

Speak↑: Hi-Fi Prototype

“Better lectures, powered by real-time student data”

Team

Name	Email	Role
Brad Reyes	breyes28@	Developer
Karen Gomez	kgomez@	User Testing
Nick Akiona	nakiona@	Design
Reid Watson	rawatson@	Manager and Documentation

Problem and Solution

Giving an engaging, interesting, and effective lecture to more than 50 students isn't easy. Lectures occur infrequently, and most instructors don't get actionable feedback when students are confused. This problem can be equally frustrating for students, who become bored when they feel confused by lecture content.

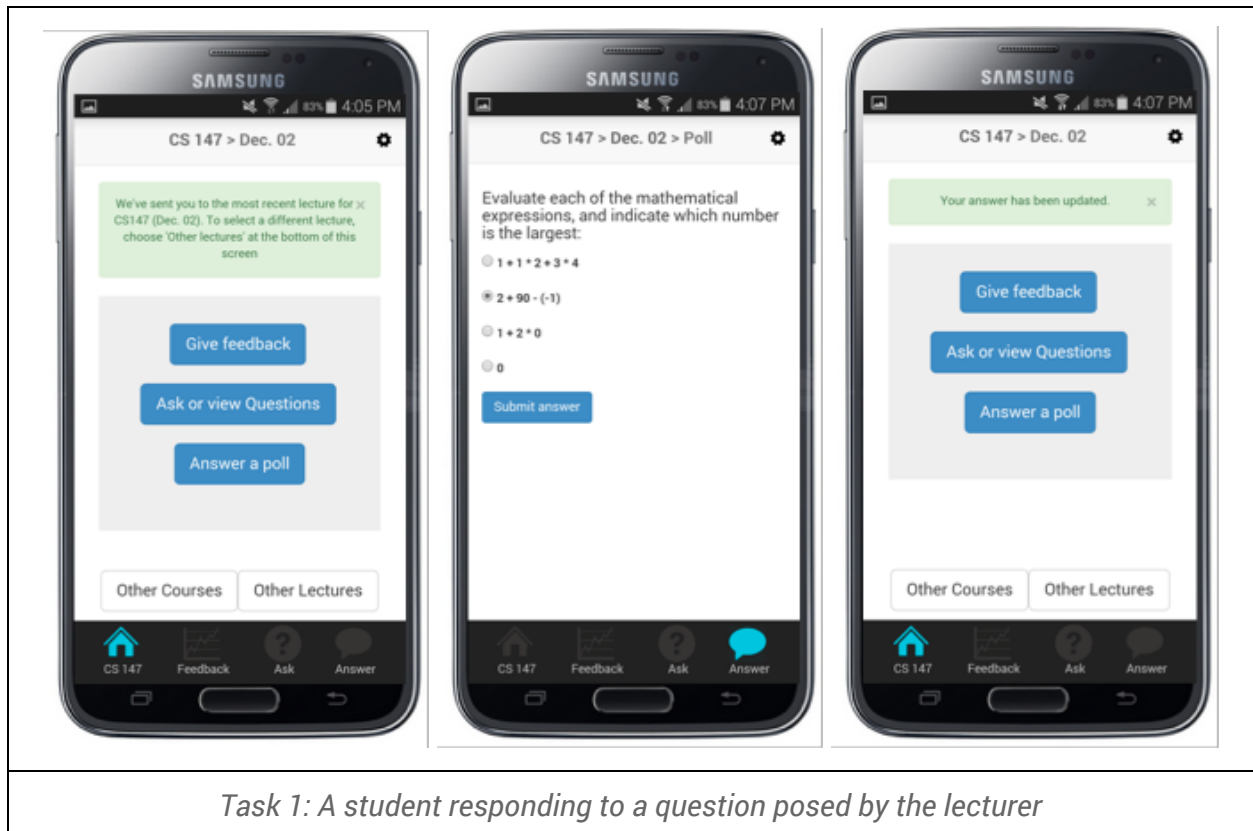
SpeakUp aims to provide real time data about student understanding to lecturers, and offer concrete ways for lecturers to keep students interested. SpeakUp allows students to easily indicate their confusion when watching lectures, respond to “clicker questions” in real time, and helps instructors improve their course content with real time feedback on engagement and clicker questions.



Tasks

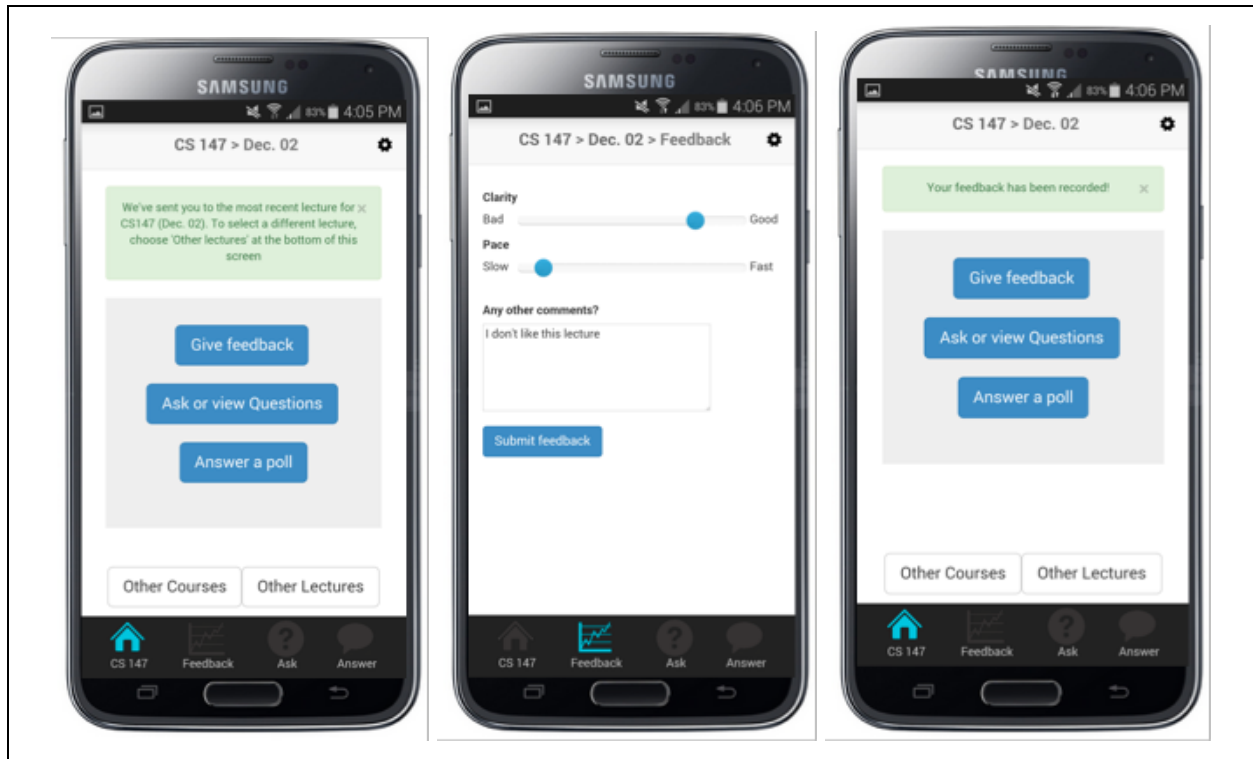
Simple Task: *Respond to a question posed by the lecturer*

This task allows the lecturer to get immediate feedback on the students' understanding by easily seeing how many students correctly answer a question posed during class. Many professors use iClicker questions to test the students' understanding, and we felt it would be easy to integrate this functionality in our application.



Medium Task: *Provide feedback on a lecture*

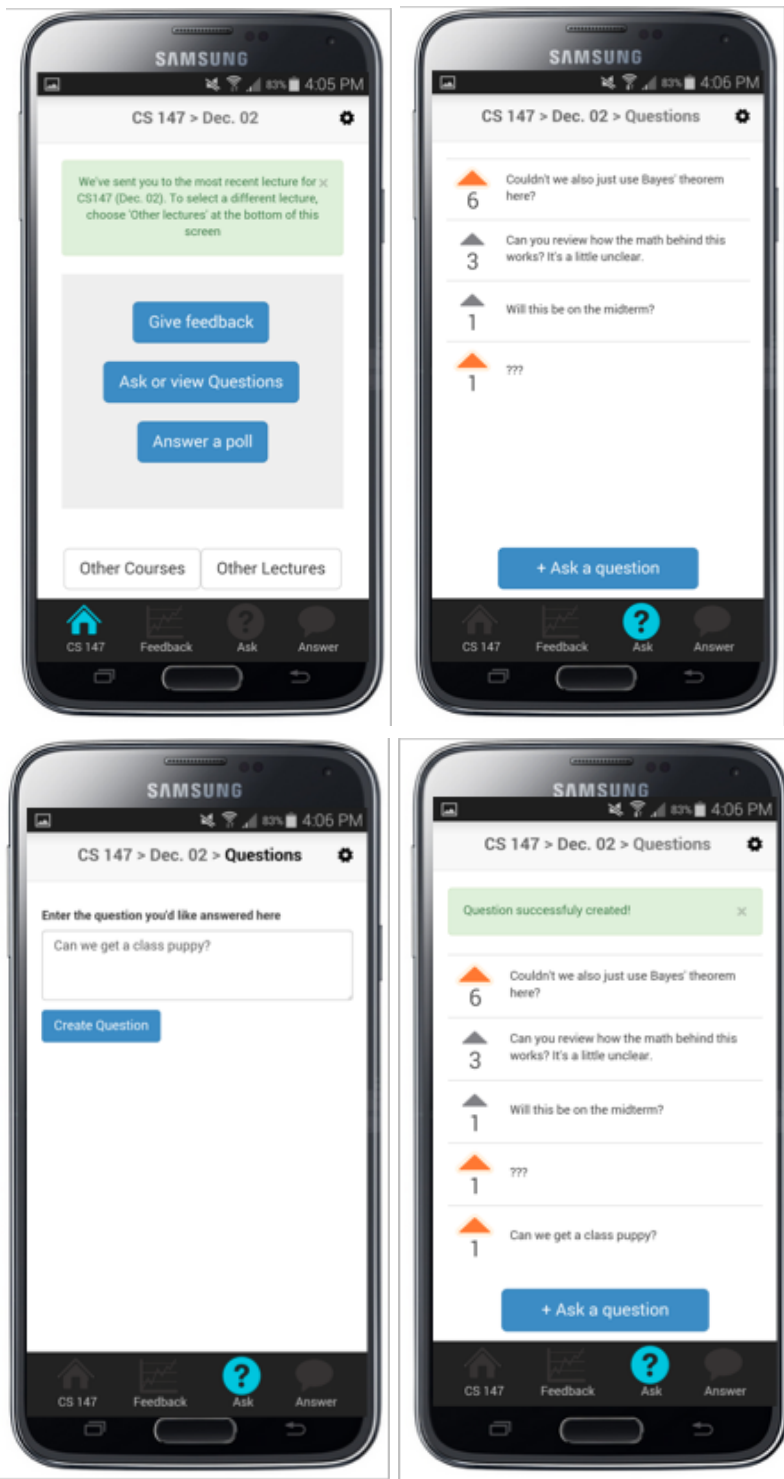
Although there exist ways for students to give lecturers feedback, many of these methods happen after a class has finished, such as course evaluations that take place at the end of the quarter. Our primary goal with the creation of this application, was to provide an easy way for students to give lecturers feedback on the material being presented in a timely manner, so that this feedback can be incorporated into future lectures. This task allows students to give the lecturer feedback on the pace and clarity of the lecture during class, so that this information can be used to improve lecture and, in turn, the learning experience for the student.



Task 2: A student giving the lecturer feedback

Complex Task: *Ask the lecturer a question*

This task allows students to pose a question to the lecturer during class and also gives other students the ability to vote on questions that they would like answered. Sites like Piazza allow students to ask questions after lecture has occurred, so we wanted to create a way for students to ask the lecturers questions as the material is being presented in class.



Task 3: A student asks the lecturer a question

Major Usability Problems Addressed

Heuristic Evaluation from Mid-Fi Prototype:

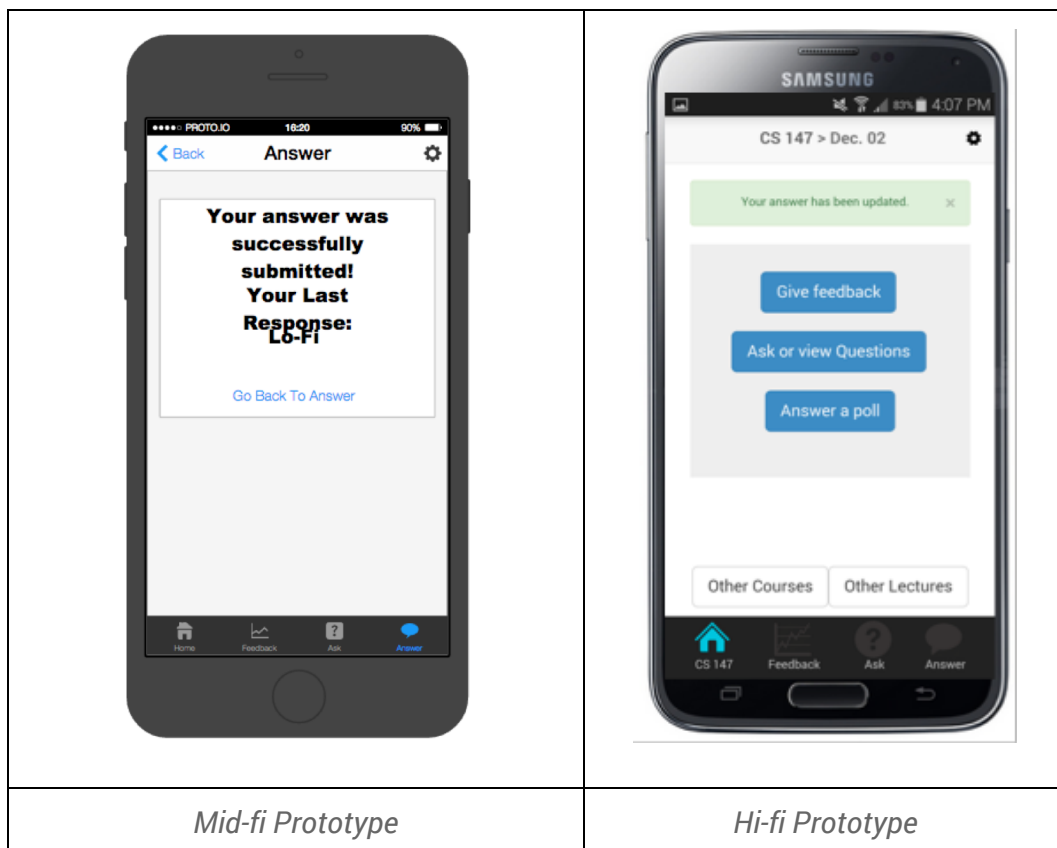
- Violation: [H2-3. User control and freedom] [Severity 3]
Description: “There is no option to remove a class once it is added. Add a ‘Remove’ button.”
Changes: No visible changes were implemented between the mid-fi and hi-fi prototypes. Removing classes already occurs in two ways. Firstly, a student can go into his or her setting and remove the class from the class list. Since students will rarely remove classes, we decided that there should not be a button dedicated to removing classes on the home screen. Secondly, students can set their classes to be removed automatically a few weeks after the class has ended. This feature is also within settings. These features are not visible since the settings screen has not been implemented for the hi-fi prototype.
- Violation: [H2-1 Visibility of system status] [Severity 3]
Description: “After the user clicks the plus button on the add class page, they are taken back to the home screen, which now contains the newly added class. Although the user should be able to verify that the class was successfully added by looking at the list of courses and checking for presence of the newly added class, there should also be some dialogue that lets the user know that they successfully added the class. The system should implement brief feedback message on the home screen that tells the user ‘CS 147 has been successfully added to your class list.’ The interface should make it very obvious that the user has successfully added a class, either with dialogue and feedback messages or with significantly more noticeable changes to the page.”
Changes: We agree that a notification for a successfully added classes would be useful, and we have worked to make clear and attractive notifications for user actions. However, the “add class” page has not been implemented for the hi-fi prototype so this change will not be visible.
- Violation: [H2-3 User control and freedom] [Severity 3]
Description: “After the student clicks add a class, they are taken to a page that allows them to input the class information. However, there is nothing that lets the user know where they are in the app, such as an Add Class header at the top of the app. Although there is only one step between the homepage and the add class page, a header/title would still help to ensure that the user is always clear on exactly where they are in the app. This is especially important given that the homepage and the add class page are similar in layout and design, meaning that the user could possibly experience confusion navigating between the two pages.”

Changes: As stated earlier, the 'Add Class' page has not been implemented for the hi-fi prototype. Therefore, the updated header for this page will not be a visible change.

4. Violation: [H2-6. Recognition rather than recall] [Severity 3]
Description: "Selecting a class in progress in the home screen is confusing. Perhaps adding instructions 'Select current lecture' on top might be helpful."
Changes: In pursuit of a minimalist design, we have chosen not to add this help text for home screen. We saw no confusion or hesitation when testing this screen on usability test participants. Therefore, we disagree with this violation and have chosen not to implement any changes.
5. Violation: [H2-2. Match between system and the real world] [Severity 3]
Description: "The user does not need to select the current lecture. The app can get the schedule of the class and automatically display the options for the current lecture."
Changes: In the mid-fi prototype we chose not to implement this feature because we still wanted the student to be able to view older lectures and did not have an obvious method of changing lectures if we did not force the user to select it initially. However, in the hi-fi prototype we implemented a class specific screen the allows the user to easily change the lecture. This allowed us to automatically place the user into the most current lecture of the class he or she selected.



6. Violation: [H2-3. User control and freedom] [Severity 3]
Description: “It is impossible to skip a question. Sometimes a wrong answer is penalized and not being able to skip a question can harm the user’s grade. Add a ‘Skip’ button.”
Changes: A lecture can only have one active question at a time. When the lecturer closes a question or poses a new question, the old question will no longer be displayed. A user may simply refrain from answering a question, if desired. There is no need for a ‘Skip’ button.
7. Violation: [H2-3. User control and freedom] [Severity 3]
Description: “The student cannot re-answer a question. The user might accidentally click the wrong button or decide to change her answer. The app should allow multiple submissions while there is still time left to answer a question. Keep current question active until time is up.”
Changes: In our hi-fi prototype, we made the question remain active for as long as the professor leaves the question open. This gives students the ability to update their question until the professor closes the poll.



8. Violation: [H2-2] Match between system and the real world [Severity 3]
Description: “The feedback icon for the student is the same as that of the teacher (ie. the graph icon). Although it makes sense to have this as the icon for teachers (since the feedback page displays line graphs similar to the icon), this is a much less intuitive icon for the student feedback page. The sliders and text input on the feedback page do not correspond the line graph icon, which should be changed in favor of a symbol more directly related to the content of the page.”
Changes: We could not find a better icon for the feedback screen, so we ended up keeping icon the same, which is the graph icon. Because there is a label under each icon, the student will know how to navigate to the feedback page.

9. Violation: [H2-5. Error prevention] [Severity 4]
Description: “It is easy to not notice the ‘Answer’ section of the menu when the instructor has asked a question. Make ‘Answer’ section more noticeable when active or alert the user every time a question is asked.”
Changes: The intended use for this feature is that the lecturer poses a question during class and gives the students time to answer the question right away. For this reason, we felt it was unnecessary to add a notification when a question is posed. Moreover, since the application is for use when students are in lecture, adding a notification would alert students who were not in class to answer the question, which is not the intended usage for this feature.

Additional Changes:

1. Change: Class Landing Page
Description: The biggest change we made between the mid-fi and hi-fi prototypes was the addition of a class landing page. This page helps the user navigate through our three tasks and provides clear ways to change the current lecture or class. The user will reach this page after selecting a class/lecture or after submitting feedback or answering a question. Notifications on this page provided the user feedback on his or her previous actions.
Reasoning: We created this landing page for two reasons. First, in our mid-fi prototype we had several separate notification screens. We wanted to unify these screens and allow users to easily navigate to all other parts of the app from this single screen. Second, we needed a clean way for a user to change lectures in a manner that fixed violation five in the heuristic evaluation.



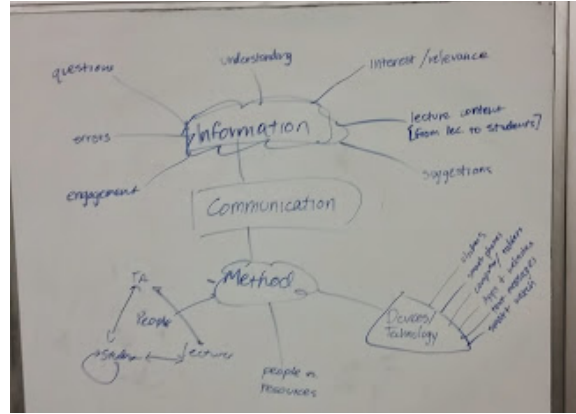
Once a class is chosen, the user is taken to a page from which any task can be completed.

Design Evolution

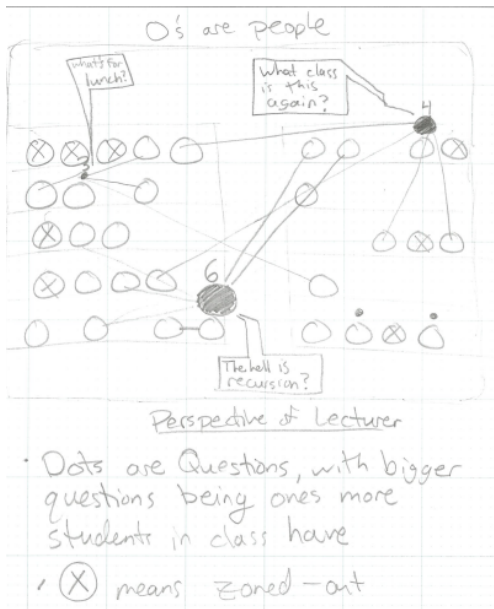
Our design was first conceived after doing our interviews with potential customers of our product. By implementing the master-apprentice model, we were able to analyze our customer's work practice. From that, we were able to perform a task analysis to try and pinpoint the important tasks our app would help our the user perform. After brainstorming ideas that lined up with our contextual inquiry results, we started sketching very raw UI designs for our future app. We narrowed down the three best project ideas from our multiple sketches and picked the one that had the most significance, feasibility, and interest within our group.



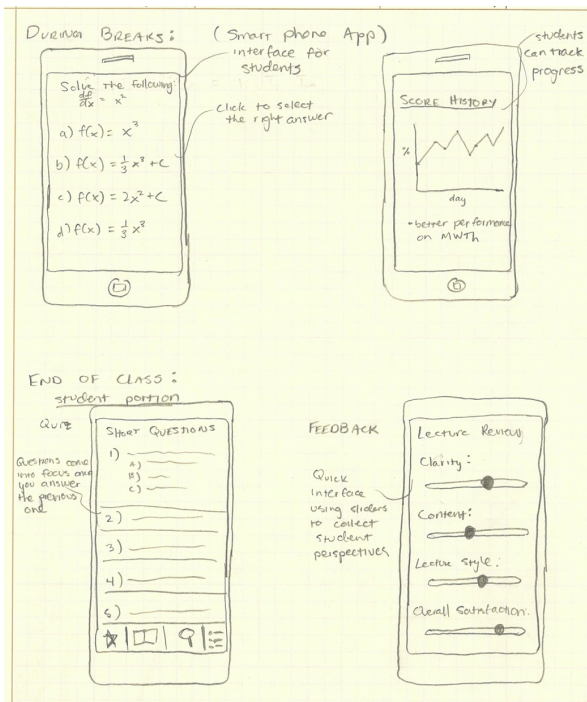
Using the master-apprentice model to investigate our customer's work practice. This shot was taken from where one of our participants was sitting during a class



Once we understood our customer's work practice, we had a brainstorming session to stir up ideas so we could make sketches for the UI.

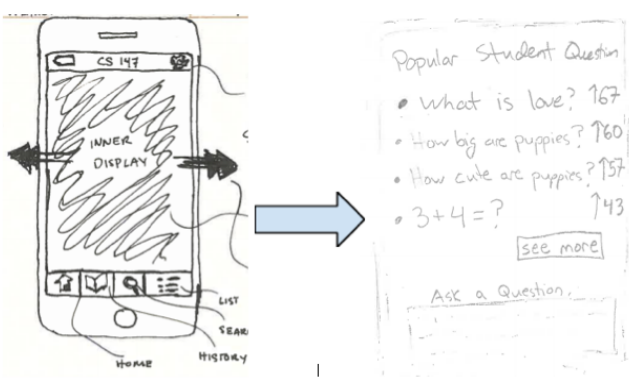
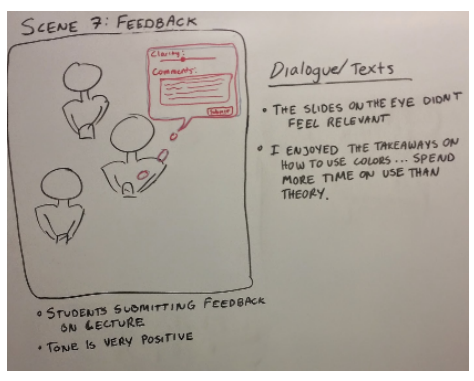


One of the three finalists for project ideas. It ultimately lost to the sketch on the right. It featured a Google Glass design that would allow lecturers to physically see popular questions above the students heads.



The sketch we decided to work off of. We picked it because of its significance, feasibility and general interest within our group. It actually has some of the tasks that are in the hi-fi prototype!

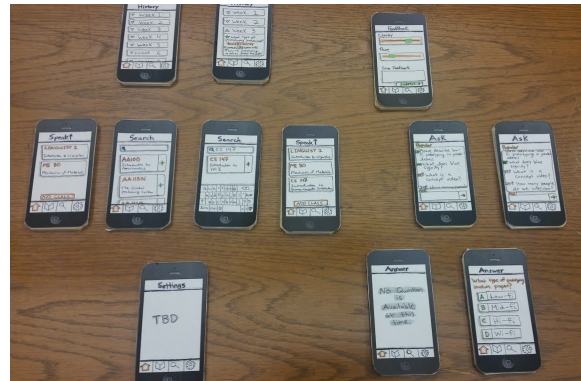
With that in mind, we created a concept video that helped us further define and improve our tasks by storyboarding them and creating scenarios that our customers will potentially have to perform. This storyboarding process allowed us to refine our last UI design choice by making the UI centered around making the three tasks accessible to our customer base. By fleshing out these scenarios in a concept video, we saw our scenarios and completion of tasks in action, thus making us confident in moving forward to user testing.

	
<p>Our first design that is centered around the three tasks we created from the contextual inquiry phase. The user must swipe to get to the screen regarding a specific task. Here is an example of what our UI storyboard looked like trying to answer a question.</p>	<p>We also storyboarded our scenarios for the concept video. By creating these scenarios, we were able to refine our tasks, which would play an important part in the rest of our design.</p>

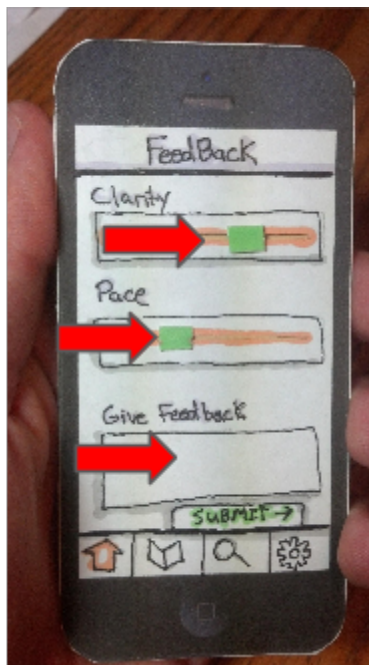
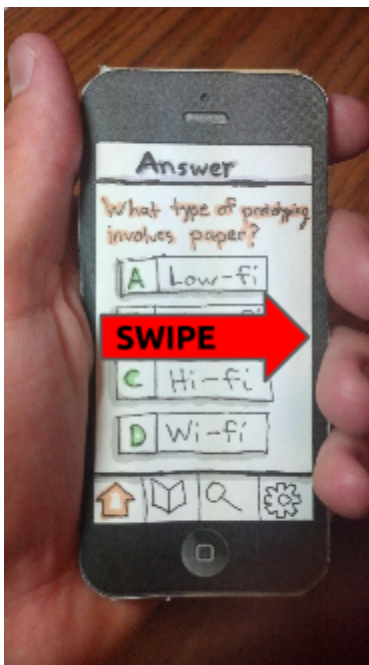
The next step was to create a lo-fi prototype and test it on users to see what is and isn't intuitive. For the most part, we kept the design we created at the end of the concept video project. After testing it on some college students (since our student UI will mostly be used by that demographic) and a lecturer (with our lecturer UI), we obtained important usability test results to incorporate into our next design phase. One of the most important results we obtained was the fact that users never found our swipe mechanic intuitive as all three students got stuck when they needed to swipe to get to the next screen. Another important one involved some ambiguity throughout the system as a whole, including confusion about the difference between the ask and feedback tasks and the fact that users would place the "Pace" slider in opposite directions when giving feedback, even though they were told to give feedback to a lecturer who spoke too fast.



One of the participants for the usability test. A group member would act as the “computer” while another would take notes on the situation.



The complete student UI. This has all the screens a user can link to, and some even are versions of one screen but scrolled, if a user ever did pick up on the scrollable screens.

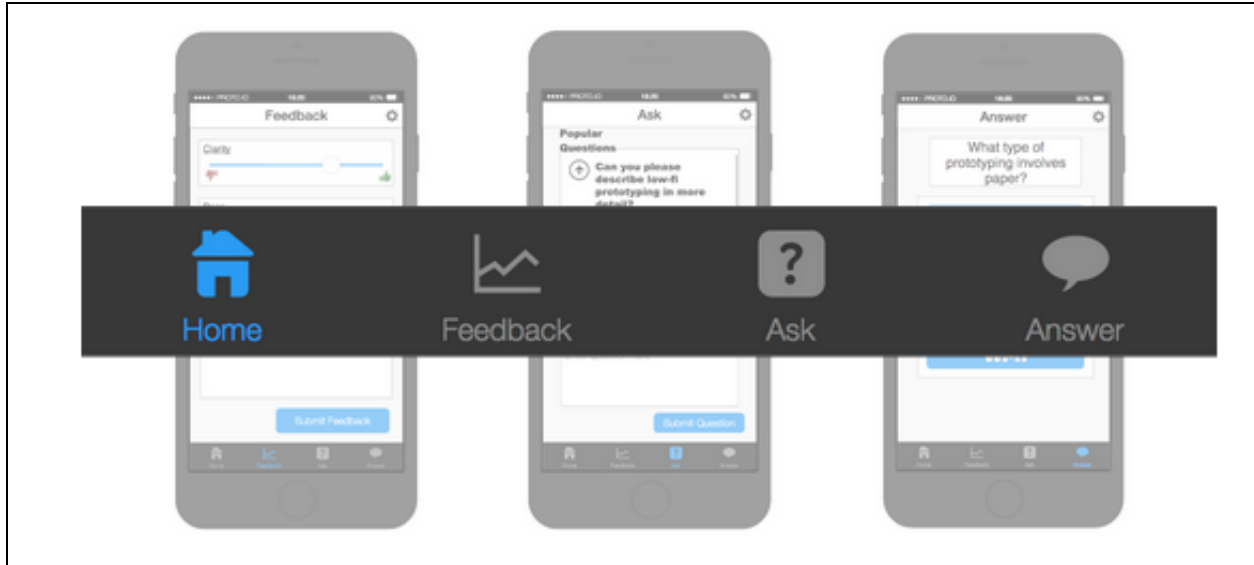


An example of a task a participant would have to complete. This one involves submitting feedback to the lecturer, but it also demonstrates one of the biggest problems with our lo-fi prototype design: the unintuitive swipe mechanic to access the three tasks.

After that phase, we moved onto the mid-fi prototype. Changes we made for that stemmed directly from our usability test results. We decided to change the unintuitive swipe mechanic and instead put a navbar at the bottom of the screens to make access to all three major tasks

very easy. We also fixed the problems with ambiguity since we made the description of pace much clearer and we clearly separated the tasks of ask and feedback with the inclusion of the navbar.

Without the limitations of a paper prototype, we were also able to solve the problem regarding the user's difficulty to see if an area was scrollable or not.



Our biggest change from the lo-fi prototype to the mid-fi, we got rid of the swipe mechanic and the navbar with rarely user icons and replaced them with a new navbar with a home button and the three major tasks our app does!



We also changed the ambiguity of the pace slider (putting "Too Fast" and "Too Slow" on the edges).

Completing one of the tasks (in this case it's answer). It's much more intuitive to click on the navbar to get to one of these tasks than the swipe we had before.

With the prototype we made, a group of evaluators performed a heuristic evaluation on our project, giving us a lot of insight about what we could possibly change for the hi-fi prototype. You can look more into this in the “Major Usability Problems Addressed” section earlier in the report, but some of the more important changes involved the home button, the auto selection of lecture, and updating answers in the Answer tab. Now, there is a class landing page, which fixed one of the violations described in that section. Additionally, the evaluation raised a great point about the ability to change answers when the poll is still open. For screenshots, full descriptions and changes from the mid-fi to hi-fi prototype, please visit the “Major Usability Problems Addressed” section.

Prototype Implementation

To implement our hi-fi prototype, we created a web app using ruby on rails for the backend and bootstrap for styling on the front end. By writing a web app in ruby on rails, we were able to develop the app faster than developing a native app. Additionally, ruby on rails made it very easy to implement a fully functional database for all of the course, student, and lecture information. Using bootstrap allowed us to implement a very aesthetically pleasing and minimalistic design that very closely matches our mid-fi prototype. Touch events were the only substantial drawback in using these tools. The main difference between our web app and a native app is that the feedback ‘sliders’ do not slide. Bootstrap’s sliders do not handle touch events so a user currently has to tap the bar to move the slider instead of using the sliding motion.

Our hi-fi prototype is a very close approximation of a fully implemented web app. Our database and backend are fully functional and all of the main features have been implemented. Our prototype only uses the Wizard of Oz technique when it comes to data that was supposedly created by the lecturer interface. The lecturer interface has not been implemented yet, so questions and lecture information had to be hardcoded into the database. As such, the same question from the lecturer is always visible and is never closed.

Additionally, all of the information used to fill the databases for demos had to be hardcoded. This includes the courses listed, lectures for each course, and questions already posed by students. Since this is a lot of data to fill in, not all of the possible paths have preexisting information. For example, the oldest lecture for CS147 does not have a question posed by the lecturer or questions already posed by students.

In our hi-fi prototype, we fully implemented our three primary student tasks. As such, there is not much development left for this interface beyond stylistic refinement and additional testing. The key pieces missing from the student interface are a functional ‘add classes’ button and a working ‘settings’ interface. Once we implement these features, the student interface will be complete and functional.

However, in order to create a complete product, we have to create the entire instructor interface. This step requires a substantial amount of coding, but the design work and the database have already been completed. From there, Speakup will be ready for full user testing and can be further developed by having a trial class use Speakup for an entire quarter.