

**School of Computer Science & Engineering**

**SC2006 Software Engineering Lab Group Z60**

**Software Requirement Specification**

for

**ParkIt**

**Prepared By :**

**TEAM SOFTIES**

Hsieh Boh Yang (U2122869J)

Khoo Yong Quan

Lou Sim Teng (U2122859L)

Li Luyun (U2120326J)

Leong Kai Yang (U2121339E)

Lim Perry

15th April 2023

**Table of Contents**

[**1. Introduction 4**](#_heading=h.1fob9te)

[**1.1 Purpose 4**](#_heading=h.3znysh7)

[**1.2 Document Conventions 4**](#_heading=)

[**1.3 Intended Audience and Reading Suggestions 4**](#_heading=)

[**1.4 Product Scope 4**](#_heading=)

[**2. Overall Description 5**](#_heading=)

[**2.1 Product Perspective 5**](#_heading=h.2s8eyo1)

[**2.2 Product Functions 6**](#_heading=)

[**2.3 User Classes and Characteristics 6**](#_heading=)

[**2.4 Operating Environment 7**](#_heading=)

[**2.5 Design and Implementation Constraints 7**](#_heading=)

[**2.6 User Documentation 7**](#_heading=)

[**2.7 Assumptions and Dependencies 7**](#_heading=)

[**3. External Interface Requirements 8**](#_heading=)

[**3.1 User Interfaces 8**](#_heading=h.2jxsxqh)

[**3.2 Hardware Interfaces 11**](#_heading=)

[**3.3 Software Interfaces 11**](#_heading=)

[**3.4 Communications Interfaces 11**](#_heading=)

[**4. Functional Requirements 12**](#_heading=)

[**4.1 Login 12**](#_heading=h.r6pmjy6jn0i3)

[**4.2 Sign Up 12**](#_heading=h.37bbyvu02nb2)

[**4.3 Forget Password 12**](#_heading=h.9op1n5y4then)

[**4.4 Map Screen 12**](#_heading=h.lyy4wnuefa3u)

[**4.5 Settings Screen 13**](#_heading=h.vsh3ha8v3lj1)

[**5. Other Nonfunctional Requirements 13**](#_heading=h.3whwml4)

[**5.1 Performance Requirements 13**](#_heading=h.2bn6wsx)

[**5.2 Reliability Requirements 13**](#_heading=)

[**5.3 Safety Requirements 13**](#_heading=h.vkttfy8h63o4)

[**5.4 Security Requirements 13**](#_heading=)

[**6. Data Dictionary 14**](#_heading=)

[**7. Other Requirements 14**](#_heading=h.cqe18xmg0t7d)

[**7.1 Installations 14**](#_heading=h.3pns0pt2ydg6)

[**7.2 Running of App 15**](#_heading=h.bem0wpfb3x3f)

[**8. Appendix 15**](#_heading=h.oocb9wxyv0t)

[**8.1 Use Case Diagram and Description 15**](#_heading=h.xfngee8sn47k)

[**8.2 Class Diagram 21**](#_heading=h.wodsfqjcjmqf)

[**8.3 State Machine Diagram / Dialog Map 22**](#_heading=h.52ey4d7r9wqt)

[**8.4 System Architecture Diagram 23**](#_heading=h.hvi82or0ohmv)

[**8.5 Sequence Diagrams 24**](#_heading=h.512lxxhgg3ah)

[**8.6 Unit Testing 29**](#_heading=h.rtwf36pspfls)

[**8.6.1 Black Box Testing 29**](#_heading=h.i18opwv59iq4)

[**8.6.2 White Box Testing 40**](#_heading=h.wsu1dqiiawfd)

[**8.7 App Demo 43**](#_heading=h.6651obmw1mrj)

# Introduction

## Purpose

This SRS provides a detailed description of ParkIt’s software requirements. It outlines the functional and non-functional requirements of our mobile application, including our use case models, class diagrams, sequence diagrams, state machine diagram, system architecture diagram and other documentation. This document will also serve as a reference for the development team, stakeholders and clients, ensuring everyone involved in this project is clear and understands its objectives.

## Document Conventions

This SRS document follows a font style of Times, font size 18 for the headings. The sub headings follow a font style of times, font size 14. The rest of the document follows a font style of Arial, font size 11.

## Intended Audience and Reading Suggestions

This SRS document is intended for all stakeholders involved in the making and implementation of the ParkIt software system, including:

1.3.1 Software Developers

To understand the constraints and technical details of the system.

1.3.2 Project Managers

To ensure the project is on track with time, within budget, and according to client’s needs.

1.3.3 Quality Control Testers

To ensure the system meets the functional and non-functional requirements.

1.3.4 End Users

To aid users in understanding how ParkIt’s features work

## Product Scope

ParkIt is a cross platform mobile application built using React Native and Expo CLI. It can alleviate the stress of drivers who have difficulty in finding parking lots. ParkIt displays the car parks within a 2km radius of the user-input destination, and provides users with an option to switch to Google Maps for directions to the selected car park. Along with the list of car parks displayed to the user, ParkIt provides users with their real-time availability as well as pricing information.

# Overall Description

## Product Perspective

ParkIt is a mobile application created to help users locate nearby car parks at a selected destination. Our app also shows the rates and number of available lots of these car parks to aid in their decision of selecting a desired car park.

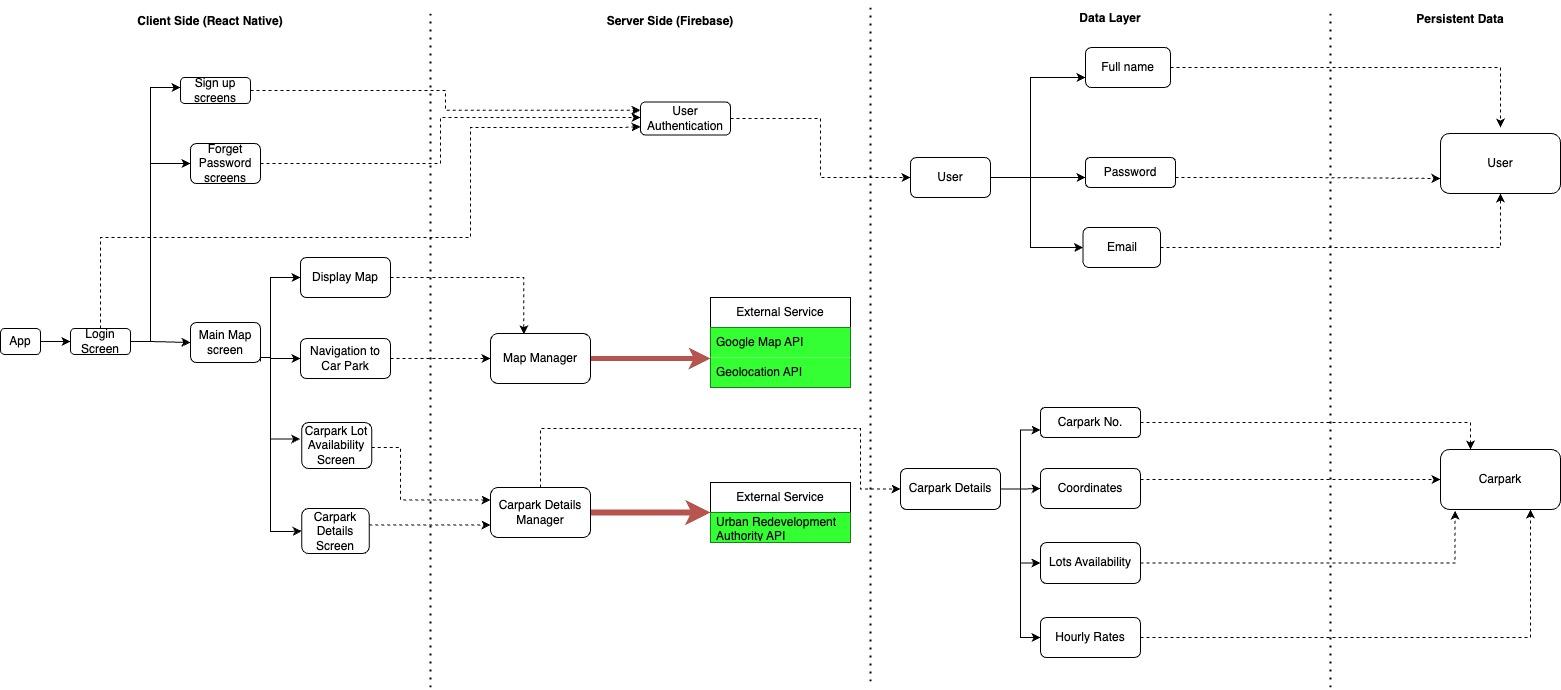
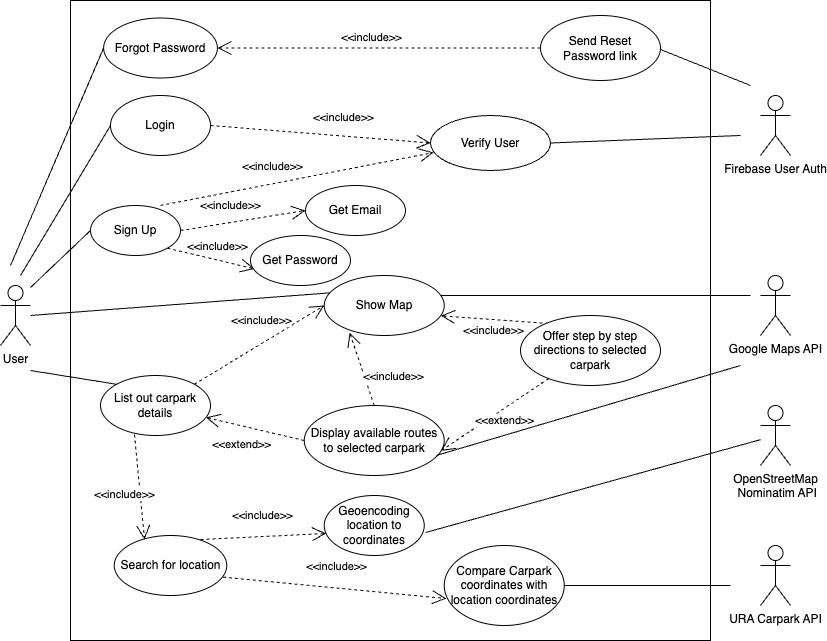


Diagram 1: ParkIt’s System Client-Server Architecture Diagram

ParkIT makes use of Application Programming Interfaces(API) such as Urban Redevelopment Authority API and Google Maps API for car park details as well as navigation.

## Product Functions



ParkIt must minimally be able to:

* Register users with a new account
* Allow users to login to their account with their verified email and password
* Allow users to search for a location and display the details of car parks within a 2km radius around it
* Allow users to reset their password for their account
* Allow users to logout of their accounts

## User Classes and Characteristics

Target users are those who are unfamiliar with car parks around Singapore and would like more convenience in searching for a suitable car park to park their vehicles at.

## Operating Environment

ParkIt works on both Android and IOS devices as a mobile application. It uses Firebase for user authentication. Access to external APIs such as URA.gov API, Google Maps API and OpenStreetMap Nominatim API are also required for full functionality of our ParkIt application.

## Design and Implementation Constraints

ParkIt is currently only available in English and thus might not be usable for users who are unfamiliar with English. Additionally, ParkIt depends on the URA.gov API to provide real time information on the car parks. Thus, any availability or accuracy issues with the API will result in the same issues on ParkIt. Lastly, ParkIt also uses OpenStreetMap Nominatim API, which is an API that converts worldwide text location into coordinates. The international nature of the API means users have to enter Singapore specific locations e.g “Bukit Timah” or add “Singapore” to their text e.g “Orchard Singapore” or risk converting a location that is not in Singapore.

## User Documentation

ParkIt’s source code has been made available at <https://github.com/bohyanggg/ParkIt> for public access. A README.md file has been provided to guide developers on cloning the app if they wish to further develop it. A similar guide can also be found in part 7 of this SRS.

## Assumptions and Dependencies

2.7.1 Availability of Internet Connection: It is assumed that users of the application will have a stable and reliable internet connection to access the application and its features.

2.7.2 Proper Input from Users: It is assumed that users will provide accurate and valid input while using the application, such as entering a valid text location into the search bar

2.7.3 Third-Party APIs: The application may have dependencies on third-party APIs for features such as carpark details data. Any changes or updates to these third-party APIs may impact the functionality of the application.

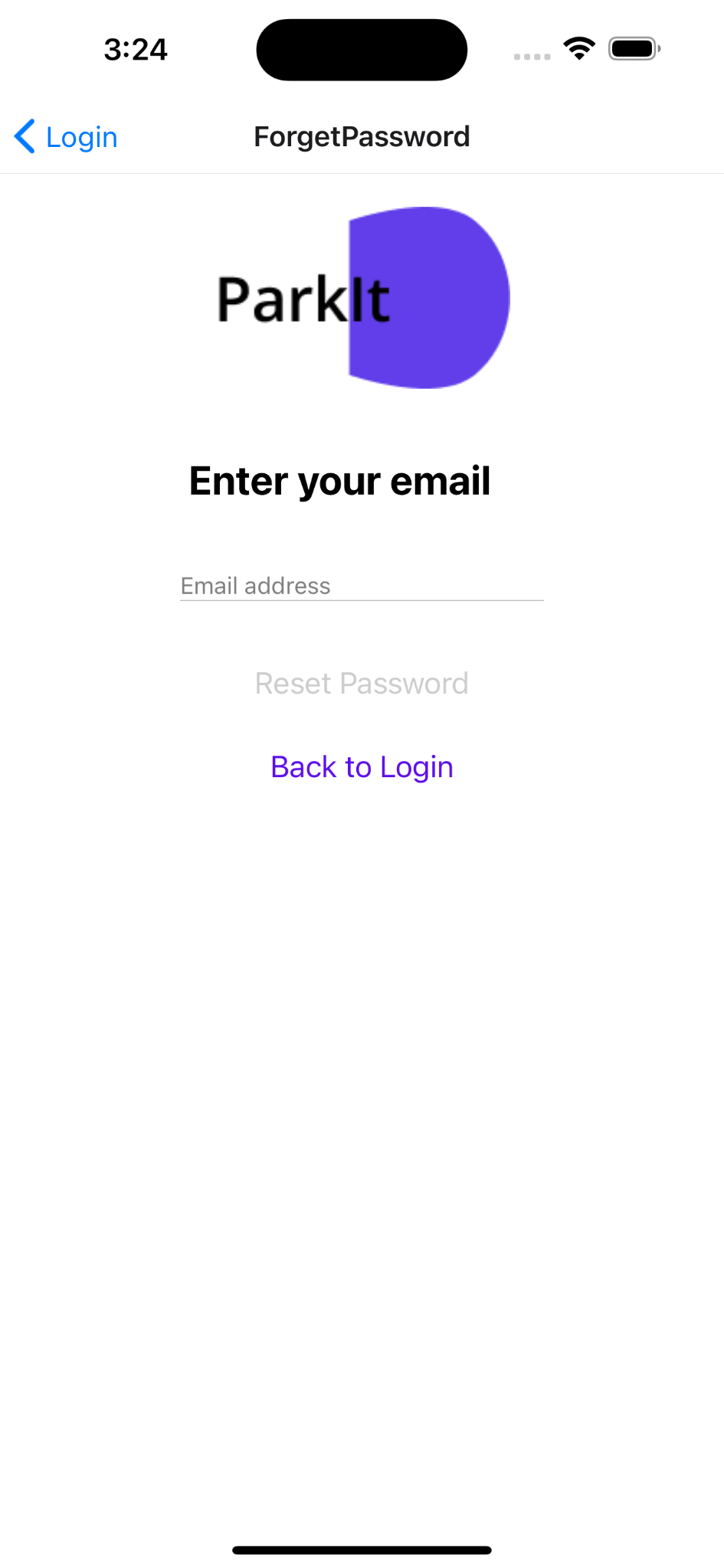
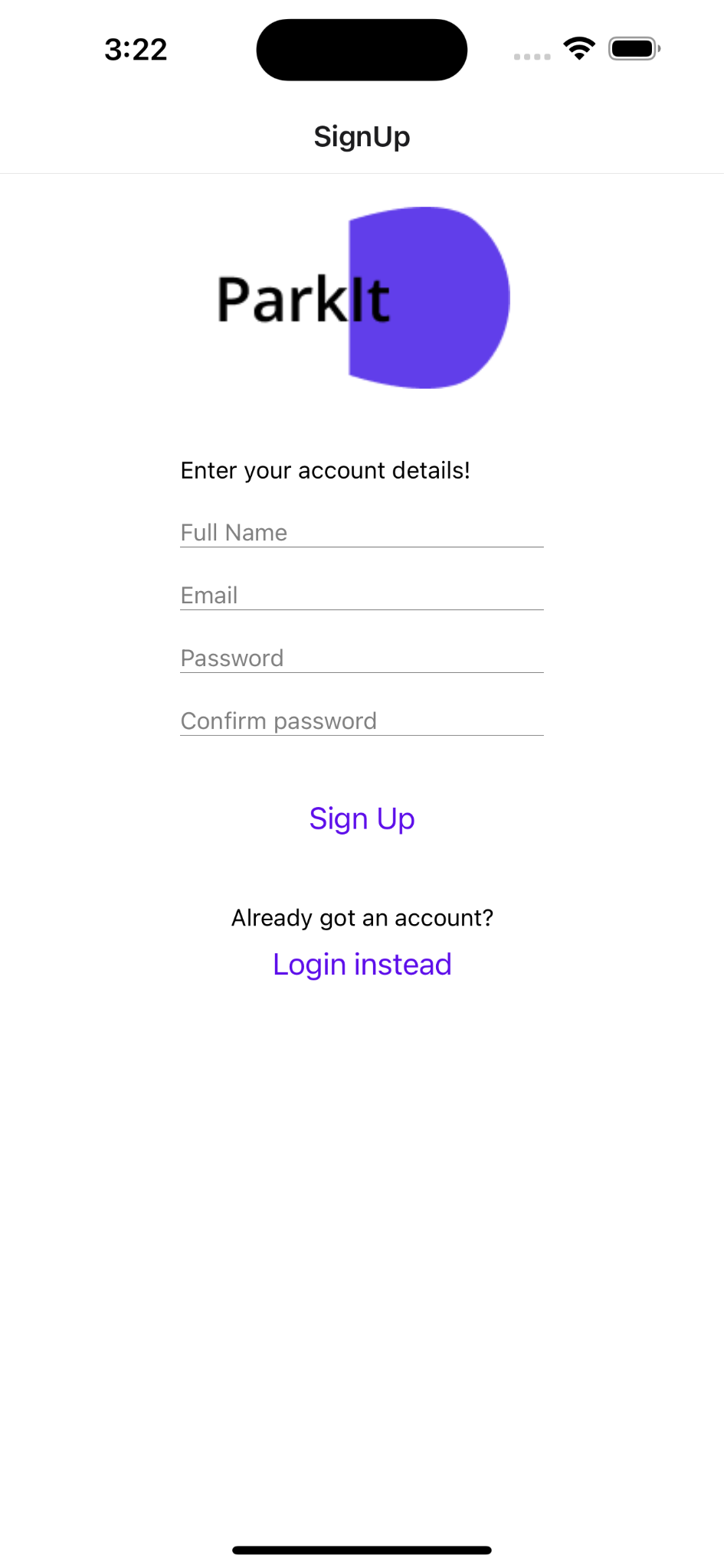
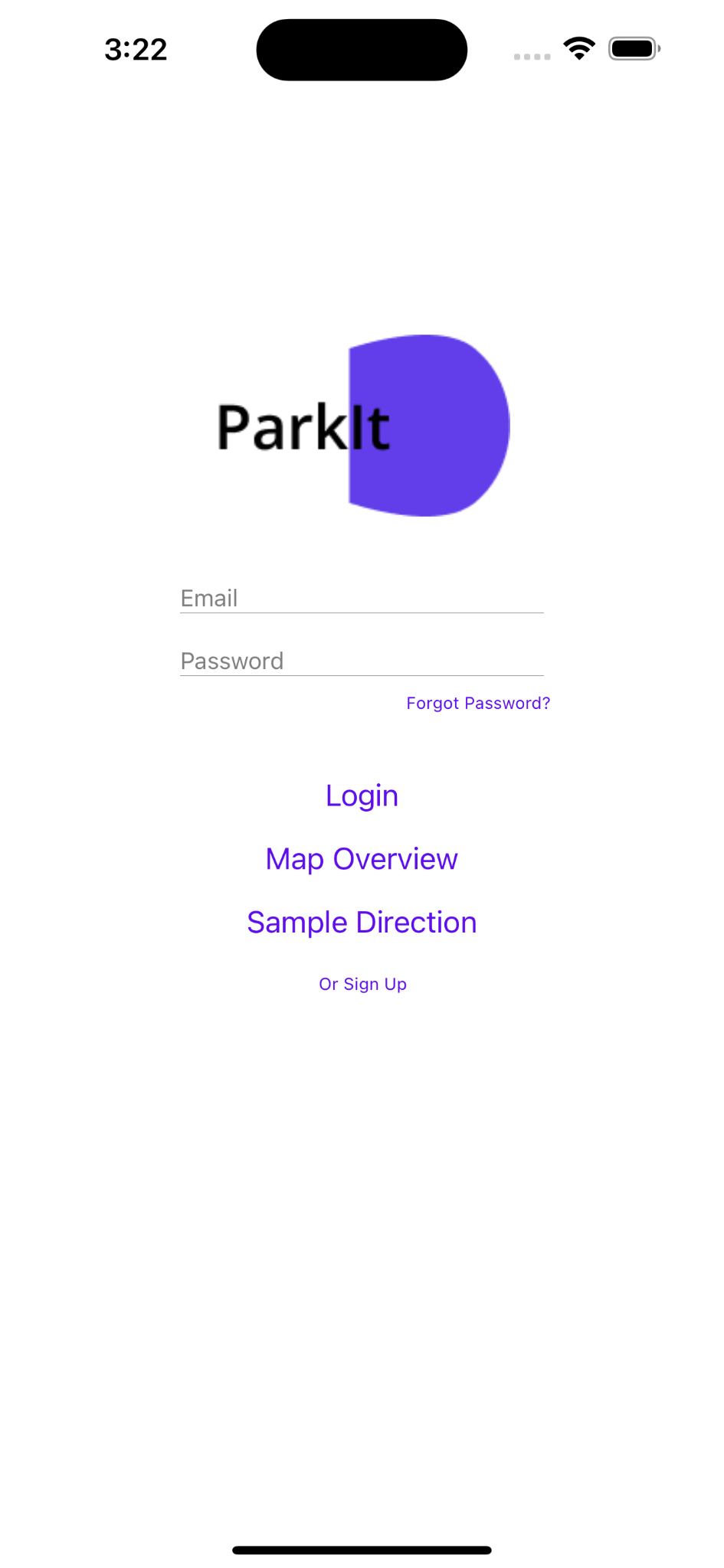
2.7.4 External Services: The application may rely on external services, such as Firebase, email notifications, which may have their own dependencies and limitations that can affect the overall performance and availability of the application.

# External Interface Requirements

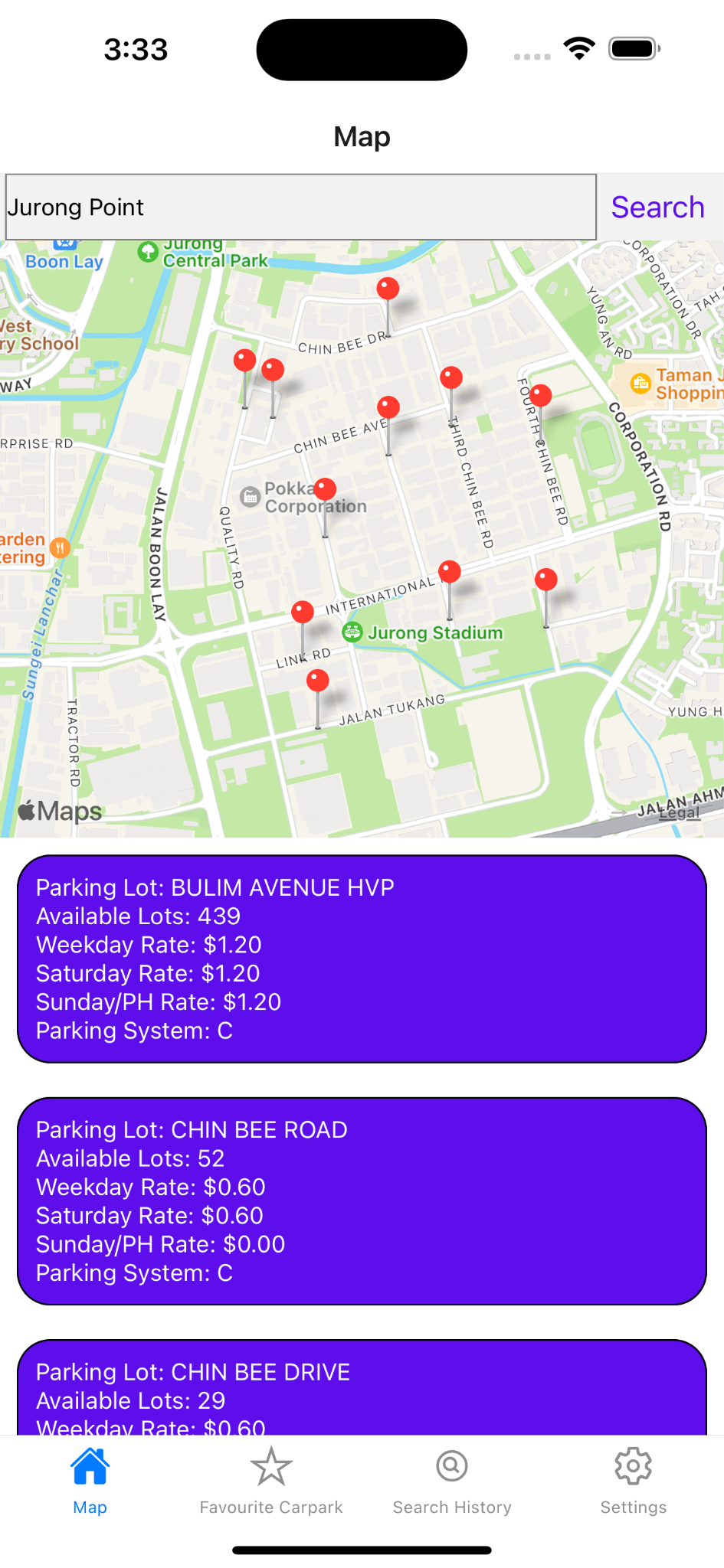
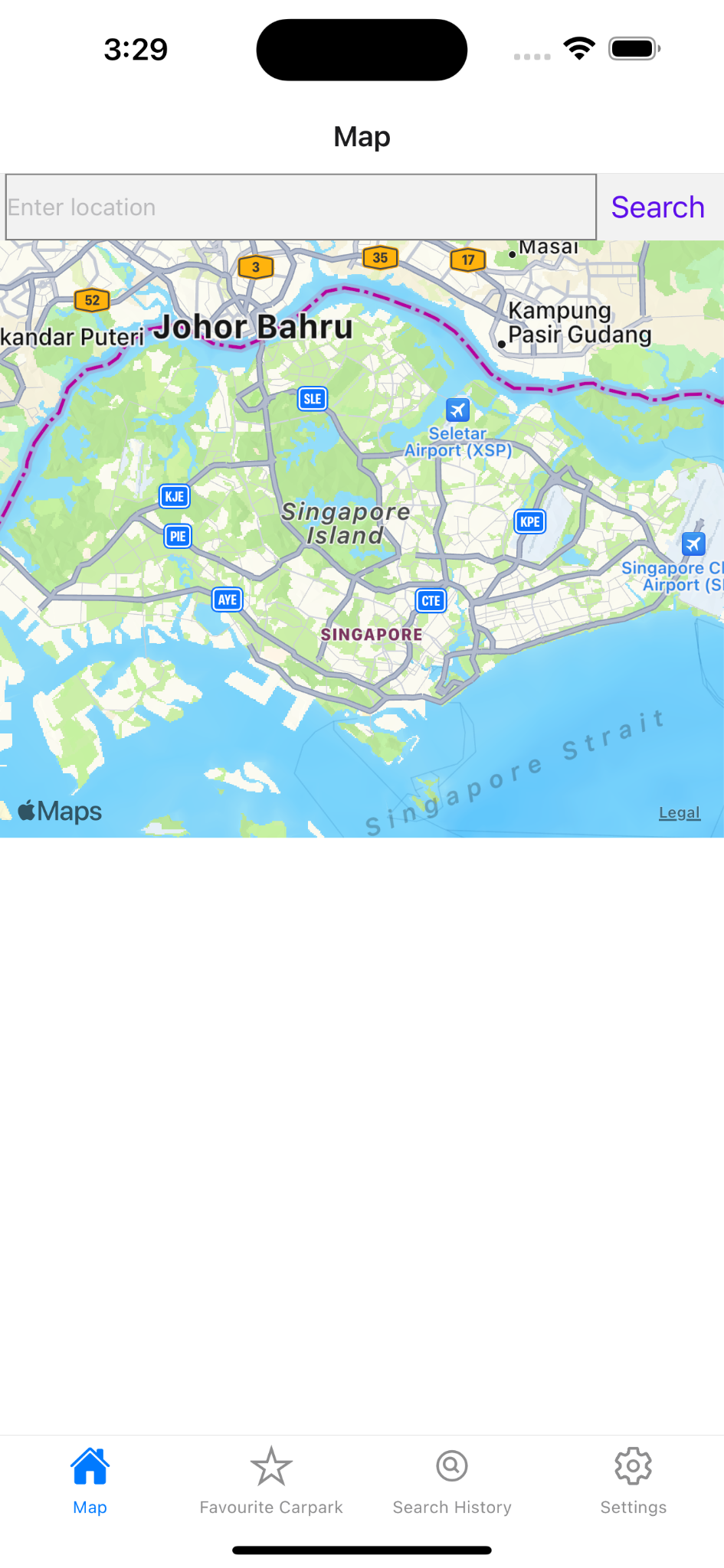
## User Interfaces

Users can access the various features of ParkIt through its highly interlinked screens. Users will be able to navigate through the various screens through buttons.

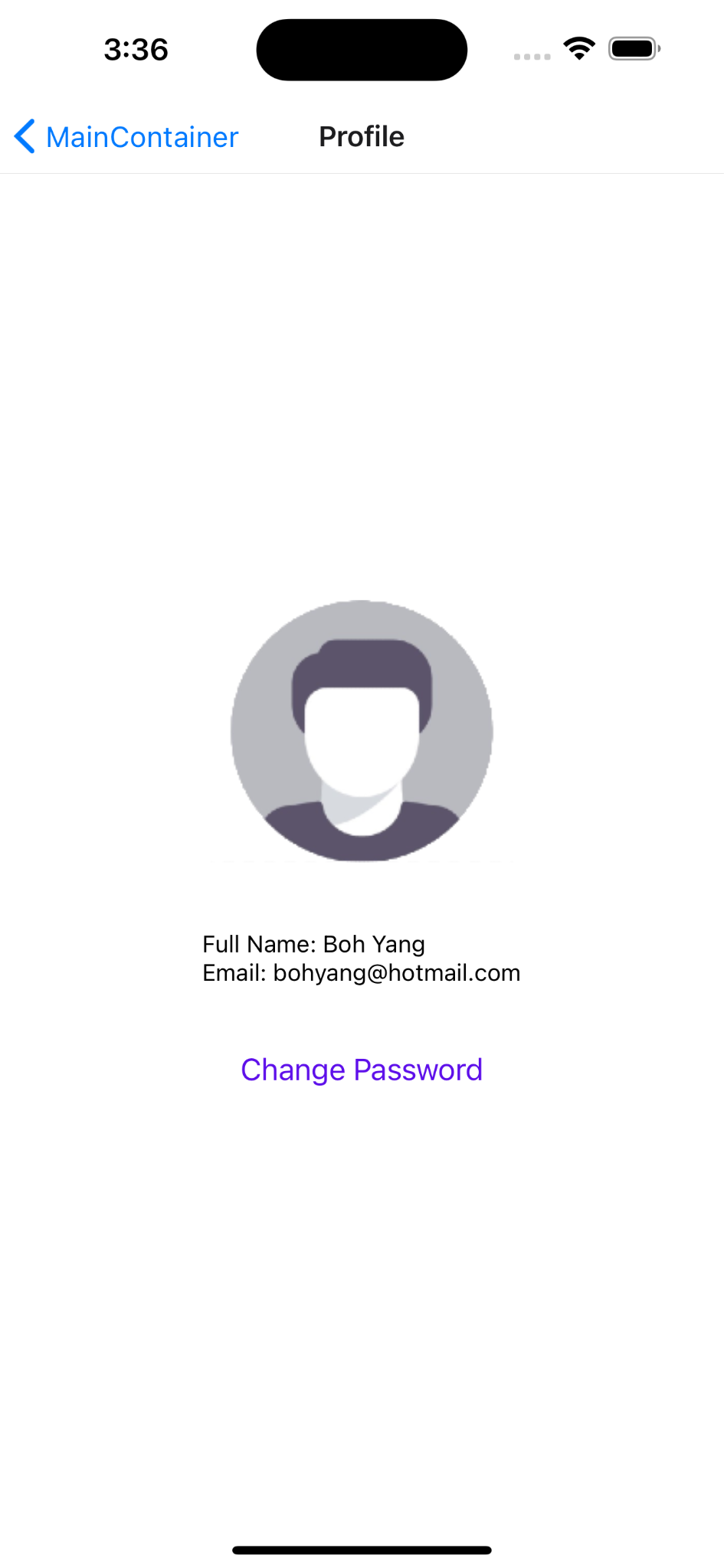
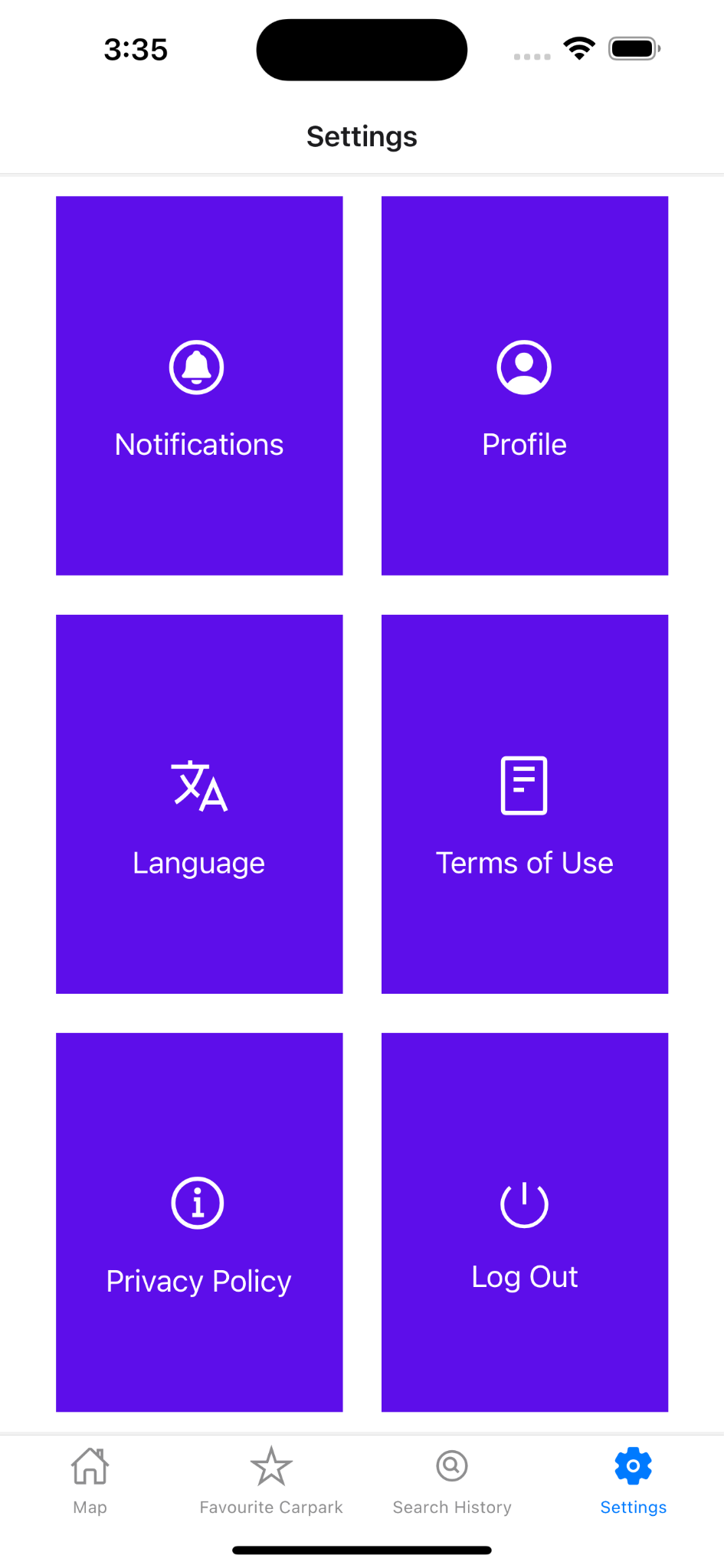
Of the 3 screens shown below, the one on the left shows the login screen where users are prompted to enter their email and password to gain access to their account. The middle screen shows the sign-up screen where users are prompted for their details such as full name, email, and password. The screen on the right shows the forget password screen where users are prompted to enter their email to receive a recovery mail for them to reset their passwords.

****

Moving on to the main screens as shown below. The screen on the left shows the map upon opening. After the user inputs a location into the search bar, a list of nearby car parks are listed out and markers are placed on their locations to allow easier viewing by the user.



Lastly, we have the settings screen (left) , which consists of the profile screen (right) as well as a logout button.



## Hardware Interfaces

The logical and physical characteristics of the interfaces between the software product and the hardware components of the system in ParkIt includes the use of APIs to obtain data and services over the internet, and the use of standard communication protocols to interact with the hardware components of the device. The OpenStreetMap Nominatim API provides geocoding services that can be used to convert a location (e.g. an address or place name) to its corresponding coordinates (latitude and longitude) in WGS84 projection. The URA API provides data on government operated car parks in Singapore while Google Maps API provides mapping, geolocation services and a range of functions for displaying maps, markers, and other geographic data. The APIs also require a valid API key, which is used to authenticate requests and track usage. URA API also requires users to get a new token every 24 hours using the API key. The above APIs can be queried using HTTP requests, and returns data in JSON format, which can be easily parsed and used by the app.

ParkIt can be used on any device that supports React Native and JavaScript, and uses standard protocols for communicating with the APIs, including HTTP/HTTPS for the OpenStreetMap Nominatim and URA APIs, and JavaScript API calls for the Google Maps API. ParkIt interacts with the hardware components of the device to obtain user input, display output, and communicate with the APIs.

## Software Interfaces

ParkIt is built using React Native and Firebase. React Native is a JavaScript framework for building user interfaces. Firebase provides the backend features for the application such as the user authentication. ParkIt also integrates with external APIs such as Google Maps and URA API.

The list of dependencies for ParkIt can be found in the ‘package.json’ file within the source code.

## Communications Interfaces

ParkIT requires an internet connection and must be able to communicate with components such as routers and servers to connect to Firebase. Data transfer rate will be determined by the user’s internet speed. The Firebase server provides data synchronization, ensuring that the app always displays the most up-to-date information. Communication security is also provided by Firebase, which uses SSL/TLS encryption to protect data communication.  
  
 ParkIT also requires communication with external APIs such as Google Maps API and URA API.

# Functional Requirements

## Login

4.1.1 The system must allow for users to log in with their verified email and password.

4.1.1.1 The system must verify that the user’s account is verified.

4.1.1.2 The system must be able to direct users to the Map Screen after login.

## Sign Up

4.2.1 The system must allow for users to sign up for an account.

4.2.1.1 The system must request for the user's full name, email and password.

4.2.1.2 The system must ensure a valid email address was keyed in

4.2.1.3 The system must ensure that the password keyed in has at least 8 characters

4.2.1.4 The system must require the user to type in their password twice to ensure the accuracy of the password

4.2.1.5 The system must issue a verification email for the user to verify their account

## Forget Password

4.3.1 The system must allow for users to reset their account password.

4.3.1.1 The system must be able to send a link to the user’s email to allow them to

reset their password

## Map Screen

4.4.1 The system must be able to display the user’s current location

4.4.2 The system must allow for users to search for a location

4.4.2.1 The system must be able to display a map view of the user’s selected location and the surrounding area

4.4.2.2 The system must be able to display the car parks within a 2km radius of the selected location and highlight them on the map.

4.4.2.3 The system must be able to display the car park specific information for each of the carparks

## Settings Screen

4.5.1 The system must be able to display the user’s profile information

4.5.2 The system must be able to display a button for the user to log out of their account.

# Other Nonfunctional Requirements

## Performance Requirements

5.1.1 ParkIt must be compatible with the latest version of Android and IOS.

5.1.2 Response time of ParkIt should be no longer than 3 seconds after user’s input.

5.1.3 The car park details and makers should render within 30 seconds of user search.

5.1.4 ParkIt must be able to store at least 10000 user account records.

## Reliability Requirements

5.2.1 ParkIt must ensure that errors do not crash the application.

## Safety Requirements

5.3.1 Error messages should be displayed to the user if invalid inputs are entered

## Security Requirements

5.4.1 ParkIt must ensure that every user has a valid email linked to a verified account.

5.4.2 ParkIt must ensure that users will only have access to its main features if they are logged in with a verified account.

# Data Dictionary

|  |  |
| --- | --- |
| Term | Definition |
| Route | A path the user can take to get from point A to point B |
| Car Park | An area or building where cars or other vehicles may be parked |
| Vehicle | A vessel used for transporting people or goods, especially on land, such as a car, motorbike, or electric vehicle |
| Car Park Lots | Spots that a vehicle could be parked at, usually represented by a number |
| Season Parking | A scheme that allows users to park their vehicle at a carpark on a regular basis at a fixed monthly rate for different types of vehicles and at different season parking rates. |
| URA | The Urban Redevelopment Authority is the national urban planning authority of Singapore, and a statutory board under the Ministry of National Development of the Government of Singapore. |

# Other Requirements

## Installations

7.1.1 To install and run the app, you will need to have Node.js and npm installed on your computer. Node.js is a runtime environment that allows you to run JavaScript code outside of a browser. npm is a package manager that lets you install and manage dependencies for your project.

7.1.2 To install Node.js and npm, you can follow the instructions on their official website: <https://nodejs.org/en/>

7.1.3 To run the app on your mobile device, you will need to have the Expo Go app installed on your device. To install Expo Go, you can follow the instructions from: <https://expo.dev/client>

7.1.3 Once you have Node.js and npm installed, you need to install the node modules that are required for the app. These are the libraries and packages that provide various functionalities and features for the app.

7.1.4 To install the node modules, you need to navigate to the project folder in your terminal and run the following command: npm install

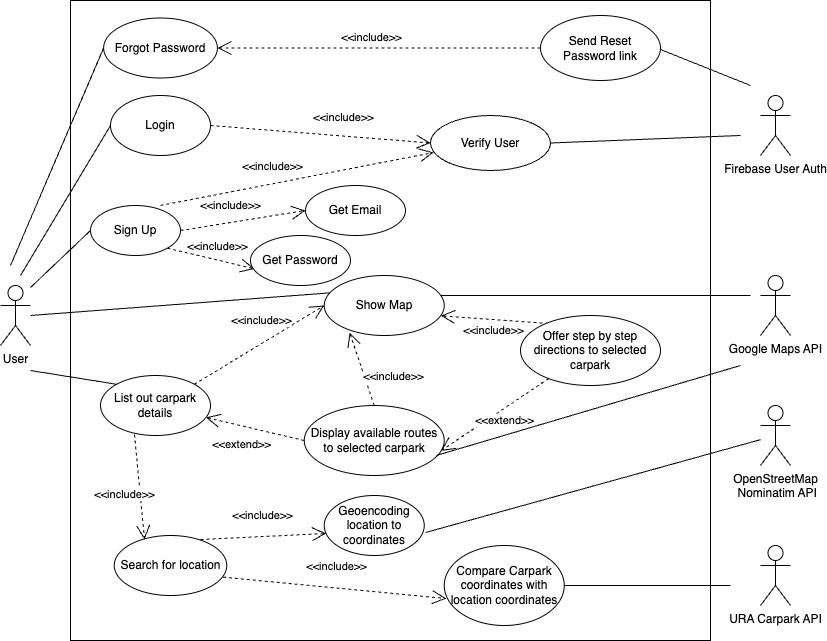
## Running of App

7.2.1 To run the app, enter the following command in your terminal: npx expo start

7.2.2 This will start a local server and provide you with a QR code which you can scan on your device and subsequently launch the Expo Go app.

# Appendix

## Use Case Diagram and Description



|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | CPK.001 | | |
| Use Case Name: | Sign up | | |
| Created By: |  | Last Updated By: | Boh Yang |
| Date Created: | 09/02/2023 | Date Last Updated: | 15/04/2023 |

|  |  |
| --- | --- |
| Actor: | App User (Initiating actor), Firebase |
| Description: | User signs up for an account |
| Preconditions: | 1. User has a valid email address which is not yet linked to an account |
| Postconditions: | 1. User creates an account |
| Priority: | Essential |
| Frequency of Use: | Only once per user |
| Flow of Events: | 1. Users will click on Sign up button 2. Users will be brought to the Sign up page 3. Users will be prompted to enter an email address, a password as well as a password confirmation. 4. System will send a verification email with a unique link to the user's email address. 5. Users will click the link and verify their account 6. Firebase saves the user’s account information. |
| Alternative Flows: | AF1: User already has an account and clicks on login instead   1. System directs user to login page |
| Exceptions: | EX1: User’s email address cannot be verified   1. System gives an “invalid email” alert and prompts user to re-enter their password   EX2: User’s email address is already linked to an account   1. System gives an “email already in use” alert and prompts user to login instead |
| Includes: | NIL |
| Special Requirements: | 1. App Account System must store up to tens of thousands of users account data |
| Assumptions: | NIL |
| Notes and Issues: | NIL |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | CPK.002 | | |
| Use Case Name: | Login | | |
| Created By: |  | Last Updated By: | Boh Yang |
| Date Created: | 09/02/2023 | Date Last Updated: | 15/04/2023 |

|  |  |
| --- | --- |
| Actor: | App user (Initiating actor), Firebase |
| Description: | App user logs in to their account or resets their password |
| Preconditions: | 1. App user has an account |
| Postconditions: | 1. App user logs in to their account |
| Priority: | Essential |
| Frequency of Use: | Low, only log in once, will remain logged in unless user logs out |
| Flow of Events: | 1. User chooses the login option 2. User enters their username and password in the login page 3. App verifies the account login information with the App Account System 4. If account information is verified, user is logged in to the app |
| Alternative Flows: | AF Step 2: If user forgets their account password   1. User chooses the forget password option 2. User is brought to the app account password reset screen 3. A verification link is sent to the user’s registered email 4. User clicks on verification link 5. If successful, user is prompted to enter a new password 6. The password is updated and the account login information is updated in the App Account System 7. User returns to login page |
| Exceptions: | NIL |
| Includes: | NIL |
| Special Requirements: | 1. App must respond to user’s actions within 3s 2. App Account System must store up to thousands of users account data 3. App must recover from errors within 1 minute |
| Assumptions: | NIL |
| Notes and Issues: | NIL |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | CPK.003 | | |
| Use Case Name: | Display current location | | |
| Created By: |  | Last Updated By: | Boh Yang |
| Date Created: | 09/02/2023 | Date Last Updated: | 15/04/2023 |

|  |  |
| --- | --- |
| Actor: | App User (Initiating actor) |
| Description: | System displays current location of user |
| Preconditions: | 1. User has an app account 2. User is logged into their account 3. User has a GPS-enabled device |
| Postconditions: | Current location of driver is shown |
| Priority: | Essential |
| Frequency of Use: | Frequent |
| Flow of Events: | 1. User click on display current location option 2. System will receive user’s GPS information 3. System display user’s current location |
| Alternative Flows: | NIL |
| Exceptions: | EX1: User does not have GPS enabled   1. App displays message “User location is not available” |
| Includes: | NIL |
| Special Requirements: | 1. App must respond to user’s actions within 3s 2. App must recover from errors within 1 minute 3. App must use a reliable API to ensure accurate data 4. App must have a error logging and feedback feature 5. App must support up to 10000 users concurrently |
| Assumptions: | 1. GPS data by user’s device is accurate |
| Notes and Issues: | NIL |

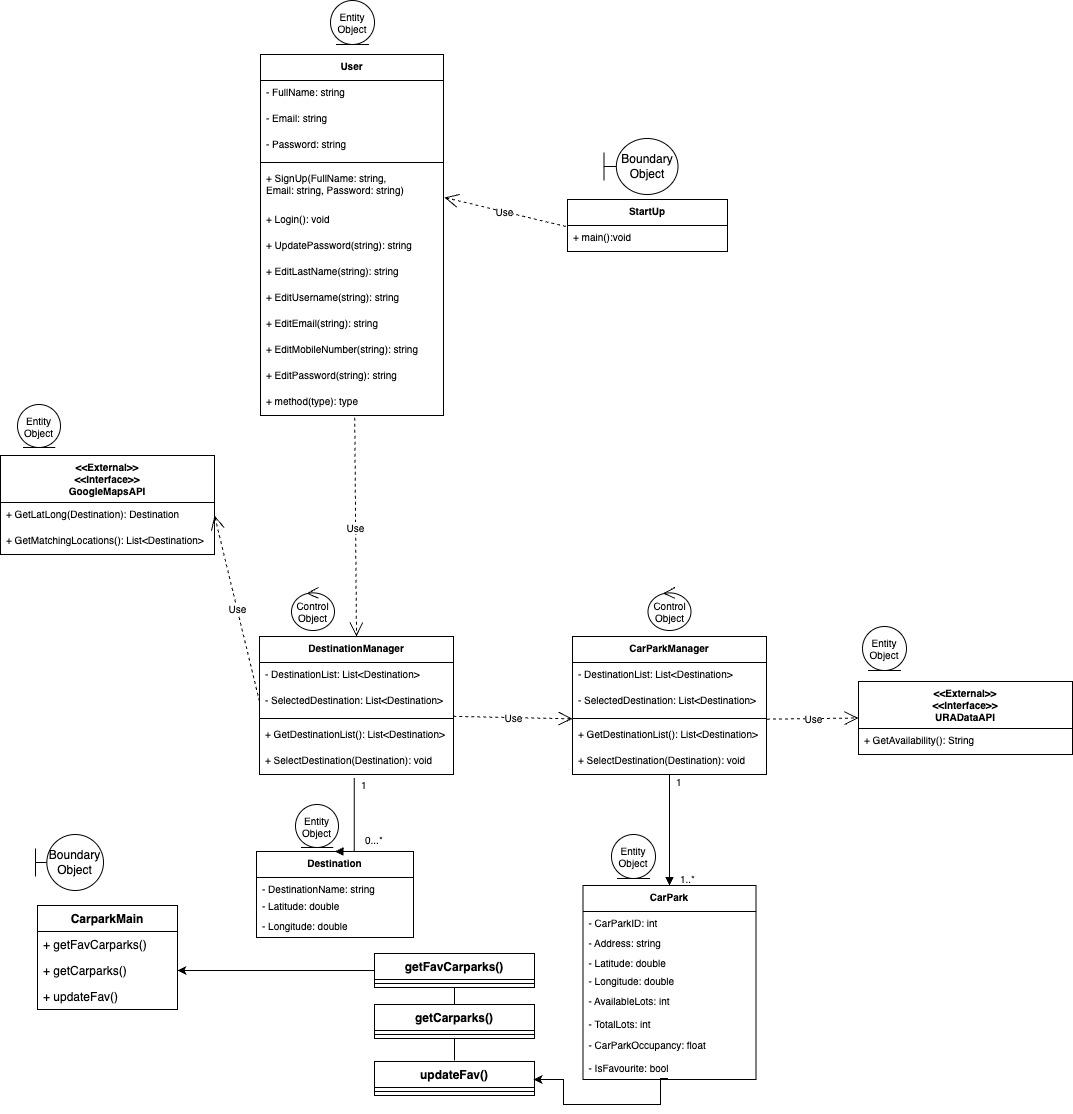
|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | CPK.004 | | |
| Use Case Name: | Search for location and display nearby car parks and their details | | |
| Created By: |  | Last Updated By: | Kai Yang |
| Date Created: | 09/02/2023 | Date Last Updated: | 15/04/2023 |

|  |  |
| --- | --- |
| Actor: | App User (Initiating actor) |
| Description: | User searches for a location and the system displays all car parks within 2 km of the selected location. |
| Preconditions: | 1. User has an account 2. User is logged in to their account |
| Postconditions: | 1. App displays all car parks within range of the location |
| Priority: | Essential |
| Frequency of Use: | High |
| Flow of Events: | 1. User inputs location into search bar 2. App displays car parks within 2 km from location along with their hourly rates, parking system type and number of slots available |
| Alternative Flows: | NIL |
| Exceptions: | EX1: If there are no car parks within distance range   1. App displays the message “No car parks within range of specified location!” 2. System prompts user to search for another location |
| Includes: | NIL |
| Special Requirements: | 1. App must use a reliable API to ensure accurate data 2. App must support up to 10000 users concurrently 3. App must render car parks within 30 seconds |
| Assumptions: | 1. Car park information fetched from API is accurate |
| Notes and Issues: | 1. Accuracy and availability of this use case depends on the accuracy and availability of the API. 2. The initial range to determine which car parks show up on the map is set to distance <= 2km |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | CPK.005 | | |
| Use Case Name: | Display available routes | | |
| Created By: |  | Last Updated By: | Boh Yang |
| Date Created: | 09/02/2023 | Date Last Updated: | 15/04/2023 |

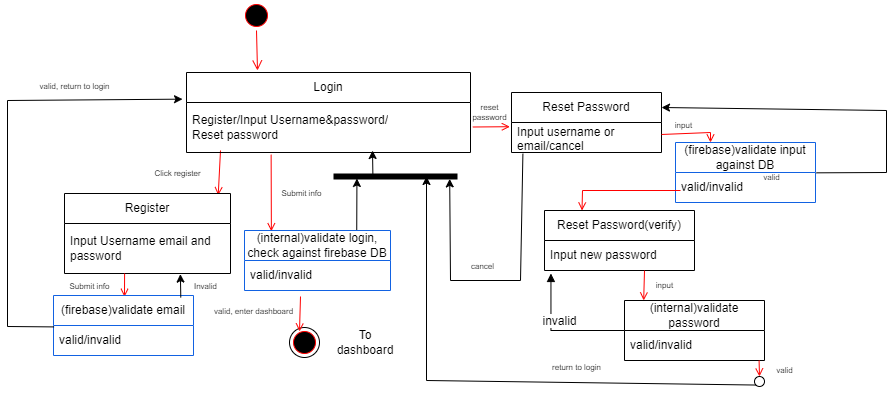
|  |  |
| --- | --- |
| Actor: | App User (Initiating actor) |
| Description: | App displays route to car parks |
| Preconditions: | 1. User has an account 2. User is logged into their account 3. User has searched for a specific location 4. User has selected a car park 5. User has google maps installed on their device |
| Postconditions: | App displays step by step navigations to selected car park |
| Priority: | Essential |
| Frequency of Use: | Moderate |
| Flow of Events: | 1. User selects a car parks of their choice 2. User clicks on button to switch to Google Maps 3. Car Park information is given to Google Maps. 4. Google maps displays route |
| Alternative Flows: | NIL |
| Exceptions: | NIL |
| Includes: | CPK.005 |
| Special Requirements: | 1. App must recover from errors within 1 minute 2. App must use a reliable API to ensure accurate data 3. App must support up to 10000 users concurrently |
| Assumptions: | 1. GPS data is accurate 2. Car Park information is accurate 3. Google Map is working as intended |
| Notes and Issues: | 1. Even if the APIs used are reliable, they can still have down time and this will affect the app usage 2. The initial range to determine which car parks show up on the map is still to be determined 3. The slot availability of car parks might change as the user is navigating to the selected carpark |

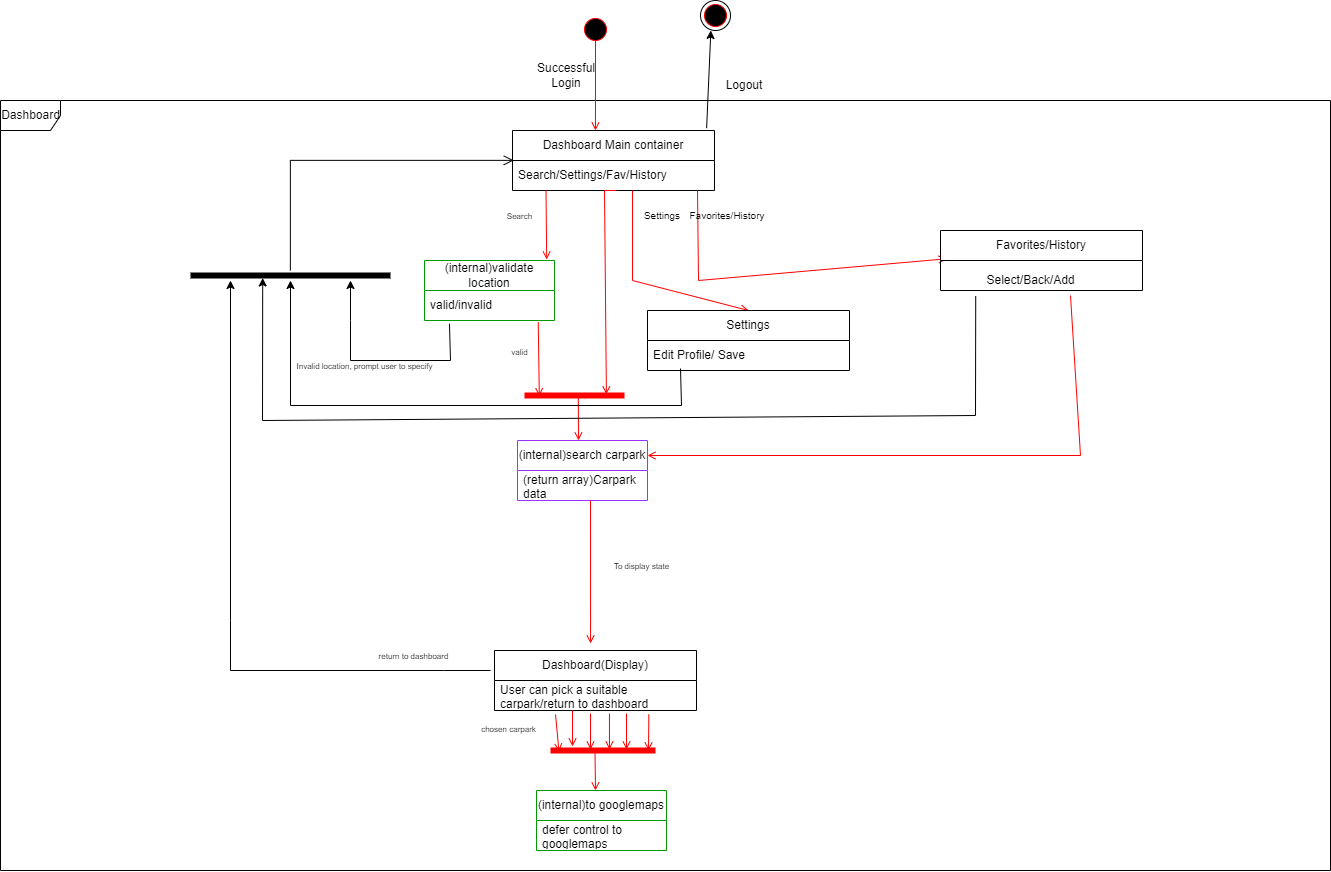
## Class Diagram

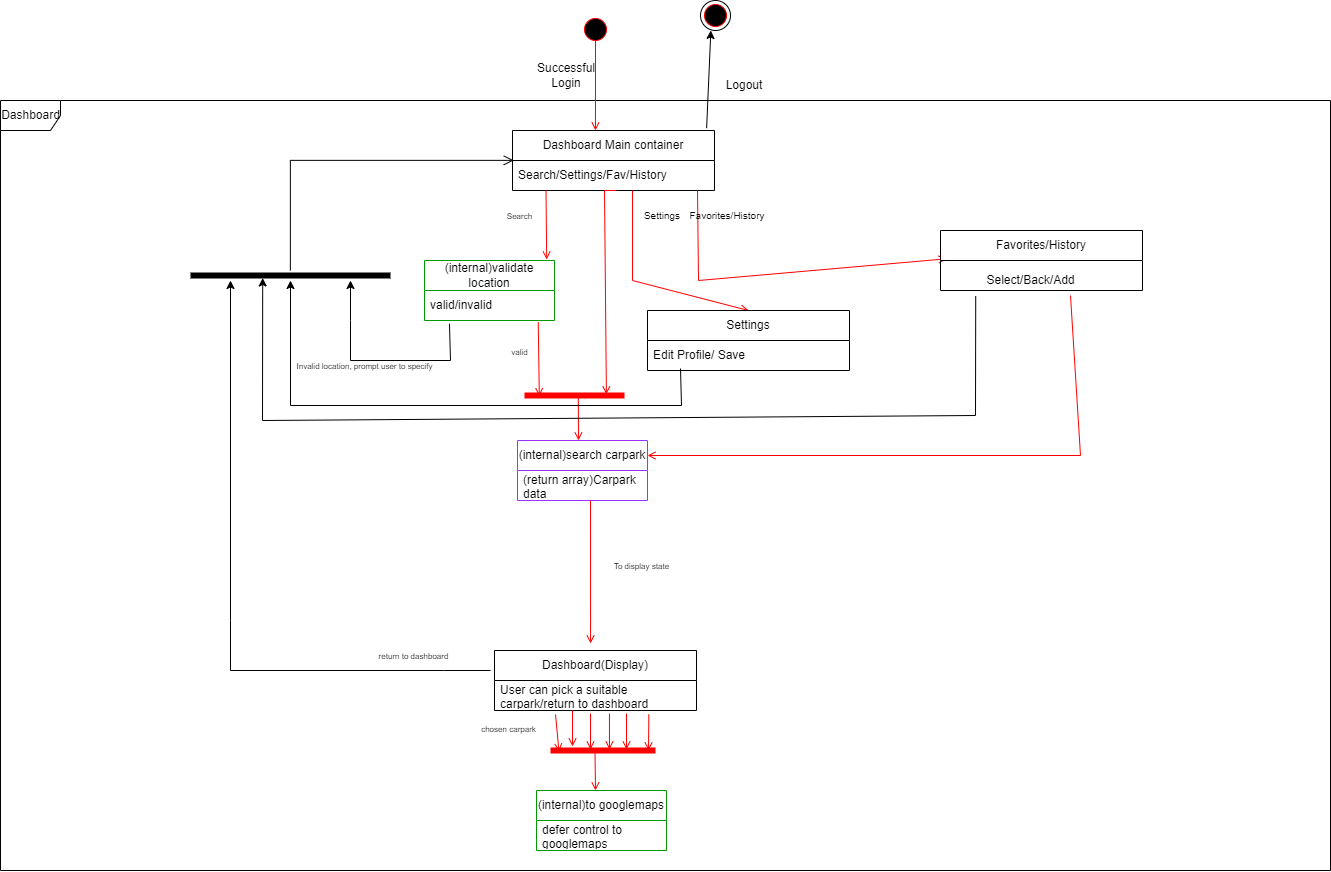


## State Machine Diagram / Dialog Map

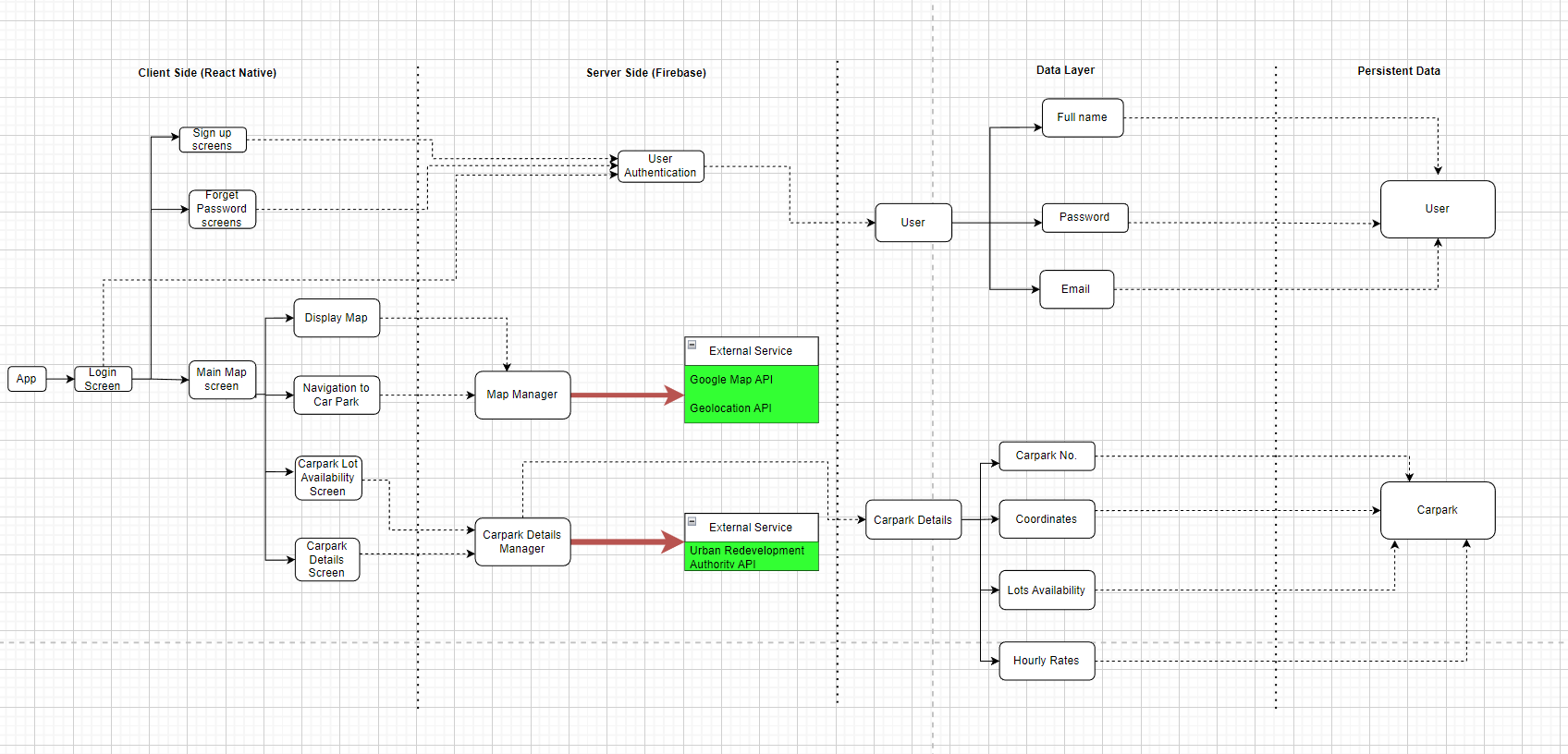
**Login States**



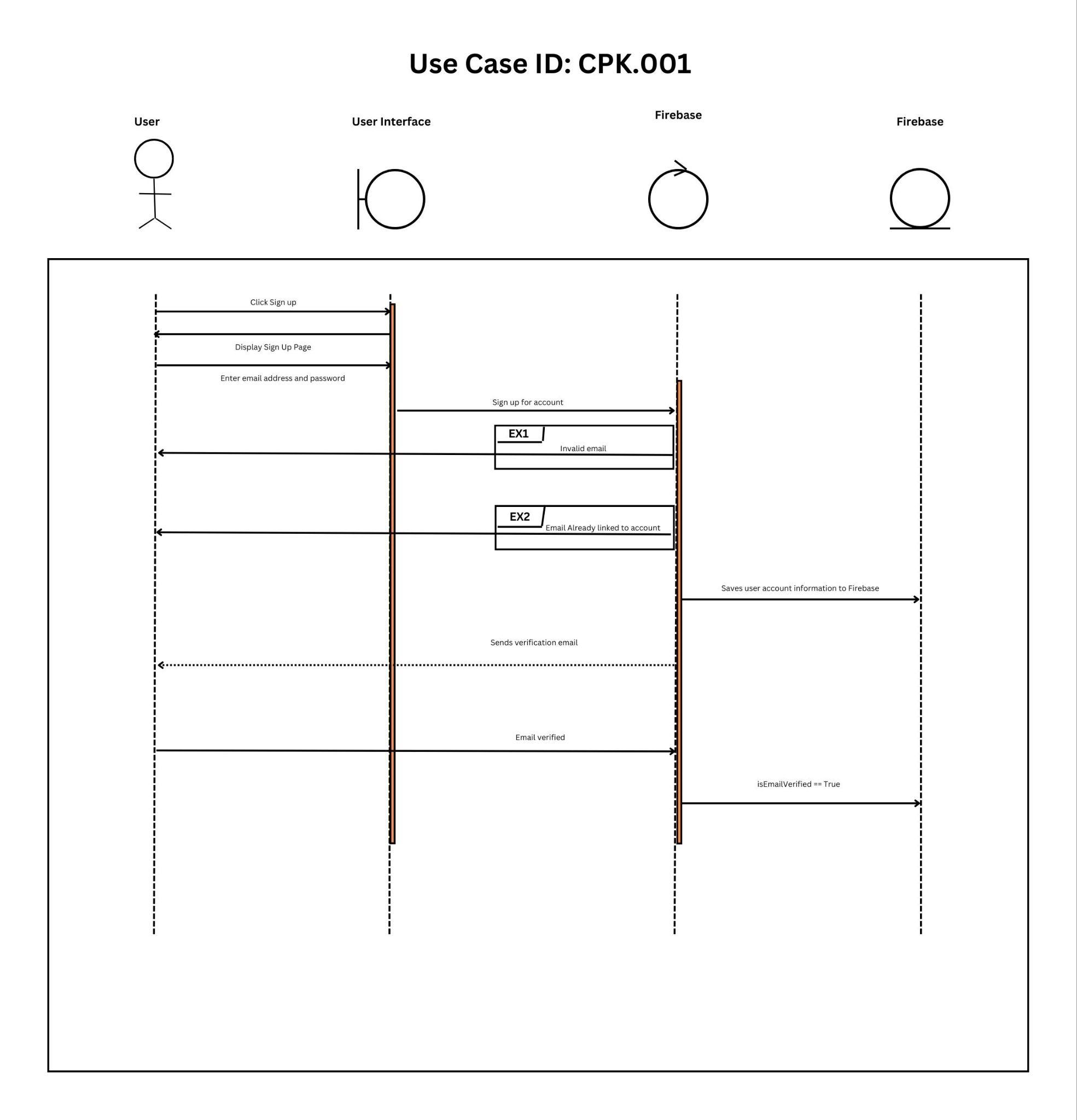
**Dashboard(Main Container) States**

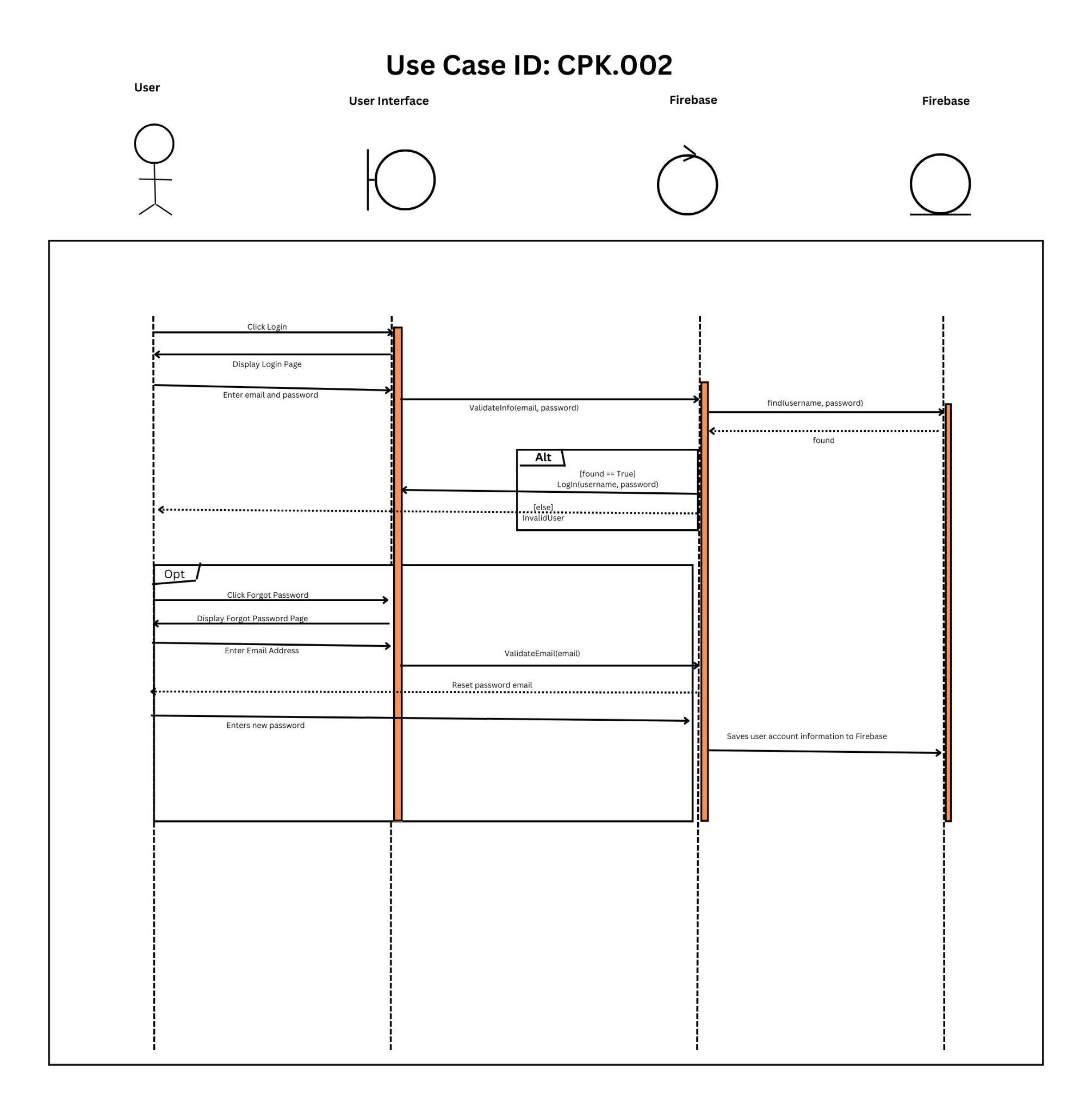


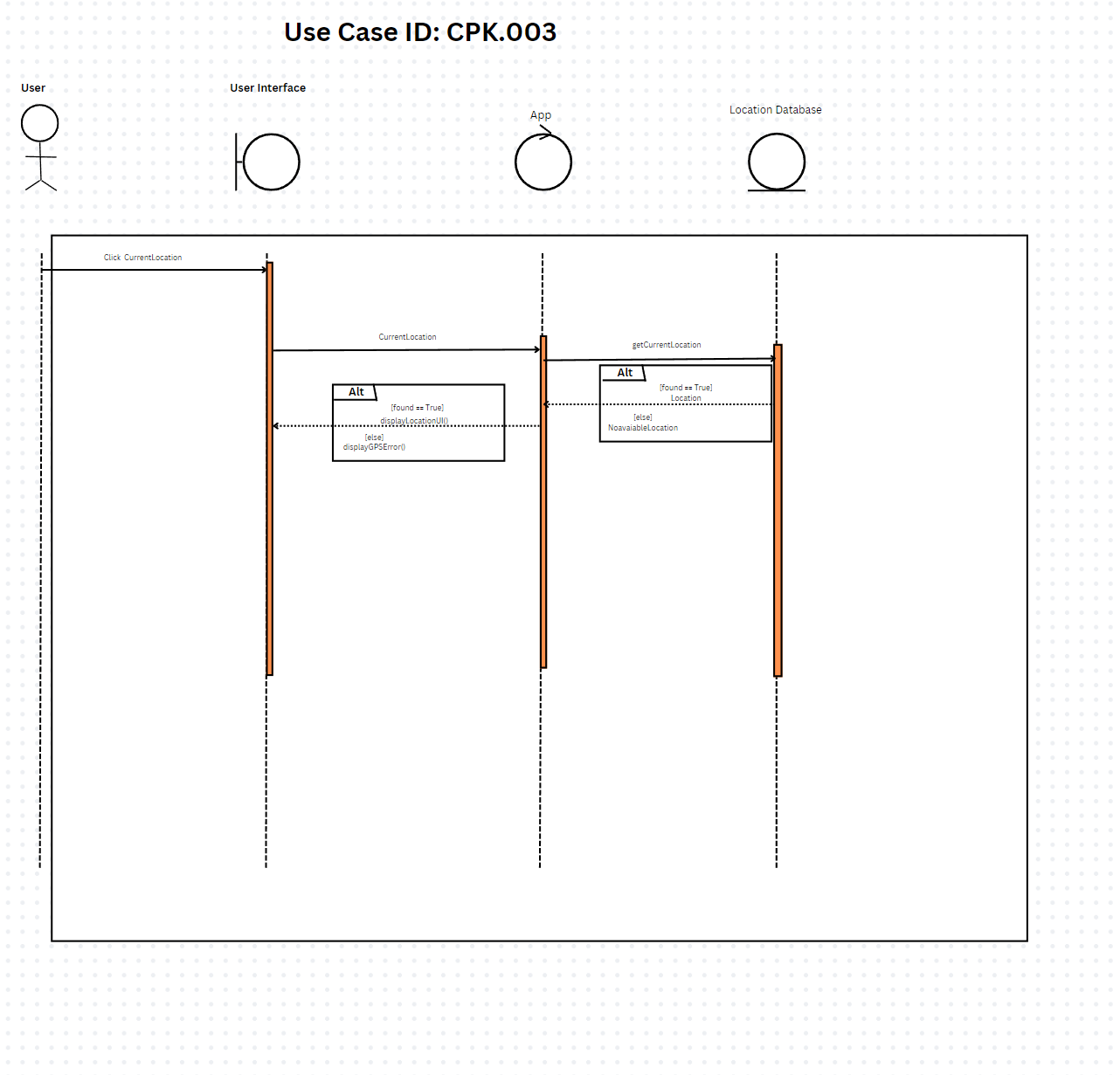
## System Architecture Diagram

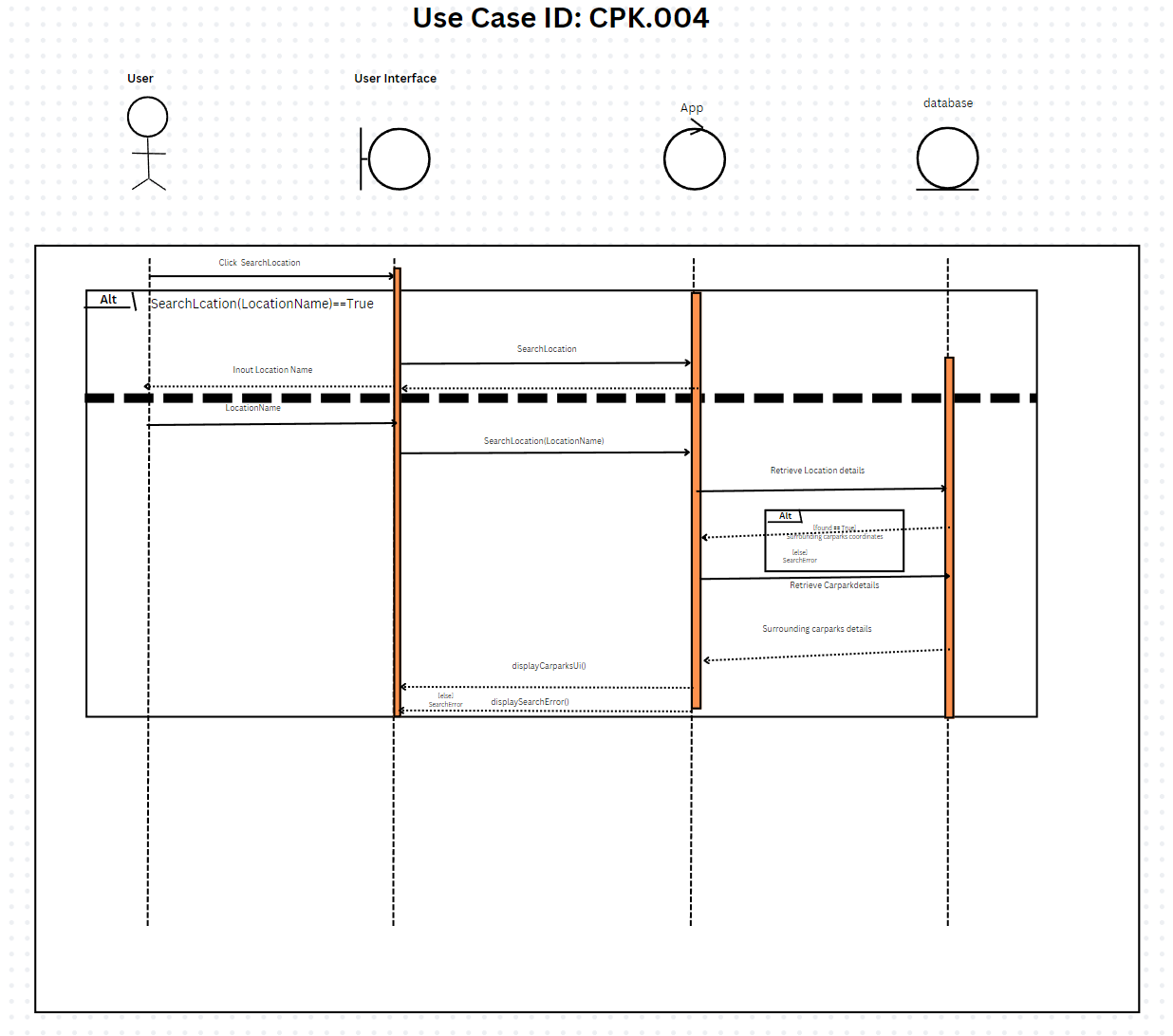


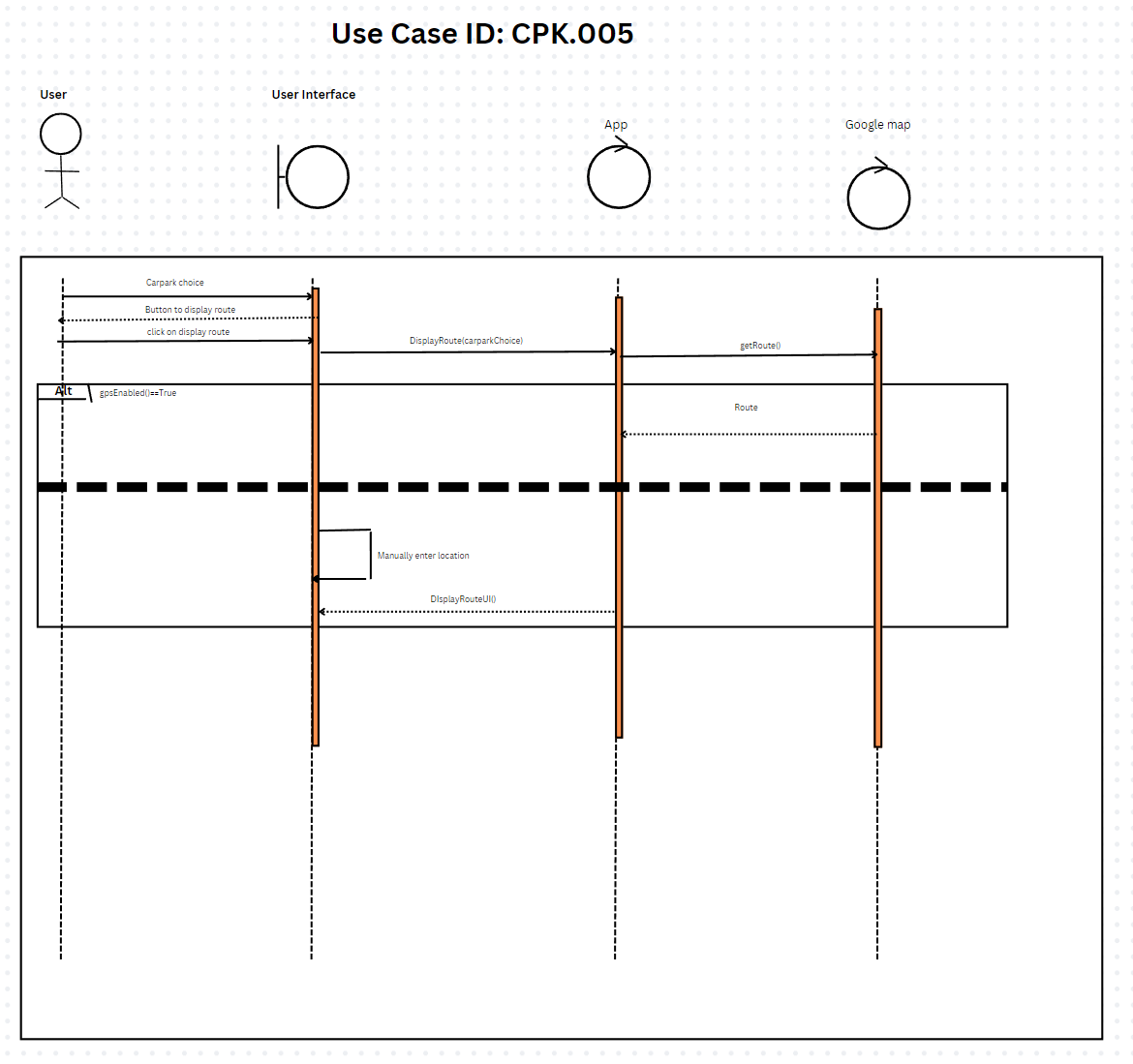
## Sequence Diagrams











## Unit Testing

### Black Box Testing

1. Register Account

|  |  |  |  |
| --- | --- | --- | --- |
| Test Id | Scenario | Expected Result | Actual Result |
| 1i | User keys in password that do not match both ‘Password’ and ‘Confirm Password’ field | The system displays error message that says "Passwords dont match" | The system displays error message that says "Passwords dont match" |
| 1ii | User registers with an email that is already registered | The system displays error message that says "Email already in use" | The system displays error message that says "Email already in use" |
| 1iii | User registers with a weak password (less than 8 characters, with at least 1 number and 1 letter) | The system displays error message that says "Password must have minimum of eight characters, at least one letter and one number" | The system displays error message that says "Password must have minimum of eight characters, at least one letter and one number" |
| 1iv | User registers with invalid email (no @) | The system displays error message that says "Invalid email" | The system displays error message that says "Invalid email" |
| 1v | User registers all fields with valid information | The system displays alert for email verification | The system displays alert for email verification |

Specific Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| Test Id | Scenario | Expected Result | Actual Result |
| 1i | Input fields:  1. Full Name: Andy Tan Xiao Ming  2. Email: cars123@gmailcom  4. Password: 4ndyLovesCars  5. Confirm Password: 4ndyLovesCar  Click on ‘Sign Up’ button | The system displays error message that says "Passwords dont match" | The system displays error message that says "Passwords dont match" |
| 1ii | Input fields:  1. Full Name: Andy Tan Xiao Ming  2. Email: user1@gmail.com  4. Password: 4ndyLovesCars  5. Confirm Password: 4ndyLovesCars  Click on ‘Sign Up’ button | The system displays error message that says "Email already in use" | The system displays error message that says "Email already in use" |
| 1iii | Input fields:  1. Full Name: Andy Tan Xiao Ming  2. Email: cars123@gmail.com  4. Password: AndyLovesCars  5. Confirm Password: AndyLovesCars  Click on ‘Sign Up’ button | The system displays error message that says "Password must have minimum of eight characters, at least one letter and one number" | The system displays error message that says "Password must have minimum of eight characters, at least one letter and one number" |
| 1iv | Input fields:  1. Full Name: Andy Tan Xiao Ming  2. Email: cars123.gmmail.com  4. Password: 4ndyLovesCars  5. Confirm Password: 4ndyLovesCars  Click on ‘Sign Up’ button | The system displays error message that says "Invalid email" | The system displays error message that says "Invalid email" |
| 1v | Input fields:  1. Full Name: Andy Tan Xiao Ming  2. Email: cars123@gmailcom  4. Password: 4ndyLovesCars  5. Confirm Password: 4ndyLovesCars  Click on ‘Sign Up’ button | The system displays alert for email verification | The system displays alert for email verification |

1. Login

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Id** | **Scenario** | **Expected Result** | **Actual Result** |
| 2i | User performs login without verifying email | The system displays message “Please verify your email before signing in”. | The system displays message “Please verify your email before signing in”. |
| 2ii | User performs login after verifying email | Application home page is loaded and map is displayed | Application home page is loaded and map is displayed |
| 2iii | User performs login after verifying email on a device with different OS. | Application home page is loaded and map is displayed | Application home page is loaded and map is displayed |

Specific Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Id** | **Scenario** | **Expected Result** | **Actual Result** |
| 2i | On IOS device:  User checks mailbox for verification email  User ignores verification email  Input Fields:  1. Email: [cars123@gmail.com](mailto:cars123@gmail.com)  2. Password:  4ndyLovesCars  Click on ‘Login’ button | The system displays message “Please verify your email before signing in”. | The system displays message “Please verify your email before signing in”. |
| 2ii | On IOS device:  User checks mailbox for verification email  User clicks on link in verification email  Input Fields:  1. Email: [cars123@gmail.com](mailto:cars123@gmail.com)  2. Password:  4ndyLovesCars  Click on ‘Login’ button | Application home page is loaded and map is displayed | Application home page is loaded and map is displayed |
| 2iii | On Android Device:  Input Fields:  1. Email: [cars123@gmail.com](mailto:cars123@gmail.com)  2. Password:  4ndyLovesCars  Click on ‘Login’ button | Application home page is loaded and map is displayed | Application home page is loaded and map is displayed |

1. Search Screen

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Id** | **Scenario** | **Expected Result** | **Actual Result** |
| 3i | User queries valid destination in Singapore | Pin and respective data of car parks within 2km radius will be displayed | Pin and respective data of car parks within 2km radius will be displayed |
| 3ii | User enters invalid destination | The system displays error message and no carparks will be displayed | The system displays error message and no carparks will be displayed |
| 3iii | Navigate to the search screen without an internet connection | The application will prompt the user to reconnect and load the screen again | The application will prompt the user to reconnect and load the screen again |

Specific Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Id** | **Scenario** | **Expected Result** | **Actual Result** |
| 3i | Input Field:  1. Enter Location:  Jurong Point  Click on ‘Search’ button | Pin and respective data of car parks within 2km radius will be displayed | Pin and respective data of car parks within 2km radius will be displayed |
| 3ii | Input Field:  1. Enter Location:  Taco Bell  Click on ‘Search’ button | The system displays error message and no carparks will be displayed | The system displays error message and no carparks will be displayed |
| 3iii | Turn off Wifi  Input Field:  1. Enter Location:  Jurong Point  Click on ‘Search’ button | The application will prompt the user to reconnect and load the screen again | The application will prompt the user to reconnect and load the screen again |

1. Navigation

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Id** | **Scenario** | **Expected Result** | **Actual Result** |
| 4i | User clicks on pin of destination carpark and clicks the navigation button | The application opens up an instance of Google Maps and displays route from user’s current location to destination carpark | The application opens up an instance of Google Maps and displays route from user’s current location to destination carpark |
| 4ii | User clicks on pin of destination carpark and clicks the navigation button with no internet connection | The application prompts user to reconnect and load the screen again | The application prompts user to reconnect and load the screen again |

## 

Specific Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Id** | **Scenario** | **Expected Result** | **Actual Result** |
| 4i | User clicks on pin of carpark named ‘SECOND CHIN BEE ROAD’  Click on the navigation button | The application opens up an instance of Google Maps and displays route from user’s current location to destination carpark | The application opens up an instance of Google Maps and displays route from user’s current location to destination carpark |
| 4ii | Turn off Wifi  Click on pin of carpark named ‘SECOND CHIN BEE ROAD’  Click on the navigation (right arrow) button | The application prompts user to reconnect and load the screen again | The application prompts user to reconnect and load the screen again |

1. Search History

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Id** | **Scenario** | **Expected Result** | **Actual Result** |
| 5i | User has not searched anything in the app before | The search history page returns “No recent searches” | The search history page returns “No recent searches” |
| 5ii | User query destination in the app | The search history page returns name of destination searched | The search history page returns name of destination searched |

## 

Specific Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Id** | **Scenario** | **Expected Result** | **Actual Result** |
| 5i | User has not searched anything in the app before | The search history page returns “No recent searches” | The search history page returns “No recent searches” |
| 5ii | Input Field:  Enter Location: Jurong Point  Click Search | The search history page returns name of destination searched | The search history page returns name of destination searched |

1. Settings

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Id** | **Scenario** | **Expected Result** | **Actual Result** |
| 6i | User checks whose profile the application is logged in to | The application displays ‘Full Name’ and ‘Email’ fields input by user during registration | The application displays ‘Full Name’ and ‘Email’ fields input by user during registration |
| 6ii | User logs out of the application | The application displays the Login page | The application displays the Login page |

Specific Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Id** | **Scenario** | **Expected Result** | **Actual Result** |
| 6i | Click ‘Settings’  Click ‘Profile’ | The application displays ‘Full Name’ and ‘Email’ fields input by user during registration | The application displays ‘Full Name’ and ‘Email’ fields input by user during registration |
| 6ii | Click ‘Settings’  Click ‘Logout’ | The application displays the Login page | The application displays the Login page |

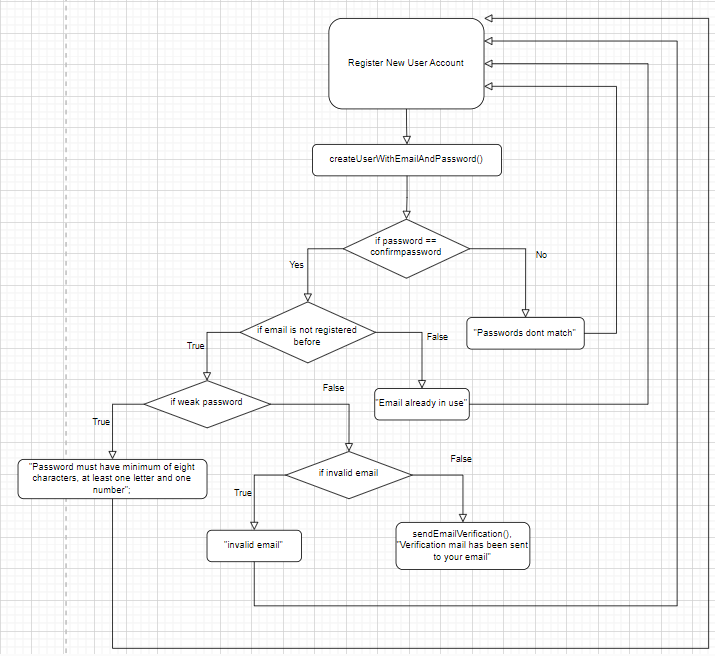
1. Map Overview and Sample Direction

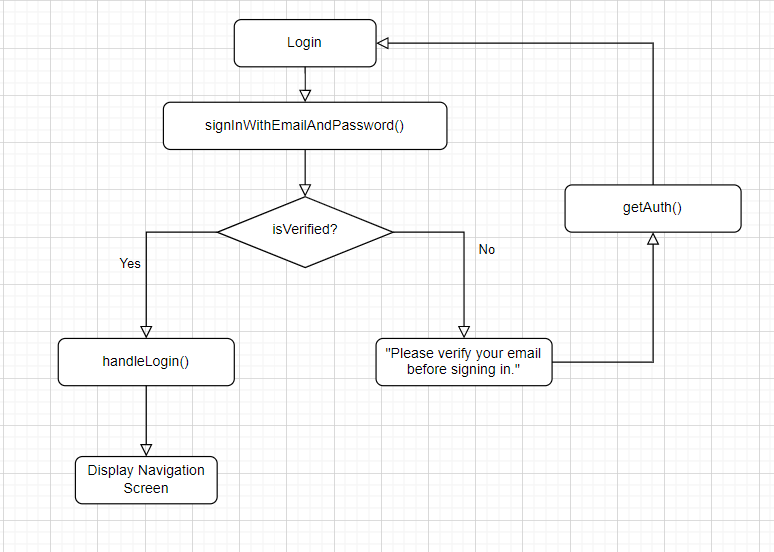
|  |  |  |  |
| --- | --- | --- | --- |
| **Test Id** | **Scenario** | **Expected Result** | **Actual Result** |
| 7i | User has not logged in and queries Top 5 carparks | The application displays map with pins of top 5 rated carparks in Singapore | The application displays map with pins at top 5 rated carparks in Singapore |
| 7ii | User has not logged in and queries how to get to the destination | The application displays an instance of Google Maps with directions to destination input in the search bar | The application displays an instance of Google Maps with directions to destination already input in the search bar |

1. Forgot Password

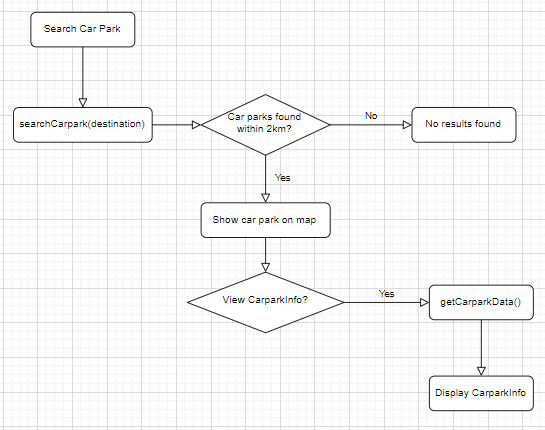
|  |  |  |  |
| --- | --- | --- | --- |
| **Test Id** | **Scenario** | **Expected Result** | **Actual Result** |
| 8i | User clicks ‘Forget Password’ button | The application prompts for user email and a message ‘Please check your junk email for a reset password link.’ . User receives an email containing password reset link in his/her registered email | The application prompts for user email and a message ‘Please check your junk email for a reset password link.’ . User receives an email containing password reset link in his/her registered email |

### White Box Testing

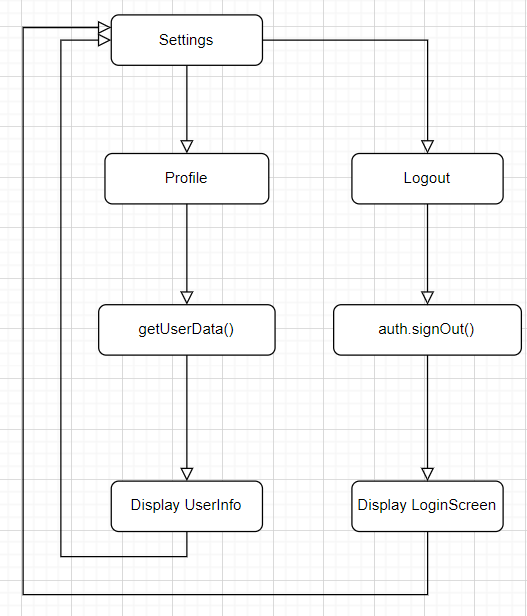
1. Register Account 
2. Login



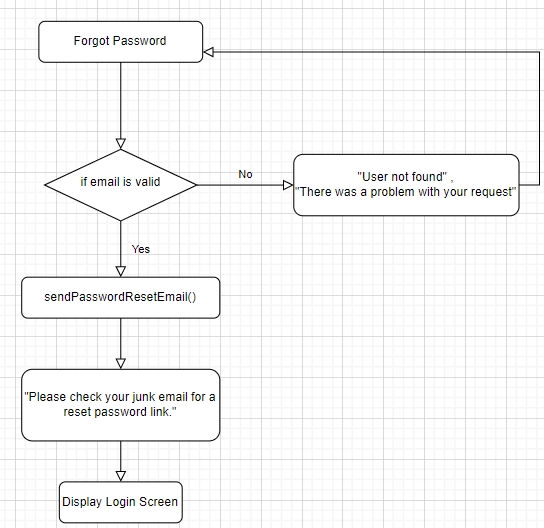
1. Search Screen



1. Settings



1. Reset Password



## App Demo

The demonstration of our App can be viewed at the following link   
<https://youtu.be/5hKf5ObB2pQ>