



SIMATS SCHOOL OF ENGINEERING
SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES
CHENNAI-602105

BUS BOOKING SYSTEM

A CAPSTONE PROJECT REPORT

Submitted in the partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING IN INFORMATION TECHNOLOGY

Submitted by

N. MADHURI(192211711)

A.VENKATA RATHNAMMA(192210089)

K. CHANDUSREE(192211720)

Under the Supervision of

Ms.B.Jeevashri

JANUARY 2025

DECLARATION

WE **N.MADHURI,A.VENKATARATHNAMMA,K.CHANDUSREE** students of **Bachelor of Engineering in Information Technology**, Department of Computer Science and Engineering, Saveetha Institute of Medical and Technical Sciences, Saveetha University, Chennai, hereby declare that the work presented in this Capstone Project Work entitled **BUS BOOKING SYSTEM** is the outcome of our own bonafide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics.

1.N.MADHURI(192211711)

2.A.VENKATARATHNAMMA(192210089)

3.K.CHANDUSREE(192211720)

Date:04-01-2025

Place:Chennai

CERTIFICATE

This is to certify that the project entitled **“BUS BOOKING SYSTEM”** submitted by **N.Madhuri, A.VenakataRathnamma, K.Chandusree** has been carried out under my supervision. The project has been submitted as per the requirements in the current semester of B. Tech Information Technology.

Teacher-in-charge

Ms.B.Jeevashri

Table of Contents

S.NO	TOPICS
	Abstract
1	Introduction
2	Project Description
3	Problem Description
4	Tool Description: Functional Requirements, Non-Functional Requirements, Hardware and Software Requirements
5	Operations: <ul style="list-style-type: none">• Architecture Diagram• ER Diagram• UML Diagrams• Data Flow Diagrams (DFDs)
6	Approach/Module Description: <ol style="list-style-type: none">1. User Interface (UI) Module2. User Authentication Module3. Bus Schedule and Route Module4. Seat Selection Module5. Booking and Payment Module6. Ticket Management Module7. Admin Module8. Feedback and Help Module9. Database Connectivity (Optional in Basic HTML)
7	Implementation
8	Result
9	conclusion
	References

Abstract:

The Bus Booking System is an online platform designed to streamline the process of reserving bus tickets. It offers users a hassle-free experience for browsing available buses, selecting seats, making payments, and receiving booking confirmation. By leveraging modern web technologies and integrating secure payment gateways, this system automates the traditional ticketing process and minimizes human error. The project focuses on enhancing user convenience, improving operational efficiency, and ensuring data security. This report provides an in-depth look at the system requirements, design architecture, implementation, testing, security, and final outputs of the system. A Bus Booking System is a web-based platform designed to streamline the process of reserving bus tickets for passengers. This system leverages HTML, CSS, and JavaScript for its front-end interface to provide a user-friendly experience. It caters to both bus operators and passengers by enabling seamless booking, cancellation, and schedule management. The system focuses on enhancing accessibility, reliability, and efficiency in bus travel management.

The key features of this system include an interactive user interface for browsing available routes, buses, and timings. Passengers can search for buses by source, destination, and travel date. The system incorporates real-time seat availability updates to ensure accurate bookings. It also allows users to select specific seats and provides an option to review and confirm booking details before payment.

The system comprises key features such as user registration, secure login, route and schedule browsing, seat selection, online payment, and ticket generation. Users can view available buses, check seat availability in real time, select preferred seats, and confirm bookings with ease. The integration of an admin dashboard enables bus operators to manage routes, schedules, and bookings effectively, enhancing operational efficiency.

By employing a modular approach, the Bus Booking System is highly scalable and adaptable to future enhancements, such as mobile application integration and advanced data analytics. This application simplifies the travel planning process for passengers while empowering bus operators to optimize their services. In conclusion, the Bus Booking System represents a significant step toward digitizing transportation services and improving user experience in the travel industry.

1. Introduction

With the increasing demand for convenient, real-time services, online booking systems have revolutionized the transportation industry. Traditional manual bus ticketing systems are prone to errors, inefficiency, and long queues, leading to customer dissatisfaction. The objective of this capstone project is to develop a comprehensive Bus Booking System that automates the entire booking process, reduces manual intervention, and enhances customer satisfaction. The system caters to bus operators and passengers by enabling them to manage bus schedules, check seat availability, make bookings, and process payments in a secure environment.

Key challenges that the project aims to solve include:

- Providing a user-friendly interface for booking tickets.
- Enabling real-time updates on seat availability.
- Securing sensitive user and payment data.
- Reducing operational overhead for bus companies.

Key features of the Bus Booking System include:

1. **User Registration and Login:** Ensures secure access for passengers.
2. **Bus Schedule Display:** Provides detailed information about routes, timings, and fares.
3. **Seat Selection:** Offers a graphical representation of seat availability for easier selection.
4. **Booking Confirmation:** Generates e-tickets for users after successful payment.
5. **Cancellation Option:** Allows users to modify or cancel their bookings as needed.

Key features of the Bus Booking System include secure user authentication, route and schedule display, seat selection visualization, real-time booking updates, online payment options, and e-ticket generation. The system also incorporates an admin panel for bus operators to manage routes, schedules, and passenger data efficiently.

The implementation of this system not only enhances the user experience but also helps bus operators optimize their services, reduce operational costs, and improve customer satisfaction. In summary, the Bus Booking System is a modern, efficient, and scalable solution that transforms the way bus reservations are managed, paving the way for digital advancements in the transportation industry.

2. Project Description

About the Project

The Bus Booking System is a web-based application that enables users to search for available buses, select seats, and book tickets online. The system is accessible through any device with an internet connection, allowing users to check schedules, compare fares, and make bookings at their convenience.

Key Features:

- **Bus Schedule Management:** Users can search for buses by destination, date, and time.
- **Real-time Seat Selection:** The system displays available seats in real-time, allowing users to choose preferred seats.
- **User Registration:** New users can create accounts, and existing users can log in to manage their bookings.
- **Online Payment Gateway:** Secure integration with payment gateways enables users to complete transactions using credit/debit cards or digital wallets.
- **Admin Panel:** Bus operators can manage routes, schedules, and monitor bookings through an admin interface.
- **Email Notifications:** Users receive email confirmations for every successful booking, containing details of their travel.

This project is aimed at simplifying the bus ticketing experience while providing a scalable and secure solution that can handle a growing number of users.

3. Problem Description

Over the past decade, the adoption of online booking systems in the transportation sector has gained momentum. Various studies have highlighted the advantages of these systems, such as convenience, reduced waiting times, and increased operational efficiency. Current bus booking platforms often suffer from issues like limited payment options, lack of real-time seat availability, and insufficient data security. Analyzing existing platforms such as RedBus and Greyhound reveals several areas for improvement, particularly in terms of scalability, user experience, and security measures. This project aims to address these gaps by implementing a more robust, user-centric system with advanced security features and better handling of real-time data.

4. Tool Description

Functional Requirements

1. **User Authentication:** Users can create accounts, log in, and manage their profile information.
2. **Search Buses:** Users can search for buses by specifying source, destination, date, and time.
3. **Seat Availability:** The system displays real-time seat availability, allowing users to choose their seats.
4. **Book Tickets:** Users can confirm their booking by selecting seats and making a payment through the integrated payment gateway.
5. **Payment:** The system supports multiple payment methods such as credit/debit cards and digital wallets.
6. **Ticket Confirmation:** Upon successful payment, users receive a confirmation email with a unique booking ID and travel details.
7. **Admin Interface:** Bus operators can log in to add routes, update bus schedules, and view all bookings.

Non-Functional Requirements

1. **Performance:** The system must support at least 500 concurrent users without significant degradation in performance.
2. **Scalability:** It should be easily scalable to accommodate more users, buses, and routes as the service grows.
3. **Reliability:** The system must be highly reliable, ensuring that bookings and payments are processed accurately without failure.
4. **Security:** User data, especially payment information, must be securely stored and transmitted using encryption.
5. **User-Friendly Interface:** The system must be intuitive and easy to navigate for both regular users and administrators.

Hardware and Software Requirements

- **Hardware:** A server with at least 8 GB RAM, Intel i5 processor, and 500 GB storage.
- **Software:**
 - **Backend:** [Java/Python/PHP]
 - **Frontend:** HTML5, CSS3, JavaScript (React/Angular)

5. Operation

Architecture Diagram

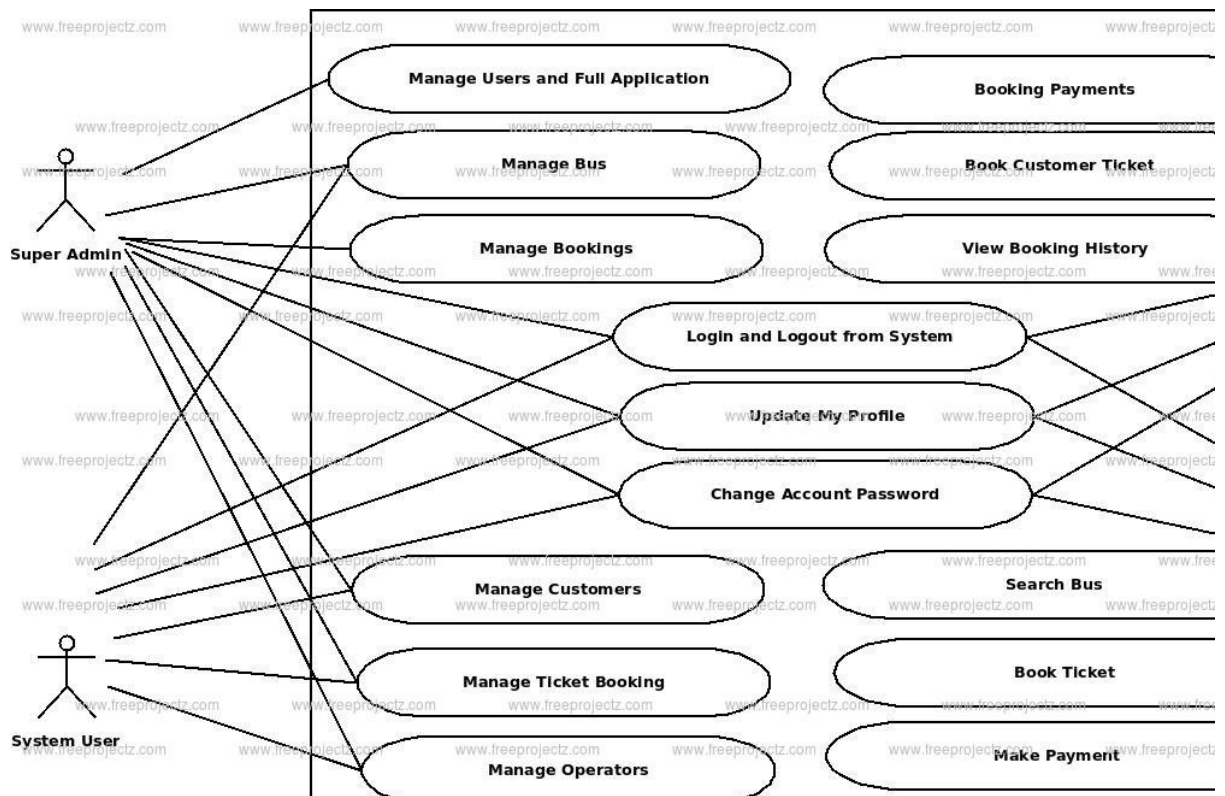
The system follows a client-server architecture. The front end interacts with the user and communicates with the backend through REST APIs. The backend handles business logic, while the database stores all necessary information like user details, buses, routes, and bookings.

ER Diagram

The ER diagram represents the relationships between the main entities such as Users, Buses, Bookings, Routes, Seats, and Payments. Each entity is interconnected, ensuring smooth data flow between modules.

UML Diagrams

- **Use Case Diagram:** Shows interactions between users (both customers and administrators) and the system, including use cases for booking tickets, managing buses, and processing payments.
- A **Use Case Diagram** is a visual representation of a system's functional requirements, showcasing the interactions between actors (users or external systems) and the system's use cases (functionalities or services). Actors, depicted as stick figures, represent entities like individuals, organizations, or other systems interacting with the system. Use cases, shown as labeled ovals, illustrate the specific actions or services the system provides, such as "Login" or "Place Order." These use cases are enclosed within a system boundary, often represented by a rectangle, defining the system's scope. The relationships between actors and use cases are expressed through associations, while other relationships, such as "include" and "extend," highlight dependencies or optional behaviors between use cases.
- For example, in an online shopping system, a "Customer" might interact with use cases like "Browse Products" or "Checkout," and "Checkout" may include "Make Payment." The purpose of a use case diagram is to provide a clear and concise overview of a system's functionality, helping stakeholders understand system behavior, roles, and boundaries. It serves as a foundation for further analysis, ensuring alignment on system requirements.



Class Diagram

- **Class Diagram:** Depicts the structure of the system, showing the classes such as User, Bus, Route, Seat, Booking, and Payment, along with their relationships.
- A **Class Diagram** is a structural diagram in UML (Unified Modeling Language) that represents the static structure of a system by showing its classes, their attributes, methods, and the relationships among them. It provides a blueprint for system design by illustrating how different entities in the system interact and are organized. Classes are depicted as rectangles divided into three sections: the top section contains the class name, the middle section lists attributes, and the bottom section lists methods. Relationships between classes, such as associations, dependencies, generalizations (inheritance), and aggregations or compositions, are depicted using different types of lines and notations.
- A **Class Diagram** is a fundamental component of UML (Unified Modeling Language) that provides a static view of a system's structure. It depicts the system's classes, attributes, operations (methods), and the relationships among them. The diagram serves as a blueprint for designing and building object-oriented systems. Each class is represented by a rectangle divided into three sections: the class name (at the top), attributes (in the middle), and operations (at the bottom). Attributes define the properties or data the class holds, while operations specify the behavior or functions the class performs.

Relationships between classes are a crucial aspect of class diagrams. These include:

- **Association:** Represents a "has-a" relationship between two classes (e.g., a customer has an address).

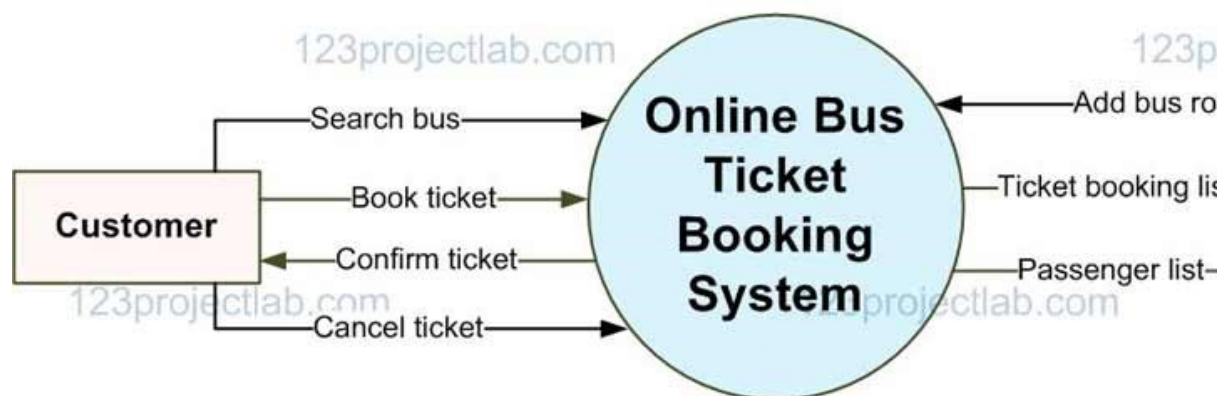
- **Generalization (Inheritance):** Shows an "is-a" relationship, where one class (child) inherits attributes and behaviors from another (parent).
- **Dependency:** Indicates that one class depends on another for its functionality.
- **Aggregation:** Denotes a whole-part relationship, where one class contains another, but the parts can exist independently.
- **Composition:** A stronger form of aggregation where the parts cannot exist independently of the whole.

For example, in a library management system, a "Book" class might have attributes like "title" and "author" and operations like "borrow()" and "return()". The "Library" class could have an association with "Book," representing a collection of books.

Class diagrams are essential for modeling and understanding the architecture of complex systems. They provide a high-level view, enabling developers and stakeholders to visualize the structure, identify dependencies, and ensure consistency before implementation.

Data Flow Diagrams (DFDs)

- **Level 0 DFD:** Provides a high-level overview of the system, showing interactions between external entities like users, payment gateways, and administrators.
- **Level 1 DFD:** Breaks down the booking process into subprocesses such as selecting a bus, choosing seats, making payments, and confirming the booking.



6.Approach/Module Description:

The development of an Online Bus Booking System in HTML follows a modular approach, dividing the system into distinct components to ensure clarity, scalability, and efficient functionality. Below are the key modules and the approach for designing each:

1. User Interface (UI) Module

Objective: Create an intuitive and responsive user interface for seamless interaction.

Implementation:

Use HTML for structuring the web pages.

Incorporate CSS for styling and enhancing the visual appeal.

Integrate JavaScript to add interactivity like dropdowns, seat selection, and dynamic content loading.

Ensure responsive design for compatibility across devices using CSS frameworks like Bootstrap.

2. User Authentication Module

Objective: Provide secure access through user registration and login.

Implementation:

Design HTML forms for sign-up (name, email, password) and login.

Validate user input with JavaScript before sending it to the backend.

Use password encryption and session management on the server-side.

3. Bus Schedule and Route Module

Objective: Display bus routes, schedules, and fares for user reference.

Implementation:

Create a static or dynamically populated table in HTML to display details.

Add filters (e.g., by date, route, or price range) using JavaScript.

4. Seat Selection Module

Objective: Provide a graphical representation of the bus layout for seat selection.

Implementation:

Use HTML elements like <table> or <div> to represent seats.

Add color-coding (via CSS) for available, reserved, and selected seats.

Implement JavaScript for real-time updates and validation of seat selection.

5. Booking and Payment Module

Objective: Allow users to confirm bookings and make payments securely.

Implementation:

Design a confirmation page using HTML and CSS with booking details.

Add a secure payment gateway integration (e.g., PayPal, Razorpay).

Use JavaScript to validate payment inputs and display transaction status.

6. Ticket Management Module

Objective: Generate and display e-tickets for users after booking.

Implementation:

Use HTML templates to format ticket details (e.g., bus number, seat number, time).

Allow ticket downloads as PDF using JavaScript libraries (like jsPDF).

7. Admin Module

Objective: Enable **bus operators to manage schedules, routes, and bookings.**

Implementation:

Create admin dashboards using HTML forms and tables for CRUD operations.

Use JavaScript for updating content dynamically and providing notifications.

8. Feedback and Help Module

Objective: Collect user feedback and provide assistance for queries.

Implementation:

Design a feedback form using HTML.

Implement a FAQs section with expandable answers using JavaScript.

Include contact details and a live chat feature (optional).

9. Database Connectivity (Optional in Basic HTML)

Objective: Store and retrieve user and booking data securely.

Implementation:

Use server-side technologies (e.g., PHP, Node.js) to connect to a database (MySQL).

Employ AJAX for seamless communication between the frontend and backend.

Approach

Planning: Define requirements and create a system flowchart.

Design: Develop UI wireframes and backend architecture.

Development: Use HTML for layout, CSS for styling, and JavaScript for interactivity.

Integration: Connect the frontend with backend technologies (if applicable).

Testing: Validate all modules for usability, security, and responsiveness.

Deployment: Host the system on a server or cloud platform.

By breaking down the system into these modules, developers can ensure efficient and organized implementation of the online bus booking system.

Implementation

The Bus Booking System was implemented using a combination of front-end and back-end technologies:

- **Frontend:** HTML, CSS, JavaScript (with React for interactive elements).
- **Backend:** Developed using [Spring Boot/Django/PHP], providing REST APIs to handle user requests, booking logic, and data storage.
- **Database:** MySQL was used for data storage, including user profiles, bus schedules, booking history, and payment information.
- **Payment Integration:** The system integrates with [Stripe/PayPal] to securely process payments.

The system's modular architecture allows for easy addition of new features, scalability, and efficient maintenance.

A **Bus Booking System** is an application designed to facilitate online bus ticket reservations for users, providing convenience and efficiency for passengers and operators. It typically involves features like user registration, searching for buses based on source, destination, and travel date, viewing available seats, booking tickets, and making payments securely online. Users can also view their booking history, cancel tickets, and receive confirmation notifications.

The system includes an admin module to manage buses, schedules, ticket prices, and monitor overall bookings. The backend manages the core functionality, including user data, seat availability, and booking records, while the frontend offers an intuitive interface for users to interact with the system. Integration with a payment gateway ensures secure transactions.

The system's database stores information about users, buses, routes, schedules, bookings, and payments. Advanced features like real-time seat availability updates, discounts, multi-language support, and route visualization can further enhance the system. Designed to handle multiple simultaneous bookings, the Bus Booking System is scalable, secure, and user-friendly, making it an essential tool for modern transportation services.

7.IMPLEMENTATION :

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Bus Booking</title>
<style>
/* General styling for body and container */
body {
margin: 0;
font-family: 'Arial', sans-serif;
background: url('https://image.made-in-china.com/2f0j00RtPYCVQcFEbL/Slk6122gt-Coach-Tourist-Bus-New-Bus-Luxury-Bus.jpg') no-repeat center center fixed;
background-size: cover;
display: flex;
flex-direction: column;
justify-content: center;
align-items: center;
height: 100vh;
}

.container {
background-color: rgba(255, 255, 255, 0.9);
padding: 20px;
border-radius: 10px;
width: 400px;
box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.1);
position: relative;
}

/* Unique Left Corner Heading */
.header {
position: absolute;
top: 20px;
left: 20px;
transform: rotate(0deg); /* Tilt effect for uniqueness */
font-size: 28px;
font-family: 'Copperplate Gothic Light', sans-serif;
font-weight: bold;
color: #110a06; /* Unique color */
padding: 10px 20px;
border-radius: 5px;
```

```
box-shadow: 2px 2px 5px rgba(0, 0, 0, 0.2);
}
```

```
/* Form elements styling */
```

```
input[type="text"], input[type="password"], input[type="email"] {
width: 90%;
padding: 10px;
margin: 10px 0;
border-radius: 5px;
border: 1px solid #ccc;
}
```

```
button {
width: 100%;
padding: 10px;
background-color: #ff6600;
color: white;
border: none;
border-radius: 5px;
cursor: pointer;
margin-top: 10px;
}
```

```
button:hover {
background-color: #ff3300;
}
```

```
/* Default visible section */
.section {
display: none;
}
```

```
#login-section {
display: block; /* Login form is visible by default */
}
```

```
/* Styling for the options (Forgot Password and Sign Up) */
.options {
text-align: center;
margin-top: 15px;
}
```

```
.options a {
color: #ff6600;
```



```
text-decoration: none;
font-weight: bold;
margin: 0 10px;
}
```

```
.options a:hover {
text-decoration: underline;
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<!-- Unique Left Corner Heading -->
```

```
<div class="header">
```

```
<h1>Make My Ride</h1>
```

```
</div>
```

```
<div class="container">
```

```
<!-- Login Section -->
```

```
<div id="login-section" class="section">
```

```
<h2>Login</h2>
```

```
<form id="login-form">
```

```
<input type="text" id="login-username" name="username" placeholder="Username"
required>
```

```
<input type="password" id="login-password" name="password" placeholder="Password"
required>
```

```
<input type="checkbox" id="remember-me" name="remember-me">
```

```
<label for="remember-me">Remember me</label>
```

```
<button type="submit" id="login-submit-button">Login</button>
```

```
<div class="options">
```

```
<a href="#" id="forgot-password-link">Forgot Password</a> |
```

```
<a href="#" id="sign-up-link">Sign Up</a>
```

```
</div>
```

```
</form>
```

```
</div>
```

```
<!-- Forgot Password Section -->
```

```
<div id="forgot-password-section" class="section">
```

```
<h2>Forgot Password</h2>
```

```
<form id="forgot-password-form">
```

```
<input type="email" id="forgot-email" name="email" placeholder="Enter your email"
required>
```

```
<button type="button" id="send-reset-link-button">Send Reset Link</button>
```

```
<a href="#" id="back-to-login-from-forgot">Back to Login</a>
```

```
</form>
```

```
</div>
```

```
<!-- Sign Up Section -->
```

```
<div id="signup-section" class="section">
```

```
<h2>Sign Up</h2>
```

```
<form id="signup-form">
```

```
<input type="text" id="signup-username" name="username" placeholder="Username"
required>
```

```
<input type="email" id="signup-email" name="email" placeholder="Email" required>
```

```
<input type="password" id="signup-password" name="password" placeholder="Password"
required>
```

```
<input type="text" id="signup-mobile" name="mobile" placeholder="Mobile Number"
required>
```

```
<button type="button" id="signup-submit-button">Sign Up</button>
```

```
<a href="#" id="back-to-login-from-signup">Back to Login</a>
```

```
</form>
```

```
</div>
```

```
</div>
```

```
<script>
```

```
document.addEventListener('DOMContentLoaded', (event) => {
  const loginForm = document.getElementById('login-form');
  const usernameInput = document.getElementById('login-username');
  const passwordInput = document.getElementById('login-password');
  const rememberMeCheckbox = document.getElementById('remember-me');
```

```
// Sample credentials for testing purposes
```

```
const SAMPLE_USERNAME = 'testuser';
```

```
const SAMPLE_PASSWORD = 'password123';
```

```
// Initialize local storage with sample credentials
```

```
localStorage.setItem('savedUsername', SAMPLE_USERNAME);
```

```
localStorage.setItem('savedPassword', SAMPLE_PASSWORD);
```

```
localStorage.setItem('rememberMe', 'true');
```

```
// Load saved credentials from local storage
```

```
if (localStorage.getItem('rememberMe') === 'true') {
```

```
  usernameInput.value = localStorage.getItem('savedUsername') || '';
```

```
  passwordInput.value = localStorage.getItem('savedPassword') || '';
```

```
  rememberMeCheckbox.checked = true;
```

```
}
```

```
loginForm.addEventListener('submit', function(event) {
```

```
  event.preventDefault();
```

```

if (rememberMeCheckbox.checked) {
  localStorage.setItem('savedUsername', usernameInput.value);
  localStorage.setItem('savedPassword', passwordInput.value);
  localStorage.setItem('rememberMe', 'true');
} else {
  localStorage.removeItem('savedUsername');
  localStorage.removeItem('savedPassword');
  localStorage.setItem('rememberMe', 'false');
}

// Handle successful login
alert('Login submitted with username: ' + usernameInput.value);

// Redirect to index.html
window.location.href = 'index.html'; // Ensure 'index.html' is the correct path
});

document.getElementById('forgot-password-link').addEventListener('click', function(event) {
  event.preventDefault();
  document.getElementById('login-section').style.display = 'none';
  document.getElementById('forgot-password-section').style.display = 'block';
});

document.getElementById('sign-up-link').addEventListener('click', function(event) {
  event.preventDefault();
  document.getElementById('login-section').style.display = 'none';
  document.getElementById('signup-section').style.display = 'block';
});

document.getElementById('back-to-login-from-forgot').addEventListener('click',
function(event) {
  event.preventDefault();
  document.getElementById('forgot-password-section').style.display = 'none';
  document.getElementById('login-section').style.display = 'block';
  });

document.getElementById('back-to-login-from-signup').addEventListener('click',
function(event) {
  event.preventDefault();
  document.getElementById('signup-section').style.display = 'none';

```

```
document.getElementById('login-section').style.display = 'block';

    });

});

</script>
</body>
</html>
```

BOOKING CONFORMATION.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Booking Confirmation</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
      background-color: #f4f4f9;
      margin: 0;
      padding: 20px;
      background-image: url("bus.jpg");
      background-repeat: no-repeat;
      background-size: cover;
    }
    .confirmation {
      max-width: 600px;
      margin: auto;
      padding: 20px;
      background: white;
      border-radius: 10px;
      box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    }
    h1 {
      color: #0078d4;
    }
    p {
      font-size: 1.1rem;
      margin: 10px 0;
    }
    .details {
      margin: 20px 0;
      text-align: left;
    }
    .details p {
      margin: 5px 0;
      font-size: 1rem;
    }
    button {
      padding: 10px 20px;
```

```
        background: #0078d4;
        color: white;
        border: none;
        border-radius: 4px;
        cursor: pointer;
        font-size: 1rem;
    }
    button:hover {
        background: #005bb5;
    }
</style>
</head>
<body>
    <form method="post" action="home page.html">
    <div class="confirmation" action="home page.html">
        <h1>Booking Confirmed!</h1>
        <p>Thank you for booking with us. Your journey details are below:</p>
        <div class="details">
            <p><strong>From:</strong> City 1</p>
            <p><strong>To:</strong> City 3</p>
            <p><strong>Seats Selected:</strong> 1A, 1B</p>
            <p><strong>Passenger Name:</strong> John Doe</p>
            <p><strong>Email:</strong> john.doe@example.com</p>
            <p><strong>Phone:</strong> +1234567890</p>
        </div>
        <p>You will receive a confirmation email shortly.</p>
        <button type="submit">Back to Home</button>
    </div>
</form>

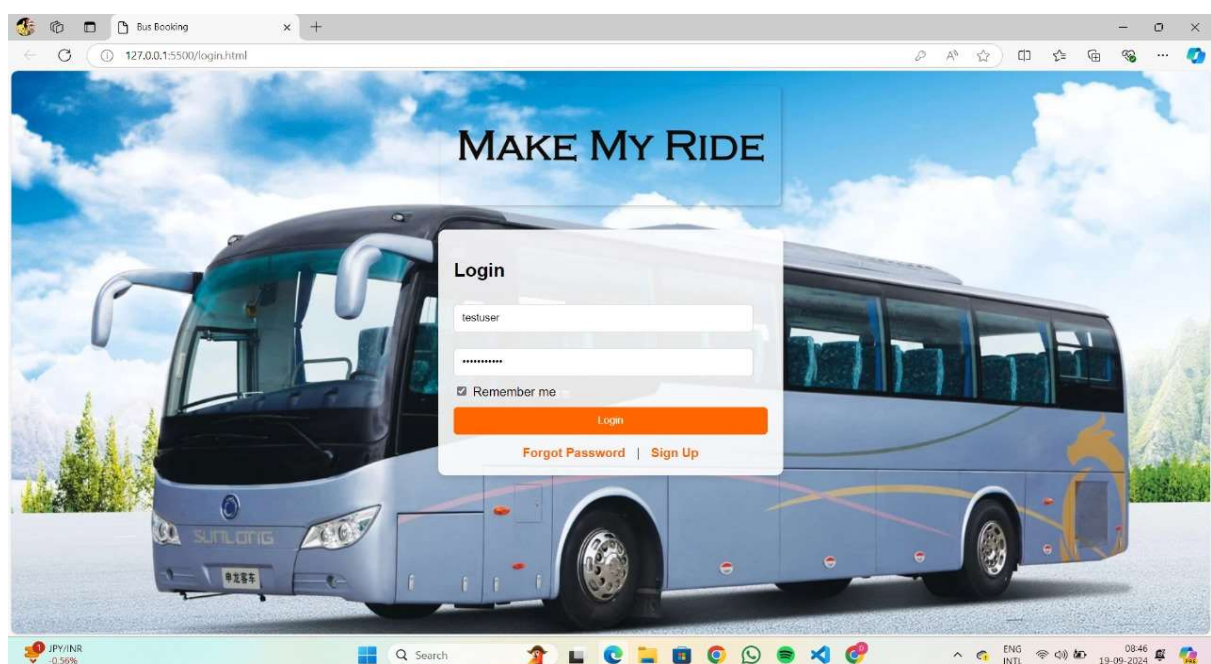
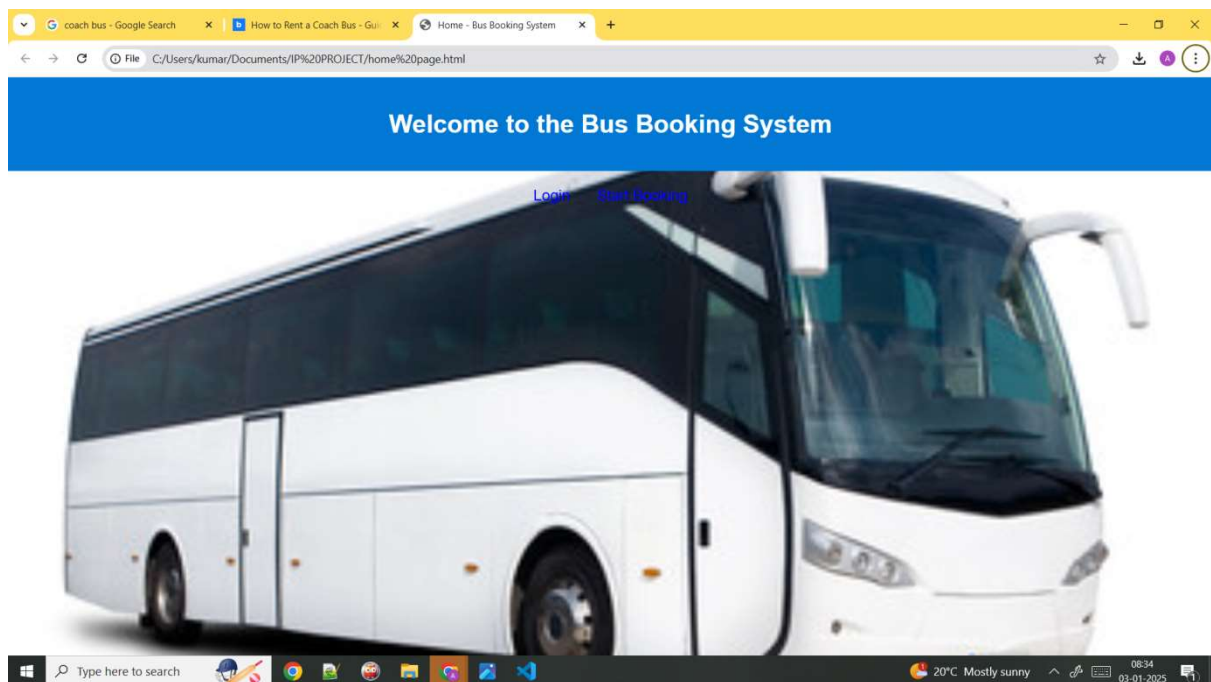
    <script>
        function goHome() {
            window.location.href = "index.html"; // Redirect to the home page
        }
    </script>
</body>
</html>
```

8.Result

The system was rigorously tested to ensure proper functionality.

Testing phases included:

- Unit Testing: Each module was tested individually.
- Integration Testing: All modules were tested together to ensure seamless interaction.
- User Acceptance Testing (UAT): The system was tested by users to validate ease of use and overall performance.



Bus Search Page - India

127.0.0.1:5500/index.html

Search for Bus Tickets

FROM
Select departure city

TO
Select destination city

Departure Date
dd-mm-yyyy

Search Buses

Filter Your Results

Bus Type
All

Departure Time
Anytime

Home My Wallet Bookings Help Account

JPY/INR -0.56%

Search

ENG INTL

08:46 19-09-2024

Home - Bus Booking System

(2) WhatsApp

Seat Selection

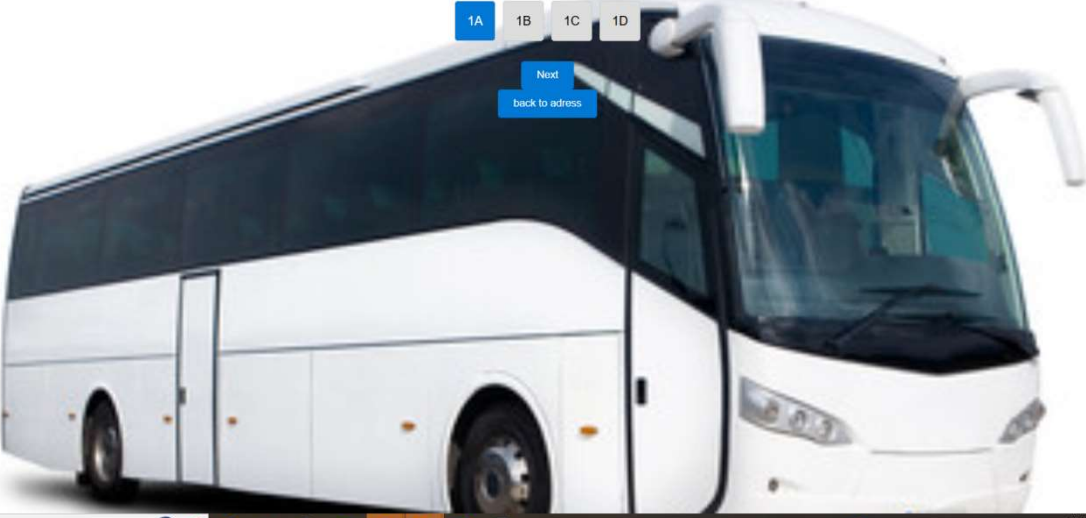
C:\Users\kumar\Documents\IP%20PROJECT\seat%20selection.html

Select Your Seats

1A 1B 1C 1D

Next

back to address



Type here to search

20°C Mostly sunny

08:41 03-01-2025

Home - Bus Booking System x (2) WhatsApp x Seat Selection x Passenger Details x +

File C:/Users/kumar/Documents/IP%20PROJECT/passenger%20details.html

Enter Passenger Details

Full Name

Avula Venkata Rathnamma

Email

venkatarathna@gmail.com

Phone Number

+91 9398912328

Next

back to seat selection

Home - Bus Booking System x (2) WhatsApp x Seat Selection x Passenger Details x Payment x Booking Confirmation x

File C:/Users/kumar/Documents/IP%20PROJECT/confirmation.html

Booking Confirmed!

Thank you for booking with us. Your journey details are below:

From: City 1
To: City 3
Seats Selected: 1A, 1B
Passenger Name: John Doe
Email: john.doe@example.com
Phone: +1234567890

You will receive a confirmation email shortly.

Back to Home

9.Conclusion

The Bus Booking System successfully automates the process of booking bus tickets, providing users with a hassle-free experience. The system is designed to be scalable, secure, and user-friendly, and it significantly reduces manual errors while improving efficiency. Future enhancements could include mobile application development, integration with GPS tracking, and support for multi-language interfaces.

9.1 FUTUTE SCOPE

To protect sensitive user data, several security measures were implemented:

- **Encryption:** User passwords and payment details are encrypted using SSL/TLS protocols.
- **Authentication and Authorization:** Only registered users can access the booking features, with different privileges for administrators.
- **Input Validation:** All user inputs are validated to prevent SQL injection and XSS attacks.
- **Secure Payment Gateway:** Integrated a secure payment API that handles transactions safely.

References:

1. Books on System Design & Software Engineering:

- Sommerville, I. (2011). *Software Engineering* (9th ed.). Addison-Wesley.
- Pressman, R. S. (2014). *Software Engineering: A Practitioner's Approach* (8th ed.). McGraw-Hill.
- Fowler, M. (2002). *Patterns of Enterprise Application Architecture*. Addison-Wesley.

2. Database Management and System Architecture:

- Elmasri, R., & Navathe, S. B. (2015). *Fundamentals of Database Systems* (7th ed.). Pearson.
- Connolly, T., & Begg, C. (2014). *Database Systems: A Practical Approach to Design, Implementation, and Management* (6th ed.). Pearson.

3. Web Development and User Interface Design:

- Duckett, J. (2014). *HTML and CSS: Design and Build Websites*. Wiley.
- Krug, S. (2014). *Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability*. New Riders Publishing.

4. Agile Methodology and Project Management:

- Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide*.
- Cobb, C. G. (2011). *The Project Manager's Guide to Mastering Agile: Principles and Practices for an Adaptive Approach*. Wiley.

5. Case Studies & Technical Reports on Bus Booking Systems:

- Li, Y., & Tang, M. (2016). "Design and Implementation of an Online Bus Ticket Booking System." *International Journal of Innovative Research in Computer and Communication Engineering*, 4(7), 132–140.
- Radhika, B. G., et al. (2019). "A Study on Bus Reservation System." *International Journal of Advanced Research in Computer Science and Software Engineering*, 9(4), 32–38.

6. API and Online Payment Integration References:

- PayPal API Documentation: <https://developer.paypal.com/docs/api/>
- Stripe Payment Gateway Integration Guide: <https://stripe.com/docs/development>