

RAILWAY RESERVATION SYSTEM

A MINI PROJECT REPORT

Submitted by

Manoj Kumar S	230701178
Mohamed Hasan Mohamed Nazeer	230701187
Mohamed Ikram	230701188

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE



RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

2024 - 25

BONAFIDE CERTIFICATE

Certified that this project report “**RAILWAY RESERVATION SYSTEM**” is the bonafide work of
“**MANOJ KUMAR S(230701178),MOHAMED HASAN MOHAMED NAZEER (230701187),**
MOHAMED IKRAM(230701188)”

who carried out the project work under my supervision.

Submitted for the Practical Examination held on _____

SIGNATURE

Mrs. K. MAHESMEENA,
Assistant Professor,
Computer Science and Engineering,
Rajalakshmi Engineering College,
(Autonomous),
Thandalam, Chennai - 602 105

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT:

The **Railway Reservation System** is a robust tool designed to streamline and optimize the management of railway ticket bookings and passenger information. This system facilitates efficient tracking of passenger data, ticket reservations, and journey details, providing a centralized solution for railway administrators and travelers to manage and monitor travel activities. By offering a clear overview of passenger details, train schedules, seat availability, and booking statuses, the system enables informed decision-making for both railway staff and passengers.

Through its structured organization of booking data and real-time information capabilities, the Railway Reservation System ensures smooth and efficient travel operations. This supports better travel planning, aids in journey tracking, and encourages improved passenger satisfaction. Users can search for trains, book tickets, and view travel details, maintaining transparency and accountability within the reservation process.

The system features an intuitive user interface that includes an admin dashboard, enabling administrators to oversee and manage ticket bookings, monitor train schedules, and analyze trends in passenger travel patterns. With advanced security protocols and efficient database management, the system ensures that all passenger information is accurate, secure, and up-to-date.

Overall, the Railway Reservation System enhances the efficiency of travel management processes and empowers railway staff and passengers to achieve greater convenience and collaboration in the travel experience.

TABLE OF CONTENTS

Chapter 1

1 INTRODUCTION

- 1.1 Introduction
- 1.2 Objectives
- 1.3 Modules

Chapter 2

2 SURVEY OF TECHNOLOGIES

- 2.1 Software Description
- 2.2 Languages
 - 2.2.1 Java
 - 2.2.2 SQL

Chapter 3

3 REQUIREMENTS AND ANALYSIS

- 3.1 Requirement Specification
 - 3.1.1 Functional Requirements
- 3.2 Hardware and Software Requirements
- 3.3 ER Diagram

Chapter 4

4 PROGRAM CODE

- 4.1 Backend details and Program Code

Chapter 5

5 OUTPUTS

- 5.1 Outputs

Chapter 6

6 CONCLUSION

- 6.1 Conclusion

Chapter 7

7 REFERENCES

- 7.1 References

Chapter 1: INTRODUCTION

1.1 INTRODUCTION

Effective management of railway reservations is crucial for ensuring smooth operations, maintaining accurate travel records, and providing a positive experience for passengers and staff. The Railway Reservation System offers a streamlined, user-friendly platform designed to securely and efficiently manage booking and passenger information. With a comprehensive suite of tools, railway administrators can monitor ticket reservations, train schedules, and passenger data with ease.

The system provides core functionalities, including passenger registration, secure login, and reservation management (seat availability, booking status, and journey details). Additionally, it maintains essential information such as personal details, travel history, and payment records, ensuring all reservation-related data is accessible in one place. These features empower administrators, staff, and passengers with a holistic view of travel information, fostering better decision-making and collaboration.

Built with Java and MySQL, the Railway Reservation System leverages Java for backend logic and an intuitive GUI developed using JFrame. MySQL supports data storage and retrieval, offering a secure and reliable foundation for managing reservation records. The Java Swing-based interface enhances usability, delivering a modern and interactive experience for users.

This report details the system's development, architecture, and technology integration, demonstrating how Java, MySQL, and NetBeans combine to create a secure and efficient railway reservation solution. The system aims to provide a seamless, reliable user experience to address reservation management needs with industry-standard performance.

1.2 OBJECTIVES

- **To develop a centralized database** for securely managing passenger profiles, ticket bookings, and travel records.
- **To enable efficient handling of train schedules and seat availability**, providing real-time updates for passengers and railway administrators.
- **To provide a secure login system for authentication**, ensuring the confidentiality of passenger data.
- **To allow users to easily access and view essential travel details**, including personal information, booking status, and journey history, through a streamlined interface.
- **To ensure compliance with data security standards** for information storage and management.

1.3 MODULES

1. **Passenger Registration Module**

This module captures essential passenger information, such as name, email, password, and travel preferences. Data is securely stored in the database. The module ensures that only authorized users can access booking records by implementing credential storage with secure encryption and validation mechanisms.

2. **Profile Management Module**

This module provides a centralized view for users to access and manage passenger profiles. Administrators can view critical information such as passenger IDs, booking history, payment details, and travel preferences. The system ensures data accuracy and security through robust authentication mechanisms.

3. **Reservation Management Module**

This module facilitates ticket booking and seat allocation for each passenger. Passengers can check seat availability and make reservations, while administrators can update and manage booking records in real time.

4. **Journey Tracking Module**

This module records travel details and booking status. Passengers can view their journey information, including train schedules, seat details, and booking confirmation. Administrators can track train schedules and manage delays or updates.

5. **Admin Dashboard Module**

The dashboard provides administrators with tools to oversee booking records, monitor seat availability, and generate reports on travel trends. Role-based access ensures data security and restricts unauthorized access.

6. **Database Management Module**

This module is responsible for securely storing and retrieving all passenger, train, and booking data. MySQL serves as the backend database, ensuring efficient data management and high reliability.

Chapter 2: SURVEY OF TECHNOLOGIES

2.1 SOFTWARE DESCRIPTION

The **Railway Reservation System** utilizes a combination of technologies to ensure robust and efficient functionality. The backend is supported by a relational database management system (RDBMS), while the frontend features an interactive and user-friendly interface built using Java Swing. Middleware technologies enable seamless communication between the backend and frontend.

2.1.1 Java

- **Role:** Java serves as the primary programming language for both backend logic and GUI development.
- **Usage:**
 - Backend: Handles operations like Admin and User registration, Train Schedules, fair details and ticket bookings.
 - Frontend: A JFrame-based GUI provides an intuitive and interactive user experience.
 - Middleware: Java Database Connectivity (JDBC) ensures seamless communication with the MySQL database.
- **Advantages:**
 - Platform independence for cross-platform compatibility.
 - Built-in security features to protect sensitive student data.

2.1.2 MySQL

- **Role:** MySQL is used as the relational database for storing and managing all Admin, User and train information.
- **Usage:**
 - Stores Admin and User profiles, train schedule, fair details, and ticket details.
 - Efficient SQL queries enable quick retrieval and management of large datasets.
- **Advantages:**
 - Reliable, open-source database management.
 - Ensures data integrity and supports complex queries for academic reporting.

Chapter 3: REQUIREMENTS AND ANALYSIS

3.1 REQUIREMENT SPECIFICATION

3.1.1 Functional Requirements

- **User Authentication and Authorization**
 - Enable secure passenger registration and login.
 - Maintain session details for logged-in users.
 - **Passenger Registration and Profile Management**
 - Allow creation and updating of passenger profiles with personal and travel information.
 - Provide a home screen displaying passenger details, such as name, passenger ID, booked tickets, and travel history.
 - **Reservation and Journey Tracking**
 - Enable real-time booking status.
 - Update and display journey details immediately after ticket confirmation.
 - **Database Records Management**
 - Use unique identifiers (e.g., passenger ID) for managing records.
 - Separate tables for passenger profiles, ticket bookings, train schedules, and payment records for efficient data handling.
-

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements

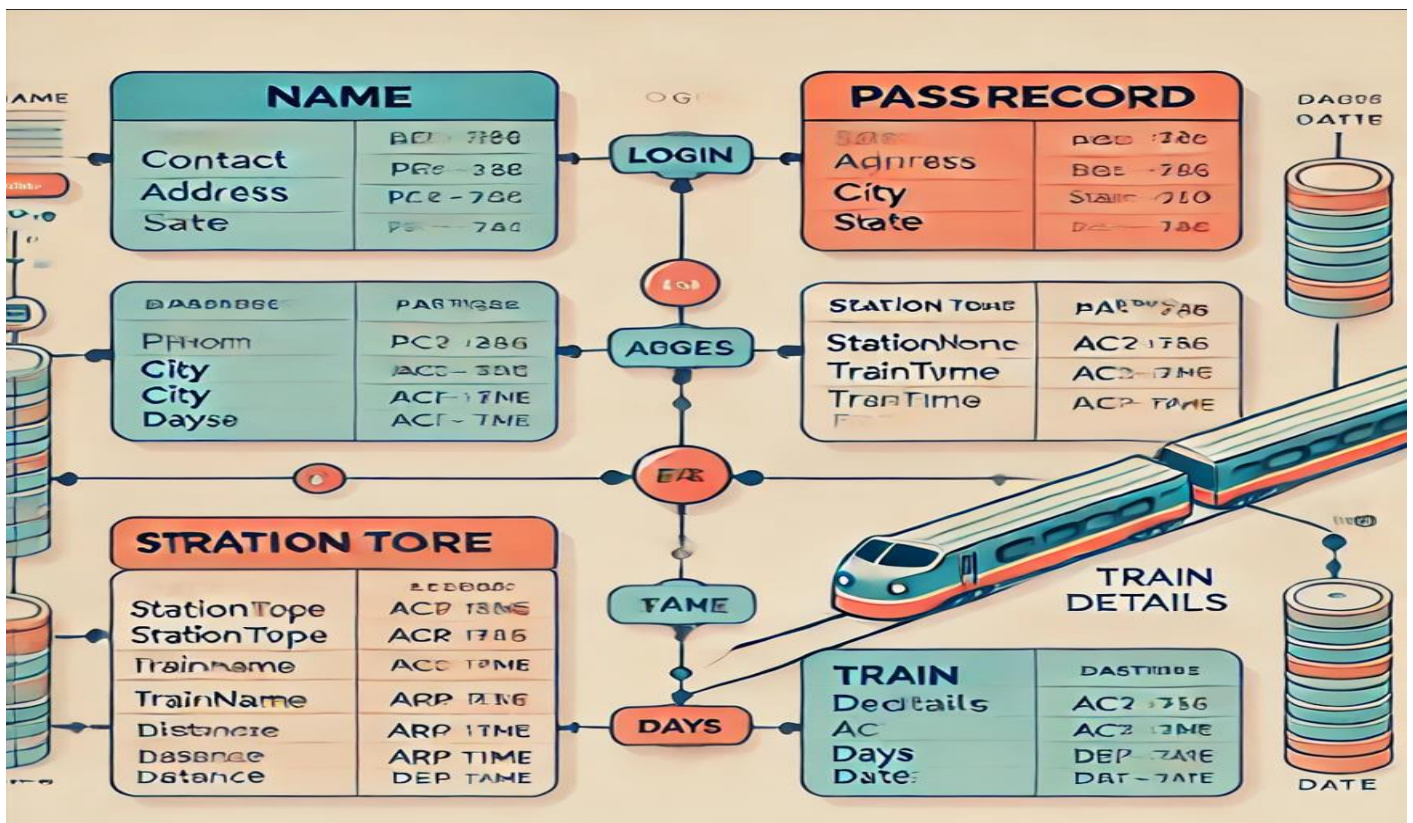
- Processor: Intel Core i3 or equivalent for smooth processing.
- RAM: 4 GB or higher to handle concurrent database operations.
- Storage: At least 500 MB for application files and database storage.
- Monitor Resolution: 1024 x 768 or higher.

Software Requirements

- Operating System: Windows 10 or higher.
- Frontend: Java Swing (JFrame-based interface).

- Backend: MySQL for database management.
- IDE: NetBeans for development.
- Version Control: Git for code versioning and collaboration.

3.3 ER DIAGRAM



Chapter 4 :PROGRAM CODE

1. Backend details

Tables

Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> login		10	InnoDB	latin1_swedish_ci	16.0 KiB	-
<input type="checkbox"/> passrecord		1	InnoDB	latin1_swedish_ci	16.0 KiB	-
<input type="checkbox"/> traindetails		13	InnoDB	latin1_swedish_ci	16.0 KiB	-
<input type="checkbox"/> ulogin		3	InnoDB	latin1_swedish_ci	16.0 KiB	-
4 tables	Sum	27	InnoDB	utf8mb4_general_ci	64.0 KiB	0 B

Admin Login

Name	Contact	Address	Email	City	State	UserName	Password
sdf	123	st	fdxg	dfg	Andhra Pradesh	dfsg	dfg
ss	123	dfs	asd	wr	Andhra Pradesh	dsf	123
sss	12	as	as	as	Andhra Pradesh	as	112233
asdfg	1234567890	asf	ds	jp	Victoria	sa	112233
urse	1234567890	add	urse@gmail.com	jp	Northern Territory	urse123	12345678
urse1	123456789	asfh	urse1@gmail.com	jo	Victoria	urse1234	123456
abcd	123456789	asdfgjh	abcd@123	jlkk	Tasmania	abcd123	asdfghj
admin	1234	abcd	abcd@gmail.com	chennai	Tamil Nadu	admin	admin
hasan	1234	dd	dd	che	Andhra Pradesh	hn	hn
Hari	12345678	address	h10@gmail.com	Chennai	Tamil Nadu	H10	H10

Ticket Details

pName	pGender	pAge	pAction	StationFrom	StationTo	TrainNo	Distance	TrainType	SeetFare	trainName	ArrTime	DepTime
RAM	Male	20	AC1	CHENNAI	MUMBAI	101	2000	Passenger	2000	EXPR	22:00	10:00

Train Schedules

TrainType	TrainNo	Name	StationFrom	StationTo	ArrTime	DepTime	SLFare	AC3Fare	AC2Fare	AC1Fare	Distance	Days	date
train	1	train	train	train	train	train	train	train	train	train	train		
cmbTrainType	sdf	sdf	dsfsd	sd	sd	sd	dsf	dsf	sdf	sdf	sdf		
cmbTrainType	123	asd	asd	as	arrTime	depTime	12	21	23	45	23	S,M,T	
Item 2	1231	123	as	sd	11:11	11:11	af	asd	asd	asd	asd	sdas	sd
SupperFast	12	dszf	awd	asd	12:12	12:42	12	12	12	12	12	sds	12/12/12
Local	112233	lacto	jp	aj	11:11	12:10	231	343	543	654	134	S,M,W	12/12/12
SupperFast	11223344	asdfghjkl	jp	jp	09:10	11:11	213	321	431	543	123	S,T,W	11/11/11
Local	456	fshf	ndbxjcf	hds bhc	11:23	12:45	hnsdbch	jdsbjkcd	jbdjc	ds bjc	dmn	334	dhns c
Passenger	8789	bvb ,z	n bj,	dfbadf	23:16	11:55	dfbd	gdb	db	dgbd	bdgxb	2	23:23
Express	12345	bbbb	bbbb	bbbb	12:12	02:56	jdbkjsa	dbsvz	ndbsmnfv	hdvhfjs	jnsbzbv	gjd	bbbb
Passenger	11	hnn	chennai	kel	11:00	22:00	500	1000	1500	2000	200	S,M,W	12/11/2024
Passenger	111	has	chennai	kerala	11:00	22:00	500	1000	1500	2000	200	S,M,W	13/11/2024
Passenger	101	EXPR	CHENNAI	MUMBAI	22:00	10:00	500	1000	1500	2000	2000	S,M,W	19/11/2024

User Login

Name	Contact	Address	Email	City	State	UserName	Password
M	1 1		m@gmail.com	m	Tamil Nadu	M	M
nn	11222	aa	aa@gmail.com	che	Andhra Pradesh	nn	nn
RAM	12345678	ADDRESS	R7@gmail.com	CHENNAI	Tamil Nadu	R7	R7

2. Method for Initializing Components

```
/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code"> // GEN-BEGIN: initComponents
private void initComponents() {
    jLabel1 = new javax.swing.JLabel();
    btnTicketBooking = new javax.swing.JButton();
    btnTrainSchedule = new javax.swing.JButton();
    btnTrainFind = new javax.swing.JButton();
    btnFareEnquiry = new javax.swing.JButton();
    btnAddTrain = new javax.swing.JButton();
    btnPassengerDetails = new javax.swing.JButton();
    btnLogout = new javax.swing.JButton();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

    jLabel1.setFont(new java.awt.Font("Tahoma", 1, 36)); // NOI18N
    jLabel1.setForeground(new java.awt.Color(0, 0, 204));
```

```
jLabel1.setText("ADMIN DASHBORD");
```

3. JDBC Connectivity

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class AdminLoginJDBC {

    public static void main(String[] args) {
        // Database credentials
        String url = "jdbc:mysql://localhost:3306/railway_system"; // Replace with your
        database name
        String username = "root"; // Replace with your MySQL username
        String password = "password"; // Replace with your MySQL password

        // Declare the Connection and Statement objects
        Connection conn = null;
        Statement stmt = null;

        try {
            // Step 1: Load the JDBC driver for MySQL
            Class.forName("com.mysql.cj.jdbc.Driver");

            // Step 2: Establish the connection to the database
            conn = DriverManager.getConnection(url, username, password);

            // Step 3: Create a Statement object to send SQL commands
            stmt = conn.createStatement();

            // Step 4: Execute a query to select data (example: fetching admin login info)
            String sql = "SELECT * FROM admin_table"; // Replace with your actual table name
            ResultSet rs = stmt.executeQuery(sql);

            // Step 5: Process the ResultSet
            while (rs.next()) {
                // Assuming 'id', 'username', and 'password' are columns in your 'admin_table'
                int id = rs.getInt("id");
                String dbUsername = rs.getString("username");
                String dbPassword = rs.getString("password");

                // Print the result (or use this data for your login validation)
                System.out.println("ID: " + id + ", Username: " + dbUsername + ", Password: " +
                dbPassword);
            }
        }
```

```

        // Close the ResultSet
        rs.close();

    } catch (SQLException se) {
        // Handle SQL errors
        se.printStackTrace();
    } catch (Exception e) {
        // Handle Class.forName errors
        e.printStackTrace();
    } finally {
        // Step 6: Close the resources
        try {
            if (stmt != null) stmt.close(); // Close the Statement
        } catch (SQLException se) {
            se.printStackTrace();
        }
        try {
            if (conn != null) conn.close(); // Close the Connection
        } catch (SQLException se) {
            se.printStackTrace();
        }
    }
}
}
}

```

4. Admin Dashboard

```

// Button action listeners for the Admin Dashboard
private void btnTicketBookingActionPerformed(java.awt.event.ActionEvent evt) {
    TicketBookingForm tbf = new TicketBookingForm();
    tbf.setVisible(true);
    this.dispose();
}

private void btnTrainScheduleActionPerformed(java.awt.event.ActionEvent evt) {
    TrainScheduleForm ts = new TrainScheduleForm();
    ts.setVisible(true);
    this.dispose();
}

private void btnTrainFindActionPerformed(java.awt.event.ActionEvent evt) {
    TrainBetweenStation tbs = new TrainBetweenStation();
    tbs.setVisible(true);
    this.dispose();
}

```

```

private void btnFareEnquiryActionPerformed(java.awt.event.ActionEvent evt) {
    TrainFareEnquiryForm tfq = new TrainFareEnquiryForm();
    tfq.setVisible(true);
    this.dispose();
}

private void btnAddTrainActionPerformed(java.awt.event.ActionEvent evt) {
    TrainsAdd ta = new TrainsAdd();
    ta.setVisible(true);
    this.dispose();
}

private void btnPassengerDetailsActionPerformed(java.awt.event.ActionEvent evt) {
    PassengerDetailsForm pdf = new PassengerDetailsForm();
    pdf.setVisible(true);
    this.dispose();
}

private void btnLogoutActionPerformed(java.awt.event.ActionEvent evt) {
    HomeScreen hs = new HomeScreen();
    hs.setVisible(true);
    this.dispose();
}

```

5. Admin Login

```

// Import packages and setup UI for AdminLogin
import java.awt.HeadlessException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.swing.JOptionPane;

public class AdminLogin extends javax.swing.JFrame {

    public AdminLogin() {
        initComponents();
    }

    // Method to initialize the form components
    @SuppressWarnings("unchecked")
    private void initComponents() {
        // Setting up the components like labels, buttons, text fields
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
    }

```

```

jLabel3 = new javax.swing.JLabel();
txtUserName = new javax.swing.JTextField();
btnLogin = new javax.swing.JButton();
btnCancel = new javax.swing.JButton();
btnForgotPassword = new javax.swing.JButton();
btnRegistration = new javax.swing.JButton();
jScrollPane1 = new javax.swing.JScrollPane();
txtPasword = new javax.swing.JPasswordField();

// Initialize components
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

// Add action listeners for buttons
btnLogin.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnLoginActionPerformed(evt);
    }
});

btnCancel.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnCancelActionPerformed(evt);
    }
});

btnForgotPassword.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnForgotPasswordActionPerformed(evt);
    }
});

btnRegistration.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnRegistrationActionPerformed(evt);
    }
});
}
}

```

6. User Login

```

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class UserLogin extends JFrame {

    // Declare components

```

```
private JLabel lblUsername, lblPassword, lblMessage;
private JTextField txtUsername;
private JPasswordField txtPassword;
private JButton btnLogin;

public UserLogin() {
    // Set up the frame
    setTitle("User Login");
    setSize(400, 250);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLocationRelativeTo(null); // Center the window

    // Initialize components
    lblUsername = new JLabel("Username:");
    lblPassword = new JLabel("Password:");
    lblMessage = new JLabel("");
    lblMessage.setForeground(java.awt.Color.RED); // Error message in red

    txtUsername = new JTextField(20);
    txtPassword = new JPasswordField(20);

    btnLogin = new JButton("Login");

    // Layout setup
    setLayout(null);

    lblUsername.setBounds(50, 30, 100, 30);
    txtUsername.setBounds(150, 30, 200, 30);

    lblPassword.setBounds(50, 70, 100, 30);
    txtPassword.setBounds(150, 70, 200, 30);

    btnLogin.setBounds(150, 110, 100, 30);

    lblMessage.setBounds(50, 150, 300, 30);

    // Add components to frame
    add(lblUsername);
    add(txtUsername);
    add(lblPassword);
    add(txtPassword);
    add(btnLogin);
    add(lblMessage);

    // ActionListener for login button
    btnLogin.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
```



```

        // Get username and password
        String username = txtUsername.getText();
        char[] password = txtPassword.getPassword();

        // Simple validation (you can replace this with database verification)
        if (username.equals("admin") && String.valueOf(password).equals("password123"))
        {
            lblMessage.setText("Login Successful!");
            lblMessage.setForeground(java.awt.Color.GREEN);
            // Open the next window or dashboard after successful login
            // Example: new AdminDashboard().setVisible(true);
            // dispose(); // Close the login window after successful login
        } else {
            lblMessage.setText("Invalid username or password!");
            lblMessage.setForeground(java.awt.Color.RED);
        }
    }
    });
}

public static void main(String[] args) {
    // Run the login window
    SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
            new UserLogin().setVisible(true);
        }
    });
}
}

```

7. Train Schedule

```

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;
import java.sql.*;
import java.util.ArrayList;

public class TrainScheduleForm extends JFrame {
    private JTextField txtFromStation, txtToStation;
    private JButton btnViewSchedule;
    private JTextArea txtScheduleDetails;

    public TrainScheduleForm() {
        // Initialize the form components
        setTitle("Train Schedule");
    }
}

```

```

setSize(400, 400);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setLayout(null);

JLabel lblFrom = new JLabel("From Station:");
lblFrom.setBounds(30, 30, 100, 30);
add(lblFrom);

txtFromStation = new JTextField();
txtFromStation.setBounds(150, 30, 150, 30);
add(txtFromStation);

JLabel lblTo = new JLabel("To Station:");
lblTo.setBounds(30, 70, 100, 30);
add(lblTo);

txtToStation = new JTextField();
txtToStation.setBounds(150, 70, 150, 30);
add(txtToStation);

btnViewSchedule = new JButton("View Schedule");
btnViewSchedule.setBounds(30, 110, 150, 30);
add(btnViewSchedule);

txtScheduleDetails = new JTextArea();
txtScheduleDetails.setBounds(30, 150, 300, 200);
txtScheduleDetails.setEditable(false);
add(txtScheduleDetails);

// Add action listener for view schedule button
btnViewSchedule.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        viewTrainSchedule();
    }
});
}

private void viewTrainSchedule() {
    String fromStation = txtFromStation.getText();
    String toStation = txtToStation.getText();
    StringBuilder scheduleDetails = new StringBuilder();

    try {
        Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/train_system", "root",
"password");
        Statement stmt = conn.createStatement();

```

```

        String query = "SELECT * FROM trains WHERE from_station = '" + fromStation + "' AND
to_station = '" + toStation + "'";
        ResultSet rs = stmt.executeQuery(query);

        if (rs.next()) {
            // Display train schedule details
            do {
                String trainName = rs.getString("train_name");
                String departureTime = rs.getString("departure_time");
                String arrivalTime = rs.getString("arrival_time");
                scheduleDetails.append("Train: " + trainName + "\n");
                scheduleDetails.append("Departure: " + departureTime + "\n");
                scheduleDetails.append("Arrival: " + arrivalTime + "\n\n");
            } while (rs.next());

            txtScheduleDetails.setText(scheduleDetails.toString());
        } else {
            JOptionPane.showMessageDialog(this, "No trains available for the selected route.");
            txtScheduleDetails.setText("");
        }
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Error: " + e.getMessage());
    }
}

public static void main(String[] args) {
    TrainScheduleForm form = new TrainScheduleForm();
    form.setVisible(true);
}
}

```

8. Ticket Booking

```

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

public class TicketBookingForm extends JFrame {
    private JTextField txtFromStation, txtToStation, txtTravelDate, txtPassengerName;
    private JButton btnCheckAvailability, btnBookTicket;

    public TicketBookingForm() {
        // Initialize the form components
        setTitle("Ticket Booking");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

```

```
setLayout(null);

JLabel lblFrom = new JLabel("From Station:");
lblFrom.setBounds(30, 30, 100, 30);
add(lblFrom);

txtFromStation = new JTextField();
txtFromStation.setBounds(150, 30, 150, 30);
add(txtFromStation);

JLabel lblTo = new JLabel("To Station:");
lblTo.setBounds(30, 70, 100, 30);
add(lblTo);

txtToStation = new JTextField();
txtToStation.setBounds(150, 70, 150, 30);
add(txtToStation);

JLabel lblDate = new JLabel("Travel Date:");
lblDate.setBounds(30, 110, 100, 30);
add(lblDate);

txtTravelDate = new JTextField();
txtTravelDate.setBounds(150, 110, 150, 30);
add(txtTravelDate);

JLabel lblPassengerName = new JLabel("Passenger Name:");
lblPassengerName.setBounds(30, 150, 120, 30);
add(lblPassengerName);

txtPassengerName = new JTextField();
txtPassengerName.setBounds(150, 150, 150, 30);
add(txtPassengerName);

btnCheckAvailability = new JButton("Check Availability");
btnCheckAvailability.setBounds(30, 200, 150, 30);
add(btnCheckAvailability);

btnBookTicket = new JButton("Book Ticket");
btnBookTicket.setBounds(200, 200, 120, 30);
add(btnBookTicket);

// Add action listener to check availability
btnCheckAvailability.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        checkTrainAvailability();
    }
});
```

```

// Add action listener to book ticket
btnBookTicket.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        bookTicket();
    }
});
}

private void checkTrainAvailability() {
    String fromStation = txtFromStation.getText();
    String toStation = txtToStation.getText();
    String travelDate = txtTravelDate.getText();

    try {
        Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/train_system", "root",
"password");
        Statement stmt = conn.createStatement();
        String query = "SELECT * FROM trains WHERE from_station = '" + fromStation + "' AND
to_station = '" + toStation + "' AND travel_date = '" + travelDate + "'";
        ResultSet rs = stmt.executeQuery(query);

        if (rs.next()) {
            JOptionPane.showMessageDialog(this, "Train Available! Proceed to Book.");
            btnBookTicket.setEnabled(true); // Enable the booking button if train is available
        } else {
            JOptionPane.showMessageDialog(this, "No train available for the selected route and
date.");
            btnBookTicket.setEnabled(false);
        }
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Error: " + e.getMessage());
    }
}

private void bookTicket() {
    String fromStation = txtFromStation.getText();
    String toStation = txtToStation.getText();
    String travelDate = txtTravelDate.getText();
    String passengerName = txtPassengerName.getText();

    try {
        Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/train_system", "root",
"password");
        String query = "INSERT INTO ticket_bookings (from_station, to_station, travel_date,
passenger_name) VALUES (?, ?, ?, ?)";

```

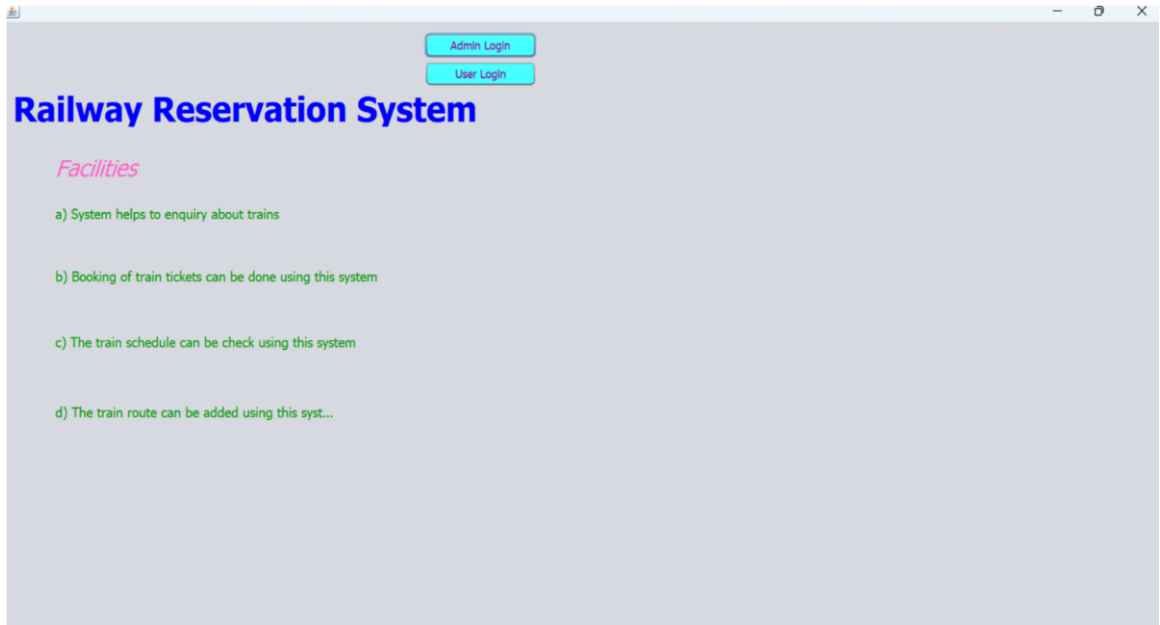
```
PreparedStatement pstmt = conn.prepareStatement(query);
pstmt.setString(1, fromStation);
pstmt.setString(2, toStation);
pstmt.setString(3, travelDate);
pstmt.setString(4, passengerName);

int rowsAffected = pstmt.executeUpdate();
if (rowsAffected > 0) {
    JOptionPane.showMessageDialog(this, "Ticket booked successfully!");
} else {
    JOptionPane.showMessageDialog(this, "Booking failed. Try again later.");
}
} catch (SQLException e) {
    JOptionPane.showMessageDialog(this, "Error: " + e.getMessage());
}
}

public static void main(String[] args) {
    TicketBookingForm form = new TicketBookingForm();
    form.setVisible(true);
}
}
```

Chapter 5: OUTPUTS

1. Home Page:



A screenshot of a web browser window displaying the home page of a 'Railway Reservation System'. The page has a light gray background. At the top right, there are two cyan buttons: 'Admin Login' and 'User Login'. Below these, the title 'Railway Reservation System' is displayed in a large, bold, blue font. Underneath the title, the word 'Facilities' is written in a pink, italicized font. A list of four facilities follows, each preceded by a green letter and a description: 'a) System helps to enquiry about trains', 'b) Booking of train tickets can be done using this system', 'c) The train schedule can be check using this system', and 'd) The train route can be added using this syst...'. The browser window includes standard OS controls (minimize, maximize, close) in the top right corner.

Admin Login
User Login

Railway Reservation System

Facilities

- a) System helps to enquiry about trains
- b) Booking of train tickets can be done using this system
- c) The train schedule can be check using this system
- d) The train route can be added using this syst...

2. Admin login



A screenshot of a web browser window displaying the 'ADMIN LOGIN' page. The page has a light gray background. The title 'ADMIN LOGIN' is centered at the top in a bold, blue font. Below the title, there are two labels in red: 'User Name' and 'Password'. Each label is followed by a white text input field. At the bottom of the page, there are four cyan buttons: 'Login', 'Registration', 'Cancel', and 'Forget Password'. The browser window includes standard OS controls (minimize, maximize, close) in the top right corner.

ADMIN LOGIN

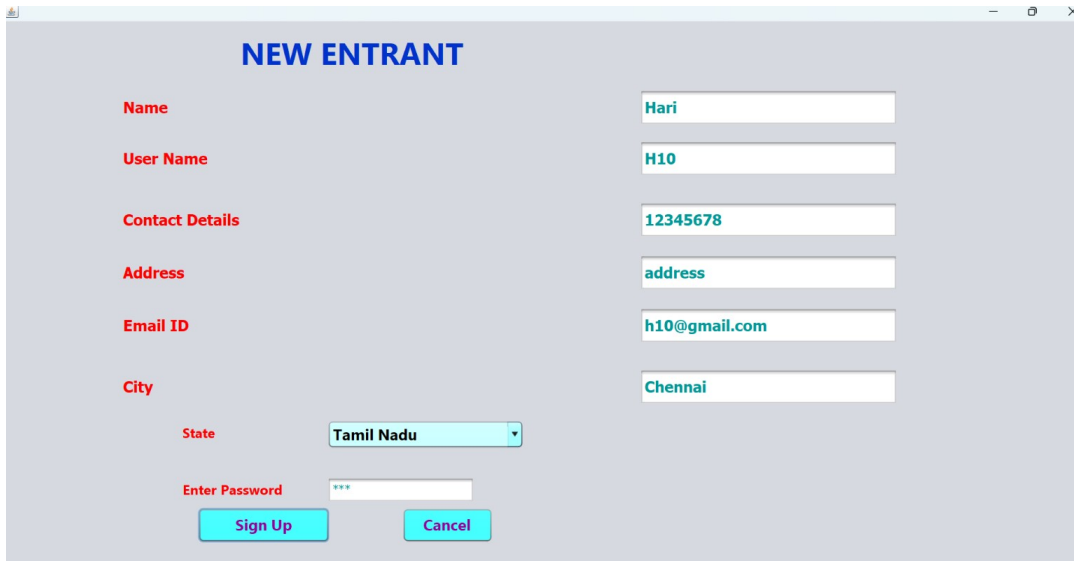
User Name

Password

Login Registration Cancel

Forget Password

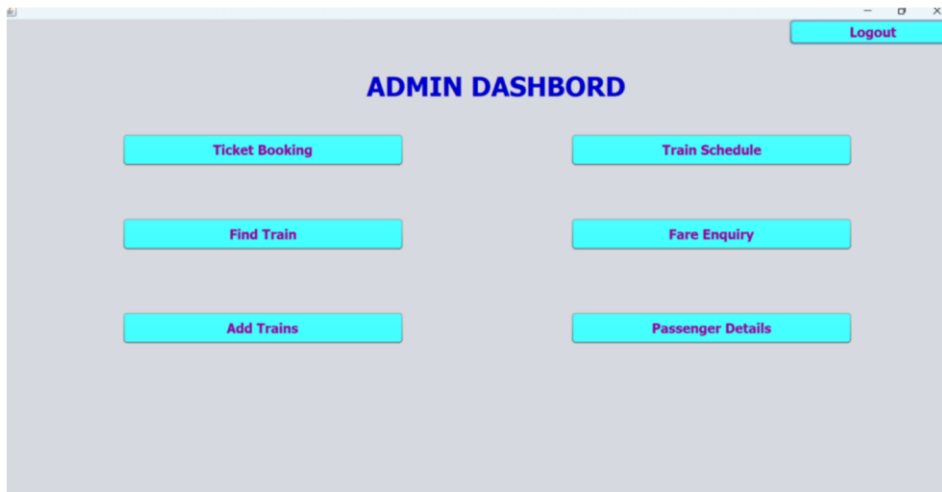
3. Sign up



A screenshot of a web application window titled "NEW ENTRANT". The form contains several input fields for user registration. The labels for the fields are in red text, and the input values are in blue text. At the bottom, there are two buttons: "Sign Up" and "Cancel".

Field Label	Input Value
Name	Hari
User Name	H10
Contact Details	12345678
Address	address
Email ID	h10@gmail.com
City	Chennai
State	Tamil Nadu
Enter Password	****

4. Admin Dashboard



A screenshot of a web application window titled "ADMIN DASHBORD". The interface features a "Logout" button in the top right corner and six main functional buttons arranged in a 3x2 grid. All buttons are blue with black text.

Function
Ticket Booking
Train Schedule
Find Train
Fare Enquiry
Add Trains
Passenger Details

5. Adding train details

Select Train: Passenger

Train No: Name: Date:

From: To: Arrival Time: Departure Time: Days:

Sleeper Fare: AC3 Fare: AC2 Fare: AC1 Fare: Distance:

Select an Option

Do you want to Add?

6. Train Schedule Page

TRAIN SCHEDULE

Train Number

SR No.	Station From	Arrival	Station To	Departure	Distance	Day
101	CHENNAI	22:00	MUMBAI	10:00	2000	S,M,W

7.User Dashboard

NEW USER

Name

User Name

Contact Details

Address

Email ID

City

State

Enter Password

Sign Up **Cancel**

RAM

R7

12345678

ADDRESS

R7@gmail.com

CHENNAI

Tamil Nadu

**

USER LOGIN

User Name

Password

Login **Registration** **Cancel**

Forgot Password

R7

**

8.Ticket Booking

BACK

TICKET BOOKING

FROM

CHENNAI

TO

MUMBAI

JOURNEY DATE

19/11/2024

SEARCH

BACK

PASSENGER DETAILS

Train Number

101

Seat Type

AC1

From

CHENNAI

Gender

Male

To

MUMBAI

Name

RAM

Age

20

Book Ticket

Reset Form

Booked Ticket

Train Name	EXPR	Train No	101	Train Type	Passenger
Pass Name	RAM	Age	20	Gender	Male
From	To	Total Distance	Seat Type		
CHENNAI	MUMBAI	2000	AC1		
Departure	Arrived				
10:00	22:00				
	Total Fare	2000			

Chapter 6: Conclusion

6.1 Conclusion

The development of the Railway Reservation System represents a significant advancement in ticket booking and travel management, enhancing accessibility to essential train schedules and passenger information. This project successfully achieves its core objectives of simplifying the reservation process, managing passenger records, and providing a comprehensive view of travel itineraries, ultimately improving operational efficiency and decision-making processes.

Built using Java with a JDBC connection to a MySQL database, the system is secure, robust, and efficient in handling real-time travel data. The use of Java JFrame ensures a user-friendly, sleek interface, making interactions such as ticket booking, and schedule updates intuitive and streamlined. MySQL contributes to efficient database management, supporting high volumes of passenger data with optimal performance and reliability.

With comprehensive functional capabilities, including user authentication, ticket management, train schedule tracking, and fare calculations, the system meets critical requirements for railway reservation. Nonfunctional aspects such as security, performance, scalability, and maintainability further reinforce the system's operational stability, ensuring administrators, operators, and passengers experience uninterrupted, secure, and responsive service.

In conclusion, the Railway Reservation System effectively addresses key challenges in travel management through an integrated approach, employing robust backend technologies and a user-focused design. It sets a standard for efficiency and security in reservation management, providing a scalable solution that can adapt to future enhancements while consistently delivering an exceptional user experience and reliable travel management tools.

Chapter 7: REFERENCE

7.1 REFERENCES

1. <https://www.javatpoint.com/java-tutorial>
2. <https://www.wikipedia.org/>
3. <https://www.w3schools.com/sql/>
4. SQL | Codecademy