Software Requirements Specification

for

Carpark Companion

Version 1.0 approved

Prepared by

Au Yew Rong Roydon (U2021424J)

Low Lee Hang (U2021604G)

Michelle Lam Su-Ann (U2021548K)

Chloe Heng (U2022247J)

Ooi Yi Hang (U2022327K)

Chua Gim Aik (U2022142B)

Rachel Chua (U2021383J)

OneOne35 (BCS1)

8th April 2022

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
All	17/01/22	First draft of SRS Added all diagrams and written documentation	1.0
Roydon Au Low Lee Hang	26/01/22	Update changes in use case descriptions	1.1
Chua Gim Aik Chloe Heng Ooi Yi Hang	30/04/22	Update changes in functional and non-functional requirements, as well as use case diagrams	1.2
Rachel Chua Michelle Lam	1/2/22	Update changes in sequence diagrams and dialog map	1.3
Roydon Au	14/2/22	Update design principles	1.4
All	8/4/22	Completed documentation	2.0

1. Introduction

1.1 Purpose

The purpose of this document is to provide a detailed description of the course of actions required during the design and development of *Carpark Companion* v1.0 – an Android mobile application. The main objective of this application is to help its users conveniently find vacant car-parks that are in close proximity to their target destination. This document will also provide an insight into the features of *Carpark Companion*, as well as its functional and non-functional requirements.

1.2 Document Conventions

This document features the use of font Times New Roman size 12 for body text. Each new heading uses Times New Roman of size 18 and is stylised to be bold. Each subheading uses Times New Roman size 14 and is stylised to be bold.

For numbering our functional requirements and detailed use case descriptions we have used a nested (indented) numbering system. For instance, a title with heading "1" will possess subheadings numbered as "1.1", "1.2", etc. This convention applies to all subheading levels as well.

For sub features in some of our system features, a bullet point is used to separate them.

1.3 Intended Audience and Reading Suggestions

This document is intended to be read by individuals who are involved in the development and distribution of the application including developers, project managers, software engineers, software testers, marketing staff, documentation writers, and other potential users who will be updating and supervising the *Carpark Companion* application. This document is also intended to be read by users who will be using the application.

The entirety of this SRS contains the following documentation and readers are recommended to follow the sequential order starting from the Introduction, consequently to the rest of the document until the appendix as follows:

1. Introduction (current position)
2. Overall description
3. External interface requirements
4. System features
5. Design considerations
6. Other non-functional requirements
7. Other requirements
8. System design
9. Appendix A: Glossary
10. Appendix B: Analysis models
11. Appendix C: Testing

For a clear understanding of the application and its features, functional and non-functional requirements, external interfaces, system features and supporting diagrams need to be considered. Use case diagrams, use case descriptions, class diagrams, sequential diagrams, dialog maps and data dictionary are provided for developed and testers reference.

Users who will be using the application are advised to read section 3 to preview the user interface and understand the system requirements for using this application, and then section 3 to fully understand the capabilities of the application, and finally section 4 to understand how their data is protected. Users who may have difficulties using the system can refer to section 2.6 to find a video demonstrating the use of the application.

1.4 Product Scope

Carpark Companion 1.0 is a mobile application developed to be used on Android mobile platforms. It is intended to be a parking assistance application to aid drivers in finding car parks with vacant parking spaces that are in close proximity to the user's target destination. This user-friendly mobile application also aims to provide users with a personalised experience, allowing for more convenient usage when they use the application.

1.5 References

1.5.1 Flutter SDK Documentation

The following documentation on Flutter SDK found on their website is used:
https://flutter.dev/

1.5.2 Government of Singapore Datasets and API

Carpark and carpark availability data is found on the Singapore government data portal:
https://data.gov.sg/

1.5.3 Land Transport Authority (LTA) Datamall API

Carpark availability data for the purpose of an alternative data source is provided by LTA datamall:
https://datamall.lta.gov.sg/content/dam/datamall/datasets/LTA_DataMall_API_User_Guide.pdf

2. Overall Description

2.1 Product Perspective

The *Carpark Companion* application is a self-contained mobile application. Despite existing applications in the market serving a similar purpose, we find that they do not fulfil all the drivers' parking needs. Hence, the main objective of this application is to make the process of finding parking spaces for all drivers more efficient and convenient. The idea for this application was designed to unify the strengths of Google Maps and government data from data.gov.sg. The former is a strong standalone API capable of providing visualisations onto a map, but it lacks sufficiently detailed information. Whereas, the government data lacks proper data visualisation but contains a plethora of useful information. Additionally, the design philosophy of the *Carpark Companion* application is simplicity - this means that the application must focus on fulfilling its main objective (carpark finding) and avoid distracting the user from its purpose with excessive features and effects. This can be observed in all the implemented features, each one solving a unique problem in carpark finding.

By combining the mentioned two APIs, and implementing additional features that users might need, our goal is to create a more enhanced and personalised application than existing solutions.

2.2 Product Functions

To serve the purpose above, the following features are included in the application:

1. **Search for Nearby Car Parks:** Users can search for nearby car parks in 2 ways, either (1) users can interact with the map interface by dragging to their intended destination or, (2) users can enter their intended destination in the search bar. The application will then display markers of car parks in close proximity to the users' intended destination on the map interface.

2. **Modify Display of Results with Filter Function:** Allow users to customise the resulting car parks they are searching for, by filtering according to the following options:
    a. Car park type
    b. Parking system type
    c. Free parking
    d. Short term parking
    e. Night parking

3. **Viewing Information of a Carpark:** Users can select a specific marker/carpark on the map interface. The application will display a half-detail page, containing the carpark's address, id number, and number of vacant lots at the moment. From there, users can further choose to view more details about the carpark, such as carpark type, free parking, etc.

4. **Lot Rememberer:** Allow users to make a note of their current parking space location. Users can enter details such as the carpark name, the lot number, and set a timer to calculate duration and parking fare spent at that location.

5. **Login or Sign in System:** Users have to be logged in to use the following features below (i.e. 6., and 7.). If user does not have an existing account yet, they can create one. Users will be prompted to enter a valid email and password. If either information entered is invalid, the user will be prompted to re-enter their details accordingly.

6. **Favourites:** Allow users to save and remember a list of car parks under the Favourites interface. This is only possible if they are logged in. Otherwise, they will be prompted to log in or create a new account before they can use this feature.

7. **Forum:** Users can create a comment to share their feedback or opinion of any carpark they've visited. Users can also delete their previous comments accordingly. This feature is only possible if they are logged in. Otherwise, they will be prompted to log in or create a new account before they can use this feature.

2.3 User Classes and Characteristics

2.3.1 Owners of Small to Medium-Sized Vehicles in Singapore

Carpark Companion mainly aims to target local drivers or small to medium-sized 4-wheeled vehicles in Singapore. Among those that drive such vehicles, our primary focus would be on those that wish to reduce the hassle and time wasted when searching for car parks manually.

The user interface is fairly straightforward, and hence many different types of users – ranging from novice users to advanced users – will be able to utilise the application without any prior experience. For a seamless experience on *Carpark Companion,* the user will only need to know the basic utilisation techniques of a smartphone mobile application.

2.4 Operating Environment (OE)

OE-1: The Carpark Companion Mobile Application will be compatible with smartphones running on **Android API 28** upwards.

OE-2: Carpark Companion must not have any data conflicts with other applications that make use of the Carpark API from data.gov.sg website, LTA's datamalland Google Maps public API. To achieve this, the Carpark Companion modelled data objects with respect to the data formats used by the aforementioned data services.

OE-3: Google Maps service is required to support the Carpark companion MapView and Navigate functionality.

OE-4: The target mobile device must have a minimum **internal storage of 2 GB**.

OE-5: Dataset from the data.gov.sg website will be parsed into Google Firebase. The *Carpark Companion* mobile application will make a request to retrieve data about nearby car parks from Firebase. The generation of nearby car park markers will make use of Google Maps. User login and authentication will be handled by Google Firebase Authentication. In the event of data retrieval problems (e.g. slow fetching, cloud provider severs connection) from the mentioned cloud database (due to the limitations of a free cloud solution), the application falls back on the direct API calls to the government data sources (i.e. data.gov).

OE-6: It goes without saying that the application is only functionable where there is internet connectivity or cellular data services.

2.5 Design and Implementation Constraints

Memory

Carpark Companion retrieves data about the destination during the user's input for destination in the first page from Google Maps, and suggests them in a drop down. The data about the destination is stored locally in the phone's internal storage.

Cache

The application must store the data locally each time a search is made. Once the application is closed, all data that was stored in the cache must be deleted. This allows for any real time changes there might be in the car-park details on the data.gov.sg information.

While this saves the user some memory space on their mobile phone, it costs some extra time in performance as overhead from retrieving data in each search is needed.

Development Toolkits and Technologies

Developers must use Android Studio IDE with Flutter SDK installed to develop the application in both Android OS.

Development and testing for the Android version of this mobile application may be carried out on any computers with any variations in OS (Windows, Mac OS or Linux).

In addition, POSTMAN, a third-party API testing platform will be used to test the API services for responsiveness, data extraction logic and data modelling on the dart codes.

Language Constraints and Programming Standards

Developers must develop the application using Flutter SDK and dart programming language.

Security Standards

Only developers and software architects are allowed full access to the source code. To access firebase console, login is required as well.

2.6 User Documentation

Carpark Companion is user oriented and simple to use. The different functions are simplified to only a few steps in each screen.

UD-1: No physical user manual is provided

UD-2: When a user first opens the application(just downloaded) the help tutorial will be invoked to guide users on the function of different buttons. Tutorials are provided in a dialog box with arrows pointing to the specific button.

UD-3: Users can find a live demonstration video of the application online.

2.7 Assumptions and Dependencies

Assumptions:

1. The user has an Android device.
2. The user's Android device has access to a stable internet connection such that it can access google map and retrieve real time carpark vacancy from the API.
3. User's phone location detection feature is enabled
4. The application's business requirements in terms of data protection and use of publicly available information are highly dependent on the current Singapore government policies as of April, 2022. Should the government change any of its policies with respect to data protection and use of available information - the system will have to be updated as necessary.
5. The user has an active email account
6. The user has Google Maps installed in their Android device.
7. The external APIs that the application is relying on are functioning.
8. Usage of application is restricted to Singapore only.

3. External Interface Requirements

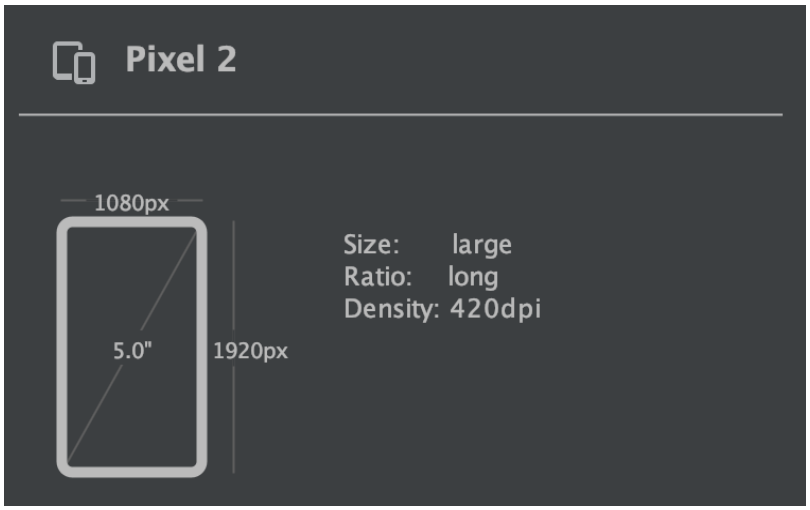
3.1 User Interfaces

Carpark Companion is a highly interactive application with a simple and consistent layout. The interactive features include buttons, commonly used icons, a search bar and dialog boxes. For ease of use every button and icons are clearly labelled. Besides that, dialog boxes also clearly show what type of information is required by users. This ensures that users are aware of the outcome of their actions when clicking on each of them.

The following screen layouts require user interaction to execute further main actions, which will be further elaborated in the following section of the document.

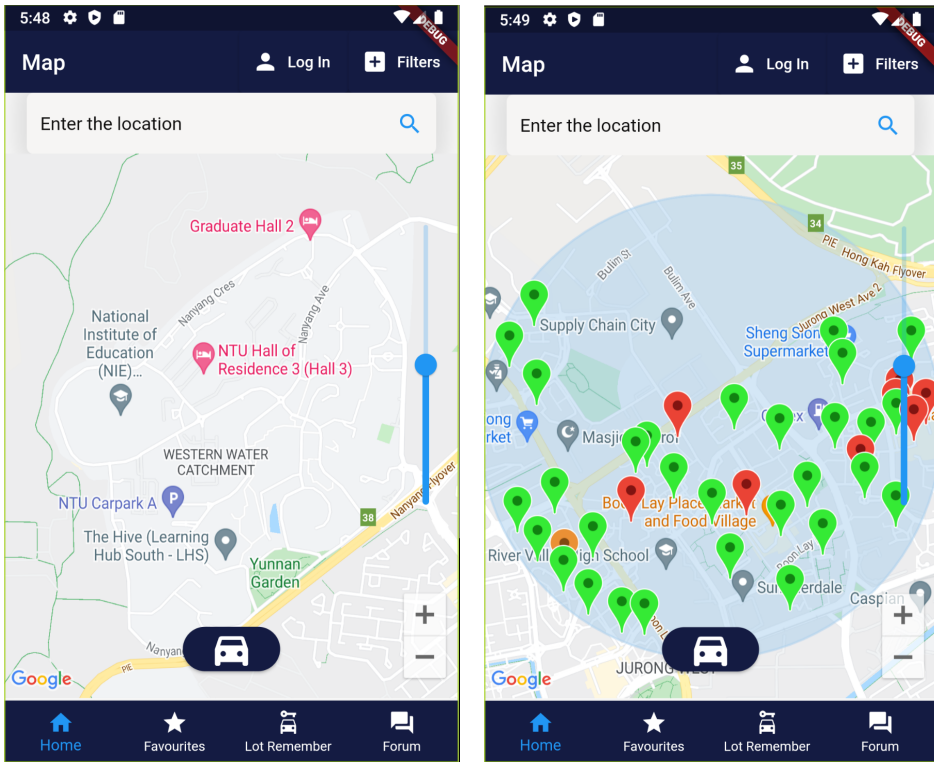
3.1.1 Style Guide

Carpark Companion will be primarily developed to be viewed in a portrait format. Car park Companion must adhere to a strict style guide to promote ease of use and consistency in branding. Car park Companion must adhere to the following style constraints:



The respective screen layouts designed are detailed below:

3.1.2 Screen 1 – Map Interface



The user can look for nearby car parks in 2 ways:

1. Dragging the map to their target destination
2. Using the search bar to key in their target destination

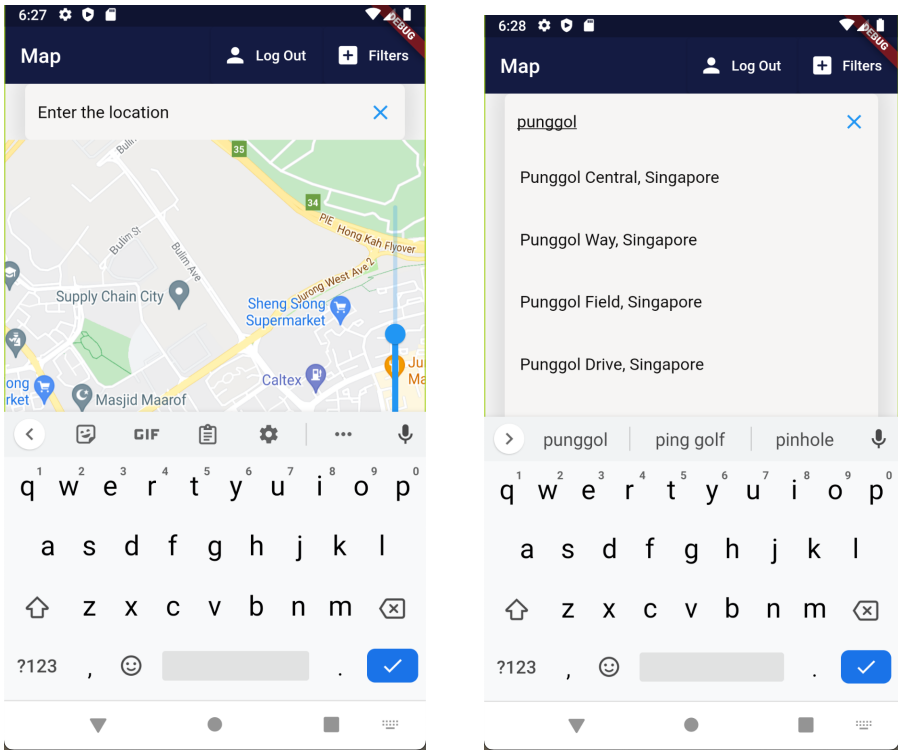
The former allows users to display nearby car parks without using the search bar, but by interacting with the map. Once they have dragged the map to their intended destination, users can click the ‘car’ button at the bottom of the screen. All nearby car parks will then be displayed to the user, in the form of location pins.

Resulting location pins displayed are colored accordingly, to indicate car-park vacancy.

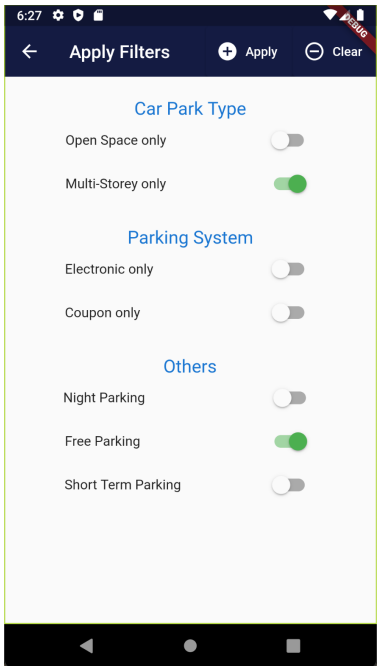
1. Green – High vacancy
2. Orange – Moderate vacancy
3. Red – Low vacancy

Users can adjust the search distance (a circular search radius) by adjusting the distance slider at the right side of the screen. Users can also adjust the zoom size of the map, with the ‘+’ and ‘–’ buttons at the bottom right of the screen.

3.1.3 Screen 2 – User Enters their intended Destination



Alternatively, using a simple and intuitive search bar, users can enter their specific intended destination. When the user keys in their intended destination, the application will offer suggestions in a dropdown list. The user can select an option and the map will be redirected to the intended destination selected. Similar to before in 3.1.2, the user can click on the 'car' button at the bottom of the screen and all nearby car parks will be displayed.



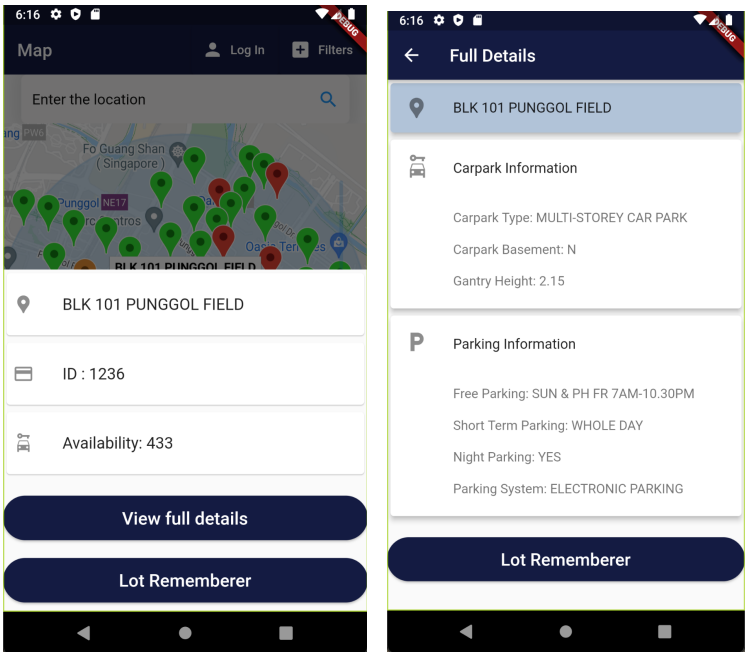
3.1.4 Screen 3 – Carpark Filters

Users can easily navigate to the selection of filters by clicking on the filter button at the top right corner of the 'Home' page. Depending on the user's preference, he/she can toggle the respective filter options off or on. Once users click on 'Apply', the filter is saved and future searches will only show carpark pins that match the specified filters.

To reset all filters to its default state, users can click on 'Clear' at the top right corner of the filters screen.

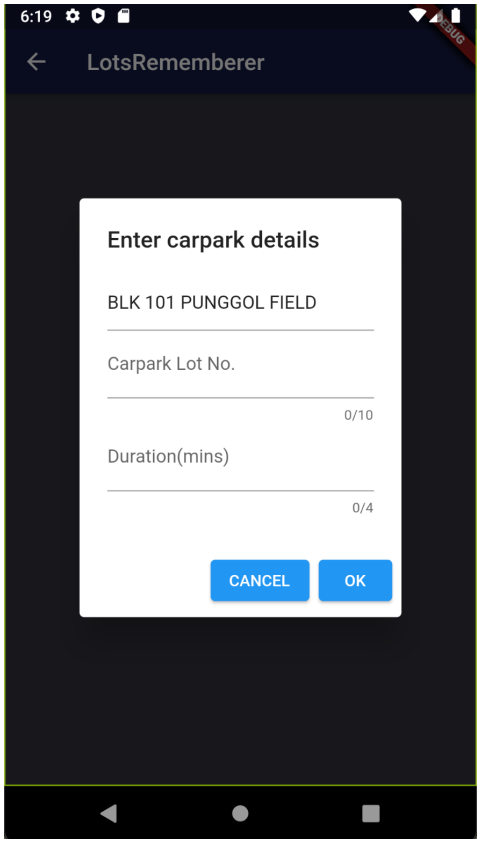
The various filters are as shown in the corresponding application screenshot. Filtering as the name of the filter suggests (i.e. if filtering by 'Night Parking' → Carpark results returned in future search must allow Night Parking).

3.1.5 Screen 4 & 5 – Carpark Information Interfaces (Half and Full Detail Interface)



The details of each carpark are displayed in two alternate interfaces.

The half-detail interface will appear when the user clicks on the location pin. The user can further navigate to the full detail interface by clicking on "View full details", whereby they can view other details about the carpark.

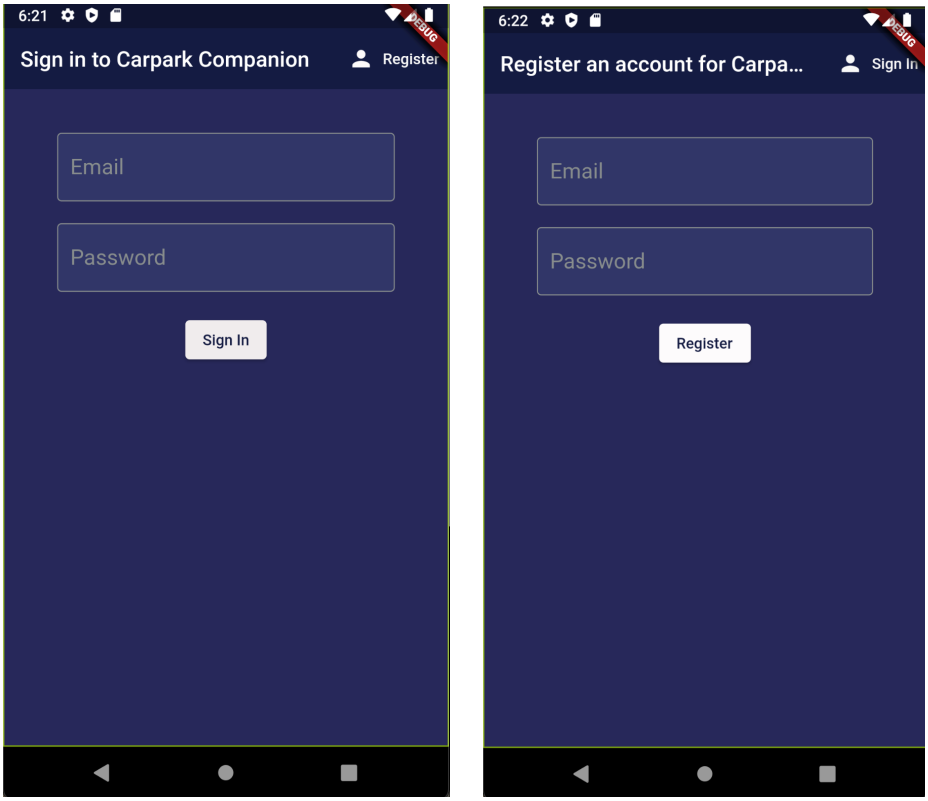


3.1.6 Screen 6 – Lot Rememberer Interface

Users can add the car park they have chosen into the 'Lot Rememberer' page by clicking on 'Lot Rememberer' button, from either the half detail or full detail page in 3.1.5. The name of the current carpark selected will be automatically filled in. Users can then enter other details, such as the carpark lot number, and the duration he/she wishes to park his/her car for.

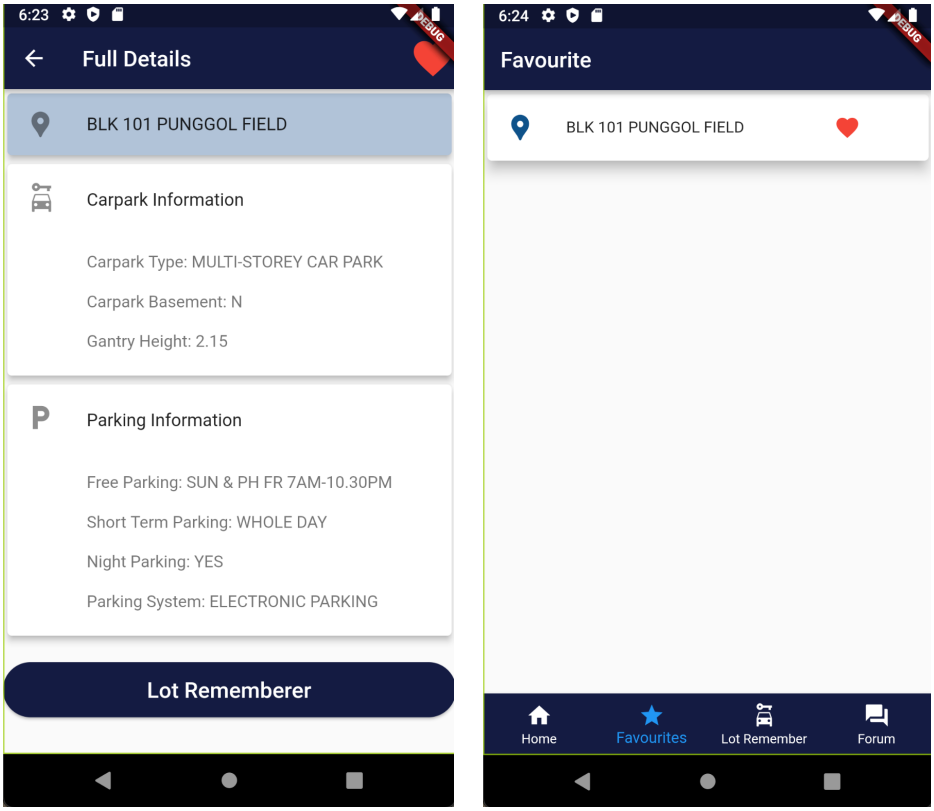
Users can also choose to manually enter the carpark name or edit the carpark name, if they wish.

3.1.7 Screen 7 – Login and Sign-Up Interface



Login page requires the user to enter a valid username and password combination. For new users, they can register an account with a valid email and a valid password.

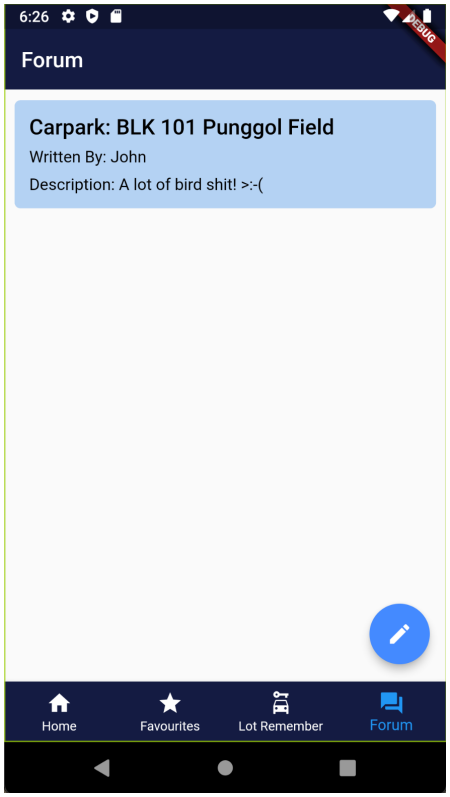
3.1.8 Screen 8 – Favourites Interface



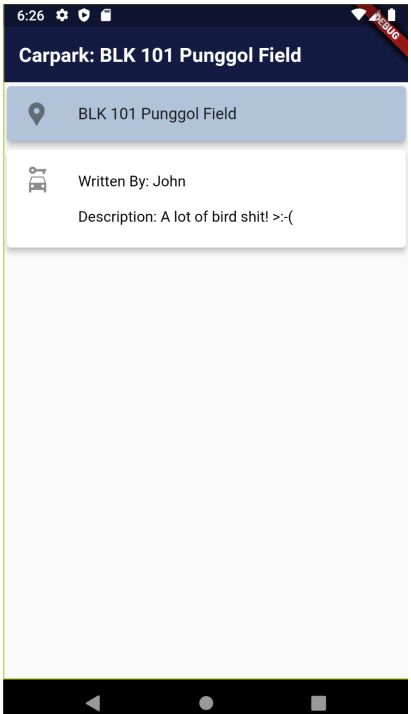
At the top right of the full detail interface, users can click on the 'heart' button to add the car park to their list of favourites. Their added car parks can be viewed in the favourites page.

To remove the car park from their list of favourites, users can simply click the now-lighted red 'heart' button.

3.1.9 Screen 9 – Forum Interface



Users can click on the 'pencil' button at the bottom right of the screen to create a new forum for a specific car park. The user will then be prompted to enter the relevant details. Once entered, the forum will be successfully created and displayed on the forum page.



If the user wants to view the full forum and its details, they can click on the forum directly. They will be redirected to the page with full detail of the forum.

3.2 Hardware Interfaces

To run the application, the user must use a mobile phone that operates on a minimum **Android API 28** version operating system.

The mobile phone must be connected to a WiFi network or cellular data.

The mobile phone should have enough storage space (**at least 2GB**) to download and store data the application is processing.

3.3 Software Interfaces

For Development:

OS: Windows 10 and Mac OS X are the main computer platforms used for development

Tools: Android SDK version 32.1.0-rc1, Flutter SDK 2.10.2, Android Studio (version 2021.1) Android Emulator and/or Android phone with Android 9.0++ pie with developer mode enabled.

Data format from external sources: CSV and JSON (from APIs)

Data sharing: Data sharing is performed between classes via public variables and public classes via import statements and getter functions For instance, some data objects were defined in 'main.dart' (the heart of our app) which were subsequently globally accessed by several other classes. This was intentionally performed to enable a global state - i.e. the entire application sees only one piece of critical truth. In other cases, classes would implement private variables that would then subsequently be accessed using class getter functions as per object-oriented design principles.

API 1: Google Maps Platform > Google Places API

- Places API: Integrate Google's Place details, search, and autocomplete into your apps.

  Data Input: Textual information with string or substring of user's destination sent from the Carpark Companion application to the web service by HTTP GET Request protocol. This is to get a list of matching destinations to suggest users via the Drop Down.

  Data returned: List of predicted matching destinations is returned to the application via either JSON or XML format. This will be suggested to users and one of them will be selected as an anchor point to search for car parks.

- Map SDK for Android: Bring the real world to your users with dynamic maps for the web and mobile.

  Data Input: Current location of the device is sent from the device to the web service by HTTP GET Request protocol.

Data returned: Dynamic map with markers, camera and view which add markers and control aspects of the camera including position, zoom level, and bearing.

API 2: Carpark Availability and HDB Carpark data API from Data.gov.sg

Data Input: API HTTP GET request detailing resource to be extracted and relevant query parameters is sent to the API service provider. 2 separate calls made: (1) for car park information (2) for car park availability

Data returned: carpark, carpark vacancy and capacity information for each carpark object, updated in real-time. Data format is in JSON. Note: Carpark information requires several fetches due to data being retrieved in pages of 100 car parks per fetch. Carpark vacancy data can be fully extracted in a single call.

API 3: Carpark Availability API from LTA

Data Input: API HTTP GET request detailing resource to be extracted, API KEY header information for authentication, and other relevant parameters is sent to the API service provider.

Data returned: carpark vacancy information for each car park object, updated in real-time. Data format is in JSON. Note: requires several fetches due to data being retrieved in pages of 500 car parks per fetch.

The software requires the installation of the following software dependencies mentioned in pubspec.yaml:

```
version: 1.0.0+1

environment:
 sdk: ">=2.15.1 <3.0.0"
dependencies:
 flutter:
   sdk: flutter
 firebase_database: ^9.0.6
 firebase_core: ^1.12.0
 firebase_auth: ^3.3.7
 syncfusion_flutter_sliders: ^20.1.47

 # The following adds the Cupertino Icons font to your application.
 # Use with the CupertinoIcons class for iOS style icons.
 cupertino_icons: ^1.0.2
 flutter_signin_button: ^2.0.0
 firebase_auth_platform_interface: ^6.1.11
 firebase_auth_web: ^3.3.7
 firebase_core_platform_interface: ^4.2.4
 provider: ^6.0.2
 geolocator: ^8.2.0
```

```
 #places_service: ^0.1.0
 geocode: ^1.0.1
 search_map_place_updated: ^0.0.4
 google_maps_flutter: ^2.1.2
 location: ^4.2.0
 flutter_countdown_timer: ^4.1.0
 simple_timer: ^2.0.0
 get: ^4.6.1
 #flutter_polylinr_points: ^0.2.6
 meta: ^1.7.0
 auto_size_text: ^3.0.0
 #Favourites button
 favorite_button: ^0.0.4
 get_storage: ^2.0.3


dev_dependencies:
 flutter_test:
   sdk: flutter
```

3.4 Communications Interfaces

Carpark Companion needs a steady wifi connection to ensure access to google map and APIs. Forum and favourite requires login in authentication to connect to the database and retrieve relevant data.

4. System Features

4. 1 Register

4.1.1 Description and Priority

First time users will be able to create an account after clicking on the register button. They will be directed to a page to enter their email and password that is required for account creation.

Priority: Medium

4.1.2 Stimulus/Response Sequences

Preconditions: User does not have a registered account for the application

Stimulus:

1. User clicks onto "Register" that is on the Home page to create an account.
2. The system will direct the user to the register page.
3. The system will prompt the user to enter a valid email and account password.
4. The user attempts to register by clicking on the "register" button.
5. The system will verify if the information entered is valid.
    - 5.1 Invalid email with incorrect format entered
        - 5.1.1 System will display an error message showing that invalid email is entered.
        - 5.1.2 System will request for a new valid email to be entered.
    - 5.2 Invalid password with password length of less than 6 characters
        - 5.2.1 System will display an error message showing that invalid password is entered.
        - 5.2.2 System will request for a new valid password of length more than or equal to 6 characters to be entered.
    - 5.3 Empty email and password field.
        - 5.3.1 System will display an error message "Required fields are not filled".
        - 5.3.2 System will request for a valid email and password to be entered.
6. The system creates a new account with the email and password entered and sends this data back to the Firebase.
7. The system then logs the user into the account and returns to the Home page.

Postconditions: User will have a registered account with the application.

4.1.3 Functional Requirements

REQ-1-1:

1. The system must allow the user to create an account.
    - 1.1. The system must request the user to enter email and password.

1.2 The system must verify that no field is left empty, and that all fields are valid.

1.3. The system must verify that the email provided is valid and unique which has not been used for registration before. This is done through checking with the Firebase that stores the existing users data.

1.4 The system must verify that the password is at least 6 characters long.

REQ-1-2:

1. The system must update the data of the newly created account into the Firebase.

REQ-1-3:

1. The system must redirect the user to the Home page upon successful registration.

REQ-1-4:

1. The system must be able to extend to Log In Feature 4.2 to log in to his/her account if he/she has already created an account before.

4. 2 Log In

4.2.1 Description and Priority

Users must log into his/her account to favourite, use the lot rememberer function and access the forum. The system will check if the user has an account registered with the application before logging the user in.

Priority: High

4.2.2 Stimulus/Response Sequences

Preconditions: User is not logged into the application.

Stimulus:

1. User clicks on the “Login” icon on the Home page.
2. The system directs the user to the login page.
3. User enters email and password into the fields of the login page.
4. User attempts to login by clicking on the login button.
5. The system will verify that there are no empty fields.
    5.1 Empty Field found
        5.1.1 System will display an error message “Required fields are not filled”.
        5.1.2 System will request for a valid email and password to be entered.
6. The system will verify if the email entered is a registered email with the application.
    6.1 Unregistered email and invalid password entered
        6.1.1 System will display an error message “Invalid email/password”.
        6.1.2 System will request for a valid registered email and password.
7. The system will verify if the password matches with the password of the registered email.
    7.1 Wrong password entered

7.1.1 System will display an error message “Incorrect password has been entered. Please re-enter password”.

7.1.2 System will request for the correct password.

8. On successful verification, the user is logged into the application and directed back to the Home page.

Postconditions: User is logged into the application

4.2.3 Functional Requirements

REQ-2-1:

1. The system must allow the user to enter his/her email and password.
2. The system must verify that no empty fields are left.

2.1 If any empty fields are found, the following message is displayed: “Required fields are not filled”.

REQ-2-2:

1. The system must verify that the entered email and password are valid and have an existing account associated with it by checking through the existing user data in the Firebase.

1.1. If the entered email and password do not have an existing account associated with it, the following message is displayed: “Invalid email/password”.

1.2 If the password entered is incorrect but the email is a registered email, the following message is displayed: “Invalid password has been entered. Please re-enter password”.

1.2. If the entered email and password are valid and have an existing account associated with them, direct the user to the Home page.

REQ-2-3:

1. The system must be able to extend to Register Feature 4.1 for users to register an account instead if they are first time users.

4.3 Navigation Bar

4.3.1 Description and Priority

The Navigation Bar allows the user to switch between the 4 main page features, Home, Favourites, Lot Remember and Forum. The Navigation Bar will always be displayed on the bottom of the screen so that users can navigate to the 4 main screens at any point in time.

Priority: High

4.3.2 Stimulus/Response Sequences

Preconditions: None

Stimulus:

1. User clicks on to the Home icon
    1.1 The system will direct the user to the Home page.
2. User taps on to the Favourites icon
    2.1 The system will direct the user to the favourites main page
3. User taps on to the Lot Rememberer icon
    3.1 The system will direct the user to the Lot Rememberer main page
4. User taps on to the Forum icon
    4.1 The system will direct the user to the Forum main page.

Postconditions: The user will be able to access the main page of the 4 features: Favourites, Lot Rememberer, Forum and Home from the current main page he/she is at.

4.3.3 Functional Requirements

REQ-3-1:

1. The system must show the navigation bar at all times at the bottom of the main pages of each feature.

REQ-3-2:

1. The navigation bar must consist of the below icons at all times:
    1.1 Home Icon
    1.2 Favourites Icon
    1.3 Lot Rememberer Icon
    1.4 Forum Icon

REQ-3-3:

1. The system must direct the user to the main page of each feature depending on the icon chosen by the user.
    1.1 Home Icon chosen
        1.1.1 The system must direct the user to the Home page.
    1.2 Favourites Icon chosen
        1.2.1 The system must direct the user to the Favourites main page.
    1.3 Lot Rememberer Icon chosen
        1.3.1 The system must direct the user to the Lot Rememberer main page.
    1.4 Forum Icon chosen
        1.4.1 The system must direct the user to the Forum main page.

REQ-3-4:

1. The system must allow the user to extend to the below features after directing the user to the main page that has been selected.
    - 1.1 Home Icon chosen
        - 1.1.1 Feature 4.4 Google map
        - 1.1.2 Feature 4.5 Distance Slider with Search Car Button
        - 1.1.3 Feature 4.7 Search Bar
    - 1.2 Favourites Icon chosen
        - 1.2.1 Feature 4.9 Favourites
    - 1.3 Lot Rememberer Icon chosen
        - 1.3.1 Feature 4.10 Lot Rememberer
    - 1.4 Forum Icon chosen
        - 1.4.1 Feature 4.11 Forum

4.4 Google map

4.4.1 Description and Priority

The integrated Google maps will be displayed and allows the user to interact with it. This feature is integrated into the Home page of the application, which is the default landing map page that users will see when they first open the application.

Priority: High

4.4.2 Stimulus/Response Sequences

Preconditions: None

Stimulus:

1. The user is able to scroll to their desired location in the map by moving their fingers while pressing on the map.
2. The system will display the Google Map location which the user has scrolled to.
3. A blue circle indicating an initial radius of 1km will be shown on the Google Map, with the location the user has scrolled to being the centre of reference.
4. The user presses on the "-" icon to zoom out.
    - 4.1 The system will display the Google Map with a zoomed out view according to how much the user pressed.
5. The user presses on the "+" icon to zoom in.
    - 5.1 The system will display the Google Map with a zoomed in view according to how much the user pressed.

Postconditions: The user is able to interact with the Google Map to scroll to their desired location on the map and adjust the zoom level on the map.

4.4.3 Functional Requirements

REQ-4-1:

1. Users must be allowed to adjust their view of this map by using the following buttons or gestures:
    1.1 Pressing "-" to zoom out.
    1.2 Pressing "+" to zoom in.
    1.3 Scrolling across the screen by moving their fingers while keeping it pressed on the map.

REQ-4-2:

1. A blue circle around the centre location of the map will be displayed. This circle will represent an initial 1km radius with the location the user has scrolled to being the centre of reference.

4.5 Distance Slider with Search Car

4.5.1 Description and Priority

The distance slider allows the user to increase/decrease the search radius (metres) of carpark objects to be displayed on the map. The Search Car button must be pressed each time the distance slider is changed in order to see the change in display. This feature is integrated into the Home page of the application, which is the default landing map page that users will see when they first open the application. This feature is also used in conjunction with Display Carpark markers Feature 4.6 to view the car parks on the map.

Priority: Medium

4.5.2 Stimulus/Response Sequences

Preconditions: The Search Car button must be pressed in order for car parks markers to be displayed on the Google map.

Stimulus:

1. The user scrolls the circle on the distance slider scale to adjust to the desired radius in which he/she wants to view the car parks at.
2. The user presses the Search Car button on the bottom centre of the display.
3. The map will then refresh and display the car park markers within the radius set by the user.

Postconditions: The user will be able to view car parks in the vicinity according to the

distance radius they have chosen.

4.5.3 Functional Requirements

REQ-5-1:

1. The user must be able to adjust the radius of the blue circle by using the distance slider to adjust it to any radius the user wants.
    1.1 The maximum radius on the scale that will be allowed is 1500 metres and the minimum radius that will be allowed is the default 500 metres.
2. The system is required to ensure that the distance scale on the slider is kept at the same distance set by the user unless altered by the user.

REQ-5-2:

1. The system must be able to extend to Display Carpark Markers Feature 4.6 in order to view the car park markers.

REQ-5-3:

1. The system must be able to extend to the Google map Feature 4.4 in order to view the car park markers on the Google map.

REQ-5-4:

1. The user must be able to press on the Search Car button.
    1.1 The user must be able to see the markers each representing one car park.
    1.2 The user must see a difference in car park markers shown in the display if the distance slider has been altered.
    1.3 The user must be able to see a blue circle indicating the radius chosen using the distance slider and all car park markers must be within this circle.
    1.4 The user must be able to see a change in the radius of the blue circle if the distance slider has been altered and all new car parks to be displayed must be within this circle.
    1.5 The system must be able to call the Firebase API to retrieve the details of the car parks that need to be displayed.

4.6 Display Car Park Markers and Google Directions

4.6.1 Description and Priority

The car park markers will be displayed on the Google map (Feature 4.4), where each car park on the map will be represented by 1 marker. Each marker will have 3 different possible colours at any one time, corresponding to the number of vacant lots at the car park currently. Upon clicking on a car park marker, a blue Google Directions button will appear on the display as well, this will allow the user to get the directions to the chosen car park. These features are integrated into the Home page of the application, which is the default landing map page that users will see when they first open the application.

Priority: High

4.6.2 Stimulus/Response Sequences

Preconditions: The user must click the Search Car button.

Stimulus:

1. User clicks on the Search Car button.
2. The system will call the Firebase API to retrieve the relevant car park details.
3. The system will display the car parks with the corresponding coloured markers at the location the user scrolled to or the entered location from the Search Bar feature in 4.7.
4. All car park markers displayed will be within the blue circle radius mentioned in Google map Feature 4.4.
5. User clicks on a certain car park marker.
6. The system will display the car park name instead of the original marker.
7. The system will display a blue Google Directions button.
    - 7.1 If the user has Google Directions application installed in their device, upon clicking the button, the system will direct the user to this external application, where directions to the location of the chosen car park will be given.
    - 7.2 If the user does not have Google Directions application, the system will show an error that no external application compatible is found and return to the Home page.
8. User clicks onto the name of the car park.
9. The system will direct the user to a Half details page of that car park. This half detail page will consist of the following information:
    - 9.1 Name of Car park
    - 9.2 ID of car park
    - 9.3 Number of vacant lots at the car park currently

Postconditions: The user will be able to view car parks that are represented by car park markers.

4.6.3 Functional Requirements

REQ-6-1:

1. User must be able to see a carpark marker representing each carpark
    - 1.1 Car park markers will have 3 possible colours at any one time depending on the number of vacant lots currently. The possible colours are:

1.1.1 Green: High availability
1.1.2 Yellow: Moderate availability
1.1.3 Red: Low availability

REQ-6-2:

1. User must be able to click on a car park marker to display the half details page of that particular carpark which includes:
    1.1 Name of car park
    1.2 Address of car park
    1.3 Number of vacancies
2. After the system directs the user to the half details page, the user must be able to extend to the full details page by clicking on the "View full details" button.
3. User must be able to click on "View full details" button to display the full details page of that car park which includes:
    3.1 Name of car park
    3.2 Address of car park
    3.3 Carpark type
    3.4 Car park Basement availability
    3.5 Gantry Height
    3.6 Free parking availability
    3.7 Night parking availability
    3.8 Short-term parking availability
    3.9 Parking system of Car park

REQ-6-3:

1. Car Park markers displayed must be able to be filtered by its details according to the user's input. This extends to Apply/remove filters Feature 4.8.

REQ-6-4:

1. User must be able to access other functions from the details pages
    1.1 If the user is logged in, a favourite button will appear on the full details page which allows the user to add or remove that carpark to his/her favourites list. The user can then view this carpark under Favourites. This extends to Favourites Feature 4.9.
    1.2 A Lot Remember button will appear on both the half detail page and full detail page. This allows the user to apply the Lot Rememberer function on the chosen car park. The user can then view this lot saved under Lot Rememberer. This extends to Lot Rememberer Feature 4.10.

REQ-6-5:

1. The system must be able to extend to the Distance Slider with Search car button Feature 4.5 if a different radius is selected by the user on the Distance Slider and the car park markers to be displayed will be different.

REQ-6-6:

1. The system must be able to extend to the Google map Feature 4.4 in order for the car park markers to be displayed on the map.

REQ-6-7:

1. The system must be able to direct the user to the Google Directions external application if the user has this application in his/her device.
2. If the user does not have Google Directions in his/her device, the system must direct the user back to the Home page.

4.7 Search Bar

4.7.1 Description and Priority

User can use the search bar to enter the location they would like to go to. The user will be directed to the Google map view of the entered location. This feature is integrated into the Home page of the application, which is the default landing map page that users will see when they first open the application.

Priority: High

4.7.2 Stimulus/Response Sequences

Preconditions: None

Stimulus:

1. User clicks on the search bar and enters a location.
2. The system will show a dropdown list of locations matching what the user has entered.
3. The user clicks on to any one of the results from the dropdown list of locations shown.
4. The system will call the Google map API to retrieve the relevant details of the location selected to update Google map view.
5. The system will move the map to the location entered.
6. The system will display the Google map view of the entered location.

Postconditions: User will be able to see the Google map view of the entered location.

4.7.3 Functional Requirements

REQ-6-1:

1. The system must be able to extend to Google map Feature 4.4 in order for the Google map view to be updated and displayed.

REQ-6-2:

1. The system must allow the user to enter their desired location in the Search Bar.
2. The destination must be typed out by the user.
    2.1 As the user types in the destination, the application must show a dropdown list of locations matching what the user has entered.
        2.1.1 The location details must be received from Google map API.
    2.2 After the user has selected a location from the dropdown, the application must bring the user to the entered location shown on Google map.

4.8 Apply/Remove filters

4.8.1 Description and Priority

The user will be able to apply filters based on specific details of the car parks he/she is looking for. The map will only display the car park markers that correspond to the user's chosen filters. This feature is integrated into the Home page of the application, which is the default landing map page that users will see when they first open the application.

- Applying filters (Sub-Feature 1):
    - The user is able to filter car parks by toggling the switches corresponding to each filter they want to select.
    - The car parks displayed on the Home page will only consist of the car parks that meet the filtering conditions chosen by the user and radius as mentioned in Distance Slider with Search Car button Feature 4.5.
- Remove all filters (Sub-Feature 2):
    - The user is able to remove all previously chosen filters under Sub-Feature 1.
    - The car parks displayed on the Home page will consist of all unfiltered car parks in the vicinity of the chosen location and radius previously selected by the user.

Priority: Medium

4.8.2 Stimulus/Response Sequences

Preconditions: User must click on the "Apply Filters" button on the Home page.

Stimulus:
Sub Feature 1 (Applying filters)

1. User clicks on the "Apply Filters" button.
2. The system will direct the user to the filter page which consists of the entire list of available filters.
    2.1 Each filter will have a toggle switch next to it, with its default colour being no colour, indicating that the filter is off.
3. User clicks on the toggle switch.
4. The system will display a green colour on the toggle switch indicating that the chosen filter is on.
5. User clicks on to the "Apply" button on the filter page after selecting all the required filters.
6. The system will direct the user back to the Home page.
7. The user adjusts the distance slider to what he/she wants and presses the Search Car button.
8. The system will display car parks according to the filters chosen. The car parks will be represented by car park markers and displayed within the blue circle indicating the radius selected previously as mentioned in Display Carpark markers and Google Directions Feature 4.6

Sub Feature 2 (Remove all filters)

1. User clicks on the "Apply filters" button again.
2. The system will redirect the user back to the filter page. With the previously chosen filters toggles still remaining in green colour and the not selected ones in its default colour.
3. The user clicks on the "Clear" button.
4. The system will clear all the filters previously chosen by the user and direct the user back to the Home page.
5. The user adjusts the distance slider to what he/she wants and presses the Search Car button.
6. The system will display all the unfiltered car parks near the chosen location.
7. The system will also toggle off all filters previously selected and all toggles will be in their default mode in the filter page.

Postconditions: The car park markers depending on the filters the user has chosen or removed will be displayed on the map in the Home page.

4.8.3 Functional Requirements

REQ-8-1:

1. The system will extend to the Display Carpark markers and Google Directions Feature 4.6 and display the car parks in the format as mentioned in this feature.

REQ-8-2:

1. The system will extend to the Distance Slider with Search Car button Feature 4.5 for the user to adjust the distance radius and view the car park markers within the selected radius on the map.

REQ-8-3:

1. The system must allow the user to apply filters by toggling on the switches of the given filters below.
    - 1.1 The filters available includes:
        - 1.1.1 Carpark Type
            - 1.1.1.1 Open Surface
            - 1.1.1.2 Multi-storey
        - 1.1.2 Parking System Type
            - 1.1.2.1 Electronic
            - 1.1.2.2 Coupon
        - 1.1.3 Free Parking
        - 1.1.4 Night Parking
        - 1.1.5 Short-term Parking
    - 1.2 The system will only display a map showing the filtered car park markers based on the filters selected by the user.
    - 1.3 The system must allow the user to change the filters applied whenever required and the corresponding car park markers will be filtered again.
    - 1.4 The system must ensure that selected filters will have its toggle remain as green as the user alternates between the map page and the filter page.

REQ-8-3:

1. The system must allow the user to clear all the filters when necessary.
    - 1.1 Any applied filters must be removed and the toggle will return back to its default mode of no colour.
    - 1.2 All unfiltered car park markers near the chosen location of the user must be displayed on the map.

4.9 Favourites

4.9.1 Description and Priority

The user will be able to add the selected carpark to their favourites list. This feature will only be enabled if the user has a registered account and is logged in to this said account.

- Viewing list of past favorited car parks (Sub-Feature 1):
  - The user should be able to view the full list of car parks that they have favorited before provided that they are logged in to a registered account.
  - The list of car parks will be retrieved from the FireBase based on the account ID which is identified using the user's email.
- Add a new favourite car park (Sub-Feature 2):
  - The application will allow users to add the current car park they have selected to the current list of favourites they have.
  - The updated list of favourites will be stored into the FireBase allowing the user to access these favorited car parks again after the user has logged out.
- Remove an existing favourite car park (Sub-Feature 3):
  - The application will allow users to remove a selected car park from its current list of favourite car parks.
  - The updated list will be stored in FireBase as well.

Priority: Medium

4.9.2 Stimulus/Responses Sequences

Preconditions: User must have a registered account and is logged in to the account.

Stimulus:

- Sub-Feature 1 (View List of Favourite car parks)

1. The user will click on the Favourites icon on the Navigation Bar to view the list of car parks that have been favorited before.
2. The system will retrieve the list of favourite car parks from Firebase using the user's account ID.
3. The system will display the list of past favourite car parks.

   2.1 If the user is not logged into a registered account, an empty list with no car parks will be shown and the login button will be present instead.

   2.2 If the user is logged into a registered account, a list of previously favorited car parks will be shown. With the oldest car park at the top and latest car park at the bottom of the list.

- Sub-Feature 2 (Add a new favourite car park)

1. The user will click on the car park marker of the new car park to be added as favourite.
2. The system will display the name of the selected car park.
3. The user will click on the name of the car park.
4. The system will direct the user to the half details page of the car park.
5. The user clicks on the "View full details" button.
6. The user will be directed to the full details page of the selected car park.
7. The user clicks on the heart icon on the top right to add the car park to favourites. The heart icon before clicking is in its default mode with no colour.
8. The system will update the list of favourite car parks into Firebase.
9. The system will direct the user to the favourites page displaying the new updated list of favourite car parks.

- Sub-Feature 3 (Remove an existing favourite car park)
    - Flow 1

1. The user will click on the Favourites icon on the Navigation Bar.
2. The system will retrieve the list of past favourite car parks based on the user's account ID from the Firebase.
3. The system will display the list of past favourite car parks.
4. The user clicks on the heart shaped icon on the car park to be removed from the current list of favourites.
5. The system will remove the car park from the list of favourites and update the list into Firebase.
6. The system will direct the user to the Home page.
    - Flow 2

1. The user will click on the car park marker of the car park to be removed from the list of favourites.
2. The system will direct the user to the half detail page of the car park.
3. The user clicks on the "View full details" button.
4. The user will be directed to the full details page of the selected car park.
5. The user clicks on the heart icon that is currently red in colour on the top right to remove the car park from current favourites.
6. The system will remove this car park from the current list of favourites and update the list into Firebase.
7. The system will direct the user back to the Home page.

Postconditions: The user will have an updated list of favorited car parks and will be able to view this list in the favourites page.

4.9.3 Functional Requirements

REQ-9-1

1. The user must be logged in to be able to view his/her favourites list.
2. The system must be able to extend to Log in Feature 4.2 for the user to log in to his/her account and access the list of favourites if he/she is not logged in.

REQ-9-2:

1. The system must be able to save the updated favourites list to the Firebase when the user does the following:
    - 1.1 Updates the list by adding or removing
    - 1.2 Logs out of account
    - 1.3 Shuts down the application

REQ-9-3:

1. The user must not be able to access or view other users' favourites list.

REQ-9-4:

1. The system must be able to extend to Navigation Bar Feature 4.3 for the user to access the favourites page from any main page he/she is currently at.
2. The user must be able to view the car park full details by clicking on the car park under the user's favourites list.

REQ-9-5:

1. If the user is logged in, the user must be able to view his/her car park favourites list in the favourites page.
2. The user must be able to add or remove a carpark from his/her current favourites list. The user will do this by performing either actions listed below:
    - 2.1 Clicking on the heart button in the chosen car park's full details page.
    - 2.2 Clicking on the heart button of the chosen car park in the favourites page that contains the entire list of favourited car parks.
3. The system must ensure that the above actions to add or remove a carpark from his/her current favourites list will set the heart icon to its default mode of no colour when removing or clicked mode where the heart icon will be red in colour when adding.

4.10 Lot Rememberer

4.10.1 Description and Priority

The user will be able to record the carpark and carpark lot information at which he/she has parked the vehicle. A timer will also be shown based on the user's input and will alert the user when the timer is up.

Priority: Medium

4.10.2 Stimulus/Responses Sequences

Precondition: The user is currently at the half detail or full detail page of a certain car park.

Stimulus:

1. The user clicks on the "Lot Rememberer" icon either from the half detail or full details page of the car park.
2. The system will direct the user to the Lot Rememberer main page.
3. The user clicks on to the "Lot rememberer" button in the centre of the page.
4. The system will retrieve the car park name and address from Firebase.
5. The system will display a pop up message with the car park name and address automatically filled in by the system.
6. The user enters in the car park lot and duration that he/she will park the vehicle for in the pop up message and clicks on save.
7. Alternatively, the user can deleted the pre-fetched address and enter the desired carpark location
8. The system will save the Lot Rememberer information on local cache.
9. The system will direct the user back to the Lot Rememberer main page where it will display the car park name, lot number and estimated duration left at the car park based on the timer set.

Postcondition: The user is able to save the car park details for the car park lot he/she has parked at and a timer is activated to remind the user that the entered duration has ended when the time is up.

4.10.3 Functional Requirements

REQ-10-1:

1. The system must automatically fill in the car park name and address into the pop up message if the Lot Rememberer button was clicked either from the full detail or half detail page of the chosen car park.
2. The user must be able to enter the carpark lot in the pop up message.
3. The user must be able to enter an estimated duration in the pop up message to set the timer.
4. The system must be able to save the Lot Rememberer information in local cache.
5. The system must be able to display the Lot Remember information on the Lot Remember main page after the user clicks save on the pop up message.

5.1 The car park name and address will be displayed.
5.2 The timer will be displayed with a countdown on the time remaining.
5.3 The application will notify the user when the timer is up.

REQ-10-2:

1. The user must only be able to remember one lot at a time.
   1.1 Only a maximum of 1 entry of Lot Remember information will be displayed on the Lot Rememberer main page at any point in time.
   1.2 When the timer is up it must prompt the user to remove that current lot from the page and allow the user to add another lot.

REQ-10-3:

1. The system must be able to extend to the Navigation Bar Feature 4.3 for the user to access the Lot Rememberer main page.

4.11 Forum

4.11.1 Description and Priority

A forum is created for users to publish a post regarding a particular car park. This allows users to inform other users on any important information such as car park closure, accidents etc.

- Viewing all posts in the forum (Sub-Feature 1)
  - The user will be able to view all the posts of various car parks by clicking on the Forum button on the Navigation Bar.
- Publishing a post in the forum (Sub-Feature 2)
  - The user will be able to make his/her own post.
  - The user will be asked to input the carpark's name and his/her username and the description in the post.
- Deleting a post in the forum (Sub-Feature 3)
  - The user will be able to delete his/her own post in the forum.

Priority: Low

4.11.2 Stimulus/Responses Sequences

Precondition: User must have a registered account and is logged into it.

Stimulus:

- Sub-Feature 1 (Viewing all posts in the forum)

1. The user will click on the Forum Icon on the Navigation Bar.
2. The system will retrieve all the past posts from the Firebase.

2.1 If the user is not logged in, the system will not retrieve any information.

3. The system will display all the posts made before.
    3.1 If no posts were made before, the system will display “No entries!”.
    3.2 If the user is not logged in, no posts will be displayed.
        3.2.1 The login button will be displayed instead.

- Sub-Feature 2 (Publishing a post in the forum)

1. The user will click on the Forum Icon on the Navigation Bar.
2. The user will click on the Write Icon on the bottom right of the display.
    2.1 If the user is not logged in, the system will not show the Write Icon.
        2.1.1 The login button will be displayed.
3. The system will direct the user to the write post page.
4. The user must enter the car park name, user name and description he/she wants to write regarding the car park.
5. The user clicks on the “Save” button.
6. The system will update the new post into the Firebase.
7. The system retrieves all posts from Firebase.
8. The system will display all the posts including the newly added post by the user.

- Sub-Feature 3 (Deleting a post in the forum)

1. The user will click on the Forum Icon on the Navigation Bar.
2. All posts will be retrieved from Firebase and displayed on the Forum main page.
    2.1 If the user is not logged in, the system will not retrieve and display any information.
        2.2.1 The login button will be displayed.
3. User will slide left on the post he/she wants to delete.
4. The system will update the deleted post into the Firebase.
5. The system will display all the posts excluding the post that was deleted by the user.

Post-conditions: The user will be able to view all posts regarding different car parks in the forum. They will also be able to add their own posts regarding a specific car park as well as delete their own previous posts.

4.11.3 Functional Requirements

REQ-11-1:

1. The system must allow the user to make a post.
2. The system must direct the user to the write post page.
3. The system must allow the user to click on the “Save” button to successfully create a post and publish it.
4. The user can make posts following the below rules:

4.1 Post description cannot exceed 1500 characters.

4.2 The user can make as many posts as they want to.

REQ-11-2:

1. The user must be able to delete their own post.
    1.1 Users must not be able to delete other user's posts.

REQ-11-3:

1. The user must be able to see all the posts regarding the different car parks in Singapore.
    1.1 All posts will be saved to and retrieved from FireBase accordingly.

REQ-11-4:

1. The system must be able to extend to the Navigation Bar Feature 4.3 for the user to access the Forum main page.
2. The system must be able to extend to the Log In Feature 4.2 for the user to log in to his/her account and access his/her posts.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

1. Search results shall not take longer than **5 seconds** to load
2. Logging in shall not take longer than **2 seconds** for the system to process
3. Switching between and entering views (i.e. List and Map View) shall not take longer than **5 seconds.**
4. Entering a different screen (excluding those mentioned in item **3**) shall not take longer than **2 seconds.**
5. Car park objects (i.e. pins for map view OR records for list view) shall load/refresh upon user input (e.g. slider adjustment, applying filter, search) within **5 seconds.**
6. The application shall not take longer than **2 seconds** to load out-of-screen content when the user performs scrolling (i.e. moving the map, scrolling detail page).
7. The application must be able to handle loading up to **50 car park pins** on the map in map view at all times.
8. The application shall be able to refresh and update the number of lot vacancies of the car park objects every **1 minute.**
9. While the application is loading data, the user shall still be able to navigate the application. (i.e. screen doesn't freeze and grey out in anticipation of data loading).

5.2 Safety Requirements

1. Users are recommended to use the application only when they are in stationary mode if they are driving a vehicle.
2. Users are recommended to refrain from revealing any personal information in the forum page as Carpark Companion is not responsible for any loss incurred due to the usage of personal information by other users.

5.3 Security Requirements

1. The application shall not store/share data to third parties regarding the location of the user and car-parks used.
2. The application shall not share the user's history of the app usage to third party companies.
3. The application must not return any form of user data (e.g. live location) when answering any fetch request.

5.4 Software Quality Attributes

1. Reliability

Carpark Companion will provide accurate and correct results for queries requested by the user. The white and blackbox testing for the application in Appendix C has shown that there is a low error rate.

2. Reusability and Extensibility
   Carpark Companion's current implementation is done in such a way that it allows for easy extension of the application to include a wider range of functionalities in the future. A few possible future functionalities include:
   - Filtering based on Vehicle type such as Electric Car
   - Automatic online payment of parking fare

   The MVC architecture is also used, where the UI is separated from the core data model. Carpark Companion currently makes use of such architecture, with the model component that is responsible for retrieving and updating data is done in a separate file i.e carparkAPIinit.dart.

   The View and Controller components are separated as well, where the View component which is in charge of generating the UI is under a subcategory of files known as screens i.e landingMap.dart and the Controller component which acts as the intermediary between the Views and the model, enabling the connection between the 2 components is also done in another separate file i.e APIs.dart. This makes it easy for reusability as the data model, business logic are reusable across the different UIs.

   If the project was chosen for further extension, where changes and addition of new UIs are required, the MVC architecture as described makes it easy to do so. This will allow developers to redesign the UI without being greatly concerned about significantly affecting the other functionalities of the application.

3. Availability
   Carpark Companion ensures its availability to its users at all times. It will function as intended so long as a stable internet connection is present, and the user's android device is able to meet the hardware and software requirements as mentioned in 3.2 and 3.3. The only exception to this rule is when the application requires backend adjusting and it affects the data fetching backend processes - in such an event, Carpark Companion will be temporarily unavailable.

4. Maintainability
   Since the system uses an MVC architecture to decouple the UI from the data model and controller logic, it makes it easy to maintain the system and fix any bugs that occur since they will be isolated in their own specific components.

5.5 Business Rules

1. Users do not have to register for an account to use the application but they will only be able to access basic system features provided by the application. They will not be allowed to use the personalised features listed below:
    - 4.9 Favourites
    - 4.11 Forum

2. Users who have registered for an account and are logged in during the usage of the application will have access to all features available in the application.

5.6 Further Improvements

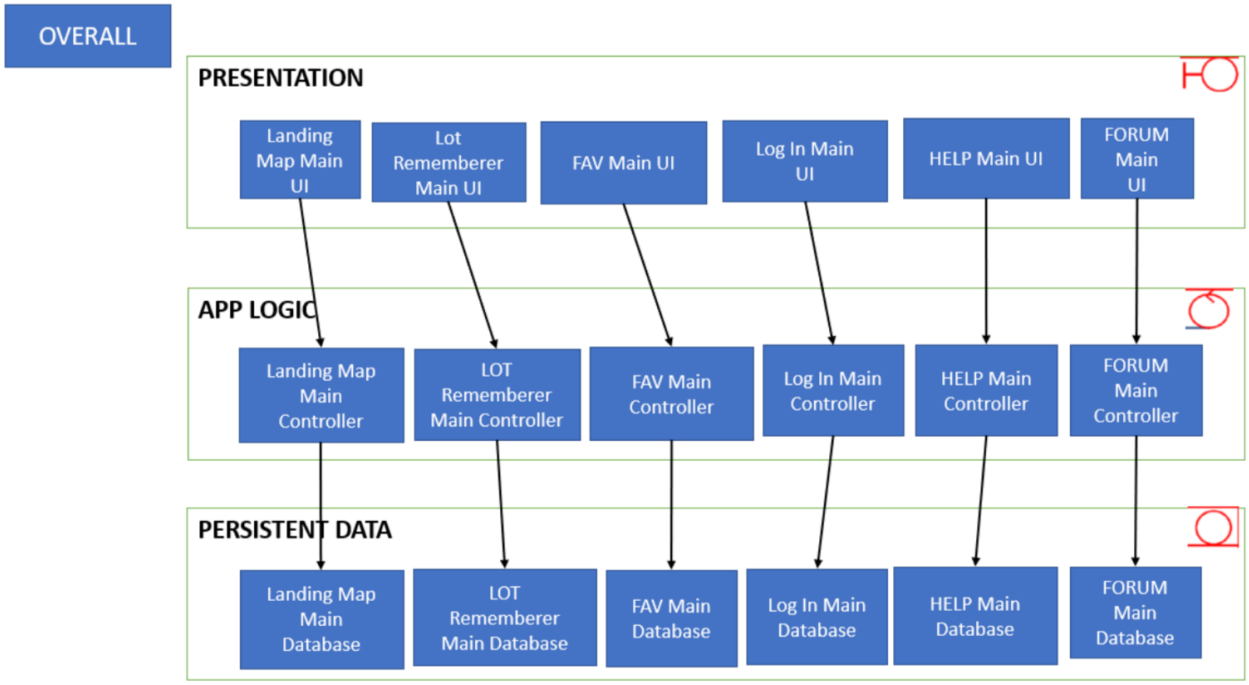
Future of smart parking - Our current application mostly includes basic and personalised features, hence in order to further improve our application, we aim to implement even more advanced features such as integration of a smart parking system where there is smart payment and booking system for car park lots. Another example will be smart notifications that will learn the user's parking activity or preferences using Machine Learning. This will enable for greater advanced personalised features as well, which meets the main goal of our application which is to cater to our customer’s needs as much as possible. Improvements can also be implemented on our current features, such as adding more advanced filters, being able to auto complete the user input in the search bar etc.

Data Security - Moreover, with the rise in cyber threats in the world today, we acknowledge that our application could be a potential target of cyber attacks. Thus, to further improve the security of our application, we aim to implement safeguards that will allow for stable application security in Car park Companion. This will be especially crucial since we also plan to implement advanced features as mentioned above, and these would require sensitive data such as credit card information.

Government Collaboration - Lastly, as most of our data is from data.gov.sg, hence we do face restricted access to data due to government regulations. This results in limited information available which is a limiting factor in our application. Hence, we aim to reduce this lack of information as much as possible in the future through collaborations with the government, gaining permission for greater access of data regarding parking in Singapore. This will hopefully enable us to provide an improved version of Car park Companion that will meet almost all drivers needs.

6 Design Considerations

6.1 System Architecture Diagram



6.2 Key Design Issues

6.2.1 Identifying and Storing Persistent Data

Our application requires the information about the car parks available in Singapore and we will be extracting these data through an API call via data.gov.sg. Since we will be using these car parks information frequently, we will be storing these information in a database in Firebase. By doing so, it will allow the application to easily access the required information by invoking the Firebase user functions.

Our application also requires the user to login to use some of the features such as Favourites and Lot Remember. The information of the users such as their favourite lists are recorded and stored as persistent data in the database in Firebase. These will be done through mapping the objects to the database. This allows us to retain the required information even if the user logs out/exits the application.

6.2.2 Providing Access Control

We used Firebase as our database and implemented a login using Firebase functions. Users can only use the features Favourite, Lot Rememberer and Forum if they login. With regards to the rest of the application, to promote ease of usage, login is not required since there is no private content at risk.

6.2.3 Strategy, Factory and Facade Pattern

We have implemented abstract classes and interfaces to introduce loose coupling and high cohesion. Moreover, with the implementation of abstract classes we promote encapsulation to limit information between classes.

Problem: For instantiation and storing of objects, all instantiation would go through the startup class.

Solution: We create an abstract class called main app which has multiple classes inheriting it. These classes would override the mainapp methods and contain their own set of instantiations as their attributes rather than having all instantiations in one class.(Loose coupling) This allows the creation/instantiating of classes dynamically at runtime enhancing flexibility in object creation.

Problem: A database manager class contains all methods which communicate with different databases. (High coupling)

Solution: We created a DBManager Interface where multiple manager classes would inherit from it. Each manager contains their own methods and attributes to control the information between the databases and other entities.

6.2.4 Decoupling by abstraction

We tried to achieve loose coupling as much as possible to prevent any unnecessary hard coding or dependencies between each class. For our filters function, we created an abstract class called Filter whereby each filter type is an extension of Filter. This allows us to standardise the main functions of each filter and any changes made can be applied throughout all childs classes of Filter. It also allows us to add any new filters in the future without reimplementing a concrete class.

6.2.5 Observer Pattern

Promotes coupling between the Subjects and the Observer.

To introduce the observer pattern, we designed our updating of car parks in the area to be disconnected from the objects. The subject is the list of car parks while the objects are the different screens, other databases etc. When the user slides on the distance slider or scrolls on the screen, the list of car parks would then be updated/pulled down from the external database. The screens and other databases would then be updated based on the new list. Hence this ensures the reusability of the list of car parks and objects independently of each other.

6.2.6 UI Design Principles

In our design, we implemented the MVC architecture in our system. Hence all boundary classes would communicate to entities through controller classes. When the user presses a button, the controller classes would communicate the state change to the entities and would then update the boundary/user interface according to the state changes of the entities.

7. Other Information

Appendix A: Glossary

1. Data Dictionary

Terms	Definition
Application	Refers to an instance of the mobile application, *(insert app's given name)*, which is designed to help users scout for suitable car parking spots in Singapore.
User	Refers to the person interacting with (and consuming the content provided by) the mobile application stated above. Most likely a driver or passenger commuting on Singapore roads.
Car Park	Refer to a single instance of a car park object. A car park is defined as an area or building where cards or other vehicles may be stationed ("parked") temporarily
Car Park Type	This refers to labels and car park categories that a car park object might possess. The car park types are exhaustively classified as follows: - Surface - Multi-storey
Vehicle Type	This refers to labels and vehicle categories that a user's vehicle might belong to. The vehicle types are exhaustively classified as follows: - Car - Lorry - Van - Motorcycle
Address	This refers to the address of a specified car park object. The address would include information like: - Road/street name - Block number (if applicable) - Postal Code
Free Parking	This refers to a label that Car Park object might have that indicates that it offers parking free of charge

Fares	This refers to the pricing rates imposed on a given car park. This information may not always be available
Nearby Facilities	This refers to facilities that are located near a car park. By our definition, these are facilities that are 300 metres away or fewer from the car park. Identified facilities are listed exhaustively as shown below: - Petrol Station - Cash Card Top-Up Machine
Description	This refers to other static (unchanging) information not already covered. And may include but are not limited to: - Number of storeys - Whether car park is sheltered
Vacancy	This refers to the number of car park lots available at a given car park
Distance	For our application this refers exclusively to either of the following two things: (1) Distance away from live location (user) (2) Distance away from desired destination (searched input)
Favorite	This refers to the user's ability to add a car park into the 'Favorites List' for future reference
Favorites List	This refers to a list of car parks that the user has 'favourited'
Map	This exclusively refers to the map of Singapore's streets and roads
Carpark Markers	This refers to car park entities/objects marked on the map using geospatial data (i.e. coordinates)
Car Park Record	This refers to car park entities/objects listed as a record in a list of car park records. Each record is a 'block' of information providing the details of unique car parks returned by the application.
Map View	This refers to the application's ability to toggle into a map-focused representation of the car park data

	This mode shows car parks as pins on the street map.
List View	This refers to the application's ability to toggle into a list representation of car park data. This mode shows the car parks as records in a sequential list of records.
Live Location	This refers to most mobile devices' GPS capability, in which the user's current location can be pinpointed and fed as input for some application features to work.
Search Radius	This refers to a circular radius around the user's live location and this radius determines the maximum distance from the user in which a car park will be detected.
Lot Rememberer	This refers to the application's function in helping users remember where they parked their cars. This feature can be accessed through a car park's detail page and records the following information: - Car park address - Car park name - Car park lot number - Car park floor number - Duration of parking - Fares incurred (using car park charging rate multiplied by duration, taking into account the period of charging)
Home page	This by definition will be the application's map view. This will always be the user's initial entry point of view when: - Entering the application - Pressing the 'go-to-home' button The home page acts as the default screen for the entire application.
Direction	In the context of this application, this exclusively refers to the applications' guidance feature in which users are offered directions to get to their desired car park destination. This is achieved with the support of Google Maps' Direction functionality.
Car park Forum	This refers to a dedicated 'blog' page unique to individual car parks. Via this page, users can share noteworthy

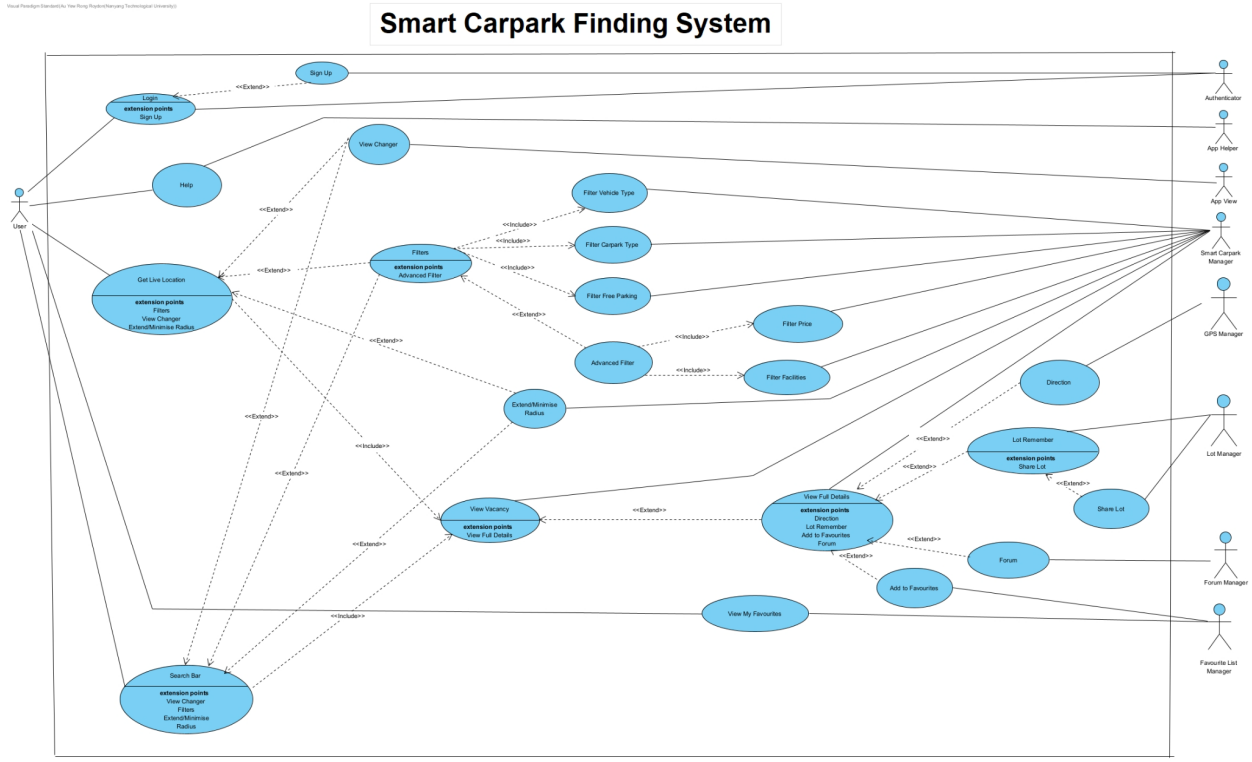
	remarks with other users.

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

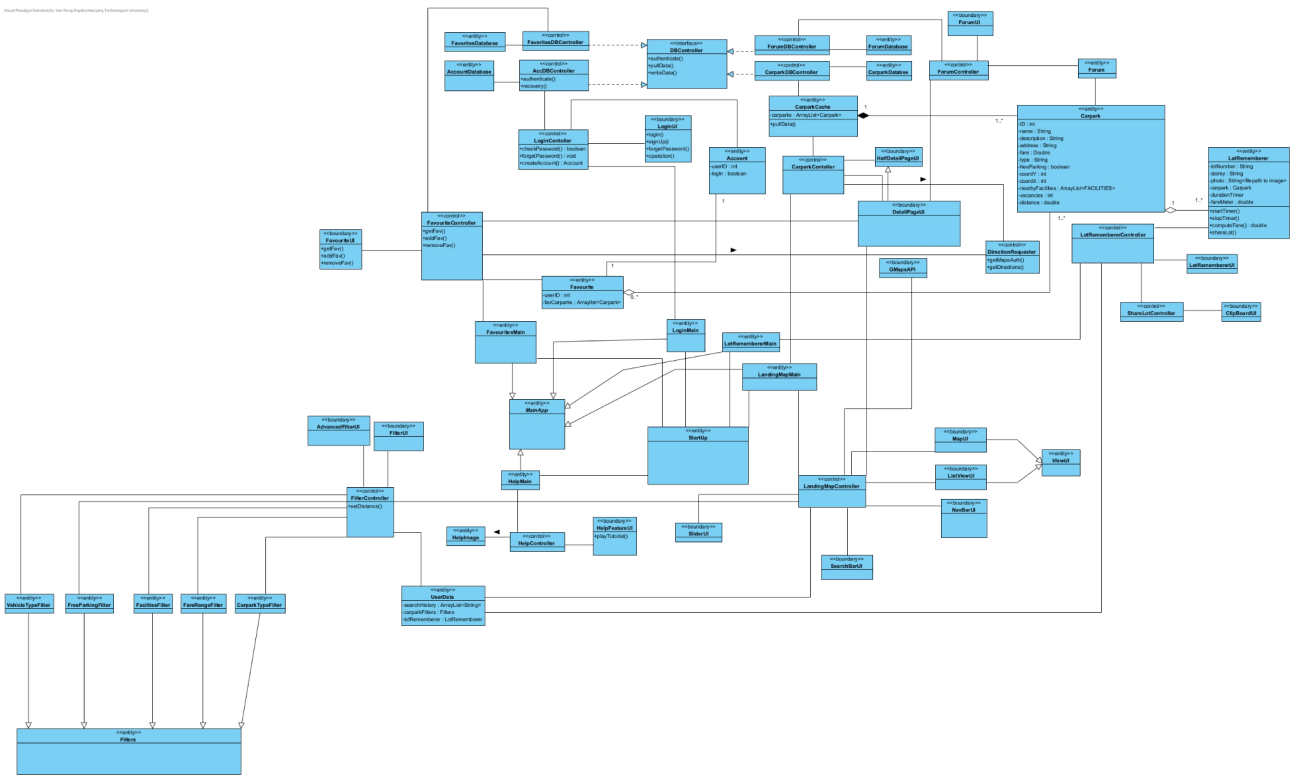
Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

1. Use Case Diagram

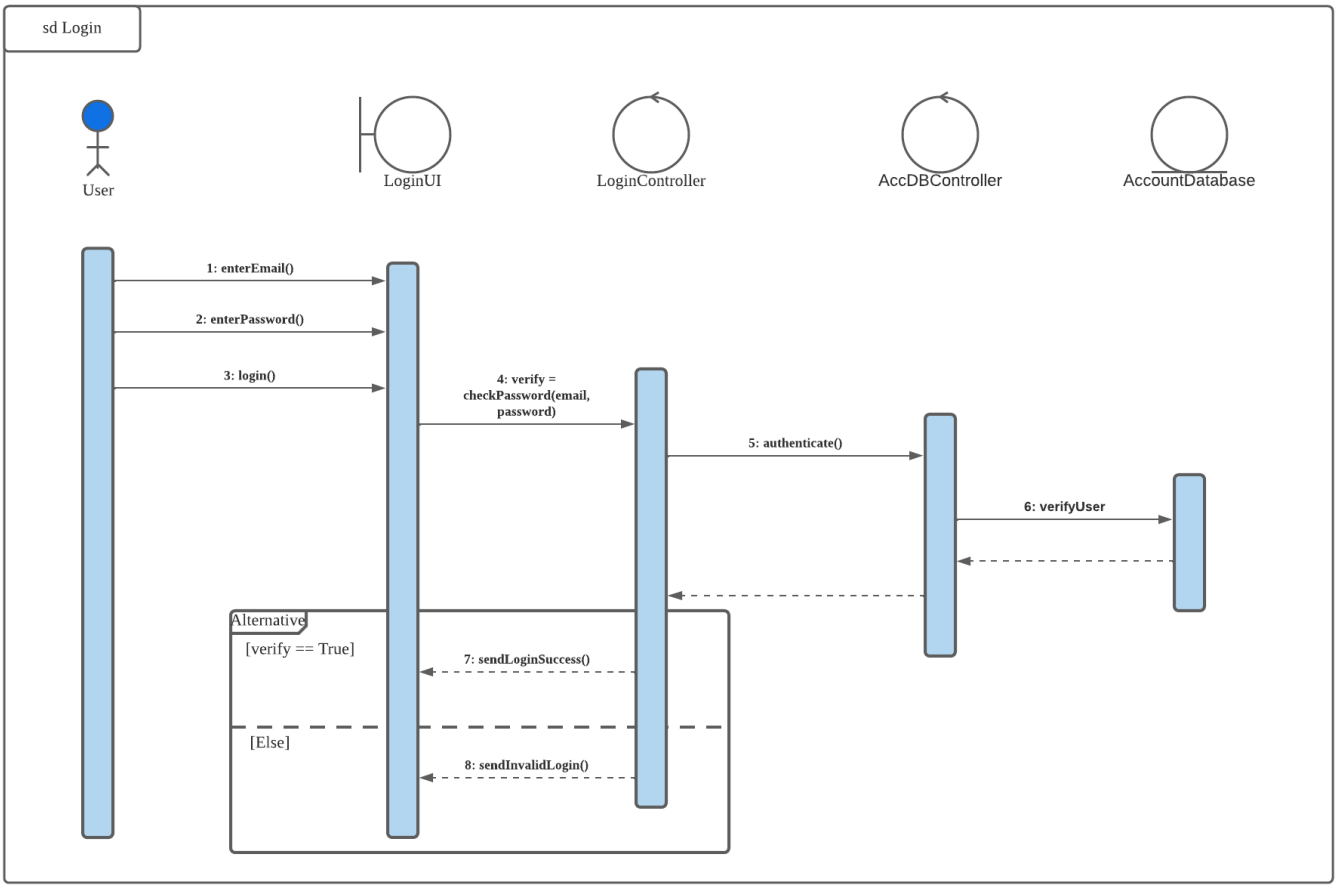


2. Class Diagram

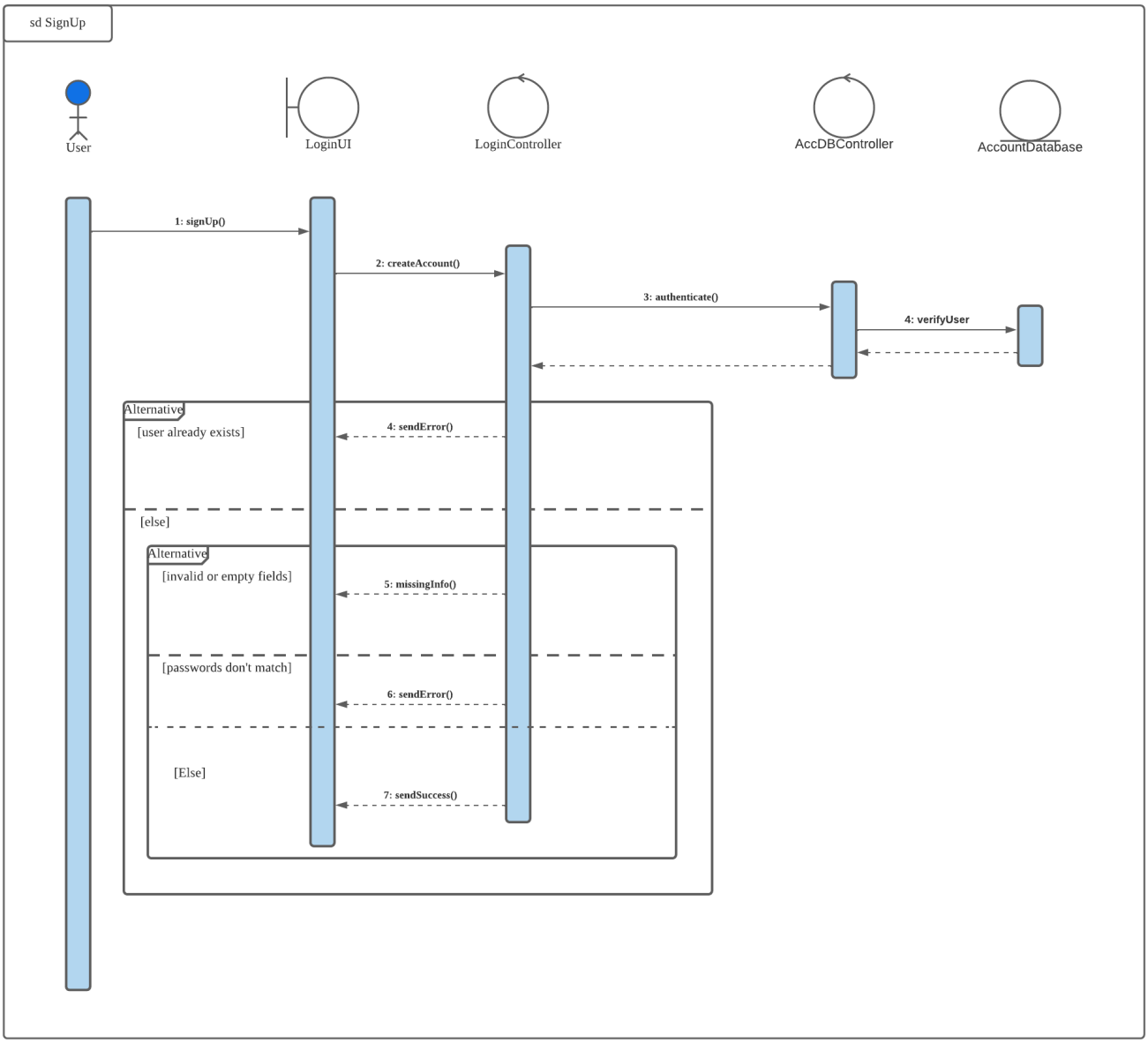


3. Sequence Diagrams

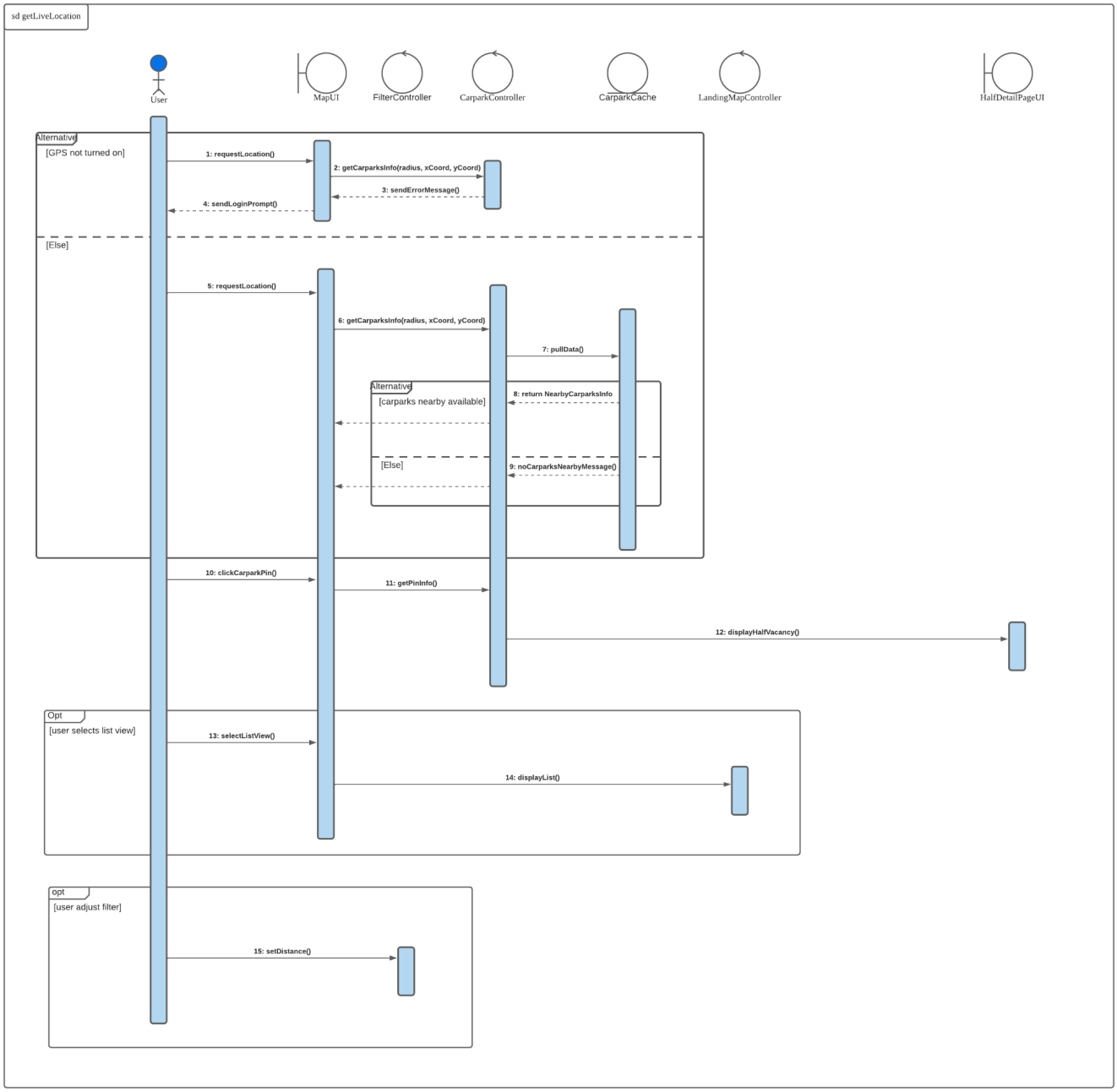
3.1 Login



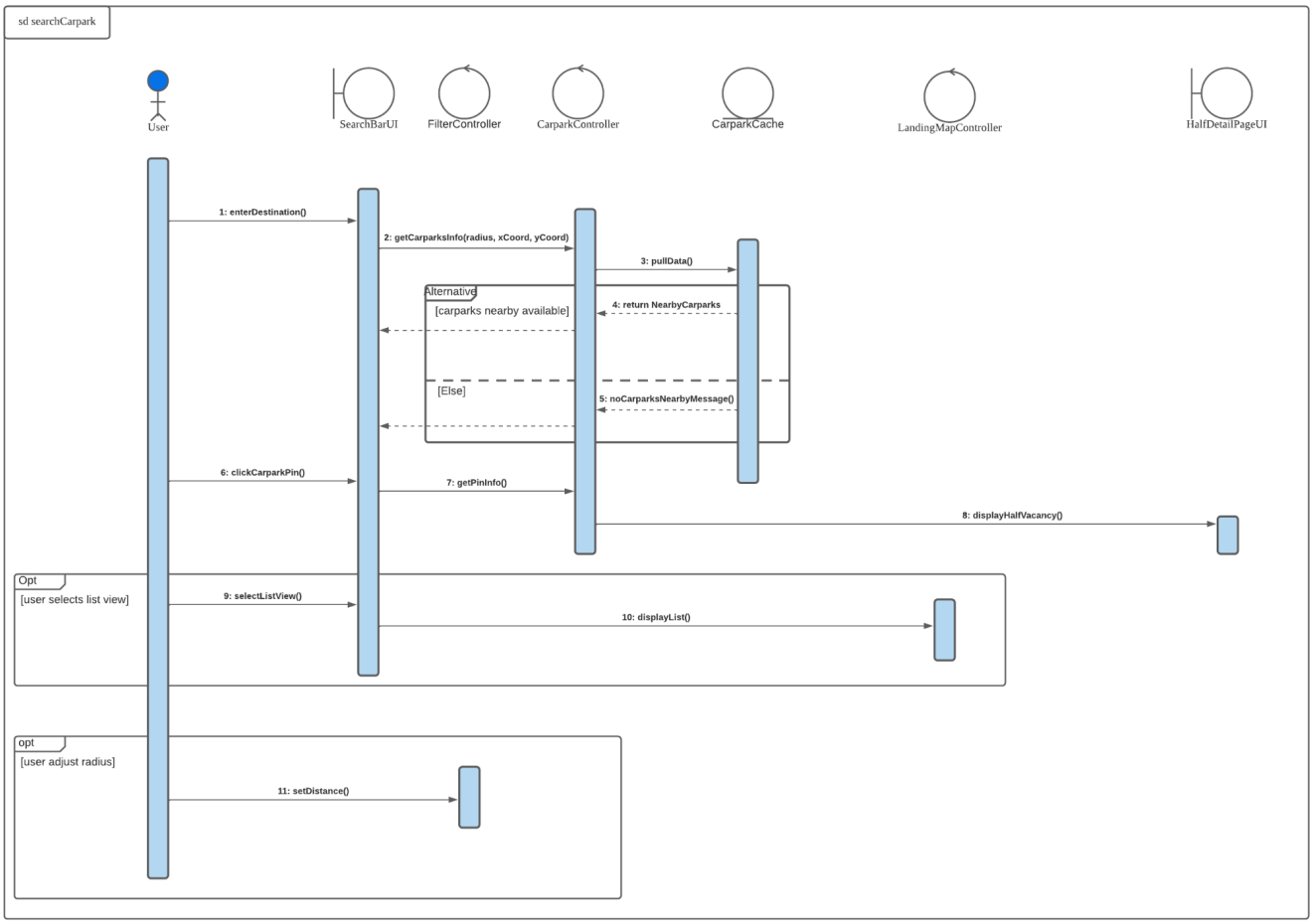
3.2 Sign up


sd SignUp
User
LoginUI
LoginController
AccDBController
AccountDatabase
1: signUp()
2: createAccount()
3: authenticate()
4: verifyUser
Alternative
[user already exists]
4: sendError()
[else]
Alternative
[invalid or empty fields]
5: missingInfo()
[passwords don't match]
6: sendError()
[Else]
7: sendSuccess()

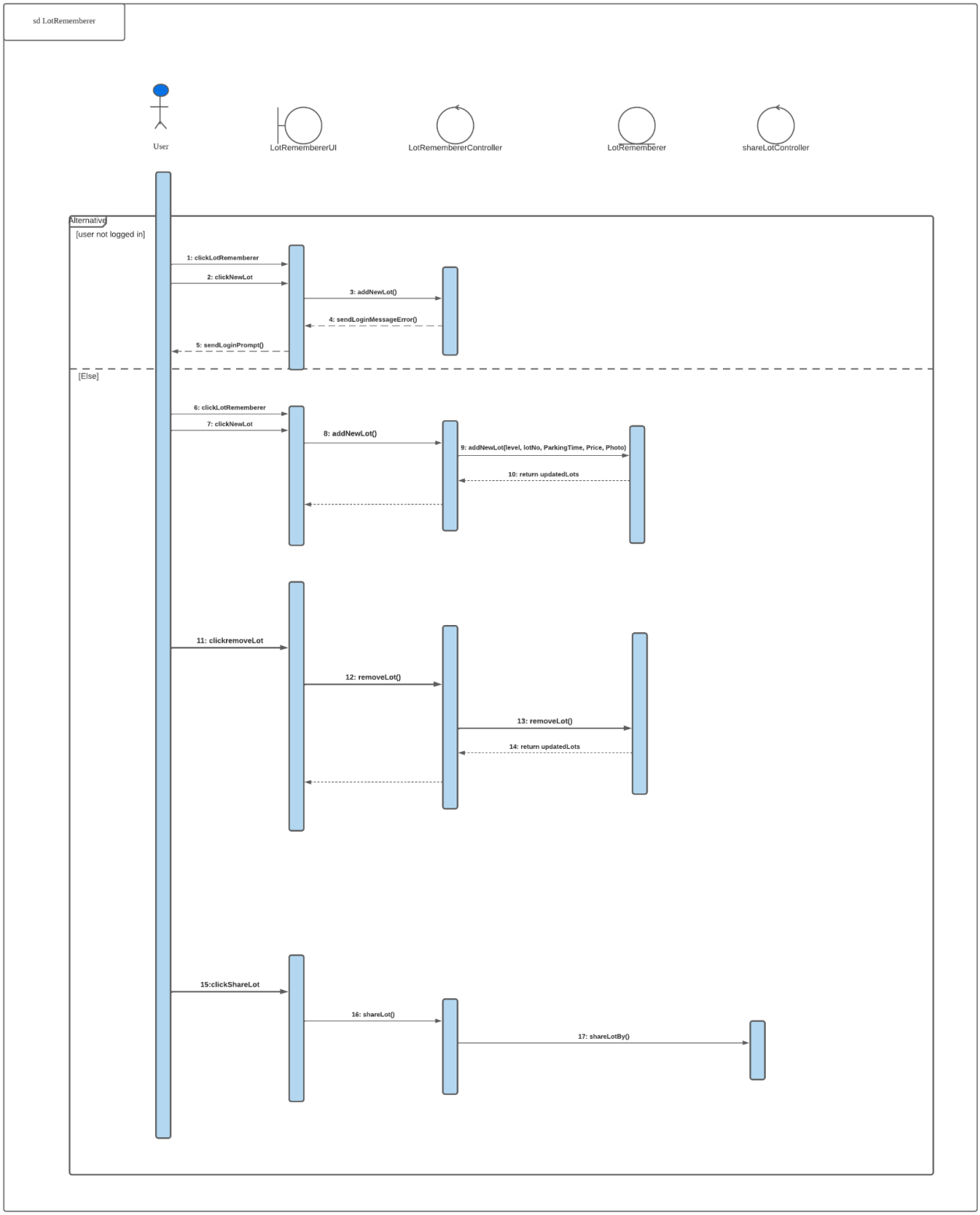
3.3. Get Live Location



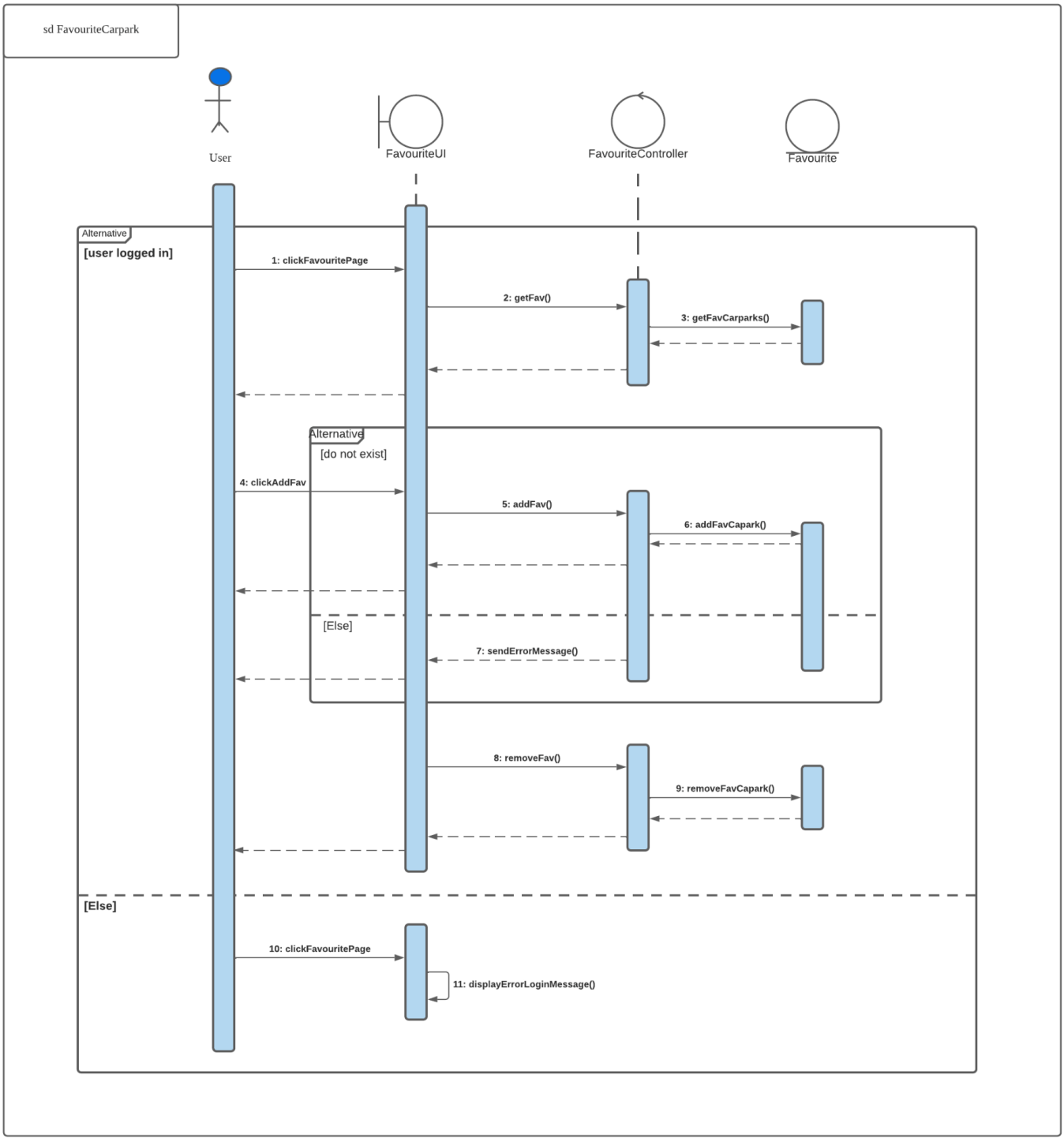
3.4 Search Carpark



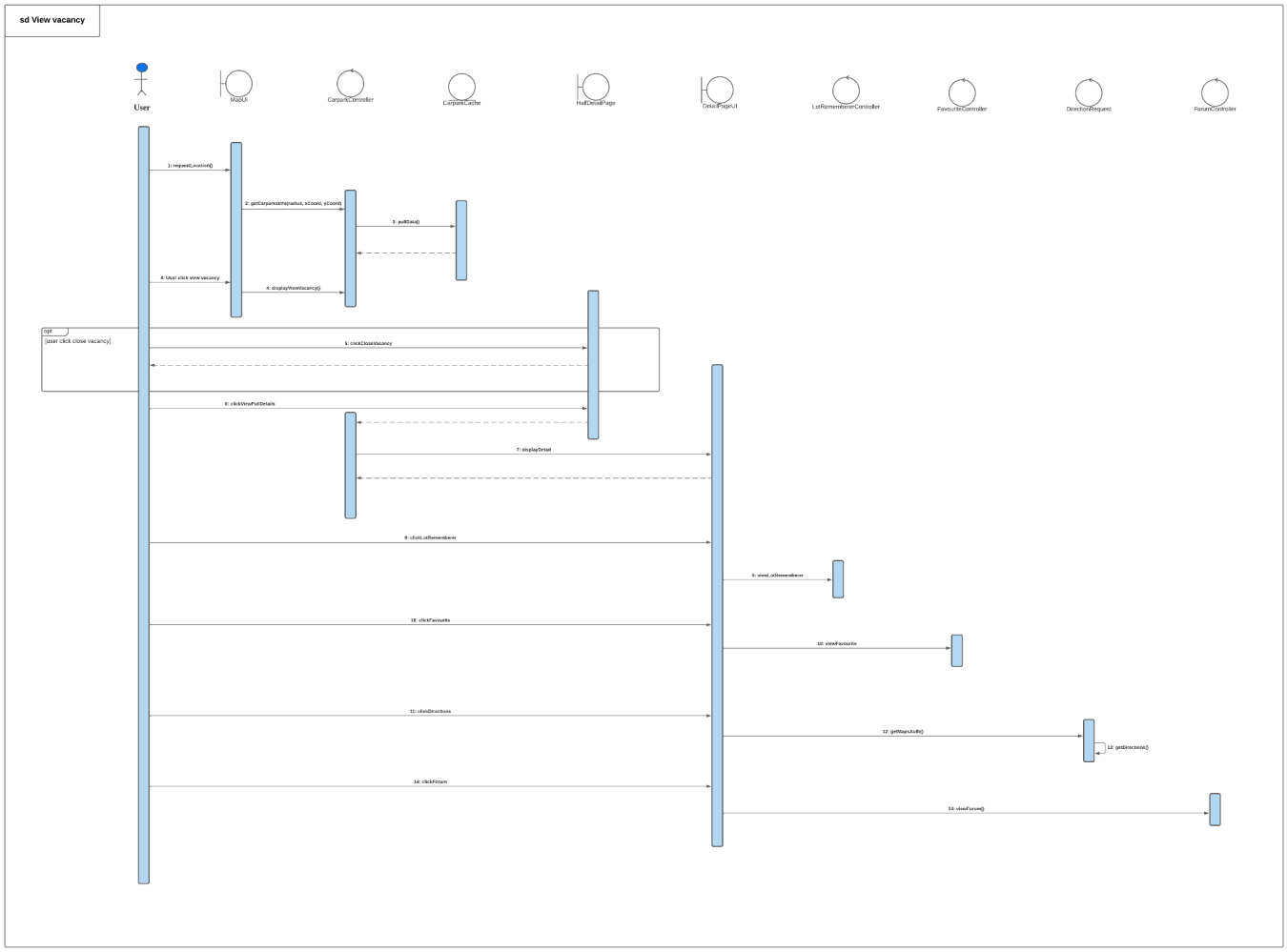
3.5 Lot Rememberer



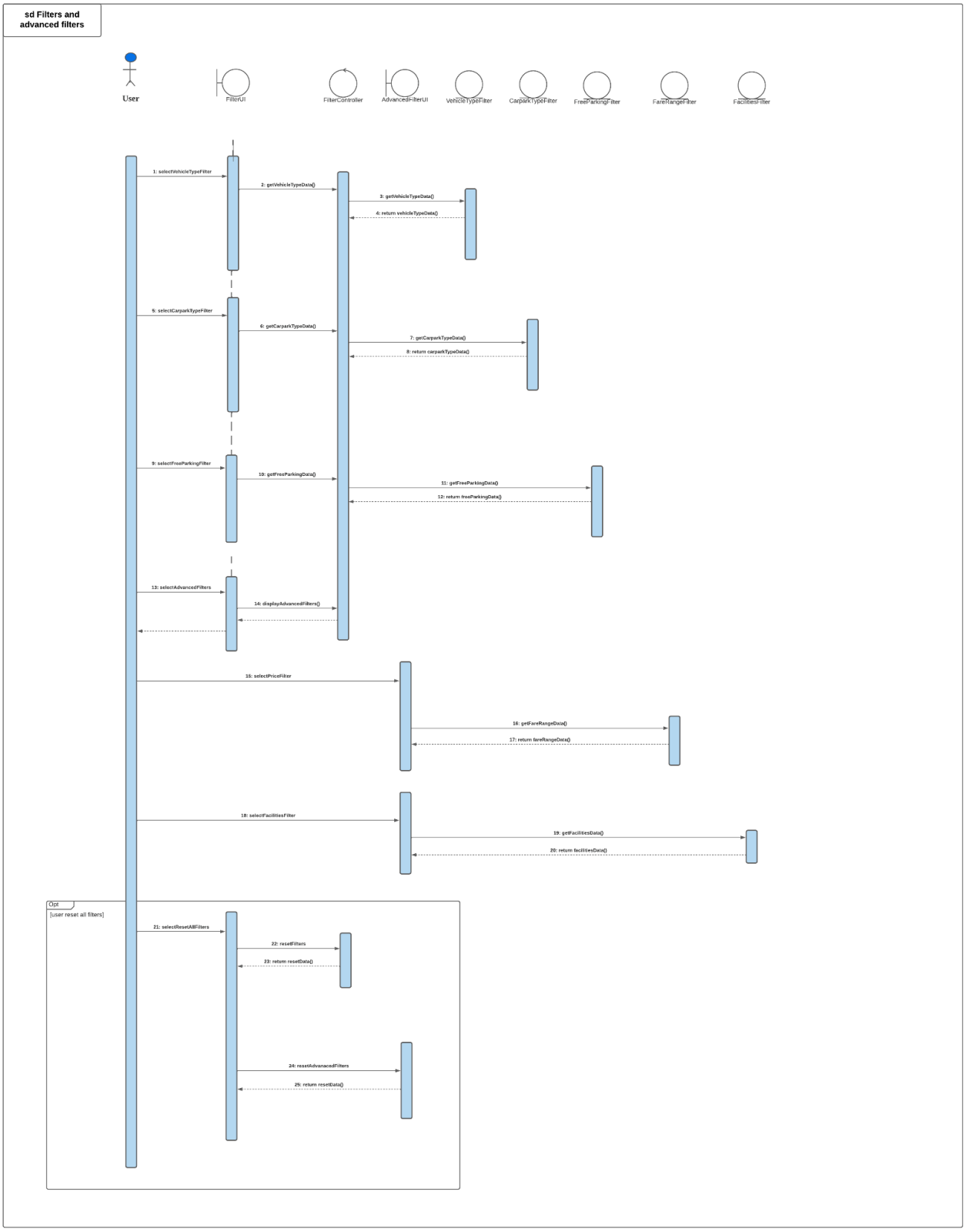
3.6 Favourite Carpark



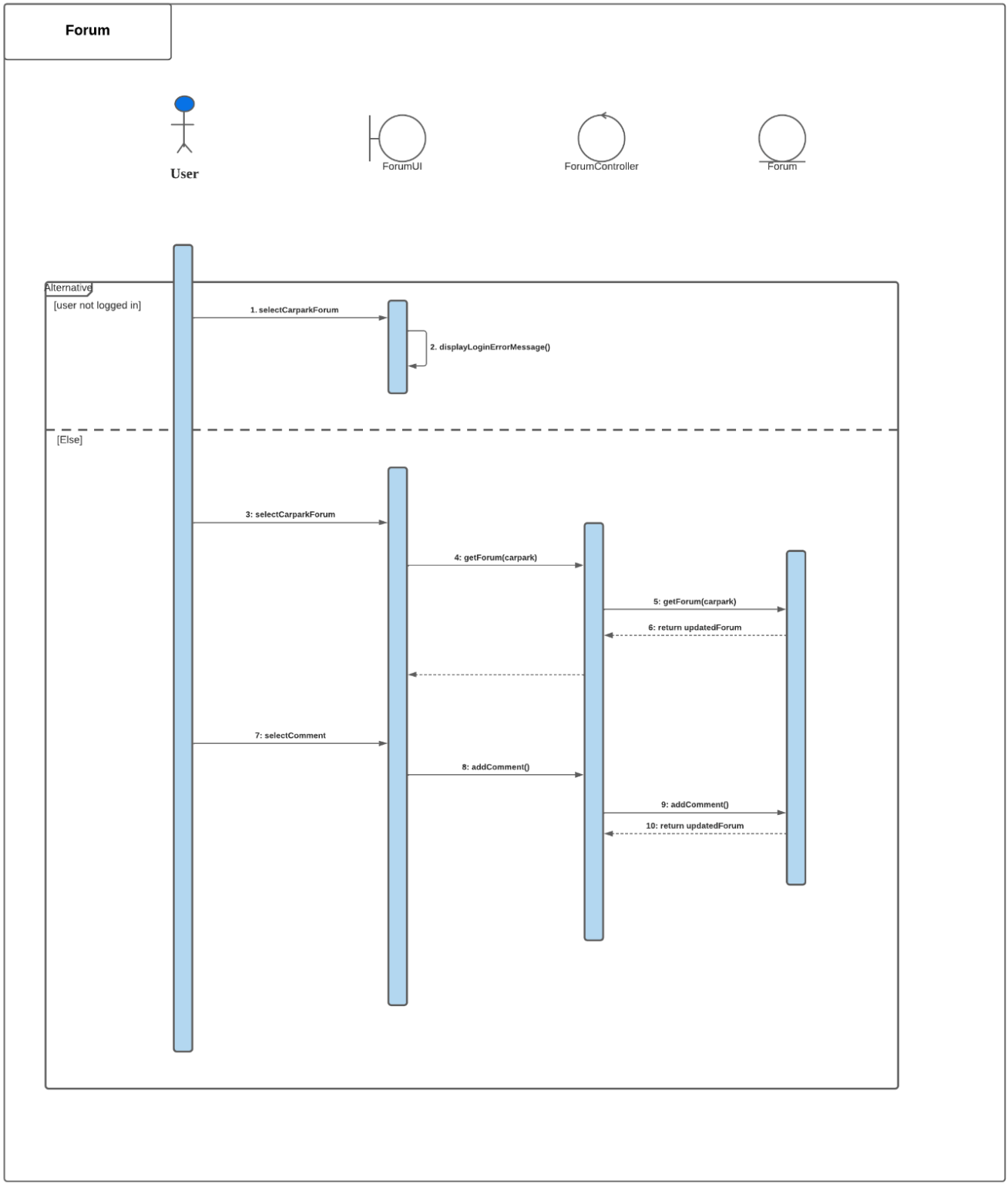
3.7 View Vacancy



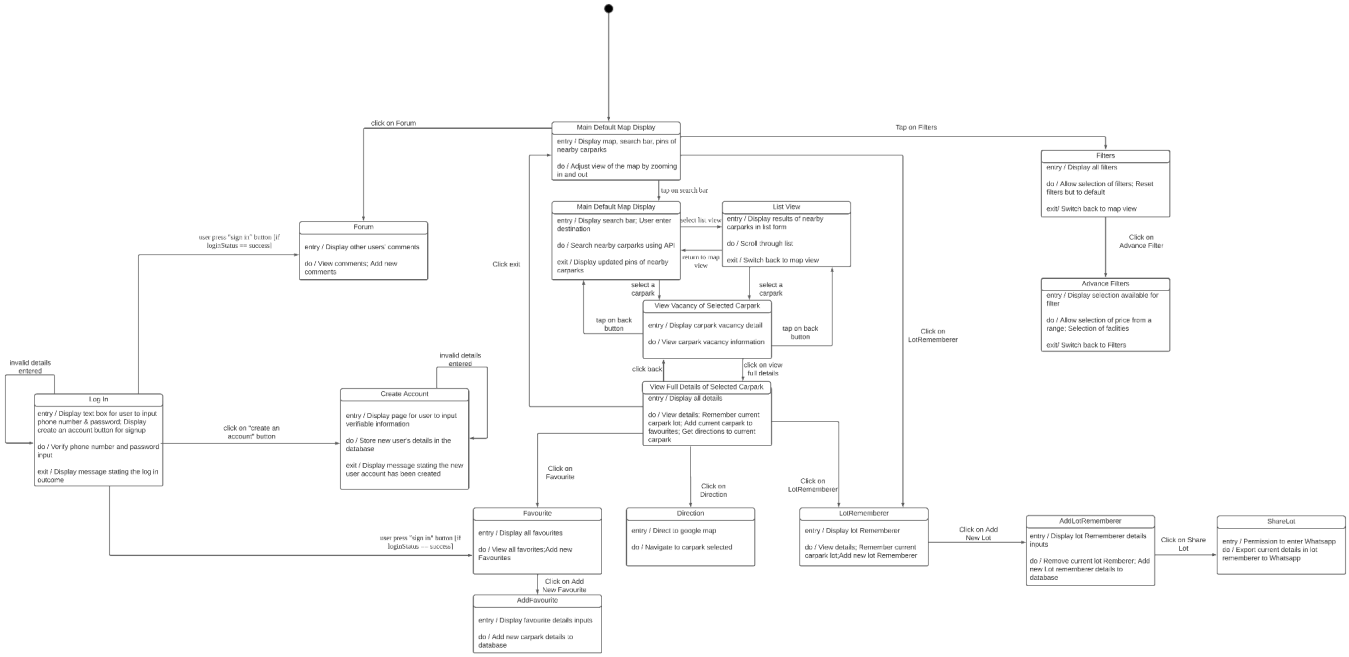
3.8 Filters and Advanced Filters



3.9 Forum


Forum
User
ForumUI
ForumController
Forum
Alternative
[user not logged in]
1. selectCarparkForum
2. displayLoginErrorMessage()
[Else]
3: selectCarparkForum
4: getForum(carpark)
5: getForum(carpark)
6: return updatedForum
7: selectComment
8: addComment()
9: addComment()
10: return updatedForum

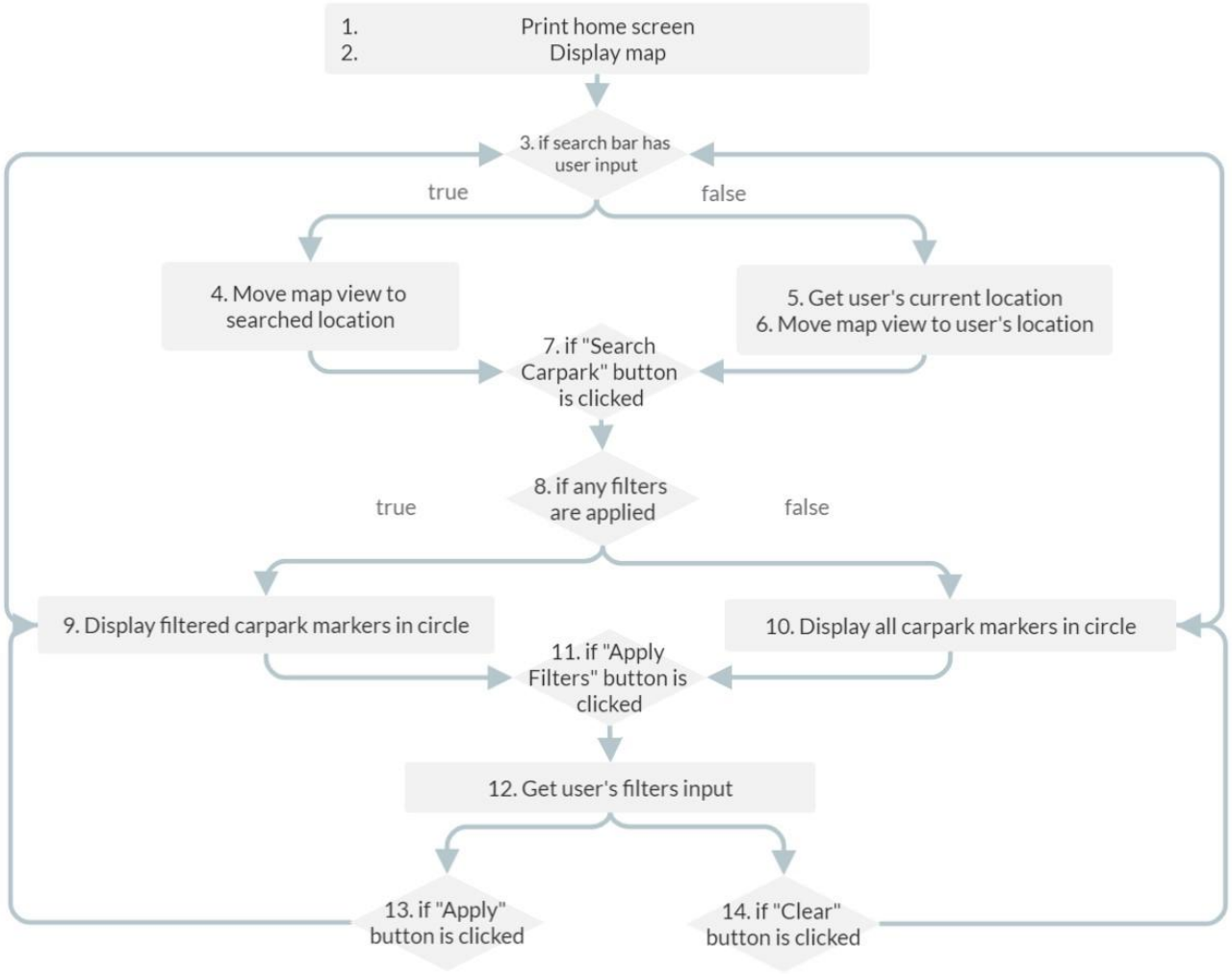
4. State Machine Diagram



Appendix C: Testing

1. White Box Testing - Display map and filters

This tests the display map and filters function. It first prints out the home page and google maps. It then checks if the user has searched for a location. If there is, the map shifts to that particular location and if not, it gets the user's current location and the map shifts to that location. It then checks if any filters for car parks are applied. If there is, only the filtered set of car park markers will be displayed and if not, all the car park markers will be displayed. The user can then input into the search location bar or click on apply filters which will loop back and repeat the steps accordingly.



One set of basis paths

1. 1,2,3,5,6,7,8,10
2. 1,2,3,4,7,8,10
3. 1,2,3,4,7,8,10,11,12,13,9
4. 1,2,3,4,7,8,9,11,12,14,10

4 test cases

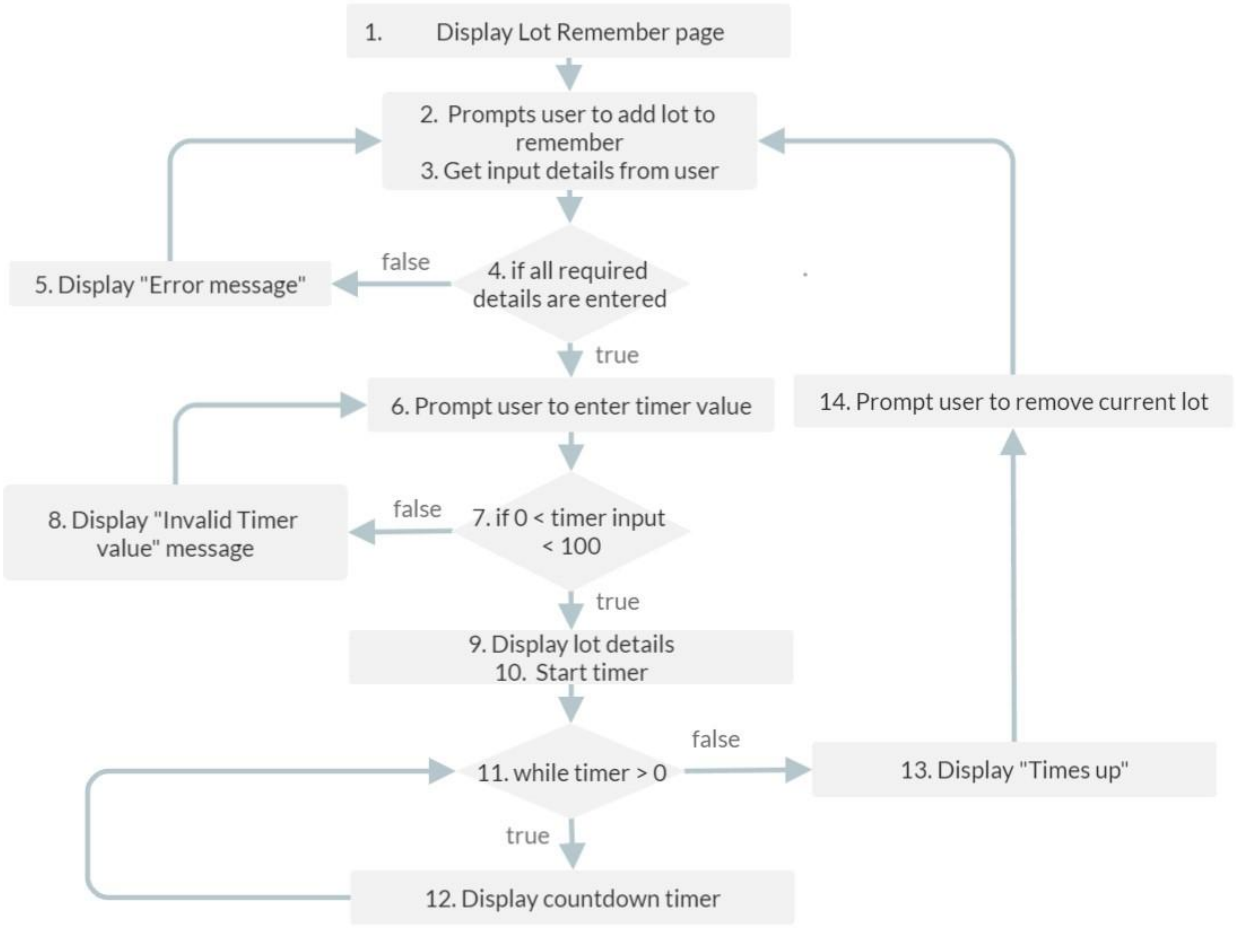
1. User clicks on Refresh
2. User only searches for a location in the map.
3. User searches for a location and applies filters.
4. User applies filters, applies a different set of filters and then clears all the filters.

4 actual execution paths

1. 1,2,3,5,6,7,8,10
2. 1,2,3,5,6,7,8,10,3,4,7,8,10
3. 1,2,3,5,6,7,8,10,3,4,7,8,10,11,12,13,9
4. 1,2,3,5,6,78,10,11,12,13,9,11,12,13,9,11,12,14,10

2. White box Testing - Lot remember

This tests the Lot Remember function of the app. When there is currently no lot, it prompts the user to add a new lot to remember. It will then make sure all required fields are entered and valid. If all is valid, the saved lot with a countdown timer will be displayed. The timer will continue to display until the timer is up which will then change to display “Times Up!”. It then prompts the user to delete that lot and add a new lot to remember for a future occurrence.



One set of basis paths

1. 1,2,3,4,5,8
2. 1,2,3,4,5,6,7,9,10
3. 1,2,3,4,5,6,7,**9,10**,11,12

4 test cases

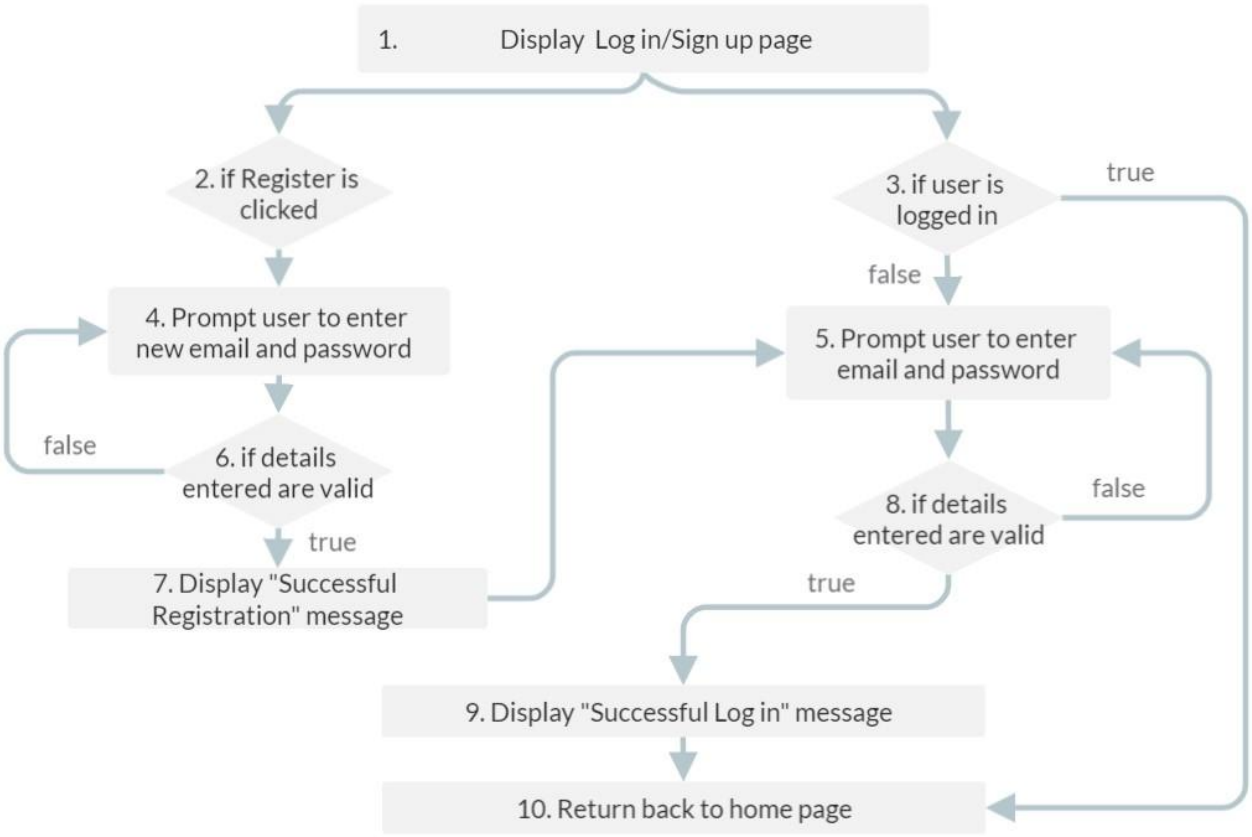
1. User doesnt do anything
2. User inputs invalid details
3. User inputs invalid timer value
4. User inputs details and timer is running
5. User inputs details and timer is up

4 actual execution paths

1. 1
2. 1,2,3,4,5,8,2
3. 1,2,3,4,6,7,8,6,7,9,10
4. 1,2,3,4,5,6,7,9,10,9,10,9,10,9,10…
5. 1,2,3,4,5,6,7,9,10,9,10,9,10,9,10,9,10,9,10,11,12,2

3. White box Testing - Log in & Sign up

This tests the log in and sign up functions. If the user is already logged in, it will go back to the home page and if not, it will prompt the user to log in. If the details are correct it will then be successful and go back to the home page and if it's not correct it will prompt the user to re-enter the details again. If the user clicks on the register, it will prompt the user to enter details to create a new account. If the details are valid it will be successfully created and then continue to prompt users to log in. If not, it will prompt the user to re-enter the new details again.



One set of basis paths

1. 1,3,10
2. 1,3,**5,8**,9,10
3. 1,2,**4,6**,7,**5,8**,9,10

4 test cases

1. User is logged in
2. User logs in successfully
3. User logs in with incorrect details and tries again
4. User registers for a new account with valid details
5. User registers for a new account with invalid details and tries again

4 actual execution paths

1. 1,3,10
2. 1,3,5,8,9,10
3. 1,3,5,8,5,8,5,8,9,10
4. 1,2,4,6,7,5,8,9,10
5. 1,2,4,6,4,6,7,5,8,9,10

1. Black Box Testing - Favourites

Generic Testing

S/N	Description	Expected outcome	Actual outcome
1.	Attempts to access favourites without logging in	The system prompts user to log in	The system prompts user to log in
2.	User is logged in and can see current favourites	The list of the user's favourite car parks saved previously are displayed	The list of the user's favourite car parks saved previously are displayed
3.	Add new car park to favourites list	The selected carpark is displayed in the list of the user's favourites car parks	The selected carpark is displayed in the list of the user's favourites car parks
4.	Remove carpark from favourites list	The selected carpark is removed in the list of the user's favourites car parks	The selected carpark is removed in the list of the user's favourites car parks

Specific Testing

S/N	Scenario	User Input	Expected outcome	Actual outcome
1.	Logged in: False		The system prompts user to log in	The system prompts user to log in
2.	Logged in: True Saved list: null	Add carpark A Add carpark B	Favourites list: ● Carpark A ● Carpark B	Favourites list: ● Carpark A ● Carpark B
3.	Logged in: True Saved list: ● Carpark A ● Carpark B ● Carpark C	Add carpark D	Favourites list: ● Carpark A ● Carpark B ● Carpark C ● Carpark D	Favourites list: ● Caprark A ● Carpark B ● Carpark C ● Carpark D
4.	Logged in: True Saved list: ● Carpark A ● Carpark B	Add carpark D Add carpark E Remove carpark B Remove carpark C	Favourites list: ● Carpark A ● Carpark D ● Carpark E	Favourites list: ● Carpark A ● Carpark D ● Carpark E

	● Carpark C			
5.	Logged in: True Saved list: ● Carpark A ● Carpark B ● Carpark C	Remove all car parks A, B and C	Favourites list: null	Favourites list: null

2. Black Box Testing - Display map and applying filters

Generic Testing

S/N	Description	Expected outcome	Actual outcome
1.	User opens app and does nothing	The map is displayed with all carpark markers.	The map is displayed with all carpark markers.
2.	User searches for a specific location in search bar	The map moves to the searched location	The map moves to the searched location and displays all carpark markers
3.	User searches for a specific location in search bar and applies some filters	The map moves to the searched location and displays curtain carpark markers based on filters applied	The map moves to the searched location and displays curtain carpark markers based on filters applied
4.	User changes filter different from current filters.	The displayed carpark markers changes based on the new filters applied	The displayed carpark markers changes based on the new filters applied
5.	User clears all applied filters	All carpark markers are displayed.	All carpark markers are displayed.
6.	User clicks on current location button	The map moves back to user's current location	The map moves back to user's current location

Specific Testing

S/N	Scenario	User Input	Expected outcome	Actual outcome
1.	Search bar: null Filters: null	Search bar: null Filters: null	Map: current location Carpark markers: All	Map: current location Carpark markers: All
2.	Search bar: null Filters: null	Search bar: Ang mo kio avenue 10 Filters: null	Map: Ang mo kio avenue 10 Carpark markers: All	Map: Ang mo kio avenue 10 Carpark markers: All
3.	Search bar: Ang mo kio avenue 10	Search bar: Tampines street 27	Map: Tampines street 27	Map: Tampines street 27

	Filters: null	Filters: null	Carpark markers: All	Carpark markers: All
4.	Search bar: null Filters: null	Search bar: Ang mo kio avenue 10 Filters: Turn ON Multi storey carpark	Map: Ang mo kio avenue 10 Carpark markers: ONLY multi storey car parks	Map: Ang mo kio avenue 10 Carpark markers: ONLY multi storey car parks
5.	Search bar: null Filters: null	Search bar: Boon lay west avenue 4 Filters: Turn ON Multi storey carpark & Free parking	Map: Boon lay west avenue 4 Carpark markers: ONLY multi storey car parks AND with free parking	Map: Boon lay west avenue 4 Carpark markers: ONLY multi storey car parks AND with free parking
6.	Search bar: null Filters: ON Multi storey carpark & Free parking	Search bar: Marina Square Mall Filters: Turn OFF Multi storey carpark & Free parking Turn ON Open space carpark & Night parking	Map: Marina Square Mall Carpark markers: ONLY open space car parks AND with night parking available	Map: Marina Square Mall Carpark markers: ONLY open space car parks AND with night parking available
7.	Search bar: null Filters: ON Multi storey carpark & Free parking	Search bar: Jcube Shopping Mall Filters: clear filters	Map: Jcube Shopping Mall Carpark markers: All	Map: Jcube Shopping Mall Carpark markers: All

3. Black Box Testing - Login

Generic Testing

S/N	Description	Expected outcome	Actual outcome
1.	User logins with a valid account email and password	Successful login. The map is displayed with all carpark markers.	Successful login. The map is displayed with all carpark markers.
2.	User logins with a valid existing email, but an invalid/incorrect password	Unsuccessful login. The system prompts the user to re-enter details again.	Unsuccessful login. The system prompts the user to re-enter details again.
3.	User logins with an invalid email, but password is valid	Unsuccessful login. The system prompts the user to re-enter details again.	Unsuccessful login. The system prompts the user to re-enter details again.
4.	User clicks on login without entering their details into all required fields.	Unsuccessful login. The system prompts the user to enter the required details into the respective fields.	Unsuccessful login. The system prompts the user to enter the required details into the respective fields.

Specific Testing

S/N	Scenario	User Input	Expected outcome	Actual outcome
1.	Email: Valid Password: Valid	Email: happy@gmail.com Password: 123456	Successful login. User redirected to map interface, with all carpark markers displayed.	Successful login. User redirected to map interface, with all carpark markers displayed.
2.	Email: Valid Password: Incorrect/Invalid	Email: happy@gmail.com Password: 1234	Unsuccessful login. User prompted to re-enter details.	Unsuccessful login. User prompted to re-enter details.
3.	Email: Invalid Password: Valid	Email: happy Password: 123456	Unsuccessful login. User prompted to	Unsuccessful login. User prompted to

			re-enter details.	re-enter details.
4.	Email: Valid Password: Empty	Email: happy@gmail.com Password: NULL	Unsuccessful login. User prompted to enter all required details into respective fields.	Unsuccessful login. User prompted to enter all required details into respective fields.

4. Black Box Testing - Sign Up

Generic Testing

S/N	Description	Expected outcome	Actual outcome
1.	User enters valid email and password	Account creation successful. User successfully logged in. The map is displayed with all carpark markers.	Account successfully created. User successfully logged in. The map is displayed with all carpark markers.
2.	User enters an email that already exists	Account creation unsuccessful. The system prompts the user to re-enter email as current email already exists.	Account creation unsuccessful. The system prompts the user to re-enter email as current email already exists.
3.	User enters an invalid email address, that is in an incorrect format	Account creation unsuccessful. The system prompts the user to re-enter an email in a valid form.	Account creation unsuccessful. The system prompts the user to re-enter an email in a valid form.
4.	User enters a password that has a length less than 6	Account creation unsuccessful. The system prompts the user to re-enter a new password, as the current password entered is too short.	Account creation unsuccessful.The system prompts the user to re-enter a new password, as the current password entered is too short.

Specific Testing

S/N	Scenario	User Input	Expected outcome	Actual outcome
1.	Email: Valid Password: Valid	Email: john1@gmail.com Password: 987654	Account creation successful and the user logged in successfully. User redirected to map interface, with all carpark markers displayed.	Account created successfully and the user logged in successfully. User redirected to map interface, with all carpark markers displayed.
2.	Email: Already exists Password: Valid	Email: happy@gmail.com Password: 987654	Account creation unsuccessful. User prompted to re-enter email.	Account creation unsuccessful. User prompted to re-enter email.
3.	Email: Invalid Password: Valid	Email: john1 Password: 987654	Account creation unsuccessful. User prompted to re-enter a valid email.	Account creation unsuccessful. User prompted to re-enter a valid email.
4.	Email: Valid Password: Invalid	Email: john1@gmail.com Password: 9876	Account creation unsuccessful. User prompted to re-enter a longer password.	Account creation unsuccessful. User prompted to re-enter a longer password.