Morgan Hodge 10779528

Introduction

GitHub Repository : https://github.com/Plymouth-University/mainassessment-Morgan101h/tree/main

YouTube Video Link: https://youtu.be/PItYe9gUlqA

This report documents the creation of an Android native application programmed using java that is tailored to the set requirements.

The purpose of this document is to give the reader a better understanding of how this application was created, how it works, and the rationale behind it. Consideration has been given to the strategic planning process, including the creation of low and high-fidelity prototypes, paper-based prototypes, and design patterns. Legal, Ethical, Social and Professional (L.E.S.P) issues have been given due consideration and will be discussed in more detail during section three of this report.

However, the scope of designing an application goes beyond its look and functionality. This report will discuss how the code works, why it was implemented in the way it was and how certain aspects of the code behave. This report is designed to be read by those who are not yet familiar with programming in order to support their understanding and knowledge acquisition of the subject.

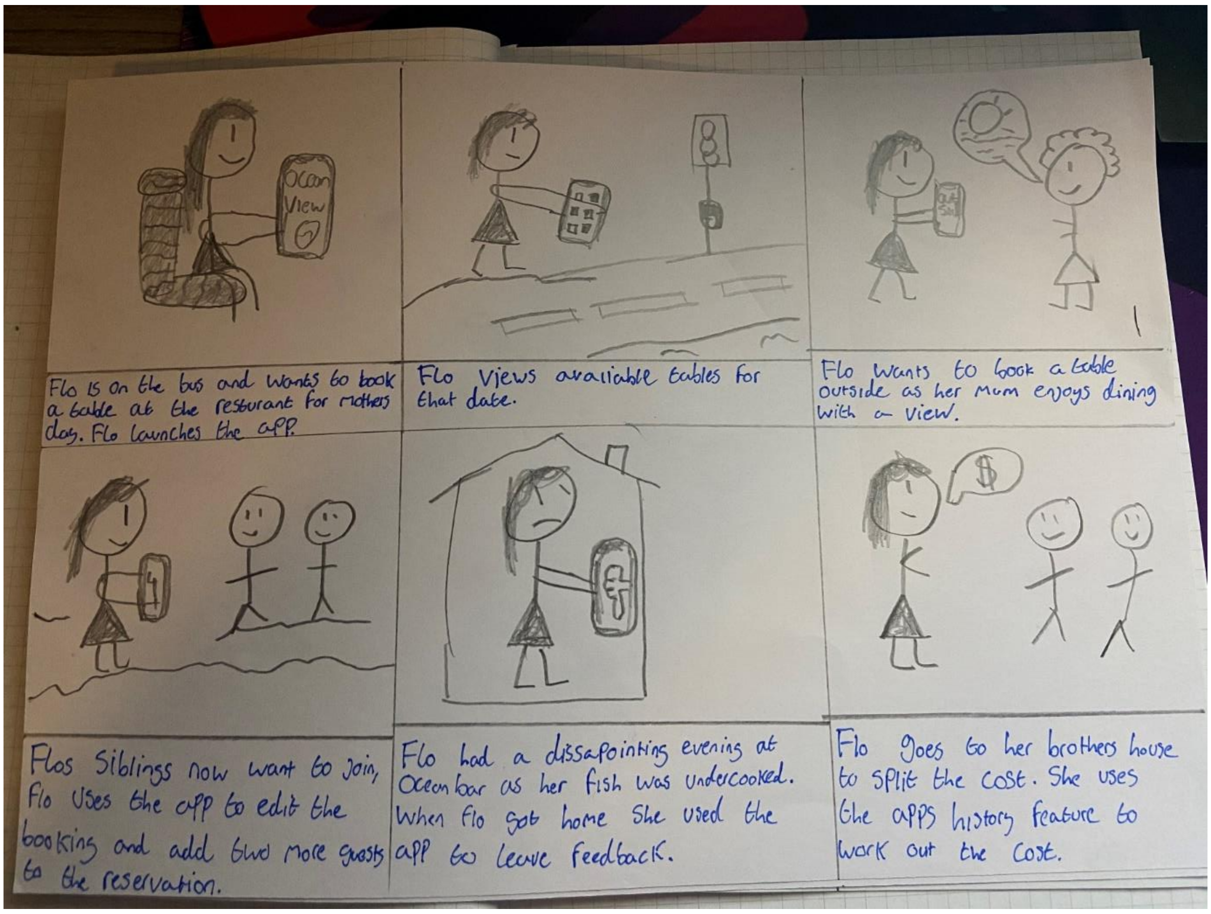
Morgan Hodge 10779528

Background

A large restaurant situated near the sea aims to enhance customer experience by developing a mobile application that facilitates the booking of tables via mobile phones. The restaurant offers reservation options for breakfast, lunch and dinner, With both inside and outside seating areas. The maximum table size available for reservation is ten people, due to the reservations high demand; bookings also need to be made a week in advance.

The potential users of this application will mostly consist of customers and staff. The customers will be using this app to create bookings, view the menu, check opening times and to leave/read feedback. However, the staff will be using the app to add new items to the menu, read and filter reviews, manage customer accounts and make any necessary changes such as opening times if they were to be changed.

I have created a storyboard to help the reader of this document understand and view who the potential users may be and what the application will be used for:


Ocean View
Flo is on the bus and wants to book a table at the resturant for mothers day. Flo launches the app.
Flo views availiable tables for that date.
Flo wants to book a table outside as her Mum enjoys dining with a view.
Flos Siblings now want to Join, Flo Uses the app to edit the booking and add two more guests to the reservation.
Flo had a dissapointing evening at Ocean bar as her fish was undercooked. When Flo got home she used the app to leave feedback.
Flo goes to her brothers house to split the cost. She uses the apps history feature to work out the cost.

Morgan Hodge 10779528

Legal, Ethical, Social and Professional Problems that may arise

Ethical consideration was given to customers privacy by obtaining consent from the user before collecting personal data, as the customer needs to be aware of what data is being collected, how it will be used and have the option to opt in or out. This issue will be addressed by having a TextView Box on the login and register page stating what information will be collected and if the user proceeded to make an account, that will then be taken as consent to the developers.

When creating the app, it was crucial to comply with data protection laws such as the General Data Protection Regulation as this application collects, processes and stores user data. Failure to comply with this law can result in financial penalties, reputational damage, legal actions, lawsuits and more. Therefore, it is crucial to follow this legislation, this was addressed by ensuring that user data was stored safely, and this was done by storing it on a secure API and locally on the developer's computer using local storage. The application is transparent with the user, informing the user in the registration page that their data will be stored.

Due to their being a feedback and reviews section on the application it creates ethical and professional problems. Manipulating or fabricating user reviews and ratings on the app can lead to ethical concerns that can damage the reputation of the restaurant and manipulate customers into a false impression of the restaurant.

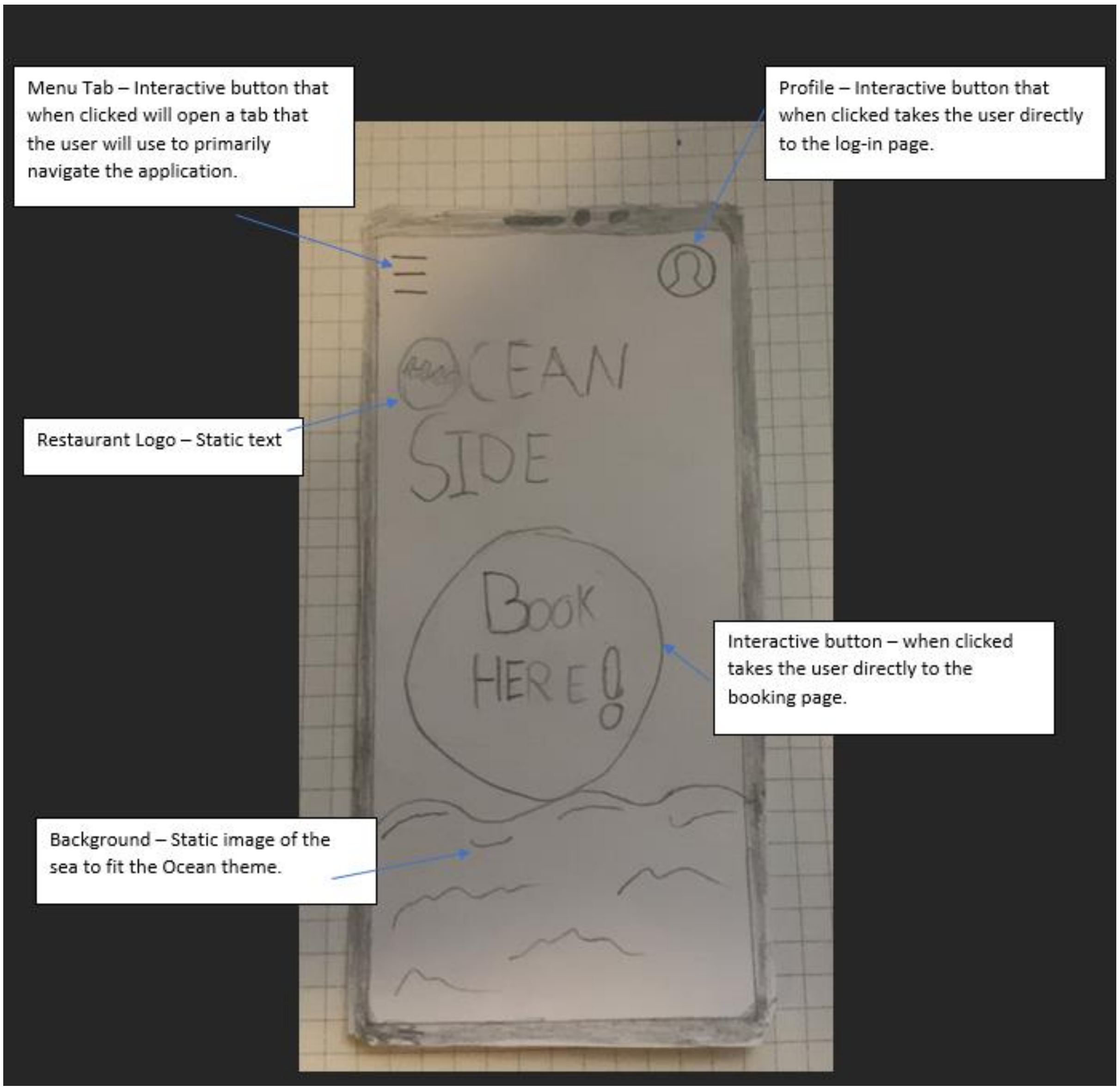
This issue has been planned to be resolved by only allowing users that have already dined at the restaurant to leave reviews. This will mitigate any malicious users that may want to leave negative and fabricated reviews to damage the restaurants reputation, as before this change any user that has an account could leave a review.

Design

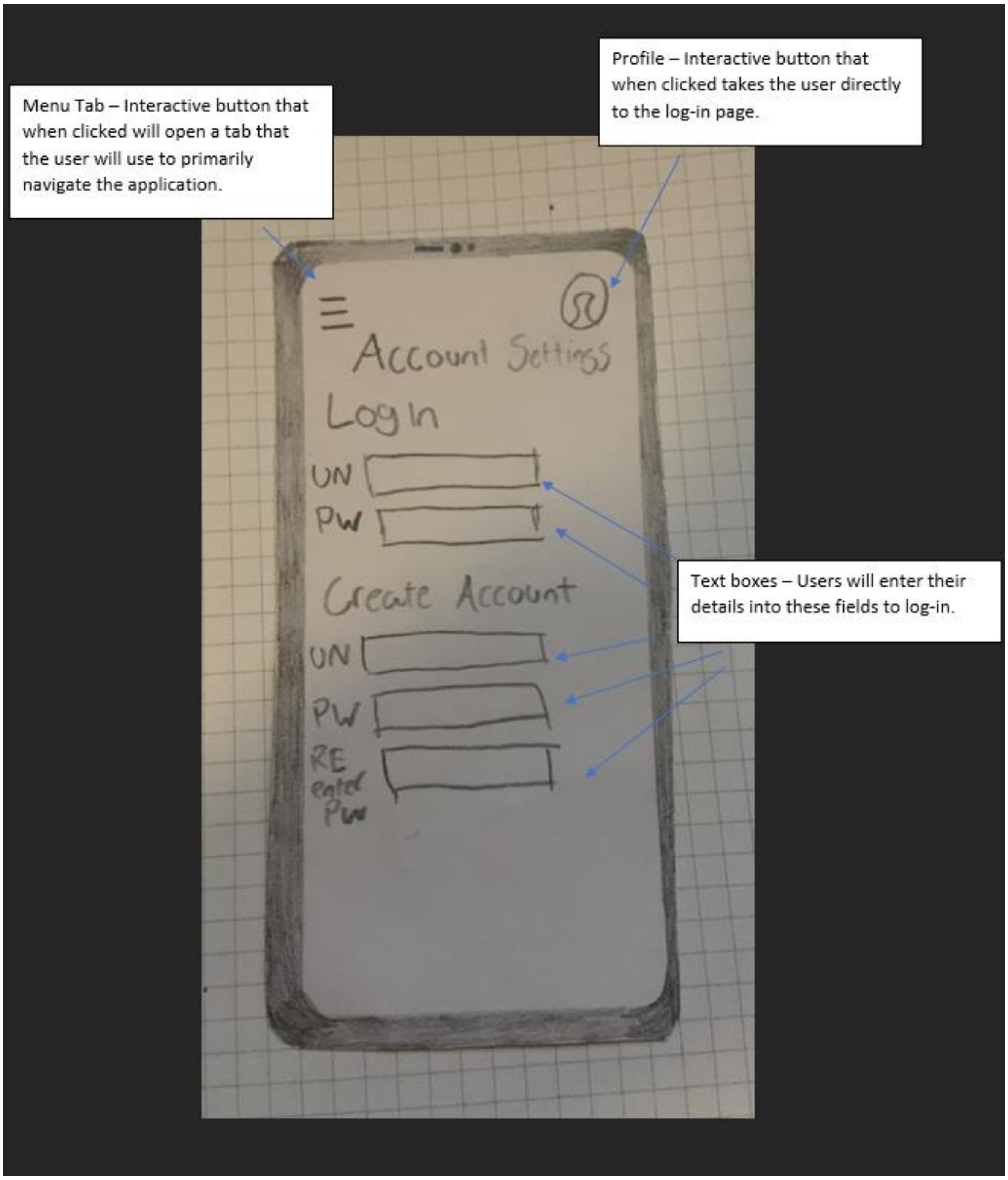
Low- Fidelity Prototype

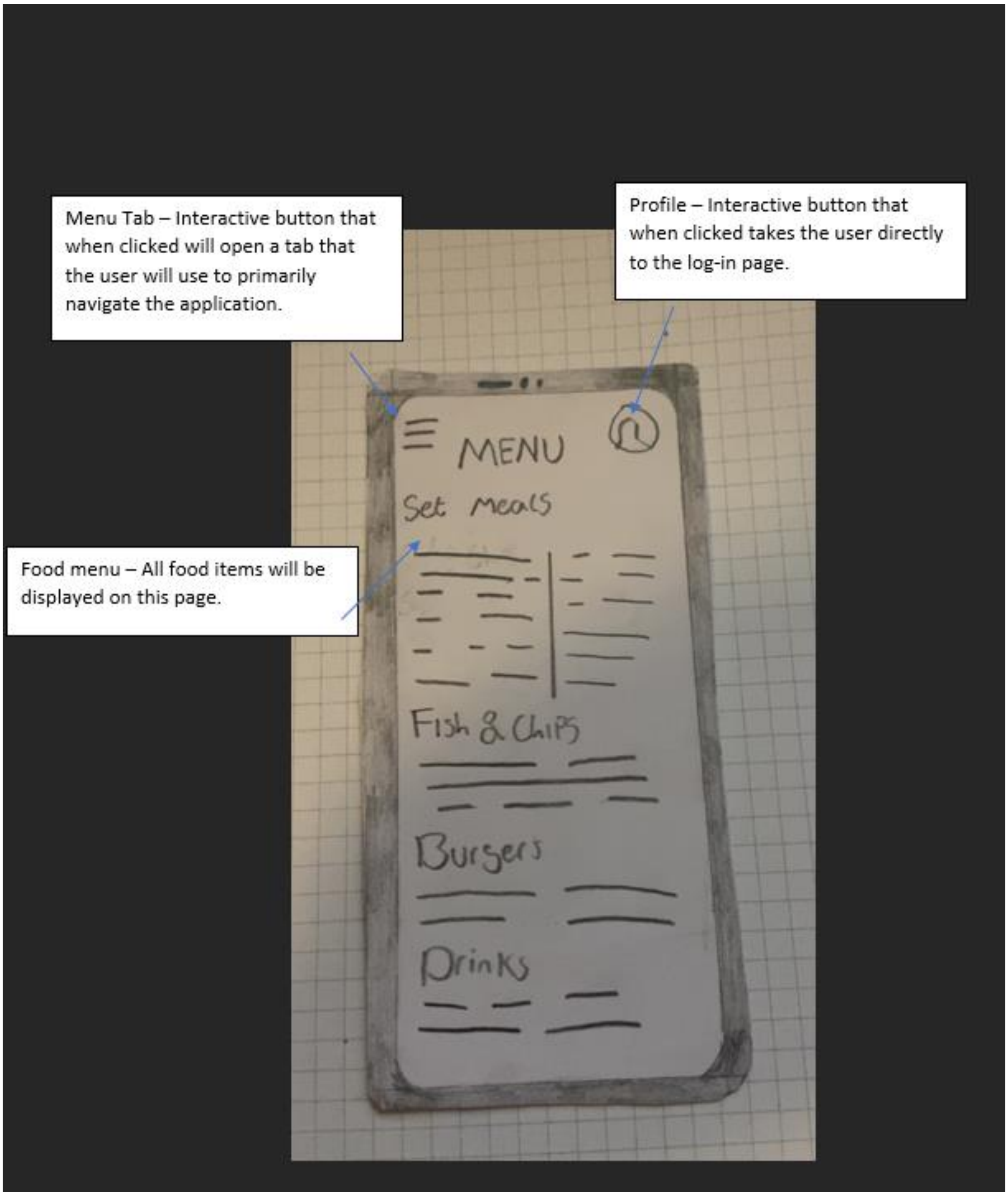
The goal of a Low Fidelity Prototype is to mainly represent the functions of the app. This was particularly important during the early stages of development as it allowed us to explore more ideas and modify the design of the app before creating the High-Fidelity Prototype. For this specific design method, A paper-based prototype was created.

Paper Based Prototype:

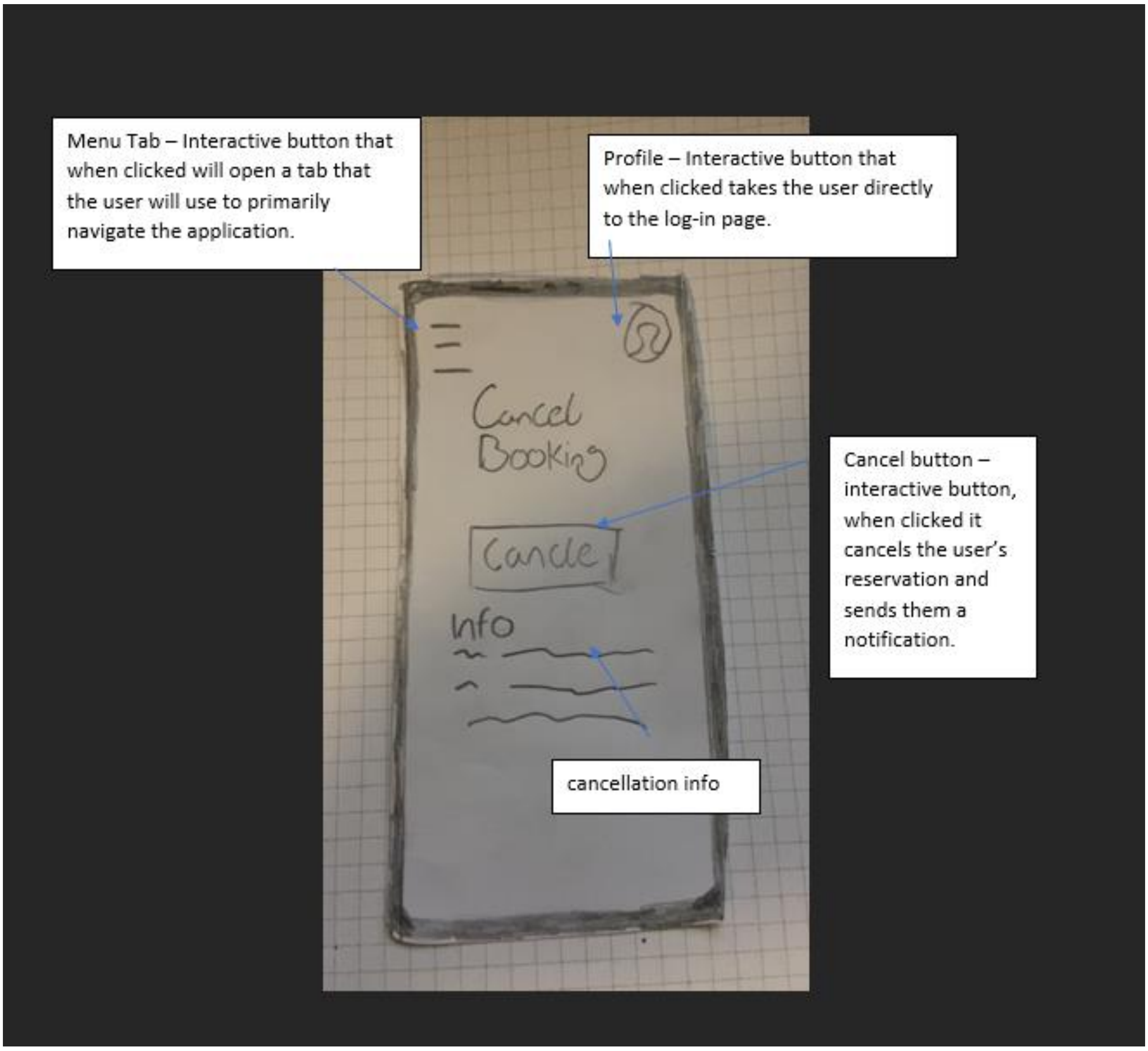

Menu Tab – Interactive button that when clicked will open a tab that the user will use to primarily navigate the application.
Profile – Interactive button that when clicked takes the user directly to the log-in page.
OCEAN SIDE
Restaurant Logo – Static text
BOOK HERE!
Interactive button – when clicked takes the user directly to the booking page.
Background – Static image of the sea to fit the Ocean theme.

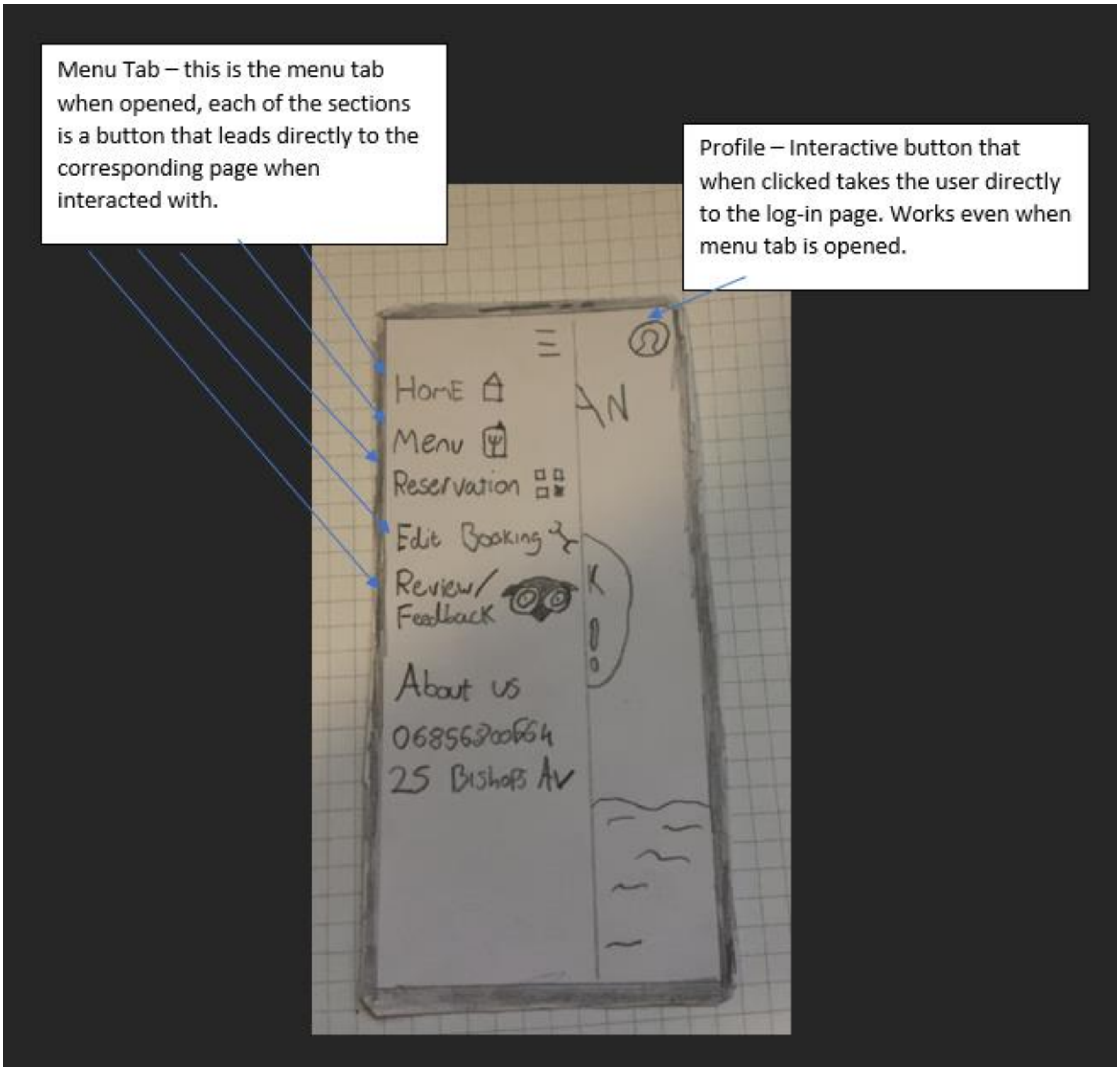
Morgan Hodge 10779528


Menu Tab – Interactive button that when clicked will open a tab that the user will use to primarily navigate the application.
Profile – Interactive button that when clicked takes the user directly to the log-in page.
Account Settings
Log In
UN
PW
Create Account
UN
PW
RE enter PW
Text boxes – Users will enter their details into these fields to log-in.


Menu Tab – Interactive button that when clicked will open a tab that the user will use to primarily navigate the application.
Profile – Interactive button that when clicked takes the user directly to the log-in page.
Food menu – All food items will be displayed on this page.
MENU
Set meals
Fish & Chips
Burgers
Drinks

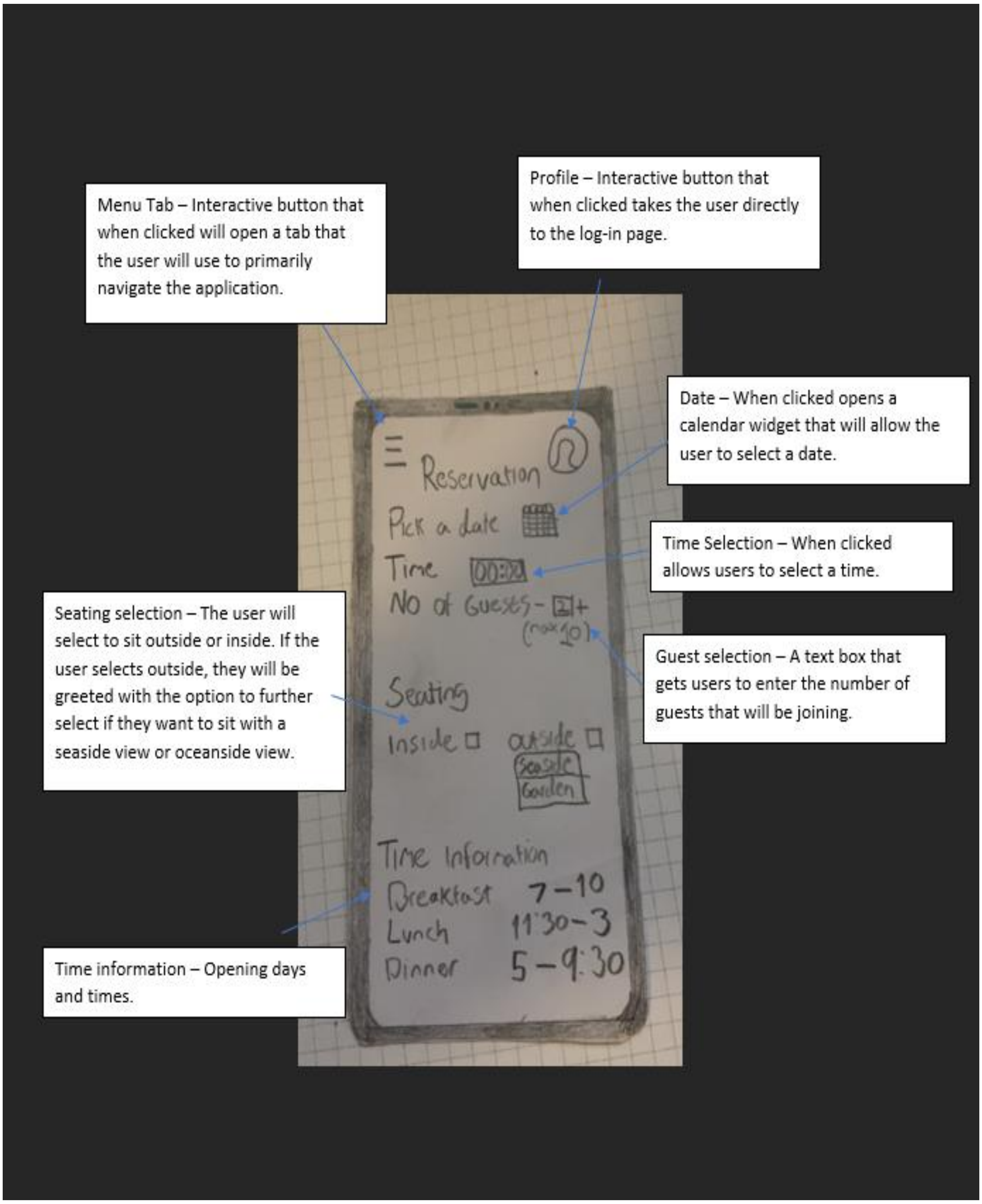
Morgan Hodge 10779528


Menu Tab – Interactive button that when clicked will open a tab that the user will use to primarily navigate the application.
Profile – Interactive button that when clicked takes the user directly to the log-in page.
Cancel Booking
Cancle
Cancel button – interactive button, when clicked it cancels the user's reservation and sends them a notification.
Info
cancellation info



Menu Tab – this is the menu tab when opened, each of the sections is a button that leads directly to the corresponding page when interacted with.

Profile – Interactive button that when clicked takes the user directly to the log-in page. Works even when menu tab is opened.



Menu Tab – Interactive button that when clicked will open a tab that the user will use to primarily navigate the application.

Profile – Interactive button that when clicked takes the user directly to the log-in page.

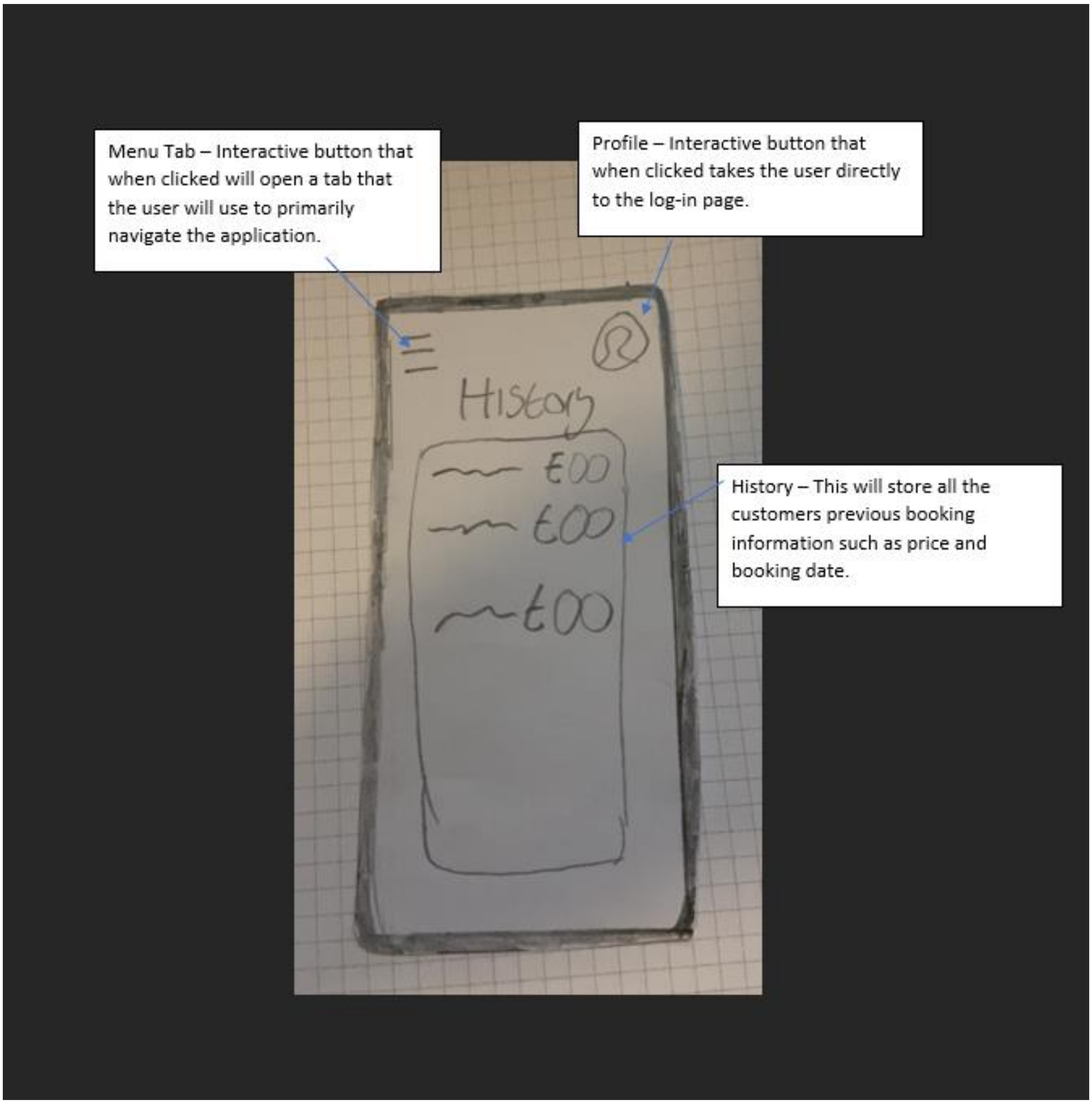
Date – When clicked opens a calendar widget that will allow the user to select a date.

Time Selection – When clicked allows users to select a time.

Seating selection – The user will select to sit outside or inside. If the user selects outside, they will be greeted with the option to further select if they want to sit with a seaside view or oceanside view.

Guest selection – A text box that gets users to enter the number of guests that will be joining.

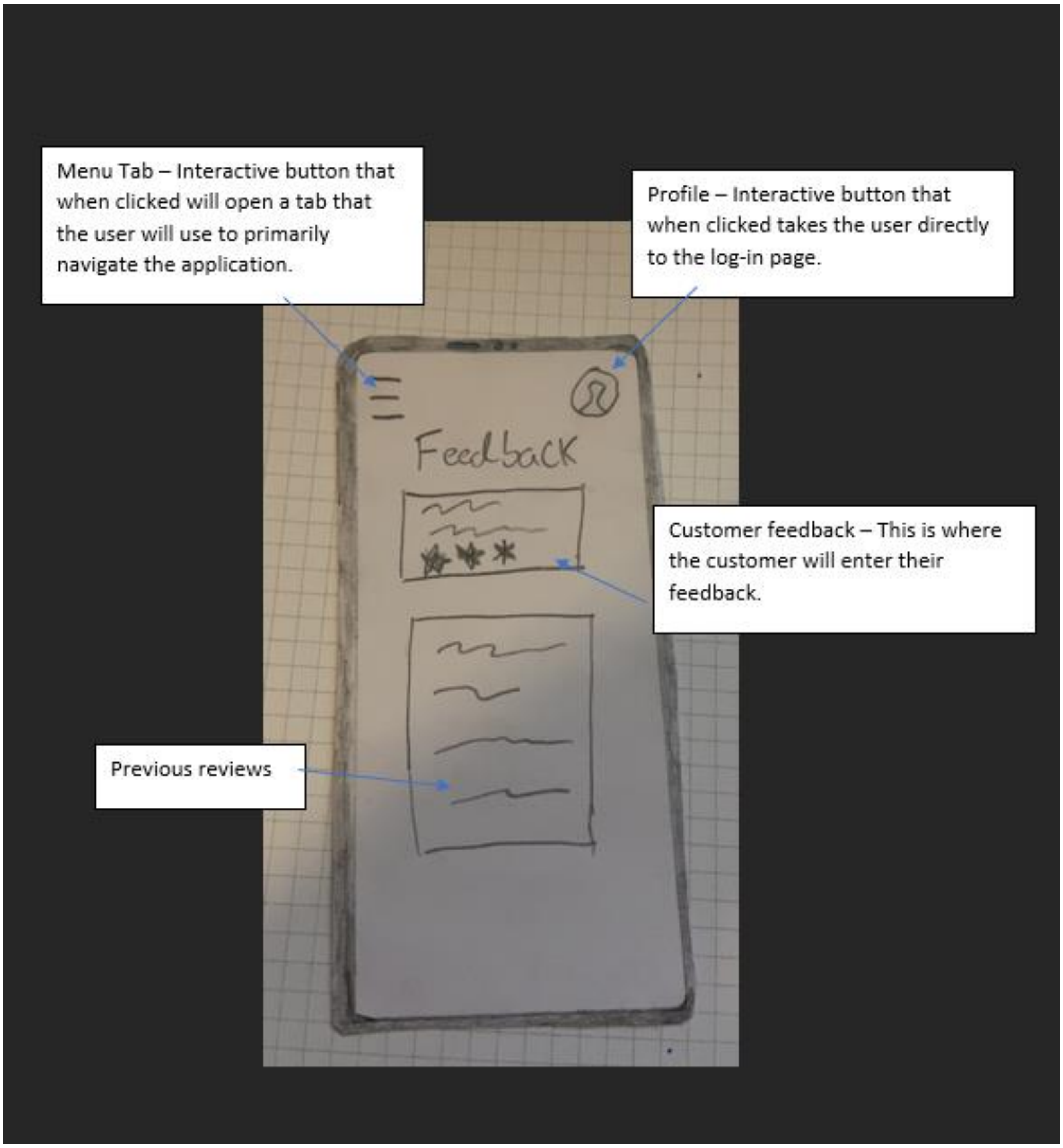
Time information – Opening days and times.



Menu Tab – Interactive button that when clicked will open a tab that the user will use to primarily navigate the application.

Profile – Interactive button that when clicked takes the user directly to the log-in page.

History – This will store all the customers previous booking information such as price and booking date.



Menu Tab – Interactive button that when clicked will open a tab that the user will use to primarily navigate the application.

Profile – Interactive button that when clicked takes the user directly to the log-in page.

Customer feedback – This is where the customer will enter their feedback.

Previous reviews

Morgan Hodge 10779528

<u>High- Fidelity Prototype</u>

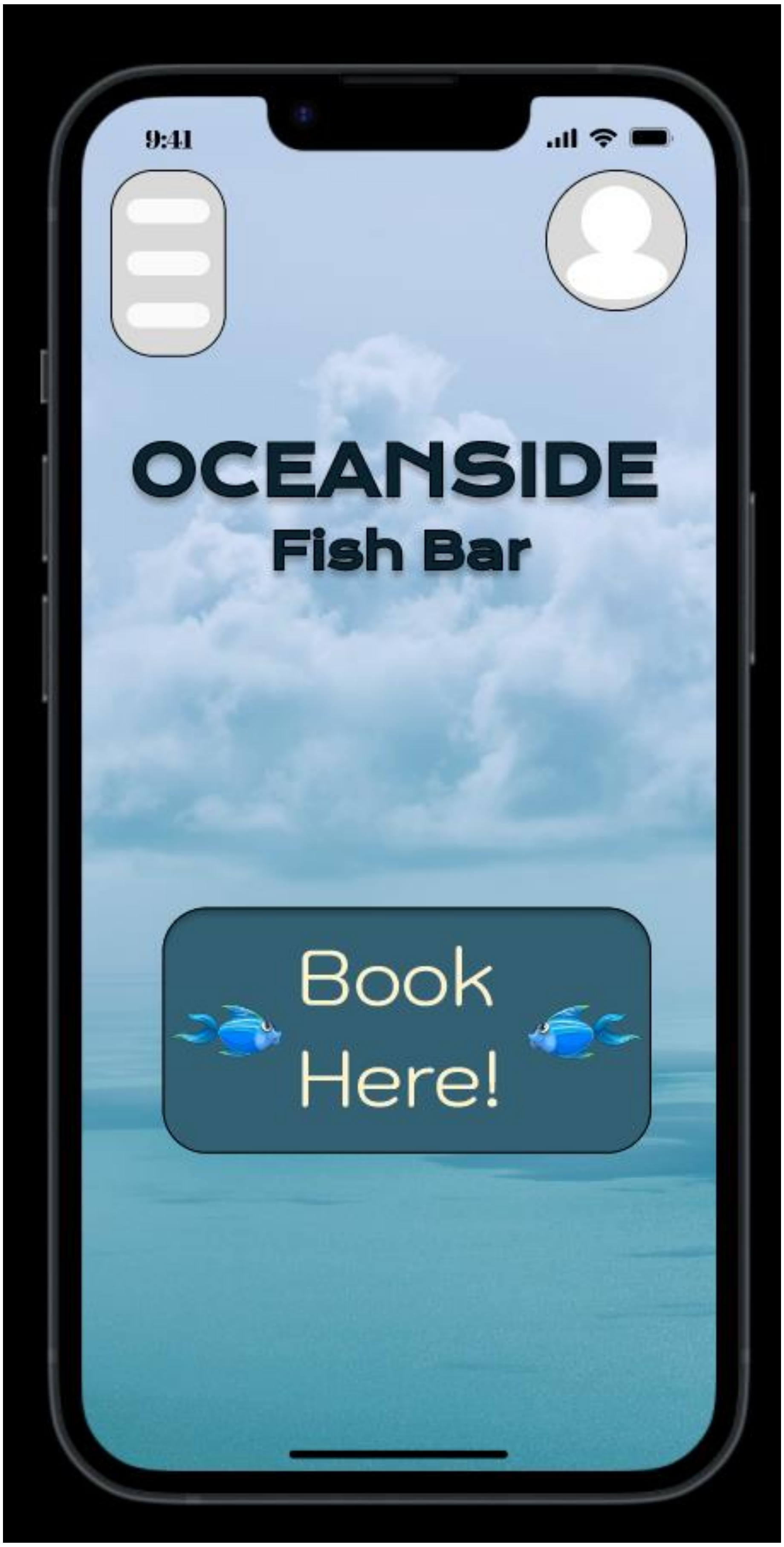
High Fidelity prototypes are more detailed and closely resemble the final product in terms of visual design, interaction functionality and content. These types of design methods are useful for simulating the user experience more closely and locating any errors or missing requirements.

For this Prototype method, a software called Figma was used. Figma is a designing tool that allows for use of various design elements and components to create a responsive design.

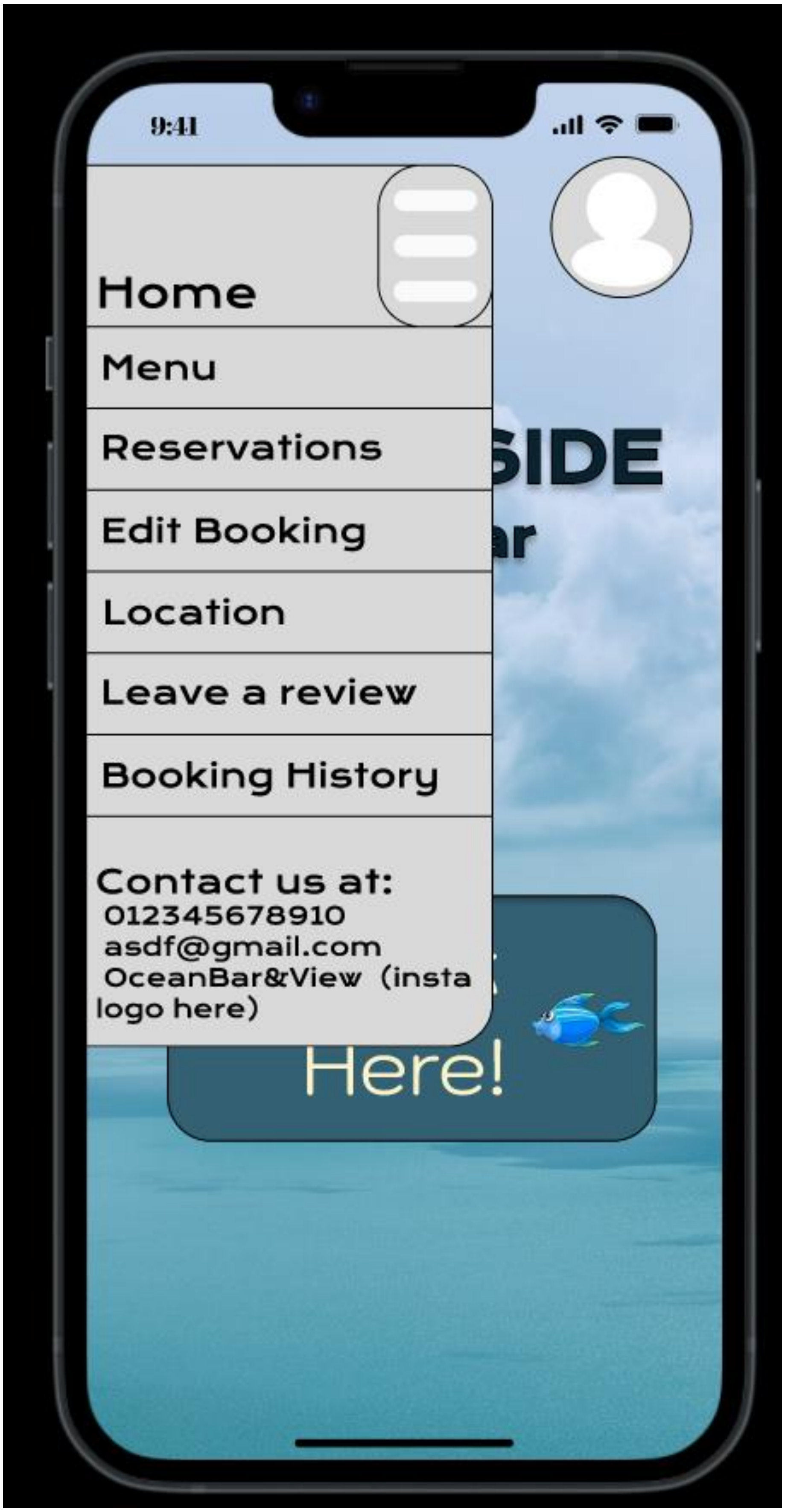
<u>Link to the Figma Prototype</u>

Screenshots of the Figma Prototype:

Morgan Hodge 10779528



Morgan Hodge 10779528



9:41

Home

Menu

Reservations

Edit Booking

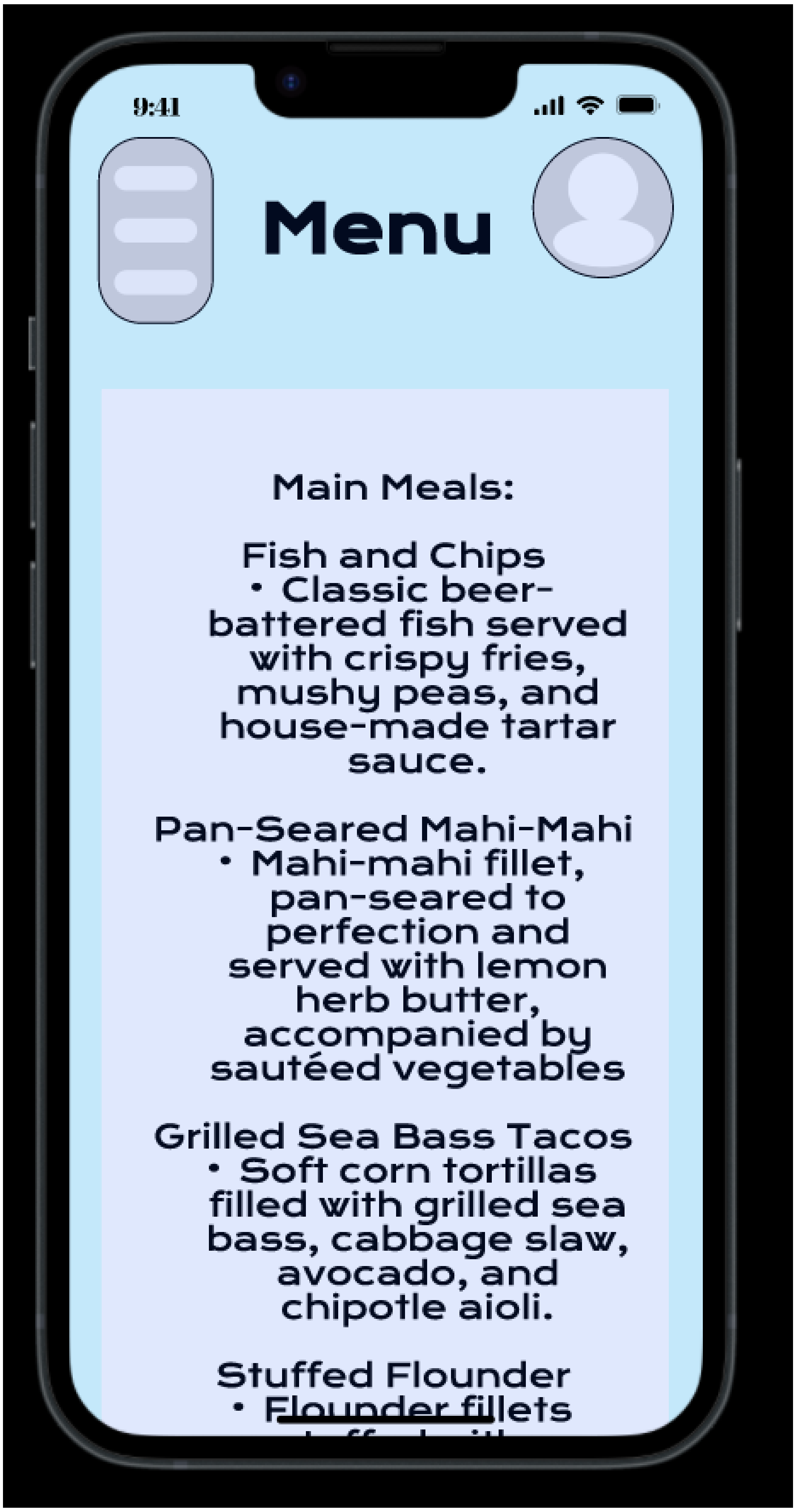
Location

Leave a review

Booking History

Contact us at:
012345678910
asdf@gmail.com
OceanBar&View (insta logo here)

Morgan Hodge 10779528



Main Meals:

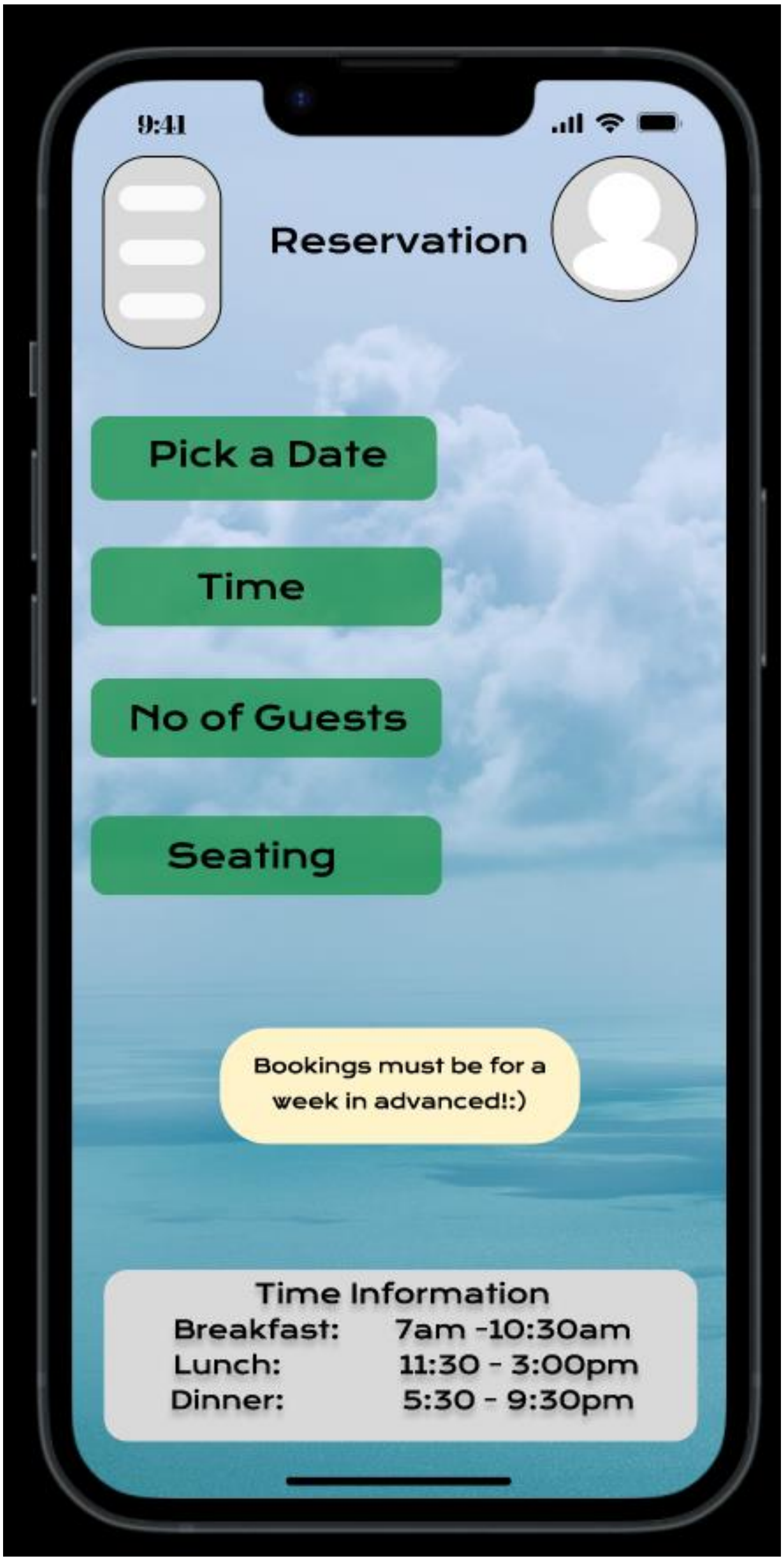
Fish and Chips
• Classic beer-battered fish served with crispy fries, mushy peas, and house-made tartar sauce.

Pan-Seared Mahi-Mahi
• Mahi-mahi fillet, pan-seared to perfection and served with lemon herb butter, accompanied by sautéed vegetables

Grilled Sea Bass Tacos
• Soft corn tortillas filled with grilled sea bass, cabbage slaw, avocado, and chipotle aioli.

Stuffed Flounder
• Flounder fillets

Morgan Hodge 10779528

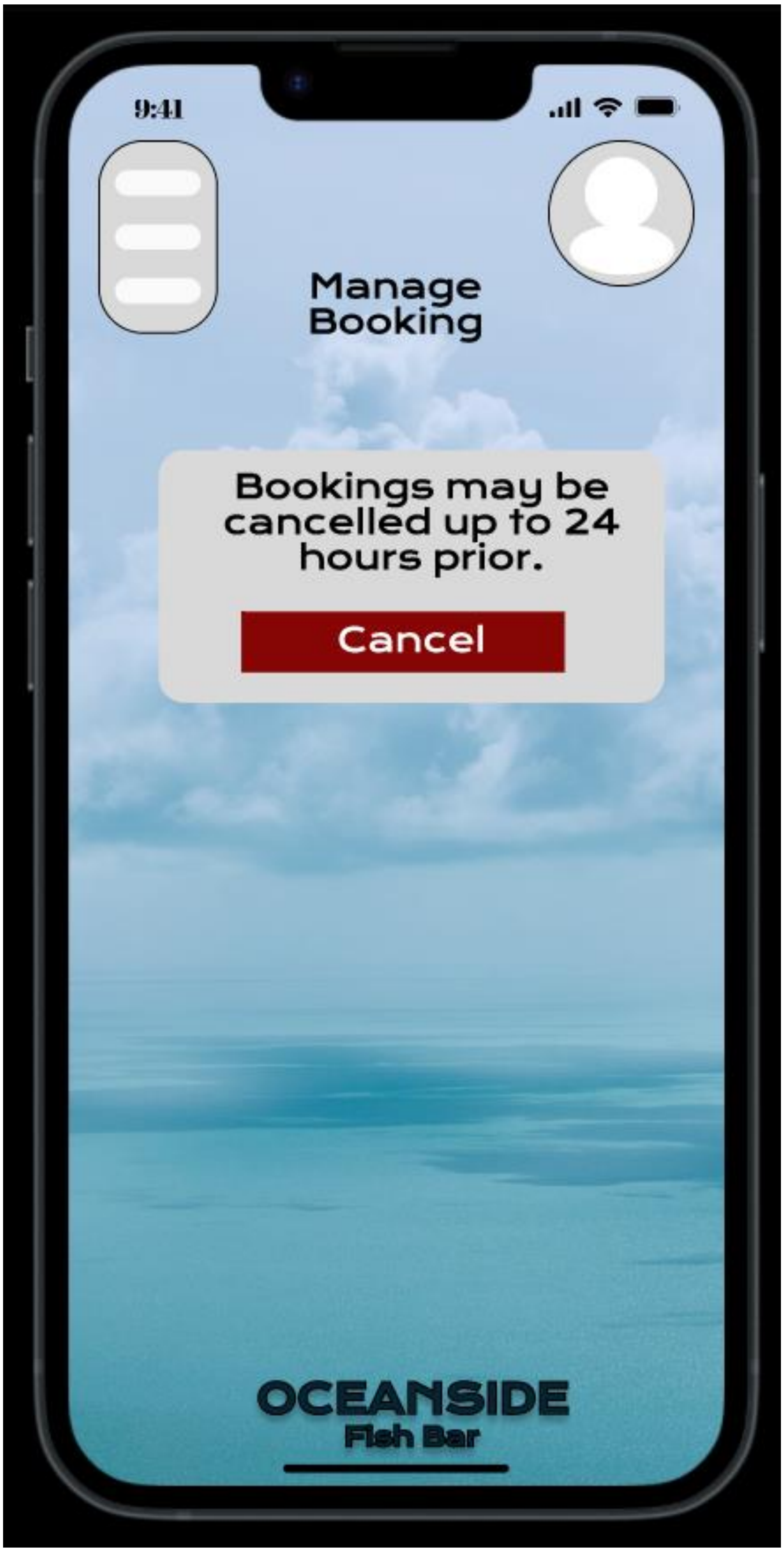


Bookings must be for a week in advanced!:)

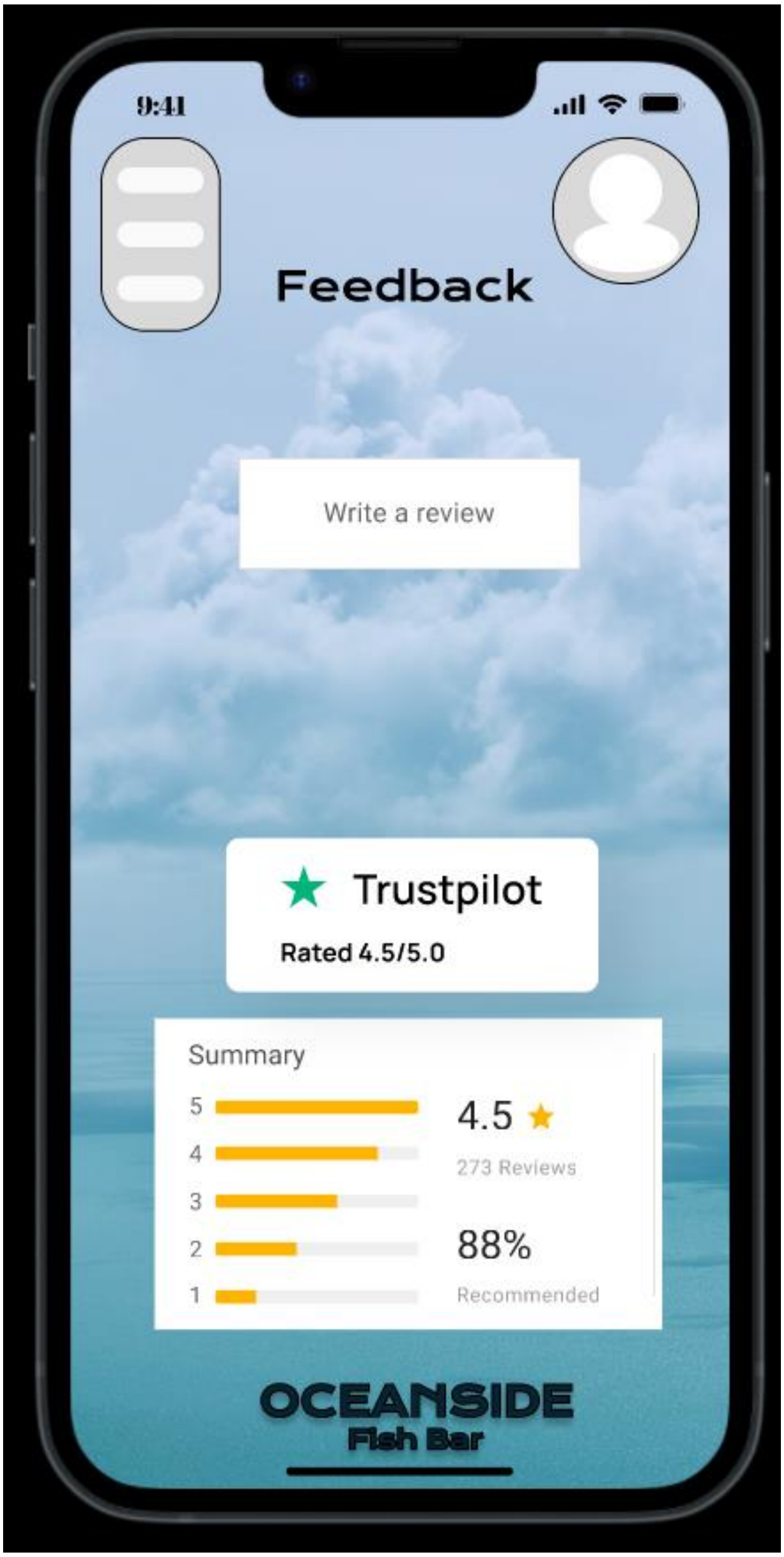
Time Information

Breakfast:	7am -10:30am
Lunch:	11:30 - 3:00pm
Dinner:	5:30 - 9:30pm

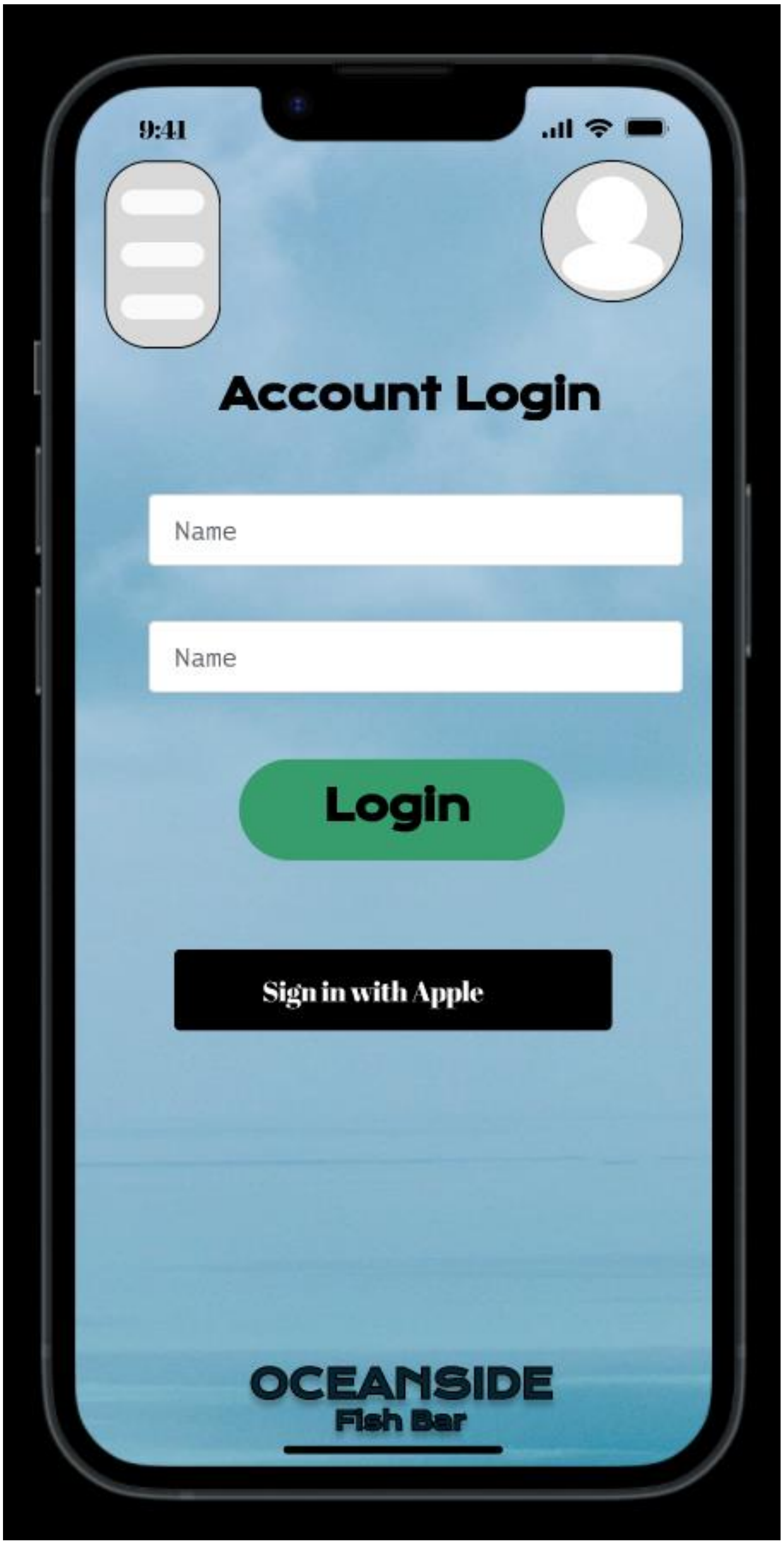
Morgan Hodge 10779528



Morgan Hodge 10779528



Morgan Hodge 10779528



Morgan Hodge 10779528

Conceptual model

Conceptual model is a form of design that provides a high-level understanding of the entities, functions, relationships and constraints within the restaurant mobile application, it can serve as a foundation for further development.

Entities

Customer: Represents users of the application

Booking: Represents a reservation made by a customer

Table: Represents a seating are in the restaurant with attributes such as location (Inside and outside), and size (Maximum 10 people)

Functions

Make Reservations:

- Customers can create a new booking specifying date, time, and meal type (breakfast, lunch, or dinner).
- Customers choose seating preference (inside/outside, seaside/garden).
- Bookings must be made at least a week in advance due to high demand.

View and Edit Booking:

- Customers can view and edit details of their existing reservations.
- Options to edit include date, time, meal type, and seating preferences.

Cancel Reservation:

- Customers can cancel a booking up to 24 hours in advance.

Notifications:

- App sends notifications to users when a booking is approved or declined.
- Notifications for updates or cancellations are pushed to inform users promptly.

User Account:

- Users need to log in to their accounts to access reservation functionalities.
- Account management includes updating personal information and preferences.
- Users can set their favourite meals or specify preferred seating locations.

Notification Preferences:

- Users can toggle notification settings on/off based on their preferences.

Morgan Hodge 10779528

Relationships

Customer - Booking

- One-to-Many relationship as a customer can have multiple bookings.
- Many-to-One relationship as multiple customers can be associated with a single booking.

Booking - Table:

- Many-to-One relationship as multiple bookings can be associated with a single table.
- One-to-Many relationship as a table can be associated with multiple bookings.

Constraints

- Bookings must be made at least a week in advance.
- Cancellations are allowed up to 24 hours before the reservation time.
- Tables have a maximum size of 10 people.
- Users must log in to access reservation features.

Preferences

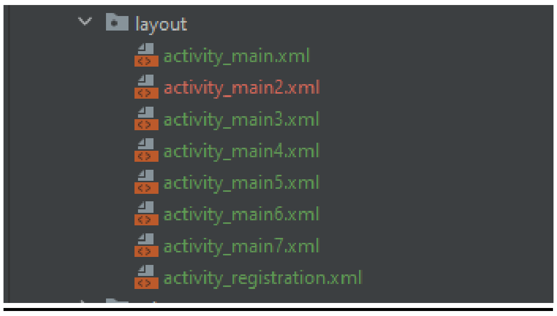
- Users can set preferences for favourite meals.
- Users can specify preferred seating locations within the restaurant.

Implementation

Once completing the design of the application (How it will look and function) and stating all the requirements, the next stage was to implement the design.

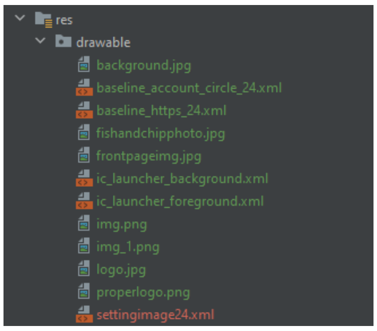
The first stage of implementation was to create the .XML files. This is where the GUI of the design was created, for each page of the application an 'Activity' had to be programmed. XML files are only for how the app will look to the user, all the logic and functionality are written within the JAVA files.

Activity Files:



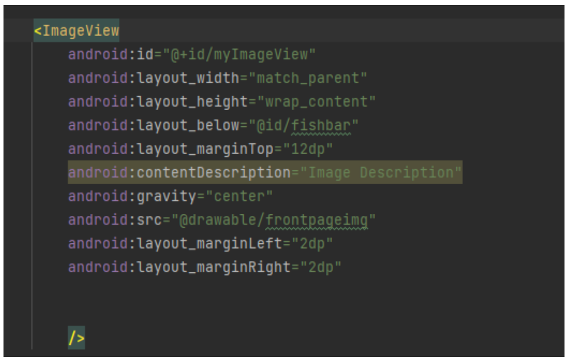
```
layout
    activity_main.xml
    activity_main2.xml
    activity_main3.xml
    activity_main4.xml
    activity_main5.xml
    activity_main6.xml
    activity_main7.xml
    activity_registration.xml
```

In total 8 Activities were created. When creating the layout, photos and icons needed to be implemented into the app, to do this we had to insert images into the 'Drawable' folder:



```
res
  drawable
      background.jpg
      baseline_account_circle_24.xml
      baseline_https_24.xml
      fishandchipphoto.jpg
      frontpageimg.jpg
      ic_launcher_background.xml
      ic_launcher_foreground.xml
      img.png
      img_1.png
      logo.jpg
      properlogo.png
      settingimage24.xml
```

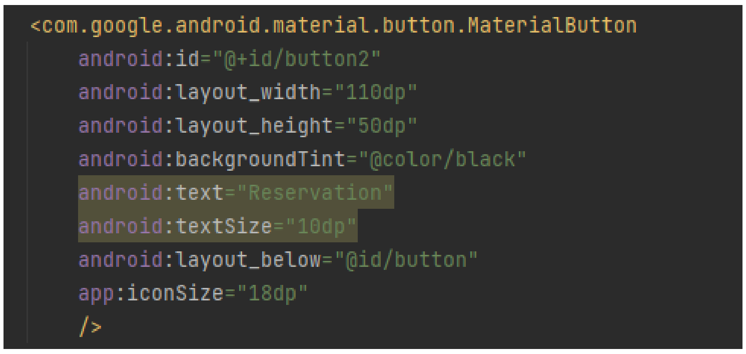
This allowed the photos to then be inserted into the .XML by using the 'Image View' tag:



```
<ImageView
    android:id="@+id/myImageView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/fishbar"
    android:layout_marginTop="12dp"
    android:contentDescription="Image Description"
    android:gravity="center"
    android:src="@drawable/frontpageimg"
    android:layout_marginLeft="2dp"
    android:layout_marginRight="2dp"


    />
```

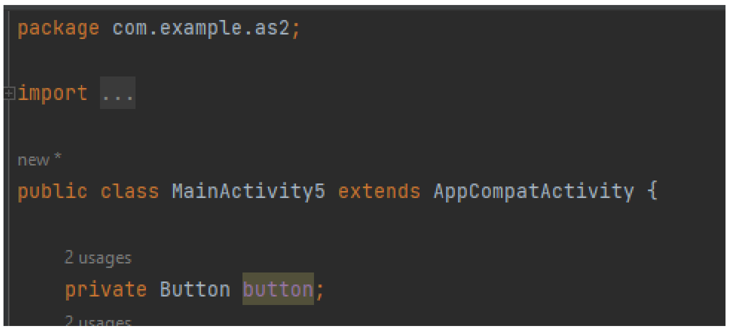
Morgan Hodge	10779528

The application heavily consisted of using buttons to navigate between the different Activities and to perform functionalities such as ‘booking’ and “saving changes”. The buttons were first declared in .XML using the tag “<com.google.android.material.button.MaterialButton>” to create the button within the file.



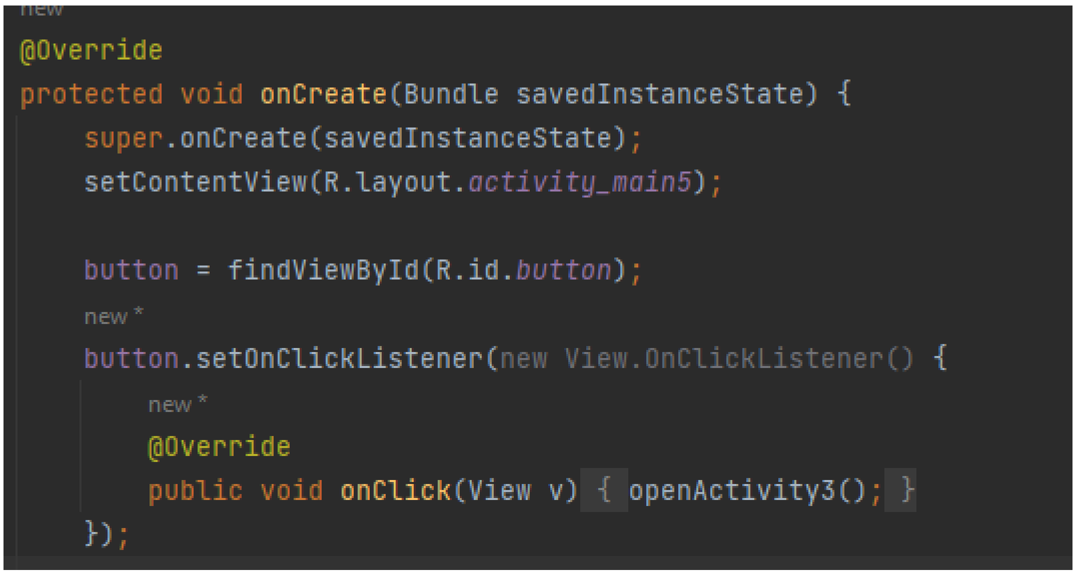
```
<com.google.android.material.button.MaterialButton
    android:id="@+id/button2"
    android:layout_width="110dp"
    android:layout_height="50dp"
    android:backgroundTint="@color/black"
    android:text="Reservation"
    android:textSize="10dp"
    android:layout_below="@id/button"
    app:iconSize="18dp"
    />
```

However, the buttons would not perform any functionalities until they were programmed in the Java class. Firstly, the button needed to be declared:



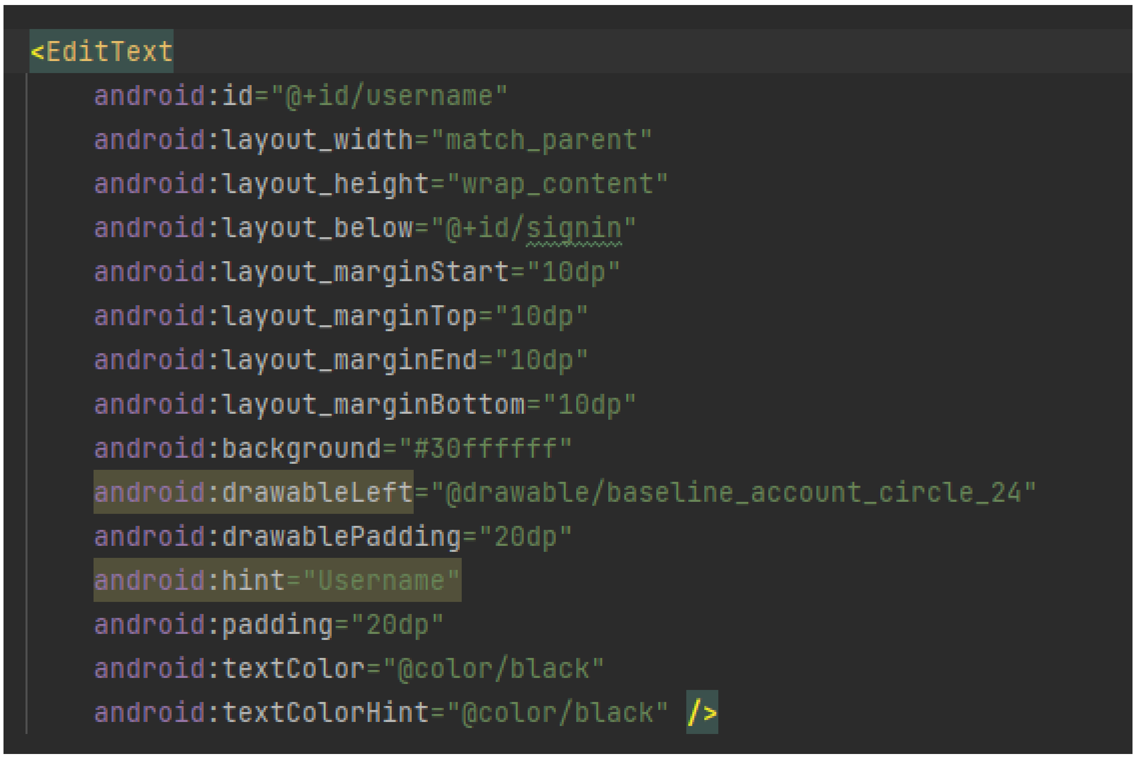
```
package com.example.as2;

import ...

new *
public class MainActivity5 extends AppCompatActivity {

    2 usages
    private Button button;
```

Then the logic for the button was written under OnCreate. The button was set up using a View.OnClickListener. When a button is clicked, it calls the ‘OpenActivity()’ method, to start another activity. This is what navigates a user from one activity to another.



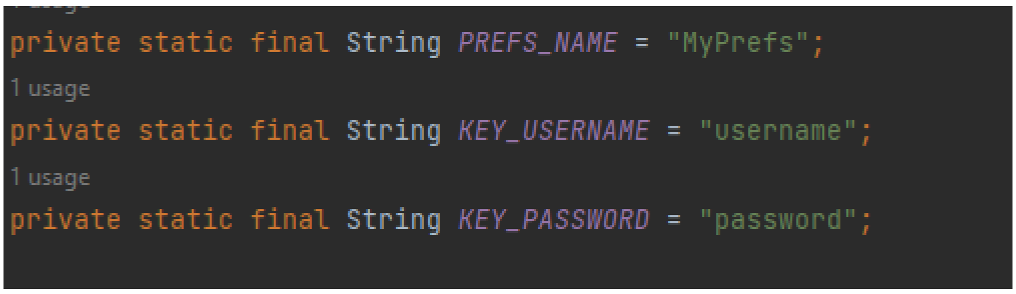
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main5);

    button = findViewById(R.id.button);
    new *
    button.setOnClickListener(new View.OnClickListener() {
        new *
        @Override
        public void onClick(View v) { openActivity3(); }
    });
```

When the user enters their username and password to log in, their credentials are stored under local storage, so the computer remembers their details for the future. In order for the users to enter any text, <EditText> had to be created within .XML. This allows users to enter and edit text within a textbox.

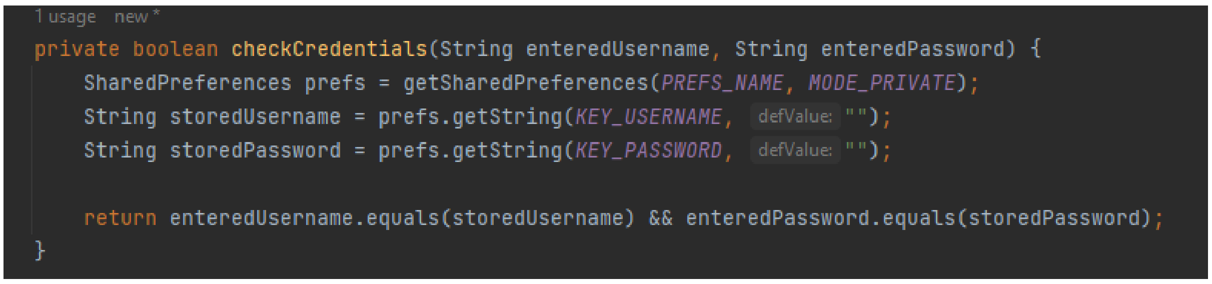


```
<EditText
    android:id="@+id/username"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/signin"
    android:layout_marginStart="10dp"
    android:layout_marginTop="10dp"
    android:layout_marginEnd="10dp"
    android:layout_marginBottom="10dp"
    android:background="#30ffffff"
    android:drawableLeft="@drawable/baseline_account_circle_24"
    android:drawablePadding="20dp"
    android:hint="Username"
    android:padding="20dp"
    android:textColor="@color/black"
    android:textColorHint="@color/black" />
```

The information that was entered then got sent and stored in local storage.

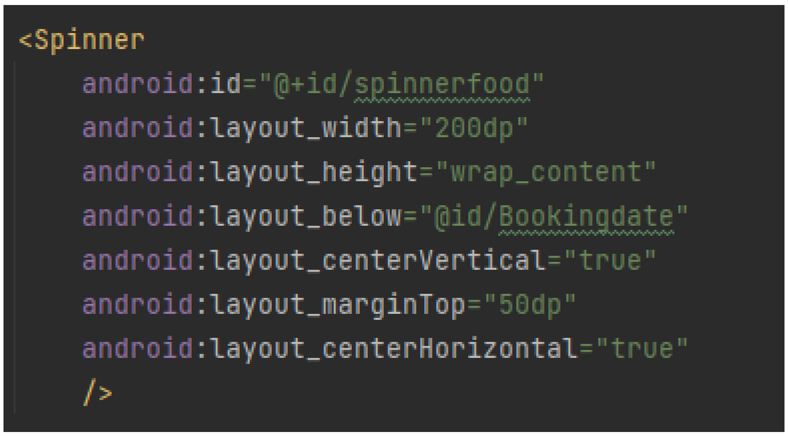


```
private static final String PREFS_NAME = "MyPrefs";
1 usage
private static final String KEY_USERNAME = "username";
1 usage
private static final String KEY_PASSWORD = "password";
```



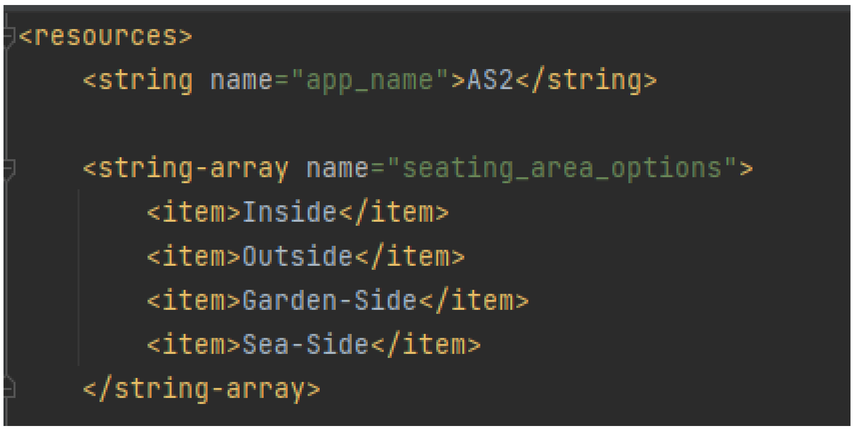
```
1 usage   new *
private boolean checkCredentials(String enteredUsername, String enteredPassword) {
    SharedPreferences prefs = getSharedPreferences(PREFS_NAME, MODE_PRIVATE);
    String storedUsername = prefs.getString(KEY_USERNAME,  defValue: "");
    String storedPassword = prefs.getString(KEY_PASSWORD,  defValue: "");

    return enteredUsername.equals(storedUsername) && enteredPassword.equals(storedPassword);
}
```

When creating the Reservation page it consisted of <TextView> ‘s and <spinner>’s for customers to enter their booking details. <Spinner>’s allows for a user to select an specific option from a variety of choices, in this instance spinners were used to select a seating location, meal time and number of guests.



```
<Spinner
    android:id="@+id/spinnerfood"
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:layout_below="@id/Bookingdate"
    android:layout_centerVertical="true"
    android:layout_marginTop="50dp"
    android:layout_centerHorizontal="true"
    />
```

To declare the attributes of a spinner, the strings.xml file had to be appended to:

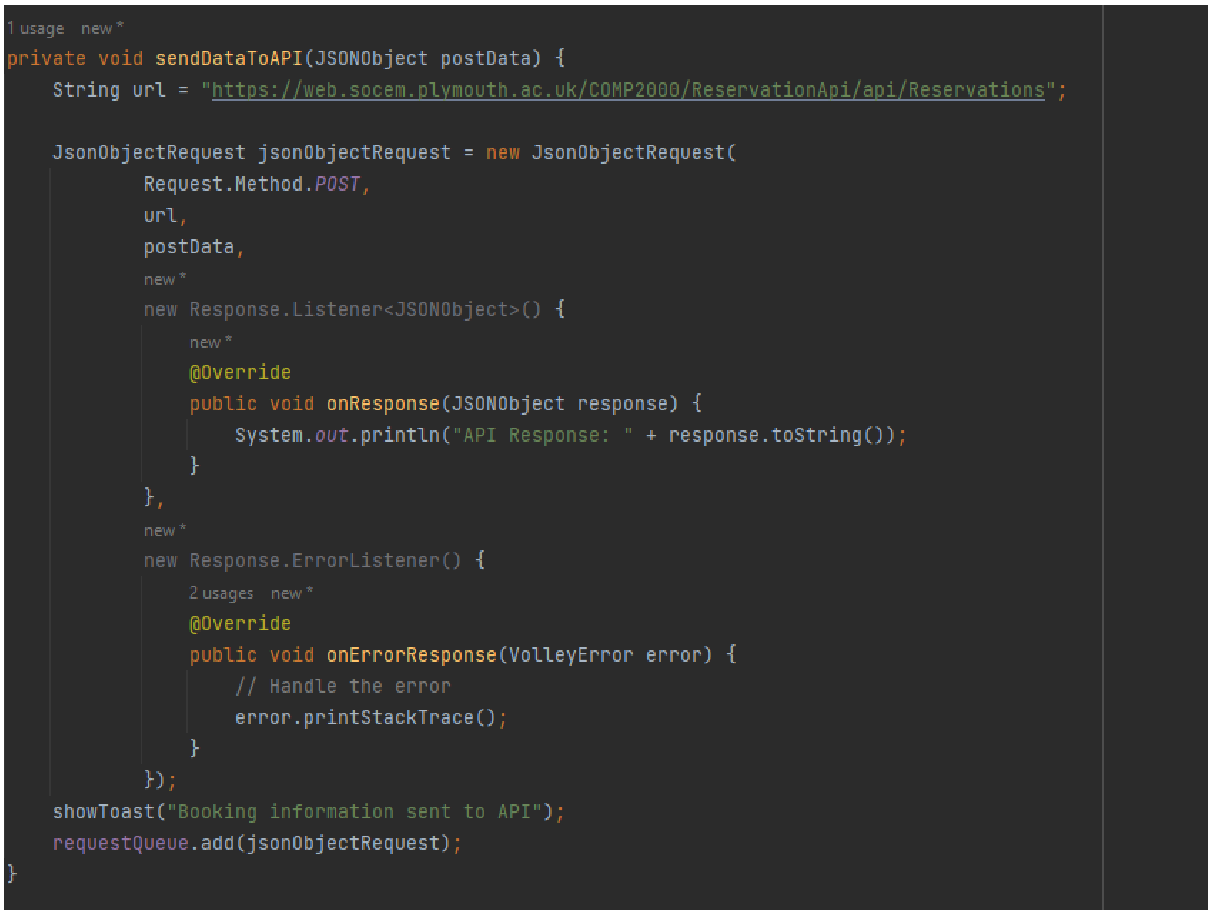


```
<resources>
    <string name="app_name">AS2</string>

    <string-array name="seating_area_options">
        <item>Inside</item>
        <item>Outside</item>
        <item>Garden-Side</item>
        <item>Sea-Side</item>
    </string-array>
```

After the user had completed the reservation form the “Book Now” button had to be clicked. Once this button has been clicked the details of the booking would be uploaded to the API.

An API is used to facilitate communication between the application and external systems or services. This can lead to a more interconnected and efficient overall system if the app ever needed to collaborate with any other services, platforms or devices.

The API was implemented using:



```
1 usage   new *
private void sendDataToAPI(JSONObject postData) {
    String url = "https://web.socem.plymouth.ac.uk/COMP2000/ReservationApi/api/Reservations";

    JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(
            Request.Method.POST,
            url,
            postData,
            new *
            new Response.Listener<JSONObject>() {
                new *
                @Override
                public void onResponse(JSONObject response) {
                    System.out.println("API Response: " + response.toString());
                }
            },
            new *
            new Response.ErrorListener() {
                2 usages   new *
                @Override
                public void onErrorResponse(VolleyError error) {
                    // Handle the error
                    error.printStackTrace();
                }
            });
    showToast("Booking information sent to API");
    requestQueue.add(jsonObjectRequest);
}
```

There was a lot more code inserted to make the API work within Activity4.java that you can view Here.

Design Patterns

The primary design pattern adopted into the implementation was the Model View Controller (MVC) Design pattern. MVC is a combination of other patterns, it is loosely followed here:

Model: ‘SharedPrefrences’ for local data storage and ‘JSONObject’ for API request payload.

View: XML layout files for defining the UI elements.

Controller: The ‘MainActivity’ class demonstrates the interaction between the model and the view, handling the user input, updating data, and triggering actions.

Testing

UI Design in terms of Heuristic Principles:

1. Visibility – The UI is tailored to give the user a more user-friendly interface. We accomplished this by having clear indicators on each button and where it will take the user, and by having background text on the <TextViews> so the users know what information to input.

2. Match – We kept the language used in the application as formal English so the users can understand the words, phrases and concepts used and not get confused.

3. User Control – 'Emergency Exits' were clearly marked in our application so the users can easily escape from pages they did not want to be on. They can press on the button in the top left to navigate safely to the correct page, or swipe left on the screen to navigate back a page.

4. Consistency – The consistency throughout the app was strong. The favourite meal options corresponded with the menu items.

5. Help Users recognize, diagnose and recover from errors – When an error occurs, for example, a user tries to log into an account that doesn't exists. The user will be greeted with a notification that states there is an error, and what the error is.

6. Error Prevention – Testing plans were created to test for errors before the launch of the app

7. Recognition - There are clear labels on buttons for users to recognise what functions they represent. The icons and photos used are clearly recognisable for the vast majority of users.

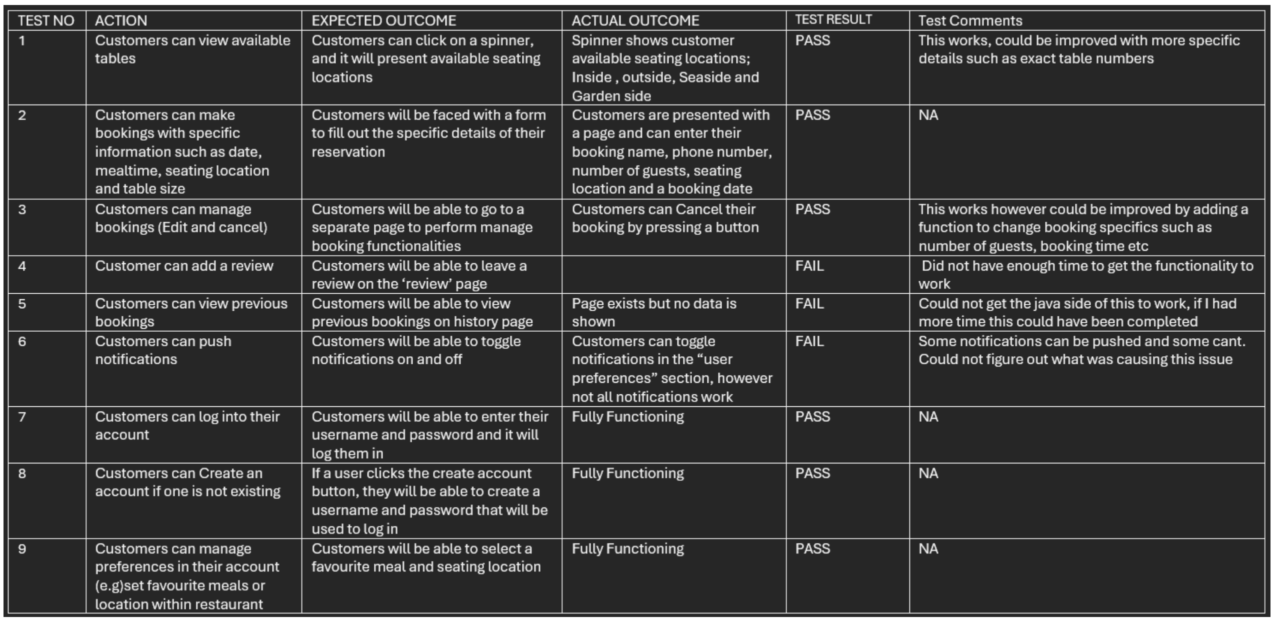
8. Flexibility – Users can customise their preferences under "User Preferences", this allows them to select a favourite seating location and set meal.

9. Aesthetic - The app presents a minimalist design by only presenting vital information to the users. There is no repetitive information, and the colours and fonts used a clear and concise.

10. Help – There are multiple indicators to the user on what data will need to be entered /selected in appropriate scenarios, so the user does not get confused.

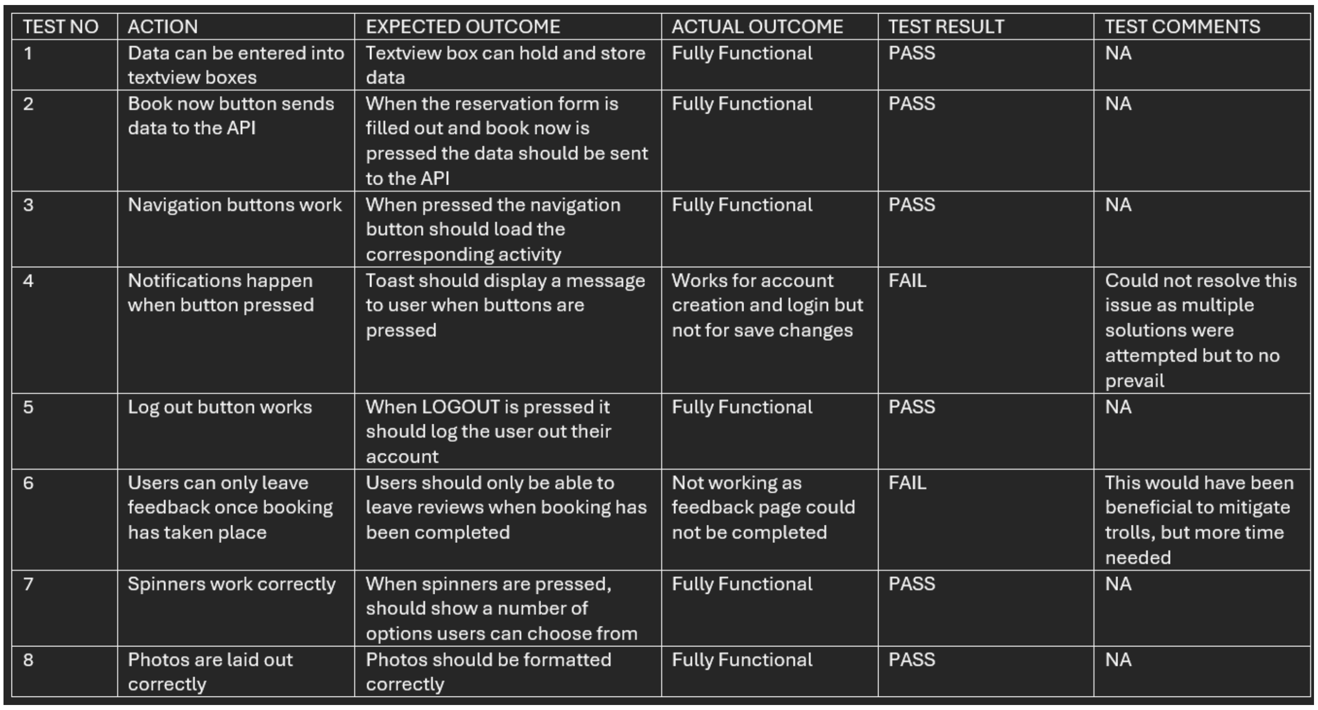
Morgan Hodge 10779528

Requirements Testing Table



TEST NO	ACTION	EXPECTED OUTCOME	ACTUAL OUTCOME	TEST RESULT	Test Comments
1	Customers can view available tables	Customers can click on a spinner, and it will present available seating locations	Spinner shows customer available seating locations; Inside , outside, Seaside and Garden side	PASS	This works, could be improved with more specific details such as exact table numbers
2	Customers can make bookings with specific information such as date, mealtime, seating location and table size	Customers will be faced with a form to fill out the specific details of their reservation	Customers are presented with a page and can enter their booking name, phone number, number of guests, seating location and a booking date	PASS	NA
3	Customers can manage bookings (Edit and cancel)	Customers will be able to go to a separate page to perform manage booking functionalities	Customers can Cancel their booking by pressing a button	PASS	This works however could be improved by adding a function to change booking specifics such as number of guests, booking time etc
4	Customer can add a review	Customers will be able to leave a review on the 'review' page		FAIL	Did not have enough time to get the functionality to work
5	Customers can view previous bookings	Customers will be able to view previous bookings on history page	Page exists but no data is shown	FAIL	Could not get the java side of this to work, if I had more time this could have been completed
6	Customers can push notifications	Customers will be able to toggle notifications on and off	Customers can toggle notifications in the "user preferences" section, however not all notifications work	FAIL	Some notifications can be pushed and some cant. Could not figure out what was causing this issue
7	Customers can log into their account	Customers will be able to enter their username and password and it will log them in	Fully Functioning	PASS	NA
8	Customers can Create an account if one is not existing	If a user clicks the create account button, they will be able to create a username and password that will be used to log in	Fully Functioning	PASS	NA
9	Customers can manage preferences in their account (e.g)set favourite meals or location within restaurant	Customers will be able to select a favourite meal and seating location	Fully Functioning	PASS	NA

Implementation Testing Table



TEST NO	ACTION	EXPECTED OUTCOME	ACTUAL OUTCOME	TEST RESULT	TEST COMMENTS
1	Data can be entered into textview boxes	Textview box can hold and store data	Fully Functional	PASS	NA
2	Book now button sends data to the API	When the reservation form is filled out and book now is pressed the data should be sent to the API	Fully Functional	PASS	NA
3	Navigation buttons work	When pressed the navigation button should load the corresponding activity	Fully Functional	PASS	NA
4	Notifications happen when button pressed	Toast should display a message to user when buttons are pressed	Works for account creation and login but not for save changes	FAIL	Could not resolve this issue as multiple solutions were attempted but to no prevail
5	Log out button works	When LOGOUT is pressed it should log the user out their account	Fully Functional	PASS	NA
6	Users can only leave feedback once booking has taken place	Users should only be able to leave reviews when booking has been completed	Not working as feedback page could not be completed	FAIL	This would have been beneficial to mitigate trolls, but more time needed
7	Spinners work correctly	When spinners are pressed, should show a number of options users can choose from	Fully Functional	PASS	NA
8	Photos are laid out correctly	Photos should be formatted correctly	Fully Functional	PASS	NA

Morgan Hodge 10779528

Evaluation

The design completes the goal of gathering and storing user bookings. However, there are areas for improvement such as the review page is not functional, I could not get the review page to work in the amount of time I had left, despite working on this project almost every day for 3 weeks I could not get this section on the app to work.

I could also improve on the notifications as only some features displayed notifications on the screen. I am unsure why some did not work as the toast command was in the correct space, despite trying many different methods, I could not get the notification to work on every aspect of the design.

The way users select a booking date in this design is by typing the date in a textview using YY-MM-DD format. I believe if I had more time, I could have improved this by using a calendar view as this would have been a more user-friendly way than entering the date manually.

In conclusion I believe that despite not completing every aspect of the design, the end goal was still good. I believe this because it performs many of the requirements and can handle bookings to the API. In the future I will need to ensure I manage my time better so I can complete a more thorough and detailed app.