**Online Train Ticket Booking System**

**A MINI-PROJECT REPORT**

**Submitted by**

**Sanjay Kumar.K 230701289**

**Sharukesh.S 230701308**

**In partial fulfillment of the award of the degree** **of**

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**



**RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI**

**An Autonomous Institute**

**CHENNAI**

**NOVEMBER 2023**

# 

# BONAFIDE CERTIFICATE

Certified that this project “Online Train Ticket Booking System ” is the

bonafide work of “Sanjay Kumar.k and Sharukesh.S” who carried out the project work under my supervision.

SIGNATURE SIGNATURE

MR.SARAVANA GOKUL.G MRS.JANANEE

ASSISTANT PROFESSOR ASSISTANT PROFESSOR

Dept. of Computer Science and Engg, Dept. of Computer Science and Engg,

Rajalakshmi Engineering College Rajalakshmi Engineering College

Chennai Chennai

# This mini project report is submitted for the viva voce examination to be held on \_\_\_\_\_\_

**Project Documentation: Train**

**Ticket Booking System**

1. **Project Overview** 
   1. Project Title

Train Ticket Booking System

* 1. Objective:

The objective of this project is to develop a Java-based desktop application for train ticket booking. The system provides functionalities for users to book train tickets, search train schedules, and view their booking history. Additionally, it includes an admin panel for managing train schedules and viewing bookings.

1. **System Features:**
   1. User Features
      1. **User Registration and Login**
         * Secure authentication system for users.
      2. **Train Search**
         * Search for available trains using source and destination locations.

# 3. Ticket Booking

o Book tickets by selecting train, date, and the number of seats.

4. **View Booking History**

o Displays all bookings made by the user.

# 5. Notifications

o Email notifications for successful bookings.

2.2 Admin Features

1. **Admin Login**
   * Secure login for admin users.
2. **Train Management**
   * Add, update, and delete train details.

# 3. View Bookings

o View all user bookings.

# 4. Dashboard

o Overview of system usage and analytics.

# 3. Technology Stack

3.1 Programming Language

* Java (Swing for GUI)

3.2 Database

* MySQL

3.3 Tools

* MySQL Workbench
* IDE (IntelliJ IDEA, Eclipse, or NetBeans)

# 4. Application Architecture

4.1 File Structure

TrainTicketBookingSystem/

├── Main.java # Entry point of the application

├── User/

│ ├── Login.java # User login functionality

│ ├── Register.java # User registration functionality

│ ├── UserDashboard.java # User dashboard screen

│ ├── SearchTrain.java # Train search screen

│ ├── BookTicket.java # Ticket booking screen

│ ├── BookingHistory.java # User booking history screen

│

├── Admin/

│ ├── AdminLogin.java # Admin login screen

│ ├── AdminDashboard.java # Admin dashboard screen

│ ├── AddTrain.java # Screen to add train details

│ ├── ViewBookings.java # Screen to view all bookings ├── Models/

│ ├── Train.java # Train details model

│ ├── User.java # User details model

│ ├── Ticket.java # Ticket booking model

├── Utils/

│ ├── DatabaseConnection.java # Handles database connections

│ ├── ValidationUtils.java # Utility methods for input validation

1. **Database Schema**

5.1 Tables

# 1. users

oStores user details (e.g., user\_id, username, password, email, etc.).

# 2. trains

oStores train details (e.g., train\_id, train\_name, source, destination, seat\_capacity, etc.).

# 3. tickets

oStores ticket booking details (e.g., ticket\_id, user\_id, train\_id, seats\_booked, booking\_date, etc.).

# 4. admins

oStores admin login credentials.

5.2 MySQL Schema:

CREATE DATABASE TrainBooking;

USE TrainBooking;

CREATE TABLE users (

user\_id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(100) NOT NULL,

password VARCHAR(100) NOT NULL,

email VARCHAR(100) NOT NULL

);

CREATE TABLE trains (

train\_id INT AUTO\_INCREMENT PRIMARY KEY,

train\_name VARCHAR(100) NOT NULL,

source VARCHAR(100) NOT NULL,

destination VARCHAR(100) NOT NULL, seat\_capacity INT NOT NULL

);

CREATE TABLE tickets (

ticket\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT,

train\_id INT,

seats\_booked INT NOT NULL,

booking\_date DATE NOT NULL,

FOREIGN KEY (user\_id) REFERENCES users(user\_id),

FOREIGN KEY (train\_id) REFERENCES trains(train\_id)

);

CREATE TABLE admins (

admin\_id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(100) NOT NULL,

password VARCHAR(100) NOT NULL

);

1. **Implementation Details**

6.1 Core Functionalities

# 1. Database Connection

o The DatabaseConnection.java file provides a reusable method for establishing a connection with the MySQL database.

# 2. GUI Implementation

o Swing is used for creating user interfaces (e.g., login forms, dashboards, and search screens).

# 3. Back Button Navigation

o Each screen has a back button implemented using CardLayout to allow users to navigate between screens.

Main.java:

package main;

import screens.SearchTrain; import screens.BookTicket; import javax.swing.\*; import java.awt.event.ActionEvent; import java.awt.event.ActionListener;

public class Main extends JFrame {

public Main() { setTitle("Train Ticket Booking System"); setSize(400, 400); setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(null);

// Create buttons for navigation

JButton searchTrainButton = new JButton("Search Trains");

JButton bookTicketButton = new JButton("Book Ticket");

JButton adminPanelButton = new JButton("Admin Panel");

JButton exitButton = new JButton("Exit");

// Set bounds for buttons

searchTrainButton.setBounds(100, 50, 200, 30); bookTicketButton.setBounds(100, 100, 200, 30); adminPanelButton.setBounds(100, 150, 200, 30); exitButton.setBounds(100, 200, 200, 30);

// Add action listeners searchTrainButton.addActionListener(new ActionListener() {

@Override public void actionPerformed(ActionEvent e) { new SearchTrain();

}

});

bookTicketButton.addActionListener(new ActionListener() {

@Override public void actionPerformed(ActionEvent e) {

String trainNoStr = JOptionPane.showInputDialog(null, "Enter Train Number to Book:"); if (trainNoStr != null) { try {

int trainNo = Integer.parseInt(trainNoStr); new BookTicket(trainNo);

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(null, "Invalid Train Number.");

}

}

}

});

adminPanelButton.addActionListener(new ActionListener() {

@Override public void actionPerformed(ActionEvent e) { // Replace with admin login check if required new AdminPanel();

}

});

exitButton.addActionListener(new ActionListener() {

@Override public void actionPerformed(ActionEvent e) {

int confirm = JOptionPane.showConfirmDialog(null, "Are you sure you want

to exit?", "Exit", JOptionPane.YES\_NO\_OPTION); if (confirm == JOptionPane.YES\_OPTION) {

System.exit(0);

}

}

});

ConnectionManager.java:

package database;

import java.sql.Connection; import java.sql.DriverManager; import java.sql.SQLException;

public class ConnectionManager {

private static final String URL = "jdbc:mysql://localhost:3306/TrainTicketBooking"; // Update with your DB details private static final String USERNAME = "root"; // Update with your username private static final String PASSWORD = "root123"; // Update with your password

public static Connection getConnection() throws SQLException {

try {

Class.forName("com.mysql.cj.jdbc.Driver"); } catch (ClassNotFoundException e) { throw new SQLException("MySQL JDBC Driver not found", e);

}

return DriverManager.getConnection(URL, USERNAME, PASSWORD);

}

}

RegisterUser.java:

package ui.user;

import javax.swing.\*;

import database.ConnectionManager;

import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.sql.Connection; import java.sql.PreparedStatement;

public class RegisterUser extends JFrame { private JTextField usernameField, emailField; private JPasswordField passwordField;

public RegisterUser() { setTitle("User Registration"); setLayout(null);

JLabel usernameLabel = new JLabel("Username:"); usernameLabel.setBounds(50, 50, 100, 30); add(usernameLabel);

usernameField = new JTextField(); usernameField.setBounds(150, 50, 150, 30); add(usernameField);

JLabel emailLabel = new JLabel("Email:"); emailLabel.setBounds(50, 100, 100, 30); add(emailLabel);

emailField = new JTextField(); emailField.setBounds(150, 100, 150, 30); add(emailField);

JLabel passwordLabel = new JLabel("Password:"); passwordLabel.setBounds(50, 150, 100, 30); add(passwordLabel);

passwordField = new JPasswordField(); passwordField.setBounds(150, 150, 150, 30); add(passwordField);

JButton registerButton = new JButton("Register"); registerButton.setBounds(150, 200, 100, 30); add(registerButton);

registerButton.addActionListener(new ActionListener() {

@Override public void actionPerformed(ActionEvent e) { registerUser();

}

});

setSize(400, 300); setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setVisible(true);

}

private void registerUser() {

String username = usernameField.getText();

String email = emailField.getText();

String password = new String(passwordField.getPassword());

try (Connection conn = ConnectionManager.getConnection()) {

String sql = "INSERT INTO Users (username, email, password) VALUES (?,

?, ?)";

PreparedStatement stmt = conn.prepareStatement(sql); stmt.setString(1, username); stmt.setString(2, email); stmt.setString(3, password); stmt.executeUpdate();

JOptionPane.showMessageDialog(this, "User Registered Successfully!");

} catch (Exception ex) {

JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage());

}

}

public static void main(String[] args) { new RegisterUser();

}

}

SearchTrain.java:

package screens;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import database.ConnectionManager;

import java.awt.\*; import java.sql.Connection; import java.sql.PreparedStatement; import java.sql.ResultSet;

public class SearchTrain extends JFrame { private JTextField sourceField, destinationField; private JTable trainTable;

public SearchTrain() { setTitle("Search Train"); setSize(800, 600); setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); setLayout(new BorderLayout());

// Search panel

JPanel searchPanel = new JPanel(new GridLayout(3, 2, 10, 10)); searchPanel.setBorder(BorderFactory.createTitledBorder("Search Train"));

sourceField = new JTextField(); destinationField = new JTextField(); searchPanel.add(new JLabel("Source:")); searchPanel.add(sourceField); searchPanel.add(new JLabel("Destination:")); searchPanel.add(destinationField);

JButton searchButton = new JButton("Search"); searchButton.addActionListener(e -> searchTrains()); searchPanel.add(new JLabel()); searchPanel.add(searchButton);

// Train table

trainTable = new JTable(new DefaultTableModel(new Object[]{"Train No", "Train Name", "Source", "Destination", "Available Seats"}, 0));

JScrollPane scrollPane = new JScrollPane(trainTable);

add(searchPanel, BorderLayout.NORTH); add(scrollPane, BorderLayout.CENTER);

setVisible(true);

}

private void searchTrains() {

String source = sourceField.getText().trim();

String destination = destinationField.getText().trim();

if (source.isEmpty() || destination.isEmpty()) {

JOptionPane.showMessageDialog(this, "Source and Destination cannot be empty!");

return;

}

try (Connection conn = ConnectionManager.getConnection()) {

String query = "SELECT train\_no, train\_name, source, destination, available\_seats FROM Train WHERE source = ? AND destination = ?"; PreparedStatement stmt = conn.prepareStatement(query); stmt.setString(1, source); stmt.setString(2, destination);

ResultSet rs = stmt.executeQuery();

DefaultTableModel model = (DefaultTableModel) trainTable.getModel(); model.setRowCount(0);

while (rs.next()) { model.addRow(new Object[]{

rs.getInt("train\_no"), rs.getString("train\_name"), rs.getString("source"), rs.getString("destination"), rs.getInt("available\_seats")

});

}

if (model.getRowCount() == 0) {

JOptionPane.showMessageDialog(this, "No trains found for the selected route.");

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(this, "Error while searching trains: " + ex.getMessage());

}

}

public static void main(String[] args) { new SearchTrain();

}

}

BookTicket.java:

package screens;

import javax.swing.\*;

import database.ConnectionManager;

import java.awt.\*; import java.sql.Connection; import java.sql.PreparedStatement; import java.sql.ResultSet; import java.util.Date;

public class BookTicket extends JFrame { private JTextField trainNoField, passengerNameField, emailField, seatCountField;

public BookTicket(int trainNo) { setTitle("Book Ticket"); setSize(400, 400); setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); setLayout(new GridLayout(6, 2, 10, 10));

// Fields for booking trainNoField = new JTextField(String.valueOf(trainNo)); trainNoField.setEditable(false); passengerNameField = new JTextField(); emailField = new JTextField(); seatCountField = new JTextField();

add(new JLabel("Train No:")); add(trainNoField); add(new JLabel("Passenger Name:")); add(passengerNameField); add(new JLabel("Email:")); add(emailField); add(new JLabel("Number of Seats:")); add(seatCountField);

JButton bookButton = new JButton("Book Ticket"); bookButton.addActionListener(e -> bookTicket()); add(new JLabel()); add(bookButton);

setVisible(true);

}

private void bookTicket() { int trainNo = Integer.parseInt(trainNoField.getText());

String passengerName = passengerNameField.getText().trim(); String email = emailField.getText().trim(); int numberOfSeats;

try {

numberOfSeats = Integer.parseInt(seatCountField.getText());

} catch (NumberFormatException e) {

JOptionPane.showMessageDialog(this, "Invalid seat count.");

return;

}

if (passengerName.isEmpty() || email.isEmpty() || numberOfSeats <= 0) { JOptionPane.showMessageDialog(this, "Please fill in all fields correctly.");

return;

}

try (Connection conn = ConnectionManager.getConnection()) {

// Check seat availability

String checkSeatsQuery = "SELECT available\_seats FROM Train WHERE train\_no = ?";

PreparedStatement checkStmt = conn.prepareStatement(checkSeatsQuery); checkStmt.setInt(1, trainNo);

ResultSet rs = checkStmt.executeQuery();

if (rs.next()) { int availableSeats = rs.getInt("available\_seats");

if (availableSeats < numberOfSeats) {

JOptionPane.showMessageDialog(this, "Not enough seats available.");

return;

}

} else {

JOptionPane.showMessageDialog(this, "Train not found.");

return;

}

// Deduct seats and book ticket

String bookTicketQuery = "INSERT INTO Ticket (train\_no, passenger\_name, email, number\_of\_seats, booking\_date) VALUES (?, ?, ?, ?, ?)";

PreparedStatement bookStmt = conn.prepareStatement(bookTicketQuery); bookStmt.setInt(1, trainNo); bookStmt.setString(2, passengerName); bookStmt.setString(3, email);

bookStmt.setInt(4, numberOfSeats); bookStmt.setDate(5, new java.sql.Date(new Date().getTime()));

bookStmt.executeUpdate();

String updateSeatsQuery = "UPDATE Train SET available\_seats = available\_seats - ? WHERE train\_no = ?";

PreparedStatement updateStmt = conn.prepareStatement(updateSeatsQuery); updateStmt.setInt(1, numberOfSeats); updateStmt.setInt(2, trainNo);

updateStmt.executeUpdate();

JOptionPane.showMessageDialog(this, "Ticket booked successfully!");

} catch (Exception ex) {

JOptionPane.showMessageDialog(this, "Error while booking ticket: " + ex.getMessage());

}

}

public static void main(String[] args) { new BookTicket(12345); // Replace with example train number

}

}

AdminPanel.java:

package main;

import javax.swing.\*;

import admin.AddTrain; import admin.AdminLogin; import admin.ViewBookings;

import java.awt.\*; import java.awt.event.ActionEvent; import java.awt.event.ActionListener;

public class AdminPanel extends JFrame {

public AdminPanel() { // Set up the frame setTitle("Admin Panel"); setSize(400, 400); setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); setLayout(new GridLayout(5, 1, 10, 10));

// Create buttons for various admin actions

JButton loginButton = new JButton("Admin Login");

JButton manageTrainButton = new JButton("Manage Trains");

JButton viewBookingsButton = new JButton("View Bookings");

JButton logoutButton = new JButton("Logout");

JButton exitButton = new JButton("Exit");

// Button actions loginButton.addActionListener(new ActionListener() {

@Override public void actionPerformed(ActionEvent e) {

// Show login screen new AdminLogin(); dispose();

}

});

manageTrainButton.addActionListener(new ActionListener() {

@Override public void actionPerformed(ActionEvent e) {

// Show add train screen new AddTrain(); dispose();

}

});

viewBookingsButton.addActionListener(new ActionListener() {

@Override public void actionPerformed(ActionEvent e) {

// Show view bookings screen new ViewBookings(); dispose();

}

});

logoutButton.addActionListener(new ActionListener() {

@Override public void actionPerformed(ActionEvent e) {

// Show the login screen again new AdminLogin(); dispose();

}

});

exitButton.addActionListener(new ActionListener() {

@Override public void actionPerformed(ActionEvent e) {

int confirm = JOptionPane.showConfirmDialog(null, "Are you sure you want

to exit?", "Exit", JOptionPane.YES\_NO\_OPTION); if (confirm == JOptionPane.YES\_OPTION) {

System.exit(0);

}

}

});

// Add buttons to the panel add(loginButton); add(manageTrainButton); add(viewBookingsButton); add(logoutButton); add(exitButton);

// Make the window visible setVisible(true);

}

public static void main(String[] args) { new AdminPanel();

}

}

AdminLogin.java:

package admin;

import javax.swing.\*; import java.awt.\*;

import java.awt.event.ActionEvent; import java.awt.event.ActionListener;

public class AdminLogin extends JFrame { private JTextField usernameField; private JPasswordField passwordField;

public AdminLogin() { setTitle("Admin Login"); setSize(400, 300); setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); setLayout(new GridLayout(3, 2, 10, 10));

// Username and password fields add(new JLabel("Username:")); usernameField = new JTextField(); add(usernameField);

add(new JLabel("Password:")); passwordField = new JPasswordField(); add(passwordField);

// Login button

JButton loginButton = new JButton("Login"); loginButton.addActionListener(new ActionListener() {

@Override public void actionPerformed(ActionEvent e) {

String username = usernameField.getText();

String password = new String(passwordField.getPassword()); if (username.equals("admin") && password.equals("admin123")) { // Hardcoded credentials

JOptionPane.showMessageDialog(null, "Login successful!"); new AdminDashboard(); dispose();

} else {

JOptionPane.showMessageDialog(null, "Invalid credentials. Please try again.");

}

}

});

add(new JLabel()); add(loginButton);

set Visible(true);

}

public static void main(String[] args) { new AdminLogin();

}

}

AdminDashboard.java:

package admin;

import javax.swing.\*; import java.awt.\*;

public class AdminDashboard extends JFrame {

public AdminDashboard() { setTitle("Admin Dashboard"); setSize(400, 400); setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); setLayout(new GridLayout(4, 1, 10, 10));

// Buttons for navigation

JButton addTrainButton = new JButton("Add Train");

JButton viewBookingsButton = new JButton("View Bookings");

JButton logoutButton = new JButton("Logout");

addTrainButton.addActionListener(e -> new AddTrain()); viewBookingsButton.addActionListener(e -> new ViewBookings()); logoutButton.addActionListener(e -> { new AdminLogin(); dispose();

});

add(addTrainButton); add(viewBookingsButton); add(logoutButton);

setVisible(true);

}

public static void main(String[] args) { new AdminDashboard();

}

}

ViewBookings.java:

package admin;

import javax.swing.\*; import javax.swing.table.DefaultTableModel;

import database.ConnectionManager;

import java.awt.\*; import java.sql.Connection; import java.sql.ResultSet; import java.sql.Statement;

public class ViewBookings extends JFrame {

public ViewBookings() { setTitle("View Bookings"); setSize(800, 600); setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Table for bookings

JTable bookingTable = new JTable(new DefaultTableModel(new

Object[]{"Booking ID", "Train No", "Passenger Name", "Email", "Seats", "Date"}, 0));

JScrollPane scrollPane = new JScrollPane(bookingTable);

add(scrollPane, BorderLayout.CENTER);

loadBookings((DefaultTableModel) bookingTable.getModel());

setVisible(true);

}

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| private void loadBookings(DefaultTableModel model) { try (Connection conn = ConnectionManager.getConnection()) {  String query = "SELECT \* FROM Ticket";  Statement stmt = conn.createStatement();  ResultSet rs = stmt.executeQuery(query);  while (rs.next()) { model.addRow(new Object[]{  rs.getInt("ticket\_id"), rs.getInt("train\_no"), rs.getString("passenger\_name"), rs.getString("email"), rs.getInt("number\_of\_seats"), rs.getDate("booking\_date")  });  }  } catch (Exception ex) {  JOptionPane.showMessageDialog(this, ex.getMessage());  }  }  }      Train.java:  package models;    public class Train { private int trainNo; private String trainName;  33 | "Error | loading | bookings: | " | + |

private String source; private String destination; private int totalSeats; private int availableSeats;

// Constructor

public Train(int trainNo, String trainName, String source, String destination, int

totalSeats, int availableSeats) { this.trainNo = trainNo; this.trainName = trainName; this.source = source; this.destination = destination; this.totalSeats = totalSeats; this.availableSeats = availableSeats;

}

// Getters and Setters public int getTrainNo() { return trainNo;

}

public void setTrainNo(int trainNo) { this.trainNo = trainNo;

}

public String getTrainName() { return trainName;

}

public void setTrainName(String trainName) { this.trainName = trainName;

}

public String getSource() { return source;

}

public void setSource(String source) { this.source = source;

}

public String getDestination() { return destination;

}

public void setDestination(String destination) { this.destination = destination;

}

public int getTotalSeats() { return totalSeats;

}

public void setTotalSeats(int totalSeats) { this.totalSeats = totalSeats;

}

public int getAvailableSeats() { return availableSeats;

}

public void setAvailableSeats(int availableSeats) { this.availableSeats = availableSeats;

}

// Override toString for better representation

@Override public String toString() { return "Train{" +

"trainNo=" + trainNo +

", trainName='" + trainName + '\'' +

", source='" + source + '\'' +

", destination='" + destination + '\'' +

", totalSeats=" + totalSeats +

", availableSeats=" + availableSeats +

'}';

}

}

Ticket.java:

package models;

import java.util.Date;

public class Ticket { private int ticketId; private int trainNo; private String passengerName; private String email; private int numberOfSeats; private Date bookingDate;

// Constructor

public Ticket(int ticketId, int trainNo, String passengerName, String email, int

numberOfSeats, Date bookingDate) {

this.ticketId = ticketId; this.trainNo = trainNo; this.passengerName = passengerName;

this.email = email; this.numberOfSeats = numberOfSeats; this.bookingDate = bookingDate;

}

// Getters and Setters public int getTicketId() { return ticketId;

}

public void setTicketId(int ticketId) { this.ticketId = ticketId;

}

public int getTrainNo() { return trainNo;

}

public void setTrainNo(int trainNo) { this.trainNo = trainNo;

}

public String getPassengerName() { return passengerName;

}

public void setPassengerName(String passengerName) { this.passengerName = passengerName;

}

public String getEmail() { return email;

}

public void setEmail(String email) { this.email = email;

}

public int getNumberOfSeats() { return numberOfSeats;

}

public void setNumberOfSeats(int numberOfSeats) { this.numberOfSeats = numberOfSeats;

}

public Date getBookingDate() { return bookingDate;

}

public void setBookingDate(Date bookingDate) { this.bookingDate = bookingDate;

}

// Override toString for better representation

@Override public String toString() { return "Ticket{" +

"ticketId=" + ticketId +

", trainNo=" + trainNo +

", passengerName='" + passengerName + '\'' +

", email='" + email + '\'' +

", numberOfSeats=" + numberOfSeats +

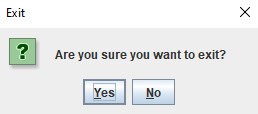
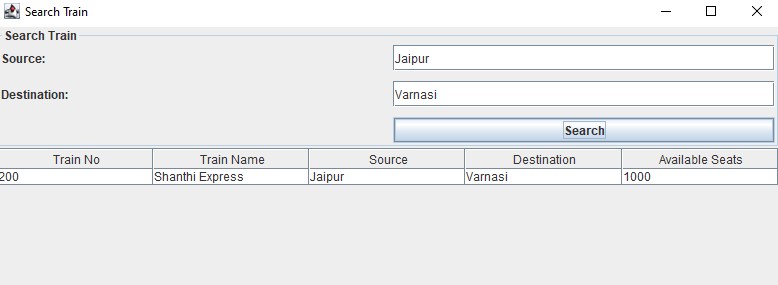
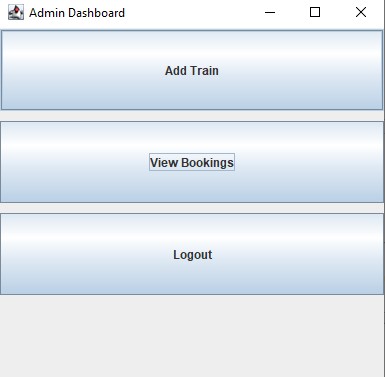
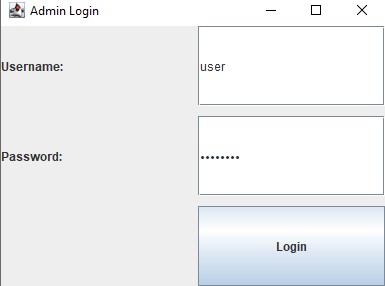
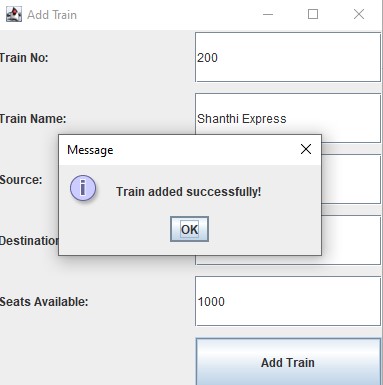
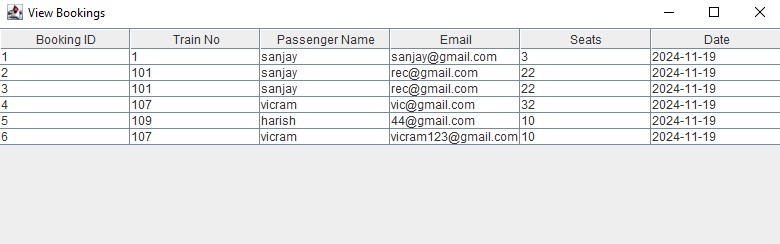
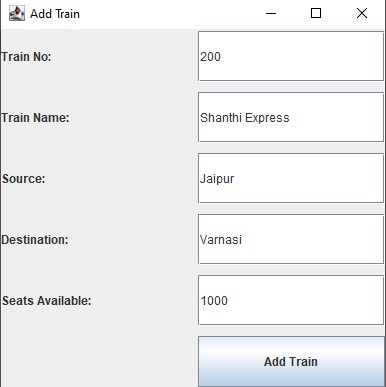
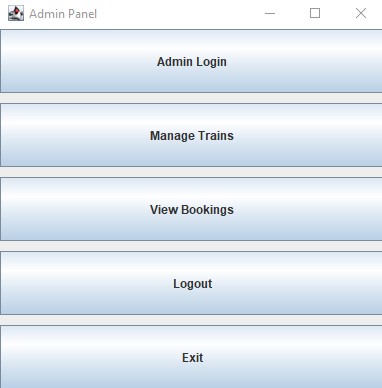
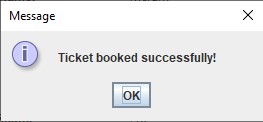
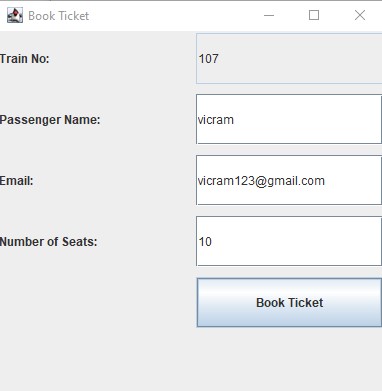
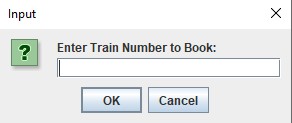
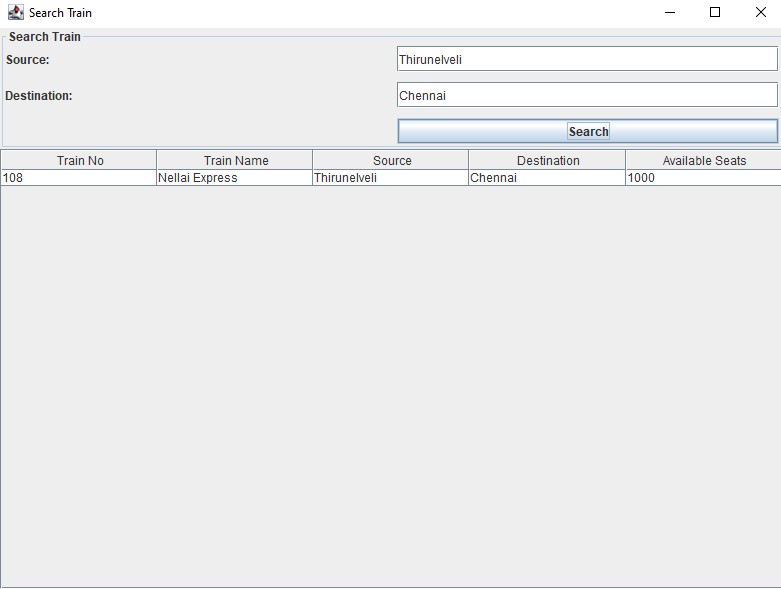
", bookingDate=" + bookingDate +

'}';

}

}

**Screenshots:**



**Conclusion:**

The Train Ticket Booking System project demonstrates a comprehensive approach to designing a desktop application with Java Swing and MySQL. It effectively incorporates user and admin functionalities, ensuring secure and efficient train ticket booking.

Future Enhancements:

1. Integration with online payment systems.
2. Improved UI/UX design using frameworks like JavaFX.
3. Real-time train schedule updates via APIs.

References:

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