Software Requirements Specification

for

ParkNow

Version 1.3 approved

Prepared by

Wang Yangming
Keith Heng Jin Sheng
Lee Pak Xin, Joel
Ye Wint Myint Myat
Yung Caleb

Nanyang Technological University, Team Overflow

29th Oct, 2023

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Yung Caleb	Sept 9	Modifying to add in past deliverables	1.0
Yung Caleb	Nov 4	Add Introductions, User Interface Mockup	1.1
Yung Caleb	Nov 6	Add System Architecture and other details	1.2
Yung Caleb	Nov 8	Update with final user interface look	1.3

1. Introduction

1.1 Purpose

This document's goal is to provide a full description of the ParkNow application. It will explain the system's goal and features, its interfaces, what the system will do, the limitations under which it must operate, and how the system will respond to external stimuli.

1.2 Document Conventions

Software Requirement Specification Format : This document follows IEEE standard. Priorities of higher level requirements are inherited by detailed level requirements.

Font : Times New Roman

Main Header : Size 18, Bold

Subsection Header : Size 14, Bold

Content : Size 11

Further conventions on the terms used could be found at Appendix A – Data Dictionary Section

1.3 Intended Audience and Reading Suggestions

This document is for

1. Developers who would like to continue the development of our software. This document provides detailed documentation of how the application is implemented. It also helps to keep track of our team's progress and ensure the product is in line with our intended goals.

2. Product Managers and marketing staff, to help them understand our product's scope, along with the benefits and limitations to be better able to pitch our project to potential investors.

3. Testers who want to understand the relevant components which they are testing.

4. Clients and/or users who would like to understand the underlying application and so that they can also recommend suggestions or request changes.

This document will run through the description of the application, user interfaces, functional requirements, nonfunctional requirements, and other requirements for ParkNow. Reading the document in order can assist in understanding the application better. Skipping to another section without prior understanding of the prior sections is not recommended.

Here is an overview of each section. The overall description will help in understanding the higher-level objective of ParkNow, along with design considerations or limitations of the application. The interfaces will show the developers and testers an expected view along with the design elements of the application from a software engineering perspective. The functional and non-functional requirements provide an understanding of the respective requirements for each section. In additional requirements, we provide the data dictionary to understand certain keywords used within ParkNow, this can help all readers in understanding the terms used in this document. Lastly, the appendix is for reference use.

1.4 Product Scope

ParkNow is a user-centric solution dedicated to revolutionising the parking experience for drivers in Singapore, with a particular focus on the bustling Central Business District (CBD). It goes above and beyond the typical parking app by offering a seamless and efficient way to locate available public car parks in these high-demand areas. ParkNow aims to make parking hassle-free for both local residents and visitors exploring the CBD.

What truly sets ParkNow apart is its commitment to providing real-time data and route recommendations. This means that, with ParkNow, you're not just searching for parking; you're gaining access to a wealth of information that helps you make informed decisions. Real-time parking availability data means that you can avoid the frustration of circling the CBD endlessly, saving you both time and stress.

The route recommendations feature is another game-changer. ParkNow does not just help you park; it helps you get there efficiently. It suggests the best routes to your destination, taking into account traffic conditions, road closures, and other factors that might impact your journey. This proactive approach to parking and navigation doesn't just make life easier for drivers; it's a key component of our broader mission to improve urban mobility and reduce congestion in Singapore.

By offering a glimpse of the future of parking in Singapore, ParkNow is not just a parking app; it's a visionary tool that's reshaping how we interact with urban infrastructure. With ParkNow, you're not just finding a parking spot; you're unlocking a world of convenience and efficiency. Our commitment to real-time data and smart route recommendations is part of our ongoing effort to create a smarter, more accessible, and less congested CBD.

1.5 References

I. IEEE 830-1998 Template

II. HTTP Status Code - https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

III. RESTful API - https://developer.mozilla.org/en-US/docs/Glossary/REST

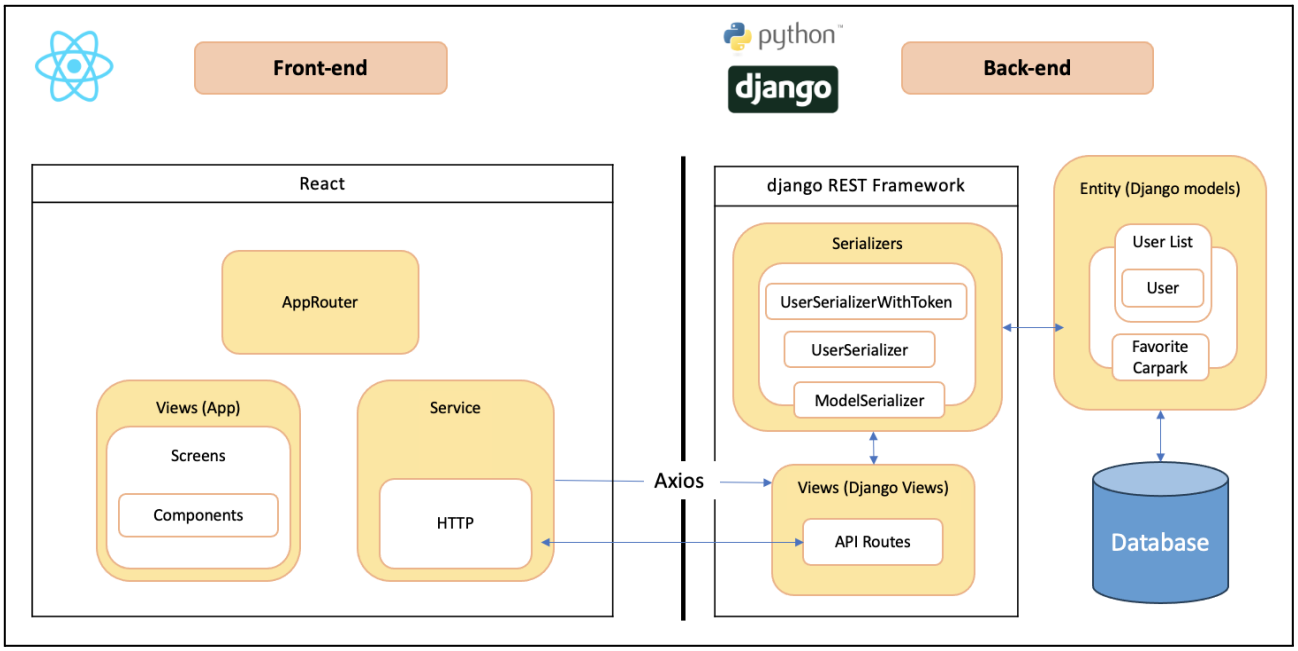
IV . React Native Framework - https://flutter.dev/

V . Python Django Framework - https://www.djangoproject.com/

VI. MySQL Database - https://www.mysql.com/

2. Overall Description

2.1 Product Perspective



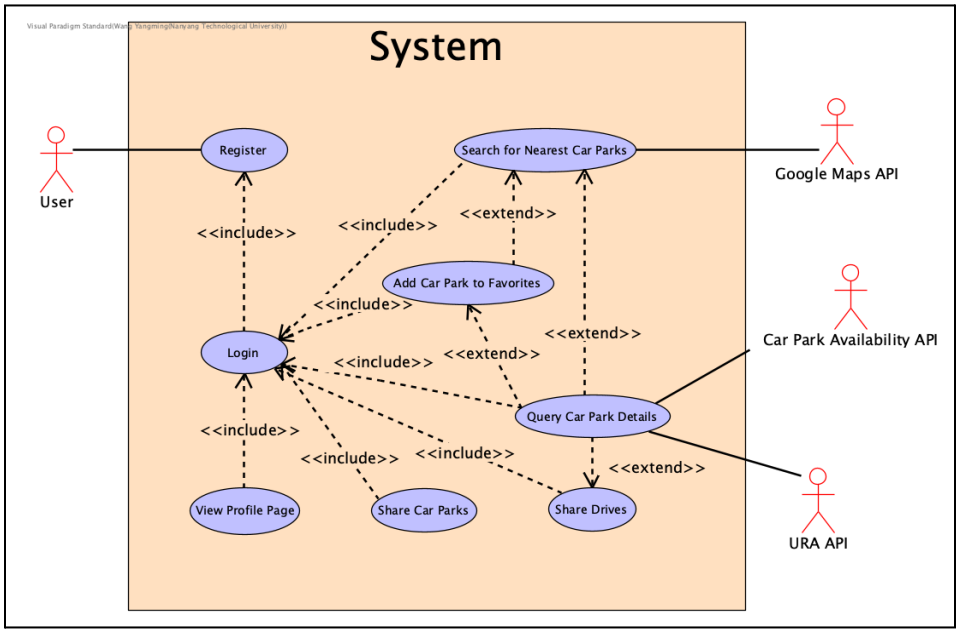
ParkNow is newly-developed. It is meant to connect Google Maps APIs and the datasets provided publicly by data.gov.sg (data.gov). Google Maps API will provide us with the Map, Route, Autofill functions.

Other products similar to ours already exist in the market. However, those products are navigation apps with parking as an additional function of parking. Our product focuses mainly on finding cheap available public parking spaces locally in Singapore's CBD.

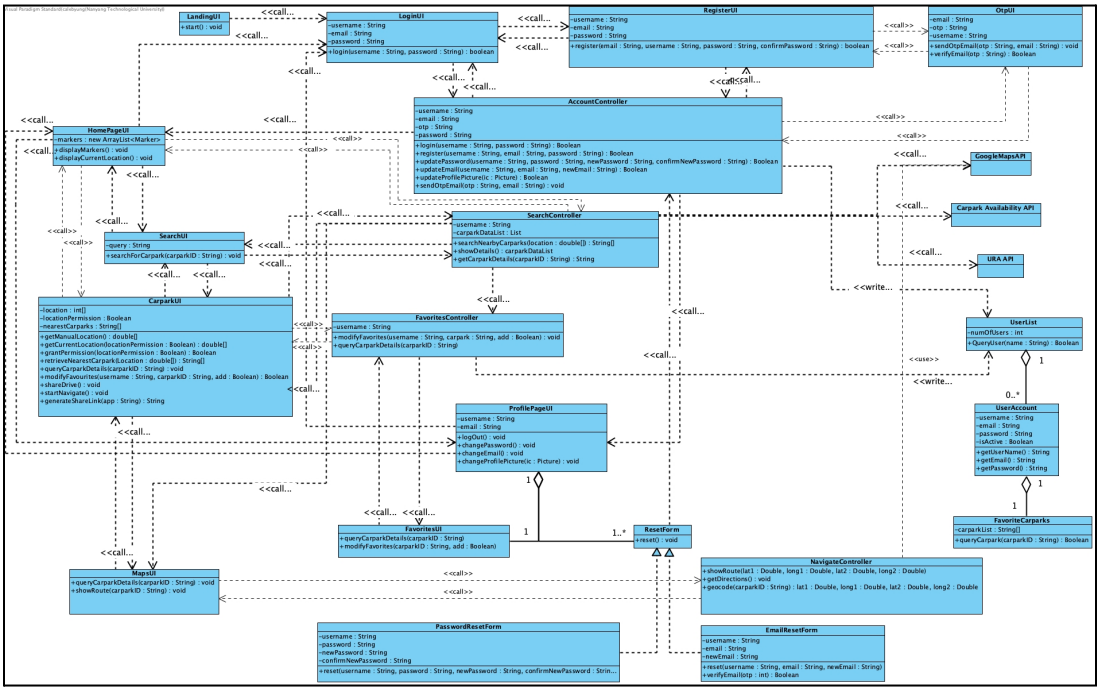
2.2 Product Functions

Our Use Case Descriptions can be found in our Use Case documentation. The major functions of ParkNow will be the search, navigation of nearby carparks, and their details such as availability and rates.

2.2.1 User Case Diagram

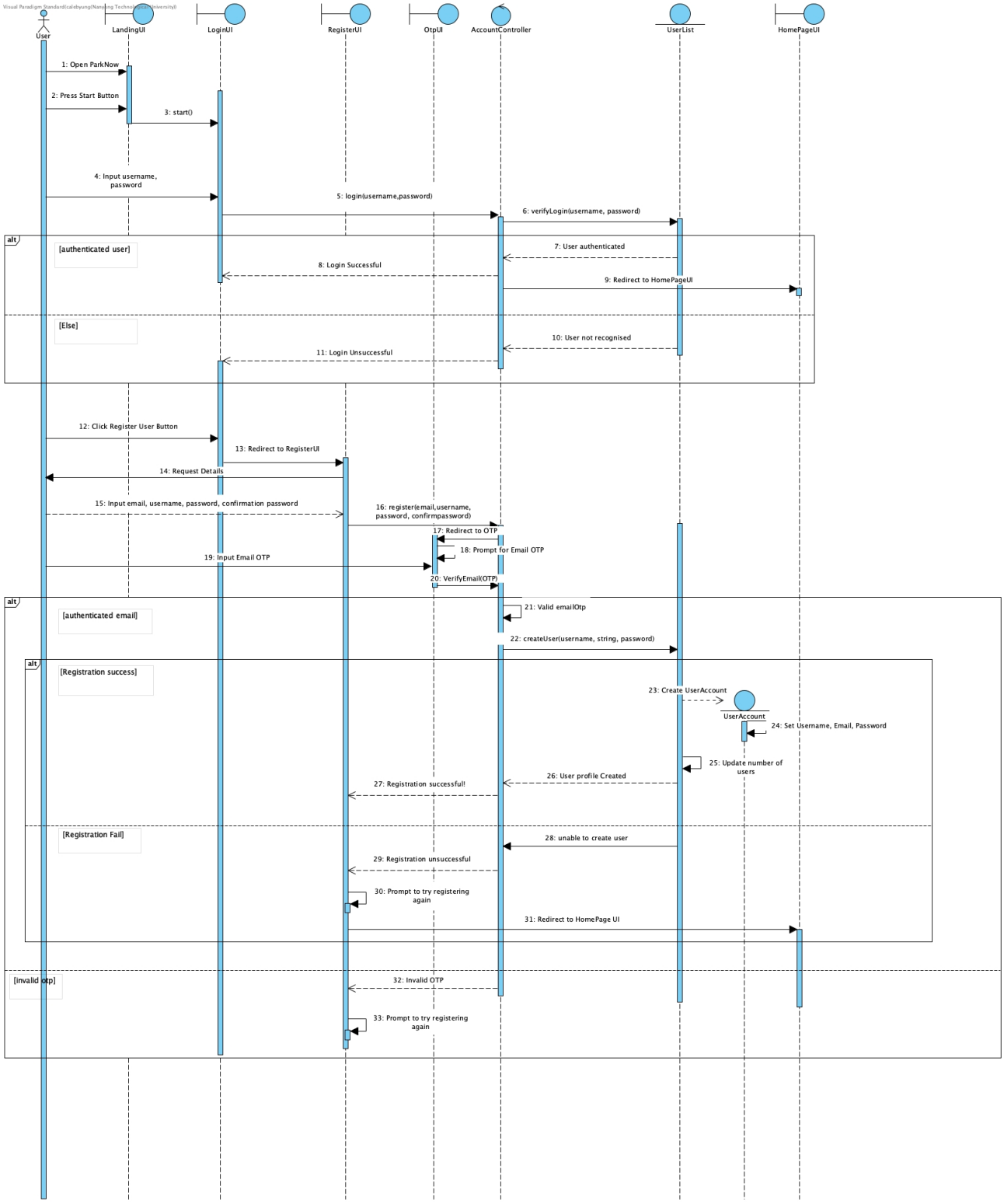


2.2.2 Class Diagram

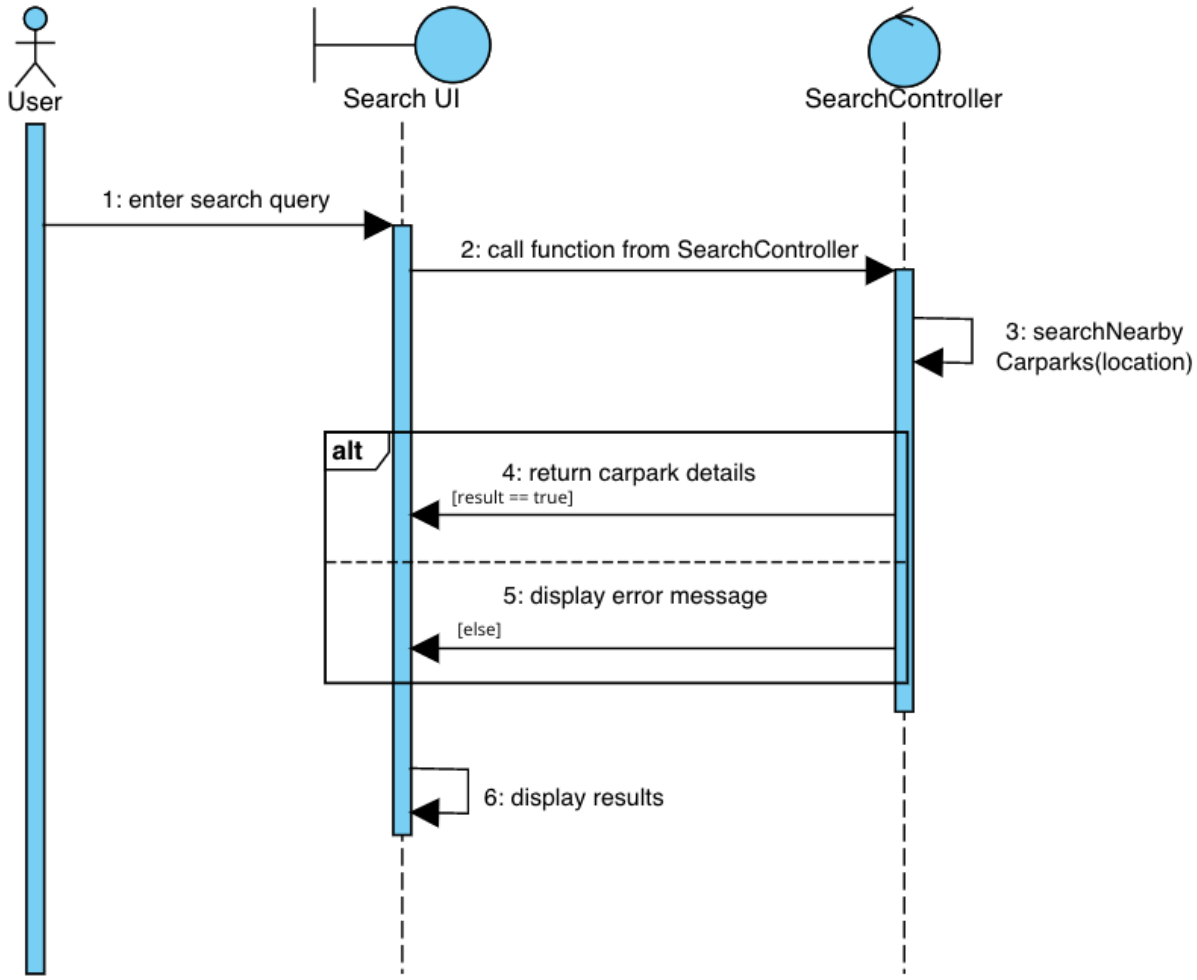


2.2.3 Sequence Diagram

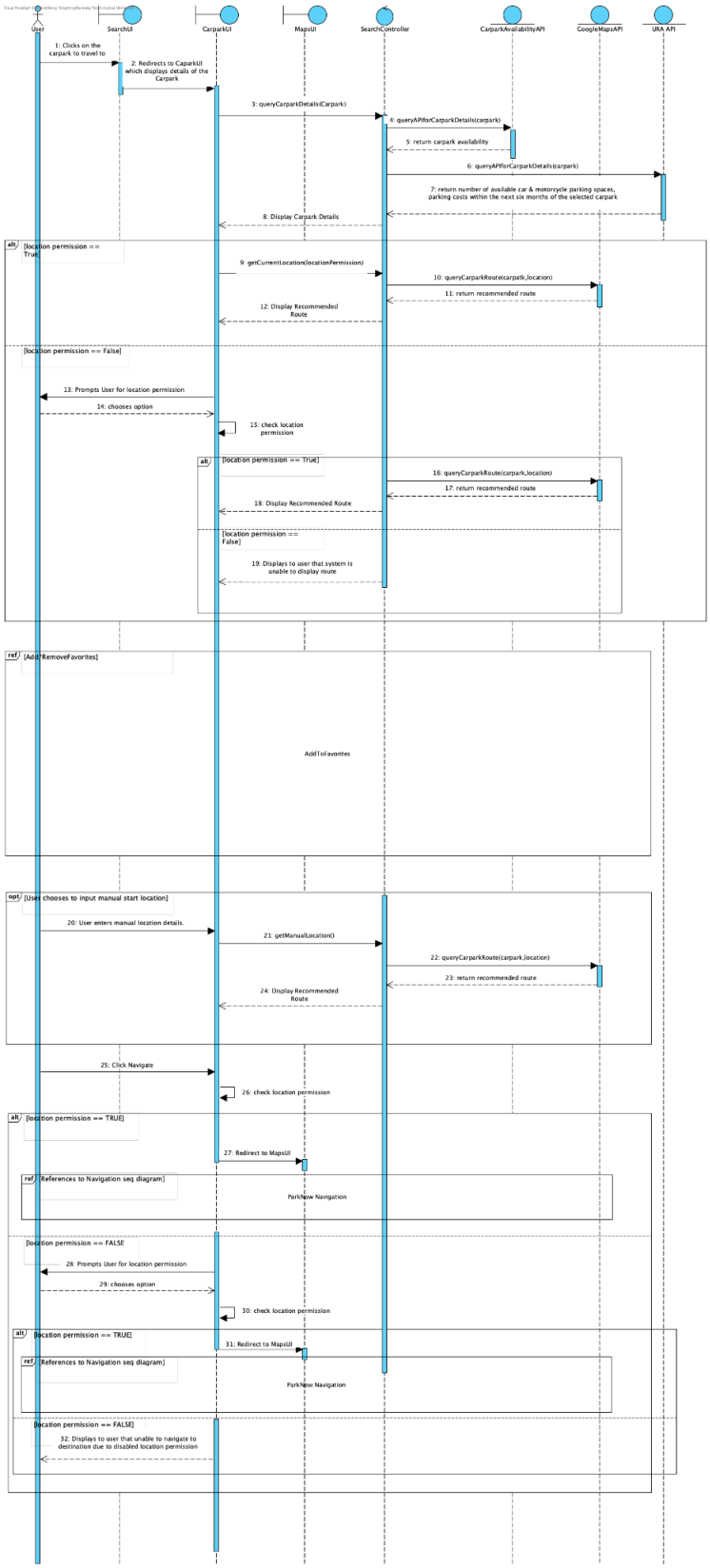
2.2.3.1 Login Register



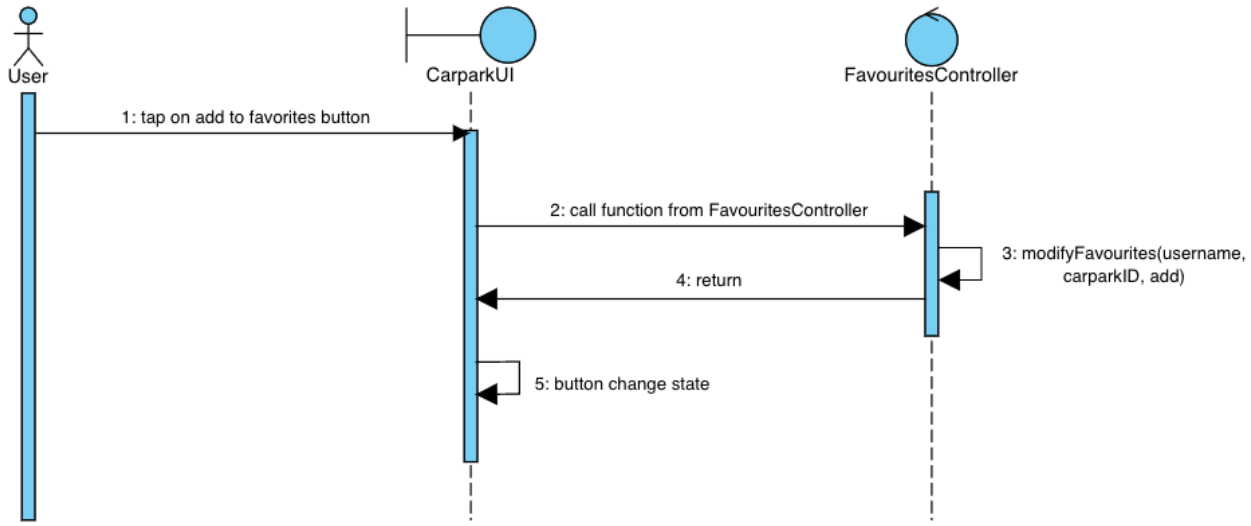
2.2.3.2 Search for Carpark



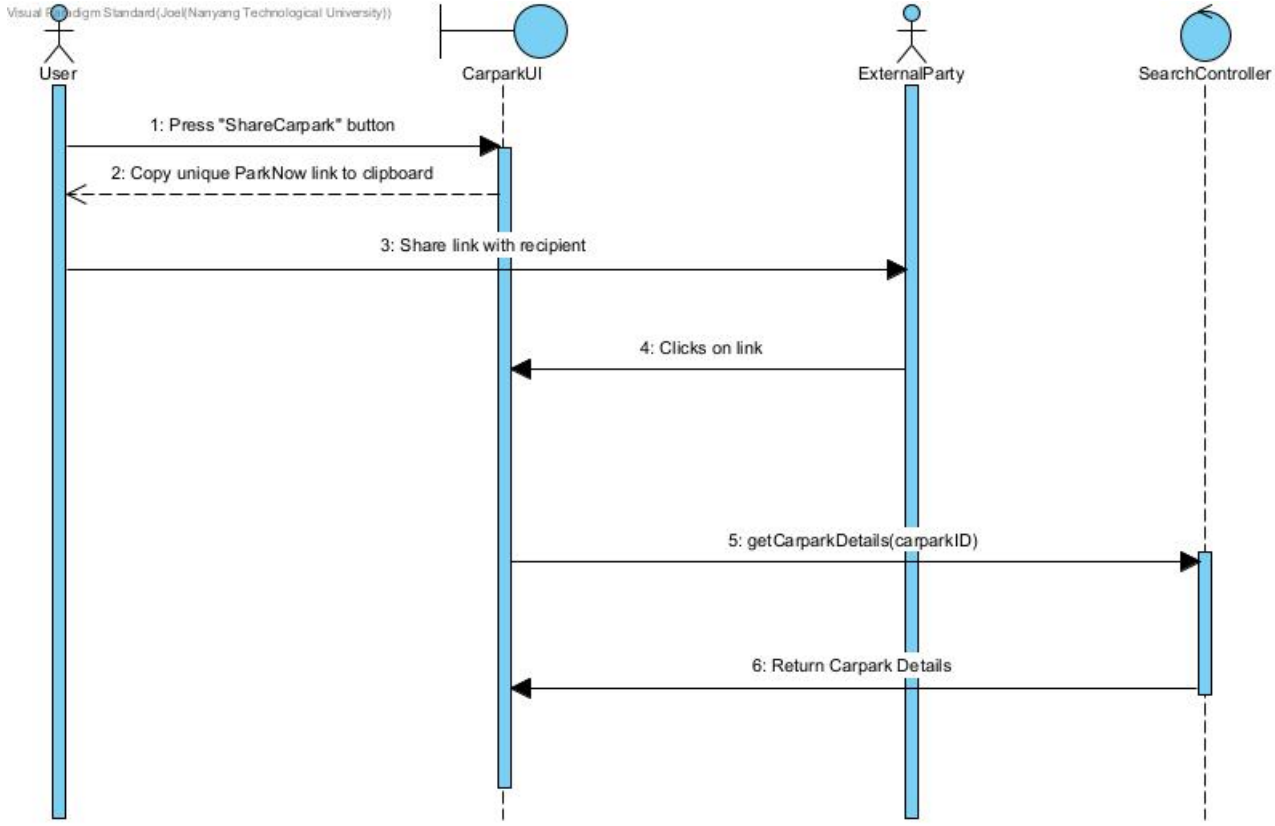
2.2.3.3 Search Details



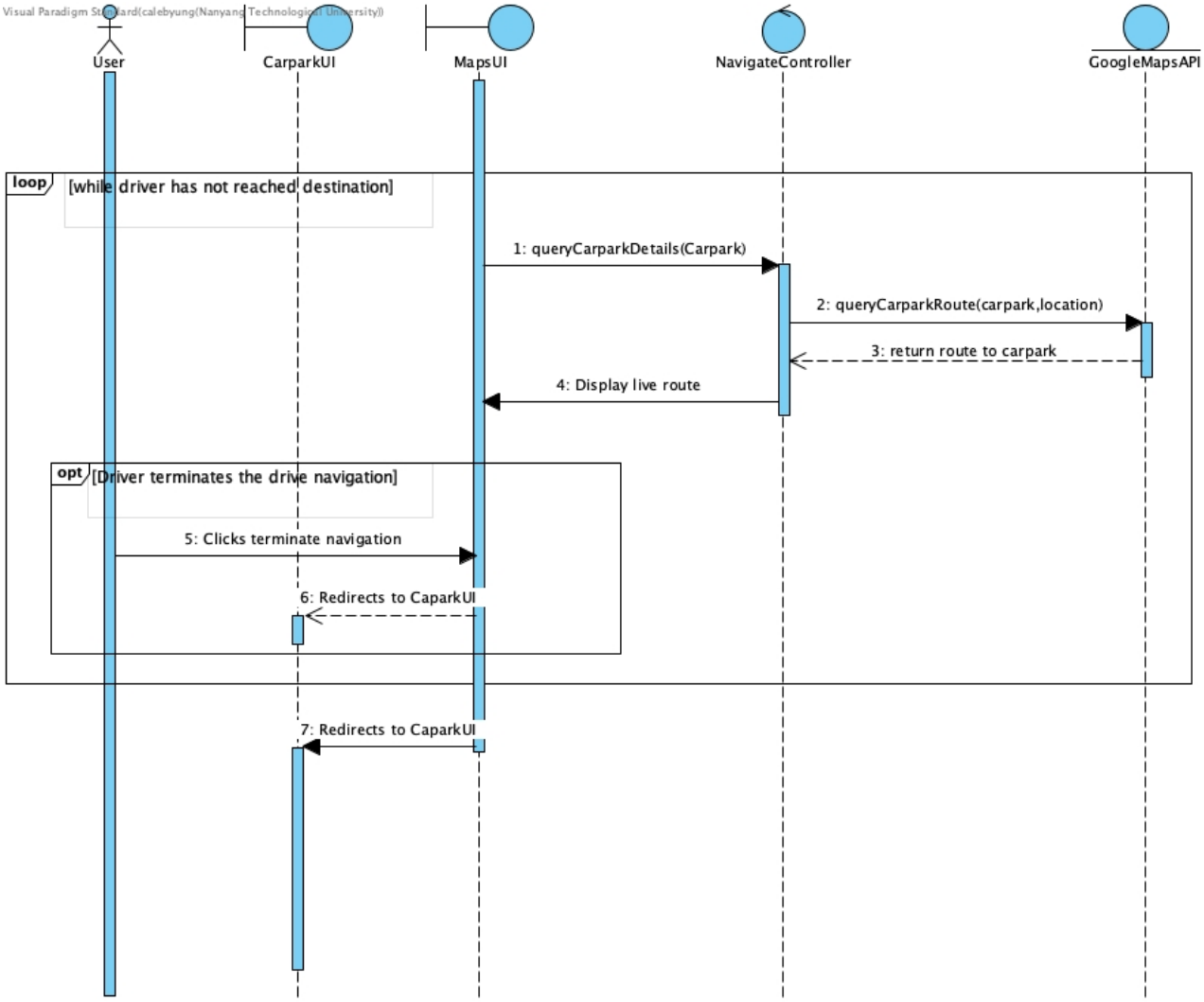
2.2.3.4 Add to Favourites



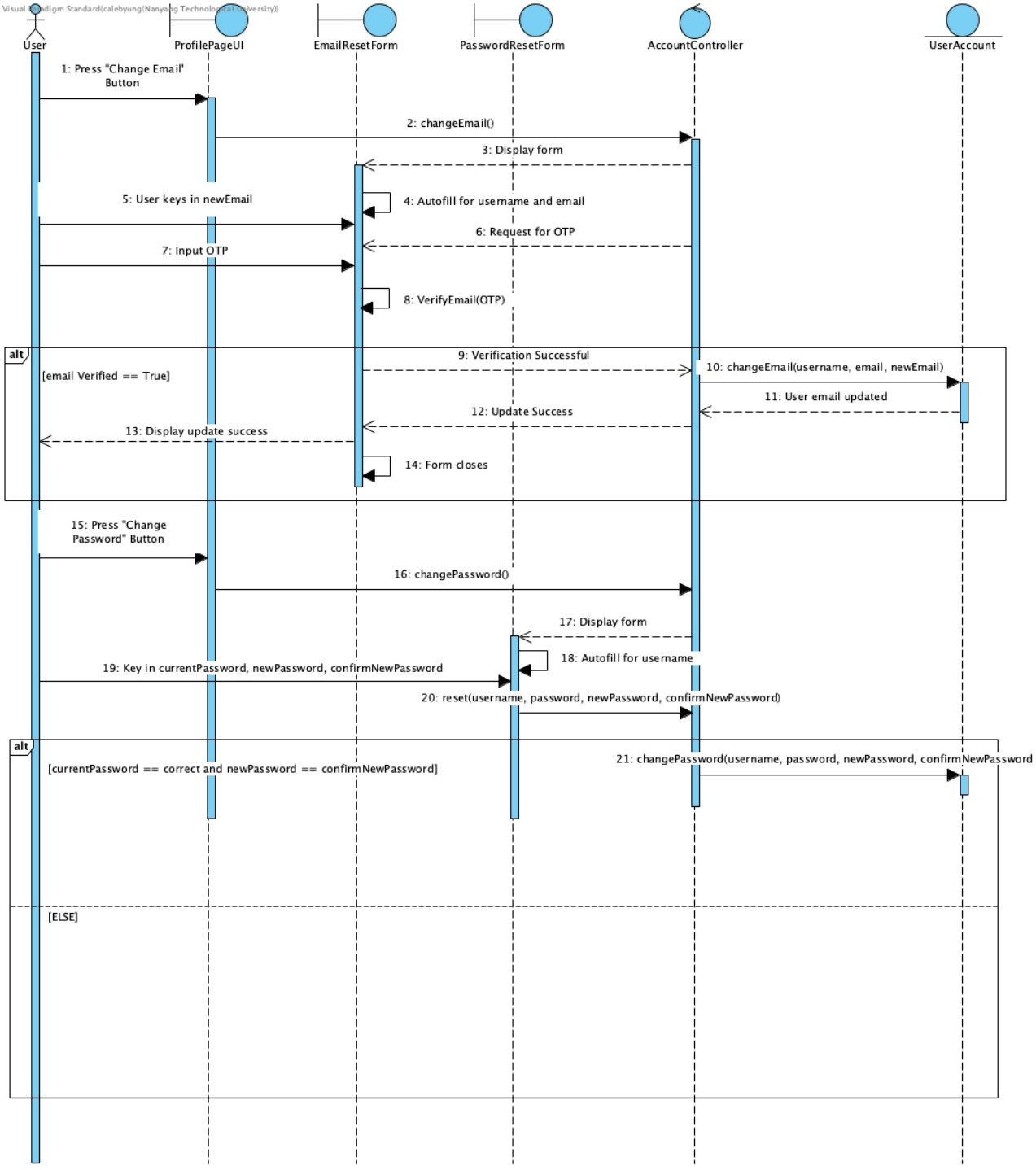
2.2.3.5 Share Carpark



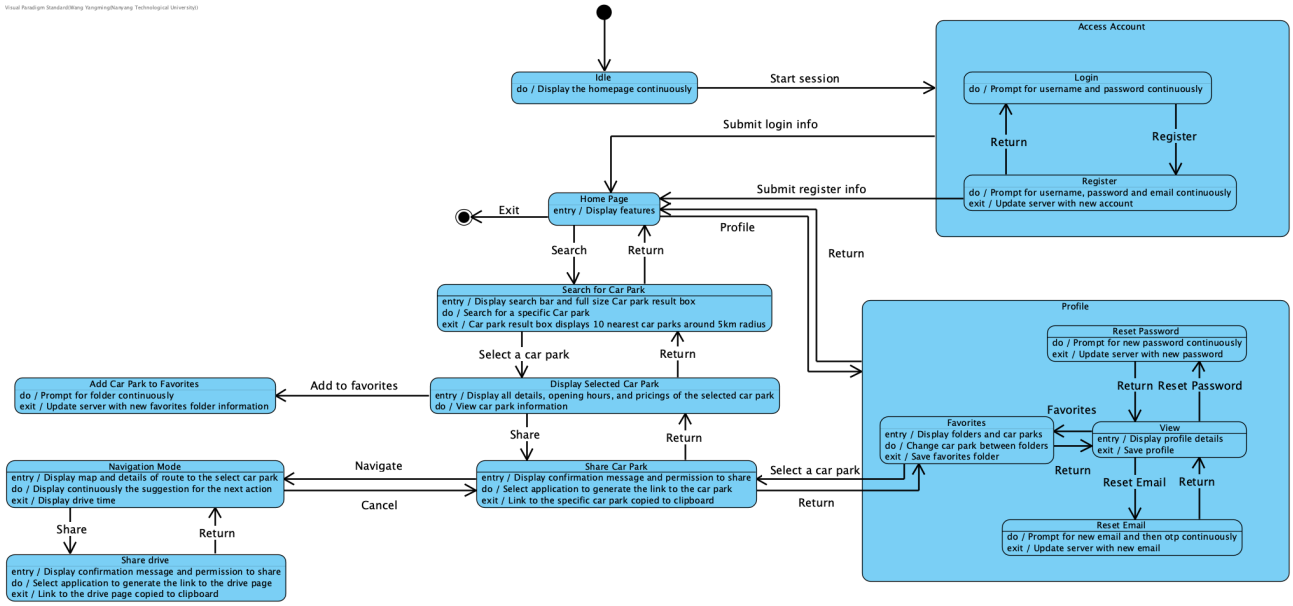
2.2.3.6 Navigation



2.2.3.7 Manage Profile



2.2.4 Dialog Map



2.3 User Classes and Characteristics

Driver

This class of users will use the application to search for their destinations and find nearby available carparks. Also, each driver can favourite and unfavourite carparks.

2.4 Operating Environment

Operating environment for our application is as listed below.

React Native

The front end will be developed entirely in JavaScript. This will allow ParkNow to be supported on both IOS and

Django

The back end will be developed entirely in Python.

mySQL

The application will have to parse data retrieved from mySQL where our dataset is stored.

Google Maps

Our application will utilise the Google Maps API to integrate the Maps, Routes, Autofill functions.

2.5 Design and Implementation Constraints

Our product is developed with ReactNative, Django, and mySQL. Once the software is delivered to our client, the responsibility of future maintenance will be passed to the client's organisation. Thus, the client's maintenance team will need to have prior knowledge of and experience with ReactNative, Django and mySQL.

2.6 User Documentation

ParkNow is a mobile application created with a focus on providing a user-friendly and efficient experience for individuals searching for free public car parks in the central area. To enhance user experience, the application incorporates an introductory feature designed for first-time users. The application is made to be user friendly and intuitive. This helps users understand and utilise the key features of the application, making it easier to navigate and maximise its utility.

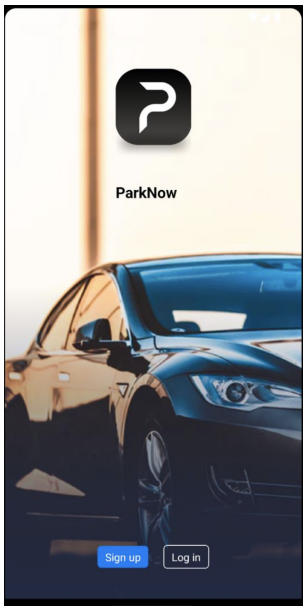
2.7 Assumptions and Dependencies

Our product assumes that users would enter a popular tourist location, and there are carparks listed under URA nearby.

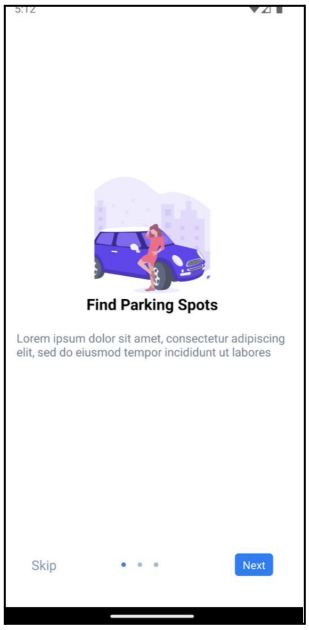
3. External Interface Requirements

3.1 User Interfaces

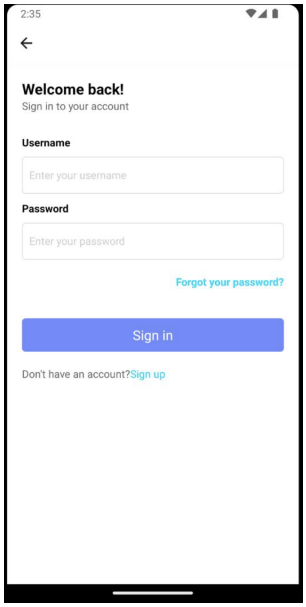
Screenshot 1: Starting page



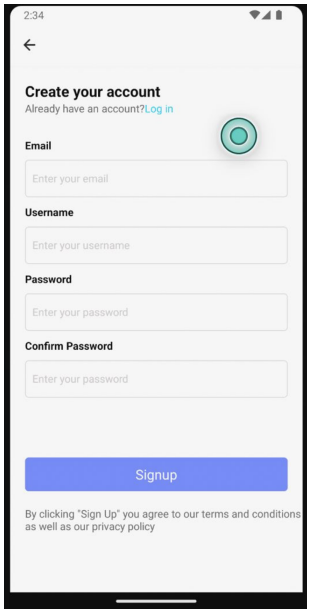
Screenshot 2 : Landing page



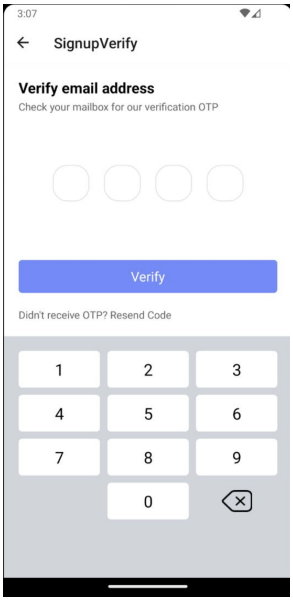
Screenshot 3 : Introductory Page



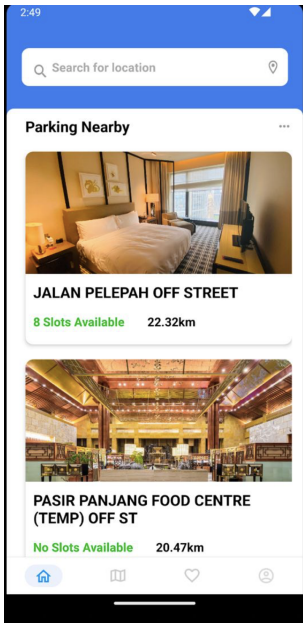
Screenshot 4 : Login Page



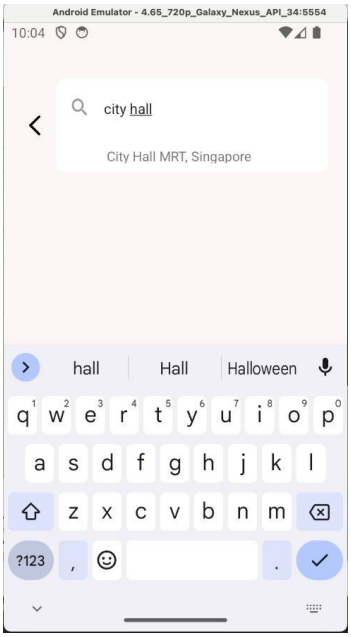
Screenshot 5 : Registration Page



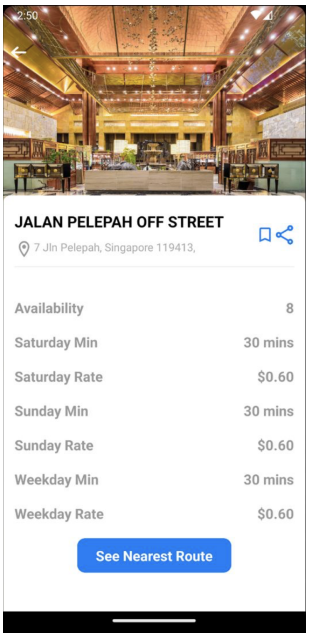
Screenshot 6 : OTP Page



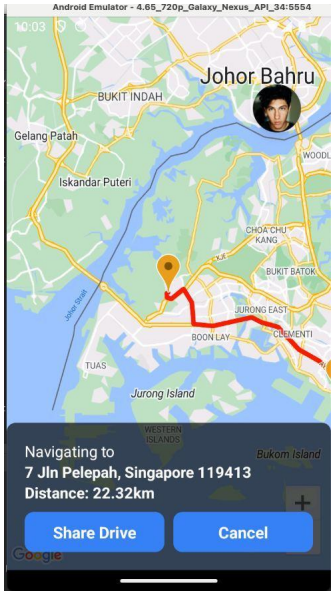
Screenshot 7 : Home page



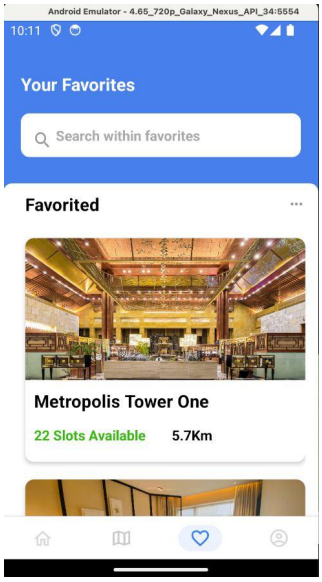
Screenshot 8 : Search Bar



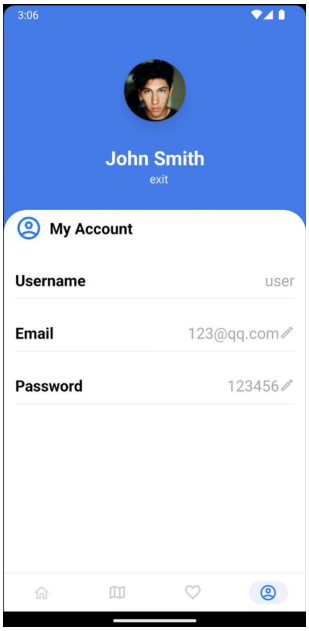
Screenshot 9 : Carpark Details



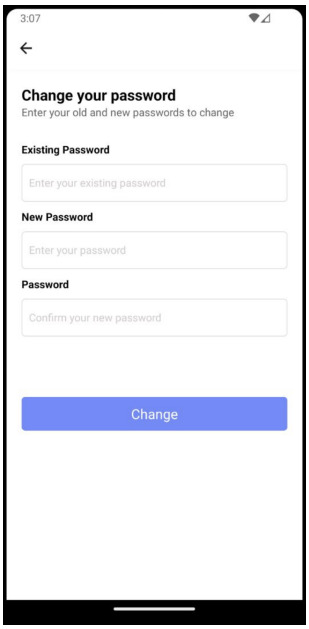
Screenshot 11 : Navigation Page



Screenshot 12 : Favourites Page



Screenshot 13 : Profile page



Screenshot 14 : Change Password

3.2 Hardware Interfaces

IOS :

- Minimum Version: IOS 10
- Recommended Version: IOS 14

Android mobile devices :

- Minimum Version: Android 4.1
- Recommended Version: Android 13

3.3 Software Interfaces

Software Used	**Description**
React Native	React Native is a JavaScript framework for writing real, natively rendering mobile applications for iOS and Android. It's based on React, Facebook's JavaScript library for building user interfaces, but instead of targeting the browser, it targets mobile platforms.
Django	Django is a high-level Python web framework that enables rapid development of secure and maintainable websites.
mySQL	MySQL is a relational database management system A relational database stores data in separate tables rather than putting all the data in one big storeroom. The database structure is organised into physical files optimised for speed.

3.4 Communications Interfaces

In our product, the communication with the APIs uses the Django REST framework. React Native handles all the front-end rendering and routing while Django serves as a backend service that provides data and authentication.

4. System Features

4.1 Registration

4.1.1 Description and Priority

For first-time users, the application will direct them to the registration page to input verifiable information to create an account The username created must contain a minimum of 8 and a maximum of 32 alphanumeric characters, and it must be unique in the system. The email given must be of a correct email format and never registered in the system. The password given must contain at least 1 uppercase character, 1 special character, and contain at least 8 characters. The system must send an OTP to the user's email address.

Priority: High

4.1.2 Stimulus/Response Sequences

The user taps on the Register button at the login page and will be brought to the Registration page.

4.1.3 Functional Requirement

REQ-1: Application must provide account creation process for first time users using verifiable information.

REQ-1.1: Verifiable information must include email.

REQ-1.2: The application must provide error messages when invalid, and all the text fields must be emptied when the user's login details are rejected.

REQ-1.3: OTP will be sent to the user's email.

REQ-2: The system must create an account for the user upon verification of login details.

REQ-2.1: The record must contain either a username or an email address.

REQ-2.2: The record must contain a password.

REQ-3: The system must log the user into the main page of the system.

4.2 Login

4.2.1 Description and Priority

The user must have a registered account. After launching the application, the login page will be shown. The user must enter their username and their password. If the user's credential is valid, the user will be successfully logged in and redirected to the Home Page. Otherwise, a message will be sent.

Priority: High

4.2.2 Stimulus/Response Sequences

The user opens ParkNow and is brought to the login page.

4.2.3 Functional Requirement

REQ-1: The application must provide a user login process.

REQ-1.1: The login interface must include text fields for User ID and Password entry.

REQ-1.2: The User ID field must accept the user's created username

REQ-1.3: If the credentials are invalid, the application must provide an error message, and login details will be cleared from input fields.

REQ-2: The system must log the user into the main page of the system.

4.3 Search for Nearest Car Parks

4.3.1 Description and Priority

The system's landing page must include a search bar for users. Upon clicking the search bar, The search bar will accept input related to the user's location, which can be either their current location or a location specified by the user, and the user's destination. Additionally, the system should be able to identify the input location using the Google Maps API and match to the closest carparks to the given location.

Priority: High

4.3.2 Stimulus/Response Sequences

Upon accessing the system's home page, users will have access to a search bar for location input.

4.3.3 Functional Requirement

REQ-1: The system's home page must feature a search bar.

REQ-1.1: The search bar should accept location input, which can be either the user's current location or a location specified by the user.

REQ-1.2: The system should utilise the Google Maps API to identify the input location.

REQ-1.3: In response to a user's search request, the system must display a list of up to 10 car parks located within a 5-kilometre radius from the specified location, by utilising the URA API.

REQ-1.4: The system must sort the listed car parks from the shortest to the longest linear distance from the user's current location.

4.4 Query Carpark Details

4.4.1 Description and Priority

The users will be able to select a car park from the search result list. Upon selection, the system must access real-time car park details from the URA API for the selected car park. This information should include the number of parking spaces, and parking costs within six months for the selected carpark. Additionally, the system should provide a recommended route to the selected car park, including an estimated travel time by car.

Priority: High

4.4.2 Stimulus/Response Sequences

After selecting a car park from the search result list, the system will retrieve real-time information and present a recommended route to the user.

4.4.3 Functional Requirement

REQ-1: Users must be able to select a car park from the search result list.

REQ-1.1: Upon selection, the system must return real-time car park details from an API for the selected car park.

REQ-1.2: The system must display the number of available car parking spaces in the selected car park.

REQ-1.3: The system must display parking costs for the selected car park within the next six months.

REQ-2: The system must provide a recommended route to the selected car park.

REQ-2.1: The system should report the estimated travel time by car to reach the selected carpark.

4.5 Add Car Park to Favourites

4.5.1 Description and Priority

The users will be able to add car parks to their favourites folder. The system will allow users to view the car parks in the favourites folder. When a user selects a car park from the favourites folder, the system must access real-time data for the selected car park, including the number of car parking spaces, the number of motorcycle parking spaces, and parking costs within six months.

Priority: Medium

4.5.2 Stimulus/Response Sequences

The user adds car parks to their favourites folders and is able to view them in their favourites folder. When selecting a car park from the favourites folder, the system will retrieve real-time information.

4.5.3 Functional Requirement

REQ-1: The application must allow users to add car parks to the favourites folder.

REQ-1.1: The system must provide the option to display the car parks in the favourites folder.

REQ-1.2: When a user selects a car park from the favourites folder, the system must access real-time data from an API for the selected car park.

4.6 View Profile Page

4.6.1 Description and Priority

The system should include a profile page to display user information. This page should allow users to view their current profile picture, username, and email address. Users will be able to change their profile picture, email address, and password through the profile page. Furthermore, the system should allow users to log out directly from the profile page.

Priority: High

4.6.2 Stimulus/Response Sequences

When accessing the profile page, users can view and manage their user information.

4.6.3 Functional Requirement

REQ-1: The application must provide a profile page for users.

REQ-1.1: The application must display the user's current profile picture.

REQ-1.2: The application must display the user's username.

REQ-1.3: The application must display the user's email address.

REQ-1.4: The application must allow the user to change their email address.

REQ-1.5: The application must allow the user to change their password.

REQ-2: The application will allow the user to logout in the profile page.

4.7 Share Car Parks

4.7.1 Description and Priority

Users will be able to share information about car parks. This feature should allow users to choose the type of information they want to generate via a menu. The system should support various sharing options, including copying the car park's location in plain text to the clipboard, generating links to Google Maps, and Apple Maps, all of which can be copied to the clipboard. Additionally, if the user is logged in, they should be able to access the generated link in ParkNow.

Priority: Low

4.6.2 Stimulus/Response Sequences

When the user chooses to share information about car parks, a prompt will appear to allow them to select the type of information that they want to generate.

4.6.3 Functional Requirement

REQ-1: The application must allow users to share information about car parks.

REQ-1.1: The application must allow users to choose the type of information to generate via a menu.

REQ-1.2: The system must be able to copy the location of the car park in plain text to the clipboard.

REQ-1.3: TThe application must be able to generate a link to Google Maps on the clipboard.

REQ-1.4: The application must be able to generate a link to Apple Maps on the clipboard.

5. Other Requirements

5.1 Usability Requirements

The user must not spend more than 10 minutes to create an account. It should be intuitive and non-frustrating.

5.2 Performance Requirements

1. The application must be able to fetch the carpark information within 1000 ms when the user enters the destination location.

2. If network connection is not available, the application must display an informative pop-up box telling the user to try again later.

    - This is to inform users that there is a network error and to ask for their understanding.

5.3 Security Requirements

1. The application must only be accessible to users with an account.

2. User authentication must be done on mySQL.

3. User password must not be available to developers.

4. Passwords entered are not displayed in plain text.

5.4 Reliability Requirements

1. The application must load within 10s after launching the application.
2. The system must perform without failure in 95 percent of use cases during a month.

5.5 Maintainability Requirements

1. The mean time to restore the system (MTTRS) following a system failure must not be greater than 10 minutes. MTTRS includes all corrective maintenance time and delay time.

5.6 Availability Requirements

1. The web dashboard must be available to SG users 99.98 percent of the time every month during business hours GMT.

5.7 Scalability Requirements

1. The system must be scalable enough to support 10,000 visits at the same time while maintaining optimal performance.

5.8 Compatibility Requirements

1. The application must support devices of the following versions:
    a. iOS:
        i. Minimum Version: IOS 10
        ii. Recommended Version: IOS 14
    b. Android:
        i. Minimum Version: Android 4.1
        ii. Recommended Version: Android 13

Using versions below may result in the application being vulnerable to security flaws in previous Versions.

5.9 Localization Requirements

1. The system must be able to display in English.

5.10 Database Availability

Our product currently uses mySQL database to store information. The limited extensibility urges us to look for other solutions to act as our database for future updates, e.g. Microsoft SQL Server.

Appendix A: Glossary

Data Dictionary

Term	Description
Front-end	The front-end refers to the user interface (UI) and user experience (UX) components of our software. It is the part of the system that users directly interact with.
Back-end	The back-end is the behind-the-scenes portion of the system that is not directly accessible to users. It is responsible for storing, processing, and manipulating data, as well as managing the core functionality of the software.
API	An API serves as an intermediary between the front-end and back-end of our software, enabling the transfer of data and requests between these two components. It defines the methods and protocols for communication.
Login details	Login details consist of a unique user ID and an associated password. These credentials are used for user authentication, allowing access to the system.
User ID	A User ID is a unique identifier used for user authentication and recognition within our system. It can be either username or email address.
Username	A username is a user-specific alphanumeric identifier with a defined format, typically chosen by the user during registration. A username must have a minimum of 8 and a maximum of 32 alphanumeric characters.
Email address	An email address serves as a User ID and is another unique identifier with a specific format. It consists of a user-specific name followed by the "@" symbol and a domain name (e.g., "username@example.com"). Email addresses must comply with standard email format guidelines.
OTP	An OTP is a single-use, randomly generated password that serves as a secondary key for verifying the identity of a user. OTPs are typically sent to the user's registered email address and are used for secure login or authentication.
Account	An account is a secure and personalised digital entity within our system that is associated with an individual user. It serves as a container for storing user-specific information, preferences, and activities.

User	A user is any individual who utilizes our system. A user must have successfully logged into the system using their login details to access the system features.
Location	Location refers to any name, address, or postal code that is verifiable using Google Maps API.
Distance	Distance refers to linear distance within our system.
Route	A route is a planned path or course that is taken or followed to reach a specific destination or travel from one location to another.
mySQL	MySQL is an Oracle-backed open source relational database management system (RDBMS) based on Structured Query Language

Appendix B: Analysis Models

Class Diagram