Student number: **190344928**

CM3050 Mobile Development
Final Report

March, 2023

Table of Contents

1. Introduction

Car Rental is an application designed to implement some of the features found in a full-fledged car rental mobile application.

Users can create a profile, browse through and filter a list of available cars, and save their car rental bookings. They can also see the list of their bookings and cancel them.

This is a React Native application that is built upon Expo, *an open-source platform for making universal native apps for Android, iOS, and the web with JavaScript and React* (see).

2. Running the app

2.1. Expo Snack link

The following Expo Snack link can be used to launch the application: https://snack.expo.dev/@yaredsha/car-rental.

2.2. Running locally

Please include the following files and folders for the app to run locally:

Folders:

`folders`

Files:

`App.js`

`app.json`

`MyBookingsComponent.js`

`MyBookingService.js`

`MyData.js`

`MyFilterComponent.js`

`MyProfileCellComponent.js`

`MyProfileComponent.js`

`MyProfileService.js`

`MyRentalCellComponent.js`

`MyRentalComponent.js`

`MyRentalService.js`

`MyRentalService.test.js`

`MyTabsComponent.js`

`package.json`

2.3. Github Repository

The code is also available in my Github repository under the `final` folder: https://github.com/yaredsha/cm3050-mobile-development.

3. Concept development

The hardest and time-consuming part of this project for me was coming up with a development idea. First, I had been planning to create a grocery list app for months. Nevertheless, I later abandon the idea because I was concerned that a shopping list app would not have enough room for icons and photos to give the app a "state of the art" appearance. The possibility that the app will merely provide a list of texts was not exciting to me.

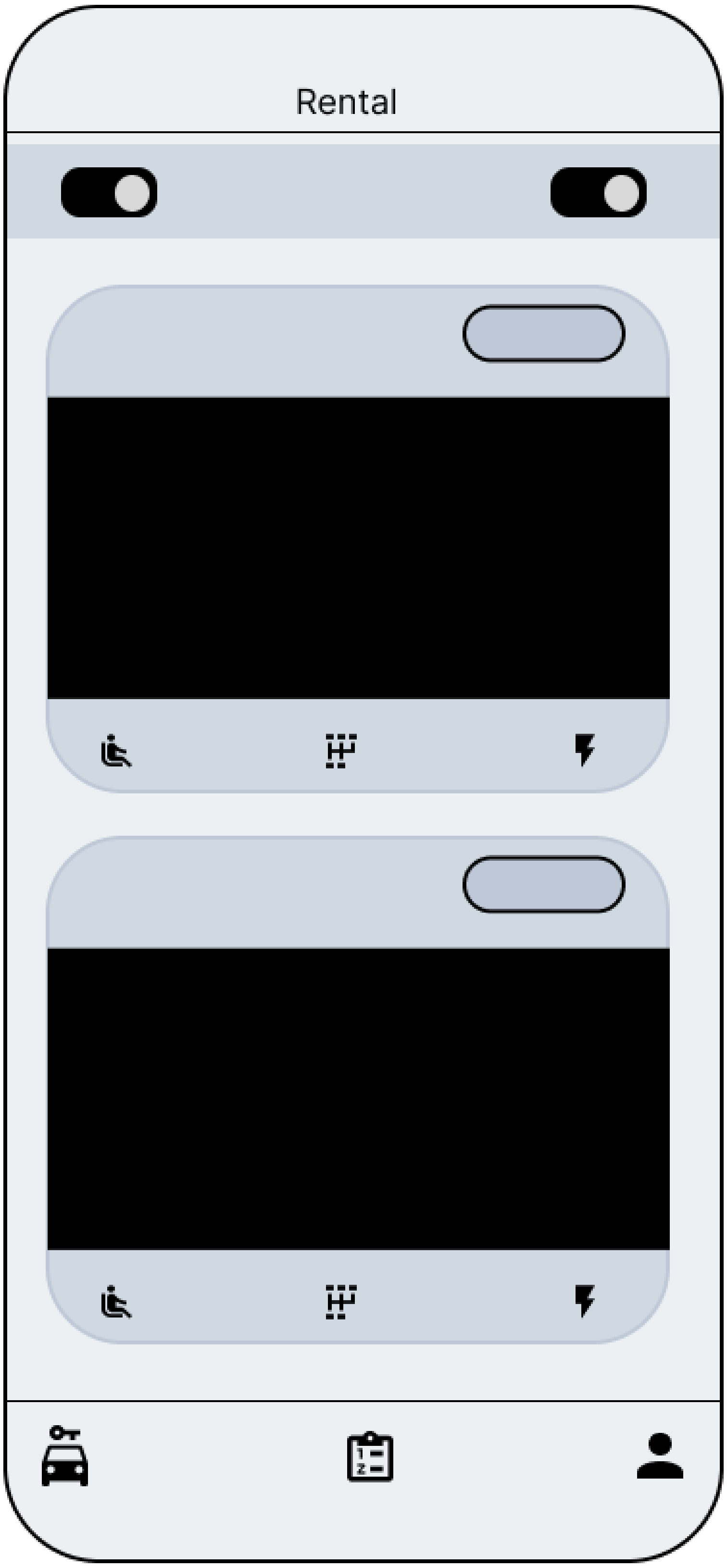
I abandoned some additional app concepts for various reasons. Some of them would have heavily relied on cloud and external api services, which I was concerned would not work during grading. The notion of creating a game was another one I abandoned after days of contemplation. I believed that no one would understand the game's rules.

After spending too much time on various concepts, I finally came up with a simple car rental app. The requirements I set for the app are that it should be simple to understand and use, should not rely on external data sources and services (if possible) but should be complex enough to demonstrate the core React Native concepts such as a well-designed and self-contained custom class components that do manage their own states and communicate and handle events via props.

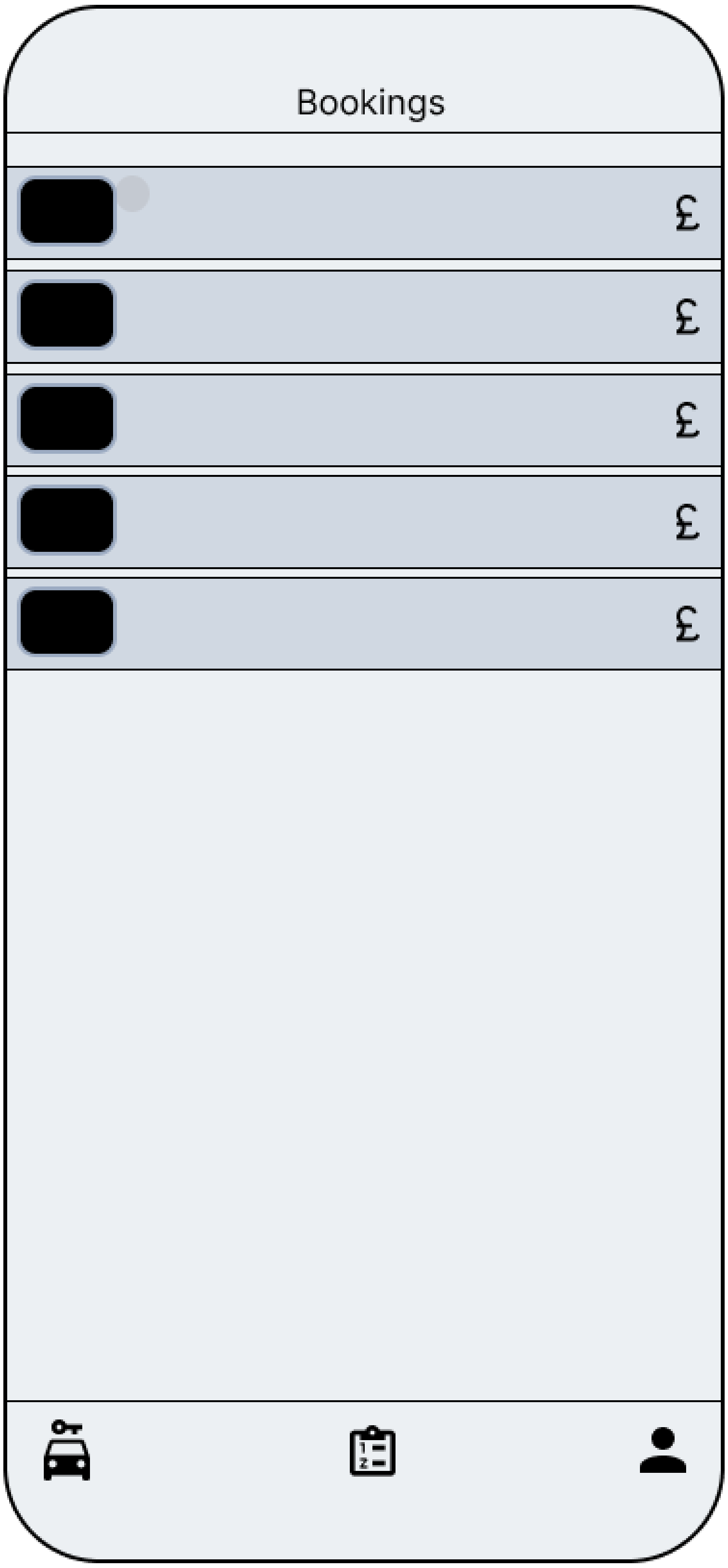
The navigation of the application should be handled by a bottom tab navigation. The application should enable users to browse through a list of cars. It should give enough information about each car such as its model, image, engine type, rental price, etc. so that a user is in a position to make decisions. Users should also have the possibility to apply filters to help ease their selection experience. Users should be able to create a personal profile and book a car. Further, they should be able to view the list of their booked cars and cancel their booking.

4. Wireframing

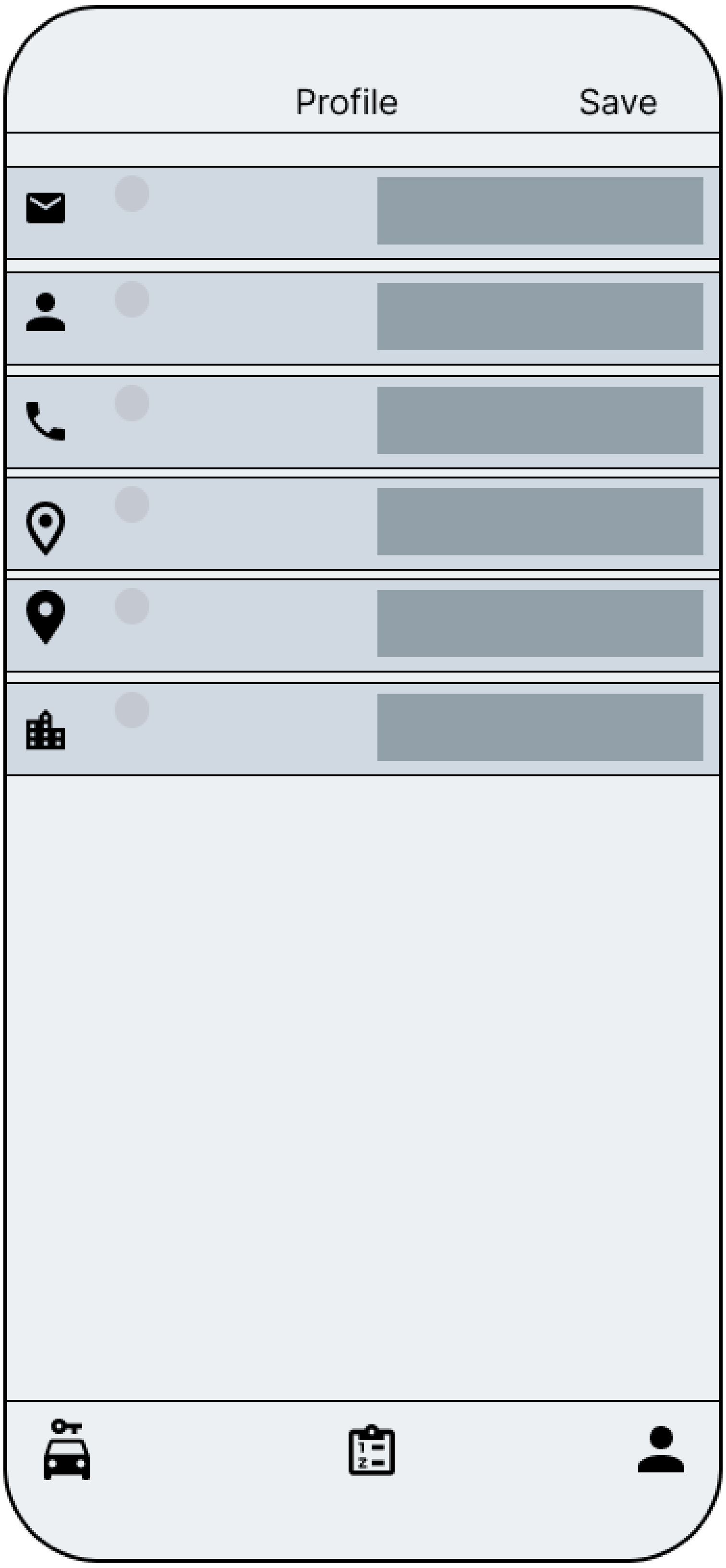
4.1. Rental screen wireframe



4.2. Bookings screen wireframe



4.3. Profile screen wireframe



5. User feedback

My family members assisted me in testing the app's early iterations. The feedback was very helpful in selecting the ideal colour schemes for the app and in spotting bugs. My son was very helpful in this regard.

6. Development

The application is built modular with a class component for each of the UI concerns/areas. Following React Native's pattern, each component manages its own state. Event propagation to parent components is accomplished by parent components providing event handlers as component properties (props).

This modular approach makes the application code easier to understand and maintain. Further, finding and fixing bugs is also much easier as each component has its own responsibility that can be easily isolated.

Components are responsible for displaying GUI elements and managing GUI states. Besides the components, there are also services that are responsible for operations that are not directly related to GUIs such as reading and saving data.

All custom components and service files start with the prefix „My" to make it easier to find them. I will briefly describe the main services and custom components and try to explain how they are plugged in together to build a functioning modular application.

6.1. MyTabsComponent

The application uses `NavigationContainer` (see) as a top-level navigator with a dark theme. It is the main component that is rendered in `App.js`. The custom `MyTabsComponent` is rendered inside the `NavigationContainer`.

`MyTabsComponent` responsible for defining the screens and children elements of each tab. This is where the bottom tab icons and tab screen titles are defined. There are three custom components that are responsible for the three tab screens: `MyRentalComponent`, `MyBookingsComponent` and `MyProfileComponent`.

6.2. MyRentalComponent

This component is the main component for the first tab `Rental` screen where users can see the list of cars, use filters to reduce the list, and make bookings by using the `Book` button. It has two main custom components. The custom component `MyFilterComponen` is responsible for placing the filter switch controls at the top of the screen and the component `MyRentalCellComponent` renders a custom table cell that displays the information for a car. The component `MyRentalCellComponent` is a custom cell for each car inside a `Tableview` component that is in turn inside a `ScrollView` and is a such builds a scrollable list of cars.

6.3. MyBookingsComponent

The component `MyBookingsComponent` is responsible for rendering the second `Bookings` tab screen where customers can see the list of their bookings. The list provides functionality for users to select bookings and cancel/delete them.

This is relatively a simple component that iterates through a user's list of bookings (Array.map) and renders a scrollable table containing the images, models, and prices of booked cars.

6.4. MyProfileComponent

This is the component that is responsible for rendering the third tab screen of the application. It enables users to create and update their profiles.

It uses the custom component `MyProfileCellComponent` to render each of the profile fields. The component `MyProfileCellComponent` renders a table cell with a TexInput component as cellAccessoryView thereby providing a convenient editable control for the user.

6.5. Component Styling

FlexBox layout is extensively used to arrange GUI components on the screen. This helped to achieve a good distribution and arrangement of GUI components with less time and effort.

6.6. MyRentalService

The `MyRentalService` service offers functionalities for the components to read data from the main `MyData.js` data file. All cars and their attributes come from the `MyData.js` data file through this service.

The `getAllCars` method returns all cars found in the `MyData.js` data file without applying any filters.

The `getCarsByFilter` method returns a list of cars that match the given filter criteria, which can include features like electric or fuel cars, cars with manual or automatic gear-shift, etc. It uses an AND operator to combine multiple filter criteria, i.e. a car must have all the features that match all a given criteria so that it can be in the result list.

There is a unit test for testing different aspects of this service (see Unit testing)

The method `getCarsByIds` returns all cars that match the list of car ids given.

6.7. MyBookingService

The service `MyBookingService` is used by the custom components to read, save and delete bookings. The service saves the list of user bookings to the Async Storage in a JSON format and transforms the JSON format to JavaScript Array during data reading. Saving to the Async Storage enables the bookings data to persist and be available after the application restart.

6.8. MyProfileService

Finally, the service `MyProfileService` provides functionalities to the custom components to save, read and delete the user's profile data. It too converts the profile data into JSON format during saving and transforms the JSON data into JavaScript object during reading.

Unlike the service `MyBookingService` though, it uses the SecureStore so the data is saved encrypted as the user profile data is sensitive and needs better protection.

7. Unit testing

Unit tests are written on top of the Jest JavaScript unit testing framework. Use the below command line to run the unit tests:

```
$ npm run test
```

The test file `MyRentalService.test.js` contains different test cases that aim to test the service `MyRentalService` as it is the service that has a logic (filtering) to be tested. The other services just save and read data to the stores. Besides, testing the stores directly is not possible rather the test is normally written against a mock store.

8. Evaluation

The most challenging part for me was to come up with an idea for the app. This is where it took me too much time to arrive at the final decision. This process consumed much of my time and left me with less time for actual development.

When it comes to the actual development and the code, I think the application is based on a good and clean structure that can be easily expanded by adding more custom components and services. More features such as retrieving data from and saving data to a server using a REST API could be added. REST client libraries such as axios can make the task easier in this regard.

Further, online location and map services could be added to the app to enable the user to select pick-up and drop-off of rental locations and also to be used when editing the user's address.

One of the positive experiences I have had developing this app is working with FlexBox layout. I really enjoyed playing and experimenting with FlexBox styling. It helped me to place and arrange GUI controls easily with little effort. It would have taken me much time and effort without the help that comes with FlexBox.

When it comes to React Native programming, in particular, I encountered two main problems which in the end I was able to fix. These challenges helped me to learn new areas of React Native.

The first challenge was reading and using data from the data storage inside a constructor of a class component. Both the Async Storage and the SecureStore need `await` for the data from the storages to be used to update the component state during initialisation. But `await` cannot be used inside a class constructor. I learned that React Native class components provide a convenient method called `componentDidMount` *that is invoked immediately after a component is mounted* (see). I used this method as an alternative to component constructors in such scenario.

The second challenge had to do with propagating events from a parent component to a child component. Normally, the parent uses the `props` to attach an event handler (a function) to get events and actions from a child component. But when I did the save button on the profile screen, I faced the problem of a parent component propagating an event to the child component. I learned that parent state changes are propagated through the method `componentDidUpdate` which comes with a class component. The custom component `MyProfileComponent` uses this method to watch for the profile save event that comes from the parent navigator component.

9. Image sources

1. List of BMW vehicles. (2023, February 27). In Wikipedia. https://en.wikipedia.org/wiki/List_of_BMW_vehicles

2. List of Audi vehicles. (2023, February 24). In Wikipedia. https://en.wikipedia.org/wiki/List_of_Audi_vehicles

3. List of Mercedes-Benz vehicles. (2023, March 3). In Wikipedia. https://en.wikipedia.org/wiki/List_of_Mercedes-Benz_vehicles

4. List of Volkswagen vehicles. (2023, February 24). In Wikipedia. https://en.wikipedia.org/wiki/List_of_Volkswagen_vehicles

5. List of Toyota vehicles. (2023, March 8). In Wikipedia. https://en.wikipedia.org/wiki/List_of_Toyota_vehicles

6. List of Renault vehicles. (2023, February 2). In Wikipedia. https://en.wikipedia.org/wiki/List_of_Renault_vehicles

7. List of Peugeot vehicles. (2023, January 23). In Wikipedia. https://en.wikipedia.org/wiki/List_of_Peugeot_vehicles

8. Logo and splash: https://unsplash.com/photos/o23qzl3obeI

9. Unsplash License: https://unsplash.com/license