Software Requirements Specification

for

SpotEase

Version 1.3 approved

Prepared by
Lee Yun Jia,
Jan Chen Jie,
Manasi Singh,
Glynis Looi Xin Lin,
Lee Lin Yang Glenn,
Kathri Arachchige Don Mayukhi Nimesha Siriwardana

SCSD C5, Nanyang Technological University

9 April 2025

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Lee Yun Jia, Glynis Looi	27 March 2025	Initial write up for Introduction, Overall Description and External Interface Requirements	1.0
Lee Yun Jia, Glynis Looi	28 March 2025	Initial write up for System Features, Other Nonfunctional requirements and Data Dictionary	1.1
Lee Yun Jia, Glynis Looi	29 March 2025	Initial write up for Other Requirements and Appendix	1.2
Lee Yun Jia, Glynis Looi	29 March 2025	Initial write up for Black Box and White Box Testing Test Cases	1.2
Glynis Looi	9 April	Finalize whole document	1.3

1. Introduction

1.1 Purpose

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

1.2 Document Conventions

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

1.3 Intended Audience and Reading Suggestions

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

1.3.1 Software Developers

To understand the constraints and technical details of the system.

1.3.2 Project Managers

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

1.3.3 Quality Control Testers

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

1.3.4 End Users

To aid users in understanding how SpotEase's features work

1.4 Product Scope

SpotEase is a mobile app built with React Native and Expo CLI to tackle parking challenges in Singapore's dense urban environment. With limited spaces and fluctuating costs, finding parking can be frustrating and time-consuming.

SpotEase uses real-time data on parking availability to reduce search time and optimize navigation. Designed for commuters, ride-hailing drivers, and tourists, the app enhances convenience and improves the overall driving experience in high-demand areas.

Our target users of this application include daily commuters, casual drivers, and delivery or ride-hailing drivers who face challenges finding parking in Singapore's high-demand areas.

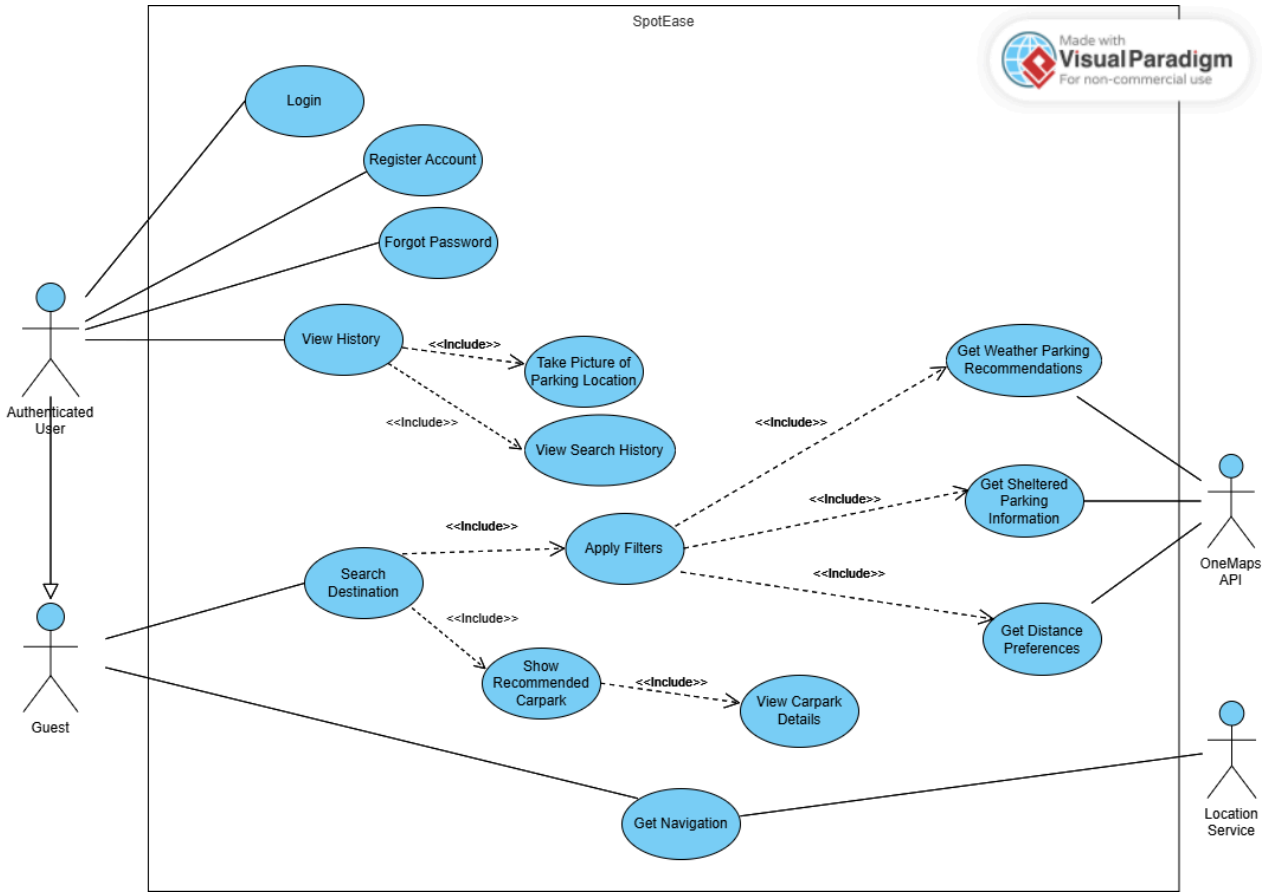
2. Overall Description

2.1 Product Perspective

SpotEase is a mobile application created to help users locate nearby car parks at a selected destination. Our app also shows the number of available lots of these car parks and customized accordingly to the weather and distance preference to aid their decision of selecting a desired car park.

SpotEase makes use of Application Programming Interfaces (API) such as OneMap API and NEA 2-hour Weather Forecast API for car park details as well as navigation.

2.2 Product Functions



SpotEase must minimally be able to:

- Register users with a new account
- Allow users to login to their account with their verified email and password
- Allow users to search for a location and display the details of car parks
- Allow users to reset their password for their account
- Allow users to logout of their accounts

2.3 User Classes and Characteristics

Primary user classes for the Spotease app include frequent drivers and tech-savvy users. Frequent Drivers, such as daily commuters and regular shoppers, will use the app often and prioritize fast, efficient parking searches and detailed information. Tech-savvy users will appreciate the filtering options for specific parking needs.

2.4 Operating Environment

SpotEase works on both Android and IOS devices as a mobile application. It uses MongoDB for user authentication. Access to external APIs such as OneMap API and NEA 2-hour Weather Forecast API are also required for full functionality of our SpotEase application.

2.5 Design and Implementation Constraints

SpotEase is currently available in English and thus might not be usable for users who are unfamiliar with English. Additionally, SpotEase depends on the NEA 2-hour Weather Forecast API to provide real time information on weather. Thus, any availability or accuracy issues with the API will result in the same issues on SpotEase.

2.6 User Documentation

SpotEase's source code has been made available at frontend link and backend link for public access. A README.md file has been provided to guide developers to start and clone the application and if they wish to make further improvements and changes to it. A guide can also be found in part 7 of this SRS.

2.7 Assumptions and Dependencies

2.7.1 Internet Connection

It is assumed that users of the application will have a stable and reliable internet connection to access the application.

2.7.2 Input from Users

It is assumed that users will provide accurate and valid input when using the application, such as entering a valid location in the search bar

2.7.3 Third-Party APIs

The application may have dependencies on third-party APIs for features such as Singapore's weather data. Any changes or updates to these third-party APIs may impact the functionality of the application.

2.7.4 External Services

The application may rely on external services, such as MongoDB, email notifications, which may have their own dependencies and limitations that can affect the overall performance and functionality of the application.

2.8 Software Engineering Practices

We followed a Waterfall approach during the initial planning phase to clearly define the overall requirements, high-level system architecture, and project timeline. This helped us set a structured foundation before diving into implementation.

We adopted an incremental development approach, where specification, development, and validation occurred concurrently. For instance, while one team member was implementing a function, others were simultaneously drafting specifications for the next function and validating the functionality of previously completed components. This workflow allowed for continuous progress and quick iteration.

We also structured our system using the three-layered architecture, dividing the program into three key components:

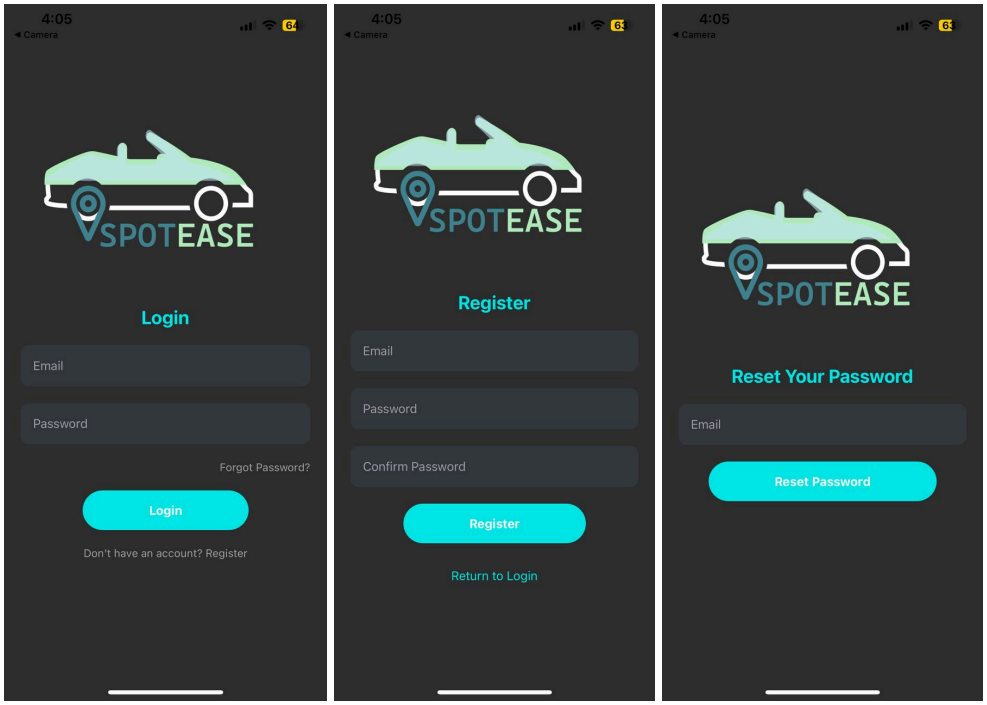
1. Presentation layer (Frontend): Handles the user interface and interaction
2. Application logic layer (Backend): Contains the core logic and processes, such as data handling and communication between layers.
3. Persistence layer (Database): Manages data storage, including user accounts and map data.

3. External Interface Requirements

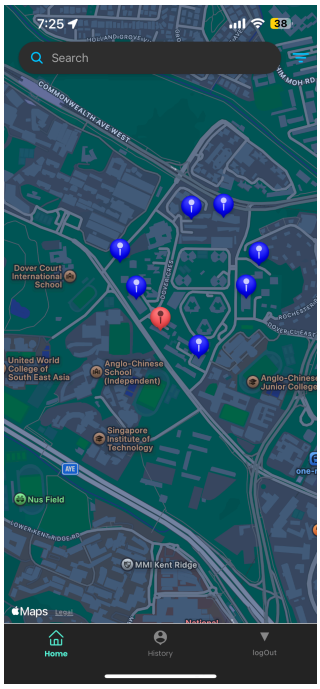
3.1 User Interfaces

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

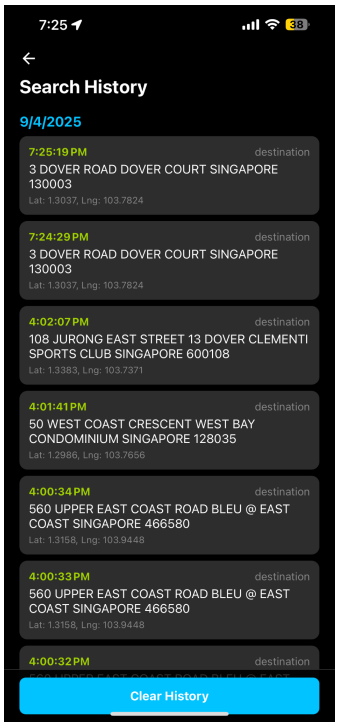
The screen one on the left shows the login screen where users are prompted to enter their email and password to login to their account. The middle screen shows the registration screen where users are required to input their details such as email, password and confirm password. The screen on the right shows the forget password screen where users are required to input their email to receive a recovery email and one-time password (OTP) for them to reset their passwords.

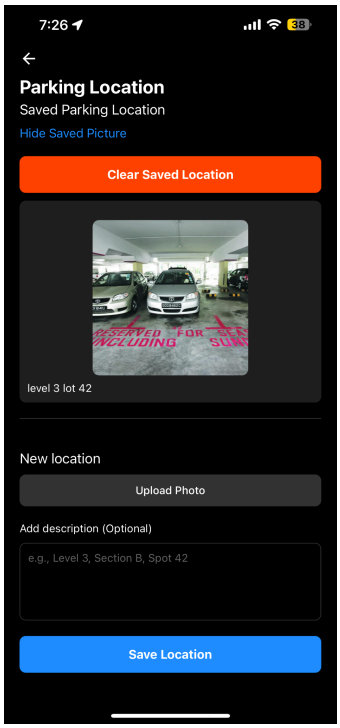


Moving on to the main screens as shown below, the screen on the left shows the map upon opening. After the user inputs a location into the search bar, a list of nearby car parks is listed out and **markers are placed on their locations to allow easier viewing by the user.**



Lastly, we have the history tab which consists of search history. Search history allows users to view their previously searched car parks while parking location allows users to capture a picture of their parking location and add optional description to it.





3.2 Hardware Interfaces

The logical and physical characteristics of the interfaces between the software product and the hardware components of the system in SpotEase includes the use of APIs to obtain data and services over the internet, and the use of standard communication protocols to interact with the hardware components of the device. The OneMap API provides geocoding, routing, and mapping services. This API converts user-input locations into geographic coordinates (latitude and longitude) using the WGS84 projection. The API enables real-time route suggestions to optimize navigation to nearby parking spaces. NEA 2-Hour Weather Forecast API supplies real-time weather updates, helping users make better parking choices by considering current or predicted weather conditions.

SpotEase runs on any device that supports React Native and JavaScript, including iOS and Android smartphones. It leverages standard APIs and protocols to ensure smooth integration, high responsiveness, and consistent performance across platforms.

3.3 Software Interfaces

SpotEase is built using React Native, a JavaScript framework for creating cross-platform mobile applications with a unified codebase. It handles the user interface, including navigation, real-time updates, and location-based features. MongoDB serves as the database, storing user profiles, parking preferences, search history, and real-time parking availability data. It enables efficient data management and retrieval to support personalized user experiences. SpotEase also integrates with external APIs such as OneMap API, Car park availability API and NEA 2-hour Weather Forecast API.

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

3.4 Communications Interfaces

SpotEase requires an internet connection and must be able to communicate with components such as routers and servers to connect to MongoDB. Data transfer rate will be determined by the user’s internet speed.

SpotEase also requires communication with external APIs such as OneMap API, Car park availability API and NEA 2-hour Weather Forecast API.

4. System Features

4.1 Login

4.1.1 The user must be able to register for an account
4.1.1.1 The user must input a valid email address
4.1.1.2 Passwords must include at least one uppercase letter, one number, one special character, and a length of 8 to 20 characters
4.1.1.3 The system must not allow identical email addresses

4.1.2 The user must be able to log in to their account
4.1.2.1 The system must be able to validate the corresponding username and password

4.1.3 The user must be able to continue with a guest account
4.1.3.1 When logging in with a guest account, their search history will not be saved.

4.1.4 The system must display a “Forgot Password” field
4.1.4.1 When ‘Forgot Password’ is chosen, the user must input their registered email address to receive an email from the system, allowing them to reset their password

4.2 Home Page

4.2.1 The system must display an interactive map on the home page, with a search bar and filter icon

4.2.2 The user must be able to zoom in or out on the map

4.2.3 The user must be able to move the map in all directions

4.2.4 The user’s current location must be shown on the interactive map

4.2.5 The system must display the history, home and logout tab at the bottom of the page

4.3 Search Page

4.3.1 User must be able to enter a location in search bar

4.3.2 Users can click on the filter button to choose sheltered parking and/or weather parking recommendations and a slider that allows users to set the radius for car park search results

4.3.3 The search bar must include placeholder text “Search” to guide users

4.4 Car Park Filtering Page

4.4.1 The system must provide 3 filter options for car parks

4.4.1.1 Weather Parking: The system recommends sheltered car parks to users when rain is forecasted within the next two hours.
4.4.1.2 Sheltered Parking: The system must allow users to filter car parks that have shelters
4.4.1.3 Distance Preference: The system filters and displays car parks based on the radius set by the user.

4.4.2 Filter parameters must be able to be toggled
4.4.2.1 The user must be able to enable or disable each filter option independently

4.5 Result Page

4.5.1 The system must display search result on a map view

4.5.2 The system must display car parks with available spaces within a 5000-meter radius of the user's input location
4.5.2.1 If no parking spaces are available within that 5000-meter search radius, the system would return no car parks.

4.5.3 Upon clicking the search button, the system must dynamically generate and display a dropdown menu with available car parks based on the user's search input and selected filters
4.5.3.1 If no parking spaces fulfill the specified filter requirements, the system would return no car parks.
4.5.3.2 The system must update the search results dynamically based on the selected filters

4.5.4 The system must display the available parking car parks as markers on the map

4.5.5 The system must display detailed information of each car park
4.5.5.1 Information such as the number of available parking spaces,sheltered parking, and distance from the user's current location and car park address should be displayed
4.5.5.2 The distance to the car park must be accurate to the nearest 0.1km

4.6 Navigation Page

4.6.1 The system must display a "Navigate" button when the user selects a car park from the search results

4.6.2 When the user clicks the "Navigate" button, the system must calculate the most optimal route from the user's current location to the selected car park
4.6.2.1 The system must determine the optimal route based on factors such as shortest distance and traffic conditions
4.6.2.2 The calculated route must account for the user's current location, which should be dynamically obtained using location services

4.6.3 The system must display the calculated route visually on the map interface

4.6.4 Turn-by-turn directions must be provided to guide the user to the selected car park

4.7 Parking Location Memory Feature Page

4.7.1 The system must allow users to capture and save a picture of their parking location and add an optional text description
4.7.1.1 The system must provide an option for the user to choose a picture of their parking location using their device photo album.
4.7.1.2 The system must allow users to edit the text description of the parking location

4.7.2 The system must attach a date and time stamp to the saved parking location entry

4.7.3 A photo of the parking location is stored locally on the device until the user manually presses “Clear Saved Location”.

4.8 History Page

4.8.1 The system must have a button from the home page to lead to a history page

4.8.2 The user must be able to see their search history in the history page
4.8.2.1 The search history must be sorted from the most recent search to oldest

4.8.3 Only users with registered accounts must be able to view their search history

4.8.4 Guest accounts must not have access to view search history

5. Other Nonfunctional Requirements

5.1 Performance Requirements

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

5.1.2 Response time of SpotEase should be no longer than 3 seconds after user's input

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

5.2 Safety Requirements

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

5.3 Security Requirements

5.3.1 SpotEase must ensure that every user has a valid email linked to a verified account

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

5.4 Software Quality Attributes

5.4.1 The user interface should have an intuitive and user-friendly interface. 80% of first-time users should be able to use the application effectively on their first try.

5.4.2 The default language of the system is English.

5.5 Business Rules

5.5.1 Users
All users should be able to search and view car park information.

5.5.2 System Administrators
System administrators configure system settings, manage API integration and oversee the general system performance.

6. Data Dictionary

Term	Definition
API	API stands for Application Programming Interface, which is a set of protocols for two or more computer programs to communicate with each other.
User	An individual who uses the application to search for the parking lot.
GPS	Global Positioning System
HDB	Housing & Development Board
Location Services	Services which provide GPS coordinates such as current coordinates and destination coordinates.
Starting Location	The point where the User wishes to begin his/her journey. It is his/her current physical location, as indicated by his/her phone GPS data.
Destination	The intended location the User wishes to park his/her car at the parking lot.
Travel Route	The path taken to get from the User's starting point to the intended destination. Upon gathering the User's input of the starting location and destination, the application will display a list of shortest time travel routes available.
Travel Time	The amount of time taken for the User to get from his/her starting point to his/her intended destination.
Result List	Contains all the possible travel routes between the starting location and the destination.
Result Entry	One of the possible travel routes between the starting location and the destination found in the result list.
Filter	A selection bar where users can selectively extract information that meets predefined criteria.
Total spaces	Total number of parking spaces in all the car parks.
Available spaces	Number of currently available parking spaces.
Sheltered	Indicates if the car park has sheltered parking.

7. Other Requirements

7.1 Installations

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

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

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

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

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

7.2 Running of App

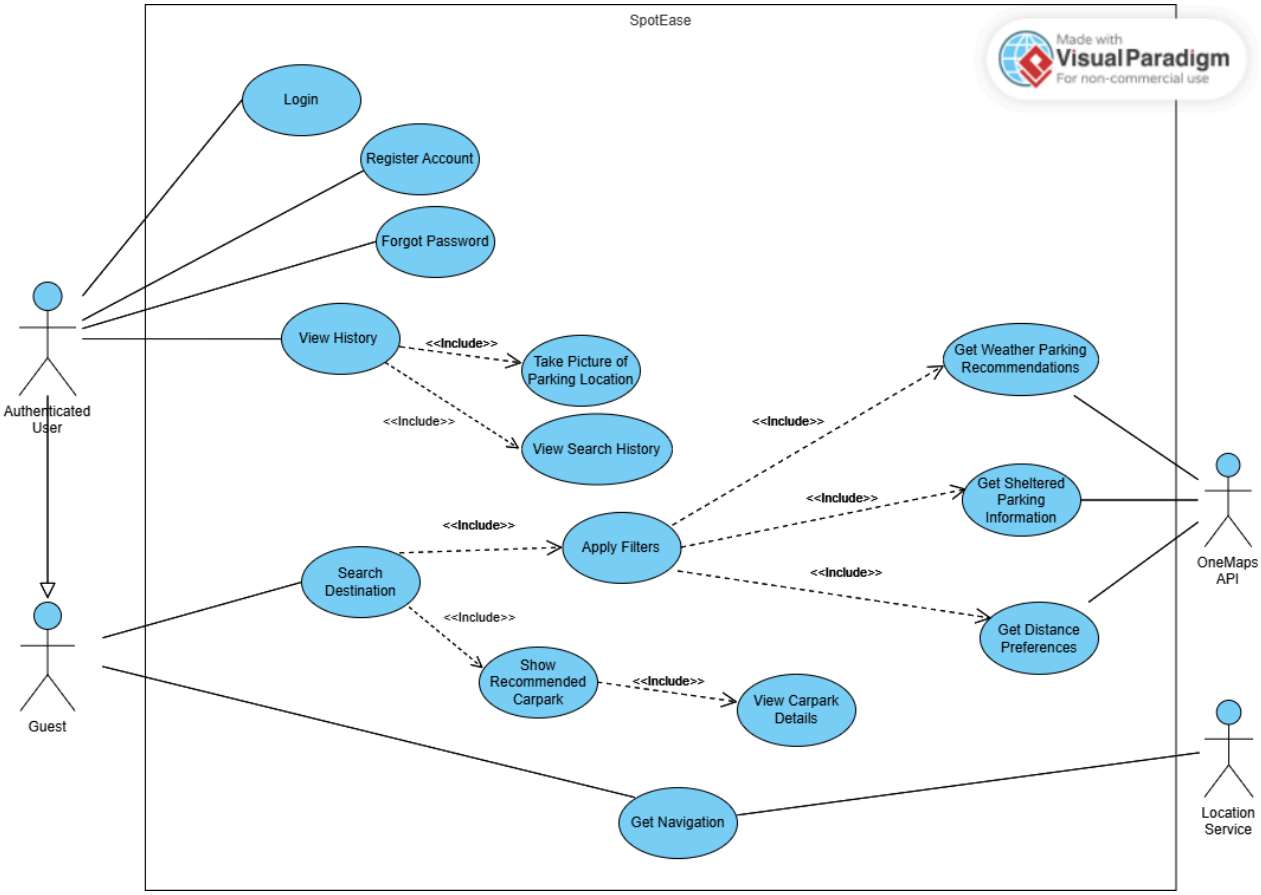
7.2.1 To run the app, enter the following command in your terminal: npx expo start

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

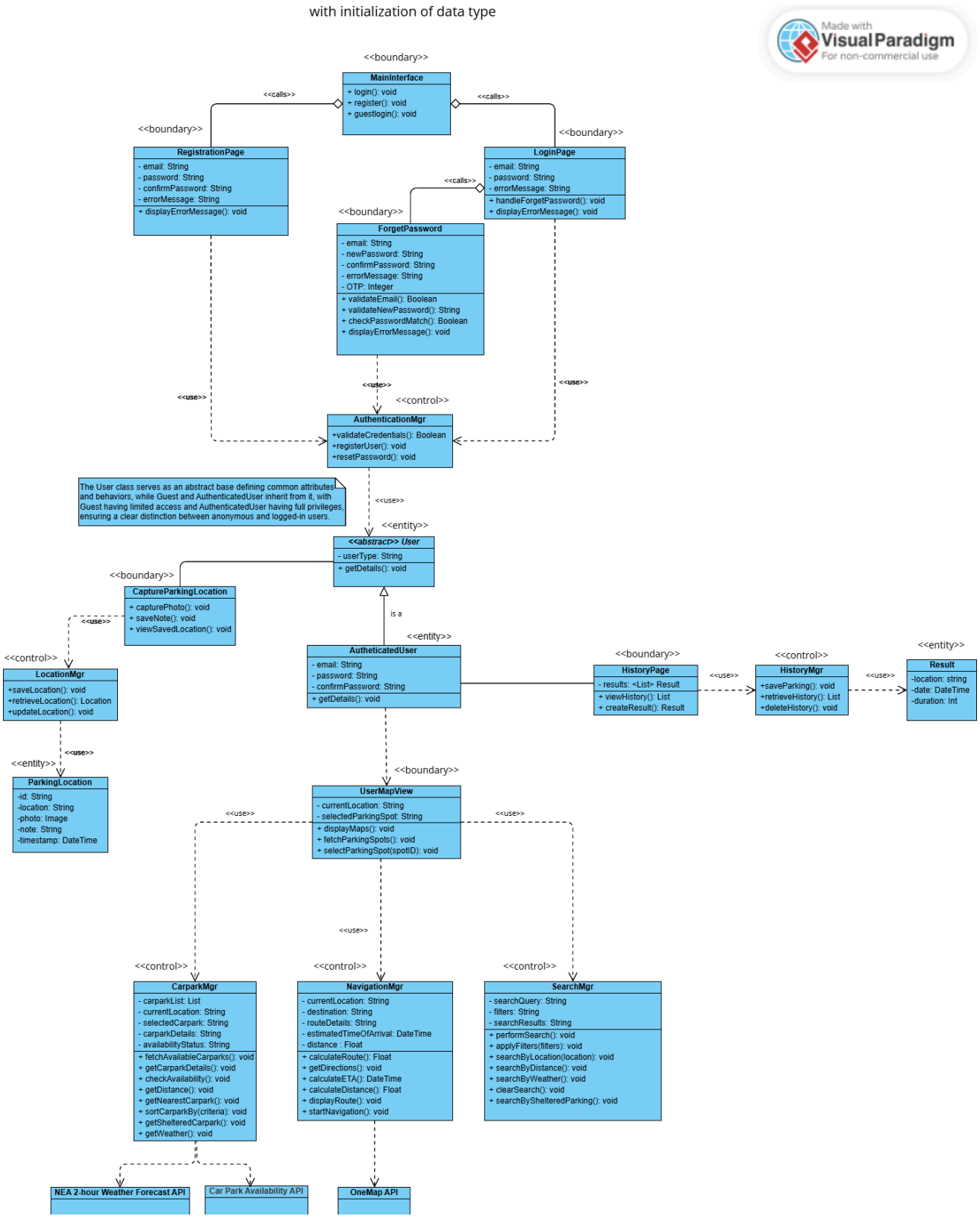
8. Appendix

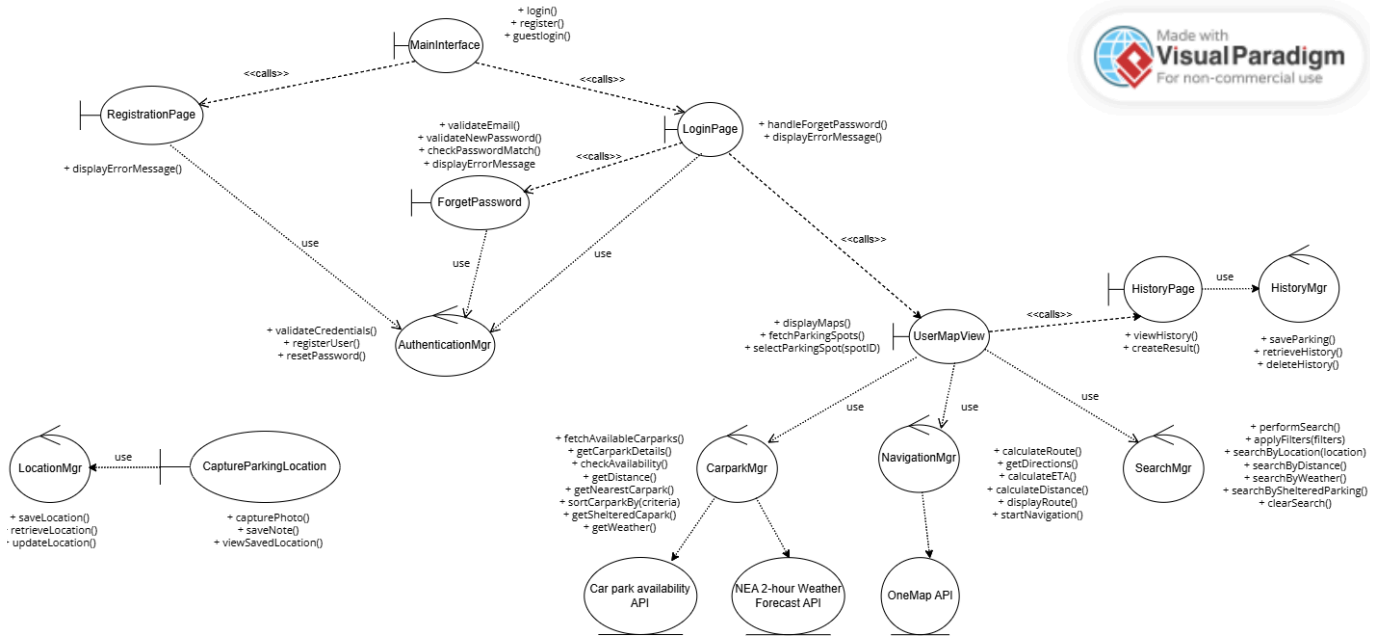
8.1 Use Case Diagram and Description

use case description document

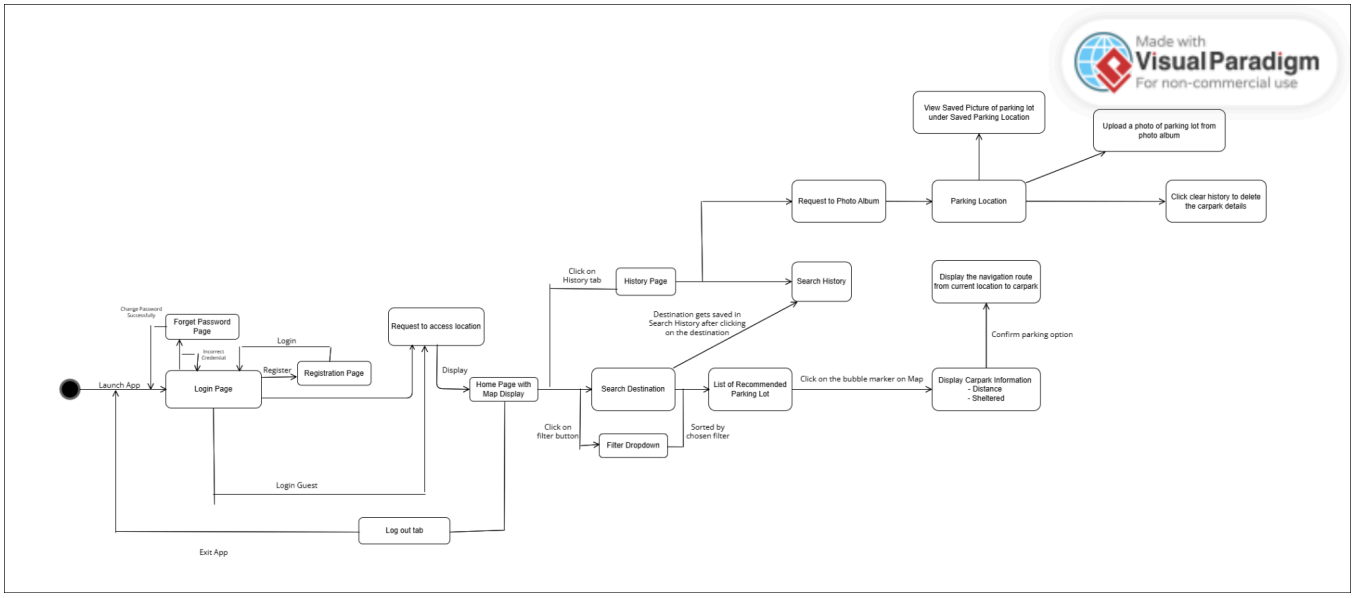


8.2 Class Diagram / Key boundary and control classes

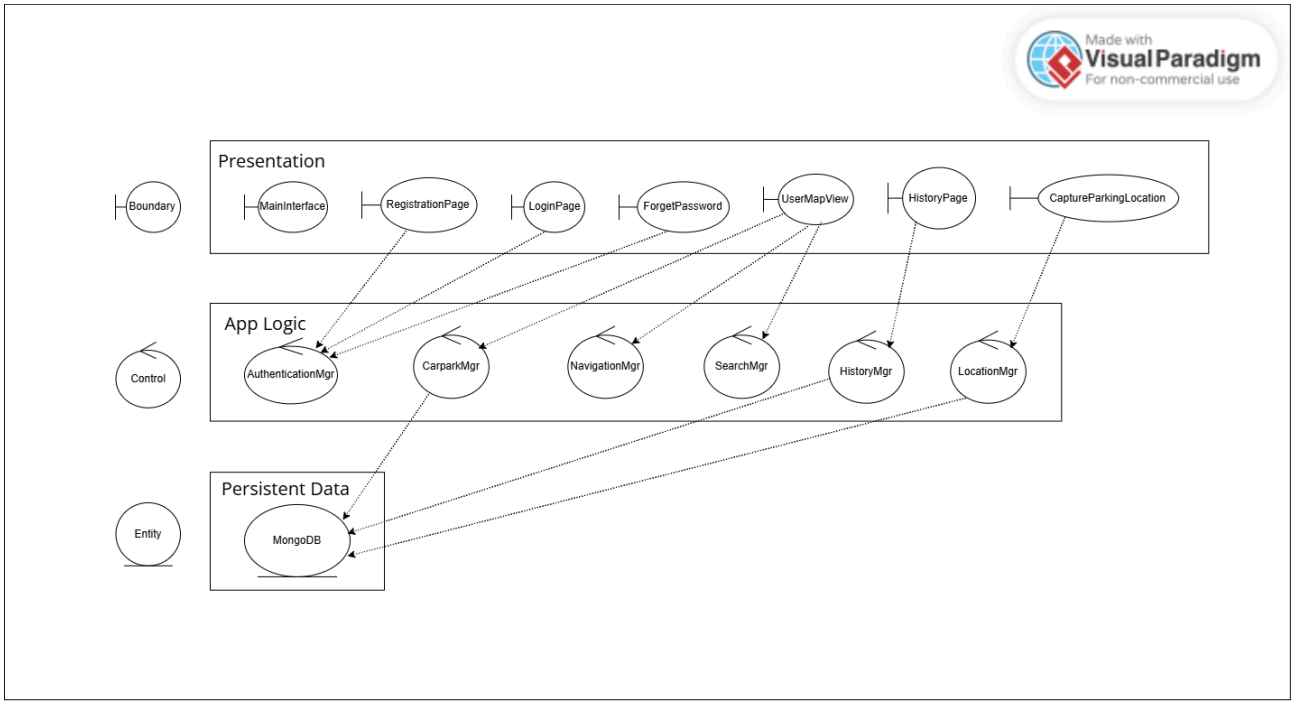




8.3 State Machine Diagram / Dialog Map

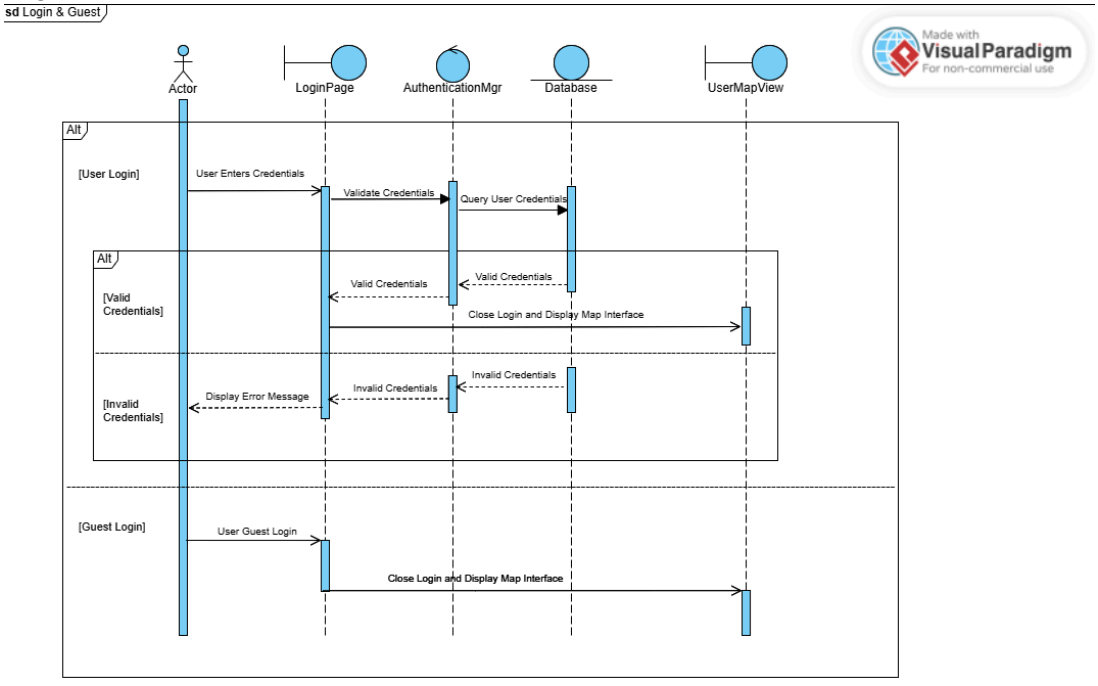


8.4 System Architecture Diagram

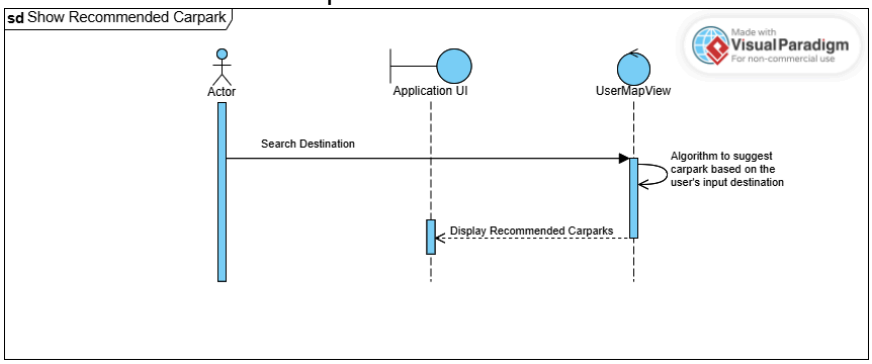


8.5 Sequence Diagrams

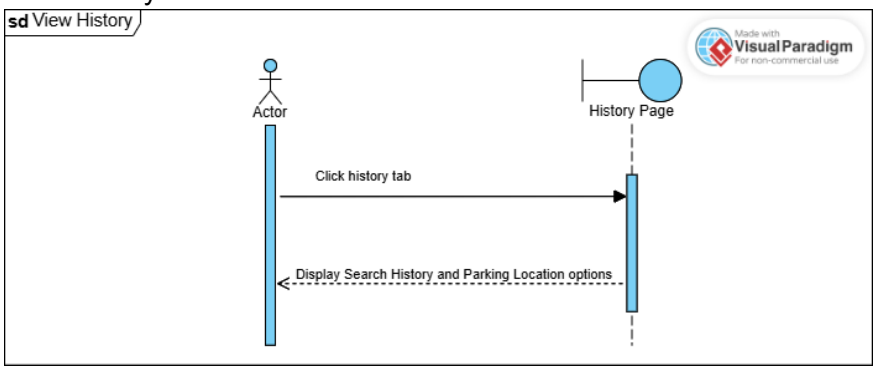
Login & Guest



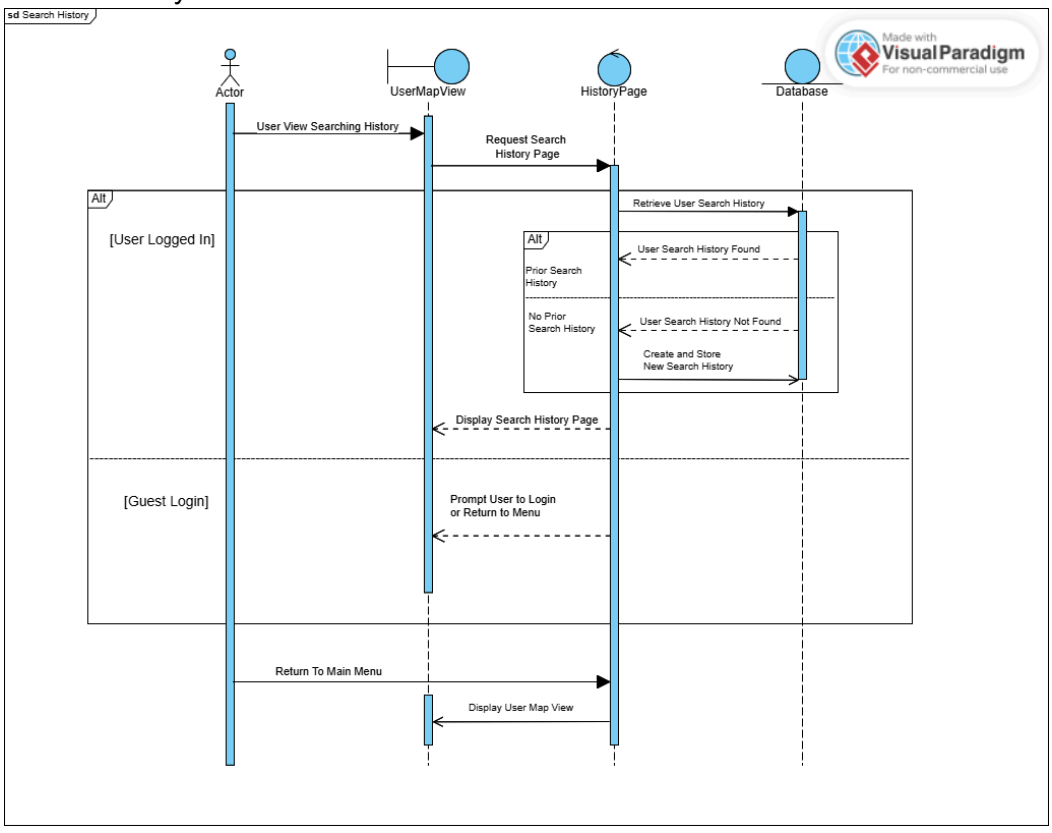
Show recommended car park



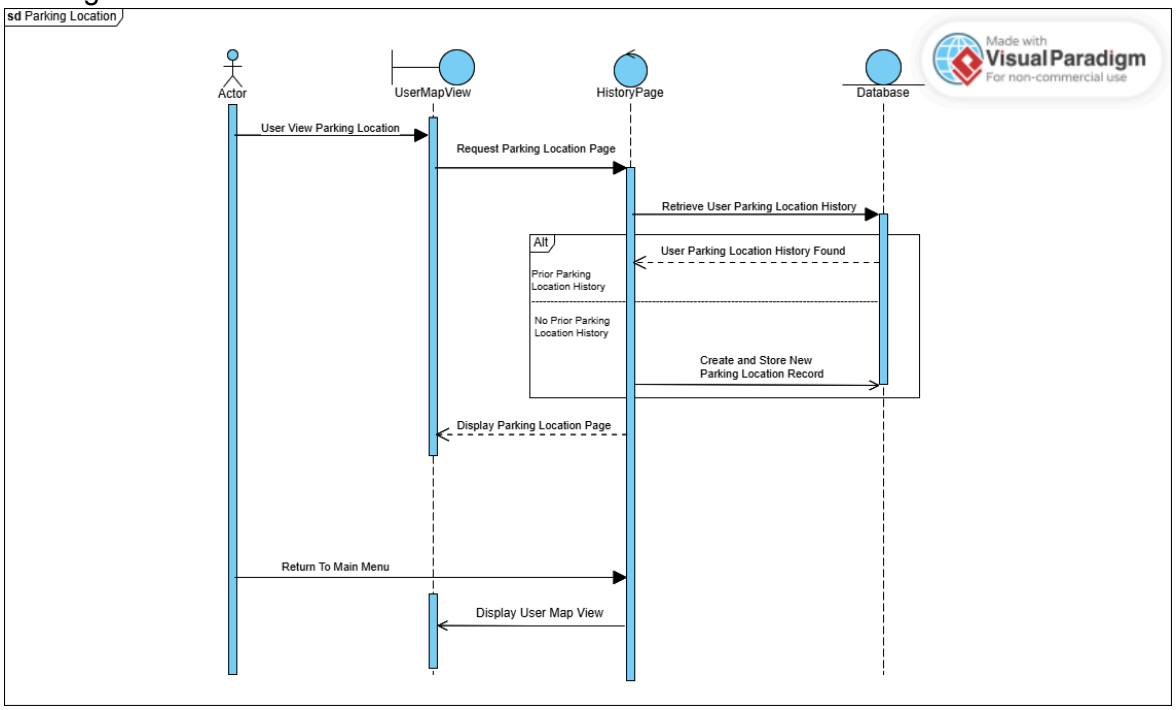
View history



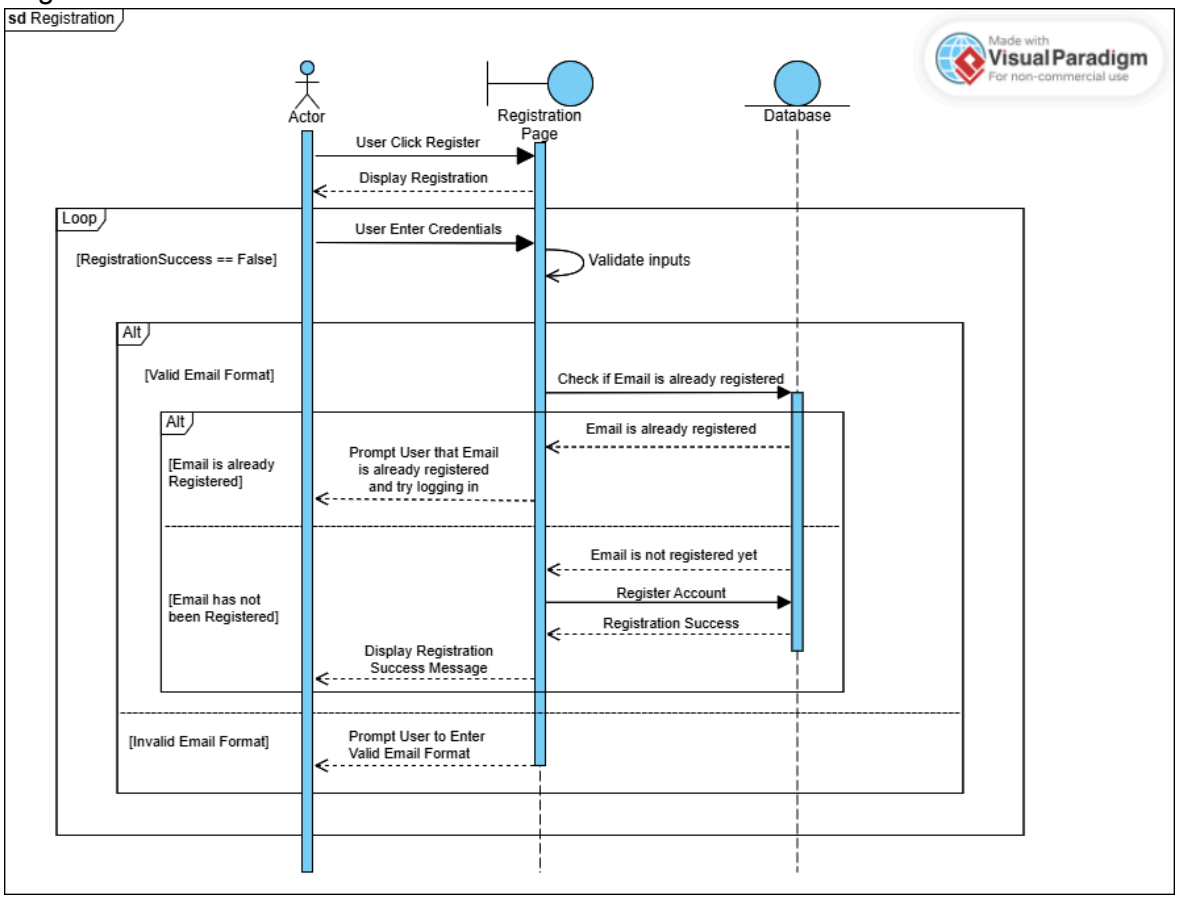
Search history



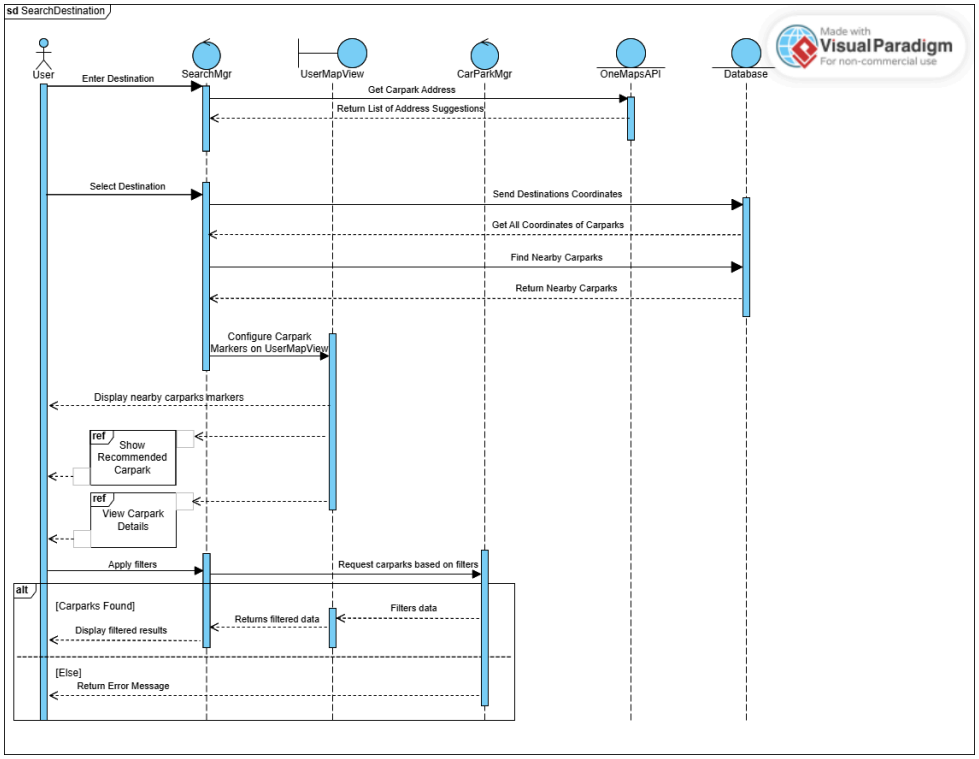
Parking location



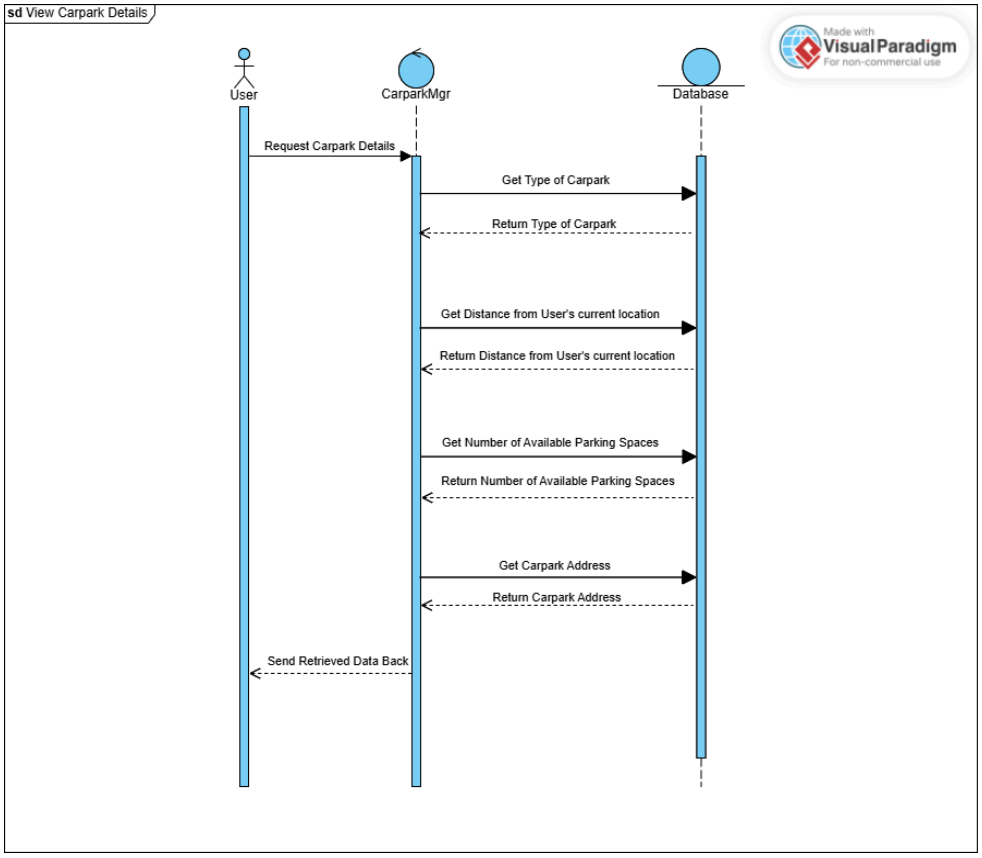
Registration



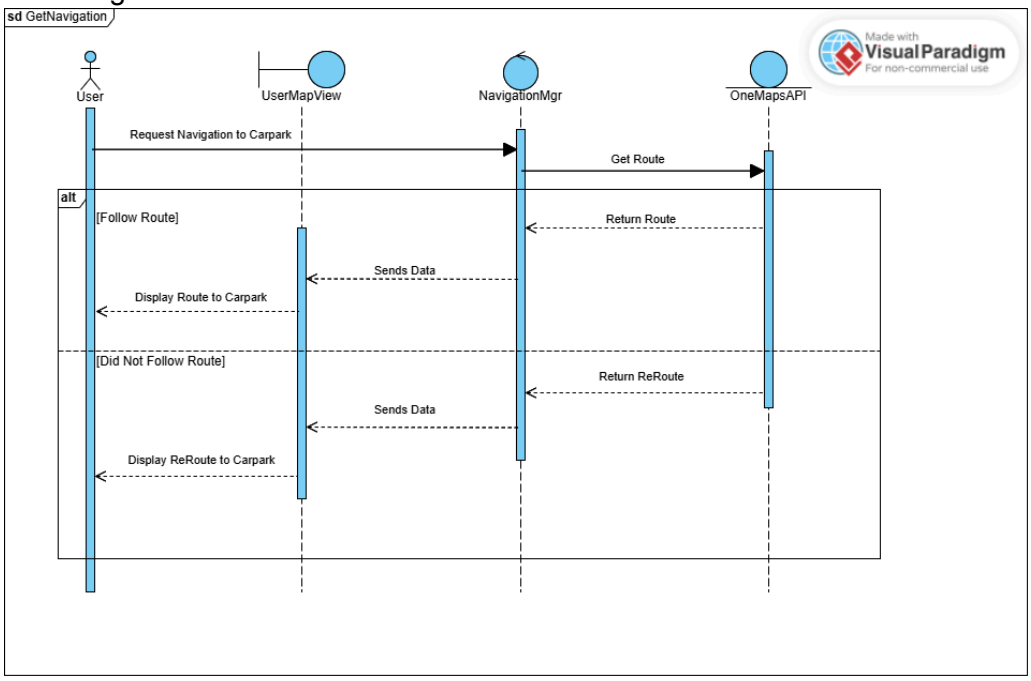
Search destination



View car park details



Get navigation



8.6 Unit Testing

8.6.1 Black Box Testing

1. Register Account

Input Parameters: Email, Password, Confirm Password

Test ID	Scenario	Expected Result	Actual Result	Pass / Fail
1a	User leaves any input parameters empty	The system displays an error message that says “Email and Password cannot be empty”	The system displays an error message that says “Email and Password cannot be empty”	Pass
1b	User registers with invalid email format (no @domain.com)	The system displays an error message that says “Invalid email format. Please enter a valid email (e.g., example@domain.com)”	The system displays an error message that says “Invalid email format. Please enter a valid email (e.g., example@domain.com)”	Pass
1c	User keys in password that does not match both ‘Password’ and ‘Confirm Password’ field	The system displays an error message that says “Passwords do not match! Please try again.”	The system displays an error message that says “Passwords do not match! Please try again.”	Pass
1d	User registers with a	The system displays an	The system displays an	Pass

	weak password (less than 8 characters, did not have at least one number and one special character)	error message that says “Password must be at least 8 characters long and contain at least one number and one special character.”	error message that says “Password must be at least 8 characters long and contain at least one number and one special character.”	
1e	User registers with an email that is already registered	The system displays an error message that says “User already exists! Try logging in.”	The system displays an error message that says “User already exists! Try logging in.”	Pass
1f	User registers with valid input fields	The system displays message “Account registered successfully”	The system displays message “Account registered successfully”	Pass

2. Login
Input parameters: Email, Password

Test ID	Scenario	Expected Result	Actual Result	Pass / Fail
2a	User leaves any input parameters empty	The system displays an error message that says “Email and Password cannot be empty. Try again.”	The system displays an error message that says “Email and Password cannot be empty. Try again.”	Pass
2b	User logins with invalid email format (no @domain.com)	The system displays an error message that says “Invalid email format. Please enter a valid email (e.g., example@domain.com)	The system displays an error message that says “Invalid email format. Please enter a valid email (e.g., example@domain.com)	Pass
2c	User logins with email that has not been registered yet	The system displays an error message that says “Failed to login”	The system displays an error message that says “Failed to login”	Pass
2d	User logins with invalid password	The system displays an error message that says “Failed to login”	The system displays an error message that says “Failed to login”	Pass
2e	User logins with valid input fields	The system displays the message “Login successful!” and application home page is loaded and map is displayed	The system displays the message “Login successful!” and application home page is loaded and map is displayed	Pass

3. Forgot Password
Input parameters: Email

Test ID	Scenario	Expected Result	Actual Result	Pass / Fail
3a	User keys in invalid email format (no @domain.com)	The system displays an error message that says “Invalid email format. Please enter a valid email (e.g., example@domain.com)	The system displays an error message that says “Invalid email format. Please enter a valid email (e.g., example@domain.com)	Pass
3b	User keys in valid email	The system displays the message “Check your email for the password reset OTP”	The system displays the message “Check your email for the password reset OTP”	Pass
3c	User keys in invalid email	The system displays an error message that says “User does not exists”	The system displays an error message that says “User does not exists”	Pass

4. Reset Password
Input parameters: OTP, Password, Confirm Password

Test ID	Scenario	Expected Result	Actual Result	Pass / Fail
4a	User did not key in OTP within 1 minute	The system displays the message “OTP has expired. Please request a new one.”	The system displays the message “OTP has expired. Please request a new one.”	Pass
4b	User leaves any input parameters empty	The system displays an error message that says “All fields are required. Try again.”	The system displays an error message that says “All fields are required. Try again.”	Pass
4c	User resets with a weak password (less than 8 characters, did not have at least one number and one special character)	The system displays an error message that says “Password must be at least 8 characters long and contain at least one number and one special character.”	The system displays an error message that says “Password must be at least 8 characters long and contain at least one number and one special character.”	Pass
4d	User keys in incorrect OTP	The system displays an error message that says “OTP is incorrect. Try again.”	The system displays an error message that says “OTP is incorrect. Try again.”	Pass

4e	User keys in password that does not match both ‘Password’ and ‘Confirm Password’ field	The system displays an error message that says “Passwords do not match! Please try again.”	The system displays an error message that says “Passwords do not match! Please try again.”	Pass
4f	User resets with valid input fields	The system displays the message “Password reset successfully” and is redirected to the login page	The system displays the message “Password reset successfully” and is redirected to the login page	Pass

5. Search Screen
Input parameters: Destination address

Test ID	Scenario	Expected Result	Actual Result	Pass / Fail
5a	User enters a valid destination in the search bar.	The system displays a dropdown list of nearby addresses based on the user’s input.	The system displays a dropdown list of nearby addresses based on the user’s input.	Pass
5b	User enters an invalid destination in the search bar.	The system returns no car parks.	The system returns no car parks.	Pass
5c	User selects a destination from the dropdown list.	The system displays nearby car parks based on the selected destination.	The system displays nearby car parks based on the selected destination.	Pass
5d	User views car park details for specific parking locations.	The system displays the car park address, number of available parking spaces, distance from the user's current location and type of car park.	The system displays the car park address, number of available parking spaces, distance from the user's current location and type of car park.	Pass
5e	User applies filter (ie. Sheltered parking, distance preferences and weather parking recommendations) to refine search results	The system processes filter criteria and displays filtered results, showing car parks that match the selected filters.	The system processes filter criteria and displays filtered results, showing car parks that match the selected filters.	Pass

6. Navigation
Input parameters: None

Test ID	Scenario	Expected Result	Actual Result	Pass / Fail
6a	User requests navigation to the selected car park.	The system displays the navigation interface which includes the map view with current location, selected car park location, recommended route, alternative routes, visual instructions and distance to next turn.	The system displays the navigation interface which includes the map view with current location, selected car park location, recommended route, alternative routes, visual instructions and distance to next turn.	Pass
6b	User deviates from route	The system recalculates routes and displays the navigation interface which includes the map view with current location, selected car park location, recommended route, alternative routes, visual instructions and distance to next turn.	The system recalculates routes and displays the navigation interface which includes the map view with current location, selected car park location, recommended route, alternative routes, visual instructions and distance to next turn.	Pass

7. History
Input parameters: None

Test ID	Scenario	Expected Result	Actual Result	Pass / Fail
7a	User selects the “History” tab from the bottom of the app.	The system displays the following options, Search History and Parking Location	The system displays the following options, Search History and Parking Location	Pass

8. Search History
Input parameters: None

Test ID	Scenario	Expected Result	Actual Result	Pass / Fail
8a	Authenticated users select the “Search History” option from the history page and has prior search history	The system retrieves search history from stored records and displays a list of past searches including the date and time of	The system retrieves search history from stored records and displays a list of past searches including the date and time of	Pass

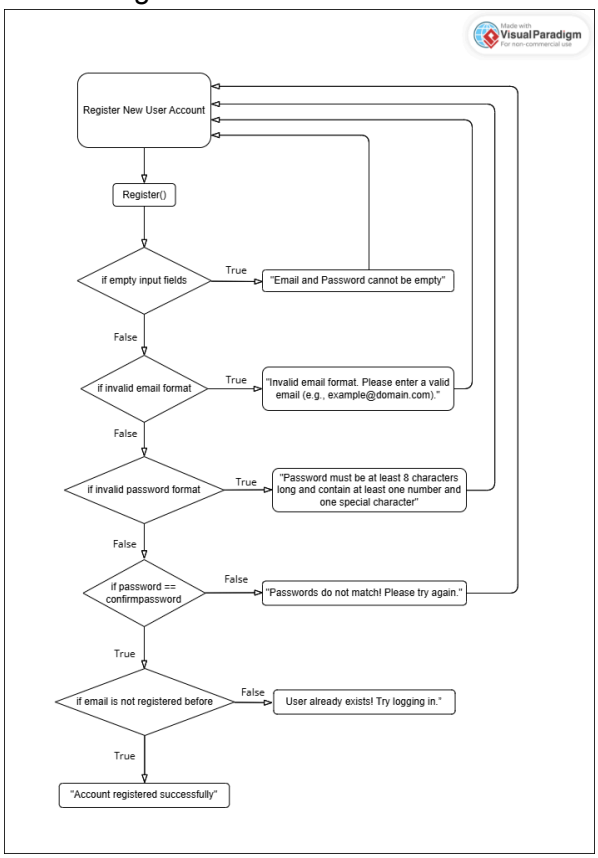
		search location as well as the destination.	search location as well as the destination.	
8b	Authenticated users select the “Search History” option from the history page and does not have prior search history	The system displays a message that says “No search history available.”	The system displays a message that says “No search history available.”	Pass
8b	Guest users select the “Search History” option from the history page	The system prompts users that they are not authenticated.	The system prompts users that they are not authenticated.	Pass

9. Parking Location
Input parameters:

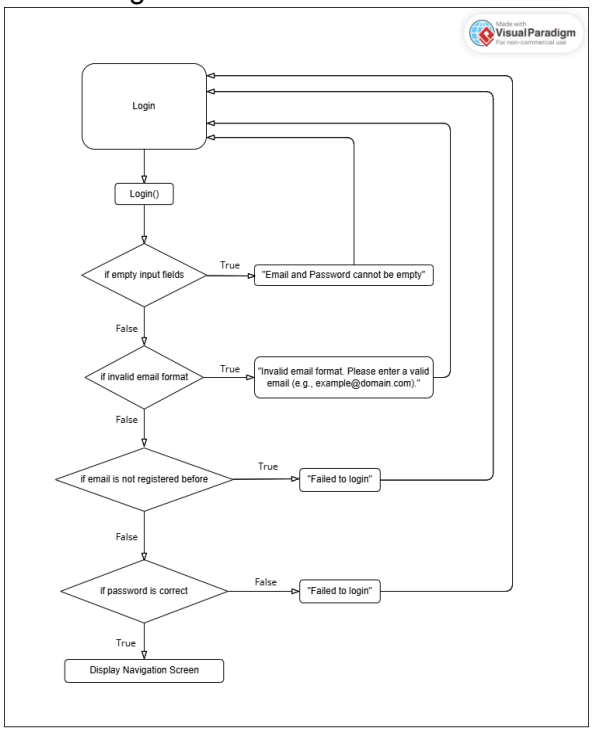
Test ID	Scenario	Expected Result	Actual Result	Pass / Fail
9a	Users select the “Parking Location” option in the history page.	The system displays the following options: “New location” and “Add description”.	The system displays the following options: “New location” and “Add description”.	Pass
9b	Users select the “New location” and “Add description” options.	The system prompts users to grant access to their photo album and select a photo to upload. Users can also optionally provide a description. The system displays a successful message that says “Parking location saved successfully!”	The system prompts users to grant access to their photo album and select a photo to upload. Users can also optionally provide a description. The system displays a successful message that says “Parking location saved successfully!”	Pass
9c	Users selects the “Clear Saved Location”	The system deletes the saved parking location and description.	The system deletes the saved parking location and description.	Pass

8.6.2 White Box Testing

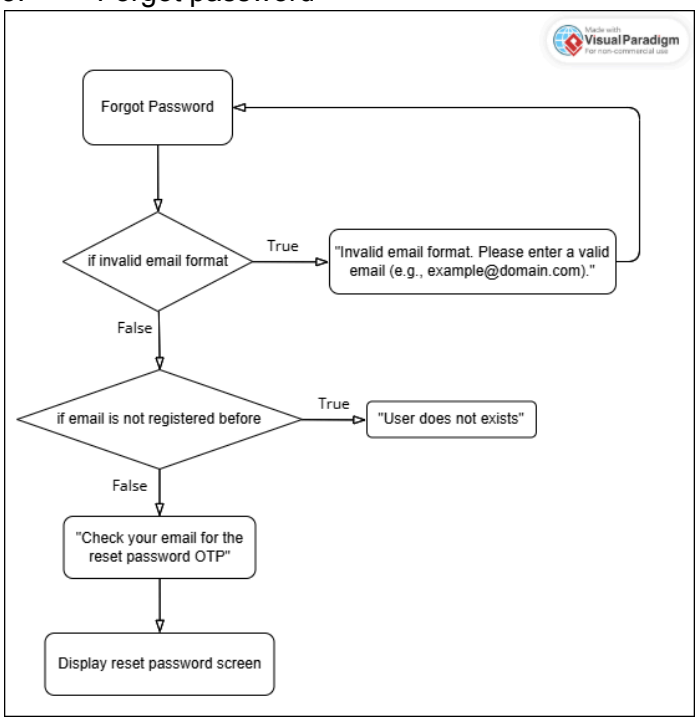
1. Registration



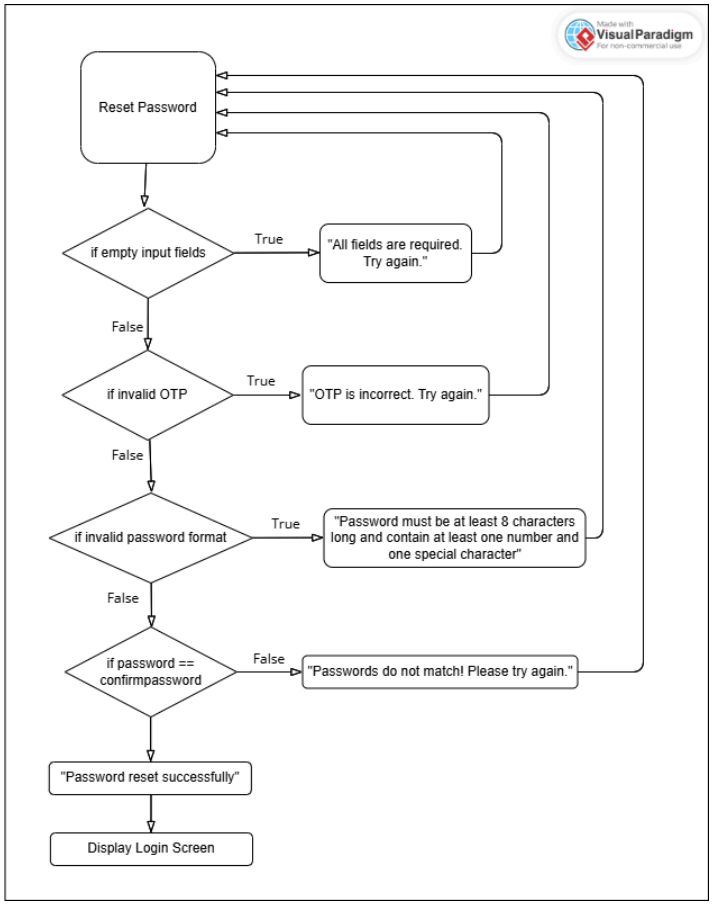
2. Login



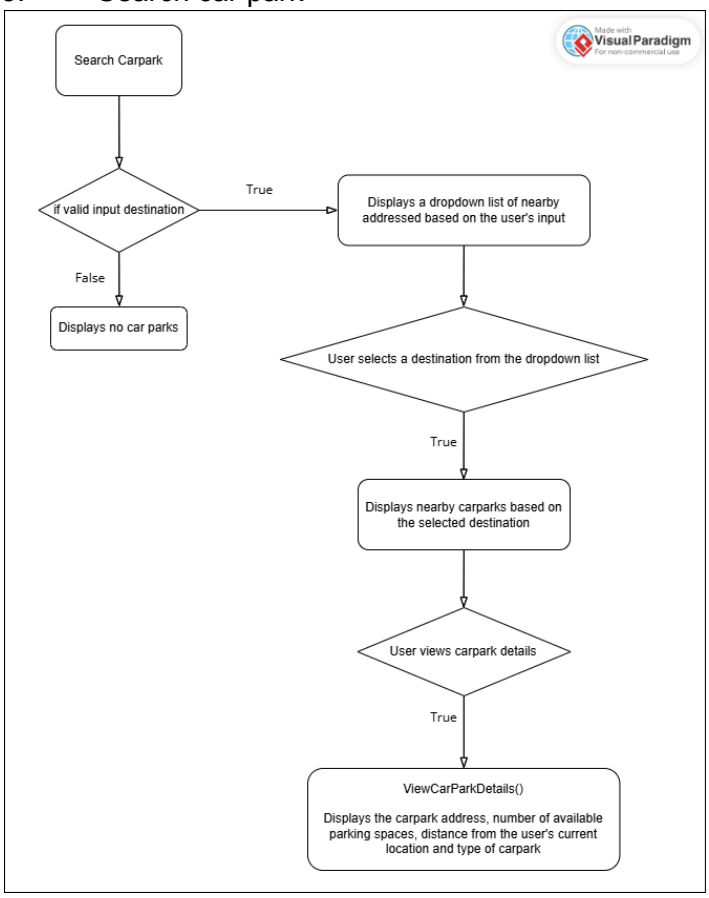
3. Forgot password



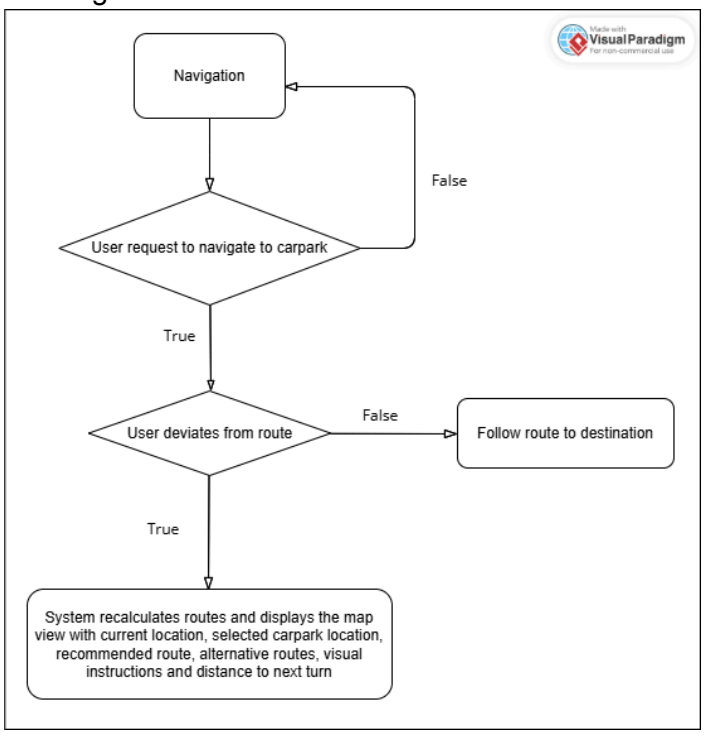
4. Reset password



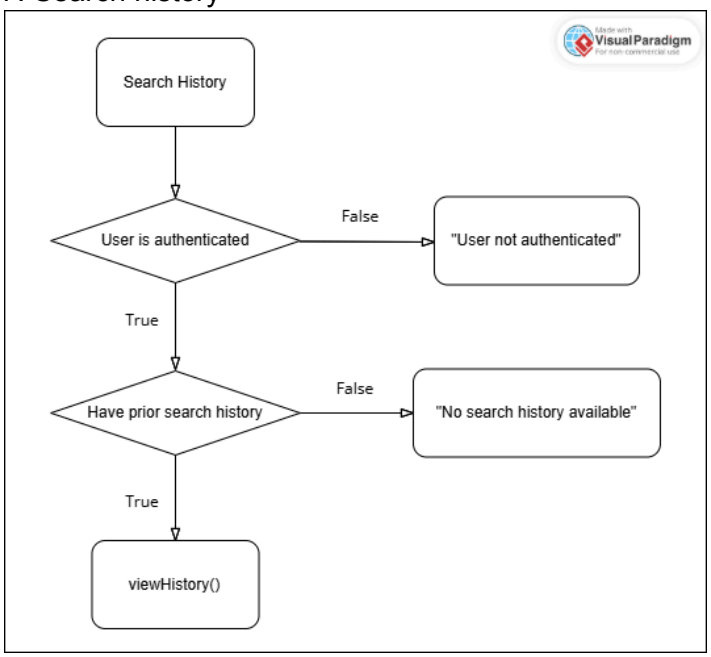
5. Search car park



6. Navigation



7. Search history



8.7 App Demo

The demonstration of our App can be viewed at the following link.

8.8 Project Management

Our team maintained effective project management through regular weekly meetings, where we discussed upcoming tasks, reviewed progress, and planned our next steps. These meetings helped ensure everyone was aligned and could contribute effectively to different parts of the project.

For version control, we used GitHub and worked mainly on the main branch, updating it periodically with new features and fixes. To prevent data loss or accidental overwrites, we made it a point to regularly back up our code, either through manual backups or by saving copies in separate folders. Although we did not use separate branches, we communicated clearly about who was working on what to avoid merge conflicts and maintain smooth collaboration.

This combination of consistent communication and disciplined code management helped keep the project organized and on track.