Assessment 3 - ReactJS: BigBrain

0. Change Log

- 04/04: Please note, if you are having trouble with react-router-dom I suggest upgrading react-scripts `npm install react-scripts@5`
- 10/04: Submit command updated; multiple choice definition expanded on

1. Background & Motivation

In March 2022 you and your friends pitched a startup idea to produce *An innovative lightweight quiz platform for millenials* that will *revolutionise the secondary and tertiary education market for years*. You pitched this solution in the form of a web-based application, and called this quiz application **BigBrain**.

A week later you received a tentative $50,000 investment from an Angel Investor pending you producing a working MVP of the application.

Shortly after you discussed the functionality and feature set with your friends, and wrote out a RESTful specification / interface together so that you can split up the front-end and back-end work between the group.

You (and optionally another one of your friends) decided to work on building the front-end. You wrote a list of requirements and functionalities your frontend should adhere to (described in `section 2`). You also decided to complete this application in `ReactJS`, a declarative framework for building single page applications. This front-end will interact with a Restful API that your team members are producing, based on the pre-defined interface.

Because your MVP is only going to be demonstrated once, your team considers it imperative that your front-end is thoroughly tested.

To satisfy modern tastes and expectations you have also decided to ensure that the UI/UX and Accessibility standards are very high.

This assignment is the process you building the front-end for that MVP to the standards described. This assignment is closely modelled off the popular game kahoot. If you're not familiar with the game, we would recommend spending the time to try it out so that you can get a feel for how this application may function.

2. The Front-end (Work to do)

Navigate to the `frontend` folder and run `yarn install` to install all of the dependencies necessary to run the ReactJS app. Then run `yarn start` to start the ReactJS app.

A series of features below need to be implemented in your ReactJS app to operate in conjunction with the backend (section 3).

2.1. Feature 1. Admin Auth (12% for solo, 10% for pairs)

2.1.1. Login Screen

- A unique route must exist for this screen
- User must be able to enter their `email` and `password`.
- If the form submission fails, a reasonable error message is shown
- A button must exist to allow submission of form

2.1.2. Register Screen

- A unique route must exist for this screen
- User must be able to enter their `email` and `password` and `name`
- A button must exist to allow submission of form

2.1.3. Logout Button

- On all screens that require an authorised user, a logout button exists.
- This logout button, when clicked, returns you to the login screen.

2.2. Feature 2. Admin Creating & Editing a Game (24% for solo, 20% for pairs)

2.2.1. Dashboard

- A unique route must exist for this screen
- A dashboard of all games is displayed, where each game shows the title, number of questions, a thumbnail, and a total time to complete (sum of all individual question times)
- Each game listed should have a clickable element relating to it that takes you to the screen to edit that particular game
- A button exists on this screen that allows you to create a new game, provided a name for the game. Clicking it creates a new game on the server and adds another visible game to the dashboard. ** A button exists on this screen that allows you to delete a particular game.

2.2.2. Edit BigBrain Game

- A unique route must exist for this screen that is parameterised on the game ID
- This screen allows users to select the question they want to edit
- This screen allows users to delete a particular question, or add a new question

2.2.3. Edit BigBrain Game Question

- A unique route must exist for this screen that is parameterised both on the Game ID and the question ID
- Editable items on this page include:
    - The question type (multiple choice, single choice)
        - Single choice questions have multiple answers the player can guess, but only one is correct
        - Multiple choice questions have multiple answers the player can guess, but multiple are correct and they must select all correct ones
    - The question itself (as a string)
    - Time limit that users have to answer the question
    - Points for how much the question is worth
    - The ability to optionally attach a URL to a youtube video, or upload a photo, to enhance the question being asked).
    - Anywhere between 2 and 6 answers, that each contain the answer as a string

2.3. Feature 3. Admin Start, Stop, Results of game (12% for solo, 10% for pairs)

2.3.1. Starting a game

- On the dashboard page, add the ability to start a stopped game
- When the game is started, a popup is displayed that shows the session ID of the game as a string
- This session ID should be able to be copied by some kind of "Copy Link" button/element. When this item is clicked, a direct URL is copied to the clipboard. When going to this URL, the users should be given play screen (described in `2.4`) with the session code already pre-populated.

2.3.2. Stopping a game

- On the dashboard page, the ability to stop a started game.
- When the game is stopped, a popup appears that prompts the admin "Would you like to view the results?" If they click yes, they are taken to the screen described in `2.3.3`

2.3.3. Getting the results of a game

- A unique route must exist for this screen that is parameterised on the session ID
- Once the screen loads, it should display the following:
    - Table of up to top 5 users and their score
    - Bar/Line chart showing a breakdown of what percentage of people (Y axis) got certain questions (X axis) correct
    - Some chart showing the average response/answer time for each question
    - Any other interesting information you see fit

2.4. Feature 4. Player able to join and play game (12% for solo, 10% for pairs)

2.4.1. Play Join

- A unique route must exist for this screen
- A user is able to enter a session ID and their own name to attempt to join the session. If succesful, they're taken to `2.4.2`.

2.4.2. Play Game

- On this screen the user is given the current question being asked. This consists of:
    - The question text
    - A video or image depending on whether it exists.
    - A countdown with how many seconds remain until you can't answer anymore.
    - A selection of either single or multiple answers, that are clickable.
- The answer shall be sent to the server the moment the user starts making selections. If further selections are modified, more requests are sent
- When the timer hits 0, the answer/results of that particular question are displayed
- The answer screen remains visible until the admin advances the quiz question onto the next question.

2.4.3. Game Results

- After the final question is answered, a page is displayed showing the key results:
    - The player's performance in each question

2.5. Advanced Features (0% for solo, 10% for pairs)

2.5.1. Game Upload

- For `2.2.1`, when a new game is created, the user can optionally upload a .csv or .json (you choose) file containing the full data for a game. The data structure is validated on the frontend before being passed to the backend normally. You should provide a copy of an example data file in your project repo ()
- If you implement this feature, you must attach an example .csv or .json into your repo in the project folder. This file must have name `2.5.json` or `2.5.csv`. This is so we can actually test that it works while marking.

2.5.2. Lobby

- If a quiz is active, but has yet to move into position 0 (i.e. is still in position -1), then a player lives in a state of limbo. Construct a "lobby" screen that is pleasant and entertaining for users while they await for the quiz to begin.

2.5.3. Past quiz results

- Allow admins to access a page whereby they can see a list of previous sessions for a quiz, and then view results for those previous sessions as well.

2.5.4. Points system

- Devise a more advanced points system whereby a player's score if the product of the time taken to complete a question (i.e. speed) and the number of points a question is worth.You can
- This points system should be explained (in writing) on the results screen for both admins and players.

2.6. Linting

- Linting must be run from inside the `frontend` folder by running `yarn lint`.

2.7. Testing

As part of this assignment you are required to write some tests for your components (component testing), and for your application as a whole (ui testing).

For **component testing**, you must:

- Write tests for different components (3 if solo, 6 if working in a pair)
- For each of the components, they mustn't have more than 50% similarity (e.g. you can't test a "Card" component and a "BigCard" component, that are virtually the same)
- Ensure your tests have excellent **coverage** (look at all different use cases and edge cases)
- Ensure your tests have excellent **clarity** (well commented and code isn't overly complex)
- Ensure your tests are **designed** well (logical ordering of tests, avoid any tests that aren't necessary or don't add any meaningful value)
- (We encourage you to only use shallow component rendering)

For **ui testing**, you must:

- Write a test for the "happy path" of an admin that is described as:
    1. Registers successfully
    2. Creates a new game successfully
    3. (Not required) Updates the thumbnail and name of the game successfully (yes, it will have no questions)
    4. Starts a game successfully
    5. Ends a game successfully (yes, no one will have played it)
    6. Loads the results page successfully
    7. Logs out of the application successfully
    8. Logs back into the application successfully
- (If working in a pair) also required to write a test for another path through the program, describing the steps and the rationale behind this choice in `TESTING.md`

Tests must be run from inside the `frontend` folder by running `yarn test`.

2.8. Other notes

- The port you can use to `fetch` data from the backend is defined in `frontend/src/config.json`
- The data structure of a "question" is open ended - it's not defined explicitly in the backend. Because of this, the backend has 3 wrapper functions defined in `backend/src/custom.js` that it uses to extract meaning from your custom data structure. **You will have to implement these as you build out your frontend**b>.
- For users of typescript, there is an alternatively `.eslintrc` file being collaborated here. Do not change it unless given approval on forum.

3.1. The Frontend

Navigate to the `frontend` folder and run `yarn install` to install all of the dependencies necessary to run the ReactJS app. Then run `yarn start` to start the ReactJS app.

Please note that some properties that the backend takes in are defined as blank objects. These are objects that can be defined by you, as the backend will simply store your object on some routes and then return it to you on other routes (i.e. the backend doesn't need to understand the schema of some objects you pass it). An example of this object is all of the data associated with a quiz.

This approach we've taken is actually designed to make the assignment *easier*, as it gives you control without having to worry about backend architecture.

3.2. The Backend (provided)

You are prohibited from modifying the backend. No work needs to be done on the backend. It's provided to you simply to power your frontend.

The backend server exists in your individual repository. After you clone this repo, you must run `yarn install` in `backend` directory once.

To run the backend server, simply run `yarn start` in the `backend` directory. This will start the backend.

To view the API interface for the backend you can navigate to the base URL of the backend (e.g. `http://localhost:5005`). This will list all of the HTTP routes that you can interact with.

Your backend is persistent in terms of data storage. That means the data will remain even after your express server process stops running. If you want to reset the data in the backend to the original starting state, you can run `yarn reset` in the backend directory. If you want to make a copy of the backend data (e.g. for a backup) then simply copy `database.json`. If you want to start with an empty database, you can run `yarn clear` in the backend directory.

Once the backend has started, you can view the API documentation by navigating to `http://localhost:[port]` in a web browser.

The port that the backend runs on (and that the frontend can use) is specified in `frontend/src/config.js`. You can change the port in this file. This file exists so that your frontend knows what port to use when talking to the backend.

4. Constraints & Assumptions

4.1. Languages

- You must implement this assignment in ReactJS. You cannot use other declarative frameworks, such as AngularJS, or VueJS.
- You must use ReactJS solutions wherever possible, and avoid doing any direct DOM manipulation unless completely unavoidable (check with course staff).
- You can use any CSS libraries that you would like, such as bootstrap or material-ui.
- You are able to use and install any library that is available to install via `yarn install`.

4.2. Browser Compatibility

- You should ensure that your programs have been tested on one of the following two browsers:
    - Locally, Google Chrome (various operating systems) - make sure is latest version.
    - On CSE machines.

4.3. Using code found online

- You may use small amounts (< 10 lines) of general purpose code (not specific to the assignment) obtained from a site such as Stack Overflow or other publically available resources. You should attribute clearly the source of this code in a comment with it. You can not otherwise use code written by another person.

4.4. Other Requriements

- The specification is intentionally vague to allow you to build frontend components however you think are visually appropriate. Their size, positioning, colour, layout, is in virtually all cases completely up to you. We require some basic criteria, but it's mainly dictating elements and behaviour.
- Besides those described to avoid, you may use any other packages available on yarn.
- The use of universal CSS is banned - you must use either CSS libraries (e.g. material-ui) or styled components.

5. Teamwork

This assignment may be completed in a team of two (pair). However, you are also welcome to complete it on your own, if you choose. The groups were organised and coordinated by the course coordinator separately.

If you formed a pair, you will be unable to leave your pair unless under extreme circumstances. You will be assessed together for the assignment.

If your contributions to the assignment are not approximately equal, then the teaching staff may make discretionary calls based on your gitlab history to award different marks to each student.

Please note: Your contributions will be measured based on the lines and commits contributed via gitlab. Please commit via your own machine or account. If you're in a pair, your contributions will not be considered yours if it is your partner who pushes the code to gitlab.

6. Marking Criteria

Your assignment will be hand-marked by tutor(s) in the course according to the criteria below.

Criteria	Weighting	Description
Functionality of the Feature Set + Mobile Responsiveness	60%	• Features implemented that satisfy requirements as outlined in `2.1`, `2.2`, `2.3`, `2.4`, and `2.5` (for pairs). • Features implemented in a mobile responsive way that work on screens as small as 400px wide, 700px high • Responsive design will contribute up to one quarter of the marks of this section • You **MUST** update the `progress.csv` file in the root folder of this repository as you complete things partially or fully. The valid values are "NO", "PARTIAL", and "YES". Updating this is necessary so that your tutor knows what to focus on and what to avoid - giving them the best understanding of your work and provide you with marks you have earned.
Linted Code	5%	• Submitted code is completely `eslint` compliant based on provided eslint configuration file.
Code Style	10%	• Your code is clean, well commented, with well-named variables, and well laid out. • Code follows common ReactJS patterns that have been discussed in lectures
Testing	15%	• Two thirds (10%) of the marks received from complying with requirements in section `2.7` in relation to **component testing** • One third (5%) of the marks received from complying with requirements in section `2.7` in relation to **ui testing** • Describe your approach to testing in `TESTING.md`
UI/UX & Accessibility	10%	• Your application is usable and easy to navigate. No obvious usability issues or confusing layouts/flows. • Your application makes intelligent use of UI/UX principles and patterns discussed in the UI/UX lectures. • Your application follows standard accessibility lessons covered in lectures. • Describe any attempts you've made to improve the UI/UX or Accessibility in `UIUX.md`
(Bonus Marks) Extra Features	5%	• Implementation of extra features that are not included in the spec. • Extra features should be non-trivial, have a clear justification for existing, and show either a form of technical, product, or creative flare. • Any extra features written down in `BONUS.md` in the project folder • Any bonus marks that extend your ass3 mark above 100% will

		bleed into other assignment marks, but cannot contribute outside of the 80% of the course that is allocated for assignment marks • **Expectations placed on solo groups will be half of that of pairs to achieve the same mark.**

7. Originality of Work

The work you submit must be your own work. Submission of work partially or completely derived from any other person or jointly written with any other person is not permitted.

The penalties for such an offence may include negative marks, automatic failure of the course and possibly other academic discipline. Assignment submissions will be examined both automatically and manually for such submissions.

Relevant scholarship authorities will be informed if students holding scholarships are involved in an incident of plagiarism or other misconduct.

Do not provide or show your assignment work to any other person — apart from the teaching staff of COMP6080.

If you knowingly provide or show your assignment work to another person for any reason, and work derived from it is submitted, you may be penalized, even if the work was submitted without your knowledge or consent. This may apply even if your work is submitted by a third party unknown to you.

Every time you make commits or pushes on this repository, you are acknowledging that the work you submit is your own work (as described above).

Note you will not be penalized if your work has the potential to be taken without your consent or knowledge.

PLEASE NOTE: To ensure the originality of your work, we are requiring that you regularly commit your work to git throughout the weeks this assignment has been released. Regular and small commits and critical. Failures to commit regularly (or at minimum, failures to commit in small chunks) may result in allegations of plagiarism

8. Submission

This assignment is due *Thursday 21st of April, 10:00am*.

To submit your assignment, you must complete the following two steps in order:

- Ensure you've pushed all of your code to your gitlab master branch. You can check if you've done this properly by seeing what code is on the gitlab site on your master branch.
- Run the following command on a CSE terminal (SSH, vlab): `$ 6080 submit ass3 [groupname]` where `[groupname]` is the group name you have been assigned on gitlab.

This will submit the latest commit on master as your submission.

It is your responsibiltiy to ensure that your code can run successfully when cloned fresh from Gitlab.

For pairs, only one team member needs to submit.

9. Late Submission Policy

If your assignment is submitted after this date, each hour it is late reduces the maximum mark it can achieve by 2%.

For example if an assignment you submitted with a raw awarded mark of 85% was submitted 5 hours late, the late submission would have no effect (as maximum mark would be 90%). If the same assignment was submitted 20 hours late it would be awarded 60%, the maximum mark it can achieve at that time.