

Huh?

Real-Time Student Feedback for Professors

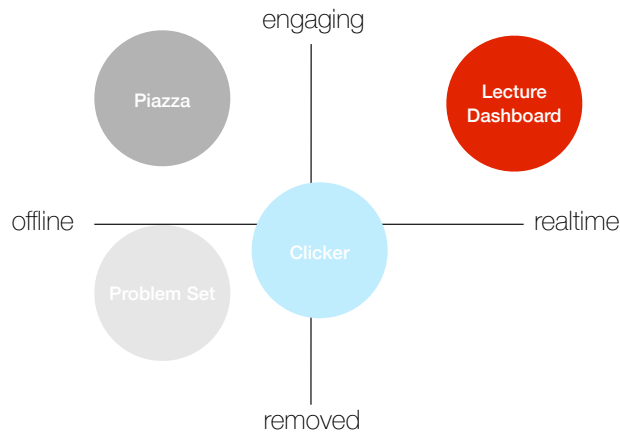
Idea

Lecture Dashboard is a web application that provides a simple way for lecturers to gather feedback from students in realtime by providing the lecturer with qualitative and quantitative data. Students will be able to mark that they are confused and provide information about what they are confused about. Lecturers will be able identify shared places of student confusion and will be able to effectively course-correct a lecture.

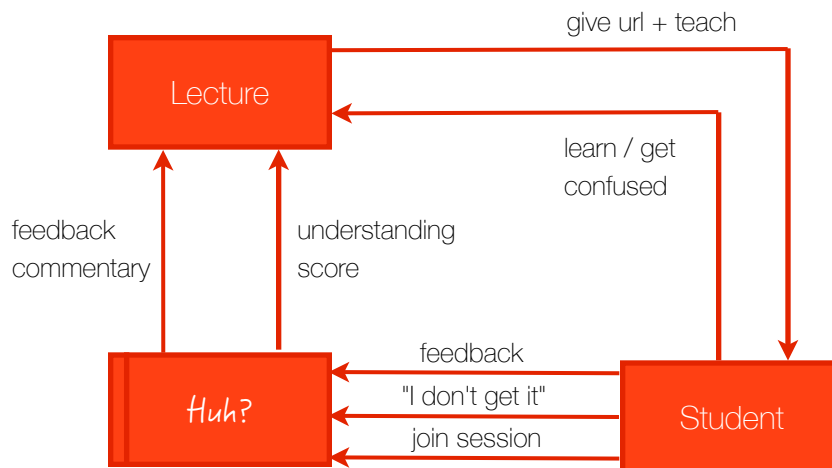
Purpose

The purpose of LectureDashboard are as follows:

1. **Provide the professor with data about the classroom.** Currently, many solutions require lecturers to pause lecture to get feedback from the students. Additionally, students do not feel comfortable telling the classroom that they do not understand a topic. *Huh?* allows students to give anonymized feedback in real-time without pausing the lecture.
2. **Reduce student confusion during lectures.** Often times, when a student is confused by a concept, it results in a long period of confusion because lecture concepts progressively build. *Huh?* empowers students to take an active role in their learning, and stop their confusion by requesting clarification.
3. **Improve the quality of lectures for both professors and students.** *Huh?* is a mutually beneficial system that aims to equip students with the ability to let their professors know in realtime when they are not following. Lecturers will use this feedback to get a better sense of what topics confuse students, which improves the quality of lectures for both professors and students.



Context Diagram



Concepts

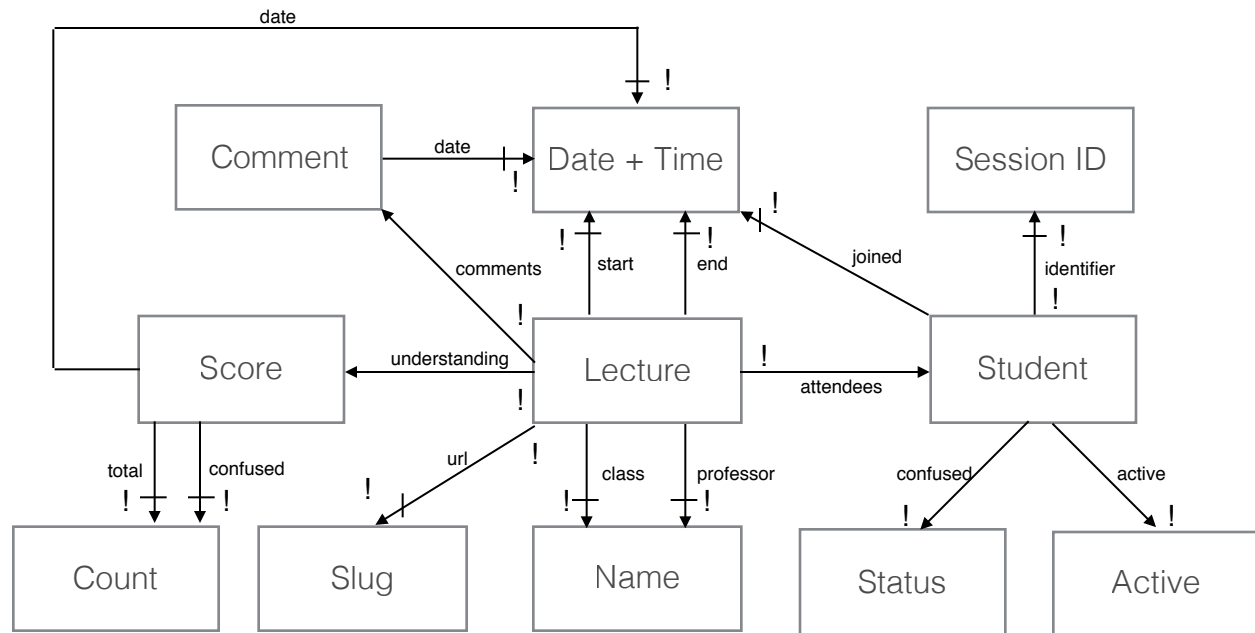
Understanding Score: a quantitative score inversely proportional to student confusion that is displayed to the professor to give them a sense of understanding at a given time for the lecture

Feedback Commentary: comments and questions by students listening in real-time that are displayed on the lecturers dashboard screen in-line with the graph

Lecture Session: a url generated and provided by the lecturer for students to ask questions and indicate their understanding status in realtime

Student Status: toggle switch adjusted by the student to indicate if they are confused or not at the current time

Data Model



Security

Injection Attack: Stringify and sanitize all database input to protect against MongoDB injection attacks

Cross-Site Request Forgery: Use CSRF tokens as hidden fields in our forms to ensure all requests are user initiated and legitimate.

Javascript Injection: Using Handlebars will protect our site from this by escaping any injected scripts

Undesired Traffic: The teacher will provide the url at the beginning of Lecture. This ensures only the students attending the lecture have access to the Huh? Lecture.

Authentication: Our scheme requires no authentication and no actual login, so there are some security considerations about abuse. To prevent a user from hitting the "I don't get it" button thousands of times, we will use a cookie to track a user who has visited the page and store them for persistency. If a user erases their cookies, they will automatically be logged out, so even if they hit the "I don't get it" button again, their vote will only count once. A malicious user could potentially open up multiple browsers and vote multiple times. We could check MAC or IP addresses to prevent this misuse, but the implementation complexity is not worth it. Additionally as we intend to support mobile browsers, we cannot use MAC addresses as this is made unavailable in iOS8.

User Interface

Create Lecture

A Web Page

http://huh.mit.edu/

Create Lecture

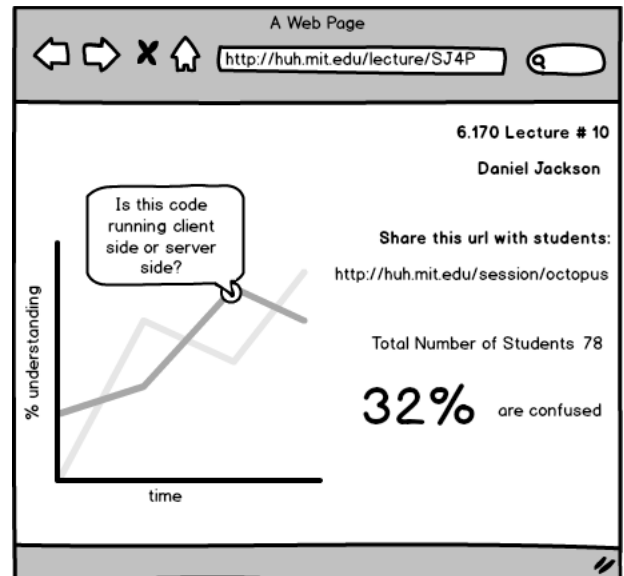
Lecturer Name

Class Name

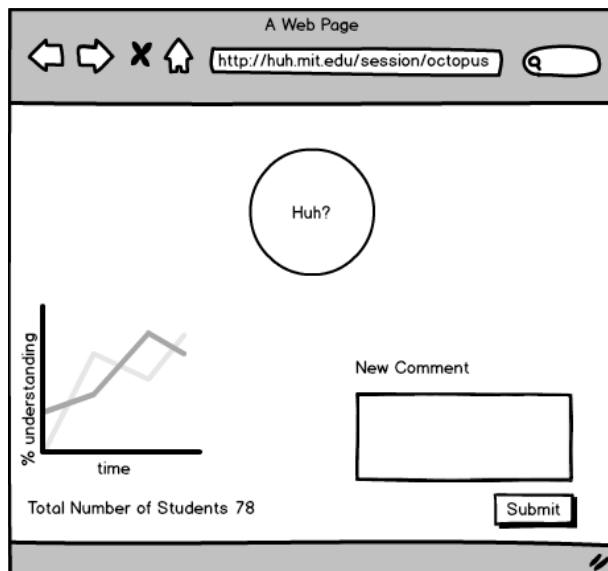
Start Time

End Time

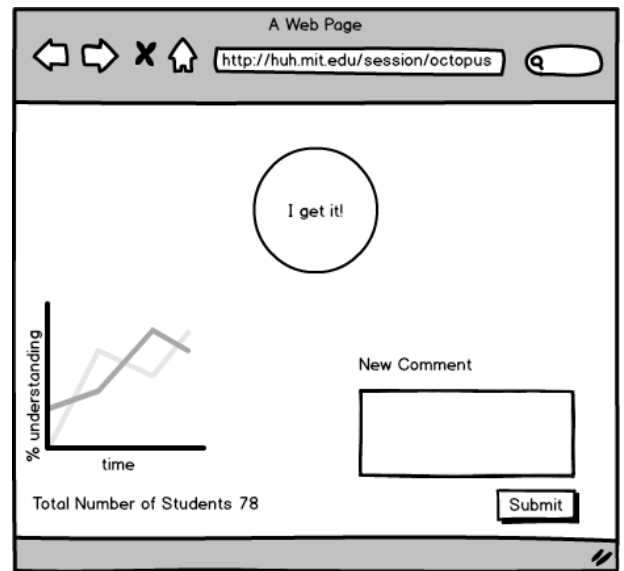
Lecture Dashboard



Student Dashboard (Not Confused)

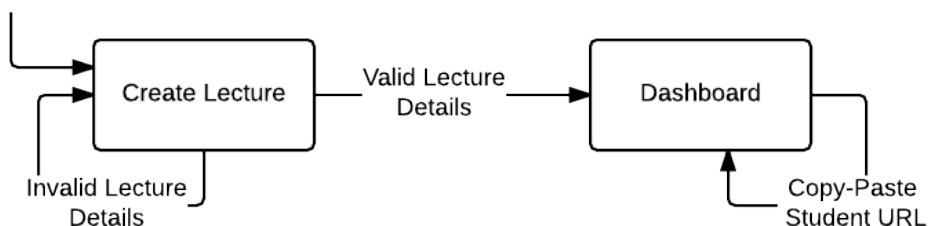


Student Dashboard (Confused)

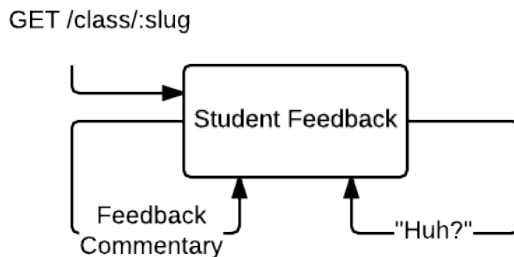


Lecturer Flow

GET/



Student Flow



Design Challenges

Should we include the ability to comment?

- Yes, it allows students to give more feedback on the material.
- No, it introduces design risks for how questions are represented to the students and professor, which may lead to more confusion.

Resolution: We believe that questions will be useful to include, but not for the MVP, as we need more time to implement them and test their functionality

Should we model confusion as a state or event?

- State, probably more intuitive, but requires student to change their state back when they are no longer confused.
- Event, more difficult to model the current state of the class, and possibly more confusing for students, but students don't have to worry about maintaining their state.

Resolution: We will use the state model as, although it is something the student has to maintain, it is easier to understand. It will also be more simple for us to implement - the event model has several design risks we would have to solve.

How do we calculate understanding score? (Specifically, for calculating what percentage of students are confused, what do we use as the population size?)

- Have professor enter number of students when creating lecture. The professor knows the number of students enrolled, but lecture attendance is often far lower than enrollment numbers. Using this metric would either require the professor to guess what attendance will be or use a value that will be far too high. However, this would be fairly simple to implement.
- Use number of current student sessions. We may run into issues with skew if we find that users usually only open the site if they are confused.

Resolution: We will use the number of student sessions. We believe it is more likely to give an accurate result. We are not overly concerned with skew for now, and if it is a major problem, it is not difficult to change this.

Do we need user authentication?

- For professors. This would allow them to store data about their lectures in one place and prevent other users from accessing the lecture dashboard page.
- For students. This would allow classes to maintain lists of allowed users and allow students to look back on past data.
- None at all. This is simpler to implement and we don't have any particularly sensitive data or actions within our application.

Resolution: We will not have authentication. We will use a system similar to that used by Doodle or Whenisgood of having an admin link given to the professor and a participant link to be posted for students.

How will we prevent trolls from voting multiple times by opening multiple tabs?

Discussed in Security section