HelpList

Santos H., Andres C., Minmin H.

Value proposition

Anonymous in-class asking, ranking and answering questions for college lectures

Problem/Solution Overview

College students can't always ask questions when they get confused in lecture, which may lead to not being able to understand the remainder of the lecture. Furthermore, it is not always apparent which questions are the ones that the majority of students are confused about to faculty, or to students who are willing to help. We are building an unintrusive mobile platform where college students can stay engaged and understand lectures by anonymously asking questions, get their questions answered on the spot and after class, and help others learn by answering questions.

Tasks and final interface scenarios

Simple task: ask a question in lecture

Santos, an easily distracted junior student, is confused about the point the professor is talking about in class and he needs to understand that point quickly in order to understand the rest of lecture. He wants to be able to ask a question to understand the material currently being presented but he is also intimidated to speak out loud in front of the whole class. He wishes there were a way to rapidly ask a question anonymously to the rest of the class and get it answered.

Medium task: give an answer

Andres, an average college student, taking advanced chemistry, has received a lot of help in the past and would like to return the favor and give help to people who are confused in his classes. He would like to be able to easily scroll through the most relevant questions that people have in his class and rapidly give an answer to those he is able to give an answer to.

Hard task: find a previously asked question

Minmin, an EE major grad student, is studying for her chemistry final and rereading her notes from earlier on in the class. She sees something very confusing, and she wants to check to see if that particular question has already been asked by some other confused student beforehand. That way, she would have an immediate answer instead of having to ask the question herself and waiting for somebody to answer it. She wants to be able

to access questions from the particular lecture where that topic was covered and search for a question similar to the one she has.

We chose these three tasks based on our value proposition. We wanted to help students learn better in large lectures and enhance the interactivity of large lectures through an easy, quick, and anonymous way of asking for help during lectures. Students should be able to quickly pull out their phone and ask a question without getting too distracted, or also ask a question when they are rereading their notes at home. We also wanted to provide a simple way of giving help when help is needed. We provide a way to rank the questions, so that the most relevant questions bubble up to the top, so that those questions are more visible and can get answered more rapidly. Finally, we wanted an easy way of leveraging the existing resources - questions that have been asked previously. Students should be able to search through the entire database of questions for a class and find questions that have already been asked that are similar to questions they may be having.

Design evolution

Initial sketches

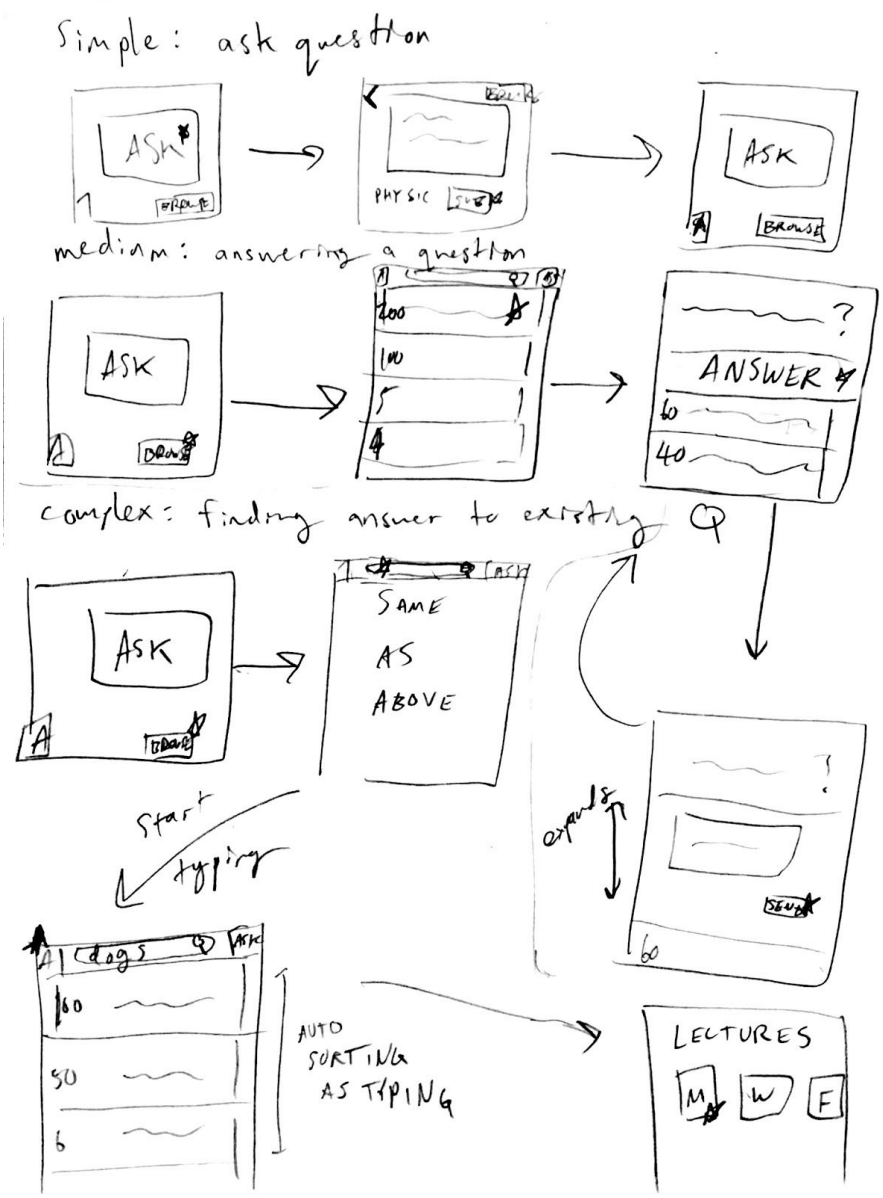


Figure 1. The initial sketches of the three tasks

We did not do any user testing on the initial sketches. They only time they were exposed to the public was when we were deciding which application to develop (between this idea and a couple of others). The testers did not go in depth in the application.

Low-fidelity prototype

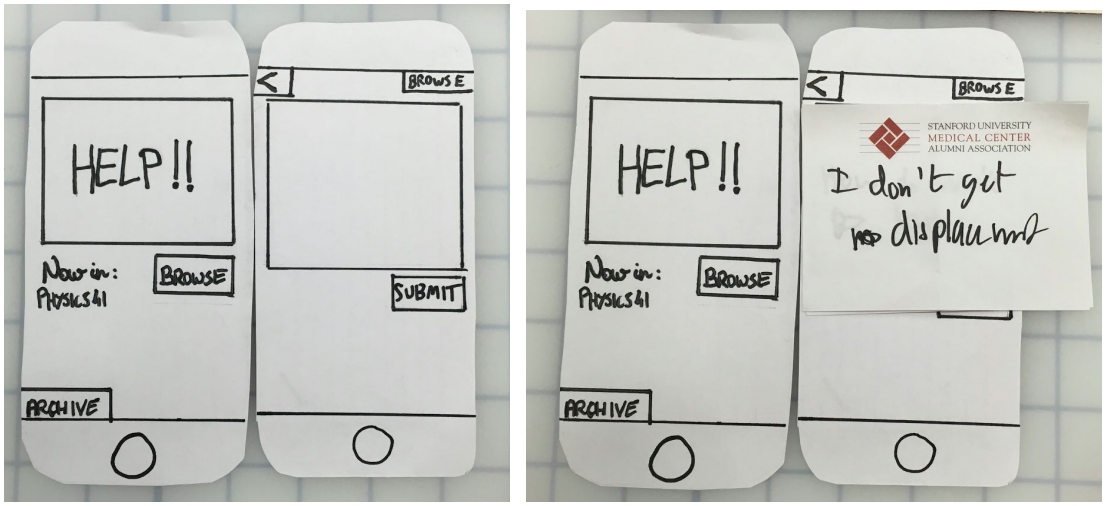


Figure 2(a). Low-fidelity prototype for the simple task (ask a question)

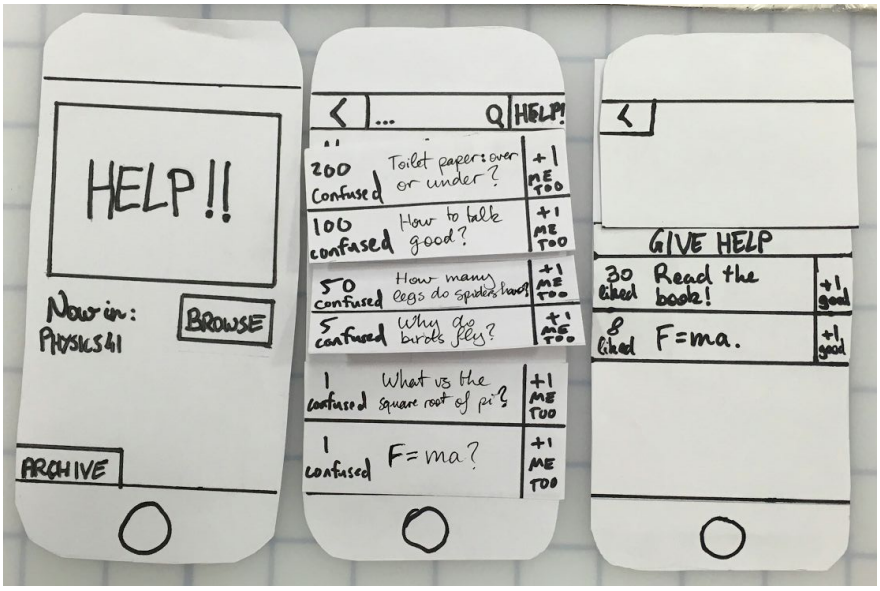


Figure 2(b). Low-fidelity prototype for the medium task (given an answer)

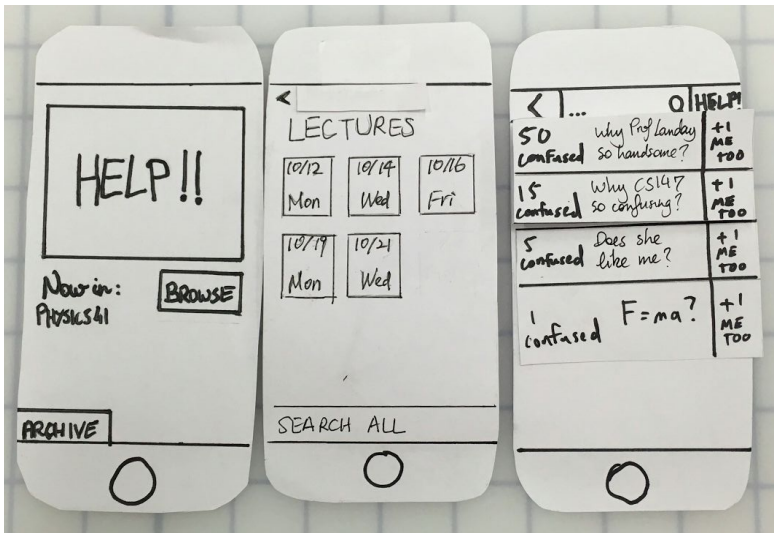


Figure 2 (c). Low-fidelity prototype for the hard task (find a previously asked question)

We used the low-fidelity prototype shown above to conduct user testing. Three Stanford undergraduates were recruited outside large lecture halls to participate in the testing. They were given some basic information about the purpose of the application, then asked to complete the tasks. One of our team acted as the facilitator, giving them basic information about what task to complete, another acted as the application, switching out their screens based on their interactions, and the last was a note-taker. The testing results and design changes that are in response to the discoveries are summarized in the table below.

Table 1. Problems discovered from user testing and design changes made in response

Problem found	Design change
Two participants separately had trouble finding the search bar on the browse screen	Add grey text inside the search bar that reads “search” and that immediately disappears when clicked on.
Two participants had trouble realizing that once you’ve typed an answer in the “give help” section, you have to tap “give help” again to record your answer to the question.	Once a user taps on the “give help” button, and the dialogue box appears, the “give help” text will change to “submit”.
All three of the participants did not immediately see the “Archive” button, to search through older lectures.	Change the text of the button to “previous lectures” and make the button much larger
The search process is a bit too complicated to use while listening to lecture.	Reduce the multiple search methods into one simple way (search bar on top of every

	screen) that users have got used to through using other mobile apps.

Medium-fidelity prototype

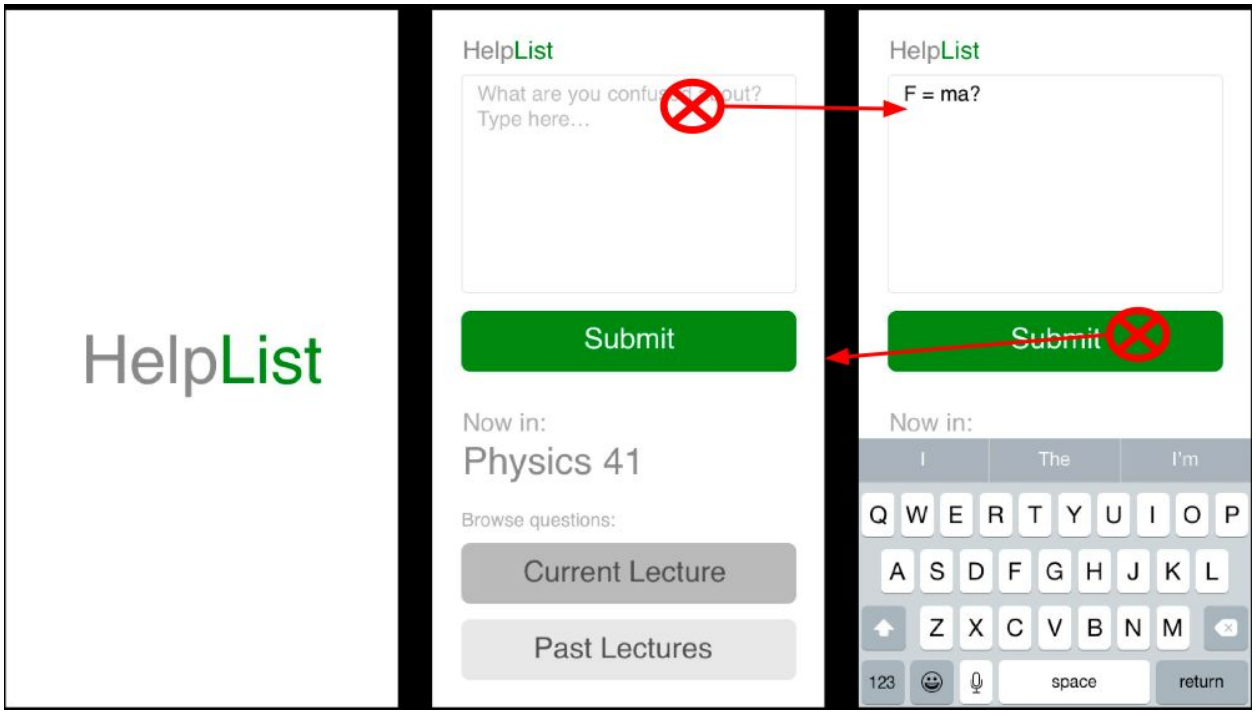


Figure 3 (a). Medium-fidelity prototype screenshots for the simple task of asking a question (red circle with a cross means tapping)

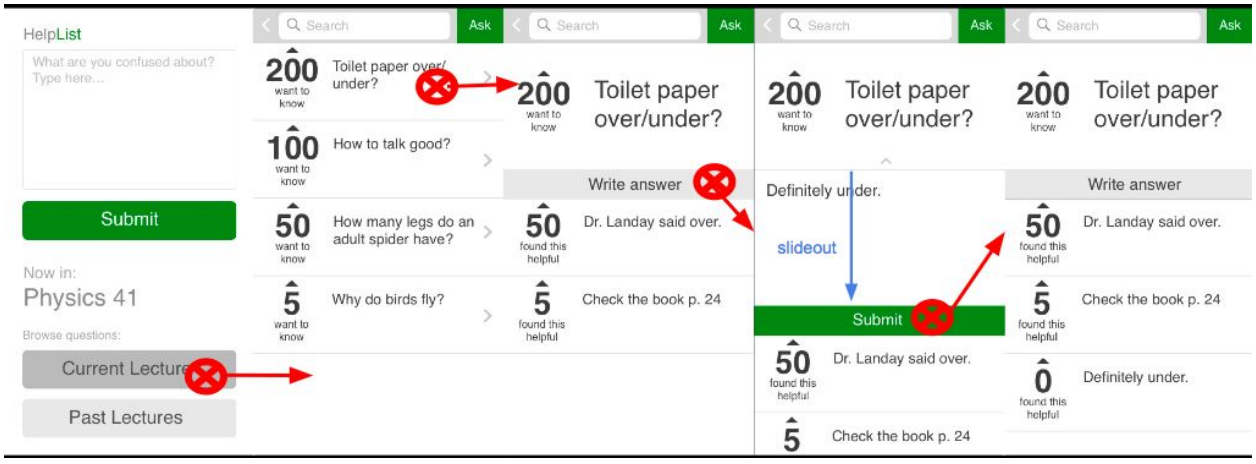


Figure 3 (b). Medium-fidelity prototype screenshots for the medium task of answering a question (red circle with a cross means tapping)

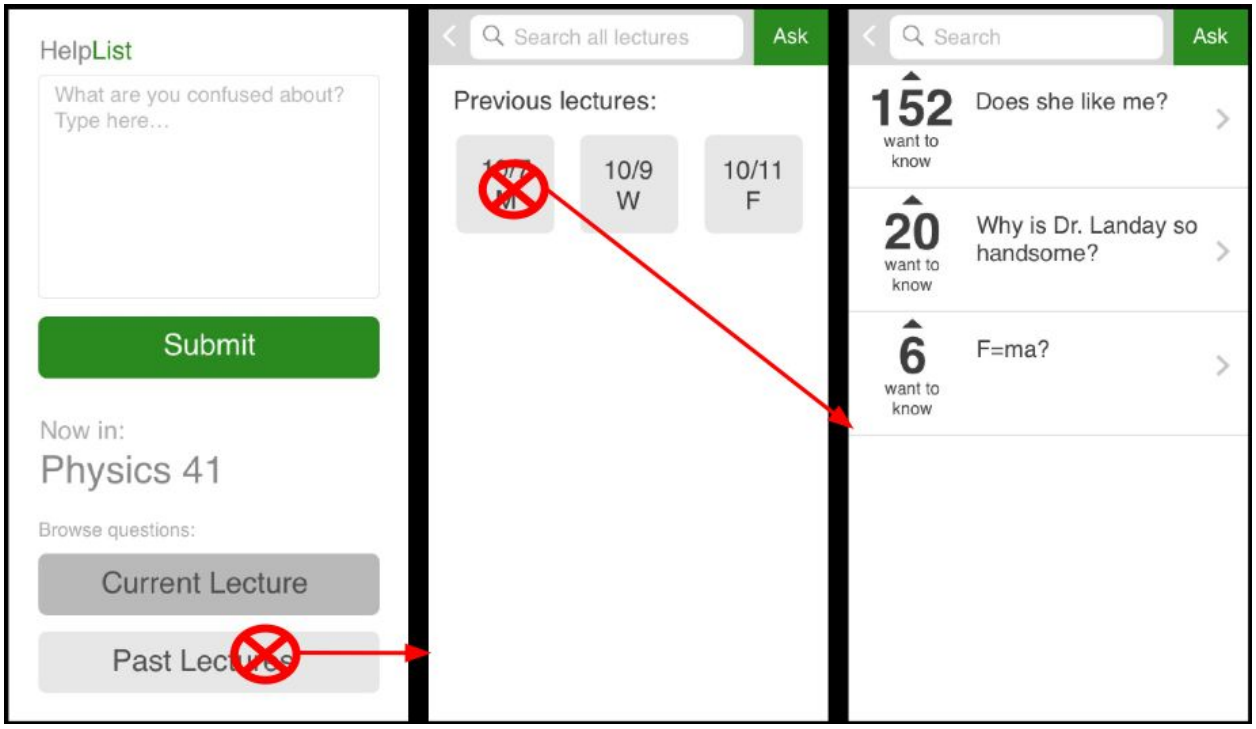
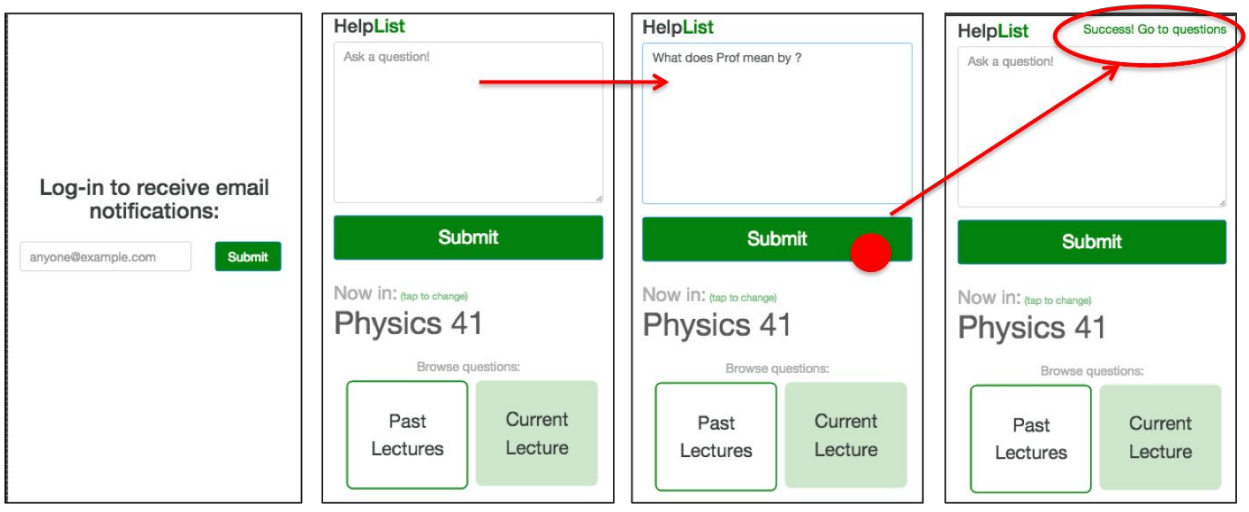


Figure 3 (c). Medium-fidelity prototype screenshots for the hard task of finding a previously asked question (red circle with a cross means tapping). User can search in the search bar to find questions with the searched key words.

We used the medium-fidelity prototype for heuristic evaluation. The evaluation results and our responses are described in the next section.

<u>High-fidelity prototype</u>

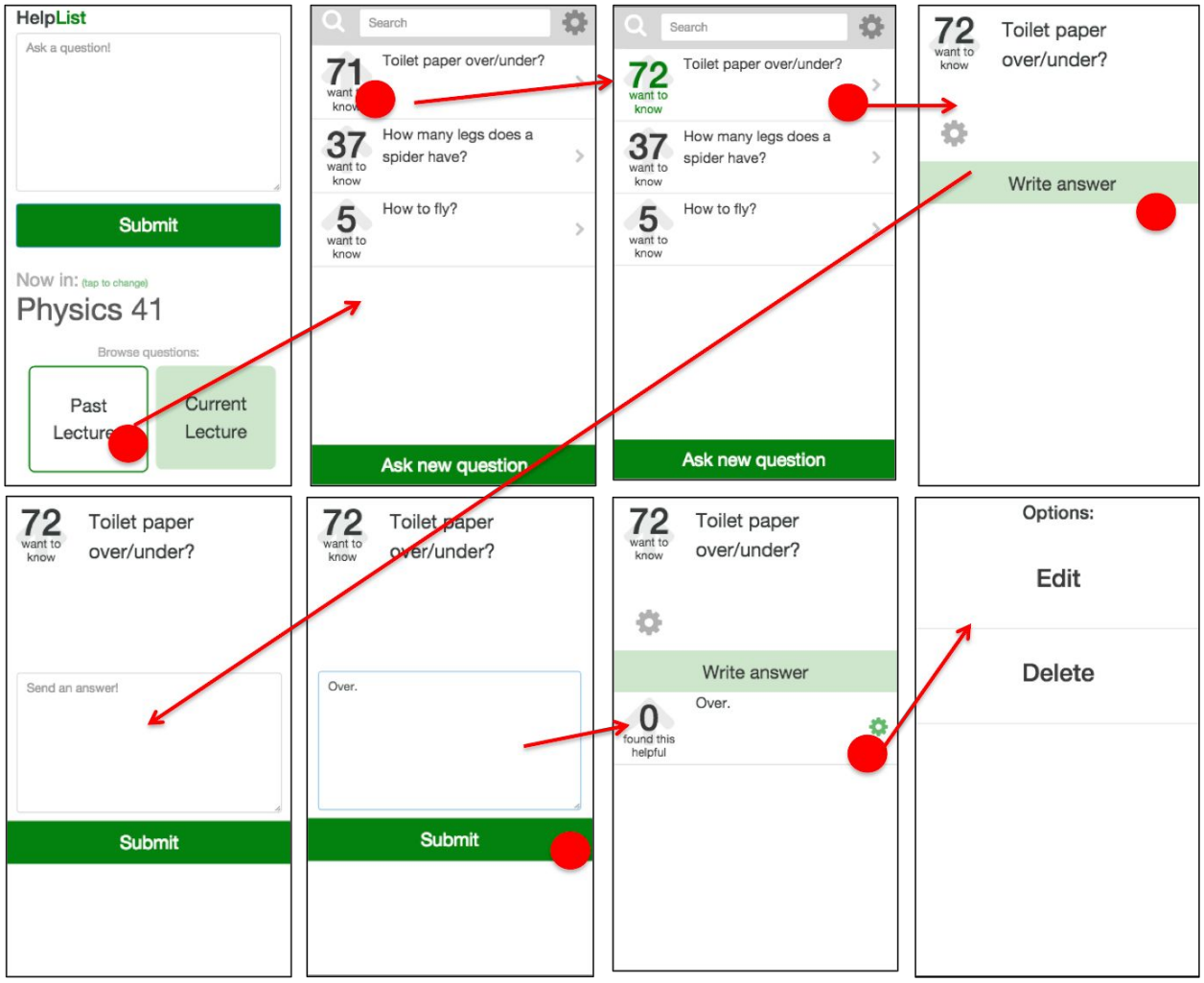
Storyboard for Task 1: Ask a question



User can log in with their email to get notifications when the questions they asked get answered. The main screen has a large text box where user can type in their question, tapping submit will

submit the question and a notification will appear to inform user that the question has been submitted.

Storyboard of Task 2: Answer a question



User can tap on either “past lecture” or “current lecture” to go to the question list. Then user can tap on a specific question, tap on “write an answer”, type in their answer, then tap on “submit”. User can edit and/or delete their answer by tapping on the gear next to the answer.

User can upvote a question or an answer by tapping on the number next to the question or answer. The number has a up-pointed arrow underneath to indicate the upvoting function.

Task 3: Find a previously asked question
User can tap on either “past lecture” or “current lecture” to go to the question list. On the question list screen, user can search in the search bar.

Major usability problems addressed

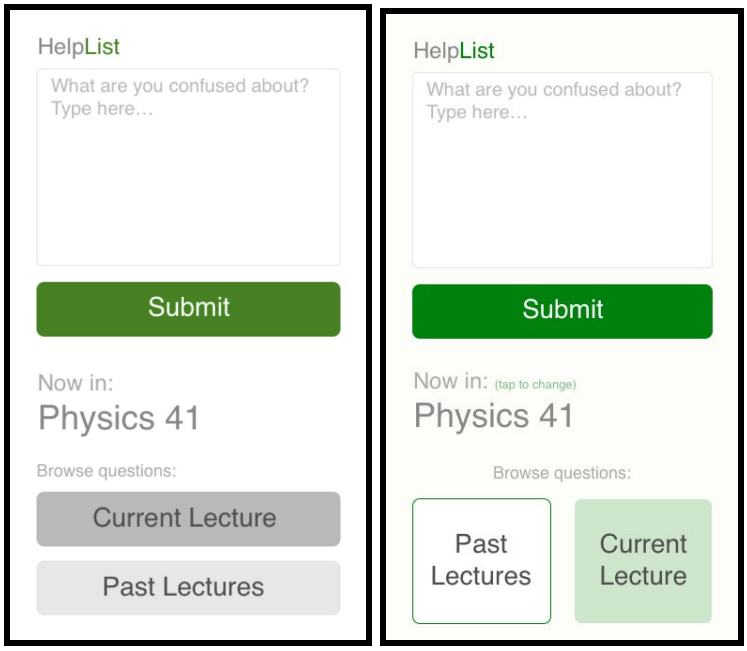
Problem [severity level]	Response	Rationale
[H2-4 Consistency & Standards] [Severity 4] The "ask" button next to the search bar looks like it should complete a search.	Remove the "Ask" button next to the search bar. Put the "Ask" button at the bottom of the screen as a green clickable banner.	Eliminate potential confusion about the "Ask" button. At the same time, keep the convenience of being able to ask questions whenever. Green button-like banner at the bottom would serve as a visual affordance.
[H2-2 Match between system and the real world] [Severity 3] When I submit a question, I expected to be taken to the page with all of the questions and see my question on there.	We decided not to direct the user to a different screen where all questions are listed, instead we will show a "your question has been submitted" pop-up to tell the user that his/her question has been recorded.	Taking the user to the screen with all the questions could distract the user when the app is used in lecture. We want our user to focus on the lecture and has an easy and quick method to submit a question when one comes up.
[H2-2 Match between system and the real world] [Severity 3] Why is the search bar on the page where I'm inspecting a question and its answers? I have no intuition as to what I would be searching for on this page.	We removed the search bar and the "ask" button on that screen.	The evaluators are right about this. The search bar is redundant and even confusing in that screen - what database would it be searching through from that page?
[H2-8 Aesthetic and minimalistic design] [Severity 4] The "write answer" button looks like a heading, not a button.	We changed the color of the button to green, our theme color. And we made the button appear to be more clickable.	We decided to incorporate the visual affordance that every clickable button would be green. This goes along with that theme and should make it more clear.
H2-6 Recognition rather than recall] [Severity 3] It would be nice if from the	We decided not to include the answers or how many answers a particular	We want to make the screen clean and easy to read. The most important

page that lists the questions in the current lecture, you could see if a particular question had answers (or how many answers).	question has on that screen.	thing to the user is to spot the question that he/she is interested in. It would make the screen very cluttered with all the other unimportant information. Also, it is easy for the user to tap on the question to see the answers.
[H2-3 User Control and Freedom] [Severity 4] As of right now, there is no way to "escape" this class. Automatically going into class trades off against flexibility.	We will change the class number text (e.g., Physics 41) into a green button where user can tap on and change the class.	The evaluators are right. We will make the change to add the flexibility to switch between classes, but will still automatically log the user into the class they are currently in (based on location) to make it as unintrusive as possible for users to ask a question.
[H2-9 Help users recognize, diagnose, and recover from errors] [Severity 4] When I went to past lectures, it isn't really obvious how I get back. I know there is a back button, but that doesn't really feel natural in this hierarchy	We decided to not change our design.	The back button should serve its function well. The user would click on one of the lectures and go into the list of the questions for that lecture. If they want to go back, they would simply click on the back button. This is a commonly encountered situation in mobile apps.
[H2-4 Consistency and Standards] [Severity 3] While the "Ask" button on a lecture screen takes the student to the initial ask question screen, this isn't consistent with the button on that page – which reads	We will remove the "Ask" button next to the search bar on a lecture screen and put a banner at the bottom of the screen that reads "Ask a question" or "submit a question".	Positioning the "ask a question" button at the bottom would remove the confusion. And as the user should be familiar with typing in a text box and click on "submit" button from using many other mobile apps. The submit button on the home screen

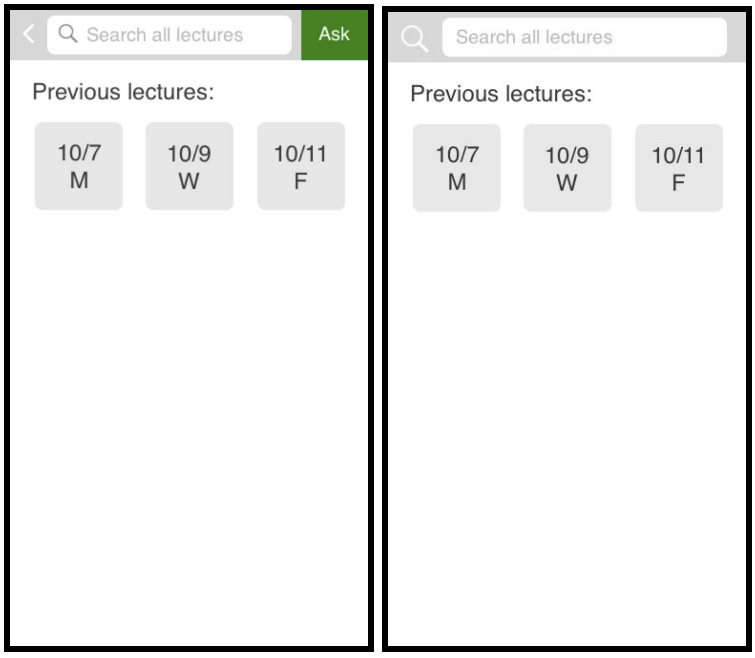
“Submit”.		would not cause confusion, because if they attempt to click it without having asked a question an error message will pop up informing them that they have to write a question before attempting to submit.
H2-7 Flexibility & Efficiency of Use] [Severity 3] When submitting an answer to a question, once the user types something it isn’t immediately apparent how to exit from that state and either submit/delete their reply entirely.	In our high-fidelity prototype, user would be able to exit, submit and delete their answers.	This problem seen in the medium-fidelity prototype is due to the limitation of the prototyping tools.
[H2-4 Consistency and Standards] [Severity 3] When asking questions in a Past Lecture, the Ask button is the same as in other places, so it is not clear that the user is asking a question in that previous class.	We decided to move the ask button to the bottom of the screen here as well, to remain consistent with our design. This should remove all confusion as it did above.	The user would tap on a particular lecture to go to that lecture’s question list and ask a question there. This would help the user to realize which lecture he/she is posting question in.
[H2-6 Recognition Rather Than Recall] [Severity 3] At any given question, the user can’t tell which lecture they are currently in. It would be nice for users to be able to tell which lecture the questions they are currently looking at are	We decided not to change our design.	The user would tap on a particular lecture to go to that lecture’s question list and ask a question there. This task flow would help the user to realize which lecture he/she is posting question in. There is also no logical place to add this information without cluttering the screen.

from.		
[H2-2 Match between system and the real world] [Severity 3] It would be helpful to get notified if a question that I upvoted was answered.	We are adding email notifications to alert a user when a question they asked has been answered. in addition, we are allowing users to “follow” a question, in which case they will be notified if it gets answered.	These notifications will let you know when you receive an answer to your question. However, there are many reasons for which you may upvote a question: maybe you just think it is a good question and want it to be more visible. Therefore we are going with the “follow” button for notifications.
[H2-3 User Control and Freedom][Severity 4] After submitting a question the user has no way to see it, edit it, delete it, or get a notification if the question is answered.	Same as above: this was not evident in the medium-fi prototype because of the nature of it, but in the hi-fi prototype the user will get notified when a question they submitted or followed gets answered. we decided to not allow edit functionality, but we are allowing users to delete questions.	The notifications will enable the user to quickly know when the receive an answer. we decided to not allow edit functionality because if the question already for answered editing it would make the responses wrong, and same thing with upvotes. However, we are allowing the users to delete a question that they have asked, or an answer that they have submitted. deleting a question will also delete all the responses tied to it.

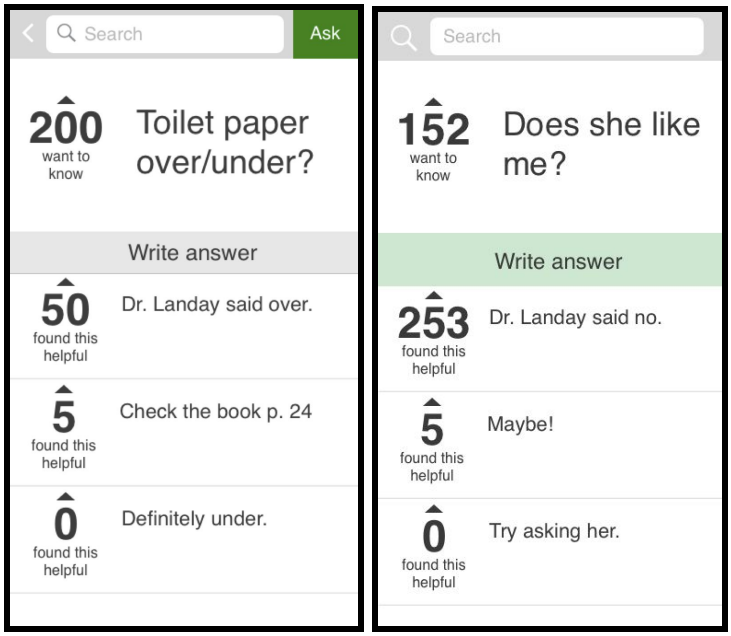
Here are some before and after screenshots from the medium fi prototype made in sketch that reflect the major changes. The after screenshots are those that are getting implemented in the hi-fi prototype. The before shots are always on the left, and the after screenshots are always on the right.



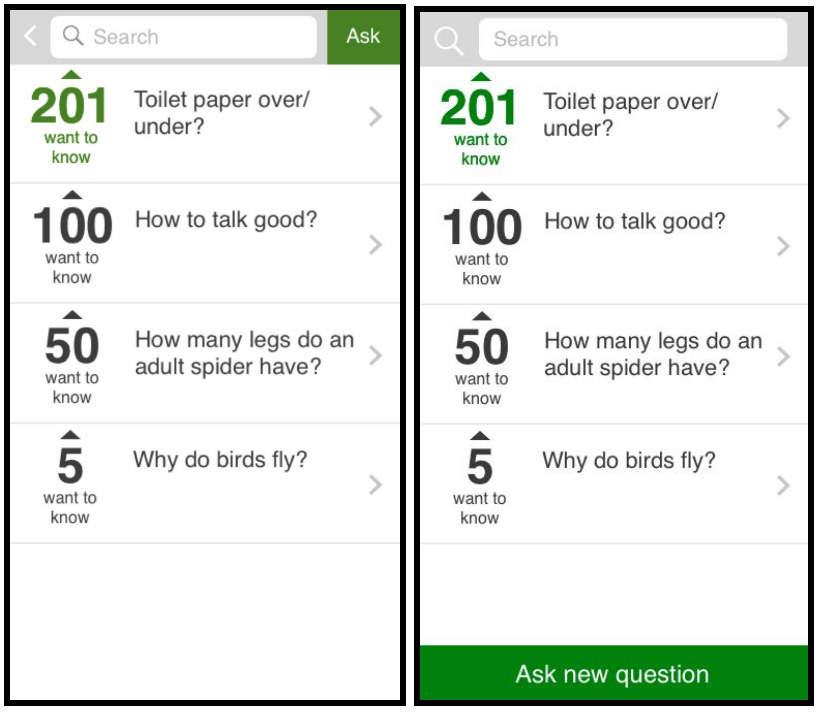
These pictures are the before and after-change of the home screen. The main things we changed are the current lecture and past lecture buttons, and the “now in” section. We made the buttons green to seem more clickable, and we added the “tap to change” note after “now in” to let the user know that they can switch out of that class. This will make the user capable of switching to different classes or checking different feeds when they are not in the classroom.



These are the before and after-change screenshots of the previous lectures screen. The main thing we changed is the removal of the “ask” button from this screen, as it does not make much sense to have it here (what feed would it be posting to?)



These are the before and after-change screenshots for the screen for a specific question. The main changes are the removal of the “ask” button, because it is unnecessary and redundant here, and the change of the color of the “write answer” banner to green in order for it to seem more like an action button rather than simply a banner.



Finally, these are the before and after-change screenshots of the questions index screen, where you can scroll through the feed for a class session and view all the questions. the main differences are that we remove the “ask” button from the top, as its

location made it ambiguous as to whether it completed a search, and moved it to a banner at the bottom of the screen that always stays present even while you scroll.

Another main change is that there is no longer any “back” button on any of the screens. The rationale for this is that due to our general lack of knowledge in coding native mobile applications, we are making our application a mobile web application. Therefore, users will be informed in the prototype readme that they should use the back button in their browser to get back to the previous screen. This also allows us to put the “Q” symbol next to the search bar to make the search function more obvious.

Prototype implementation

We implemented our high-fidelity prototype as a mobile web application due to our very limited experiences of developing native mobile applications. We used HTML, CSS and Ruby on Rails for the development.

This is somewhat less than ideal, because the design of our application would ideally lend itself to a native mobile application. However, because of the limitations mentioned above, we have decided to go with a mobile web application, which would also lend itself well to an iphone screen.

This tool definitely helped because one person on our team had experience developing those, so that was a huge asset. Another advantage that the mobile web app confers is that we do not need back button, as the user can just click the back button on whichever browser they are using.

However, there are some drawbacks to making a mobile web application. First off, it would simply be more cumbersome for the user to open a browser and navigate to the site, rather than simply opening an application on their phone. In addition, making a mobile web application makes it so that we cannot use push notifications, which are specific to native mobile applications and would be the most convenient for users. Therefore, to substitute that, the idea is that users would input their email when they logged into the application, and they would get email notifications when a question they asked or followed got answered. However, this is hypothetical because we are not going to be coding in the notifications due to technical limitations and our limited skillset.

Wizard of oz techniques that we used include:

1. Pre-defined class: Location based automatic sign-in is not trivial to implement. Given the time constraint, we decided to pre-define a class (Physics 41), and during user testing, the users will all be in that class.

2. Search function: we will ask user to search a pre-defined question in user testing, as we have a limited number of questions available (see hardcoded data)

Hard-coded data in the high-fidelity prototype include:

1. Existing questions in current and previous lectures
2. Previous lectures and their dates
3. Class list for a user

Were we to continue developing this application in the future, there are some important steps we could take to make the application fully functional.

The first one of these would be to be to incorporate location settings, to be able to log people into their classes directly when they are located inside those classrooms. They would still be able to log out of the feed of that class and switch to another one, but for ease of use, if they are in a class, they will be automatically logged into the feed of that class. However, this would also require coordinating with Stanford's administration, and getting a list of all the classes in a given quarter and their locations, then somehow setting up that database. If that was implemented, then the application would be essentially fully functional and ready to be used.

Summary and Future work

We conducted an iterative UI design process over the quarter. We came up with our project based on the needfinding interviews. We modified our designs based on the user testing with low-fidelity prototypes. We further polished our designs based on heuristic evaluation feedback. For the high-fidelity prototype, we implemented the user interfaces and some of the backend functionalities. Some hard-coded data is used in the high-fidelity prototype due to the time constraint. However, the implemented prototype is sufficient for future user testing. Further work needs to be done on the backend to fully support the some of designed functionalities, such as automatically signing into classes.