electurefy

Jennifer Farman - Team Manager
Daniel Hok - Development & User Testing
Juan Posadas Castillo - Design & Development
Nikhita Obeegadoo - Documentation & User Testing

Value Proposition

Make lecture a more effective learning tool for both students and teachers.

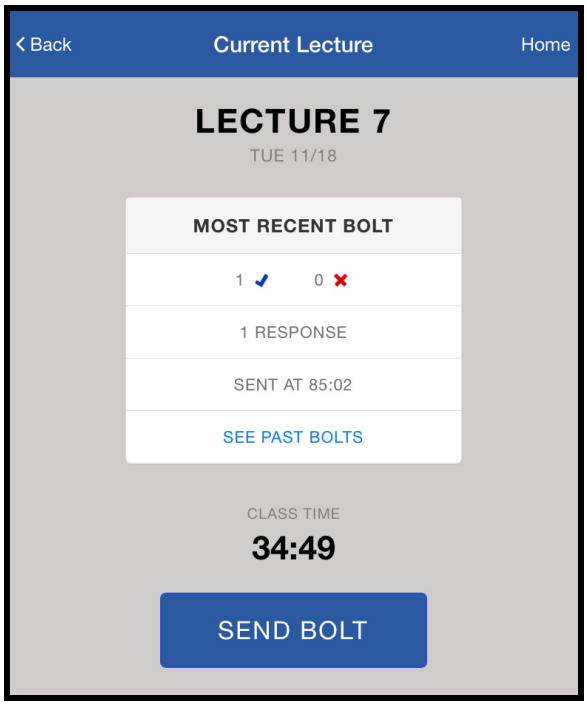
Problem and Solution Overview

Today's colleges have all it takes for success: dedicated lecturers and motivated students. However, in a lecture hall with a ratio of a hundred students to one instructor, it is hard - if not impossible - for students to effectively communicate their understanding levels.

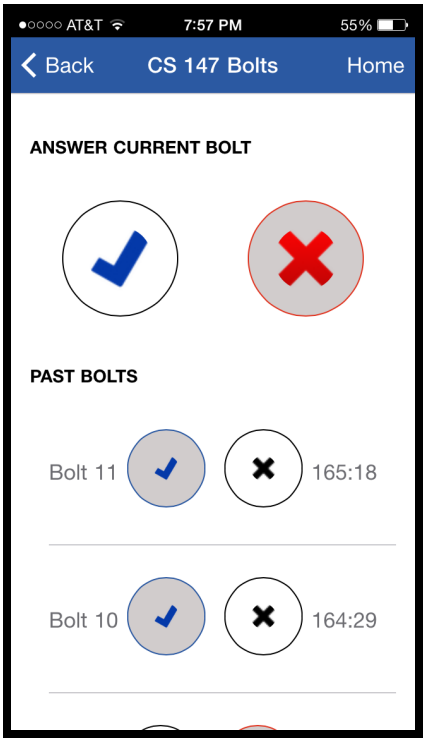
Electurefy's mission is to bridge the communication gap between students and instructors. By allowing instructors to prompt students for feedback at specific moments during lecture, Electurefy provides a simple, user-friendly and non-distracting way for students to indicate their understanding levels to the instructor, thereby making lecture a more effective learning tool and providing instructors with actionable data that allows them to adapt lecture to students needs.

The basic approach of our app is described and illustrated as follows:

1. Instructors can prompt students for feedback via "bolts".



2. Students can respond to “bolts” by indicating confusion or understanding.



Tasks & Final Interface Scenarios

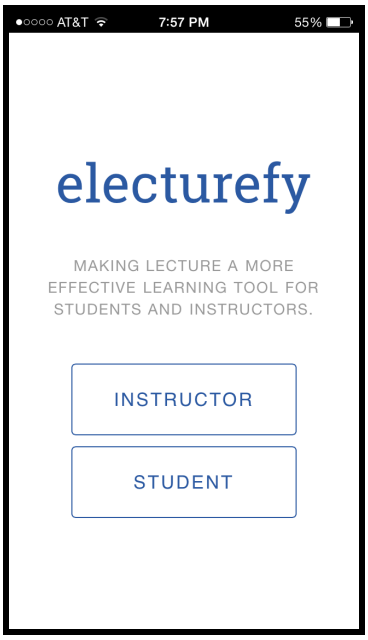
Our three tasks are as follows:

<u>Simple task : Monitoring student confusion in real time</u>
Task performed by Instructor

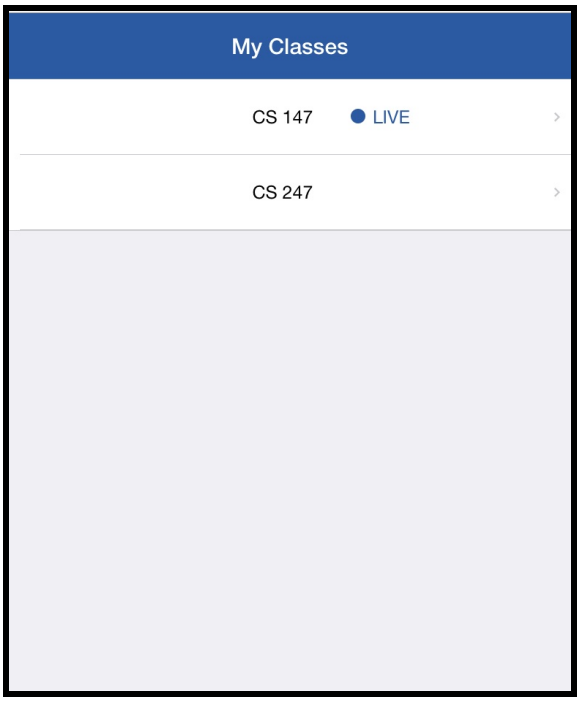
This task is performed by the instructor. During lecture, he/she sends out a “bolt”, which allows students to indicate either *confusion* (indicated by a red cross) or *understanding* (indicated by a green tick). Bolts can be sent to either all of the students in class, or a random subsection of the students (the instructor can switch between these options in “settings”). The instructor can see the student response data as the students reply to the bolts (e.g: 30 confused, 10 understood), and can also swipe between all the bolts that have been sent out during that lecture to see the response breakdowns for previous bolts.

Why was this task chosen?
This task was chosen because it captures, in its very essence, the purpose of the app, which is to allow the instructor greater insight into the student experience.

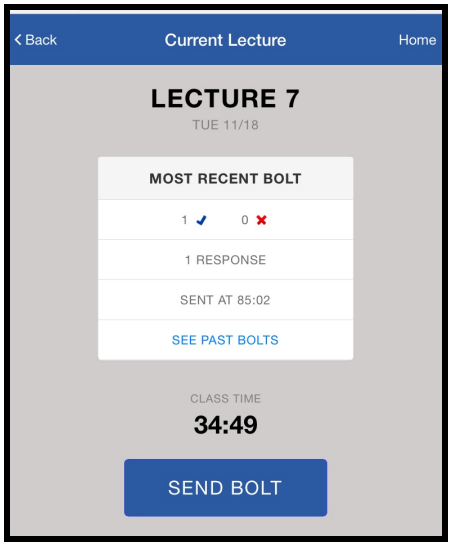
Storyboard of this task



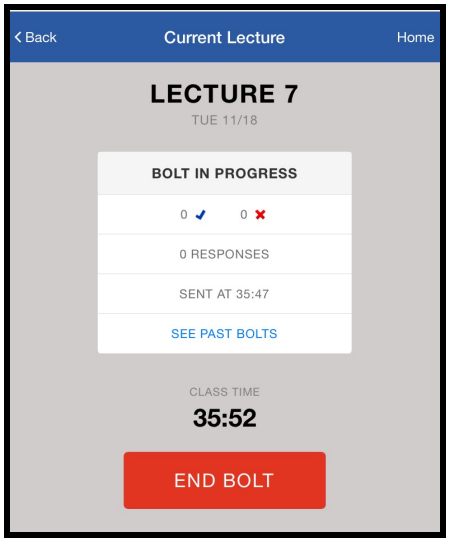
Step 1: The instructor opens the app and is direction to this login page.



Step 2: The instructor is directed to a “My Classes” page, from which he can choose the lecture which is currently live.



Step 3: The instructor can send a bolt, requesting student feedback.



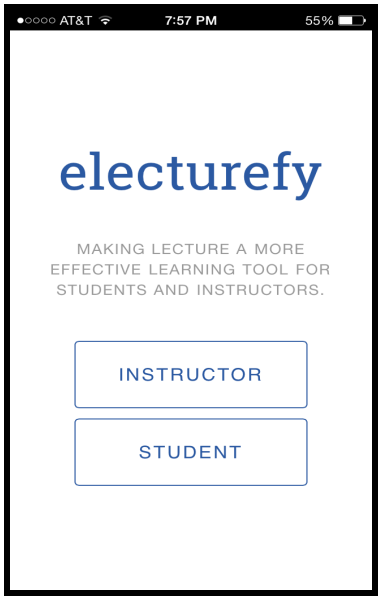
Step 4: The instructor can end the bolt, after which no more responses will be added.

<u>Moderate task:</u>
Submitting feedback (ie. responding to “bolts”)

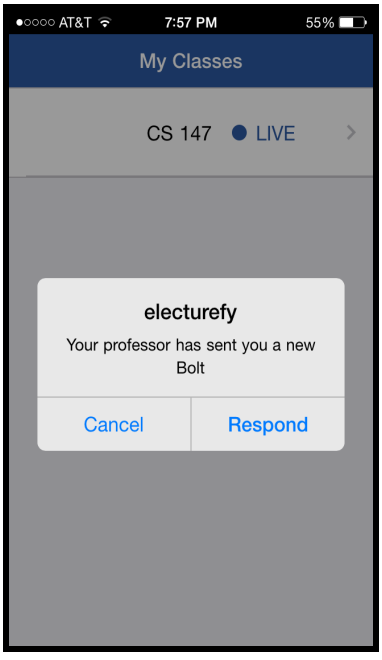
Task performed by Student
The student responds to “bolts,” sent out by the instructor, to indicate whether they are experiencing confusion or understanding at that particular point in lecture. This data is immediately sent back to the instructor, who can then adapt to the student feedback.

Why was this task chosen?
This task shows the other side of the essential task of allowing instructors a greater insight into the student experience.

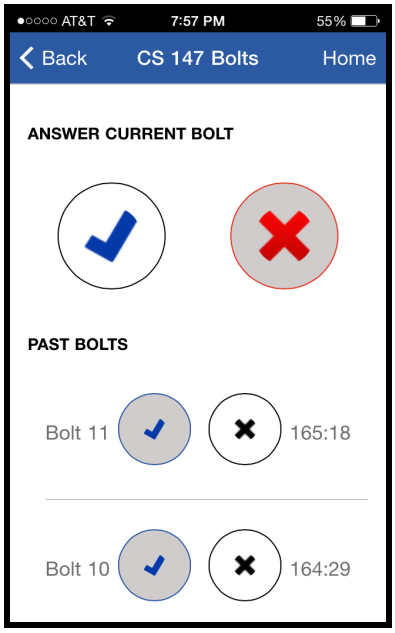
Storyboard of this task



Step 1: The student logs in.



Step 2: The student receives a notification when the professor sends a bolt. He/She clicks “Respond”



Step 3: The student can choose to indicate understanding by clicking on the check, or confusion by clicking on the cross.

Complex task:
Reviewing (overall and individual) performances (Instructor)

The instructor can review performance from two different perspectives:

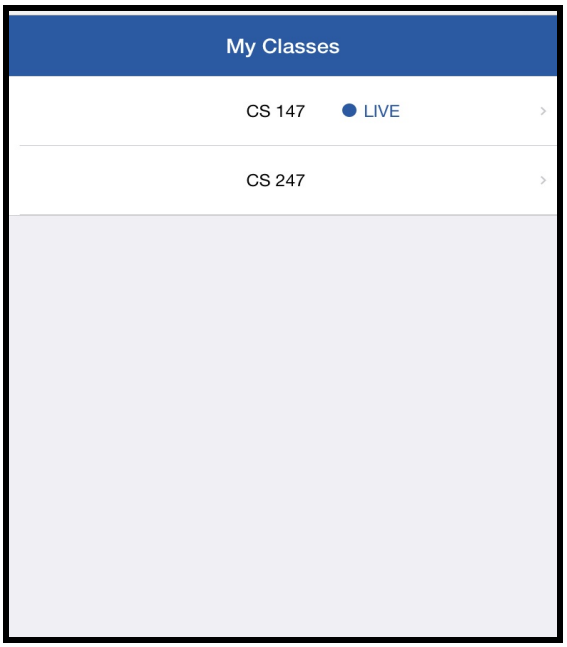
- The instructor can look at meaningful data pulled out from the wealth of data available, in terms of Overall & Best/Worst performance.

- The professor can also look at data for a specific lecture, which includes response data for each bolt sent during that lecture.

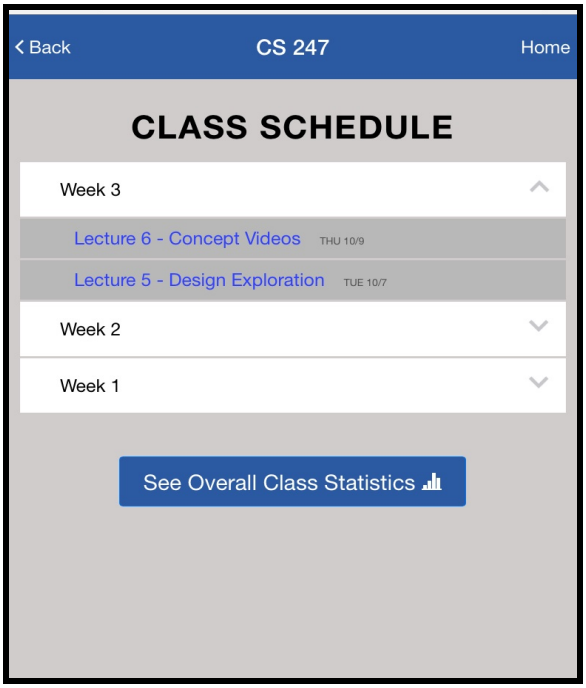
Why was this task chosen?
This task transforms the app from an instantaneously useful lecture tool to a tool with long-term benefits, as the instructor can now use the app not only to improve students' understanding during lecture, but also improve further iterations of the course.

Storyboard of this task

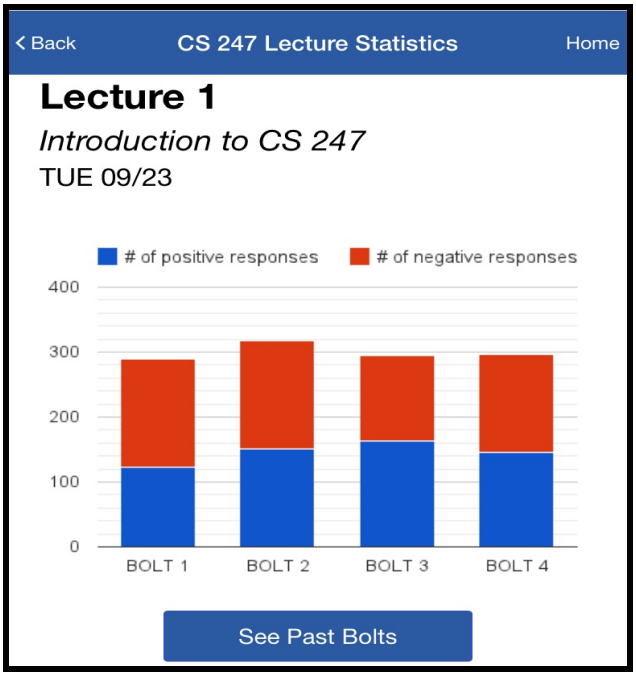
Step 1: The professor can choose a lecture which is not ongoing from the “My Classes” page.



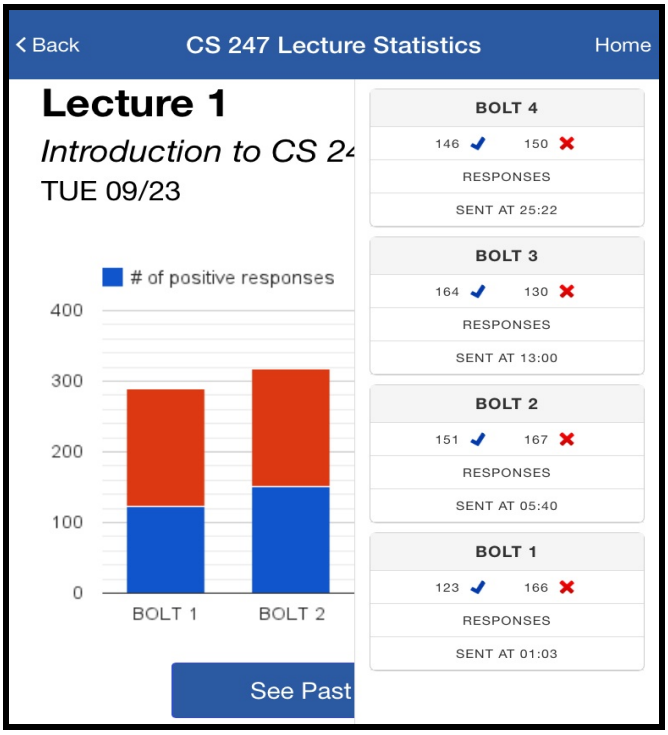
Step 2: The instructor can choose any specific class which has already taken place, or can choose to “See Overall statistics”. Let us say he chooses a specific class.



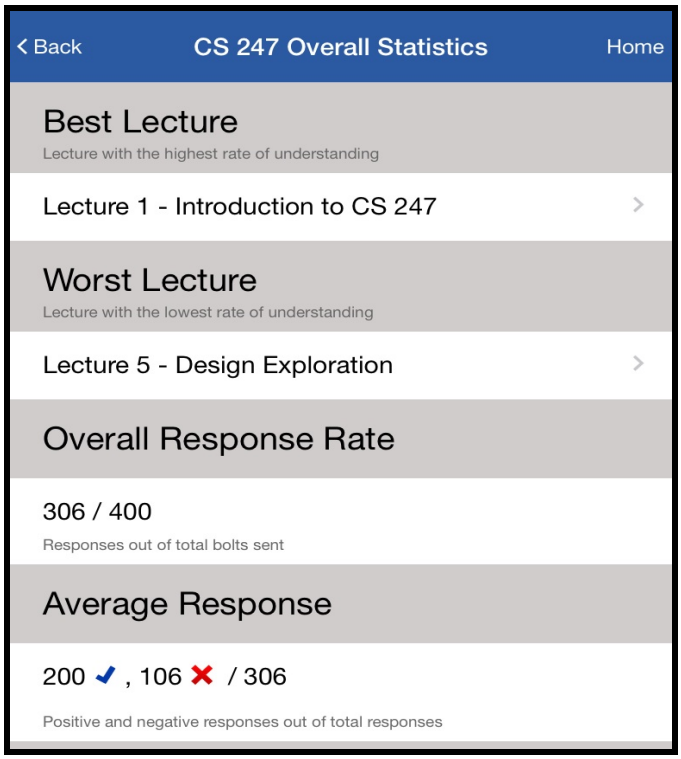
Step 3: The instructor views the response rate and distribution in a graphical format.



Step 4: By clicking on “See Past Bolts”, a pop-up sliding screen appears on the side which allows the instructor to see the details of each individual bolt sent out during that lecture.



Step 5: If, during step 3, the instructor chose “See Overall Statistics”, then he is led to a page which shows overall aggregate data for the course, in order to give him a holistic comprehension of the students’ engagement with the class.



Major Usability Problems Addressed

We considered each level 3 or 4 heuristic violation, and the report for each violation is as follows:

“Current Lecture” screen

1. **Violation**: No confirmation message after clicking “Send Bolt”

The problem was that there is no way of making sure whether a bolt had truly been sent or not. We remedied to this violation by transforming the “Send Bolt” button to “End Bolt” as soon as a bolt is sent. Additionally, the bolt information window gets populated by a new bolt. Thus, there is now no possible confusion about whether a bolt has been successfully sent.

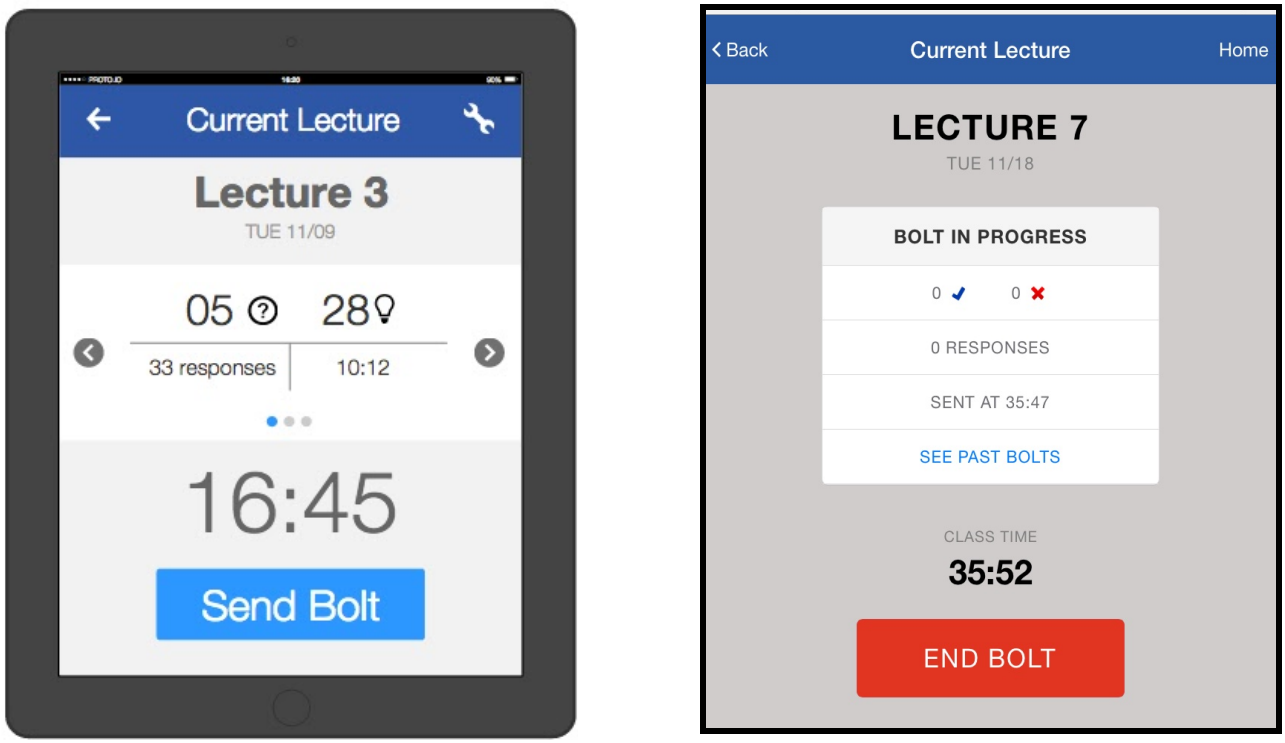


Diagram 1: While our medium-fi prototype (left) didn't provide any confirmation as to whether a bolt was sent or not, the hi-fi prototype (right) makes it very clear using the "End bolt" icon.

2. **Violation**: Users can send endless bolts that confuse the students

This problem was also fixed by transforming the "Send Bolt" button to "End Bolt" as soon as a bolt is sent (refer to diagram 1).

3. **Violation**: If attendance is low, randomly selecting bolt recipients may miss people who attend lectures.

As principle, we would solve this problem by requiring users to sign in at the start of class, and then creating the "random" sample from the students present in class, rather than the whole class student population.

4. **Violation**: The status of the bolt is only known in settings

We changed how bolts worked by having "Start" and "End" buttons instead of having set bolt time lengths like we did before. This allowed for much more flexibility on the side of the instructor. As for the setting that allowed the professor to send bolts to <100% of the class, we were not able to fully implement that setting and therefore couldn't display it on the Current Lecture screen.

5. **Violation**: Icon meanings are unclear

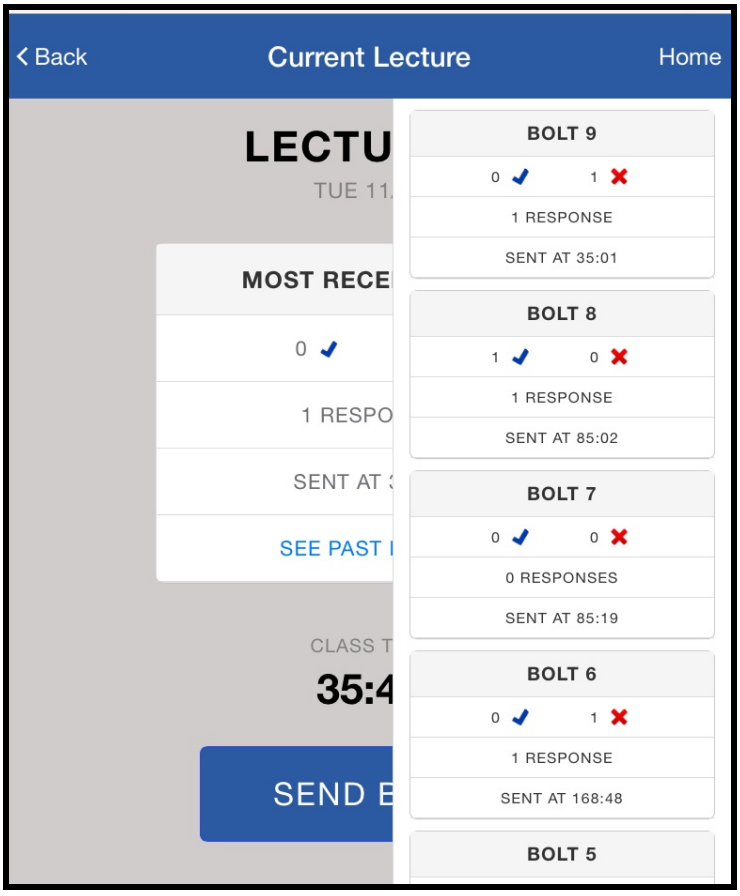
Since the clarity of the response icons are central to our app, we changed the icons to a blue check and a red cross, which are much clearer.

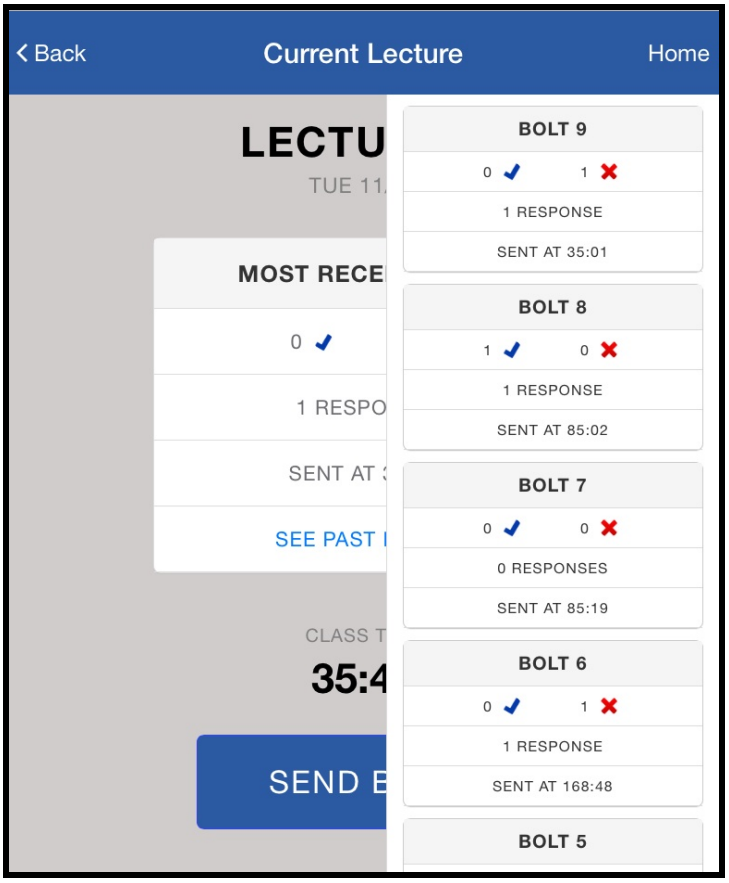
This can be seen in Diagram 1.

6. **Violation**: The dynamic panel is an inefficient way to view bolt statistics, as it doesn't have any indicator or shortcut for most recent bolt.

The default screen displays the most recent bolt. Clicking "send bolt" makes the most recent bolt appear on the dynamic panel. A pop-up screen appears on the side when "View Past Bolts" is clicked, allowing the instructor to see labelled details of all previous bolts sent. See Diagram 2.

Diagram 2: Clear, well-labelled details of past and present bolts are available.





7. **Violation**: There are too many numbers that look like times on this screen, and this fosters confusion in users' minds.

Our fix was to label the numbers in order to eliminate confusion. Once again, please refer to Diagram 2.

Lecture Statistics Screen

1. **Violation**: The color scheme and 100% stack graph do not directly visualize the contrast between the bulb and question.

We spent much time thinking of this violation, but we believe that we approached the color scheme in exactly the way that was outlined in lecture and are convinced that our present color scheme is effective.

Settings Screen

1. **Violation**: Users are constrained to defined times for responses.

Adding an “End Bolt” button allows the Professor to directly control when he/she wants to stop bolt responses, based on how the response rate has been going. Please see Diagram 1.

Schedule Screen

1. **Violation**: Lectures are in chronological order while users may want to access most recent ones first.

We thought this was a very pertinent point, and changed the display order of lectures to display the most recent ones first, as evidenced in Diagram 3.

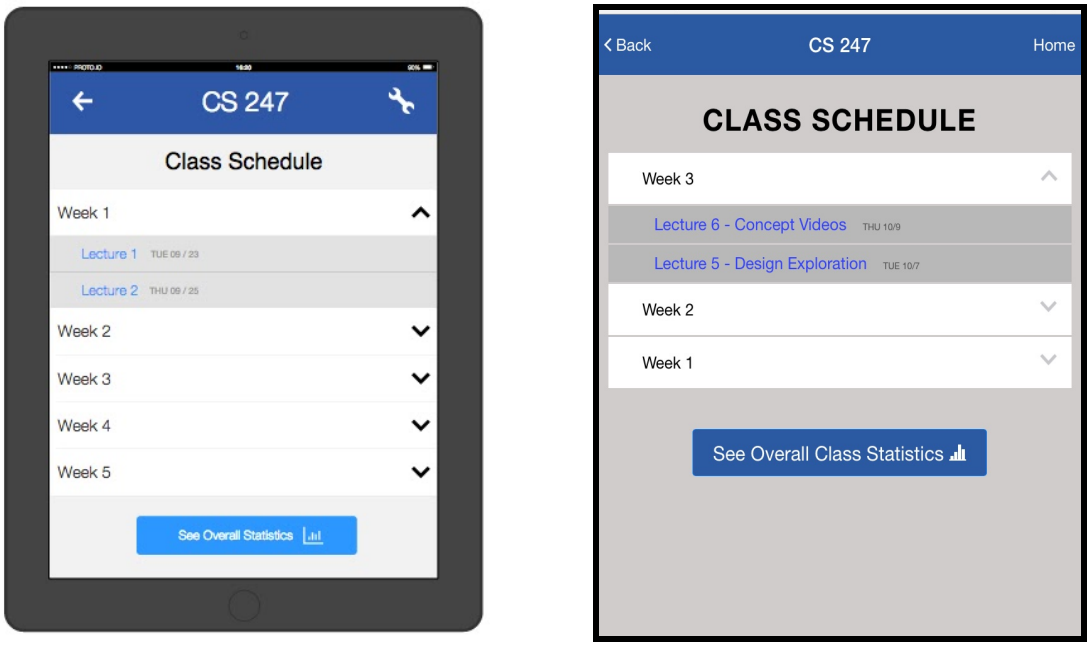


Diagram 3: In the hi-fi prototype (right), the order of lectures is from most recent to oldest, and lectures are labelled by both number and title/name. This is in improvement to the mid-fi prototype (left), in which lectures are ordered from first to last and lecture names are not displayed. In both prototypes, the “Overall Class Statistics” button is prominently displayed.

2. **Violation**: Lecture names are not informative.

Once again, we understood that this would create confusion, among users, and included the lecture title in each lecture description. See Diagram 3.

3. **Violation**: “Overall Class Statistics” button not apparent in the screen.

We disagree with the point that the Overall Stats is not apparent enough, and therefore didn’t change the position or size of the Overall Stats button.

Lecture Stats Screen

1. **Violation**: Response rate or number of responses lacking.

The response rate was already on the screen, and after extensive discussion we decided against changing its size or position.

2. **Violation**: Bolt numbers are unnecessary.

Once again, we discussed this violation, but ultimately came to the conclusion that the bolt numbers are necessary to keep track of the bolts, and for longer-term recording purposes.

Overall Stats Screen

1. **Violation**: The definitions of Best and Worst lectures ambiguous.

In order to remedy to the confusion created by this problem, we added descriptions to the “Best” and “Worst” lectures. Please see Diagram 4.

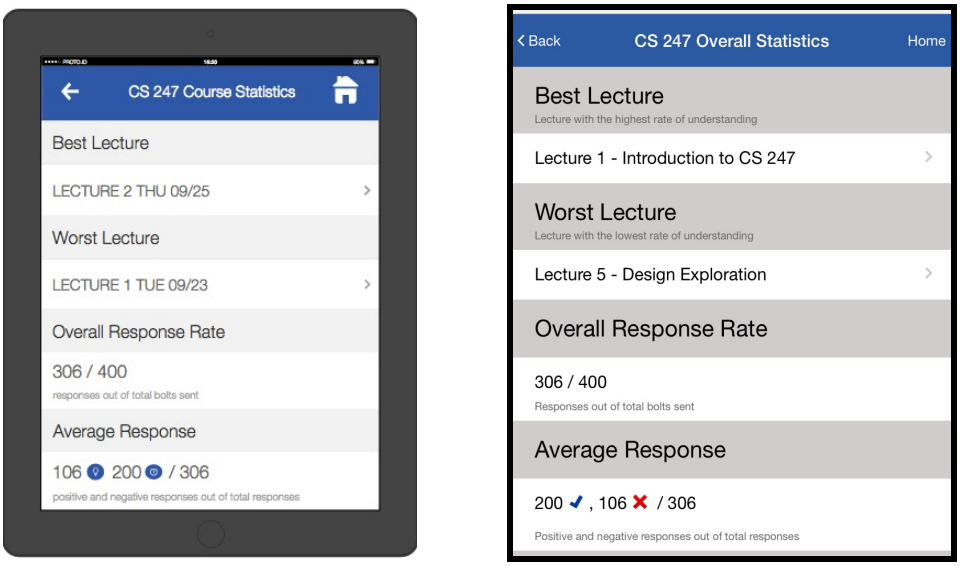


Diagram 4: In the medium-fi prototype, (left) there is no explanation for “Best Lecture” and “Worst lecture”. This has been remedied to in the Hi-Fi prototype (right).

Student UI / Answer Bolt Screen

1. **Violation**: Unclear whether a student should or shouldn't take action when they first see the "answer bolt" screen.

We added an immediate notification, so that the student is immediately notified of the need for action.

2. **Violation**: The meaning of the question mark and light bulb is unclear

We changed the icons to a red cross and blue check, as mentioned previously. Please see diagram 5.

3. **Violation**: Previous bolts seem to be still accessible.

We labeled past bolts and current bolts. Please see diagram 5.

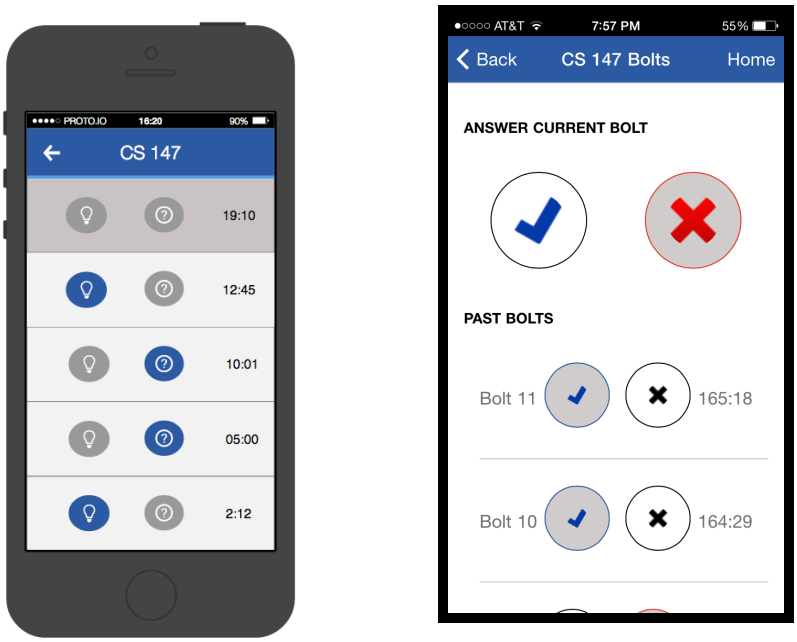


Diagram 5: In the medium-fi prototype, (left) the choice is between a light bulb and a question mark, which can be confusing and non-intuitive, and there is no clear demarcation between past and present bolts. In the hi-fi prototype (right), the choice between a blue cross and a red cross is unmistakable, and there is a clear difference between current and past bolts.

Additional changes

We did not implement any changes in addition to those suggested by the heuristic evaluation, as those already encapsulated most (if not all) of what we felt required transformation in the app given the time constraints.
We did not have to change anything due to platform usability or standard issues.

Design Evolution

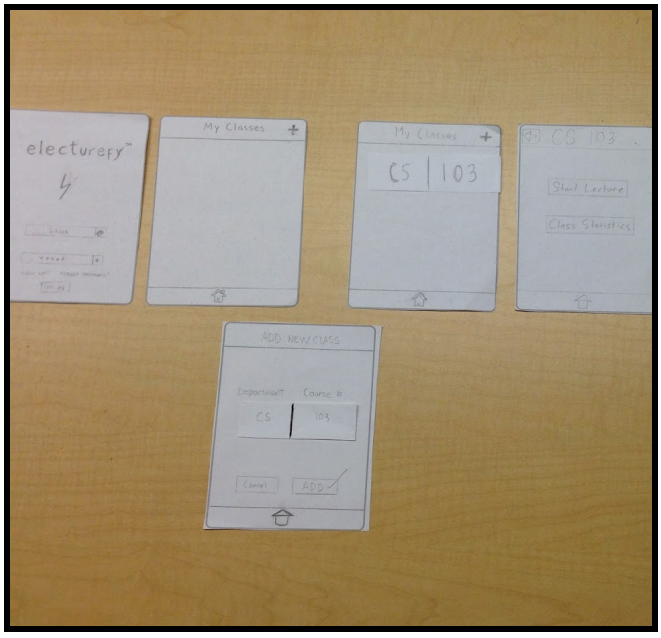
Our design underwent significant changes throughout the quarter, based on the feedback received from:

1. Our CA and peers during section presentations
2. Usability testing with participants from our target audience using the master-apprentice model
3. Heuristic evaluations

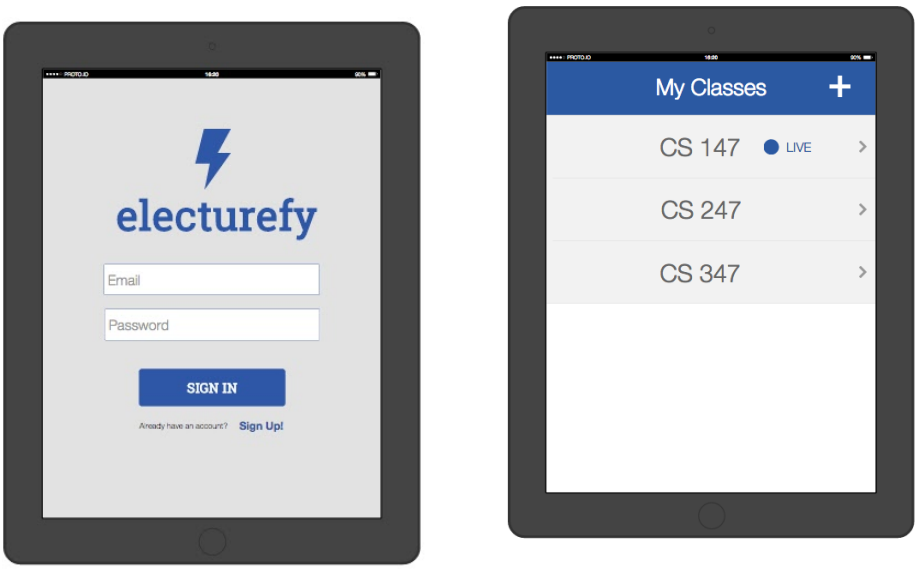
<u>Some designs were kept constant…</u>

Some of our screens been more or less preserved from our low-fi prototype, as shown below. These include:

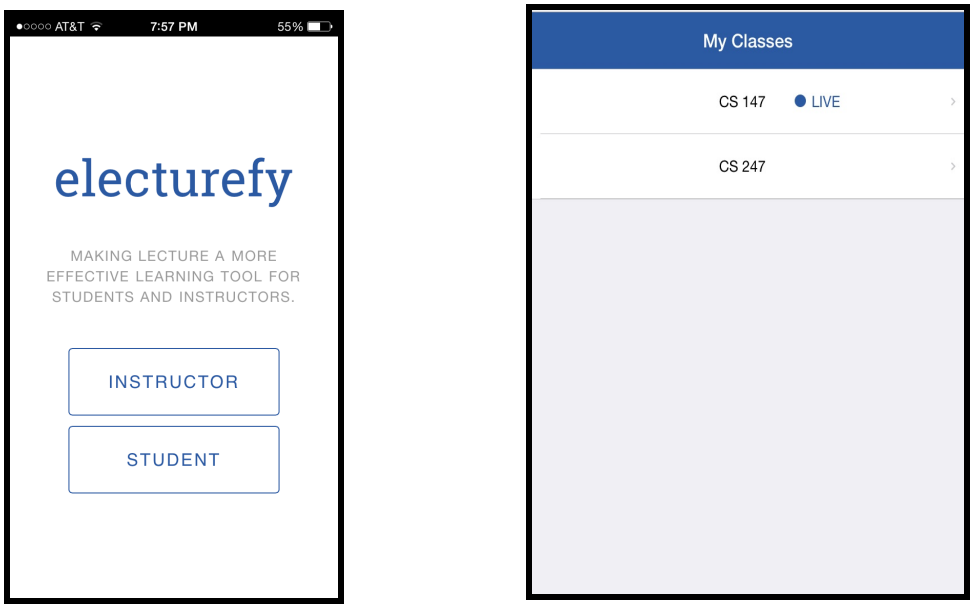
1) Our preliminary login and “my classes” screens



A. The medium-fi prototypes of the login and “my classes” screens

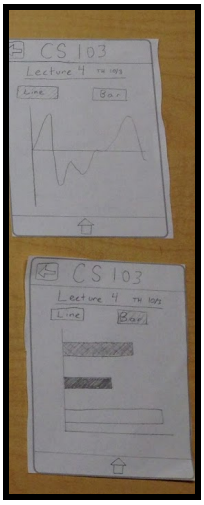


B. The medium-fi prototype of the login and “My classes” screens.

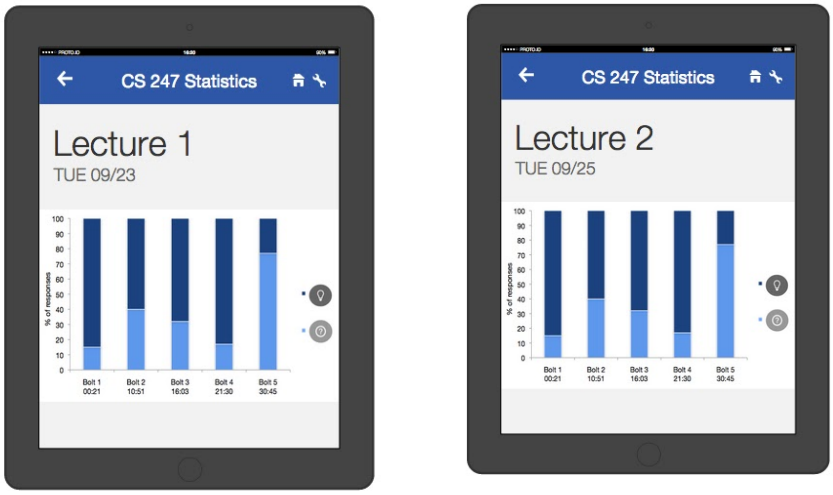


C. The hi-fi prototype of the login and “My classes” screens.

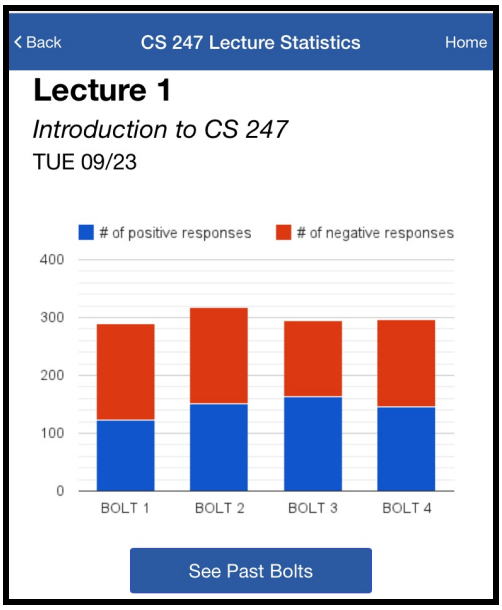
2) Our graphical screens



A. Low-fi prototype of graphical screens



B. Medium-fi prototype of graphical screens



C. Hi-fi prototype of graphical screen

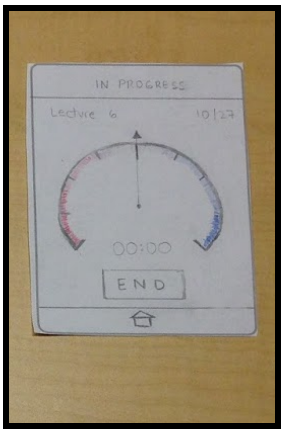
… While others changed radically

Student response screen

The most important screen of of our app - the screen used by the student to indicate confusion or understanding - has completely changed from a dial that indicates varying degrees of student confusion and can be constantly monitored by students, to a screen that provides the temporary binary choice between a cross (to indicate confusion) and a tick, depending on when the instructor decides to “send” or “end” the bolt.

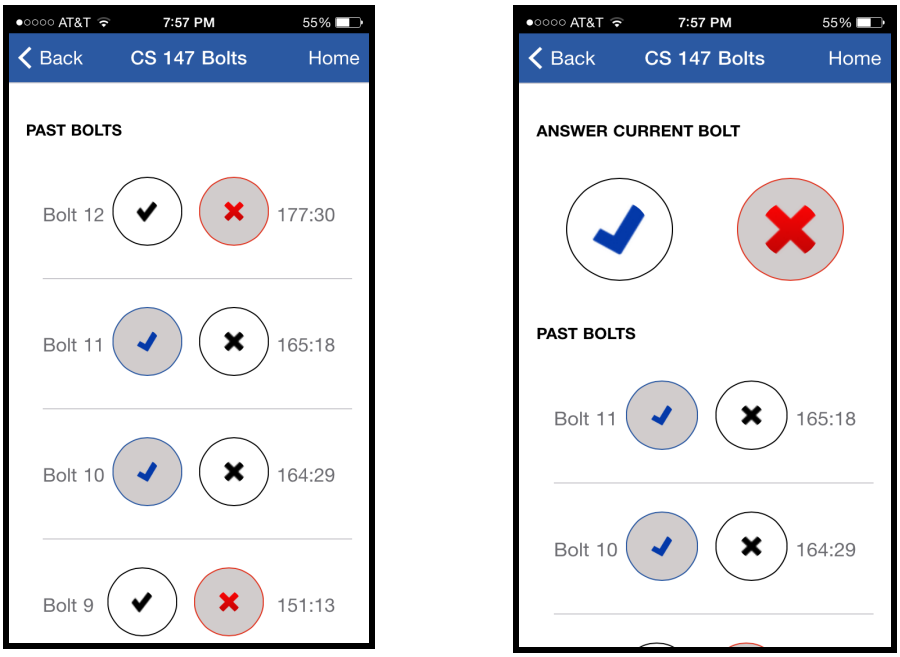
This was a major change, due mostly to feedback we received during our low-fi prototype presentation and user testing. Some of the most compelling points in favor of the change are summarized below:

1. Giving the students the possibility to constantly fiddle with the dial might result in high levels of distraction and less engagement in the class. This was a worry expressed by all the instructors who tested the app as part of user testing.
2. The hand motion required to move across the dial is unintuitive.
3. The fact that the dial provides a continuous range might result in skewed and inaccurate data e.g: what would be the difference between a 68% and 72% understanding rate?
4. We also thought it would be harder to implement a dial than a binary choice.



A. Low-fi prototype of student response screen.

B. Medium-fi prototypes of student response screens

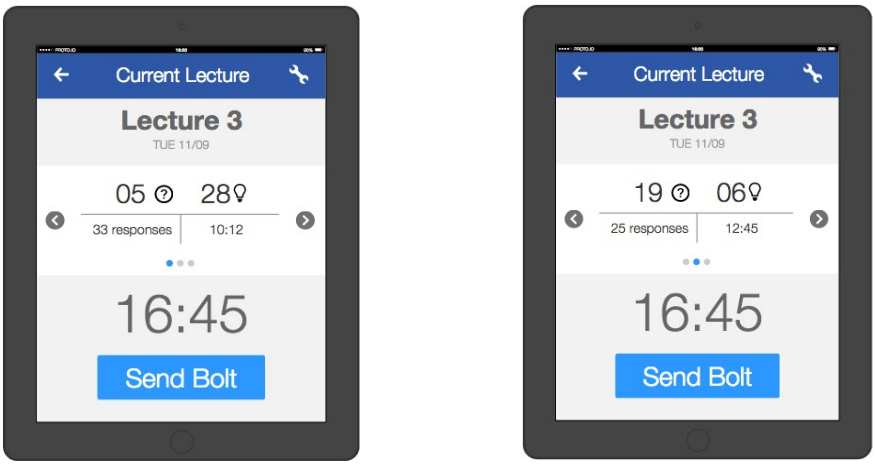


C: Hi-fi prototypes of student response screens

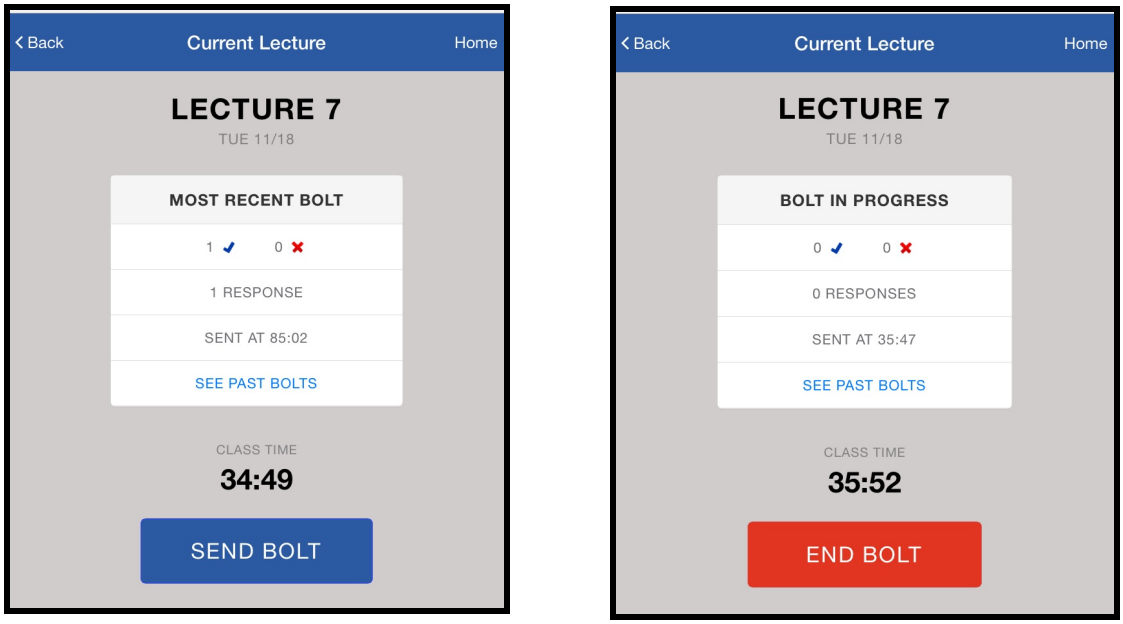
Instructor "Current Lecture" screen

Since one of the major points raised was the lack of control over the time limit for a bolt, we thought it would be a good idea to implement a "Send bolt" and an "End bolt" functionality that would allow the instructor to control when and for how long students use the app.

A: In the low-fi prototype, these functionalities were non-existent as we were, at that point, thinking of implementing a dial.



B. The medium-fi prototypes for the "Send Bolt" functionality. At this point, the "End bolt" functionality doesn't exist yet.



C. The hi-fi prototypes for “Send Bolt” and “End Bolt”.

Prototype Implementation

Tools Used

Since our team lacked the expertise to code in iOS, we felt it was better to go with a **HTML/CSS/JS** front-end framework on top of a **node.js** backend. Most of the team had some sort of exposure to the tools necessary for Web Development, making task distribution and code collaboration possible. At the same time, the learning curve for more complicated features was less steep, and we were able to help each other out more. We also believed that it was necessary to go with a web framework that was mobile friendly, and that allowed our mobile web app to look and feel like a native iOS app. After some research we found a framework called **framework7**. We also used **socket.io**, which is a node.js module that was primarily used for communication between the user and student. As for our database, we used **Parse**

How The Tools Helped

Node.js
By using a node module called express, we were able to generate all necessary routes and app structure using node. Node was especially helpful because it was written in javascript, and most of us had familiarity with javascript or java.

Framework 7
This framework provided a good single page application front-end layout which included handlebars as the templating language, and **require.js** as the library that would load scripts

dynamically. At the same time, **Framework7** came with a css library to make elements and components look like native iOS apps.

Socket.io
It allowed us to create sockets between both our student app and instructor app through our **node.js** backend in a fast and simple way.

Parse
Parse has a really good Javascript API documentation, which was really helpful when trying to query, put or update specific values on the database. At the same time, the fact that it was cloud-based, allowed to us to have our data accessible even when we switched from iOS to **framework7.**

How The Tools Did Not Help

Framework 7
There was a lack of documentation for the CSS elements, and we had to rather find them manually in the .css file. At the same time, since it wasn't a very common framework used for iOS development, whenever we had specific questions about the platform that were not on the documentation, we weren't able to find posts or tutorials from other developers who had the same issues.

Hard-coded Data

In some case we did not use the Parse database, and instead we directly hard-coded many of the elements because of time constraints and static nature of the content. The data that was hard-coded is as follows:

1. The bolts data for both the current lecture and past lectures (hard-coded on Parse)
2. The lecture information for our past lectures view (hard-coded on front-end)
3. The graphs representing the bolt data under each lecture (hard-coded on front-end)
4. The lectures offered at Stanford for adding and searching classes. This was feature was not implemented at the end, but we have the data. (hard-coded on Parse)

Wizard Of Oz Techniques

We implemented real interaction between student and instructor using socket.io, and therefore are no longer using any Wizard of Oz Techniques.

What might we add?

Here is some of the functionality that we might add in the future:

1. Querying / Adding / Removing classes
2. Class settings for randomized bolt sending, automatic duration of bolts, etc.
3. Log-in and sign-up for students and instructors.
4. Ability to send more information than just 'yes' or 'no' when responding to a bolt, like a comment or remark.