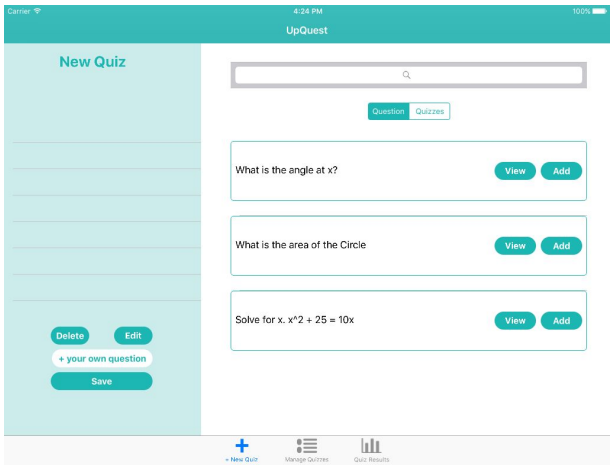
UpQuest

The Student Data That Keeps on Giving

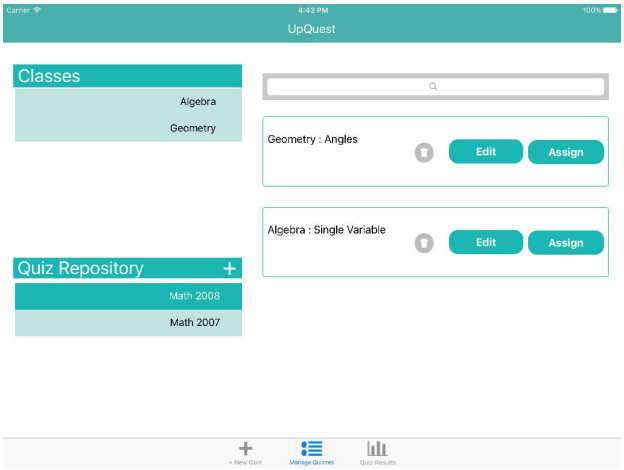
Kat Guo, Matt Chen, Tyler Berbert

Problem & Solution

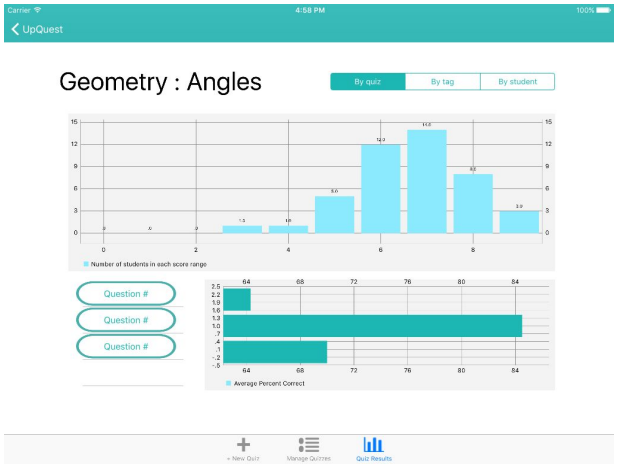
K-12 teachers often start the school year with limited information about the knowledge levels of their students. They also find it hard to track students' progress over time. Our application allows teachers to give formative assessments to their students and quickly understand their strong and weak points.



New quiz view



Quiz management view

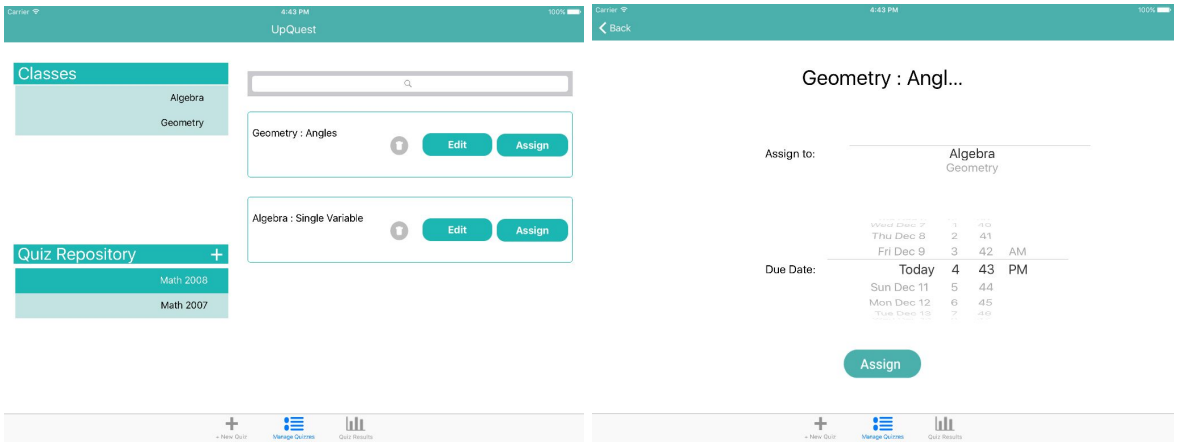


Quiz results view

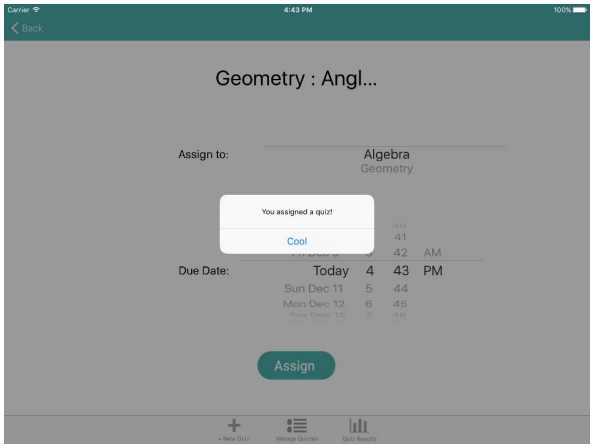
Tasks & Final Interface Scenarios

Easy: The teacher manages all quizzes given to students

We chose this task as the easy one because it is mainly assigning quizzes from the repository and editing due dates of quizzes that are given to students.



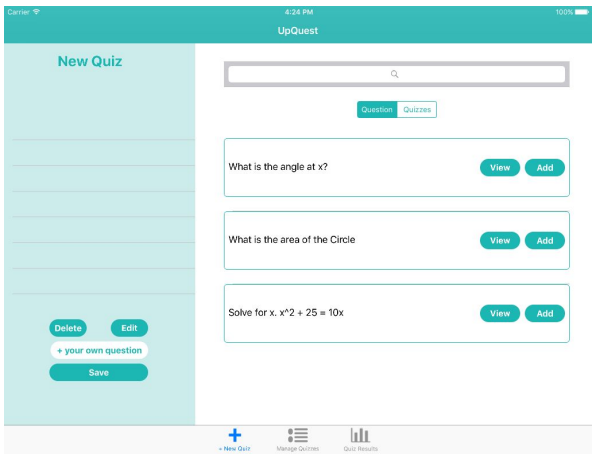
From quiz repository, click assign button to select the due date and class.



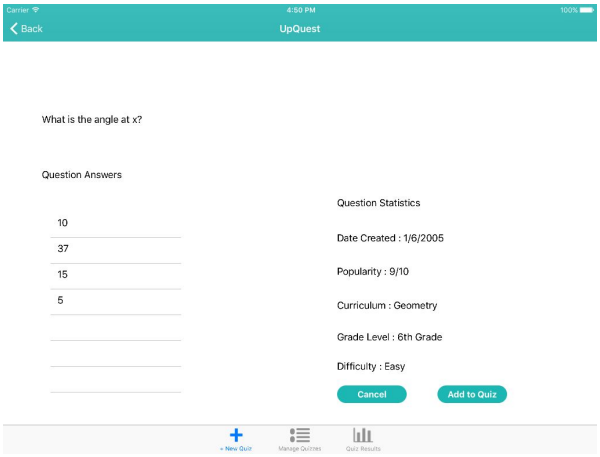
Click on assign button here lead to a pop-up of confirmation.

Medium: The teacher creates custom quizzes for students

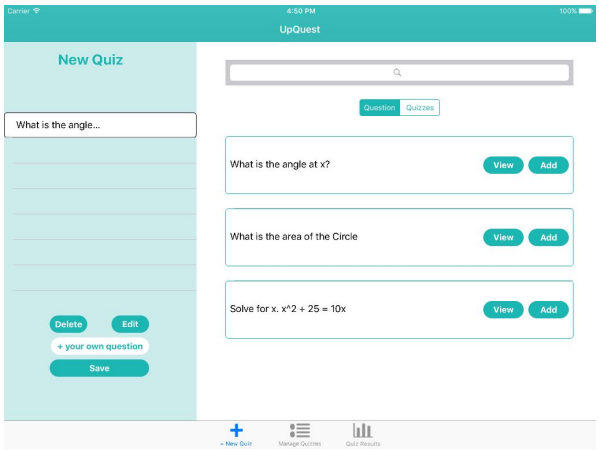
We chose this task as the medium one because it requires teachers to select questions from our quiz bank.



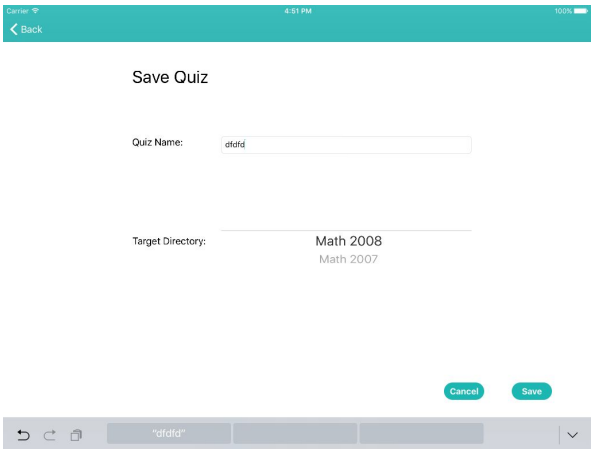
Click on “view” button to view a question



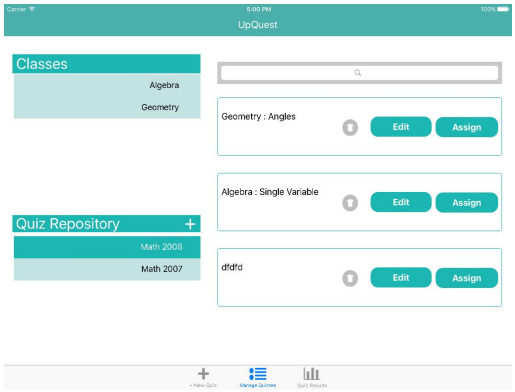
Click on “add to quiz” button



Click on “save” button



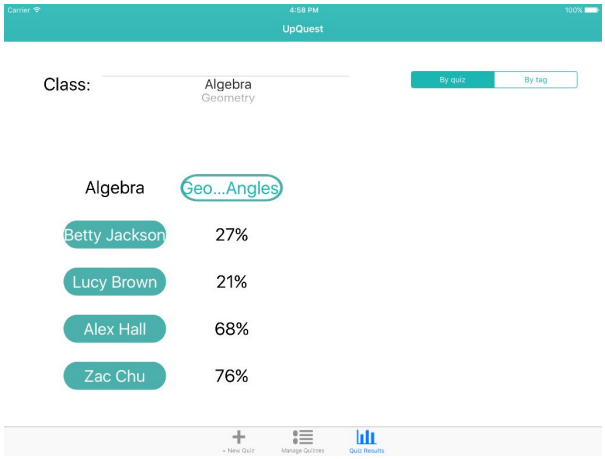
Type in quiz name and click “save”



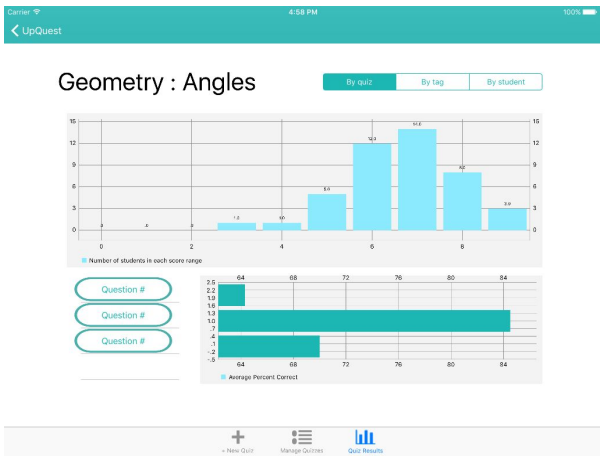
Now the new quiz is in quiz repository

Hard: The teacher checks the student's progress

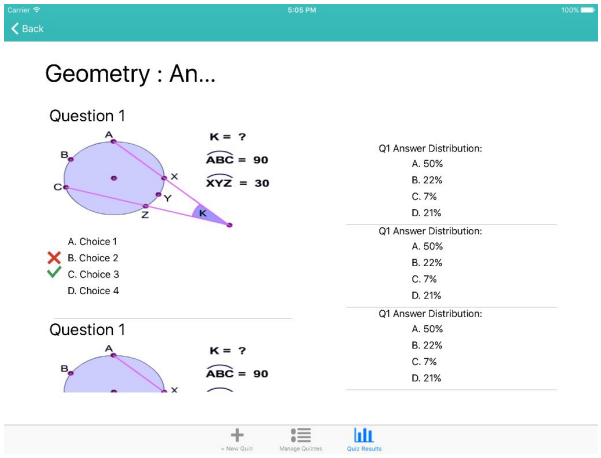
We chose this task as the hard one because there are many ways to present the data and it not only entails teachers checking the results, but also interpreting and making use of the information.



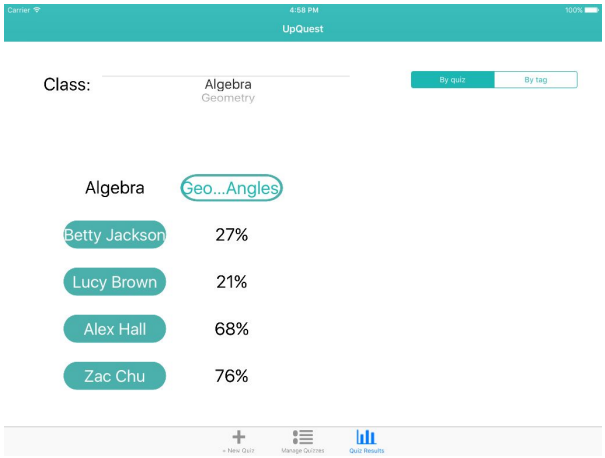
Click on a quiz to see the results

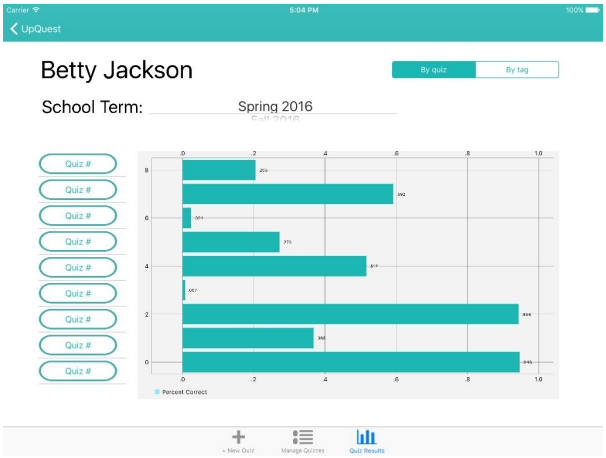


Click on a question to see the answers



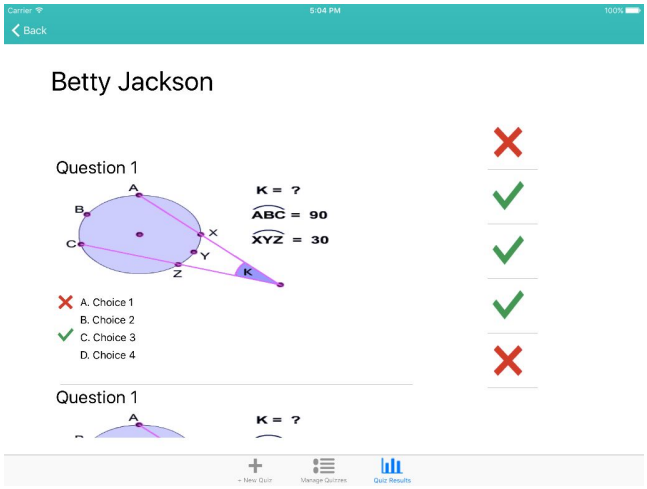
Answer distribution of questions in a quiz





Click on a student to see her overall progress

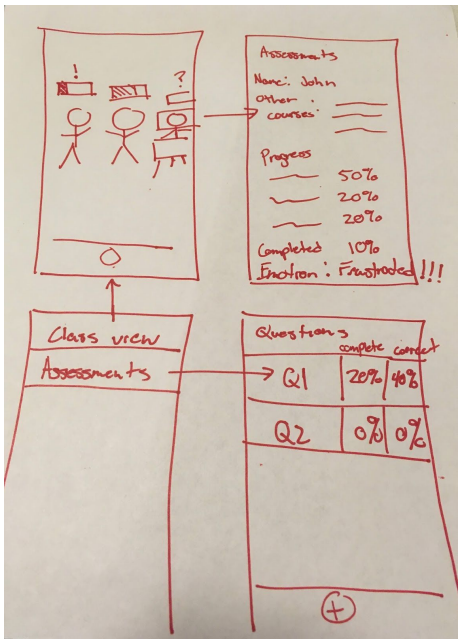
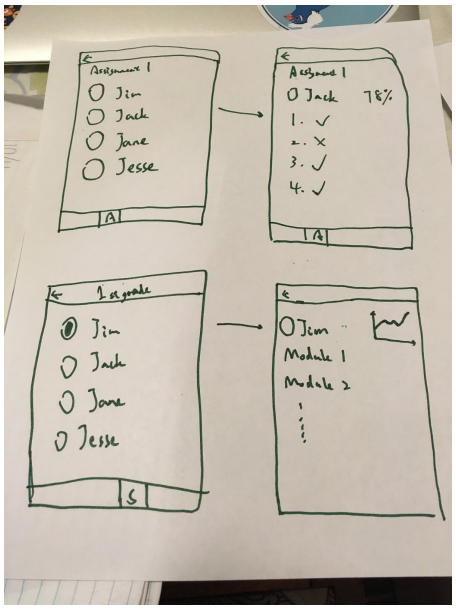
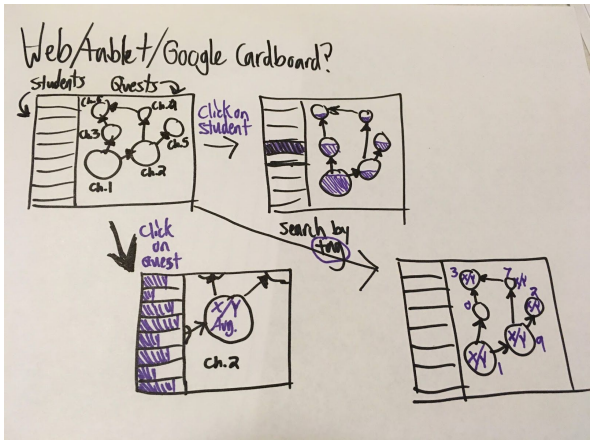
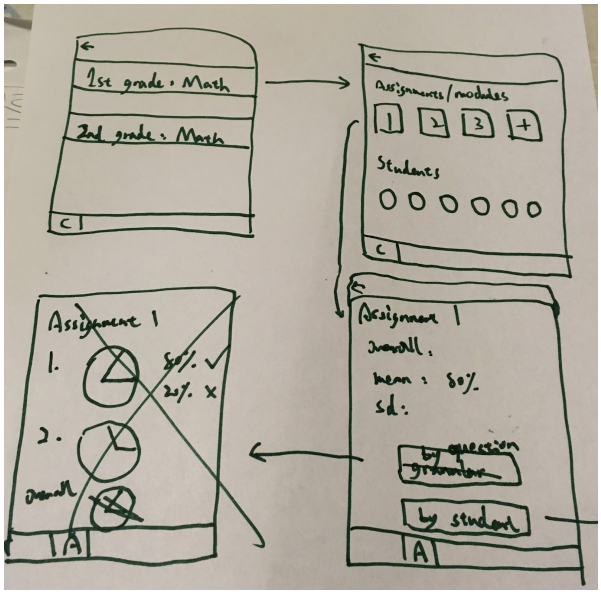
Click on a quiz to see her result

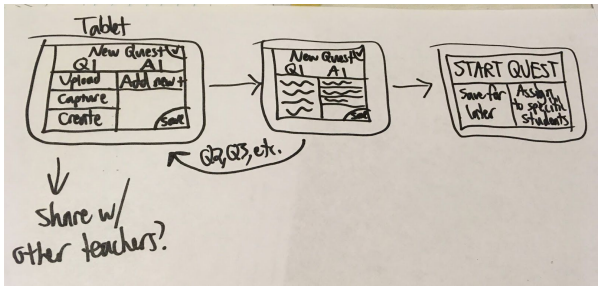
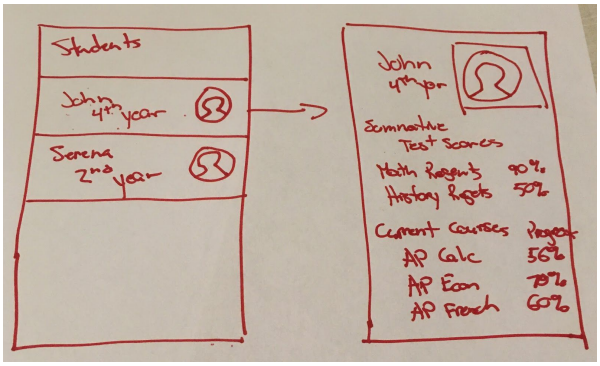


Detailed quiz results for a student's one quiz

Design Evolution

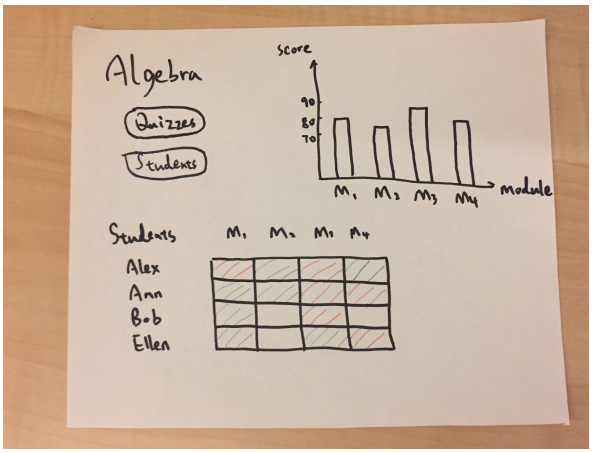
Initial sketches

<table>
<tr>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td>Concept Sketch Series #1 : Augmented reality capability for iphone</td>
<td>Concept Sketch Series #2 : Teacher iphone student view</td>
</tr>
<tr>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td>Concept Sketch Series #3 : Concept maps on a tablet display for modules</td>
<td>Concept Sketch Series #4 : Teacher iphone quiz/module view</td>
</tr>
</table>

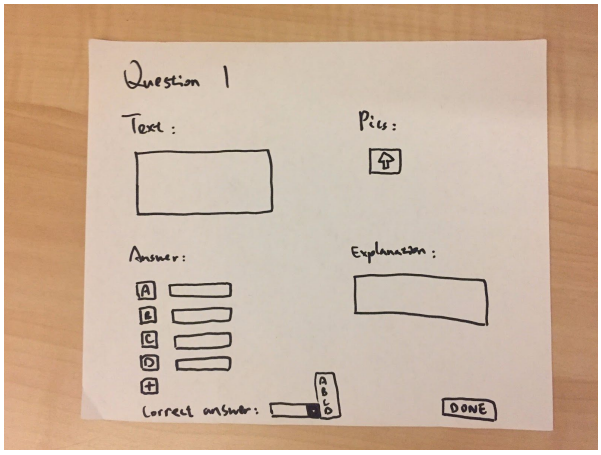
	
Concept Sketch Series #5 : Teacher tablet photo quiz importer	**Concept Sketch Series #6 :** Teacher iphone student profile view

Above are the initial sketches we made. At this point we had both teacher and student's view. We decided to make a web app on teachers' end and an iPhone app on students' end because bigger screen makes it easier to display data to teachers, while mobile app is more convenient for students to take quizzes anytime and anywhere they want.

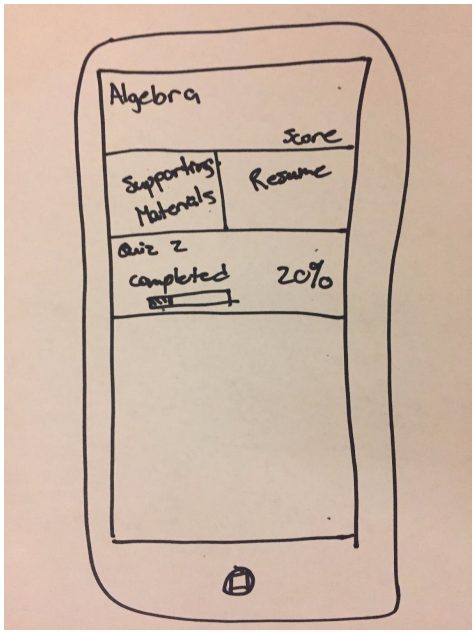
Low-fi prototype



Quiz results view



New quiz view

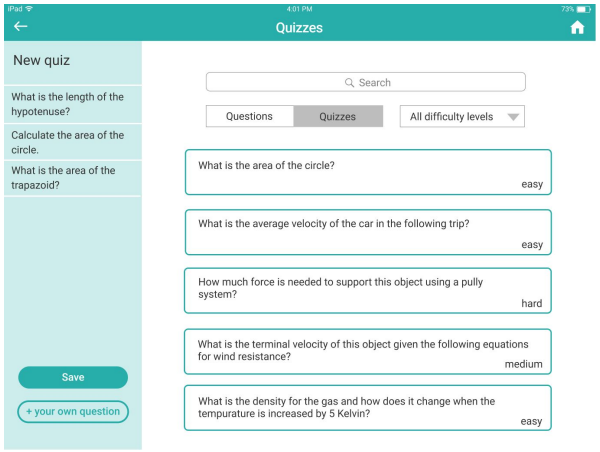


Student view

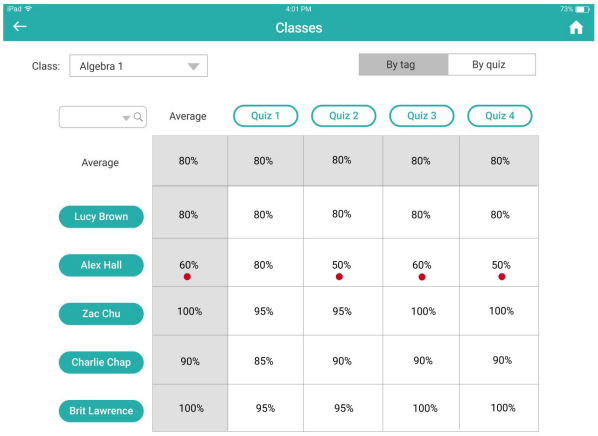
In our low-fi prototype teachers use our web app to create quizzes by manually adding questions, students answer the questions on their phones, and then teachers will see the results. After our user tests with 3 high school teachers, we found out that manually adding quizzes is too much work for them, so we added a crowd-sourced quiz bank, which allows teachers to create quizzes efficiently. They also suggested that action buttons could be made more clear.

We decided to take out the student view and only focus on teacher's side, which would be more realistic for this project's scope. Thus in order to meet the course requirement we had to change teacher's app from web to tablet app.

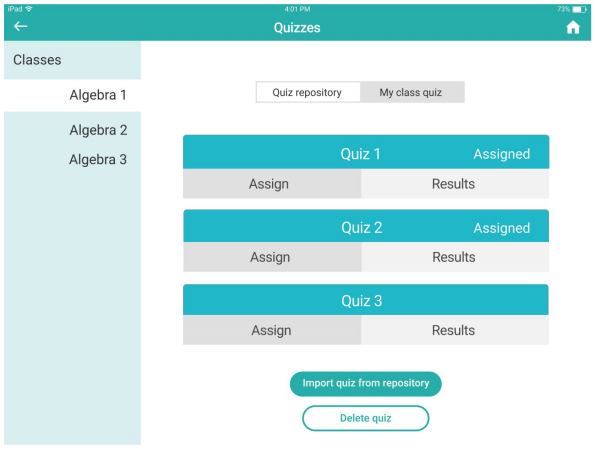
Med-fi prototype



Quiz creation view



Quiz results view

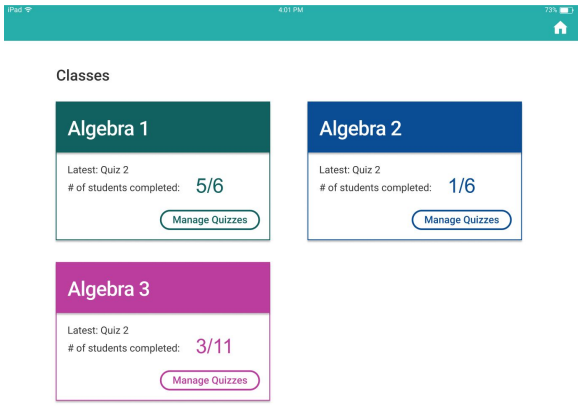


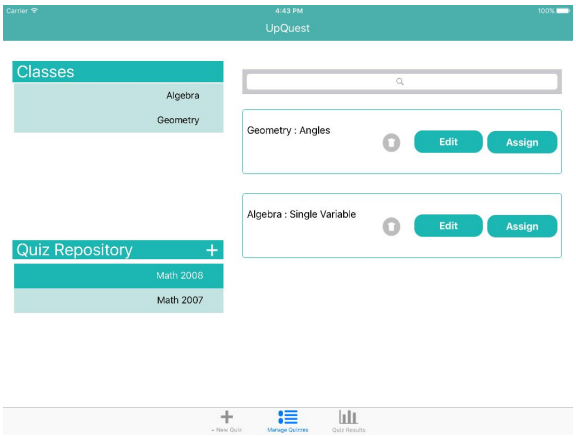
Quiz management view

Major Usability Problems Addressed

Level 3 Heuristic violations:

1. Tasks not delineated clearly enough from the beginning
   [H2-1: Visibility of System Status]



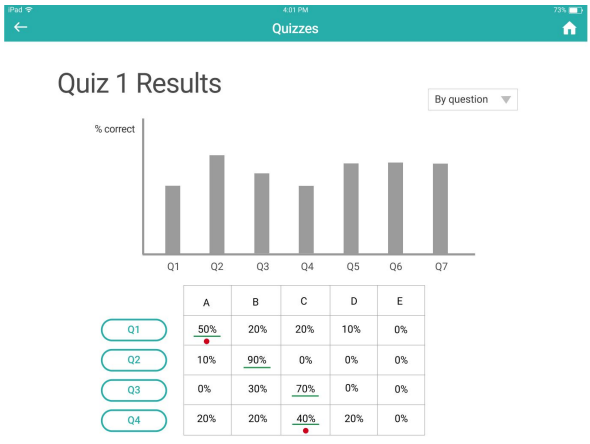


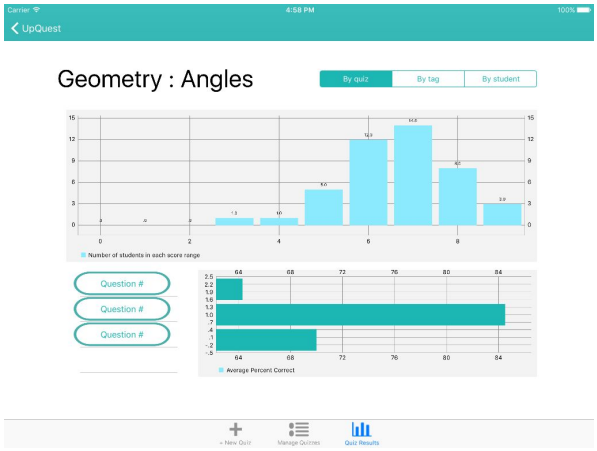
Before *After*

Rationale: in the older version we had a landing page as a dashboard. Creating and managing quizzes were wrapped by the same button ”Manage Quizzes”, and tapping on the tiles of classes would lead to quiz results. To switch between tasks, users have to tap on the home button in the top right corner. In the latest iteration, we took out the dashboard and added a tab bar at the bottom, which is always present. It separates the 3 tasks clearly and makes it easy to switch tasks.

2. Quiz results page hard to decipher. “ABCDE” and colored shapes are confusing.

   [H2-2: Match between system and the real world]
   [H2-8: Aesthetic and Minimalist Design]

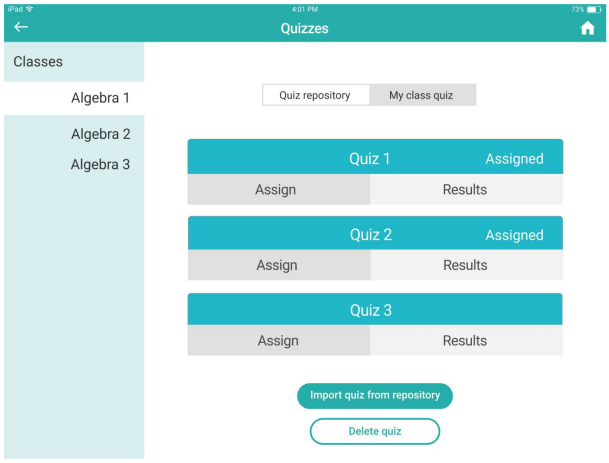


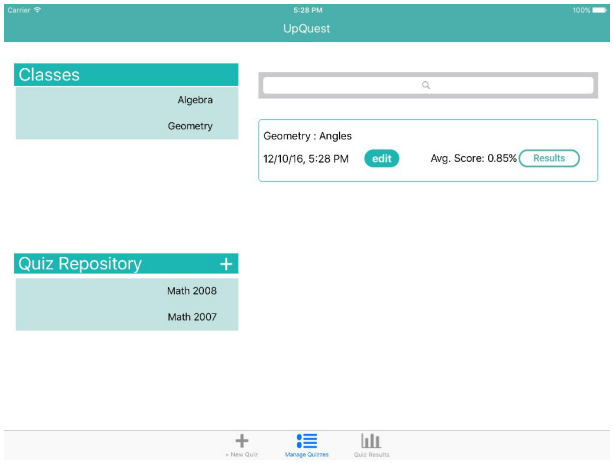


Before *After*

Rationale: the next 2 violations both address on the confusions around quiz results page. In the older version, we had a histogram of the correction rate for each question and a chart that shows answer distributions for each question. Green lines under the percentage signal the right answer, while red dots signal that the percentage is under 60%. In the latest iteration, we made a huge change. We now have a histogram of student scores to show the overall performance of the class. Right under it are bar charts representing the correction rate of each question/tag/student. Tapping on the buttons left to the bars lead to the details of the question. We believe this way we avoid the information overload and unnecessary repetition that was present in our older prototype.

3. No way to edit assigned quiz
   [H2-5: Error Prevention]





Before *After*

Rationale: in our previous version, there's no way to edit a quiz once it is assigned. We added an "edit" button in the latest prototype, which allows one to edit the due date of an assigned quiz. But to edit the quiz itself one still needs to go to the quiz repository, because the same quiz can be assigned to different classes, changing it in the repository would be more efficient.

Other changes:

1. We incorporated the quiz repository and assigned quizzes into one page. On the left side the teacher can see all the classes he's teaching and all the quiz repository folders he has. This change allows users to see the parallel relationship between repository and assigned quizzes, and allows a quick overview of all the classes and repository folders at first glance.
2. In the previous version, to assign a quiz the teacher needs to import it from the quiz repository and then assign. Now he can assign a quiz directly from the quiz repository. We believe it streamlines the experience of assigning quizzes.
3. To delete quizzes from the quiz repository, we used to have a delete button at the bottom. It was not clear how it functioned. Now we have a trash can icon for each quiz.
4. To check individual student's progress, we used to have a histogram and tiles with colored answers. Now we only use bars to represent the student's scores in the quizzes, so that we have room for buttons. Tapping on the buttons on the left side leads to details of the quizzes. This way we reduce information overload and repetition.

Prototype Implementation

What tools were used?

We programmed our app in the Xcode 8.0 environment using Swift 3. The Xcode environment was very useful in mapping directly between our figma storyboard implementation to the program storyboard built into the environment. Though the drag and drop method used to link visual elements to their respective controllers was initially unintuitive coming from a traditional programming background, it proved useful in the end. One difficulty with the storyboard however was the difficulty in collaborating. Since the source code of the storyboard is not user readable, the merge conflicts with this file were difficult to resolve.

Additionally the simulator in the environment was useful in quickly building the application to visually test if the elements were working as expected. In addition to the native platform we used the charting library to generate graphs used for student

performance visualizations. The library was helpful in not having to program the graphing visualizations at a low level and just focusing on the content.

Hard coded data and wizard of oz techniques

We hard coded all of the student data, quiz questions, and structure in the back end. If this application were to be deployed the student data would be received from a student client application and the quiz questions would be pulled from a back end database of quizzes on our servers. Additionally we hard coded the images which were used in both the detailed question view and the detailed student answer views. There were no wizard of oz techniques used.

What is missing and what might you add in the future?

The search functionality as well as the tagging functionality used to support it is missing in this prototype. While the UI elements are there, the backend support is missing since this is primarily a design prototype. The editing functionality of assigned quizzes is not yet implemented, either.

Summary

Through needfinding, ideation, iterative prototyping, we built UpQuest in 10 weeks. It is by no means perfect, but it is now a working prototype that offers multiple complex functionalities. With a user-centric approach, we developed an iPad app that allows teachers to better understand their students' progress and custom their teaching accordingly. Our crowd-sourced quiz bank, tagging system, and data visualization tools are what make the app suitable for this purpose. Hopefully we will continue iterating the app to make it more useful and user-friendly.