Software Requirements Specification

for

CarparkLEY?

Version 5.0 approved

Prepared by ChienMing

Willis Yang [redacted]
Glenn Tay [redacted]
Tu Xianan [redacted]
Kundu Koushani [redacted]
Mamuduri Paulani [redacted]

Nanyang Technological University, Singapore

18 April 2021

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Willis	10/2/21	Use case diagram and Security update	2.0
Everyone	24/2/21	Sequence Diagram, Dialog map, Class Diagram	3.0
Everyone	30/3/21	Changes to sequence diagram	4.0
Everyone	17/4/21	Complete the document	5.0

1. Introduction

1.1 Purpose

This document lays out a plan for the development of 'CarparkLEY?', a carpark application by Team ChienMing. The intended readers of this document are current and future developers working on CarparkLEY?. This document will comprise, but not restrict to, a summary of the system functionality, the functional and nonfunctional requirements, use-case models, class diagrams, sequence diagrams, and other analysis models.

1.2 Document Conventions

Described below is the style and format used for this document

- Section Heading
  - Font: Times
  - Face: Bold
  - Size: 18
- Subsection Heading
  - Font: Times
  - Face: Bold
  - Size: 14
- Main Body Content
  - Font: Times
  - Face: Normal
  - Size: 12

1.3 Intended Audience and Reading Suggestions

This document is intended for any developer, project manager, software tester or a user involved in the design, development or testing of the application CarparkLEY.

Users of this system are drivers. They can be Motorcycle, car, or heavy vehicle drivers. This document serves to inform users on the main functionality of the system and give a clearer overview of the software requirements. External Interface Requirements (Section 3) and System Features (Section 4) are most useful to the users.

Developers and Project managers involved in maintaining and enhancing the system will use this document to review capabilities and understand individual components of the system. This document allows developers to determine which section of the system must be focussed on for improvements or modification. It is recommended that the developers read this document from the beginning.

For software testers (Quality Assurance Engineers), this document shall act as a guideline to create relevant test cases. It shall also serve as the starting point to understand the architecture of the entire system before conducting tests. Main areas of reading shall be System Features (Section 4) and Non-Functional Requirements (Section 5) for system testing and requirements verification. Furthermore, those who wish to visualise the working of the application can view the interface too.

1.4 Product Scope

This project consists of the 'CarparkLEY' Application. This application allows users to search for car parks near his/her destination, check how many empty lots there are in real-time, and detailed rates and other information such as operating hours, type of parking system.

We had observed that carparks in town areas are very hard to find and usually have limited vacancy as compared to residential areas. Drivers may also have to circle around their destination many times to find carparks that may not even have vacancies.

Hence, this application aims to address this problem by allowing drivers to check carpark vacancies before actually going to the carpark. He/she can also view the carpark rates, allowing them to compare rates across different carparks and make an informed decision as to whether they should park at that particular car park or park at a different car park.

1.5 References

- Google Firebase is used as the database system for the application - https://firebase.google.com/
- Flutter (https://flutter.dev/) and Dart (https://dart.dev/) are used for the codebase of the application.
- Visual Paradigm UML Software (https://www.visual-paradigm.com/) is used to generate the various diagrams.

2. Overall Description

2.1 Product Perspective

CarparkLEY is an application that intends to help drivers in their daily lives to locate carparks that are near their destination. It provides the drivers an easy and direct way of searching and locating carparks when their destination carparks are already full.

Drivers would have access to this app,where they would be able to search for the carpark themselves. The primary features in this application are: Searching Carpark, Locating Carpark, Navigating to Carpark, and Saving Carpark to Favourites. Searching for carpark is the main feature of the app where drivers will input their destination and search for nearby carparks. From there, the user is able to know more information about the selected carpark and even be directed to the carpark from wherever they are via Google Maps.

2.2 Product Functions

The major functions are as follows:

2.2.1 Search for nearby carpark

2.2.1.1 User inputs their destination and the app will return a list of nearby carparks from their destination.

2.2.1.2 Users can sort the list according to distance, number of empty lots, or rates.

2.2.1.3 Users can view information on the selected carpark by clicking on it.

2.2.2 Navigate to Carpark

2.2.2.1 Users will click on the Navigate button which will open up Google Maps for them.

2.2.2.2 Google Maps will then direct them to the carpark.

2.2.3 Saving Carpark to Favourites

2.2.3.1 When logged in, users are able to save carparks to their favourites list and view it without having to search again.

2.3 User Classes and Characteristics

CarparkLEY is for motor car, motor bike and heavy vehicle drivers, who are looking for a carpark near their destination.

2.4 Operating Environment

- Operating Platform for Application:
  - Development platform: Flutter
  - Operating System platform: Android and iOS

- Database Server
  - Google FireBase

- API used for carpark location and direction
  - URA API
  - Places API

2.5 Design and Implementation Constraints

2.5.1 Loading Time: Because of the use of multiple APIs and having to convert the latitude and longitude coordinates of the carparks' locations, this causes the search for the carparks to be quite long and results will only be generated after awhile (roughly 2-3 minutes).

2.5.2 Mobile-Only Application: The CarparkLEY can only be run on Android and iOS mobile devices. As such, users on PC will be unable to use the application.

2.5.3 Security: The login system for our application uses a simple authentication and might not be the most secure way of protecting users' information against hackers. Additional security features need to be outsourced.

2.5.4 API used: The URA API mostly returns carpark around town areas, and hence it will be rather difficult to search for carparks in the non town areas. Additional carpark APIs need to be outsourced.

2.6 User Documentation

A demonstration video will be submitted which will show the users all the features of the application and how to use them.

2.7 Assumptions and Dependencies

2.7.1 Project Schedule

Due to the tight deadline, the team must complete the application within 13 weeks. After 13 weeks, the application must be fully functional and error-free, and must satisfy all functional and non-functional requirements.

2.7.2 Connectivity

Internet access is required as users must log in. Searching carparks and being directed to carparks via Google Maps will also require internet access.

2.7.3 Users' Hardware Capability

Users should have an Android or iOS mobile phone capable of running this application.

2.7.4 Authentication

Our app will require authentication in the form of user email and password. Authentication is also required as a form of confirmation pin to the email when the user registers for an account in our app.

2.7.5 Assumptions

2.7.5.1 Users are aware of how to use web and mobile applications.

2.7.5.2 Users of the app know how to navigate using the touchscreen.

2.7.5.3 Users will have a good internet connection.

2.7.5.4 Users' accounts are already registered in the database.

2.7.5.5 Users will check their email and confirm registration personally and manually.

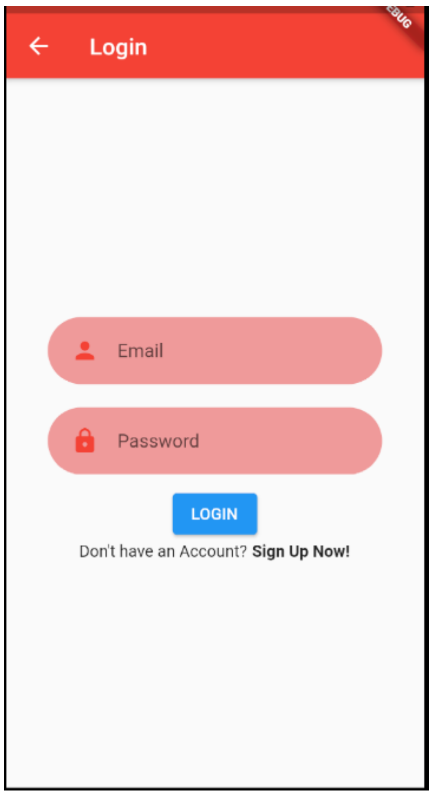
2.7.5.6 Users know how to use Google Maps for directions.

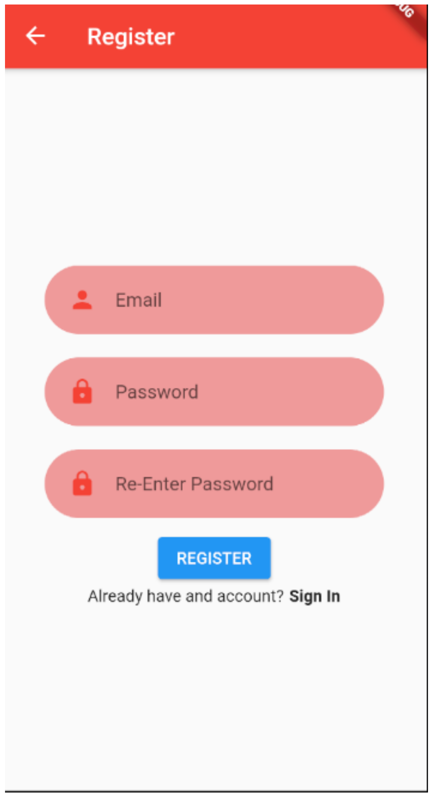
2.7.5.7 Users know the name of their destination.

3. External Interface Requirements

3.1 User Interfaces

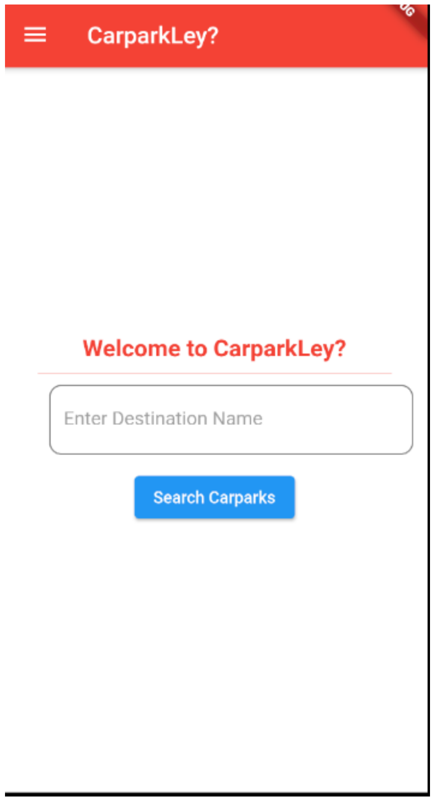
3.1.1 Login/Register Page





User has to log in with his email and password and be validated by the system to enter the app.

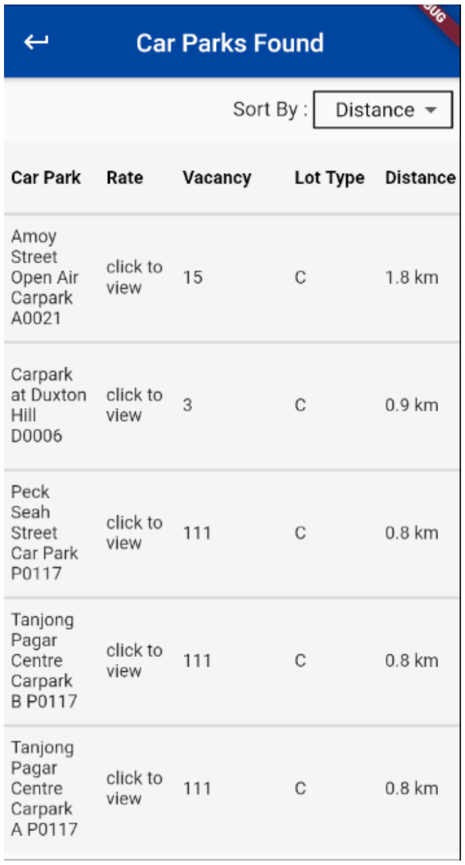
3.1.2 Home Page



Users can access various contents and features of this app through the main page once they have logged in.

- Navigation Menu: Users can access their favourites, settings and account.
- Favourites: Selecting favourites allows users to access their favourites list.
- Settings: Selecting settings allow them to change their vehicle type and font size.
- Input Destination: The input box utilises Google Maps function to search for the destination and will be linked to our other APIs to locate the nearby carparks.
- Sign Out: Users can log out of their account.

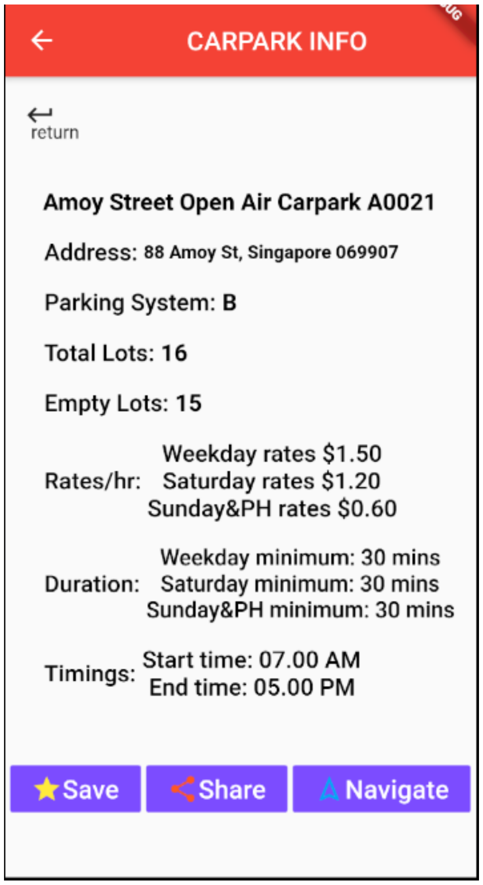
3.1.3 Carpark Results Page



Car Park	Rate	Vacancy	Lot Type	Distance
Amoy Street Open Air Carpark A0021	click to view	15	C	1.8 km
Carpark at Duxton Hill D0006	click to view	3	C	0.9 km
Peck Seah Street Car Park P0117	click to view	111	C	0.8 km
Tanjong Pagar Centre Carpark B P0117	click to view	111	C	0.8 km
Tanjong Pagar Centre Carpark A P0117	click to view	111	C	0.8 km

Users can see the list of carparks that is generated after searching for the carparks at their destination. They are able to sort the carparks by distance, number of empty lots or rates.

3.1.4 Carpark Information Page



Amoy Street Open Air Carpark A0021

Address: 88 Amoy St, Singapore 069907

Parking System: B

Total Lots: 16

Empty Lots: 15

Rates/hr: Weekday rates $1.50
Saturday rates $1.20
Sunday&PH rates $0.60

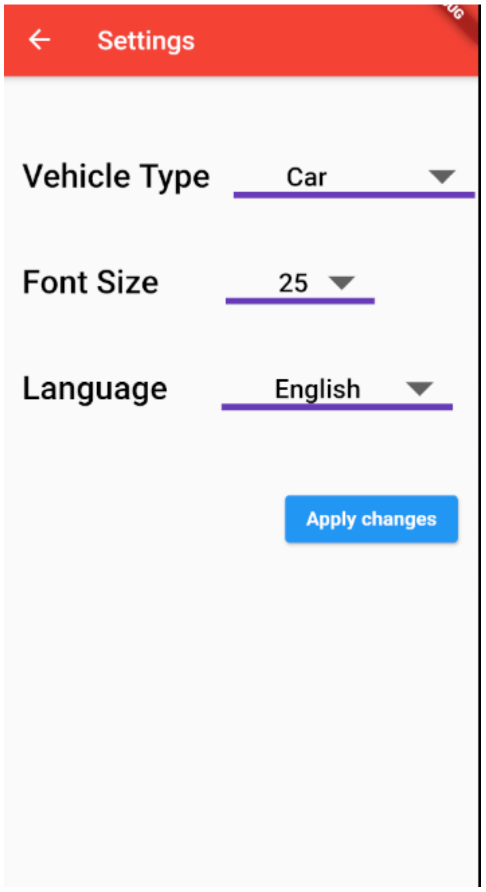
Duration: Weekday minimum: 30 mins
Saturday minimum: 30 mins
Sunday&PH minimum: 30 mins

Timings: Start time: 07.00 AM
End time: 05.00 PM

Users can view the information of the carparks they select to know more about it such as their full rates and pricing system.

- Save: Users can save the selected carpark into their favourite list so that they can access it without having to search the destination again.
- Share: Users can share the selected carpark via messaging to other users.
- Navigate: Users will be redirected to Google Maps where it will direct the user to the selected carpark from their current location.

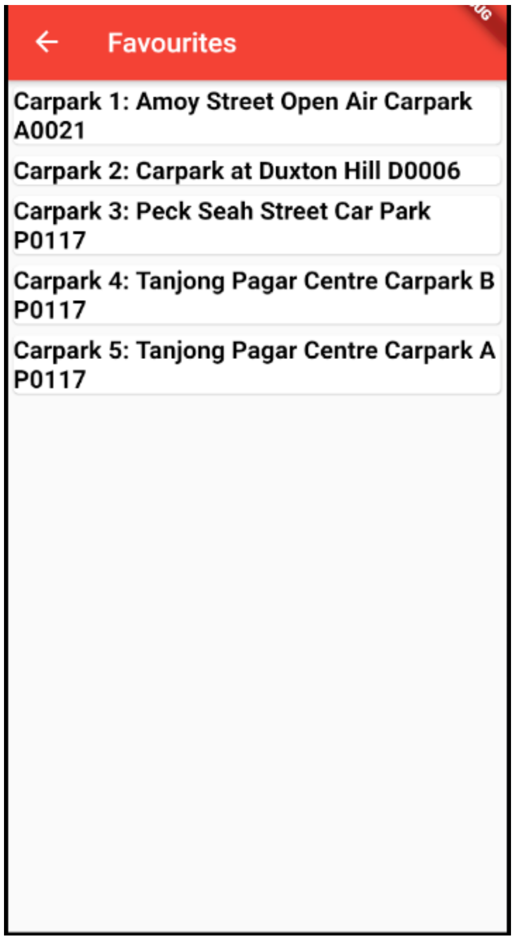
3.1.5 Settings Page



Vehicle Type Car
Font Size 25
Language English

Users can change their vehicle type, system font size and system language.

3.1.6 Favourites Page



Carpark 1: Amoy Street Open Air Carpark A0021

Carpark 2: Carpark at Duxton Hill D0006

Carpark 3: Peck Seah Street Car Park P0117

Carpark 4: Tanjong Pagar Centre Carpark B P0117

Carpark 5: Tanjong Pagar Centre Carpark A P0117

Users can access their saved carparks, but they can only access it when they are logged in to the app.

3.2 Hardware Interfaces

3.2.1 CarparkLEY Application

The application requires a mobile device that supports Android or iOS with internet connection capability.

3.3 Software Interfaces

3.3.1 CarparkLEY Application

- The application runs on an Android or iOS mobile device.
- The application will be searching for the carparks near the users' destination using numerous APIs.

- The application will also be collecting the users' saved carparks and uploading them into the Firebase. The application will then collect the data from the Firebase and display it in the Favourites Page for the users.

3.4 Communications Interfaces

HTTPS Communication

- HTTP protocol encrypted by TLS which provides secure communication over the network when retrieving real-time information using APIs, as well as the associated web server the client is communicating thus helping to guard against the man-in-the-middle attack.
- In addition, it also provides bidirectional encryption of communications between the client and server and helps guard against any eavesdropping attempt or tampering of data.

Location Service

- Users must enable location services in order to utilise the full features of the app like Navigating to carpark.

4. Functional Requirements

4.1 Main Interface

4.1.1 The system must check that the user has set up the application - Enabled Location Tracking, Set vehicle type and Logged in

4.1.1.1 The system must check if the user has enabled Location Tracking

4.1.1.1.1 The system must ask for permission to access user location if Location Tracking is not enabled

4.1.1.2 The system must check if the user has set vehicle type

4.1.1.2.1 The system must redirect the user to Settings to set vehicle type if the user has not set Vehicle Type

4.1.1.3 The system must prompt the user to log in if the user is not logged in

4.1.2 The system must generate a list of car park locations according to user input

4.1.2.1 The system must allow the user to fill in his/her desired destination

4.1.2.1.1 The system must be able to validate if the destination exists

4.1.2.1.2 The system must display an error message if validation fails

4.1.2.2 The system must allow the user to indicate the order to sort the car parks to be shown first in the generated list - Distance from Destination, Rates and Number of Empty Lots

4.1.2.2.1 The default option for sort will be the shortest Distance from Destination first

4.1.2.3 After the user confirms his/her choice by pressing the “Search” button, the system must redirect the user to a list of available car parks in the order indicated by the user after car parks are found

4.1.2.4 The system must notify the user if all car parks are full with a pop-up

4.1.3 The system must show the Name of the car park, Distance from Destination, Rates and Number of Empty Lots for each carpark, in a row

4.1.4 The system must display the car park information when the user selects a specific car park from the list

4.1.4.1 The car park information must have the following components - Address, Available Lots, Total Lots, Parking System, Full Parking Rates information, Navigate button, Share button and Favourite button

4.1.4.1.1 The system must redirect the user to Google Maps when the user clicks on the Navigate button.

4.1.4.1.1.1 Google Maps must navigate the user to the selected car park

4.1.4.1.1.2 The system must notify the user if the car park becomes full during navigation with a notification

4.1.4.1.1.3 The system must notify the user if the originally full car park becomes empty during navigation with a notification

4.1.4.1.2 The system must allow the user to share the following information of the car park when the user clicks on the Share button - Name of the car park and address, Available Lots and Total Lots at the time of sharing, Full Parking Rates information, Google Maps link of the car park location

4.1.4.1.2.1 The system must check if the user is logged in before allowing the user to share the car park information

4.1.4.1.2.2 The system must generate a message containing the car park information for the user

4.1.4.1.2.3 The system must allow the user to choose the platform the user wants to share the message to

4.1.4.1.2.4 The system must redirect the user to the chosen platform to share the message

4.1.4.1.3 The system must allow the user to save the car park into the user’s Favourites when the user clicks on the Favourites button.

4.1.4.1.3.1 The system must check if the user is logged in before allowing the user to save the car park into his/her Favourites

4.1.4.1.3.2 The system must save the car park into a database under the user's account

4.2 Login

4.2.1 The system must contain a link to redirect the user to the Registration page

4.2.2 The system must be able to validate the email and password text fields

- 4.2.2.1 The system must be able to validate if the email text field is filled
- 4.2.2.2 The system must be able to validate if the password text field is filled
- 4.2.2.3 The system must display a corresponding message when any of the above validation fails

4.2.3 The system must validate the credentials provided by the user before logging in

- 4.2.3.1 The system must display an error message when the email provided is not valid
- 4.2.3.2 The system must display an error message when the email provided is not registered
- 4.2.3.3 The system must display an error message when the password does not match with the user in the database
- 4.2.3.4 The system must display an error message when the account has not been activated

4.2.4 The system must redirect the user to the main interface after successful login

- 4.2.4.1 The system must load in User Vehicle Type and Favourite List from the database to the application

4.3 Registration

4.3.1 The system must allow the user to create an account

- 4.3.1.1 The system must allow the user to fill in his/her email address, password and confirmed password
  - 4.3.1.1.1 The system must validate that all the text fields are filled up
  - 4.3.1.1.2 The system must validate that the email address provided by the user is valid
  - 4.3.1.1.3 The system must validate that the email address has not already been registered
  - 4.3.1.1.4 The system must validate that the password contains at least 8 characters
  - 4.3.1.1.5 The system must validate that the confirmed password matches the password
  - 4.3.1.1.6 The system must add the user's email and password to the database
  - 4.3.1.1.7 The system must set the user account activation as false
- 4.3.1.2 The system must send a confirmation pin to the user's email to authenticate account creation
  - 4.3.1.2.1 The system must allow the user to input the confirmation pin
  - 4.3.1.2.2 The system must validate the confirmation pin

4.3.1.2.3 The system must set the user account activation as true

4.3.1.3 The system must redirect the user to the main interface upon successful account creation

4.3.1.4 The system must prompt the user to set his vehicle type

4.4 Favourites

4.4.1 The system must check if the user has logged in before allowing user to access his/her Favourites

4.4.2 The system must be able to access the database and retrieve a list of car parks that the user has favourited

4.4.3 The system must be able to generate a list of favourite car parks of the user

4.4.4 The system must allow the user to select the car park from the list and redirect them to the car park information in 4.1.4

4.4.5 The system must allow user to remove car park from Favourites list

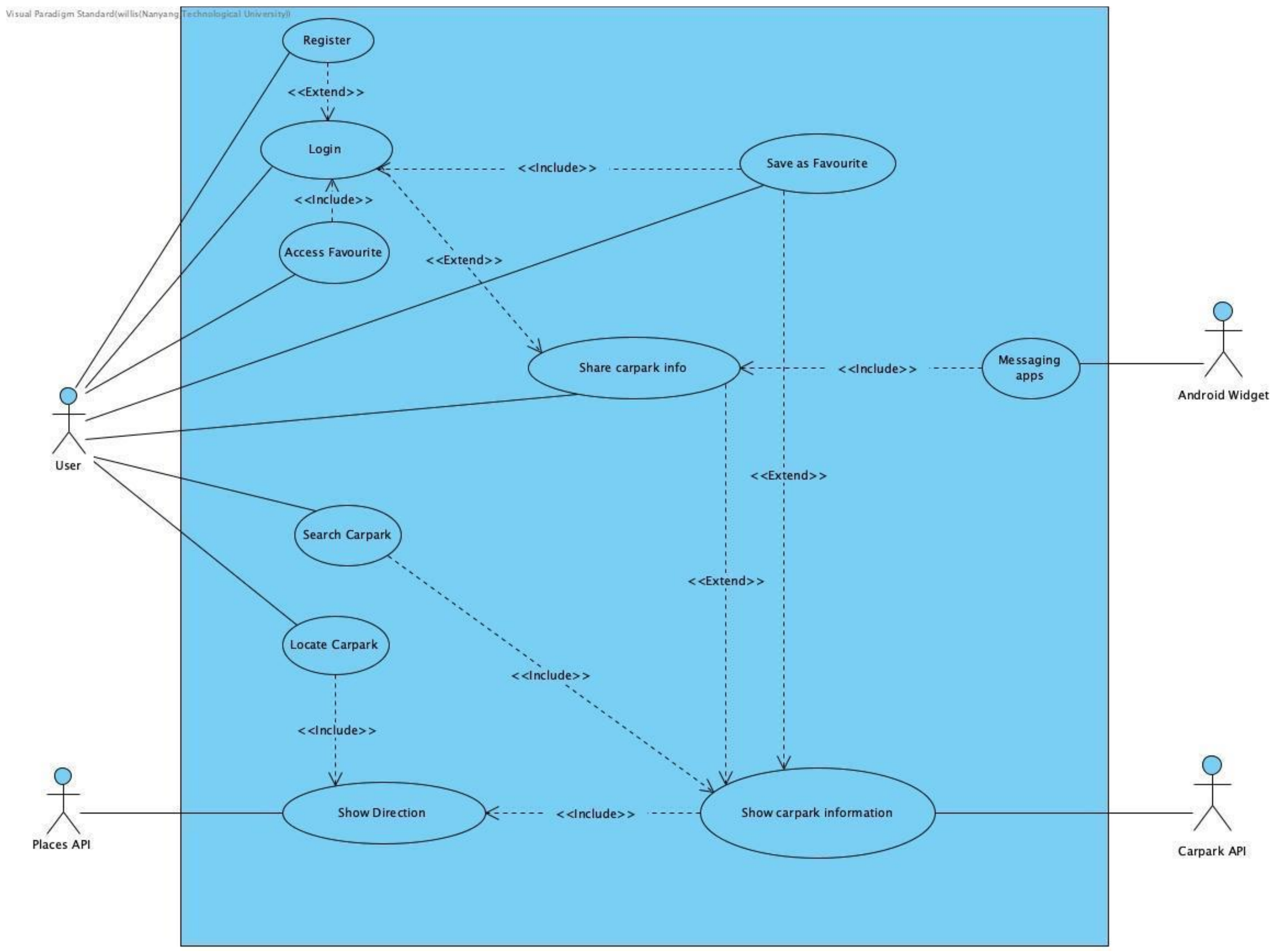
4.5 Settings

4.5.2 The system must allow the user to select vehicle types: Car, Motorbike or Heavy Vehicle

Future implementation:

- The system must allow the user to change to one of the following languages - English, Chinese, Malay or Tamil
- The system must allow the user to change the font size displayed in the application
- The system must contain a Help section that contains common issues faced by users and suggestion to rectify the issue

4.6 Use case diagram


Visual Paradigm Standard(willis(Nanyang Technological University))
Register
<<Extend>>
Login
<<Include>>
Save as Favourite
<<Include>>
Access Favourite
<<Extend>>
Share carpark info
<<Include>>
Messaging apps
Android Widget
User
<<Extend>>
Search Carpark
<<Extend>>
Locate Carpark
<<Include>>
<<Include>>
Places API
Show Direction
<<Include>>
Show carpark information
Carpark API

4.7 Use Case Descriptions

Use Case ID:	#1		
Use Case Name:	Register		
Created By:	Glenn	Last Updated By:	Xianan
Date Created:	9/2/21	Date Last Updated:	17/4/21

Actor:	User (Initiating Actor)
Description:	The user can register an account in order to save carpark as favourites or to share carpark information and directions with people in his contacts list
Preconditions:	1. The user must have the application open
Postconditions:	The user's account that was created must be saved in a database for future logins by the user.
Priority:	Medium
Frequency of use:	Moderate
Flow of events:	1. The user enters his email address 2. The user enters his password 3. The user enters his password again 4. The application will inform the user that an account has been created 5. A confirmation pin email will be sent to user 6. The user enters the confirmation pin 7. the application informs user that account is activated
Alternative flows:	If user enters a password that is less than 8 characters long: 1. Application will inform that the inputted password is too short

	If user enters a email address that is already registered in the application: 1. Application will inform that the email address is already registered If user enters a email address that is invalid: 1. Application will inform that the email address is not valid If user entered confirmation pin incorrectly: 1. Application will inform that the user that the pin is incorrect, and to try again
Exceptions:	If user enters the pin incorrectly 3 times: 1. A new pin will be sent to the user's email address 2. User repeats step 5
Includes:	N/A
Special Requirements:	1. Database can only store hashed password
Assumptions:	N/A
Notes and issues:	

Use Case ID:	#2		
Use Case Name:	Login		
Created By:	Glenn	Last Updated By:	
Date Created:	9/2/21	Date Last Updated:	

Actor:	User(Initiating Actor)

Description:	The user can login to his account to save a carpark as favourites or to share carpark information and directions with people in his contacts list
Preconditions:	1. The user must have the application open 2. The user must have an existing account in the application
Postconditions:	The application will inform user that login is successful
Priority:	Medium
Frequency of use:	Moderate
Flow of events:	1. User enters his email address 2. User enters his password 3. Application informs user that he is successfully logged in
Alternative flows:	If the user enters an incorrect email: 1. Application will inform that the email address is not registered If user enters an incorrect password: 1. Application will inform user that the inputted password is incorrect, and to try again
Exceptions:	N/A
Includes:	N/A
Special Requirements:	N/A
Assumptions:	N/A
Notes and issues:	

Use Case ID:	#3
Use Case Name:	Access favourites

Created By:	Glenn	Last Updated By:	Xianan
Date Created:	9/2/21	Date Last Updated:	17/4/21

Actor:	User(Initiating Actor)
Description:	The user can access his caparks which he had saved as favourites. The 'favourites' feature allows users to view carpark information faster as well as directions to the carpark, with greater convenience as compared to manually searching the carpark in the home page.
Preconditions:	1. The user must have the application open 2. The user must have an existing account in the application 3. The user must be logged in to his account
Postconditions:	The application will inform user that login is successful
Priority:	Medium
Frequency of use:	Moderate
Flow of events:	1. Application checks if user is logged in the application 2. User views his favourite carparks
Alternative flows:	If user is not logged in: 1. Application will inform user that he is required to log in
Exceptions:	If user does not have an account: 1. Application will inform user that he needs to have an account in order to access his favourite carparks
Includes:	#2 Login
Special Requirements:	N/A
Assumptions:	N/A
Notes and issues:	

Use Case ID:	#4		
Use Case Name:	Search carparks		
Created By:	Glenn	Last Updated By:	Xianan
Date Created:	9/2/21	Date Last Updated:	17/4/21

Actor:	User(Initiating Actor)
Description:	The user can search for carparks near his destination that are within 2km. The user will be shown the rates(for first hour parking), number of empty lots, and distance from the destination for each carpark. User will be able to sort the results.
Preconditions:	1. The user must have the application open 2. The user must have inputted his destination
Postconditions:	The list of carparks within 2km from user's destination is shown(nearest carparks are shown at the top)
Priority:	High
Frequency of use:	High
Flow of events:	1. User inputs his destination, and his choice of sorting 2. Application shows carparks within 2km from user's destination 3. User sort by number of empty lots/distance from destination
Alternative flows:	N/A
Exceptions:	N/A

Includes:	#7 Input destination
Special Requirements:	N/A
Assumptions:	N/A
Notes and issues:	

Use Case ID:	#5		
Use Case Name:	Show carpark information		
Created By:	Glenn	Last Updated By:	Xianan
Date Created:	9/2/21	Date Last Updated:	17/4/21

Actor:	User(Initiating Actor)
Description:	The application will show a list of carparks within 2km from user's destination. The list of carparks will show(for each carpark): 1. The name of carpark 2. The empty of empty lots out of the total of available lots 3. The distance from the destination From the list of carparks, user can click on a carpark to: 1. View parking rates a. Rates for weekdays, weekends and public holidays b. The first hour rate, subsequent rates 2. View carpark parking system a. Electronic/coupon

	3. View Address of carpark 4. View empty lots out of available lots 5. Share carpark information with someone in their contacts list 6. Save carpark as 'favourites'
Preconditions:	1. The user must have the application open 2. The user must have an existing account in the application 3. The user must be logged into the application 4. The user must have inputted a destination
Postconditions:	N/A
Priority:	Medium
Frequency of use:	Moderate
Flow of events:	1. Application shows list of carparks within 2km from user's destination 2. User clicks on one of the carpark 3. User views full details of the parking rates of that carpark 4. User can share carpark information, or save carpark as 'favourites'
Alternative flows:	N/A
Exceptions:	N/A
Includes:	N/A
Special Requirements:	N/A
Assumptions:	N/A
Notes and issues:	

Use Case ID:	#6

Use Case Name:	Locate Carpark		
Created By:	Paulani	Last Updated By:	
Date Created:	9/2/21	Date Last Updated:	

Actor:	User (initiating actor)
Description:	System locates carpark that user has chosen
Preconditions:	1. Mobile must be connected to wifi/mobile data
Postconditions:	The user will be redirected to Google maps
Priority:	High
Frequency of use:	High
Flow of events:	1. User must click on navigate button in the car park information page 2. System must redirect the user to google maps for directions
Alternative flows:	N/A
Exceptions:	N/A
Includes:	#5 Show direction
Special Requirements:	N/A
Assumptions:	N/A
Notes and issues:	

Use Case ID:	#7		
Use Case Name:	Show direction		
Created By:	Glenn	Last Updated By:	
Date Created:	9/2/21	Date Last Updated:	

Actor:	Google maps(Initiating Actor)
Description:	Google maps will show the route to the carpark that the user wants to reach
Preconditions:	1. The user must have the application open 2. The user must have entered his destination 3. The user must have selected a carpark from the list of carparks to view
Postconditions:	N/A
Priority:	High
Frequency of use:	High
Flow of events:	1. Google maps will show the route and guide user to his destination carpark
Alternative flows:	N/A
Exceptions:	N/A
Includes:	#9 Show carpark information
Special Requirements:	N/A

Assumptions:	N/A
Notes and issues:	

Use Case ID:	#8		
Use Case Name:	Share carpark information		
Created By:	Glenn	Last Updated By:	
Date Created:	9/2/21	Date Last Updated:	

Actor:	User(Initiating Actor)
Description:	The user can share carpark information - Name of the car park and address, Available Lots and Total Lots at the time of sharing, Full Parking Rates information, Google Maps link of the car park location, with someone from their contacts list
Preconditions:	1. The user must have the application open
Postconditions:	The receiver in their contacts list will receive a message containing the carpark information(in the description)
Priority:	Medium
Frequency of use:	Moderate
Flow of events:	1. User clicks ‘share carpark info’ 2. User selects messaging platform 3. User selects someone in their contacts list 4. Application will inform the user ‘successfully shared’

Alternative flows:	N/A
Exceptions:	N/A
Includes:	N/A
Special Requirements:	N/A
Assumptions:	N/A
Notes and issues:	

Use Case ID:	#9		
Use Case Name:	Save as favourites		
Created By:	Glenn	Last Updated By:	Xianan
Date Created:	9/2/21	Date Last Updated:	17/4/21

Actor:	User(Initiating Actor)
Description:	The user can save a carpark as ‘favourites’. The ‘favourites’ feature allows users to view carpark information faster as well as directions to the carpark, with greater convenience as compared to manually searching the carpark in the home page. User can sort favourites by: “most frequent” and “most recent”.
Preconditions:	1. The user must have the application open 2. The user must have an existing account in the application 3. The user must be logged into the application 4. The user must have input a destination 5. The user must have viewed a carpark information

Postconditions:	The carpark will be saved as 'favourites' under the user's account and the user will be able to access it under his settings. This information will be saved in the application database.
Priority:	Medium
Frequency of use:	Moderate
Flow of events:	1. User clicks 'save as favourites' 2. Application will save the carpark name into the database 3. Application will inform user that capark information has been successfully saved
Alternative flows:	If carpark has been saved as favourites: 1. Users can remove it from favourites anytime.
Exceptions:	N/A
Includes:	N/A
Special Requirements:	N/A
Assumptions:	N/A
Notes and issues:	

4.7 Data Dictionary

Term	Definition
Destination	The location that the user is heading to.
Rate	The fee for parking at the carpark. Depending on the carpark, it can be charged hourly, half-hourly etc. In our app, we will show the first hour of the fee for the list for simplicity purposes.
Distance	How far the carpark is from the destination.
Vacancy	The number of empty lots in the carpark.
Vehicle Type	The type of vehicle that the user is driving. There are 3 types: 1. Motor Car

	2. Motor bike 3. Heavy vehicle
Carpark List	Shows the available carparks within a 2km radius from destination. User can sort it based on distance, rate and vacancies.
User	Driver
Carpark Information	Detailed information of the carpark, showing the full rates list, instead of the simple rate information shown in the carpark list, and other information like address, type of parking system.
Favourites	A list where the user can save the carparks and access it easily.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

5.1.1. The system must open Google Map when the user clicks the Navigate button on the Car Park Information screen within 3 seconds.
5.1.2. The system must not crash at any process during the app's lifetime.

5.2 Safety Requirements

5.2.1. When the user is inputting a destination, a message notification will be shown to remind drivers to not use their phone while driving to prevent any accidents.

5.3 Security Requirements

5.3.1. The system will only store hashed passwords into the database to prevent hacking attempts.
5.3.2. On login, the system will compare the salt-hashed version of the customer's password with the hashed password stored in the database and will allow entry only if they match.
5.3.3. System will mask the password to prevent onlookers from stealing the password.

5.4 Usability Requirements

5.4.1. System must strive for consistency.
    5.4.1.1. System must have a consistent sequence of actions for similar situations.
    5.4.1.2. System must have a consistent layout throughout the application (e.g. fonts, labels, colors).
5.4.2. System must offer informative feedback.

5.4.2.1. System is to display appropriate feedback when user input is invalid.
5.4.2.2. System is to display appropriate feedback when an exception occurs.

5.4.3. System must permit easy reversal of actions.
5.4.3.1. Users must be able to save and remove favourite carpark locations easily.

5.4.4. System must reduce short term memory load.
5.4.4.1. The interface design has to be simple and straightforward.
5.4.4.2. The user must be able to retrieve information on his favourite carpark location easily.

5.4.5. System must support internal locus of control.
5.4.5.1. Users must be able to terminate redirection from app to Google Maps.

5.5 Reliability Requirements

5.5.1. Data must be stored in the event that the system crashes or undergoes a system reboot.
5.5.1.1. When users re-enter the app, users do not have to re-enter the destination.
5.5.1.2. If the user saves the carpark as 'favourites' before the app crashes, the user must be able to access 'favourites' option once the app is running again and the carpark must still be there.

6. Other Requirements

6.1. Black box testing

6.1.1. Login

Test Case ID	Scenario	Expected Result	Actual Result
1	Login with valid email and password	The App displays home page for user to continue	The App displays home page for user to continue
2	Login with invalid email	The App prompts user to try again with appropriate feedback to write the email in the proper format.	The App prompts user to try again with appropriate feedback to write the email in the proper format.
3	Login with unregistered email	The App prompts user to try again with appropriate feedback that the email is not registered.	The App prompts user to try again with appropriate feedback that the email is not registered.
4	Login with wrong password	The App prompts user	The App prompts user to

		to try again with appropriate feedback that the password is wrong.	try again with appropriate feedback that the password is wrong.
5	Login without activating account	The App will prompt user to go to their email and verify their account	The App will prompt user to go to their email and verify their account

ID	Email	Password	Validate account	Expected Result	Actual Result
1	Test email	testpassword	true	Login successful	Login successful
2	Invalid email	Testpassword	true	User cannot login - error message	User cannot login - error message
3	unregistered email	testpassword	false	User cannot login - error message	User cannot login - error message
4	Test email	Wrong password	true	User cannot login - error message	User cannot login - error message
5	Test email	testpassword	false	User cannot login - error message	User cannot login - error message

6.1.2. Register

Test Case ID	Scenario	Expected Result	Actual Result
1	Register with valid email and password	The App displays validate email for user to activate account	The App displays validate email for user to activate account
2	Register with invalid email	he App prompts user to try again with appropriate feedback to write the email in the proper format	The App prompts user to try again with appropriate feedback to write the email in the proper format
3	Register with registered email	The App prompts user to try again with appropriate feedback	The App prompts user to try again with appropriate feedback

		that the email is already registered	that the email is already registered
4	Register with weak password	The App prompts user to try again with appropriate feedback to use a password with more than 6 characters	The App prompts user to try again with appropriate feedback to use a password with more than 6 characters

ID	Email	Password	Expected Result	Actual Result
1	Test email	Test password	Registration successful	Registration successful
2	Invalid email	Test password	User cannot register - error message	User cannot register - error message
3	registered email	Test password	User cannot register - error message	User cannot register - error message
4	Test email	Weak password	User cannot register - error message	User cannot register - error message

6.1.3. Search Destination

Test Case ID	Scenario	Expected Result	Actual Result
1	Valid destination with sufficient vacancy is searched	The App displays Carparks found	The App displays Carparks found
2	No carparks found	The App displays error message	The App displays error message

6.1.4. Saving Favourites

Test Case ID	Scenario	Expected Result	Actual Result

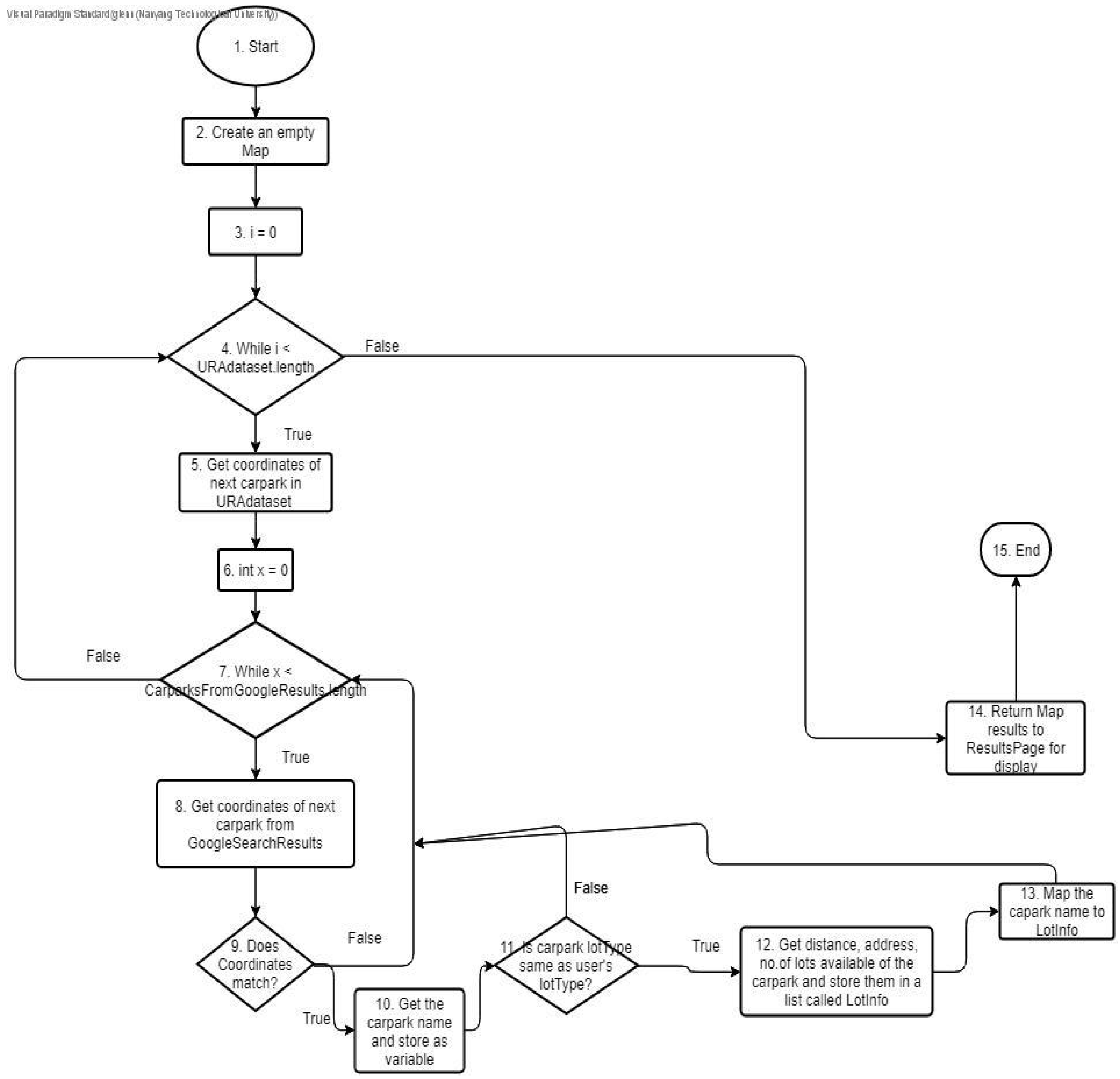
1	User is logged in and favourites a carpark	The App appends the favourited carpark to the favourites list	The App appends the favourited carpark to the favourites list
2	User is not logged in and favourites a carpark	The App prompts user to log in to access this feature	The App prompts user to log in to access this feature

6.1.5. Accessing Favourites

Test Case ID	Scenario	Expected Result	Actual Result
1	User is logged in and accesses favourites	The App displays all the carparks favourited	The App displays all the carparks favourited
2	User is not logged in and accesses favourites	The App prompts user to log in to access this feature	The App prompts user to log in to access this feature

6.1.2 White Box testing

6.1.2.1 locateCarparks() method in CarparkMgr


Visual Paradigm Standard(glen (Nanyang Technological University))
1. Start
2. Create an empty Map
3. i = 0
4. While i < URAdataset.length
False
True
5. Get coordinates of next carpark in URAdataset
6. int x = 0
7. While x < CarparksFromGoogleResults.length
False
True
8. Get coordinates of next carpark from GoogleSearchResults
9. Does Coordinates match?
False
True
10. Get the carpark name and store as variable
11. Is carpark lotType same as user's lotType?
False
True
12. Get distance, address, no.of lots available of the carpark and store them in a list called LotInfo
13. Map the capark name to LotInfo
14. Return Map results to ResultsPage for display
15. End

Cyclomatic Complexity(CC) = no. of binary decision points + 1

= 4 + 1 = 5

Hence no.of basis paths needed is 5

<u>Basis paths</u>:

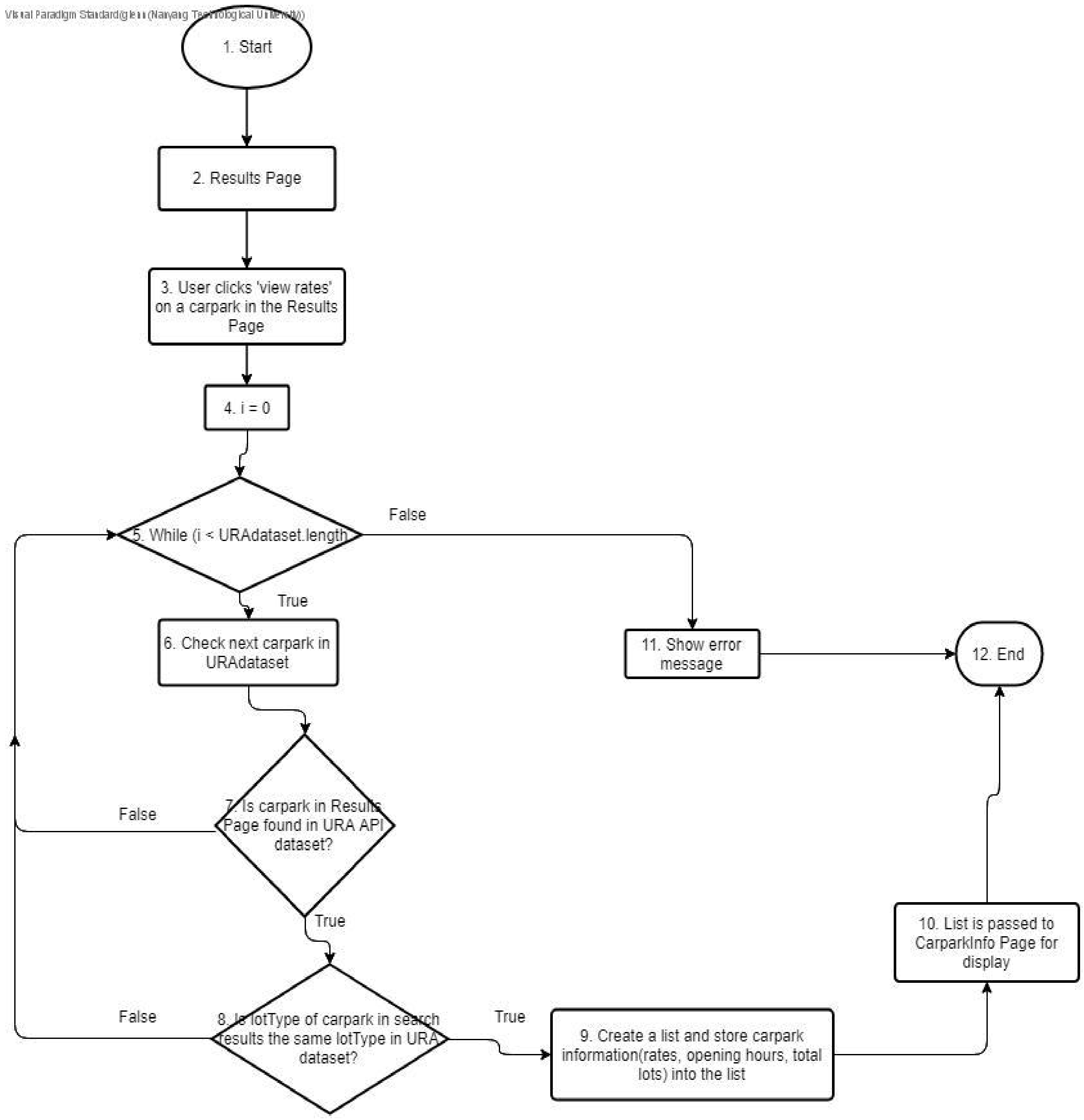
1. 1, 2, 3, 4, 14, 15(infeasible)
2. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 7, 4, 14, 15
3. 1, 2, 3, 4, 5, 6, 7, 8, 9, 7, 8, 9, 10, 11, 12, 13, 7, 4, 14, 15

4. 1, 2, 3, 4, 5, 6, 7, 8, 9, 7, 8, 9, 10, 11, 7, 8, 9, 10, 11, 12, 13, 7, 4, 14, 15
5. 1, 2, 3, 4, 5, 6, 7, 8, 9, 7, 4, 14, 15

Test cases

1. Infeasible for URAdataset to be empty at the start
2. The carpark in GoogleSearchResults is in URAdataset and the lotType is the same and there are no more iterations in GoogleSearchResults and URAdataset to be done.
3. The first carpark in GoogleSearchResults is not found in first iteration of URAdataset but the second carpark in GoogleSearchResults is found in the first iteration of URAdataset and their lotType is the same, and there are no more iterations in GoogleSearchResults and URAdataset to be done.
4. The first carpark in GoogleSearchResults is not found in the first iteration of URAdataset, the second is found in the first iteration of URAdataset, but their lotType is not the same. The third carpark in GoogleSearchResults is found in the first iteration of URAdataset and their lotType is the same, and there are no more iterations in GoogleSearchResults and URAdataset to be done.
5. The last carpark in GoogleSearchResults is not found in the last iteration of URAdataset

6.1.2.2 getCarpark_info() method in MainControlMgr


Visual Paradigm Standard(glenn(Nanyang Technological University))
1. Start
2. Results Page
3. User clicks 'view rates' on a carpark in the Results Page
4. i = 0
5. While (i < URAdataset.length
False
True
6. Check next carpark in URAdataset
11. Show error message
12. End
False
7. Is carpark in Results Page found in URA API dataset?
True
10. List is passed to CarparkInfo Page for display
False
8. Is lotType of carpark in search results the same lotType in URA dataset?
True
9. Create a list and store carpark information(rates, opening hours, total lots) into the list

Cyclomatic Complexity(CC) = no. of binary decision points + 1

= 3 + 1 = 4

Hence no.of basis paths needed is 4

<u>Basis paths</u>:

1. 1, 2, 3, 4, 5, 11(infeasible)
2. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12
3. 1, 2 ,3, 4, 5, 6, 7, 5, 6, 7, 8, 9, 10, 12

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

Test cases

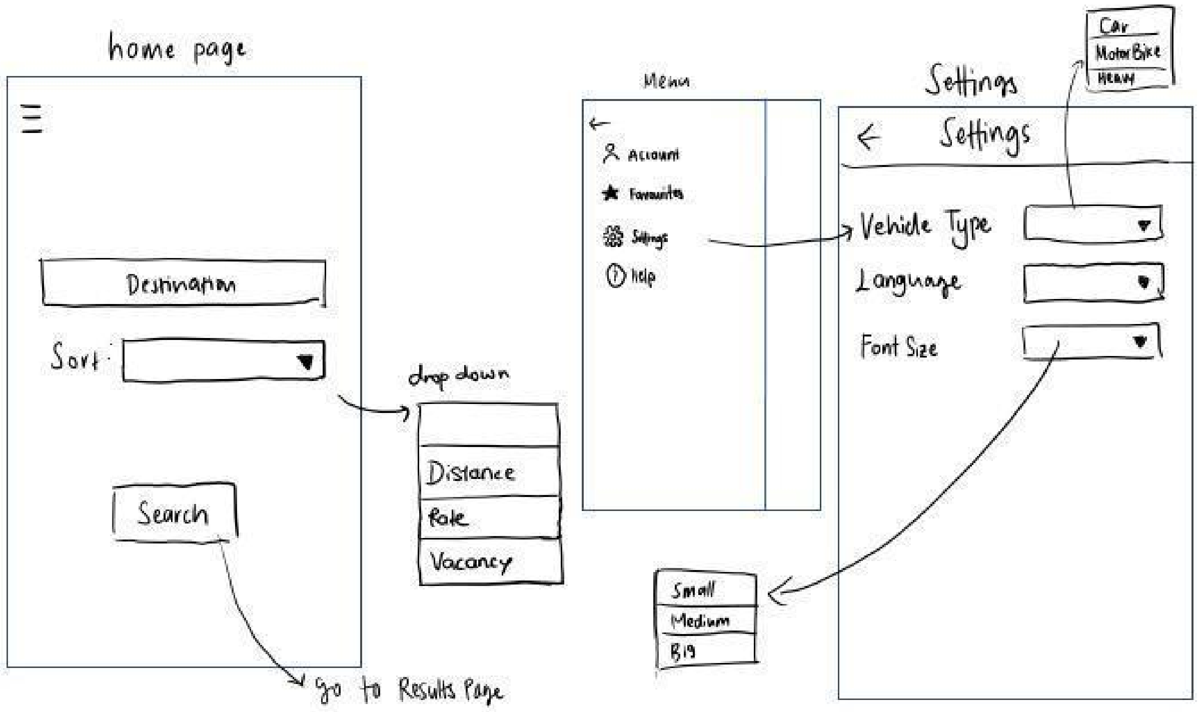
1. Unlikely for carpark to not be in URAdataset because carpark results were obtained from URAdataset
2. Carpark is found in URAdataset and the lotType is the same
3. The first carpark is not found in URAdataset while the second carpark is found in URAdataset and the lotType is the same
4. The first carpark is found in URAdataset but their lotType is not the same, the second carpark is found in URAdataset and the lotType is the same.

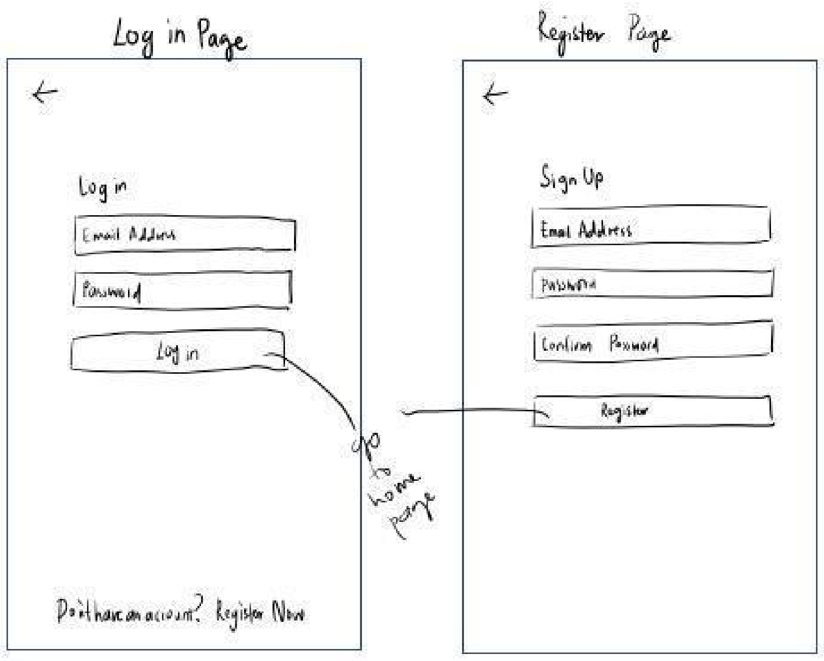
6.2 Software Design Principles and Patterns

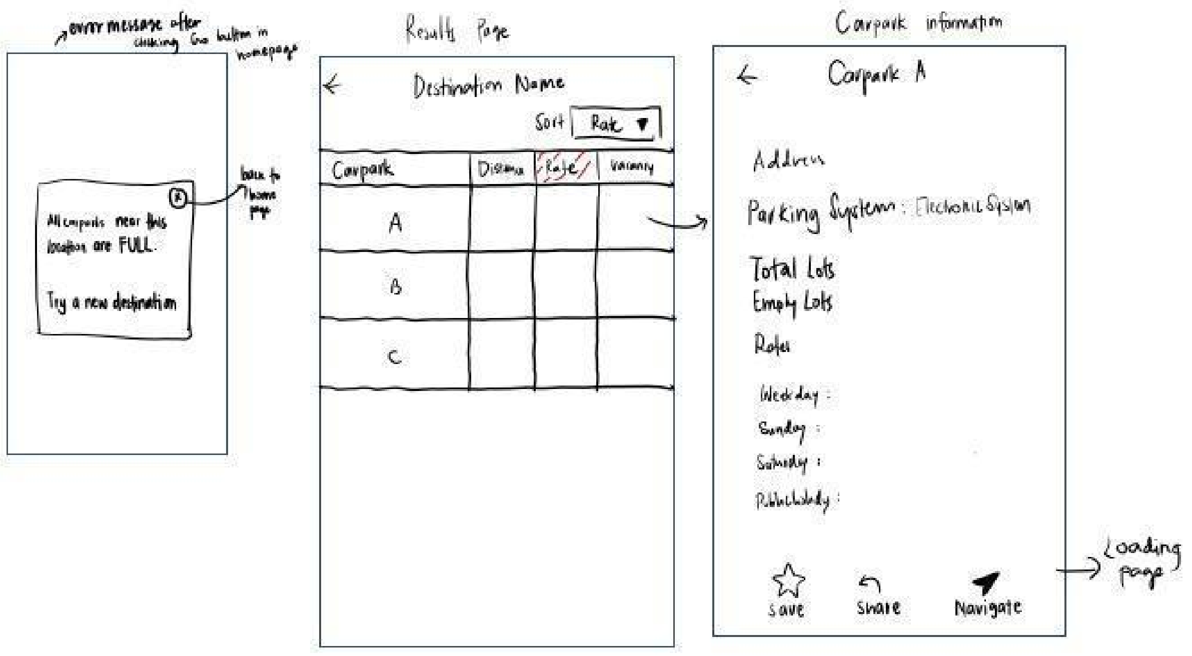
6.2.1 Facade Pattern

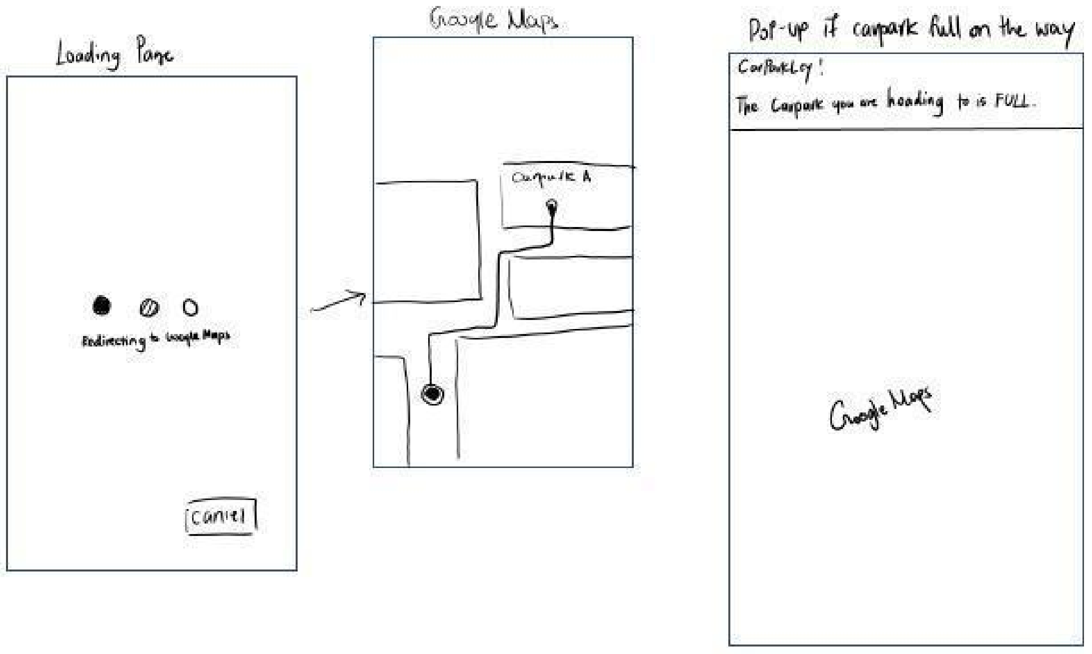
In our system, we have a main UI containing all the other UIs like Login, Register and GoogleMaps UI. The main UI is the constant screen that the user will always see, while the other interfaces satisfy specific needs of the user. Therefore we separated the three UIs and made them part of the main UI by composition. The main UI is loosely coupled with the 3 UIs and is open for extension if we want to add other UIs in the future as well. It will be closed for modification when we make changes to the other 3 UIs as the code in the main UI will not be affected by the changes.

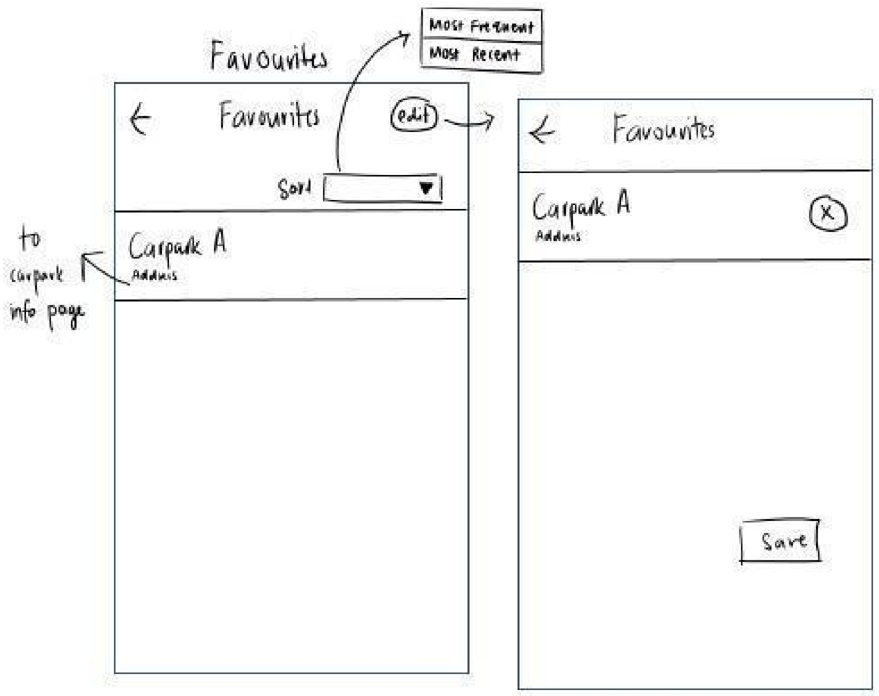
Appendix A: UI Mockup






error message after clicking Go button in homepage
All carparks near this location are FULL.
Try a new destination
back to home page
Results Page
Destination Name
Sort
Rate
Carpark
Distance
Rate
Vacancy
A
B
C
Carpark information
Carpark A
Address
Parking System : Electronic System
Total Lots
Empty Lots
Rates
Weekday :
Sunday :
Saturday :
Public Holiday :
Save
Share
Navigate
Loading page


Loading Page
Redirecting to Google Maps
Cancel
Google Maps
Carpark A
Pop-up if carpark full on the way
CarParkLoy !
The Carpark you are heading to is FULL.
Google Maps


Favourites
Most Frequent
Most Recent
Favourites
edit
Sort
to carpark info page
Carpark A
Address
Favourites
Carpark A
Address
X
Save

Appendix B: Analysis Models

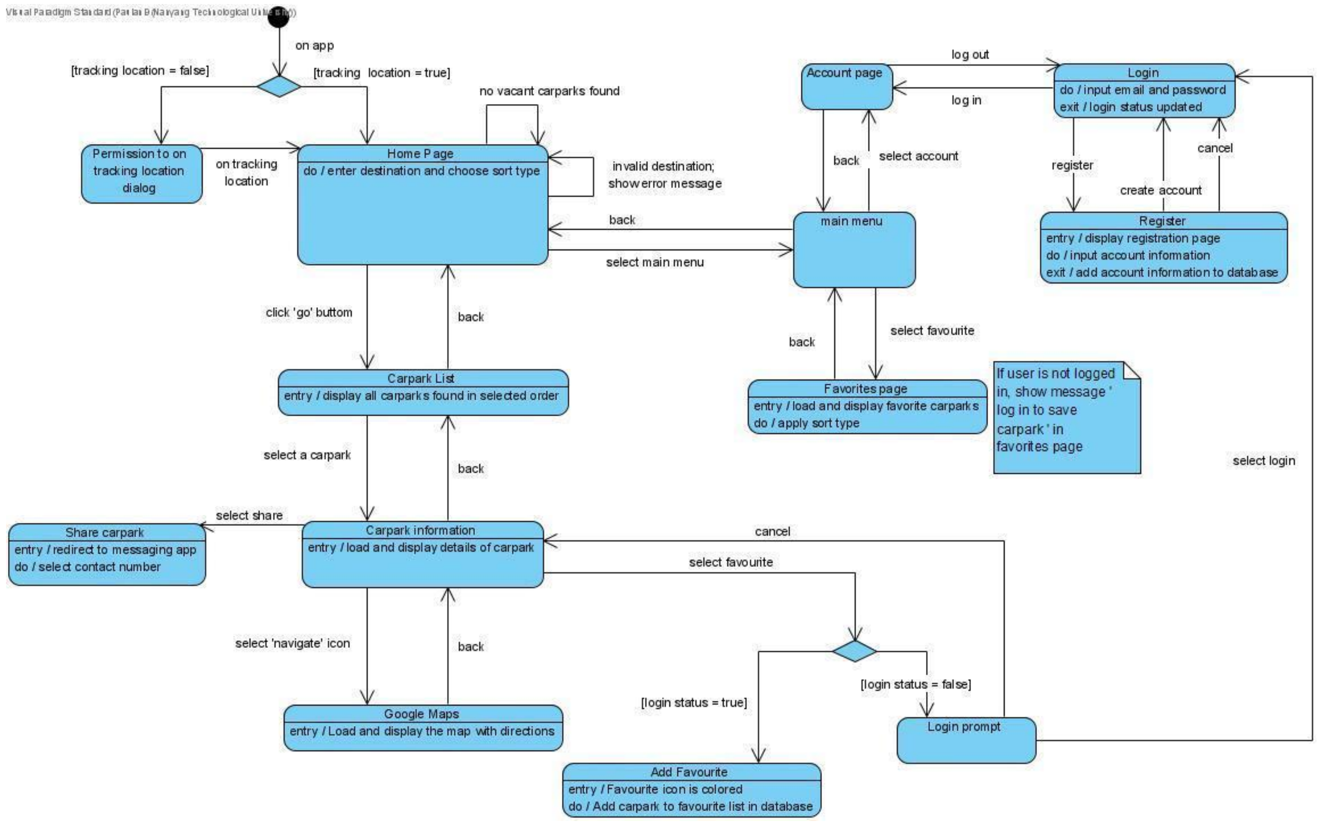

Visual Paradigm Standard (Pan Jiali B(Nanyang Technological University))
on app
[tracking location = false]
[tracking location = true]
Permission to on tracking location dialog
on tracking location
no vacant carparks found
Home Page
do / enter destination and choose sort type
invalid destination; show error message
back
select main menu
main menu
Account page
log out
log in
Login
do / input email and password
exit / login status updated
back
select account
register
create account
cancel
Register
entry / display registration page
do / input account information
exit / add account information to database
click 'go' buttom
back
Carpark List
entry / display all carparks found in selected order
back
select favourite
Favorites page
entry / load and display favorite carparks
do / apply sort type
If user is not logged in, show message ' log in to save carpark ' in favorites page
select login
select a carpark
back
select share
Share carpark
entry / redirect to messaging app
do / select contact number
Carpark information
entry / load and display details of carpark
cancel
select favourite
select 'navigate' icon
back
[login status = true]
[login status = false]
Google Maps
entry / Load and display the map with directions
Login prompt
Add Favourite
entry / Favourite icon is colored
do / Add carpark to favourite list in database

Figure 1 Dialog Map

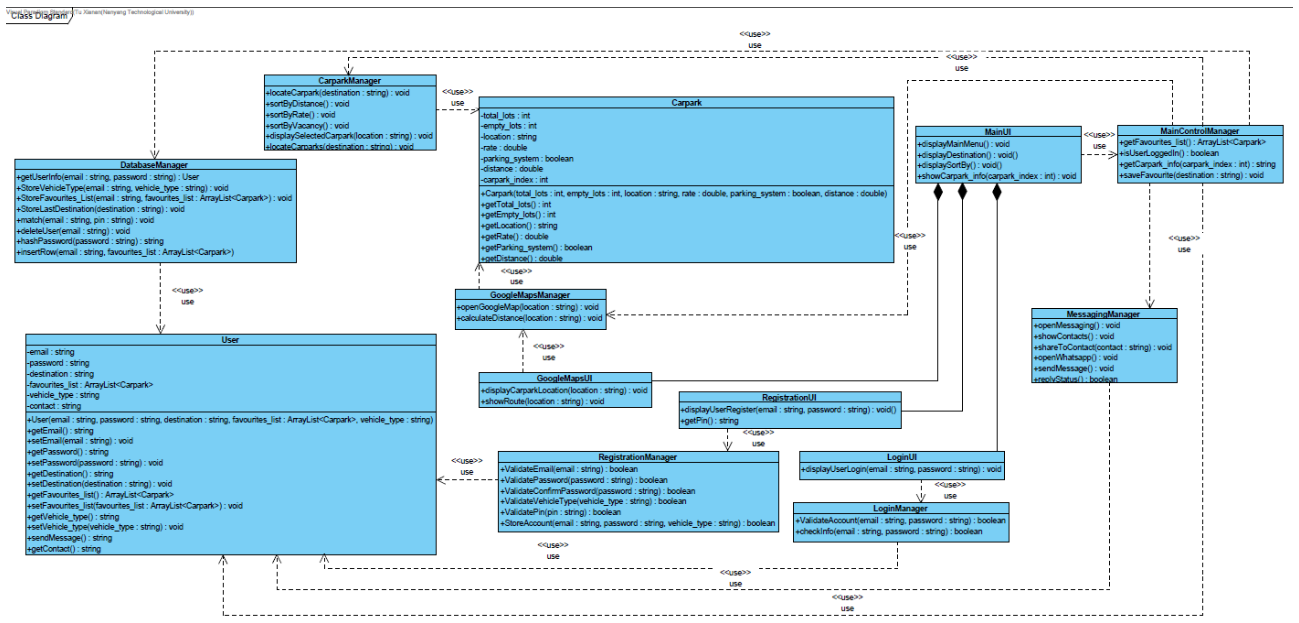

Visual Paradigm Standard(Tu Xianan(Nanyang Technological University))
Class Diagram
<<use>>
use
CarparkManager
+locateCarpark(destination : string) : void
+sortByDistance() : void
+sortByRate() : void
+sortByVacancy() : void
+displaySelectedCarpark(location : string) : void
+locateCarparks(destination : string) : void
Carpark
-total_lots : int
-empty_lots : int
-location : string
-rate : double
-parking_system : boolean
-distance : double
-carpark_index : int
+Carpark(total_lots : int, empty_lots : int, location : string, rate : double, parking_system : boolean, distance : double)
+getTotal_lots() : int
+getEmpty_lots() : int
+getLocation() : string
+getRate() : double
+getParking_system() : boolean
+getDistance() : double
MainUI
+displayMainMenu() : void
+displayDestination() : void()
+displaySortBy() : void()
+showCarpark_info(carpark_index : int) : void
MainControlManager
+getFavourites_list() : ArrayList<Carpark>
+isUserLoggedIn() : boolean
+getCarpark_info(carpark_index : int) : string
+saveFavourite(destination : string) : void
DatabaseManager
+getUserInfo(email : string, password : string) : User
+StoreVehicleType(email : string, vehicle_type : string) : void
+StoreFavourites_List(email : string, favourites_list : ArrayList<Carpark>) : void
+StoreLastDestination(destination : string) : void
+match(email : string, pin : string) : void
+deleteUser(email : string) : void
+hashPassword(password : string) : string
+insertRow(email : string, favourites_list : ArrayList<Carpark>)
GoogleMapsManager
+openGoogleMap(location : string) : void
+calculateDistance(location : string) : void
MessagingManager
+openMessaging() : void
+showContacts() : void
+shareToContact(contact : string) : void
+openWhatsapp() : void
+sendMessage() : void
+replyStatus() : boolean
User
-email : string
-password : string
-destination : string
-favourites_list : ArrayList<Carpark>
-vehicle_type : string
-contact : string
+User(email : string, password : string, destination : string, favourites_list : ArrayList<Carpark>, vehicle_type : string)
+getEmail() : string
+setEmail(email : string) : void
+getPassword() : string
+setPassword(password : string) : void
+getDestination() : string
+setDestination(destination : string) : void
+getFavourites_list() : ArrayList<Carpark>
+setFavourites_list(favourites_list : ArrayList<Carpark>) : void
+getVehicle_type() : string
+setVehicle_type(vehicle_type : string) : void
+sendMessage() : string
+getContact() : string
GoogleMapsUI
+displayCarparkLocation(location : string) : void
+showRoute(location : string) : void
RegistrationUI
+displayUserRegister(email : string, password : string) : void()
+getPin() : string
RegistrationManager
+ValidateEmail(email : string) : boolean
+ValidatePassword(password : string) : boolean
+ValidateConfirmPassword(password : string) : boolean
+ValidateVehicleType(vehicle_type : string) : boolean
+ValidatePin(pin : string) : boolean
+StoreAccount(email : string, password : string, vehicle_type : string) : boolean
LoginUI
+displayUserLogin(email : string, password : string) : void
LoginManager
+ValidateAccount(email : string, password : string) : boolean
+checkInfo(email : string, password : string) : boolean

Figure 2 Class Diagram

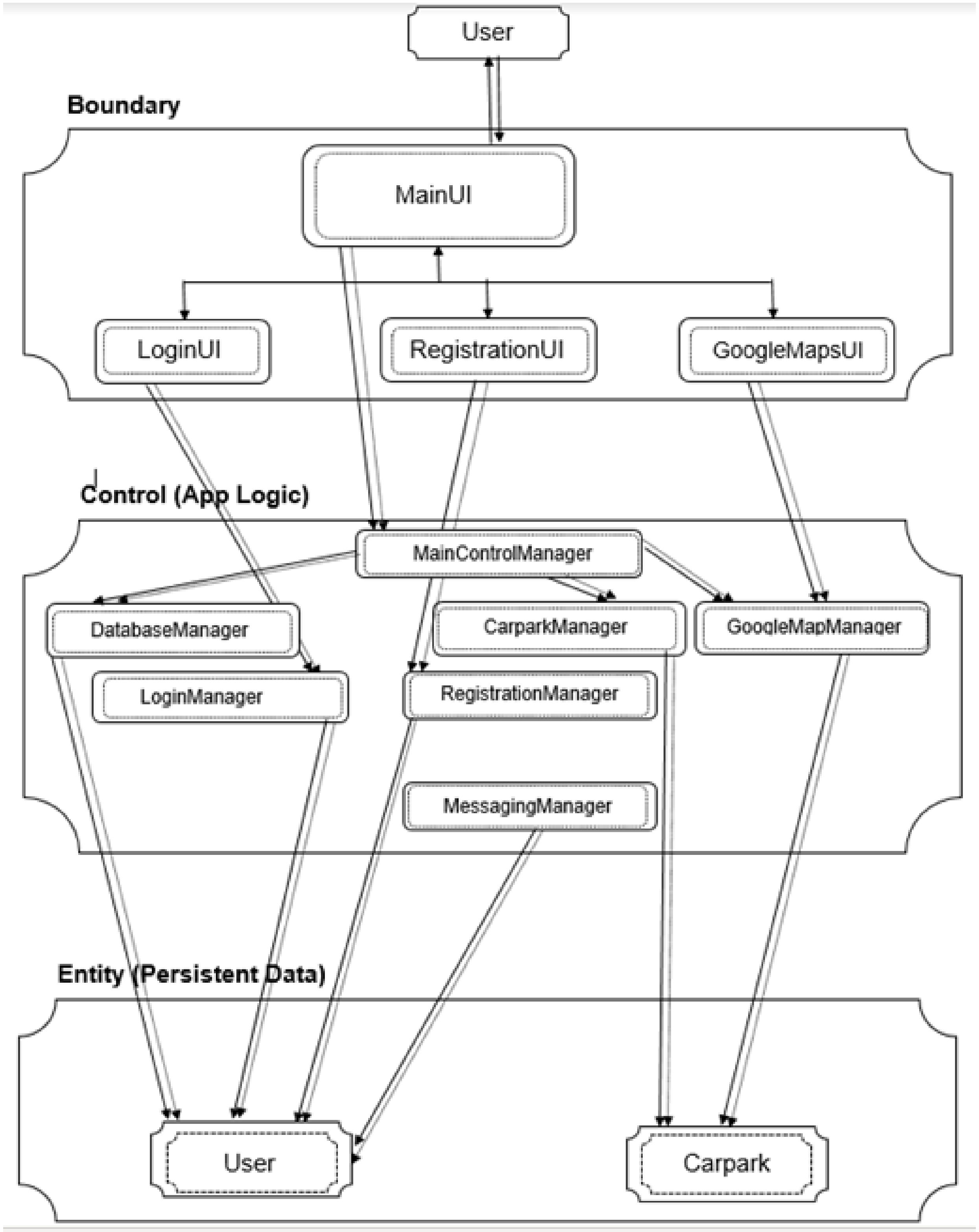


Figure 3 System Architecture

We have adopted the 3-Layered Architecture in our design where we grouped our components according to their class stereotypes.

In the first boundary or presentation layer, we have the Main UI which the user will see the moment they open the app. Depending on the functionalities that the user requires, the Main UI may instantiate other UIs like the Login or Registration UI. In order to satisfy the various requests issued by the user, the UIs will pass the request to the next layer, which is the Control or App Logic layer.

The various manager classes in this layer will access information from various APIs or databases, process the data if needed, and pass the output back to the Presentation layer for the UI to display to the user. In the bottom most layer, we have the entity or the persistent data layer. This is where raw data of the different users and car parks will be stored. Objects in these layers are stored in our databases or APIs and the Managers classes in the Control layer will be able to request for these data when needed.

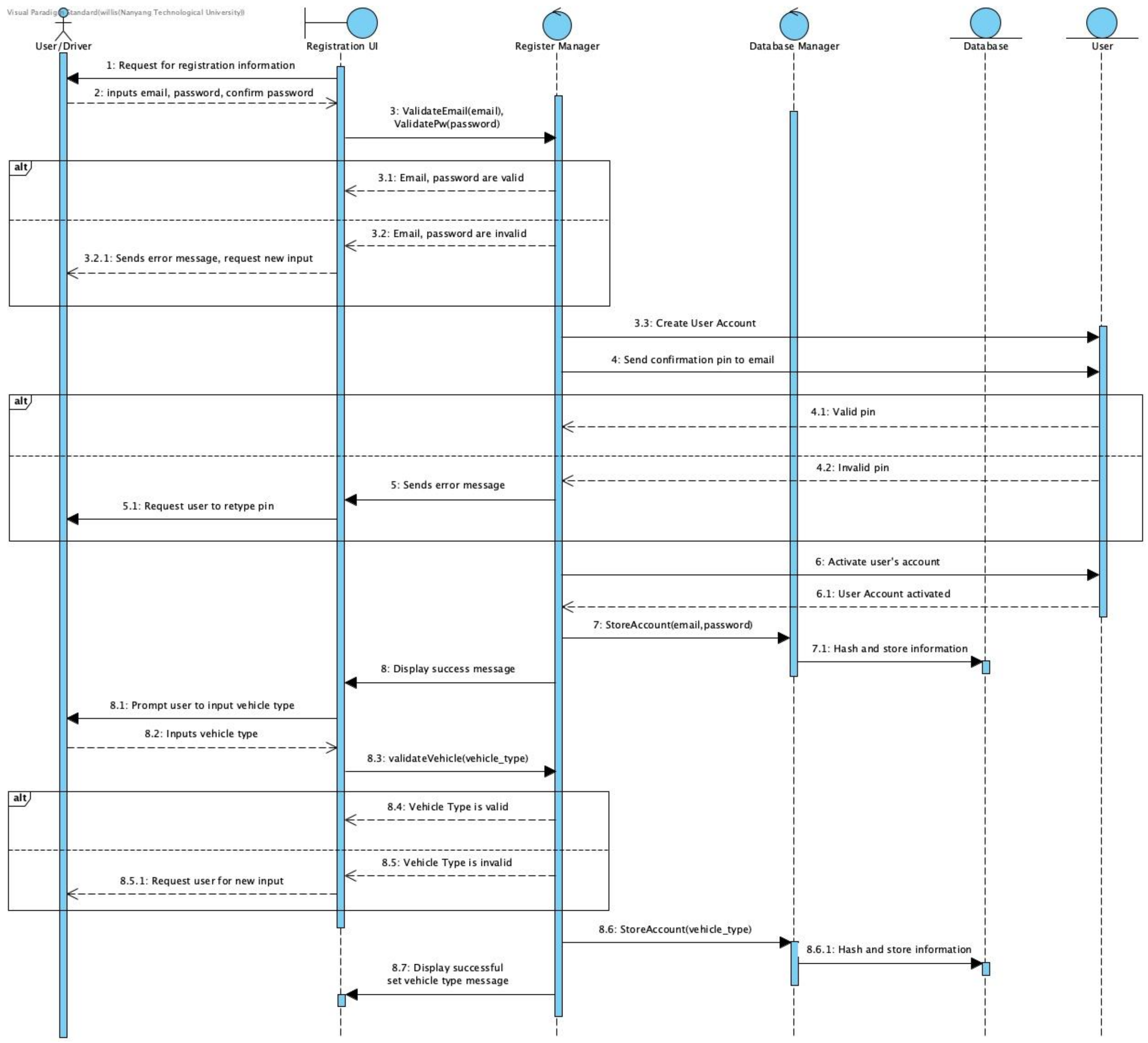

Visual Paradigm Standard(willis(Nanyang Technological University))
User/Driver
Registration UI
Register Manager
Database Manager
Database
User
1: Request for registration information
2: inputs email, password, confirm password
3: ValidateEmail(email), ValidatePw(password)
alt
3.1: Email, password are valid
3.2: Email, password are invalid
3.2.1: Sends error message, request new input
3.3: Create User Account
4: Send confirmation pin to email
alt
4.1: Valid pin
4.2: Invalid pin
5: Sends error message
5.1: Request user to retype pin
6: Activate user's account
6.1: User Account activated
7: StoreAccount(email,password)
7.1: Hash and store information
8: Display success message
8.1: Prompt user to input vehicle type
8.2: Inputs vehicle type
8.3: validateVehicle(vehicle_type)
alt
8.4: Vehicle Type is valid
8.5: Vehicle Type is invalid
8.5.1: Request user for new input
8.6: StoreAccount(vehicle_type)
8.6.1: Hash and store information
8.7: Display successful set vehicle type message

Figure 4 Use Case #1: Registration Sequence Diagram

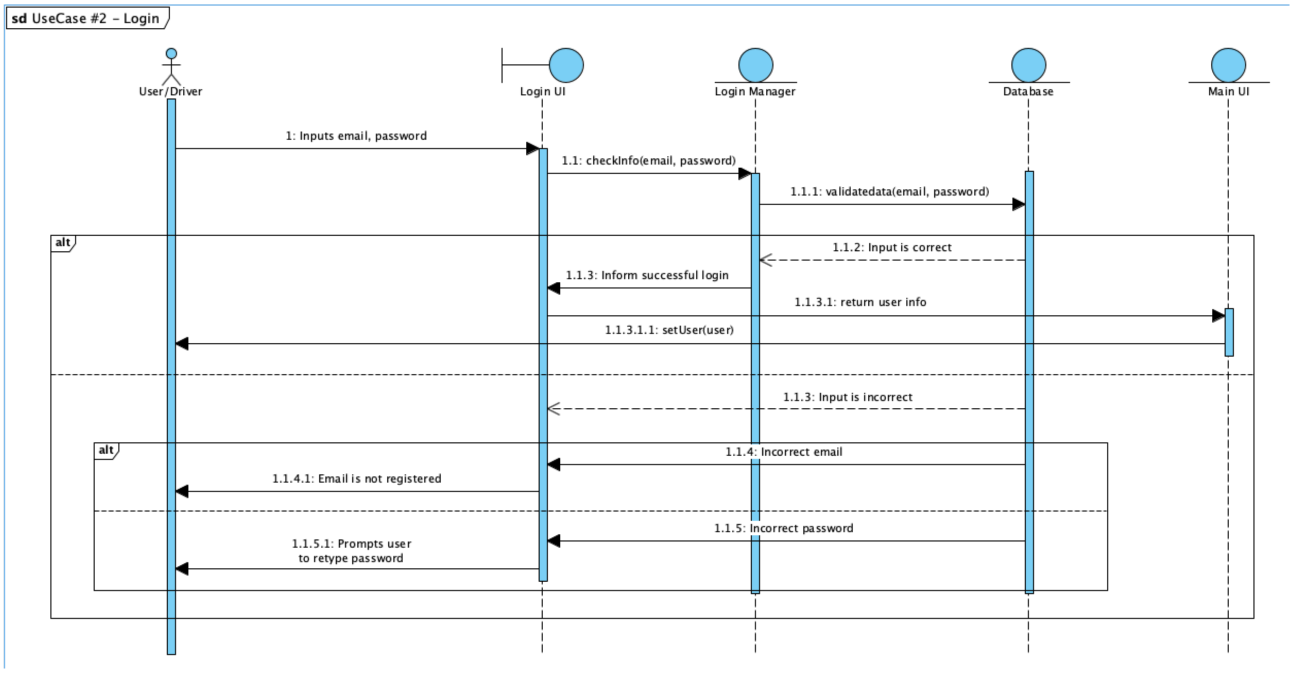

sd UseCase #2 – Login
User/Driver
Login UI
Login Manager
Database
Main UI
1: Inputs email, password
1.1: checkInfo(email, password)
1.1.1: validatedata(email, password)
alt
1.1.2: Input is correct
1.1.3: Inform successful login
1.1.3.1: return user info
1.1.3.1.1: setUser(user)
1.1.3: Input is incorrect
alt
1.1.4: Incorrect email
1.1.4.1: Email is not registered
1.1.5: Incorrect password
1.1.5.1: Prompts user to retype password

Figure 5 Use Case #2: Login Sequence Diagram

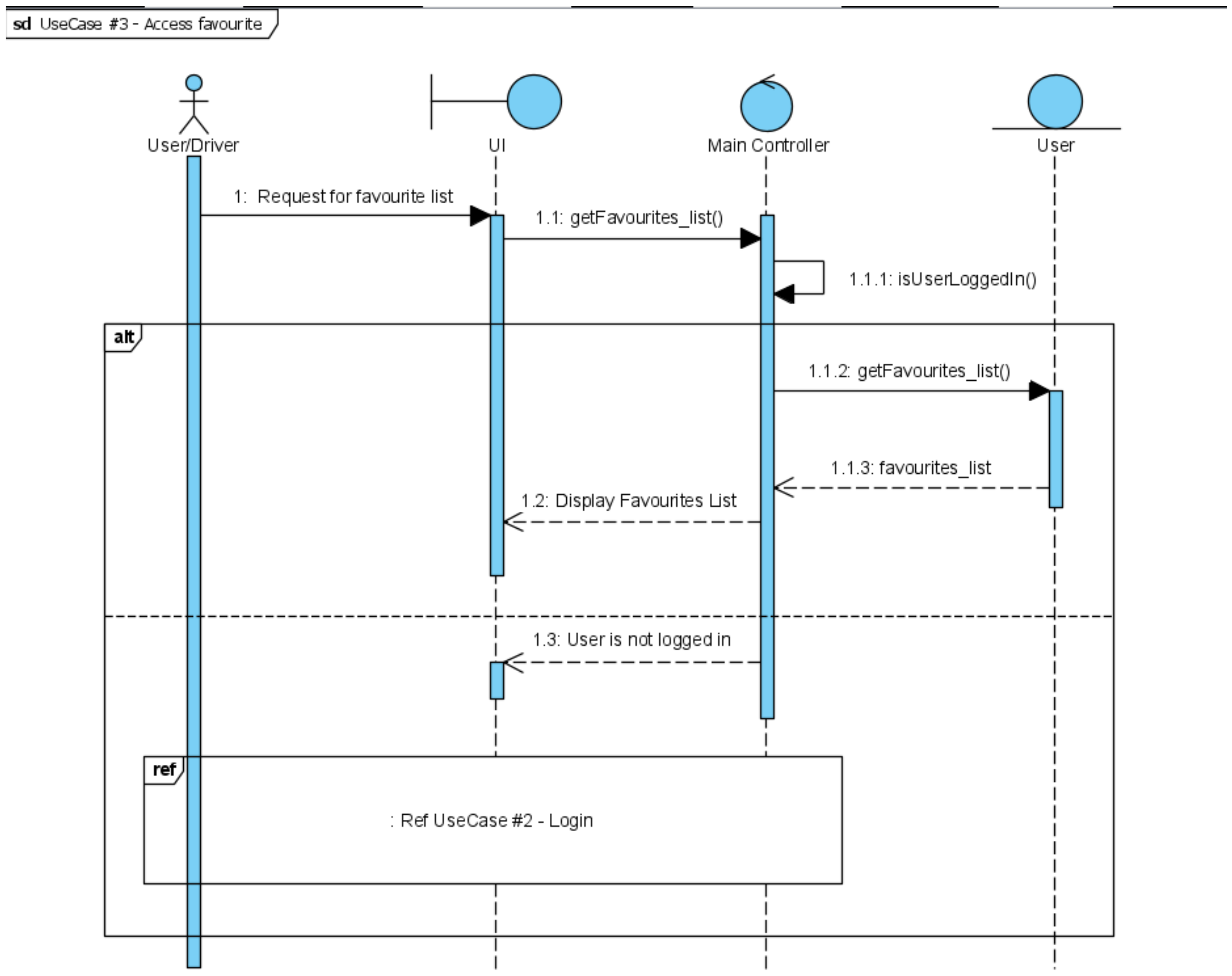

sd UseCase #3 - Access favourite
User/Driver
UI
Main Controller
User
1: Request for favourite list
1.1: getFavourites_list()
1.1.1: isUserLoggedIn()
alt
1.1.2: getFavourites_list()
1.1.3: favourites_list
1.2: Display Favourites List
1.3: User is not logged in
ref
: Ref UseCase #2 - Login

Figure 6: Use Case #3: Access Favourites Sequence Diagram

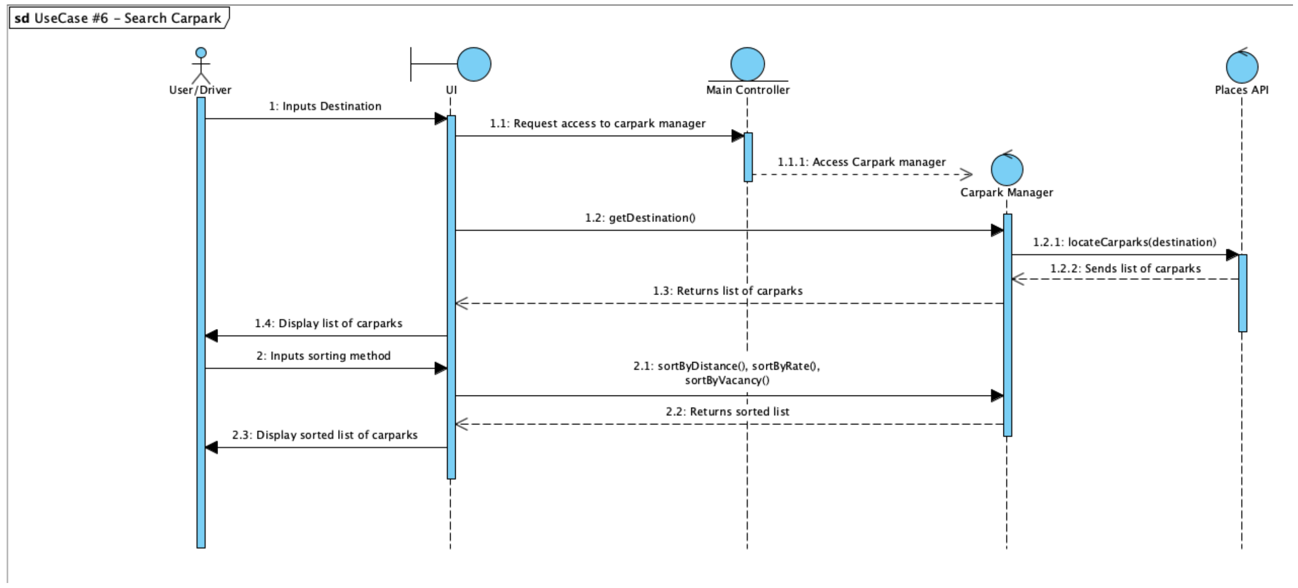

sd UseCase #6 - Search Carpark
User/Driver
UI
Main Controller
Places API
Carpark Manager
1: Inputs Destination
1.1: Request access to carpark manager
1.1.1: Access Carpark manager
1.2: getDestination()
1.2.1: locateCarparks(destination)
1.2.2: Sends list of carparks
1.3: Returns list of carparks
1.4: Display list of carparks
2: Inputs sorting method
2.1: sortByDistance(), sortByRate(), sortByVacancy()
2.2: Returns sorted list
2.3: Display sorted list of carparks

Figure 7: Use Case #4: Search Carpark Sequence Diagram

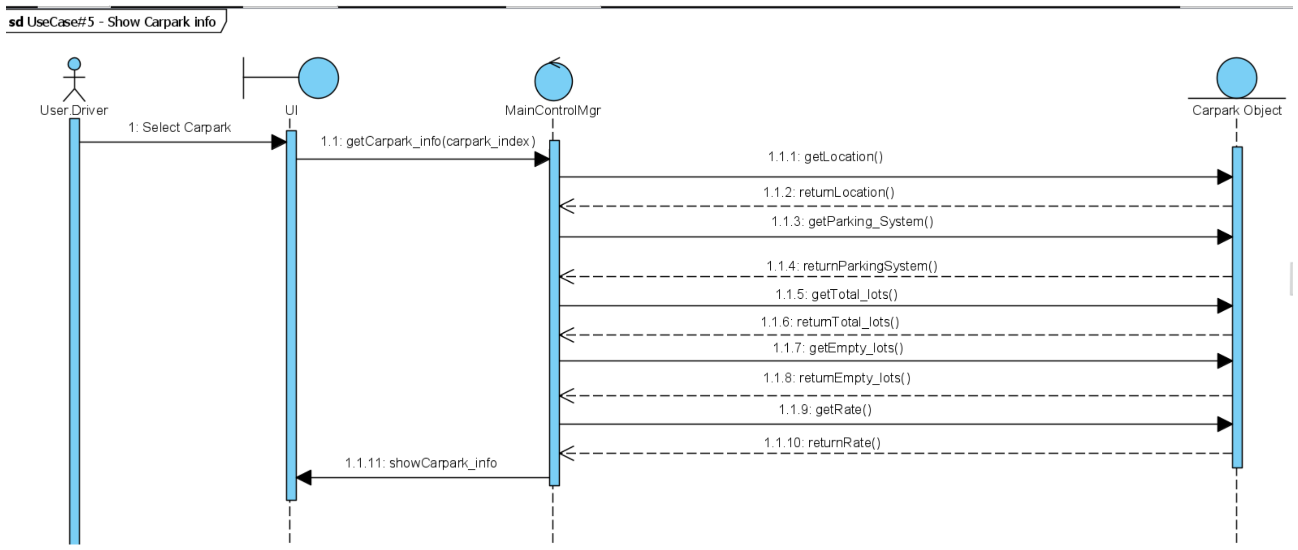


Figure 8: Use Case #5: Show carpark info Sequence Diagram

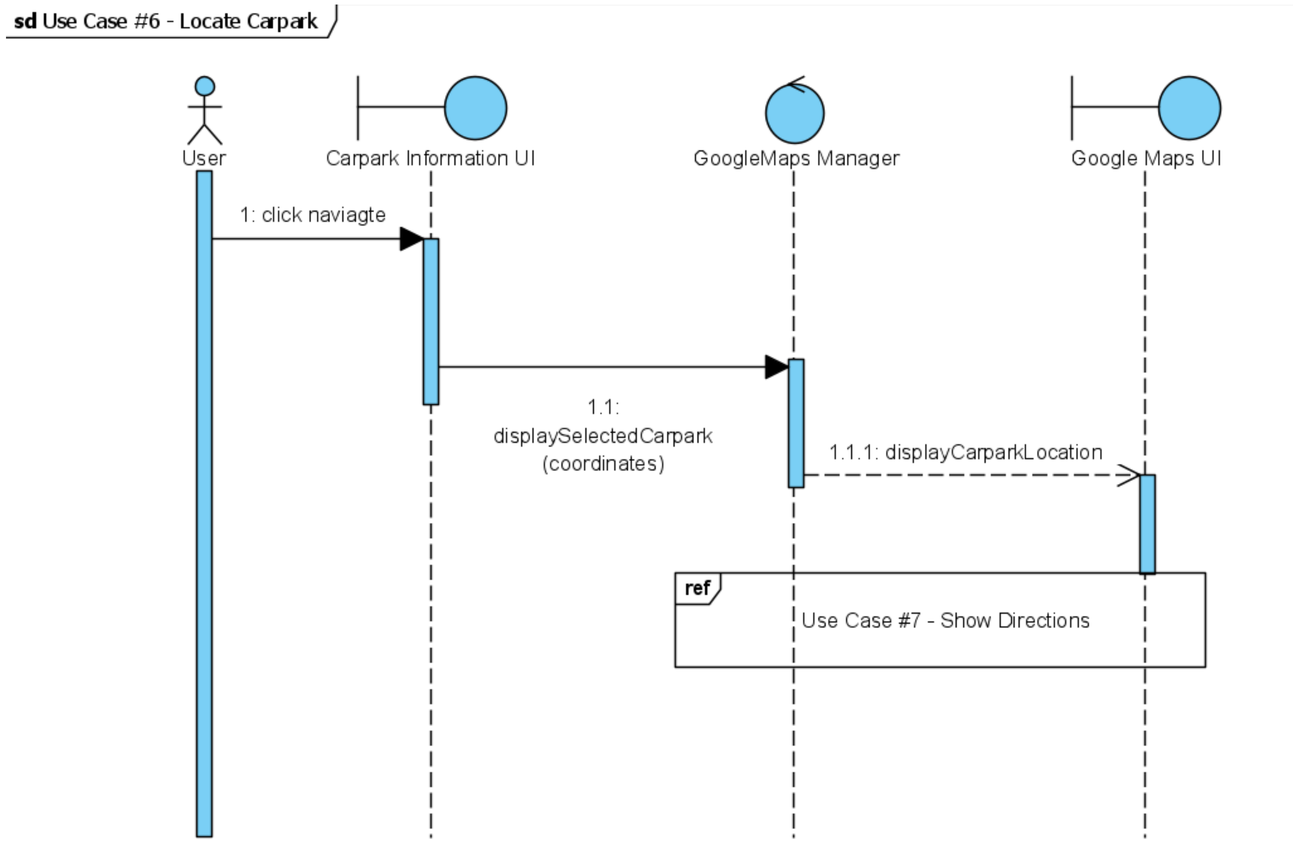

sd Use Case #6 - Locate Carpark
User
Carpark Information UI
GoogleMaps Manager
Google Maps UI
1: click naviagte
1.1: displaySelectedCarpark (coordinates)
1.1.1: displayCarparkLocation
ref
Use Case #7 - Show Directions

Figure 9: Use Case #6: Locate carpark Sequence Diagram

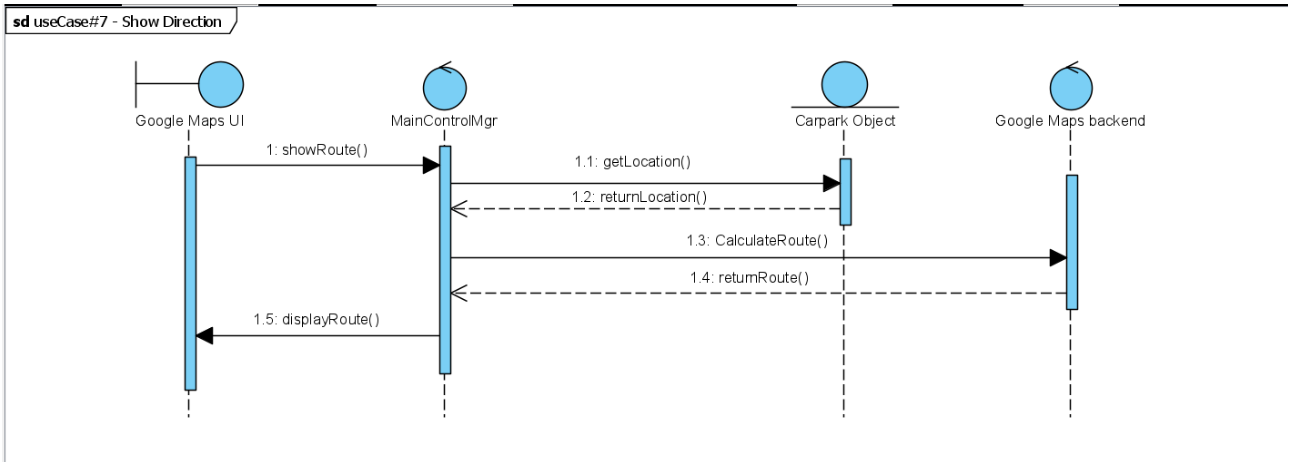


Figure 10: Use Case #7: Show direction Sequence Diagram

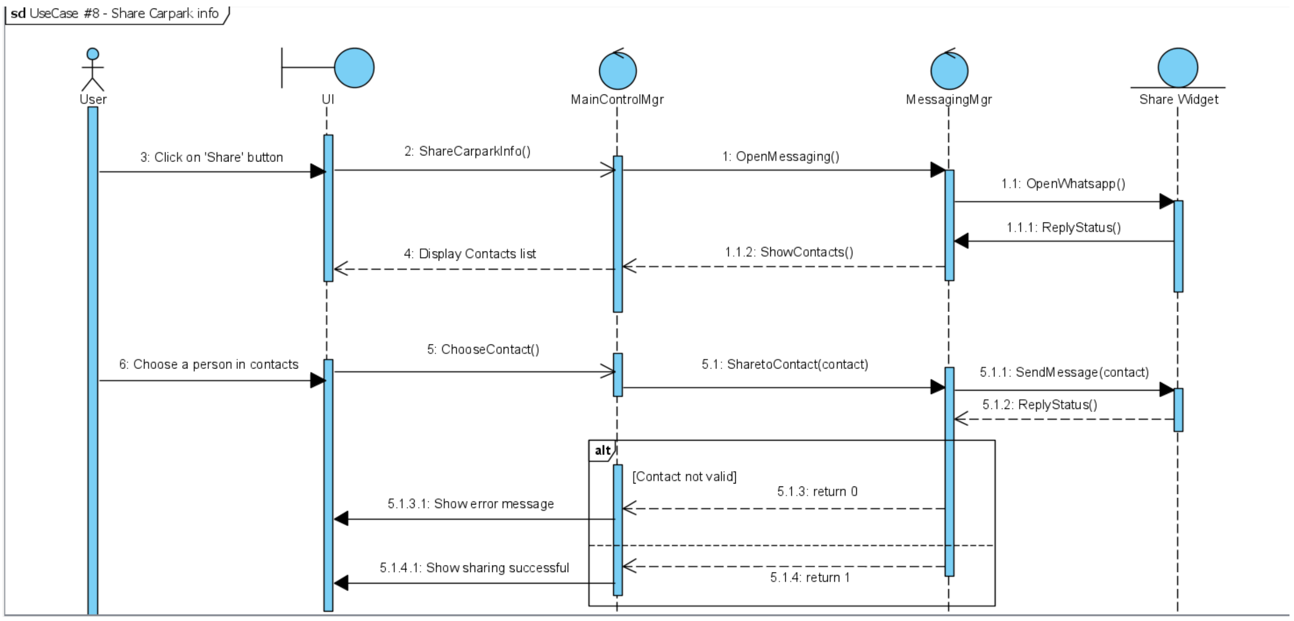


Figure 11: Use Case #8: Share carpark info Sequence Diagram

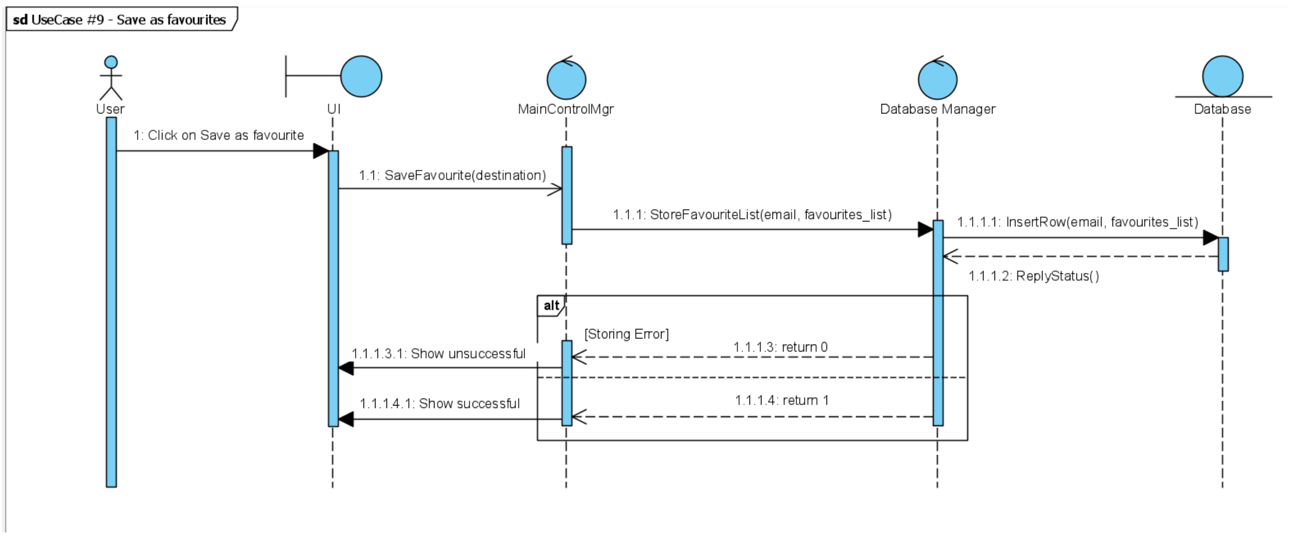

sd UseCase #9 - Save as favourites
User
UI
MainControlMgr
Database Manager
Database
1: Click on Save as favourite
1.1: SaveFavourite(destination)
1.1.1: StoreFavouriteList(email, favourites_list)
1.1.1.1: InsertRow(email, favourites_list)
1.1.1.2: ReplyStatus()
alt
[Storing Error]
1.1.1.3: return 0
1.1.1.3.1: Show unsuccessful
1.1.1.4: return 1
1.1.1.4.1: Show successful

Figure 12: Use Case #9: Save as favourites Sequence Diagram