EDUvote Design Doc

Annie Tang, Casey O'Brien, Qui Nguyen, Kathryn Siegel

Overview

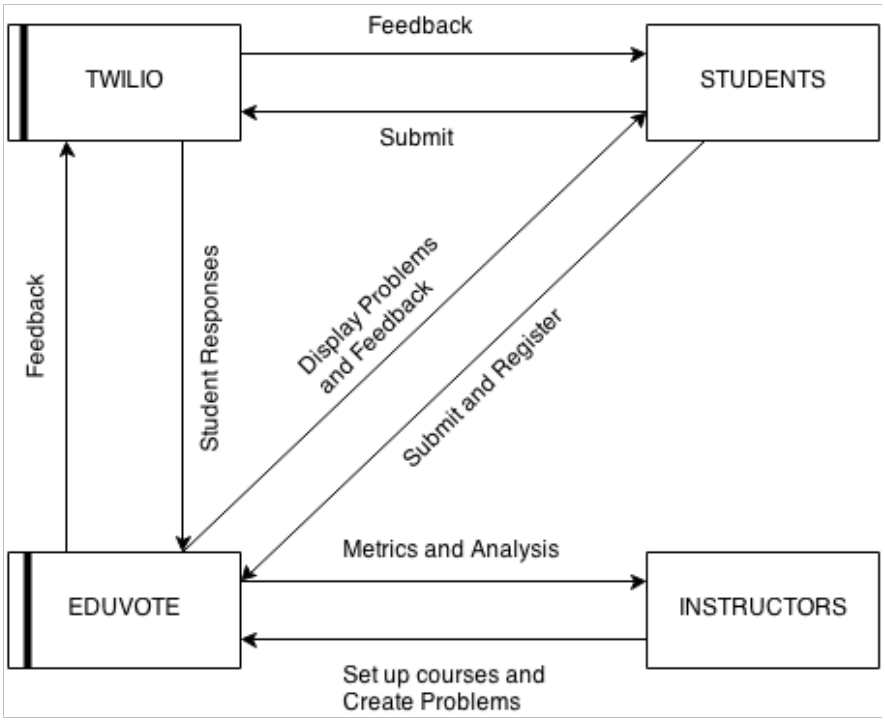
Purpose and Goals *(Annie)*

Description: EDUvote is an app that allows students to submit answers to questions asked in class and enables instructors to track and analyze student responses. Instructors will be able to create multiple choice or numeric questions through our app interface and display them to the class by projecting an app view onto a screen. Then, students can submit responses to those questions through text messages. The app registers these responses by utilizing the Twilio API and has the ability to determine whether they are right or wrong. Feedback is sent using the Twilio API. After responses are collected, instructors can view summaries and analysis of the responses.

Purpose & Goals: To facilitate participation and student/professor interaction within the classroom by creating a voting application with low costs, a simple interface and detailed analytics.

Motivation: Many classes at MIT try to foster interaction and keep track of attendance by posting a short question and having students use Turning Point Clickers to answer them. Clickers work as follows: there is a receiver system controlled by the professor or TA in the classroom. Questions are displayed via projector on a screen, and can be opened to responses or closed by the professor or TA via the receiver system. While a question is open, students submit responses by tuning their clicker to the class channel and clicking the appropriate answer to the displayed multiple choice question. However, clickers are unnecessarily expensive ($40) for both students and instructors, and they give no feedback directly to students after a submission. Additionally, students often cheat by giving other students their clickers. Instructors would also benefit from detailed analysis, which is not readily accessible with current clicker software.

Context Diagram *(Annie)*



Key Concepts *(Qui)*

Student: a person enrolled in a course section, who will need to respond to the questions of that section's instructor.

Instructor: a person who teaches a course section, and will ask questions to the students in the section.

Question: something that students can give a response to. It can be multiple-choice or free response, and it may or may not have correct answer(s).

Response: an answer to a question. It should be either a number or a short string (like A, B, C, etc).

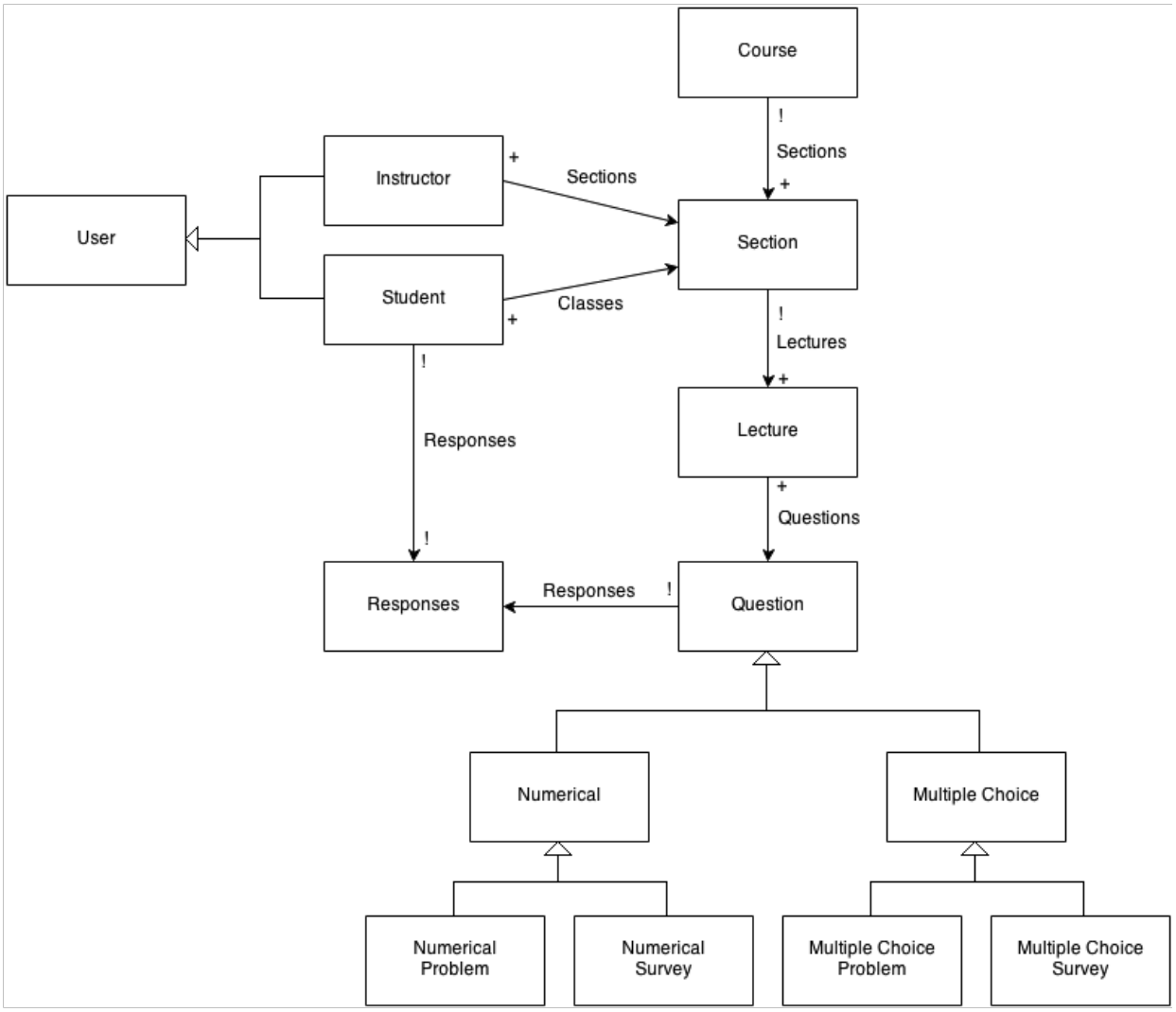
Course: A subject offered in a semester, such as 18.02 (Calculus II) Fall 2011 or 5.111 (Principles of Chemical Science) Spring 2013.

Section: A course may have several sections, where each section meets at a different time. For example, there may be a section of 18.02 that meets at 10AM Mondays and Wednesdays, and another section of 18.02 that meets at 2PM Mondays and Tuesdays.

Lecture: A lecture is a single meeting of a section. Each section has many lectures throughout

the semester, where each lecture is associated with a specific date and time.

Data Model *(Casey)*



Clarifying Notes:

- A *Course* is an overall class, something like 8.01
- A *Section* is a particular offering of the course, something like MW9-11
- A *Lecture* is a particular class session, something like Lecture 4 on 11/7
- A *Question* can be *Numerical* (single number solution) or *Multiple Choice*. It can also be a *Problem* (has a correct answer) or a *Survey* (no correct answer, just gathering information)

Feature Descriptions *(Qui)*

Creation of questions: Instructors will be able to create questions and associate them with lectures.

Display view for questions: Instructors can show questions to their students from the application.

Receive responses from students via text message: Students can send text messages containing their response to a predetermined phone number, and the responses will be received by EDUvote.

Send feedback to students on their responses via text message: After a student submits a response to a question, EDUvote can send text messages containing feedback. This feedback can be customized by instructors.

Receive responses and send feedback directly: Students who cannot or do not want to text message can answer questions directly on the EDUvote website.

Analysis of student responses: Instructors will be able to view a summary of responses for each question. In addition, they will be able to see other analyses such as: responses grouped by student, responses over time, responses across sections, etc.

Display responses over time to students: Each student can view their response to each question in course sections they are registered for, and the correct answer (if applicable).

Security concerns *(Katie)*

Security Requirements:

- Students cannot access the professor-side interface, and cannot create/edit/destroy problems that the professor has created.
- Students cannot view problems that the professor has not yet released to the class.
- Must securely store phone numbers so that only professors and the app can send notifications to students.

Potential Risks:

- Student submits a number that is not his/her own number and helps a friend cheat for the whole semester.
- Hacker gains another user's access token, and utilizes this to send fraudulent answers, thus sabotaging another's grade.
- User sends so many messages that the Twilio queue gets clogged.

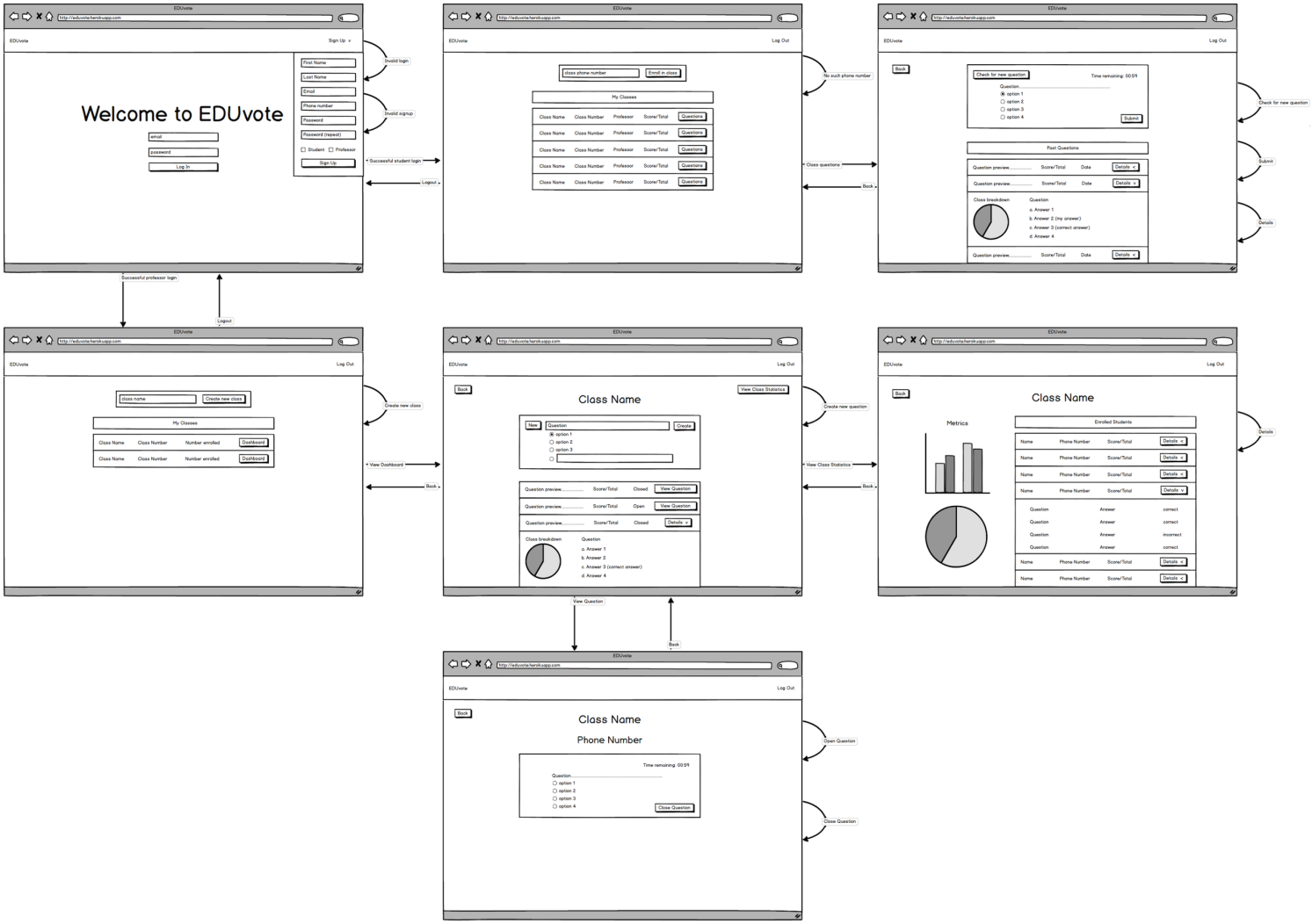
Threat Model:

- Minimal information is stored for each user; thus, this app would not be a target for criminals, etc.
- Verifications are sent via text message to the user upon correct submission, so it would be hard to spoof a sender without the student being notified.
- Only one answer may be registered per open question, thus preventing the Twilio queue from being backed up.

Mitigations:

- Only the users' names, emails, and phone numbers will be stored. This information is not particularly sensitive.
- User access control to prevent viewing of students' information by everyone except the professor of a class for which the student is registered.
- We will rely on Twilio to send text messages and to receive messages
- Registration requires verification via text message, double checking that the phone number registered is correct.
- Implement session timeouts to avoid insecurities involved with copying cookies or leaving a computer open.
- We will use SSL to encrypt cookies, so hackers would need to decrypt a cookie to get a user's access token.
- We will use rails' protect_from_forgery method to further encrypt information being sent by the app.
- The Twilio API requires an authentication token to be passed in each request, increasing the security of the app with regards to sending and receiving messages.

User Interface *(Katie)*



Preliminary Design Challenges *(Katie, Annie)*

1. Problem: Possible lack of cell service in the classroom
    - Considerations:
        - Students could use Wifi
        - Online portal for students would communicate with Twilio via Wifi
    - Solution:
        - We will create an online voting platform that students can use if there is no service in the classroom

2. Problem: Some students don't have phones or have a limited data plan.
    - Option 1: Have students submit answers through an online system

    - Pros: all students have access to computers, especially in TEAL classrooms. Also, this allows for the interface to be more complex and give more feedback.
    - Cons: laptops are more distracting and the interface is more complicated to use. Also, repeatedly taking out and putting back one's laptop in the middle of class disrupts class unnecessarily.
- Option 2: Have students submit answers through their phones
    - Pros: most students have phones, and this is a simple system that has no embellishments (simply text messaging). This solution can be easily implemented via the Twilio API.
    - Cons: Phones also present distractions, and professors may have issues getting students to put away their phones in between questions. Also, not all students have the ability to text or possess phones, so choosing this option would limit the scope of the app to just people with smartphones and data plans.
- Decision: We will implement both options, but favor developing the phone-centric version of the app for the MVP. We reasoned that most people possess phones, which should allow us to adequately demonstrate our app. In the final product, we will create an online interface for voting so that students without phones or texting plans will not be excluded.

3. Problem: Calculating trends over time (i.e. how how a class is improving on answering the correct answers) from data.

- Considerations:
    - Performance might be an issue if we are looking up an entire database table each time the view is called
    - Could make use of middle-tier database caching to achieve high scalability and performance
        - Since we are looking at simple database tables that only store a few values, caching might be too much effort than its worth
    - We could create sophisticated methods in our models that can easily be called upon in our controllers to display different kinds of analytics
- Solution:
    - We will first implement without database caching. Because our tables are relatively simple, performance should not be an issue. However, we will decide after the MVP is implemented whether or not we should do some sort of database caching.

4. Problem: Twilio can only handle 1 text per second for each phone number

- Considerations:
    - Issue of processing speed for collecting submissions
    - Possible lagging between getting the answers and sending feedback
    - Could have multiple numbers for a single classroom, and divide up who texts to which number
- Solution:
    - Depending on how many students are registered for a class, we will generate a

number of different phone numbers that all belong to a single class. Groups of students will text into different numbers so that the speed that Twilio processes texts will not be too much of an issue