**NANYANG TECHNOLOGICAL UNIVERSITY**

**CZ2006 SOFTWARE ENGINEERING**

**Car Park App**

**SSP1 Team NTU SE**

**Team Members:**

Devansh Koppar (Leader) (U1823660K)

Ng Wee Hsuan (U1920658D)

Glenda Hong Zixuan (U1921784A)

Supraja Naganathan (U2021507E)

Chantharojwong Kasidis (U2020731L)

Lim Jiayi Joey (U2021903G)

Contents

[**1. Product Description**](#_r2jzj1qabdvt) **3**

[1.1. Purpose](#_f3or90mlb7vt) 3

[1.2. Scope](#_n9d5jnt5nons) 3

[1.3. Users and Stakeholders](#_gw5f6l81atsw) 3

[1.4. Assumptions](#_r90xl1ccdqkb) 3

[1.5. Constraints](#_b804ayvxq6ja) 4

[1.6. Initial UI Mockups](#_cpojcxsdd6l8) 4

[**2. Functional Requirements**](#_i6nu71nrmo8b) **10**

[2.1. Use Case Diagrams](#_de2twfqya0qt) 13

[2.2. Use Case Descriptions](#_17kjtvg14jlu) 14

[2.3. Class Diagram](#_expu6lg6x09a) 37

[2.4. Sequence Diagram](#_fcwy8huz9qlt) 38

[2.5. Dialog Map](#_ueozm02y9h87) 46

[**3. Non-Functional Requirements**](#_enfig6ntgxea) **47**

[**4. Interface Requirements**](#_yl2pwsj5wkkn) **49**

[**5. Architecture Design**](#_k3r9qxv5gaj1) **50**

[5.1. System Architecture Diagram](#_t0v7u780p1sp) 50

[5.2. Design Pattern](#_jt251ifq1a1z) 51

[**6. Data Dictionary**](#_1oo4qzf8jn7o) **52**

[**7. Testing**](#_x6y6rs2unguf) **54**

[7. 1. Black Box Testing](#_9xqvr4wvrj0h) 54

[7.2. White Box Testing](#_f4zfwmn9yjnb) 55

# 1. Product Description

## 1.1. Purpose

Our application’s main purpose is to allow car drivers to search for available lots in HDB car parks across Singapore based on either a user inputted location or their current GPS location.

## 1.2. Scope

The application will be available in the Google Play Store for all Android phones.

## 1.3. Users and Stakeholders

Our application targets drivers in Singapore with an Android phone who live in HDBs or will park in a HDB carpark. The stakeholders of this project consists of car drivers who use Android, team SE, and the government.

The drivers will use the application to search for nearby car parks and locate empty lots. Team SE aims to ease the driving experience of Singapore drivers by providing a smooth and seamless platform to check available lots before heading to the car park physically, thus saving time and energy. The government will assist Team SE by providing the API located in weblink <https://data.gov.sg/dataset/carpark-availability>.

## 1.4. Assumptions

In order for the application to work, we made two assumptions:

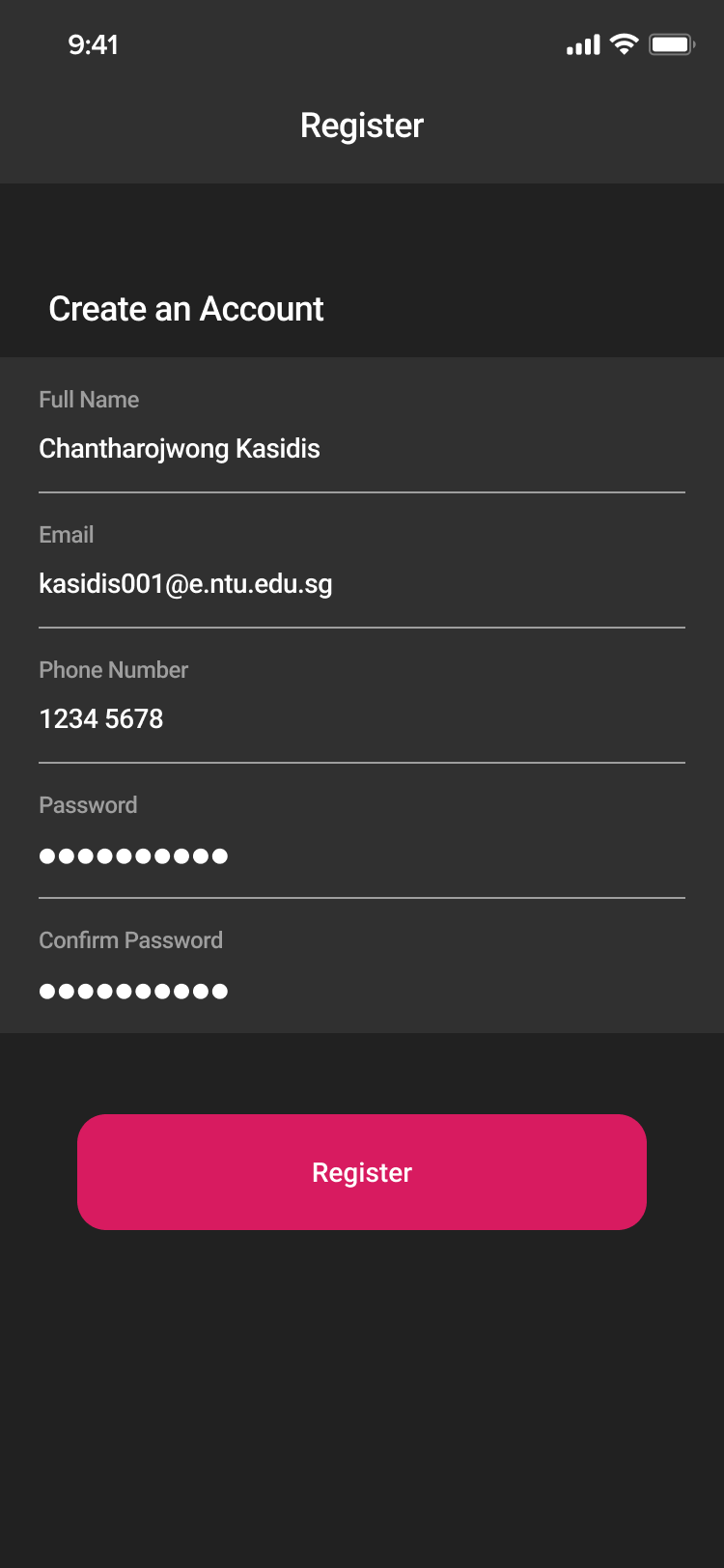
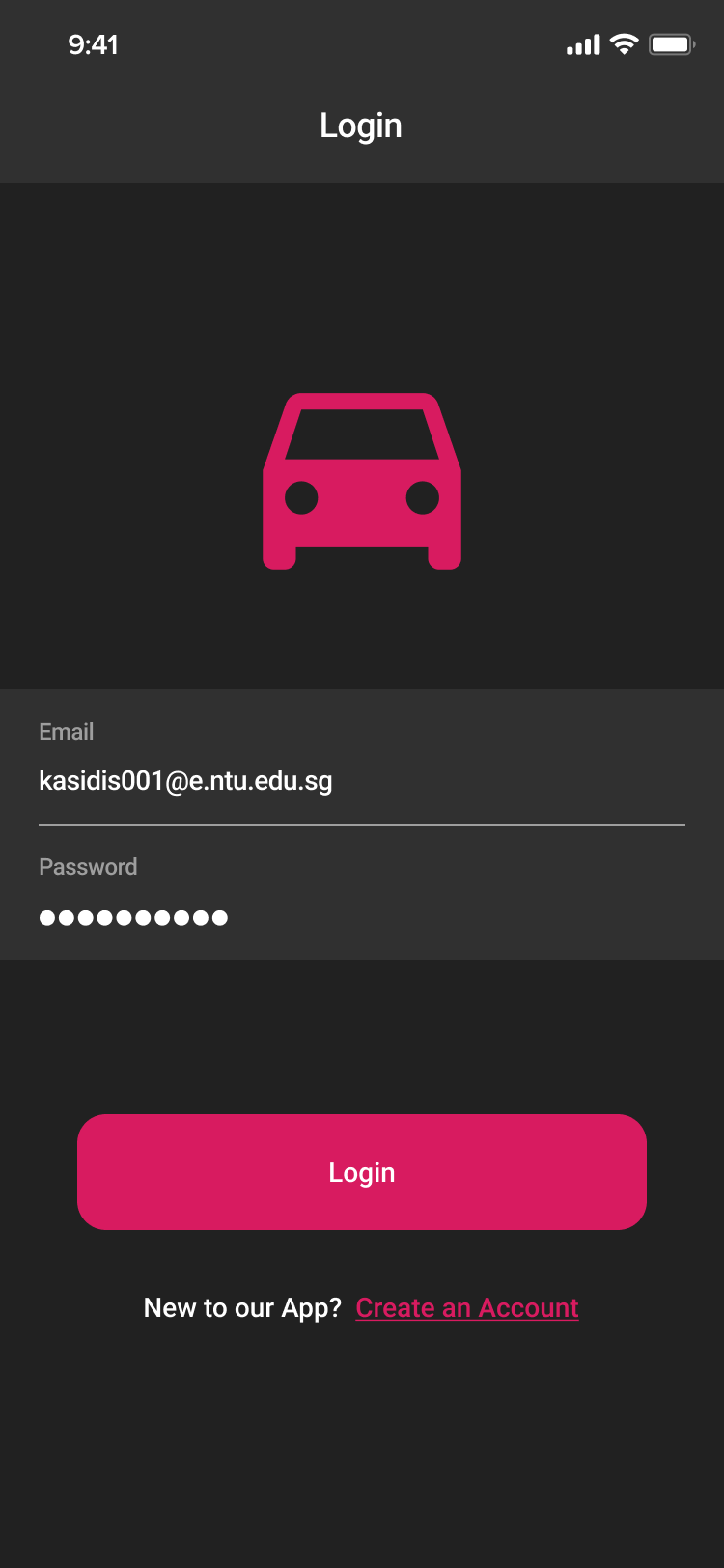
1. Users would have a GPS-enabled Android device capable of running the application.
2. Users would have access to the Internet either through Wi-Fi or cellular data.

## 1.5. Constraints

Our application currently only supports Android OS running version 2.12.0 and above.

The application is currently available only in English.

## 1.6. Initial UI Mockups

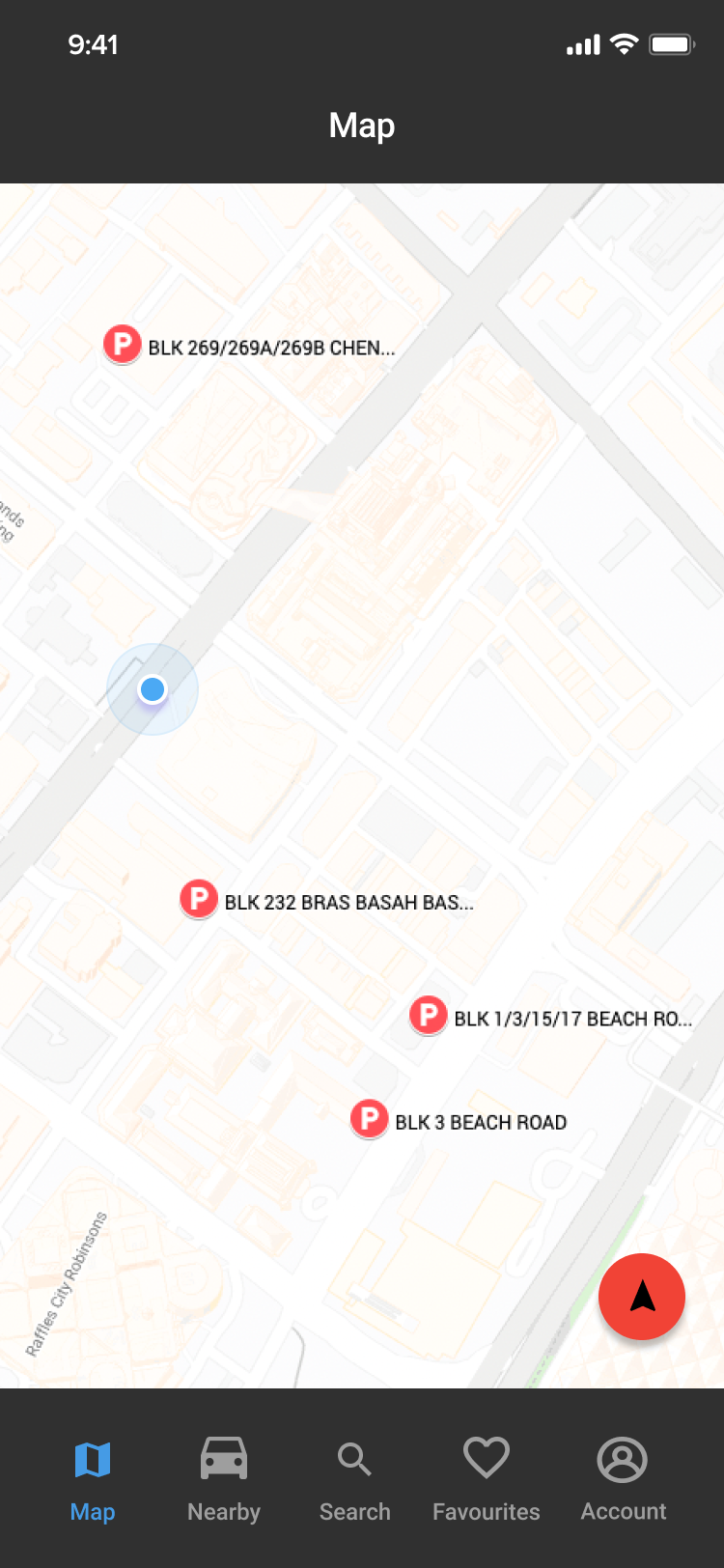
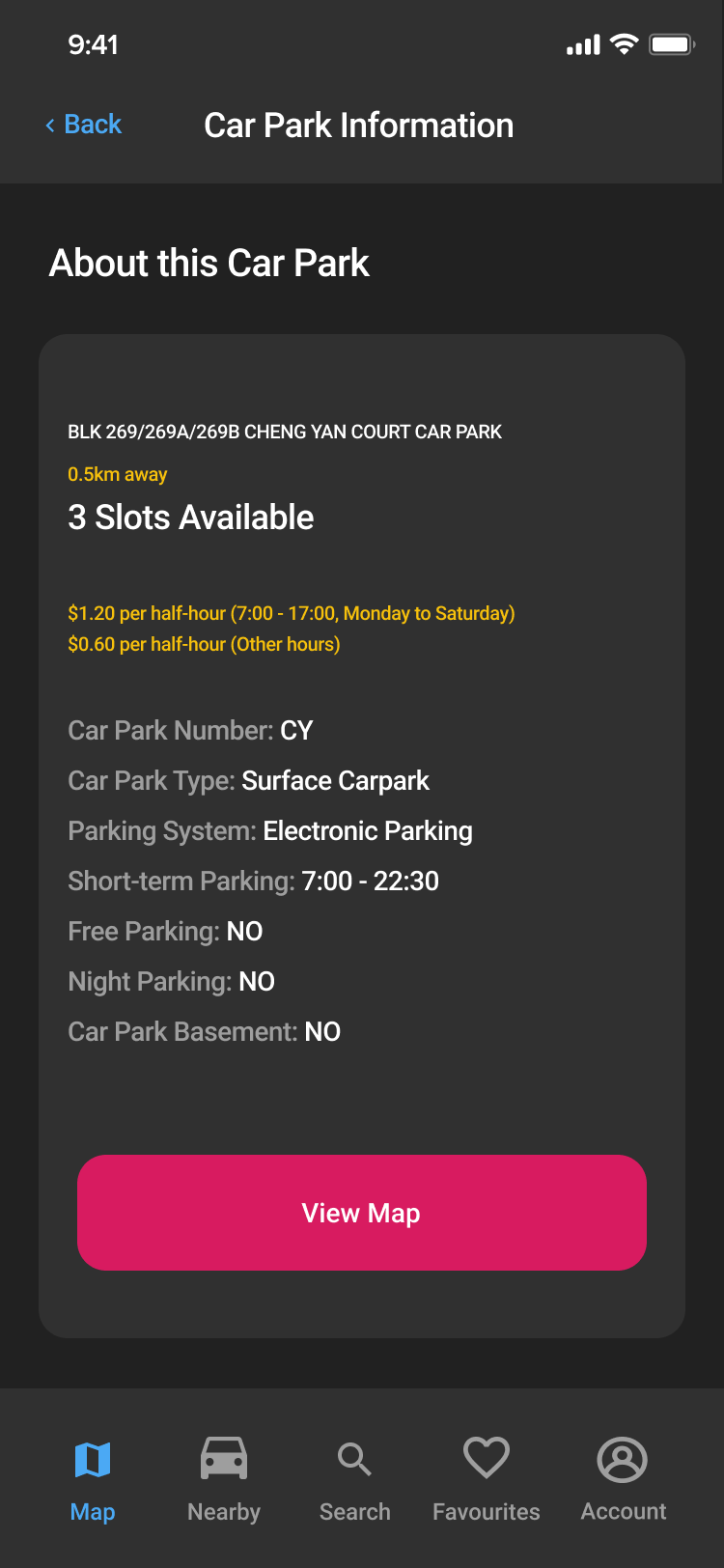


# 

# 

*Figure 1: Login and Registration Screens*

Figure 1 illustrates the user interface for login and registration. The login screen displays the Email and Password field, a registration link, and a login button. New users can create an account by filling in their information and password at the registration screen. The system will display error messages if the fields are invalid or authentication fails.



# 

# 

# 

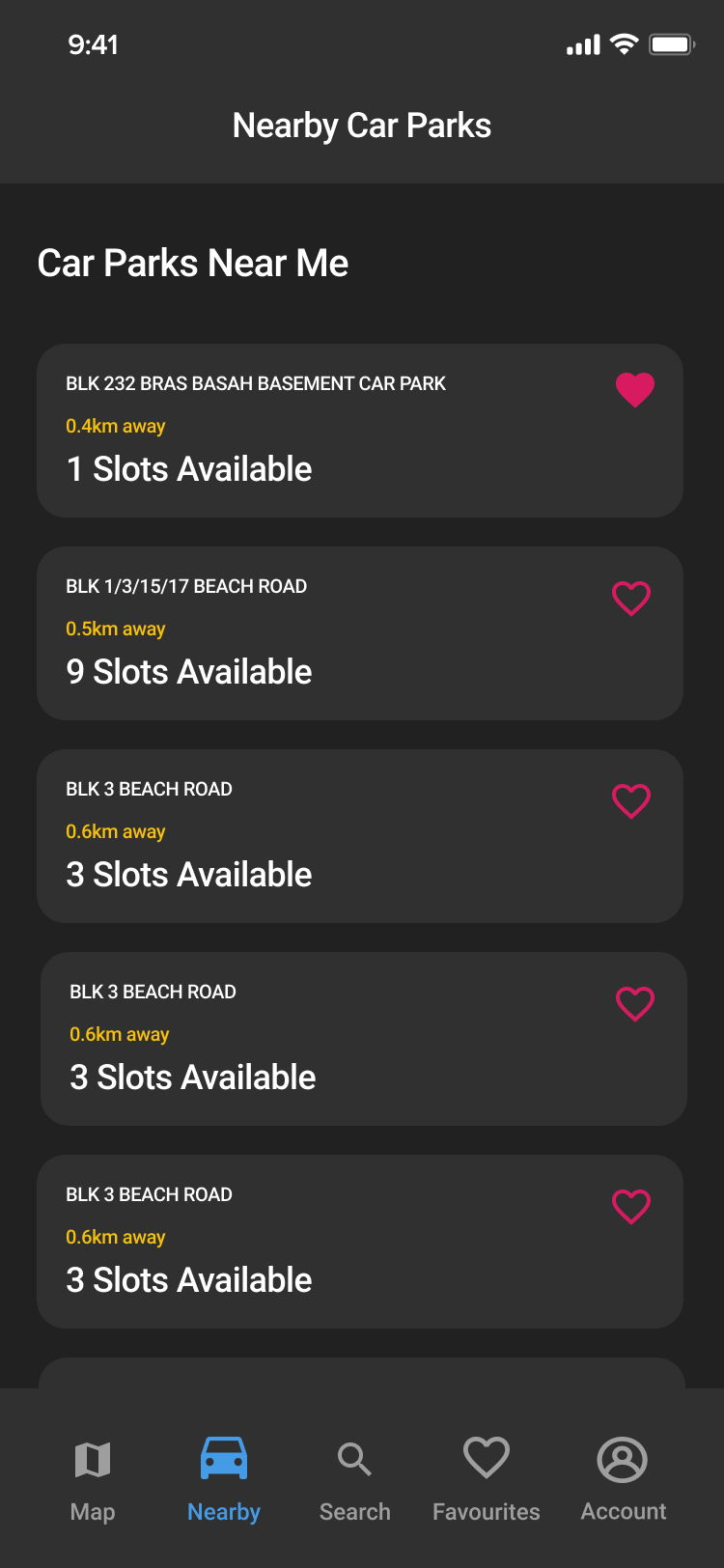
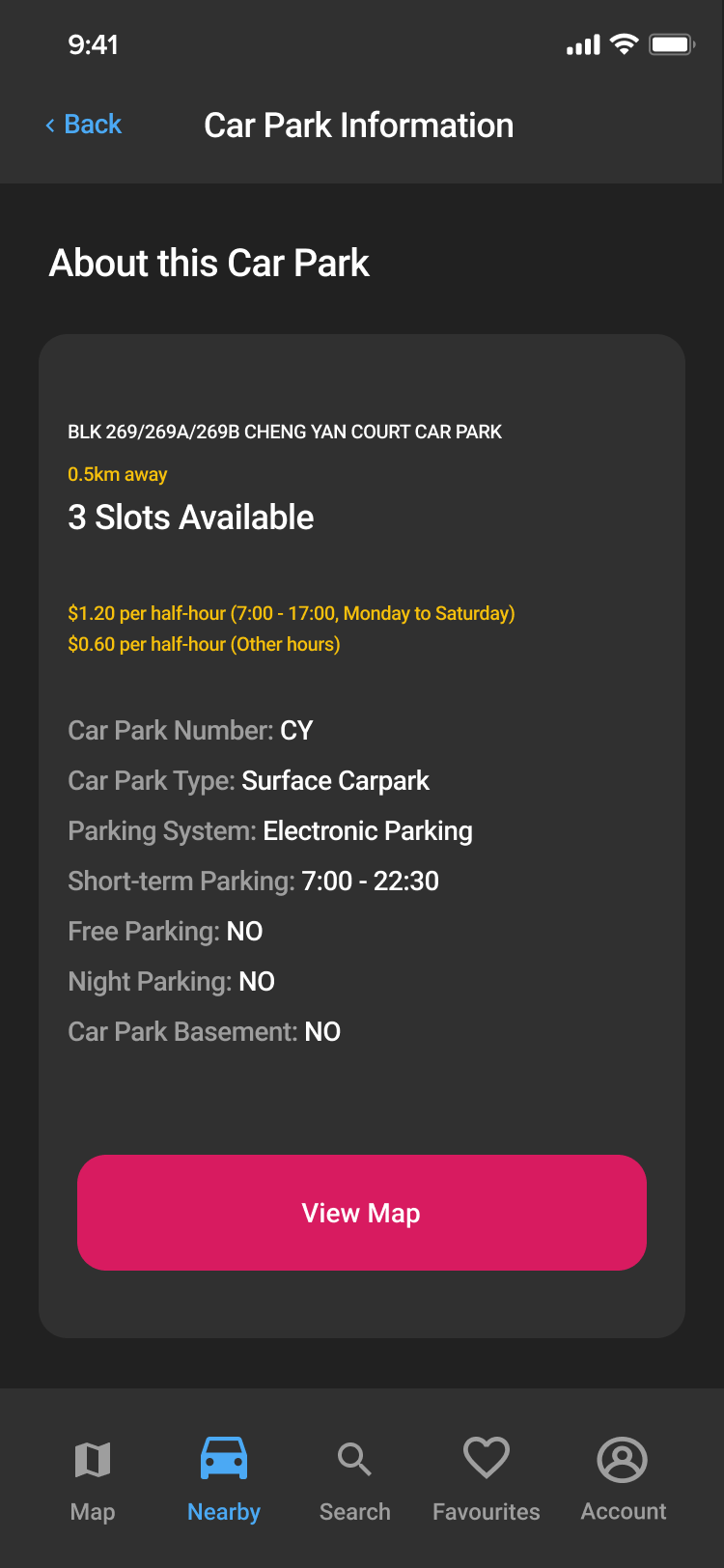
# 

# 

# 

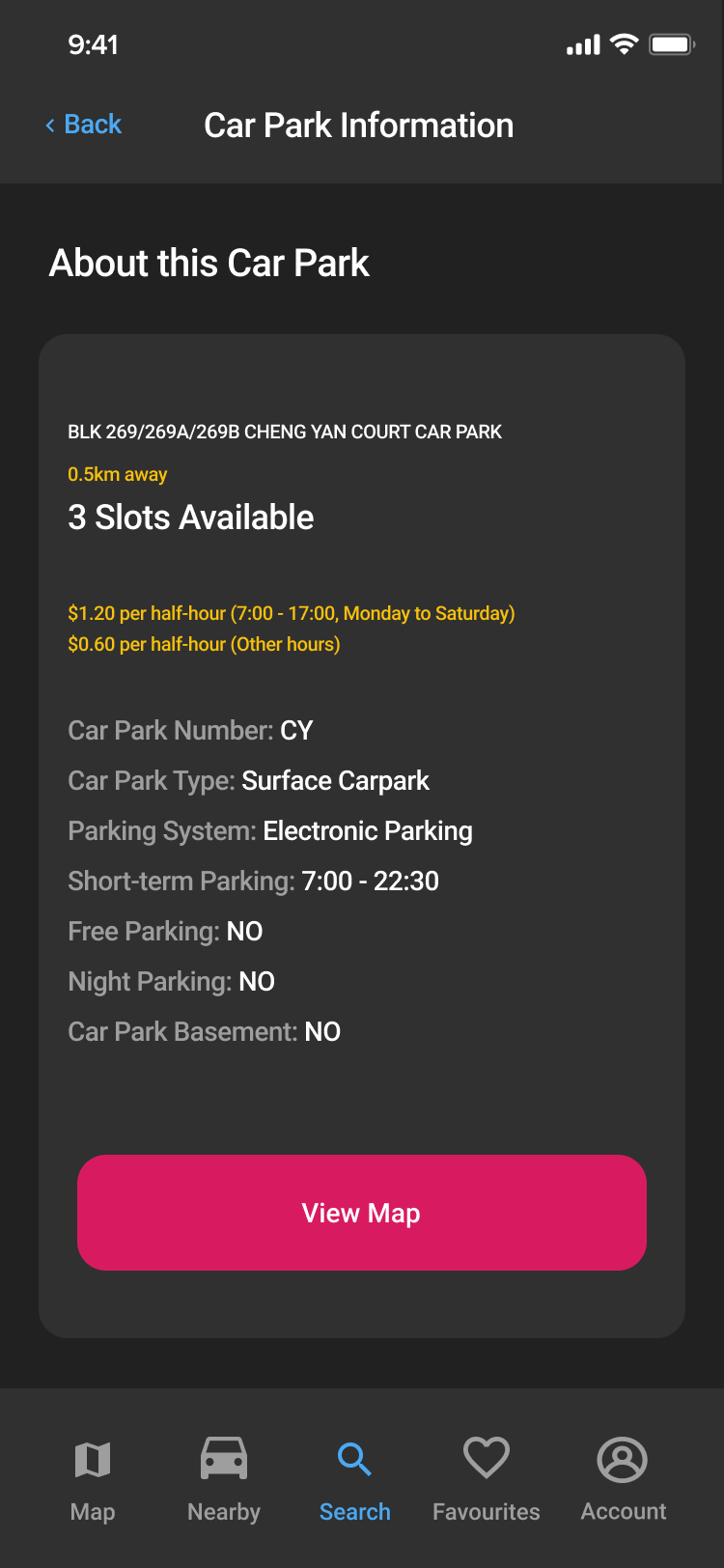
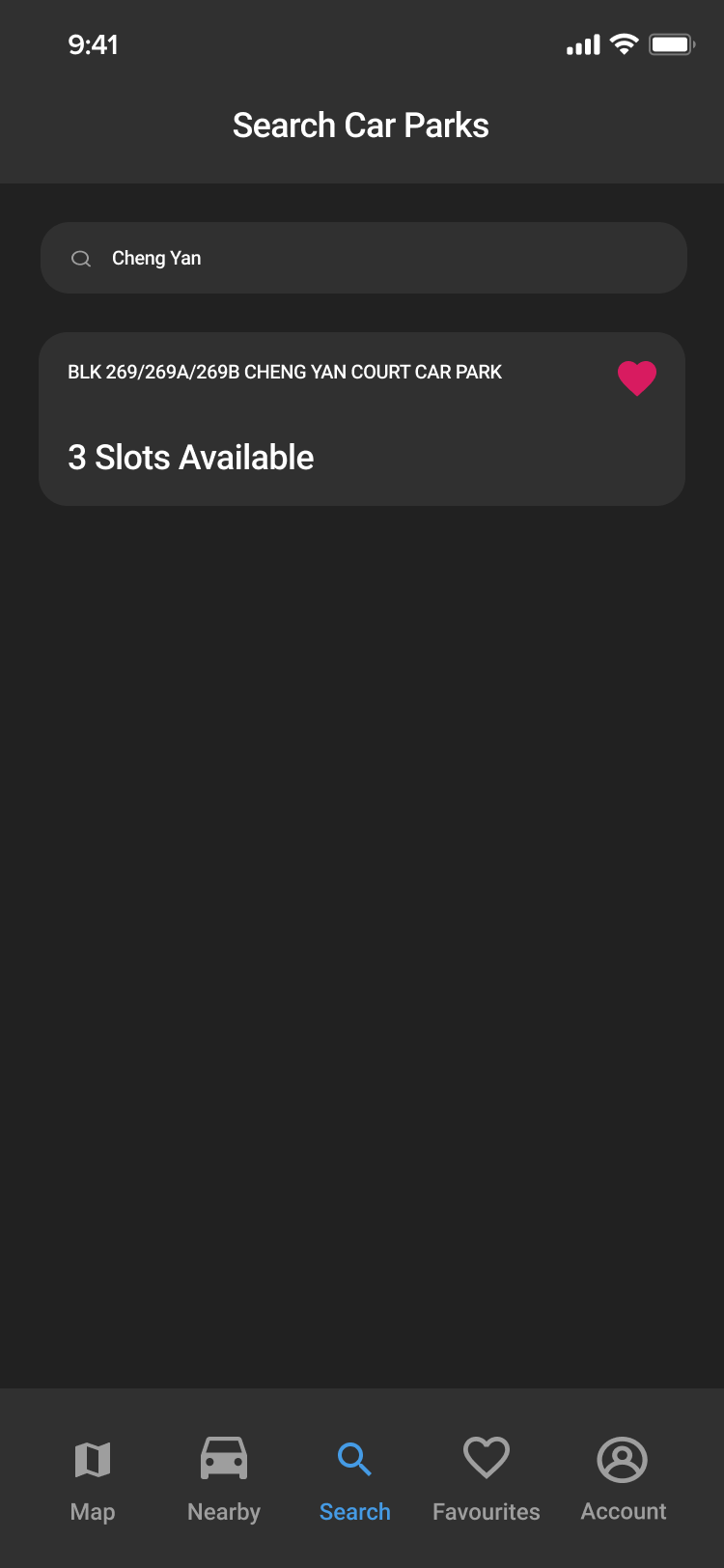
*Figure 2: Map Screen and Car Park Information Popup Screen*

Figure 2 illustrates the map screen and a car park information popup from the map screen. Whenever a user taps a car park on the map, a car park information popup will appear. Users can also open Google Maps from the map screen to navigate to the car park. The car park information screen displays additional details about the car park and users can click back to go back to the map screen.



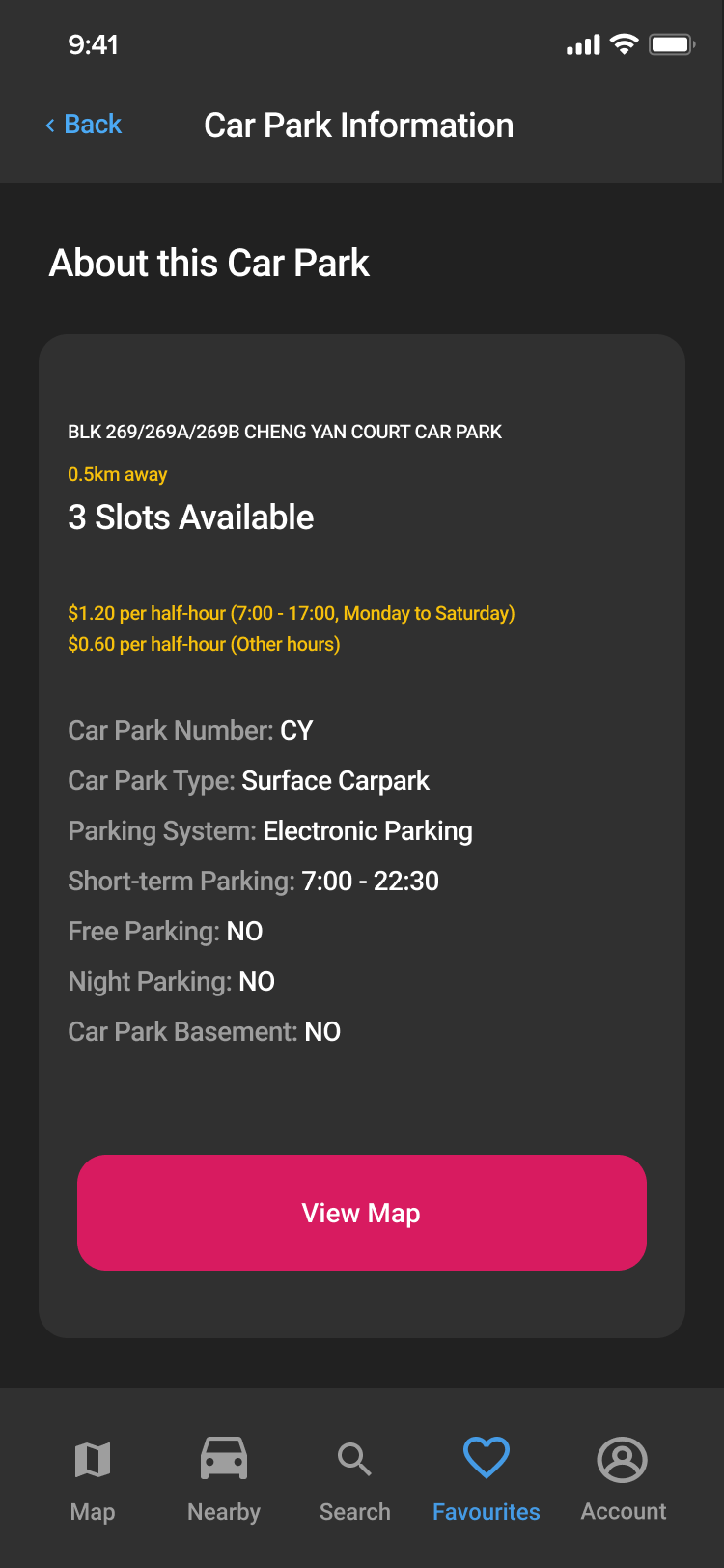
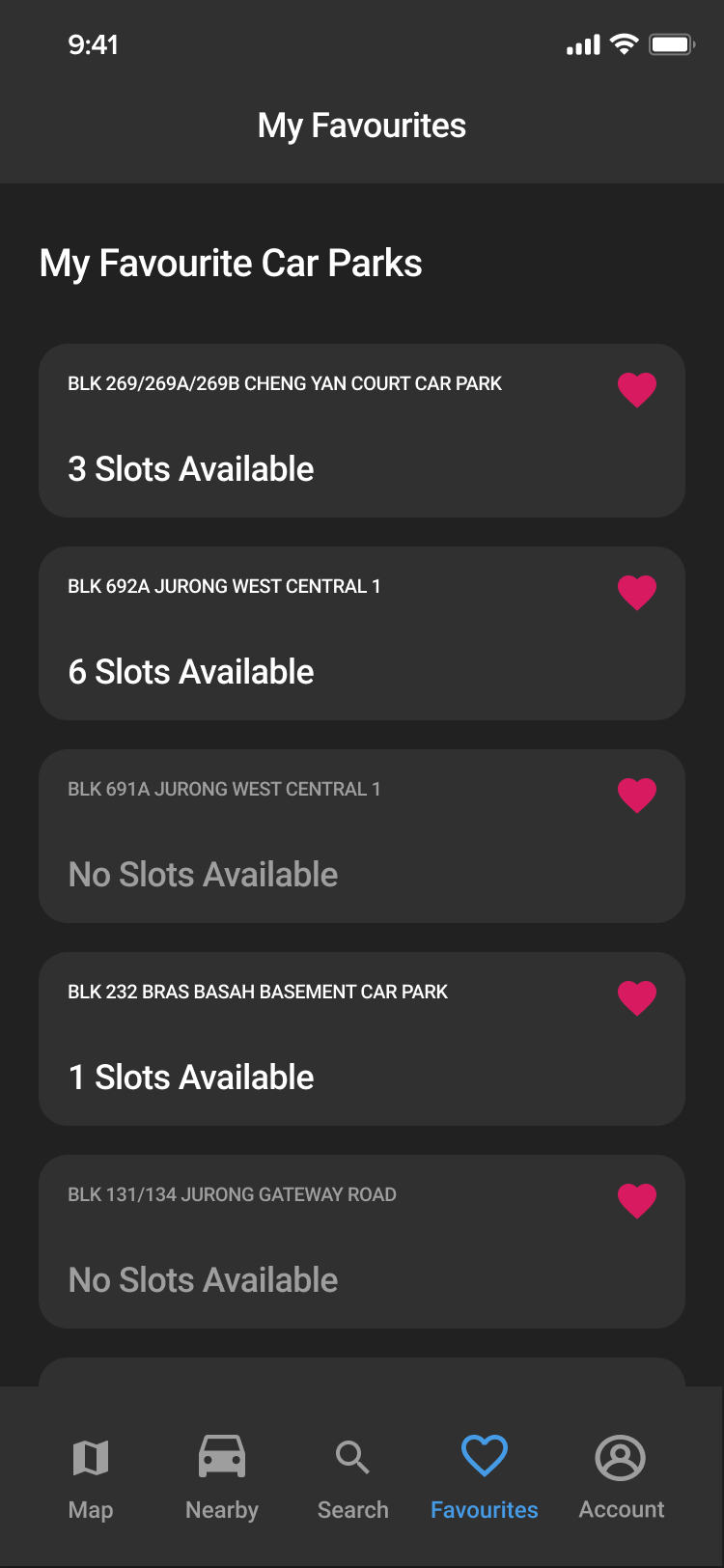
*Figure 3: Nearby Car Parks Screen and Car Park Information Popup Screen*

Figure 3 illustrates the nearby car parks screen and a car park information popup from the nearby car parks screen. The nearby car parks screen displays car parks within 10 kilometers radius from closest to furthest. A car park card contains car park address, distance to car park, and number of car park slots available. Users can also tap a heart icon on a car park card to add or remove that car park from favourites. Whenever a user taps a car park card, a car park information popup will appear. The car park information screen displays additional details about the car park and users can click back to go back to the nearby car parks screen.



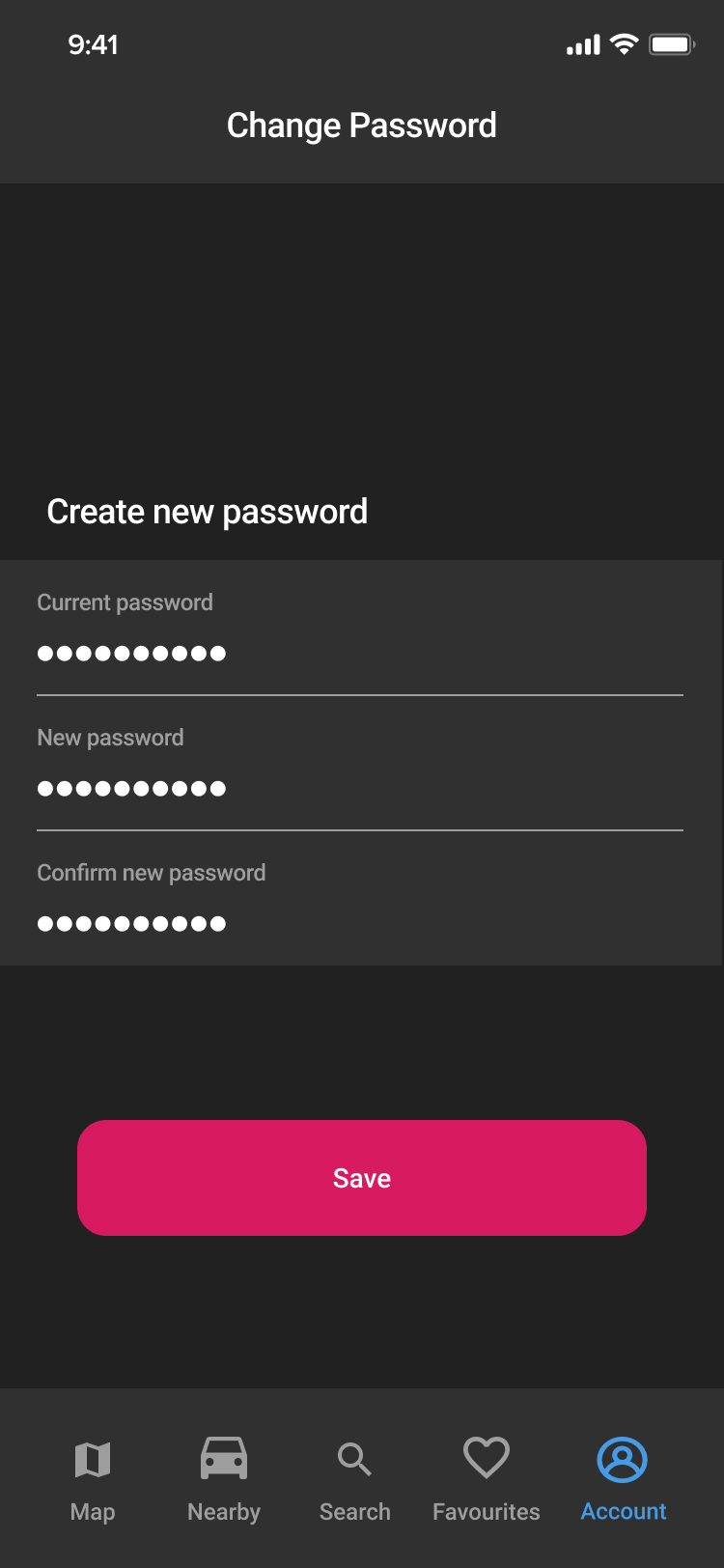
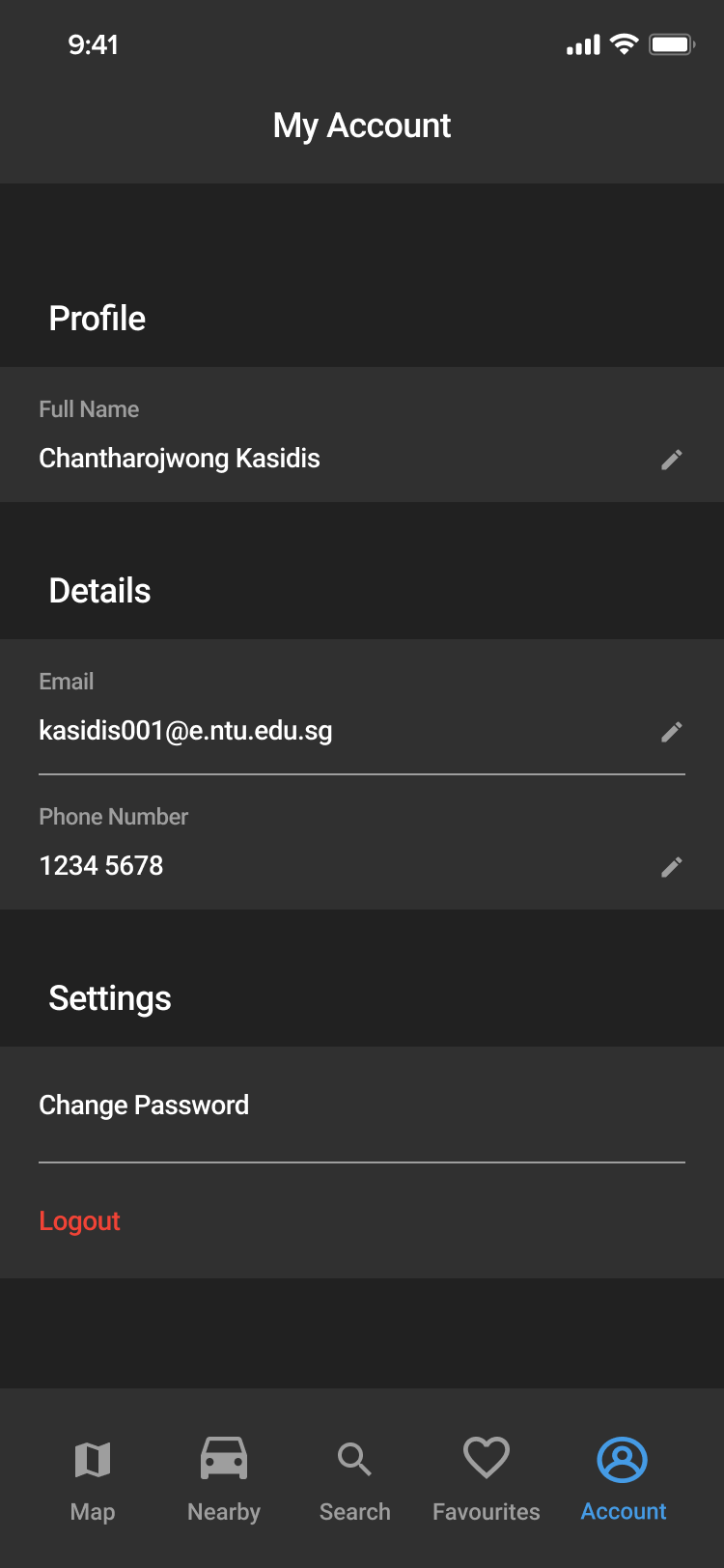
*Figure 4: Search Screen and Car Park Information Popup Screen*

Figure 4 illustrates the search screen and a car park information popup from the search screen. The search screen by default displays all car parks in alphabetical order. When the user enters characters in the search box, the screen will display car parks that match the search query. Whenever a user taps a car park, a car park information popup will appear. The car park information screen displays additional details about the car park and users can click back to go back to the search screen.



*Figure 5: My Favourites Screen and Car Park Information Popup Screen*

Figure 5 illustrates the my favourites screen and a car park information popup from the my favourites screen. My favourites screen displays all car parks added to favourites by the user from closest to furthest. A car park card contains car park address, distance to car park, and number of car park slots available. Users can also tap a heart icon on a car park card to add or remove that car park from favourites. Whenever a user taps a car park card, a car park information popup will appear. The car park information screen displays additional details about the car park and users can click back to go back to my favourites screen.



*Figure 7: My Account and Change Password Screens*

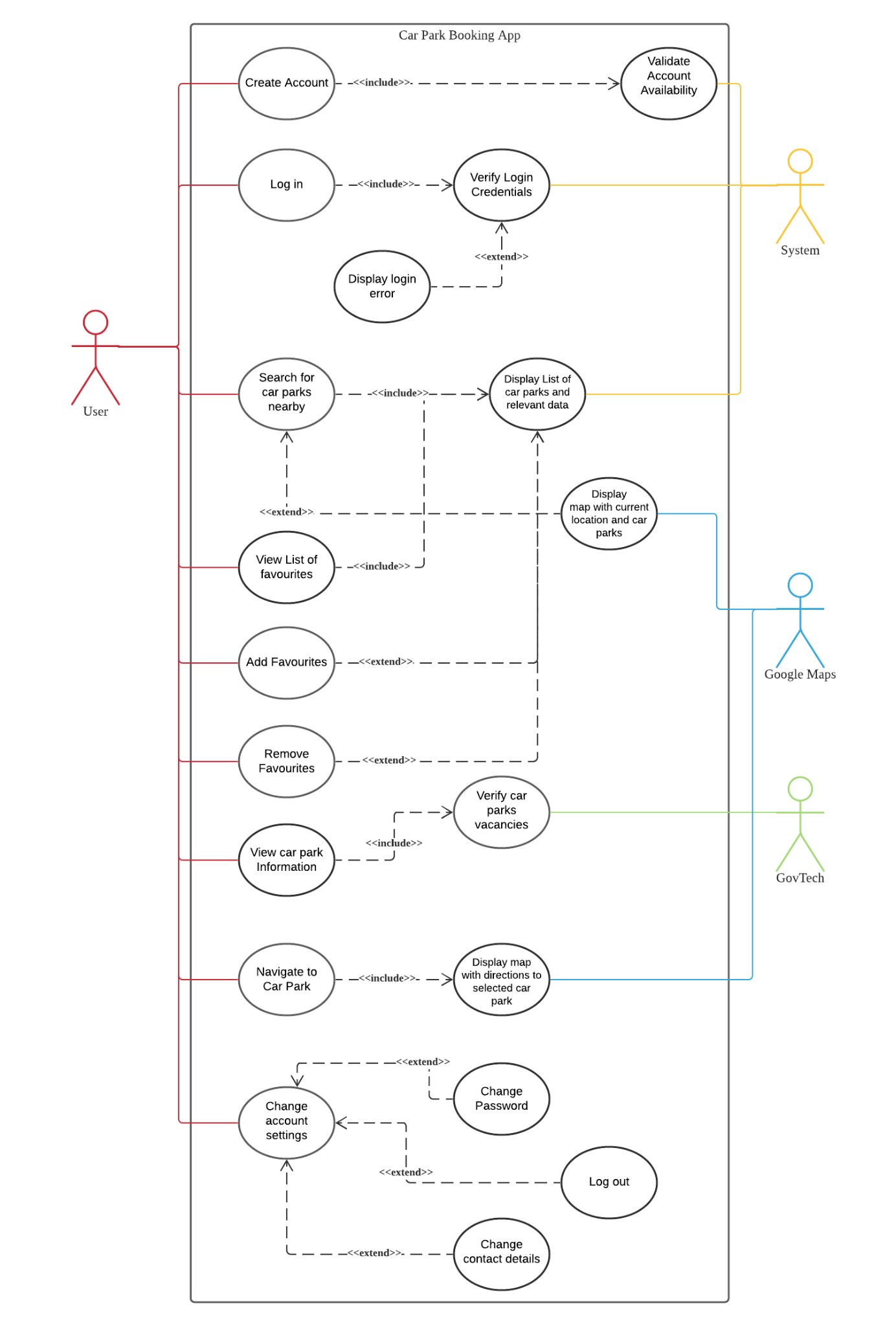
Figure 7 illustrates the my account and change password screens. Users can change personal information and account password from my account screen. In the change password screen, the system requires the user to enter an old password before making any changes. The system will display error messages if the fields are invalid. Users can also logout from my account screen to go back to the login screen.

# 

# **2. Functional Requirements**

1. **Registration**
   1. The application must allow the user to register for a new account.
   2. During registration, the application must validate that the user has entered valid information for the account fields.
      1. The account must have the following fields:
         1. Name
         2. Email address
         3. Password
         4. Confirm password
         5. Phone number
      2. The application must validate that the password is valid.
         1. The application must validate that the password contains at least 8 characters.
         2. The application must validate that the password field and the confirm password field contain the same value.
      3. The application must validate that the email address is valid.
         1. The application must check that the email address is not null or an empty string.
      4. The application must validate that the phone number is valid
         1. The application must check that the phone number has exactly 8 digits.
      5. The application must only create an account after it has validated that all account field information is valid.
2. **Login and Security**
   1. The application must allow the user to login into existing accounts using the account field information.
      1. The login credentials must include the account email address and password.
   2. If the user enters invalid login credentials, the application must return an error message.
3. **Car park information**
   1. The application must allow to user to view information for each car park:
      1. The car park address
      2. The distance between the carpark and the user’s current location
      3. The number of empty lots in the car park at the current time
      4. The car park rates
      5. The car park number
      6. The car park type
      7. The car park parking system
      8. The car park short-term parking
      9. The car park free parking
      10. The car park night parking
      11. The car park basement
4. **Map**
   1. The application must be able to display car parks on a map.
   2. The application must be able to display the user’s current location on a map.
      1. The user’s location marker must always be centered in the middle of the map.
   3. The application must allow the user to get directions to a specific car park by redirecting the user to Google Maps.
5. **Search** 
   1. The application must allow the user to search for nearby car parks with location
      1. The application must list car parks that are within 10km of the user’s current location.
   2. The application must allow the user to search for car parks using keywords.
      1. The application must list car parks with addresses that contain the keywords. (e.g. “Jurong”)
6. **Favourites**
   1. Favourites must be tied to an account.
   2. The application must allow the user to favourite car parks.
   3. The application must allow the user to unfavourite car parks.
   4. The application must allow the user to view the account favourites list.
7. **Account management**
   1. The application must allow the user to manage the account field information.
      1. The application must allow the user to view the account field information.
      2. The application must allow the user to edit the account field information
8. **Interface with other systems**
   1. The application must be able to retrieve the real-time data of available car parks using the GovTech Carpark Availability API
   2. The application must be able to retrieve the user’s current location using the Flutter Geolocation Plugin
   3. The application must allow the user to choose a car park and get directions to the car park using Google Maps.
9. **Formats for information to be processed.**
   1. Price rate
      1. Prices must be displayed in Singapore Dollars (i.e. S$) and have two decimal places (e.g. S$1.00)
   2. Location
      1. Distance units must be in kilometers (i.e. km) and have one decimal place (e.g. 1.2km).

## **2.1. Use Case Diagrams**

****

## **2.2. Use Case Descriptions**

| **Use Case ID:** | 1 | | |
| --- | --- | --- | --- |
| **Use Case Name:** | Login | | |
| **Created By:** | Supraja | **Last Updated By:** | Supraja |
| **Date Created:** | 6th September 2021 | **Date Last Updated:** | 8th September 2021 |

| **Actor:** | User |
| --- | --- |
| **Description:** | User enters login credentials |
| **Preconditions:** | 1. User account(s) must already exist in the system database 2. Phone must be connected to WiFi/mobile data |
| **Postconditions:** | 1. User is able to search for a lot in any HDB car park 2. User is able to favorite car parks |
| **Priority:** | High |
| **Frequency of Use:** | 1 - 3 per lifetime |
| **Flow of Events:** | 1. User enters their credentials in the login interface 2. User selects the login button 3. System performs Use Case 2 (Verify Login Credentials) 4. User is logged into their account |
| **Alternative Flows:** | AF-S1: User resets password before logging in   1. User clicks on “Forgot Password?” 2. User enters their account’s email address. 3. User follows the instructions included in the forgot password page 4. System updates the account’s password in the database. 5. Return to Step 1. |
| **Exceptions:** | - |
| **Includes:** | 2. Verify login credentials |
| **Special Requirements:** | - |
| **Assumptions:** | - |
| **Notes and Issues:** | - |

| **Use Case ID:** | 2 | | |
| --- | --- | --- | --- |
| **Use Case Name:** | Verify Login Credentials | | |
| **Created By:** | Wee Hsuan | **Last Updated By:** | Wee Hsuan |
| **Date Created:** | 7th September 2021 | **Date Last Updated:** | 26th October 2021 |

| **Actor:** | System |
| --- | --- |
| **Description:** | System will verify the login credentials based on whether the entered username and password has a match in the system’s database. |
| **Preconditions:** | 1. User account must already exist in the system database 2. Phone must be connected to WiFi/mobile data |
| **Postconditions:** | 1. User is able to favorite car parks |
| **Priority:** | High |
| **Frequency of Use:** | 1 - 3 times per lifetime |
| **Flow of Events:** | 1. User enters their registered email address and password in the respective fields on the login page. 2. User clicks on the “Login” button. 3. System validates the account by checking the user’s credentials against the system’s database. 4. System verifies the user login successfully. |
| **Alternative Flows:** | AF-S1: System detects an empty email address and/or password fields.   1. System displays an error message: “Email address or password fields cannot be empty.” 2. User fills up the empty field. 3. User clicks the “Login” button to reattempt login. 4. Return to Step 3.   AF-S2: User enter wrong credentials   1. System displays an error message: “Wrong email address and/or password. Please try again.” 2. Return to Step 1. |
| **Exceptions:** | - |
| **Includes:** | 1. Login |
| **Special Requirements:** | - |
| **Assumptions:** | - |
| **Notes and Issues:** | - |

| **Use Case ID:** | 3 | | |
| --- | --- | --- | --- |
| **Use Case Name:** | Search for Nearby Parking Lots | | |
| **Created By:** | Supraja | **Last Updated By:** | Glenda |
| **Date Created:** | 6th September 2021 | **Date Last Updated:** | 7th September 2021 |

| **Actor:** | User |
| --- | --- |
| **Description:** | To search for available car parks using a particular location |
| **Preconditions:** | 1. Phone must be connected to WiFi/ mobile data 2. User needs to turn on GPS 3. User specifies location 4. User must be logged into an account |
| **Postconditions:** | 1. User obtains nearby available car parks from the search location. 2. User obtains car parks that are available at the date and time when the search was made |
| **Priority:** | High |
| **Frequency of Use:** | 0-20 times a day |
| **Flow of Events:** | 1. User inputs his/her query into the search bar in the home page 2. System retrieves a list of available car parks and their relevant data from the GovTech databaseSystem displays a list of the car parks, with the car parks closer to the queried location listed first 3. User selects his/her preferred car park location |
| **Alternative Flows:** | 3.AC.1: User’s input does not match any keywords in search results   1. Display “No search results” and prompt user to edit his/her input 2. Return to step 1   3.AC.2.: User uses their current location to search rather than inputting one   1. User clicks on the “OPEN MAP” button on the home page 2. System retrieves a list of available car parks and their relevant data from the GovTech database based on the user’s current location 3. Google Map displays the car parks on the map 4. Return to step 4 |
| **Exceptions:** | - |
| **Includes:** | 1. Display list of parking lots and relevant data |
| **Special Requirements:** | - |
| **Assumptions:** | Google Maps uses GPS to locate car parks using data from GovTech |
| **Notes and Issues:** | - |

| **Use Case ID:** | 4 | | |
| --- | --- | --- | --- |
| **Use Case Name:** | View List of Favourites | | |
| **Created By:** | Supraja | **Last Updated By:** | Supraja |
| **Date Created:** | 6th September 2021 | **Date Last Updated:** | 7th September 2021 |

| **Actor:** | User |
| --- | --- |
| **Description:** | To retrieve the list of the user’s favourite car parks |
| **Preconditions:** | 1. Phone must be connected to WiFi/ mobile data 2. User must be logged into an account |
| **Postconditions:** | 1. A list of up to 10 favourite car parks will be displayed on the Favourites page |
| **Priority:** | Low |
| **Frequency of Use:** | 0-5 times per day |
| **Flow of Events:** | 1. User navigates to the Favourites tab 2. Application will retrieve the list of user’s favourite car parks 3. Application displays a list of the user’s favourite car parks 4. User chooses car park |
| **Alternative Flows:** | 4.AC.1 User navigates to favourites page using “My Favourites” card on the Nearby tab   1. User clicks on the “My Favourites” card on the Nearby tab 2. Return to step 2 |
| **Exceptions:** | - |
| **Includes:** | 1. Display list of parking lots and relevant data |
| **Special Requirements:** | - |
| **Assumptions:** | - |
| **Notes and Issues:** | - |

| **Use Case ID:** | 5 | | |
| --- | --- | --- | --- |
| **Use Case Name:** | Change Account Settings | | |
| **Created By:** | Supraja | **Last Updated By:** | Glenda |
| **Date Created:** | 6th September 2021 | **Date Last Updated:** | 7th September 2021 |

| **Actor:** | User |
| --- | --- |
| **Description:** | User wants to change his/her account settings |
| **Preconditions:** | 1. User account(s) exists in the system database 2. Phone must be connected to WiFi/ mobile data 3. User is logged into an account |
| **Postconditions:** | 1. Account details are changed according to what the user wanted |
| **Priority:** | Low |
| **Frequency of Use:** | 10 times per lifetime |
| **Flow of Events:** | 1. User navigates to the ‘Account’ page 2. User does one of the following to edit a specific account field:    1. User clicks on the pen icon next to the email input box to edit the email address    2. User clicks on the pen icon next to the phone no. input box to edit the phone number    3. User clicks on “change password” 3. User types in the new account field information 4. System updates the database to match the changes |
| **Alternative Flows:** | - |
| **Exceptions:** | - |
| **Includes:** | - |
| **Special Requirements:** | - |
| **Assumptions:** | - |
| **Notes and Issues:** | - |

| **Use Case ID:** | 6 | | |
| --- | --- | --- | --- |
| **Use Case Name:** | Create Account | | |
| **Created By:** | Wee Hsuan | **Last Updated By:** | Wee Hsuan |
| **Date Created:** | 7th September 2021 | **Date Last Updated:** | 7th September 2021 |

| **Actor:** | User |
| --- | --- |
| **Description:** | Registration for a new user account |
| **Preconditions:** | 1. Phone must be connected to WiFi/ mobile data |
| **Postconditions:** | 1. User will have an account that can be logged into in the future |
| **Priority:** | High |
| **Frequency of Use:** | 1-3 times per lifetime |
| **Flow of Events:** | 1. User fills up the “name”, “email address”, “password”, “confirm password” and “phone number” fields. 2. User clicks the register button. 3. System performs Use Case 9 Validate Account Availability 4. System validates that the “password” and “confirm password” fields are identical. 5. System returns a message stating that the account has been successfully created |
| **Alternative Flows:** | 6.AC.1: System detects a mismatch between the password and confirm password fields   1. System displays an error message: “The 2 passwords you have keyed in are different. Please try again” 2. Return to Step 1 |
| **Exceptions:** | - |
| **Includes:** | - |
| **Special Requirements:** | - |
| **Assumptions:** | - |
| **Notes and Issues:** | - |

| **Use Case ID:** | 7 | | |
| --- | --- | --- | --- |
| **Use Case Name:** | Navigate to Car Park | | |
| **Created By:** | Wee Hsuan | **Last Updated By:** | Wee Hsuan |
| **Date Created:** | 7th September 2021 | **Date Last Updated:** | 26th October 2021 |

| **Actor:** | User |
| --- | --- |
| **Description:** | Use in-app Maps to navigate from current location to the selected car park |
| **Preconditions:** | 1. User account(s) exists in the system database 2. Phone must be connected to WiFi/ mobile data |
| **Postconditions:** | 1. User reached the selected car park |
| **Priority:** | Medium |
| **Frequency of Use:** | 0-10 times per day |
| **Flow of Events:** | 1. User clicks on any car park information tab 2. User clicks on “View on Map” 3. User clicks on the arrow button 4. User will be navigated from current location to the selected car park |
| **Alternative Flows:** | - |
| **Exceptions:** | - |
| **Includes:** | 1. Display map with directions to selected car park |
| **Special Requirements:** | - |
| **Assumptions:** | - |
| **Notes and Issues:** | - |

| **Use Case ID:** | 9 | | |
| --- | --- | --- | --- |
| **Use Case Name:** | Validate Account Availability | | |
| **Created By:** | Wee Hsuan | **Last Updated By:** | Supraja |
| **Date Created:** | 7th September 2021 | **Date Last Updated:** | 7th September 2021 |

| **Actor:** | System |
| --- | --- |
| **Description:** | System will verify the account availability, send verification email and message and create the account |
| **Preconditions:** | 1. User account must not exists in the system database 2. Phone must be connected to WiFi/ mobile data |
| **Postconditions:** | 1. User account is created in the system’s database 2. User will be able to log in his/her registered account |
| **Priority:** | Medium |
| **Frequency of Use:** | 1-3 times per lifetime |
| **Flow of Events:** | 1. System will validate the user account availability in the system’s database. 2. System will create the user account in the system’s database. |
| **Alternative Flows:** | 9.AC.1: System detects the user account is already in the database   1. System displays an error message: “The email address you have entered is already registered.” 2. User will key in another email address. 3. User clicks on the “Register” button. 4. Return to Step 1 |
| **Exceptions:** | - |
| **Includes:** | - |
| **Special Requirements:** | - |
| **Assumptions:** | - |
| **Notes and Issues:** | - |

| **Use Case ID:** | 11 | | |
| --- | --- | --- | --- |
| **Use Case Name:** | Add to Favourites | | |
| **Created By:** | Wee Hsuan | **Last Updated By:** | Wee Hsuan |
| **Date Created:** | 7th September 2021 | **Date Last Updated:** | 26th October 2021 |

| **Actor:** | User |
| --- | --- |
| **Description:** | User can add car park entries to the list of favourites |
| **Preconditions:** | 1. User account must exist in the system database 2. Phone must be connected to WiFi/ mobile data |
| **Postconditions:** | 1. User’s favourites will be saved to the system’s database |
| **Priority:** | High |
| **Frequency of Use:** | 0-20 times per day |
| **Flow of Events:** | 1. User scrolls to view the nearby car parks on the home page. 2. User clicks on the heart shape on the top right of each car park entry. 3. System will save the user’s favourite car park to the system’s database. |
| **Alternative Flows:** | 11.AC.1 User goes to search car park   1. User types in the search field on the home page 2. System will display a list of car parks based on the user entry. 3. Return to Step 2 |
| **Exceptions:** | - |
| **Includes:** | - |
| **Special Requirements:** | - |
| **Assumptions:** | - |
| **Notes and Issues:** | - |

| **Use Case ID:** | 12 | | |
| --- | --- | --- | --- |
| **Use Case Name:** | Remove Favourites | | |
| **Created By:** | Wee Hsuan | **Last Updated By:** | Wee Hsuan |
| **Date Created:** | 7th September 2021 | **Date Last Updated:** | 7th September 2021 |

| **Actor:** | User |
| --- | --- |
| **Description:** | User can remove car park entries to the list of favourites |
| **Preconditions:** | 1. User account must exist in the system database 2. Phone must be connected to WiFi/ mobile data |
| **Postconditions:** | 1. User’s favourites will be saved to the system’s database |
| **Priority:** | High |
| **Frequency of Use:** | 0-20 times per day |
| **Flow of Events:** | 1. User scrolls to view the nearby car parks on the home page. 2. User clicks on the filled heart shape on the top right of each car park entry. 3. System will remove the user’s favourite car park from the system’s database. |
| **Alternative Flows:** | 11.AC.1 User goes to search car park   1. User types in the search field on the home page. 2. System will display a list of car parks based on the user entry. 3. Return to Step 2   11.AC.2 User goes to list of favourites   1. User clicks on the “My favourites” tab on the home page. 2. System will display the list of favourited car parks. 3. Return to step 2 |
| **Exceptions:** | - |
| **Includes:** | - |
| **Special Requirements:** | - |
| **Assumptions:** | - |
| **Notes and Issues:** | - |

| **Use Case ID:** | 13 | | |
| --- | --- | --- | --- |
| **Use Case Name:** | Display List of Car Park and Relevant Data | | |
| **Created By:** | Supraja | **Last Updated By:** | Supraja |
| **Date Created:** | 10th September 2021 | **Date Last Updated:** | 10th September 2021 |

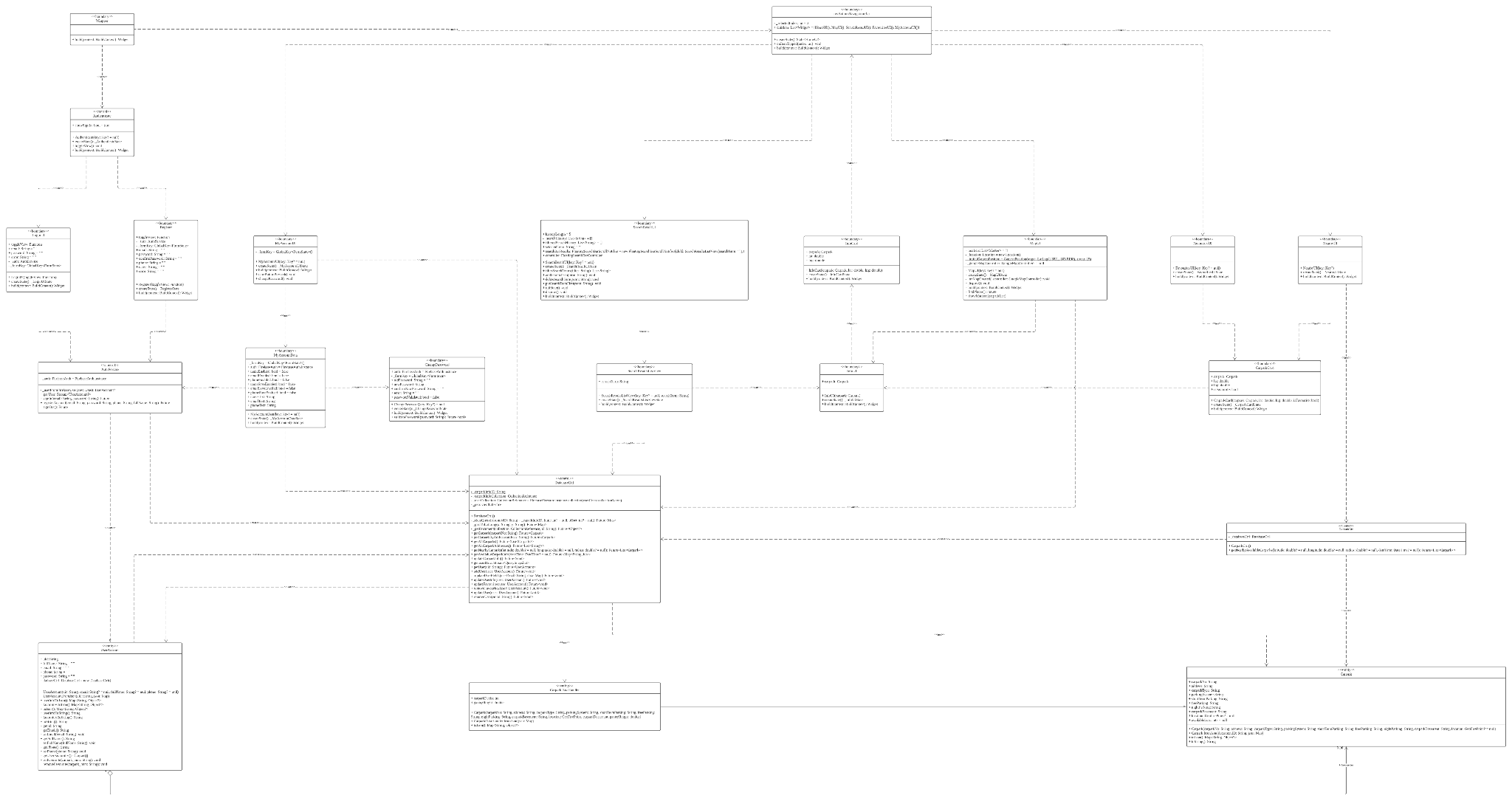
| **Actor:** | GovTech |
| --- | --- |
| **Description:** | To display the list of car parks and their relevant data |
| **Preconditions:** | 1. Phone must be connected to WiFi/ mobile data |
| **Postconditions:** | 1. GovTech produces/ displays car park data for the user’s reference |
| **Priority:** | High |
| **Frequency of Use:** | 0-10 times per day |
| **Flow of Events:** | 1. User inputs search keywords by name/ number or address 2. System will fetch the data from GovTech 3. System will provide a selectable list of parking lot 4. System will display a list of relevant car parks and their details |
| **Alternative Flows:** | 13.AF.1: User selects input from recent searches  Return to step 2  13.AF.2: User view their list of favourite car parks  Return to step 2  13.AF.3: User chooses to add favourites  Return to step 2 |
| **Exceptions:** | 13.EX.1: Input does not match any keywords of car park name, number or address   1. Display “No results found” 2. Return to step 1 |
| **Includes:** | - |
| **Special Requirements:** | - |
| **Assumptions:** | - |
| **Notes and Issues:** | - |

| **Use Case ID:** | 14 | | |
| --- | --- | --- | --- |
| **Use Case Name:** | Change Password | | |
| **Created By:** | Supraja | **Last Updated By:** | Supraja |
| **Date Created:** | 10th September 2021 | **Date Last Updated:** | 10th September 2021 |

| **Actor:** | User |
| --- | --- |
| **Description:** | To allow user to change or reset their passwords in the event that the user is unable to remember |
| **Preconditions:** | 1. Phone must be connected to WiFi/ mobile data |
| **Postconditions:** | User is able to login to their account using the new password |
| **Priority:** | High |
| **Frequency of Use:** | 3-5 times per year |
| **Flow of Events:** | 1. User navigates to the Account page and clicks on the Change Password button 2. Change Password page is opened 3. User enters current password for verification 4. User enters the same new password twice 5. Password is changed successfully |
| **Alternative Flows:** | 14.AC.1: User enters wrong password in the current password field   1. System displays an error message: “The password you have keyed is incorrect. Please try again” 2. Return to Step 1   14.AC.2: System detects a mismatch between the new password and confirm new password fields   1. System displays an error message: “The 2 passwords you have keyed in are different. Please try again” 2. Return to Step 1 |
| **Exceptions:** | - |
| **Includes:** | - |
| **Special Requirements:** | - |
| **Assumptions:** | - |
| **Notes and Issues:** | - |

## 

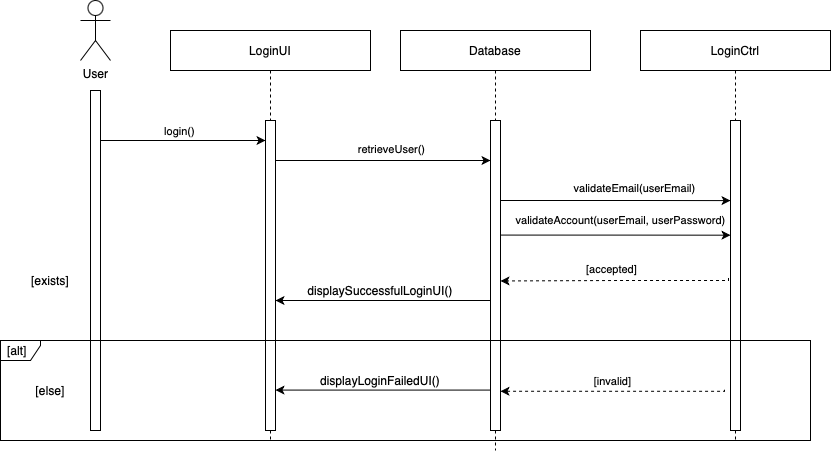
## 2.3. Class Diagram



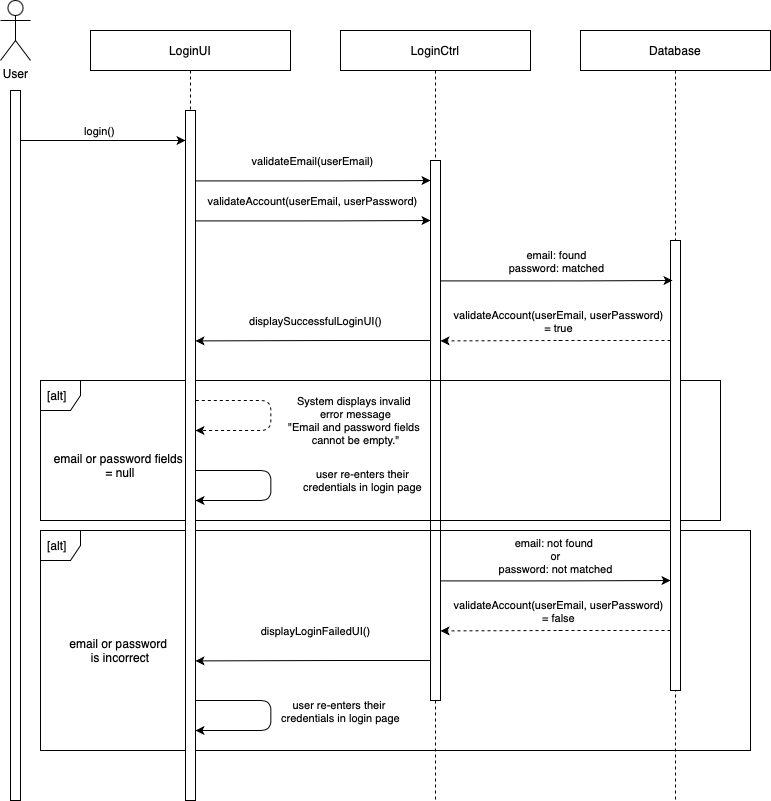
(Higher resolution PDF version of diagram available)

## 2.4. Sequence Diagram

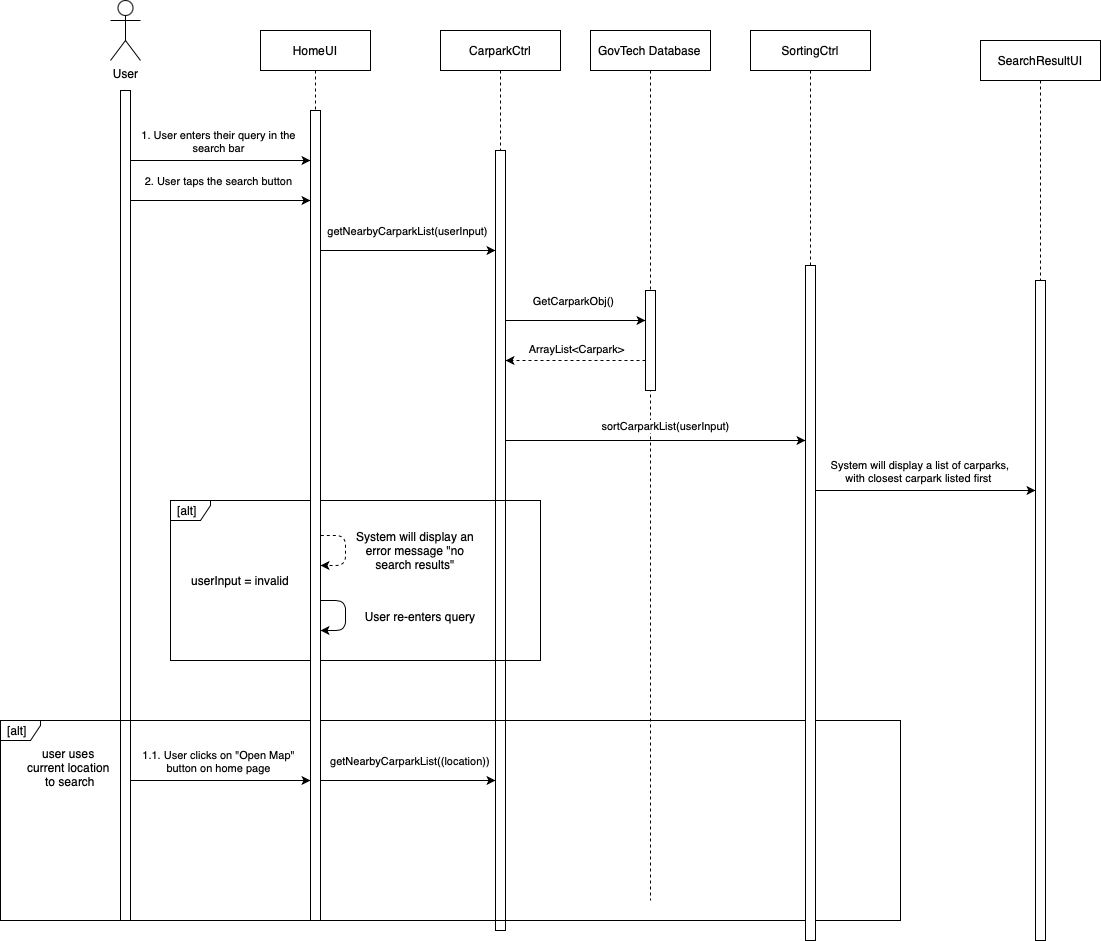
1. **Login**



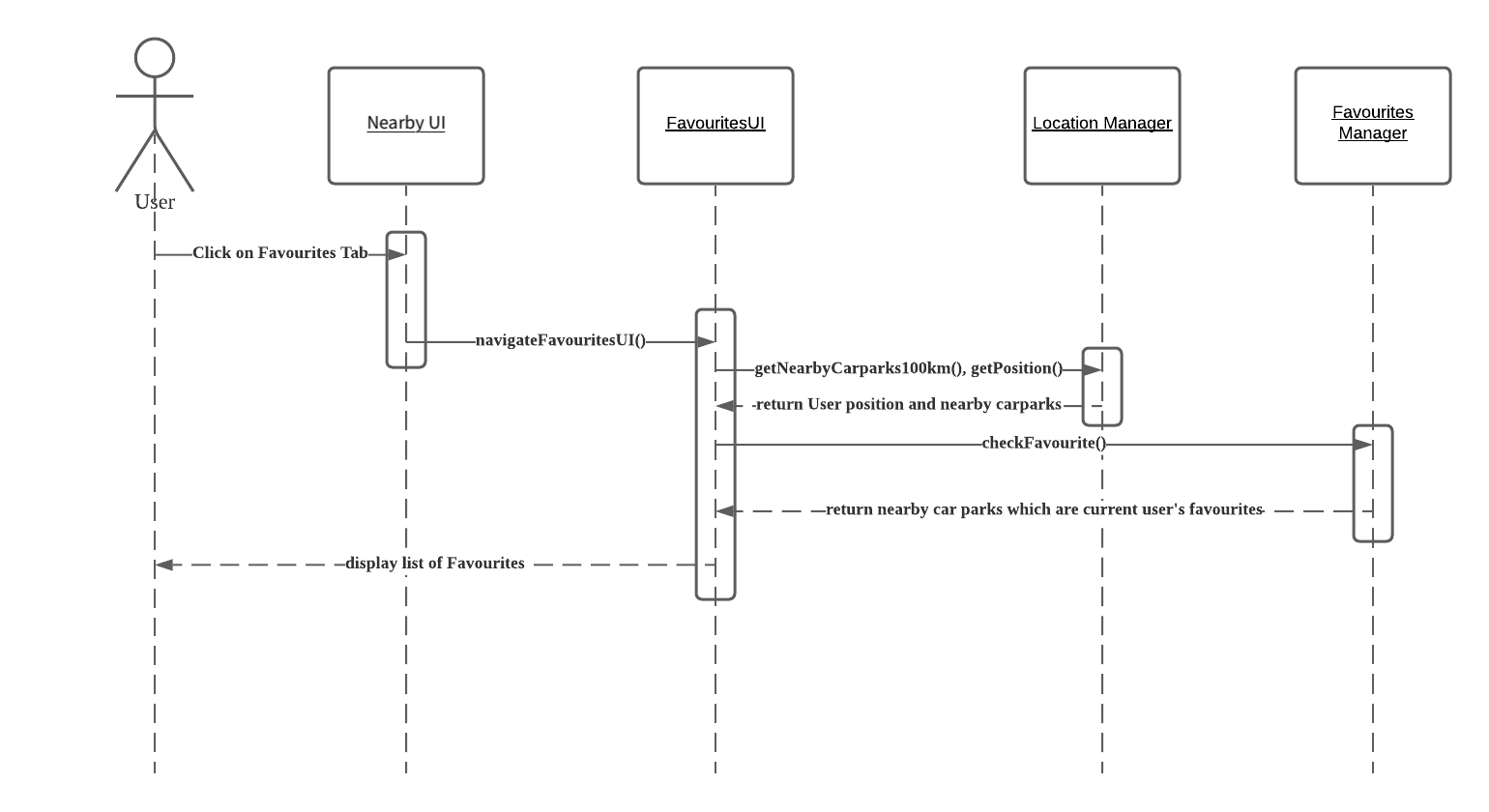
1. **Verify Login**



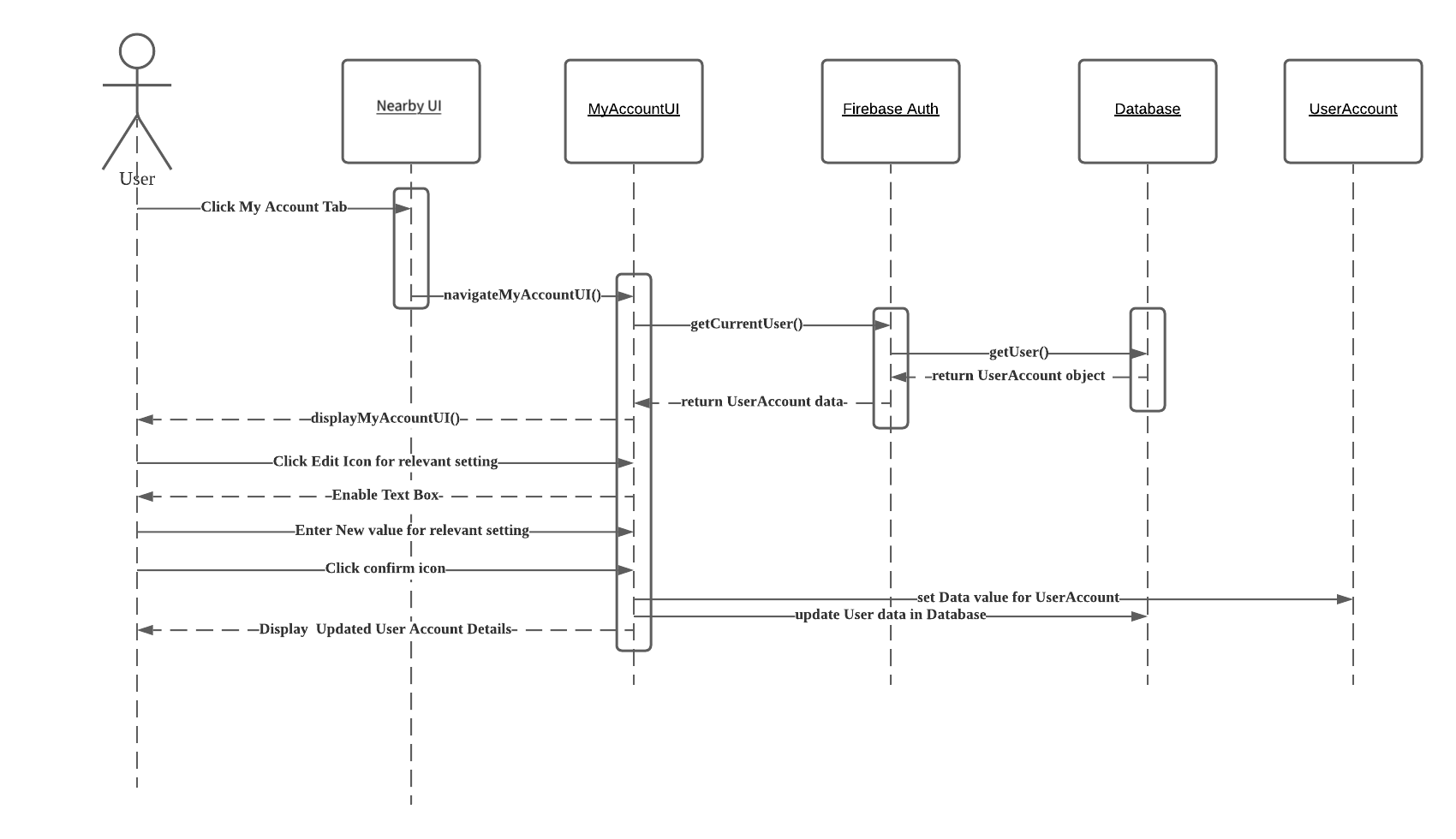
1. **Search for nearby parking lots**



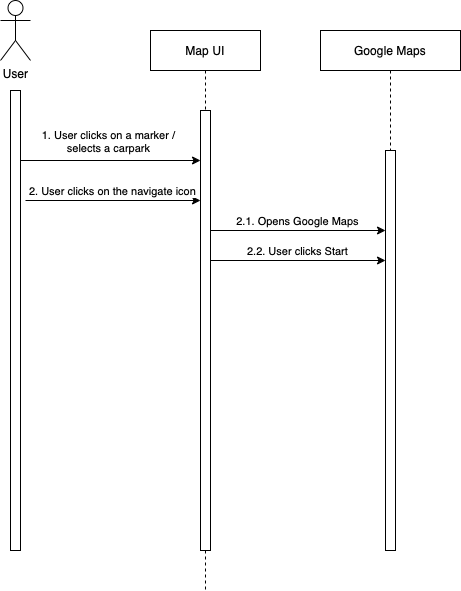
1. **View List of Favourites**



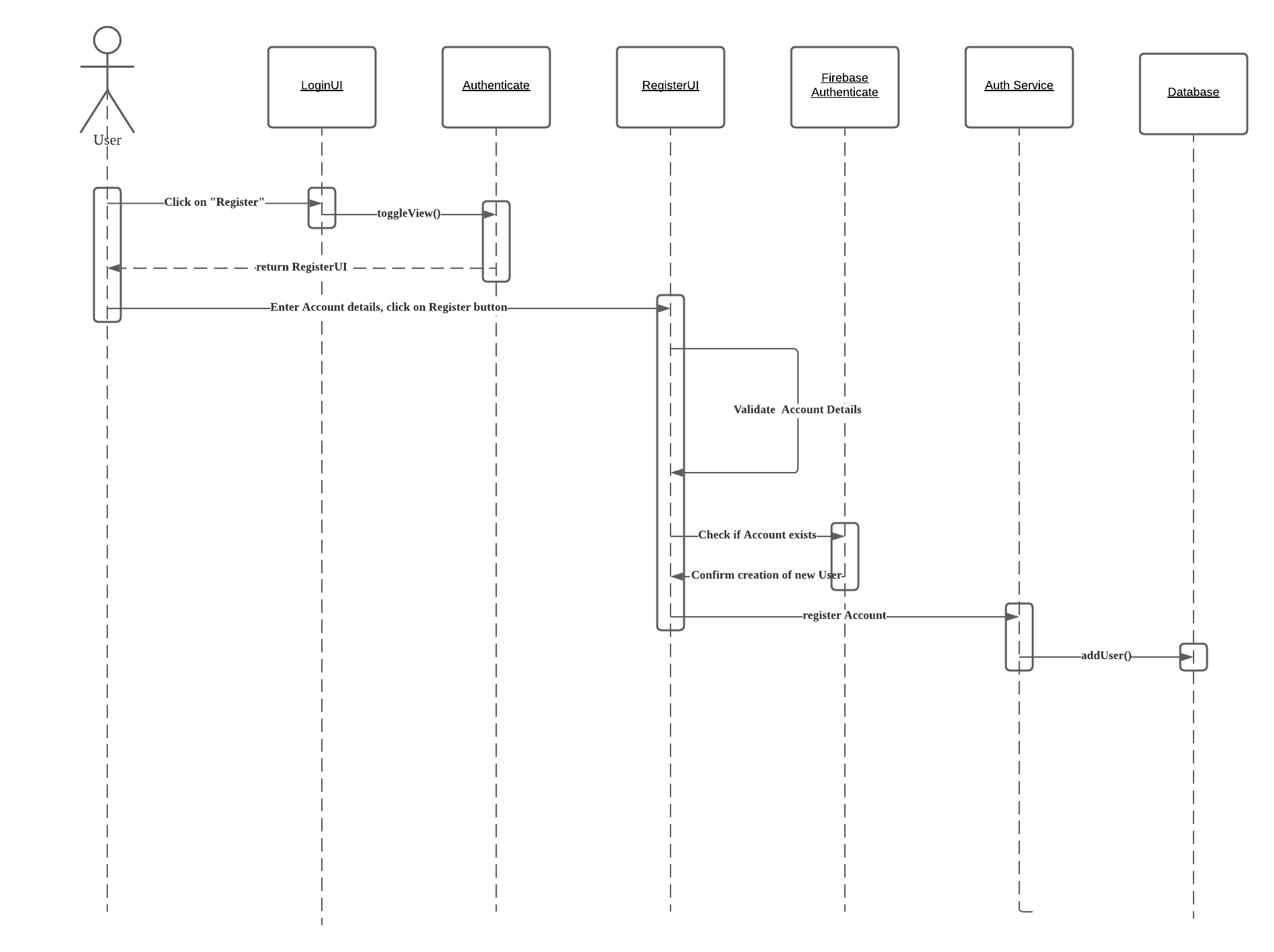
1. **Change Account Settings**



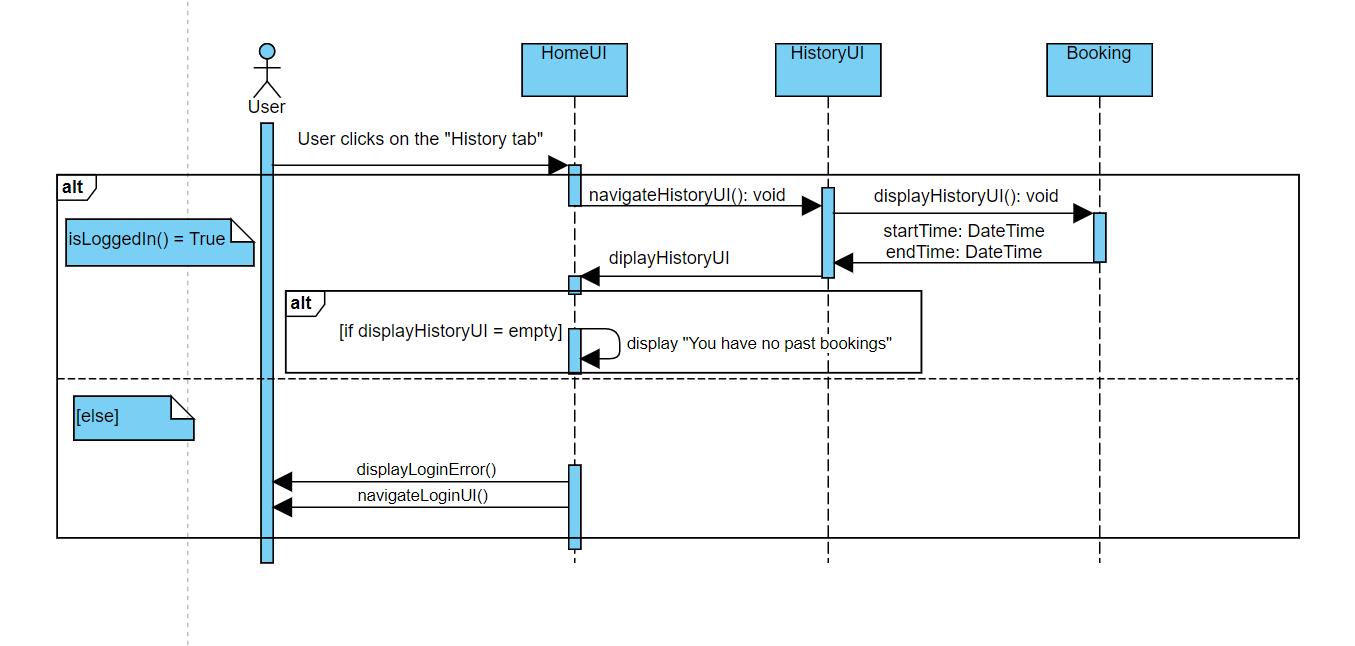
1. **Navigate to Car Park**



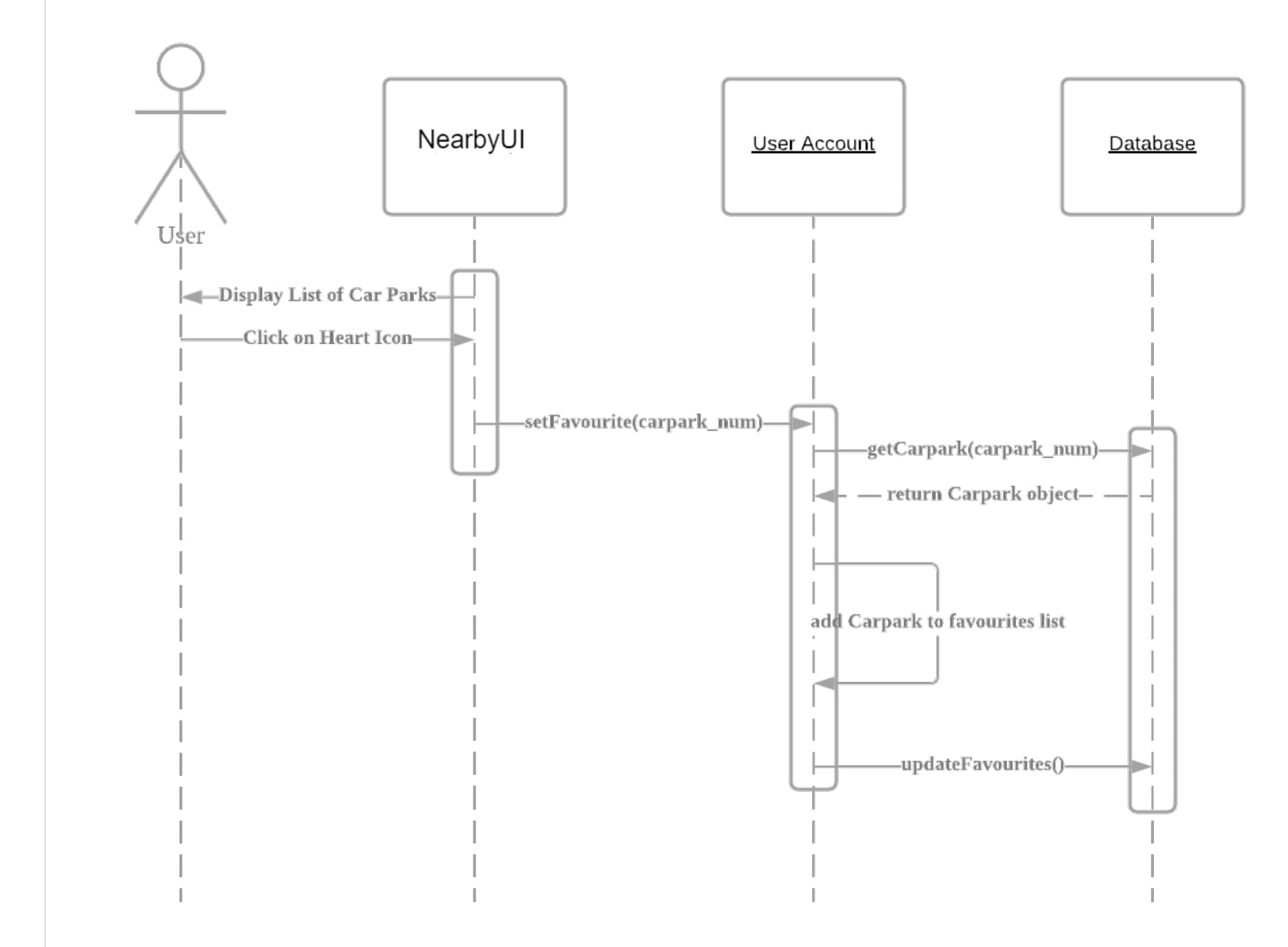
1. **Create New Account**



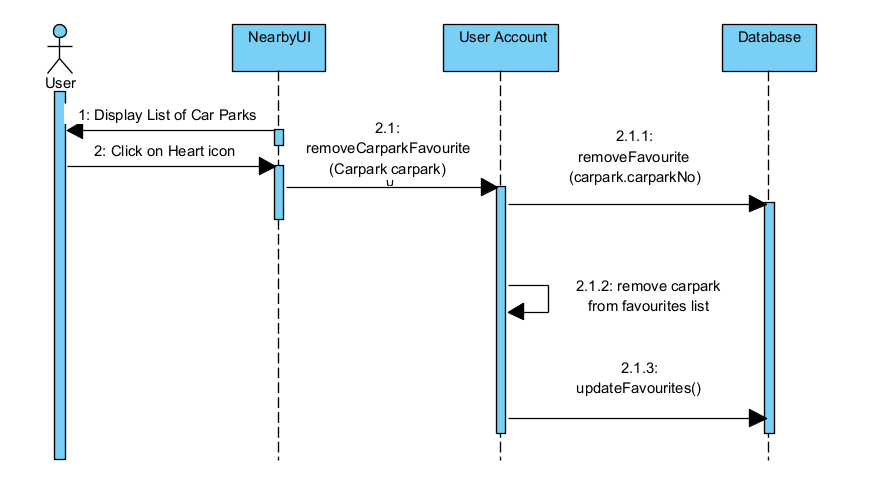
1. **Check Parking History**



1. **Add to Favourites**



1. **Remove favourites**



## 2.5. Dialog Map

# 

# 3. Non-Functional Requirements

1. **Usability Requirements**
   1. When moving from page to page, the application must not require the user to remember more than 5 chunks of information. (reduce short-term memory load)
   2. The application must allow the user to reverse an action within 5 steps. (easy reversal of actions)
      1. The reversal of actions must include:
         1. Removing bookmarks from the bookmark list.
   3. The application must return error messages (offer informative feedback) when:
      1. The user enters invalid inputs
      2. An application process was unsuccessful
   4. The application must be consistent.
      1. The application must have consistent sequences of actions for similar situations.
         1. The application must use the same transitions between pages throughout its interface.
      2. The application must have a consistent visual layout for its interface.
         1. The application must use the same font, shapes, type of labels and colour scheme throughout its interface.
         2. The application must use the same animations throughout its interface.
2. **Performance Requirements**
   1. The application must be able to maintain with little or no downtime occurring.
      1. 95% of users must be able to use the application without any system breakdown.
      2. The application must not crash during run-time.
   2. The application must be able to support real-time retrieval of information from the GovTech Carpark Availability API.
3. **Security Requirements**
   1. During login, the application must mask the password field with asterisks to prevent any potential shoulder surfing.
   2. The application must use the Firebase Authentication feature to store passwords to ensure password security.
   3. During login, the application must verify whether the user has entered the correct password using the Firebase Authentication feature to ensure password security.
4. **Extendability Requirements**
   1. The application must be designed using Model-View-Controller architecture.

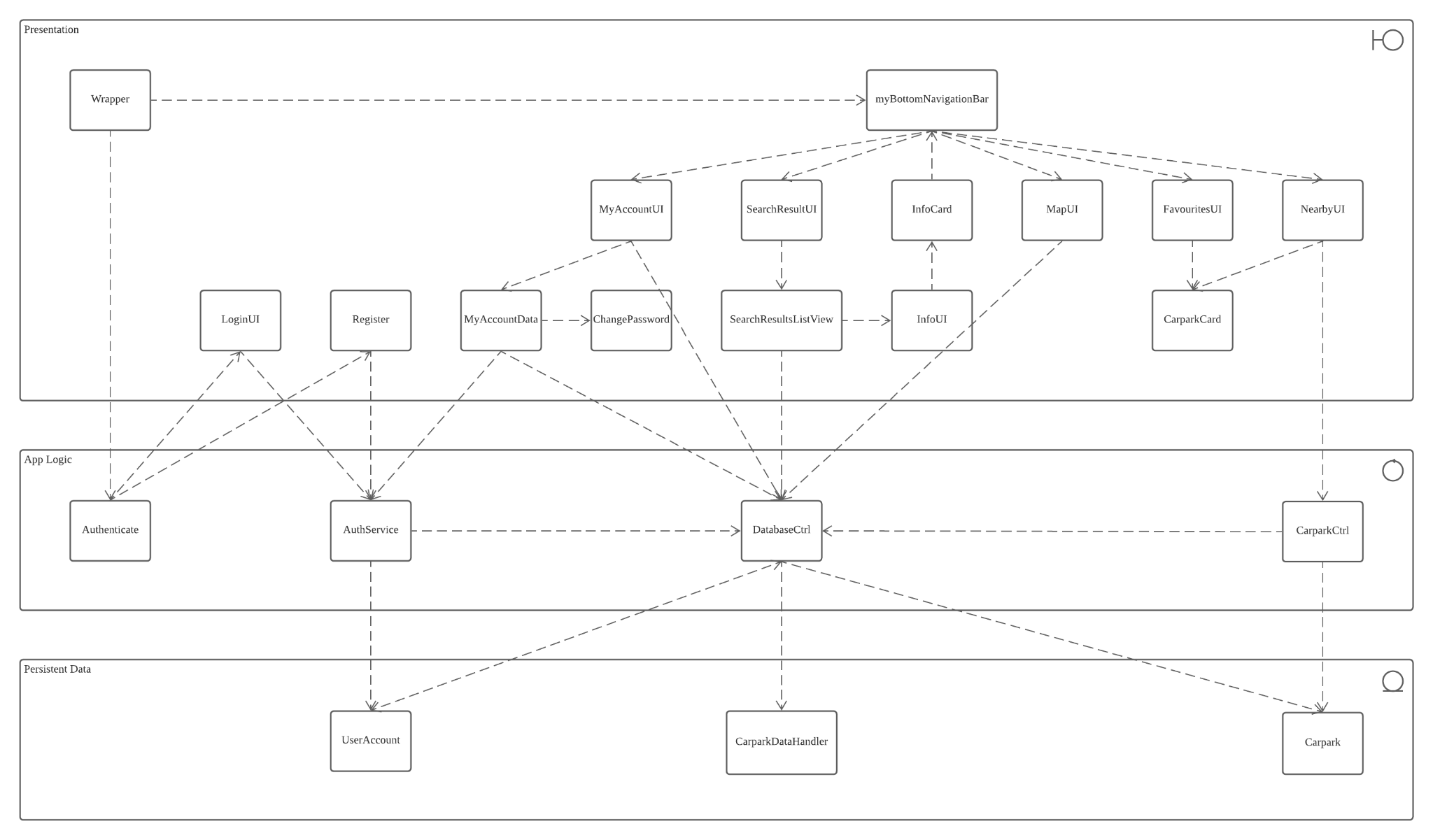
# 4. Interface Requirements

1. **Usability Requirements**
   1. The application works for all types of users who would like to search for nearby car parks.
   2. Currently the application does not support special filters of nearby car parks (e.g. night parking, etc.)
2. **Software**
   1. The application is designed to work on Android devices with location service enabled and with internet connection to locate the user’s current location.

# 

# **5. Architecture Design**

## 5.1. System Architecture Diagram



## 5.2. Design Pattern

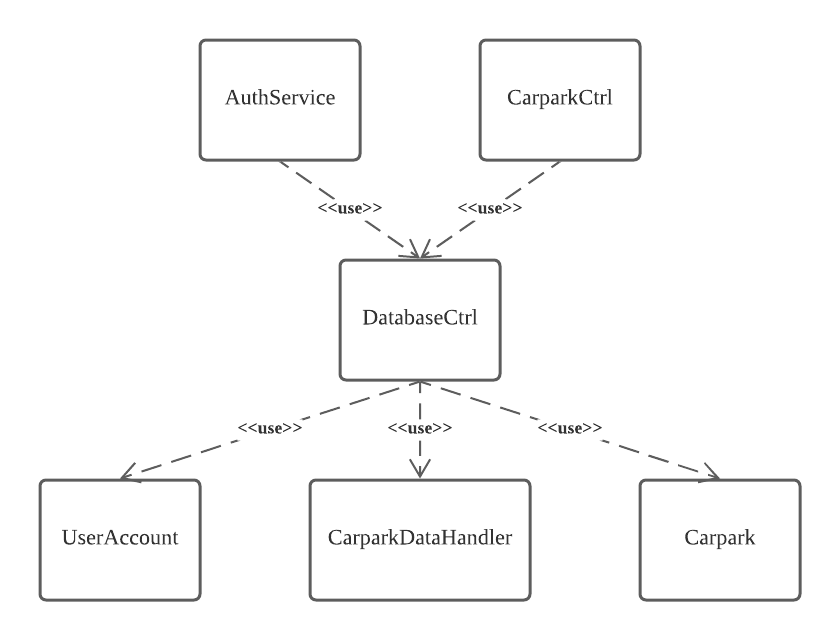
1. **Data Access Object Pattern**

**Problem**

Mixing business logic and data access logic into the same classes makes the code confusing. It is difficult to keep track of which classes perform data handling operations, what data the classes have access to, and what operations the classes perform.

**Solution**

We used the data access object (DAO) pattern to separate the business logic and data access logic into different classes.



In our implementation, the DatabaseCtrl class serves as our DAO implementation. This class handles all CRUD operations on our data. The UserAccount, CarparkDataHandler and Carpark classes are the model object classes. These classes serve as entity representations of our data. The AuthService and CarparkCtrl classes are control classes that use DatabaseCtrl. The classes implement the business logic.

**Consequences**

The benefit of using this pattern is that all data handling operations have been concentrated into a single class, which makes the operations easy to keep track of. It also means that if we want to make changes to the database in the future, only a single class needs to be edited to match the changes.

# 6. Data Dictionary

| **Term** | **Definition** |
| --- | --- |
| Car Park Type | A car park refers to an area or building where drivers can park their car temporarily. There are seven types of car park: surface car park, multi-storey car park, basement car park, covered car park, mechanised car park, mechanised and surface car park and surface/multi-storey car park |
| Car Park Rates | Rates refer to the amount charged to drivers for parking their cars in the car park. Rates could be charged per 30 minutes or per hour. |
| Car Park Number | Every car park will have a unique number to identify them |
| Car Park Address | Car park address refers to the exact location the car park is located in. |
| Car Park Basement | A boolean function (Yes/No) to show if the car park has basements |
| Car Park Lots | A car park will have multiple lots for drivers to park their car. |
| Night Parking | A boolean function (Yes/No) to show if the car park allows cars to be parked overnight |
| Free Parking | Some car parks provide free parking during certain days such as Sunday and public holiday |
| Short term parking | Some car parks only allows season parking holders to park their cars there while some car parks allow non-season parking holders to park during a timeframe or the whole day |
| Type of Parking System | Each car park has a type of parking system which refers to the payment type. There are two types of parking system: Electronic parking and Coupon parking |
| X-/Y- Coordinates | The X- and Y- coordinates locate the exact location of a car park based on the scale of the map. |
| User | User refers to anyone using the application |
| Favourites | Favourites is a feature that allows users to add their favourite or frequently used car park and users can easily retrieve this list. |
| Registration | Registration is a feature that allows new user to sign up for an account and use the application features such as bookmarks and book a car park |
| GPS | GPS refers to Global Positioning System which can detect the user’s current location |
| Shoulder Surfing | Shoulder spying is the practice of spying on the user when they are typing their credentials into an electronic device in order to obtain their password. |
| Search | Search is a feature that enables users to access available car parks based on a few filters. |
| Error message | An error message is a message displayed on the screen when an incorrect input is given by the user or when an application process is unsuccessful. |
| Account Fields | Account fields are information fields that are unique to each user account. Every account has four account fields: name, email address, password and phone number. |
| Login Credentials | Login credentials are account fields used to login to a specific user account. There are two account fields needed to login: email address and password. |

# 7. Testing

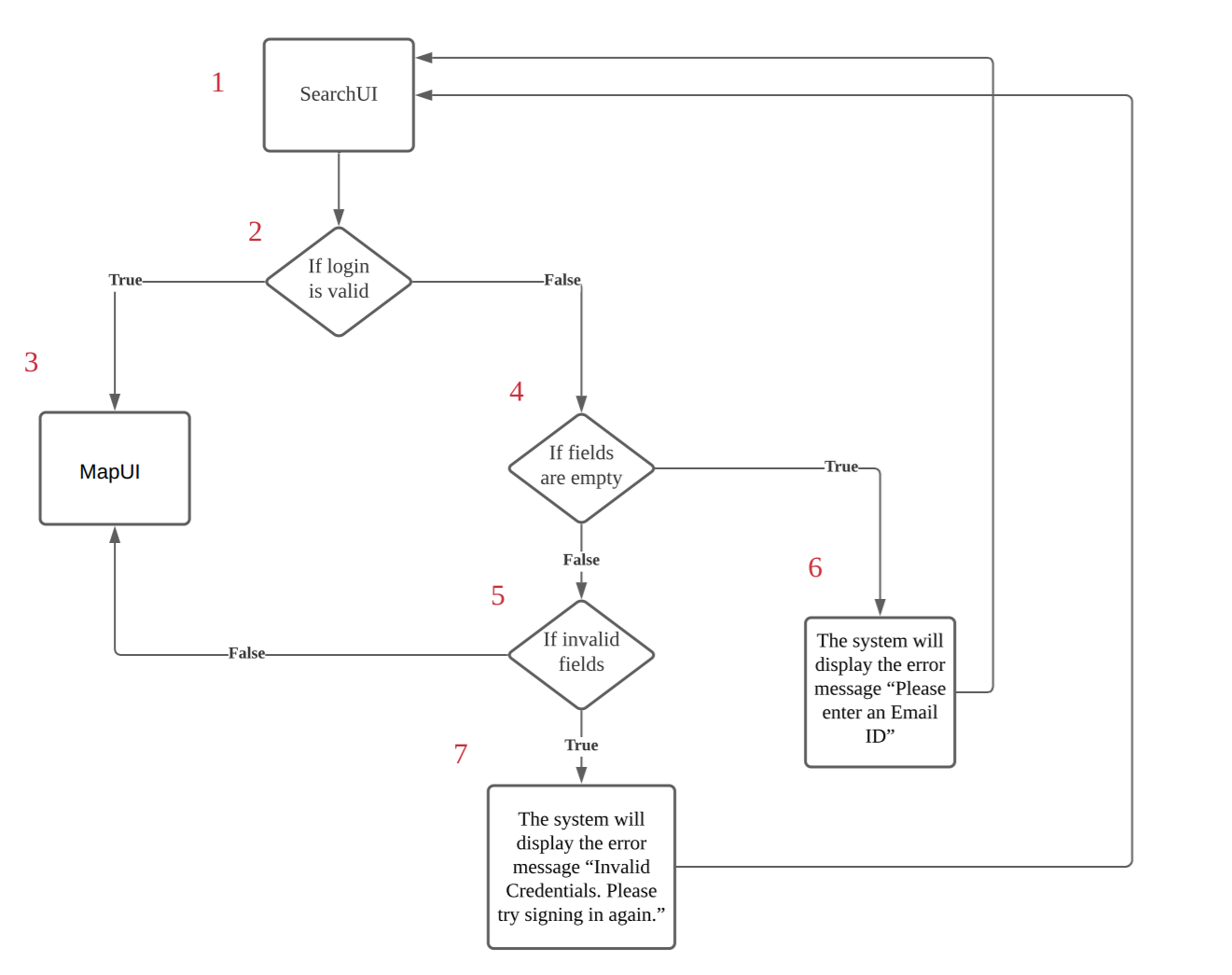
## 7. 1. Black Box Testing

1. **DatabaseCtrl**

| **Test ID** | **Scenario** | **Test Input** | **Test Oracle**  **(Expected Results)** | **Test Log**  **(Actual Results)** |
| --- | --- | --- | --- | --- |
| 1 | Pass a valid carpark number into getCarpark() | carparkNo = “A1” | A Carpark object with fields corresponding to carpark A1. | A Carpark object with fields corresponding to carpark A1. |
| 2 | Pass a valid address into getCarparkByAddress() | address= “Blk 215 Ang Mo Kio Street 22” | A Carpark object with fields corresponding to carpark A1. | A Carpark object with fields corresponding to carpark A1. |
| 3 | Pass a valid user ID into getUser() | uid = “AifwzQ2JiOVVhCMQz2Y4SYHFGGt2” | A UserAccount object with the following fields:   * email: devanshs001@e.ntu.edu.sg * favourites: an empty list * fullName: “Devansh Srinivasan” * phone: “83555026” | A UserAccount object with the following fields:   * email: devanshs001@e.ntu.edu.sg * favourites: an empty list * fullName: “Devansh Srinivasan” * phone: “83555026” |

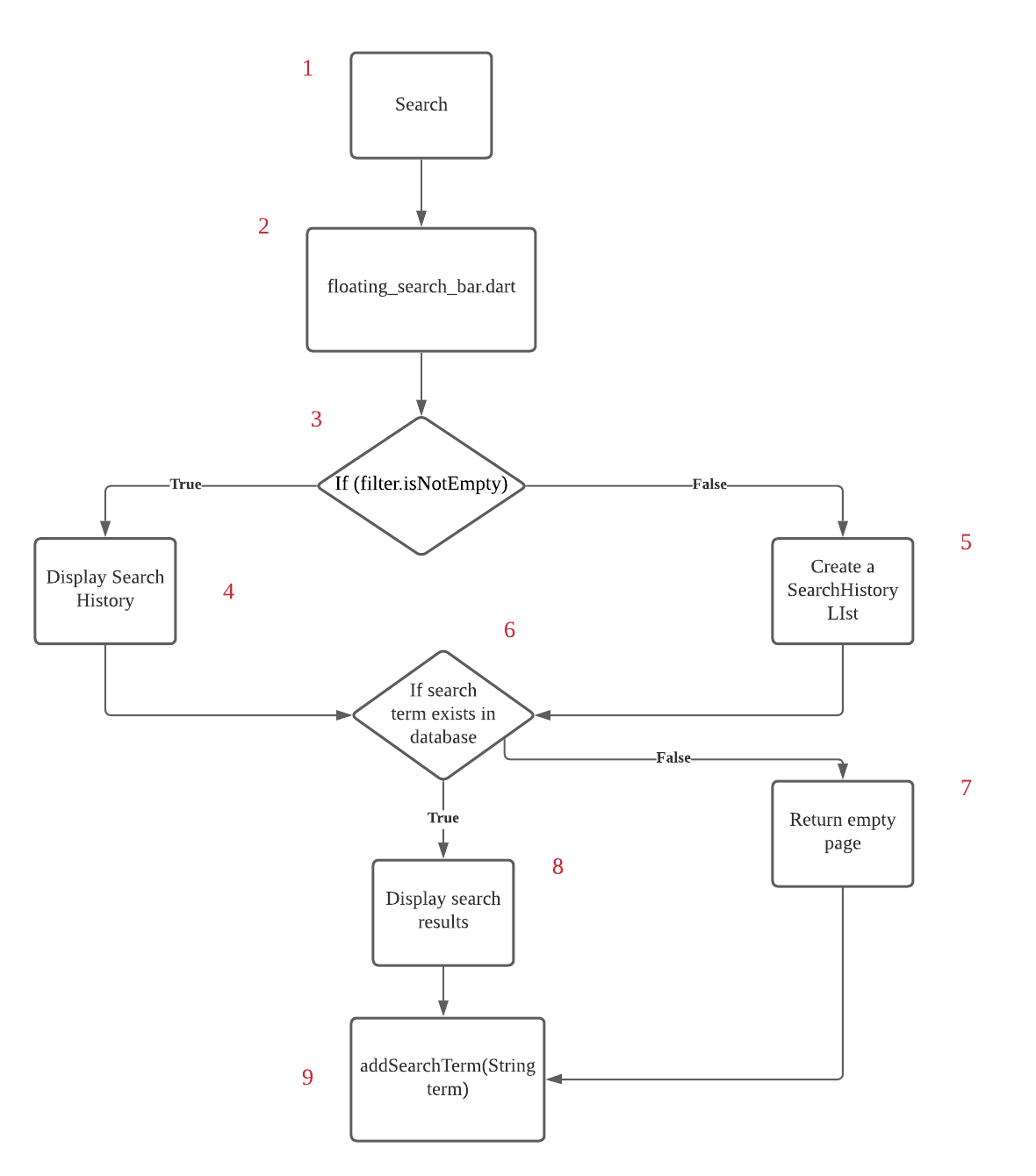
## 7.2. White Box Testing

1. **User Login**



| **Test ID** | **Basis Path** | **Test Input** | **Real Execution Paths** |
| --- | --- | --- | --- |
| 1 | 1,2,3 | **Email:**  annie@gmail.com  **Password:**  Testing123 | 1,2,3 |
| 2 | 1,2,4,5,3 | **Email:**  annie@gmail.com  **Password:**  Testing123 | 1,2,3 |
| 3 | 1,2,4,6,1 | **Email:** -  **Password:**  abc123 | 1,2,4,6,1 |
| 4 | 1,2,4,5,7,1 | **Email:** abd@hotmail.com  **Password:**  abc123 | 1,2,4,5,7,1 |

1. **Search**



| **Test ID** | **Basis Paths** | **Test Input** | **Real Execution Paths** |
| --- | --- | --- | --- |
| 1 | 1,2,3,5,6,8,9 | “Ang Mo Kio” | 1,2,3,5,6,8,9 |
| 2 | 1,2,3,4,6,8,9 | “Ang Mo Kio” | 1,2,3,4,6,8,9 |
| 3 | 1,2,3,4,6,7,9 | “ank” | 1,2,3,4,6,7,9 |
| 4 | 1,2,3,5,6,7,9 | “ank” | 1,2,3,5,6,7,9 |