A PROJECT ON
Online Bus Ticket Booking.

SUBMITTED IN

PARTIAL FULFILLMENT OF THE REQUIREMENT

FOR THE COURSE OF DIPLOMA IN MOBILE COMPUTING FROM CDMC

SUNBEAM INSTITUTE OF INFORMATION TECHNOLOGY
Hinjawadi

SUBMITTED BY:

1. Pratik Savale.

2. Rhutuja Salave.

3. Virendra Mahamuni.

UNDER THE GUIDENCE OF:

Samruddhi Phadnis.

Faculty Member

Sunbeam Institute of Information Technology, Pune

ACKNOWLEDGEMENT

A project usually falls short of its expectation unless aided and guided by the right persons at the right time. We avail this opportunity to express our deep sense of gratitude towards Mr. Nitin Kudale (Center Coordinator, SIIT, Pune) and Mr. Yogesh Kolhe (Course Coordinator, SIIT ,Pune) .

We are deeply indebted and grateful to them for their guidance, encouragement and deep concern for our project. Without their critical evaluation and suggestions at every stage of the project, this project could never have reached its present form.

Last but not the least we thank the entire faculty and the staff members of Sunbeam Institute of Information Technology, Pune for their support.

Virendra Mahamuni

0324PG-DMC

SIIT Pune

A PROJECT ON

“ONLINE BUS TICKET BOOKING”

SUBMITTED IN
PARTIAL FULFILLMENT OF THE REQUIREMENT
FOR THE COURSE OF
DIPLOMA IN MOBILE COMPUTING FROM CDMC

SUNBEAM INSTITUTE OF INFORMATION TECHNOLOGY
Hinjawadi

SUBMITTED BY:

1. Pratik Savale.
2. Rhutuja Salave.
3. Virendra Mahamuni.

UNDER THE GUIDENCE OF:
Samruddhi Phadnis.
Faculty Member
Sunbeam Institute of Information Technology, PUNE.

CERTIFICATE

This is to certify that the project work under the title ‘Online Bus Ticket Booking’ is done by Virendra Mahamuni in partial fulfillment of the requirement for award of Diploma in Mobile Computing Course.

Samruddhi Phadnis **Mr. Yogesh Kolhe**

Project Guide **Course Co-Coordinator**

Date:

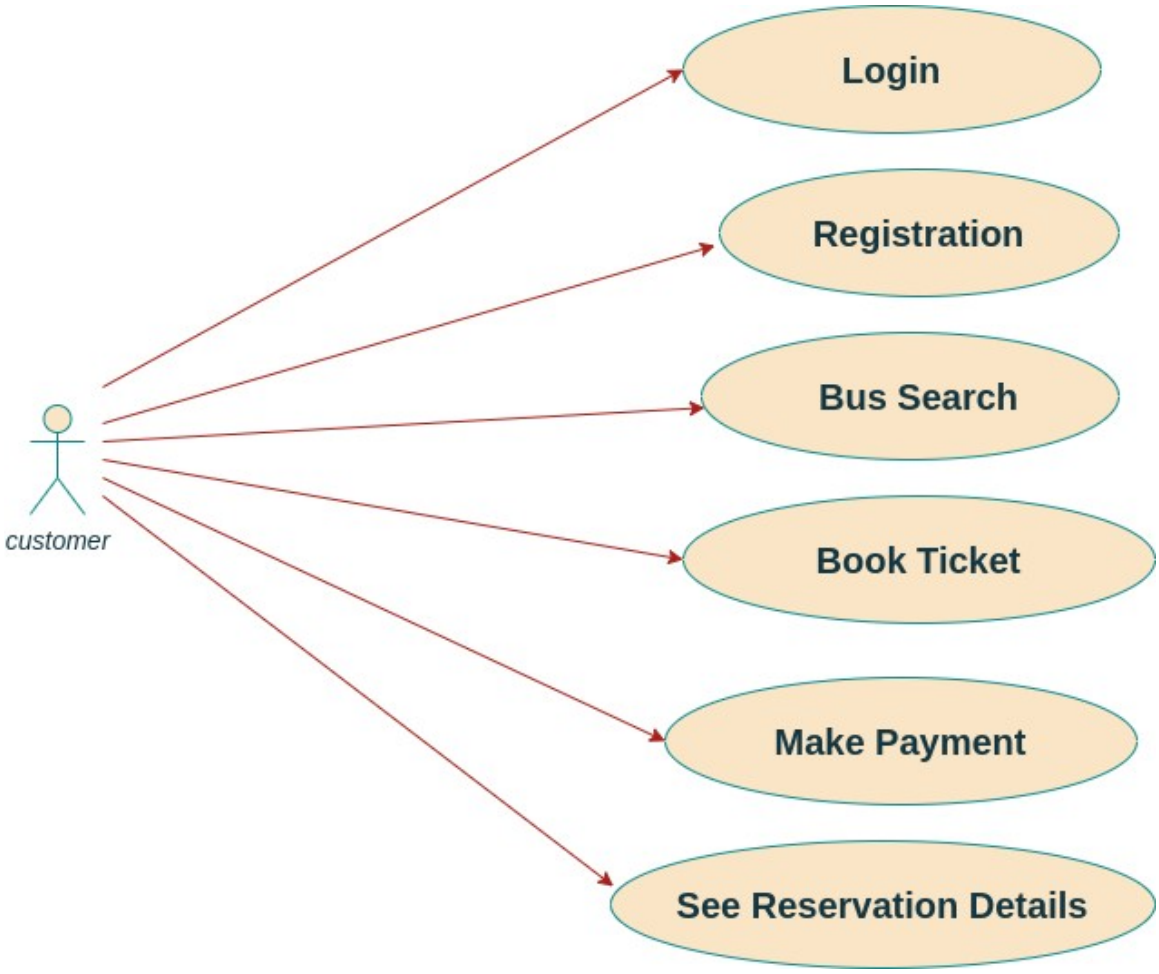
1. INTRODUCTION TO PROJECT

The Bus Booking Application is a comprehensive web-based system designed to simplify the process of bus ticket booking for users. The system allows passengers to search for available buses, select preferred seats, enter passenger details, and make payments online. The application also features an admin portal for managing buses, viewing reservations, and handling passenger details. The system is built using modern web technologies, including React for the frontend, Node.js and Express for the backend, and MySQL as the relational database management system.

This project aims to provide a user-friendly platform that facilitates seamless ticket booking, enhances customer satisfaction, and improves operational efficiency for bus operators. The application supports multiple users simultaneously, ensuring high performance and scalability to accommodate growing user demand.

2. REQUIREMENTS

2.1 FUNCTIONAL REQUIREMENTS



2.1.1 User Management

- User Registration: Users must be able to create an account by providing their first name, last name, email, and password.
- User Login: Registered users should be able to log in using their email and password.
- Password Management: Users should be able to reset their password if forgotten.
- User Profile Management: Users can update their profile information after registration.

2.1.2 Bus Search and Booking:

- Bus Search: Users can search for buses by specifying the departure location, destination, and travel date.
- Bus Listing: The system should display a list of available buses based on the search criteria, showing details like bus name, type, fare, departure time, and total available seats.
- Seat Selection: Users can select seats from the available ones in the chosen bus.
- Passenger Details Entry: For each selected seat, users must enter passenger details such as name, age, gender, boarding stop, and alighting stop.
- Booking Confirmation: After successful payment, the system should provide a booking confirmation with all relevant details.

2.1.3 Payment Processing:

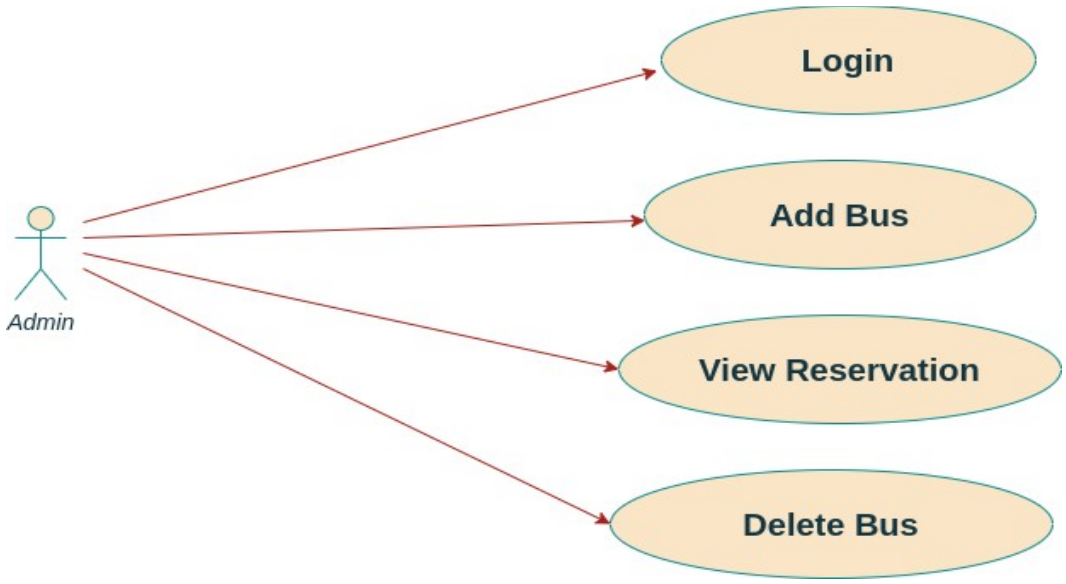
- Payment Gateway Integration: The system should integrate with a payment gateway to securely process transactions.
- Payment Receipt: Upon successful payment, a receipt should be generated and displayed to the user.

2.1.4 Reservation Management:

- View Reservations: Users should be able to view their past and upcoming reservations.
- Cancel Reservation: Users can cancel their reservations before a certain cutoff time, and the system should handle refunds according to the cancellation policy.

2.1.5 Admin Management:

- Admin Login:Admins can log in to the system using their credentials.
- Bus Management: Admins can add, update, and delete buses in the system, including specifying details like bus name, type, departure time, and fare.
- Reservation Management: Admins can view reservations for specific buses and dates, including passenger details.



2.2 NON FUNCTIONAL REQUIREMENTS

2.2.1 Security:

-Authentication and Authorization: The system should enforce strong authentication and authorization mechanisms to protect user data and restrict access to admin functionalities.

- Data Encryption: Sensitive data, such as passwords and payment information, should be encrypted both in transit and at rest.

- Input Validation: The system should validate all inputs to prevent SQL injection, XSS, and other common security vulnerabilities.

2.2.2 Performance

- Scalability: The system should be able to scale horizontally to handle increased traffic and user load.
- - Response Time: The application should provide a responsive user experience with page load times under 2 seconds.

2.2.3 Reliability:

- Availability: The system should be highly available, with minimal downtime, and should support failover mechanisms to handle server failures.

- Data Integrity: The system should ensure data consistency and integrity, particularly in transactional operations like bookings and payments.

2.2.4 Usability:

-User Interface (UI): The UI should be intuitive and easy to navigate for both regular users and admins.

- Accessibility: The system should be accessible to users with disabilities, following WCAG (Web Content Accessibility Guidelines) standards.

2.2.5 Maintainability:

- Modular Architecture: The system should be designed in a modular fashion to allow for easy updates and maintenance.
- Code Quality: The codebase should follow best practices in coding standards, be well-documented, and include unit and integration tests.

2.2.6 Compliance:

- Data Privacy:The system should comply with data privacy regulations such as GDPR (General Data Protection Regulation) and other applicable laws.
- Legal Compliance: The system should comply with relevant laws and regulations governing online transactions and data storage.

3. DESIGN

3.1 Database Design

The following table structures depict the database design.

Table1: User Table

Field	Type	Null	Key	Default	Extra
user_id	int	NO	PRI	NULL	auto_increment
email	varchar(255)	NO	UNI	NULL	
first_name	varchar(255)	NO		NULL	
last_name	varchar(255)	NO		NULL	
password	varchar(255)	NO		NULL	
role	varchar(255)	NO		user	

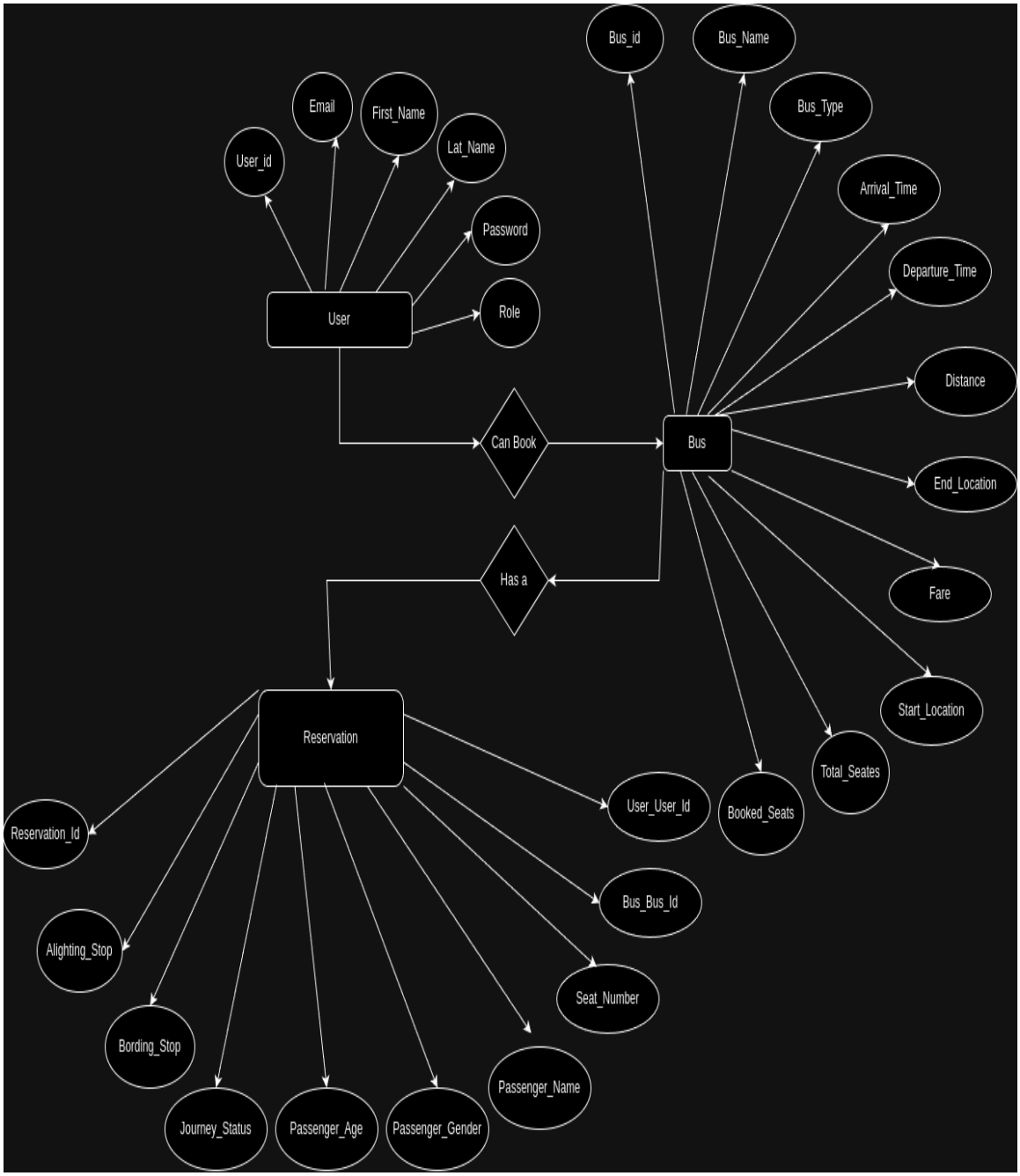
Table2: Bus Table

Field	Type	Null	Key	Default	Extra
bus_id	int	NO	PRI	NULL	auto_increment
bus_name	varchar(255)	NO		NULL	
bus_type	varchar(255)	NO		NULL	
arival_time	datetime	NO		NULL	
departure_time	datetime	NO		NULL	
distance	int	NO		NULL	
end_location	varchar(255)	NO		NULL	
fare	int	NO		NULL	
start_location	varchar(255)	NO		NULL	
total_seates	int	NO		NULL	
booked_seates	int	NO		0	

Table3: Reservation Table

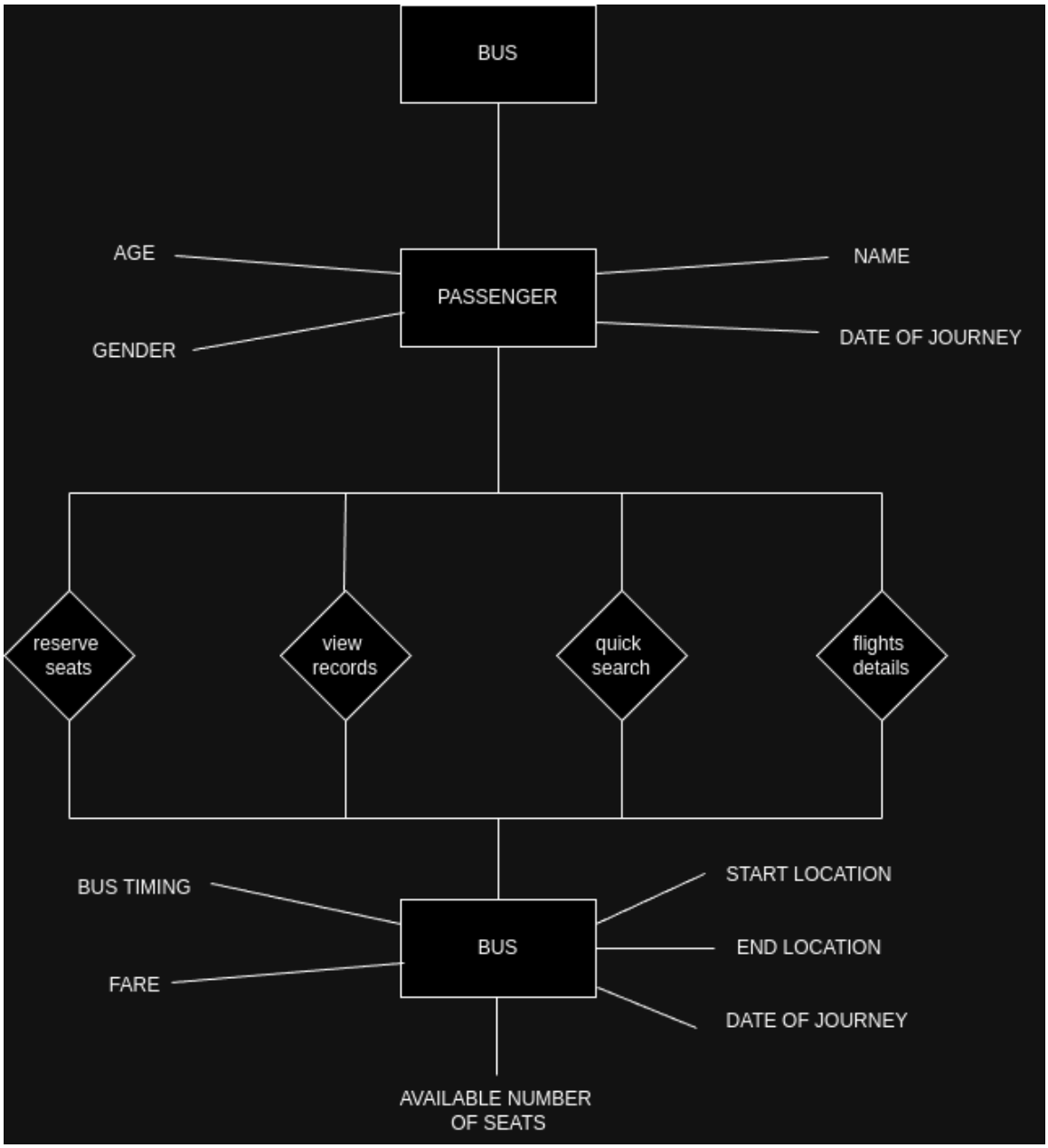
Field	Type	Null	Key	Default	Extra
reservation_id	int	NO	PRI	NULL	auto_increment
alighting_stop	varchar(255)	YES		NULL	
boarding_stop	varchar(255)	YES		NULL	
journey_status	varchar(255)	YES		NULL	
passenger_age	int	NO		NULL	
passenger_gender	varchar(255)	YES		NULL	
passenger_name	varchar(255)	YES		NULL	
seat_number	int	NO		NULL	
bus_bus_id	int	NO	MUL	NULL	
user_user_id	int	YES		NULL	

3.2 E-R Diagram:



3.3 Class Diagram :

3.4 Data Flow Diagram:



4. CODING STANDARDS IMPLEMENTED

4.1 Naming Conventions

Variables: Use camelCase for variable names (e.g., `userDetails`, `totalSeats`).

- Functions: Use camelCase for function names (e.g., `handleLogin`, `getBusDetails`).

- Classes: Use PascalCase for class names (e.g., `User`, `Reservation`).

- Constants: Use UPPER_CASE for constant names (e.g., `MAX_SEATS`, `DEFAULT_ROLE`).

4.2 Indentation and Formatting

Use 2 or 4 spaces for indentation to maintain consistency throughout the codebase.
- Keep line lengths under 80 characters where possible to enhance readability.
- Use line breaks and indentation to separate logical blocks of code.

4.3 Comments and Documentation

Use inline comments to explain complex logic and algorithms.
- Document all functions with JSDoc or equivalent to describe parameters, return values, and exceptions.
- Provide detailed documentation for key modules, classes, and APIs.

4.4 Error Handling

- Implement try-catch blocks for error handling in both synchronous and asynchronous operations.
- Ensure that meaningful error messages are logged and returned to the user, avoiding exposure of sensitive information.

- Use HTTP status codes consistently to indicate success, client errors (4xx), and server errors (5xx).

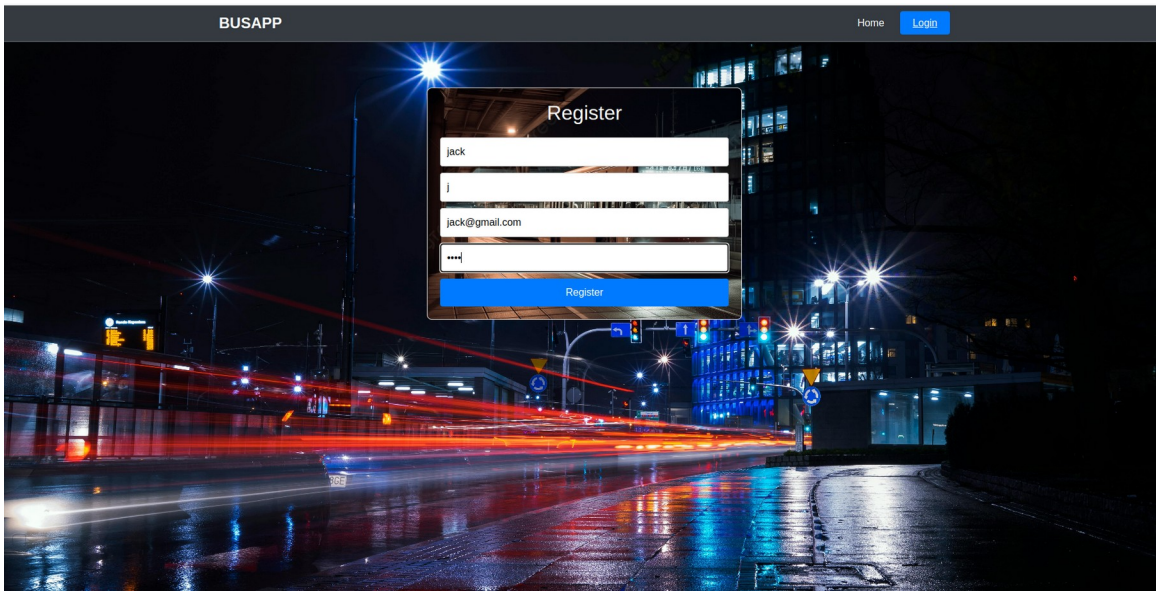
4.5 Version Control

- Use Git for version control and follow a branching strategy (e.g., GitFlow) for development.
- Ensure commits are atomic and have descriptive messages.
- Regularly merge feature branches into the main branch after code review and testing.

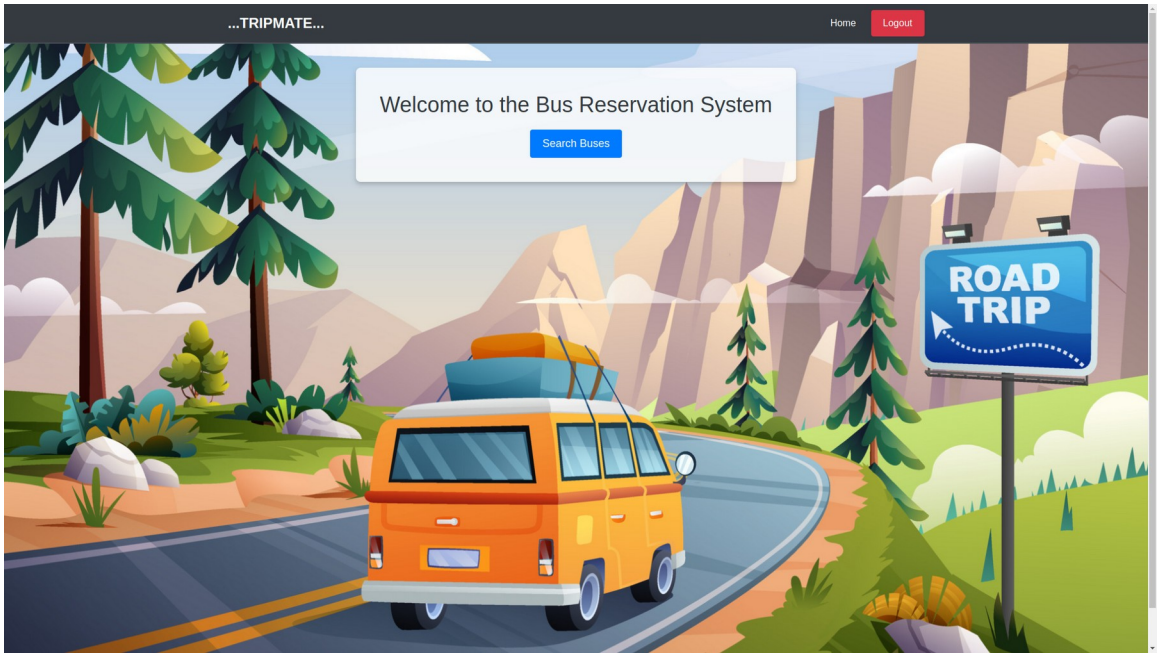
5 . UI Diagram :

Homepage:

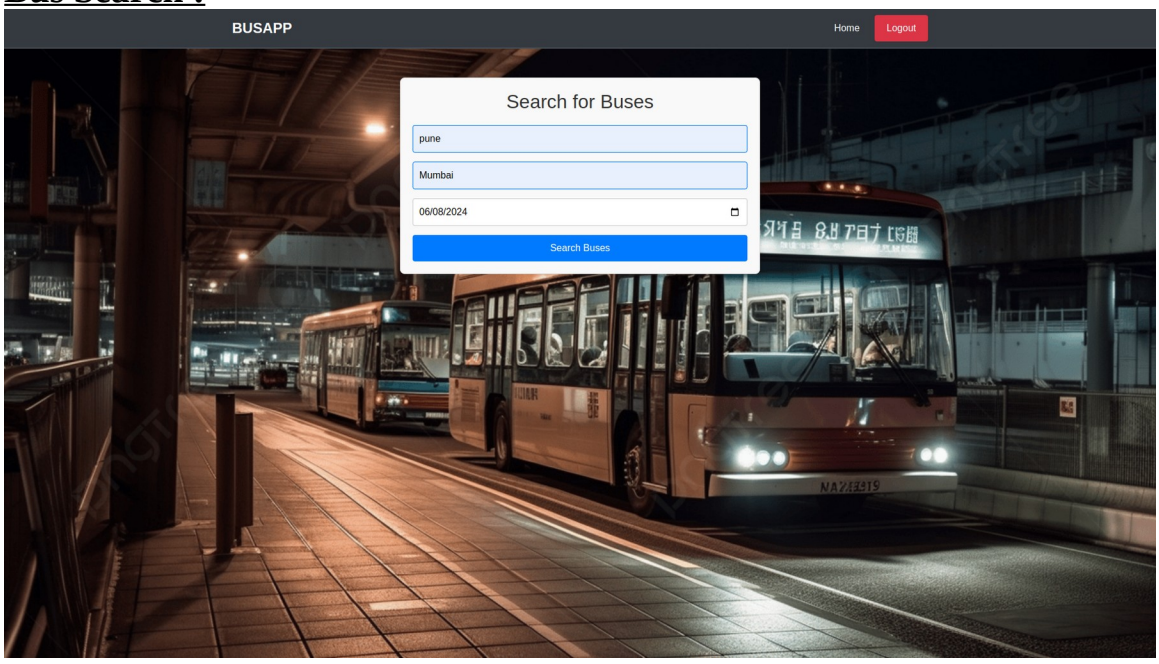
User Registration:



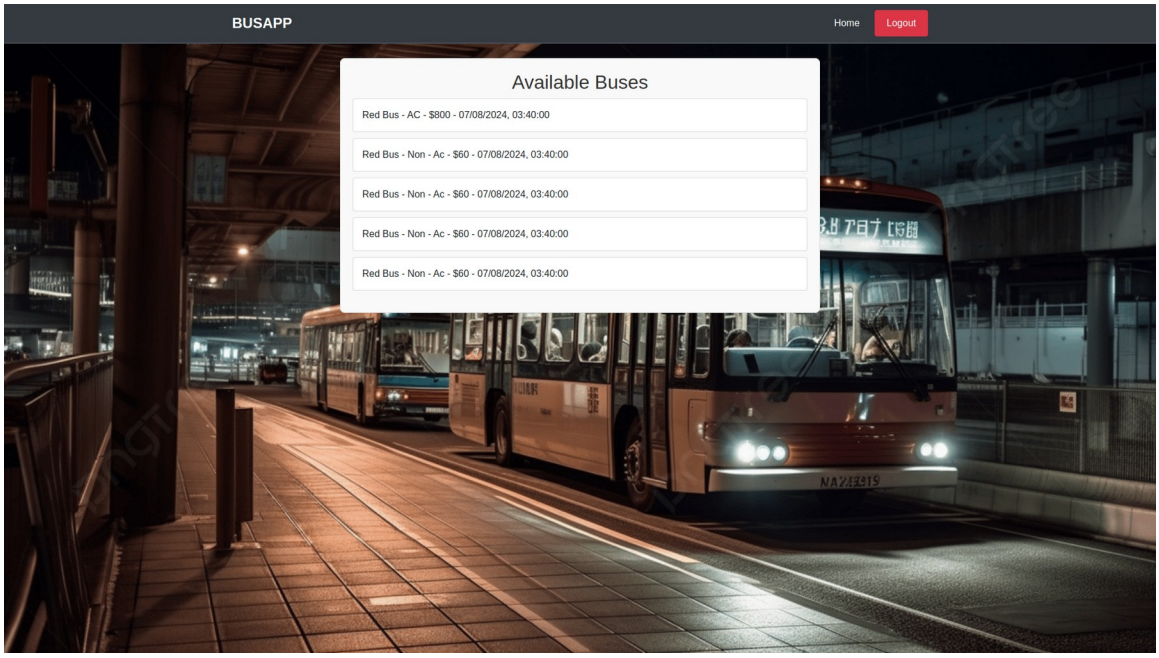
Login :



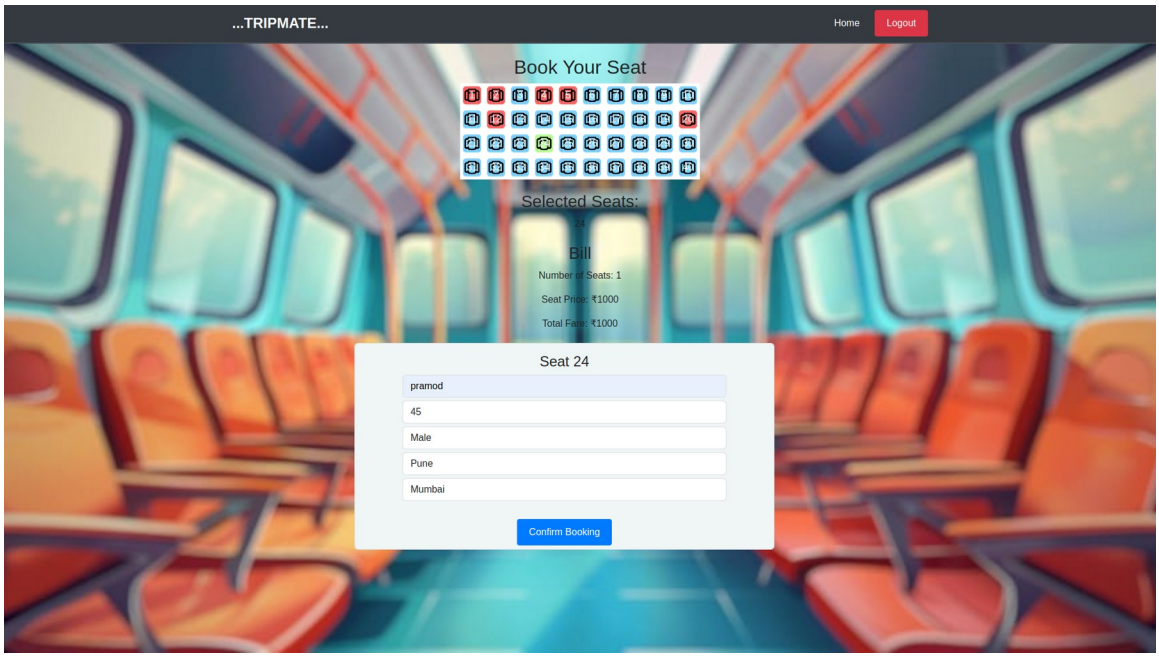
Bus Search :



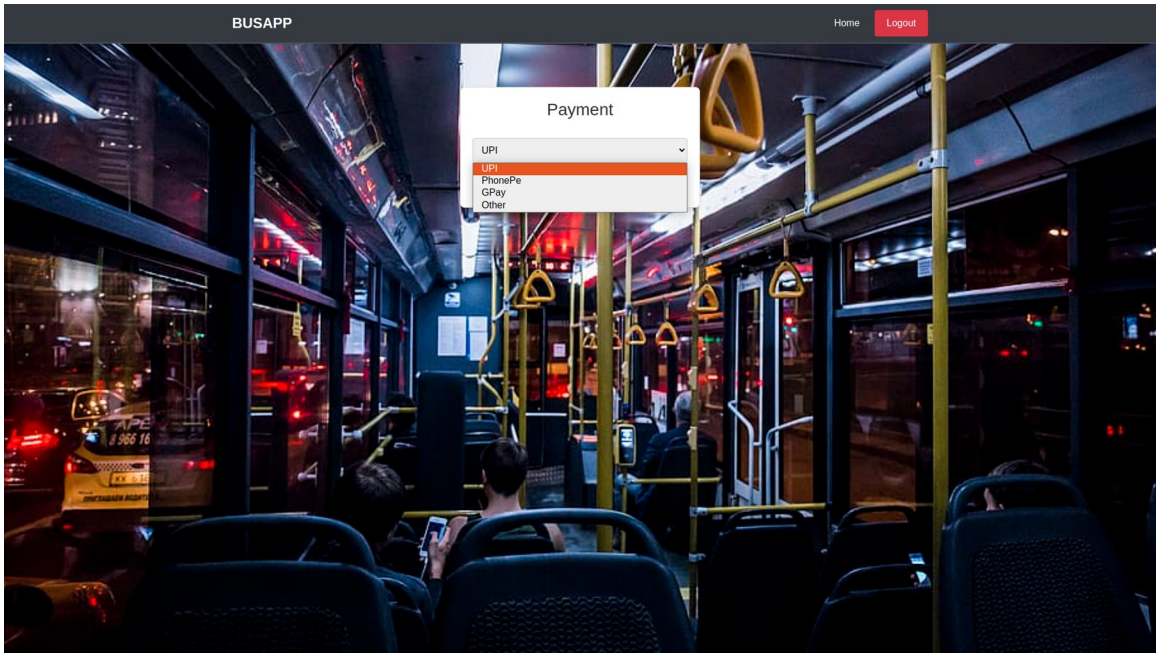
Bus List:



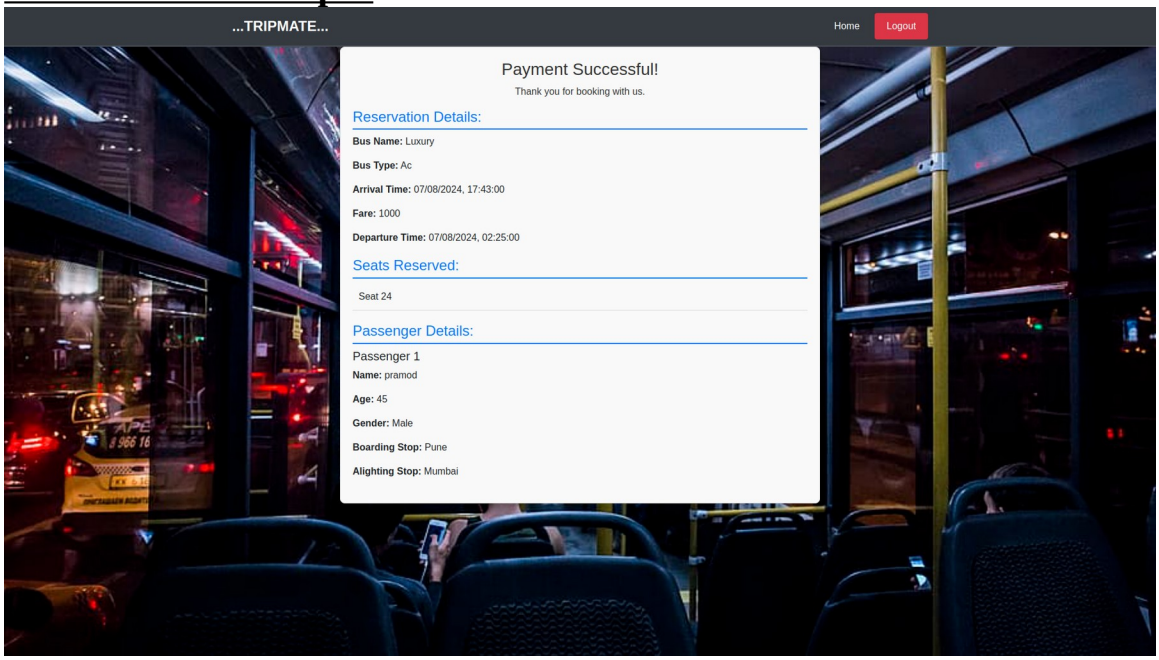
Seat Selection And Passenger Details :



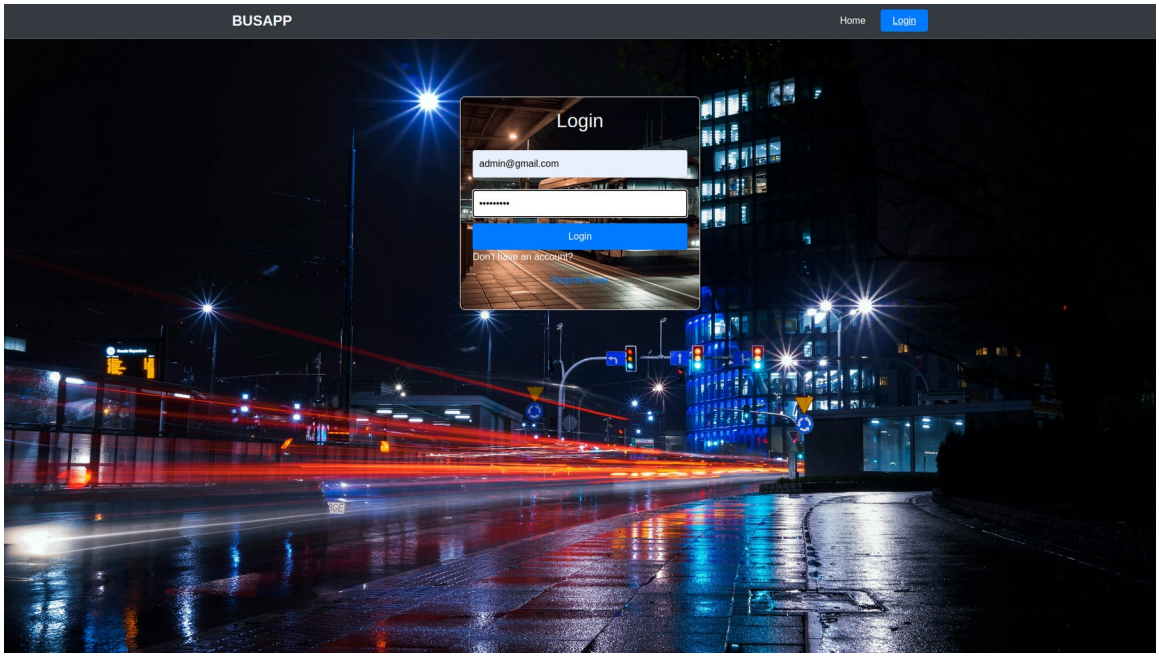
Payment Options:



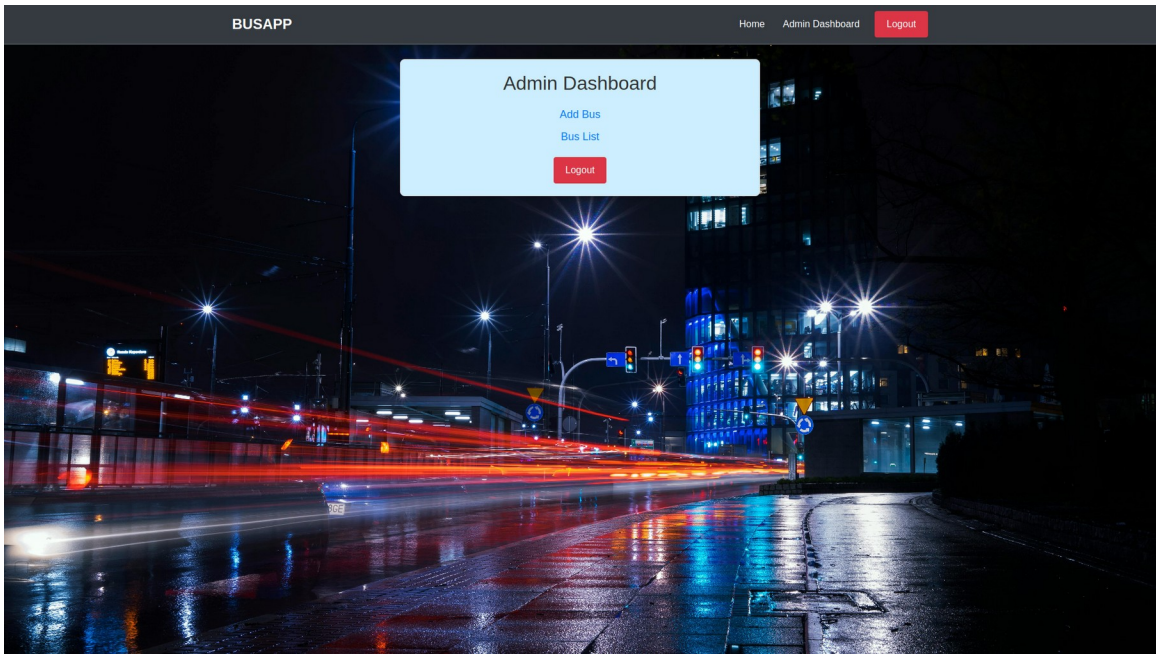
Confirmation Receipt :



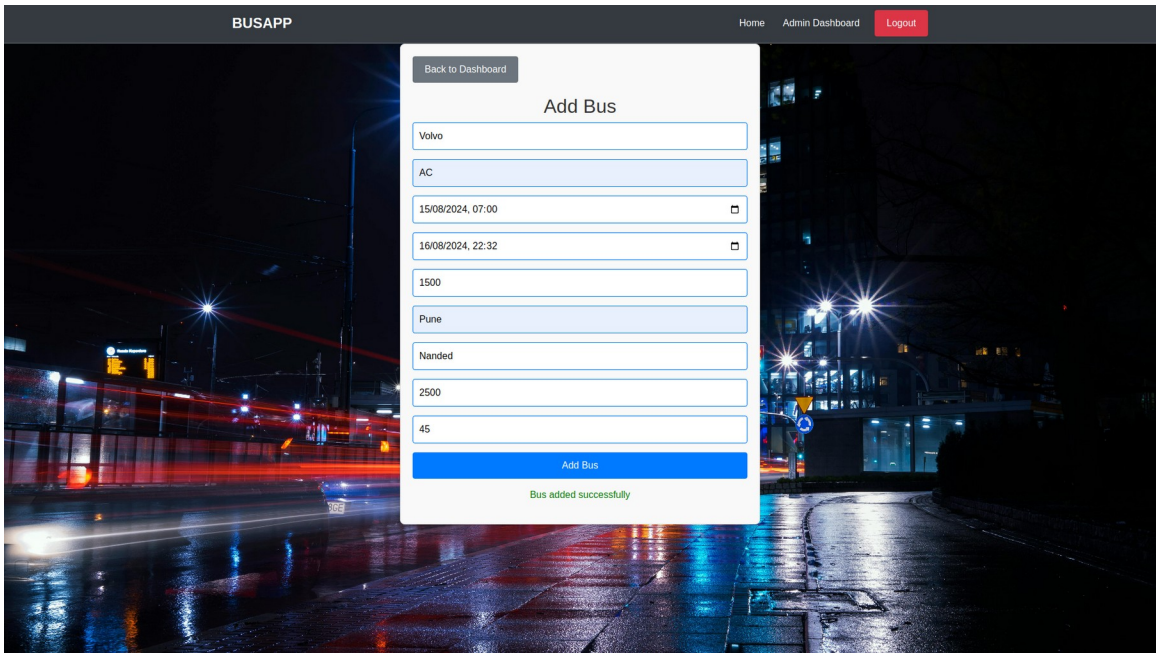
Admin Login :



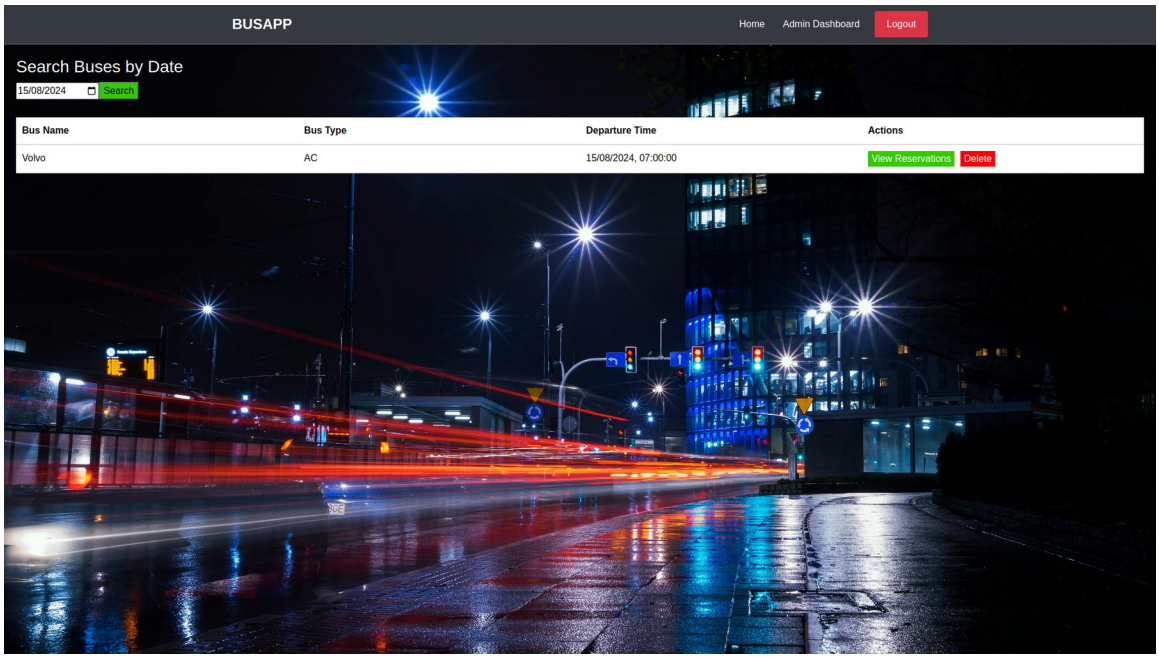
Admin Dashboard :



Add Bus :



View Reservations and Delete Bus :



6. REFERENCES:

Homepage:

1. React Documentation: https://reactjs.org/docs/getting-started.html
2. Node.js Documentation: https://nodejs.org/en/docs/
3. Express Documentation: https://expressjs.com/
4. Sequelize Documentation: https://sequelize.org/master/
5. MySQL Documentation: https://dev.mysql.com/doc/
6. Bootstrap Documentation: https://getbootstrap.com/docs/5.0/getting-started/introduction/
7. REST API Design: https://restfulapi.net/
8. JSDoc Documentation: https://jsdoc.app/
9. Git Documentation: https://git-scm.com/doc

THANK YOU