COMP 10222 - Emerging Web Technologies

Lab #3 - WebSockets with Socket.IO

Due date: Friday October 23rd at 11:59pm

Worth: 14% of total grade

Marks: 100 marks total

Learning objectives

- Create a real-time web application with Socket.io
- Use WebSockets to co-ordinate interaction between different users

Requirements

We will create a real-time web application that allows teachers to create timed questions, send those questions to students who must answer them in time, collect and display the answer results, and then repeat this process for additional questions.

Create a file named **server.js** to handle WebSocket connections and messages from both the teacher and students. Assume that a

single teacher accesses the application using the URL **http://localhost:3000/teacher**, and assume that *any number of students* access the application using the URL **http://localhost:3000/student**.

Use socket.io to implement both the server and client.

Teacher

The **teacher** should be allowed to create a question and send the question to all students with a time-limit in seconds for the student to answer the question. The teacher should be able to enter a time-limit in seconds using a range form element, with a minimum time of 10 seconds, a maximum time of 90 seconds, and 5 second steps between min and maximum times. The teacher should be able to assign a score associated with answering the question correctly (e.g. 500). The teacher should be able to pick from one of two question types, using either links, buttons or some other method (e.g. a drop-down list). The teacher should be able to send one of these two types of questions to the students:

- **Multiple-choice question** - the teacher should be able to specify a question and answer. The teacher must be able to add as many possible answers to the question as they prefer (perhaps by clicking an add button to add another option). Use checkboxes to allow the teacher to specify which options are the correct answer (the correct answer may be a single option, or multiple options... if they specify multiple options, all options must be selected for the answer to be considered correct).
- **Matching question** - a matching question involves having students find a match for each item in one column to an item in another column. The teacher should be able to specify a description for the matching question. The teacher must be

able to add as many possible matching pairs as they prefer (perhaps by clicking an add button to add another option). Use pairs of input text boxes to allow the teacher to add a matching pair.

After the teacher has entered the data required for the selected question type, the teacher should be able to click a submit button to send the question to the students.

After the teacher submits a question to the students, a timer should be displayed on the teacher page and it should countdown each second from the specified time limit to zero. Once the timer reaches zero, the teacher should have the option of creating a new question (by clicking a button or link). Before the timer reaches zero, the teacher should not have the option of creating a new question (i.e. only one question at a time).

After the teacher submits a question, the interface for creating a question should disappear, and the results of any students answering the questions should be displayed and updated in real-time as each student answers the question. The following information should be displayed in real-time:

- Total number of answers received
- Total number of correct answers received
- Percentage of answers that are correct
- A list of names of all students that answered the question correctly
- A list of names of all students that answered the question incorrectly
- Cumulative score for each student (i.e. any score obtained by answering this question and any previous questions correctly), presented in a list ordered from highest to lowest

This results information should remain displayed until the teacher utilizes the option to creates a new question, at which point the interface for doing so should appear again.

Student

When the student initially visits the application page, it should give them the option to enter their name. After the student submits their name, the student should be able to receive the next question, and the page should only display the text "Waiting for a question..." in a large font (30px or greater).

When the teacher submits a question, all the students who have already submitted their name should be presented with the question on their pages. The "Waiting for a question..." page should disappear, and an interface that allows the student to answer the question should appear. The score for the question should be presented to the student. The question text itself should be presented to the student (again use a larger font, 20px or greater), and form elements should be presented that allow the user to input and submit an answer to the question. Use checkboxes in the case of multiple choice questions. In the case of matching questions, present the first column of items as a static column of items (e.g. in a table, or as text, etc.), and give the users the ability to match items with drop-down lists next to reach item in the first column.

When the question is presented to the student, a timer should also start and be presented on the student's page that counts down from the allotted question time to zero. When the timer hits zero the student should no longer be able to answer the question (i.e. disable or hide the submit button, or somehow otherwise prevent the user from submitting an answer).

When the student submits their answer for the question, the student's page should be updated to indicate whether they answered the question correctly or incorrectly. If the student answered the question incorrectly, they should be presented with the correct answer (the format depends on the question type, but you could just 'correctly fill out' the form elements for a question type) . The student should also at this point be presented with their current, cumulative score.

When the timer reaches zero, the only text "Waiting for a question..." should be displayed again, along with the cumulative score for each student (i.e. any score obtained by answering this question and any previous questions correctly), presented in a list ordered from highest to lowest.

The backend should not contain any HTML code, or work with any HTML code. Only JSON data should be communicated back and forth from the connected teacher and students to the server.

Hints

You can test the application by running multiple browsers (Chrome, IE, FireFox), or multiple tabs (at least in the case of Chrome).

You don't need to do any advanced co-ordination of the question time between the teacher and students. When the teacher submits a question, the time for that question can be sent to the students, and the student page can use that given time to display a timer. Of course this means the timers may be very slightly off due to the time it takes to transmit the time from the teacher to the server and the server to the students. If this were a real-time system for something more serious, where a few milliseconds difference might

really matter, we might invest more time into ensuring even more precision.

The teacher and students don't need to "login". Assume that only one teacher will access the teacher page, and that any number of students may access the student page.

Note that you can use ReactJS, VueJS or other technologies for the front-end if you prefer, but for something smaller like this jQuery or vanilla JavaScript may be easier for the sake of simplicity.

Submission instructions

When you have completed the lab, put your source files in a folder called **lab3.zip** and submit it in the Dropbox.

Include your **package.json** file and your **server.js** file, but do not include your **node_modules** folder.

Statement of Authorship

Copying others people work and claiming it as your own is a serious breach of ethics. If copied work is found, all parties involved will receive a failing grade as per departmental policy. Group work is encouraged but it is expected that you do your own work. If you do, you'll learn the material and feel better for it. Since all work submitted is assumed to be your own original work, you must include the following "Statement of Authorship" in every program file you submit for grading:

"StAuth10065: I John Doe, 123456 certify that this material is my original work. No other person's work has been used without due acknowledgement. I have not made my work available to anyone else."

- Use the exact text above. No substitutions.
- Replace John Doe with your name and the number 123456 with your student ID.
- Place this text as one line as close to the top of each document as possible.
- Include StAuth10065: at the beginning of each statement.
- Failure to include this statement will cause your work to be ineligible for grading.

Marking scheme

Server	20
Teacher	20
Student	20
Timer	10
Questions	30
Total	/100

Labs submitted late will receive a 15% penalty per day. Labs more than 2 days late will receive a grade of zero. For example, a lab submitted at 11:01pm on the due date that was marked 80/100

would receive a mark of 80 - 15 = 65/100. A lab submitted at 11:01pm the day after the due date that was marked 65/100 would receive a mark of 65 - 30 = 35/100. A lab submitted at 11:01pm two days after the due date would receive a mark of zero.