Assessment 3 - ReactJS: AirBrB

0. Change Log

- 30/10: Updated Typescript setup instructions
- 1/11: Updated backend to not overwrite users details when an email already exists on the database.
- 2/11: Added correct late policy as per course policy and lecture
- 15/11: Removing (*)

1. Background & Motivation

In October 2021, following the immense financial success of your messaging platform Slackr, you were invited to several tech talks and travelled all over the country. In your travels, you've had to endure several uncomfortable hotel stays. During these stays you've been struck with another brilliant startup idea for a person to person property renting service **AirBrB**.

You've contacted your developer friends and settled on functionality, feature set and a RESTful specification/interface for AirBrB. You've decided to outsource your back-end to another company and took on the task of building the front-end (optionally with another one of your friends). 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 the team you've outsourced to 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 property renting platfrom Airbnb. If you're not familiar with the service, 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 Task

You are to build a frontend for a provided backend. This frontend shall be build with ReactJS. It shall be a single page application that does not require a refresh for state updates.

Features need to be implemented (described below) in order for your ReactJS app to meet the requirements of the task, and to operate with the backend described in 3.2.

The requirements describe a series of **screens**. Screens can be popups/modals, or entire pages. The use of that language is so that you can choose how you want it to be displayed. A screen is essentially a certain state of your web-based application.

Please note: This assignment, unlike assignment 2, has a lot of functionality available whilst not logged in. Logging in just adds extra functionality. If you're unsure what we mean by this behaviour, you can play around with the Airbnb website for comparison.

2.1. Feature Set 1. Admin Auth (10% for solo, 8% for pairs)

This focuses on the basic user interface to register and log in to the site. Login and registration are required to gain access to making bookings as a guest, leave reviews and to manage your own listings as a host.

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 confirm `password` field should exist where user re-enters their password.
- If the two passwords don't match, the user should receive an error popup before submission.
- A button must exist to allow submission of form

2.1.3. Logout Button

- A logout button, when clicked, returns you to the landing screen whilst being no longer logged in.

2.1.4. Items on all screens

- On all screens, for a user who is logged in / authorised:
  - The logout button exists somewhere
  - A button exists that will take the user to the screen to view their hosted listings.
  - A button exists that will take the user to the screen to view all listings.

2.2. Feature Set 2. Creating & Editing & Publishing a Hosted Listing (16% for solo, 14% for pairs)

For logged in users, they are able to create their own listings (as a host) that will become visible to all other users who have the option of booking it.

2.2.1. Hosted Listings Screen

- A unique route must exist for this screen
- A screen of all of YOUR listings (that you created) is displayed, where each listing shows the:
  - Title
  - Property Type
  - Number of **beds** (not bedrooms)
  - Number of bathrooms
  - Thumbnail of the listing
  - SVG rating of the listing (based on user ratings)
  - Number of total reviews
  - Price (per night)
- Each listing should have a clickable element relating to it that takes you to the screen to edit that particular listing (`2.2.3`).
- A button exists on this screen that allows you to delete a particular listing.

2.2.2. Hosted Listing Create

- On the hosted listing screen (`2.2.1`) a button should exist that allows you to create a new listing. When you click on it, you are taken to another screen that requires you to provide the following details:
  - Listing Title
  - Listing Address
  - Listing Price (per night)
  - Listing Thumbnail
  - Property Type
  - Number of bathrooms on the property
  - Property bedrooms (e.g. each bedroom could include number of beds and their type)
  - Property amenities
- Using a button, a new listing on the server is created and visibly added to the dashboard once all of the required fields have been filled out correctly.

2.2.3. Edit AirBrB Listing

- A unique route must exist for this screen that is parameterised on the listing ID.
- The user should be able to edit the following:
  - Title
  - Address
  - Thumbnail
  - Price (per night)
  - Type
  - Number of bathrooms
  - Bedrooms (incorporate editing of beds as part of bedrooms)
  - Amenities
  - List of property images
- Updates can auto-save, or a save button can exist that saves the updates and returns you to the hosted listings screen.

2.2.4. Publishing a listing

- For a listing to "go live" means that the listing becomes visible to other AirBrB users on the screen described in `2.4` .
- On the hosted listings screen described in `2.2.1` , add the ability to make an individual listing "go live".
  - A listing must have at least one availability date range (e.g. a listing could be available between 1st and 3rd of November and then between the 5th and 6th of November).

    - The way you define the availability ranges is entirely up to you. For example, you could use the following schemas:

```
//Example 1:
availability: [{ start: date1, end: date2 }, { start: date3, end: date4 }, ...];
//Example 2:
availability: [date1, date2, date3, date4, ...];
```

- (Note: If the listing has more than 1 availability range, aggregate them on the frontend and submit them all to the backend in one go when publishing the listing).

2.3. Feature Set 3. Landing Page: Listings and Search (16% for solo, 14% for pairs)

When the app loads, regardless of whether a user is logged in or not, they can access the landing screen. The landing screen displays a number of listings that you as a guest may be able to book (on another screen). We recommend you create some listings (`2.2`) with one user account, and then create a second user account to build/test `2.3` so that you can view their listing as a potential booking option.

2.3.1. Listings Screen

- A unique route must exist for this screen.
- This is the default screen that is loaded when a user accesses the root URL.
- This screen displays a list of all listings (rows or thumbnails). The information displayed in each listing is:
    - Title
    - Thumbnail of property (or video if advanced)
    - Number of total reviews
    - (any more information you want, though that's optional).
- In terms of ordering of displayed listings:
    - Listings that involve bookings made by the customer with status `accepted` or `pending` should appear first in the list (if the user is logged in).
    - All remaining listings should be displayed in alphabetical order of title.

2.3.2. Search Filters

- On this listings screen, a search section must exist for the user to filter via search parameters. You are only required to be able to search by one of the parameters described below at a time.
- The search section will consists of an input text box:

    - The input text box will take in a search string, and will search title and city location properties of listings, and only display those that match.
    - You are expected to do this matching on frontend (after loading all the results from the backend).
    - You are only required to do case insensitive substring matching (of each word in the search field), nothing more complicated.
- Other form inputs should also exist that allow the user to search by:
    - Number of bedrooms (a minimum and maximum number of bedrooms, expressed either via text fields or a slider)
    - Date range (two date fields) - only display bookings that are available for the entire date range as inputted by the user.
    - Price (a minimum and maximum price, expressed either via text fields or a slider)
    - Review ratings:
        - Sort results from highest to lowest review rating **or** from lowest to highest review rating depending
        - If there is more than one listing with the same rating, their order does not matter
- The search section must have an associated search button that will action the search to reload the results given the new filters.

2.4. Feature Set 4. Viewing and Booking Listings (9% for solo, 8% for pairs)

2.4.1. View a Selected Listing

- A unique route must exist for this screen that is parameterised on the Listing ID
- For `2.3`, when a listing is clicked on, this screen should appear and display information about a specific listing.
- On this screen the user is given the listing they have decided to view in 2.4.1. This consists of:
    - Title
    - Address (displayed as a string, e.g. 1/101 Kensington Street, Kensington, NSW)
    - Amenities
    - Price:
        - If the user used a date range for search in `2.3.2` - display **price per stay**
        - If the user did not use a date range for search in `2.3.2` - display **price per night**
    - All images of the property including the listing thumbnail (they don't have to be visible all at once)
    - Type
    - Reviews

    - Review rating
    - Number of bedrooms
    - Number of beds
    - Number of bathrooms
- On this screen if the user is logged in and they have made booking for this listing, they should be able to see the status of their booking (see `2.4.2`).
- (Note: if the user has made more than 1 booking for a listing, display the status of all the bookings)

2.4.2. Making a booking and checking its status

- On the screen described in `2.4.1`, a **logged in** user should be able to make a booking for a given listing they are viewing between the dates they are after. The user enters two dates (this includes day, month and year), and assume the dates describe a valid booking, a button allows for the confirmation of the booking.
- A user can make an unlimited number of bookings per listing even on overlapping date ranges and even if other users have already booked the property for those dates. It is up to the host to check if they have double booked their listing and accept/deny the bookings accordingly.
- A booking's length (in days) is defined based on *how many nights* a user spends at the listed property (this is how bookings are defined on all modern accommodation platforms). For example, a booking from the 15th to the 17th of November consists of 2 days in length - 15th to the 16th and 16th to the 17th. As this is a late addition to the specification, we will not be strictly enforcing how you chose to calculate a booking's length. So for the case described here, it could also be expressed as a 3 day long booking (15th, 16th and 17th as the 3 days).
- Once a booking is made, the user receives some kind of temporary confirmation on screen.

2.4.3 Leaving a listing review

- A logged in user should be able to leave a review for listings they've booked that will immidiately appear on the listing screen after it's been posted by the user. The review will consist of a score (number) and a comment (text). You can leave an unlimited number of reviews per listing.
- Please note: Normally you'd prohibit reviews until after a booking visit is complete, but in this case for simplicity we allow reviews to be left as soon as a booking's status becomes `accepted`.
- If the user has made more than 1 booking for a given listing, you can use any of their `bookingid`s for the purpose of leaving a review. Just as long as the booking has status `accepted`.

2.5. Feature Set 5. Removing a Listing, Managing Booking Requests (9% for solo, 8% for pairs)

2.5.1. Removing a live listing

- On the hosted listings screen described in `2.2.1`, add the ability to remove a live listing from being visible to other users.
- Once un-published, those who had made bookings for a removed listing will no longer be able to view it on their landing screen

2.5.2. Viewing booking requests and history for a hosted listing

- A unique route must exist for this screen that is parameterised on the listing ID
- This screen should be accessed via a button or link on the hosted listings screen `2.2.1`.
- On this screen, a list of booking requests are provided for the listing they are viewing. For each booking request, the host is able to accept/deny it.
- The screen should also display the following information about a listing:
  - How long has the listing been up online
  - The booking request history for this listing consisting of all booking requests for this listing and their status (accepted/denied)
  - How many days this year has the listing been booked for
  - How much profit has this listing made the owner this year
  - (Note: When counting the days and profits, inlcude all the bookings, past or future, that have been accepted for this year)

2.6. Feature Set 6. Advanced Features (0% for solo, 8% for pairs)

2.6.1 Advanced Listing Rating Viewing

- On hover of star rating a tool tip appears which displays the break down of how many people rated the booking (both in percentage terms and absolute terms) within each star category. (e.g. see Amazon product rating for reference)
- If you click on a particular star rating, another screen should appear (that can be closed) that shows all of the individual reviews left for that rating.

2.6.2 Listing Profits Graph

- On the screen described in 2.2.1, a graph of how much profit the user has made from all their listings for the past month must be displayed. The X axis should be "how many days ago" (0-30),

and the Y axis should be the $$ made on that particular day (sum of income from all listings).

2.6.3. Listing Upload

- For `2.2.1`, when a new listing is created, the user can optionally upload a .json file containing the full data for a listing. The data structure is validated on the frontend before being passed to the backend normally.
- If you implement this feature, you must attach an example .json into your repo in the project folder. This file must have name `2.6.json`. This is so we can actually test that it works while marking.

2.6.4 YouTube Listing Thumbnail

- For any given listing, making it possible to use a Playable YouTube video as the listing thumbnail. This youtube video URL becomes a field in the create/edit hosted listing screen.

2.7. Linting

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

2.8. 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 a user that is described as:
  i. Registers successfully
  ii. Creates a new listing successfully

iii. Updates the thumbnail and title of the listing successfully
iv. Publish a listing successfully
v. Unpublish a listing successfully
vi. Make a booking successfully
vii. Logs out of the application successfully
viii. 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`
- (If working solo) include a short rationale of the testing you have undertaken within `TESTING.md`.

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

Advice for Component Testing

- Find a simple primitive component you've written, and if you don't have one, write one. This could include a common button you use, or a popup, or a box, or an input. Often examples of these are just MUI or other library components you might have wrapped slightly
- Find one that has props of some sort - ideally a few
- Simply write some unit tests that check that for a given prop input, the component behaves in a certain way (e.g. action or visual display), etc etc

Advice for UI Testing

- Consider adding `cy.wait(1000)` if necessary to add slight pauses in your tests if you find that the page is rendering slower than cypress is trying to test.
- You can modify the command `yarn test` by modifying the script in `package.json`. You can have it run the standard ReactJS tests as well as cypress in one command. An example line would be `“test": “react-scripts test –watchAll=false && yarn run cypress open”`

2.9. Other notes

- The port you can use to `fetch` data from the backend is defined in `frontend/src/config.json`
- You can modify the eslint file throughout your work, but you need to revert it to the original for final submission to achieve the linting marks.
- This article may be useful to some students
- If you'd like to use Typescript, remove the frontend folder and instead run these commands in the main directory
  - `npx create-react-app frontend --template typescript`
  - `cd frontend`
  - `npm i eslint-config-standard-with-typescript`

3. Getting Started

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.

You will not need to do any work in the backend. However, 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).

For example, one of the ways you could choose to define the `address` object could be using the following schema:

```
address: {
        street: '1 Kensington Street'
        city: 'Kensington'
        state: 'NSW'
        postcode: '2032'
        country: 'Australia'
}
```

There are several objects that will need to be defined by you using the schema you think is best for the optimal solution. Within the `listing` object, you will need to define `address` and `metadata` completely. You will also need to define the schema for a single review and availability range to be added to the `reviews` and `availability` arrays respectively. For the `booking` object, you will need to define the schema of the `dataRange` that the booking has been made for.

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.

Please note: When special consideration is granted for one individual in a pair, it will only extend the deadline for the person who gets special consideration. It does not extend for the other individual. On top of this, the person who receives special consideration is required to email the lecturer to notify them of how the work is split up prior to deadline.

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 75% 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 *Friday 18th November, 10pm*.

To submit your assignment, simply run the following command on a CSE terminal:

```
$ 6080 submit ass3 [groupname]
```

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

If you are working in a pair, only one group member needs to submit

Dryrun

You can run a dryrun to sanity check your code runs basically by:

1. Pushing your code to master on gitlab
2. On a CSE terminal (vlab or lab machine), run `6080 ass3dryrun GROUP_NAME` where GROUP_NAME is the name of your group

9. Late Submission Policy

No late submission are accepted.