Airbnb Design Challenge

Michelle Venetucci Harvey

michelle.ann.harvey@gmail.com

November 2017

Product Choice: GitHub

Improving Developer <> Designer Workflows

GitHub has the potential to improve cross-functional collaboration throughout the product development process.

Product Opportunity & Research

Conduct basic foundational user research and competitive analysis to provide context and opportunities.

Existing GitHub Product Audit

Audit the product to contextualize problems and identify areas of opportunity or improvement.

Business Opportunity

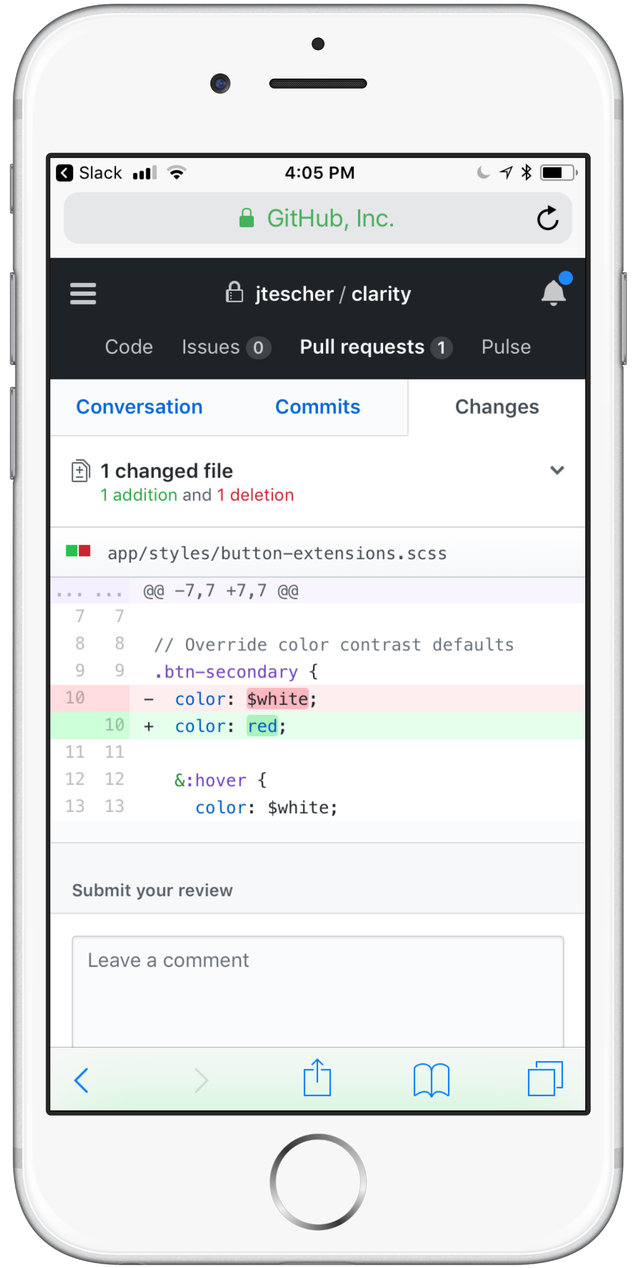
Lay out unique opportunities from a business perspective to better inform prioritization of work.

Brainstorming & Prototyping

Generate ideas and direction through rapid brainstorming activities, prototyping, and wireframing.

Final Design

Walk through the final solution from a UX and visual design perspective.



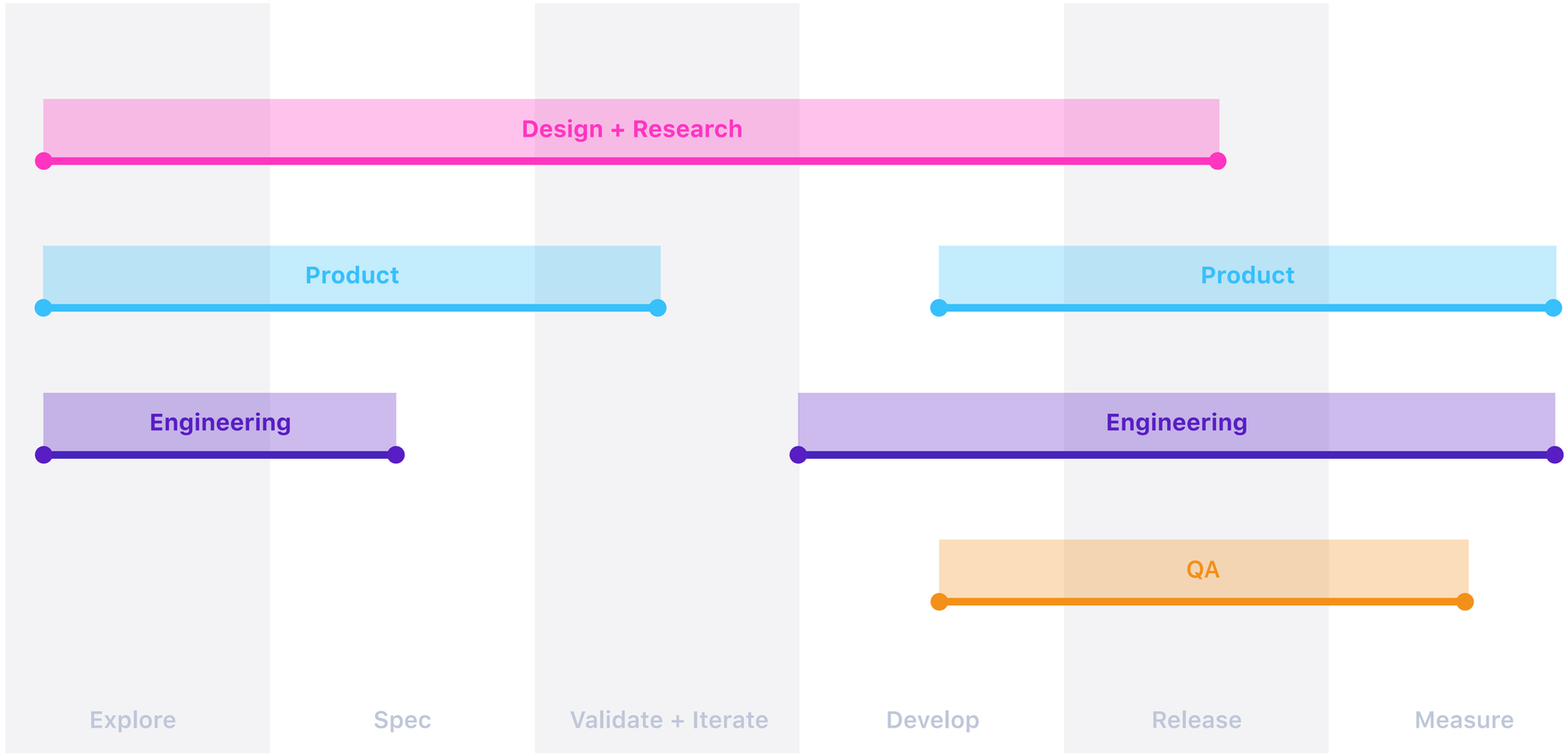
GitHub's existing mobile experience

Product Opportunity

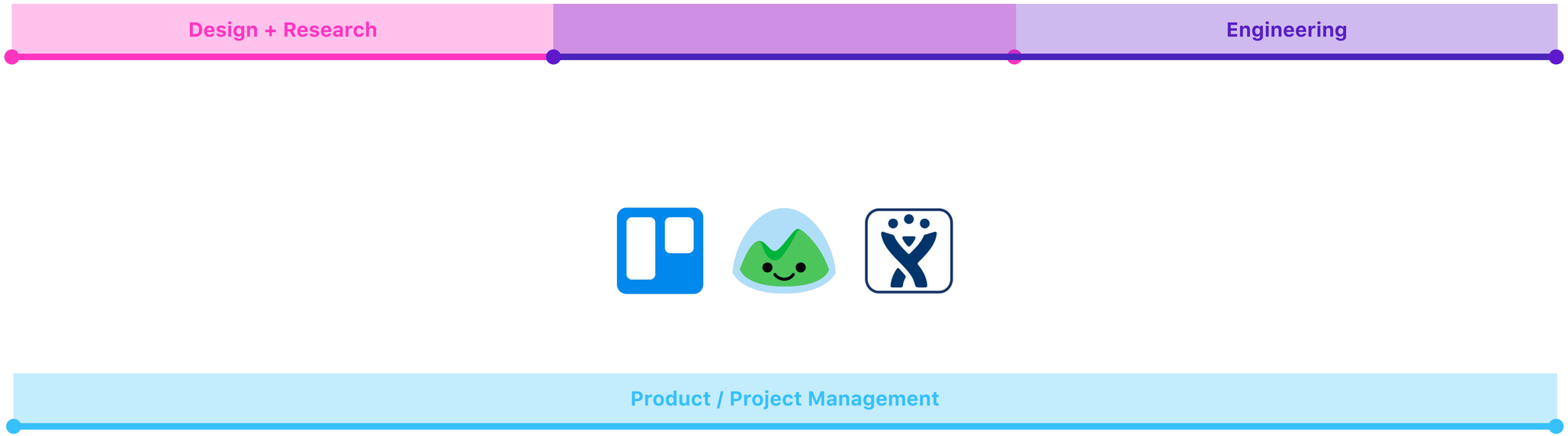
Research & Competitive Analysis

I conducted time-constrained foundational user research and looked over the competitive landscape in order to better understand the industry and start identifying the most compelling problem spaces.

There are many cross-functional touchpoints throughout a typical digital project lifecycle.



While there are currently many tools that project participants use, there aren’t many that are targeted specifically towards cross-functional collaboration and treat cross-functional partners equally.



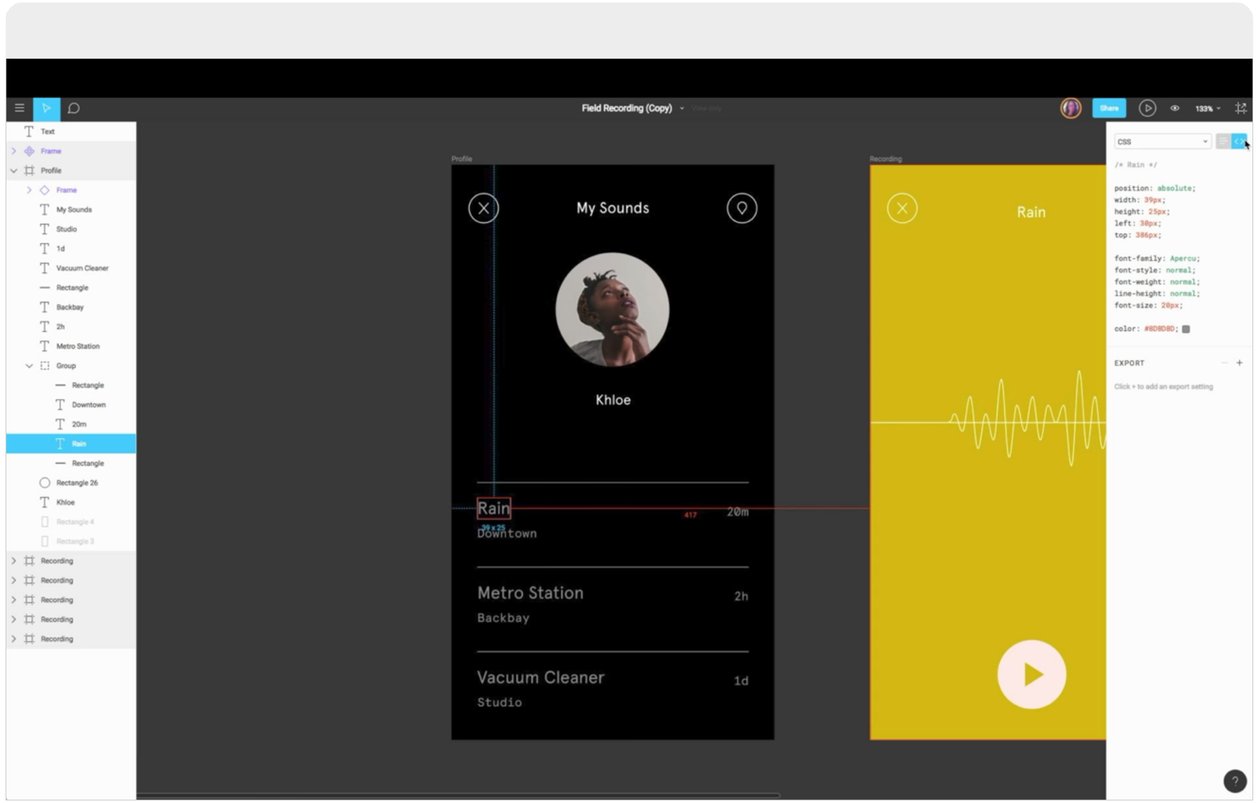
“

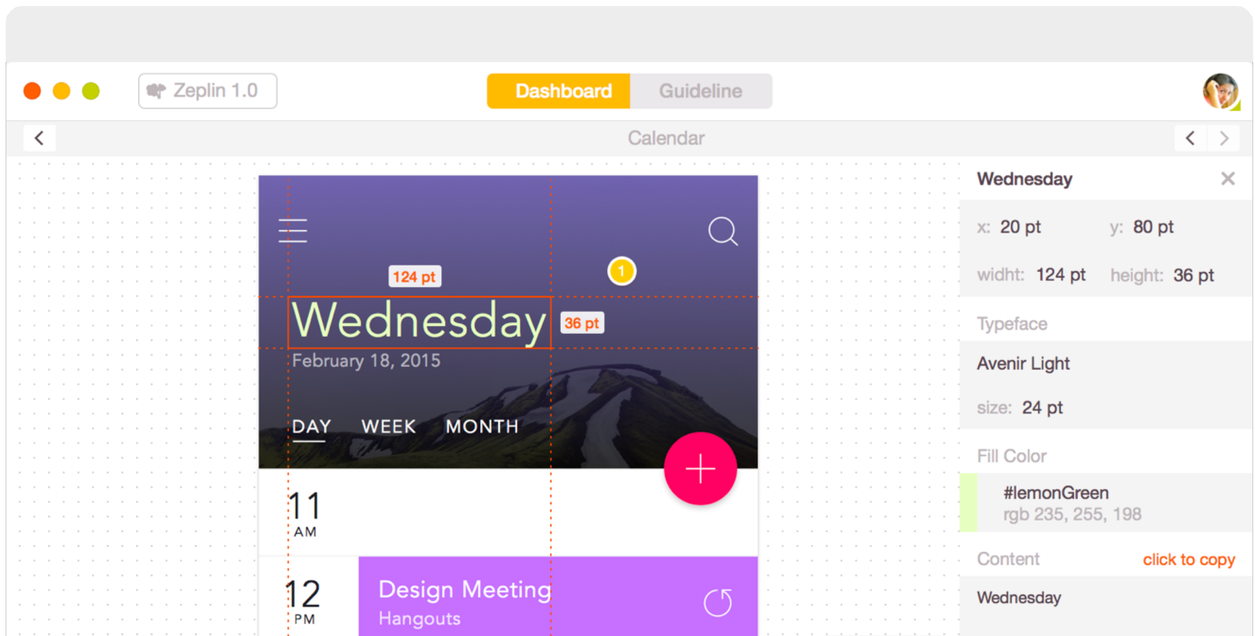
When handing off work to engineers, I always think the videos and mocks communicate the design perfectly. But what I get back often looks nothing like my intention, which ends up being an awful experience.

- Harrison, Product Designer

Many design tools have popped up recently with a main goal of improving communication between designers and developers.

These are generally seen as helpful, but remain somewhat separated from the engineering workflow, which can cause communication breakdowns.





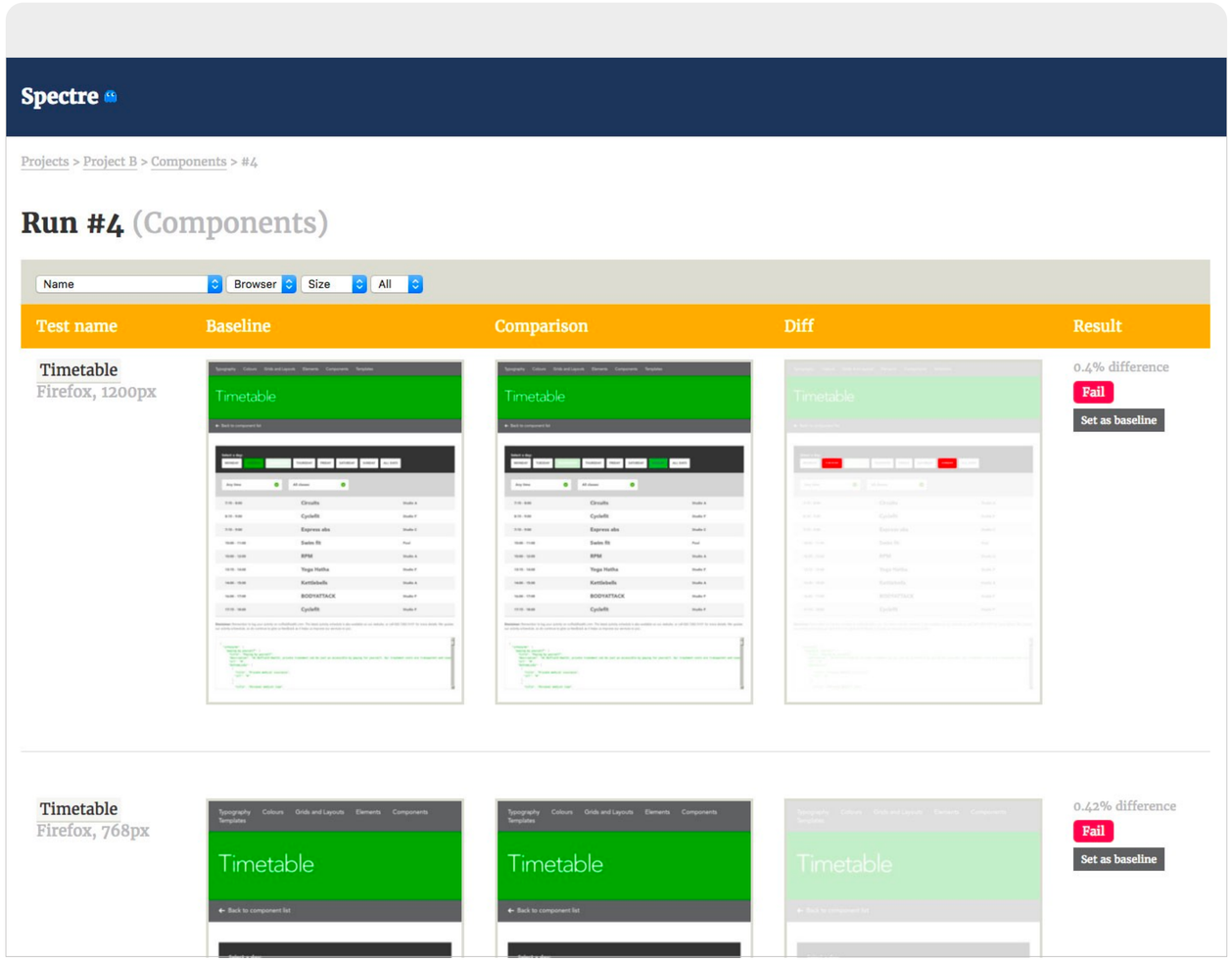
“

I Use Zeplin for redlines, but it’s not very efficient. I also use Dropbox, and it’s okay. It gets the job done for sharing files, but it’s hard to keep track of things. Google docs and Paper could be useful for comments but overall things are hard to track.

- Lin, Product Designer

There are a host of visual regression testing tools made for engineering workflows.

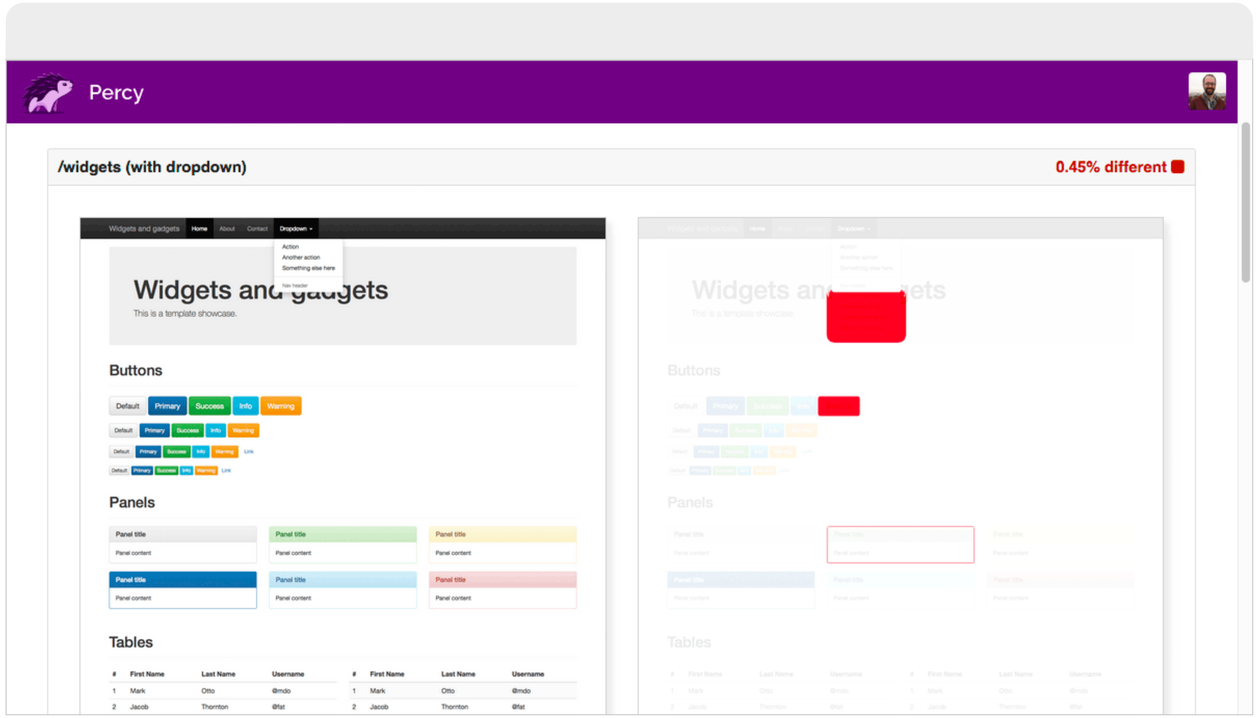
Most compare proposed CSS to previous CSS visually, and focus on engineers as the end user.

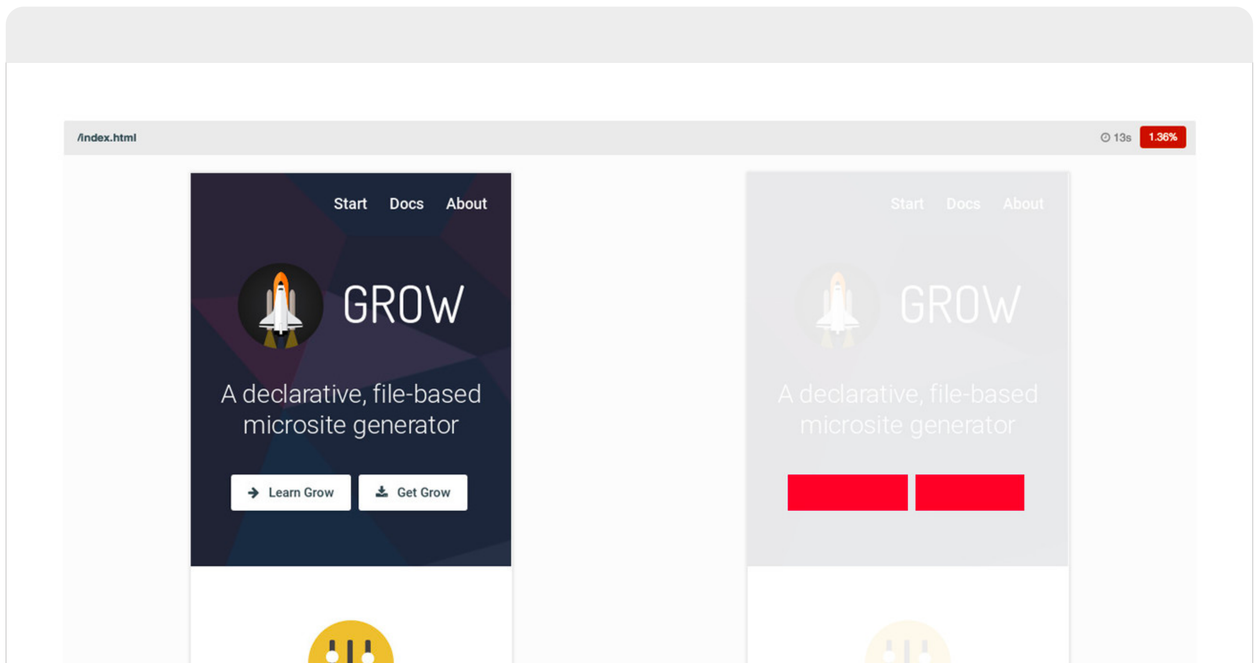


One example is percy.io, a continuous visual review tool for engineers building web apps.

> “Percy integrates directly into your development workflow to provide iterative and fast feedback about visual changes. This is what we mean by automated visual reviews.”

The fact that so many of these tools exist shows a growing need from users within this problem space, but fragmentation leads to low adoption.





“

There’s always a lot of miscommunication in the designer / developer handoff period. It’s really challenging to keep track across all the different tools we’re trying to piece together.

- Alex, Engineer

Keeping track of updates

Designers and engineers often struggle to keep track of cross-functional work.

Polish

Zeplin has improved polish for static mocks, but it’s still challenging to communicate animation values.

Communication is a barrier

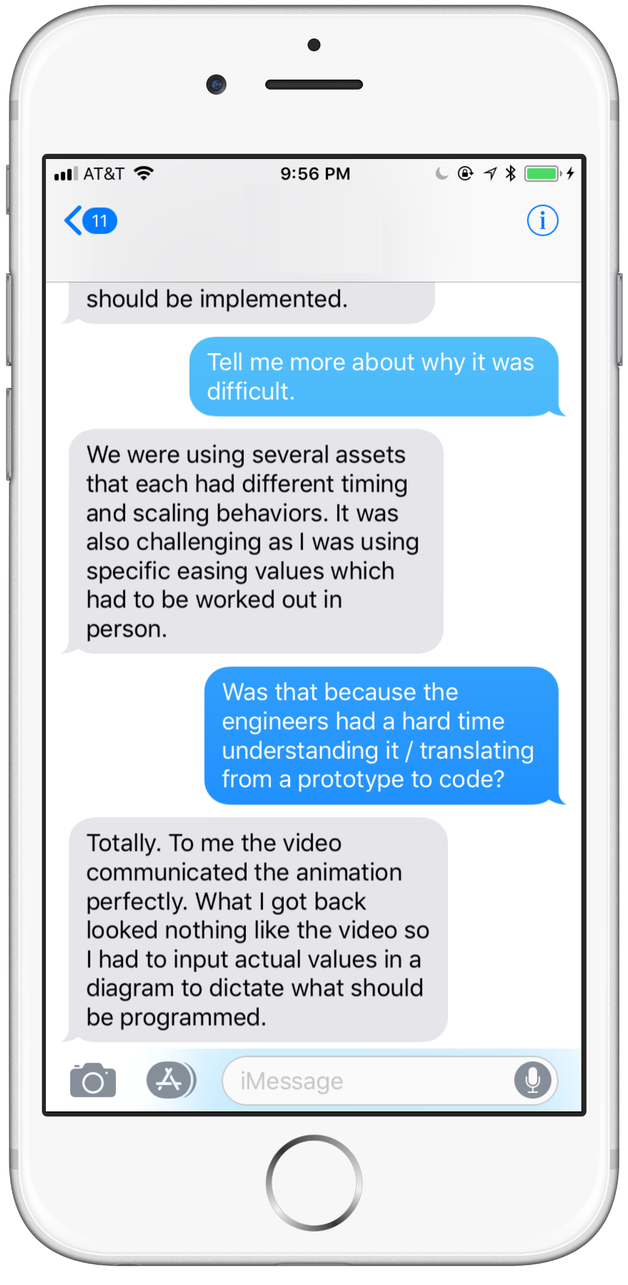
Designers and engineers feel like they communicate clearly to their cross-functional partners, but often the results don’t reflect the original intention.

Too many tools

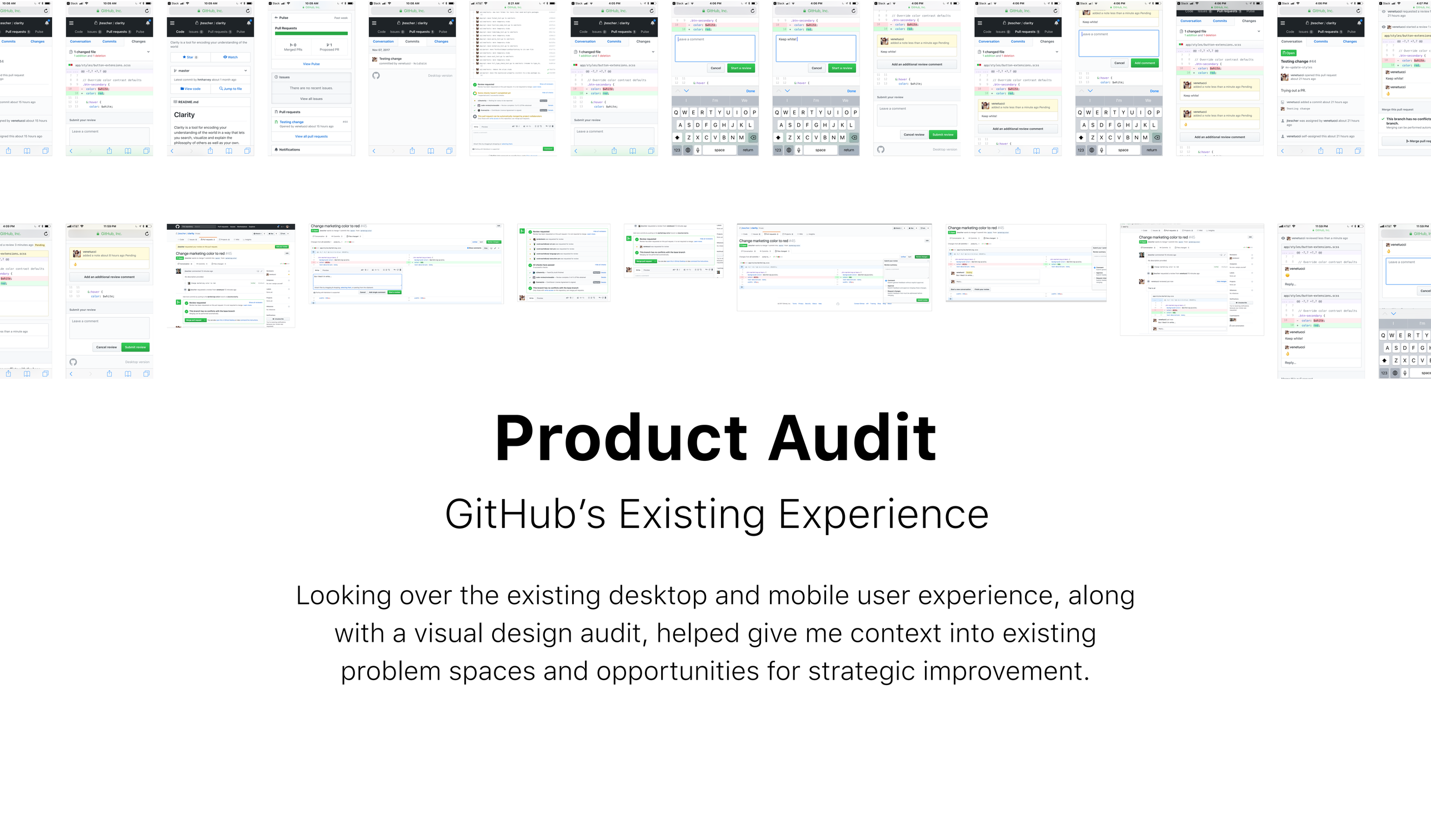
Designers and engineers are using too many tools to communicate with each other.

GitHub as a collaboration tool

GitHub feels very intimidating to everyone except engineers.



Screenshot of SMS-based user interview



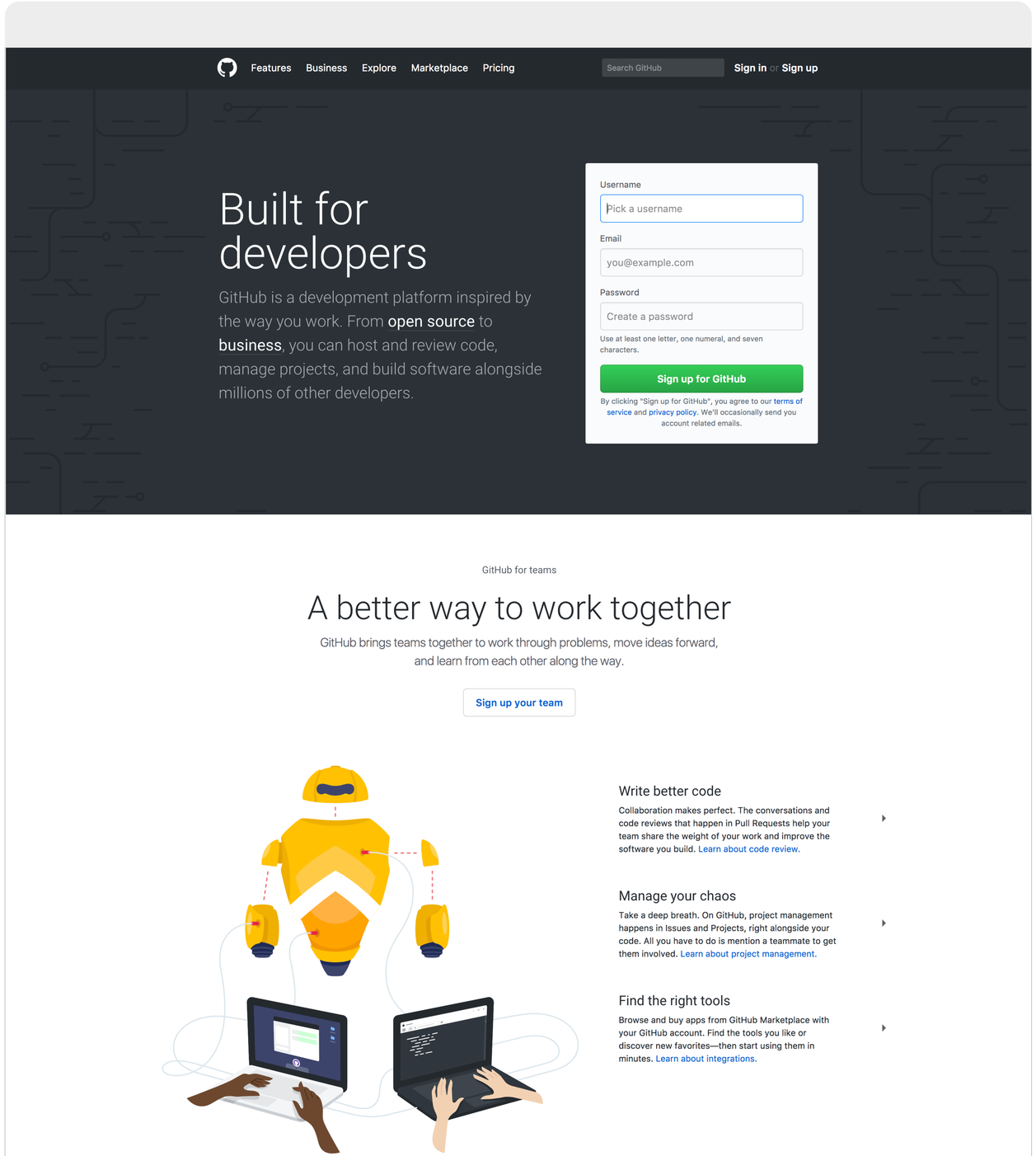
Product Audit

GitHub’s Existing Experience

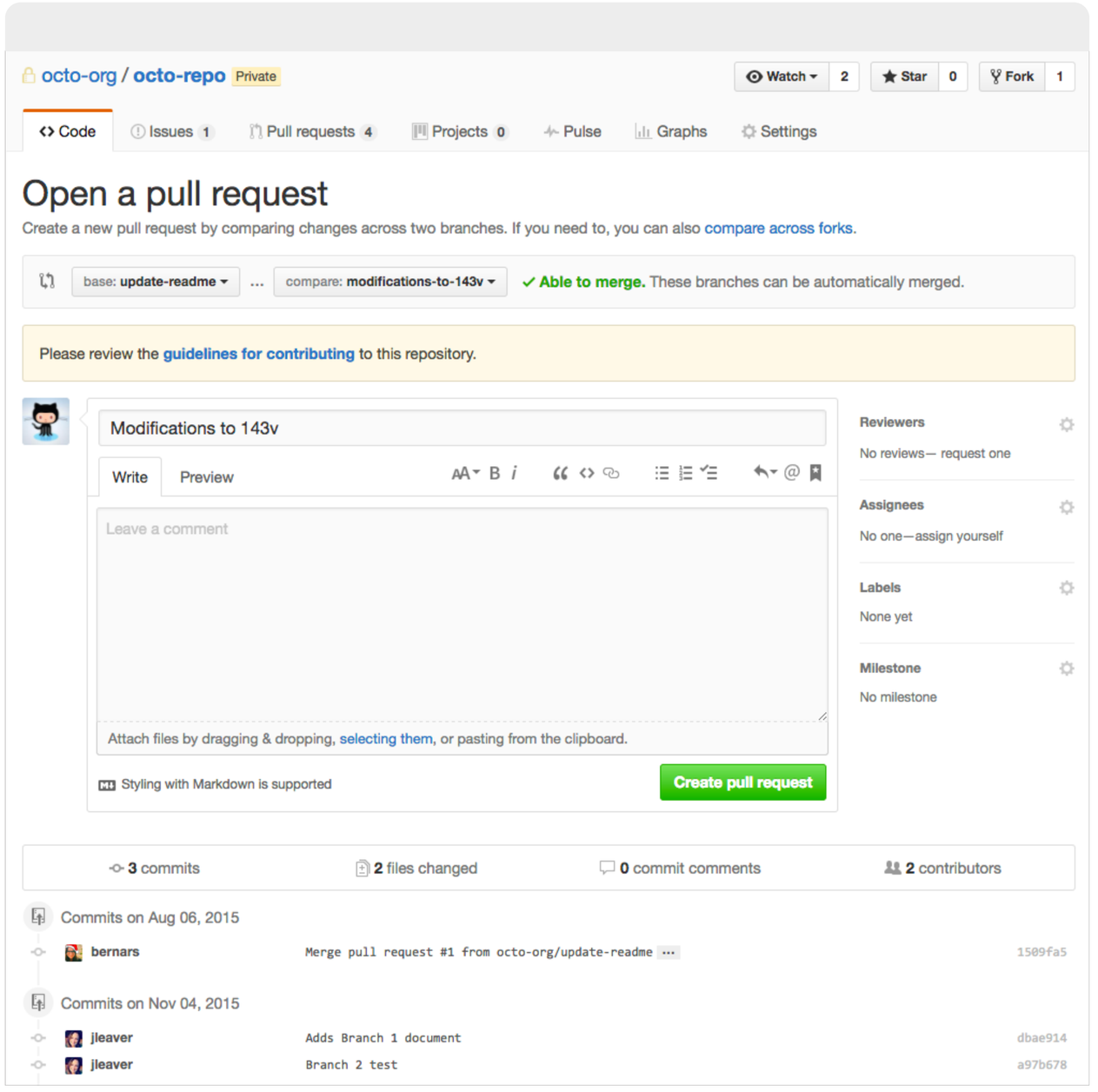
Looking over the existing desktop and mobile user experience, along with a visual design audit, helped give me context into existing problem spaces and opportunities for strategic improvement.

GitHub is a great tool for developer collaboration, but falls short in supporting cross-functional collaboration. Some examples:

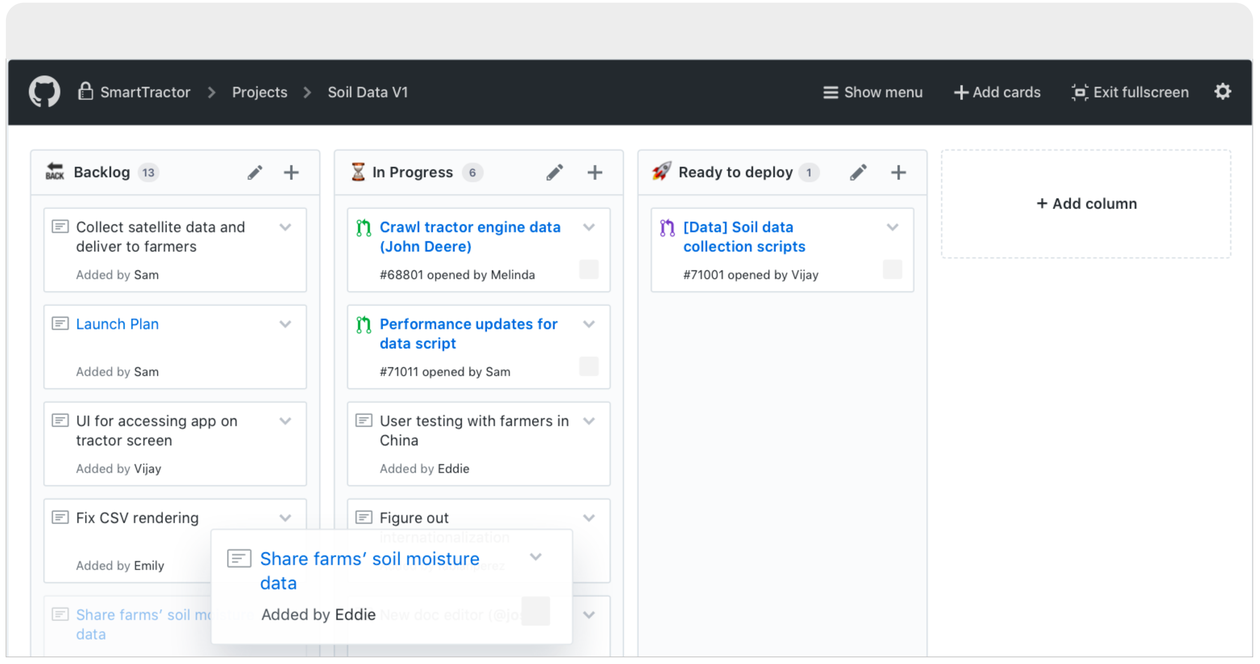
- Pull Requests (PRs) only include code reviews, even when there are visual updates that need reviewing.
- The UI is intimidating and there is no user experience geared towards designers, who may want to be involved in the review process.

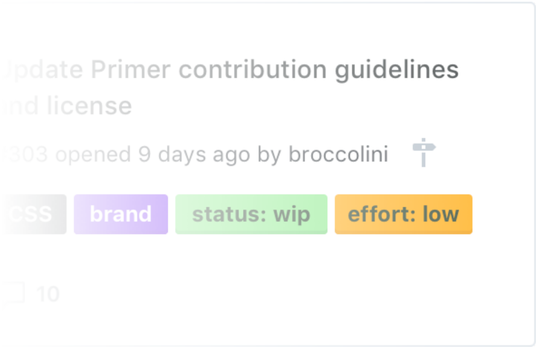


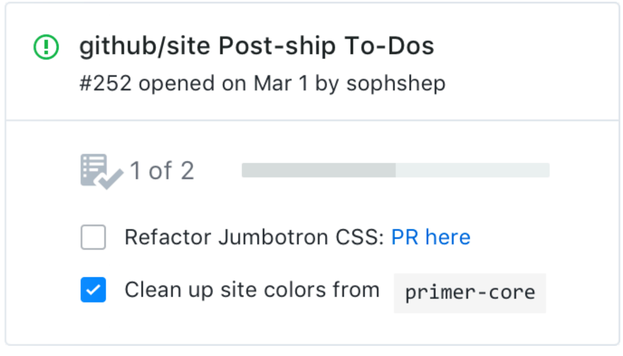
Pull Requests (PRs) are a great opportunity for design QA or as a tool for visual comparisons, but the current view is cluttered, intimidating, and doesn't include useful features for visual reviews.

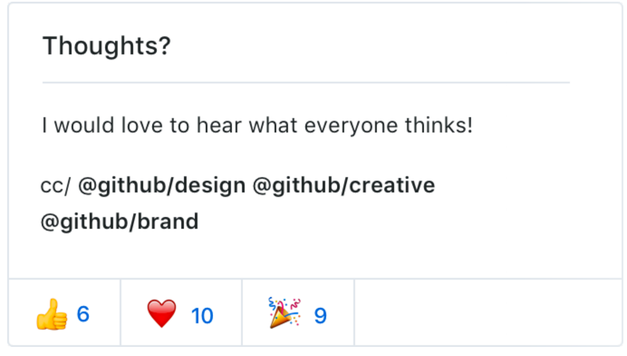


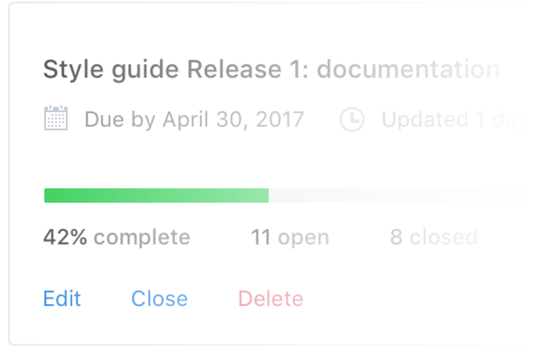
GitHub has started rolling out project management tools, but they are still very developer focused and don’t provide good collaboration features for designers.











GitHub already has features to show differences in images between different code commits, which many developers find helpful.

This creates the potential to apply similar concepts to full product screenshots, or to compare against mocks instead of previously committed code.

Read about image diffs on GitHub here

“

GitHub is scary. Yikes. I don’t see any value in it for me. I would love to get a GitHub link that I could open and actually gather some utility from, but I just don’t see a use right now.

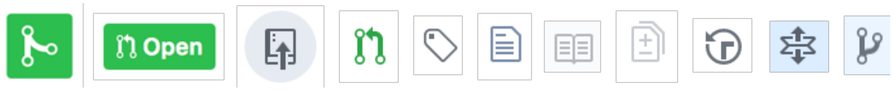
- Harrison, Product Designer

Product Audit / **GitHub Visual Audit**

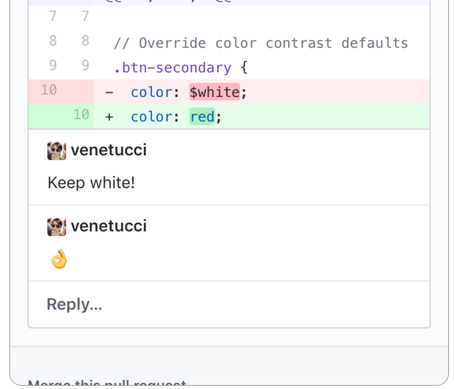
GitHub has a lot of potential areas of improvement in their visual design. GitHub also doesn't have a native experience, so mobile viewing often comes off as an afterthought.

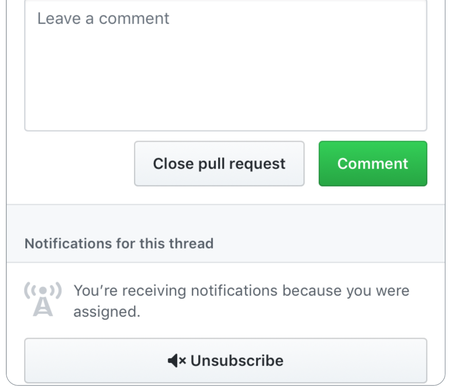
Some areas for potential improvement:

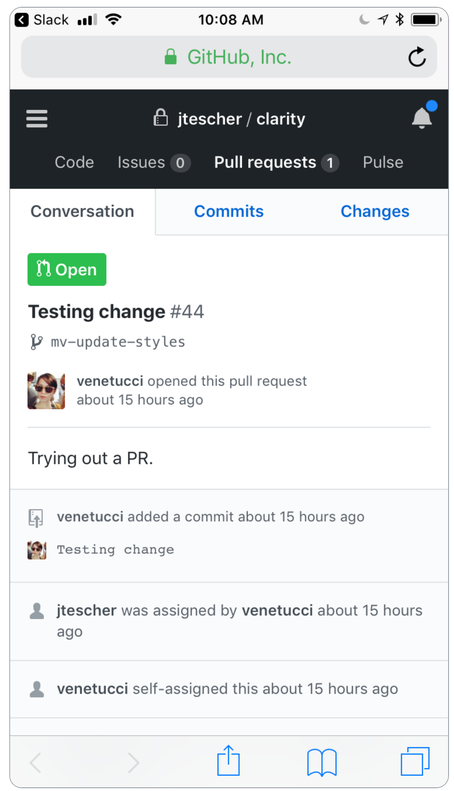
- Establish better information hierarchy (typography, spacing, etc).
- Update the color palette to be less "muddy" and more vibrant and welcoming.
- Remove ambiguous icons that increase visual clutter and don't provide helpful context.

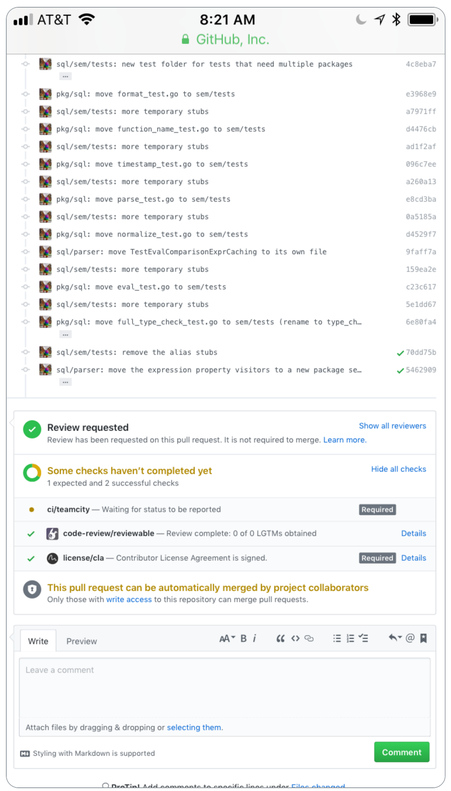


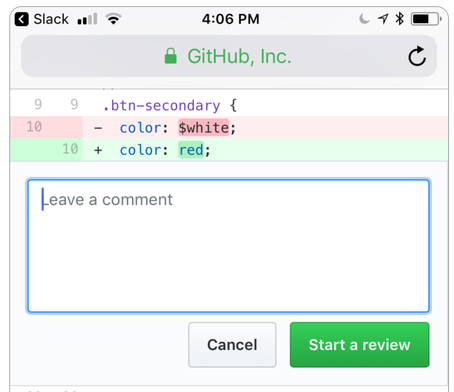
- Simplify the navigation experience and create clearer calls-to-action.

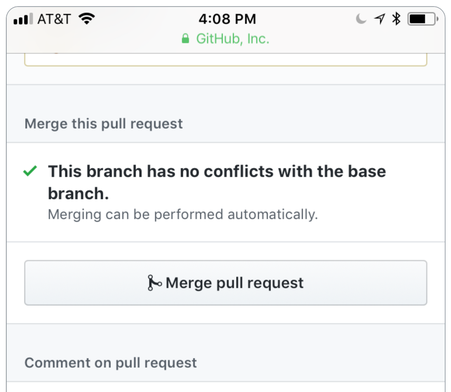


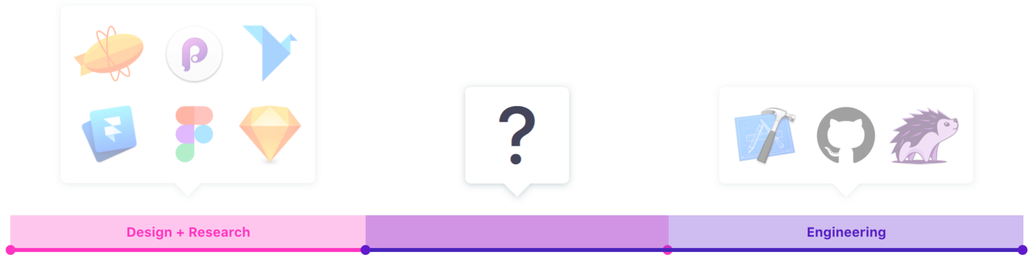












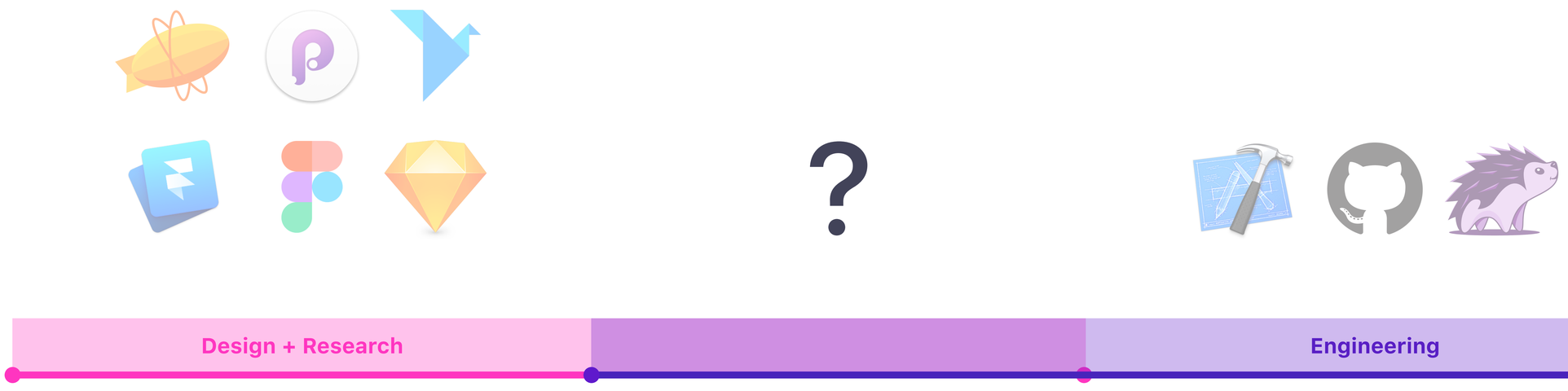
Business Opportunity

Long & Short Term Goals

Thinking through the business opportunity and positioning helped me prioritize the direction and scope of work.

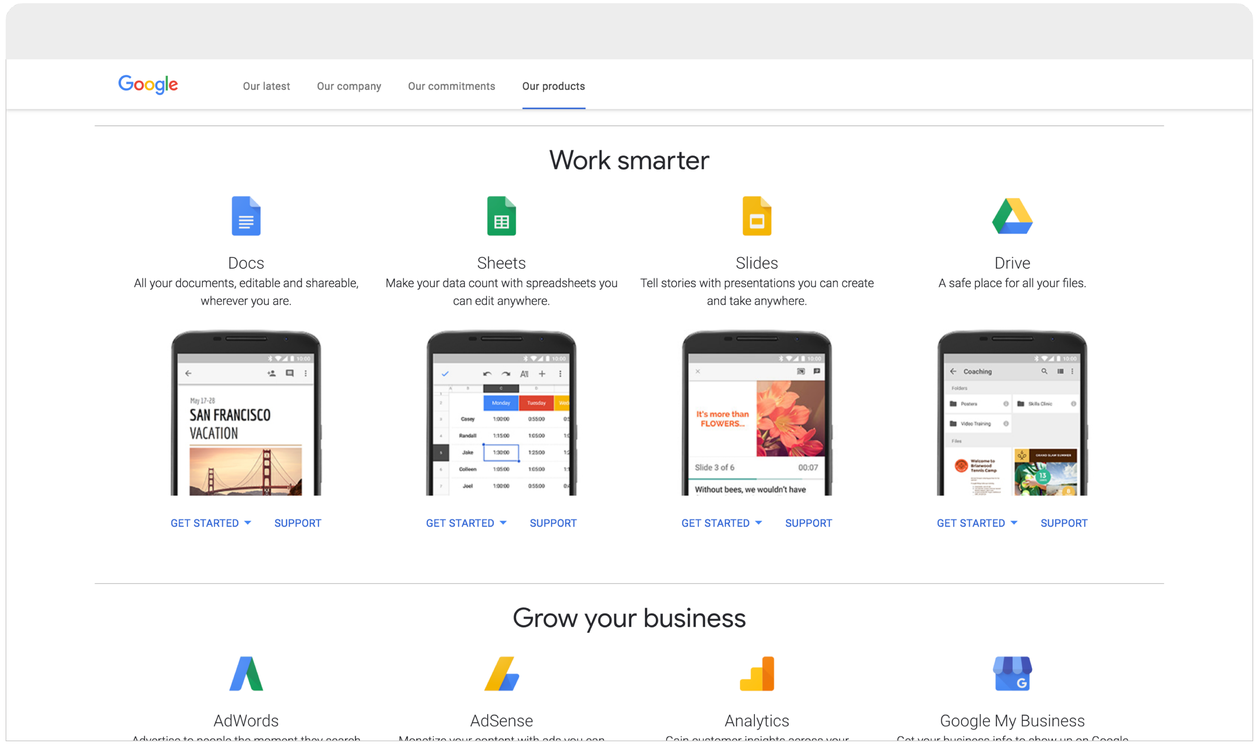
Building tools to improve designer and developer collaboration will not only increase project efficiency and quality, it also gives GitHub the opportunity to:

1. Replace a whole host of other professional subscription tools by providing companies with a holistic suite of tools.
2. Edge out current competitors by providing a more expansive suite of connected products.



GitHub as the go-to service for anyone building products, providing **a suite of end-to-end project collaboration tools** that increase cross-functional collaboration and the quality of final products. Success could look like:

- GitHub greatly expanding its user base, including designers, product managers, and researchers.
- Teams no longer subscribing to services like Trello or Zeplin, and instead using GitHub.
- Increased engagement through a more integrated holistic project experience and an expanded user base.
- GitHub edging out competitors by providing a whole suite of tools, taking a similar approach as Google Drive or Dropbox.



Google provides a suite of business tools

In the short term, a good step toward the long term vision is a feature that can immediately provide value. GitHub can build an initial version of an **integrated design process** that can immediately start **improving cross functional collaboration**.

Success can be measured through:

- Adoption rates
- Churn rates
- Increased efficiency and quality of work
- Qualitative research
- Daily Active Users (DAUs)
- Weekly Active Users (WAUs)
- Monthly Active Users (MAUs)

View full business opportunity thinking here →

Brainstorming

Strategy & Prototyping

For this project, I did a variety of brainstorming activities to make sure I explored divergent options, and then narrowed down on a solution using sketching and prototyping.

I mapped a critical path to start thinking about pain points throughout the user journey.

Design Ideation & Mock Creation

Designer: How should I communicate with developers?
Developer: Why won't the designer loop me in?

Hand Off

Designer: I'm really busy and am not sure how much to communicate.
Developer: I'm not totally sure what the designer wants.

Implementation

Designer: I feel like I have to micro-manage, or else I don't know what's going on.
Developer: I'm not sure if I'm catching all the details, and I'm not sure how to loop in design.

Review & QA

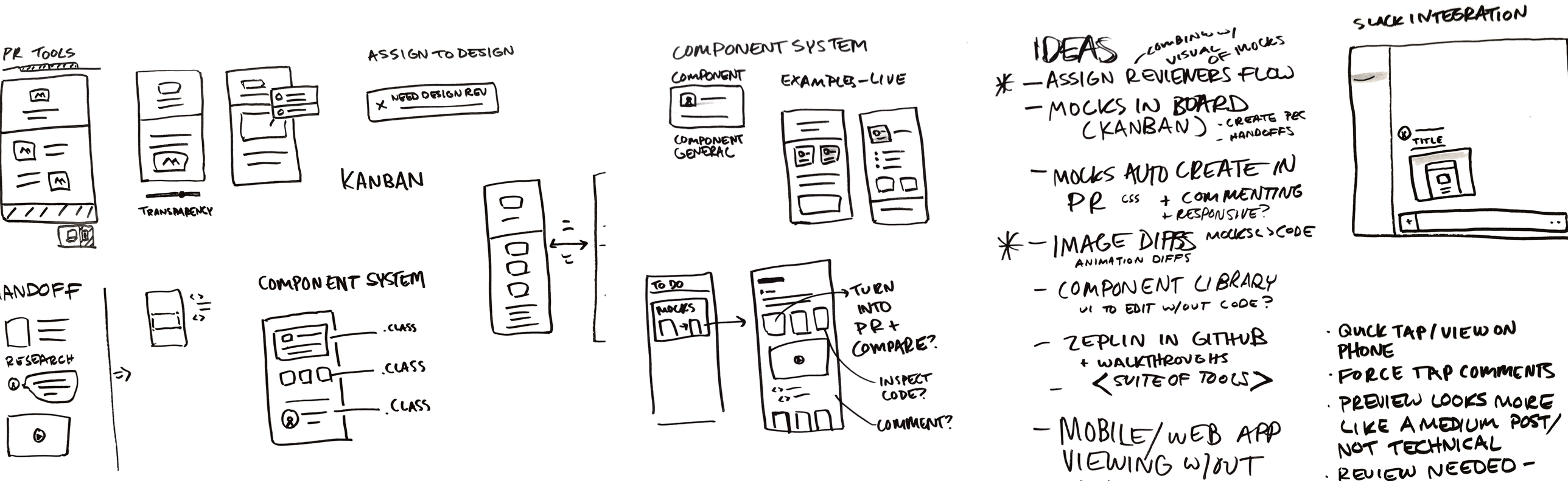
Designer: There are a lot of design bugs and no one is looping me in.
Developer: We don't have time to fix everything now, and I don't know when to loop in the designer.

Ship

Designer: This isn't as high quality as I had hoped and I feel frustrated.
Developer: This isn't as high quality as I had hoped and I feel frustrated.

I did a series of rapid timed sketching activities to produce a lot of ideas in a short amount of time.

I noted which ideas seemed the most interesting to me, and **chose three concepts to sketch out in more detail.**



Ideation / **Concept 1: Design Tools in Kanban Boards**

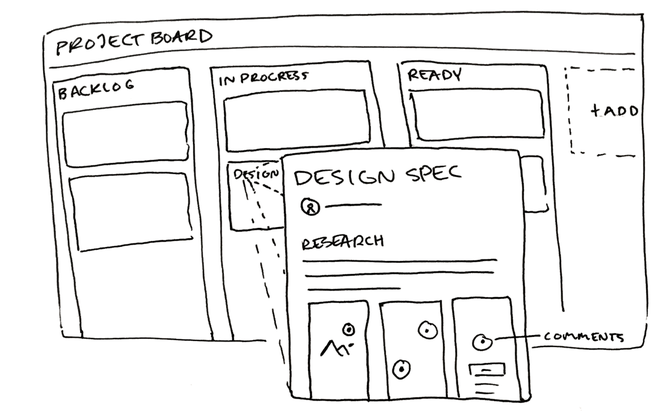
Since GitHub recently introduced kanban-style project boards, I thought it would be interesting to incorporate design into the process. Some ideas include:

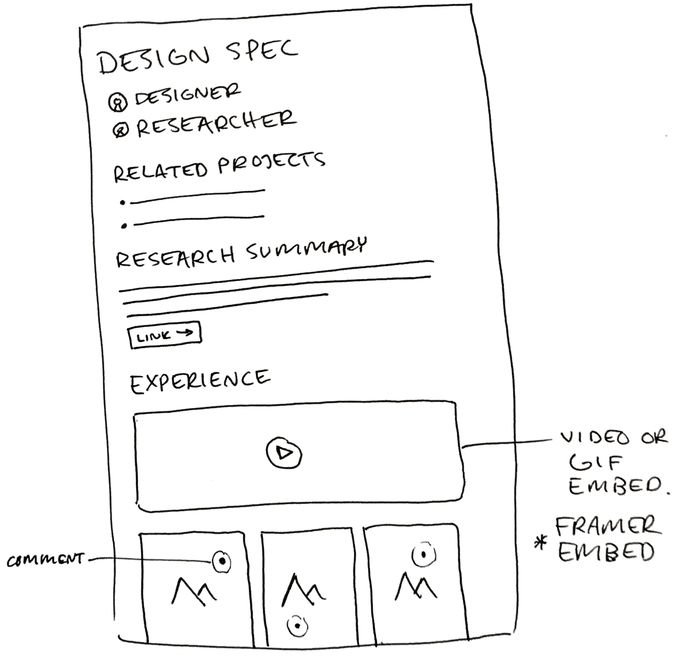
- Create a “design spec” card for the project board.

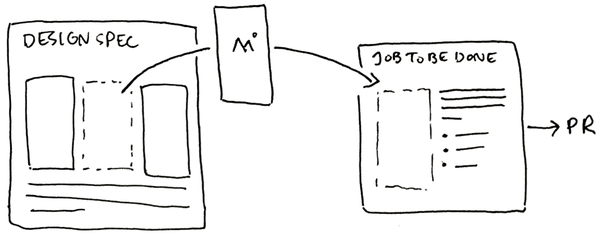
- Design specs can provide redlining features, as well as show the full intention of the design through diagramming and research.

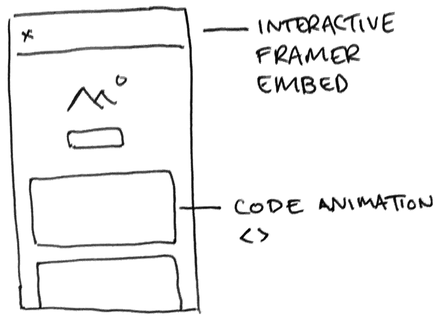
- Mocks can be pulled from design specs and attached to other cards.

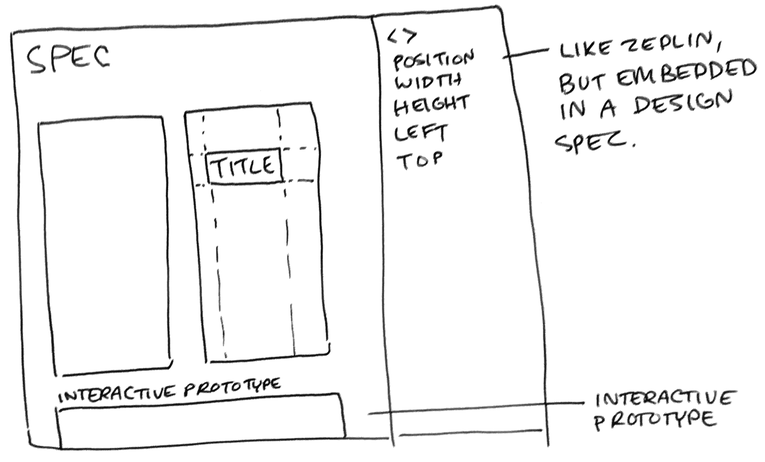
- Cards that get converted into Pull Requests could attach the mocks as a reference.

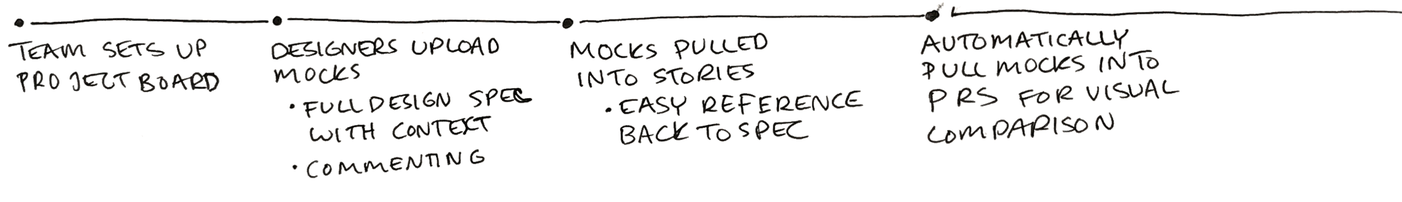






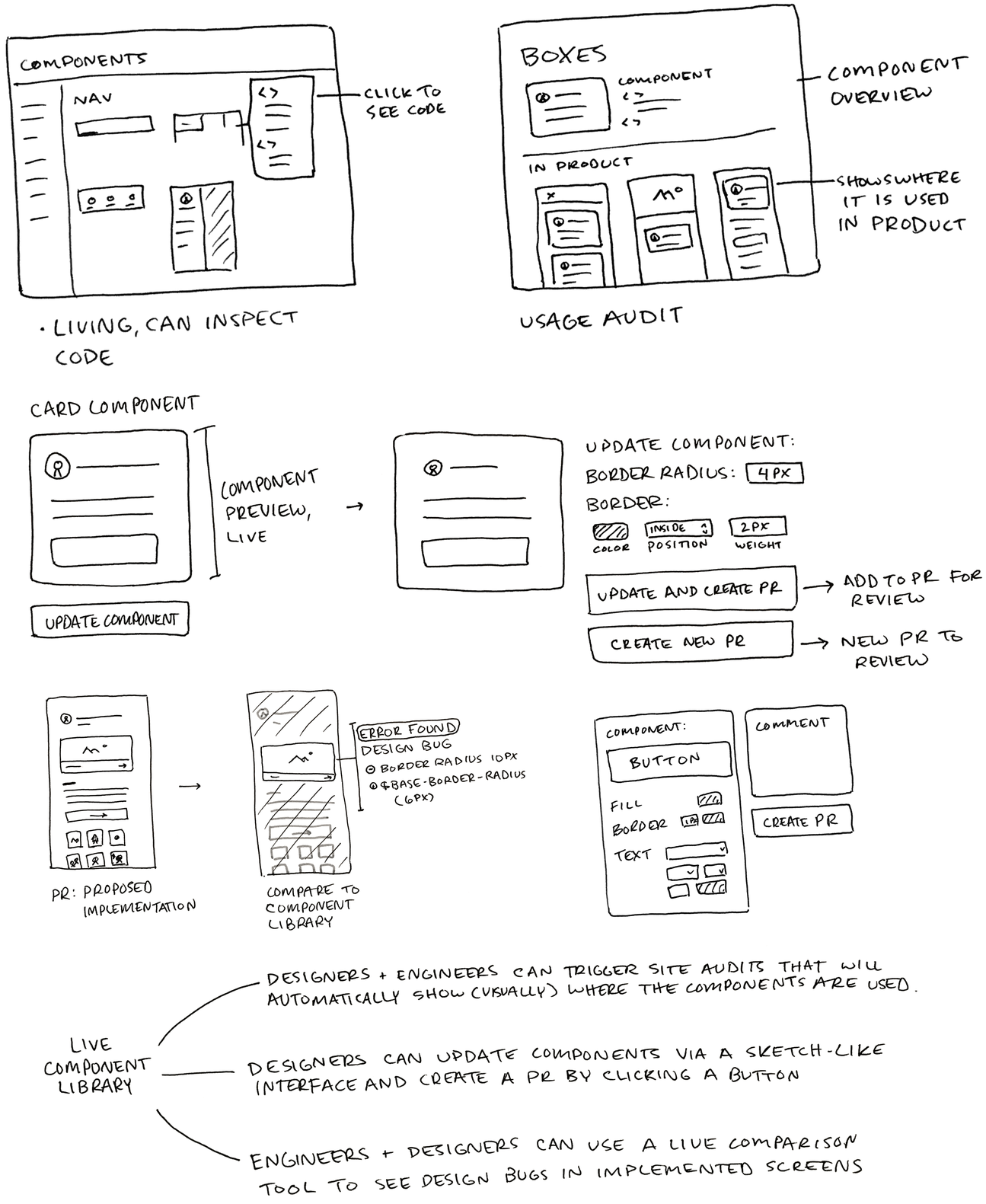






Creating component systems is always a challenge, often because designers and developers have a hard time keeping everything updated.

This concept was an exploration around creating a living component library. Designers could update components through a user-friendly UI that automatically submits code for review.

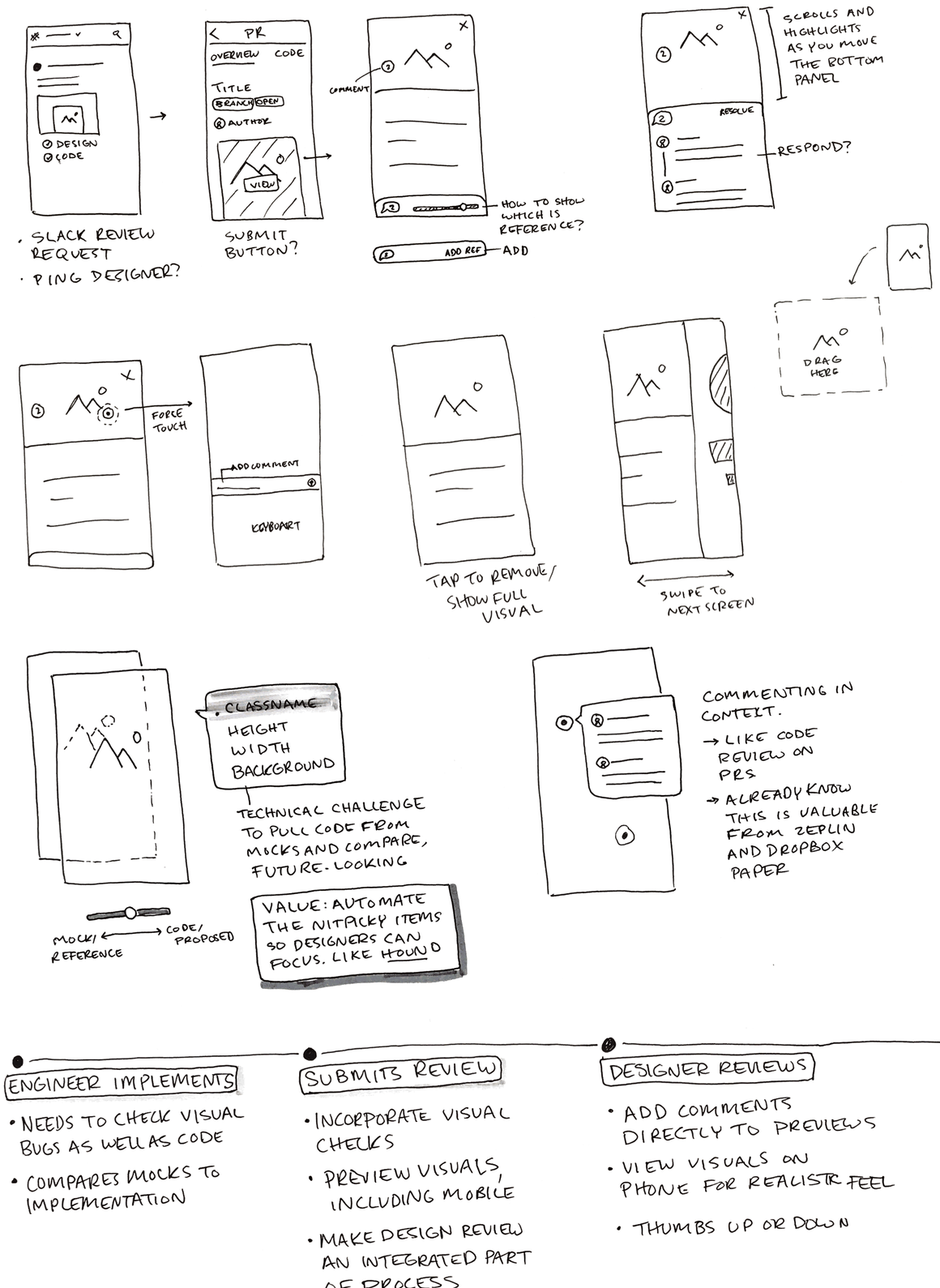


Ideation / **Concept 3: Visual Reviews in PRs**

Code review processes currently do not include design checks, even when the code is changing the design.

This concept explores updating the Pull Request flow to include automated and manual visual checks. Some of the value includes:

- Automatically tagging designers for visual reviews.
- Including visual changes alongside code changes.
- Collaboration tools for better communication.
- A potential focus on mobile to improve the quality of feedback. Viewing mobile work on an actual device is closer to the real thing and will improve feedback.



I ultimately chose the third concept: Visual Reviews in PRs

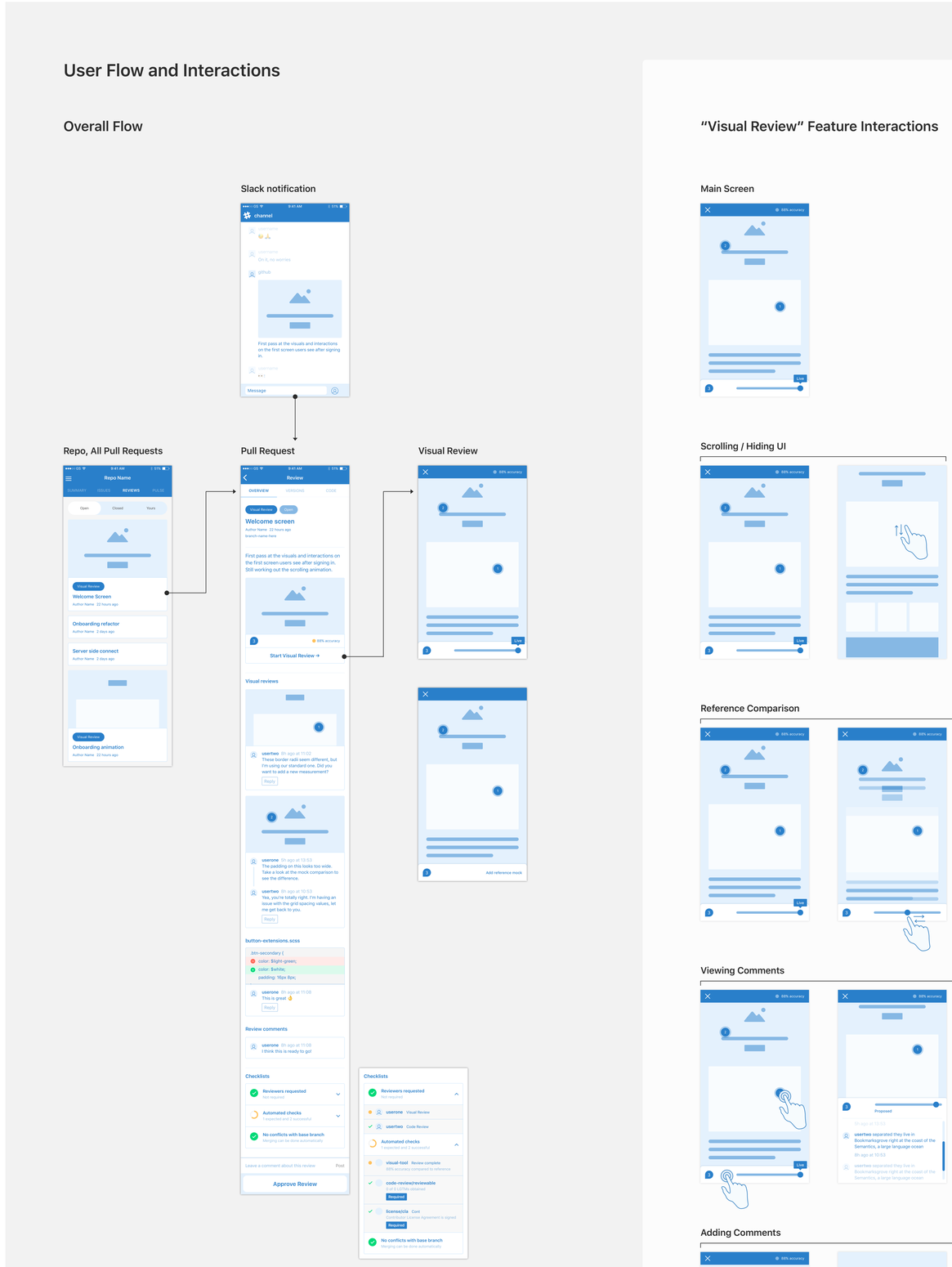
I decided to do a native version to show how it could optimize feedback for mobile work.

For a first pass at this feature, I scoped the experience to the designer’s perspective and established three main points in the user experience:

1. A “hook” to alert the designer that a review needs attention.
2. A redesigned Pull Request flow that includes checks for visuals.
3. A feature that allows designers and developers to check the accuracy of the visuals by comparing the implemented version to the reference mocks.

As an add-on, I decided to redesign the list of PRs in the main repository to apply the new visual styles to the entire flow.

View full wireframe mocks here →

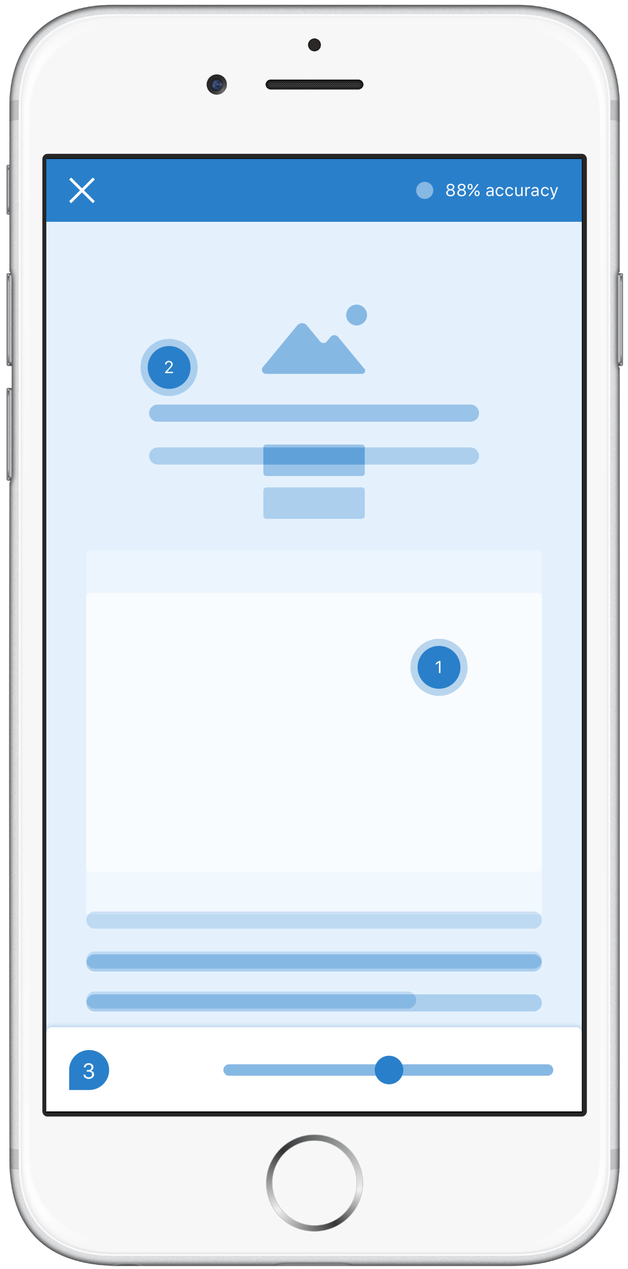


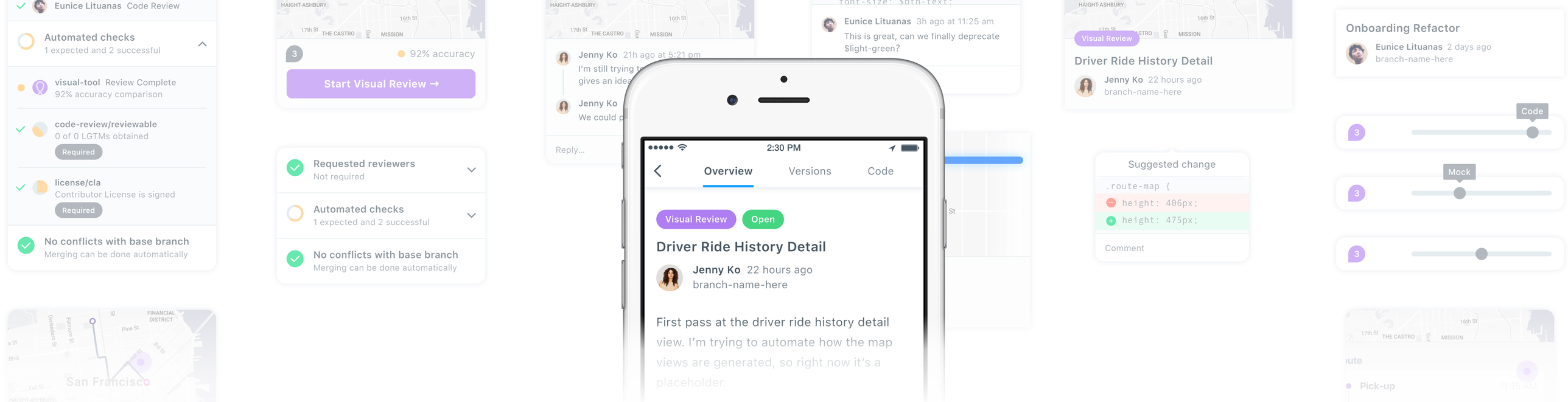
I decided to prototype the interactions for the “Visual Review” feature before moving on to visual design. Since this feature would rely heavily on interactions, it felt important to work out some of the issues before moving forward.

One improvement I made after prototyping was that I realized that the navigation bar on this screen was too heavy and obscured the view.

I also realized how future-facing the automated “best guess” code feature was when playing around with an actual prototype. I decided to make sure the final design would hold up without this feature and use it to establish future direction.

View prototype with interactions here →

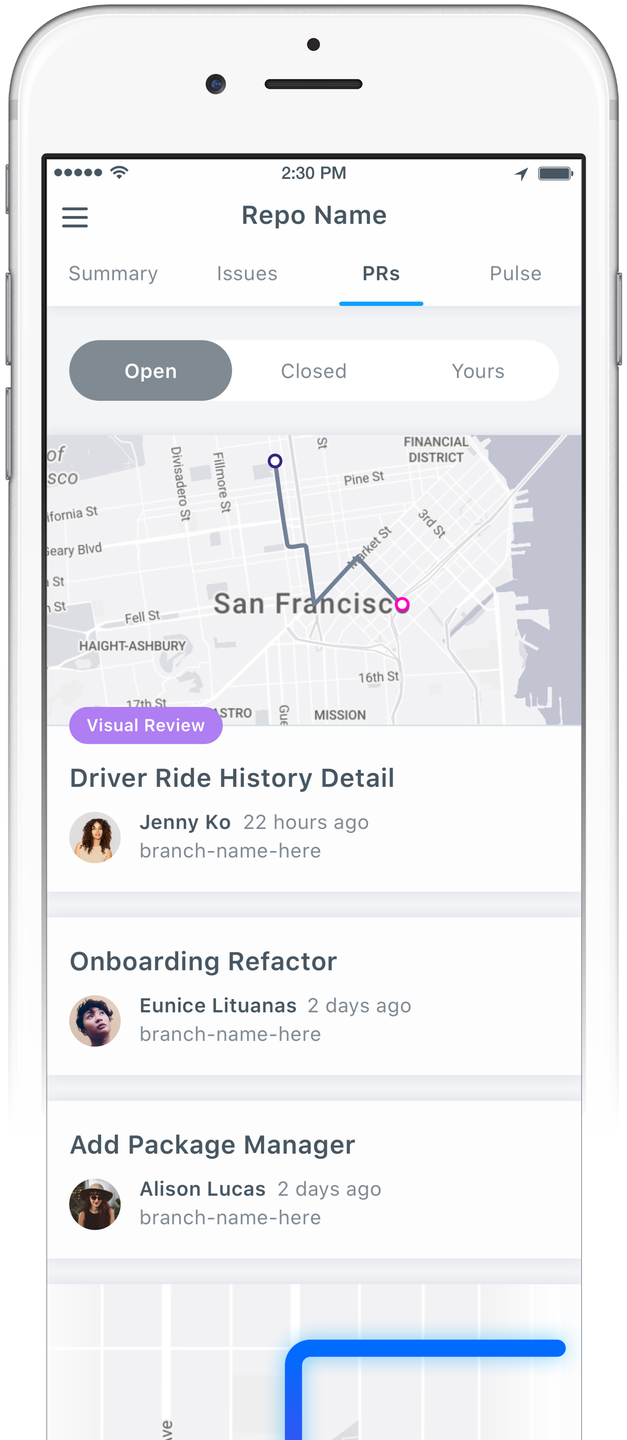


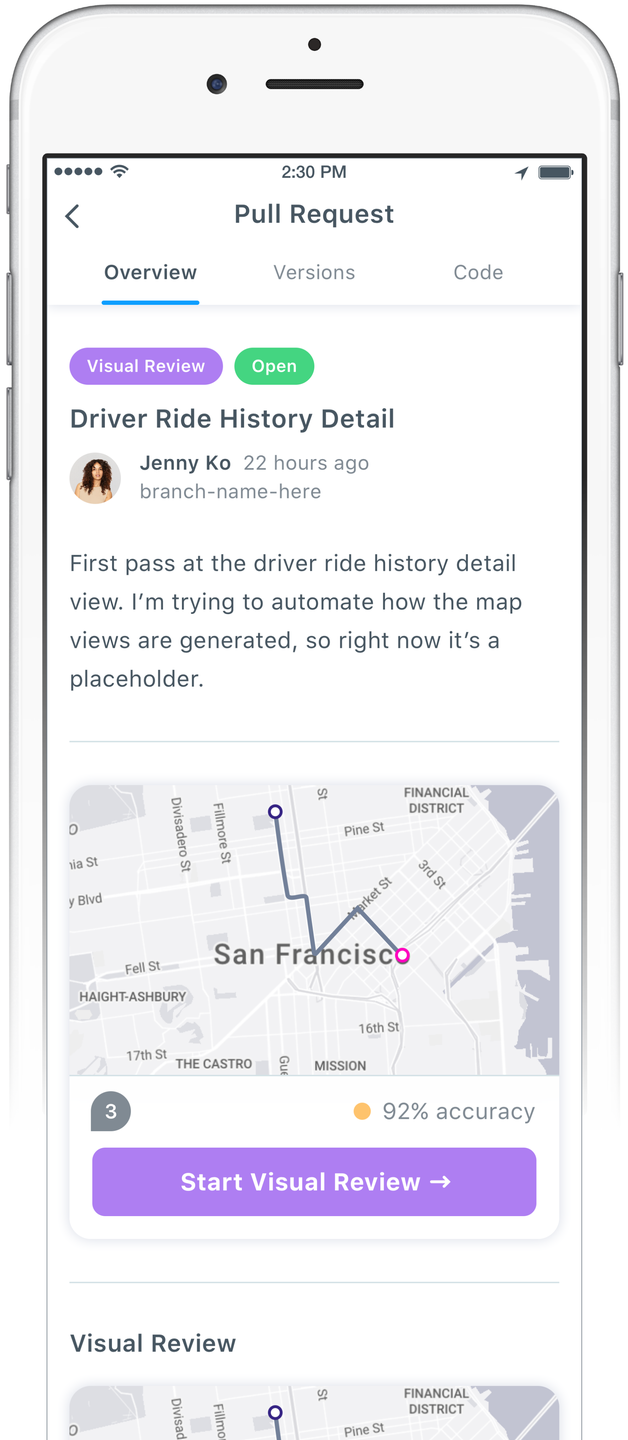


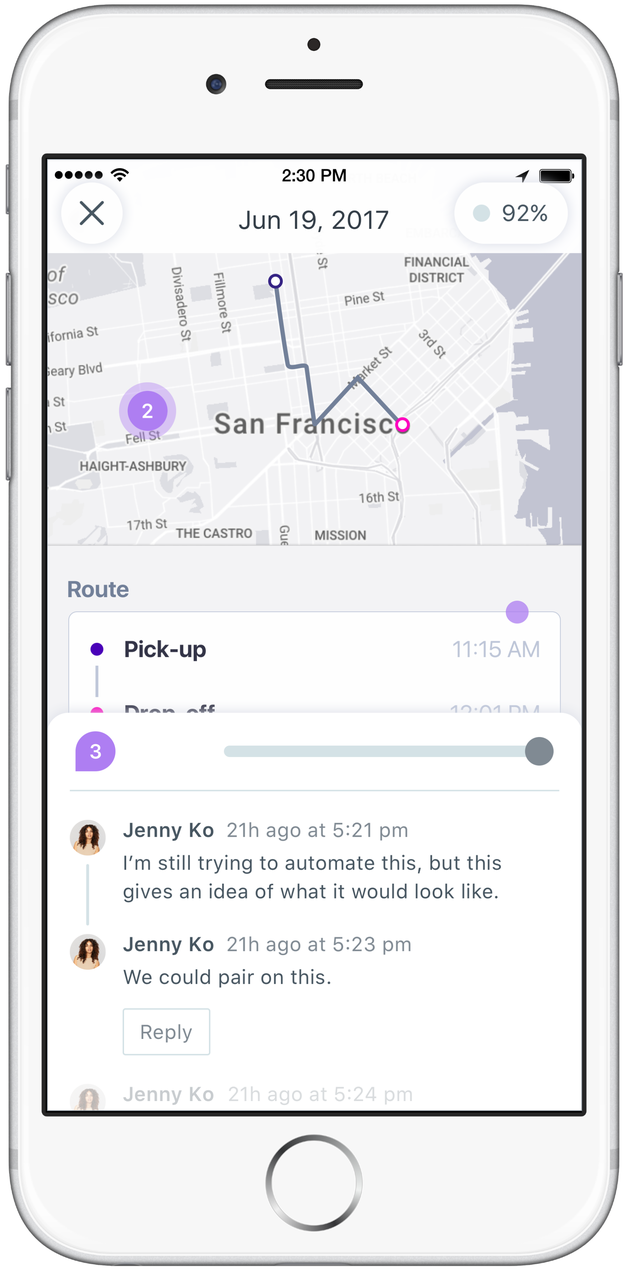
Final Design

Visual Reviews for PRs

The following is a walkthrough of the final user experience and visual design for the Visual Review feature.







Visual Review feature

Scenario: An engineer just submitted a PR for a feature that includes design work. The designer was included in the list of reviewers as a “visual reviewer”.

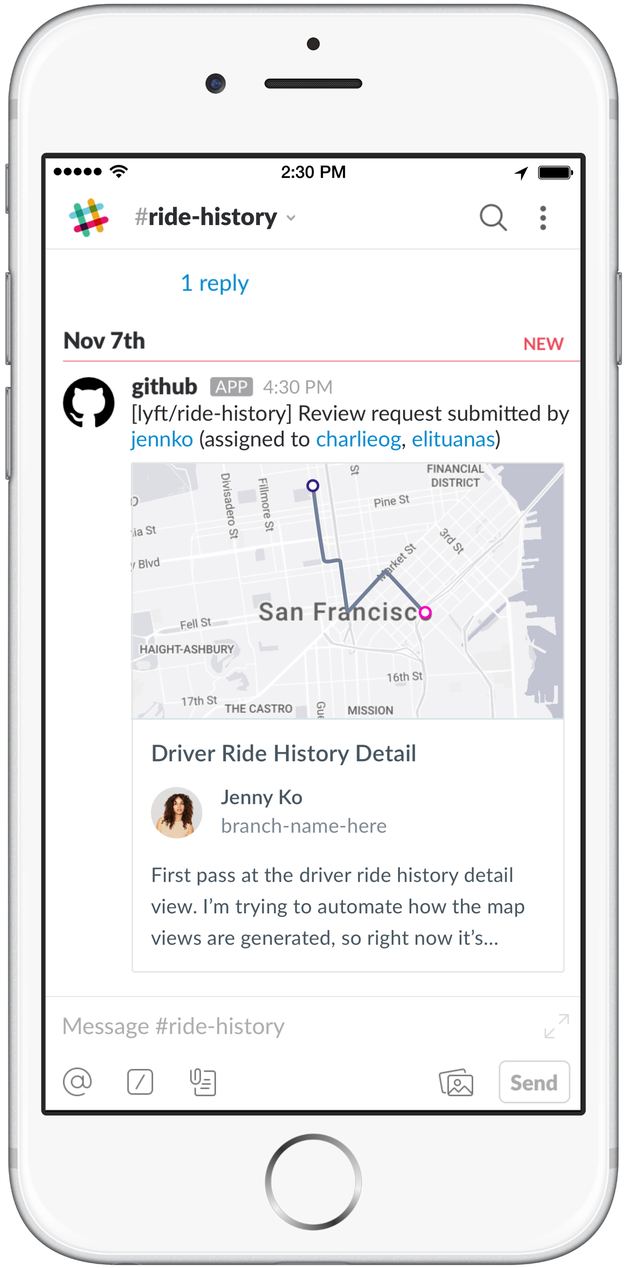
The designer gets pinged when the GitHub integration posts the PR in Slack. This is the quick entry point to the PR.

The “users” are:

Jenny Ko, PR author

Charlie Ogunse, Designer

Eunice Lituanas, Developer

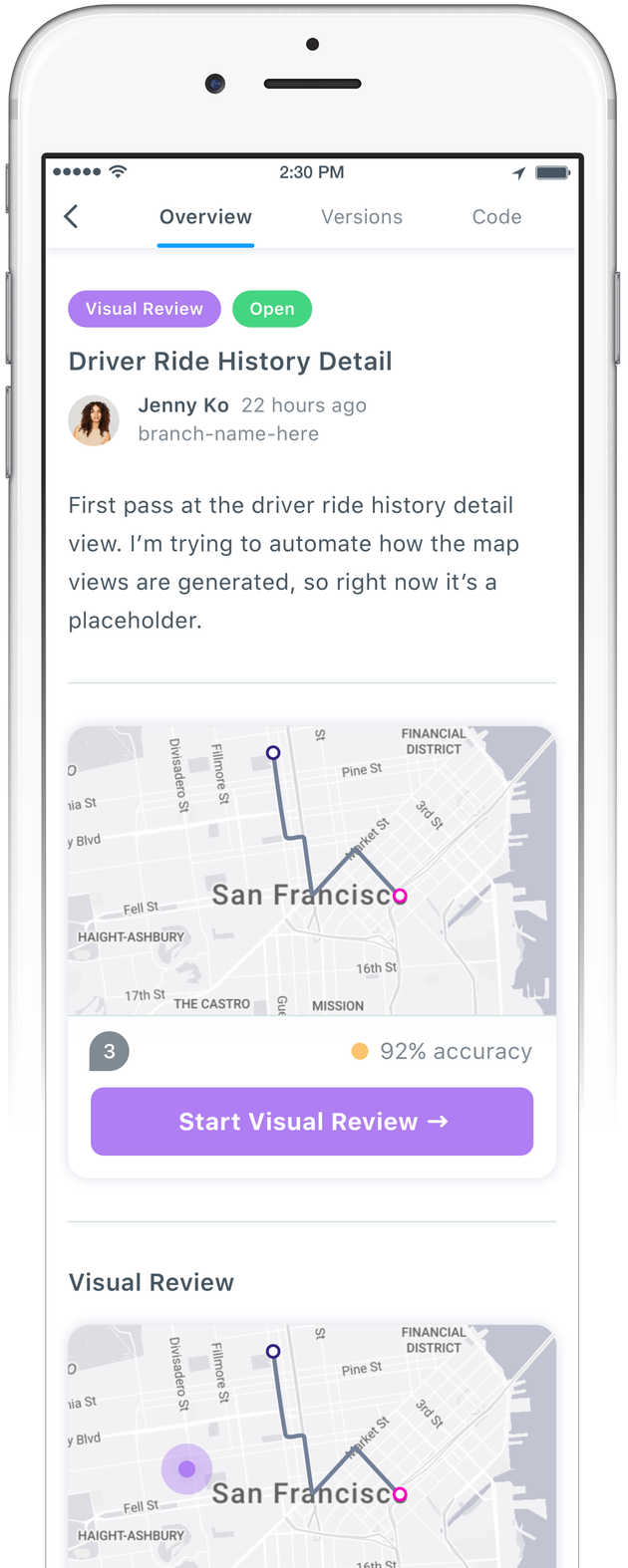


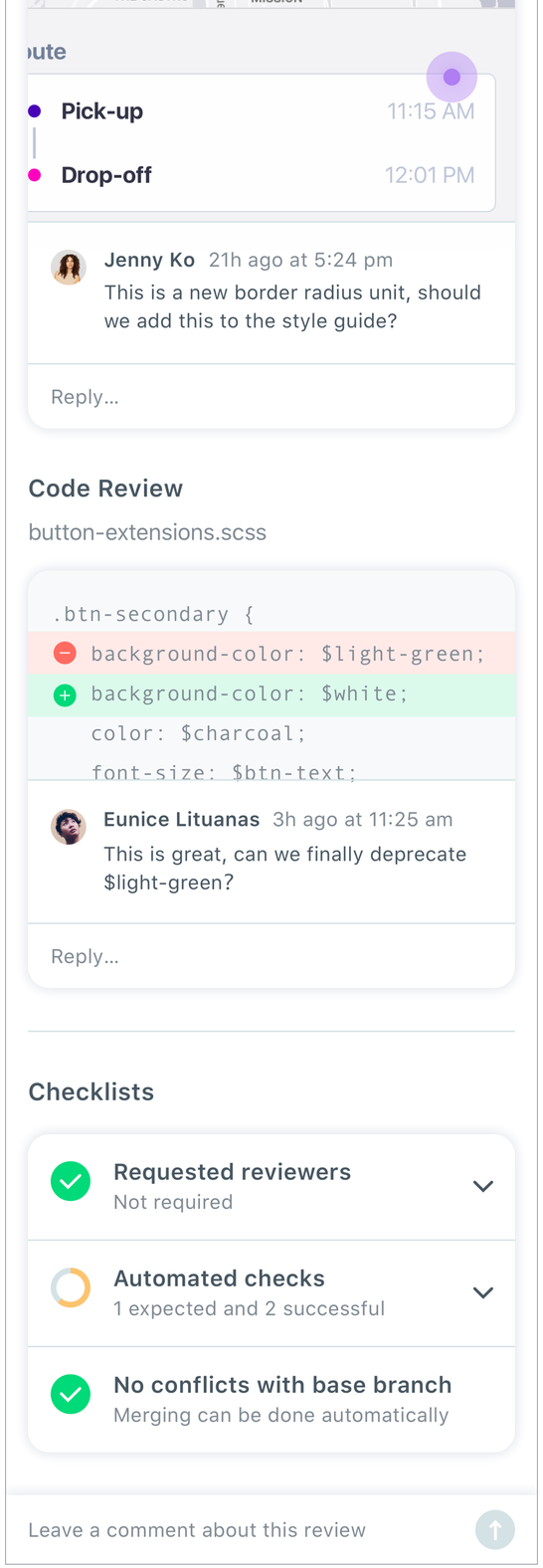
Final Design / **Pull Request With Integrated Visual Review**

This is the redesigned PR screen that the designer sees. It includes:

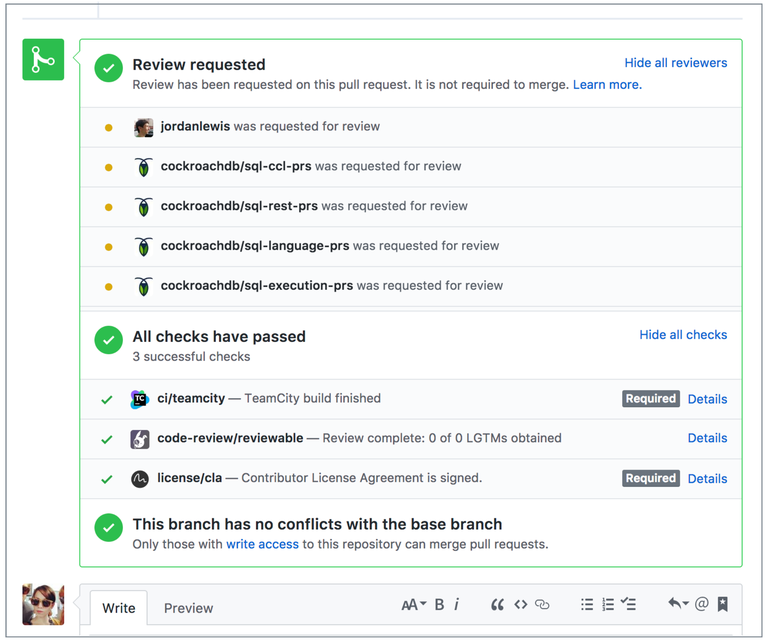
- An up-front summary section with an overview.
- A Visual Review feature incorporated into the main experience with a clear call-to-action, increasing its integration into the development process.
- Updated comment blocks for code and visual feedback.
- An updated checklist module that includes visual review checks.

View a detailed PR anatomy here →

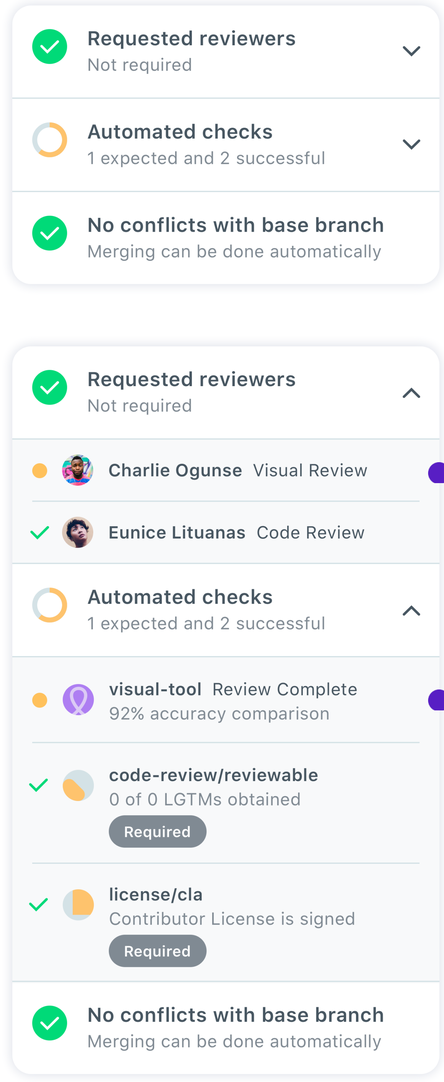




This is a closer look at the updated checklist module, which includes visual review features.



The existing "checklist" feature in PRs



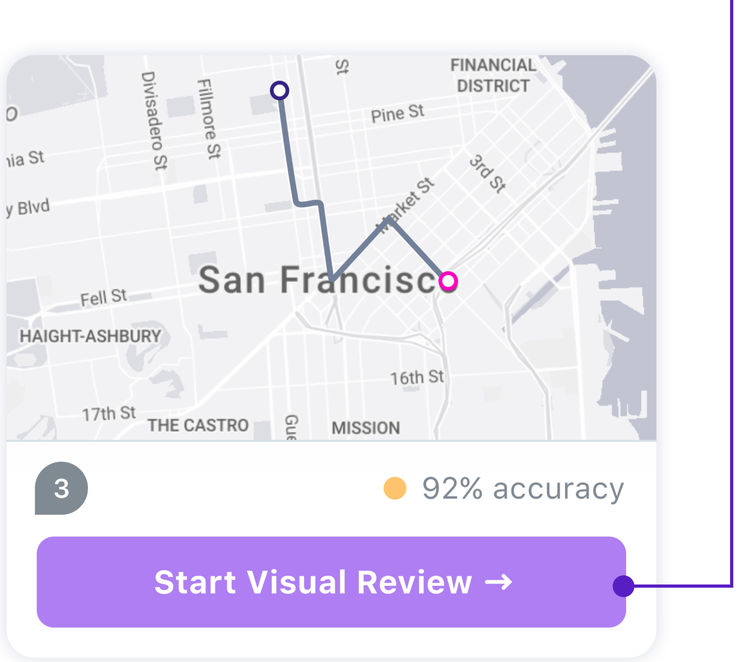
Visual Review Request
Assign designers to review visuals

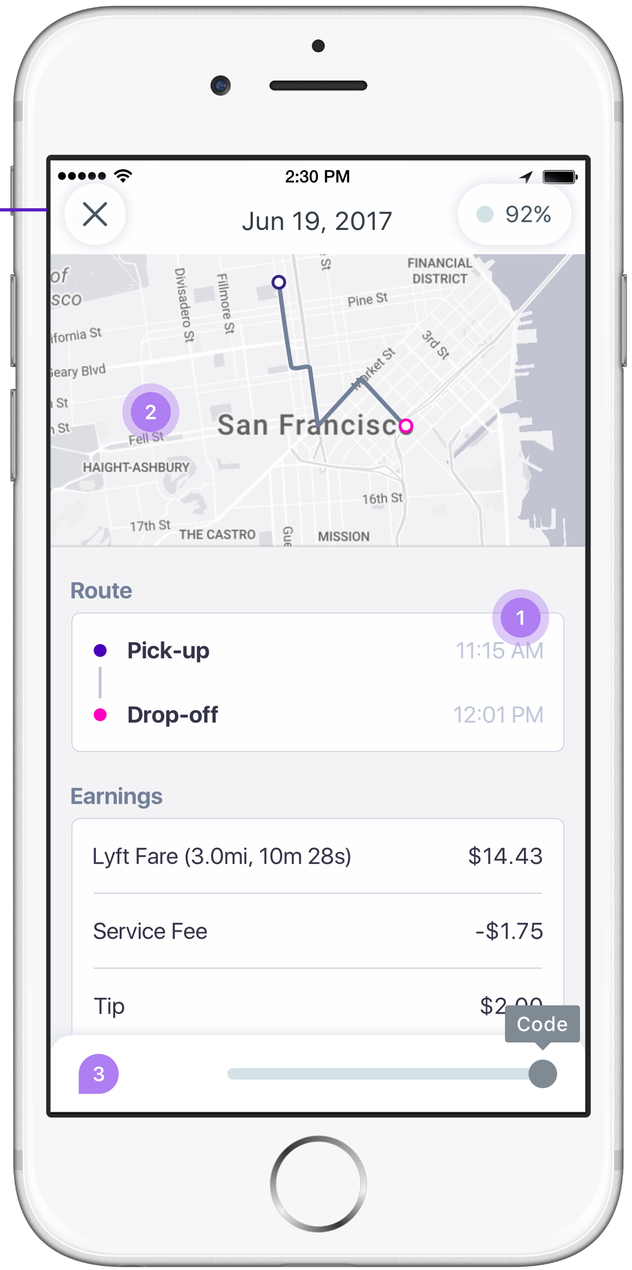
Automated Visual Comparison
Future facing idea

Updated "checklist" feature

The designer can start a visual review by tapping the module on the PR screen. This launches the Visual Review feature.

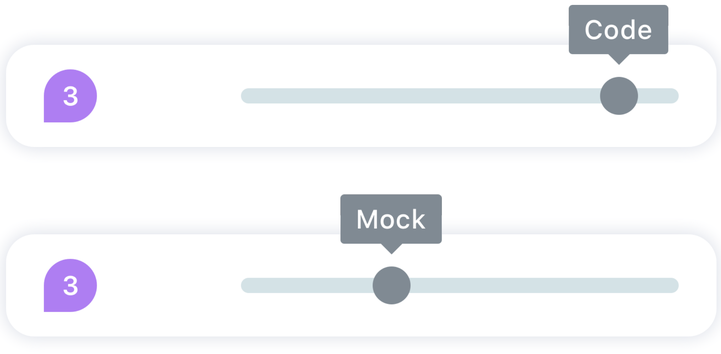
View a detailed Visual Review anatomy here →

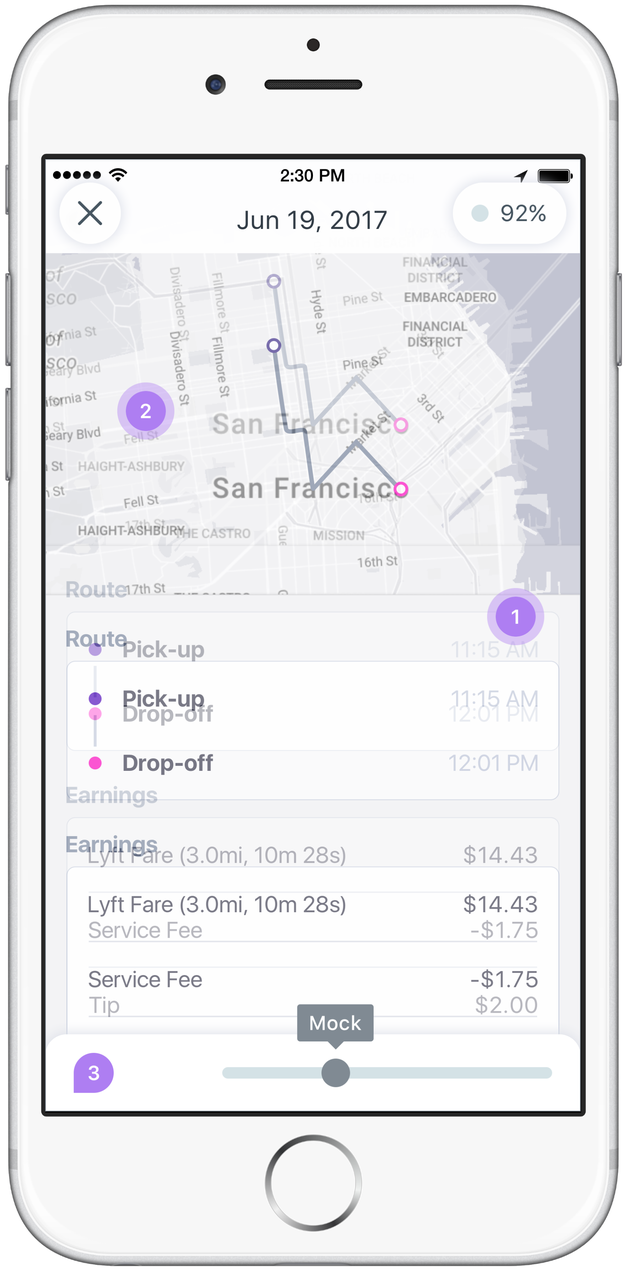




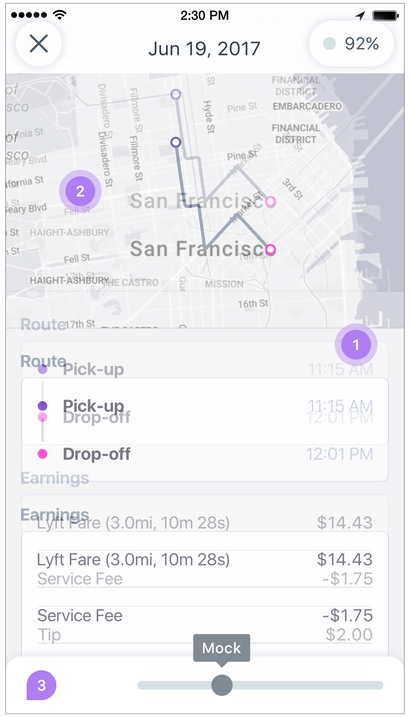
The Visual Review feature includes:

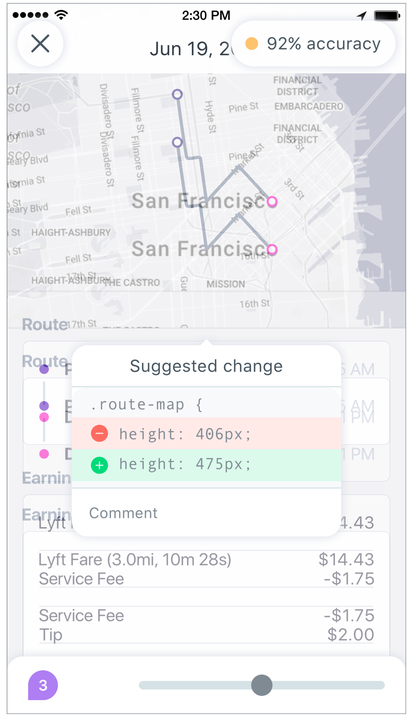
- An “onion skin” comparison between the screenshot and reference mocks.
- Commenting in context.
- The ability to swipe to see more screens.
- A future-facing idea for automatically catching design bugs and suggested CSS updates. This would be out-of-scope for the near term.

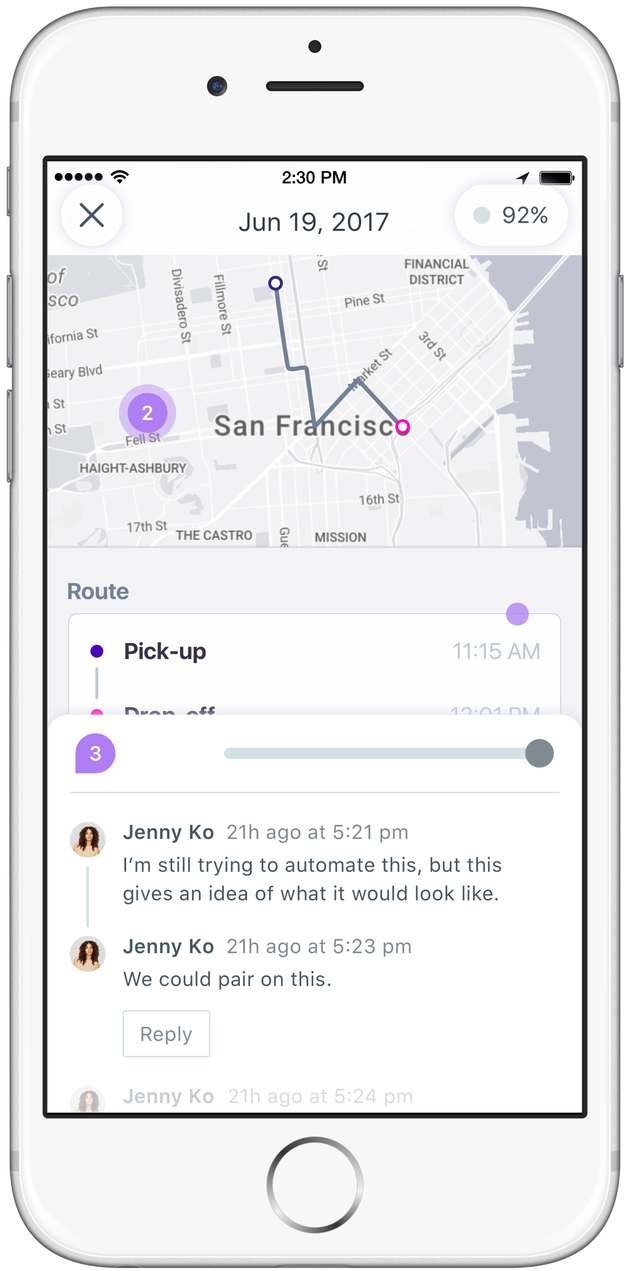


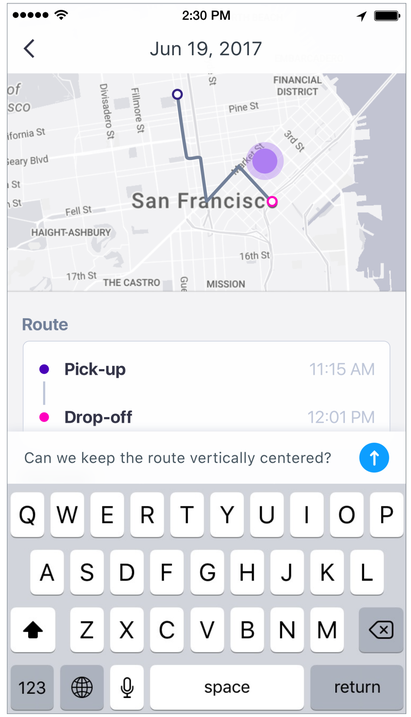


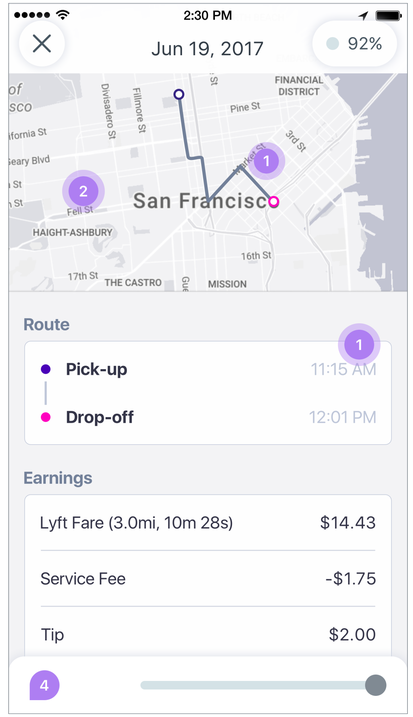
Final Design / **Visual Review Interactions**







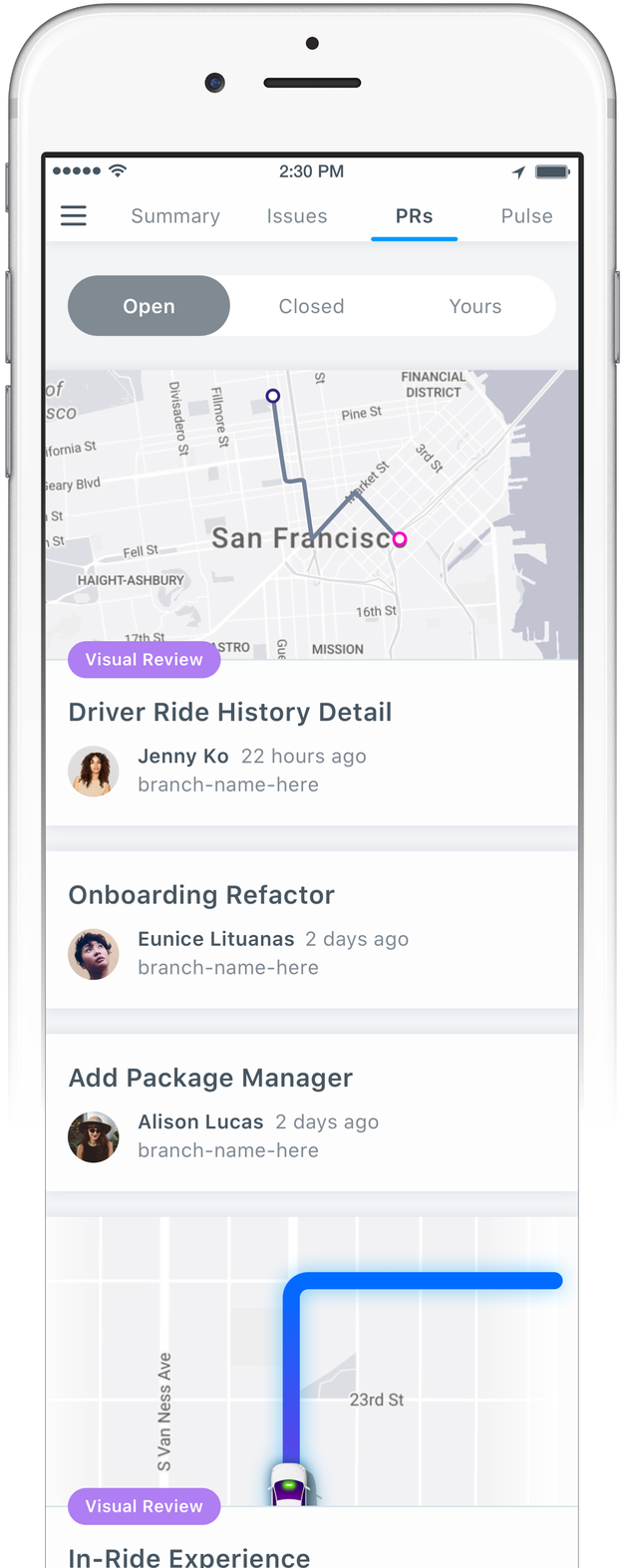




View prototyped interactions here →

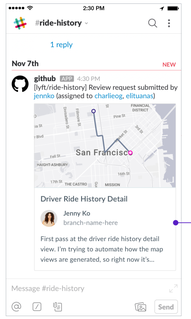
Final Design / **Repository List of PRs**

The user can also go up a level in the navigation to see the full list of PRs in the current repository.



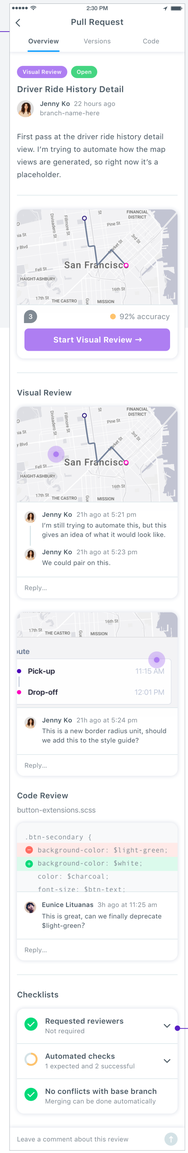
Final Design / **Overall Experience**

Slack Automatic Embed



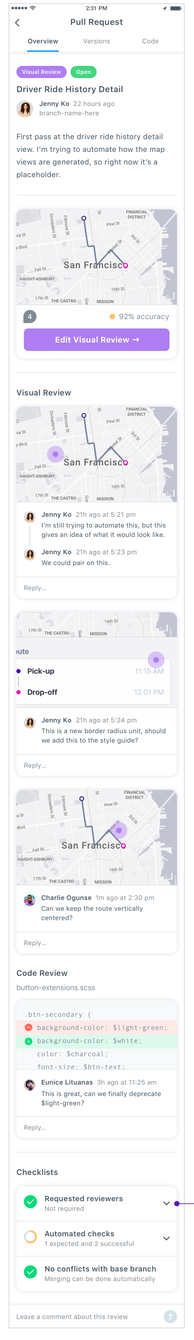
Pull Request

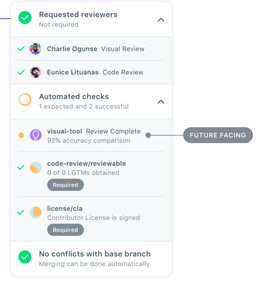
Full view (before visual review)



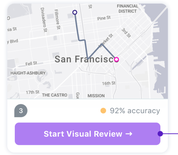
iPhone view (before visual review)

Full view (after visual review)

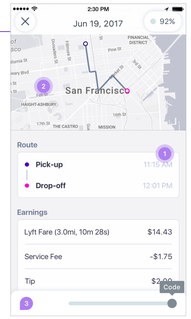




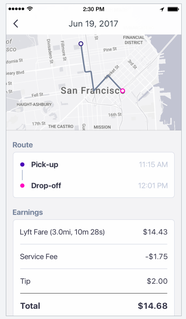
Visual Review



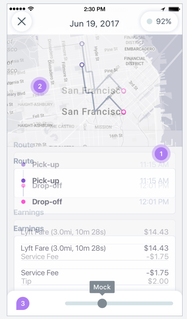
Main screen



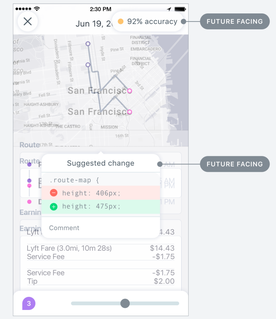
Tap to hide UI



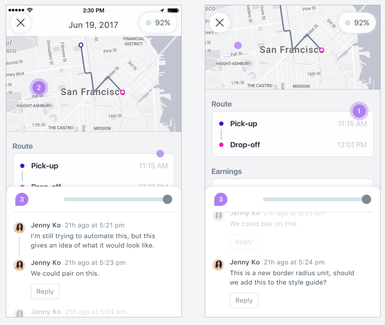
Comparison slider



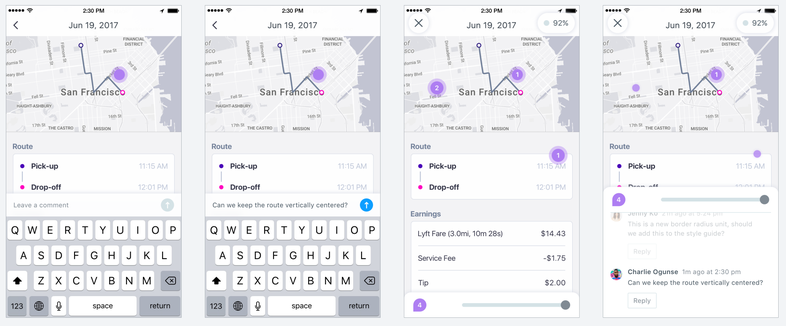
Automated visual comparison



View comments

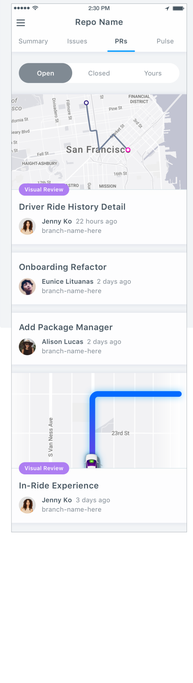


Add comments

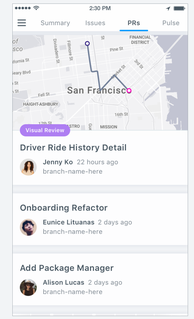


Repo Main View ("Reviews" Tab)

Full view

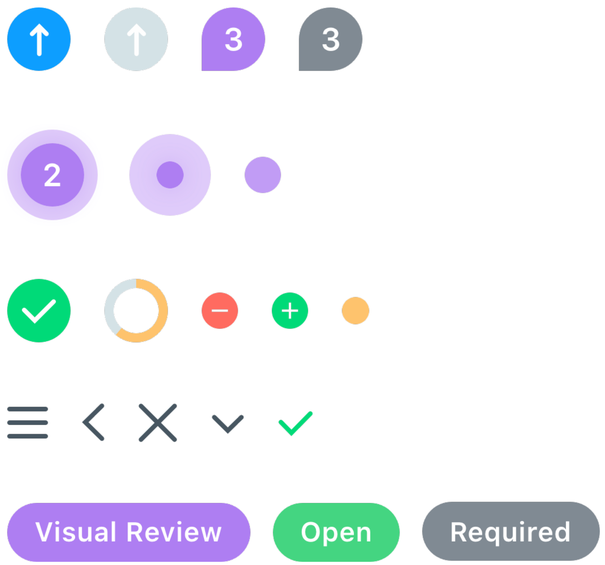


iPhone view



View all mocks and interactions here →

The final design includes an updated color palette, typography, and iconography.



Style Guide

Color Palette

#0D9EFF	#475661	#808A93	#D4E2E6	#F2F4F6	#F8F9FA
Main accent	Main text	Secondary text, placeholder text	Strokes, dividers	Background	Secondary background

#AE7EF2	#FEC36D	#44D581	#FF6B60
Visual review	Warning, pending	Open, success, addition	Blocked, error, deletion

Typography

SF UI Text

Aa Regular

Aa Semibold

Sizes @ 2x

Main headline
36pt (Semibold, 0.4)

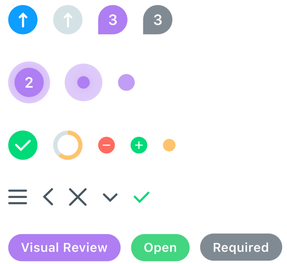
Secondary headline
34pt (Semibold, 0.39)

Body copy body copy body copy body copy body copy body copy
32pt (Regular & Semibold, 0.38)

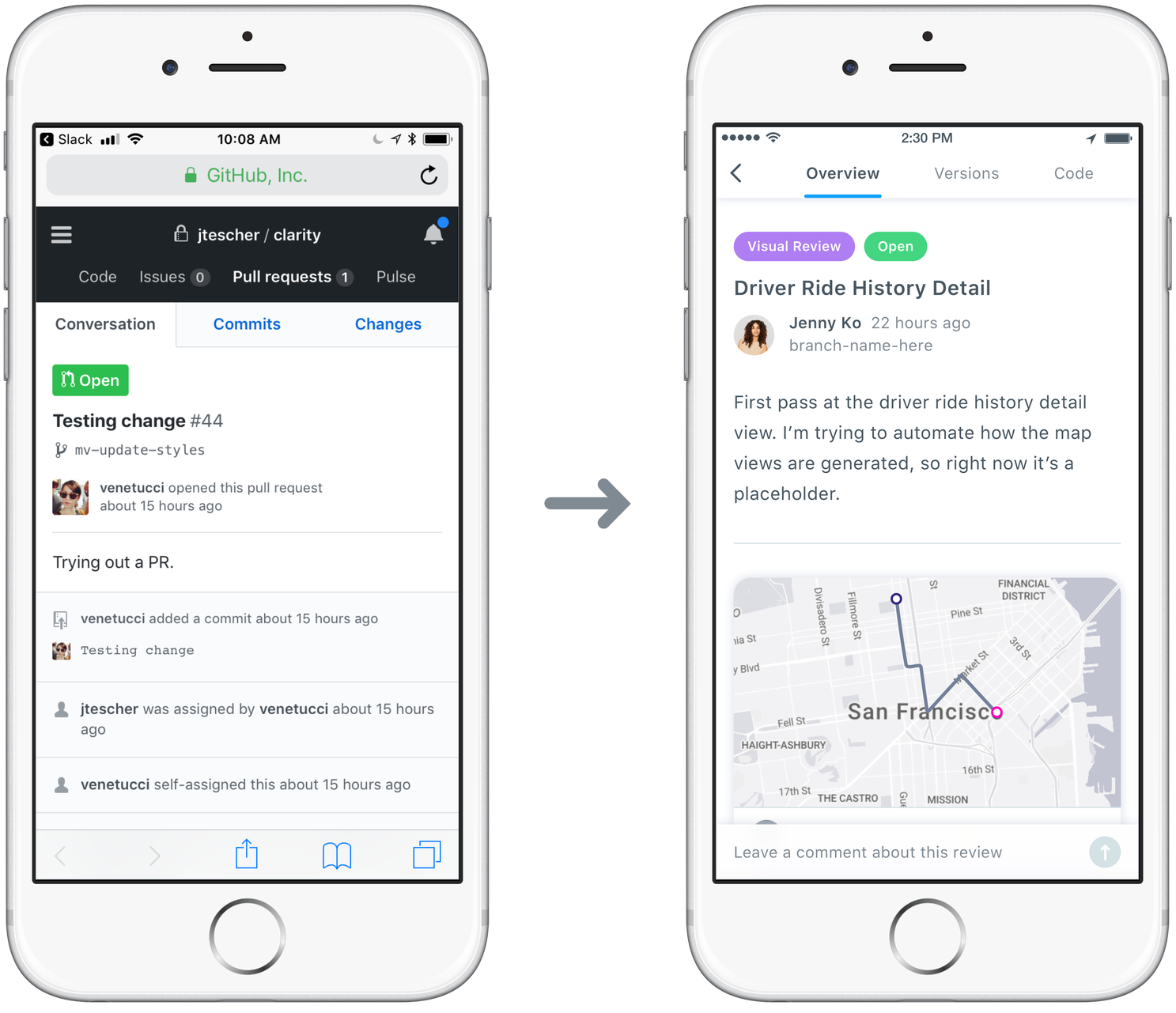
Subhead
28pt (Regular & Semibold, 0.36)

Minimum
24pt, (Semibold, 0.36)

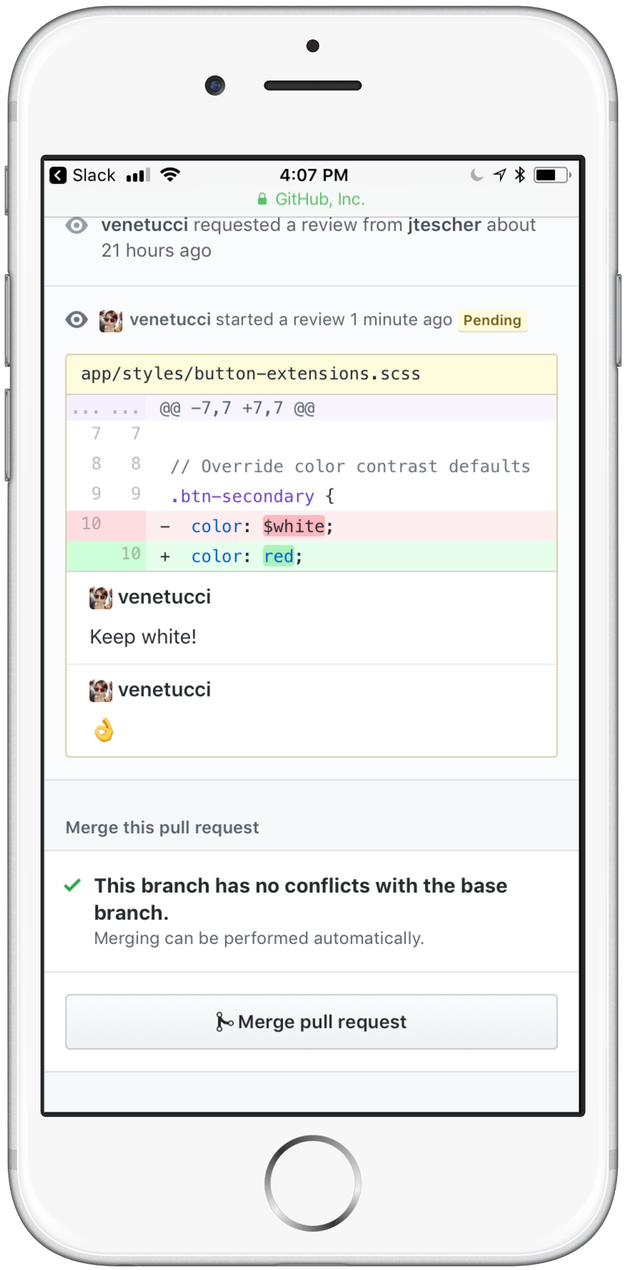
Iconography and Labels

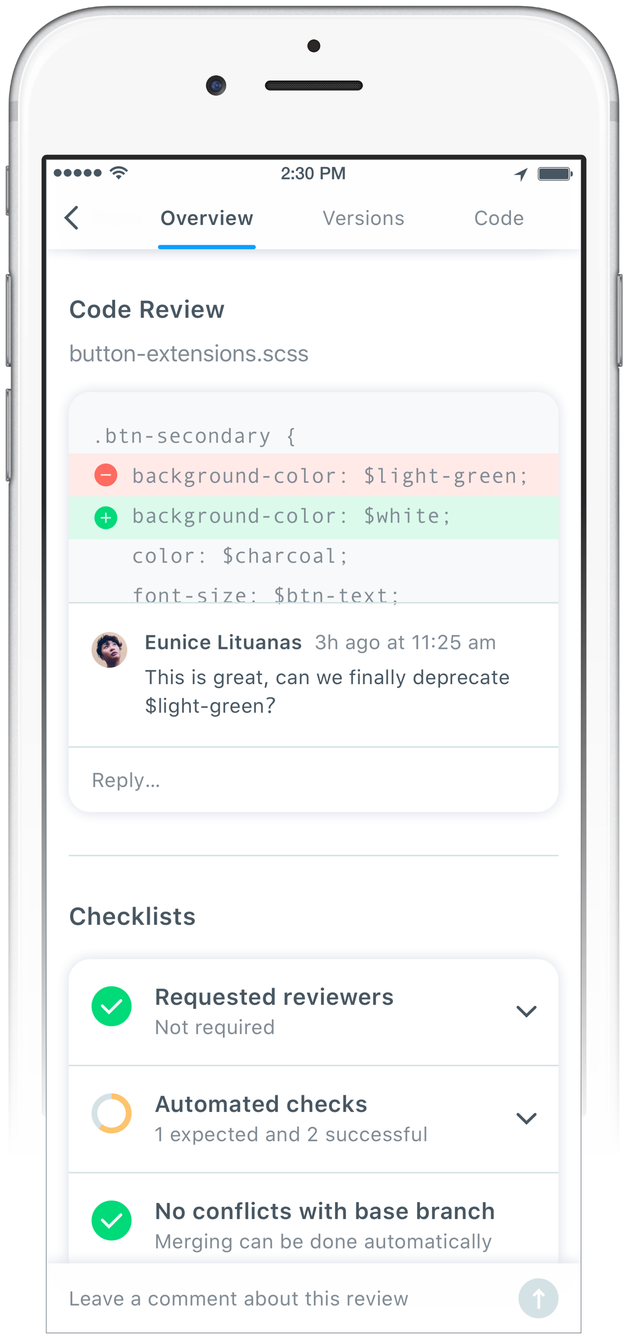


Visual Design / **Improved Information Hierarchy**



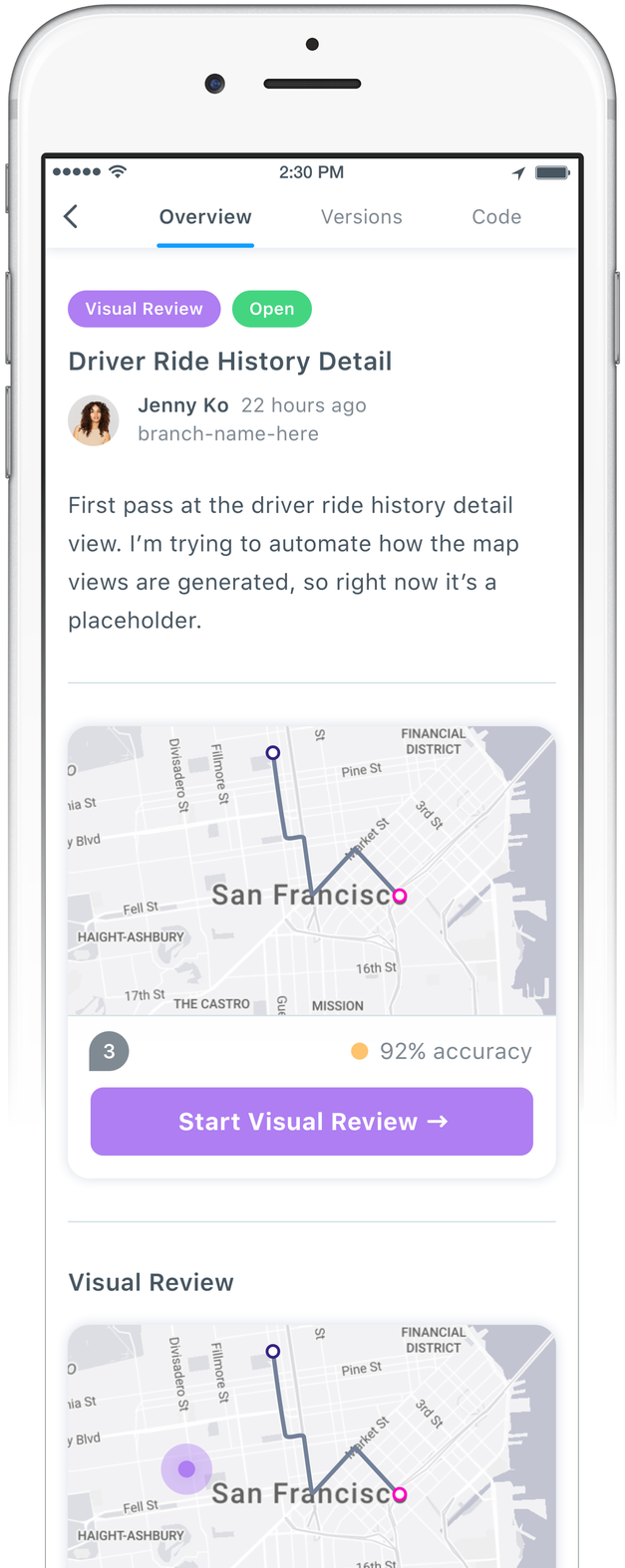
Visual Design / **Reduced Visual Clutter**

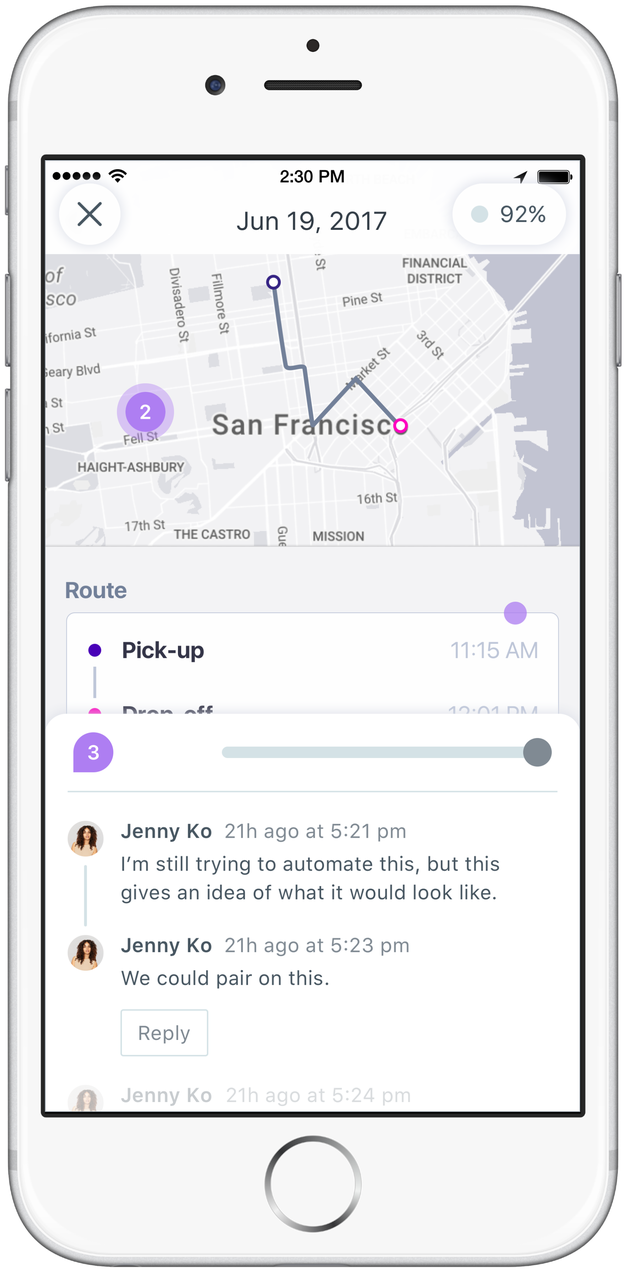




The final Visual Review feature and updated PR flow improve product development work in the following ways:

- Normalizes design reviews later in project lifecycles by integrating design into the code review process.
- Provides better tools for communication between designers and developers.
- Makes it easy to preview mobile work on a device as well as provide feedback directly on the visuals.
- Increases the ability to catch design bugs by providing a visual comparison tool.
- Proposes a direction for future improvements that can automate the design QA process. Automated tools have the potential to reduce overhead for teams and increase the quality of design in final products.





[All mocks and interactions →]

[User research overview →]

[Supplemental materials →]

Thank you