INTRODUCTION

"Welcome to Park Harbour, a cutting-edge web application designed to revolutionize urban parking solutions. In the hustle and bustle of city life, finding a convenient parking spot for your vehicle can be a daunting task. Park Harbour emerges as the ultimate solution, employing advanced technologies such as Java, ReactJS, MySQL, and Spring Boot to seamlessly connect vehicle owners with available parking spaces.

Our user-centric platform not only simplifies the parking process through real-time location-based searches and reservation functionalities but also empowers individuals to become parking providers, creating a mutually beneficial community. With a robust security infrastructure, Park Harbour prioritizes the safety of your vehicle, providing a trustworthy environment for users.

The inclusion of an admin module ensures the smooth functioning of the application, allowing for dynamic parking fee management, issue resolution, and customer care support. Join Park Harbour today to experience hassle-free parking, enhanced safety, and the opportunity to contribute to a collaborative urban parking ecosystem."

**1.1 Purpose**

The purpose of the "Park Harbour" web application is to provide a user-friendly platform that addresses urban parking challenges by connecting vehicle owners with available parking spaces. It aims to streamline the parking process, enhance safety, and create economic opportunities for individuals willing to offer their parking spaces.

**1.2 Scope**

The application will be built using Java, ReactJS, MySQL, and Spring Boot. It will support real-time location-based searches, reservation functionalities, and a dual-sided approach allowing users to find parking spaces and individuals to become parking providers.

**2. System Overview**

**2.1 System Description**

"Park Harbour" will consist of a web-based interface accessible to users and parking providers. The system will utilize Java and Spring Boot for backend development, ReactJS for the frontend, and MySQL for database management.

**2.2 Features**

•User registration and authentication

•Real-time location-based parking search

•Reservation and payment functionality

•Dual-sided system for users and parking providers

•Secure user data management

•Robust safety measures for vehicle protection

**3. Functional Requirements**

**3.1 User Module**

•User registration and login

•User profile management

•Real-time parking space search

•Reservation and payment processing

•Notification system for users and providers

**3.2 Parking Provider Module**

•Provider registration and login

•Parking space listing

•Reservation confirmation and denial

•Earnings tracking and management

**3.3 Admin Module**

•Admin login and authentication

•Parking fee management and updates

•User and provider oversight

•Issue management and resolution

•Customer care support and communication

**4. Non-functional Requirements**

**4.1 Performance**

•The system shall support a minimum of 1000 simultaneous users.

•Response time for search and reservation processes should be within 3 seconds.

**4.2 Security**

•User data and payment information shall be encrypted and securely stored.

•Access controls and authentication mechanisms shall be implemented.

**5. Database Requirements**

**5.1 Database Design**

•MySQL will be used to store user profiles, parking space details, reservations, and financial transactions.

**6. User Interface Requirements**

**6.1 User-friendly Interface**

•The application shall have an intuitive and responsive UI using ReactJS.

•Clear navigation and easy access to features.

**7. Constraints**

•The application will initially support English language only.

•Availability of real-time location data for accurate parking space suggestions.

**8. Conclusion**

The "Park Harbour" SRS outlines the requirements for the development of a comprehensive web application addressing urban parking challenges. By leveraging Java, ReactJS, MySQL, and Spring Boot, the platform aims to redefine urban parking with a focus on convenience, safety, and economic opportunities.

SYSTEM REQUIREMENTS

HARDWARE REQUIREMENTS:

* System : Pentium i3 Processor
* Hard Disk : 500 GB.
* Monitor : 15’’ LED
* Input Devices : Keyboard, Mouse
* RAM : 2 GB

SOFTWARE REQUIREMENTS:

* Operating system : Windows 10.
* Coding Language : JAVA.
* Tool : STS 3.8.19
* Database : MYSQL 8.0

SYSTEM REQUIREMENTS FOR PROJECT MAKING

**Hardware Requirements:**

**1.Minimum Server Configuration:**

•Dual-core processor (or higher)

•8 GB RAM

•Solid State Drive (SSD) for improved data access speed

•Stable internet connection with sufficient bandwidth

**2.Recommended Server Configuration:**

•Quad-core processor (or higher)

•16 GB RAM or more

•SSD for the operating system and database storage

•Redundant power supply for increased reliability

**3.Database Server:**

•Adequate storage space for the database

•Recommended: Separate server for the database for better performance

•Backup and recovery system for data protection

**4.Client Devices:**

•Personal computers, laptops, or tablets

•Internet connectivity

•Modern web browsers such as Google Chrome, Mozilla Firefox, or Safari

**Software Requirements:**

**1.Web Server:**

•Apache, Nginx, or any other suitable web server

**2.Database Management System:**

•MySQL (version compatible with Spring Boot)

•Database administration tool for managing and monitoring the database (e.g., MySQL Workbench)

**3.Backend:**

•Java Development Kit (JDK) - compatible version with Spring Boot

•Spring Boot framework

**5.Frontend:**

•ReactJS library

**6.Development Tools:**

•Integrated Development Environment (IDE) for Java development (e.g., IntelliJ IDEA, Eclipse)

•Code version control system (e.g., Git)

**7.Security:**

•Secure Sockets Layer (SSL) certificate for secure data transmission

•Firewall configuration to control access to the server

**8.Miscellaneous:**

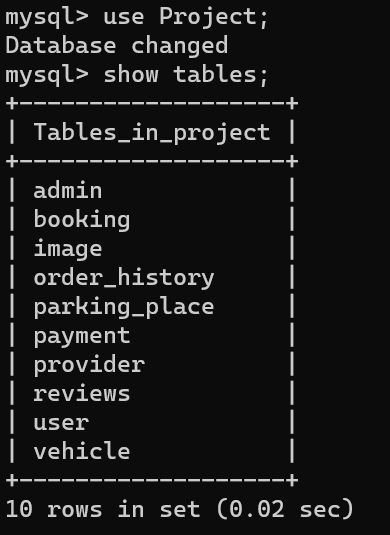
•Node.js for managing frontend dependencies

•Package manager for Java (e.g., Maven)

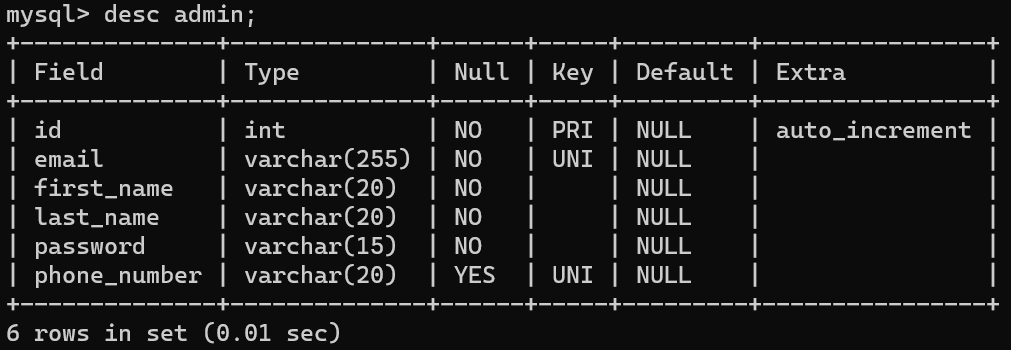
•Package manager for Node.js (e.g., npm)

Ensure that all software components are updated to the versions compatible with each other for smooth integration and functionality of the "Park Harbour" application.

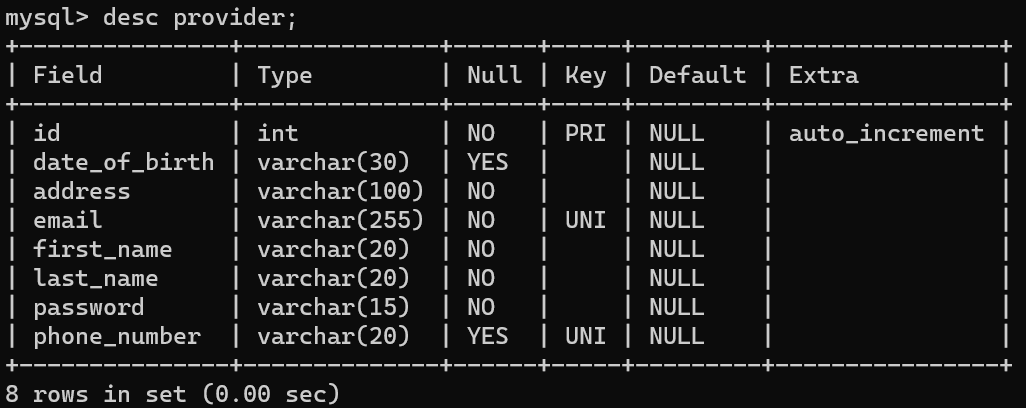
DATABASE TABLES



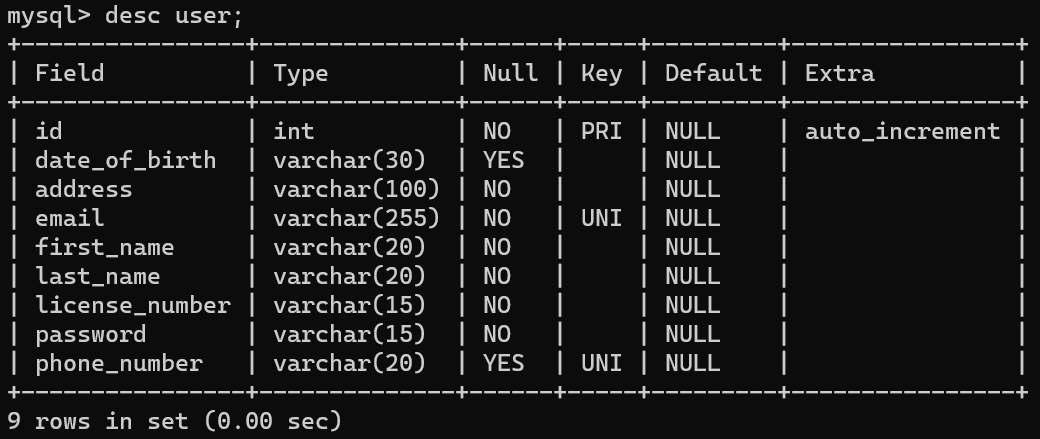
Admin Table :



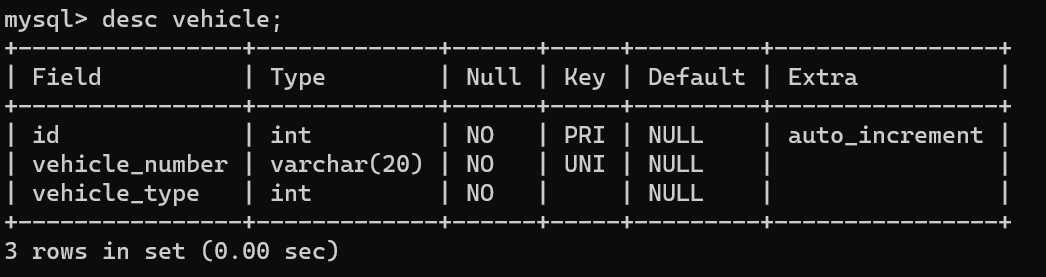
Provider Table :



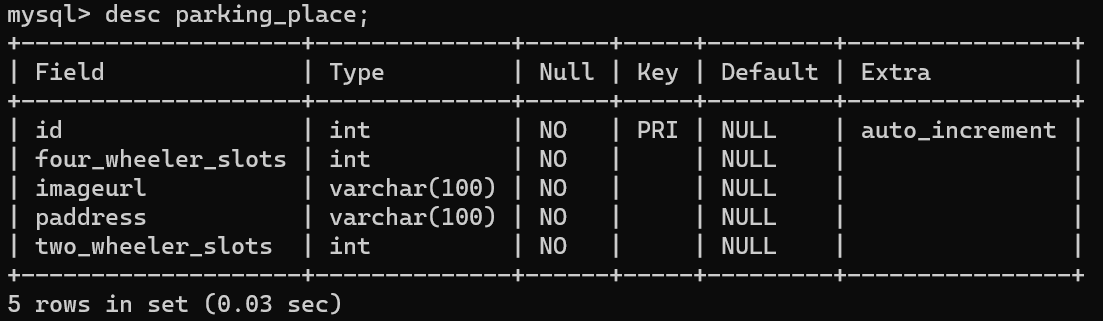
User Table :



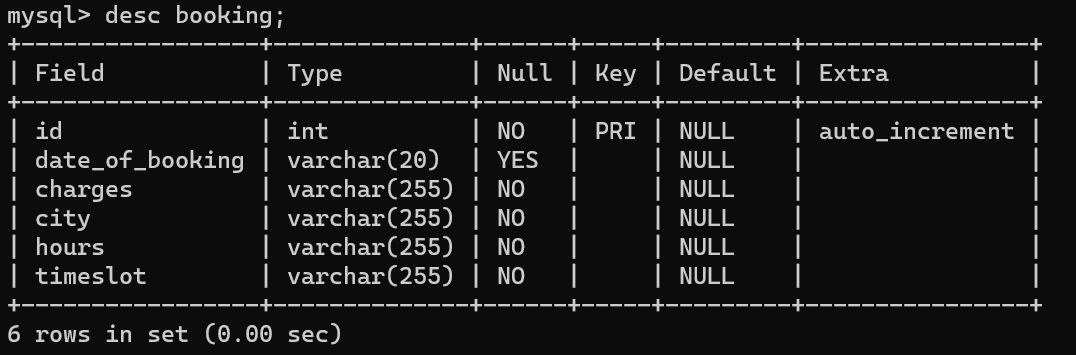
Vehicle Table :



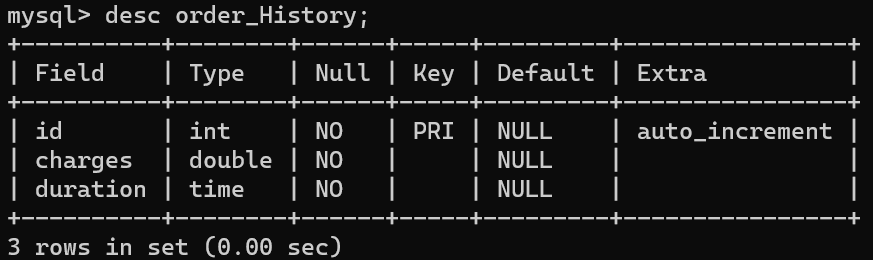
Parking Place Table :



Booking Table :



Order History Table :



Reviews Table :

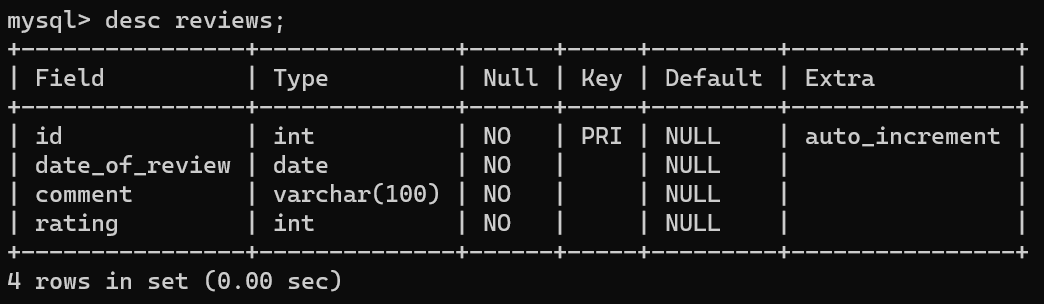
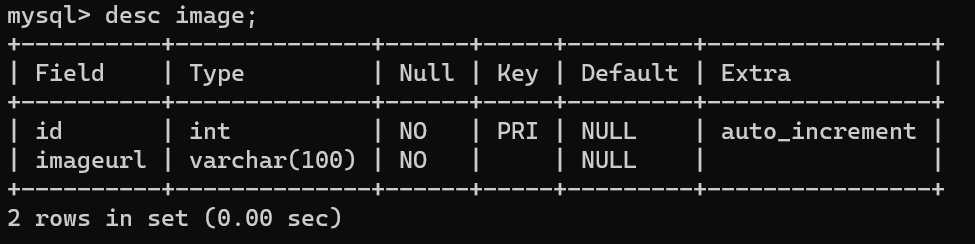
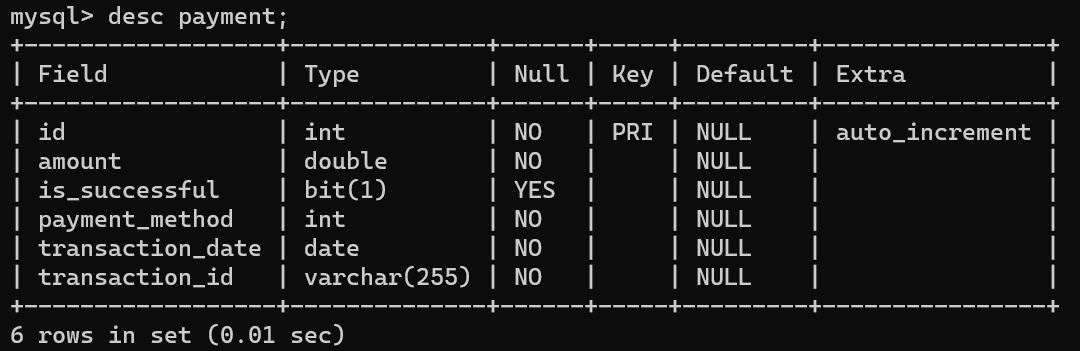


Image Table :

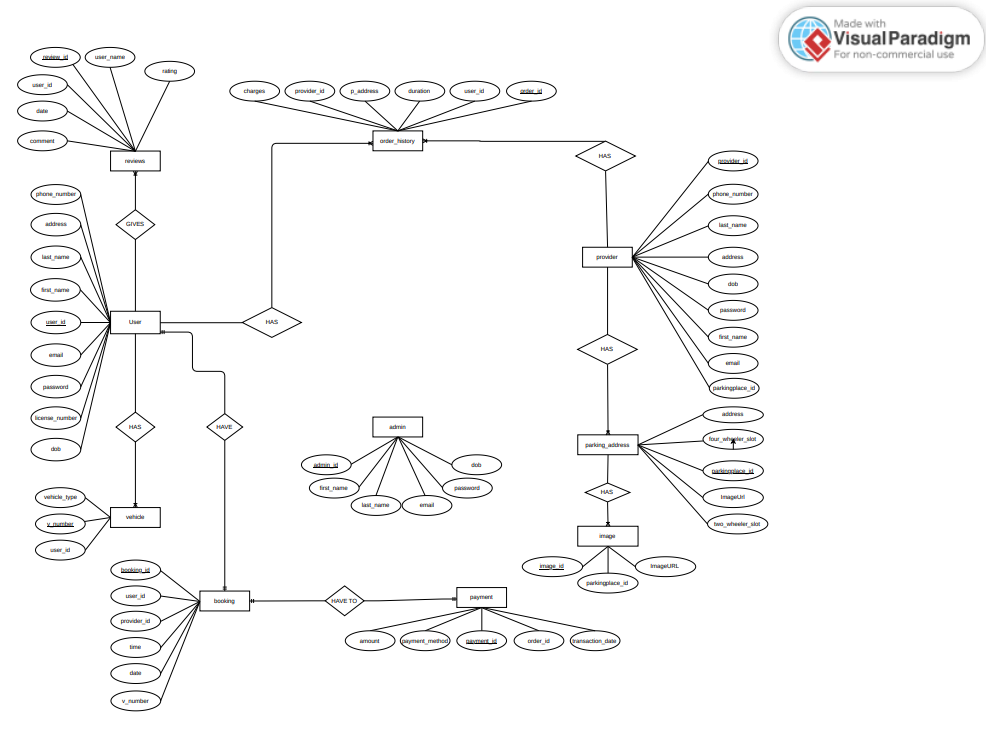


Payment Table :

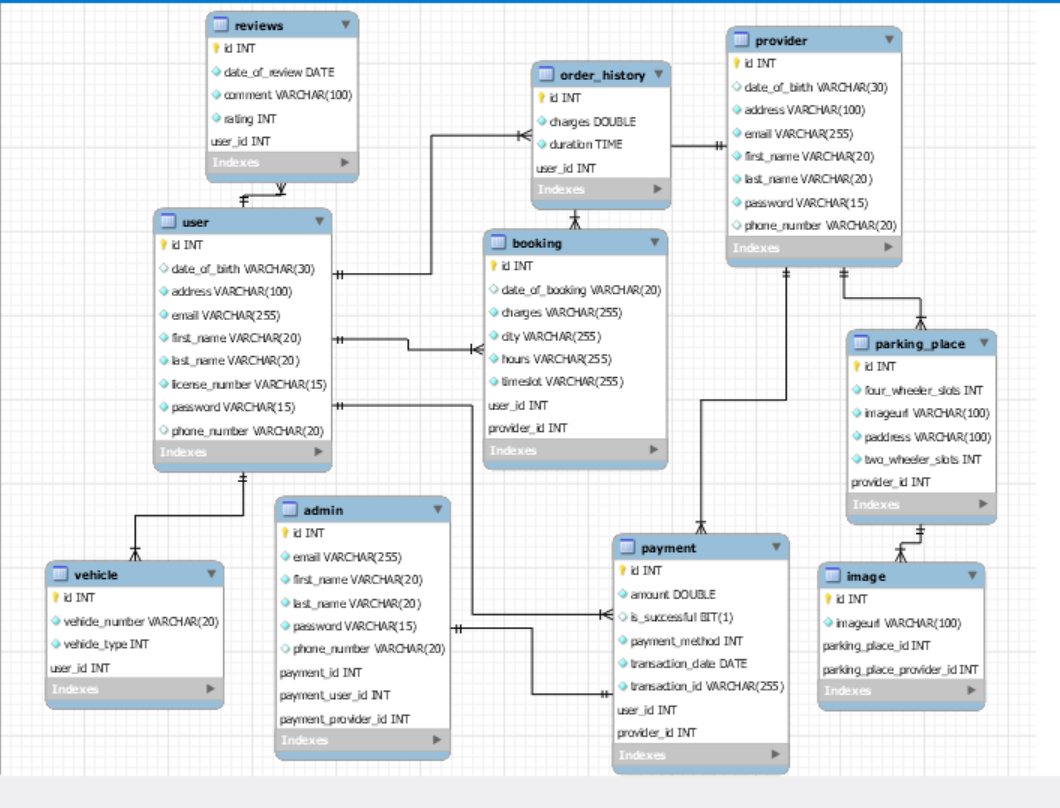


**APPENDIX-A**

ER DIAGRAM

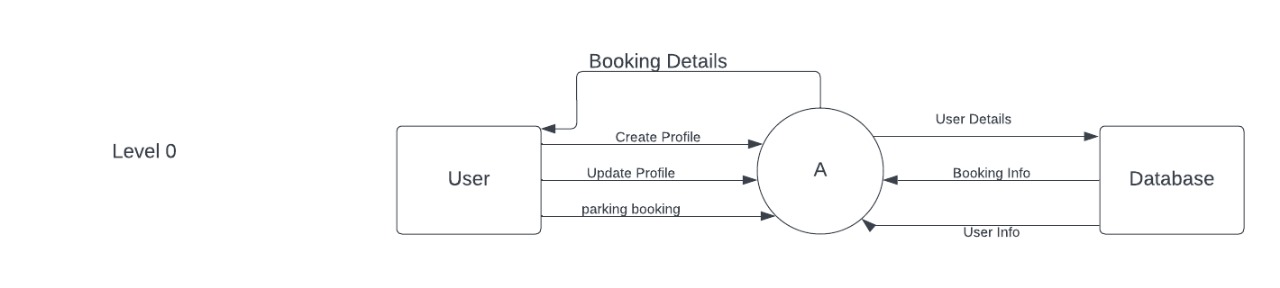


SYSTEM GENERATED ER DIAGRAM

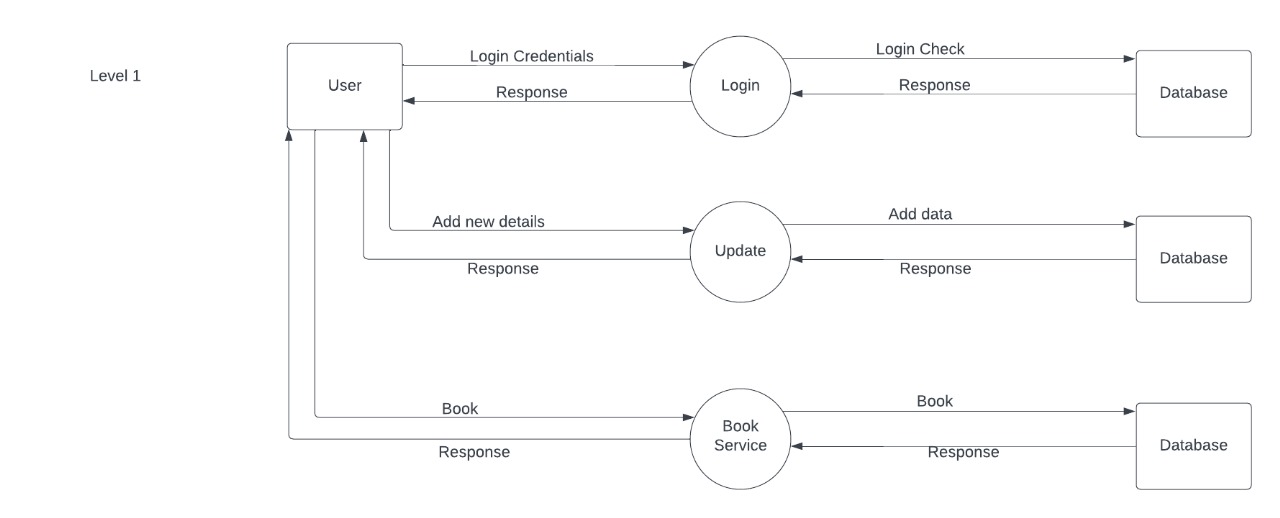


DFD DIAGRAM

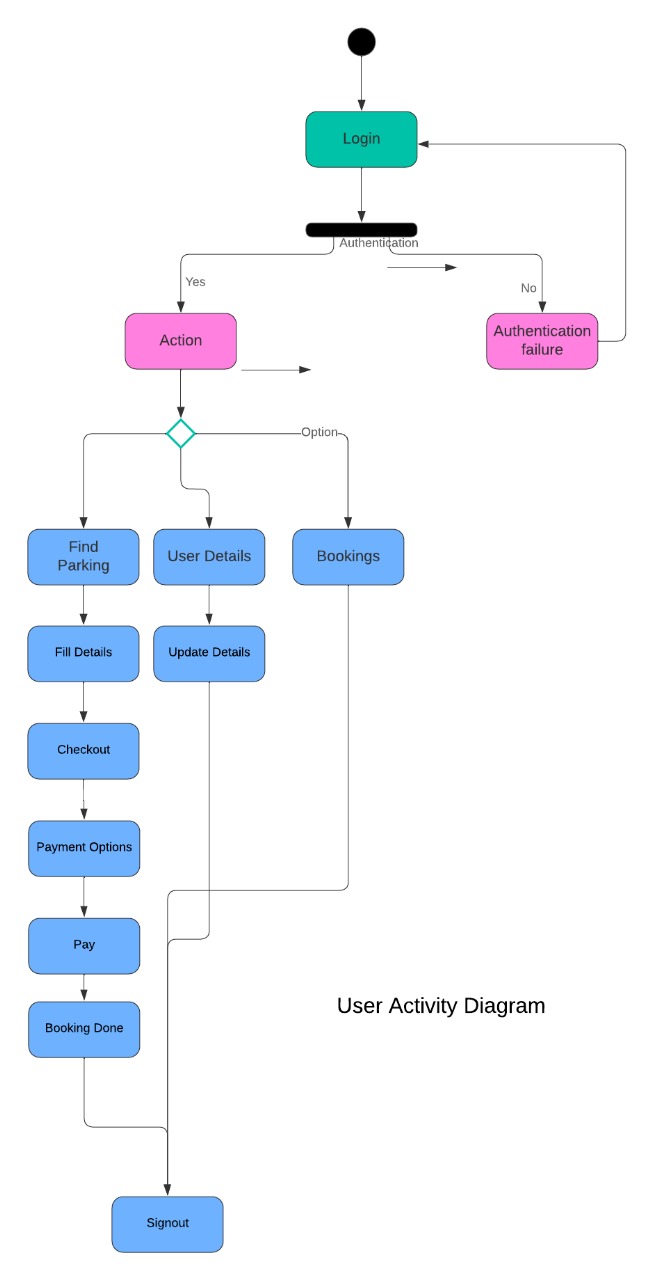
1.



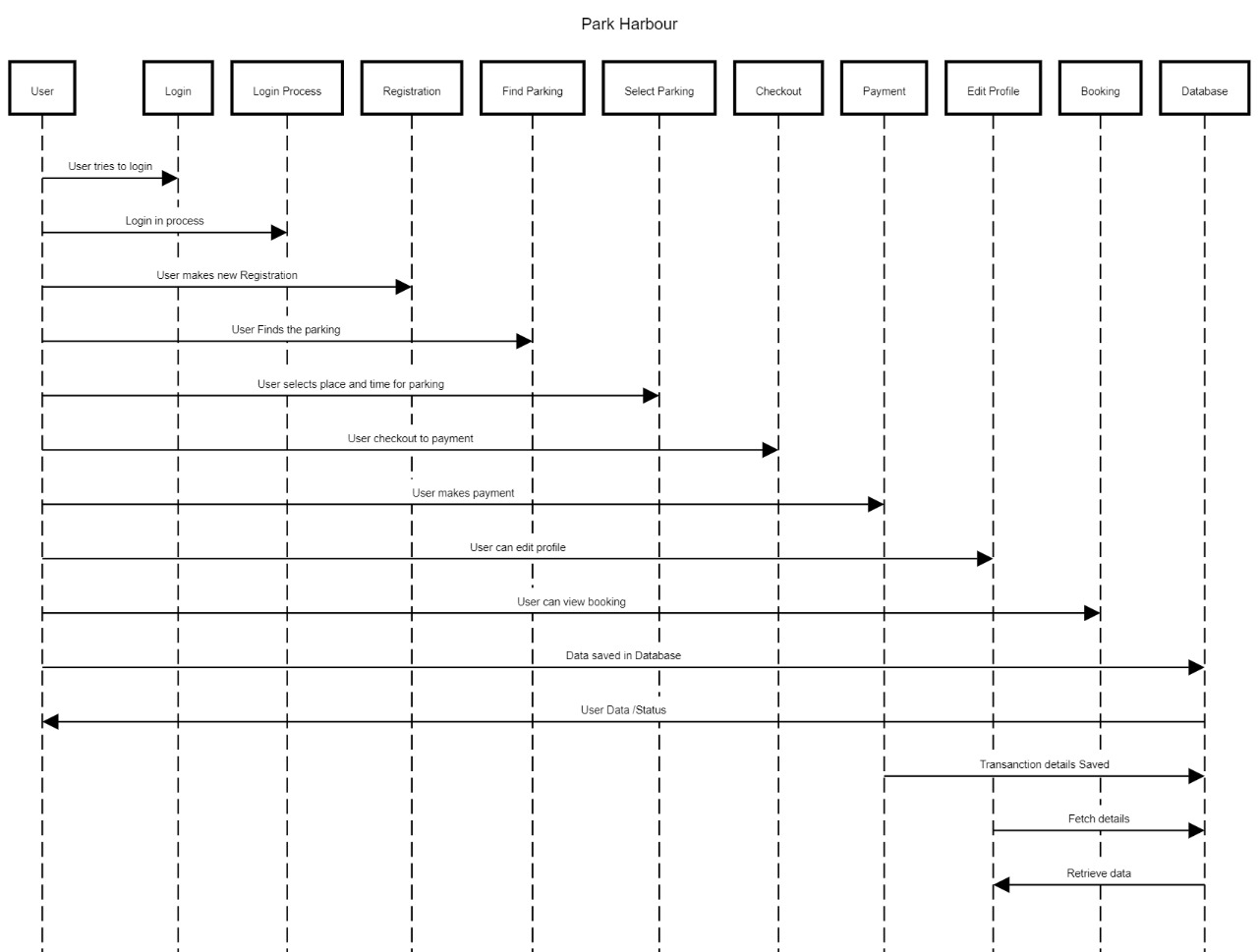
2.



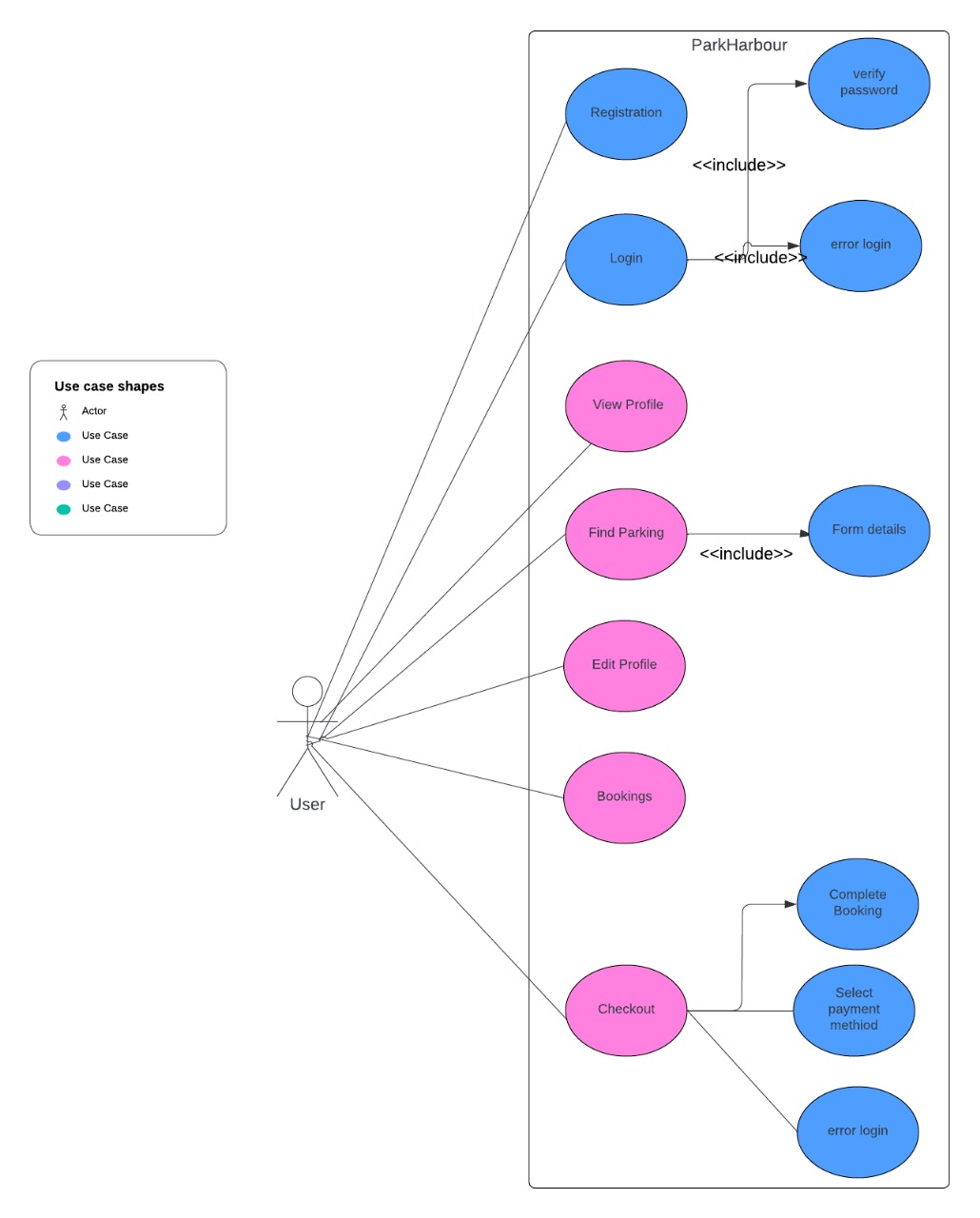
ACTIVITY DIAGRAM



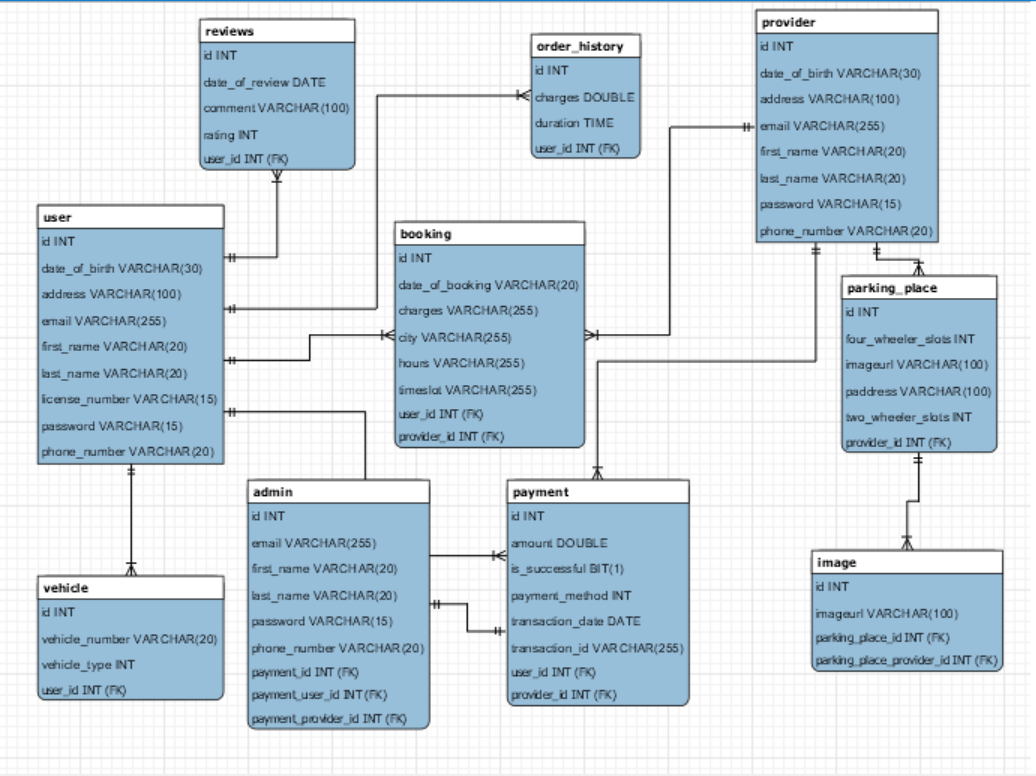
SEQUENCE DIAGRAM



USE-CASE DIAGRAM

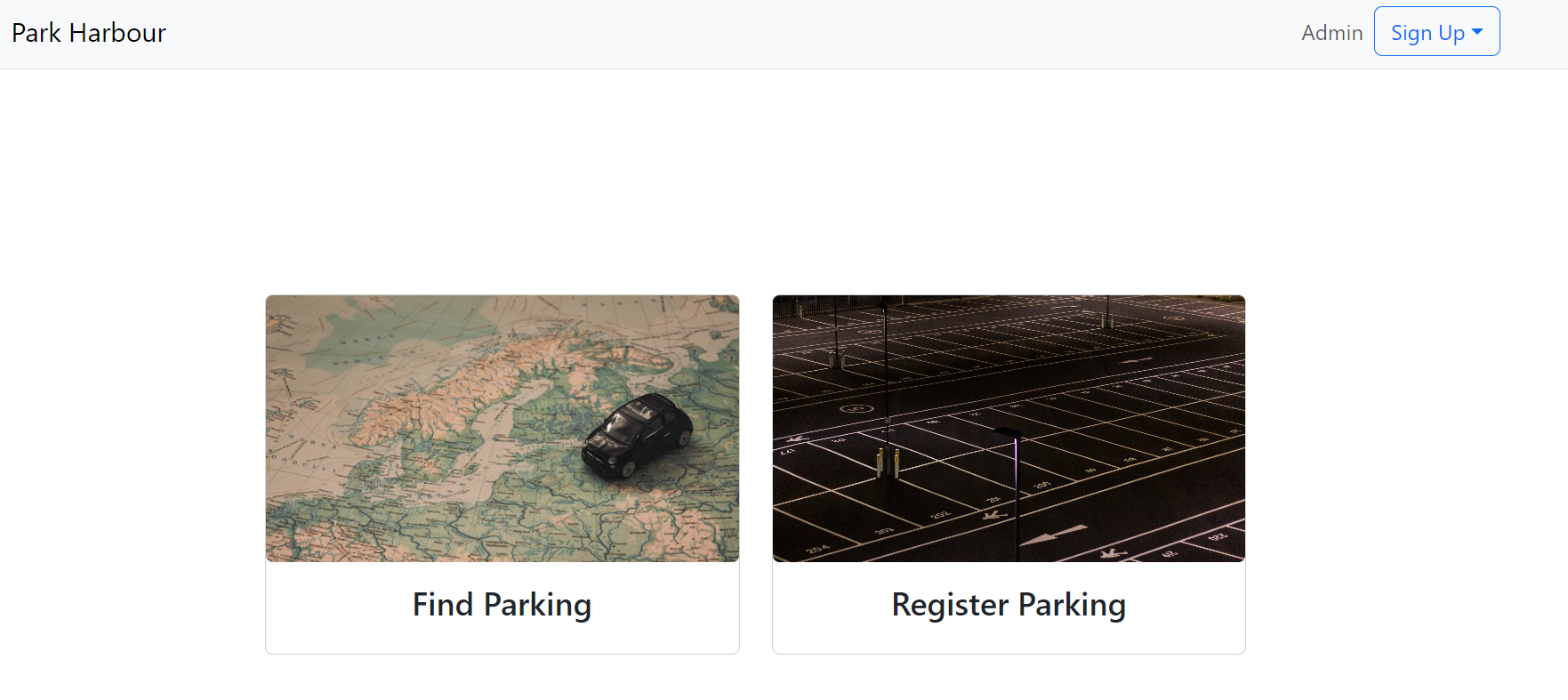


CLASS DIAGRAM

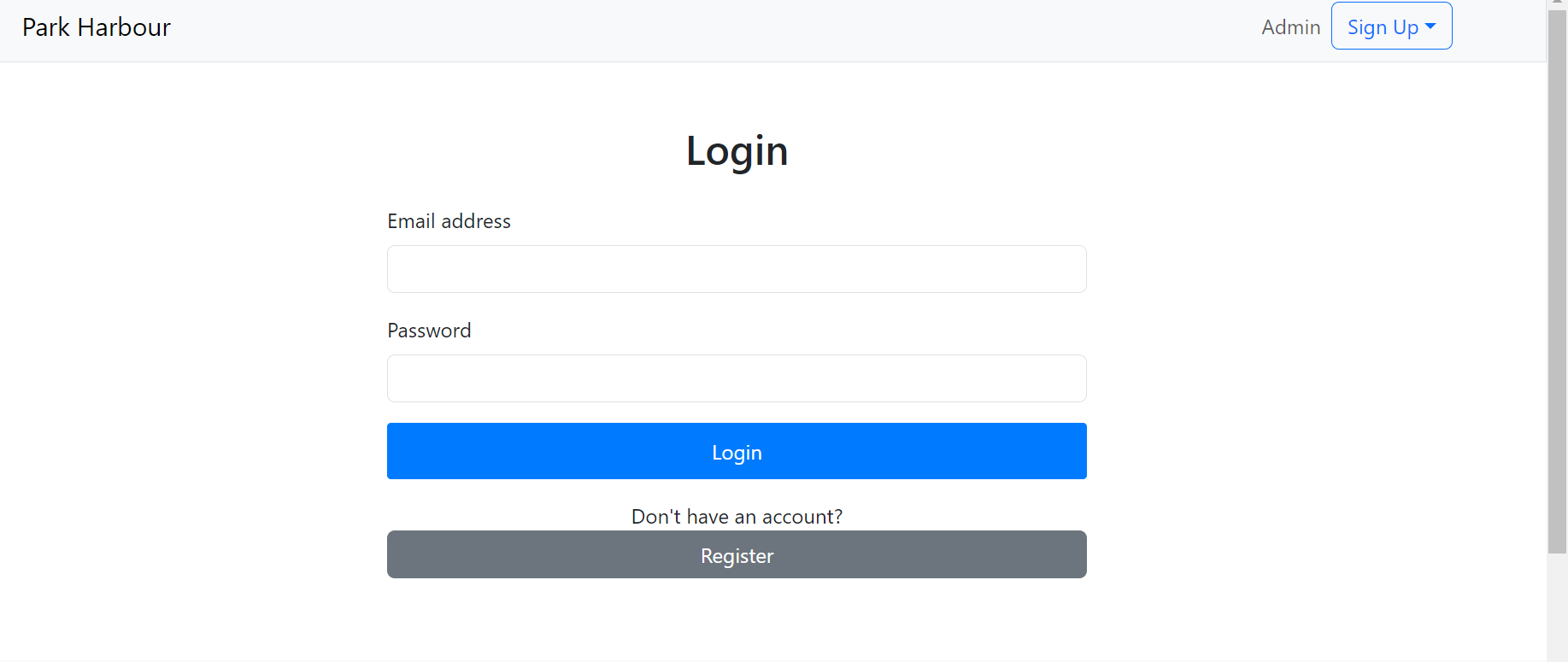


SCREENSHOTS

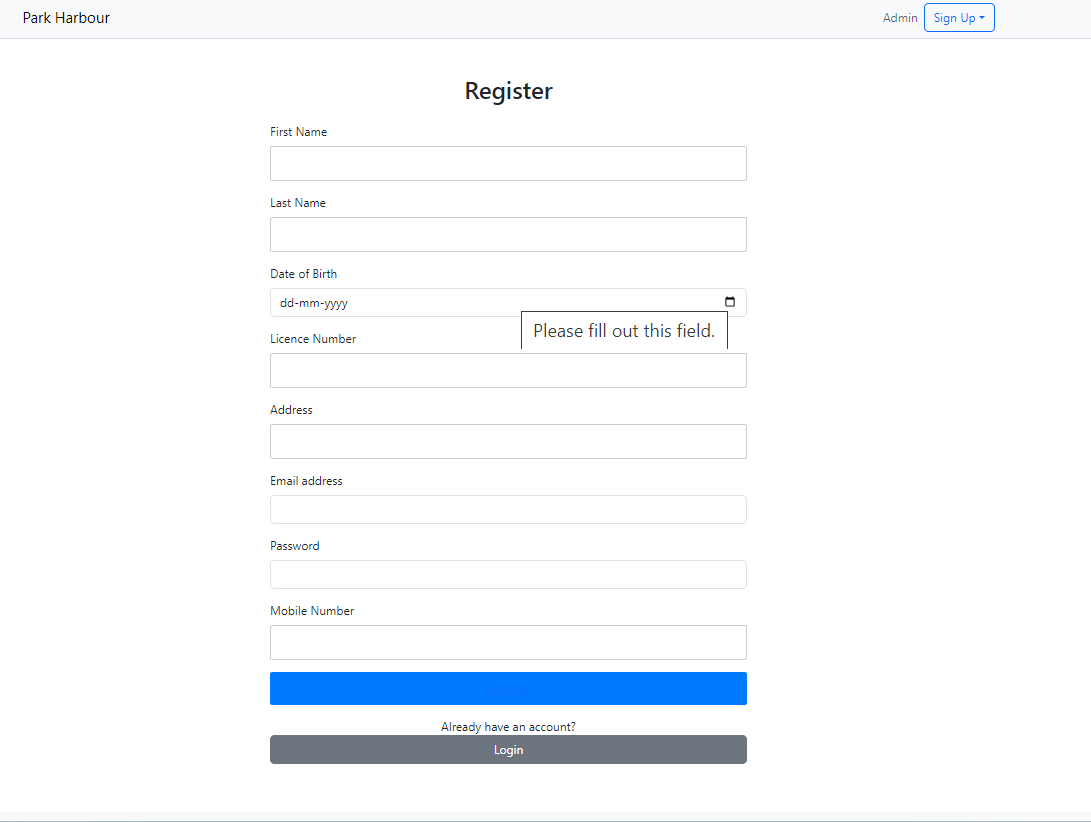
Front Page



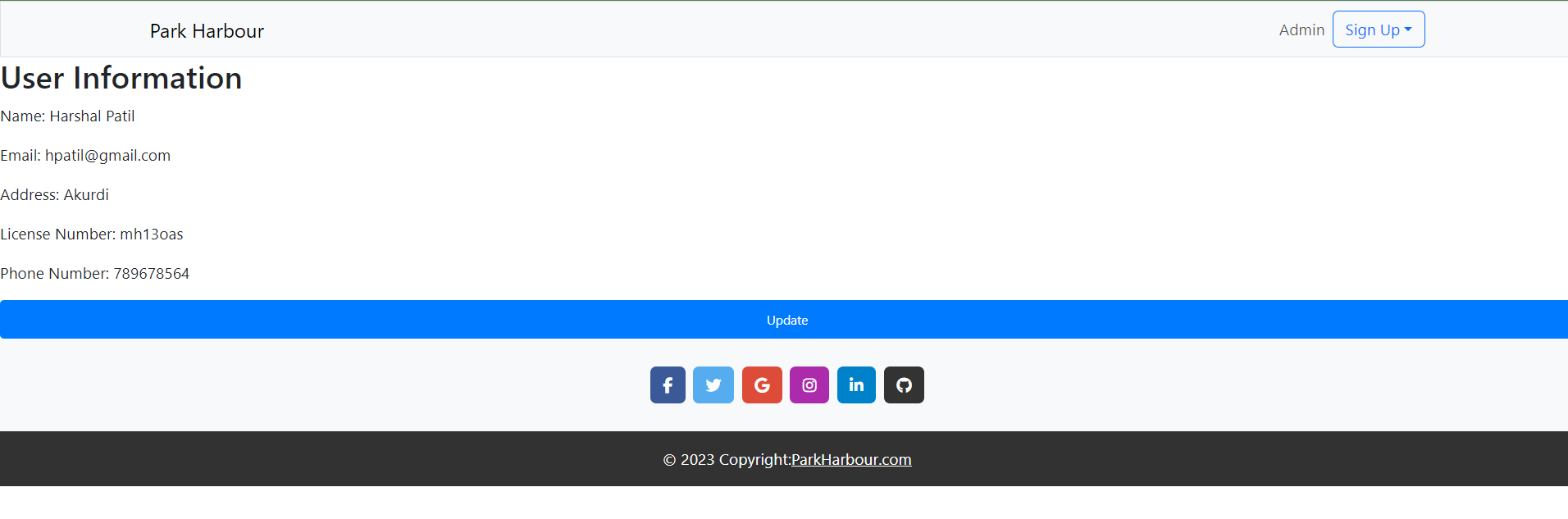
Login Page



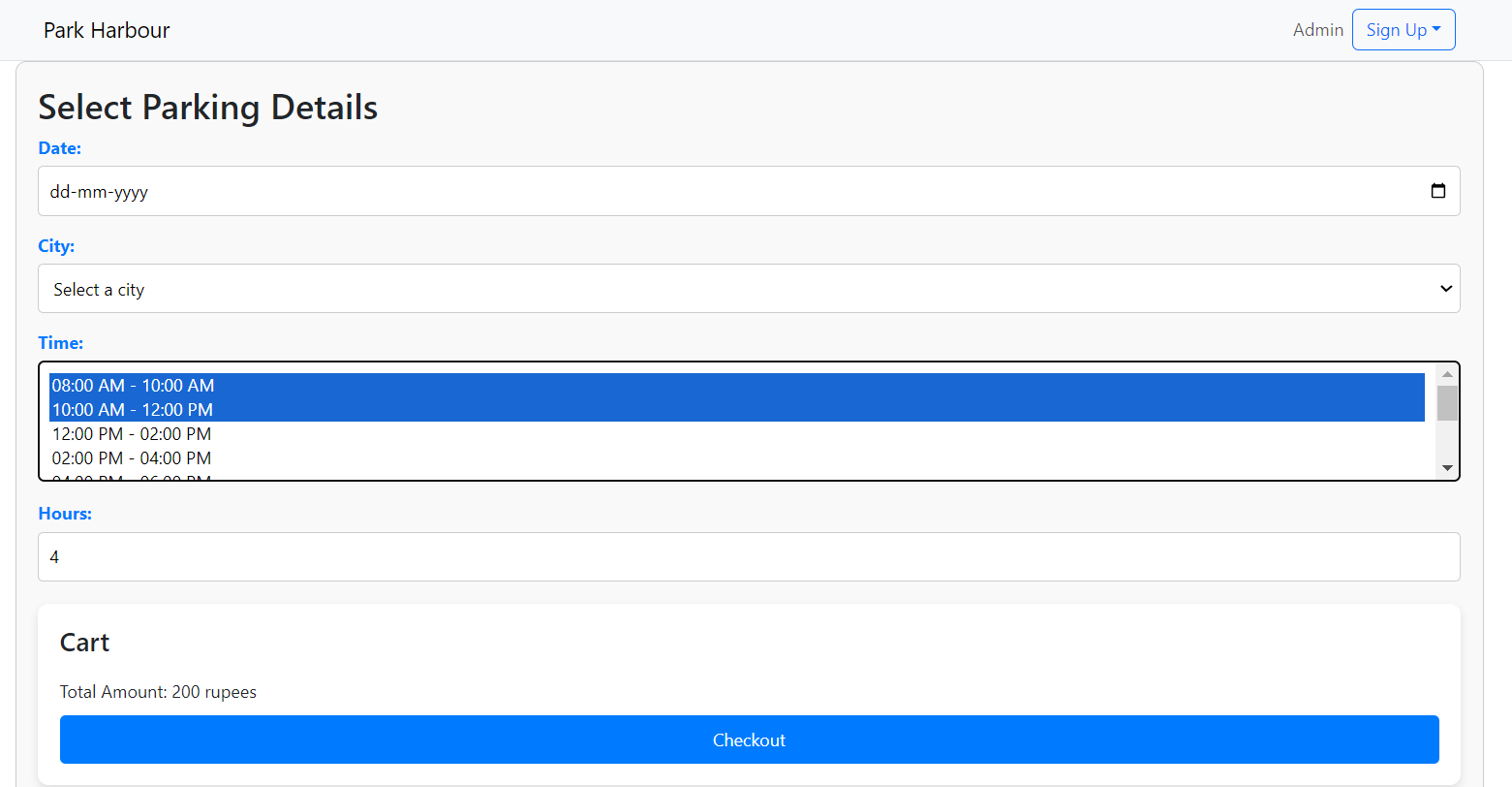
Registration Page



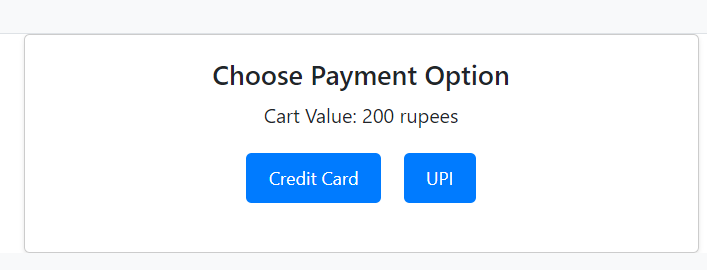
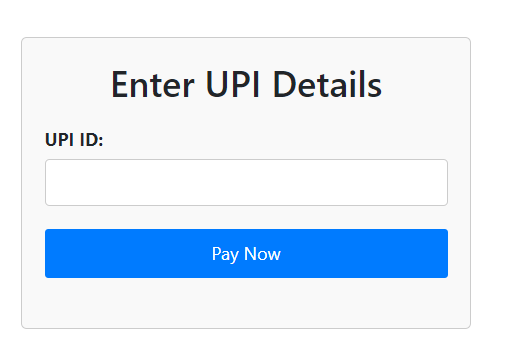
Profile Page

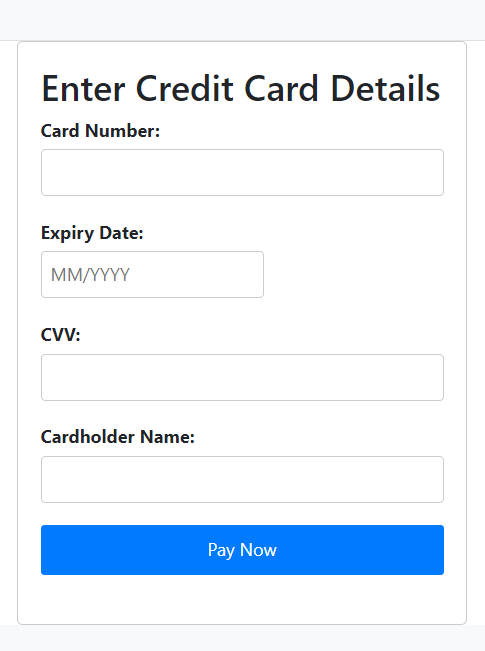


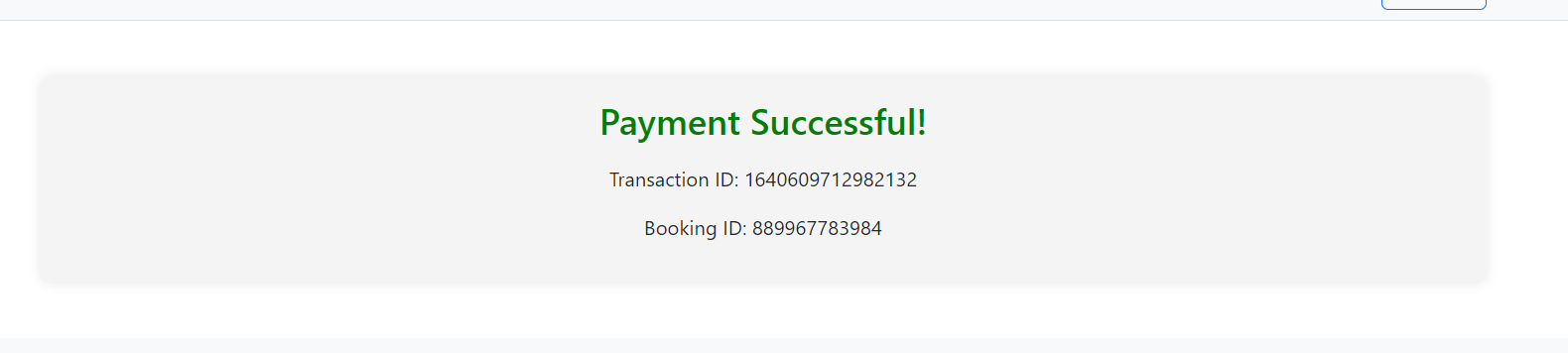
Booking Page



Payment Page





PROJECT CLOSURE REPORT - "PARK HARBOUR"

**Project Information:**

**• Project Name:** Park Harbour

**•** **Project Guide:** Mrs. Gauri Kadam

**•** **Project Team:** Aman Gupta, Harshal Ganesh Patil

**Objective:**

The objective of the "Park Harbour" project was to develop an online web application addressing urban parking challenges by connecting users with available parking spaces. The project aimed to enhance user convenience, safety, and economic opportunities for parking providers.

**Key Deliverables:**

1.User-friendly web application with real-time location-based parking searches.

2.Dual-sided system allowing users to find parking spaces and providers to offer them.

3.Admin module for overseeing application functionality and managing system parameters.

4.Integration of Java, ReactJS, MySQL, and Spring Boot technologies.

**Project Achievements:**

1.Successful implementation of core features, including user authentication, reservation, and payment processing.

2.Creation of a robust security infrastructure ensuring the safety of user data and vehicles.

3.Development of a responsive and intuitive user interface using ReactJS.

4.Inclusion of an admin module for dynamic parking fee management, issue resolution, and customer support.

**Challenges and Mitigations:**

**1.Technical Challenges**:

Overcame technical hurdles during the integration of Java and ReactJS by engaging in comprehensive testing and collaboration between development teams.

**2.Timeline Pressures:**

Mitigated timeline pressures by implementing agile project management methodologies, adjusting sprints, and setting realistic milestones.

**Lessons Learned:**

1.Effective communication is crucial for project success.

2.Regular testing and debugging should be integrated into the development process.

3.Flexibility in adapting to changing requirements is essential for project agility.

**Recommendations:**

1.Continuous monitoring and updating of technologies to maintain system security.

2.Periodic user feedback sessions to incorporate improvements.

Project Closure Sign-off: The project closure is hereby signed off by:

•Mrs. Gauri Kadam

**Acknowledgments:**

The project team expresses gratitude to all stakeholders, team members, and contributors for their dedication and efforts in making the "Park Harbour" project a success.

**Closure Report Prepared by:**  Mrs. Gauri Kadam

BIBLIOGRAPHY

<http://www.javatpoint.com/javaturorial>

<http://www.w3.org>

<http://www.wikipedia.org>

<http://www.tutuorialspoint.com/jav>

These references encompass essential resources, documentation, and best practices for the technologies and methodologies employed in the development of the “Park Harbour” project.