FINAL ASSIGNMENT COURSE PHASE DWA

the Quizzer

The Quizzer is a web application that can be used in bars, sports canteens and maybe even prisons to play quizzes as a team. A pub quiz, basically.
A typical use in a bar would look like this:

- Players are placed in *teams*. A typical team would consist of anywhere between three and eight players, this is not critical to the application, however.
- A *Quizz Night* typically has two to six teams participating.
- A Quizz Night consists of multiple *rounds*, each round contains twelve questions, chosen from three *categories*.
- Besides the teams there is also a Quizz Master who selects questions, approves or disproves answers and keeps the mood up using humour and enthusiasm.
- Every team shares a table and uses one team member's smartphone to submit answers.
- The Quizz Master uses a tablet to host the game.
- Finally, a large screen (e.g. a beamer) is used to display everyone's score, the current question and the teams that have answered this question.
- The Quizzer's questions require short answers. They are not multiple choice questions, but don't require long sentences either. For example: “What is the capital of Peru?” or “In what year did the Netherlands last win the world championship Clay Pigeon Shooting?”.
- The Quizz Master decides when a Quizz Night is over. When it is, the application selects the winners based on the results of all played rounds.

THREE SINGLE PAGE APPLICATIONS

The Quizzer actually consists of three SPAs:

1. **The Team app**
   This SPA runs on the smart phones of the teams and can do two things:
   - Apply as a team for the Quizz Night.
     Part of the application process is providing a team name. Your team then needs to be approved by the Quizz Master.
   - Display the current question and allow the team to enter and submit an answer.

2. **The Quizz Master app**
   This app runs on the Quizz Master's tablet. With it, he or she can perform these tasks:
   - Start the Quizz Night and open it for applications.
   - Approve or reject team applications.
   - Start the Quizz with all the teams that have applied.
   - Start a Quizz Round (that has twelve questions) by selecting three categories (out of a list) and pressing the "Start Round" button, for example.
   - After a Quizz Round, decide whether to play another round or to end the Quizz Night.
   - Select the next question. This is an interesting screen that will be expanded upon under the *User Interface* header.
   - *Start* the selected question by pressing a button. Only then will it show on the Score Board and each Team's app. From that moment on, teams can submit their answers until the Quizz Master closes the question, again by pressing a button.
   - Read the answers that the teams have submitted. This way the Quizz Master can decide whether he or she allows an answer that has a spelling mistake, for example.

3. **The Score Board app**
   This app does not have any interaction, but shows in real time:

- The progress: how many rounds have been played, how many questions we're into a round.
- The team names with their scores in 'Round Points' and the number of correctly answered questions per round.
  Round Points are awarded like this: After each round of twelve questions, the team that has the most correct answers is awarded 4 Round Points (RPs). The next best team is awarded 2 RPs and the third best team is awarded 1 RP. All other teams are awarded 0.1 RPs for their effort and company.
- When a question is in progress (it has been started and is not yet closed, teams can still submit answers) the following information:
  - the question;
  - the category of that question;
  - which teams have submitted an answer, but not the answer itself.
- When a question is closed, all the team answers are displayed. As soon as an answer is approved or rejected, this is also displayed and the team scores are updated.

TECHNICAL

The Quizzer will use Mongo, Express, React and Node (MERN stack), supplemented with the WebSocket protocol (the use of Redux is optional, and will result in a higher grade). All three Single Page Applications will be served by the same Node.js/Express server.
They are web apps that run in a phone or tablet's browser. They are not to be native Android or iOS apps that could be distributed through the Google Play Store or the App Store.
The three SPAs use web sockets to communicate in real-time. Examples of this real-time communication are:

- when the Quizz Master *starts* a question, it immediately appears on the team phones and the score board;
- when a team submits an answer it will immediately be displayed on the Quizz Master's tablet, and the score board will show that that team has submitted an answer.

- When the Quizz Master approves an answer, the score shown on the score board will immediately be updated.

The server can support multiple simultaneous Quizz Nights that might be held in different pubs, for example. To prevent teams from applying with the wrong Quizz Night (being held in a different pub) the teams need to enter a simple password in the application process provided by the Quizz Master.

USER INTERFACE

You don't have to design a polished UI, but do think about which screens the different apps are going to need and what they are roughly going to look like. Even the use of Bootstrap is optional, but definitely worth the effort.
A nice visual design can earn you a small bonus if the app already meets all the requirements in this document.

One thing is especially important to consider. It is important that the Quizz Master has some freedom in choosing the next question, to cater to his or her audience or to the mood of the moment. Design your own UI for the Quizz Master app that allows a Quizz Master to choose from a few questions from any of the three categories that round.

SMALL REQUIREMENTS

This is the incomplete list of small requirements:

1. Any question can appear only once per Quizz Night. When a question is done, it will not appear in the suggestions for the Quizz Master again.
   (Categories *can* appear more than once per Quizz Night.)
2. Team names must be unique; no two teams can have the same name in one Quizz Night. Team names can not be empty.
3. After a team has submitted an answer, they can change their mind and submit another answer, providing the Quizz Master hasn't closed the question yet. The answer submitted last counts.
4. Empty answers will be ignored. If a team inadvertently submits an empty answer, it will not show on the score board or for the Quizz Master's.
5. When the Quizz Master rounds up the Quizz Night, the score board will show the teams that placed first, second and third, measured by the amount of Round

Points. This final screen heavily emphasises the team that places first.

This list of small requirements is probably incomplete. If you encounter a situation where the expected behaviour is not described by this document, keep the following guide lines in mind:

- Whenever there is one straightforward solution that you can implement within a reasonable time frame, go for it.
  A “straightforward” solution should be a solution that helps the application reach it’s goal (for everyone to have a good time) and is user friendly.
- If there are multiple straightforward solutions available that are all implementable within a reasonable time frame, pick the one you like best.
- If none of the solutions you thought up contributes to a good time or to the usability of the apps, present the problem to one of your teachers.
- If the only viable solutions look like they require a lot of time, please consult a teacher.

TECHNICAL REQUIREMENTS

Your code base should include the following:

- Appropriate data validations in your Mongoose models;

You are not required to write any automated tests.

HINTS

1. Recommended WebSocket architecture

Deciding which part of the communication should be implemented using WebSockets and which part using plain old HTTP is by no means trivial. In theory you could build your entire application using only WebSockets.

The recommended approach is to implement all communication involving Quizzer data using HTTP. Whenever the server has new data, use WebSockets to notify the clients and let the client make an HTTP request to fetch the data.

This architecture is by no means the most efficient solution, but it makes it much easier to structure your app. Exclusively using WebSockets can easily lead to a nasty mess.

2. Cooperation and merge conflicts

When working together on a shared code base, merge conflicts are a part of life. You can mitigate the risk of this annoyance by defining clearly separated chunks of functionality and avoid working on the same chunk at the same time. An example of two chunks is the React- and the Express-part (but you could also define smaller chunks).

Try to merge your work often (at least daily). If you postpone merging until the end, you won't suffer any conflicts until the final day. However, the merge conflict you'll face then will probably be too big to solve before the deadline.

EXTRA FEATURES

When you're confident that you've implemented all of the aspects mentioned in this document you can consider adding a few extra features. We award bonus points for these extra features, granted that your application would get a passing mark without them.
Feel free to add your own functionalities or rules. Here are a few ideas:

- *The team selfie* — Use the HTML5 API to access the phone's camera and allow the team to take a selfie that they can send with their team application. The photo can then be shown on the score board.

- *PhoneGap* — It's a lot easier than you think to wrap the client side part of an SPA in a native Android or iOS app. A tool that is often used for this purpose is called 'PhoneGap'.
- *Funny badges* — Allow the Quizz Master to send funny or encouraging badges to teams whenever he or she wants. Examples of this could be badges saying "Funny answer", "Awkward typo", "Fast and furious", etc.
  The badges don't need to count towards the final score.
- *Thinking time* — Instead of having the Quizz Master manually close a question, you could implement a clock that starts counting down as soon as the question has started and closes the question when it reaches zero. The scoreboard could show this clock in real time.
- *History* — The app remembers the teams, allowing competition and rivalry that spans multiple Quizz Nights.
- *Automated test* – Mongoose models and/or Express routes are automatically tested by test scripts that make use of Mocha (or any other framework).