CS 194h: Assignment 14

Final Report

The Team

Austin Jones

Caroline Willis

Daniel Kharitonov

Emma Alderton

Our Value Proposition

Thundr | A Collaborative Brainstorming Tool

Problem Overview:

Through needfinding we realised that people want to use technology to increase group productivity, but are not willing to invest massive amounts of time learning how to use it.

At the same time through our personal experience we found that brainstorming sessions could be boring and time consuming. Therefore we chose to focus on a need for a better brainstorming tool that could be easy to use, engaging and available remotely.

Solution:

By the end of CS194h class we delivered our project Thundr that streamlines the brainstorming process by using voice interaction with a mobile app and persistent storage of created ideas on the web. We strived to create a simple and intuitive user interface designed to minimize time waste in joining the brainstorming session and running the task workflows. Our final app version allows for local and remote brainstorming, with results that can be easily organized and readily revisited.

Tasks

Create and Join New Brainstorm (Simple)

We chose this task because the users needed a quick and simple way to create a new brainstorm and join brainstorm sessions. This task helped us to think of methods to minimize the time and reduce barriers in joining and creating a brainstorms.

Our solution consists of the two main components: the **Web App** (projected screen visible to everyone), and the mobile **iOS App** (visible to participants on their phones). We expect one user to work as discussion facilitator and manage the web screen.

Web App:
Facilitator goes to Thundr website https://thundrweb.herokuapp.com/. From here clicking the large logo creates a new session. Six-digit code is generated for that unique brainstorm session and web screen changes to that session's canvas.

iOS App:
A participant can join by scanning the QR code displayed on the web app's session screen, which will open the Thundr app (must be pre-installed from https://web.stanford.edu/~austinhj [1]), and automatically join the current brainstorm session. Alternatively, the six-digit can be also entered manually.

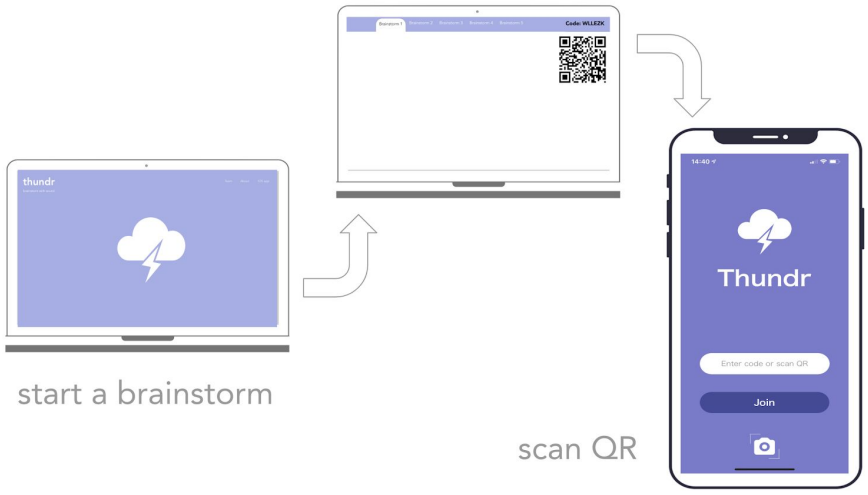


Figure 1. Create and Join Brainstorm Task Flow

[1] Due to limits on Apple student developer program, the app can only be installed to iPhones whitelisted for the use of the CS194h development team.

Submit and Edit Ideas (Complex)

Ideation is a vital part of the brainstorming process. We wanted users to feel engaged and 'in flow' when generating and submitting ideas, therefore we decided to utilize voice dictation as an avenue to achieve this. Furthermore, we wanted users to have the ability to edit ideas on their mobile apps before submission, and manage ideas on the web app after submission, a group discussion, or after expanding on an idea.

iOS:
The brainstorm participant has the option to click the 'mic button' to dictate an idea that is created on-screen while the user speaks into microphone. From here the user has a few different options. She can immediately submit the idea by pressing "submit" button, or click the mic button again to stop recording, and apply edits (if necessary). This dictate-then-edit cycle can be repeated multiple times until the idea is finalized. If the user didn't like the end result, she also have the option to click the 'x' in the top right corner and delete everything.

Web App:
After a user has submitted an idea, it shows on web app as a note. The facilitator controlling the web app can organize the ideas by dragging and editing any idea by clicking the "pencil" icon in the left top corner of note. They are also able to delete an idea altogether by clicking the 'x' in the top right corner of the note.

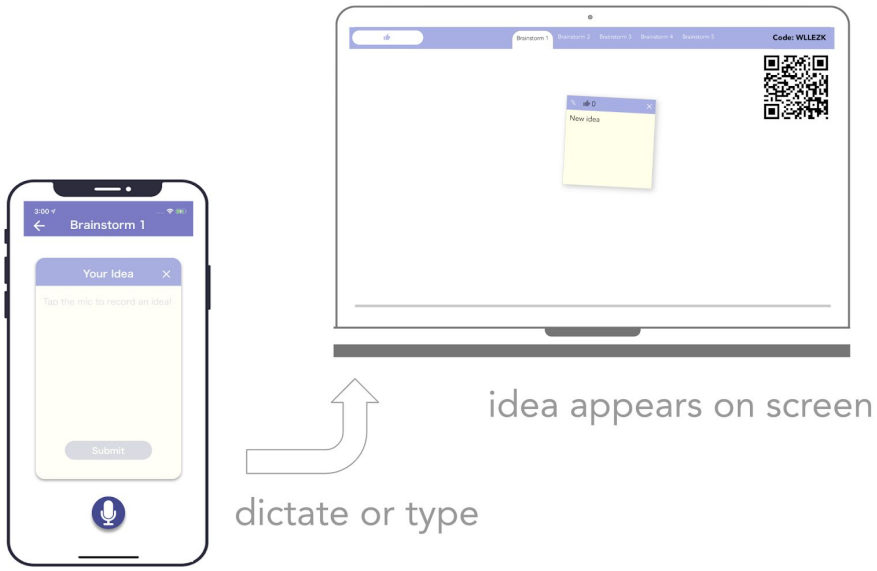


Figure 2. Idea Submission Task Flow

Vote on and Rank the Ideas (Moderate)

A quick and controlled method for users to rank their favourite ideas is important part of the brainstorm. Focusing on this task challenged us to experiment and determine the best way to deal with voting.

Web App:
To begin a voting session, facilitator clicks on the ‘vote’ button in the upper left of the brainstorm session screen of the web app. This forces all mobile client to switch into the voting screens where all submitted ideas are listed. Live votes are updated on the notes of web app as the cumulative count of all votes cast by the brainstorm participants.

iOS App:
Once the facilitator starts a voting session, the screen of the iOS App is switched into vote mode. Here the ideas from the current brainstorm are displayed. The user can then scroll through the ideas, up-vote an idea by clicking the ‘like’ button in the top right corner, or undo their vote by clicking again.

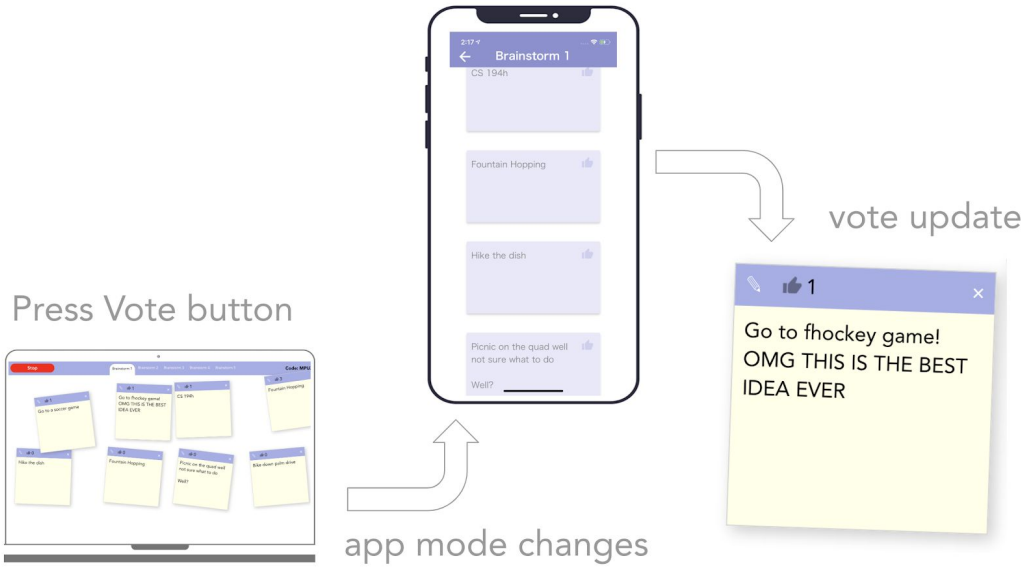


Figure 3. Voting Task Flow

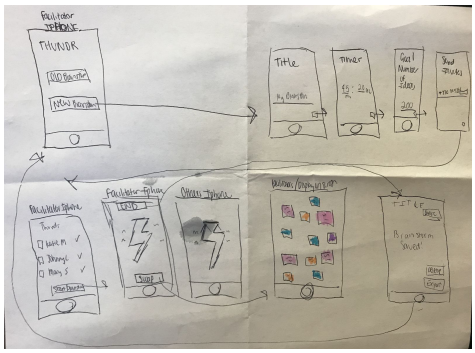
Note that the back-and-forth transition between submission and voting screens on different channels for the brainstorm participant remains seamless because it driven by a facilitator operating the web app. In a sense, mobile app only has one operating screen.

Design Evolution

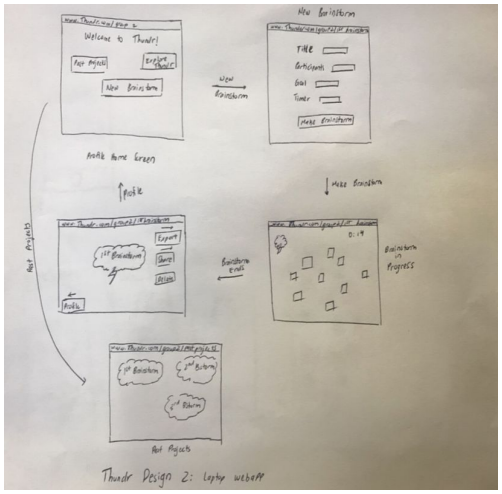
The alpha stage design involved two parts of the system: mobile and web app.

In our initial UI sketch (Figure 1), we focused on mobile app's three major tasks: *Creating a Brainstorm*, *Voting on ideas*, and *Accessing old brainstorms*. We used multiple step-by-step flows to obtain data in order to create the brainstorm. The app had a separate switch between idea recording and the brainstorm list. The plan was to make a way in which the mobile app could be self-contained for all information needed for a brainstorm. We used this design for our low-fi prototype (Tasks 1-3).

Low-Fi

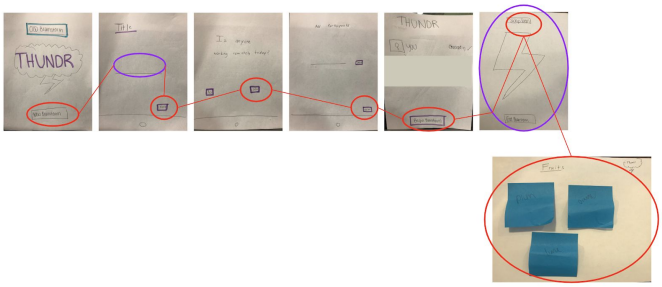


Mobile app part

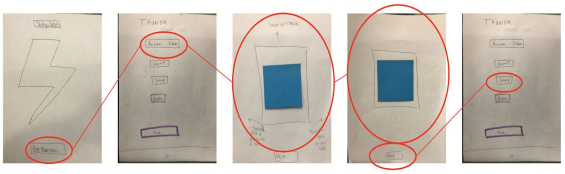


Web app part

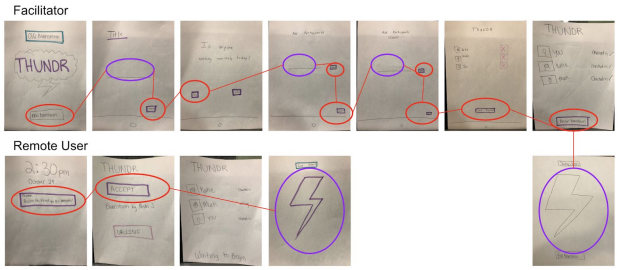
Task 1: Create and Share New Ideas



Task 2: Highlight and Present Favorite Ideas



Task 3: Work Remotely from Team



Medium-Fi

We used Marvel to create our mobile medium-fi prototype. Marvel was easy to use and allowed us to create a wireframe we envisioned. However, it did not allow for collaboration, and only one person could work on a prototype at a time. It also did not permit us to test voice interaction, creating the need for the “Wizard of Oz” testing.

Task 1: creating and sharing the ideas

Our medium-fi vision for ideation involved dictating the idea, assigning it to a particular brainstorm, and choosing the participants to share it with.

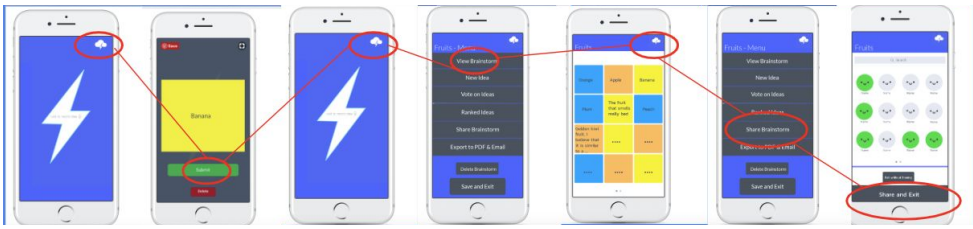


Figure 5. Med-Fi Idea submission Task Flow

Task 2: voting

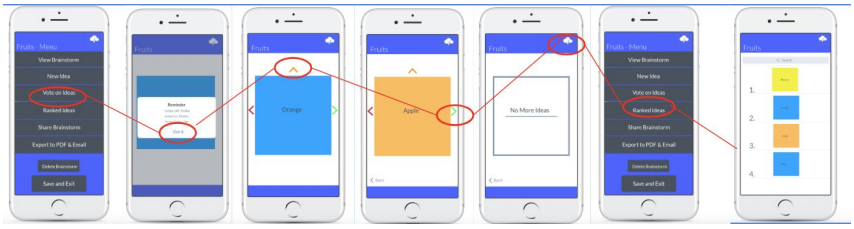


Figure 6. Med-Fi Voting Task Flow

In medium-fi prototype, we recast our ‘highlight and present ideas’ task into ‘voting’. Voting was implemented as a rather involved sequence of actions started by selecting an idea to vote, swiping it tinder-style and repeating for other ideas.

Task 3: creating and sharing a brainstorm

Upon realizing that brainstorm creation and sharing are related, we organized task three around these two actions. Our med-fi prototype included explicit steps for creating and naming a brainstorm, as well as manually adding the participants.

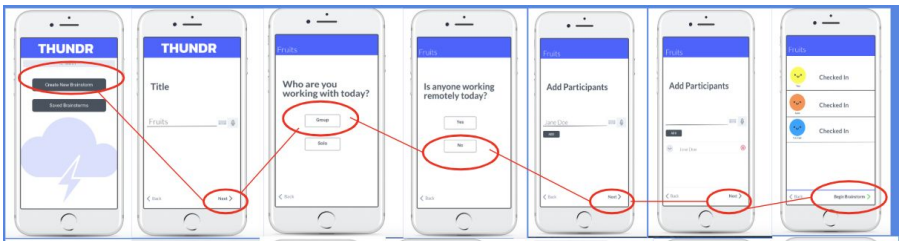
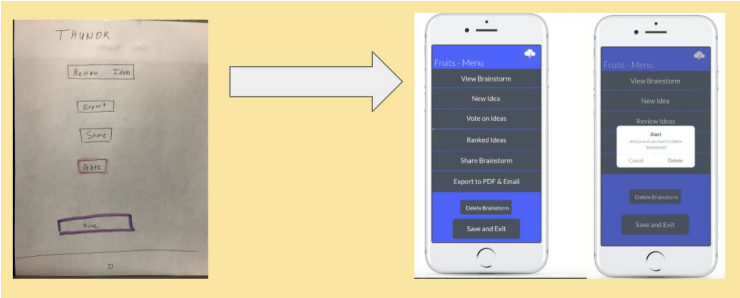


Figure 7. Med-Fi Share Brainstorm Task Flow

Rationale for changes

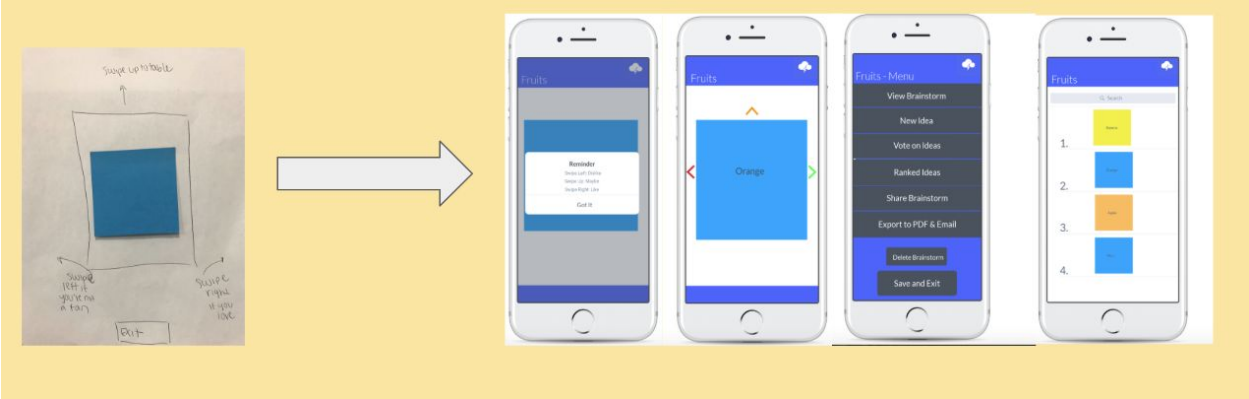
Based on feedback from our low-fi testers, we have adjusted our medium-fi prototype in a couple different ways:

1. Clarified Wording



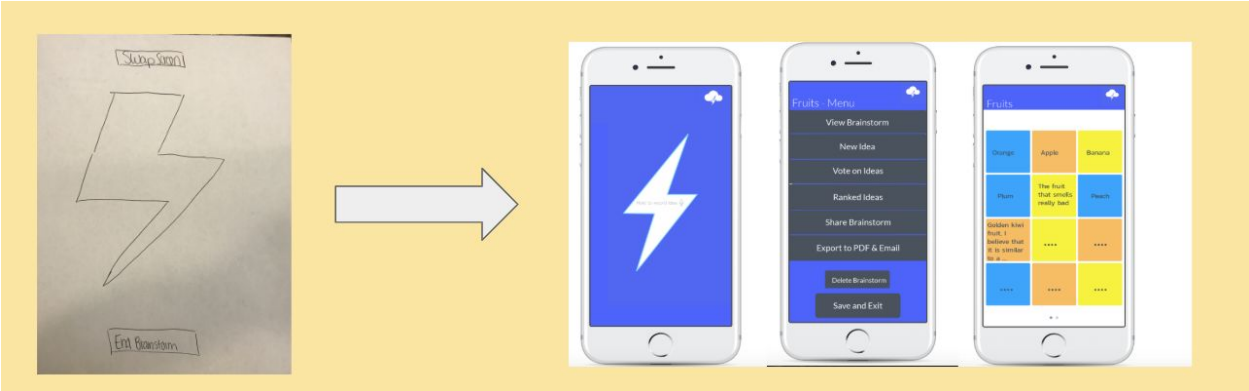
During user-testing, we discovered the participants were confused with certain word choices, such as “Delete” on the button on our paper prototype. We wanted to clarify the language so the users knew that clicking this button would delete the entire brainstorm.

2. Simplified ranking workflow



During user-testing, our participants were often confused with the purpose of “Vote” button. We first wanted to clarify that in order to vote on idea, they needed to swipe. Additionally, we added a “Rank of Ideas” menu, so users could see which ideas were ranked the highest after voting.

3. Improved screen change between idea submission and brainstorm access



From feedback from the instructors, we changed the way that a user could access the brainstorm during the session. ‘Swap Screen’ concept appeared unintuitive, so we made it part of the hamburger menu to ‘View Brainstorm’ instead.

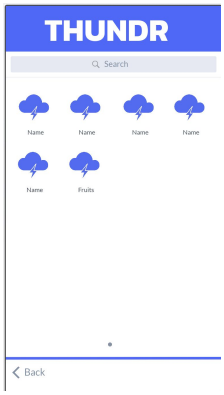
Hi-Fi #1

Upon testing the medium-fi prototype, we had a profound realization that workflow and navigation sequence that we envisioned were far too complex. Therefore, we focused on drastically reducing the number of screens and options in our mobile app, as well as adding the visual cues on what the user was expected to do.

In particular, we have deleted the notion of manually adding users to a brainstorm, and killed idea sharing with particular users. Our new assumption was that an active brainstorming session exists that everyone in a room is a part of. We have also deferred the non-essential features of exporting the brainstorm to different file formats, and viewing the ranked ideas privately on the mobile (as opposed to shared web screen).

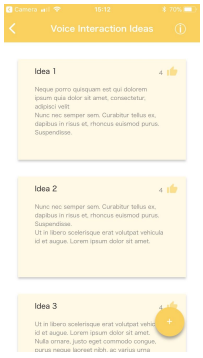
Rationale for changes
Based on the feedback from our med-fi testers, we have adjusted our hi-fi prototype in the following ways:

1. Search bar



In our medium-fi prototype the “Homepage” and “Ranked Ideas” pages both had a search function which seemed to be ambiguous in purpose. We addressed this by cueing for search with pre-filled sample searches.

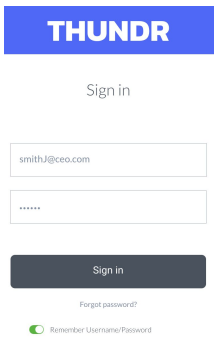
2. Idea Voting

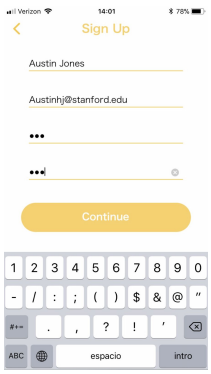


In our original tinder-like voting system there was a need for the initial prompt that tells users what the different arrows mean. There was also no guarantee that users will actually read this prompt or that they will remember what each arrow correlates to. Further, our med-fi app depended on colors for voting, which didn’t cater for color deficient people well.

We addressed these issues by changing the voting system to use a simple “thumbs up” button.

3. Sign-up/Sign-in





Removal of explicit ‘user adding’ step in the brainstorming shifted our design towards a more conventional sign-up/sign-in workflow.

With that change we were forced to think about what happens if the user does not have an account yet. We addressed this problem by creating back arrows and links between the sign-in and sign-up pages.

4. Recording the Idea

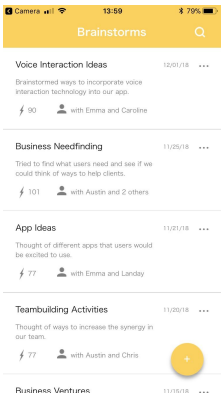
When voice-recording a new idea, the user is expected to click for start/stop. However, in the med-fi there was no visual indicator showing what was going on, i.e. whether the app is recording or not. We addressed this issue by adding a visual effect to show there is sound recording in process.

Our response to heuristic violations

In addition to test-inspired tweaks, we have made adjustments based on the group heuristic assessment of our app.

1. We rearranged the structure of mobile app, adding recent brainstorms to home screen. This change was intended to create a shortcut to review past sessions.

2. We added a floating button for creating a new brainstorm.



This made the main task of the app easily visible and accessible, making this part of the interface similar to IOS notes app.

We planned to propagate this style across the interface, taking more cues from Apple IOS Style guide.

Visual style changes

Our Hi-Fi #1 saw color scheme changed from the ‘facebook blue’ to ‘crayon yellow’. This change was purely subjective but strangely satisfying.

Limitations of our Hi-Fi #1

1. When designing the app, we had an early vision that Thundr should consist of the two components: mobile (individual) and web (collective screen sharing). However, we also deemed the mobile part to be more complex and important, so we did not invest much time in creating a web app. In our final testing and demo in CS147, the web part was driven by the ‘Wizard of Oz’.

2. We did not have the technology to test the actual voice input when submitting a note, so this part of the functionality did not work.

3. The logic of brainstorm session creation and joining was not fully flushed out, so we just assumed there is some way to tie the web and mobile frontends with some TBD backend.

Hi-Fi #2:

Transition from CS147 to CS194h gave us time to think through some major app changes, as well as to go through the additional user testing.

Most importantly, this afforded us two major insights.

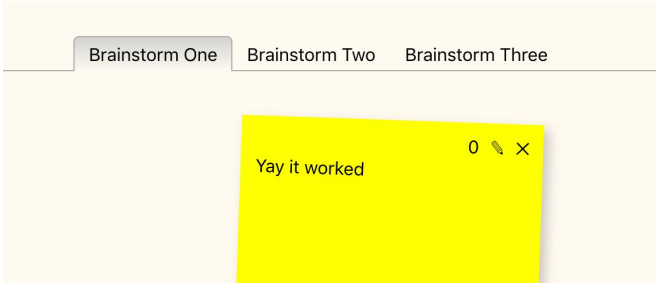
First, we realized that the brainstorm session itself was the central thing that tied the entire workflow together. This meant starting a new brainstorm and joining it could be made as simple as generating a unique code and sharing it across the participants, thus replacing the entire log-in + user adding routine. This same insight also allowed for removal of steps to archive and access past brainstorms – for as long as access code remained embedded in the web url, past brainstorms could be made accessible by regular webpage bookmarking in the browser.

Second, it became clear that to encourage close collaboration, we want the brainstorm to be managed by a facilitator – a user in charge of the web app. This allowed for further simplification of our mobile component because we could now design mobile app as a slave client, where mode of operation (idea submission or voting) are set by the facilitator for all participants at once.

Rationale for changes
Based on the above, we have adjusted our Hi-Fi prototype #2 in the following ways:

1. Feature realignment

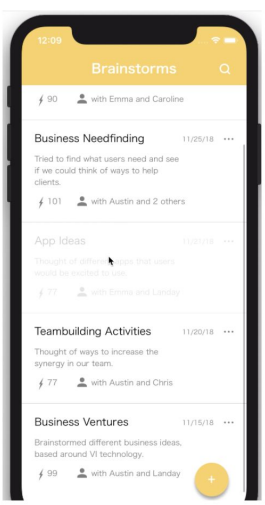
   In our Hi-Fi#1 we have kept functions like ‘search’ and ‘session wipe’. Upon realizing that a typical session involves only a handful of ideas, we dropped the search altogether. We also realized session wipe can be a part of extended business plan catering to enterprises, and is not crucial for our initial app release.

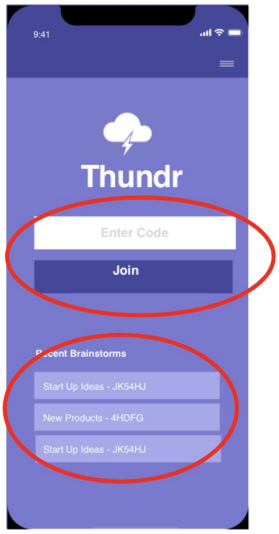


   On the other hand, one repeating feedback was that a brainstorm session may involve different topics (channels), so we decided to add this feature into the app.

2. Streamlined join process

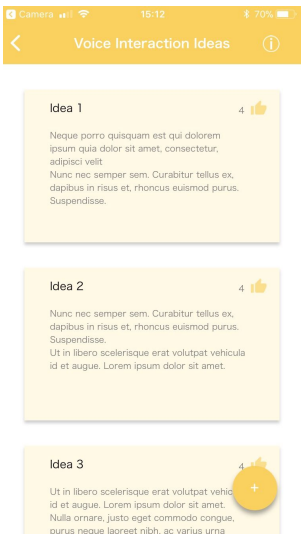
   Per the first major insight mentioned above, we replaced the in-app brainstorm creation by joining common session via code-based join screen.





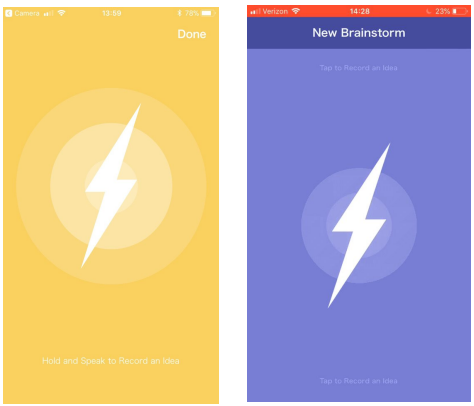
3. Slave screen switching in mobile

   Per the second major insight from above, our mobile app became effectively reduced to just two screens (joining and operational). The operational screen now switched between submission and voting modes according to actions of the facilitator, thus freeing a participants from every task unrelated to ideation.



4. Voice/Idea submission

   In our first take on the voice/NLP recognition, we implemented dictation using IBM Watson voice-to-text API. Since we had to deal with a remote server, this decision impacted our UI: now the user now needed to record her idea first, and then send it for processing.



5. Look-and-feel of the web app

   Changes to workflow cascaded down to our web app. Now the web app needed tabs for channels, and a code all participants needed in order to join (Figure 9)

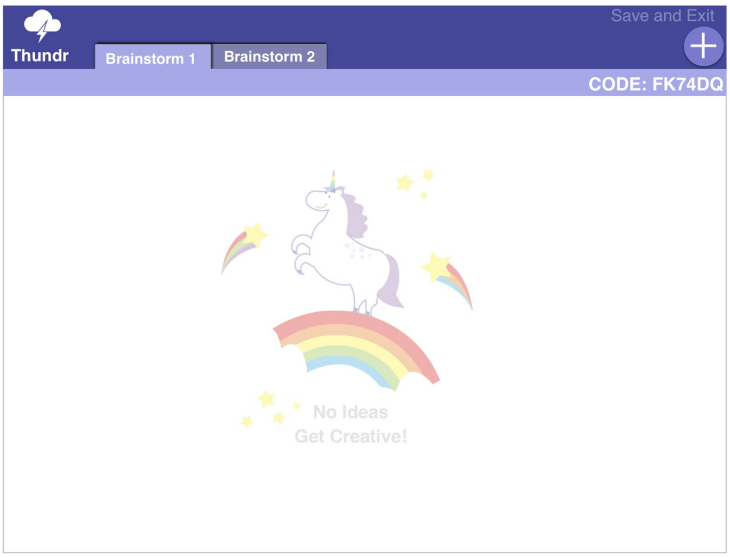


Figure 9. Workflow changes reflected in the web app

Visual style changes
Our Hi-Fi #1 had a ‘crayon yellow’ scheme, but the Hi-Fi #2 switched to ‘midnight purple’. This change was again purely subjective but was well met by the audience.

Market fit considerations for Hi-Fi #2
The market-fit planning task we performed on our prototype using Business Canvas helped us to understand our audience and realign it with feature set. In particular, we have realized that the widest market for Thundr at launch would be in non-commercial applications, where the enterprise-grade security is not required. This allowed us to limit session authentication to unique code (session ID) chosen from a large hash space, and defer session wipe feature.

We also realized that to sell Thundr as a service into the enterprise environment we must tighten up security, performance and access permissions, but it should not be our immediate priority at launch.

Final Interface

In designing our final UI we focused on streamlining the existing workflow, and synchronizing the look-and-feel across the web and the mobile parts.

With that, we delivered two major improvements: QR codes for fast session joining, and a move from the cloud-based speech recognition to on-device NLP.

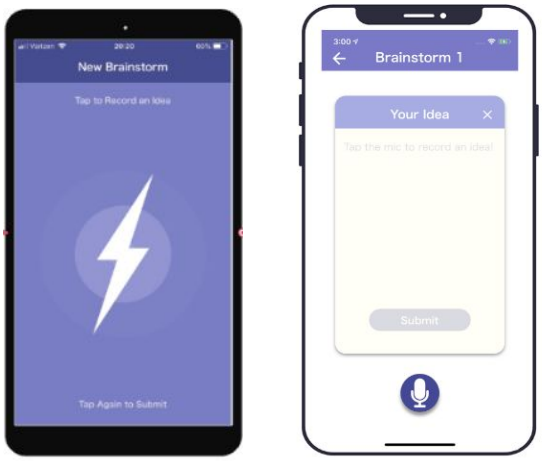
Rationale for changes

1. QR code join

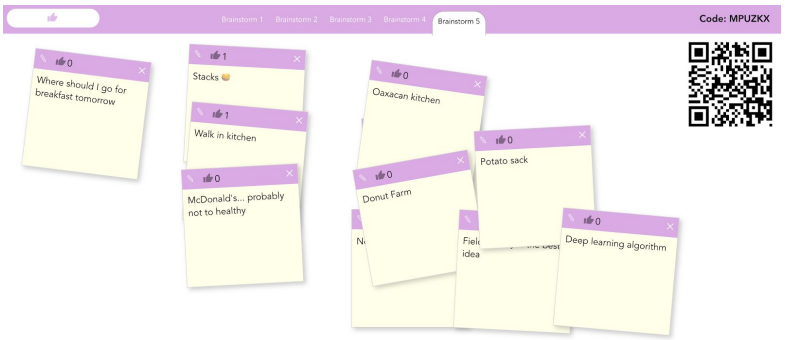
While code-based session joining introduced in Hi-Fi #2 was faster than conventional authentication, typing a random code into the mobile app was still cumbersome. To address this issue, we added a QR code into the web screen, which could be scanned on the phone (in-app or by a system camera) to join the brainstorm in one click.

2. Real-time voice recognition

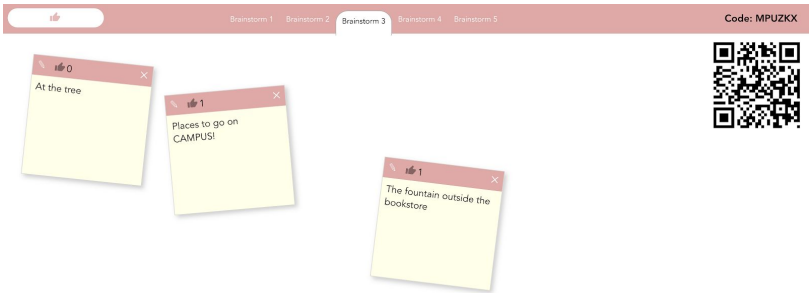
We had an early feedback on our voice NLP feature that speaking into the app and waiting for response was cumbersome and error-prone. In our final iteration we ditched cloud voice recognition in favor of on-device Apple voice-to-text API. This caused some changes to submission screen as per below:

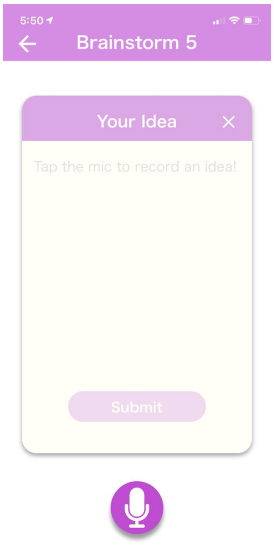


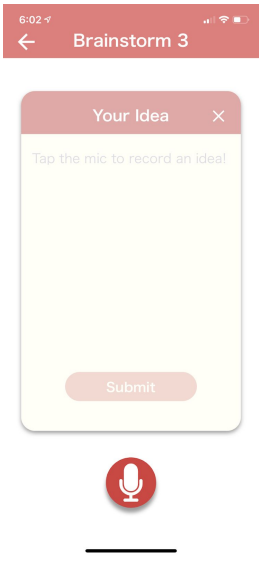
3. Color channel coding



User feedback on our ‘channels’ feature was positive, but some testers expressed confusion at what channel they are currently on. To prevent this issue, we synchronized the entire screen color scheme between mobile and web apps based on the active channel number.

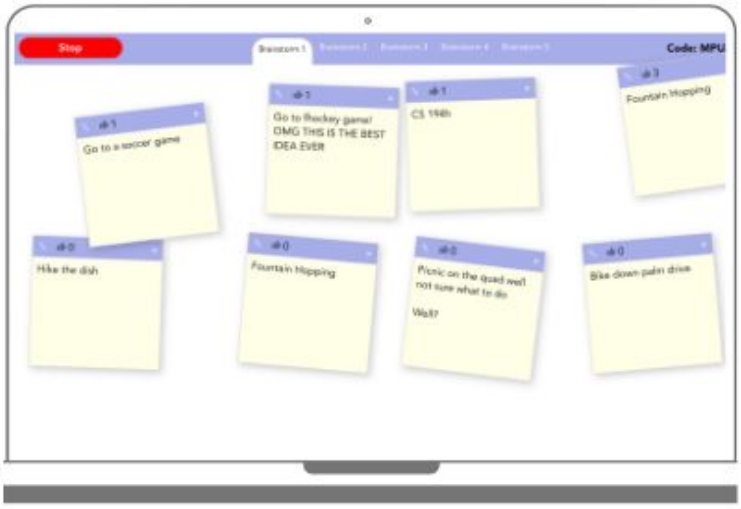


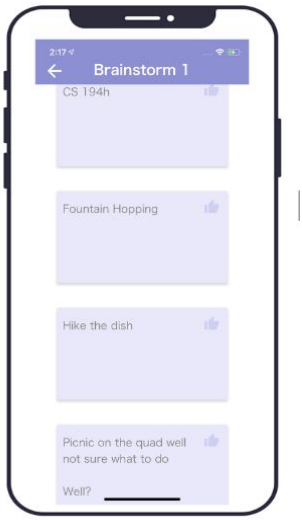




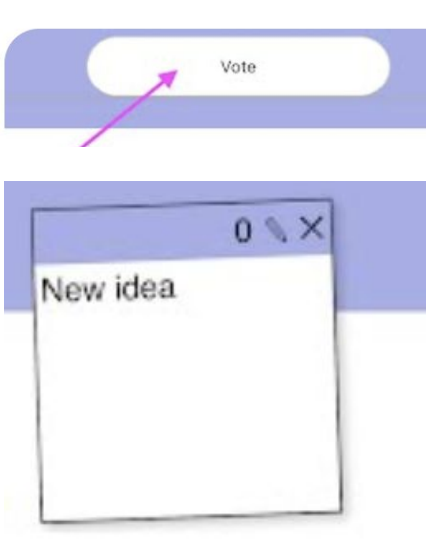
4. Voting mode color accent

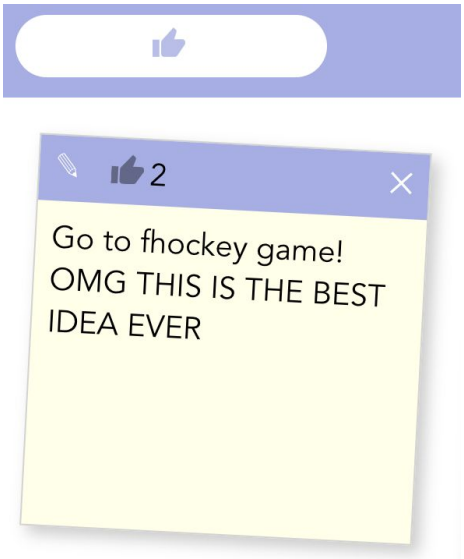
   In our Hi-Fi #2 testing, we found that sometimes users forgot to switch web app out of voting, and were confused about why they cannot submit new ideas. To prevent this, we chose to indicate the voting mode by use of a dissenting color accent (red) to highlight transiency of voting state.





5. Icon consistency and controls layout in web app

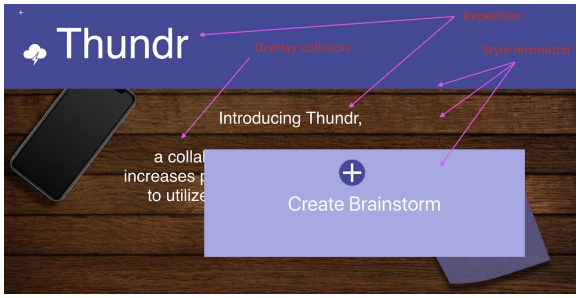


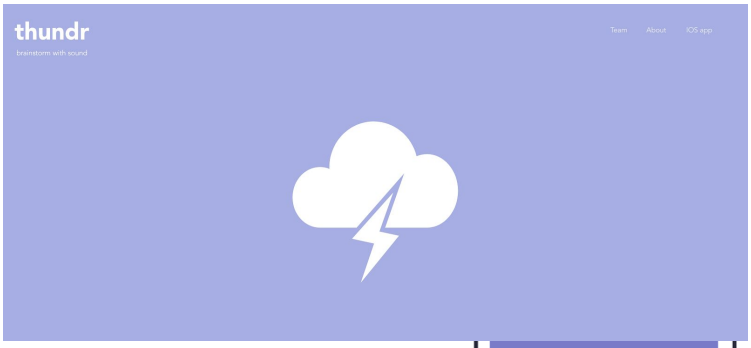


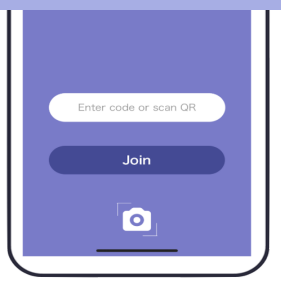
In feedback we collected at Hi-Fi#2 demo, we got the advice to place ‘delete’ and ‘edit’ buttons on notes further away to prevent accidental mistakes. We addressed that by

moving those control elements to the opposite sides of the header, and color-coding them white to differentiate from passive (informational) icons. We have also switched away from using the word ‘Vote’ on the mode button in favor of using the ‘like’ icon in multiple places across web and mobile for better symbol consistency.

6. Web app homescreen







In our Hi-Fi #2 design, the webapp homescreen was already present, but only served a singular function of starting a new brainstorm. In our final UI, we aligned the look and feel of web homepage with a mobile app, and added some information about our team and the mobile app download.

Final functionality review

Web App:

1. **Starting a new brainstorm session**
   A facilitator can start a new brainstorm session by going to the homepage at https://thundrweb.herokuapp.com/ and clicking the thunder logo. This creates a session all participants can now join using QR picture or code, and submit ideas.

2. **Organizing ideas**
   Facilitator can organize notes on-screen by dragging them around, editing or deleting them using note header controls. This simulates the whiteboard treatment of the physical post-it notes.

3. **Voting on ideas**
   Facilitator can press the ‘thumb’ button in the left upper corner to switch into voting mode. This forces mobile clients into voting and displays all current ideas on mobile screens. The web app displays tallied votes for every note.

4. **Channel switching**
   Whenever a new brainstorming topic is needed, facilitator can select one of five channels in web app. This switches context for web canvas and mobile clients.

5. **Revisiting past brainstorms**
   Brainstorm storage is persistent and available via deep-linked url in the web app. To revisit the past brainstorm, one needs to bookmark and reopen the link.

Mobile App:

1. **Joining a new brainstorm session**
   Assuming the Thundr app is preinstalled and a session was created by the facilitator, brainstorm participant scans QR code from web screen via his mobile camera, or opens the Thundr app and scans the code from there. Either action takes the client into an active brainstorm session.

2. **Submitting ideas**
   Brainstorm user can press the ‘mike’ button once and start speaking into the phone. Her speech is recognized and shown in real time. If the note reflects the idea well, she can press ‘submit’ and publish idea to web canvas right away.

   Alternatively, she can stop voice recognition by pressing the ‘mike’ button again, edit the note, dictate more content, and submit when ready. A new blank note is immediately created and made ready to submit the next idea.

3. **Voting on ideas**
   If facilitator chooses to start voting, the mobile app screen changes into idea list. A ‘thumb’ button on the right can be pressed or depressed to cast a vote. Vote submissions are shown in real time on the web app.

Left unimplemented

Thundr platform is fully working with no wizard of oz, and no planned features dropped.

Tools:
We created Thundr using React Native for mobile app, React JS for the web, and Firebase for the backend. Voice recognition was delivered with Apple IOS voice API.

React in both forms (JS and native) was pivotal for development due to its ease of use and cross-platform nature that permits development in pure Javascript.
Firebase (in flavor of legacy Firebase Database) proved to be an extremely easy to use tool that allowed us to organize the backend without configuring mongoDB and node.js, or adding any special code to keep the state synchronized between mobile and web.

We used free tiers of Google Firebase and Heroku to host our backend and frontend respectively. These tiers are not very fast, and the web app might be slow to start. Such problems are easy to solve with upgrading to paid tiers, so we did not focus on them.

The biggest issue we faced in development was our initial approach to React Native development that relied on Expo, which at the moment does not support the native IOS voice API. Expo is extremely easy to use and is well documented, but we had trouble working around its limitations. Although we were finally able to push some Expo boundaries (and employ remote voice processing with IBM Watson API), the end result was not very satisfactory. Eventually, we ditched Expo and finished the project in pure React Native, which worked surprisingly well.

Another dropped technology was Angular web framework we used in our first take on the web app. The reason for this choice was that Angular is part of CS142 syllabus and is familiar to many students. However, Angular is less expressive than React JS and stood in the way of continuity between the mobile and web source code, so we dropped it.

Finally, we faced one more issue that is not the part of the code toolchain. Unfortunately, Apple student dev program does not allow for App Store deployment, so our binary builds for IOS can only be distributed across devices registered in CS194h dev team.

Moving forward

Thundr project has a number of ‘extended reach’ goals that were not targeted at launch:

1. Android client. Android app is critical for brainstorming in the wide audience. Although it possible to generate Android app in React Native, it requires coding Android-specific voice API, and extensive testing. Since no one on our team had Android phone, we did not make this goal a priority.

2. Enterprise features. Enterprise-grade brainstorming is expected to have features like token-based access, encrypted backend storage, high-performance web and database cloud service and so on. These features were not prioritized.
3. QR-to-install. The current QR function joins a session if Thundr app is already on the phone. It is possible to redesign it in such way that QR would lead into App Store if the app is missing. Since we could not build for the App Store using student account, this feature was not targeted.
4. Limited vote counts. One feature suggested by judges at Launch Party was to limit the number of votes per brainstorm participant. We did not plan for this feature.
5. Submission of images. Another late request that we did not plan for.

Download:

The web app can be accessed by going to https://thundrweb.herokuapp.com/.
Thundr app can be installed from https://web.stanford.edu/~austinhj[2]

Makers

Our team consists of *Emma Alderton*, *Austin Jones*, *Caroline Willis*, and *Daniel Kharitonov*. Both Emma and Caroline are undergraduate Computer Science majors concentrating in Human-Computer Interaction (HCI). Austin is a co-term student in Computer Science studying HCI, and Daniel is a Ph.D student in the Management Science and Engineering department and MS. student in Computer Science in AI. The team has a mixture of app development, web development, and design skills that allowed us to create a working, easy-to-use prototype of our app that is ready for use on the App Store.

Business model

As implemented, Thundr acts as a fully functional free app, suitable for any personal, non-commercial or educational purposes. It has no explicit restrictions on scale, features, or frequency of use. Building a business out of free app is somewhat challenging, but not impossible. We think the premium version of Thundr could target businesses and include security measures that would block outsiders from reaching and accessing ideas from a brainstorm that a premium user had created. This would provide peace of mind for commercial users concerned about sensitive intellectual property. We also think that reaching into enterprise market could be made easier by partnering, so integration with existing platforms such as Slack or Salesforce can be attractive.

[2] Due to limits on Apple student developer program, the app can only be installed to iPhones whitelisted for the use of the CS194h development team.

Summary

Brainstorming can be difficult.

Thundr takes on this challenge with modern technology and intends to streamline the brainstorming process. The simplicity of our design makes it easy for users to make the jump from paper-based brainstorming to electronic format and benefit from voice input, automatic storage and anonymous voting.

We hope that our project can help to create effective and creative brainstorms, resulting in fewer “lost” ideas, active participation, and guilt-free idea ranking.