

RAILWAY RESERVATION SYSTEM

CS2333 – Object Oriented Programming Using JAVA Project Report

Submitted by

ASWIN MUTHUKUMAR -231001022

INDRANILCHAKRAVARTHY-231001064

Of

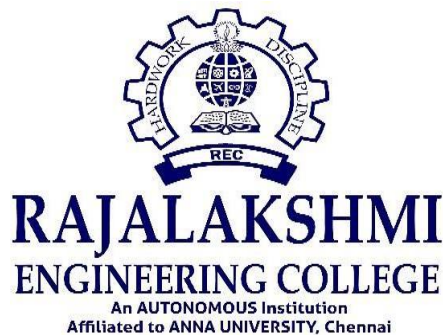
BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

RAJALAKSHMI ENGINEERING COLLEGE, THANDALAM

(An Autonomous Institution)



RAJALAKSHMI ENGINEERING COLLEGE

NOVEMBER 2024

BONAFIDE CERTIFICATE

Certified that this project titled “Railway Reservation System” is the bonafide work of **ASWIN MUTHUKUMAR (231001022), INDRANILCHAKRAVARTHY (231001064)** who carried out the project work under my supervision

SIGNATURE

Dr.P.Valarmathie

HEAD OF THE DEPARTMENT

Department of Information Technology
Rajalakshmi Engineering College

SIGNATURE

Mrs.Usha S

COURSE INCHARGE

Assistant Professor(S.G)

Department of Information Technology
Rajalakshmi Engineering College

This mini project is submitted for CS23333 – Object Oriented Programming Using JAVA held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

Table of Contents:

CHAPTER NO.	TITLE	PAGE NO.
1	1.1 ABSTRACT	5
	1.2 INTRODUCTION	5
	1.3 PURPOSE	5
	1.4 SCOPE OF PROJECT	6
	1.5 SOFTWARE REQUIREMENT SPECIFICATION	6
2	SYSTEM FLOW DIAGRAM	12
	2.1 USE CASE DIAGRAM	12
	2.ENTITY RELATIONSHIP DIAGRAM	13
	2.3 DATA FLOW DIAGRAM	14

3	MODULE DESCRIPTION	15
4	IMPLEMENTATION	16
	4.1 DESIGN	16
	4.2 DATABASE DESIGN	20
	4.3 CODE	21
5	CONCLUSION	27
6	REFERENCE	27

LIST OF FIGURES

Figure Numbers	Figure Captions	Page No.
2.1.1	Use Case Diagram	13
2.2.1	Entity Relationship Diagram	14
2.3.1	Data Flow Diagram	15
4.1.1	Login Page	17
4.1.2	Home Page	17
4.1.3	Train Booking Steps	18
4.1.4	Details of Booked Train	18
4.1.5	Details of Train Booked	19
4.1.6	Xampp	19

LIST OF TABLES

Table Number	Table Caption	Page No.
4.2.1	Train Booking Table	22

1.1 Abstract:

The JDBC-powered Railway Reservation System is designed to provide a reliable, efficient, and user-friendly platform for passengers to browse train schedules, select seats, and reserve tickets seamlessly. Leveraging Java Database Connectivity (JDBC), the system establishes a secure connection with a relational database, ensuring Realtime updates on seat availability, reservation status, and train schedules. Using JDBC transactions, the system maintains data integrity and consistency, minimizing errors and conflicts during the reservation process. Passengers can interact with an intuitive interface to search for available trains, select desired seats, and complete bookings with ease. The system is scalable, making it suitable for a variety of railway environments, from small regional stations to large national networks.

1.2 Introduction:

The Railway Reservation System (RRS) is a critical component in modernizing the way passengers book tickets and manage their travel plans. Traditionally, railway ticketing processes involved manual intervention, long queues, and high chances of errors, leading to inefficiencies and customer dissatisfaction. With the advent of technology, the Railway Reservation System offers a robust solution to streamline the ticketing process, improving efficiency, accuracy, and overall user experience. This system enables passengers to easily browse available trains, check schedules, select seats, and book tickets online or through automated kiosks, eliminating the need for manual ticket counters. The Railway Reservation System ensures real-time updates on seat availability, reservation status, and train schedules, providing passengers with the most accurate and timely information.

1.3 Purpose:

Simplify Ticket Booking: Automates the reservation process, allowing passengers to easily book tickets online and check schedules.

Enhance Passenger Experience: Provides real-time updates on train schedules and seat availability for a smoother booking experience.

Efficient Management: Enables railway authorities to manage schedules, seat allocations, and reservations efficiently.

Data Insights: Collects data for analysis to optimize train schedules, pricing, and services.

Security: Ensures passenger data and transactions are protected through secure authentication and encryption.

1.4 Scope of the Project:

The Railway Reservation System will allow passengers to search for trains, view schedules, select seats, and book tickets online with real-time updates. It will feature secure user authentication to protect personal and payment data. Administrators can manage schedules, seat availability, and generate reports on booking trends and revenue. The system will also support booking modifications and cancellations, improving flexibility.

1.5 Software Requirement Specification:

Introduction:

The Railway Reservation System (RRS) is designed to automate and streamline the process of booking train tickets, managing schedules, and seat allocations for passengers. It provides a comprehensive solution for passengers to search trains, view schedules, reserve seats, and make payments online, while also assisting railway administrators in managing operations efficiently.

Document Purpose:

This SRS document outlines the software requirements for the Railway Reservation System, detailing the design decisions, architectural considerations, functional requirements, and other essential aspects necessary for its successful Product Scope implementation. This document serves as a guide for the development, testing, and future maintenance of the system.

Product Scope:

The Railway Reservation System replaces outdated manual ticketing processes, offering an automated, user-friendly platform for booking train tickets. It supports functions like searching for trains, viewing seat availability, booking tickets, and providing real-time updates on train schedules. For administrators, it offers functionalities to manage train schedules, reservations, and generate reports.

RRS - Railway Reservation System

SRS - Software Requirements Specification.

References and Acknowledgement:

- <https://www.javatpoint.com/java-awt>
- <https://www.javatpoint.com/java-swing>
- <https://www.w3schools.com/sql/>
- <https://www.geeksforgeeks.org/introduction-to-jdbc/>

The Railway Reservation System enables passengers to search for train routes, check seat availability, make reservations, and manage bookings. It provides a centralized platform to simplify train travel for users and enhance operational efficiency for railway administrators.

Product Perspective:

Built on a client/server model, the system is optimized for the Microsoft Windows operating system. The front end is developed using Java AWT and Swing, while MySQL server powers the backend for robust and efficient data management.

Product Functionality:

- Admin Register: Allows new administrators to register on the platform.
- Admin Login: Provides secure login functionality for existing administrators.
- Add Railway: Enables administrators to add details of new railways.
- View Railway: Facilitates viewing and updating of existing railway information.
- Delete Railway: Permits removal of railways from the system database.
- Add Reservation: Supports the creation of new user reservations.

- Update Reservation: Allows users to view and modify their existing reservations.
- Remove Admin: Enables deletion of specific administrator accounts.

User and Characteristics:

Qualification: Users should have at least basic educational qualifications, such as matriculation, and be comfortable with English.

Experience: Familiarity with the university registration process is advantageous. Technical

Experience: Users are expected to have elementary knowledge of computers for optimal system interaction.

Operating Environment:

Hardware Requirements:

- Processor: Any Processor over i3
- Operating System: Windows 8, 10, 11
- Processor Speed: 2.0 GHz
- RAM: 4GB
- Hard Disk: 500GB

Software Requirements:

- Database: MySQL
- Frontend: Java (SWING, AWT)
- Technology: Java (JDBC)

Constraints:

- System access limited to administrators.
- Delete operation restricted to administrators without additional checks for simplicity.
- Administrators must exercise caution during deletion to maintain data consistency.

Assumptions and Dependencies:

- System administrators create and confidentially communicate login IDs and passwords to users. Specific Requirements:

User Interface:

The Railway Reservation System provides user-friendly, simplified interfaces for:

- a) Admin Register: Registering new administrators.
- b) Admin Login: Logging in existing administrators.
- c) View Railway: Viewing and updating existing railway information.
- d) Delete Railway: Deleting existing railways.
- e) Add Reservation: Creating new reservations for users.
- f) Update Reservation: Viewing and modifying existing reservations.
- g) Remove Admin: Deleting specific administrator accounts.

Hardware Interface:

- Screen resolution of at least 640 x 480 or above.
- Compatible with any version of Windows 8, 10, 11.

Software Interface:

- a) MS-Windows Operating System
- b) Java AWT and SWING for designing the front end
- c) MySQL for the backend
- d) Platform: Java Language
- e) Integrated Development Environment (IDE): NetBeans

Functional Requirements:

Login Module (LM):

- Users (admins) access the Login Module.
- LM supports user login with a username and password.
- Passwords are masked for security.
- Successful login verification by the database administrator is required for access.

Registered Users Module (RUM):

- After successful login, users (admins) can navigate through the application.
- Users can view detailed information about train, timing, and reservations.
- Users can update and maintain train details, including modifying timing and seat allocations.

Administrator Module (AM):

- Upon successful login, the system displays administrative functions.
- Functions include adding and updating train details.
- The "Add" function allows administrators to input new train details and remove unused entries.
- The "Update" function enables administrators to modify existing train details in the database.
- All add, update, or delete requests trigger the AM module to communicate with the Server Module (SM) for necessary database changes.

Server Module (SM):

- SM acts as an intermediary between various modules and the database (DB).
- Receives requests from different modules and formats pages for display.
- Validates and executes requests received from other modules.
- Handles communication with the database, ensuring data consistency and integrity, especially regarding train details, timing, and reservations.

Non-functional Requirements:

Performance:

The system must handle real-time reservation requests efficiently, ensuring a response time of less than 2 seconds for seat selection and confirmation. - Safety-critical failures, such as payment processing errors, must be addressed instantly to ensure a smooth user experience.

Reliability:

The system is safety-critical; in case of abnormal operation or downtime, immediate measures should be taken to resolve the issue and restore normal functionality.

Availability:

Under normal operating conditions, user requests for railway reservations, including seat selection and payment, should be processed within 2 seconds to maintain a seamless booking experience.

Immediate feedback on reservation status and confirmation should be communicated to users to enhance the overall booking process.

Security:

A robust security mechanism must be in place on the server side to prevent unauthorized access, safeguard user payment information, and ensure the integrity of the reservation system.

User privacy, including personal details, must be securely stored and managed to maintain confidentiality.

Maintainability:

Design documents outlining software and database maintenance procedures must be available to facilitate regular updates and modifications to the railway reservation system.

Administrative access should be provided for proper maintenance at both the front end and back end, ensuring the system's long-term functionality and adaptability.

2. System Flow Diagrams:

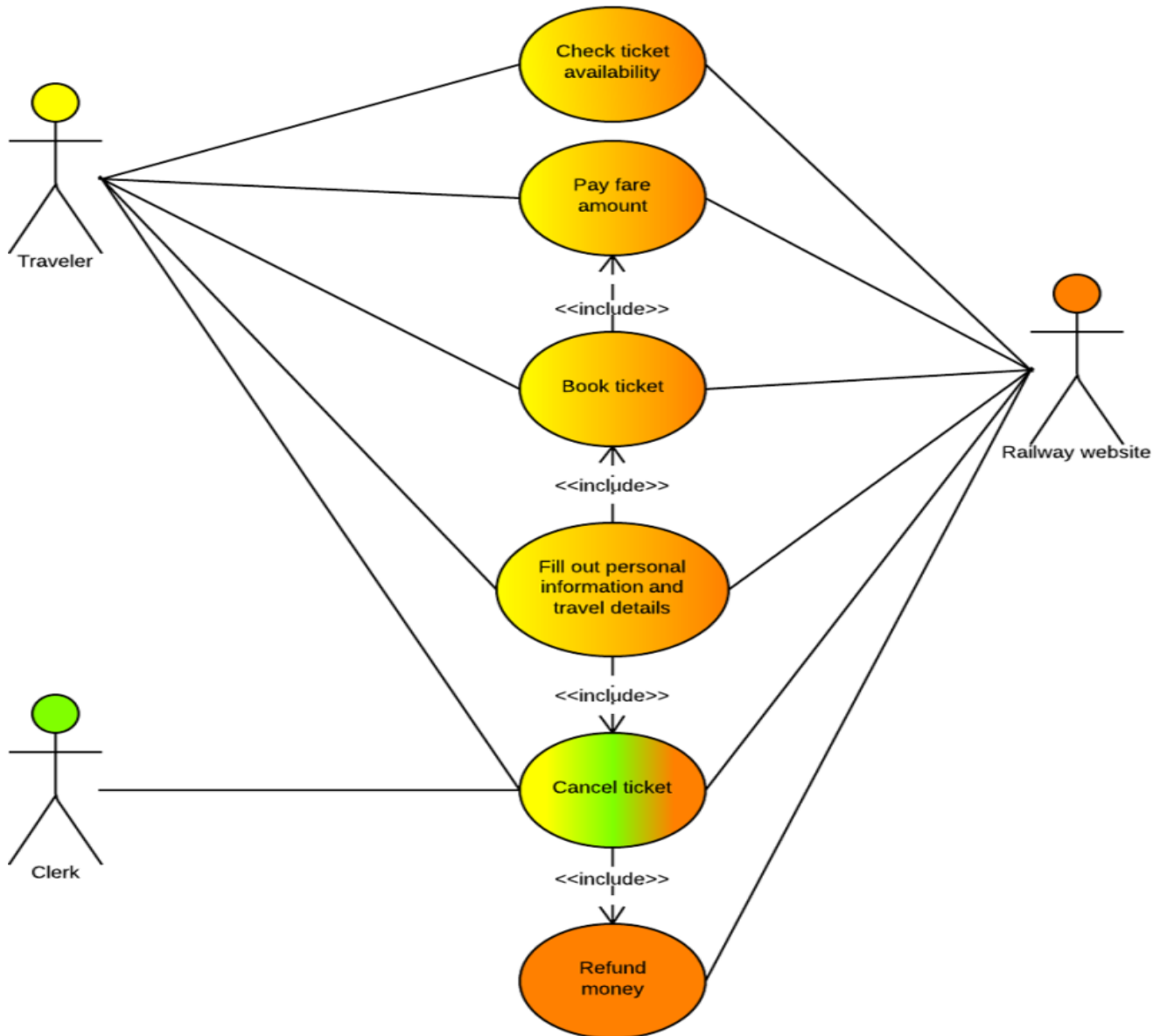


Figure 2.1.1 Use Case Diagram

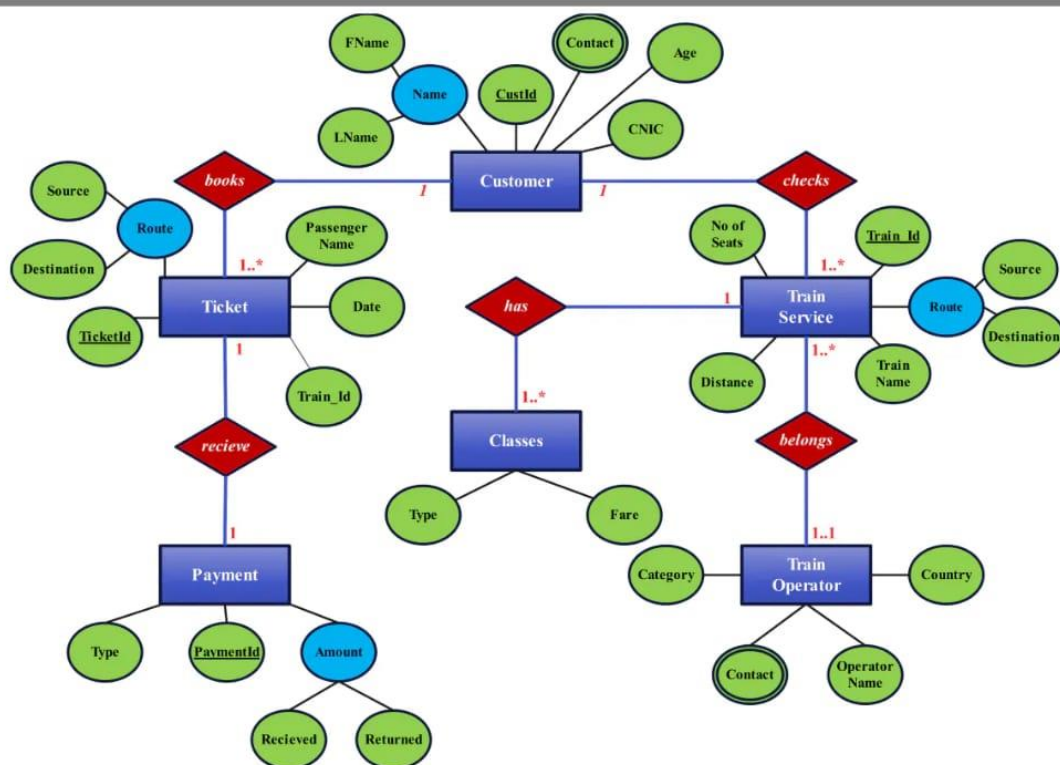


Figure 2.2.1 Entity Relationship Diagram

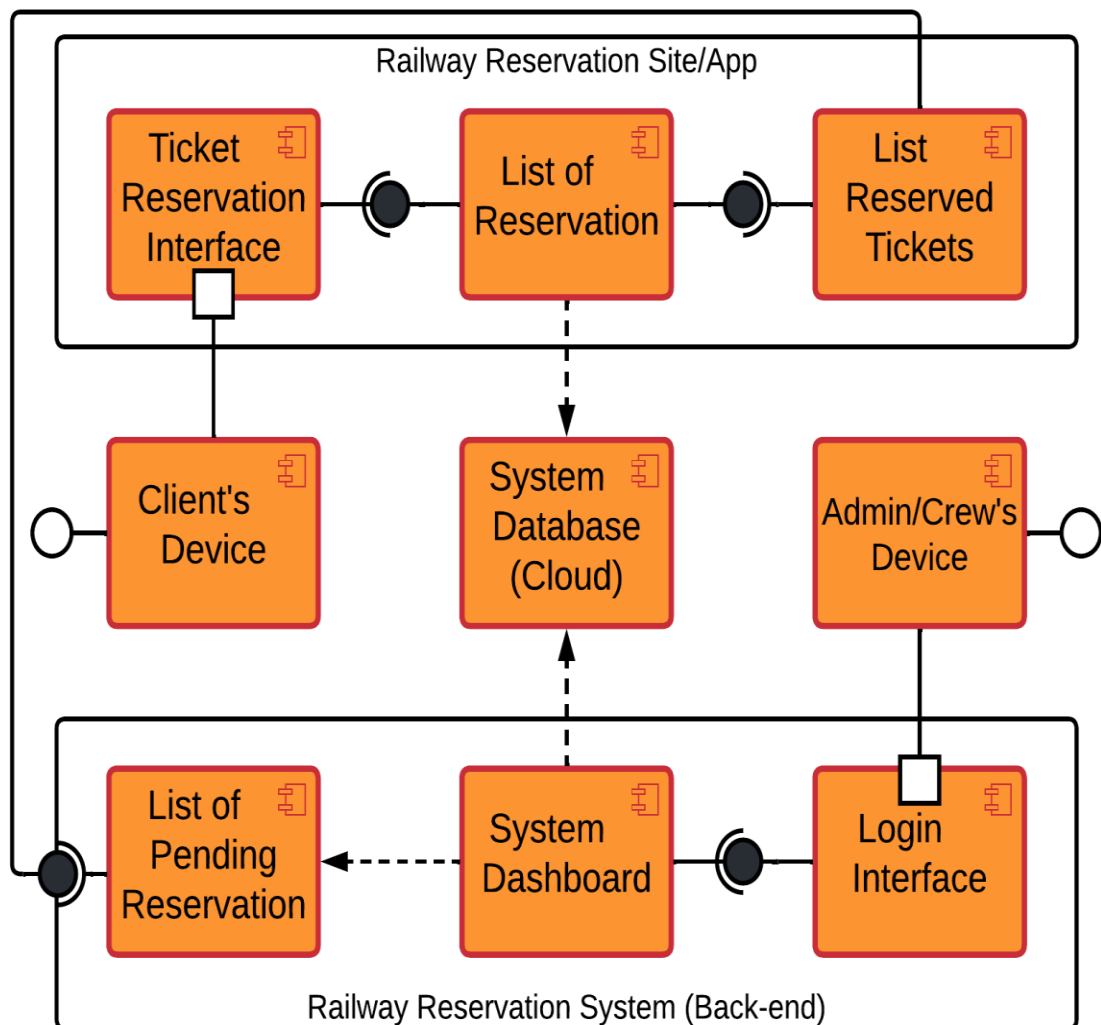


Figure 2.3.1 Data Flow Diagram

3. Module description:

1. Register:

Admin can register with the username and password for the registration

2. Login:

Admin can log in with their username and password.

3. After Login:

- **Add Train:**

In this section, admin can add details about new trains, including the title, date, and other relevant information.

- **View Railway:**

Admin can view and update railway details such as the title, no of seats and release date.

- **Delete Train:**

Admin can delete train details, removing trains from the system.

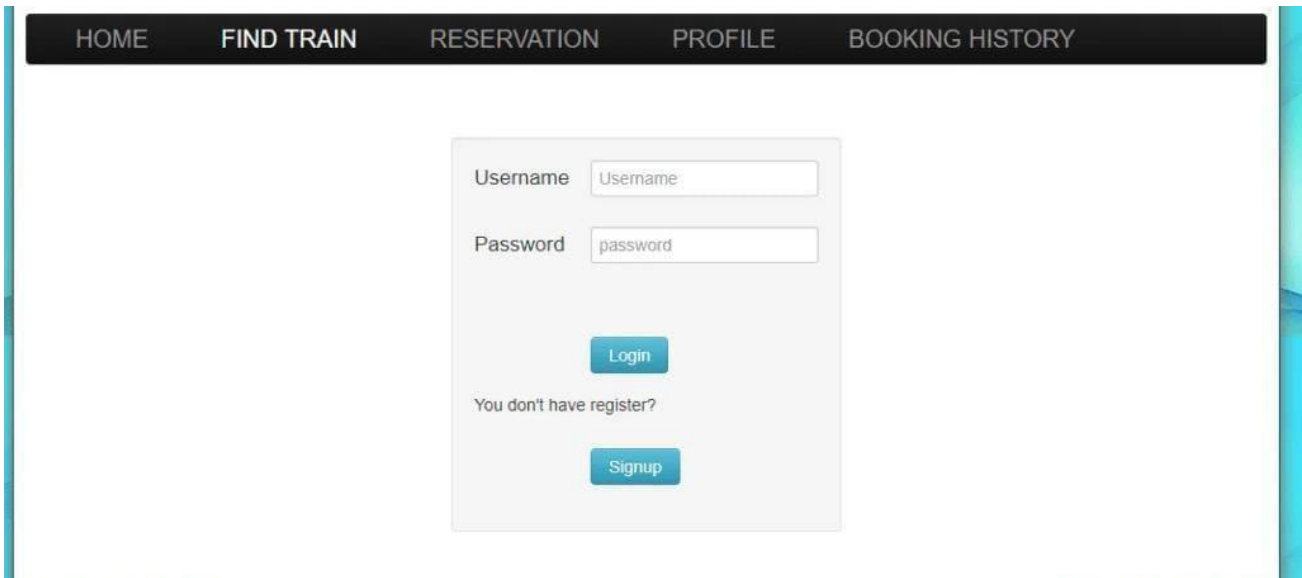
- **Add Reservation:**

Admin can add reservation details, including the trains, date, time, and number of seats.

- **Update Reservation:**

Admin can update reservation details, such as changing the date, time, or the number of reserved seats.

4. Implementation:



The screenshot shows a login page with a dark blue header containing navigation links: HOME, FIND TRAIN, RESERVATION, PROFILE, and BOOKING HISTORY. The main content area features a light gray login box. Inside the box, there are two input fields: 'Username' with the placeholder text 'Username' and 'Password' with the placeholder text 'password'. Below these fields are two buttons: a blue 'Login' button and a blue 'Signup' button. A link 'You don't have register?' is positioned between the two buttons.

Figure 4.1.1 Login Page

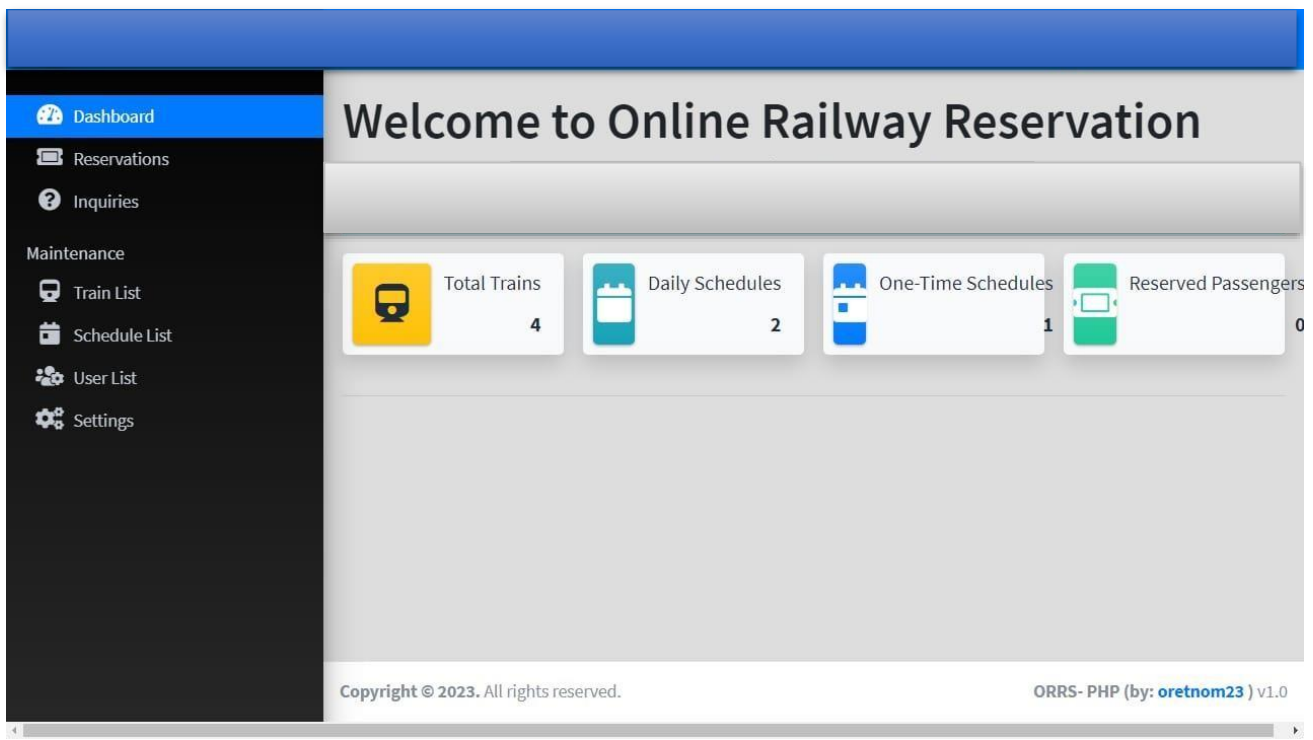


Figure 4.1.2 Home Page

localhost/orrs/?page=reserve&sid=1

09854698789 / 78945632 Admin Login

Schedule: **Everyday 07:00 AM** To: **Station 2** Economy Fare: **25**

Schedule Date

mm/dd/yyyy

Select here

Seat Type Fare Amount

(optional)

First Name Middle Name Last Name

Remove

+ Add Passenger Submit Reservation

Figure 4.1.3 Train Booking Steps

localhost/orrs/admin/?page=trains

ORRS - PHP

Online Railways Reservations System - PHP - Admin

Administrator Admin

+ Add New

Search:

Capacity Action

Class: **100** omny: **200** Action

Class: **100** omny: **200** Action

Class: **150** omny: **300** Action

Class: **150** omny: **300** Action

Previous 1 Next

Update Train Details

Train #

TIR-1001

Spaces and special characters except (-_) are not allowed in this field.

Name

Train 101

First Class Seat Capacity

100

Economy Seat Capacity

200

Save Cancel

Figure 4.1.4 Details of booked Train

Dashboard
Reservations
Inquiries

Maintenance
Train List
Schedule List
User List
Settings

List of Reservations

Show 10 entries Search:

Seat #	Schedule	Schedule Code	Passenger	Group	Action
E-001	2022-01-07 08:00	202201-0003	Cooper, Mark D	Economy	Action
E-001	2022-01-07 07:00	202201-0001	Smith, John D	Economy	Action
E-002	2022-01-07 08:00	202201-0003	Cooper,...	Economy	Action
E-002	2022-01-07 07:00	202201-0001	Blake, Claire C	Economy	Action
FC-001	2022-01-07 08:00	202201-0003	Smith, John D	First Class	Action
FC-001	2022-01-07 07:00	202201-0001	Smith, John D	First Class	Action
FC-002	2022-01-07 08:00	202201-0003	Blake, Claire C	First Class	Action

Figure 4.1.5 Details of Train Booked

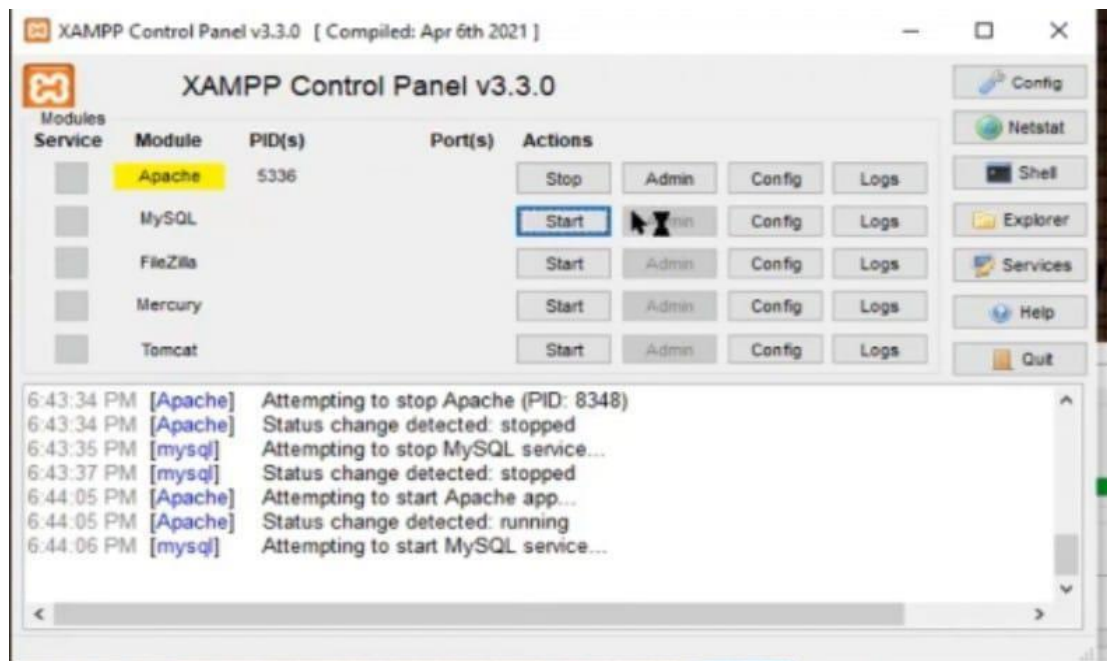


Figure 4.1.6 Xampp

4.2 Database Design:

The database design is a crucial part of the system design process, where the data elements and structures identified during the analysis stage are organized for efficient storage and retrieval. A database is a collection of related data, stored with minimal redundancy, to serve multiple users quickly and effectively. The primary goal is to ensure that database access is easy, fast, cost-effective, and flexible. Data relationships are established, and unnecessary items are removed. Normalization is performed to achieve internal consistency, minimize redundancy, and ensure maximum stability. This process helps to reduce the storage requirements, avoid data inconsistencies, and optimize updates. MySQL has been selected as the database management system for developing the necessary databases.

MySQL is a widely used, open-source relational database management system known for its reliability, scalability, and performance. It is the go-to choice for many web applications, particularly those using the LAMP (Linux, Apache, MySQL, PHP) stack. MySQL enables developers to manage large datasets efficiently, offering features like fast query processing and strong data security. Its structured query language (SQL) makes data retrieval and manipulation intuitive. MySQL's cross-platform compatibility and seamless integration with various programming languages and frameworks make it ideal for diverse applications, from small-scale websites to enterprise-level databases.

MySQL offers several advantages over other databases like PostgreSQL and Oracle. It is lightweight, making it faster and more efficient for web-based applications. MySQL's ease of use and quick setup process make it accessible for beginners, while its scalability supports large-scale systems. Compared to Oracle, MySQL is more cost-effective, particularly for startups and smaller enterprises. Additionally, MySQL's broad community support provides extensive resources and documentation. Its performance is particularly optimized for read-heavy workloads, which is ideal for applications requiring fast data retrieval, making it a preferred choice for many businesses.

Coach No = S1
 Total Km's = 1515 kms
 Fare Per Seat = 500.0 Rs

Adult's Details :							
No	Seat No	Name	Sex	Age	Senior Citizen	Amount	Status
1	S1	SANTOSHKUMAR	M	26	N	500.0	RESERVED
2	S2	SINGH	M	32	N	500.0	RESERVED
3	S3	ATANU	M	13	N	500.0	RESERVED
4	S5	SITA	F	21	N	500.0	RESERVED
5	S6	JAVA	M	13	N	500.0	RESERVED
6	S7	JSP	M	13	N	500.0	RESERVED

PAYMENT

Table 4.2.1 Train Booking Table

4.3 IMPLEMENTATIONS CODE:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.sql.*;

public class RailwayReservationSystem {
    private JFrame frame;
    private JComboBox<String> railwayDropdown;
    private JTextField customerNameField;
    private JTextField seatNumberField;
    private JButton reserveButton;
    private JTextArea reservationStatus;

    // Database credentials
    private static final String DB_URL = "jdbc:mysql://localhost:3306/theater_db";
    private static final String USER = "root";
    private static final String PASSWORD = "password";

    public RailwayReservationSystem() {
        // Setup the GUI
        frame = new JFrame("Railway Reservation System");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(500, 400);

        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(6, 2, 10, 10));

        panel.add(new JLabel("Select Railway:"));
        railwayDropdown = new JComboBox<>();
        loadRailways(); // Populate dropdown with railways from the database
    }
}
```



```

panel.add(railwayDropdown);

panel.add(new JLabel("Customer Name:"));
customerNameField = new JTextField();
panel.add(customerNameField);

panel.add(new JLabel("Seat Number:"));
seatNumberField = new JTextField();
panel.add(seatNumberField);

reserveButton = new JButton("Reserve Seat");
reserveButton.addActionListener(this::reserveSeat);
panel.add(reserveButton);

reservationStatus = new JTextArea();
reservationStatus.setEditable(false);
frame.add(panel, BorderLayout.CENTER);
frame.add(new JScrollPane(reservationStatus), BorderLayout.SOUTH);

frame.setVisible(true);
}

private void loadRailways() {
    try (Connection conn = DriverManager.getConnection(DB_URL, USER, PASSWORD);
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery("SELECT title FROM railways")) {

        while (rs.next()) {
            railwayDropdown.addItem(rs.getString("title"));
        }
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(frame, "Error loading railways: " + e.getMessage());
    }
}

```

```

private void reserveSeat(ActionEvent e) {
    String selectedRailway = (String) railwayDropdown.getSelectedItem();
    String customerName = customerNameField.getText();
    String seatNumber = seatNumberField.getText();

    if (selectedRailway == null || customerName.isEmpty() || seatNumber.isEmpty()) {
        reservationStatus.setText("Please fill out all fields.");
        return;
    }

    try (Connection conn = DriverManager.getConnection(DB_URL, USER, PASSWORD)) {
        // Get the railway ID
        PreparedStatement getRailwayIdStmt = conn.prepareStatement("SELECT id FROM railways
WHERE title = ?");
        getRailwayIdStmt.setString(1, selectedRailway);
        ResultSet rs = getRailwayIdStmt.executeQuery();

        if (rs.next()) {
            int railwayId = rs.getInt("id");

            // Insert reservation
            PreparedStatement reserveStmt = conn.prepareStatement(
                "INSERT INTO reservations (railway_id, seat_number, customer_name) VALUES (?,
?, ?)");
            reserveStmt.setInt(1, railwayId);
            reserveStmt.setString(2, seatNumber);
            reserveStmt.setString(3, customerName);
            reserveStmt.executeUpdate();

            reservationStatus.setText("Reservation successful for " + customerName + "!");
        } else {
            reservationStatus.setText("Railway not found.");
        }
    }
}

```

```

    } catch (SQLException ex) {
        reservationStatus.setText("Error: " + ex.getMessage());
    }
}

public static void main(String[] args) {
    // Load JDBC driver
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
    } catch (ClassNotFoundException e) {
        JOptionPane.showMessageDialog(null, "JDBC Driver not found: " + e.getMessage());
        return;
    }

    SwingUtilities.invokeLater(RailwayReservationSystem::new)
    }
}

```

5.Conclusion:

The Railway Ticket Booking Reservation System project, developed under expert supervision, reflects a thorough and systematic design and implementation process. Prioritizing user-centric features such as adding, viewing railways, and managing reservations, the system ensures an effortless and efficient experience for users. Enhanced security protocols, especially in the "Remove Admin" module, emphasize the system's dedication to safeguarding data and ensuring operational integrity. This project is a comprehensive solution tailored to meet current demands while remaining adaptable for future upgrades and enhancements.

6.Reference links:

<https://www.javatpoint.com/java-awt>

<https://www.javatpoint.com/java>

<https://www.localhost/orrs/java>

