ParKNGo : A Parking Management System

Submitted By

Md. Alamin Ridoy

ID: 221-35-1012

Supervised By

Mr. Khalid Been Badruzzaman Biplob

Lecturer (Senior Scale),

Department of Software Engineering,

Daffodil International University

This project report has been submitted in fulfilment of the requirements for the degree of **Bachelor of Science in Software Engineering**

DAFFODIL INTERNATIONAL UNIVERSITY

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name	:	Md. Alamin Ridoy
Date of Birth	:	12-01-2001
Title	:	ParkNGo : A Parking Management System
Academic Session	:	2022-2025

I declare that this thesis is classified as:

☐	CONFIDENTIAL	(Contains confidential information under the Official Secret Act 1997)*
☐	RESTRICTED	(Contains restricted information as specified by the organization where research was done)*
☐	OPEN ACCESS	I agree that my project to be published as online open access (Full Text)

I acknowledge that Daffodil International University reserves the following rights:

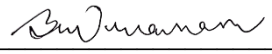
1. The Project is the Property of Daffodil International University.
2. The Library of Daffodil International University has the right to make copies of the Project for the purpose of research only.
3. The Library of Daffodil International University has the right to make copies of the Project for academic exchange.

Certified by:

Alamin	
_____	_____
(Student's Signature)	(Supervisor's Signature)
221-35-1012	Mr. Khalid Been Badruzzaman Biplob
_____	_____
Student ID	Name of Supervisor
Date:	Date:

SUPERVISOR'S DECLARATION

I/We* hereby declare that I/We* have checked this project and in my/our* opinion, this project is adequate in terms of scope and quality for the award of the degree of Bachelor of Science.



(Supervisor's Signature)

Full Name : Mr. Khalid Been Badruzzaman Biplob

Position : Lecturer (Senior Scale)

Date :

STUDENT'S DECLARATION

I hereby declare that the work in this project is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Daffodil International University or any other institution.

Alamin

_____

(Student's Signature)

Full Name : **Md. Alamin Ridoy**

ID Number : **221-35-1012**

Date : **11 December, 2025**

TITLE: ParKNGo - A Parking Management System

Md. Alamin Ridoy

Project submitted in fulfillment of the requirements
for the award of the degree of
Bachelor of Science

Department of Software Engineering

DAFFODIL INTERNATIONAL UNIVERSITY

November 2025

ACKNOWLEDGEMENTS

First and foremost, I am grateful to Almighty Allah, who has given me the strength, patience, and ability to complete this project successfully.

I would like to give my special thanks to my respected supervisor, “Mr. Khalid Been Badruzzaman Biplob” Sir, of the Department of Software Engineering, for continuous guidance, valuable suggestions, and constant encouragement through the development of this project. His expertise, constructive feedback, and support have proven to be invaluable to me.

I am also grateful to all those who participated in the survey and interviews relating to this project, whose willingness to share their opinions and experiences was of immense help to the success of my project.

I would like to take this opportunity to sincerely thank all the faculty members of the Department of Software Engineering, Daffodil International University, for their academic guidance, motivation, and support throughout my studies.

Last but not the least, I would like to extend my deepest gratitude to my parents for their unconditional love, incessant support, and relentless encouragement. Their blessings and motivation have always been the direct and constant source of my strength.

DEDICATION

I therefore declare that I have done this project under the oversight of **"Mr. Khalid Been Badruzzaman Biplob"**, **Lecturer (Senior Scale)**, **Department of Software Engineering**, **Daffodil International University**. Also declare that neither entire record nor any portion of this record has been submitted somewhere else for my degree.

ABSTRACT

The **ParKNGo Parking Management System** is a full-stack web-based application developed to address common urban parking problems by providing a smart and digital solution. The system connects customers who are searching for parking spaces with hosts who own available parking spots, making the entire parking process simple, fast, and reliable. Users can register and log in as **Customers, Hosts, or Admins**, and each role has clearly defined functionalities. Customers can search for nearby parking spots, apply filters such as location, price, and availability, and complete bookings through an easy-to-use interface. Hosts can add new parking spots, update availability, activate or deactivate listings, view customer bookings, and track their total earnings through a dashboard. Admins are responsible for managing users, approving parking listings, monitoring bookings, and ensuring smooth system operation. The application is designed to be secure, scalable, and responsive, with features such as real-time availability updates, efficient booking management, and a clean, user-friendly interface. Overall, the ParKNGo platform streamlines urban parking by improving convenience for customers, creating earning opportunities for hosts, and providing effective administrative control over the system.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	FULL FORM
JWT	JSON Web Token
OAuth	Open Authorization
UI	User Interface
API	Application Programming Interface
ERP	Enterprise Resource Planning
UAT	User Acceptance Testing
SRLC	Software Release Life Cycle

LIST OF APPENDICES

CHAPTER: 1 INTRODUCTION

1.1 Background

As the urban areas grow even faster and the number of vehicle owners increases, the users find the available parking or rentable spots to be a usual problem. The conventional parking management methods are largely manual, inefficient and time consuming and may result in frustration of the user and under utilization of resources. Meanwhile, there are numerous parking areas that are not utilized by the owners of the properties and stay in disarray and are hard to control. This scenario shows why a digital platform which can effectively match users who seek parking with hosts which have free spots is necessary. The proposed project will help fill this need by providing a web based spot booking and management system that will make the whole process simpler by automating it and having a centralized control.

1.1.1 Context and Relevance

The modern age of digitalization presupposes that people require prompt, trustworthy, and convenient internet-based solutions to the problems they face in their daily lives. The field of parking and spot management systems are yet to be established in most of the local environments particularly where manual operations prevail. The current solutions tend to be non usable, non transparent, and ill managed. The project is topical because it is able to comply with the latest ideas of smart cities and trends of digital services provision, providing a system which is well organized and user-friendly. It is customer and host friendly; thus it can be used in real life situations in city and semi-city set ups.

1.1.2 Problem Identification

The main issue this project will be dealing with is that there is no centralized and efficient platform to manage short-term parking or rentable spots. Customers have challenges when finding an available place, availability, and making bookings in an efficient way. Conversely, hosts have inadequate tools to control the listings, track the bookings and control the incomes. Also, the existing systems are mostly manual-intensive, feature few security measures, as well

as lack real-time insights, which translates to inaccurate information and inadequate user experience.

1.1.3 Purpose and Justification

This project is aimed at designing and developing a web-based system of spot booking and management that enhances accessibility, efficiency, and user experience. The system will strive to make the booking procedure easier, minimize manual handling, and present tabular dashboards of various user categories. The project is justified because it shows the real case of the use of web technologies, database management, and the principles of system design. It is also a solution to a real-life issue, as well as provides a universal core to add further improvements later like payment integration or real-time tracking.

1.1.4 Scope

The deliverables of this project involve the development of a working web application that provides user registration, user authentication, spot listing, search, booking and simple availability management. The system offers individual customer, host and administration dashboards and image uploads and database-supported operations. Nevertheless, the project does not encompass advanced offerings like the online payment gateways, real time GPS positioning and the development of mobile applications. The emphasis is still placed on providing an implementable, user friendly, and well organized web based solution within the stipulated project parameters.

1.2 Project Planning and Initiation

Project planning and initiation entail the analysis of viability of the proposed system prior to the onset of development. This phase will make sure that the project is feasible, within the reach and in line with the user requirements and resources at their disposal. The project was analyzed step by step to present it in terms of operational, technical, and financial analysis through the feasibility study. The phases assisted in defining the scope of the project, choosing the right technologies and mitigating possible risks of the project development.

Feasibility Study (Step-by-Step)

This was the stage of grasping the essence of the issue and determining the goals of the parking/spot booking management system. Current parking systems and manual operations

have been examined to find out the constraints of the current parking systems including the inability to have a centralized control of the parking systems and poor booking system. According to this analysis, project scope was outlined to accommodate the required characteristics which included user authentication, spot listing, searching, booking management and basic availability control. The device did not include some of the non-essential features such as online payment and real-time GPS tracking to ensure that the project is manageable within the time frame.

Phase 1 Preliminary Analysis & Project Scope Definition:

The market feasibility involved the analysis of the demand and applicability of digital spot booking system. Online research, informal surveys, and studying the current platforms were carried out to get insight into user expectations and frequently encountered challenges. The results showed that there is increasing demand of simple web-based booking systems, particularly in the urban setting where the problem of parking is common. This discussion revealed that the system proposed is a solution to a practical problem in the real world and has some practical applicability to users and service providers.

Phase 2 Market Feasibility Analysis (or Market Research):

The demand and usability of a digital spot booking system were analyzed in the market feasibility analysis. Informal surveys, on-line research, and observation of existing platforms were done to have the idea of what the users expect and what are the general problems. The results showed that there was an increasing demand for simple, web based booking systems particularly in cities where parking problems are common. The analysis established that the proposed system covers a practical issue in real-life setting and is practical in its application by users and service providers.

Phase 3 Technical Feasibility Analysis:

The technical feasibility study assessed the ability of the project to be developed successfully with the existing technology and skills. The system was created with the help of Spring Boot as the backend developer, MySQL as a database manager, and Angular as frontend tools that include HTML, CSS, and TypeScript and JWT can allow building a secure, scalable, and responsive smart parking system. The technologies are highly supported, open-source and are appropriate to develop a safe and scalable web application. The environment and tools used in

development were easy to get and the project was technically possible to be performed according to the planned schedule.

Phase 4 Financial Feasibility Analysis:

The cost-effectiveness of the project was determined by the financial feasibility analysis. The low cost of development is due to the fact that the system is based on open-source software and standard development tools. The cost involved is primarily restricted to the hardware usage, internet access and development time. Academic implementation does not need any licensing or deployment costs. Thus, the project can be defined as financially viable and cost-effective and can be used in academic research and on a small scale in real life.

1.3 Target User Profile and Tentative Elicitation Process

1.3.1 Target User

The parking/spot booking management system, as proposed, is to be used by various user groups that will utilize the system differently. The targeted system users and stakeholders are the following:

Customers : They are the people who find parking or rentable parking spaces, access the information, and book parking spaces via the system.

Hosts / Spot Owners : Property owners that advertise the spots available, maintain the availability, handle bookings, and track the income.

Admin : The task of this person may include the management of the user accounts, monitoring activity in the system, and compliance with the correct functioning of the system.

Maintenance and Support : Support the system and maintain it when necessary.

Project Developer : Part of the system formulation, testing and subsequent maintenance.

Academic Supervisors : To do academic evaluation, review, and validate the system.

1.3.2 User profile

Table 1: User Profile for Customer

User Class	Notes on Characteristic	Requirement Implied
Type of User	Customer (Parking Seeker)	Verification
Age Range	15 <	Performance, Acceptance
Frequency of Use	Regular	Performance, Acceptance
Computer Experience	Basic–Intermediate	Usability, Acceptance
Education	School/University	Usability, Acceptance
Language Skills	Bangla & English	Usability, Acceptance
Number of Users	Many	Performance, Scalability, Acceptance
Training	Not required	Usability, Documentation
Other Systems Used	Maps, digital payments	Compatibility, Usability
Ways of Working	Searches, filters, books, cancels spots	Usability, Performance

Table 2: User Profile for Host

User Class	Notes on Characteristic	Requirement Implied
Type of User	Host (Parking Owner / Spot Provider)	Verification
Age Range	17 <	Performance, Acceptance
Frequency of Use	Moderate	Performance, Acceptance
Computer Experience	Intermediate	Usability, Acceptance
Education	School/University	Usability, Acceptance
Language Skills	Bangla & English	Usability, Acceptance
Number of Users	Medium	Performance, Scalability, Acceptance
Training	Not required	Usability, Documentation
Other Systems Used	Banking apps	Compatibility, Usability
Ways of Working	Listing management, earnings tracking	Usability, Performance

Table 3: User Profile for Admin

User Class	Notes on Characteristic	Requirement Implied
Type of User	Admin	Verification
Age Range	24-50	Performance, Acceptance
Frequency of Use	Daily	Performance, Acceptance
Computer Experience	Advanced	Usability, Acceptance
Education	IT/CS Background	Usability, Acceptance
Language Skills	Bangla & English	Usability, Acceptance
Number of Users	Few	Performance, Scalability, Acceptance
Training	Required	Usability, Documentation
Other Systems Used	Admin tools, DB, monitoring tools	Compatibility, Usability
Ways of Working	Listing management, earnings tracking	Usability, Performance

1.3.3 Elicitation Process

In order to obtain correct, and pertinent system requirements, several requirements elicitation methods were employed during the initial project life cycle phases.

Interviews were carried out with stakeholders (hosts (the owners of the parking space), users (drivers), and system administrators) to learn about everyday parking problems, behavior during booking, and the expectations that the system managers held.

Also, online surveys would be shared with the possible users to gather a wider perspective on frequent parking challenges, most preferred system features, and usability needs. The feedback served to determine the main expectations like ease of booking, availability visibility and system simplicity.

Document analysis was also done by analyzing the existing parking processes, paper logs, booking books, and other informal ways of booking. This discussion revealed some serious concerns such as the absence of transparency, a possibility of having a double booking, and the absence of a centralized documentation system.

Moreover, field research involved going to common parking spots to see actual practice of operations and interaction with users. Lastly, prototyping and feedback were provided where initial user interface wireframes were shown to stakeholders. Their reviews were used in an iterative way to improve usability and functionality of the system.

1.4 Project Block Diagram

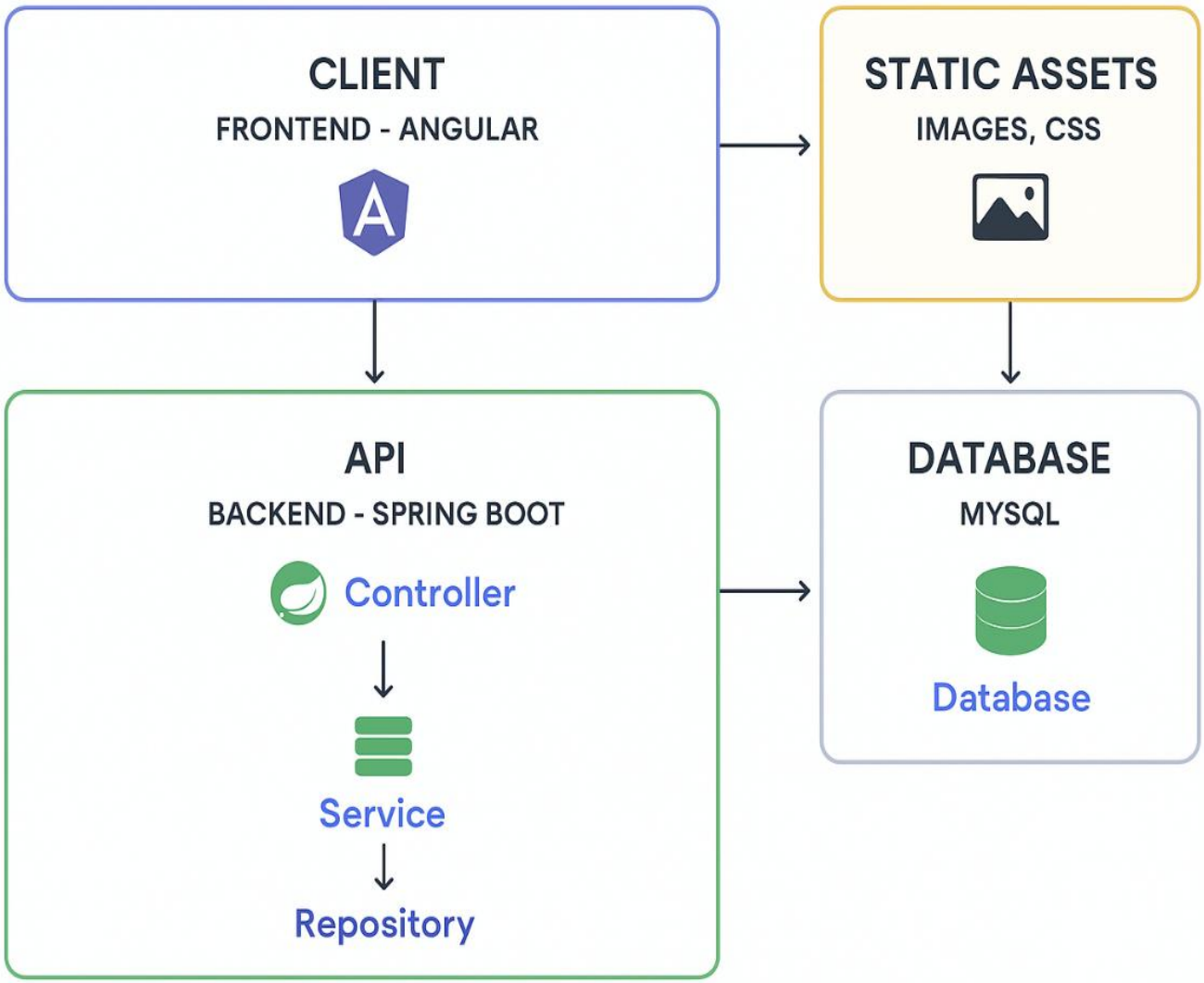


Figure 1: System Block Diagram

1.5 System Requirements

1.5.1 Hardware Requirements

Client-Side Hardware

- **Device:** Desktop, Laptop, Smartphone, or Tablet
- **Processor:** Minimum Dual-Core 2.0 GHz (Intel/AMD equivalent)
- **RAM:** Minimum 4 GB (8 GB recommended for smooth browser performance)
- **Storage:** 200–500 MB free space for browser cache and temporary files
- **Display:** Minimum screen resolution of 1366 × 768
- **Internet Connection:** Stable connection (minimum 1 Mbps; recommended 5 Mbps or higher)
- **GPU (Optional):** Integrated graphics sufficient for UI rendering

Server-Side Hardware

- **Processor:** Quad-Core 2.4 GHz or higher
- **RAM:** Minimum 8 GB (recommended 16 GB for production use)
- **Storage:** 50–100 GB SSD for application files, database, logs, and uploads
- **Network:** Reliable high-availability internet connection
- **Backup Storage:** External or cloud storage for database backups and system logs

1.5.2 Software Requirements

Client-Side Software

- **Operating Systems:**
  - Windows 10/11
  - Linux (Ubuntu/Fedora)
  - Android/iOS (via responsive web access)
- **Web Browsers (Latest Versions):**
  - Google Chrome
  - Mozilla Firefox
  - Microsoft Edge

Server-Side Software

- **Operating System:**
  - Linux (preferred for deployment)
  - Windows Server
- **Backend Technology:**
  - Spring Boot Core (Primary backend technology)

- **Frontend Technology:**
  - Angular (with Angular CLI for module/component generation)
  - Bootstrap (for UI styling)
- **Database:**
  - MySQL
  - phpmyadmin (for admin and DB management)
- **Authentication & Security:**
  - Spring Security
  - JWT + OAuth for secure API access
- **Development Tools:**
  - Visual Studio Code
  - XAMPP/WAMP for local development
  - Git and GitHub for version control
- **Runtime/SDK Requirements:**
  - Java Development Kit (JDK) 17/21
  - Node.js + npm (required for Angular)
  - Angular CLI

1.5.3 Constraints and Dependencies

Constraints

- **Technology Dependency:** System must use Angular for frontend, Spring boot for backend, and MySQL for data management.
- **Cross-Browser Requirement:** Must run on modern browsers that support Angular's latest rendering engine.
- **Authentication Rules:** Protected endpoints must require valid JWT tokens generated via Spring Security.
- **Security Constraints:** All communication must use HTTPS. Passwords must follow Spring Security complexity rules.
- **File Upload Limitations:** Maximum size restrictions on evidence uploads (set by server configuration)
- **Server Scalability Constraints:** MySQL performance depends on server RAM and disk I/O.
- **Platform Policy Constraints:** All booking, cancellation, and host listing processes must comply with ParKNGo operational policies and user-agreement rules.

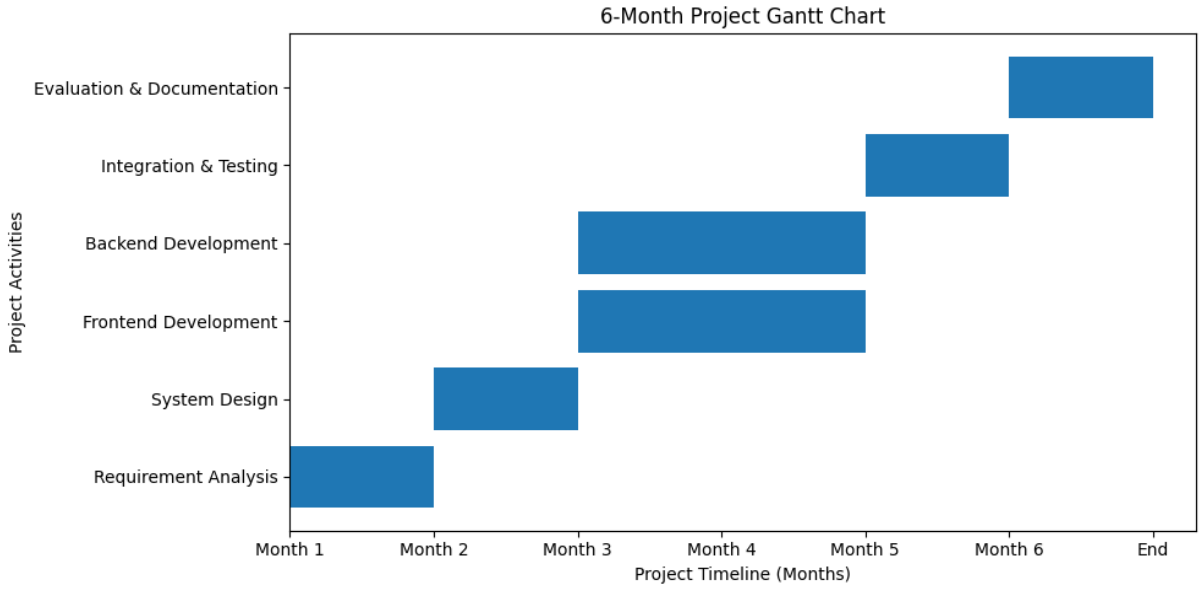
Dependencies

- **Framework Dependencies:** Angular components depend on Node.js and Angular CLI.

  The backend depends on **Spring Boot**, **Spring Web**, **Spring Data JPA**, and the **Java Runtime Environment (JRE/JDK)**.

- **Database Dependency:** MySQL database must be online and accessible for operations like user authentication, spot listings, availability updates, and booking management.
- **Third-Party Authentication Services:** OAuth provider availability affects login and token generation.
- **Version Control Dependency:** Git hosting services (GitHub/Azure) required for project tracking and collaboration.
- **Hosting Environment:** Deployment depends on a stable Linux/Windows server with proper configuration.

1.6 Project Scheduling



1.7 Summary

This chapter gave the general picture of the research suggested parking/spot booking system, its background, objectives, and importance. The current issues with the traditional parking management were presented to support the necessity of an online and centralized solution. The chapter also described the process of project planning and initiation that involves feasibility

analysis and system requirements to show the feasibility of the proposed system and its preparedness.

Moreover, target users and requirement eliciting methods were also determined to make the system match the actual user requirements. The project schedule was set so as to indicate a logical and realistic development schedule. In general, this chapter presents a solid basis on the further design and implementation stages of the project.

CHAPTER: 2 DESIGN AND IMPLEMENTATION

2.1 Introduction

This chapter is dedicated to the design and implementation of the very proposed parking/spot booking management system. It explains the process of converting the system requirements that were found in the previous chapter into an organized system design and a working web application. The chapter describes the general architecture of the system, database design, user interface plan and technical development technologies. It also describes the process of implementation of the core modules like user authentication, spot listing, booking management and dashboard capabilities. As it is evident in this chapter, the principles of design and programming methods are practically applied in order to create an efficient and easy-to-use system.

2.2 Functional Requirements

FR01	**Registration**
Description	Allows new users (Customer or Host) to register with full name, email, password, and role-specific information.
Stakeholder	Customer, Host, Admin

FR02	**User Login**
Description	Enables registered users to securely log in using email and password, receiving a session token (JWT).

Stakeholder	Customer, Host, Admin

FR03	**Spot Search & Filter**
Description	Allows users to search for available spots based on location, date/time range, price, vehicle type, and amenities.
Stakeholder	Customer, Host, Admin

FR04	**Spot Listing Manage**
Description	Allows authenticated Hosts to create a new spot listing by providing address, rate, description, availability, and uploading images.
Stakeholder	Host, Admin

FR05	**Spot Booking**
Description	Allows authenticated Customers to select a spot and time, proceed to secure payment, and confirm the reservation.
Stakeholder	Host, Admin

FR06	**Payment Processing**
Description	The system must securely process payments for bookings and disburse earnings to the Host, retaining commission.

Stakeholder	Host, Admin

FR07	**Booking Cancellation**
Description	Allows Customers and Hosts to cancel confirmed bookings, initiating a refund based on pre-defined policy rules.
Stakeholder	Customer, Host, Admin

FR08	**Listing Update/Modification**
Description	Allows the Host to modify the details (rate, description, availability) of an existing spot listing.
Stakeholder	Host, Admin

FR09	**Listing Deletion**
Description	Allows the Host to set a listing to delete it permanently, provided there are no future bookings.
Stakeholder	Host, Admin

FR10	**Dashboard View**

Description	Provides personalized dashboards for each role: Customer (upcoming bookings), Host (earnings, current listings), and Admin (system overview, reports).
Stakeholder	Customer, Host, Admin

FR11	**Report Generation**
Description	Allows the Admin to generate reports on key metrics, such as total bookings, monthly revenue, and active listings.
Stakeholder	Admin

FR12	**Logout**
Description	Allows users to safely end their session and logout from the system.
Stakeholder	All users

2.3 Non-Functional Requirements

Non-functional requirements detail the performance of the system as opposed to the functionality of the system. These specifications concentrate on such quality attributes of the system like performance, reliability, portability, usability and security. In the case of the proposed parking/spot booking management system, the non-functional requirements of the

system will make it efficient, dependable and available on various platforms and retain a satisfactory user experience.

2.3.1 Performance

- All the web pages should be loaded by the system in 3 seconds, under normal conditions.
- The online system should have a capacity of 2,000 active users at their optimum.
- The database queries are expected to take less than 200ms in 95% percent cases.
- The search operation of available spots records needs to be in 2 seconds.

2.3.2 Reliability

- The system will have 99.5% uptime when there are academic calendar periods.
- Daily backups, which are automated with the ability to recover at a point in time.
- The integrity of these data should be preserved with support of transaction about critical operations.

2.3.3 Portability

- Available with the current web browsers.
- Desktop, Laptop, Tablet, and Mobile.
- Adaptation to a variety of screens.
- Simple implementation with other server platforms.
- Light configuration needed in different settings.

2.3.4 Usability

- Easy and easy to use interface.
- Easy navigation among any kind of user.
- Minimal learning curve

2.3.5 Security

- Safe authentication with Spring security in-built encryption and hashing.
- Customer, host and administrator role based access control.
- Security against typical vulnerabilities found in the Web (SQL injection, XSS, CSRF).

- Maintenance of secure classes of sessions with a time schedule that can be configured.
- Encryption security of sensitive user data.

2.3.6 Scalability

- Favors future addition of features.
- Serves more and more users.
- Software architecture Modular.

2.3.7 Maintainability

- Well documented, clean code according to Spring the Best, Java standards.
- Portable architecture that enables the easy upgrading of architecture and addition of features.
- Full logging of debugging and monitoring.
- Version-controlled schema migration to databases.

2.3.8 Compatibility

- Java 17/21 and Spring boot 3+ support.
- Compatibility with MySQL 8.0+
- We have cross-browser compatibility with the latest web browsers.
- Mobile-friendly across the screen.

2.4 Object-oriented System design using UML

2.4.1 Use Case Diagram

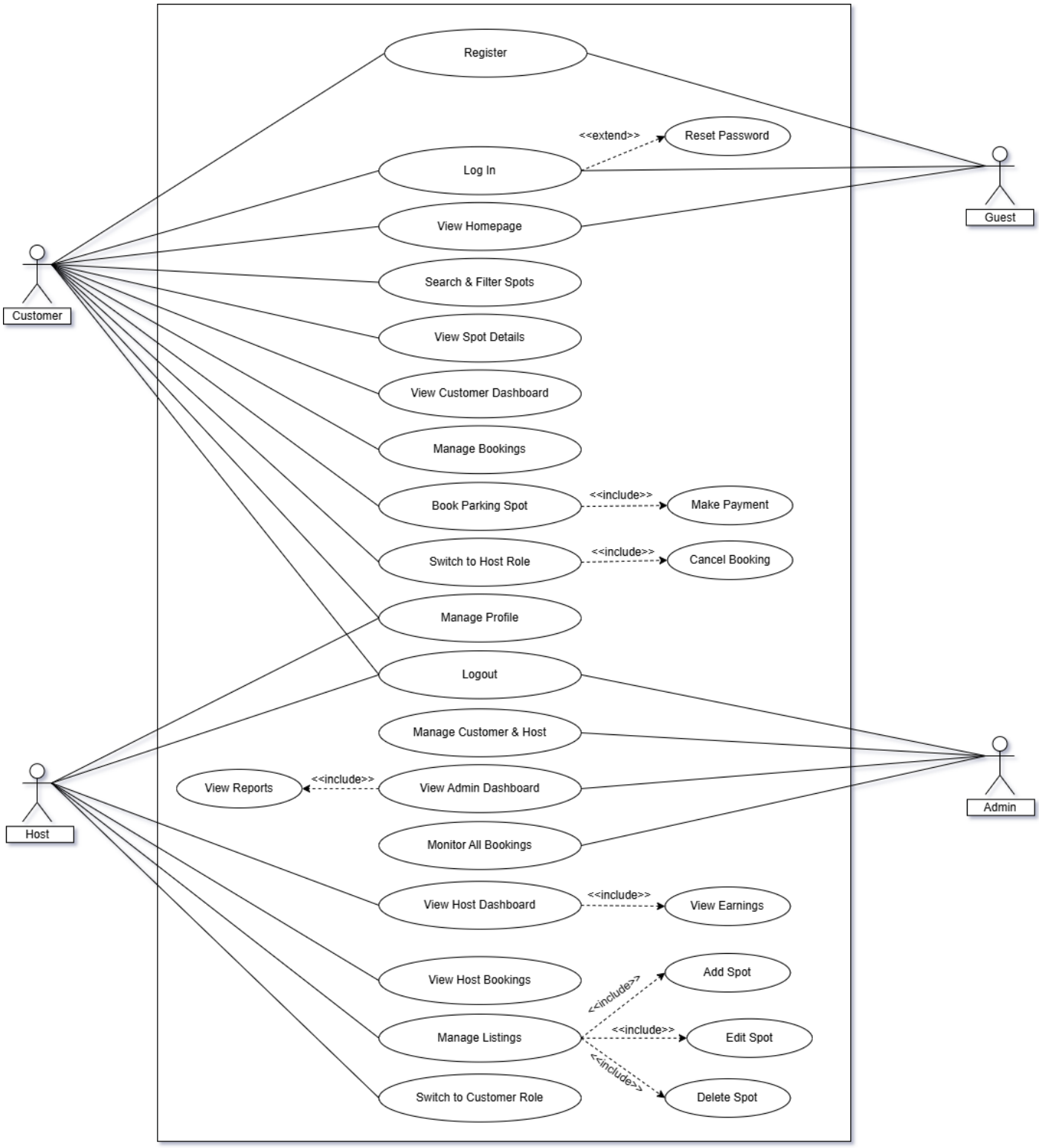


Figure 2: Use case Diagram

2.4.2 Case Description

Case Description-01: Registration

<table>
<tr><td>Use Case</td><td>Registration</td></tr>
<tr><td>Goal</td><td>Allow a new user to create an account and select a primary role (Customer or Host).</td></tr>
<tr><td>Precondition</td><td>The user is not logged in.</td></tr>
<tr><td>Success End Condition</td><td>A new user record is created in the database., and the user is automatically logged in.</td></tr>
<tr><td>Failed End Condition</td><td>Notification: "Registration failed. Email address already exists."</td></tr>
<tr><td>Primary Actors:
Secondary Actors:</td><td>Customer, Host
System</td></tr>
<tr><td>Trigger</td><td>The user navigates to the registration page and submits the form.</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>User navigates to the registration page</td></tr>
<tr><td>2.</td><td>User fills in required fields (Name, Email, Password, Primary Role).</td></tr>
<tr><td>3.</td><td>The user presses the "Register" button.</td></tr>
<tr><td>4.</td><td>System (Frontend) sends a POST request to the backend API (/auth/register)</td></tr>
<tr><td>5.</td><td>System (Backend) validates input, hashes the password, and checks for duplicate email.</td></tr>
<tr><td>6.</td><td>The system returns a success message and the authentication token (JWT). Frontend redirects the user to their new dashboard.</td></tr>
</table>
</td></tr>
<tr><td>Alternative Flows</td><td>
<table>
<tr><td>1.1</td><td>Email Exists: System returns a 409 Conflict error. Frontend displays the "Email address already exists" notification</td></tr>
<tr><td></td><td></td></tr>
<tr><td>1.2</td><td>Validation Error: User submits a weak password or invalid email format; System returns a 400 Bad Request error.</td></tr>
</table>
</td></tr>
<tr><td>Quality Requirements</td><td>Password hashing must be used. The user will fill up all the details in 30 minutes.</td></tr>
</table>

Case Description-02: User Login

<table>
<tr><td>Use Case</td><td>User Login</td></tr>
<tr><td>Goal</td><td>Allow a user (Customer, Host, or Admin) to gain authorized access to the system.</td></tr>
<tr><td>Precondition</td><td>The user must have a registered account (username/email and password).</td></tr>
<tr><td>Success End Condition</td><td>A valid JWT is generated and stored in the client, and the user is redirected to their respective Dashboard.</td></tr>
<tr><td>Failed End Condition</td><td>Notification: "Invalid credentials. Please try again."</td></tr>
<tr><td>Primary Actors:

Secondary Actors:</td><td>Customer, Host, Admin

System (Authentication Service)</td></tr>
<tr><td>Trigger</td><td>The user submits the login form.</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>User enters email & password.</td></tr>
<tr><td>2.</td><td>System validates credentials.</td></tr>
<tr><td>3.</td><td>The user is logged in successfully.</td></tr>
<tr><td>4.</td><td>System redirects to dashboard.</td></tr>
<tr><td>5.</td><td>N/A</td></tr>
<tr><td>6.</td><td>N/A</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>1.1</td><td>Password Reset: User clicks "Forgot Password," System guides them through the password reset process.</td></tr>
<tr><td></td><td>1.1.a. Try Again!!</td></tr>
<tr><td>1.2</td><td>Account Locked: If login fails repeatedly, the System temporarily locks the account.</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Authentication must use HTTPS. The JWT must be checked for validity and expiration on every protected API call.</td></tr>
</table>

Case Description-03: Add New Spot (Host)

<table>
<tr><td>Use Case</td><td>Add New Spot.</td></tr>
<tr><td>Goal</td><td>Allow an authorized Host to create and list a new parking spot available for booking.</td></tr>
<tr><td>Precondition</td><td>The user must be logged in as a Host.</td></tr>
<tr><td>Success End Condition</td><td>A new record is created in the Database.</td></tr>
<tr><td>Failed End Condition</td><td>Notification: "Listing creation failed." or "Validation failed."</td></tr>
<tr><td>Primary Actors:

Secondary Actors:</td><td>Host

System</td></tr>
<tr><td>Trigger</td><td>Host clicks " Add New Spot. "</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>The host clicks “Add Spot”.</td></tr>
<tr><td>2.</td><td>The system shows the form to enter details.</td></tr>
<tr><td>3.</td><td>Host uploads photo & fills details.</td></tr>
<tr><td>4.</td><td>The host clicks Save.</td></tr>
<tr><td>5.</td><td>The system stores data.</td></tr>
<tr><td>6.</td><td>The system confirms success.</td></tr>
</table>
</td></tr>
<tr><td>Alternative Flows</td><td>
<table>
<tr><td>1.1</td><td>Validation Error: Host fails to enter a required field</td></tr>
<tr><td></td><td></td></tr>
<tr><td>2.1</td><td>Image Upload Failure: Images fail to upload, but the listing data is saved; System returns a warning to the Host to retry image upload later.</td></tr>
</table>
</td></tr>
<tr><td>Quality Requirements</td><td>Host ownership verification must be performed by the Backend before saving the record. Spot location must be geocoded for accurate search results.</td></tr>
</table>

Case Description-04: Search & Filter Spots

Use Case	Search & Filter Spots

<table>
<tr><td>Goal</td><td>Customers can search and filter available spots that match their location and time criteria.</td></tr>
<tr><td>Precondition</td><td>None.</td></tr>
<tr><td>Success End Condition</td><td>The system returns a filtered, sorted list of spots and displayed on the Search Results page.</td></tr>
<tr><td>Failed End Condition</td><td>Notification: “No spots found.”</td></tr>
<tr><td>Primary Actors:

Secondary Actors:</td><td>Customer, Guest

System</td></tr>
<tr><td>Trigger</td><td>The user submits the search form on the homepage or search bar.</td></tr>
<tr><td>Description / Main

Success

Scenario</td><td><table>
<tr><td>1.</td><td>Customer enters location/date/filters.</td></tr>
<tr><td>2.</td><td>System fetches available spots.</td></tr>
<tr><td>3.</td><td>The system displays results.</td></tr>
<tr><td>4.</td><td>N/A</td></tr>
<tr><td>5.</td><td>N/A</td></tr>
<tr><td>6.</td><td>N/A</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>1.1</td><td>No matches found.</td></tr>
<tr><td></td><td></td></tr>
<tr><td>1.2</td><td>Server error retrieving data.</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Search response time must be fast (ideally < 500ms). Filtering logic must be accurate, especially for time-based availability</td></tr>
</table>

Case Description-05: Book a Spot

Use Case	Book a Spot
Goal	Customers should be able to book a spot.

<table>
<tr><td>Precondition</td><td>User logged in as Customer. Spot must be available.</td></tr>
<tr><td>Success End Condition</td><td>Booking is saved in DB. Confirmation shown.</td></tr>
<tr><td>Failed End Condition</td><td>Notification: “Submission Not Submitted”</td></tr>
<tr><td>Primary Actors:
Secondary Actors:</td><td>Customer
System</td></tr>
<tr><td>Trigger</td><td>The customer clicks Book Now.</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>The customer selects a spot.</td></tr>
<tr><td>2.</td><td>The system displays date/time selection.</td></tr>
<tr><td>3.</td><td>The customer chooses start & end time.</td></tr>
<tr><td>4.</td><td>The system calculates price.</td></tr>
<tr><td>5.</td><td>Customer confirms booking.</td></tr>
<tr><td>6.</td><td>System stores booking And System shows “Booking Confirmed”.</td></tr>
</table>
</td></tr>
<tr><td>Alternative Flows</td><td>
<table>
<tr><td>5.1</td><td>Time slot already taken , Booking rejected.</td></tr>
<tr><td></td><td>5.1.a. Show Error Message.</td></tr>
<tr><td>6.1</td><td>Payment failed</td></tr>
<tr><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>Quality Requirements</td><td>The user will fill up all the details in 5 minutes.</td></tr>
</table>

Case Description-06: Cancel Booking

Use Case	Cancel Booking
Goal	Let customers cancel their bookings.

<table>
<tr><td>Precondition</td><td>User logged in as Customer. Booking exists.</td></tr>
<tr><td>Success End Condition</td><td>Notification: Booking status updated to Cancelled.!!!</td></tr>
<tr><td>Failed End Condition</td><td>Notification: “Cancellation failed.”</td></tr>
<tr><td>Primary Actors:
Secondary Actors:</td><td>Customer
System</td></tr>
<tr><td>Trigger</td><td>Customer clicks Cancel on booking.</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>Customer opens My Bookings.</td></tr>
<tr><td>2.</td><td>The customer selects a booking.</td></tr>
<tr><td>3.</td><td>Customer clicks Cancel.</td></tr>
<tr><td>4.</td><td>System updates booking status.</td></tr>
<tr><td>5.</td><td>The system shows a success message.</td></tr>
<tr><td>6.</td><td>N/A</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>1.1</td><td>N/A</td></tr>
<tr><td></td><td></td></tr>
<tr><td>4.1</td><td>Booking already started, Cannot be cancelled.</td></tr>
<tr><td></td><td></td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Refund calculation logic must be precise and auditable. System must ensure that no further charges occur after cancellation.</td></tr>
</table>

Case Description-07: Delete Listing (Host)

Use Case	Delete Listing
Goal	Allow Hosts to delete their listings.

<table>
<tr><td>Precondition</td><td>The host logged in. Listing exists.</td></tr>
<tr><td>Success End Condition</td><td>Notification: Listing removed.!!!</td></tr>
<tr><td>Failed End Condition</td><td>Notification: “Delete failed.”</td></tr>
<tr><td>Primary Actors:
Secondary Actors:</td><td>Host
System</td></tr>
<tr><td>Trigger</td><td>Host clicks Delete on a listing.</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>The host opens My Listings.</td></tr>
<tr><td>2.</td><td>Host clicks Delete.</td></tr>
<tr><td>3.</td><td>The host confirms.</td></tr>
<tr><td>4.</td><td>The system deletes records.</td></tr>
<tr><td>5.</td><td>The system shows success.</td></tr>
<tr><td>6.</td><td>N/A</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>3.1</td><td>Listing has active bookings, deletion restricted.</td></tr>
<tr><td></td><td></td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Deletion should be a soft delete (status update) for auditability.</td></tr>
</table>

2.4.3.1 Activity Diagram

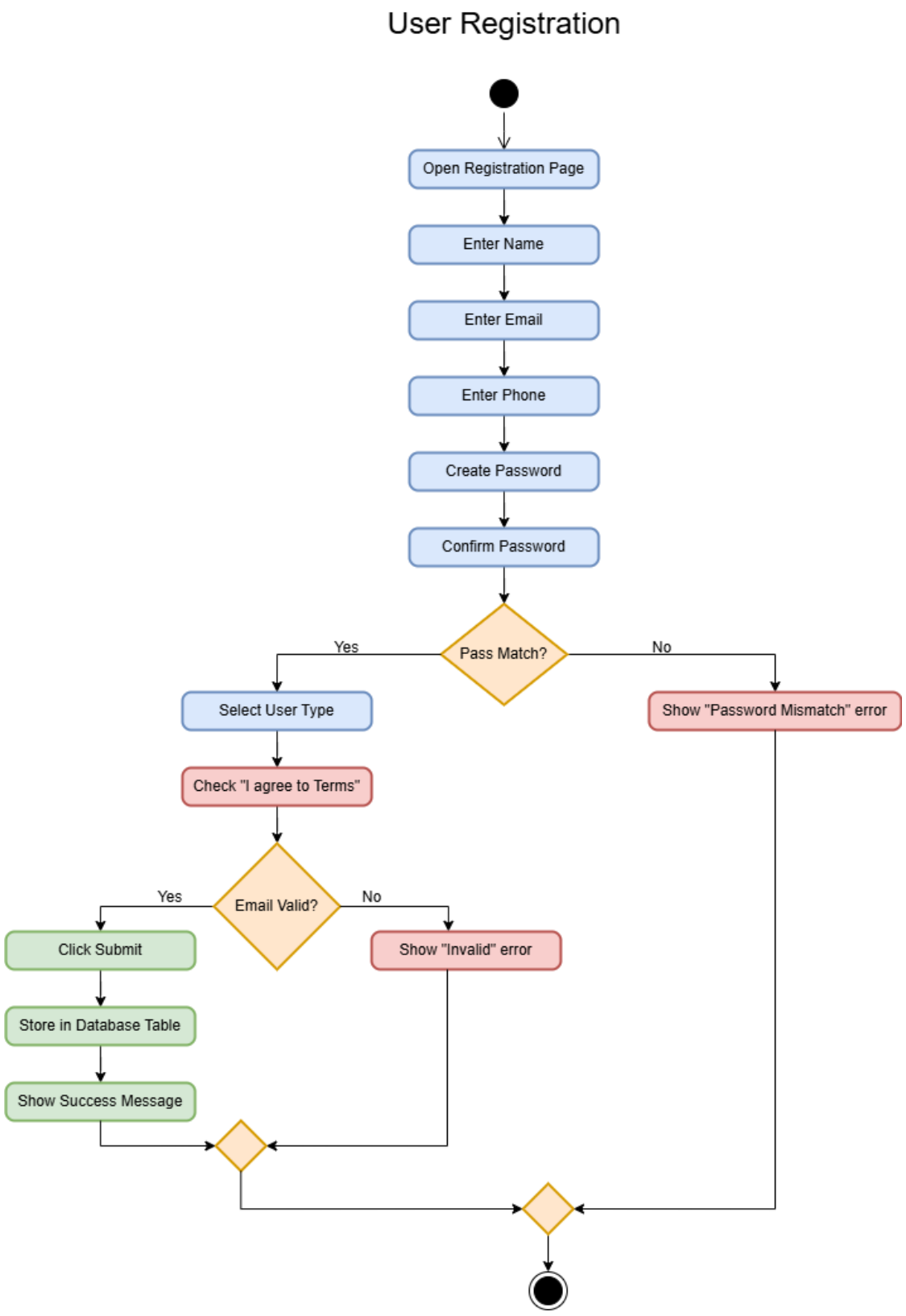


Figure 3.1 : User Registration

2.4.3.2 Activity Diagram

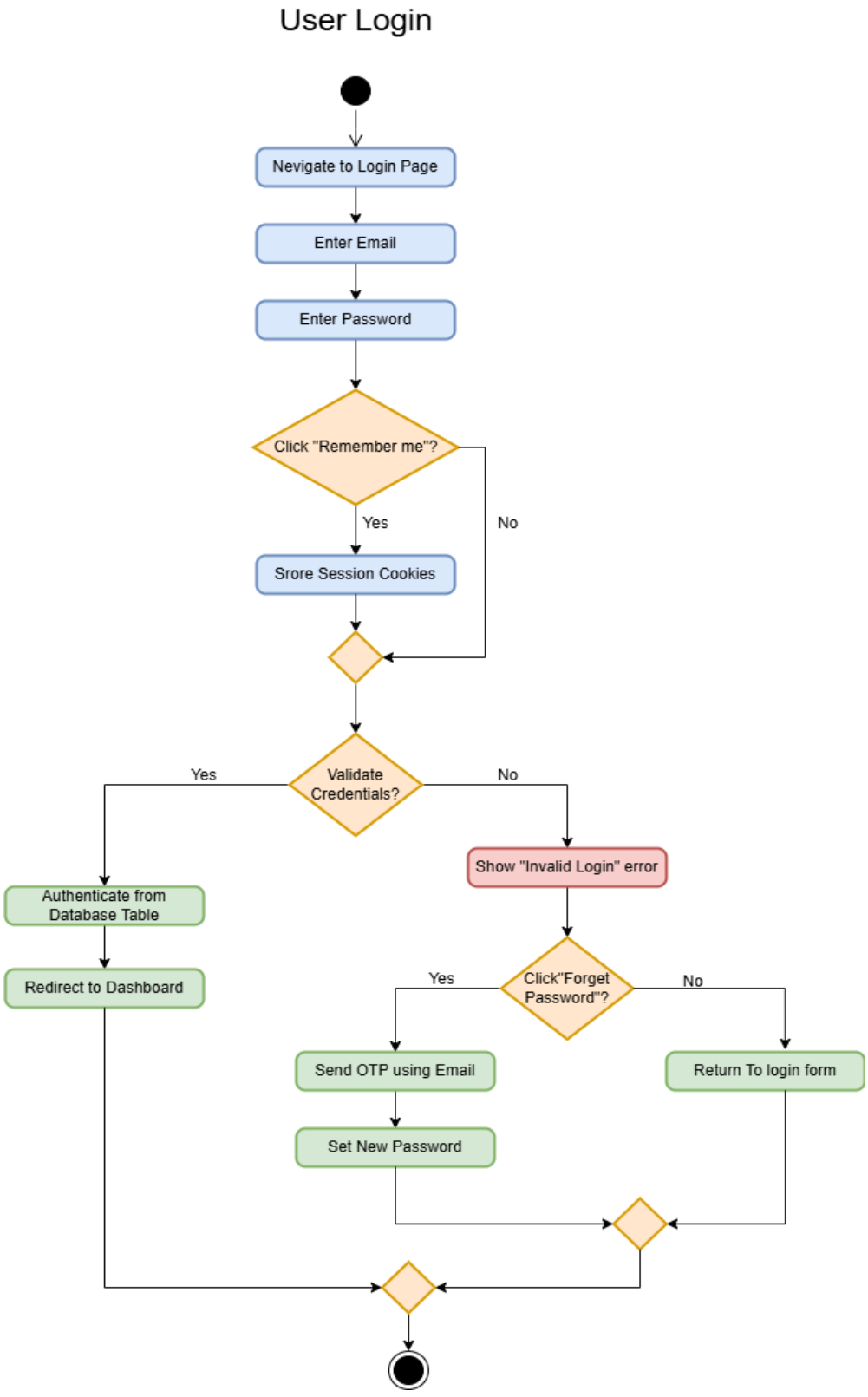


Figure 3.2 : Log In

2.4.3.3 Activity Diagram

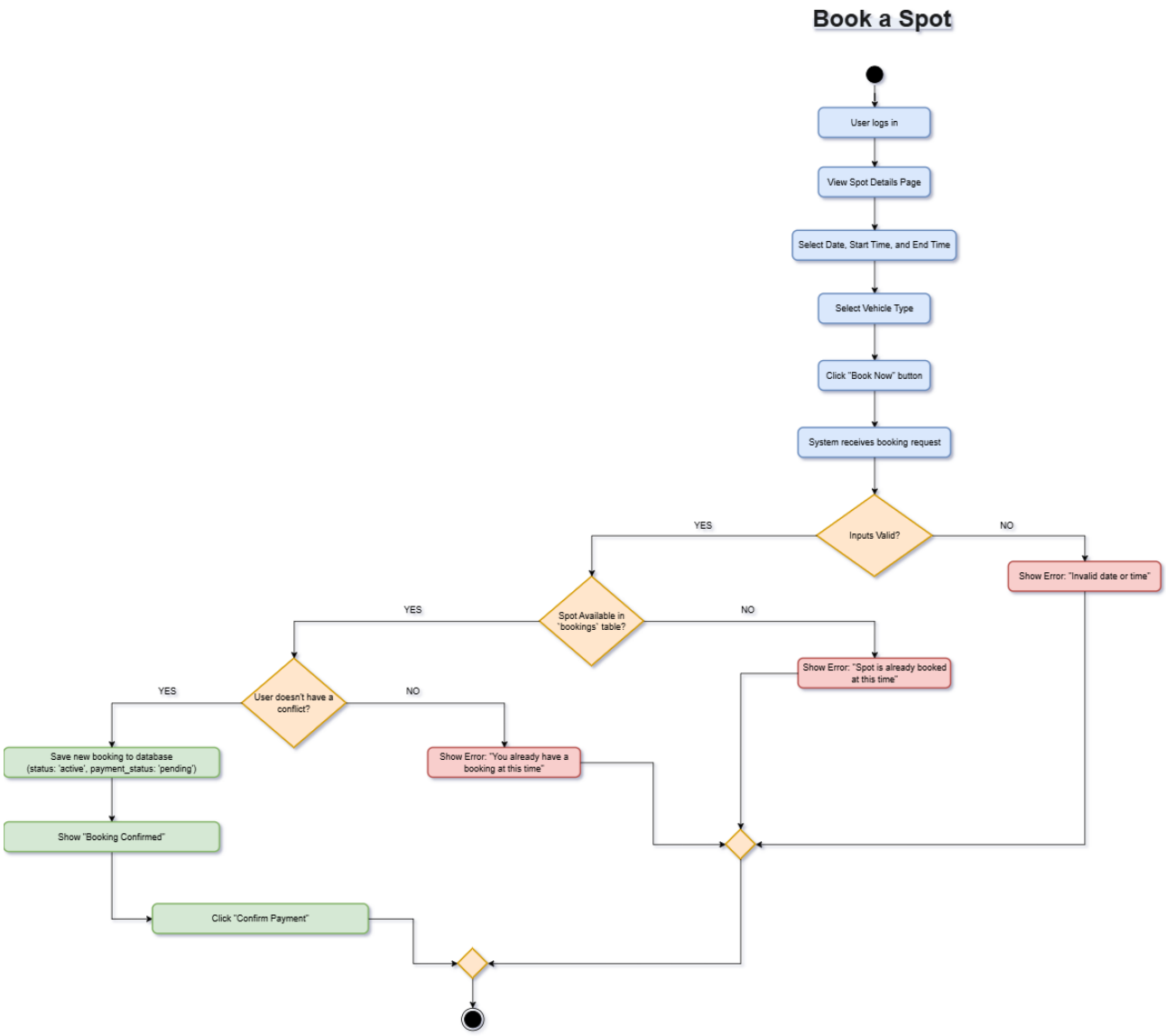


Figure 3.3 : Book a Spot

2.4.3.4 Activity Diagram

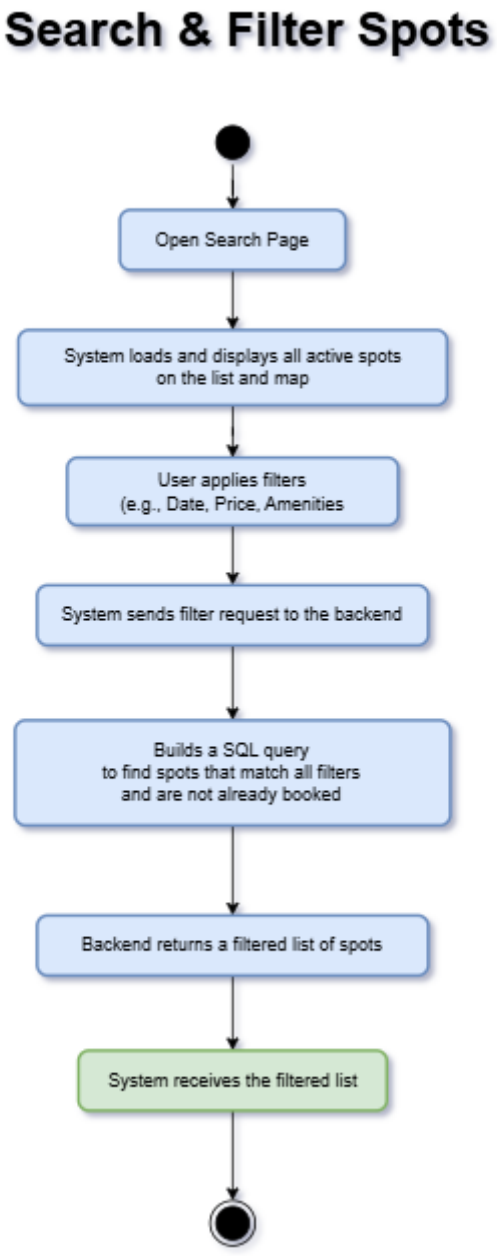


Figure 3.4: Search and filter Spots

2.4.3.5 Activity Diagram

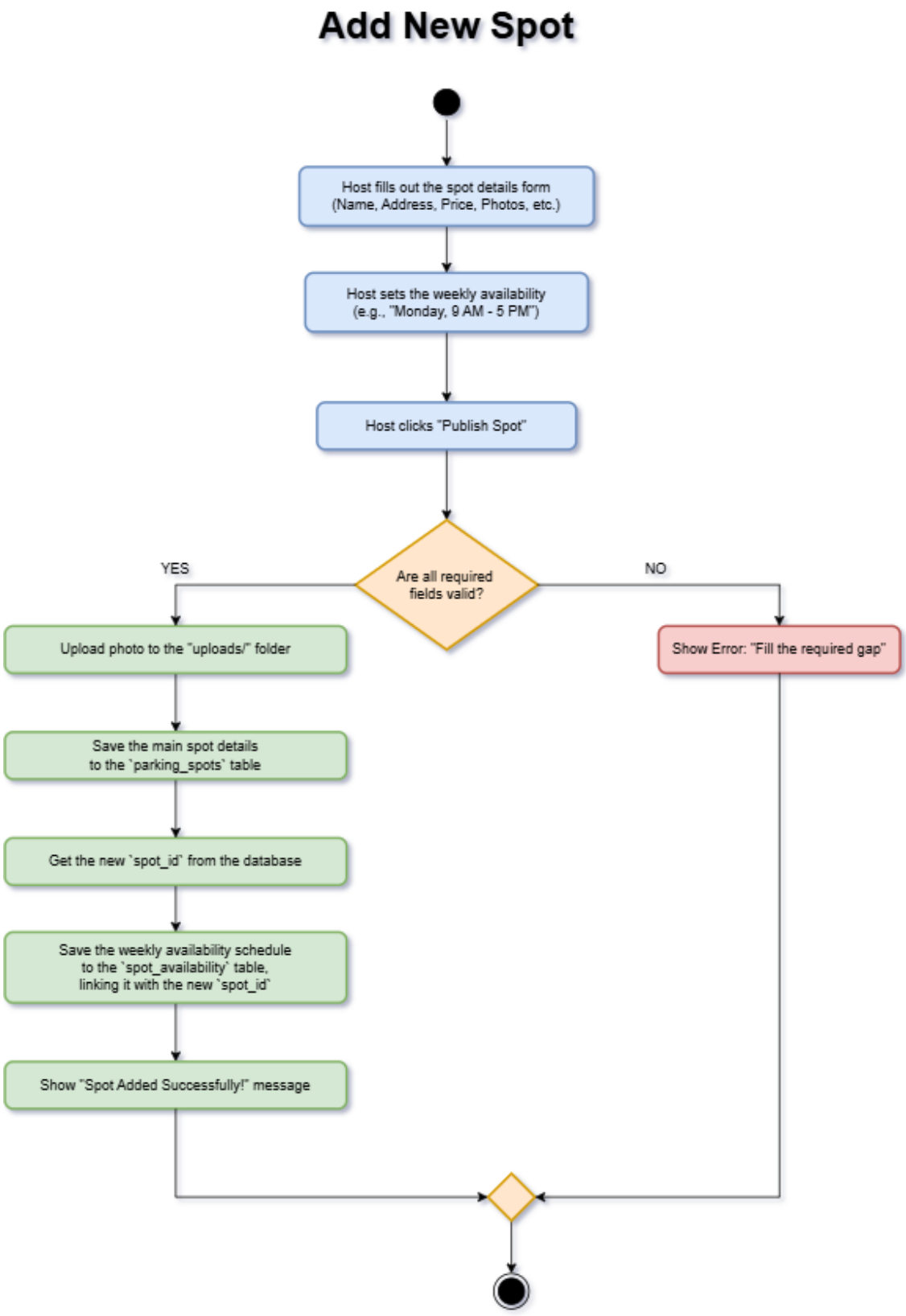


Figure 3.5: Add a slot

2.4.3.6 Activity Diagram

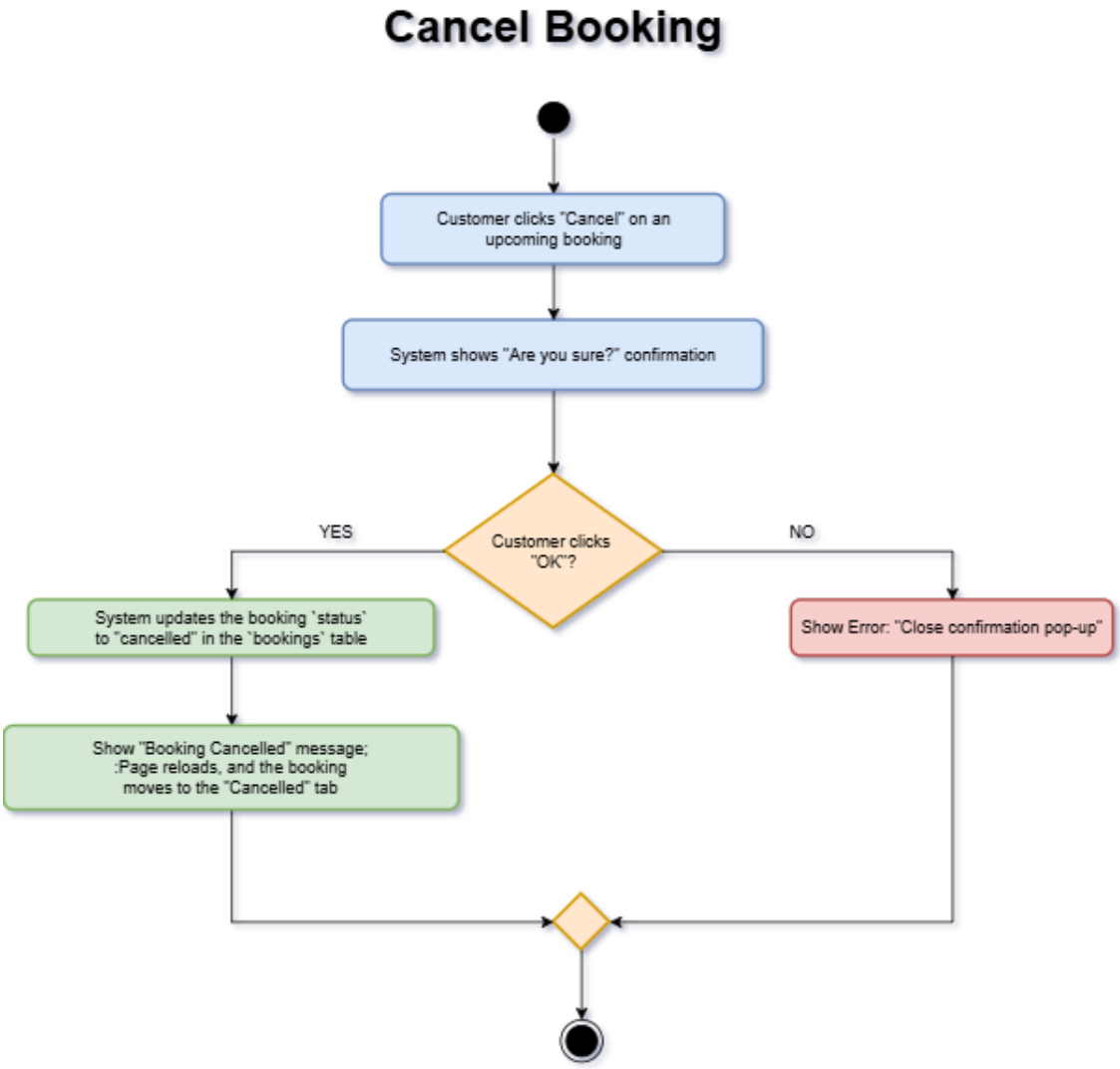


Figure 3.6: Cancel Booking

2.4.3.7 Activity Diagram

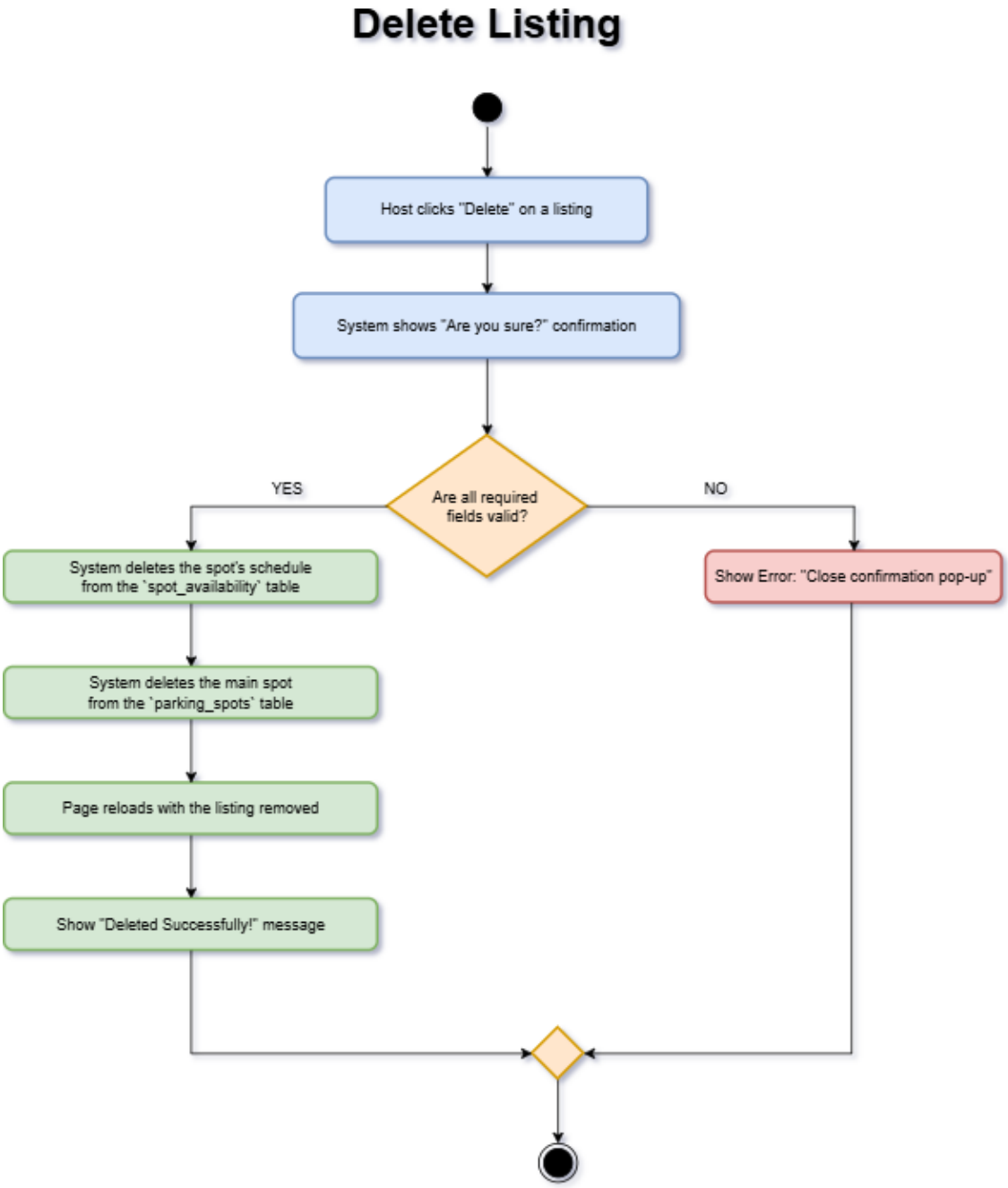


Figure.3.7: Delete Listing

2.4.4 Sequence Diagram

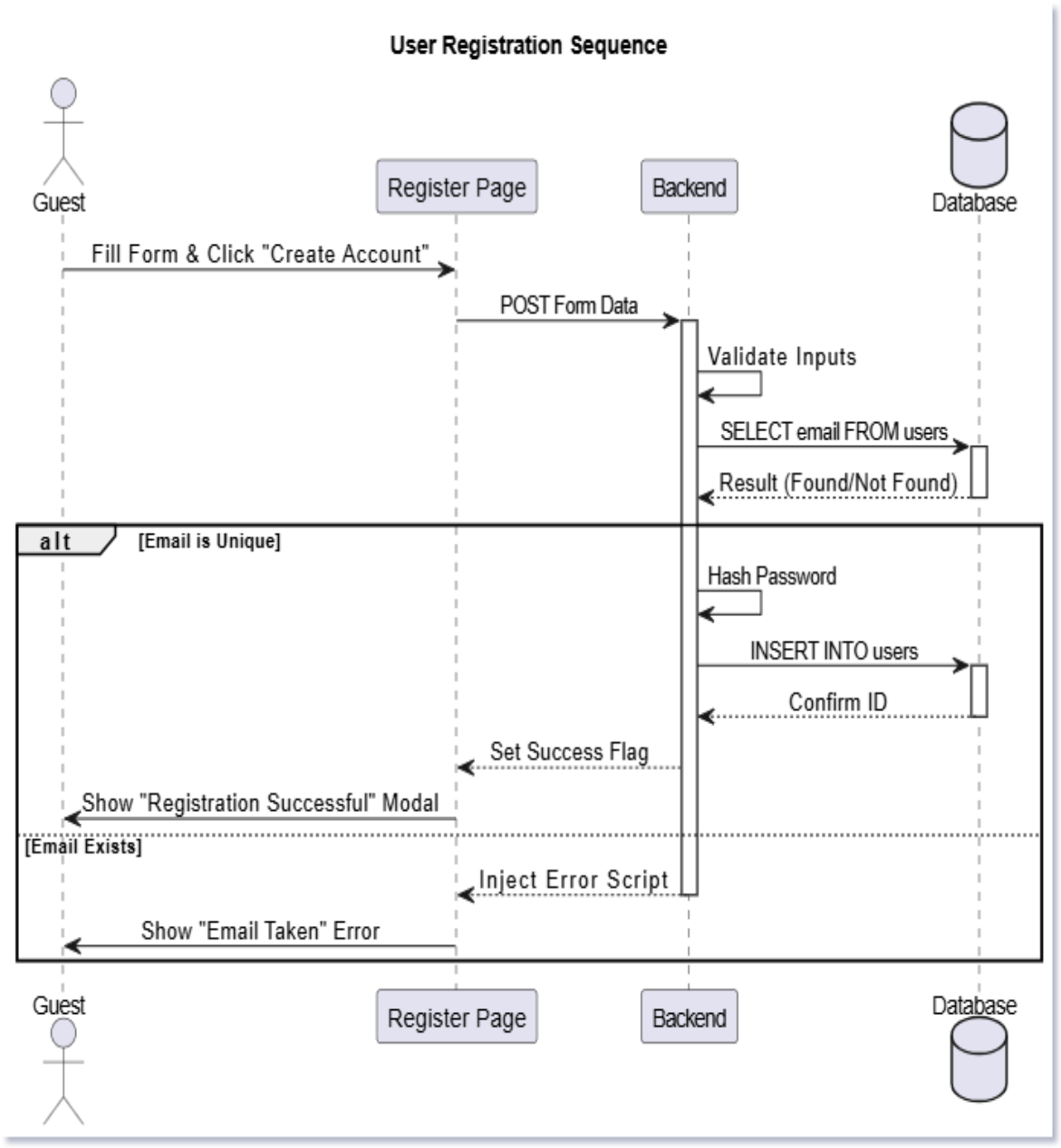


Figure 4.1: User Registration

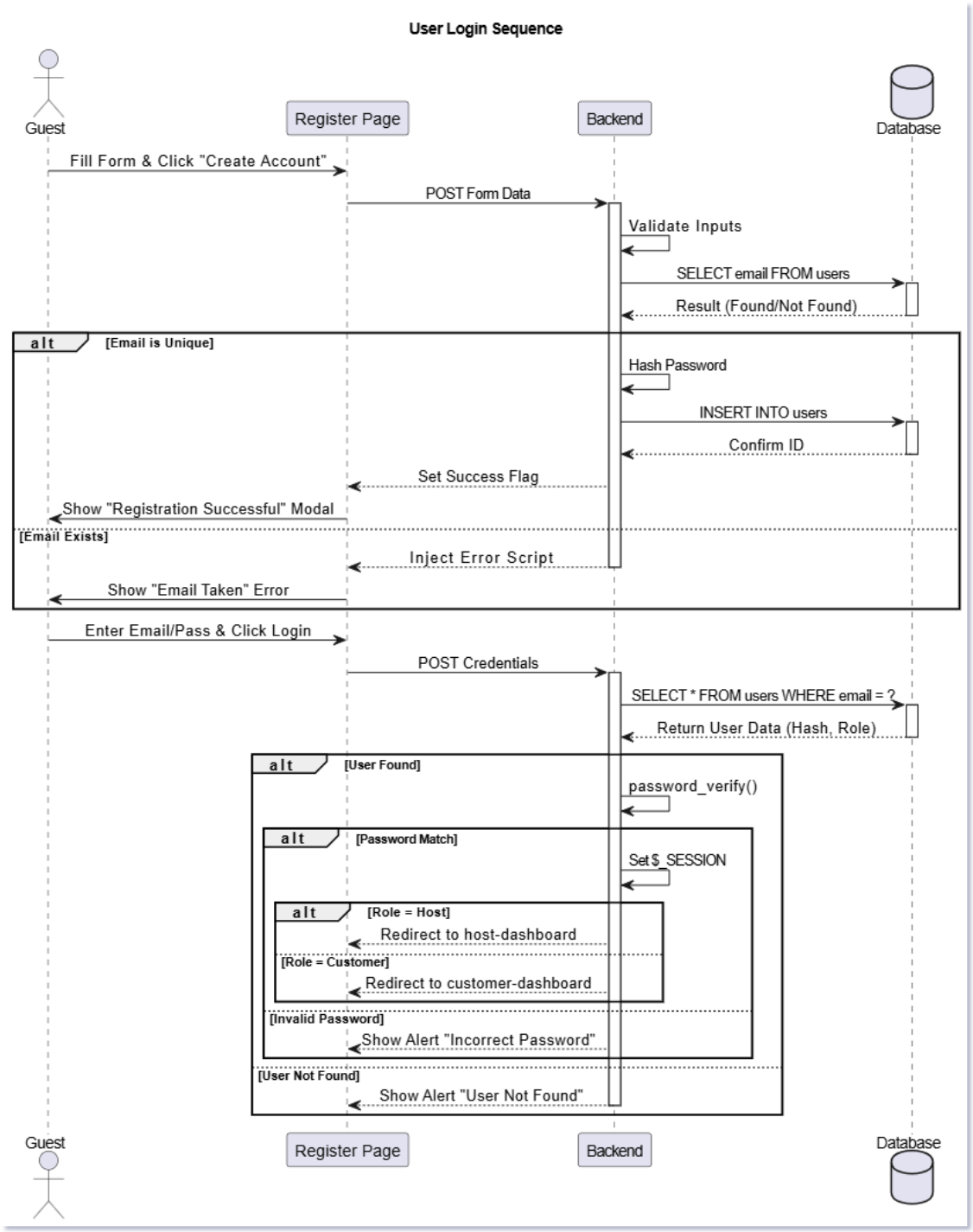


Figure 4.2: Login

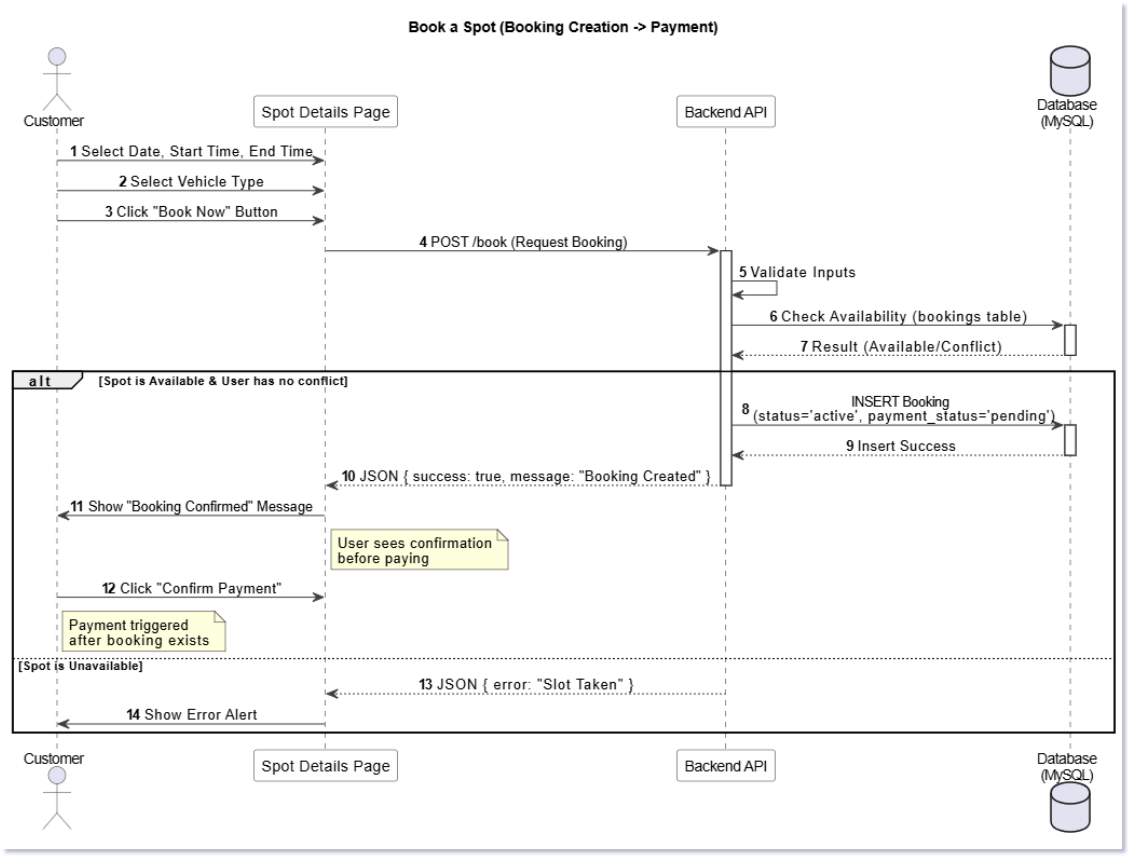


Figure 4.3: Book a Slot

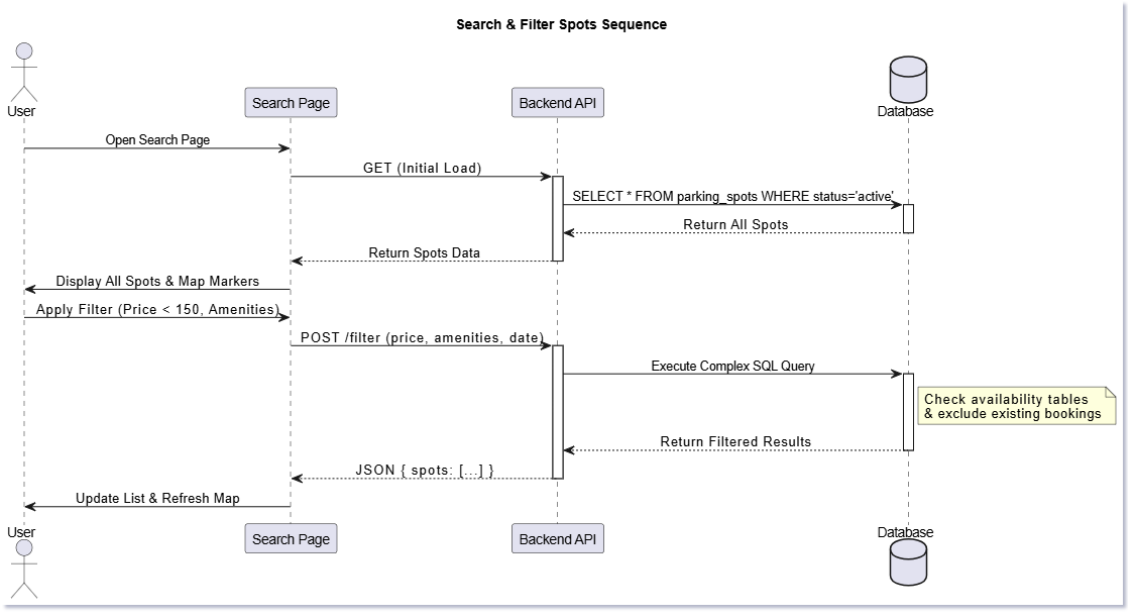


Figure 4.4: Search and Filter Spot

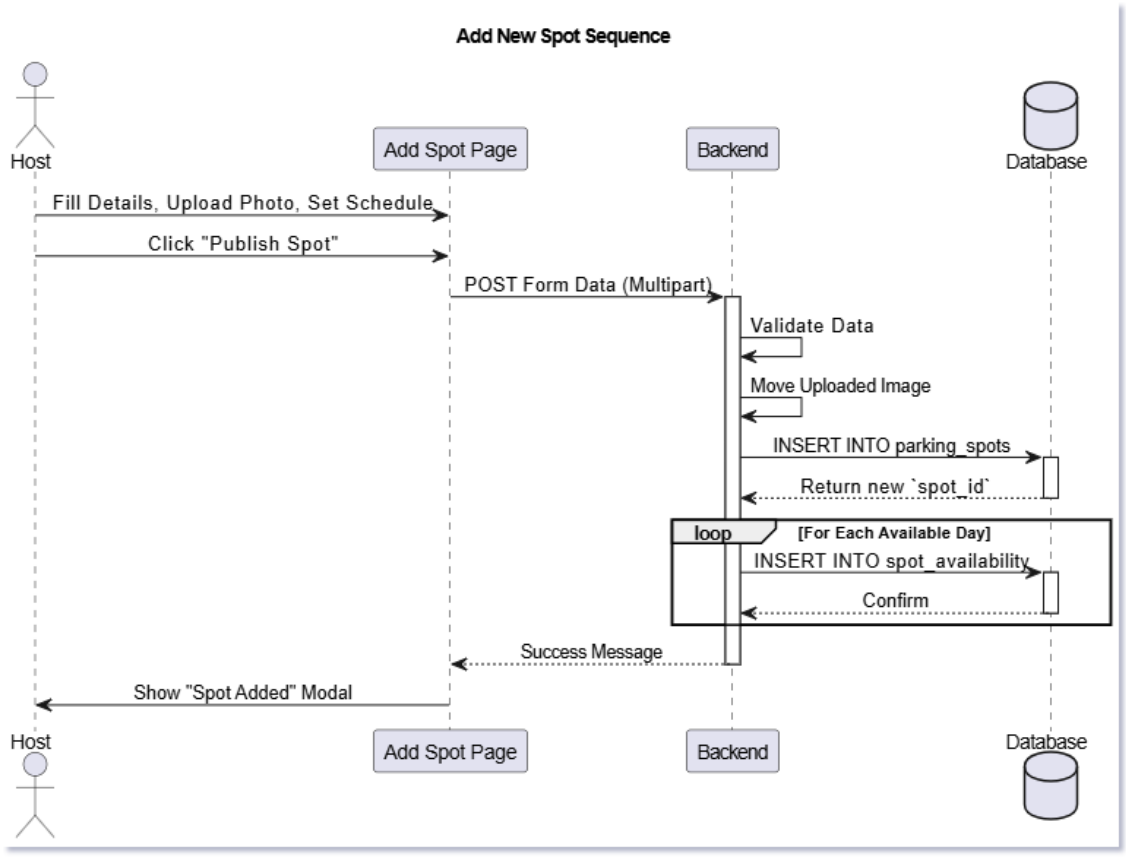


Figure 4.5: Add New Spot

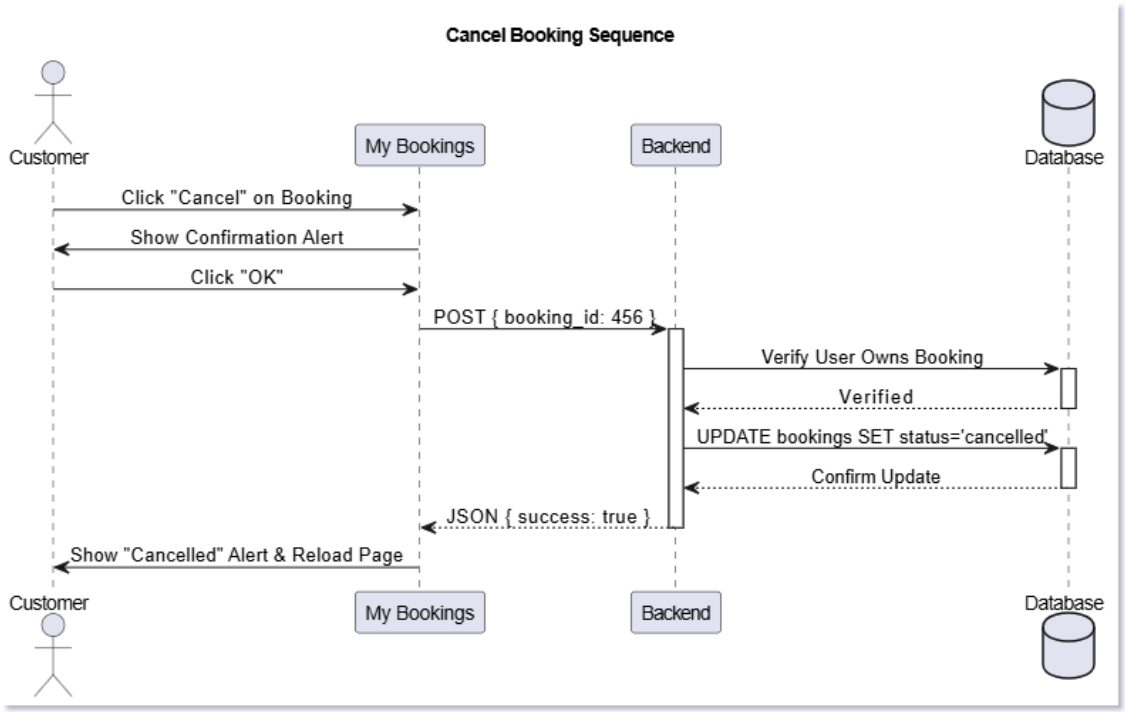


Figure 4.6: Cancel Booking

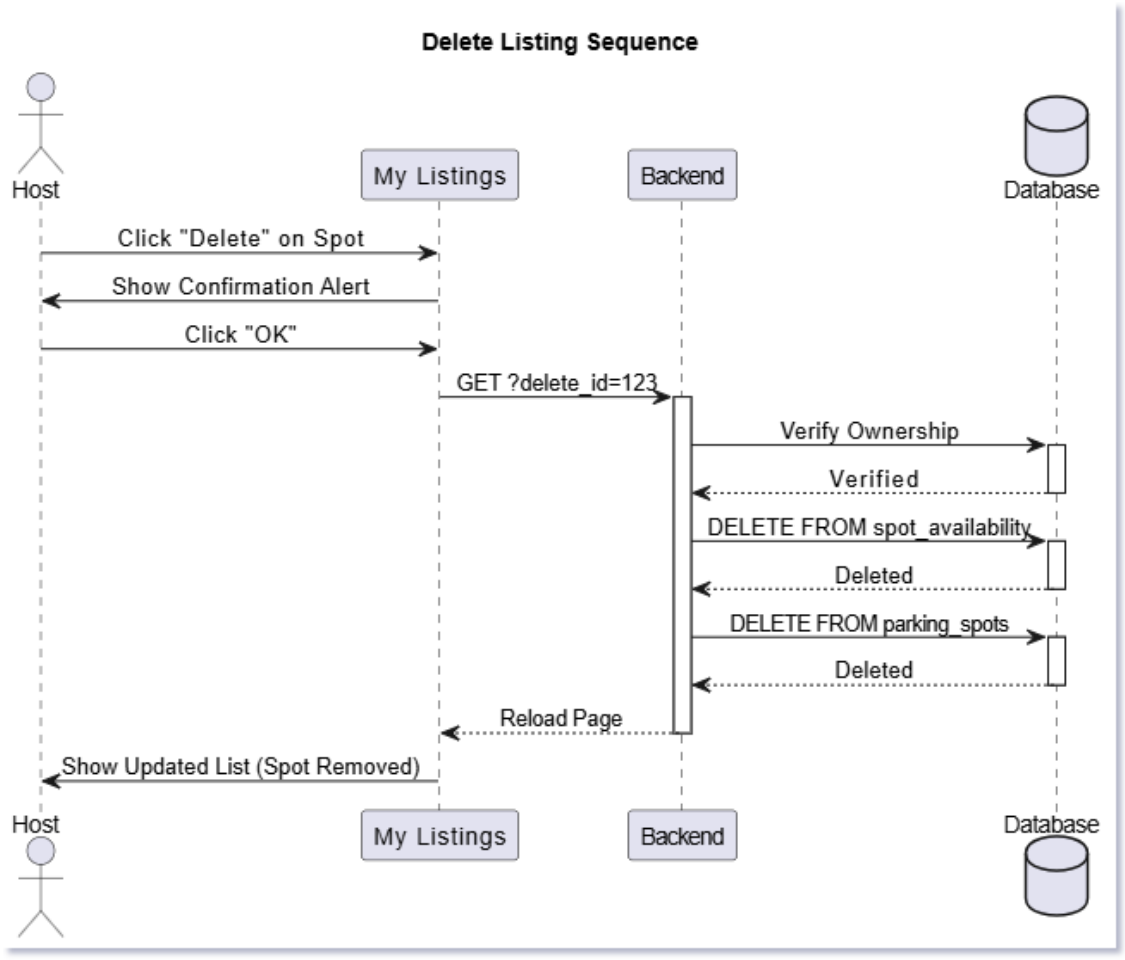


Figure 4.7: Delete Listing

2.4.5 Class Diagram

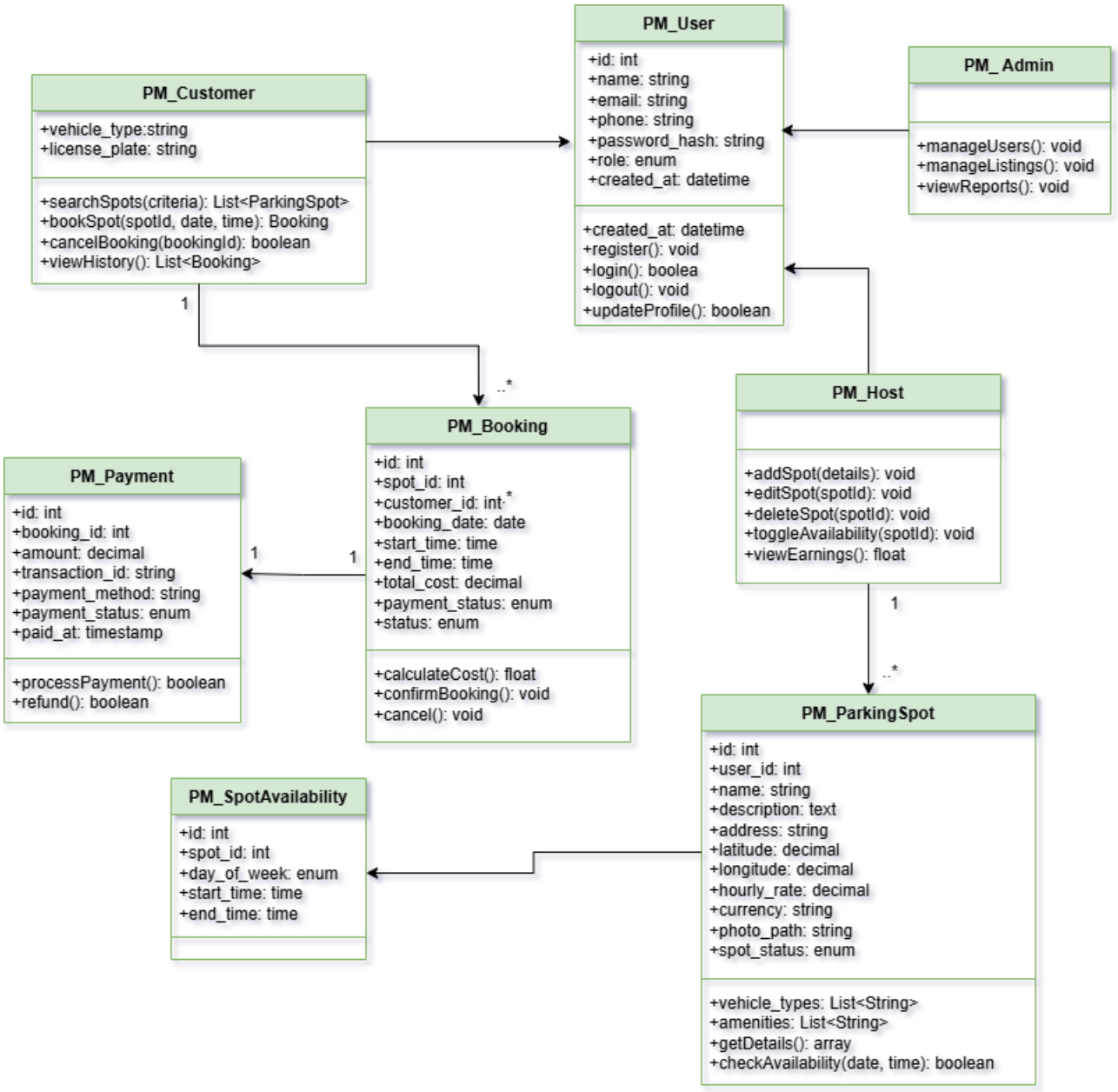


Figure 5: Class Diagram

2.4.6 ER Diagram

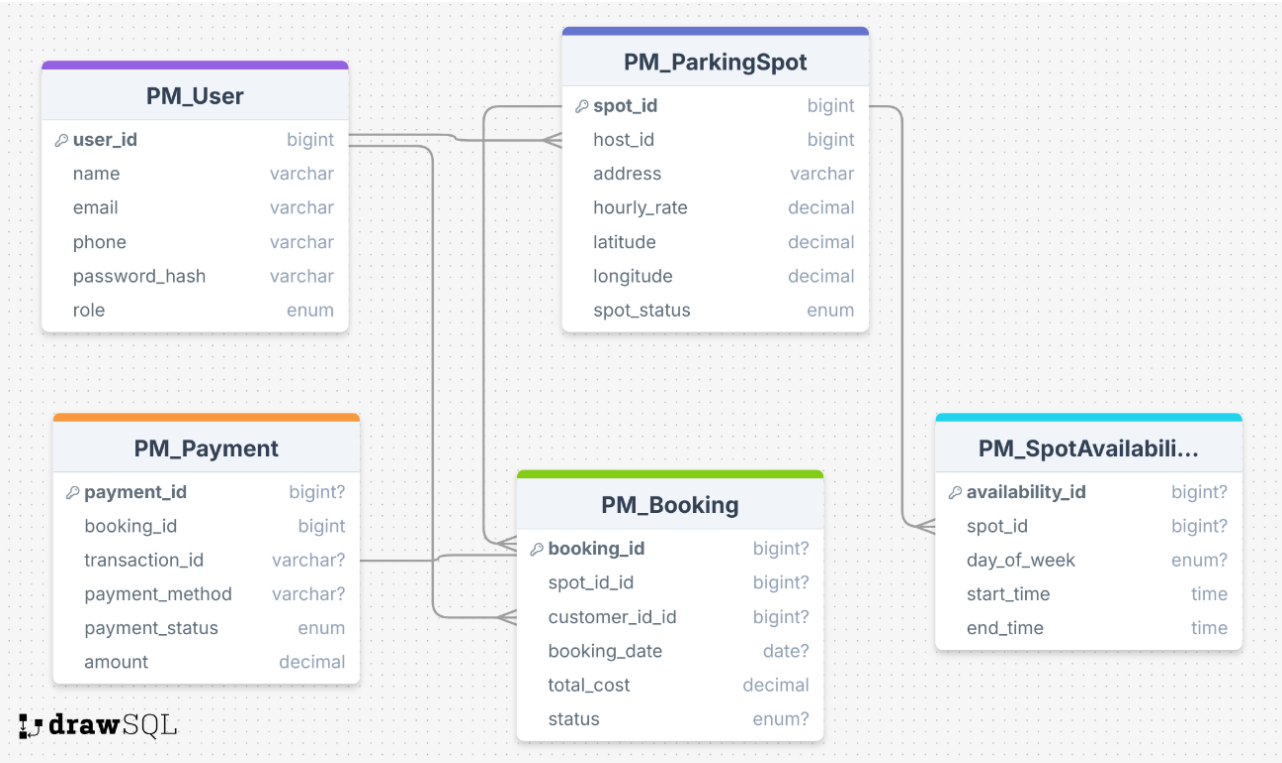
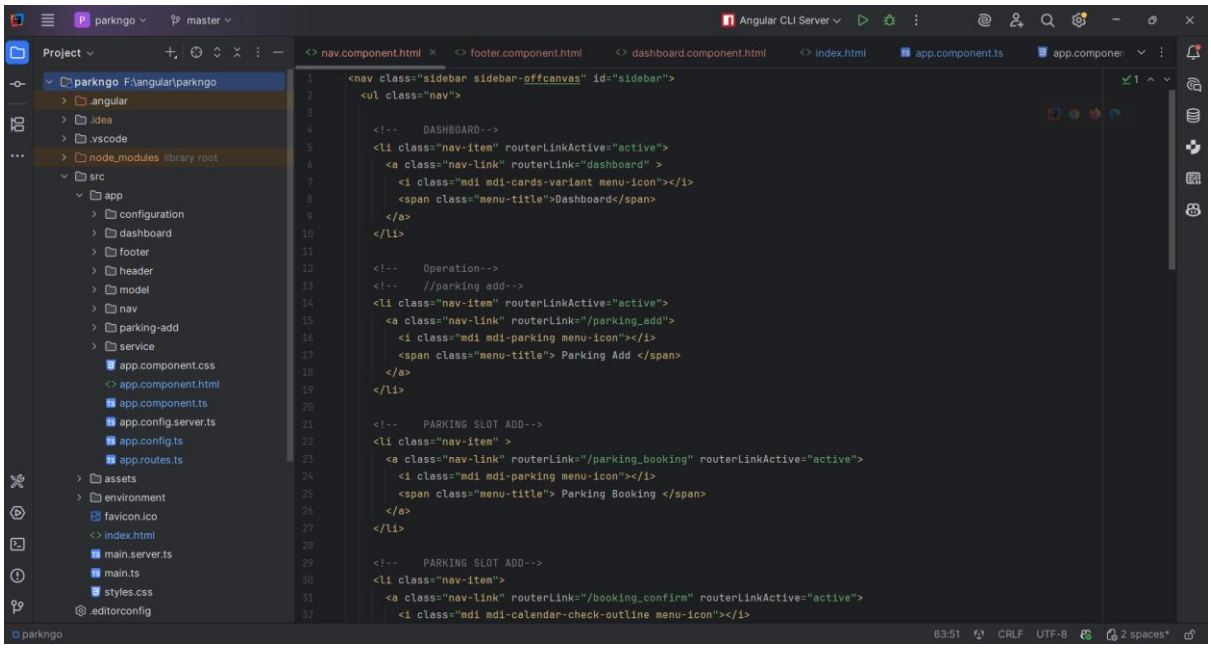
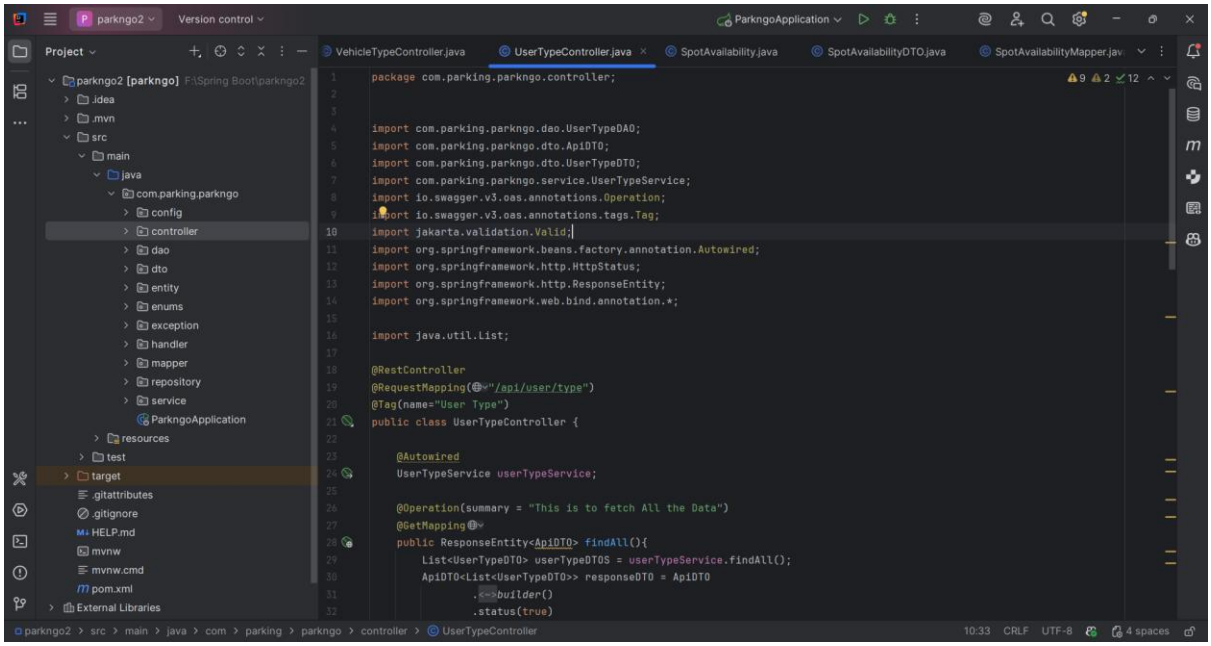


Figure 6: ER Diagram

2.5 Coding: Appendix A





2.6 Summary

In this chapter, the design and implementation of the proposed parking/spot booking management system were provided. It talked about the system architecture, functional and non-functional requirements as well as the design decisions made in order to make the system perform, be reliable and usable. The implementation strategy of the main modules including authentication, spot listing, booking management, and user dashboards were also described. This chapter as a whole illustrates how the system requirements were implemented in an effective and efficient web-based system which will be used in testing and evaluation in the follow-up chapter.

Chapter 3 Software Testing

3.1 Introduction

The development of the ParKNGo Parking Management System involves software testing as a key stage of the development life. It maintains that each feature such as user authentication to spot booking works properly, it is reliable and secure enough until the system is released to the actual users.

The main objective of the testing in this project is to ensure that the basic operations that include listing of parking spots, searching, checking real time availability, booking, cancellation, host earnings management as well as administrator controls are behaving as per the requirements. It is also tested that the users such as Customers, Hosts and Admin have a smooth error free and intuitive workflow throughout the application.

3.2 Testing Features

3.2.1 Feature to Be Tested

A. User Registration
B. User Login
C. Spot Management
D. Search & Filter System
E. Booking Management
F. Payment Processing
G. Host Earnings & Dashboard
H. Admin Panel
I. Frontend UI/UX Functionality
J. API Functionality (Backend)
K. Performance & Load Handling
L. Security Feature

3.3 Testing Strategies

3.3.1 Test Approach

The proposed parking/spot booking management system has a testing strategy that is based on a multi-layered strategy in order to validate the system entirely.

Individual components of the system that were to be unit tested were authentication logic, booking functions, and database operations to test the proper functionality in isolation. This assisted in detecting faults in logic early in the process of development.

It was then tested through an integration test to understand how the various modules communicated with each other where user registration, spot listing, booking management and dashboard updates were included. This step provided a smooth flow of data and proper communication between frontend and backend and the database.

After integrating testing, system testing was conducted on application that was wholly integrated. The system was tested in terms of functional and non-functional criteria, such as response time, usability, and simple security tests.

Lastly, the User Acceptance Testing (UAT) was done by the chosen users to confirm the real-life scenarios, and ensure that the system is prepared to be implemented and satisfies the expectations of the user.

3.3.2 Pass/Fail Criteria

A test case is passed when the result of the actual system is according to the expected outcome as stipulated in the test scenario. All the functional operations should run properly and the performance standards like reasonable response time should not be ignored.

A test case is considered to have failed when the real result is different with the expected result, the system generates the wrong output, runtime error or fails to perform as per the expected performance. Any failures are recorded, examined, given priority ranked and sorted out and before moving to next testing or going to the final deployment, the failed cases are sorted out.

3.4 System Testing (Test Cases with Report)

Test Case ID	Module	Test Description	Test Steps	Expected Result	Demo Data	Pass/ Fail
TC01	Authentication	Verify user registration with valid data	1. Open Register Page 2. Enter valid name, email,	Account created & redirected to login	Name: John Doe Email: john@test.com Pass: Test@123	Pass

			password 3. Click “Register”			
TC02	Authentication	Verify login with valid credentials	1. Open Login Page 2. Enter valid email & password 3. Click “Login”	Redirected to dashboard	Email: john@test.com Pass: Test@123	Pass
TC03	Customer – Search	Search for available parking spots	1. Login as Customer 2. Go to Search Parking 3. Enter location 4. Click Search	Matching parking spots displayed	Location: Banani Date: 12 Feb 2025	Pass
TC04	Customer – Booking	Book a parking spot successfully	1. Search a spot 2. Select time slot 3. Click Book Now 4. Confirm	Booking visible in Upcoming Bookings	Spot: Banani 11 Time: 2pm–4pm	Pass
TC05	Customer – Cancel Booking	Cancel an upcoming booking	1. Login 2. Open Upcoming Bookings 3. Click Cancel	Booking moves to Cancelled list	Booking ID: BK1022	Pass
TC06	Host – Add Spot	Host adds a new parking spot	1. Login as Host 2. Click Add Spot 3. Enter details 4. Submit	Spot added & shown in My Spots	Spot Name: Dhanmondi 7 Parking Rate: 50/hr	Pass
TC07	Host – View Earnings	Host checks earnings dashboard	1. Login as Host 2. Navigate to Earnings Dashboard	Correct totals displayed	Total Earnings: 12,550 BDT	Pass
TC08	System Security	Verify JWT secured API access	1. Call API without token 2. Call with invalid token	Unauthorized (401)	Endpoint: /api/booking Token: Invalid	Pass
TC09	UI/UX – Responsive Layout	Verify UI responsiveness	1. Open on mobile/tablet 2. Navigate pages 3. Check alignment	Layout responds cleanly	Device: iPhone 12 Browser: Chrome	Pass

3.5 Summary

The chapter talked about the ParKNGo Smart Parking System testing and evaluation to provide the functionality, reliability, and usability of the system. Different testing strategies such as the user acceptance testing were added to check both the system behavior and the internal logic. Precise pass/fail criteria were established to determine the performance of the system, its security and the accuracy of the data. In general, it is possible to state that this chapter proves that the system is in compliance with the requirements stated and is prepared to be deployed with the lowest risk.

Chapter: 4 Deployment and Maintenance

4.1 Introduction

The chapter is dedicated to the deployment and maintenance of the Smart Parking System of ParKNGo. It describes the process of getting ready the developed system to release it, deploy it to the target environment and maintain it once it is deployed. The chapter also talks about the steps that are undertaken during the maintenance of system stability, performance and long term usability. This chapter provides a systematic method of system release, monitoring, and continuous maintenance as the next stage of Software Release Life Cycle (SRLC).

4.2 Try to follow the SRLC (software release life cycle)

The implementation of the ParKNGo - Smart Parking System is based on the organized steps of the Software Release Life Cycle (SRLC) so that the system is provided in a stable condition, secure, and full of functionalities. Every SRLC stage will help to prepare the application to be used in real-world by Customers, Hosts, and Admin.

4.2.1 Pre-Alpha Phase

During this stage, all the functional and non-functional requirements of the stakeholders like drivers, owners of parking spots, and system administrators are gathered. The workflow diagrams, database design and system architecture are developed. There is no operational mechanism.

4.2.2 Alpha Phase

In the Alpha stage, the developers develop the key features, which include:

- User registration & authentication
- Search for available parking
- Spot booking and cancellation
- Host spot management
- Admin approval and monitoring
- Payment simulation

Initial integration testing and unit testing is carried out. It is still in an unstable state and is in use by the development team alone.

4.2.3 Beta Phase

Beta stage involves complete integration and testing of the system by a restricted external audience (test customers, hosts). Major tasks include:

- Fixing bugs from Alpha testing
- Enhancing user experience and UI.
- Load testing and security testing.
- APIs preparation and deployment configuration.

The feedback on users helps to improve features, including the quality of search, booking process, and dashboard efficacy.

4.2.4 Release Candidate (RC)

Once the key modules were stabilized, the Release Candidate was initiated as per the suggestions that were given by the beta users. The Release candidate was the one that was near the final version, and it had the best workflow, enhanced UI, latest security settings, and the best role-based access control. The Release Candidate was still reviewed by supervisors to be approved.

4.2.5 Production Release (Stable Version)

The production server is used to deploy the stable version of ParKNGo. Deployment includes:

- Publishing the build on the hosting service.
- Establishing the data base of production.
- Setting up of environment variables.
- Enabling SSL/HTTPS
- Versioning the release in Git

Customer, Host, and Admin have total access to the features of the system.

4.2.6 Post-Release Maintenance and Updates

Once the system has gone-live the system is under continuous maintenance:

- Resolving bugs that were logged by actual users.
- Improving functions and interface on feedback.
- Checking server performance and availability.
- Tracking and processing user behavior.
- Conducting frequent database backup.
- Installation of security patches and updates of security frameworks.

It is aimed at maintaining the system stable, fast, and secure with the increase in usage.

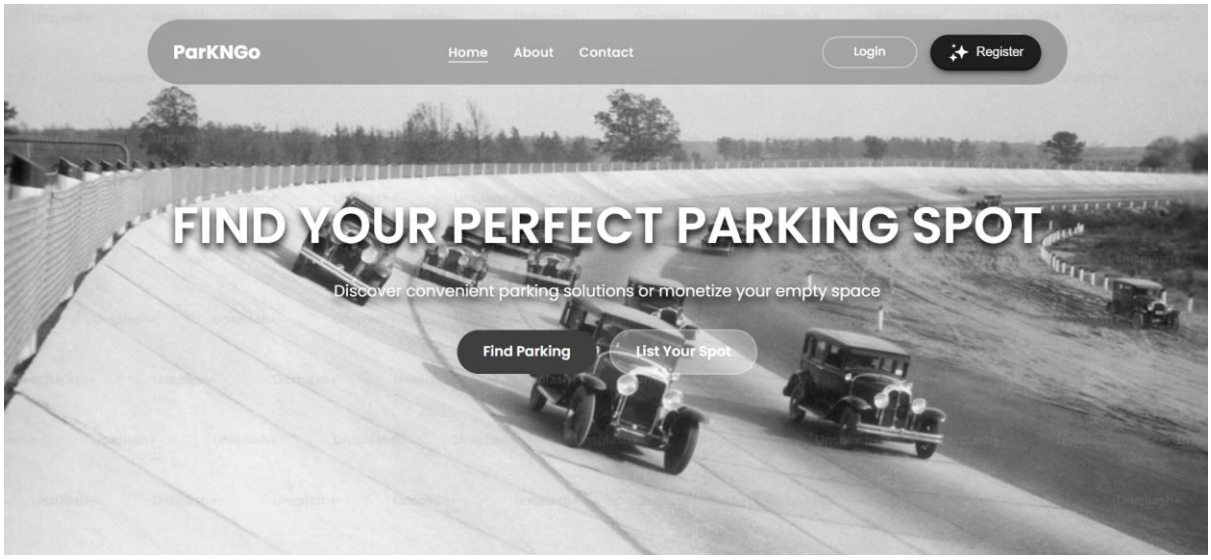
Chapter: 5 User Manual

5.1 Introduction

ParkNGo - Smart Parking System has a user manual that shows a clear and easy way of how the platform can be used by all users effectively. It outlines the possibility of the customers to find parking spots in the vicinity and reserve or cancel them, as well as manage their profiles. Hosts will get to know how to add new parking spots, provide updates on spots and see income. Administrators are instructed on the way to approve listings, keep track of system activities, and control users. The manual will ensure that all users have an easy time navigating the system as it will contain step-by-step guidelines, descriptions of features, and simple troubleshooting. All in all, this guide can make the process of using the ParKNGo system easy and convenient to any person using it.

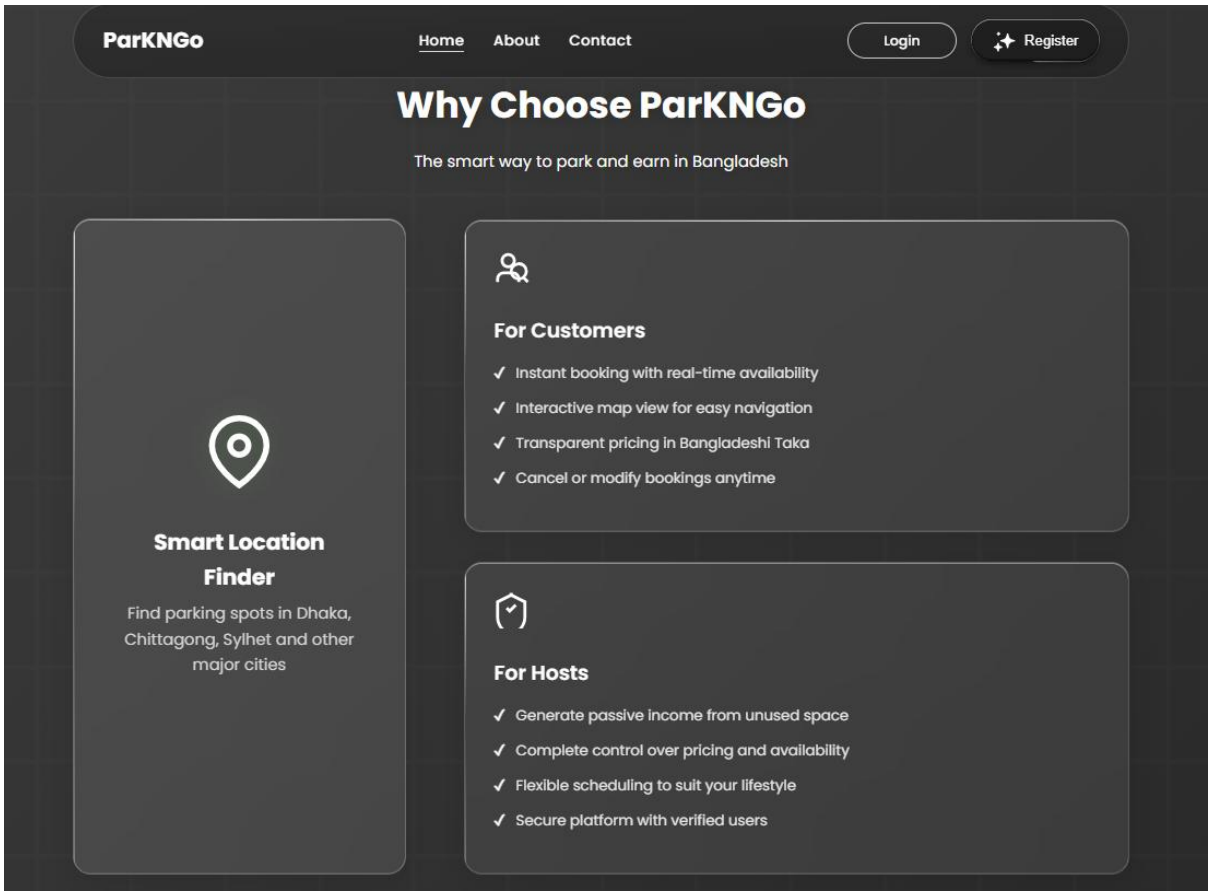
5.2 Project Functionalities

Homepage



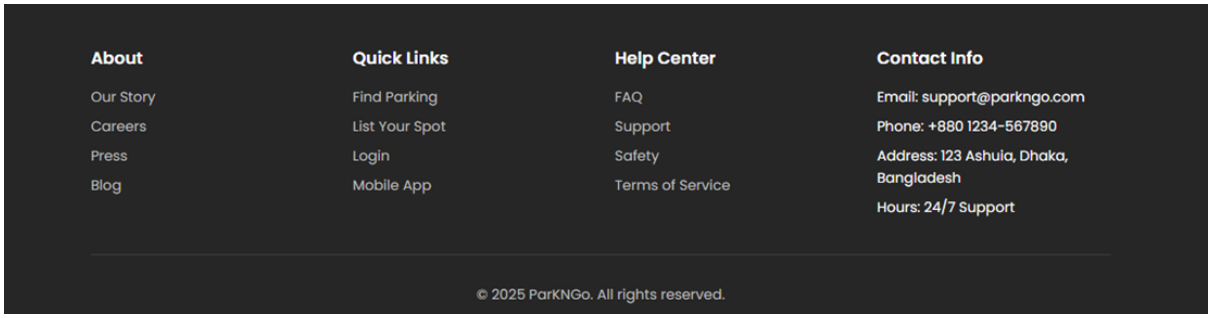
Appendix 5.2.1: **Homepage**

About Us



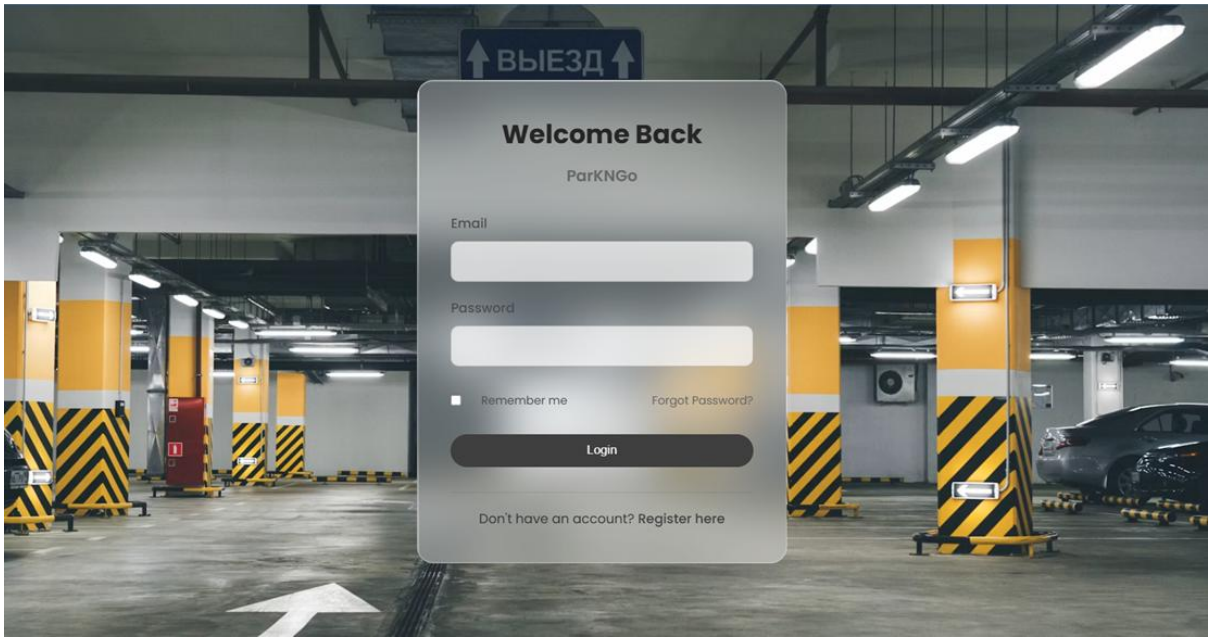
Appendix 5.2.2: About Us

Contact Us



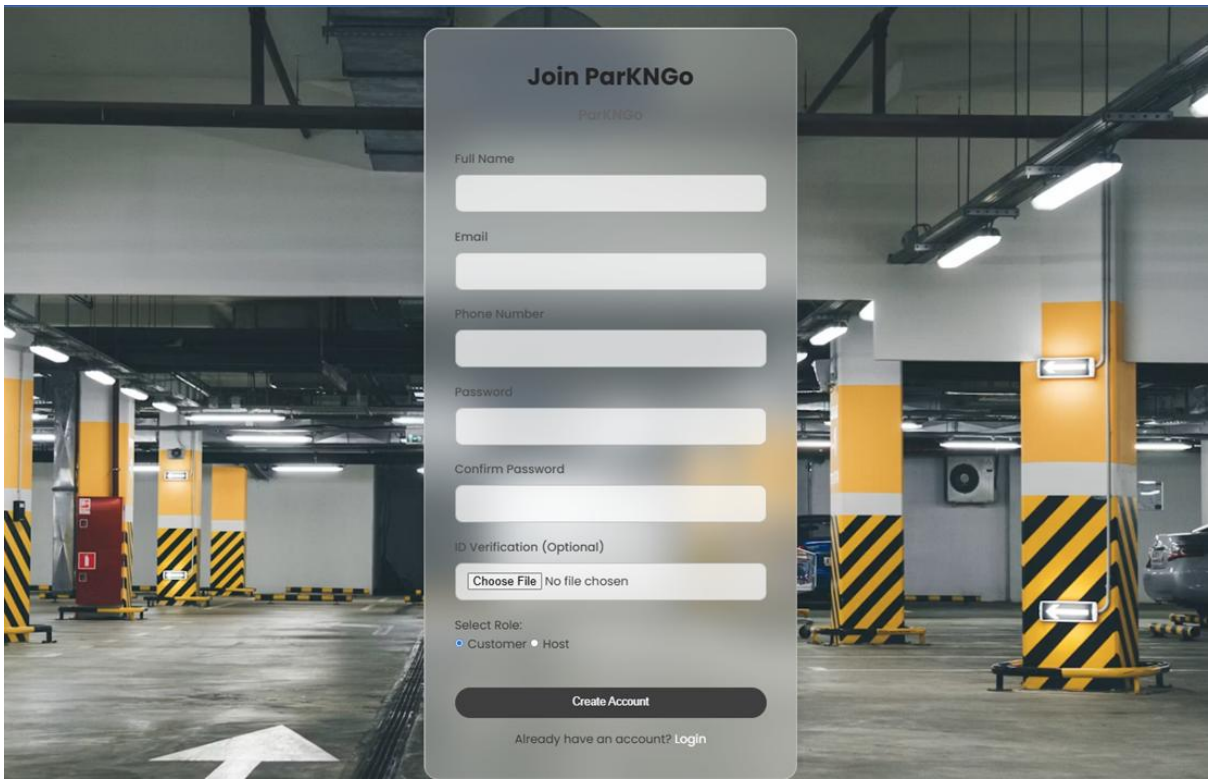
Appendix 5.2.3: Contact Us

Login



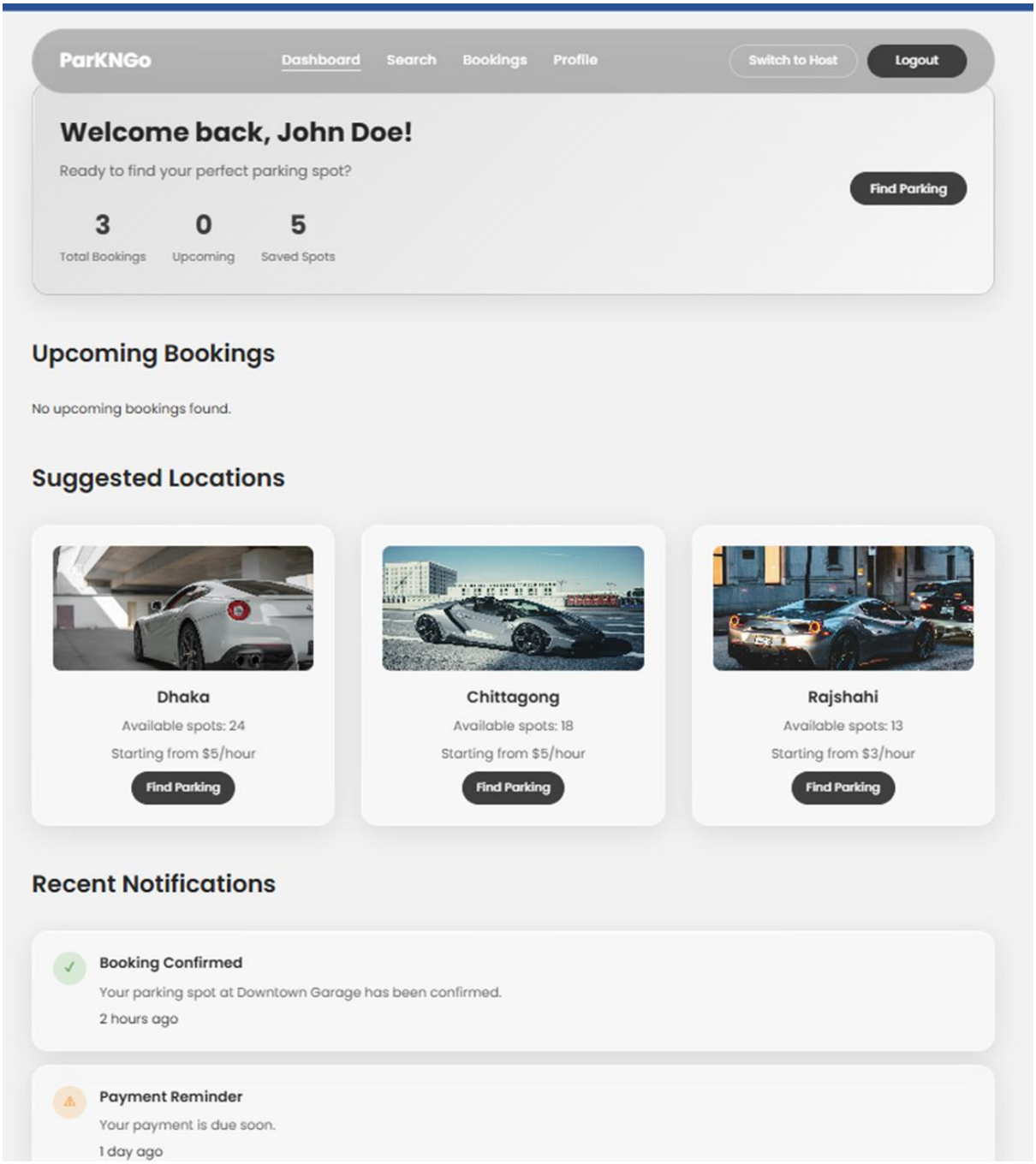
Appendix 5.2.4: Login

Sign Up



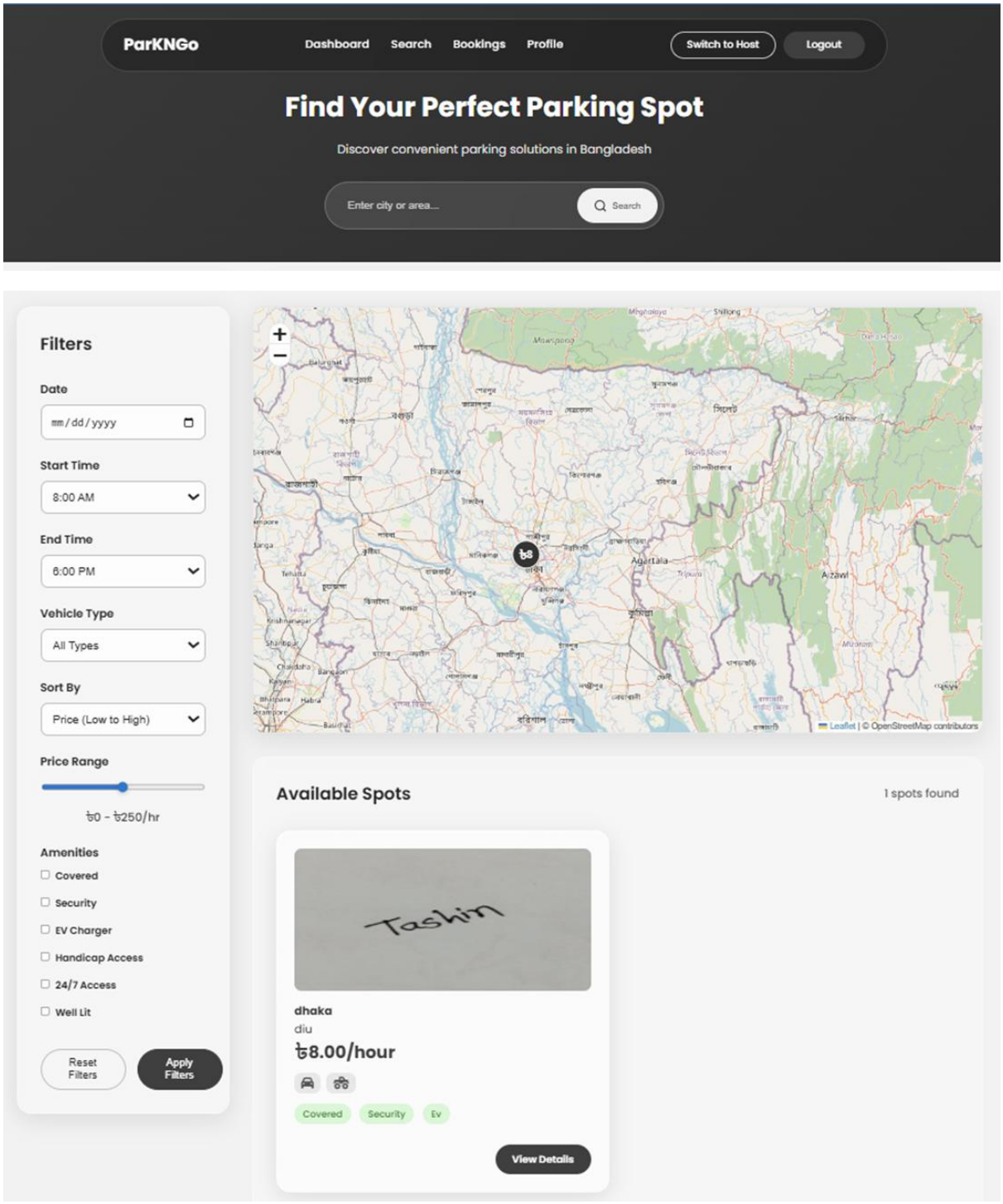
Appendix 5.2.5: Sign Up

Customer Dashboard



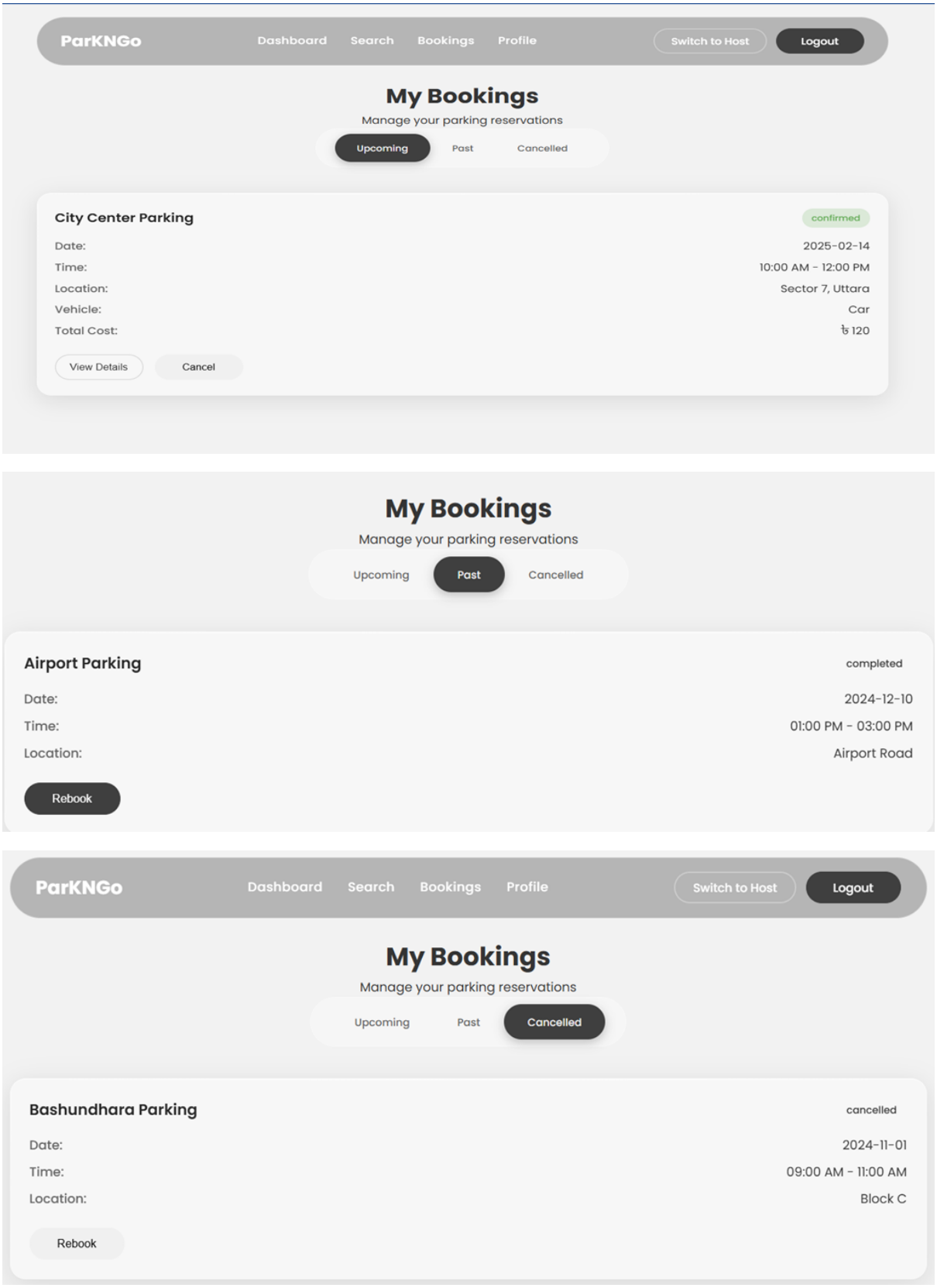
Appendix 5.2.6: Customer Dashboard

Search Spot



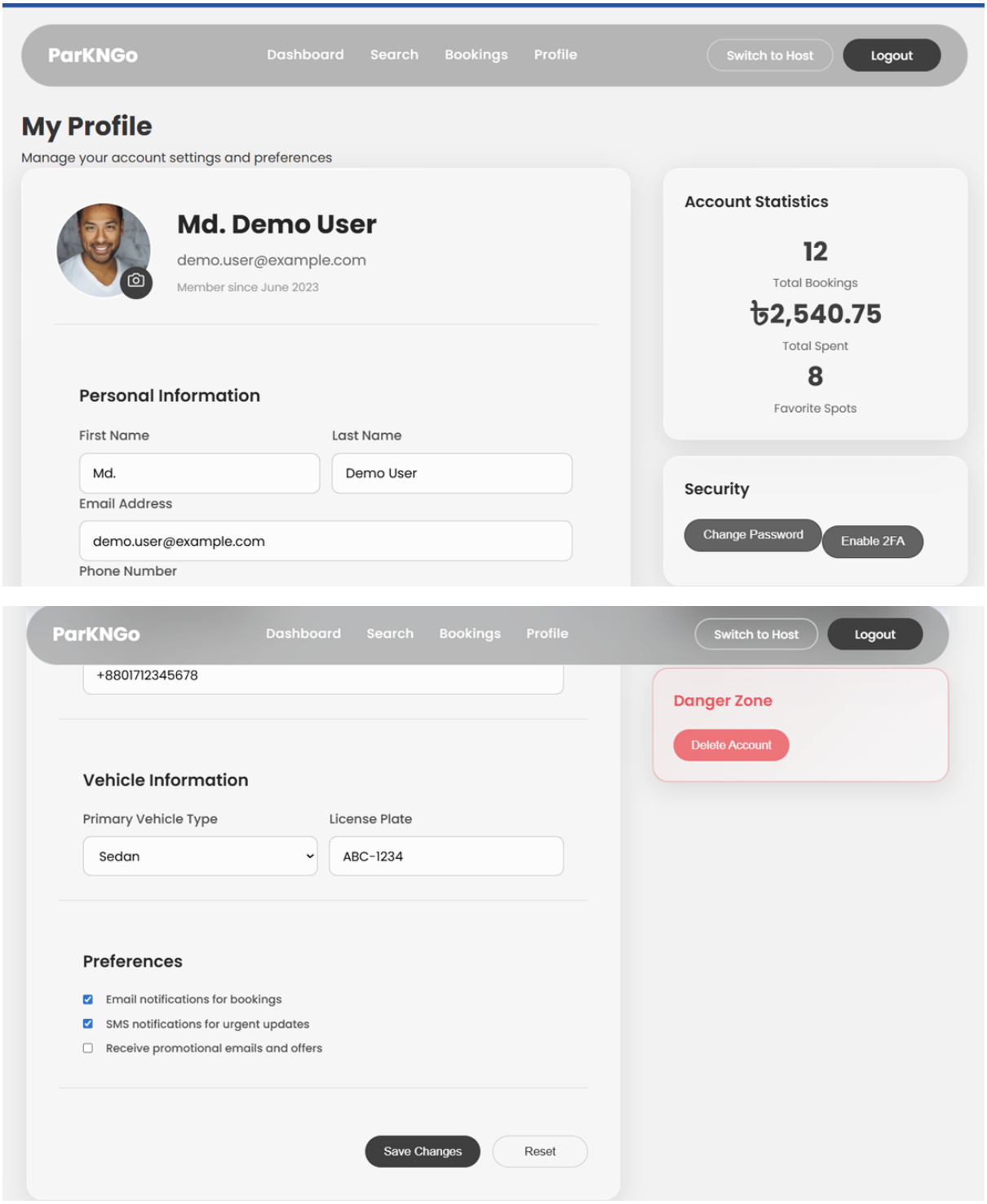
Appendix 5.2.7: Search Spot

My Bookings



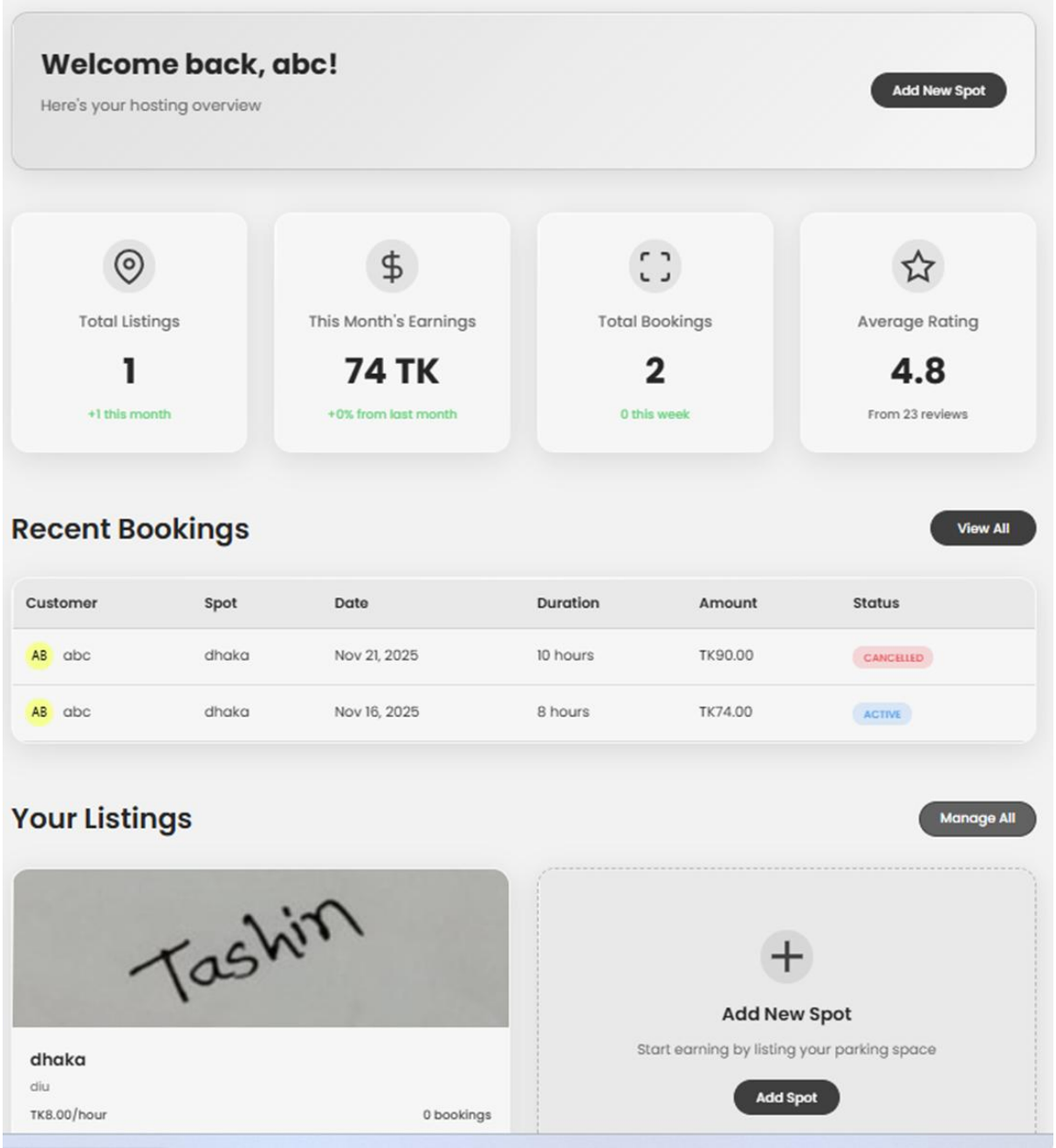
Appendix 5.2.8: **My Bookings**

Profile



Appendix 5.2.9: Profile

Host Dashboard



Appendix 5.2.10: Host Dashboard

5.3 Summary

This section provides an overview of the smart parking system wherein the users of the system gain access to the platform using role-based dashboards. The customers will be able to search and book parking spaces, locations, and manage their booking. Hosts are able to control parking space and track revenue and administrators are able to control users, parking records and system performance. The system is efficient, secure and easy to use in the management of parking.

Chapter: 6 Project Summary

6.1 Introduction

The ParKNGo - Smart Parking System is a Web-based application that is aimed at updating and simplifying the searching, booking, and managing parking spots in cities. Historically, some of the difficulties that drivers encountered have been trying to find free spaces manually, poor parking management, unavailability of real-time availability services, and improper communication between customers and the owner of the spots. The system is a solution to these challenges as it provides a user-friendly, automated and centralized parking management solution.

The system is developed with a modern technology stack consisting of Angular on the front-end, Spring Boot on the backend API, MySQL on the database management, and the security system is based on JWT with fast response and smooth scaling. Further development tools like Postman/swager, VS Code, IntelliJ IDEA, and GitHub assistance is used in making sure that development is organized and easier to maintain.

The features of the system are user registration and login, searching available parking spots, the real-time availability update, booking and cancellation, spot activation toggle, host-earnings dashboard, the system of the administration approval, and the notification service. The various groups of users, namely, Customers, Hosts and Administrators have their own dashboard with distinct functionality. UI pages, diagrams and user manual are all designed on the basis of simplicity and ease of use to give an easy experience to all users.

Altogether, the ParKNGo system can be considered an effective way to improve the efficiency of parking as it offers a digital environment that automatizes the routine activities and procedures, saves time on searching a parking space, expands the opportunities of the hosts to earn money, and improves the control of the administration. The project does not only meet academic needs and requirements but also presents a viable impactful solution that can be applied in the real world.

6.2 Project Limitation

Even though the ParKNGo system delivers an efficient and organized digital parking management solution, it still faces some limitations:

- **Requires Internet Connection:**
  The system is solely web-based and it is unable to operate offline. To book or find parking users will require a working internet connection.
- **Image & File Upload Limits:**
  The size of the file uploaded when posting an image of a parking space might be restricted by the server settings, so it is not always possible to upload large or quality photos.
- **Strict Role-Based Access:**
  The size of the file uploaded when posting an image of a parking space might be restricted by the server settings, so it is not always possible to upload large or quality photos.
- **Scalability Based on Server Resources:**
  Although this is scalable, the performance of the system can decline in case the number of users or the number of spots that it can host increases beyond the capacity of the server without appropriate upgrades.
- **Dependency on Third-Party APIs:**
  Some of the services which rely on an external service provider are notification services (email/SMS), map APIs, and payment gateways. System functionality was susceptible to downtime or delays in these services.
- **Browser Compatibility Variations:**
  Depending on the devices with low performance and old browsers, the system might not be very effective.
- **No Mobile App (Web Only):**
  The system is only accessible through a browser, there is no dedicated mobile app at the moment which can be inconvenient to some users.

6.3 Scope

The project scope will incorporate the production of a web-based intelligent parking system that will accommodate -

- User registration
- Authentication
- Parking spot listing
- Parking spot searching
- Parking spot booking
- dashboard management for customers, hosts, and administrators.

The system also comes with the provision of the basic availability control, booking history, and earnings tracking. The project however does not include the use of real-time GPS positioning, automated sensor-based parking location, online payment gateways, and the creation of mobile application. This system is mainly meant to be used in academic and small scale operations.

6.4 Future Work

A number of improvements can be reflected on further development. This involves the addition of real-time GPS positioning to update the accurate position of the user, online payment gateway to facilitate smooth transactions and the addition of Android Android and the iOS mobile applications. Two-factor authentication and role-based access control are some of the advanced security measures that can be implemented to enhance protection of data. Also, it is possible to consider AI-based parking suggestions and real-time analytics to make the system smarter and more scalable.

6.5 Conclusion

To sum up, the ParKNGo Smart Parking System is an effective resolution to the problematic situation in the sphere of traditional parking management as it offers a well-organized, user-friendly, and centralized parking management system. The project shows good use of the web development technologies, database management and software engineering guidelines. In this project, great lessons were gained in system design, testing, deployment, and teamwork. Generally, the project fulfills the set goals and includes a solid base of future developments and practical application.

REFERENCES

Academia Research Report. (2017). *IoT-enabled smart parking system: Design and implementation study*.

Angular Team. (2025). *Angular official documentation* (Version 17+). Google. https://angular.dev

Baeldung. (2024). *REST API with Spring Boot 3 and Spring Security 6*. https://www.baeldung.com

Baeldung. (2025). *Testing with Spring Boot: Unit tests, integration tests, and MockMvc*. https://www.baeldung.com

BrowserStack. (2025). *How to write a good test summary report*.

Cybertec SQL. (2024). *Creating ER diagrams with SQL and Mermaid*.

GeeksforGeeks. (2022). *Software testing: Web application testing checklist with test scenarios*. https://www.geeksforgeeks.org

Leaflet.js Team. (2025). *Leaflet maps library: Official documentation* (Version 1.9.x). https://leafletjs.com

Drow.io Project. (2025). *class diagram and flowchart syntax documentation* (Version 11.x).https://app.diagrams.net/

OpenStreetMap. (2025). *Using Leaflet with OpenStreetMap tiles: Developer guide*.

Spring.io. (2025a). *Spring Boot 3 reference documentation* (Version 3.3). https://docs.spring.io/spring-boot

Spring.io. (2025b). *Spring Data JPA documentation* (Version 3.3). https://docs.spring.io/spring-data/jpa

Spring.io. (2025c). *Spring Security documentation: Authentication, authorization, and user management* (Version 6.3). https://docs.spring.io/spring-security

Visual Paradigm. (2024). *UML class diagram tutorial and notation guide*.

Visual Paradigm Online. (2024). *Online UML diagram tool: Feature overview*