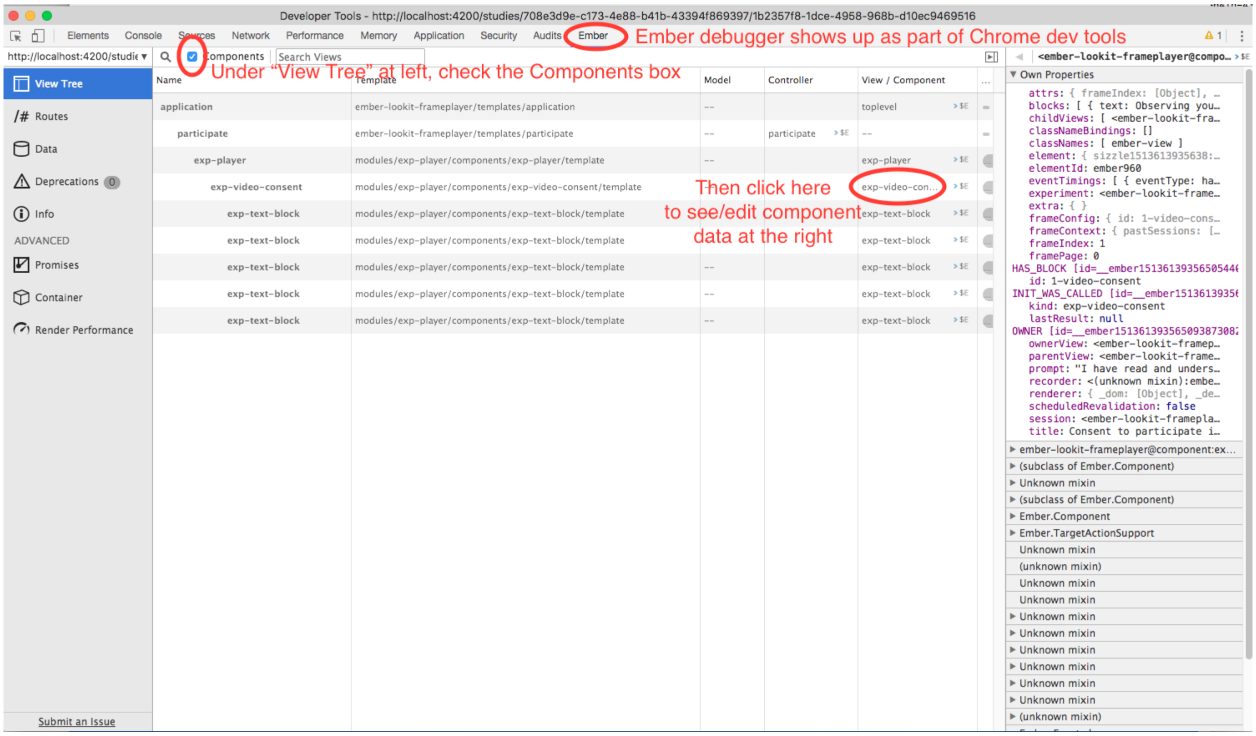
And then to keep using it, save it as a variable:

Then you can do things like try out actions, e.g. `this.send`.

When should I use actions vs functions?

Actions should be used when you need to trigger a specific piece of functionality via user interaction: eg click a button to make something happen.

Functions (or helper methods on a component/frame) should be used when the logic is shared, or not intended to be accessed directly via user interaction. It is usually most convenient for these methods to be defined as a part of the component, so that they can access data or properties of the component. Since functions can return a value, they are particularly helpful for things like sending data to a server, where you need to act on success or failure in order to display information to the user. (using promises, etc)



Ember debugger shows up as part of Chrome dev tools

Under "View Tree" at left, check the Components box

Then click here to see/edit component data at the right

Name	Template	Model	Controller	View / Component
application	ember-lookit-frameplayer/templates/application	--		toplevel
participate	ember-lookit-frameplayer/templates/participate	--	participate	--
exp-player	modules/exp-player/components/exp-player/template	--		exp-player
exp-video-consent	modules/exp-player/components/exp-video-consent/template			exp-video-con...
exp-text-block	modules/exp-player/components/exp-text-block/template			exp-text-block
exp-text-block	modules/exp-player/components/exp-text-block/template			exp-text-block
exp-text-block	modules/exp-player/components/exp-text-block/template	--		exp-text-block
exp-text-block	modules/exp-player/components/exp-text-block/template	--		exp-text-block
exp-text-block	modules/exp-player/components/exp-text-block/template	--		exp-text-block

```
<ember-lookit-frameplayer@compo…
▼ Own Properties
   attrs: { frameIndex: [Object], …
   blocks: [ { text: Observing you…
   childViews: [ <ember-lookit-fra…
   classNameBindings: []
   classNames: [ ember-view ]
   element: { sizzle1513613935638:…
   elementId: ember960
   eventTimings: [ { eventType: ha…
   experiment: <ember-lookit-frame…
   extra: { }
   frameConfig: { id: 1-video-cons…
   frameContext: { pastSessions: […
   frameIndex: 1
   framePage: 0
HAS_BLOCK [id=__ember15136139356505446
   id: 1-video-consent
INIT_WAS_CALLED [id=__ember15136139356
   kind: exp-video-consent
   lastResult: null
OWNER [id=__ember15136139356509387308
   ownerView: <ember-lookit-framep…
   parentView: <ember-lookit-frame…
   prompt: "I have read and unders…
   recorder: <(unknown mixin):embe…
   renderer: { _dom: [Object], _de…
   scheduledRevalidation: false
   session: <ember-lookit-framepla…
   title: Consent to participate i…
▶ ember-lookit-frameplayer@component:ex…
▶ (subclass of Ember.Component)
▶ Unknown mixin
▶ (subclass of Ember.Component)
▶ Ember.Component
▶ Ember.TargetActionSupport
  Unknown mixin
  (unknown mixin)
  Unknown mixin
  Unknown mixin
▶ Unknown mixin
▶ Unknown mixin
▶ Unknown mixin
▶ Unknown mixin
▶ Unknown mixin
▶ Unknown mixin
▶ (unknown mixin)
```

Fig. 1: Ember debugger tree view

Usually, you should use actions only for things that the user directly triggers. Actions and functions are not mutually exclusive! For example, an action called `save` might call an internal method called `this._save` to handle the behavior and message display consistently.

If you find yourself using the same logic over and over, and it does not depend on properties of a particular component, consider making it a util!

If you are building extremely complex nested components, you may also benefit from reading about closure actions. They can provide a way to act on success or failure of something, and are useful for : - Ember closure actions have return values - Ember.js Closure Actions Improve the Former Action Infrastructure

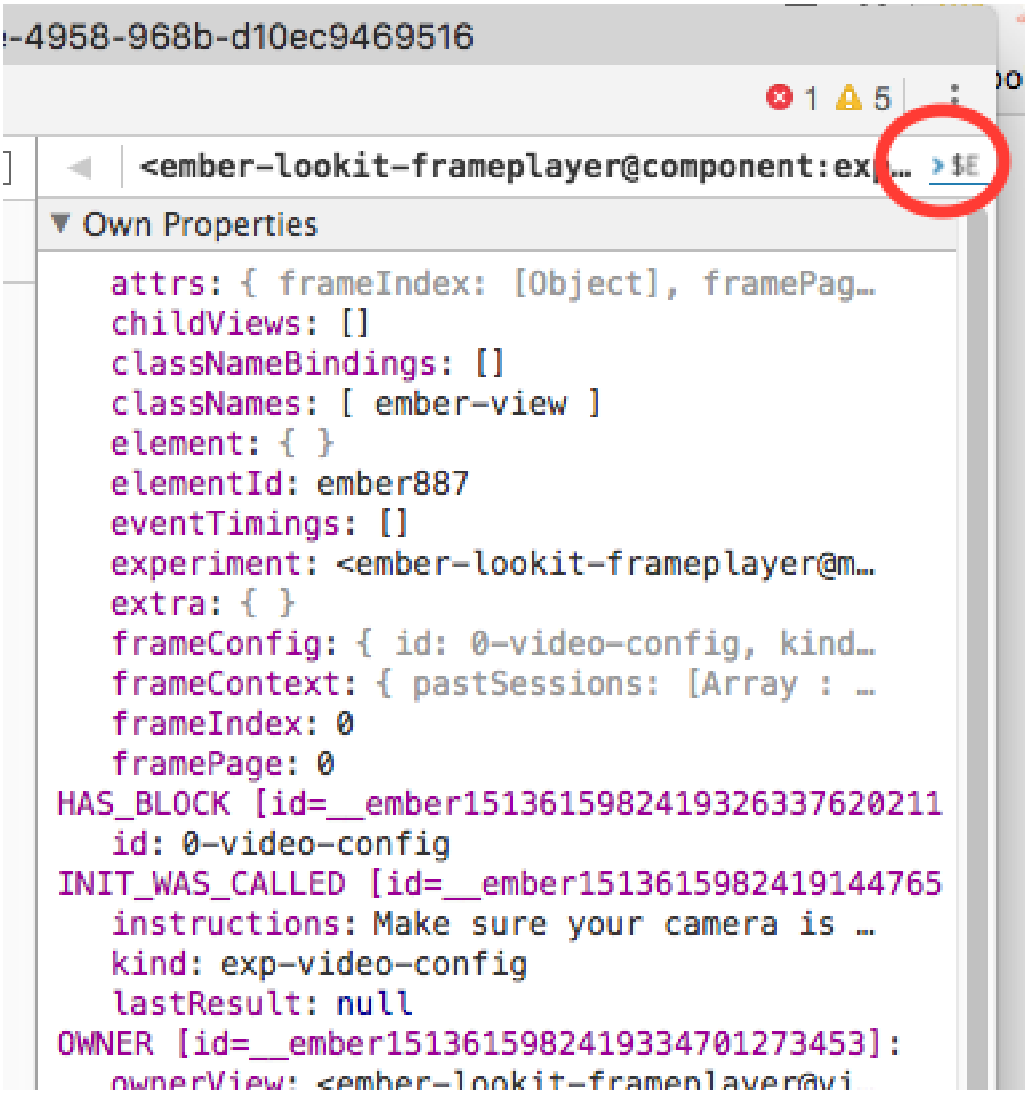
1.38.3 How to capture video in your frame

Webcam video recording during Lookit frames is currently accomplished using WebRTC as the interface to the webcam and Pipe for video streaming and processing.

Lookit frames that collect video data make use of an Ember mixin `VideoRecord` included in ember-lookit-frameplayer, which makes a `VideoRecorderObject` available for use in the code for that frame. This object includes methods for showing/hiding the webcam view, starting/pausing/resuming/stopping video recording, installing/destroying the recorder, and checking the current video timestamp.

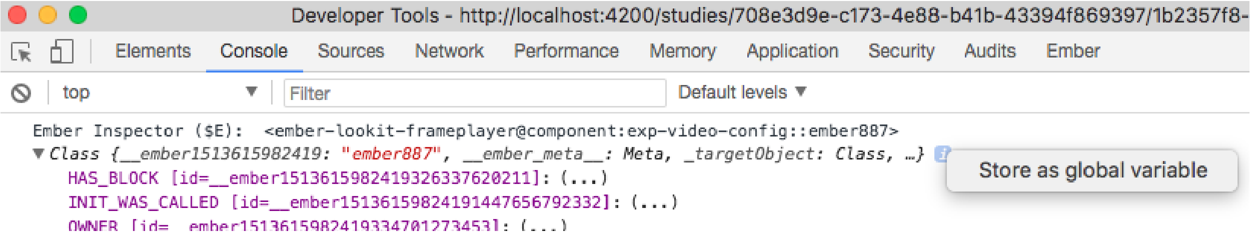
The programmer designing a new frame can therefore flexibly indicate when recording should begin and end, as well as recording video timestamps for any events recorded during this frame (e.g., so that during later data analysis, researchers know the exact time in the video where a new stimulus was presented). The name(s) of any videos collected during a particular frame as included in the session data recorded, to facilitate matching sessions to videos; video filenames also include the study ID, session ID, frame ID, and a timestamp.

To begin, you will want to add the `VideoRecord` mixin to your experiment frame. This provides, but does not in itself activate, the capability for your frame to record videos.



```
    attrs: { frameIndex: [Object], framePag…
    childViews: []
    classNameBindings: []
    classNames: [ ember-view ]
    element: { }
    elementId: ember887
    eventTimings: []
    experiment: <ember-lookit-frameplayer@m…
    extra: { }
    frameConfig: { id: 0-video-config, kind…
    frameContext: { pastSessions: [Array : …
    frameIndex: 0
    framePage: 0
HAS_BLOCK [id=__ember1513615982419326337620211
    id: 0-video-config
INIT_WAS_CALLED [id=__ember1513615982419144765
    instructions: Make sure your camera is …
    kind: exp-video-config
    lastResult: null
OWNER [id=__ember1513615982419334701273453]:
    ownerView: <ember-lookit-frameplayer@vi…
```

Fig. 2: Ember debugger send to console



```
Ember Inspector ($E):  <ember-lookit-frameplayer@component:exp-video-config::ember887>
▼Class {__ember1513615982419: "ember887", __ember_meta__: Meta, _targetObject: Class, …}
   HAS_BLOCK [id=__ember1513615982419326337620211]: (...)
   INIT_WAS_CALLED [id=__ember15136159824191447656792332]: (...)
   OWNER [id=__ember1513615982419334701273453]: (...)
```

Fig. 3: Ember debugger save variable

```
import ExpFrameBaseComponent from '../../components/exp-frame-base/component';
import VideoRecord from '../../mixins/video-record';

export default ExpFrameBaseComponent.extend(VideoRecord, {
    // Your code here
});
```

Limitations

One technical challenge imposed by webcam video streaming is that a connection to the server must be established before webcam recording can be quickly turned on, and this process may take up to a few seconds. If you are setting up your frame to create a separate video clip, you will need to design your frame to wait for recording to begin before proceeding to a portion of the trial where video data is required. This fits well with typical study designs using looking time or preferential looking, where the child's attention is returned to the center of the screen between trials; the first few seconds of the child watching the "attention grabber" are not critical and we can simply ensure that the webcam connection is established before proceeding to the actual experimental trial. When collecting verbal responses, the study frame can simply pause until the connection is established or, similarly, proceed with an initial portion of the trial where video data is not required.

You can also plan for users of your frame to turn on continuous recording using multi-frame or 'session' recordings, either using the exp-lookit-start-recording and exp-lookit-stop-recording frames or by directly setting the `startSessionRecording` and `endSessionRecording` parameters.

How it works

The VideoRecord mixin is how a new frame makes use of video recording functionality. In turn, this mixin uses the video-recorder service, which relies on Pipe. To set everything up from scratch, e.g. if you're creating Mookit, an online experimental platform for cows, you'll need to do the following:

- Make a Pipe account, and get the account hash and environment ID where you want to send videos.
- Create an Amazon S3 bucket (where video will be sent by Pipe, then renamed). Set up Pipe to send your videos to this bucket; you'll need to create an access key that just allows putting videos in this bucket. Go to IAM credentials, and make a group with the following policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "s3:ListAllMyBuckets",
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "s3:GetBucketLocation",
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Sid": "VisualEditor2",
            "Effect": "Allow",
            "Action": "s3:PutObject",
```

(continues on next page)

(continued from previous page)

```
                "Resource": [
                    "arn:aws:s3:::MYBUCKET/*",
                    "arn:aws:s3:::MYBUCKET"
                ]
            }
        ]
}
```

Then make a user, and add it to your new group. Use the keys for this user in Pipe.

- Create a webhook key in Pipe, and store it in as `PIPE_WEBHOOK_KEY` in the lookit-api Django app .env file. This will let Lookit rename the video files to something sensible upon being uploaded to S3.
- Create a webhook in Pipe for the event video_copied_s3, and send it to `https://YOURAPP/exp/renamevideo/`
- Store the `PIPE_ACCOUNT_HASH` and `PIPE_ENVIRONMENT` in the ember-lookit-frameplayer .env file. This is what lets Lookit video go to the right Pipe account.

1.38.4 Custom randomizer frames

Experimenter supports a special kind of frame called ‘choice’ that defers determining what sequence of frames a participant will see until the page loads. This allows for dynamic ordering of frame sequence in particular to support randomization of experimental conditions or counterbalancing. The goal of this page is to walk through an example of implementing a custom ‘randomizer’.

Overview of ‘choice’ structure

Generally the structure for a ‘choice’ type frame takes the form:

```
{
    "kind": "choice",
    "sampler": "random",
    "options": [
        "video1",
        "video2"
    ]
}
```

Where: - **sampler** indicates which ‘randomizer’ to use. This must correspond with the values defined in `lib/exp-player/addon/randomizers/index.js` - **options**: an array of options to sample from. These should correspond with values from the `frames` object defined in the experiment structure (for more on this, see the experiments docs)

Making your own

There is some template code included to help you get started. From within the `ember-lookit-frameplayer/lib/exp-player` directory, run:

```
ember generate randomizer <name>
```

which will create a new file: `addon/randomizers/<name>.js`. Let’s walk through an example called ‘next. The ’next’ randomizer simply picks the next frame in a series. (based on previous times that someone participated in an experiment)

```
$ ember generate randomizer next
...
installing randomizer
  create addon/randomizers/next.js
```

Which looks like:

```
/*
 NOTE: you will need to manually add an entry for this file in addon/randomizers/
↪index.js, e.g.:
import
import Next from './next';
...
{
    ...
    next: Next
}
 */
var randomizer = function(/*frame, pastSessions, resolveFrame*/) {
    // return [resolvedFrames, conditions]
};
export default randomizer;
```

The most important thing to note is that this module exports a single function. This function takes three arguments: - `frame`: the JSON entry for the 'choice' frame in context - `pastSessions`: an array of this participants past sessions of taking this experiment. See the experiments docs for more explanation of this data structure - `resolveFrame`: a copy of the ExperimentParser's _resolveFrame method with the `this` context of the related ExperimentParser bound into the function.

Additionally, this function should return a two-item array containing: - a list of resolved frames - the conditions used to determine that resolved list

Let's walk through the implementation:

```
var randomizer = function(frame, pastSessions, resolveFrame) {
    pastSessions = pastSessions.filter(function(session) {
        return session.get('conditions');
    });
    pastSessions.sort(function(a, b) {
        return a.get('createdOn') > b.get('createdOn') ? -1: 1;
    });
    // ...etc
};
```

First we make sure to filter the `pastSessions` to only the one with reported conditions, and make sure the sessions are sorted from most recent to least recent.

```
...
var option = null;
if(pastSessions.length) {
    var lastChoice = (pastSessions[0].get(`conditions.${frame.id}`) || frame.
↪options[0]);
    var offset = frame.options.indexOf(lastChoice) + 1;
    option = frame.options.concat(frame.options).slice(offset)[0];
}
else {
    option = frame.options[0];
}
```

Next we look at the conditions for this frame from the last session (`pastSessions[0].get(`conditions.${frame.id}`))`. If that value is unspecified, we fall back to the first option in `frame.options`. We calculate the index of that item in the available `frame.options`, and increment that index by one.

This example allows the conditions to "wrap around", such that the "next" option after the last one in the series circles back to the first. To handle this we append the `options` array to itself, and slice into the resulting array to grab the "next" item.

If there are not past sessions, then we just grab the first item from `options`.

```
    var [frames,] = resolveFrame(option);
    return [frames, option];
};

export default randomizer;
```

Finally, we need to resolved the selected sequence using the `resolveFrame` argument. This function always returns a two-item array containing: - an array of resolved frames - the conditions used to generate that array

In this case we can ignore the second part of the return value, and only care about the returned `frames` array.

The `export default randomizer` tells the module importer that this file exports a single item (`export default`), which in this case is the randomizer function (**note**: the name of this function is not important).

Finally, lets make sure to add an entry to the index.js file in the same directory:

```
import next from './next';

export default {
    ...,
    next: next
};
```

This allows consuming code to easily import all of the randomizers at once and to index into the `randomizers` object dynamically, e.g. (from the `ExperimentParser`):

```
import randomizers from 'exp-player/randomizers/index';
// ...
return randomizers[randomizer](
    frame,
    this.pastSessions,
    this._resolveFrame.bind(this)
);
```

1.39 Contributor Guidelines

Interested in helping write the code behind the Lookit platform? Thanks for supporting open source science! This page describes the process any would-be contributor should plan to use. We have included some beginner-friendly details in case you are new to open source projects.

The content of this page applies to all three Lookit repos: `lookit-api` (Lookit site), `ember-lookit-frameplayer` (system for displaying experiments & components to use), and `lookit-docs` (the documentation you're reading now).

Where's the code I need?

If you only want to change something about the Lookit site, without touching experiment functionality (for instance, to add a question to the demographic survey or change how studies are sorted), you will only need to run *lookit-api* and can follow the Django project installation steps. If you want to develop experiment frames or change how the experiment player works, you will need to follow the steps for local frame development, installing *both lookit-api* and *ember-lookit-frameplayer* and telling them how to talk to each other. Your changes, however, will likely be limited to *ember-lookit-frameplayer*.

1.39.1 Prerequisites

To contribute to the *lookit-api* codebase, it will be very helpful to have a (a) a strong grasp of Python and (b) some familiarity with the Django framework. Learning Python is outside the scope of these docs, but if you want someplace to start, we highly recommend Think Python. If you're already familiar with Python but haven't used the web framework Django, we highly recommend taking the time to complete the official Django tutorial.

To contribute to the *ember-lookit-frameplayer* codebase - e.g., when creating your own experiment frames - it will be helpful to have (a) a strong grasp of Javascript and (b) some familiarity with Ember.js. However, we're really not using that much of the functionality of Ember, and if you're just making some new frames, we would recommend getting started by trying out modifications of an existing frame to get your feet wet, rather than trying to learn Ember from scratch.

1.39.2 Getting started

At a high level, we are roughly following a Forking Workflow version of Gitflow as described here.

You should plan to make feature-specific branches off of the `develop` branch (for lookit-api, lookit-docs) or `master` branch (for ember-lookit-frameplayer) of a local copy of the code running on your own machine. This will keep the codebase as clean as possible.

First create your own fork of lookit-api, ember-lookit-frameplayer, and/or lookit-docs. Follow the directions for installation of lookit-api or ember-lookit-frameplayer if needed.

1.39.3 Ignoring some files

You may want to configure a global .gitignore on your machine and include your virtualenv(s) along with any files specific to your system. A sample global .gitignore is available here – you can tell git to globally ignore files specified in a .gitignore file via:

```
git config --global core.excludesfile ~/path/to/your/.gitignore_global
```

1.39.4 Add your own feature and submit a Pull Request

Keep your commit history clean and merge process simple by following these steps before starting on any new feature.

One time only, add the original repo as a remote to your fork, e.g., if you are contributing to *lookit-api* you would run a command like this:

SSH:

```
git remote add upstream git@github.com:lookit/lookit-api.git
```

HTTPS:

```
git remote add upstream https://github.com/lookit/lookit-api.git
```

Anytime a PR is merged or changes are pushed (or you're starting this process for the first time), you should run:

```
git checkout develop
git pull upstream develop
```

in order to make sure you are working with an up-to-date copy of the *develop* branch.

Once you have the most recent *develop* code, pick an issue (or create a new one) which your new feature will address and create a new branch off of *develop*. Note: our project convention is to prepend *feature/* or *hotfix/* to the feature or issue name for a richer annotation of the commit history.

If you want to create a new validation feature, for example, you might name it like this:

```
git checkout -b feature/my-validation-feature
```

Now you can run *git branch* and should see an output like this:

```
$ git branch
  develop
  master
* feature/my-validation-feature
```

Proceed with writing code. Commit frequently! Focus on writing very clear, concise commit statements and plentiful comments. If you have poor comments or zero tests, your PR will not be merged.

If you are aware of changes in the branch you forked from, rebase your branch from that changing branch (in our case that is *develop*) by running:

```
git rebase develop
```

and then resolving all merge conflicts.

On `lookit-api`, you should then update dependencies like this:

```
pip install -r requirements/defaults.txt
python manage.py migrate
python manage.py test
```

On `ember-lookit-frameplayer`, you should update dependencies using the package manager yarn.

Next, push all your local changes to your own fork. You should push your code (making sure to replace `feature/my-validation-feature` with whatever your branch is actually called):

```
git push --set-upstream origin feature/my-validation-feature
```

Prior to finalizing your commit, make sure to clean up your code to comply with PEP8. Since both black and isort are included in our development dependencies, you should just be able to run `isort -rc . --skip venv` to fix your imports, and similarly `black . --exclude=venv` to "blacken" your changes. With both commands, replace `venv` with the actual name of your virtual env directory so that you don't blacken/isort your dependencies.

When your branch is ready (you've tested your changes out, and your code has comments and tests), submit a Pull Request! To do this, go to GitHub, navigate to your fork (in this case the github extension should be /your-username/lookit-api), then click `new pull request`. Change the base to `develop` and the compare to `feature/my-validation-feature`. Finally, click *Create pull request* and describe the changes you have made. Your pull request will be reviewed by Lookit staff; changes may be requested before changes are merged into the develop branch. To allow Lookit staff to add changes directly to your feature branch, follow the directions here.

IMPORTANT: WHEN YOUR PR IS ACCEPTED, stop using your branch right away (or delete it altogether). New features (or enhanced versions of your existing feature) should be created on brand new branches (after pulling in all the fresh changes from `develop`).

1.39.5 Writing your tests

In `lookit-api`, you should generally add to or edit the `tests.py` file in the appropriate app (e.g., `exp/tests.py`). You can run tests like this:

```
python manage.py test
```

For more information see https://docs.djangoproject.com/en/2.1/topics/testing/.

In `ember-lookit-frameplayer` you should generally edit the tests under `tests/`, but as you will see there is currently very little coverage. Just try to leave it better than you found it.

In `ember-lookit-frameplayer`, you should generally add a test file under `tests/unit/components/` if you have created a new frame. As you can see, we do not have a strong convention for this yet except for randomizer frames.

To learn more about how testing is supposed to work for `ember-lookit-frameplayer`, see https://guides.emberjs.com/v2.11.0/testing/.

1.39.6 Creating a release (ember-lookit-frameplayer)

The ember-lookit-frameplayer repo is semantically versioned.

The release process is relatively manual for now because the expected workflow isn't finalized (it's currently almost entirely a one-person project).

Work should be completed and tested on a feature branch, then merged into develop.

To create a new major or minor release:

1. When a set of features is ready to release, create a release branch off of develop named `release/vX.Y.Z`
2. Change version number in package.json in the release branch.
3. Turn on readthedocs builds for the release branch.
4. Make PRs from the release branch to master and develop, and merge commit.
5. Create a new release on GitHub, exactly matching the version name used above. Include release notes explaining what has been added/changed. For major versions (backwards-incompatible changes), include step-by-step instructions for updating study protocols (e.g., "1. If your study contains a frame with `kind: "exp-lookit-oldsurvey"`, replace "exp-lookit-oldsurvey" with "exp-lookit-survey". It will work the same way, the name has just changed.")

To create a new bugfix release for the latest version:

Follow the steps above except don't turn on readthedocs builds.

To create a new bugfix release for an older version:

1. Create a new release branch off of the target release branch (e.g., `release/v3.1.5`). Increment the version in the new branch name (e.g., `release/v3.1.6`).
2. Apply appropriate patch and increment version on new bugfix branch.
3. Create a new release on GitHub, exactly matching new version name. Include release notes explaining what has been fixed.

1.40 Installation: lookit-api (Django project)

`lookit-api` is the codebase for Experimenter and Lookit, excluding the actual studies themselves. Any functionality you see as a researcher or a participant (e.g., signing up, adding a child, editing or deploying a study, downloading data) is part of the `lookit-api` repo. This project is built using Django and PostgreSQL. (The studies themselves use Ember.js; see Ember portion of codebase, ember-lookit-frameplayer.), It was initially developed by the Center for Open Science.

If you install only the `lookit-api` project locally, you will be able to edit any functionality that does not require actual study participation. For instance, you could contribute an improvement to how studies are displayed to participants or create a new CSV format for downloading data as a researcher.

Note: These instructions are for Mac OS. Installing on another OS? Please consider documenting the exact steps you take and submitting a PR to the lookit-api repo to update the documentation! For notes on Linux installation, there may be helpful information in a previous version of invoke tasks.py.

1.40.1 Requirements

This is the software you will need to have installed to run lookit-api locally.

1. If you haven't already, install Brew and install Graphviz:

```
brew install graphviz
```

2. Install and start rabbitmq via brew:

```
brew install rabbitmq && brew services start rabbitmq
```

3. You will need to have Docker installed and running.
4. Create a Postgres database using the following command:

```
docker run --name lookit-postgres -d -e POSTGRES_HOST_AUTH_METHOD="trust" -e␣
↪POSTGRES_DB="lookit" -p 5432:5432 postgres:9.6
```

5. To help with installing a specific version of python, we'll need to install asdf.
6. Add Python plugin using the following command. Here's documentation to help with errors:

```
asdf plugin-add python
```

7. Clone the lookit-api repo:

```
git clone https://github.com/lookit/lookit-api.git
```

8. At the root of the project, install python. The asdf Python plugin docs can help install older versions of python:

   ```
   asdf install
   ```

9. Install Poetry.
10. Install Python libraries:

    ```
    poetry install
    ```

11. Use invoke to run setup:

    ```
    poetry run invoke setup
    ```

    This will create a local .env file with environment variables for local development, run the Django application's database migrations ("catching up" on changes to the database structure), set up rabbitmq with queues for various task types, and create local SSL certificates. If you're curious about what exactly is happening during this step, or run into any problems, you can reference the file tasks.py.

12. Create a superuser by running:

    ```
    poetry run ./manage.py createsuperuser
    ```

Now you should be ready for anything. Going forward, you can run the server using the directions below.

1.40.2 Running the server

To run the Lookit server locally, run:

```
poetry run invoke server
```

Now you can go to http://localhost:8000 to see your local Lookit server! You should be able to log in using the superuser credentials you created during setup.

To view the HTTPS version of the local development add the `https` argument to the above command:

```
poetry run invoke server --https
```

If you are not working extensively with lookit-api - i.e., if you just want to make some new frames - you do not need to run celery, rabbitmq, or docker. For more information about these services and how they interact, please see the Contributing guidelines.

1.40.3 Running Celery

You should already have a rabbitmq server installed and running. You can check this by:

```
brew services list
```

If rabbitmq is not running, you can start it using:

```
brew services start rabbitmq
```

Then use the invoke command to start the celery worker:

```
poetry run invoke celery-service
```

1.40.4 Authentication

You can create participant and researcher accounts through the regular signup flow on your local instance. To access Experimenter you will need to add two-factor authentication to your account following the prompts. In order to access the admin interface (https://localhost:8000/__CTRL__), which provides a convenient way to access and edit records, you will need to log in using the superuser you created earlier using manage.py.

1.40.5 Handling video

This project includes an incoming webhook handler for an event generated by the Pipe video recording service used by ember-lookit-frameplayer when video is transferred to our S3 storage. This requires a webhook key for authentication. It can be generated via our Pipe account and, for local testing, stored in .env under `PIPE_WEBHOOK_KEY`.

Pipe needs to be told where to send the webhook. First, you need to expose your local /exp/renamevideo hook. You can use Ngrok to generate a public URL for your local instance during testing:

```
ngrok http https://localhost:8000
```

Then, based on the the assigned URL, you will need to manually edit the webhook on the dev environment of Pipe to send the `video_copied_s3` event to (for example) `https://8b48ad70.ngrok.io/exp/renamevideo/`.

1.40.6 Common Issues

During installation, you may see the following:

```
psql: FATAL:  role "postgres" does not exist
```

To fix, run something like the following from your home directory:

```
$../../../usr/local/Cellar/postgresql/9.6.3/bin/createuser -s postgres
```

If your version of postgres is different than 9.6.3, replace with the correct version above. Running this command should be a one-time thing.

You might also have issues with the installation of `pygraphviz`, with errors like

```
running install
Trying pkg-config
Package libcgraph was not found in the pkg-config search path.
Perhaps you should add the directory containing `libcgraph.pc'
to the PKG_CONFIG_PATH environment variable
No package 'libcgraph' found
```

or

```
pygraphviz/graphviz_wrap.c:2954:10: fatal error: 'graphviz/cgraph.h' file not found
#include "graphviz/cgraph.h"
       ^
1 error generated.
error: command 'clang' failed with exit status 1
```

To fix, try running something like:

```
$ brew install graphviz
$ pip install --install-option="--include-path=/usr/local/include" --install-option="-
↪-library-path=/usr/local/lib" pygraphviz
```

Then re-run setup.

1.41 Installation: ember-lookit-frameplayer (Ember app)

`ember-lookit-frameplayer` is a small Ember application that allows both researchers to preview an experiment and users to participate in an experiment. This is meant to be used in conjunction with the Lookit API Django project, which contains the Experimenter and Lookit applications. The Django application will proxy to these Ember routes for previewing/participating in an experiment.

In order to run the frame player as it works on Lookit, you will need to additionally install the Django app `lookit-api` and then follow the local frame development instructions to make sure it communicates with the Ember app. This way, for instance, an experiment frame will be able to look up previous sessions a user has completed and use those for longitudinal designs.

> Note: These instructions are for Mac OS. Installing on another OS? Please consider documenting the exact steps you take and submitting a PR to the lookit-api repo to update the documentation!

1.41.1 Prerequisites

You will need the following tools properly installed on your computer.

- Git
- Node.js (with NPM)
- Bower

1.41.2 Installation

Before beginning, you will need to install Yarn, a package manager (like npm).

```
git clone https://github.com/lookit/ember-lookit-frameplayer.git
cd ember-lookit-frameplayer
yarn install --pure-lockfile
bower install
```

Create or open a file named '.env' in the root of the ember-lookit-frameplayer directory, and add the following entries to use the Pipe WebRTC-based recorder: `PIPE_ACCOUNT_HASH` (reference to account to send video to) and `PIPE_ENVIRONMENT` (which environment, e.g. development, staging, or production). These are available upon request if you need to use the actual Lookit environments. (If you are doing a very large amount of local testing, we may ask that you set up your own Pipe account.) Your .env file should look like this:

```
PIPE_ACCOUNT_HASH='<account hash here>'
PIPE_ENVIRONMENT=<environment here>
```

1.41.3 Running / Development

- `ember serve`

- Visit your app at http://localhost:4200.

If you change any dependencies, make sure to update and commit the yarn.lock file in addition to package.json.

Code Generators

Make use of the many generators for code, try `ember help generate` for more details

Running Tests

- `ember test`
- `ember test --server`

Building

- `ember build` (development)
- `ember build --environment production` (production)

Writing documentation of frames

Documentation of individual exp-player components is automatically generated using YUIDoc:

- yarn run docs

At the moment, this is a manual process: whatever files are in the top level /docs/ folder of the master branch will be served via GitHub pages.

1.42 Django app implementation notes

1.42.1 Workflow: managing study states

Why Transitions

Transitions is an object-oriented state machine implemented in Python.

It's both very powerful and very simple. It's definition is a python dictionary so it can be easily serialized into JSON and stored in a database or configured via YAML. It has callback functionality for state transitions. It can create diagrams of the workflow using pygraphiz. It also ties into django model classes very easily.

How

The workflow is defined in `studies/workflow.py` in a dictionary called `transitions`. Here is a gist that explains how the pieces fit together.

Make a diagram

To make a workflow diagram in png format start a shell plus instance with `python manage.py shell_plus` and execute the following:

```
# get a study you'd like to diagram
s = Study.objects.first()
# draw the whole graph ... in which case the study you choose doesn't matter
s.machine.get_graph().draw('fancy_workflow_diagram.png', prog='dot')
# ... or just the region of interest (contextual to the study you chose)
# (previous state, active state and all reachable states)
s.machine.get_graph(show_roi=True).draw('roi_diagram.png', prog='dot')
```

Logging

There is a `_finalize_state_change` method on the `Study` model. It fires after every workflow transition. It saves the model with its updated `state` field and also creates a `StudyLog` instance making record of the transition. This callback would be the optimal place to add functionality that needs to happen after every workflow transition.

1.42.2 Permissions

Generic best practices

- Groups are an important abstraction between users and permissions.
  - If you assign permissions directly to a user it will be difficult to find out who has the permissions and difficult to remove them.
- Creating a Group just to wrap an individual permission is fine.
- Include the model name when defining model specific permissions. Permissions are referenced with app_name and permission codename.
- Always check for individual permissions. **NEVER CHECK IF SOMEONE BELONGS TO A GROUP or ‘‘is_superuser‘‘**
- `is_superuser` implicitly grants all permissions to a user. Any permissions check will return `True` if a user `is_superuser`.

Guardian, how does it work?

Django provides model-level permissions. That means that you can allow users in the Scientist group the ability to read the Report model or users in the Admin group the ability to create, read, update, and delete the Report model.

Guardian provides object-level permissions. That means that you can allow users in the Southern Scientists group the ability to read the a specific Report instance about Alabama.

Guardian does this by leveraging Django’s generic foreign key field. This means that Guardian can have a severe performance impact on queries where you check object-level permissions. It will cause a double join through Django’s ContentType table. If this becomes non-performant you can switch to using direct foreign keys.

1.42.3 Celery tasks

build_experiment task

The business requirements for this project included the ability for experiments to rely on versioned dependencies without causing conflicts between experiments.

The experiment application is dependent on the `ember-lookit-frameplayer` repo. Researchers have the ability to specify a custom github url for the `ember-lookit-frameplayer` repo. They can also specify a SHA for the commit that they would like to use. These fields are on the Build Study Page.

What happens

The build process uses celery, docker, ember-cli, yarn, and bower.

When a build or preview is requested a celery task is put into the build queue.

Inside the task, the study and requesting user are looked up in the database. If it's a preview task its current state is copied into a new variable to be saved for later, then the study is put into the state of `previewing` and saved. If it's a deployment the study is put into the state of `deployment` and saved. Since these states don't actually exist in the workflow definition this short circuits the workflow engine so that studies currently undergoing deployment or preview can neither move through the workflow or be previewed or deployed concurrently.

The SHAs are checked in the study model's metadata field, if they are empty or invalid the **HEAD** of the default branch is used. This requires HTTPS calls to github for the `ember-lookit-frameplayer` repository.

A zip file of each repo is downloaded to a temporary directory and extracted. The `lookit-frame-player` archive is extracted in the *checkout directory* (`ember_build/checkouts/{player_sha}`).

A docker image is built based on the node:8.2.1-slim image. It is rebuilt every time because it doesn't change very often and docker rebuilds of unchanged images are very fast.

The container is started passing several environment variables. It installs python3 and several other dependencies with `apt-get`. Then it installs yarn, bower, and ember-cli@2.8 globally with npm. Next it mounts the `ember-build/checkouts` directory to `/checkouts` inside the container and the `ember-build/deployments` directory to `/deployments` inside the container. It copies `ember-build/build.sh` and `ember-build/environment` into the root (`/`) of the container and executes `/bin/bash /build.sh`.

`build.sh` copies the contents of the *checkout directory* into the container (`/checkout-dir/` inside the container) for faster file access. A couple of `sed` replacements are done where there are experiment specific data that needs to be hardcoded prior to `ember-build` running. The `environment` files are copied into the correct places. Then `yarn install --pure-lockfile` and `bower install --allow-root` are run for `ember-lookit-frameplayer`. Once those have completed `ember-build -prod` is run to create a distributable copy of the app. The contents of the `dist` folder is then copied into the study output directory. The container is now destroyed.

Once the build process is finished the files in the `dist` folder are copied to a folder on Google Cloud Storage. If it's a preview they go into a `preview_experiments/{study.uuid}` folder in the bucket for the environment (staging or production). If it's a deployment they go into a `experiments/{study.uuid}` folder in the bucket for the environment (staging or production).

When the task is finished copying the files to Google Cloud Storage an email is sent to the study admins and Lookit admins.

If the task was a preview task the state of the study is set back to it's previous state. If it was a deployment the study is set to active. If the study is marked as discoverable, it will now be displayed on the lookit studies list page.

Finally, regardless of whether the task completed successfully a study log will be created. The extra field (a JSON field) will contain the logs of the image build process, the logs of the ember build process than ran inside the docker container, any raised exception, and the any logs generated by python during the entire task run. This is very helpful for debugging. Line endings are encoded for ease of storage so to read the results easily copy the contents of a study logs extra field from the admin into an editor and replace the overly escaped linebreaks `(\\\\n|\\n)` with actual line breaks `\n`. You can also use a JSON beautifier/formatter to aid readability.

build_zipfile_of_videos

This task downloads videos from MIT's Amazon S3 bucket, zips them up, uploads them to Google Cloud Storage, generates a signed url good for 30m, and emails the requesting user that URL.

- The zip filename is generated from the study uuid, a sha256 of the included filenames, and whether it's consent videos or all videos.
- If a zip file exists on Google Cloud Storage with the same name the file is not regenerated, an email with a link is immediately sent.
- After the task is completed all video files are immediately removed from the server. They still exist on s3 and Google Cloud Storage.

cleanup_builds

This finds build directories older than a day and deletes them. It's scheduled to run every morning at 2am.

cleanup_docker_images

This finds unused docker images from previous builds and deletes them. It's scheduled to run every morning at 3am.

cleanup_checkouts

This finds checkout (extracted archives of github repos) directories older than a day and deletes them. It's scheduled to run every morning at 4am.