BUS TICKET RESERVATION SYSTEM

1. Introduction

The manual bus ticketing system often leads to long queues, overbooking, and poor customer satisfaction. To overcome these challenges, a Bus Ticket Reservation Management System is proposed. The system allows customers to search trips, view real-time seat availability, book tickets, and make secure payments. It also provides features for downloading e-tickets, QR codes, and easy cancellations. Administrators can manage buses, routes, schedules, pricing, and reports through a centralized dashboard. The backend is built using Spring Boot, Spring Security (JWT), MySQL, and Hibernate. The frontend uses React.js, Bootstrap, and Axios for responsive and seamless user interaction. Role-based access control ensures security, with different privileges for Admin and Customer. This system improves efficiency, reduces revenue leakage, and enhances customer satisfaction.

2. Requirement Analysis

2.1 User Stories

To ensure a user-centric design, the project requirements were defined through user stories. This approach helps to focus development on features that provide direct value to the end-user.

Feature User Story

- Authentication - As a user, I want to register and log in securely using my credentials so I can access the system.
- Trip Search - As a customer, I want to search for buses between my source and destination so I can plan my travel.
- Seat Selection - As a customer, I want to view available seats in real time so I can choose my preferred seat.
- Booking & Payment - As a customer, I want to book tickets and pay online securely so I can confirm my travel instantly.
- E-Ticket & QR Code - As a customer, I want to download my e-ticket with a QR code so I can show it during boarding.
- Cancellations & Refunds - As a customer, I want to cancel my booking and get a refund as per policy so I have flexibility.
- Bus & Route Management - As an admin, I want to add and manage buses, routes, and schedules so I can operate efficiently.
- Reports & Dashboards - As an admin, I want to see reports on sales, occupancy, and routes so I can track performance.
- Role-Based Access - As an admin/customer, I want restricted access to only my features so that the system stays secure.

3. System Design

3.1 Architecture

The Bus Ticket Reservation Management System follows a 3-Tier Architecture to ensure modularity, scalability, and maintainability. Each tier has a distinct responsibility, enabling smooth integration and independent upgrades.

Presentation Tier (Frontend):

Implemented as a Single Page Application (SPA) using React.js with Bootstrap and CSS. This layer manages user interactions, including trip searches, seat selection, booking, payments, and admin dashboards. JWT tokens are stored securely in local/session storage for authentication.

Business Tier (Backend):

Built using Spring Boot (RESTful APIs) with Spring Security + JWT for authentication and role-based authorization. This tier processes business rules such as real-time seat inventory, booking workflows, payment handling, and cancellation policies.

Data Tier (Database):

A MySQL relational database serves as the persistence layer, storing structured data including users, buses, routes, trips, bookings, and payments. JPA/Hibernate is used for ORM to map database tables to Java entities.

This layered approach ensures separation of concerns: the frontend focuses on user experience, the backend enforces business logic and security, and the database ensures reliable data storage and retrieval.

3.2 Database Schema

The database schema is designed using a relational model to ensure data consistency, integrity, and efficient query performance. The schema includes core entities such as Users, Buses, Routes, Trips, Bookings, Payments, and Tickets.

- The Users table stores customer and admin details, with role-based access.
- The Buses table contains bus details such as bus number, type, and capacity.
- The Routes table defines source and destination cities along with travel distance.
- The Trips table links buses to routes and schedules with date, time, and fare.
- The Bookings table records customer reservations, linking users to specific trips and seats.
- The Payments table manages transaction details for bookings.
- The Tickets table generates e-tickets with QR codes for validation.
- The Trips table acts as a central entity, maintaining relationships with Buses, Routes, Bookings, and Payments.

4. Implementation

4.1 Technology Stack

The technology stack was chosen to leverage a robust, enterprise-grade backend and a modern, dynamic frontend, ensuring a scalable, secure, and maintainable application.

Backend : Spring Boot 3.5.5, Spring Security + JWT, Spring Data JPA, Java 17, Maven

Frontend : React.js ,React Router, Bootstrap, Axios

Database : MySQL

Security : JWT Authentication, Role-Based Access Control, BCrypt Password Hashing

Testing : JUnit s

Collaboration : GitHub repository with feature branching workflow

API Documentation : Swagger / OpenAPI

This stack ensures real-time seat management, secure payments, role-based access, and responsive user experience, while supporting maintainability and future scalability for additional features such as agent bookings, promo codes, and multi-operator support.

4.2 Flow Example – Login Process

- When a Customer or Admin enters credentials on the React login page, a POST request is sent to the Spring Boot backend's AuthController.
- The backend validates the credentials against the MySQL Users table.
- If valid, a JWT token is generated and returned to the frontend.
- The frontend stores the token in localStorage/sessionStorage and decodes the user role.
- The user is redirected to the appropriate dashboard (Customer or Admin).
- For future requests, the JWT is sent in the Authorization header, enabling secure, role-based access.
- Invalid credentials or expired tokens trigger 401/403 errors, prompting re-login.

5. Frontend Validation

Client-side validation was implemented to improve user experience and prevent malformed or incomplete data from being submitted to the server. The project uses Formik to manage form state and Yup to define validation schemas for various input fields.

- Email Validation: Ensures the email entered by the user is in a proper format.
- Password Validation: Enforces password complexity rules (minimum length, uppercase/lowercase letters, digits, special characters).
- Seat Selection: Checks that at least one seat is selected before proceeding to booking.

- Payment Details: Validates card numbers, expiry dates, CVV, and required billing information.
- Real-Time Feedback: Users receive immediate error messages or guidance for corrections before submission.

This approach ensures that only properly formatted and complete data reaches the backend, reducing errors, improving data integrity, and enhancing the overall booking experience.

6. Achievements

This project successfully delivered a functional prototype of a Bus Ticket Reservation Management System. Key achievements include:

- A secure JWT-based authentication and role-based access control system.
- Fully functional REST APIs for core entities such as Users, Buses, Routes, Trips, Bookings, Payments, and Tickets.
- A responsive React frontend integrated seamlessly with the backend services.
- Real-time seat selection, booking, payment processing, and e-ticket generation.
- A modular project structure allowing maintainability and scalability for future features.

7. Conclusion

The **Bus Reservation System** provides a reliable and efficient way to manage bus bookings, seat allocation, and ticketing. It simplifies the process for passengers by offering a convenient platform for searching trips, selecting seats, and making reservations, while also giving administrators effective tools for managing schedules, buses, and bookings.