Software Requirements Specification

for

ParkSmart

Version 1.2 approved

Prepared by

Lee Heng Sheng, Brandon
Tan Wen Rong
Aditi Vasudevan
Ng Qi Cheng
Goh Yue Jun Bradley
Chia Jia Yuun

Nanyang Technological University, Team ParkSmart Dev Team Limited

13/04/2025

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Brandon, Wen Rong, Qi Cheng	04/02/2025	Documented the introduction: purpose, document convention, target user, functional requirements, non-functional requirements, and UI interfaces.	1.0
Bradley, Jia Yuun, Aditi	26/03/2025	Updated the overall descriptions, UI interfaces, and System features.	1.1
Brandon,Wen Rong, Qi Cheng, Bradley, Jia Yuun, Aditi	06/04/2024	Final updates to all documentations.	1.2

1. Introduction

1.1 Purpose

The purpose of this document is to outline the software requirements for Version 1.0 of the ParkSmart web application. It defines the intended functionality, scope, and constraints of the system, serving as a reference point for stakeholders throughout the development lifecycle. By capturing both functional and non-functional requirements, this document facilitates alignment between technical and business teams to ensure the final product meets user and operational expectations.

1.2 Document Conventions

The document follows the following conventions for clarity and consistency.

Font	Times New Roman
Line Spacing	1.5
Heading 1	Font Size: 18, Font Weight: bold
Heading 2	Font Size: 14, Font Weight: bold
Heading 3	Font Size: 12
Content	Font Size: 12

1.3 Intended Audience and Reading Suggestions

This document is intended for all stakeholders of the application, including three primary user groups: drivers (end users) and application system administrators, as well as application developers, project managers, testers, and the marketing staff.

Drivers: Our primary users, comprising daily drivers and visitors who need real-time car park availability and useful car park information. Information relevant to driver functions and features is outlined in the following sections:

- 1. Introduction
- 2.6 User Documentation
- 3.1 User Interfaces
- 4. System Features.

System Administrator: Manages the forum's posts and comments, along with feedback and reports to ensure the smooth operation of the application. Information relevant to system administration functions and features is outlined in the following sections:

- 1. Introduction
- 2.6 User Documentation
- 3.1 User Interfaces
- 4. System Features

App Developers: To obtain a comprehensive understanding of the application, it would be beneficial to review the entire document. However, developers can focus on development-focused sections in the following sequence:

- 3. External Interface Requirements
- 4. System Features
- 5. Non-Functional Requirements
- Appendix A. Data Dictionary

Project Managers: To oversee the ParkSmart Project, project managers need a broad understanding of the application. However, they can focus on the user-focused sections in the following sequence:

- 2. Overall Description
- 3. External Interface Requirements
- 4. System Features

Testers: Test the application to ensure that it fulfils the user requirements. Testers can focus on the following sections in the following sequence:

- 4. System Features
- 5. Non-Functional Requirements

Marketing Staff: Responsible for marketing the application to our target users. We recommend focusing on the following sections in the sequence:

- 1. Introduction
- 2. Overall Description
- 3.1 User Interfaces

1.4 Product Scope

ParkSmart is a web application designed to enhance the parking experience at HDB car parks in Singapore by providing real-time lot availability and detailed car park information, such as location, operating hours, gantry height, and payment options. It supports user account features like sign-up, login, and password management with built-in security, as well as accessibility enhancements like language selection and dark/light mode. A community forum allows users to communicate with each other while a dedicated support page enables users to report technical issues or suggest improvements. The application is optimized for cross-browser compatibility and responsive across all devices.

1.5 References

1. MongoDB Documentation: https://www.mongodb.com/docs/
2. ExpressJs Documentation: https://expressjs.com/
3. React Documentation: https://react.dev/
4. NodeJs Documentation: https://nodejs.org/docs/latest/api/
5. Tailwind Documentation: https://tailwindcss.com/
6. Car Park Availability API: https://data.gov.sg/datasets/d_ca933a644e55d34fe21f28b8052fac63/view
7. HDB Car park Information API: https://data.gov.sg/datasets/d_23f946fa557947f93a8043bbef41dd09/view
8. OneMap Search API: https://www.onemap.gov.sg/apidocs/search
9. OneMap Advanced Minimap API: https://www.onemap.gov.sg/apidocs/maps/#advancedMiniMap

10. OneMap Authentication API: https://www.onemap.gov.sg/apidocs/authentication

2. Overall Description

2.1 Product Perspective

This application aims to improve the parking experience at HDB car parks in Singapore by allowing users to view real-time availability of parking lots and access detailed information about each car park, such as location, operating hours, gantry height, free parking, night parking option and payment types.

Additionally, the application will include user account management functionalities such as sign-up, login, and password changes, complete with security measures to protect user data. Features such as language selection and light/dark mode are included to enhance convenience and accessibility for drivers seeking their preferred parking at Singapore's HDB car parks.

To encourage community engagement, a forum feature will be included where users can share car park-related issues (e.g., faulty gantries or lot sensors), share tips, or provide reviews about specific car parks. This crowdsourced feedback mechanism helps keep information accurate and up-to-date.

A dedicated support page will be available for users to provide feedback, report technical issues, or suggest new features. This ensures the continuous improvement of the application based on real user needs and experiences.

To cater to a broad user base, the website must be compatible across widely used browsers to ensure a seamless user experience. It must also be responsive on all device viewports.

2.2 Product Functions

ParkSmart provides role-based access, with distinct features available to administrators and standard users.

(End) Users:

1. Users are able to create a new account to access the platform.
2. Existing users are able to login with their existing credentials.
3. Users are able to remain logged in even after closing the site for up to an hour.
4. Users are able to update their personal details if required.
5. Users are able to toggle between light and dark mode.
6. Users are able to select their preferred language setting (English, Chinese, Bahasa Melayu, Hindi)
7. Users can search for a specific location to retrieve nearby HDB car parks.

8. Users are able to filter the search based on their preferences, such as free parking, availability, and more.
9. Users are able to view detailed car park information about each car park, such as:
    a. Distance
    b. Availability (available lots / total lots)
    c. Short Term Parking
    d. Free Parking
    e. Night Parking
    f. Car Park Type
    g. Gantry Height (Height Limit)
    h. Payment Type
    i. Car Park Decks
    j. Basement
10. Users are able to browse forum content posted by other users.
11. Users are able to post content on the forum.
12. Users are able to edit their own posts on the forum.
13. Users are able to comment on other users' posts on the forum.
14. Users are able to edit their own comments on the forum.
15. Users are able to report other users' posts and comments on the forum.
16. Users are able to access a support page, where they are provided with answers to Frequently Asked Questions (FAQ).
17. Users are able to provide feedback to Admins on the support page.

Admin:

1. Admin is able to login using the email admin@parksmart.com and password Admin@123.
2. Admin is able to access most of the same features as Users (1-11, 13, 16-17).
3. Admin is able to moderate forum content by deleting Users' posts and comments.
4. Admin is able to access a dashboard on support page, where they are able to view:
    a. Total Feedback
    b. Average User Rating
    c. Recent Feedback, with information about:
        i. Date
        ii. Name
        iii. Email
        iv. Subject
        v. Message
        vi. Rating
    d. User reports, with information about:
        i. Username
        ii. Post Title
        iii. Comment Text
        iv. Reason
        v. Date
    e. Admin is able to navigate to the post using a View button.

2.3 User Classes and Characteristics

Target Audience: Singaporean Drivers, Visitors who require car park information

Frequency of Use:

- Daily
- Usage peaks during commuting hours and weekends

Product Functions Used:

- Sign up and login, remain logged in (1-3)
- Update personal and vehicle information (4)
- Toggle between light and dark mode (5)
- Customise language preferences (6)
- Search for parking facilities by location (7)
- Filter search based on preferences (8)
- View detailed parking facility information (9)
- Access real-time parking availability (9b)
- View and participate in community forum (10-15)
- Access frequently asked questions (16)
- Submit feedback to administrators (17)

Characteristics:

- Age Range: 18 to 60+ years
- Vehicle Types: Cars, motorcycles, trucks, and other motorized vehicles
- Technical Expertise: Low to moderate
- Security/Privilege Level: Basic user privileges
- Educational Level: Varied
- Experience: Ranges from new to experienced drivers

Target Audience: Administrator

Frequency of Use:

- Daily

Product Functions Used:

- Moderate forum content (Admin 3)
- Access feedback and report information (Admin 4)
- Sign up and login, remain logged in (User 1-3)
- Update personal and vehicle information (User 4)
- Toggle between light and dark mode (User 5)
- Customise language preferences (User 6)
- Search for parking facilities by location (User 7)
- Filter search based on preferences (User 8)
- View detailed parking facility information (User 9)
- Access real-time parking availability (User 9b)
- View and participate in community forum (User 10-11, User 13)
- Access frequently asked questions (User 16)
- Submit feedback to administrators (User 17)

Characteristics:

- Age Range: 18 to 60+ years
- Technical Expertise: High
- Security/Privilege Level: Elevated system access with administrative privileges
- Educational Level: Typically tertiary education or equivalent experience
- Experience: Familiar with system management and user oversight

2.4 Operating Environment

2.4.1 Production Environment of *ParkSmart*

Setting	Description

Browser Support	The application is designed to be compatible with modern browsers. It does not rely on any major features that lack support.
Accessibility Support	The application supports light/dark modes, automatically using the user's system preferences by default. The application supports four different languages, and saves the user preference even after the user leaves the site. The application uses a modern user interface with a global header and footer, allowing for easy navigation.
Screen Support	The application is optimised for modern mobile devices within the following specifications: ● Minimum supported resolution: 320x544 pixels ● Maximum supported resolution: 2560x1440 pixels
Connectivity Requirements	The application requires an active internet connection to communicate with the backend services.

2.4.2 Development Environment of *ParkSmart*

This subsection describes the setting of which the web application is built and tested on during the development phase.

Tech Stack Overview

Front-end

Front-end development using React, a JavaScript library maintained by Meta for building user

interfaces with reusable components. The front-end stack incorporates:

- JavaScript (ECMAScript 2023) as the core programming language
- HTML5 for content structure and markup
- CSS3 for styling and visual presentation
- TailwindCSS (version 3.4.0) as a utility-first CSS framework for rapid styling

Back-end

Back-end development using Node.js, a JavaScript runtime for server-side applications, featuring:

- Node.js (version 20.17.0) for executing JavaScript on the server
- Express.js (version 4.18.3) as a minimalist web framework for handling HTTP requests and routing

Database

Database implementation using MongoDB, a NoSQL document-oriented database system:

- MongoDB (version 7.0) for storing data in flexible, JSON-like documents without requiring a predefined schema

This represents a complete MERN stack (MongoDB, Express.js, React, Node.js) implementation, providing a JavaScript-based solution across the entire application stack.

2.5 Design and Implementation Constraints

The development and deployment of the ParkSmart application are subject to the following constraints:

API Rate Limit:

External APIs used in the application, such as the HDB Car Park Availability API and OneMap Search API, impose rate limits. High concurrency from many users may cause the number of requests to exceed the rate limit, affecting real-time data retrieval.

External API Downtime and Updates:
The application relies on third-party APIs, which may undergo downtime and updates. This may cause disruptions to the services which rely on APIs, such as the signing up, logging in, the search functionality, and more.

Hosting Limitations:
The backend is currently deployed on Render on a free payment plan. This means that the server spins down during periods of inactivity, and it takes a while to start up. This means that the first request after idle periods may be delayed, which may affect services such as signing up or logging in.

Cloud Database Availability:
External downtime or updates of the cloud database will affect services that rely on the database for information retrieval or updates.

2.5.1 Design Principles and Patterns

The code is to follow the following design patterns to enhance code maintainability and reusability:

- **Model-View-Controller (MVC):** The MVC pattern structures the system into three core parts: The Model (MongoDB schemas via Mongoose) handles data storage and retrieval, The View (React frontend) manages user interaction and display, and The Controller (Express route handlers) processes user requests and coordinates responses.
- **Repository Pattern:** The Repository Pattern helps us manage how we access and work with data in a consistent way. These repositories provide a simple interface to access data. If we ever need to change how we get the data, we only need to update the repository, and everything else in the system will continue to work smoothly. This makes our system more flexible and easier to maintain.

- **Strategy Pattern:** The Strategy Pattern is a way of making the system more flexible by allowing it to switch between different methods or strategies without modifying the core logic.

The system will also incorporate **SOLID design principles** such as:

- Single Responsibility Principle (SRP): each class should have only one well-defined responsibility/function.
- Open-Closed Principle (OCP): to allow for extension of new functionality while being closed for modification of existing code.
- Interface Segregation Principle (ISP): ensures clients only use the interfaces they require.
- Dependency Inversion Principle (DIP): ensures that high-level modules do not depend on low-level modules, such that they both rely on abstraction, building a loosely-coupled application.

2.6 User Documentation

We have created a README.md file in our GitHub Repository to guide users on how to set up the project. The following are the main steps.

2.6.1 Front-End Set Up

2.6.2

1. Clone the repository to your local machine.

   ```
   git clone https://github.com/softwarelab3/2006-FDAE-D2.git
   cd 2006-FDAE-D2
   ```

2. Install the required dependencies:

   ```
   cd lab5/app
   npm i
   ```

3. Running the Front End (after finishing Back-End set up)

Open your browser and go to http://localhost:5173 (or the port you're using for the React app). If you are on a Macbook, do not use Safari. Safari automatically blocks insecure localhost connections (see issue). If you still want to use Safari, you may try this workaround by adding

```
server: {
  hmr: {
    host: 'localhost',
  },
},
```

to the vite.config.js file. Otherwise, we recommend accessing it on other browsers.

2.6.3 Back-End Set Up

Set up MongoDB (Optional)

Our app uses a MongoDB cloud database by default. As a result, you do not need to download MongoDB by default. However, if the cloud database is down, it defaults to connecting to a local database.

If you wish to use your own local database, connect to localhost:27017 in MongoDB Compass and create a new database with the name sc2006. You may initialise your first collection as MyCollection (the name does not really matter).

If you wish to observe the cloud database, connect to mongodb+srv://wenrongtan16:7F5vZcLpytTXXl9z@wr.re7utjp.mongodb.net/. You may then observe the populated data in the cloud database.

1. Set up environment variables

   Enter the following commands in the app directory.

   ```
   echo LOCAL_MONGO_URI="mongodb://localhost:27017/sc2006" > .env
   echo
   CLOUD_MONGO_URI="mongodb+srv://wenrongtan16:7F5vZcLpytTXXl9z@wr.re7utjp.mongodb.net/sc20
   06?retryWrites=true&w=majority" >> .env
   echo ONEMAP_EMAIL="brandon02.lee@gmail.com" >> .env
   echo ONEMAP_PASSWORD="WQ!*PJwF7a#k*@" >> .env
   ```

```
echo JWT_SECRET=123456 >> .env
echo JWT_EXPIRATION=1h >> .env
echo PORT=3000 >> .env
```

2. Start the application

   Ensure you are in the app/ directory. Open up two terminals. On one terminal, run

   ```
   node server/server.js
   ```

   This connects to the server. On the other terminal, run

   ```
   npx vite
   ```

   This runs React.

Backend: Saved data is viewable on MongoDB Compass if you have connected using the correct Mongo URIs.

2.7 Assumptions and Dependencies

The ParkSmart development team developed the web application with the following assumptions:

1. Technical assumptions
   - The application will be used on a modern web browser.
   - Users will have a stable and strong internet connection, as most data is retrieved from the server.
   - API responses will be consistent.
   - The browser will have JavaScript enabled to support interactive features.

2. Infrastructure assumptions
   - Full-scale development will be supported by reliable hosting infrastructure.
   - The OneMap API services will remain available and maintain their current API structure, or be forward compatible.
   - The MongoDB database will be hosted on a dependable cloud platform.

3. User Assumptions

- Users are expected to be proficient in at least one of the four supported languages.
- Users will have valid login credentials.
- Users will be willing to provide basic profile information to participate in the forum.
- Users will engage in constructive discussions related to parking.

4. Business assumptions
    - Forum administrators will provide accurate and up-to-date information on HDB parking.
    - Content moderation and reporting features will operate as expected.
    - User authentication services will remain fully operational.

2.7.1 Front-End Dependencies

The dependencies can be found in the app/package.json file.

Dependency	Version	Purpose
react	19.0.0	JavaScript library for building user interfaces.
react-dom	19.0.0	DOM-specific methods for React.
react-router-dom	7.3.0	Routing library for React applications.
react-i18next	15.4.1	React internationalisation framework for translations.
i18next	24.2.3	Internationalisation-framework written in and for JavaScript.
react-password-checklist	1.8.1	A React Component to display the success or failure of password strength rules that updates as a user types.

geolib	3.3.4	Library to provide basic geospatial operations like distance calculation, conversion of decimal coordinates to sexagesimal and vice versa, etc.
axios	^1.8.3	Handles HTTP requests and responses between frontend and backend.
browser-image-compression	2.0.2	Javascript module to be run in the web browser for image compression.
tailwindcss	4.0.7	A utility-first CSS framework for rapidly building custom user interfaces.
@tailwindcss/vite	4.0.7	Tailwind plugin for Vite.
vite	6.2.2	Frontend build tool that significantly improves the frontend development experience.
@emotion/react	11.14.0	A library designed for writing css styles with JavaScript.
@emotion/styled	11.14.0	The styled API for @emotion/react.
@mui/material	6.4.8	A comprehensive suite of free UI tools.
@material-tailwind/react	2.1.10	An open-source library that uses Tailwind and React.

2.7.2 Back-End Dependencies

The dependencies can be found in the app/package.json file.

Dependency	Version	Purpose

express	4.21.2	Web framework for Node.js
mongoose	8.12.1	MongoDB object modeling for Node.js
bcrypt	5.1.1	Password hashing library for secure authentication
jsonwebtoken	9.0.2	Implementation of JSON Web Tokens for authentication
cors	2.8.5	Enable Cross-Origin Resource Sharing
dotenv	16.4.7	Environment variable management
multer	1.4.5-lts.1	Middleware for handling multipart/form-data (file uploads)

2.7.3 Dev Dependencies

The dependencies can be found in the app/package.json file.

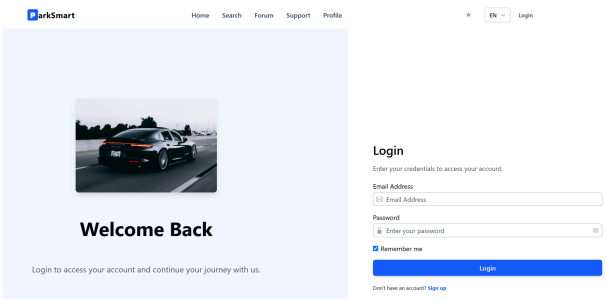
Dependency	Version	Purpose
@eslint/js	9.19.0	Official ESLint config for JavaScript, providing core linting rules.
eslint	9.19.0	The main linter that analyzes code to find and fix problems.
eslint-plugin-react	7.37.4	Adds React-specific linting rules.
eslint-plugin-react-hooks	5.0.0	Enforces the Rules of Hooks for React components.
eslint-plugin-react-refresh	0.4.18	ESLint plugin for ensuring safe usage of React Refresh (Fast Refresh).

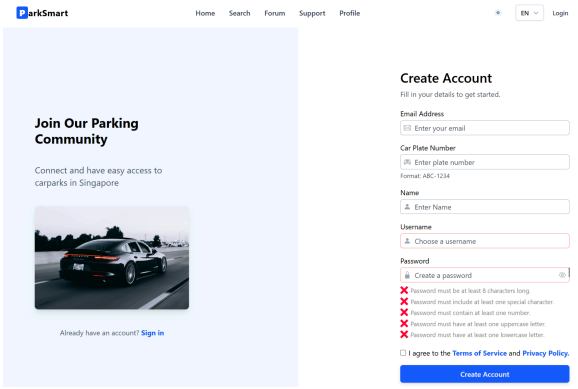
globals	15.14.0	Provides a list of global variables to help ESLint understand various environments.
prettier	3.5.3	Code formatter for consistent code style.
prettier-plugin-tailwindcss	0.6.11	Sorts Tailwind CSS classes automatically within Prettier.
vite	6.2.2	Frontend build tool that significantly improves the frontend development experience.
@types/react	19.0.8	TypeScript definitions for React.
@types/react-dom	19.0.3	TypeScript definitions for ReactDOM.
@vitejs/plugin-react	4.3.4	Vite plugin to enable React support with fast refresh and JSX transform.
i18next-browser-languagedetector	8.0.4	Detects user language from browser settings.
i18next-http-backend	3.0.2	Loads translations via HTTP (useful for async translation loading).

3. External Interface Requirements

3.1 Interfaces

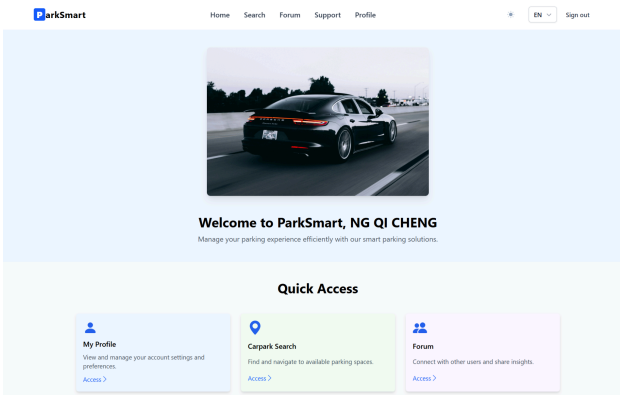
3.1.1 User Interfaces

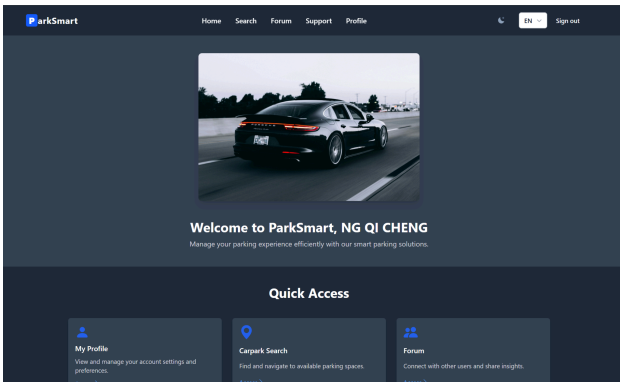
<table>
<tr>
<td> 
</td>
<td>Login / Sign-up Option
● When users open ParkSmart, they may login with an existing account. Alternatively, new users may Sign Up with a new account.

Login Page:
● Users can enter their email and password for their existing account to login.
● Once logged in, users do not need to relogin for up to an hour even after exiting the website</td>
</tr>
<tr>
<td> 
</td>
<td>Sign-up Page:
● If users click ‘Sign Up’ on the Login page, they will navigate to this page where they may key their email, car plate number, name, username and password to create a new account.
● Users may not use an existing email or username
● Users must ensure the password has:
○ 8 characters
○ 1 special character</td>
</tr>
</table>

	○ 1 number ○ 1 uppercase letter ○ One lower case letter

3.1.2 User Interfaces





Home Page

Image 1:

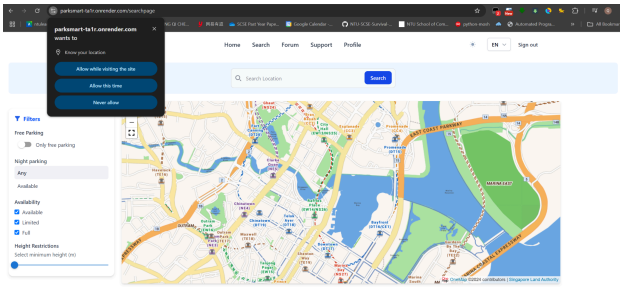
- Provides access to all ParkSmart functions including profile, carpark search, forum and feedback page
- Users may also access these features through the navigation header bar

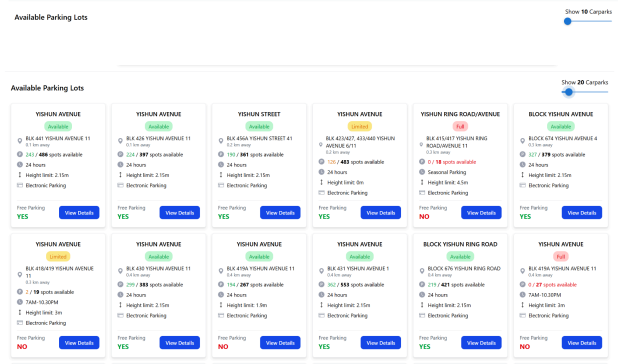
Image 2:

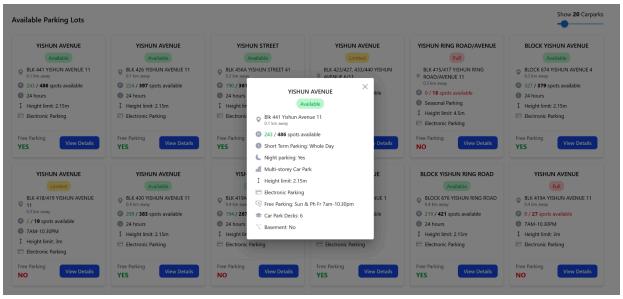
- Users can click the sun/moon icon on the navigation bar to toggle between dark/light mode

Image 3:

- User can click change language on the navigation bar to change the language of all displayed text to the the 4 national languages







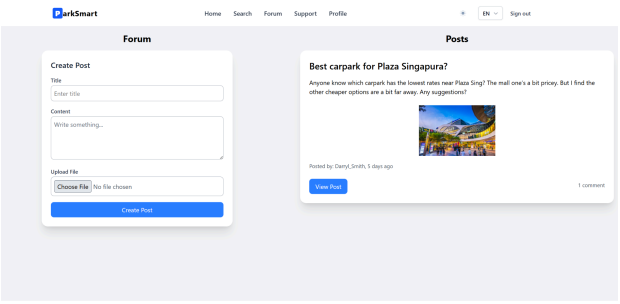
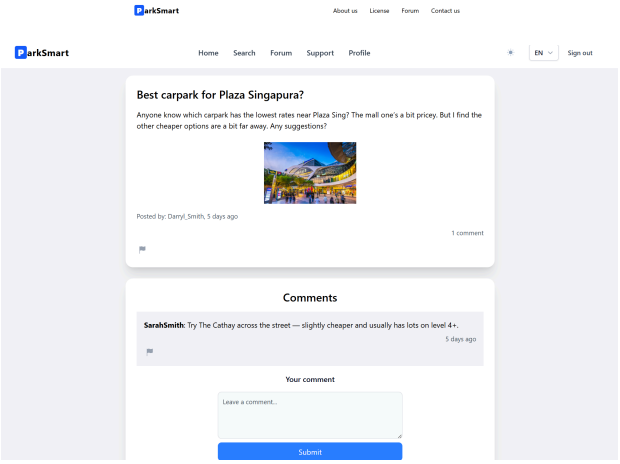
Carpark Search Page

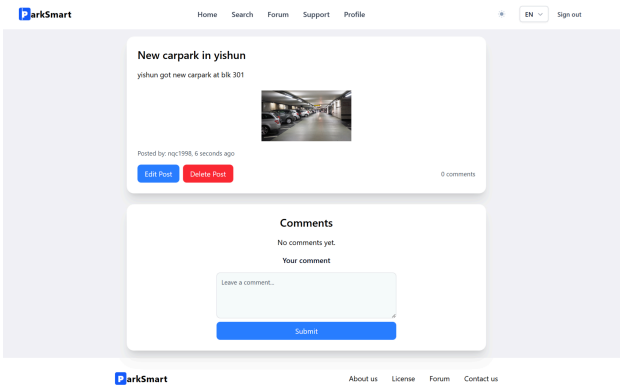
First Image:

- Shows a map which is automatically loaded to the user's current location, shown by a red marker (user must allow location permissions)
- Users may access the search bar above the map to search for a location, this will be loaded to the map
- Users may filter the resulting car parks using the filters on the left of the map, based on:
    - Free parking
    - Night parking
    - Availability
    - Gantry Height

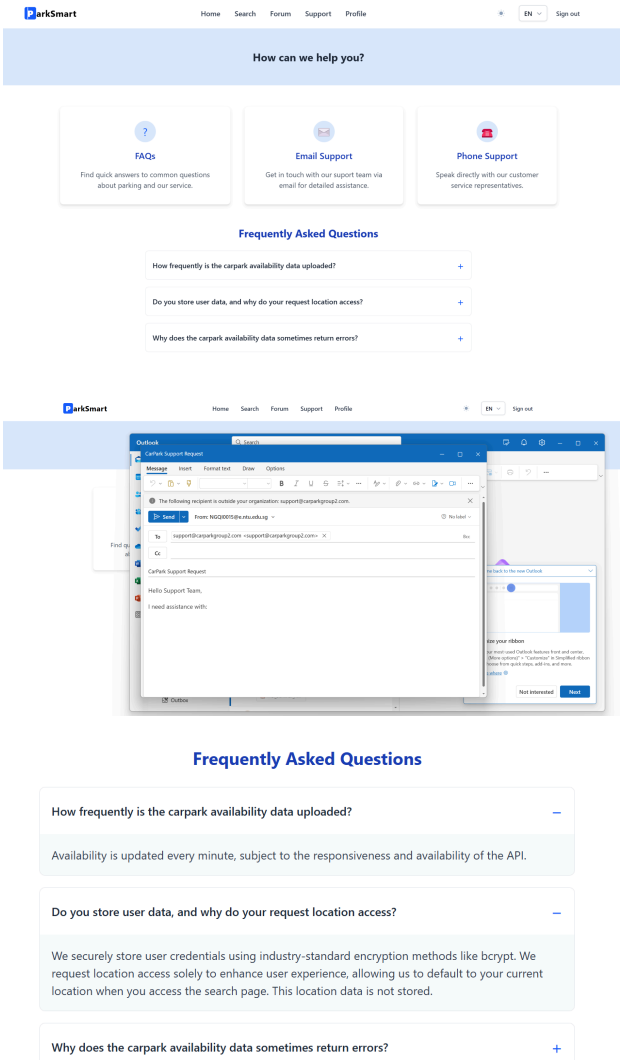
Second Image:

- As the user scrolls down, they will see the resulting car parks listed in order of distance to their current loaded map location. Each carpark will display information including:
    - Location
    - Availability
    - Timing
    - Gantry Height limit
    - Parking Type
- Users may use the slider on the top right corner to adjust the number of displayed car parks

	Third Image: ● When a user clicks ‘View More’ on a car park they may see additional details including: ○ Night Parking ○ Free parking ○ Car Park decks ○ Basement
  	**Forum Page** First Image: ● Shows the posts displayed by users sharing their car park experiences ● On the left, users may add their own post by entering a post title, post content and attaching a file(optional) then clicking submit post Second Image (View Post Page): ● When user clicks ‘View Post’ they can see the full contents of a post and the comments below ● Users can comment on a post by entering a submitting in the new comment field ● Users can delete their comment by clicking the bin icon ● Users can report posts/comments by clicking the flag icon Third Image: ● If the user views their own post they may delete/edit their post by clicking



the respective button



Support Page (User POV)

First Image:

- Displays methods of support users may take including faq, phone support and email support

Second Image:

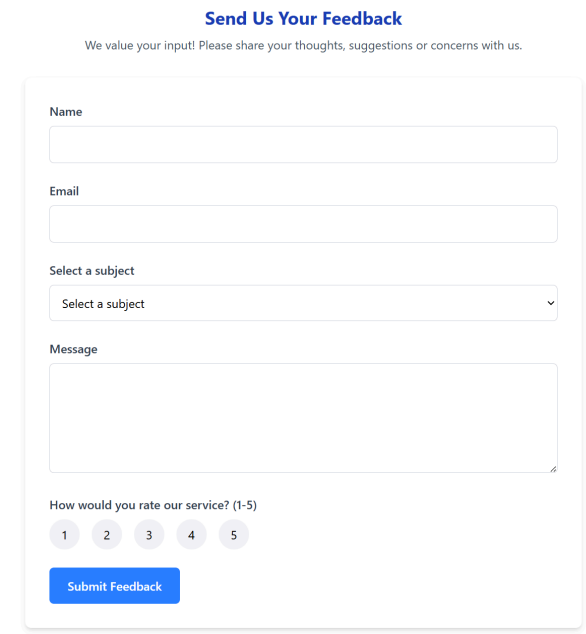
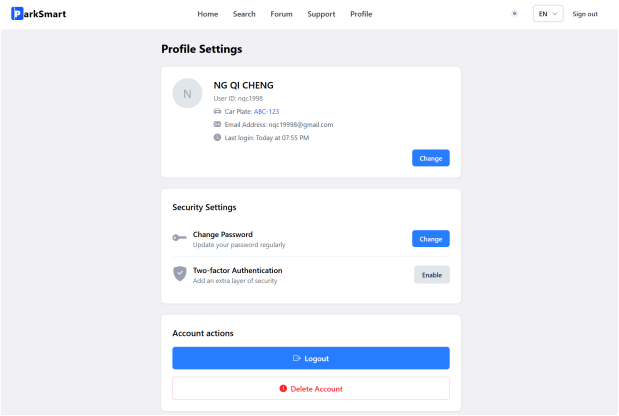
- When a user clicks email support they will be navigated to their email app to start a draft to the ParkSmart team

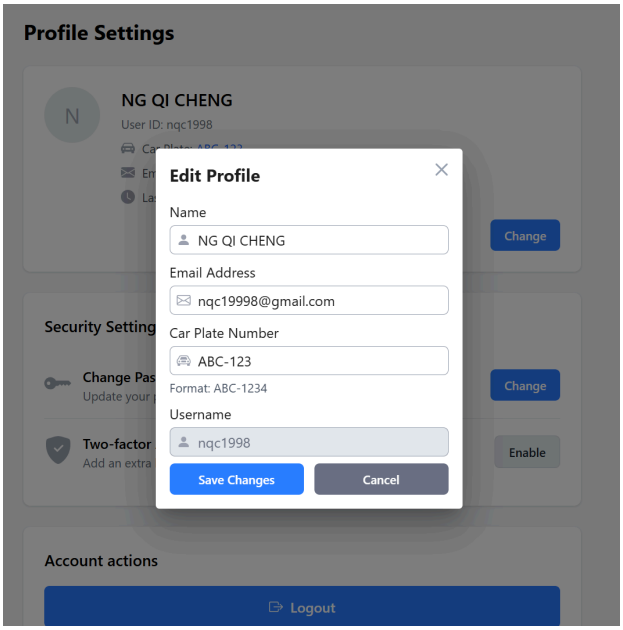
Third Image:

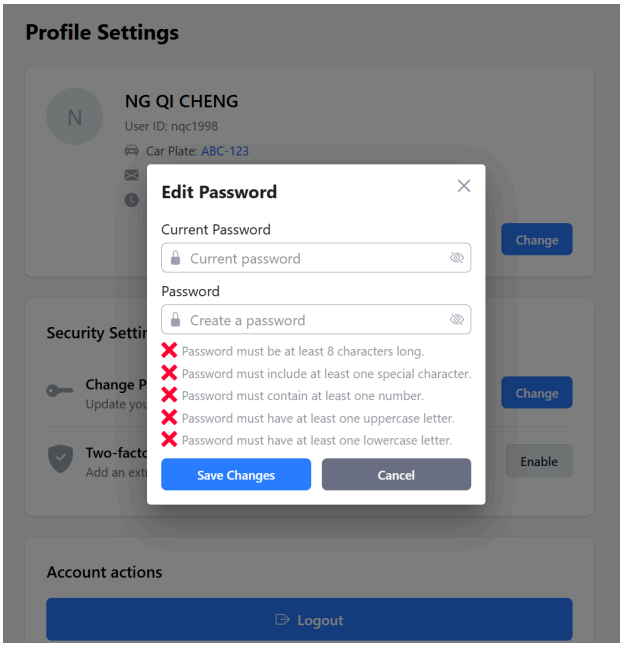
- When a user scrolls down frequently asked questions are displayed
- Users may click the '+' icon to display the answers to the FAQs

Fourth Image:

- When a user scrolls to the bottom of the page they see the feedback form field
- Users may enter their name, email, subject(dropdown menu), feedback content and service rating(click a number) then submit their feedback. The feedback will be sent to be

<table>
<tr>
<td> 
</td>
<td>reviewed by the ParkSmart admin.</td>
</tr>
<tr>
<td> 
</td>
<td>Profile Page
First Image:
• Displays the user’s personal information including name, email, username, car plate number and last logged in time.
• Users may sign out by clicking ‘Log out’
• Users may delete their account by clicking ‘Delete Account’
Second Image:
• When user clicks the first ‘Change’ button the user’s current personal details will be displayed
• Users may type and edit their current details
• Users may click ‘Save Changes’ to</td>
</tr>
</table>

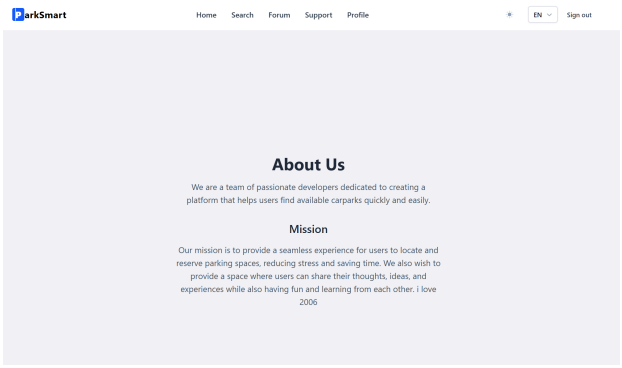
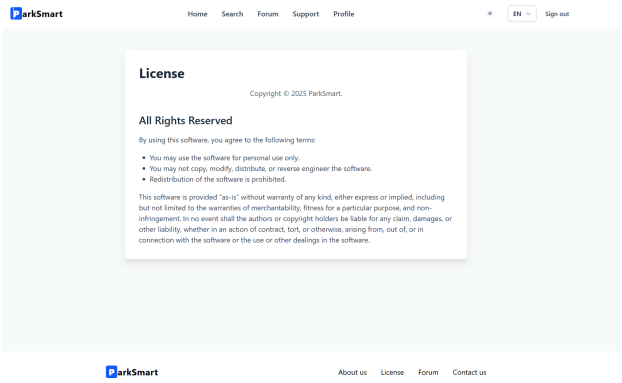




update the information. Or ‘Cancel’ to withhold the current information

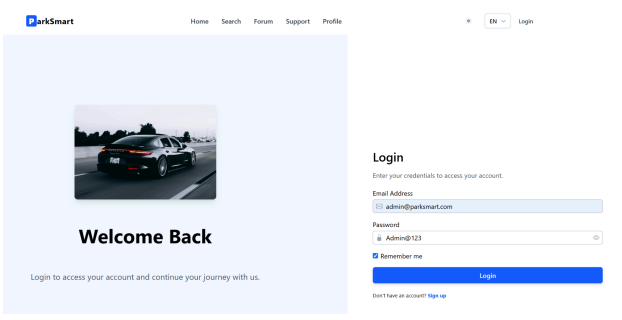
Third Image:

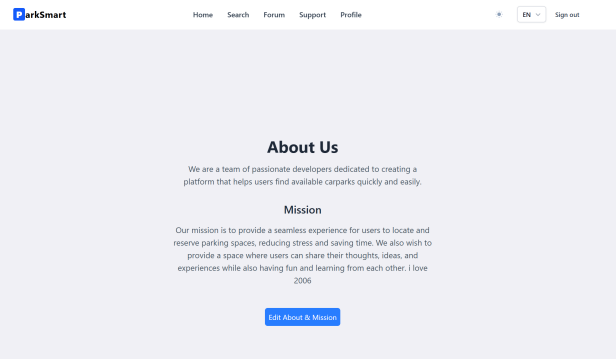
- When user clicks the second ‘Change’ button a text field for current password and new password are displayed
- Users must ensure the current password entered is accurate and the new password has:
    - 8 characters
    - 1 special character
    - 1 number
    - 1 uppercase letter
    - One lower case letter
- Users may click ‘Save Changes’ to update the information. Or ‘Cancel’ to withhold the current information

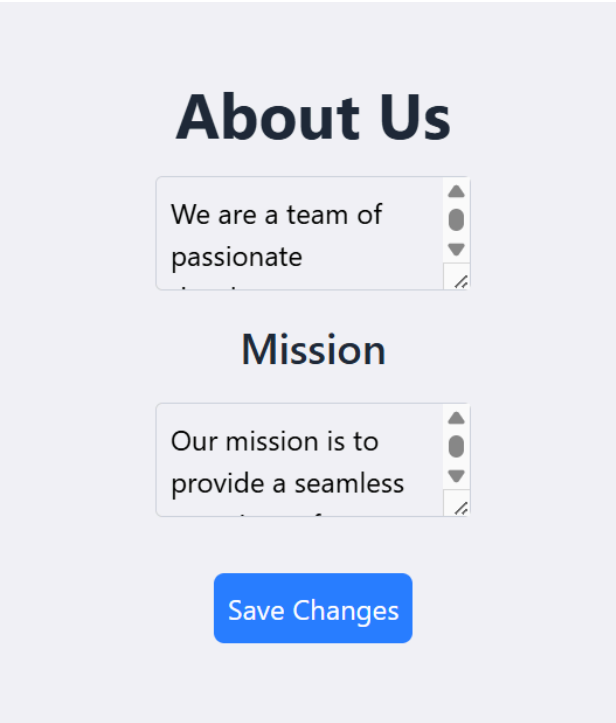
	**About Page (User POV)** This page is accessed by clicking the About page link in the Homepage. • Displays about information about ParkSmart • Displays the mission statement
	**License Page** Provides information about the terms and conditions under which the app and its content can be used.

3.1.3 Admin Interfaces

The admin account withholds all User Interfaces except for the following special access interfaces.

	**Admin Login** ParkSmart team may log in to access admin features using the following admin login.





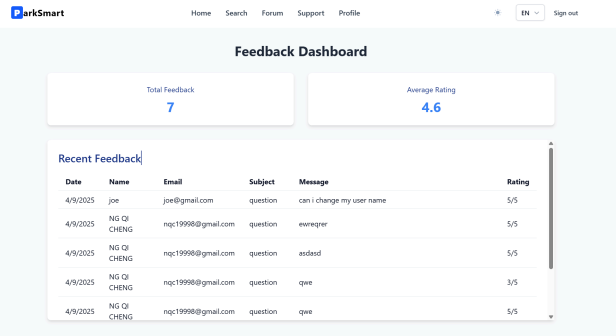
About Page (Admin POV)

Image 1:

- Displays about information about ParkSmart
- Displays the mission statement

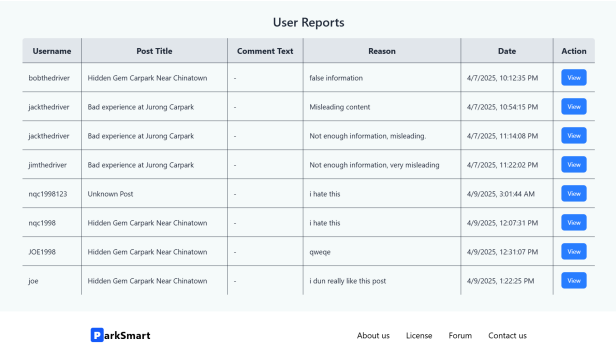
Image 2:

Admin is brought to this page when they click ‘Edit About & Mission’

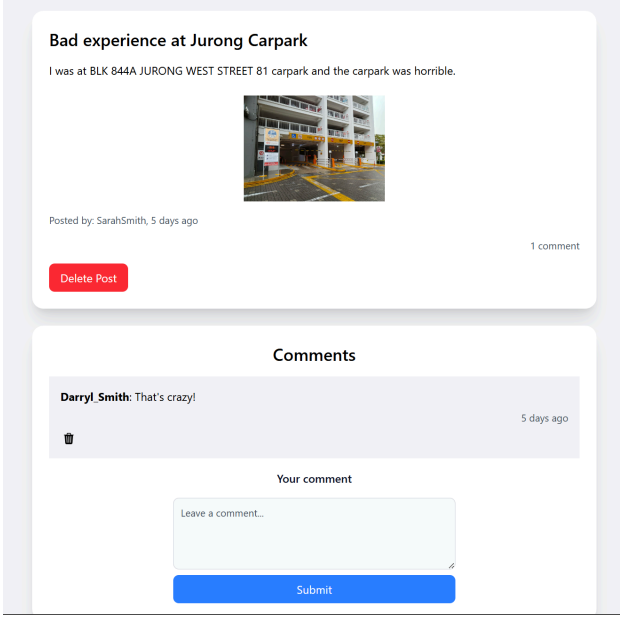


Support Page (Admin POV)

- Displays:
  - Total feedback and average rating (from user feedback form submissions)
  - Recent feedback form submissions
  - User comment/ post reports
- Admin may click ‘view post’ to



navigate to the reported post



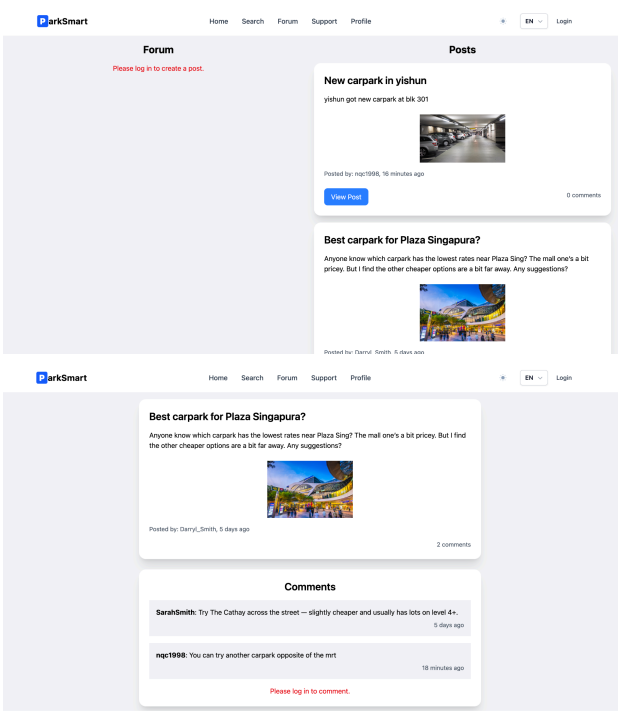
View Post Page (Admin POV)

Admin may enter this page by either clicking ‘View Post’ on the forum page or ‘View Post’ from user reports in the support page.

- Admin may click ‘Delete Post’ to remove a post from the forum
- Admin may click the bin icon beside a comment to delete it from a post
- Admin may leave a comment under a post entering their message content in the new comment field and clicking ‘Submit’

3.1.4 Guest Interfaces

If a user isn't logged in, they have limited access and can only view the Login/Sign Up Page, Search Page, Forum Page, and Support. The interfaces for the Login/Sign Up, Search, and Support pages remain identical to those available to Users. The forum page has a different interface.



Forum Page

Image 1:

- Guests may view all posts submitted on the forum page
- Guests may not create a new post

Image 2 (View Post Page: Guest POV):

- Guest may click 'View Post' to see a post's content
- Guests may view all comments under the post
- Guests may not comment under a post nor make a report

3.2 Hardware Interfaces

3.2.1 Client-Side Device Compatibility

ParkSmart is designed as a web application compatible with modern web browsers such as Chrome (version 90 and above), Firefox (version 88 and above), Safari (version 14 and above), and Edge (version 90 and above). It is fully responsive and works across various devices including laptops, desktops, tablets, and smartphones by adjusting its layout to fit different screen sizes.

An internet connection is required to use ParkSmart. Client devices should meet the following specifications:

- At least 2GB of RAM for optimal performance
- A minimum screen width of 320px (mobile) and 1024px (desktop)
- JavaScript enabled in the browser

All user interactions are handled through standard input/output devices, including display screens for output, keyboards for text input, and mouse/touchscreens for navigation and other general actions.

3.2.2 Server-Side Device Compatibility

ParkSmart backend operates on a Node.js server environment, which handles tasks like database management (CRUD operations), API requests (including OneMap integration), and other backend functionalities. The server hosting the application should meet these specifications:

- Node.js version 14.0.0 or newer
- At least 2GB of RAM (4GB recommended)
- A minimum of 2 CPU cores
- 10GB of storage for the application code, dependencies, and database (if hosted locally)
- A stable internet connection for API communication
- MongoDB version 4.4 or higher (either self-hosted or through MongoDB Atlas in the cloud)

3.3 Software Interfaces

3.3.1 APIs and Data

ParkSmart uses a few real-time APIs for operation:

Car Park Availability API
(https://data.gov.sg/datasets/d_ca933a644e55d34fe21f28b8052fac63/view)

Provides real-time car park availability data for HDB-managed car parks, including number of available lots.

HDB Car park Information API
(https://data.gov.sg/datasets/d_23f946fa557947f93a8043bbef41dd09/view)

Provides detailed information about each HDB car park, such as location, type, payment methods, gantry height, and parking rates.

OneMap Search API (https://www.onemap.gov.sg/apidocs/search)

Enables location-based searches in Singapore, allowing you to search for addresses, postal codes, and nearby places.

OneMap Advanced Minimap API
(https://www.onemap.gov.sg/apidocs/maps/#advancedMiniMap)

Allows embedding of a customizable interactive minimap, including markers, overlays, and zoom controls.

OneMap Authentication API (https://www.onemap.gov.sg/apidocs/authentication)

Used to authenticate and obtain access tokens for secure use of OneMap's secure search APIs. Expires every 3 days.

3.3.2 Database

ParkSmart employs MongoDB as its primary database for real-time CRUD operations, data persistence, and the secure storage of all data, including:

- User information (username, email, password hash, car plate numbers)
- Forum posts and associated comments
- User reports on inappropriate content
- User feedback submissions
- System content such as About and Mission statements

The application implements a connection strategy that attempts to connect to a cloud-based MongoDB instance first, with a fallback to a local MongoDB installation if the cloud connection fails.

3.3.3 Frameworks

The frameworks used in building ParkSmart are as follows:

1. Frontend:
    1. React.js (version 19.0.0)
    2. React DOM (version 19.0.0)
    3. React Router DOM (version 7.3.0)
    4. Axios (version 1.8.3)
    5. React-i18next (version 15.4.1) & i18next (version 24.2.3)
    6. React Password Checklist (version 1.8.1)
    7. Geolib (version 3.3.4)
    8. Browser Image Compression (version 2.0.2)
    9. Tailwind CSS (version 4.0.7)
    10. Material Tailwind React (version 2.1.10)
    11. Emotion (versions 11.14.0 for both @emotion/react and @emotion/styled)
    12. Material UI (version 6.4.8)
    13. Vite (version 6.2.2)
2. Backend:
    1. Node.js (version 14.0.0 or higher)
    2. Express.js (version 4.21.2)
    3. Mongoose (version 8.12.1)
    4. Bcrypt (version 5.1.1)
    5. JSON Web Token (version 9.0.2)
    6. CORS (version 2.8.5)
    7. Dotenv (version 16.4.7)
    8. Multer (version 1.4.5-lts.1)

More information about these dependencies may be found under section 2.7.

3.4 Communications Interfaces

ParkSmart employs various communication interfaces to interact with MongoDB for data storage and user authentication, as well as to ensure secure data exchange between the frontend and backend.

Protocols & Standards: The frontend communicates with the backend via API calls over HTTP/HTTPS to retrieve, update, and present data stored in MongoDB. The server serves as the core communication hub for the application, handling API requests from the frontend, processing them, and interacting with MongoDB to retrieve and modify data. Once the data is fetched from MongoDB, the server formats it in standardized JSON structures, ensuring consistency and ease of parsing by the frontend.

JWT Authentication: For secure user authentication, the system implements JSON Web Tokens (JWT). When users log in, the server validates their credentials and issues a JWT containing encoded user information. This token is then included in subsequent API requests in the Authorization header, allowing the server to authenticate and authorize user actions without requiring credentials for each request.

Mongoose ODM: To maintain consistency and structure in data exchanges with MongoDB, Mongoose Object Data Modeling (ODM) is used to organize and standardize the data models. Each model defines specific fields and validation rules, ensuring that only relevant and properly formatted information is stored in the database.

Role-Based Authorization & Access Control: The HDB Parking System enforces access control using JWT (JSON Web Token) verification middleware. Each secure API endpoint checks the JWT token to confirm that only authenticated users can access specific data and features, reducing the risk of unauthorized access.

API Key Management: To safeguard sensitive information, such as OneMap API credentials, these keys are stored in environment variables (.env files). This prevents sensitive data from being exposed in the codebase and adds an extra layer of security by managing keys separately within the

deployment environment.

Security & Encryption: User passwords are securely hashed using bcrypt before storage, ensuring that even in the event of a data breach, actual passwords remain protected. The system also implements CORS (Cross-Origin Resource Sharing) with specific origin configuration to prevent unauthorized cross-domain requests.

Error Handling & Resilience: The server employs try/catch blocks throughout the API routes to manage and log errors during communication with MongoDB. This ensures fault tolerance by catching and handling specific and general exceptions, ensuring a smooth user experience even when errors occur.

Data Transfer Optimization: The system uses Mongoose's lean() queries where appropriate to optimize data transfer rates when retrieving documents from MongoDB. Additionally, for file uploads, Multer middleware is configured to handle multipart form data with file size limits (10MB), and uploaded images are compressed (maximum 1MB) to ensure efficient and secure file transfer.

MongoDB Connection Management: The server first attempts to connect to a cloud MongoDB instance, with automatic fallback to a local instance if needed. This ensures high availability of the database connection for all communications.

4. System Features

4.1 User Features

4.1.1 Sign Up

4.1.1.1 Description and Priority

Description: A form to get the user's details and create a new account for them to view carpark information on ParkSmart.

Priority: High

4.1.1.2 Stimulus/Response Sequences

Flow of Events

1. The user enters the app and sees the landing page.
2. The user clicks the "SIGN UP" option and is directed to a sign-up page.
3. The user enters their credentials.
    a. The user provides their email.
    b. The user provides their car plate number
    c. The user provides a user ID.
    d. The user provides a password
4. The user agrees to the Terms of Service and Privacy Policy.
5. The user clicks ‘CREATE ACCOUNT’ to register their details in the system.
    a. The system validates that the email follows the correct format.
    b. The system checks if there is a duplicate email.
    c. The system checks if there is a duplicate User ID.
    d. The system validates that the provided password is secure enough.
6. The system creates a new record in the database.
7. The app prompts the user to log in.

Alternate Flows:

AF-S4: Invalid Input

AF-S4-1: Username already registered

1. App displays a cross “X” next to failed requirement
2. App remains at step S4 until requirements are met

AF-S4-2: Password strength error

1. App displays “Please match the requested format.” message
2. App remains at step S4 until requirements are met

AF-S4-3: Email format wrong

1. App displays “Please include an ‘@’ in the email address. ‘(email)’ is missing an ‘@’
2. App remains at step S4 until requirements are met

AF-S5: Email already registered

1. App displays “Email already in use.” message
2. App returns to step S2

Exceptions

EX1: Database is inaccessible

EX2: Network error

1. App displays “Error. Not connected to the internet.”

4.1.1.3 Functional Requirements

REQ-1. FR-1: User Registration Interface

REQ-1.1 The system shall provide a landing page with a visible "SIGN UP" option.

REQ-1.2 The system shall provide a dedicated registration page when the "SIGN UP" option is selected.

REQ-1.3 The registration page shall include input fields for:

- Email address
- Car plate number
- User ID

- Password

REQ-1.4 The registration page shall include checkboxes for agreeing to Terms of Service and Privacy Policy.

REQ-1.5 The registration page shall include a "CREATE ACCOUNT" button to submit registration details.

REQ-2. FR-2: Input Validation

REQ-2.1 The system shall validate that the email address follows the correct format.

REQ-2.2 The system shall validate that the email address is not already registered in the database.

REQ-2.3 The system shall validate that the User ID is not already registered in the database.

REQ-2.4 The system shall validate that the password meets the security requirements.

REQ-2.5 The system shall validate that the Terms of Service and Privacy Policy have been accepted.

REQ-3. FR-3: Error Handling

REQ-3.1 The system shall display "Username already exists" when attempting to register with an existing User ID.

REQ-3.2 The system shall display specific password strength error messages when password requirements are not met.

REQ-3.3 The system shall display "Please include an '@' in the email address. '(email)' is missing an '@'" when email format is incorrect.

REQ-3.4 The system shall display "Email already in use" when attempting to register with an existing email.

REQ-3.5 The system shall handle database accessibility errors appropriately.

REQ-4. FR-4: Account Creation

REQ-4.1 The system shall create a new user record in the database when all validation checks pass.

REQ-4.2 The system shall store the user's email, car plate number, User ID, and password (securely).

REQ-4.3 The system shall prompt the user to log in after successful account creation.

REQ-5. FR-5: System Persistence

REQ-5.1 The system shall remain on the registration page until all input requirements are met.

REQ-5.2 The system shall return to the registration page when an email is already registered.

4.1.2 Login

4.1.2.1 Description and Priority

Description: A form to ask for the user's email and password before they can access app functionalities that require user authentication.

Priority: High

4.1.1.2 Stimulus/Response Sequences

Flow of Events

1. The user enters their credentials - an email and password.
2. The system validates that the email keyed in exists in the database.
3. The system validates that the provided password matches the password tagged to the account.
4. The user is directed to the Homepage after successfully logging in.

Alternative Flow

AF-S4-1: Login password is invalid

1. App displays “Credentials are incorrect. Please try again.” message
2. Password field is emptied

AF-S4-2: Login email is invalid

1. App displays “Credentials are incorrect. Please try again.” message
2. Password field is emptied

Exceptions

EX1: Authentication System/ Database is inaccessible

EX2: Network error

1. App displays “Error. Not connected to the internet.”

4.1.2.3 Functional Requirements

REQ-1. FR-1: User Login Interface

REQ1.1 The system shall provide a login page with input fields for email and password.

REQ-1.2 The system shall provide a "LOGIN" button to submit credentials.

REQ-2. FR-2: Authentication

REQ2.1 The system shall validate that the entered email exists in the database.

REQ2.2 The system shall validate that the provided password matches the one associated with the account.

REQ2.3 The system shall direct the user to the Homepage upon successful authentication.

REQ3. FR-3: Error Handling

REQ3.1 The system shall display "Credentials are incorrect. Please try again." when an incorrect password is entered.

REQ3.2 The system shall display "Credentials are incorrect. Please try again." when an unregistered email is entered.

REQ3.3 The system shall display "Error. Not connected to the internet." when network connectivity issues occur.

REQ3.4 The system shall handle database accessibility errors appropriately.

REQ-4. FR-4: Security Management

REQ-4.1 The system shall securely compare the entered password with the stored password.

REQ-5. FR-5: System Navigation

REQ-5.1 The system shall return users to the login page after failed authentication attempts.

4.1.3 View Homepage

4.1.3.1 Description and Priority

Description: A landing page that allows authenticated users to access ParkSmart's functions

Priority: High

4.1.3.2 Stimulus/Response Sequences

Flow of Events

1. User accesses the Homepage after logging in
2. User has the option to access several features from the homepage itself:
    a. My profile
    b. Carpark Search
    c. Forun
    d. Provide feedback
    e. About
    f. License
3. On the header, users can access:
    a. Home
    b. Search
    c. Forum
    d. Support
    e. Profile

and also toggle between light and dark mode, and different languages, or signing out

4. On the footer, users can access:
    a. About Us
    b. License
    c. Forum
    d. Contact Us

Exceptions

EX1: Homepage fails to load due to system error

1. App displays “There was an error loading the page. Please try again later.”

4.1.3.3 Functional Requirements

REQ-1. FR-1: Homepage Access

REQ-1.1 The system shall display the Homepage after successful user login.

REQ-1.2 The system shall restrict Homepage access to authenticated users only.

REQ-1.3 The system shall load all Homepage elements and functionalities upon access.

REQ-2. FR-2: Main Content Navigation

REQ-2.1 The system shall provide access to "My Profile" feature from the Homepage.

REQ-2.2 The system shall provide access to "Carpark Search" feature from the Homepage.

REQ-2.3 The system shall provide access to "Forum" feature from the Homepage.

REQ-2.4 The system shall provide access to "Provide Feedback" feature from the Homepage.

REQ-2.5 The system shall provide access to "About" information from the Homepage.

REQ-2.6 The system shall provide access to "License" information from the Homepage.

REQ-3. FR-3: Header Navigation

REQ-3.1 The system shall display a header with navigation options on the Homepage.

REQ-3.2 The header shall include a "Home" navigation option.

REQ-3.3 The header shall include a "Search" navigation option.

REQ-3.4 The header shall include a "Forum" navigation option.

REQ-3.5 The header shall include a "Support" navigation option.

REQ-3.6 The header shall include a "Profile" navigation option.

REQ-3.7 The header shall include a theme toggle option for switching between light and dark modes.

REQ-3.8 The header shall include a language selection option.

REQ-3.9 The header shall include a "Sign Out" option.

REQ-4. FR-4: Footer Navigation

REQ-4.1 The system shall display a footer with additional navigation options on the Homepage.

REQ-4.2 The footer shall include an "About Us" navigation option.

REQ-4.3 The footer shall include a "License" navigation option.

REQ-4.4 The footer shall include a "Forum" navigation option.

REQ-4.5 The footer shall include a "Contact Us" navigation option.

REQ-5. FR-5: Error Handling

REQ-5.1 The system shall display "There was an error loading the page. Please try again later." when the Homepage fails to load due to system errors.

REQ-5.2 The system shall implement appropriate error logging for Homepage loading failures.

REQ-5.3 The system shall attempt to reload the Homepage when errors occur.

4.1.4 Search Car Park Location

4.1.4.1 Description and Priority

Description: A landing page with real-time updates and a map that allows users to search for the location and consolidated information of their preferred carpark in Singapore.
Priority: High

4.1.4.2 Stimulus/Response Sequences

Flow of Events

1. User enters a location or address into the search bar
2. App processes the input and retrieves matching car park locations from the Car Park Information System
3. App displays a list of car parks that match the search query, including their names and general location information
4. User can click on any car park in the list for more detailed information:
    a. Full address
    b. Spots available
    c. Short Term Parking
    d. Night parking
    e. Car Park Type
    f. Height limit
    g. Parking Type
    h. Free Parking
    i. Car Park Decks
    j. Basement

Alternate flows

AF-S2-1: No matching car parks found

1. App displays “No car parks found matching your search. Please try again with different keywords.”
2. Redirect user to enter a location or address into the search bar

Exceptions

EX1: Search query fails due to system error.

1. App displays “There was an error processing your search. Please try again later.”

EX2: API rate limit exceeded.

1. App displays “There was an overload of requests. Please try again later.”

EX3: Network error

1. App displays “Error. Not connected to the internet.”

4.1.4.3 Functional Requirements

REQ-1. FR-1: Search Interface

REQ-1.1 The system shall provide a landing page with a prominently displayed search bar.

REQ-1.2 The system shall allow users to enter location or address information into the search bar.

REQ-1.3 The system shall process search inputs to retrieve matching car park locations.

REQ-1.4 The system shall display a real-time map visualization of car park locations.

REQ-2. FR-2: Search Results Display

REQ-2.1 The system shall display a list of car parks that match the search query.

REQ-2.2 The system shall include car park names and general location information in the results list.

REQ-2.3 The system shall allow users to select individual car parks from the results list for detailed information.

REQ-2.4 The system shall update the map display to highlight relevant car parks based on search results.

REQ-3. FR-3: Car Park Information Retrieval

REQ-3.1. The system shall connect to the Car Park Information System to retrieve car park data.

REQ-3.2 The system shall process location data to identify relevant car parks.

REQ-3.3 The system shall display real-time updates of car park information where available.

REQ-4. FR-4: Error Handling

REQ-4.1 The system shall redirect users to the search input when no results are found.

REQ-5. FR-5: Performance Requirements

REQ-5.1 The system shall process search queries and display results within 2 seconds

REQ-5.2 The system shall implement appropriate throttling mechanisms to prevent API rate limit issues.

REQ-5.3 The system shall cache frequently searched locations to improve performance.

4.1.5 View Carpark Information

4.1.5.1 Description and Priority

Description: A landing page that allows users to view the complete carpark information of their preferred carpark in Singapore.
Priority: High

4.1.5.2 Stimulus/Response Sequences

Flow of Events

1. User selects a car park from the list or searches for a specific car park
2. App retrieves real-time parking availability for the selected car park

3. App displays the parking lot details including:
    a. Full address
    b. Spots available
    c. Short Term Parking
    d. Night parking
    e. Car Park Type
    f. Height limit
    g. Parking Type
    h. Free Parking
    i. Car Park Decks
    j. Basement

Alternate flows

AF-S2: No parking lots available

1. App displays “No available parking lots at the moment. Please try again later.”

Exceptions

EX1: Real-time data retrieval fails

1. App displays “Unable to retrieve parking availability. Please try again later.”

EX2: Network error

1. App displays “Error. Not connected to the internet.”

4.1.5.3 Functional Requirements

REQ-1. FR-1: Car Park Selection

REQ-1.1 The system shall allow users to select a specific car park from a displayed list.

REQ-1.2 The system shall process car park selection inputs to retrieve detailed information.

REQ-2. FR-2: Information Retrieval

REQ-2.1 The system shall retrieve real-time parking availability data for the selected car park.

REQ-2.2 The system shall access comprehensive car park details from the database.

REQ-2.3 The system shall update car park availability information at appropriate intervals to maintain data currency.

REQ-3. FR-3: Information Display

REQ-3.1 The system shall display detailed car park information including:

1. Full address
2. Number of spots available
3. Short term parking availability
4. Night parking availability
5. Car park type
6. Height limit
7. Parking type
8. Free parking periods (if applicable)
9. Number of car park decks
10. Basement parking availability

REQ-3.2 The system shall present information in a clear, organized, and user-friendly format.

REQ-4. FR-4: Availability Status

REQ-4.1 The system shall display real-time parking spot availability.

REQ-4.2 The system shall display "No available parking lots at the moment. Please try again later." when no parking spots are available.

REQ-5. FR-5: Error Handling

REQ-5.1 The system shall display "Unable to retrieve parking availability. Please try again later." when real-time data retrieval fails.

REQ-5.2 The system shall display "Error. Not connected to the internet." when network connectivity issues occur.

REQ-5.3 The system shall implement appropriate error logging for system failures.

REQ-6. FR-6: Performance Requirements

REQ-6.1 The system shall retrieve and display car park details within 2 seconds

REQ-6.2 The system shall maintain data accuracy through appropriate refresh mechanisms.

4.1.6 Access Forum

4.1.6.1 Description and Priority

Description: A landing page that allows authenticated users to view other authenticated users' comments and suggestions of the various carparks in Singapore.
Priority: High

4.1.6.2 Stimulus/Response Sequences

Flow of Events

1. User navigates to Community Page
2. App fetches forum data from the Database
3. User granted access to Forum page
4. User scrolls and views others' posts

Alternate flows

AF-S4-1: User adds comments

1. See included 'Comment' use case

AF-S4-2: User makes and shares a post

1. See included ‘Make Post’ use case

Exceptions

EX1: Database error

1. App displays “Unable to update language settings. Please try again later.”
2. App returns to the Profile Page with no language change applied.

EX2: Network error

1. App displays “Error. Not connected to the internet.”

4.1.6.3 Functional Requirements

REQ-1. FR-1: Forum Access

REQ-1.1 The system shall provide a Community Page accessible to authenticated users.

REQ-1.2 The system shall verify user authentication status before granting access to the forum content.

REQ-1.3 The system shall retrieve forum data from the Database when a user navigates to the Community Page.

REQ-1.4 The system shall display community forum content to authenticated users.

REQ-2. FR-2: Content Display

REQ-2.1 The system shall display posts and comments from other authenticated users.

REQ-2.2 The system shall organize forum content in a scrollable interface.

REQ-2.3 The system shall display car park-specific discussions appropriately categorized or tagged.

REQ-2.4 The system shall support displaying various content types (text, possibly images)

within forum posts.

REQ-3. FR-3: User Interaction

REQ-3.1 The system shall support user comment functionality as defined in the 'Comment' use case.

REQ-3.2 The system shall support user post creation as defined in the 'Make Post' use case.

REQ-3.3 The system shall differentiate between viewing, commenting, and posting functionalities.

REQ-4. FR-4: Content Retrieval

REQ-4.1 The system shall fetch forum data from the Database efficiently.

REQ-4.2 The system shall implement appropriate pagination or lazy loading for forum content.

REQ-4.3 The system shall maintain content currency through appropriate refresh mechanisms.

REQ-5. FR-5: Error Handling

REQ-5.1 The system shall display "Unable to load forum content. Please try again later." when database errors occur.

REQ-5.2 The system shall return users to an appropriate page when content cannot be loaded.

REQ-5.3 The system shall implement appropriate error logging for system failures.

REQ-6. FR-6: Performance Requirements

REQ-6.1 The system shall retrieve and display forum content within 2 seconds.

REQ-6.2 The system shall optimize content loading to minimize user wait times.

4.1.7 Make a Post (includes Comment)

4.1.7.1 Description and Priority

Description: A form to get the authenticated user's details and comments to create a post on ParkSmart's Forum page.
Priority: High

4.1.7.2 Stimulus/Response Sequences

Flow of Events

1. User has something to share
2. User types post in text box and includes image attachments if they want to
3. If there are no problems, User makes a post by clicking 'Post'
4. App creates a new document for his post and stores it in the Database
    a. User sees a post and has something to comment
    b. User types comment in provided text field and adds image attachment if he wants to
    c. If there are no problems user posts his comment by clicking 'Comment'
    d. App stores his comment in the Database by updating the document containing the post being commented on

Alternate flows

AF-S3-1: Attachment has invalid file type

1. App displays "Invalid attachment type. Please try again".
2. Attachment is not uploaded

AF-S3-2: Attachment file size is too big

1. App displays "Please do not exceed 10mb."
2. Attachment is not uploaded

AF-S3-3: User cancels post submission

1. App returns to the previous screen without submitting any post

AF-S5: User deletes post after posting

1. App sets the status of post in Database to ‘Deleted’
2. The post is no longer displayed

AF-S3-3: User cancels comment submission

1. App returns to the previous screen without submitting any comment

AF-S5: User deletes comment after posting

1. App removes comment from post document in Database and moves it to ‘Deleted Comments’
2. Comment is no longer displayed

Exceptions

EX1: Database error

1. App displays “Unable to update language settings. Please try again later.”
2. App returns to the Profile Page with no language change applied.

EX2: Network error

1. App displays “Error. Not connected to the internet.”

EX3: Post text exceeds character limit

1. App displays "Your post is too long. Please shorten your text."

EX4: Post submission fails due to system error

1. App displays "There was an error submitting your comment. Please try again later."

EX5: Comment text exceeds character limit

1. App displays "Your comment is too long. Please shorten your message."

EX6: Comment submission fails due to system error

1. App displays "There was an error submitting your comment. Please try again later."

4.1.7.3 Functional Requirements

REQ-1. FR-1: Post Creation Interface

REQ-1.1 The system shall provide a text box for users to enter post content.

REQ-1.2 The system shall allow users to attach images to their posts.

REQ-1.3 The system shall provide a "Create Post" button to submit the content.

REQ-1.4 The system shall display appropriate interface elements for post creation and submission.

REQ-2. FR-2: Post Submission

REQ-2.1 The system shall validate post content before submission.

REQ-2.2 The system shall create a new document in the Database for each submitted post.

REQ-2.3 The system shall associate posts with the authenticated user's account.

REQ-2.4 The system shall handle text content and image attachments appropriately.

REQ-3. FR-3: Comment Creation Interface

REQ-3.1 The system shall provide a text field for users to enter comments on existing posts.

REQ-3.2 The system shall allow users to attach images to their comments.

REQ-3.3 The system shall provide a "Submit" button to submit the comment.

REQ-3.4 The system shall display appropriate interface elements for comment creation and submission.

REQ-4. FR-4: Comment Submission

REQ-4.1 The system shall validate comment content before submission.

REQ-4.2 The system shall store comments in the Database by updating the document containing the post being commented on.

REQ-4.3 The system shall associate comments with the authenticated user's account.

REQ-4.4 The system shall handle text content and image attachments appropriately for comments.

REQ-5. FR-5: Attachment Handling

REQ-5.1 The system shall validate file types for image attachments in both posts and comments.

REQ-5.2 The system shall display "Invalid attachment type. Please try again" when unsupported file types are selected.

REQ-5.3 The system shall validate file sizes for attachments in both posts and comments.

REQ-5.4 The system shall display "Please do not exceed 10mb" when attachment size limits are exceeded.

REQ-5.5 The system shall prevent uploading of attachments that fail validation.

REQ-6. FR-6: Content Management

REQ-6.1 The system shall allow users to cancel post submission and return to the previous screen.

REQ-6.2 The system shall allow users to cancel comment submission and return to the previous screen.

REQ-6.3 The system shall allow users to delete their own posts after posting.

REQ-6.4 The system shall set the status of deleted posts in the Database to 'Deleted'.

REQ-6.5 The system shall no longer display deleted posts in the forum.

REQ-6.6 The system shall allow users to delete their own comments after posting.

REQ-6.7 The system shall remove deleted comments from post documents in the Database and move them to 'Deleted Comments'.

REQ-6.8 The system shall no longer display deleted comments in the forum.

REQ-7. FR-7: Content Validation

REQ-7.1 The system shall enforce character limits for post text.

REQ-7.2 The system shall display "Your post is too long. Please shorten your text" when post character limits are exceeded.

REQ-7.3 The system shall prevent submission of posts that exceed character limits.

REQ-7.4 The system shall enforce character limits for comment text.

REQ-7.5 The system shall display "Your comment is too long. Please shorten your message" when comment character limits are exceeded.

REQ-7.6 The system shall prevent submission of comments that exceed character limits.

REQ-8. FR-8: Error Handling

REQ-8.1 The system shall return users to the Profile Page when database errors prevent post or comment creation.

REQ-8.2 The system shall implement appropriate error logging for submission failures.

4.1.8 Feedback

4.1.8.1 Description and Priority

Description: A form to get the authenticated user's details and feedback regarding ParkSmart.
Priority: High

4.1.8.2 Stimulus/Response Sequences

Flow of Events

1. User selects the "Feedback" option from the navigation menu
2. App prompts the user to enter their feedback
3. User enters feedback in the provided text field
4. User submits the feedback
5. App confirms submission with a message "Thank you for your feedback"
6. Feedback is stored in the system for review

Alternate flows

AF-S2-1: User cancels feedback submission

1. App returns to the previous screen without submitting any feedback

Exceptions

EX1: Feedback submission fails due to system error

1. App displays "There was an error submitting your feedback. Please try again later."

EX2: Feedback text exceeds character limit

1. App displays "Your feedback is too long. Please shorten your message."

EX3: Network error

1. App displays “Error. Not connected to the internet.”

4.1.8.3 Functional Requirements

REQ-1. FR-1: Feedback Access

REQ-1.1 The system shall provide a "Forum" option in the header.

REQ-1.2 The system shall provide a landing page “Send us your feedback!” within the “Forum” page

REQ-1.2 The system shall display a feedback submission interface when the Feedback option is selected.

REQ-2. FR-2: Feedback Page

REQ-2.1 The system shall prompt users to enter their feedback.

REQ-2.2 The system shall provide a text field for users to enter feedback content.

REQ-2.3 The system shall provide a submission button to submit feedback.

REQ-2.4 The system shall allow users to cancel feedback submission.

REQ-3. FR-3: Feedback Submission

REQ-3.1 The system shall validate feedback content before submission.

REQ-3.2 The system shall store submitted feedback in the system for review.

REQ-3.3 The system shall display a confirmation message "Thank you for your feedback" after successful submission.

REQ-3.4 The system shall return users to the previous screen when feedback submission is canceled.

REQ-4. FR-4: Content Validation

REQ-4.1 The system shall enforce character limits for feedback text.

REQ-4.2 The system shall display "Your feedback is too long. Please shorten your message" when character limits are exceeded.

REQ-4.3 The system shall prevent submission of feedback that exceeds character limits.

REQ-5. FR-5: Error Handling

REQ-5.1 The system shall display "There was an error submitting your feedback. Please try again later" when feedback submission fails due to system errors.

REQ-5.2 The system shall implement appropriate error logging for submission failures.

4.1.9 View Profile page

4.1.9.1 Description and Priority

Description: A page that authenticated users can access to access different functions of ParkSmart.
Priority: High

4.1.9.2 Stimulus/Response Sequences

Flow of Events

1. User selects the “Profile” option from the navigation bar
2. App retrieves the user’s profile data from the User Authentication System
3. App displays the user’s profile information including email, username, vehicle number plate, and other relevant details
4. User can edit their details such as email, password, or vehicle number plate
5. User confirms changes and updates are saved to the User Authentication System
6. App confirms changes have been successfully updated

Alternate flows

AF-S4: User cancels profile updates

1. App returns to the homepage without saving any changes

Exceptions

EX1: Profile update fails due to system error

2. App displays “There was an error updating your profile. Please try again later.”

4.1.9.3 Functional Requirements

REQ-1. FR-1: Profile Access

REQ-1.1 The system shall provide a "Profile" option in the navigation bar.

REQ-1.2 The system shall retrieve the user's profile data from the User Authentication System when the Profile option is selected.

REQ-1.3 The system shall display the user's profile information, including email, username, vehicle number plate, and other relevant details.

REQ-2. FR-2: Profile Editing

REQ-2.1 The system shall allow users to edit their profile details including:

1. Email address
2. Password
3. Vehicle number plate
4. Any other relevant profile information

REQ-2.2 The system shall provide appropriate input fields for each editable profile detail.

REQ2.3 The system shall provide buttons or controls to confirm or cancel profile updates.

REQ-3. FR-3: Profile Update

REQ-3.1 The system shall save confirmed profile changes to the User Authentication System.

REQ-3.2 The system shall display a confirmation message when profile updates are successfully saved.

REQ-3.3 The system shall not save any changes when a user cancels the update process.

REQ-3.4 The system shall return users to the homepage when profile updates are canceled.

REQ-4. FR-4: Error Handling

REQ-4.1 The system shall display "There was an error updating your profile. Please try again later." when profile updates fail due to system errors.

REQ-4.2 The system shall appropriately handle and log errors related to profile data retrieval or updates.

REQ-4.3 The system shall maintain the original profile data in case of update failures.

REQ5. FR-5: Data Validation

REQ-5.1 The system shall validate any edited profile information before saving changes.

REQ5.2 The system shall notify users of any validation errors in their profile edits.

4.1.10 Change Language

4.1.11.1 Description and Priority

Description: A function that allows Users to toggle the language of their website interface between English, Chinese, Malay and Hindi on ParkSmart.
Priority: Medium

4.1.10.2 Stimulus/Response Sequences

Flow of Events

1. User enters Profile Page
2. User navigates to the settings section of the Profile Page
3. User selects the "Language" option from the settings menu
4. App displays a list of available languages
5. User selects their preferred language from the list
6. App updates the interface language to the selected language by fetching the translations from the Database
7. App confirms the language change and displays a message "Language updated

successfully."

Alternate flows

Exceptions

EX1: Language change fails due to system error

1. App displays “Unable to update language settings. Please try again later.”
2. App returns to the Profile Page with no language change applied.

EX2: Language preference not saved after session restart

1. App displays “Your language preference could not be saved. Please try again.”
2. User is prompted to reselect their language or return to default settings.

EX3: Database error

1. App displays “Unable to update language settings. Please try again later.
2. App returns to the Profile Page with no language change applied.

EX4: Network error

1. App displays “Error. Not connected to the internet.”

4.1.10.3 Functional Requirements

REQ-1. FR-1: Language Settings Access

REQ-1.1 The system shall provide access to language settings from the header.

REQ-2. FR-2: Language Selection

REQ-2.1 The system shall display a list of available languages (English, Chinese, Malay, and Hindi) when the Language option is selected.

REQ-2.2 The system shall allow users to select their preferred language from the available options.

REQ-2.3 The system shall process language selection inputs to update the interface's language.

REQ-3. FR-3: Language Implementation

REQ-3.1 The system shall retrieve appropriate translations from the Database for the selected language.

REQ-3.2 The system shall update all interface text elements to reflect the selected language.

REQ-3.3 The system shall persist language preference across user sessions.

REQ-4. FR-4: Error Handling

REQ-4.1 The system shall display "Unable to update language settings. Please try again later." when language changes fail due to system errors.

REQ-4.2 The system shall display "Your language preference could not be saved. Please try again." when language preferences cannot be persisted.

REQ-4.3 The system shall display "Error. Not connected to the internet." when network connectivity issues occur.

REQ-5. FR-5: Language Data Management

REQ-5.1 The system shall maintain a database of translations for all supported languages (English, Chinese, Malay, and Hindi).

REQ-5.2 The system shall implement appropriate caching mechanisms for language settings to improve performance.

REQ-5.3 The system shall maintain default language settings (English) when errors occur.

4.1.11 Logout

4.1.12.1 Description and Priority

Description: A function that allows users to log out of their ParkSmart account.

Priority: High

4.1.12.2 Stimulus/Response Sequences

Flow of Events

1. User taps the “Sign Out” button
2. App clears user session data
3. User is signed out and redirected to the login page

Exceptions

EX1: Sign-out fails due to a system error

1. App displays “There was an error signing you out. Please try again later.” message

4.1.12.3 Functional Requirements

REQ-1. FR-1: Sign Out Initiation

REQ-1.1 The system shall provide a "Sign Out" button in the user interface.

REQ-1.2 The system shall respond to user taps on the "Sign Out" button.

REQ-1.3 The system shall process the user's confirmation choice appropriately.

REQ-2. FR-2: Session Management

REQ-2.1 The system shall clear all user session data upon confirmation of sign-out.

REQ-2.2 The system shall terminate the active authenticated session.

REQ-2.3 The system shall redirect the user to the login page after successful sign-out.

REQ-2.4 The system shall return the user to the main page if sign-out is canceled.

REQ-3. FR-3: Error Handling

REQ-3.1 The system shall display "There was an error signing you out. Please try again later." when sign-out fails due to system errors.

REQ-3.2 The system shall implement appropriate error logging for sign-out failures.

REQ-3.3 The system shall maintain the user's session if sign-out fails.

4.2 Admin Features

4.2.1 Login

4.2.1.1 Description and Priority

Description: The registration feature allows admin to log in on ParkSmart. An account 'Admin' has already been created for the Admin of the account.

Priority: High

4.2.1.2 Stimulus/Response Sequences

Flow of Events:

1. The user enters the website and sees the login page.
2. The user enters their credentials.
    a. The user provides the Admin email
    b. The user provides the Admin password.
    c. The system validates if the email and password entered is validated.

Alternate Flows:

AF-S4-1: Login password is invalid

3. App displays "Credentials are incorrect. Please try again." message
4. Password field is emptied

AF-S4-2: Login email is invalid

3. App displays “Credentials are incorrect. Please try again.” message
4. Password field is emptied

Exceptions

EX1: Authentication System/ Database is inaccessible

EX2: Network error

2. App displays “Error. Not connected to the internet.”

4.2.1.3 Functional Requirements

REQ-1. FR-1: Admin Login Interface

REQ1.1 The system shall provide a login page with input fields for email and password.

REQ-1.2 The system shall provide a "LOGIN" button to submit credentials.

REQ-2. FR-2: Authentication

REQ2.1 The system shall validate that the entered email exists in the database.

REQ2.2 The system shall validate that the provided password matches the one associated with the account.

REQ2.3 The system shall direct the user to the Homepage upon successful authentication.

REQ3. FR-3: Error Handling

REQ3.1 The system shall display "Credentials are incorrect. Please try again." when an incorrect password is entered.

REQ3.2 The system shall display "Credentials are incorrect. Please try again." when an unregistered email is entered.

REQ3.3 The system shall display "Error. Not connected to the internet." when network connectivity issues occur.

REQ3.4 The system shall handle database accessibility errors appropriately.

REQ-4. FR-4: Security Management

REQ-4.1 The system shall securely compare the entered password with the stored password.

REQ-5. FR-5: System Navigation

REQ-5.1 The system shall return users to the login page after failed authentication attempts.

4.2.2 Profile Management

4.2.5.1 Description and Priority

Description: Admin will be able to view their Profile details, including account details and forum posts/ comments. The admin may edit their account details, edit and delete posts and comments.

Priority: High

4.2.6.2 Stimulus/Response Sequences

Flow of Events:

1. The Admin accesses the Profile page.
    a. The admin clicks on the Profile button in the header
    b. The System displays the following information:
        i. Admin Name
        ii. Admin User ID
        iii. Car Plate
        iv. Email
        v. Last Login

c. From the Profile page, the admin may
    i. Change password
    ii. Enable 2-factor authentication
    iii. Logout
    iv. Delete Account

2. The admin edits their account details
    a. The admin navigates to the “Profile” page and clicks on the “Change” button.
    b. The admin can then amend the following items. The form field will display the original values.
        i. Admin Name
        ii. Admin User ID
        iii. Car Plate
        iv. Email
    c. The admin can press the “Save Changes” button to save their changes.
    d. The system will validate the input values and prompt the user if there are invalid input values. The system will not allow the user to save their changes until the input has been corrected.
    e. The admin can press the “Cancel” button to cancel their changes.

3. The admin accesses the Forum page
    a. The admin navigates to the “Forum” page and clicks on the different posts.
    b. The System will display the different posts and allow the Admin to
        i. Delete the post.
            1. Admin clicks on “View Post”
            2. Admin clicks on “Delete post”
            3. “Are you sure you want to delete this post?” will appear, prompting Admin.
            4. Admin clicks on “Yes”
            5. Post gets deleted

ii. Delete the comment
    1. Admin clicks on the dustbin icon on the comment
    2. "Are you sure you want to delete this comment?" will appear, prompting Admin.
    3. Admin clicks on "Yes"
    4. Comment gets deleted

iii. Add a post
    1. Admin has something to share
    2. Admin types post in text box and includes image attachments if they want to
    3. If there are no problems, Admin makes a post by clicking 'Post'
    4. App creates a new document for his post and stores it in the Database

iv. Add a comment
    1. Admin sees a post and has something to comment
    2. Admin types comment in provided text field and adds image attachment if he wants to
    3. If there are no problems Admin posts his comment by clicking 'Comment'
    4. App stores his comment in the Database by updating the document containing the post being commented on

Alternative Flows

AF-S4: Admin cancels profile updates

2. App returns to the homepage without saving any changes

AF-S3-3: Admin cancels post deletion

2. App returns to the previous screen without deleting any post

AF-S3-3: Admin cancels comment deletion

3. App returns to the previous screen without deleting any comment.

Exceptions

EX1: Database error

3. App displays “Unable to update language settings. Please try again later.”
4. App returns to the Profile Page with no language change applied.

EX2: Network error

2. App displays “Error. Not connected to the internet.”

EX3: Post text exceeds character limit

2. App displays "Your post is too long. Please shorten your text."

EX4: Post submission fails due to system error

2. App displays "There was an error submitting your comment. Please try again later."

EX5: Comment text exceeds character limit

2. App displays "Your comment is too long. Please shorten your message."

EX6: Comment submission fails due to system error

2. App displays "There was an error submitting your comment. Please try again later."

EX6: Profile update fails due to system error

3. App displays “There was an error updating your profile. Please try again later.”

4.2.5.3 Functional Requirements

REQ-1. FR-1: Admin Profile Access

REQ-1.1 The system shall provide a Profile button in the header for Admin access.

REQ-1.2 The system shall display the Admin's profile information when the Profile page is accessed.

REQ-1.3 The system shall display the Admin's name, User ID, Car Plate, Email, and Last Login information.

REQ-1.4 The system shall provide options for changing password, enabling 2-factor authentication, logout, and account deletion.

REQ-2. FR-2: Admin Profile Editing

REQ-2.1 The system shall provide a "Change" button to initiate editing of account details.

REQ-2.2 The system shall allow editing of Admin Name, Admin User ID, Car Plate, and Email.

REQ-2.3 The system shall display form fields pre-populated with current values.

REQ-2.4 The system shall provide "Save Changes" and "Cancel" buttons for profile edits.

REQ-2.5 The system shall validate input values before saving changes.

REQ-2.6 The system shall prompt users when invalid input values are detected.

REQ-2.7 The system shall prevent saving changes until input has been corrected.

REQ-2.8 The system shall return to the homepage without saving changes when edits are canceled.

REQ-3. FR-3: Admin Forum Access

REQ-3.1 The system shall provide access to the Forum page for Admins.

REQ-3.2 The system shall display posts and allow Admins to interact with them.

REQ-3.3 The system shall provide a "View Post" option for each post.

REQ-4. FR-4: Admin Post Management

REQ-4.1 The system shall allow Admins to delete posts.

REQ-4.2 The system shall display a confirmation message "Are you sure you want to delete this post?" before deletion.

REQ-4.3 The system shall process post deletion when confirmed by the Admin.

REQ-4.4 The system shall cancel post deletion and return to the previous screen when deletion is canceled.

REQ-4.5 The system shall allow Admins to create new posts.

REQ-4.6 The system shall provide a text box for Admins to enter post content.

REQ-4.7 The system shall allow Admins to include image attachments in posts.

REQ-4.8 The system shall create and store post documents in the Database upon submission.

REQ-5. FR-5: Admin Comment Management

REQ-5.1 The system shall allow Admins to delete comments.

REQ-5.2 The system shall provide a dustbin icon for each comment for deletion.

REQ-5.3 The system shall display a confirmation message "Are you sure you want to delete this comment?" before deletion.

REQ-5.4 The system shall process comment deletion when confirmed by the Admin.

REQ-5.5 The system shall cancel comment deletion and return to the previous screen when deletion is canceled.

REQ-5.6 The system shall allow Admins to add comments to posts.

REQ-5.7 The system shall provide a text field for comment creation.

REQ-5.8 The system shall allow image attachments in comments.

REQ-5.9 The system shall store comments in the Database by updating the associated post document.

REQ-6. FR-6: Content Validation

REQ-6.1 The system shall enforce character limits for post text.

REQ-6.2 The system shall display "Your post is too long. Please shorten your text." when post character limits are exceeded.

REQ-6.3 The system shall enforce character limits for comment text.

REQ-6.4 The system shall display "Your comment is too long. Please shorten your message." when comment character limits are exceeded.

REQ-7. FR-7: Error Handling

REQ-7.1 The system shall display "There was an error submitting your comment. Please try again later." when post or comment submission fails.

REQ-7.2 The system shall display "There was an error updating your profile. Please try again later." when profile updates fail.

REQ-7.3 The system shall display "Unable to update language settings. Please try again later." when database errors occur.

REQ-7.4 The system shall implement appropriate error logging for system failures.

5. Nonfunctional Requirements

For a car park information application with multiple user groups, we'll need to consider several non-functional requirements. These requirements ensure the app's quality, performance, and usability. Here are some key non-functional requirements to consider:

5.1 Performance Requirements

REQ-PERF-1. The system must be able to handle up to 1,000 requests per hour.

REQ-PERF-2. Information on car park availability must be retrieved and displayed within two seconds of user request.

REQ-PERF-3. At least 80% of first-time users should be able to find car park information within one minute of their query.

5.2 Safety Requirements

REQ-SAFE-1. The system should provide accurate information about gantry height and other safety features of each car park.

REQ-SAFE-2. The system should include reporting mechanisms for users to flag safety concerns at forum.

5.3 Security Requirements

REQ-SEC-1. User data must be securely stored, including authentication details and personal information, in encrypted formats within secure databases.

REQ-SEC-2. The system must protect against SQL injection, XSS, CSRF, and brute-force attacks with regular security protocol updates and vulnerability scans.

REQ-SEC-3. All data exchanges, particularly those involving sensitive or personal information, must be encrypted using HTTPS.

REQ-SEC-4. The system must strictly enforce access controls and permissions to ensure users can only access information pertinent to their account and privilege level.

5.4 Software Quality Attributes

1. Scalability
    a. The system shall be able to scale horizontally to accommodate an increasing number of users and car parks without degradation in performance.
    b. The application must maintain performance standards even during peak usage periods, such as major events or holiday seasons.
    c. The system should be designed to easily incorporate additional car parks and geographical areas without requiring significant restructuring.

2. Reliability
    a. The system shall ensure data integrity, with real-time data (including parking availability and user accounts) being consistently accurate and promptly updated.
    b. Robust error handling and recovery mechanisms shall be in place to handle unexpected failures, ensuring minimal data loss and quick recovery.

3. Usability
    a. The interface must be easy to navigate, featuring clear instructions, well-labeled elements, and visible error messages.
    b. The account creation and login processes must not exceed three steps each.
    c. Each form field must have clear labels and placeholder text where applicable.
    d. Error messages must be specific and appear immediately upon invalid input.
    e. The navigation bar should be visible across all pages, granting easy access to core functionalities like HomePage, Parking, Forum, Support and Profile.

4. Compatibility
    a. The application shall support major mobile operating systems, ensuring broad accessibility for users.
    b. The user experience shall be consistent across all devices and screen sizes, ensuring responsiveness, accessibility, and ease of use on different mobile device sizes.

c. Compatibility shall extend to all modern web browsers, providing a seamless experience across different platforms.

5. Maintainability
    a. The system must employ a modular architecture, such as MVC, to ensure components like login, profile management, and navigation can be independently updated without system-wide disruptions.
    b. The application must develop reusable components for frequently used functionalities and UI elements to simplify future updates and ensure consistency.

6. Error handling
    a. The application should provide actionable and specific error messages to guide users effectively when errors occur.
    b. If an error happens during login, such as entering a wrong password, the system should display, "Incorrect password. Please try again or reset your password."

5.5 Business Rules

REQ-BUS-1. Unauthenticated users cannot access the homepage and profile page, with limited access to forum, and will be prompted to log in if they try to. They can access other pages like login, register, support and basic car park information pages.

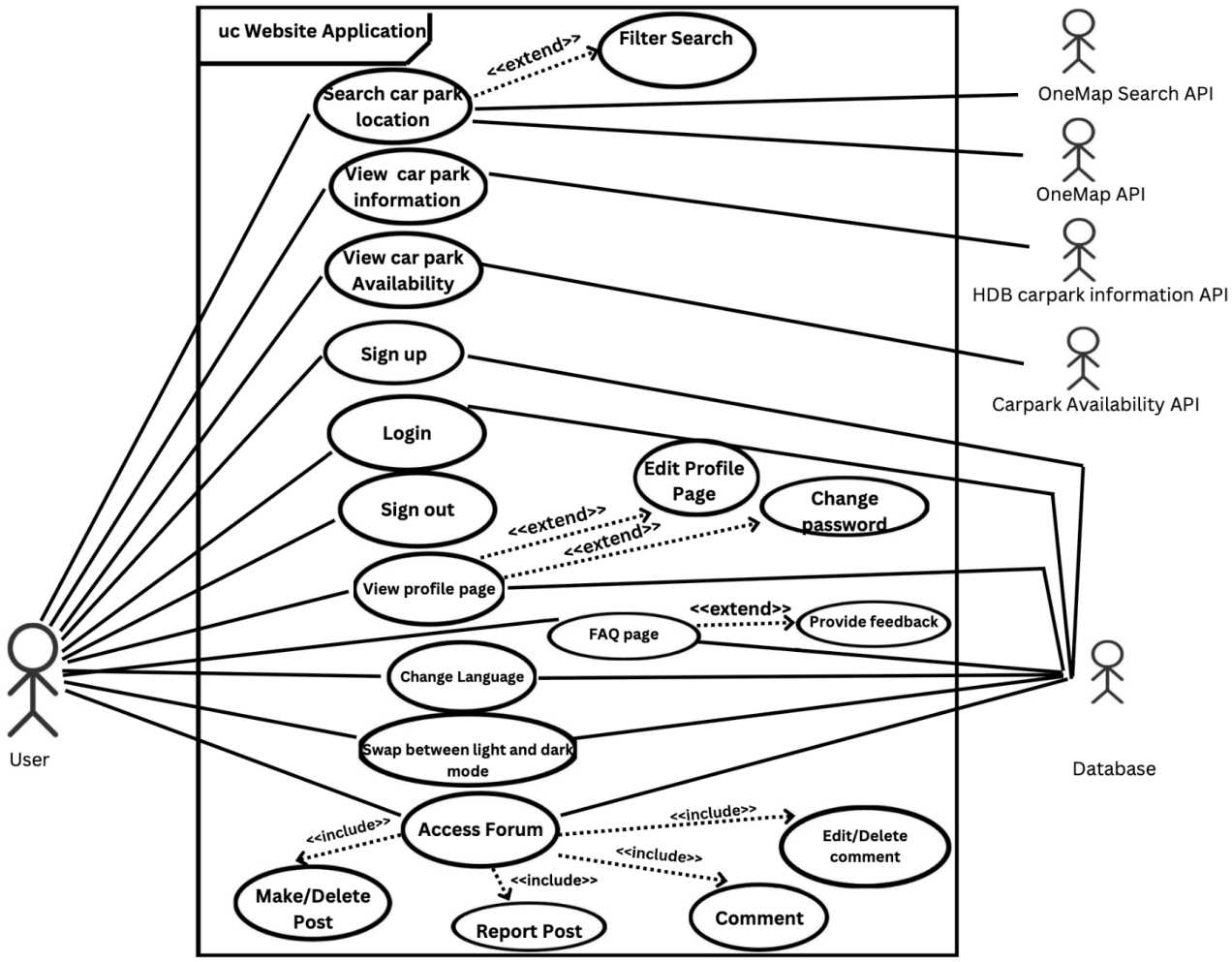
REQ-BUS-2. Authenticated users can access all pages related to car park reservations, forum discussions, and their user profile.

Appendix A: Data Dictionary

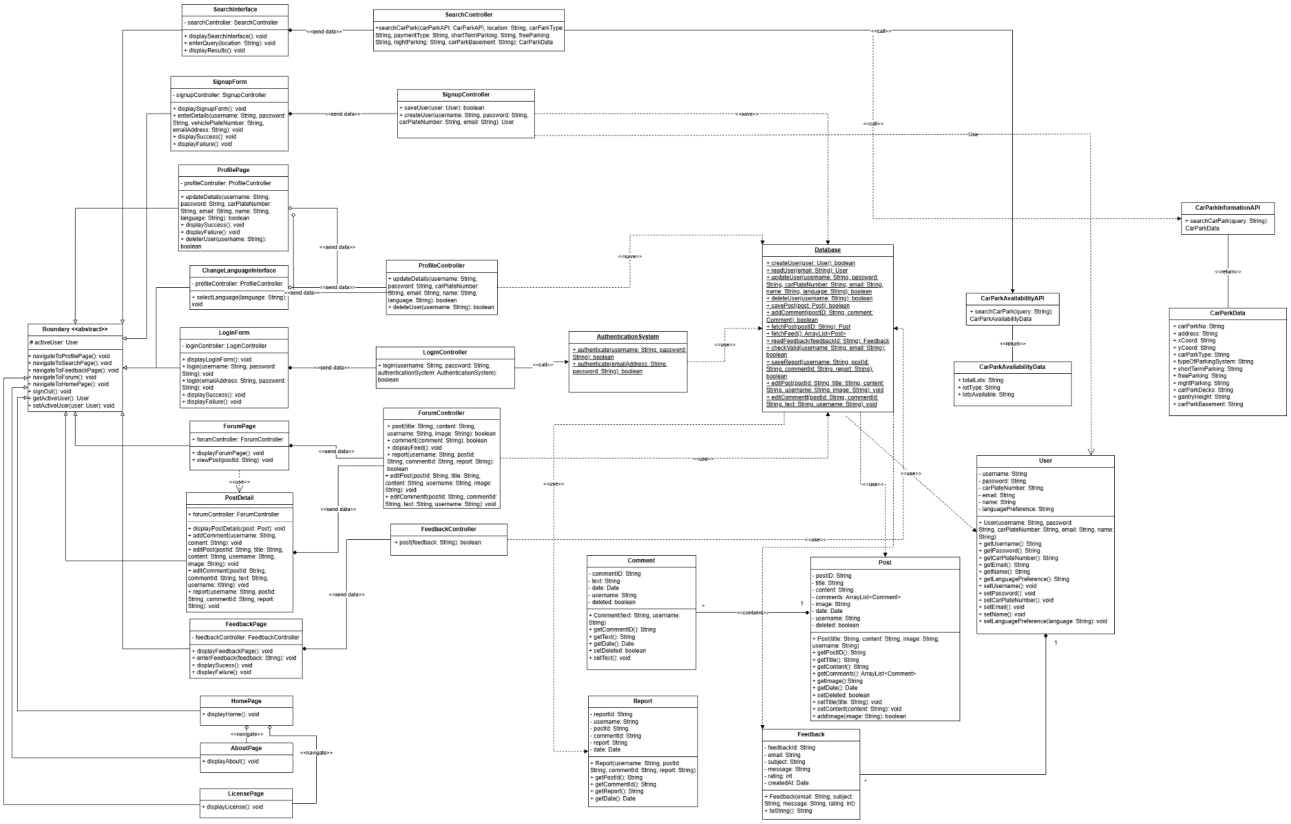
Term	Definition
User	A person using the application to check availability or reserve parking
Car Park	An area where vehicles may be parked.
Lot	A single parking spot.
Query	A user-entered search term used to locate HDB carparks based on name, location, or other attributes.
Rate	The cost of parking in an HDB carpark, which varies based on time of day and duration.
Car Park Type	The structural type of the HDB carpark.
Payment Type	The payment system used for the car park.
Gantry Height	Maximum vehicle height allowed at the carpark entrance.
Availability	The number of unoccupied parking lots in a car park at a given time.

Appendix B: Analysis Models

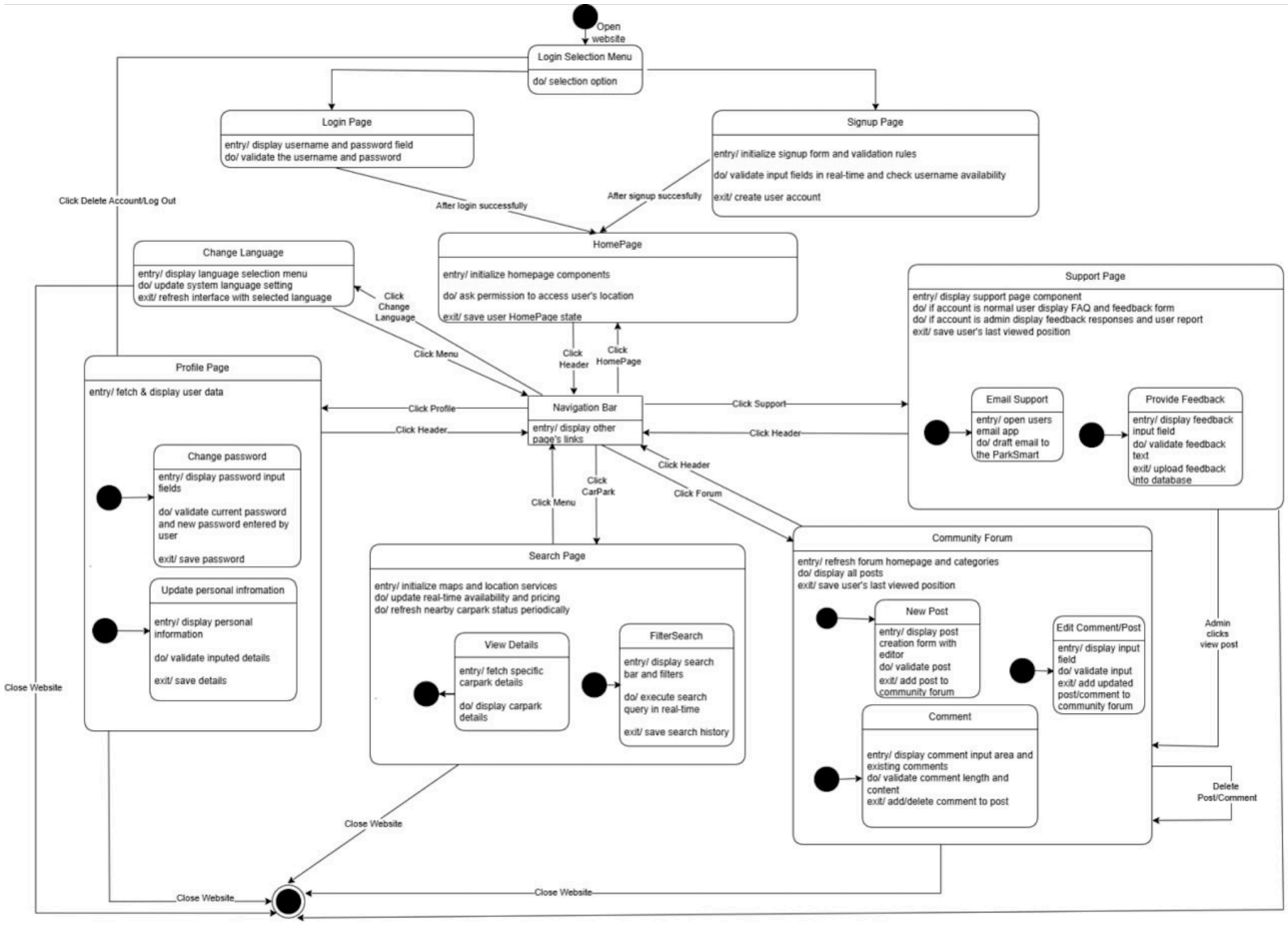
Please refer to our GitHub repository for a clearer view of the diagrams.



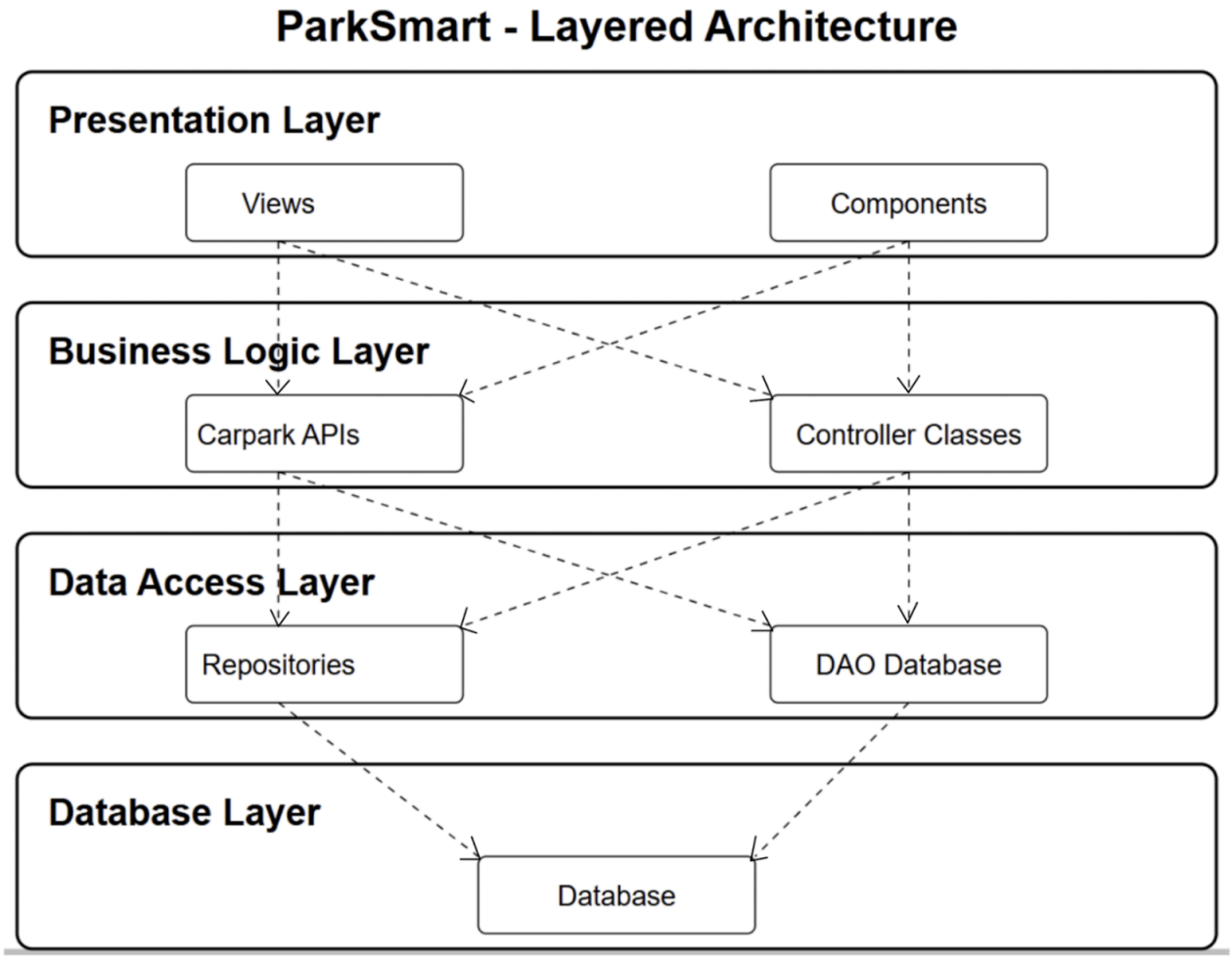
Use Case Diagram



Class Diagram


Open website
Login Selection Menu
do/ selection option
Login Page
entry/ display username and password field
do/ validate the username and password
Signup Page
entry/ initialize signup form and validation rules
do/ validate input fields in real-time and check username availability
exit/ create user account
Click Delete Account/Log Out
After login successfully
After signup succesfully
Change Language
entry/ display language selection menu
do/ update system language setting
exit/ refresh interface with selected language
HomePage
entry/ initialize homepage components
do/ ask permission to access user's location
exit/ save user HomePage state
Click Change Language
Click Menu
Click Header
Click HomePage
Support Page
entry/ display support page component
do/ if account is normal user display FAQ and feedback form
do/ if account is admin display feedback responses and user report
exit/ save user's last viewed position
Profile Page
entry/ fetch & display user data
Change password
entry/ display password input fields
do/ validate current password and new password entered by user
exit/ save password
Update personal infromation
entry/ display personal information
do/ validate inputed details
exit/ save details
Click Profile
Click Header
Navigation Bar
entry/ display other page's links
Click Support
Click Header
Email Support
entry/ open users email app
do/ draft email to the ParkSmart
Provide Feedback
entry/ display feedback input field
do/ validate feedback text
exit/ upload feedback into database
Click Header
Click Menu
Click CarPark
Click Forum
Search Page
entry/ initialize maps and location services
do/ update real-time availability and pricing
do/ refresh nearby carpark status periodically
View Details
entry/ fetch specific carpark details
do/ display carpark details
FilterSearch
entry/ display search bar and filters
do/ execute search query in real-time
exit/ save search history
Community Forum
entry/ refresh forum homepage and categories
do/ display all posts
exit/ save user's last viewed position
New Post
entry/ display post creation form with editor
do/ validate post
exit/ add post to community forum
Edit Comment/Post
entry/ display input field
do/ validate input
exit/ add updated post/comment to community forum
Comment
entry/ display comment input area and existing comments
do/ validate comment length and content
exit/ add/delete comment to post
Admin clicks view post
Delete Post/Comment
Close Website
Close Website
Close Website
Close Website

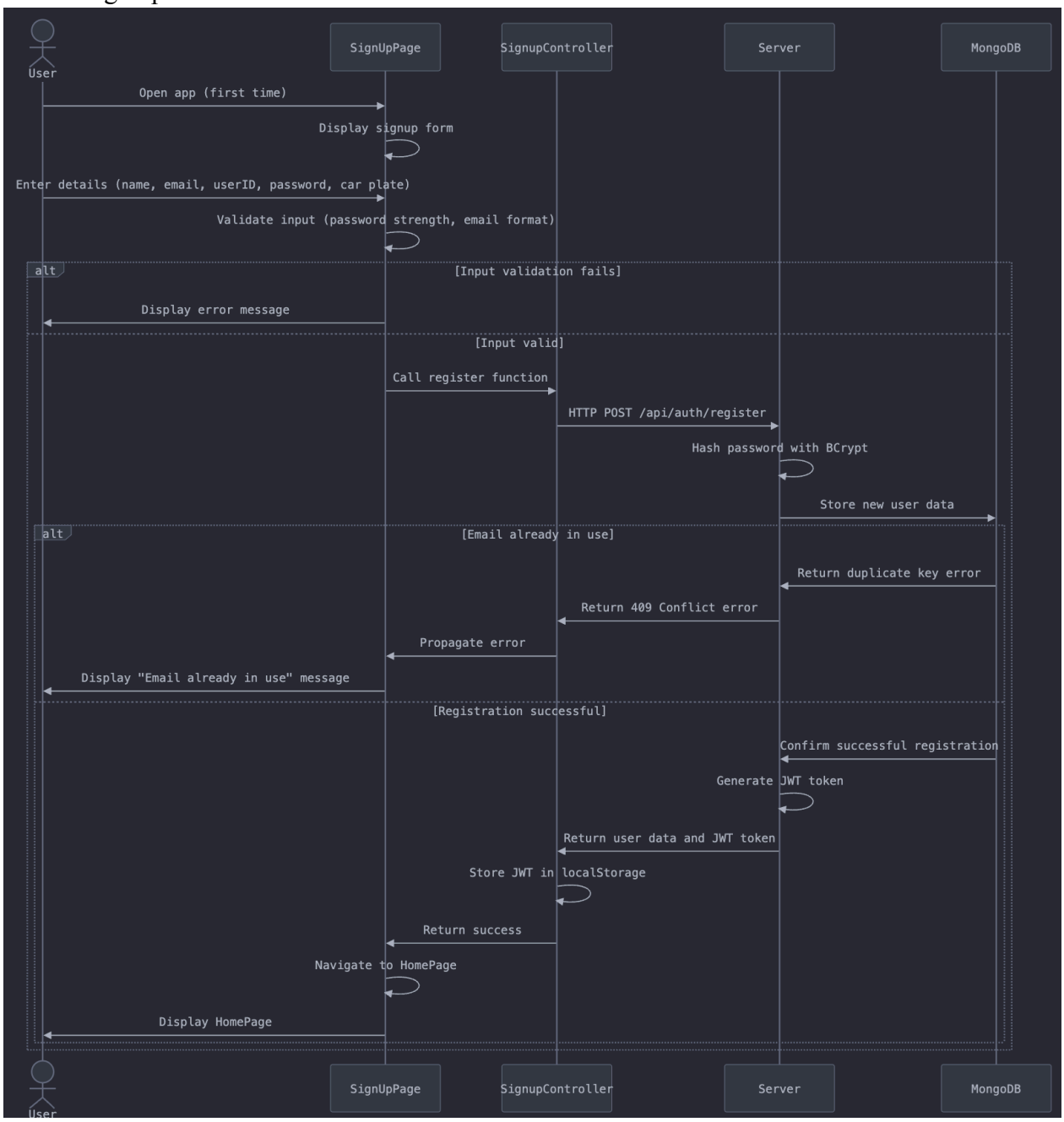
Dialog Map



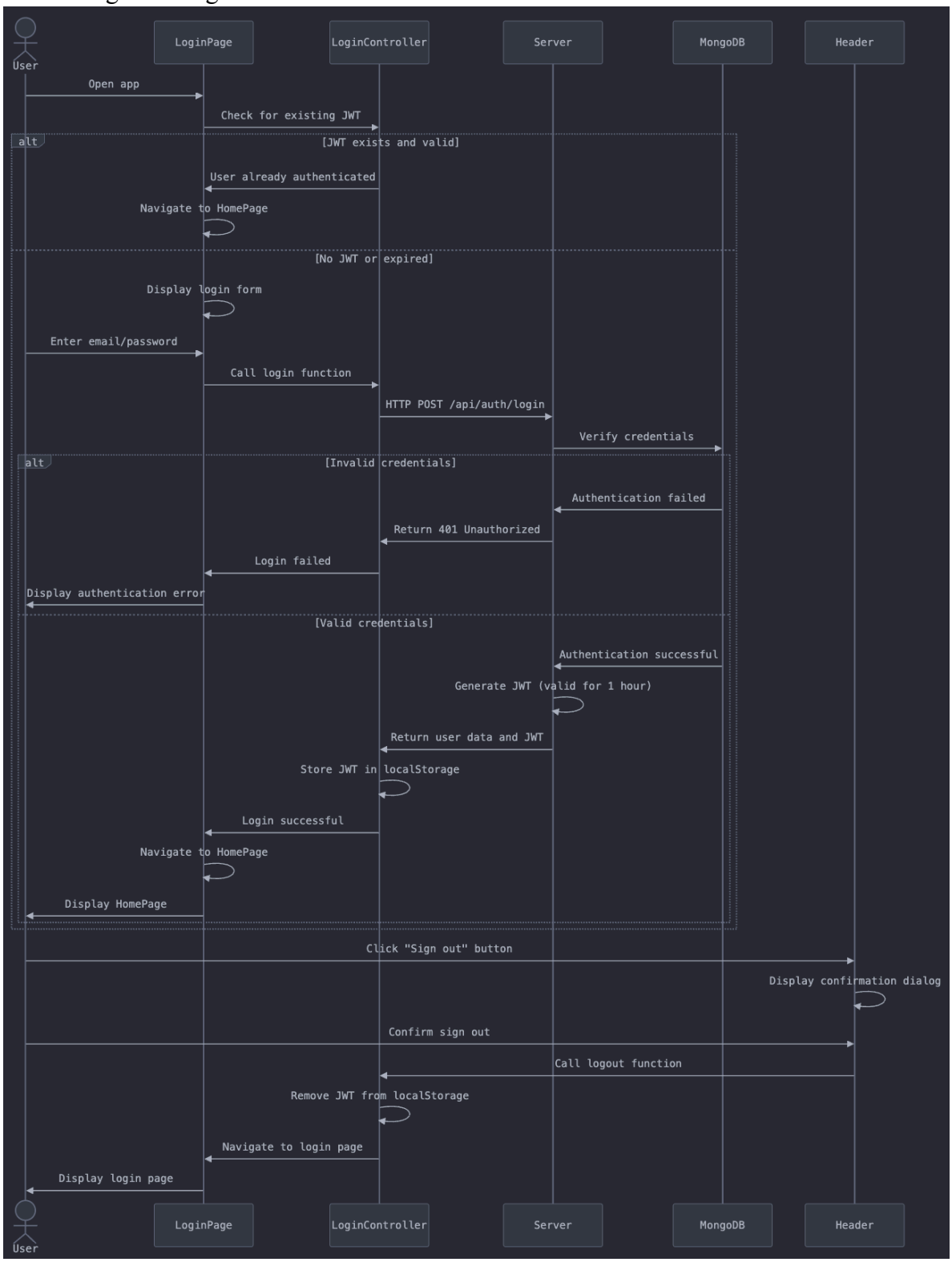
System Architecture Diagram

SEQUENCE DIAGRAMS

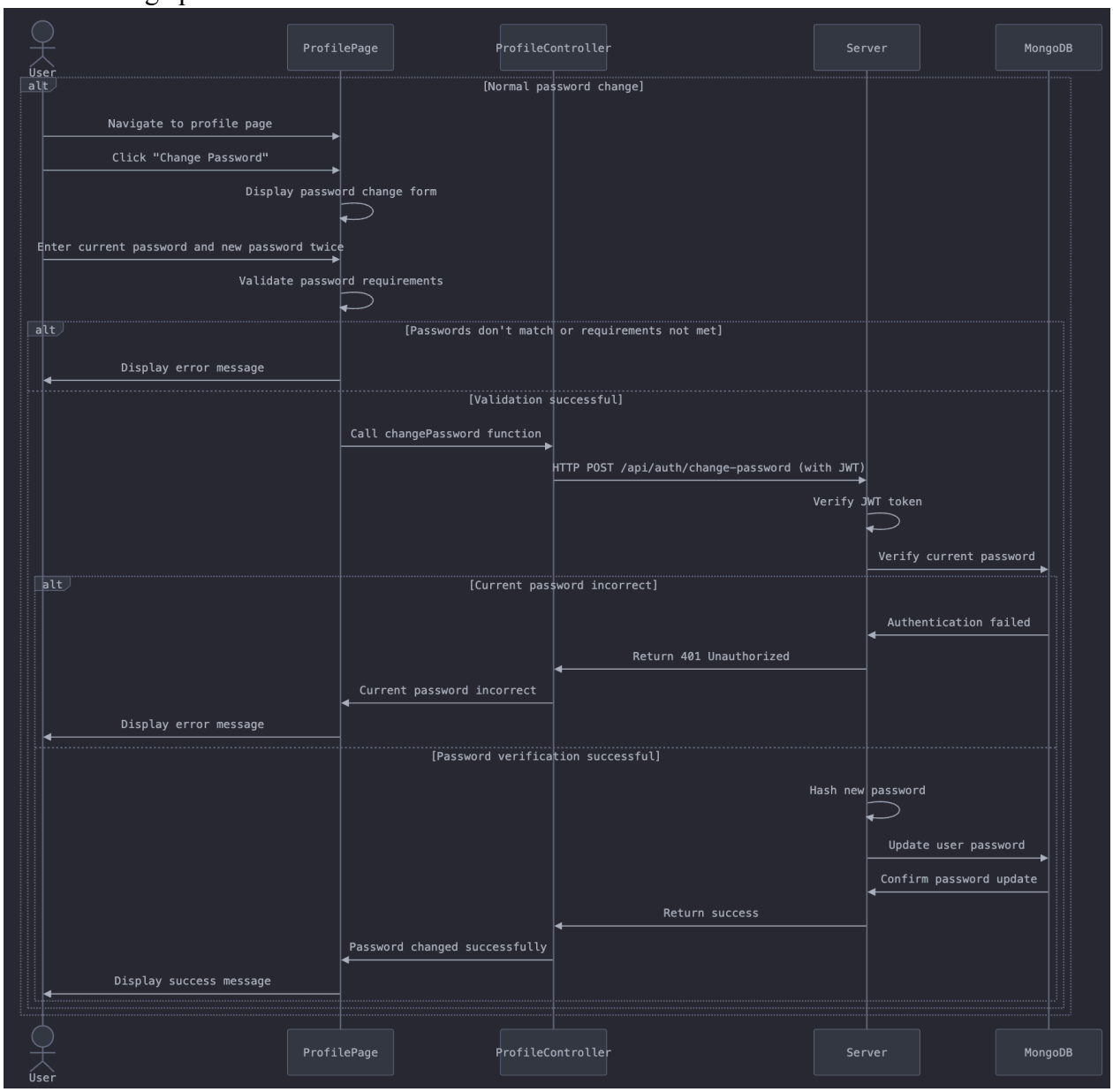
1. Sign up



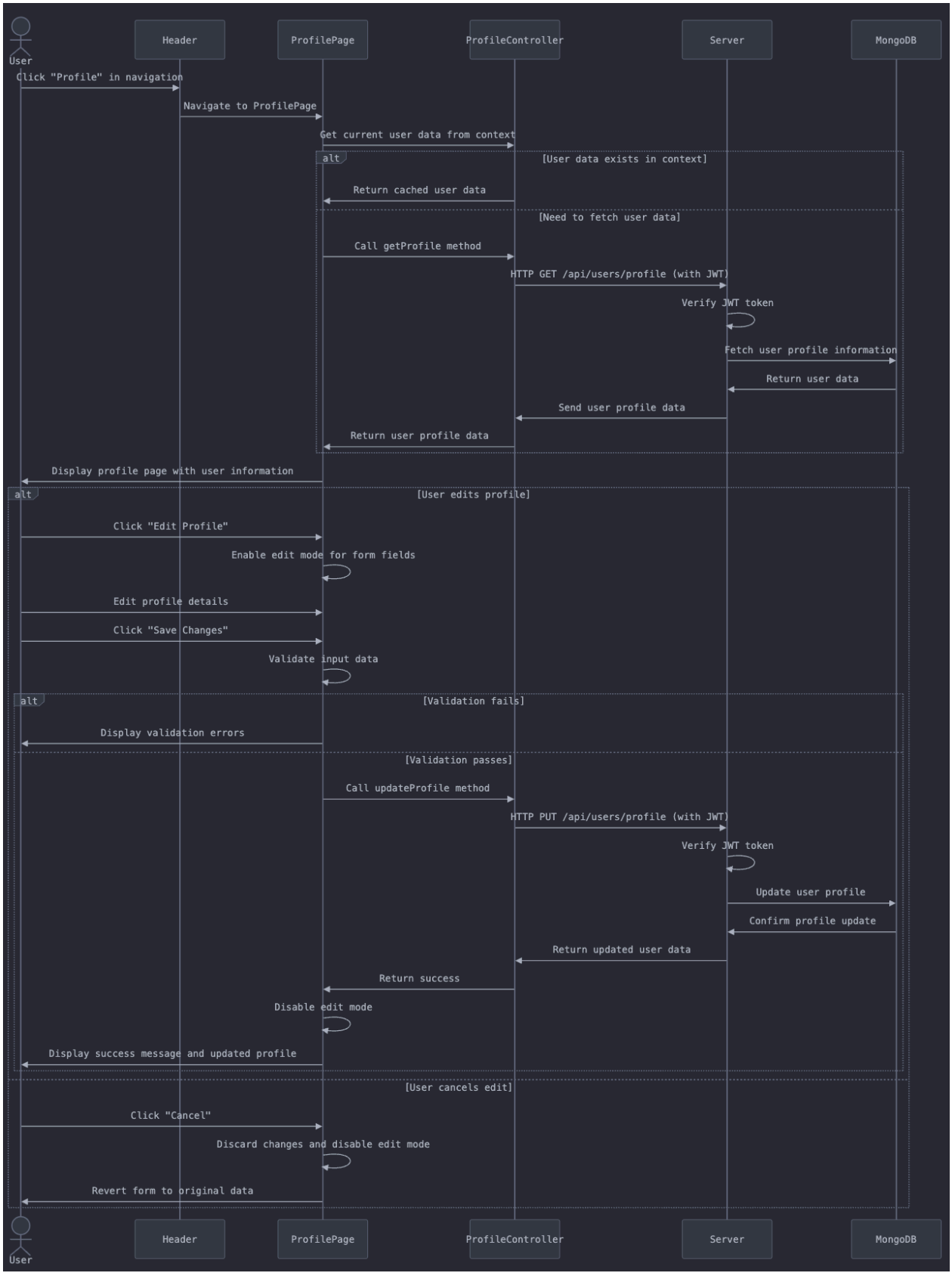
2. Login and sign out



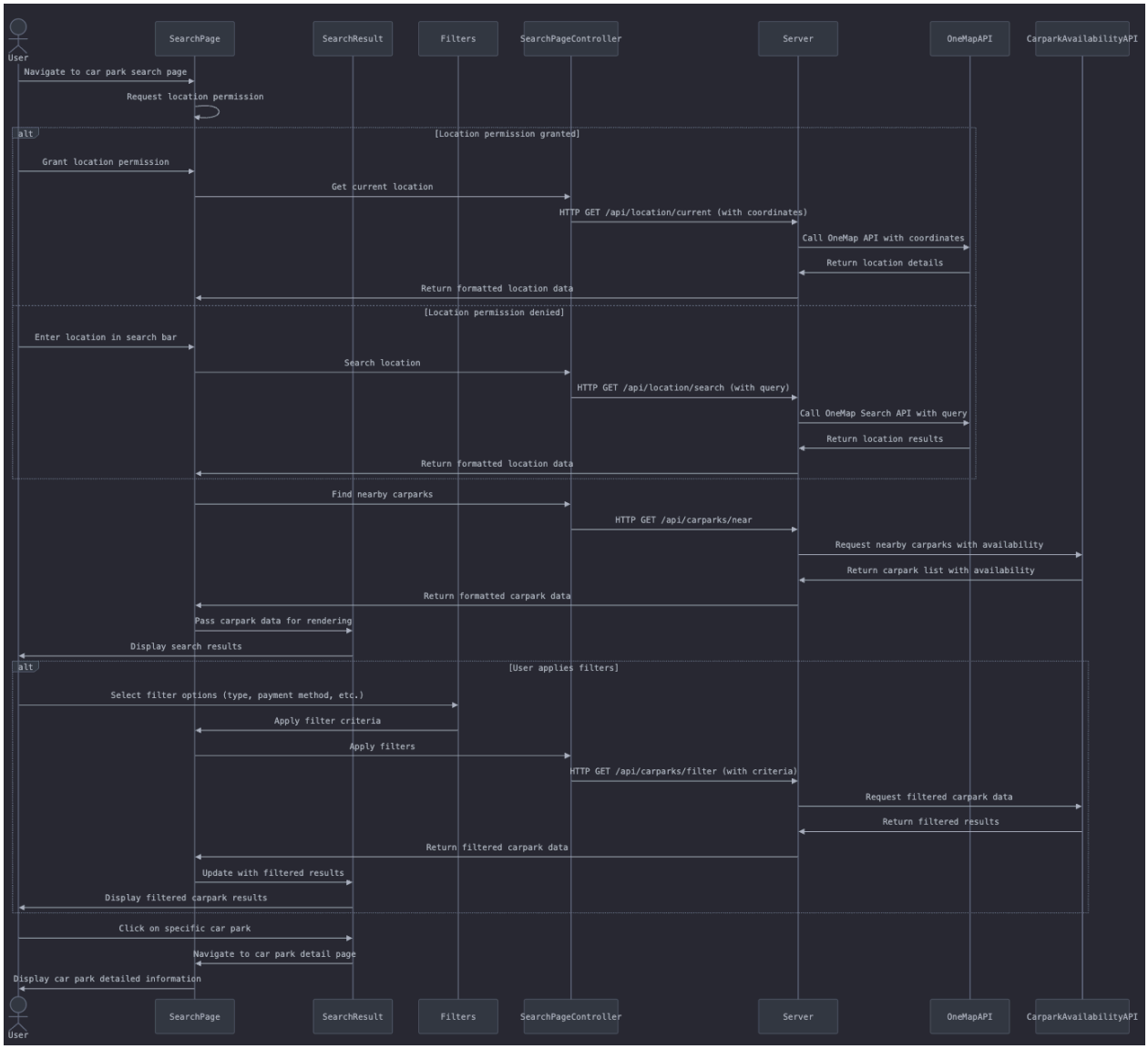
3. Change password



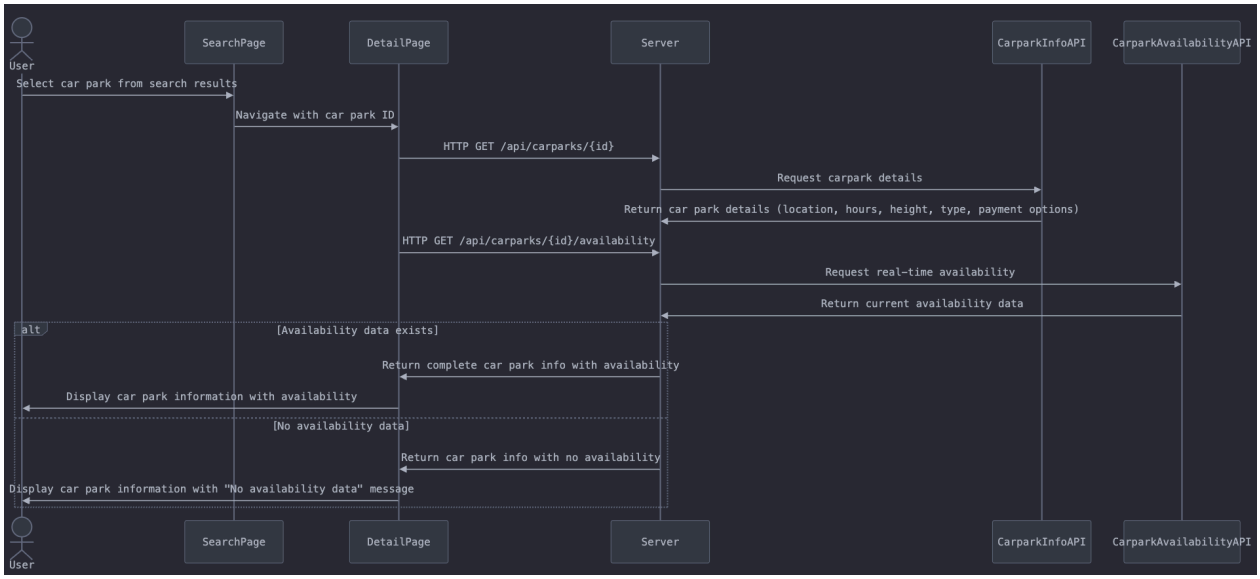
4. View and edit profile page


User
Header
ProfilePage
ProfileController
Server
MongoDB
Click "Profile" in navigation
Navigate to ProfilePage
Get current user data from context
alt
[User data exists in context]
Return cached user data
[Need to fetch user data]
Call getProfile method
HTTP GET /api/users/profile (with JWT)
Verify JWT token
Fetch user profile information
Return user data
Send user profile data
Return user profile data
Display profile page with user information
alt
[User edits profile]
Click "Edit Profile"
Enable edit mode for form fields
Edit profile details
Click "Save Changes"
Validate input data
alt
[Validation fails]
Display validation errors
[Validation passes]
Call updateProfile method
HTTP PUT /api/users/profile (with JWT)
Verify JWT token
Update user profile
Confirm profile update
Return updated user data
Return success
Disable edit mode
Display success message and updated profile
[User cancels edit]
Click "Cancel"
Discard changes and disable edit mode
Revert form to original data
User
Header
ProfilePage
ProfileController
Server
MongoDB

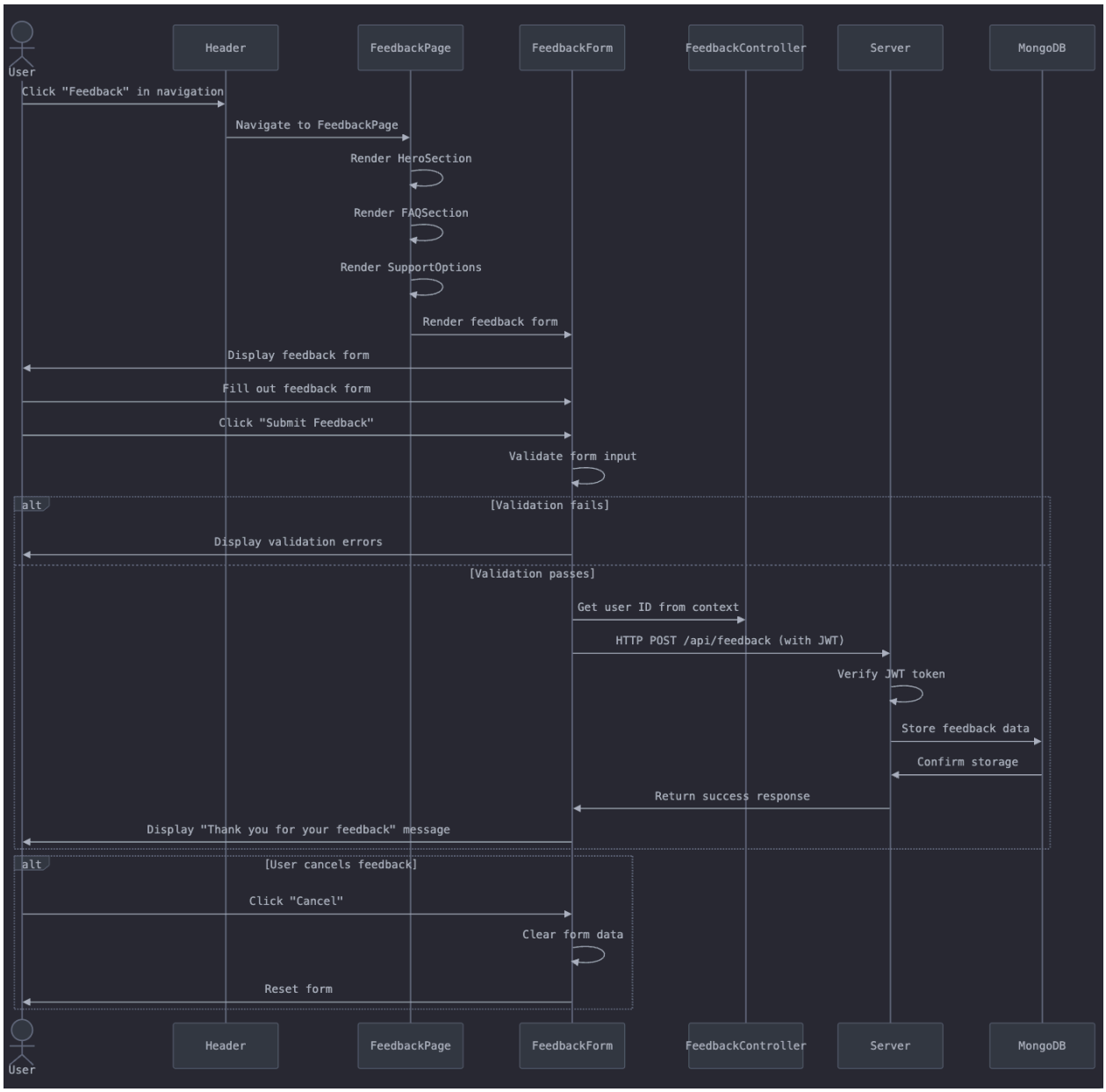
5. search carpark location



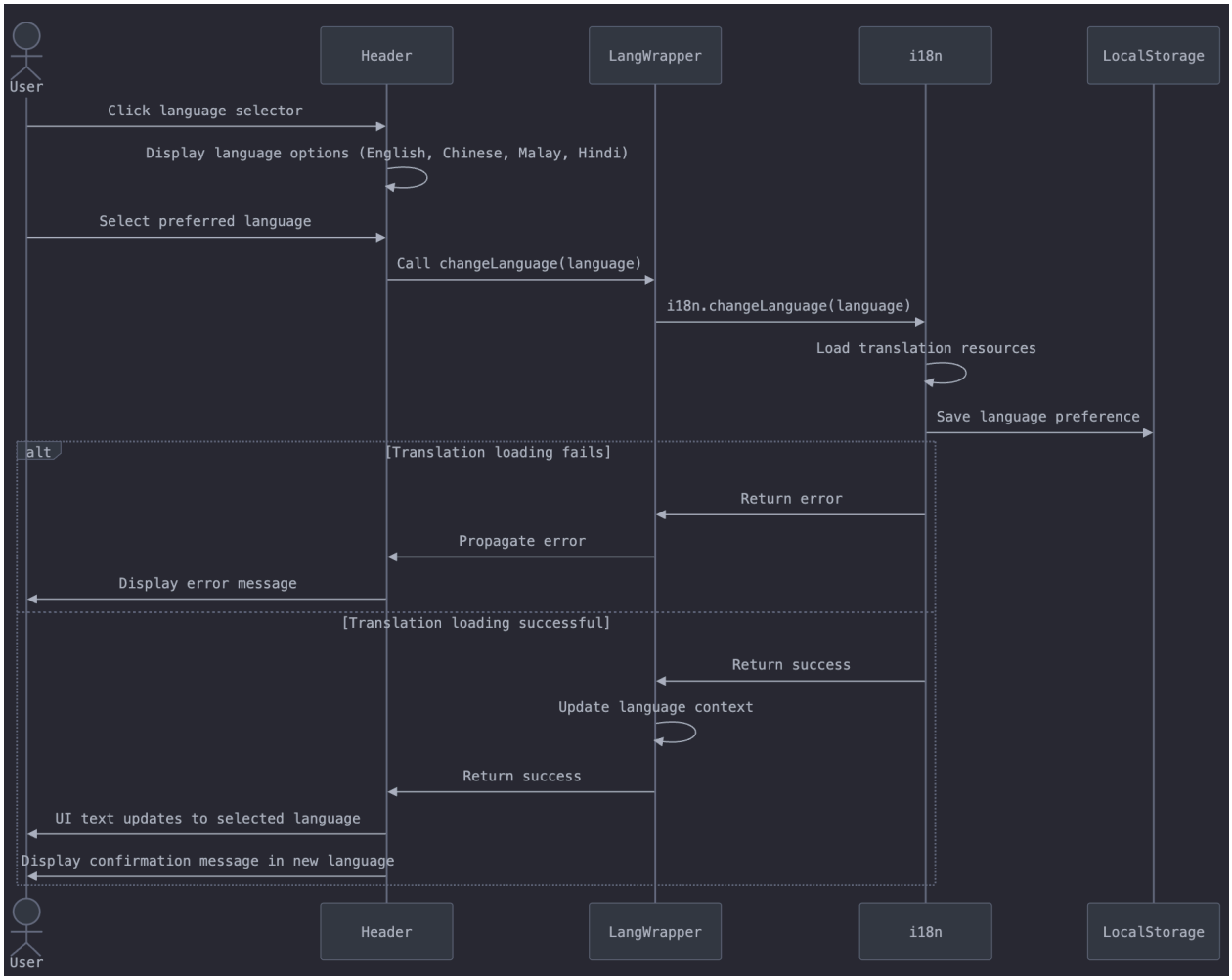
6. View carpark information



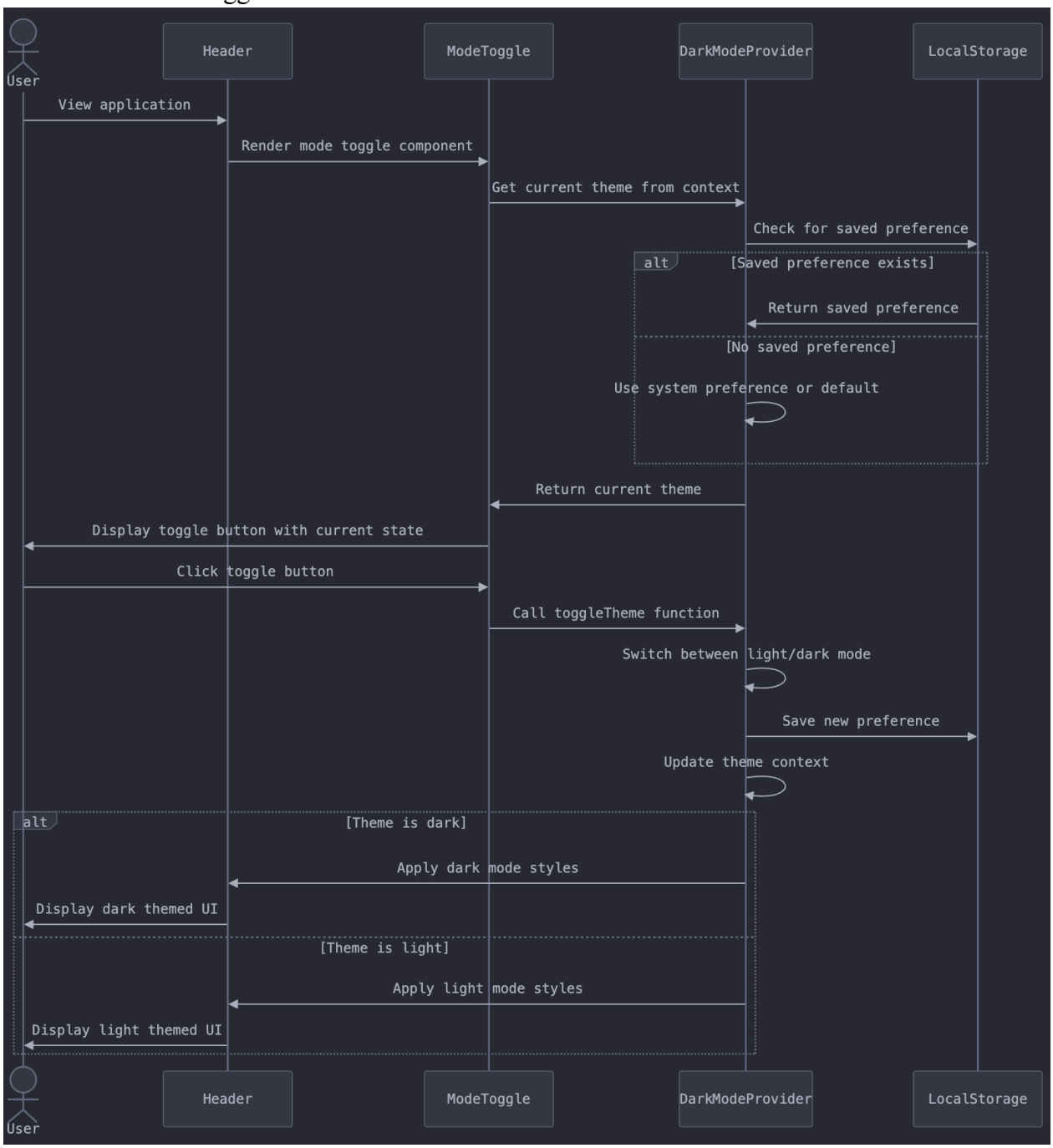
7. Provide Feedback



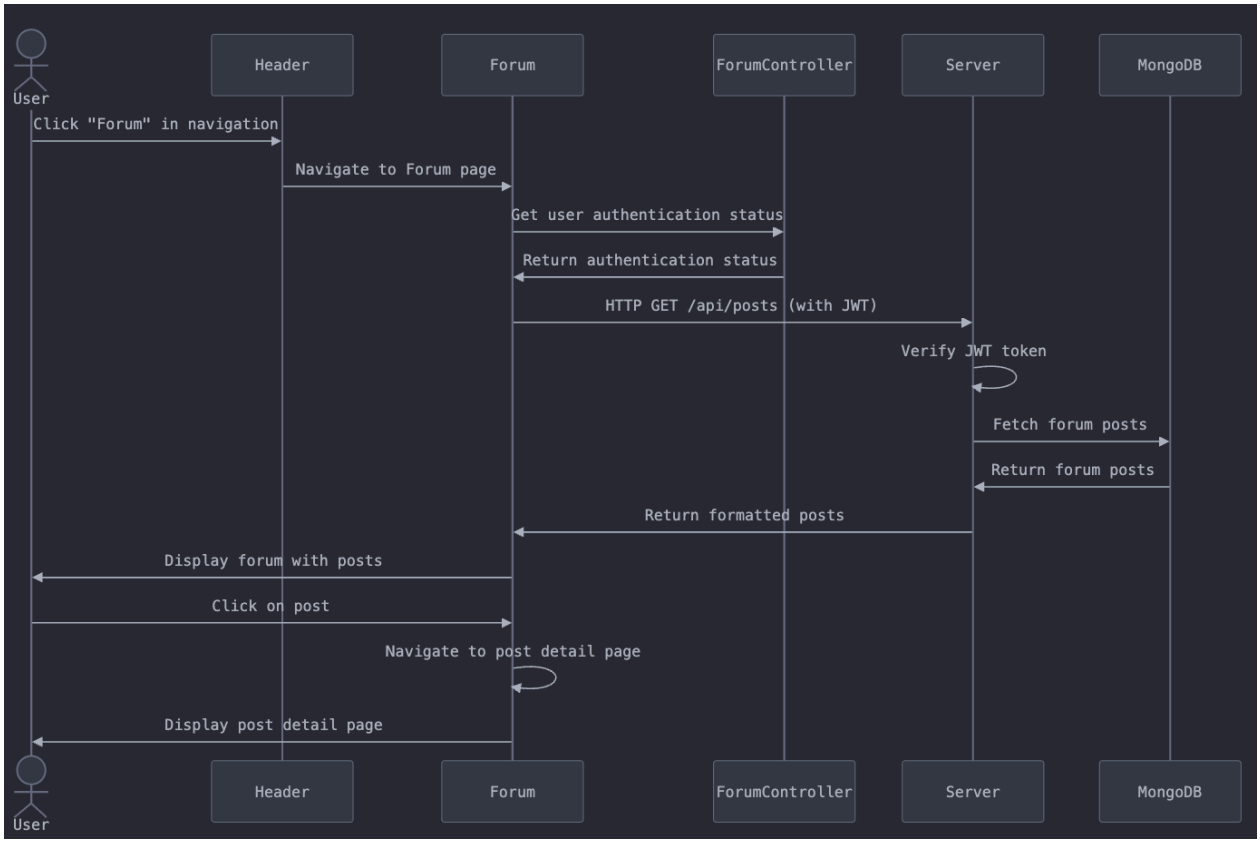
8. Change language


User
Header
LangWrapper
i18n
LocalStorage
Click language selector
Display language options (English, Chinese, Malay, Hindi)
Select preferred language
Call changeLanguage(language)
i18n.changeLanguage(language)
Load translation resources
Save language preference
alt
[Translation loading fails]
Return error
Propagate error
Display error message
[Translation loading successful]
Return success
Update language context
Return success
UI text updates to selected language
Display confirmation message in new language
User
Header
LangWrapper
i18n
LocalStorage

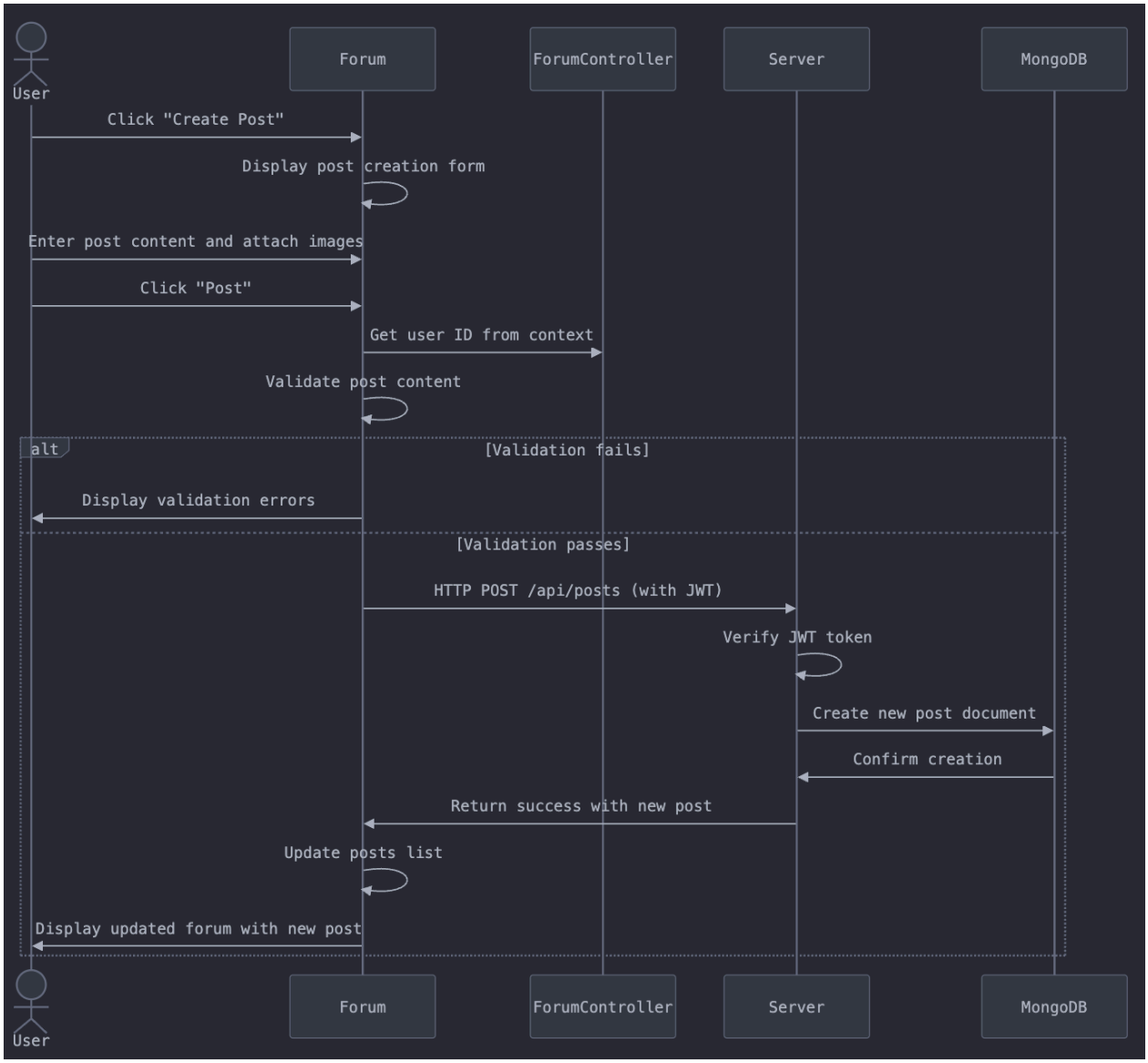
9. Dark mark toggle



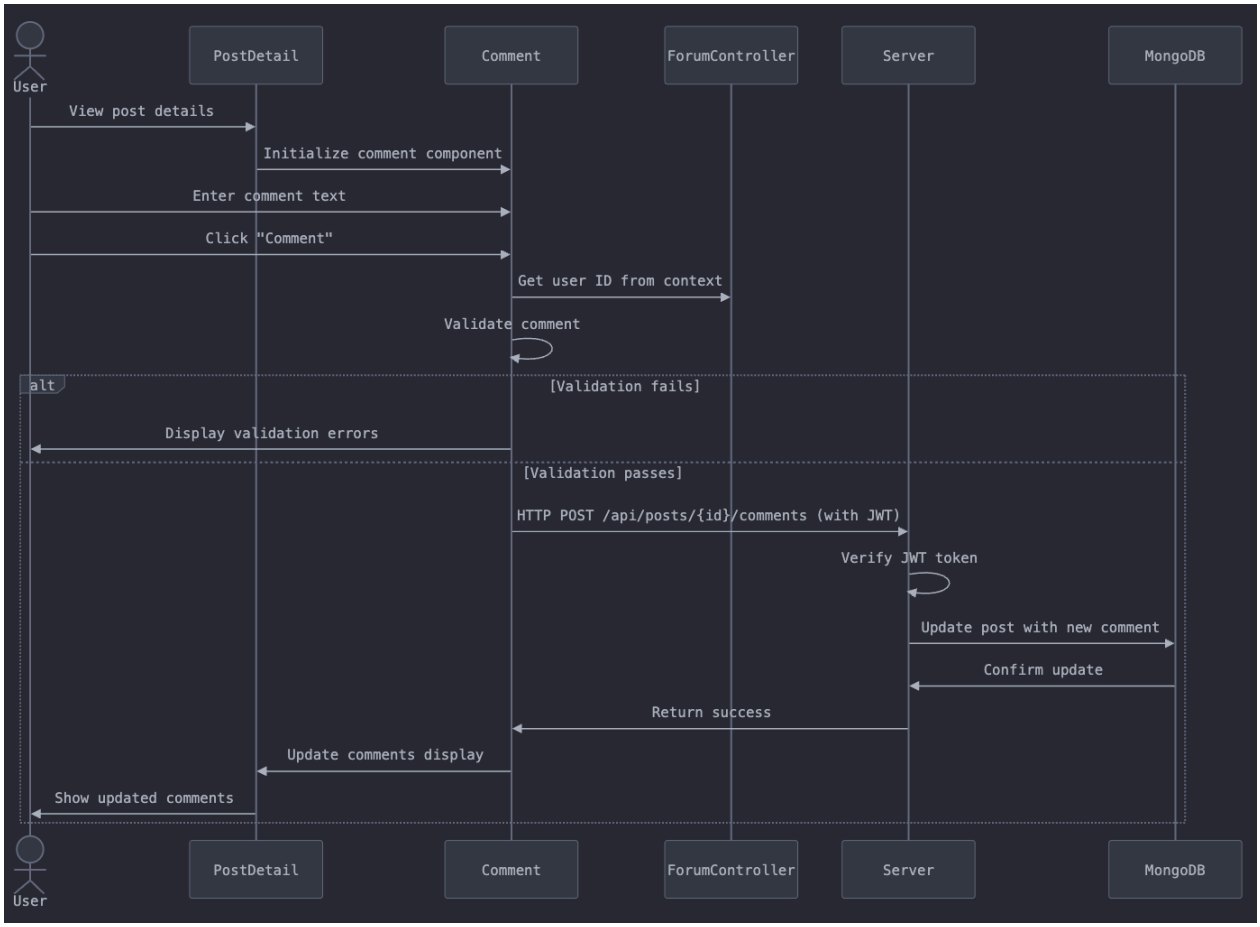
10. Access forum



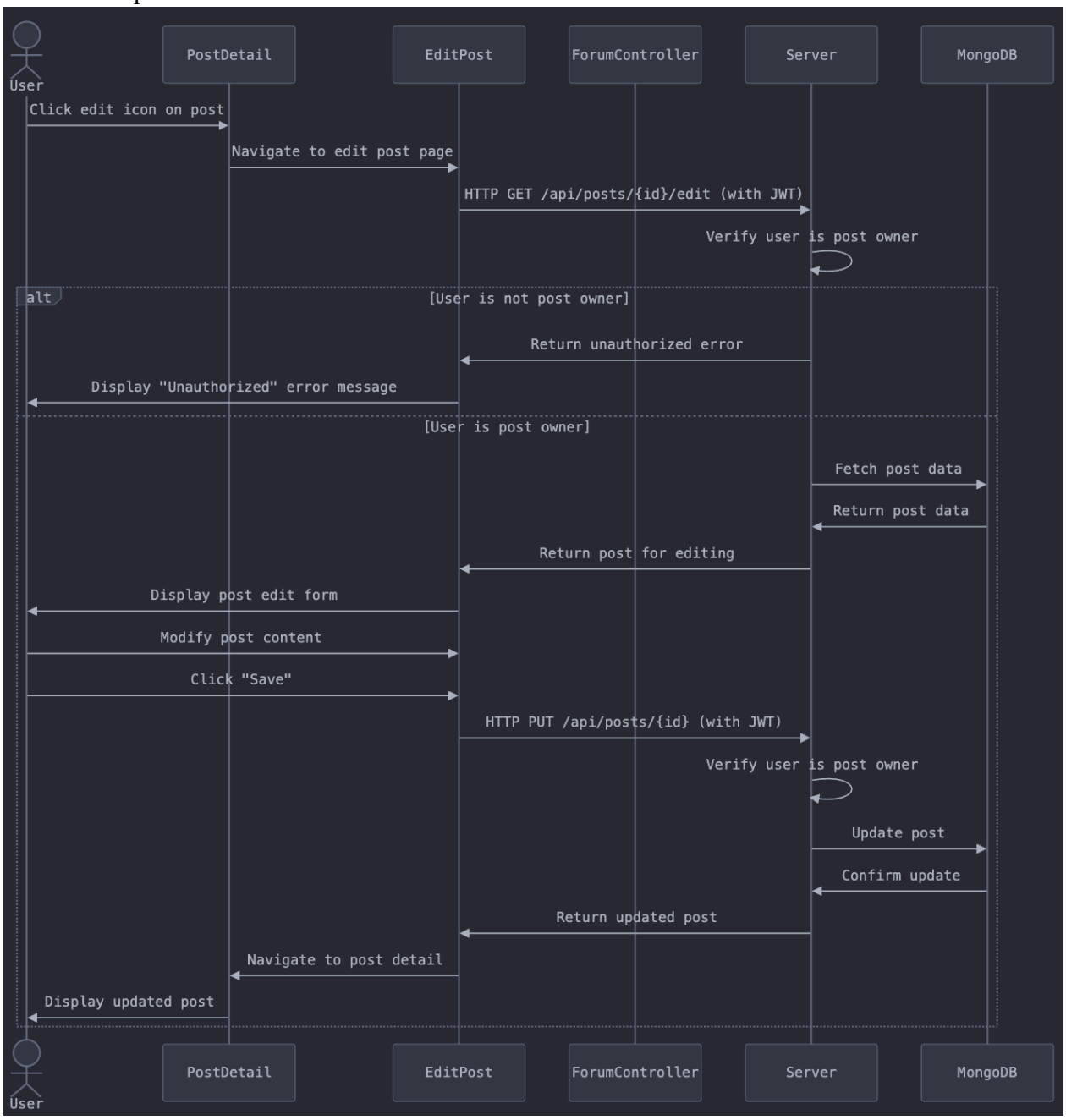
11. Create post



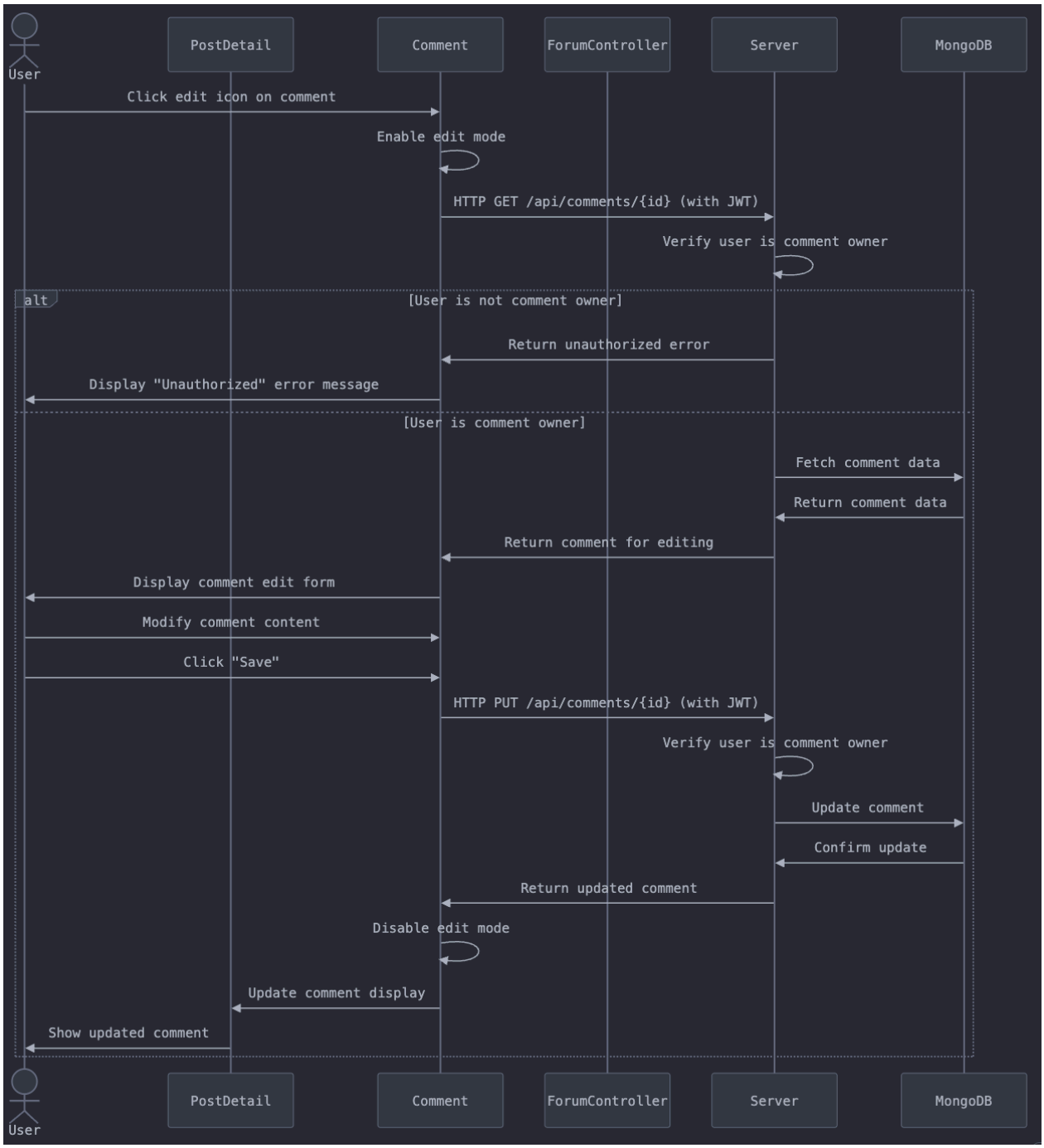
12. Add comment



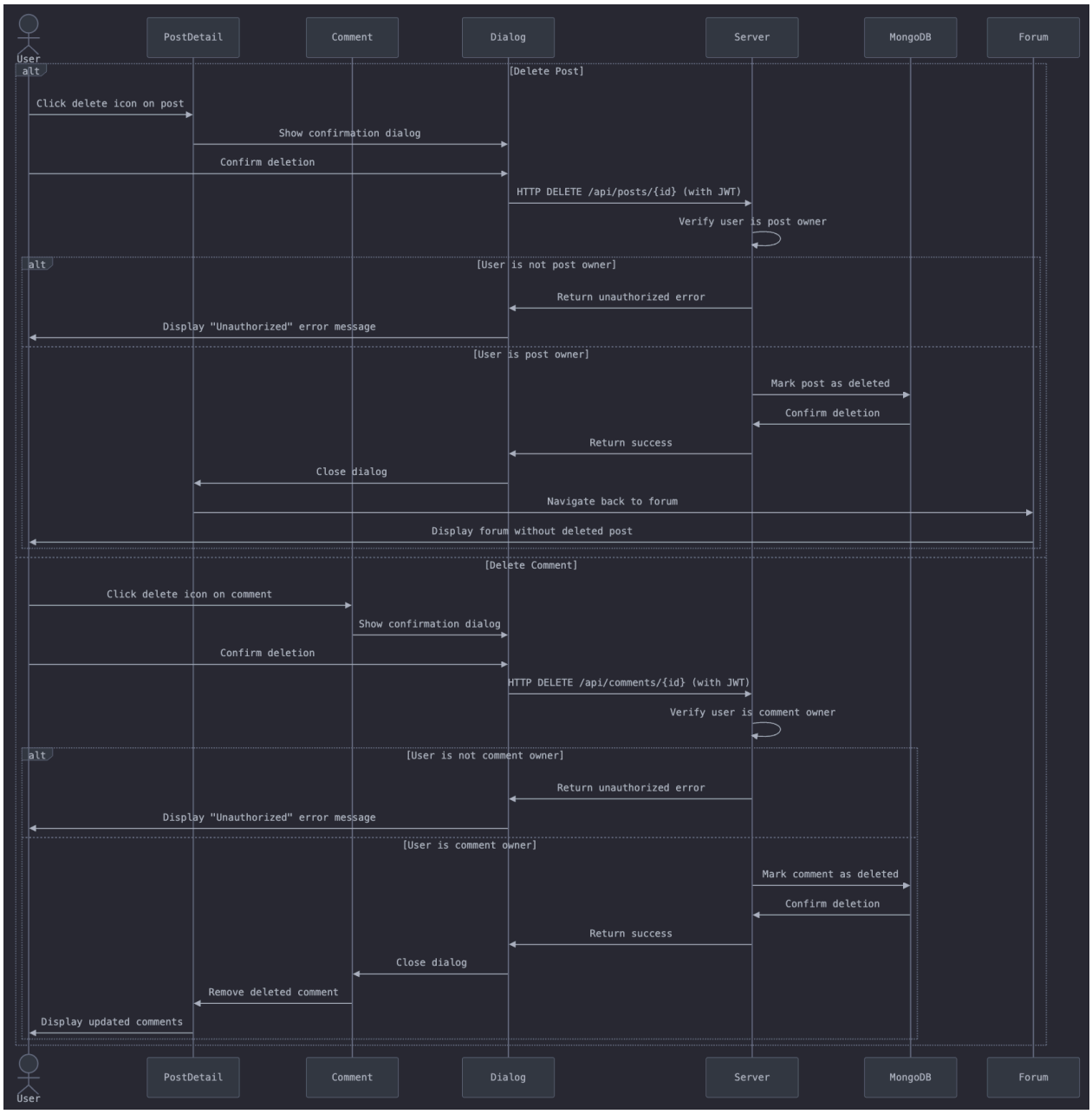
13. Edit post


User
PostDetail
EditPost
ForumController
Server
MongoDB
Click edit icon on post
Navigate to edit post page
HTTP GET /api/posts/{id}/edit (with JWT)
Verify user is post owner
alt
[User is not post owner]
Return unauthorized error
Display "Unauthorized" error message
[User is post owner]
Fetch post data
Return post data
Return post for editing
Display post edit form
Modify post content
Click "Save"
HTTP PUT /api/posts/{id} (with JWT)
Verify user is post owner
Update post
Confirm update
Return updated post
Navigate to post detail
Display updated post
User
PostDetail
EditPost
ForumController
Server
MongoDB

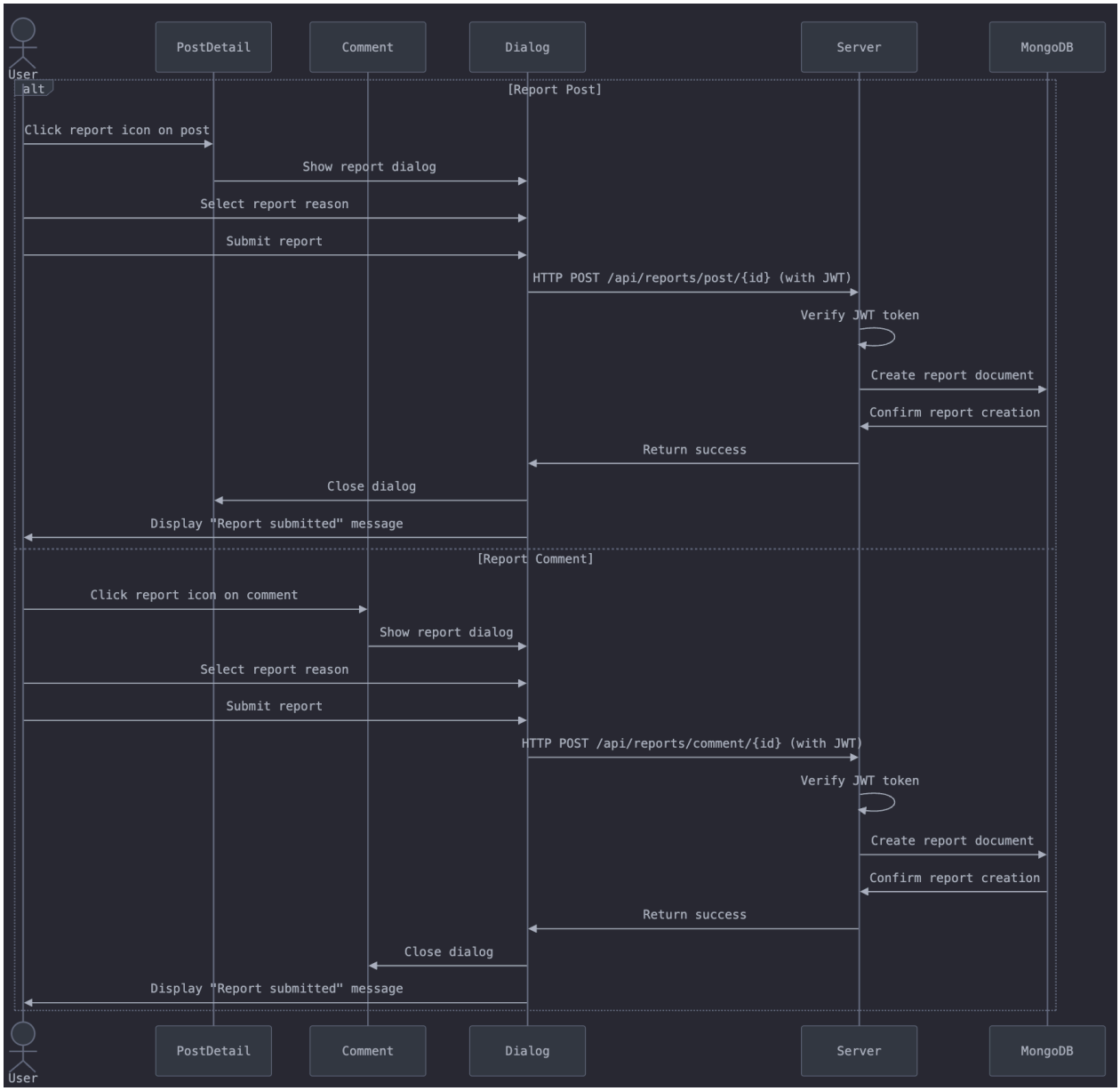
14. Edit comment



15. Delete post / comment

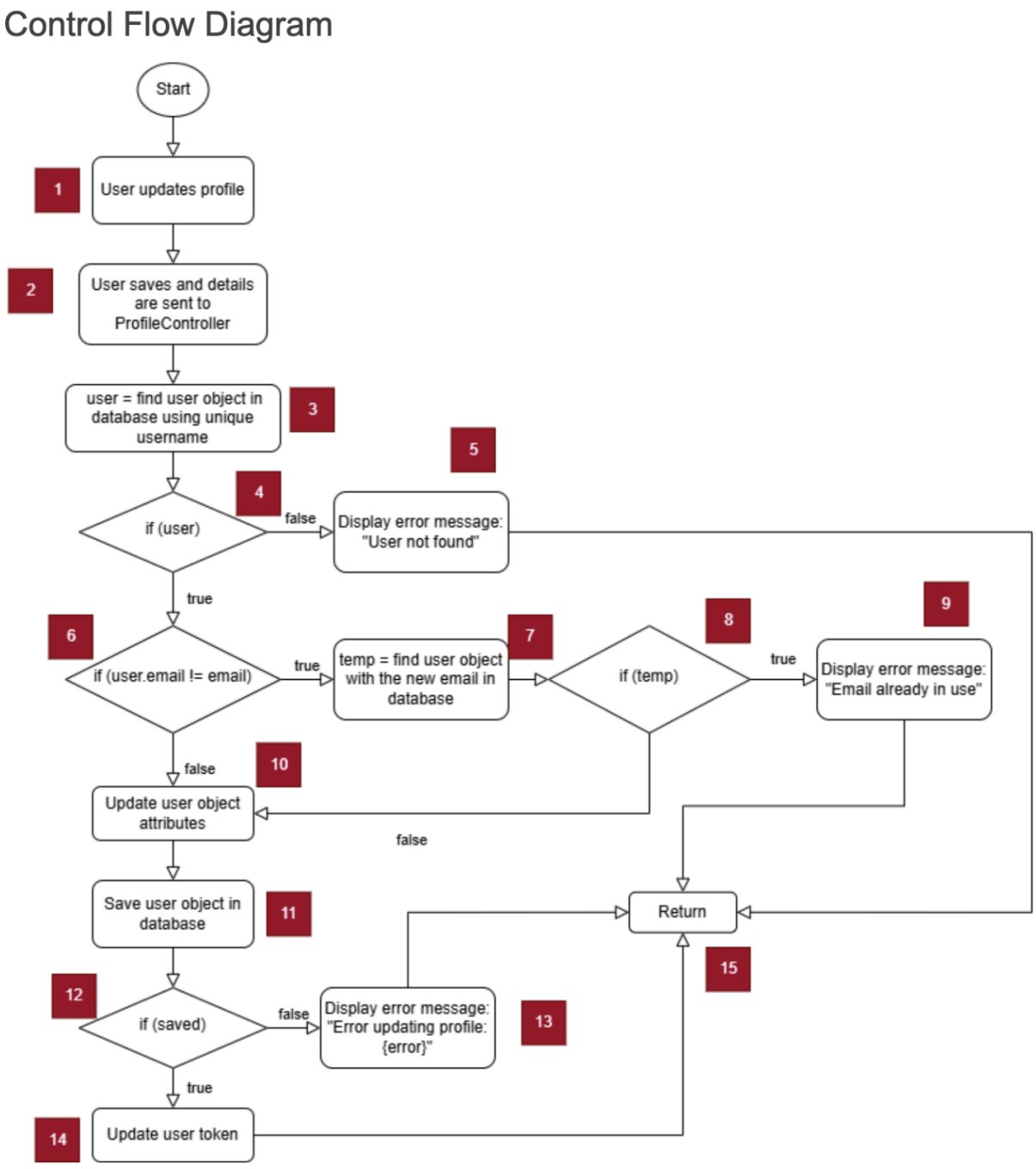


16. Report post/comment

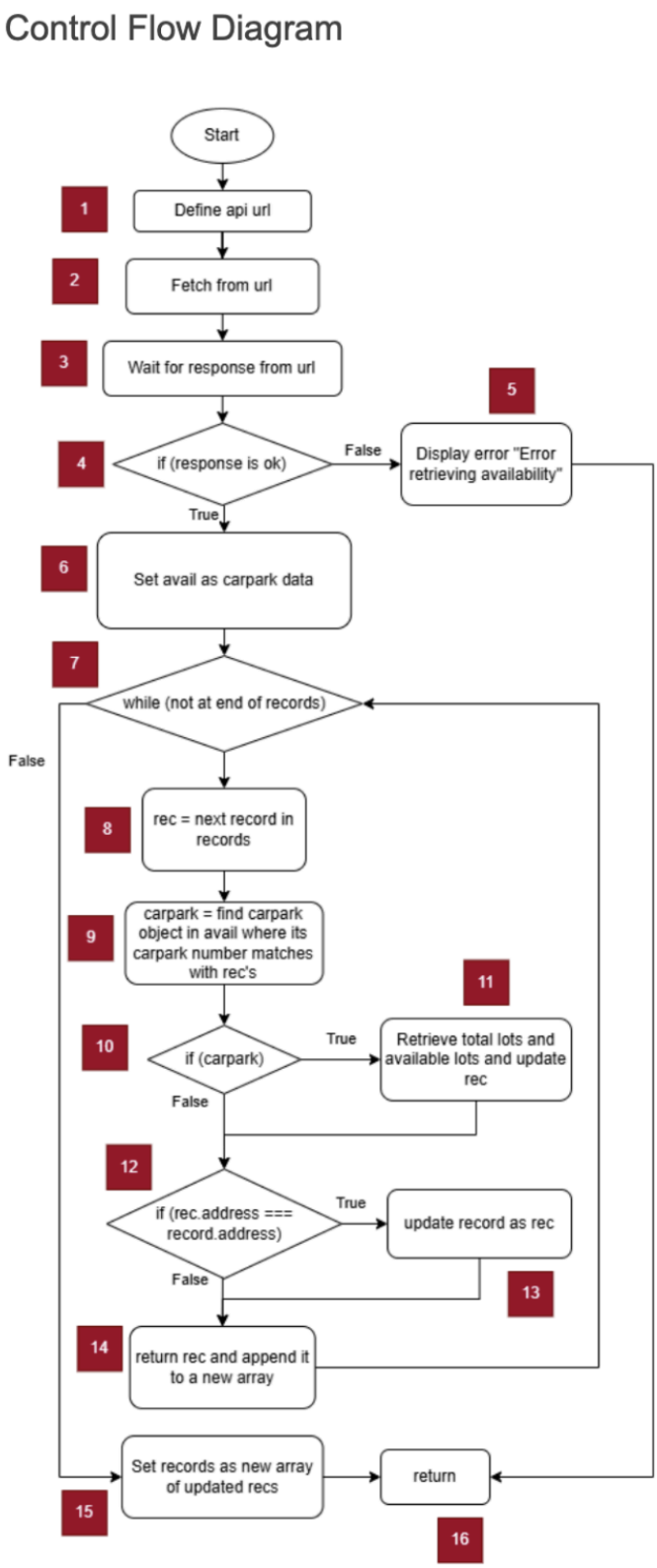


Test Input			Expected Output	Actual Output
Email	Username	Password		
newemail@gmail.com	newuser	Testing@123	No error, redirect to homepage	No error, redirect to homepage
newemail	newuser	Testing@123	Error message: “Please include an ‘@’ in the email address.’newuser’ is missing an ‘@’.”	Error message: “Please include an ‘@’ in the email address.’newuser’ is missing an ‘@’.”
existingemail@gmail.com	newuser	Testing@123	Error message: “Email already in use”	Error message: “Email already in use”
newemail@gmail.com	existinguser	Testing@123	Error message: “Username already exists”	Error message: “Username already exists”
newemail@gmail.com	newuser	Tes@123	Only requirement 1 (8 characters) is not fulfilled. Error message: “Please match the requested format.”	Only requirement 1 (8 characters) is not fulfilled. Error message: “Please match the requested format.”
newemail@gmail.com	newuser	Testing123	Only requirement 2 (1 special character) is not fulfilled. Error message: “Please match the requested format.”	Only requirement 2 (1 special character) is not fulfilled. Error message: “Please match the requested format.”
newemail@gmail.com	newuser	Testing@	Only requirement 3 (1 number) is not fulfilled. Error message: “Please match the requested format.”	Only requirement 3 (1 number) is not fulfilled. Error message: “Please match the requested format.”
newemail@gmail.com	newuser	testing@123	Only requirement 4 (1 uppercase letter) is not fulfilled. Error message: “Please match the requested format.”	Only requirement 4 (1 uppercase letter) is not fulfilled. Error message: “Please match the requested format.”
newemail@gmail.com	newuser	TESTING@123	Only requirement 5 (1 lowercase letter) is not fulfilled. Error message: “Please match the requested format.”	Only requirement 5 (1 lowercase letter) is not fulfilled. Error message: “Please match the requested format.”

Black Box Testing for SignUpController



Basis Path Test for Update User Details



Basis Path Test for Fetching Carpark Availability Data