BUS TICKET BOOKING SYSTEM USING JDBC WITH MYSQL

A MINI PROJECT REPORT

Submitted by

J JACINTH MANUEL (Reg. No. 9517202309039)

R RAGAVAN (Reg. No. 9517202309092)

D YASHWANTH (Reg. No. 9517202309124)

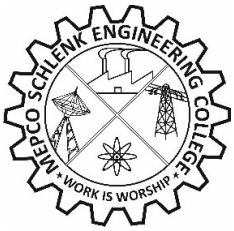
for the Practical Component

of

23AD452: Object Oriented Programming With Java Laboratory

during

IV Semester : 2024 – 2025



DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI

(An Autonomous Institution affiliated to Anna University Chennai)

MAY 2025

MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI

(An Autonomous Institution affiliated to Anna University Chennai)

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

BONAFIDE CERTIFICATE

Certified that this project report titled **Bus Ticket Booking system Using JDBC with MYSQL** is the bonafide work of J Jacinth Manuel(9517202309039), R Ragavan (9517202309092),D Yashwanth (9517202309124) who carried out this work under my guidance for the course **"23AD452: Object Oriented Programming With Java Laboratory"** during the Fourth semester.

SIGNATURE

Mrs. M.Revatheeswari,
Assistant professor,
Department of Artificial Intelligence and Data Science.
Mepco Schlenk Engineering College (Autonomous)
Sivakasi.

SIGNATURE

Dr. J. Angela Jennifa Sujana,
Professor & Head of department,
Department of Artificial Intelligence and Data Science.
Mepco Schlenk Engineering College (Autonomous)
Sivakasi.

Submitted for viva-Voice Examination held at **MEPCO SCHLENK ENGINEERING COLLEGE (Autonomous), SIVAKASI** on _____.

ACKNOWLEDGEMENT

First and foremost, we express our deepest gratitude to the **LORD ALMIGHTY** for His abundant blessings, which have guided us through our past, present, and future endeavors, making this project a successful reality.

We extend our sincere thanks to our management, esteemed Principal, **Dr. S. Arivazhagan, M.E., Ph.D.**, for providing us with a conducive environment, including advanced systems and well-equipped library facilities, which have been instrumental in the completion of this project.

We are profoundly grateful to our **Head of the Artificial Intelligence and Data Science Department**, **Dr. J. Angela Jennifa Sujana, M.Tech., Ph.D.**, for granting us this golden opportunity to work on a project of this significance and for her invaluable guidance and encouragement throughout.

Our heartfelt appreciation goes to our project guides **Mrs. M.Revatheeswari**, **M.Tech.,** Assistant Professor and **Mrs. A.Muthulakshmi, M.Tech.,** Assistant Professor, Department of Artificial Intelligence and Data Science for their unwavering support, insightful suggestions, and expert guidance. Their experience and encouragement have been vital in shaping our project and helping us bring forth our best.

We also extend our sincere gratitude to all our faculty members, lab technicians, and our beloved family and friends, whose timely assistance and moral support played a pivotal role in making this mini-project a success.

ABSTRACT

The Bus Ticket Booking System is a comprehensive management software developed using Java, incorporating Swing for the graphical user interface (GUI) and Java Database Connectivity (JDBC) for backend database interactions. Designed to streamline various bus ticketing operations, this application consolidates functionalities related to passenger registration, ticket bookings, payment processing, route management, and reporting into a user-friendly platform.

The overarching aim of the project is to enhance operational efficiency while minimizing the scope for human error that often arises from manual handling. By bridging the gap between different operational areas into a single system, the Bus Ticket Booking System facilitates smooth transitions between passenger interactions and backend processes, ultimately leading to improved customer satisfaction and streamlined bus operations.

Key components of the application include the Booking Management Panel, which allows the user to manage passenger bookings efficiently, the Passenger Management Panel for handling passenger details, and the Route Management Panel to maintain records of route availability and bus schedules. The GUI features interactive components such as combo boxes, text fields, and tables, enabling real-time data entry and presentation.

Additionally, the project includes payment functionalities, allowing bus operators to track and manage financial transactions linked to bookings, thereby ensuring accurate financial accounts. The system also incorporates a Seat Management Panel, enabling the allocation and management of seats on buses, and a Bus Management Panel, which maintains records of bus details, including bus type, capacity, and maintenance schedules.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The Bus Ticket Booking System is a comprehensive management software developed using Java, incorporating Swing for the graphical user interface (GUI) and Java Database Connectivity (JDBC) for backend database interactions. Designed to streamline various bus ticketing operations, this application consolidates functionalities related to passenger registration, ticket bookings, payment processing, route management, and reporting into a user-friendly platform.

The overarching aim of the project is to enhance operational efficiency while minimizing the scope for human error that often arises from manual handling. By bridging the gap between different operational areas into a single system, the Bus Ticket Booking System facilitates smooth transitions between passenger interactions and backend processes, ultimately leading to improved customer satisfaction and streamlined bus operations.

Key components of the application include the Booking Management Panel, which allows the user to manage passenger bookings efficiently, the Passenger Management Panel for handling passenger details, and the Route Management Panel to maintain records of route availability and bus schedules. The GUI features interactive components such as combo boxes, text fields, and tables, enabling real-time data entry and presentation.

Additionally, the project includes payment functionalities, allowing bus operators to track and manage financial transactions linked to bookings, thereby ensuring accurate financial accounts. The system also incorporates a Seat Management Panel, enabling the allocation and management of seats on buses, and a Bus Management Panel, which maintains records of bus details, including bus type, capacity, and maintenance schedules.

1.1 PROBLEM STATEMENT

The bus transportation industry today faces significant challenges in managing operations efficiently, primarily due to outdated processes and disparate systems that

handle different aspects of bus ticketing and management. Many bus operators still rely on manual methods or unintegrated software tools for ticket bookings, payment processing, and route management. This can lead to inefficiencies, such as overbooked buses, poorly managed passenger data, inaccurate billing, and an overall decrease in customer satisfaction.

Moreover, with the increasing expectations of passengers for faster booking processes, quick access to information, and personalized services, bus staff is often burdened with cumbersome tasks. Existing systems may not provide real-time data updates, which can result in discrepancies in bus availability, route schedules, and passenger information. As a result, bus operations suffer from delays and errors, which can tarnish the passenger experience and ultimately lead to a loss in revenue and reputation.

Hence, there is a pressing need for a comprehensive bus ticket booking system that centralizes all operational functions in one user-friendly platform. This solution should streamline the various processes involved in bus ticketing and management to minimize human error, improve efficiency, and enhance overall passenger satisfaction.

1.2 OBJECTIVES

The primary objectives of the Bus Ticket Booking System project are as follows:

- Centralized Management System: To develop an integrated software application that centralizes all bus ticketing and management functionalities—passenger management, ticket booking, payment processing, and route management—into a single interface. This aims to ensure seamless data flow and easy access across various operational areas.
- User-Friendly Interface: To create an intuitive and user-friendly graphical user interface (GUI) utilizing Java Swing, allowing bus staff to efficiently manage operations without extensive training. The design will prioritize ease of use, ensuring that staff can navigate the application with confidence.
- Real-Time Operations: To implement real-time data processing that allows for immediate updates of bus availability, passenger information, and payment records, thus ensuring accuracy and efficiency at operational touchpoints.

- Error Reduction: To reduce human error through automated data entry and validation procedures, ensuring that incorrect information does not lead to overbookings or complications in passenger management.
- Scalable Features: To design the application with scalability in mind, allowing for future enhancements such as online booking capabilities, mobile ticketing, advanced reporting tools, and integration with third-party services for added functionalities.
- Enhanced Passenger Experience: To improve overall customer satisfaction by streamlining passenger interactions during the booking and travel process, allowing staff to focus on providing individualized service and support.

1.3 SCOPE

The scope of the Bus Ticket Booking System project encompasses the following elements:

- Passenger Management: Enabling the addition, update, and retrieval of passenger information, including personal details and travel histories.
- Route Management: Facilitating management of bus routes, schedules, and fares, including the creation of new routes, modification of existing ones, and assignment of buses to specific routes.
- Booking Management: Allowing for the creation, modification, and cancellation of bookings, ensuring real-time updates of bus availability and passenger records.
- Seat Management: Enabling the management of seat allocations, including the assignment of seats to passengers and the tracking of seat availability.
- Payment Processing: Integrating payment functionalities to handle transactions linked to bookings and ensure accurate financial tracking.
- Bus Management: Management of bus details, including bus type, capacity, and maintenance schedules, to ensure efficient allocation of buses to routes.
- Reporting and Analytics: Features to generate reports on passenger traffic, revenue, route popularity, and bus utilization, assisting management in decision-making.
- Application Development Environment: The application will be developed using Java, with Swing for the GUI and JDBC for database connectivity.

- Database: The system will employ MySQL for data storage, ensuring efficient querying and management of bus ticketing-related information.
- Security: Implementation of security measures to protect passenger data and prevent unauthorized access to the system.
- User Management: Management of user roles and permissions, allowing administrators to control access to different features and functions of the system.

CHAPTER 2
EXISTING AND PROPOSED MODEL

2.1 EXISTING MODEL

The existing model of bus ticket booking and management typically relies on a fragmented system where multiple disparate applications handle different aspects of operations. This can include various standalone software for managing passenger bookings, payment processing, and route management.

- Disparate Systems: Bus operators often use different tools for different functions (e.g., a separate system for bookings, another for managing payments, and yet another for route scheduling). This leads to a lack of integration and continuity in operations.
- Manual Processes: Many tasks still require manual entry or adjustments, increasing the chances for human error (e.g., overbooked buses, incorrect passenger information).
- Inefficiencies: Staff may struggle to manage multiple tools, leading to delays in service. Employees might find themselves switching between systems, which can slow operations and frustrate both staff and passengers.
- Data Inconsistencies: Because data is stored in multiple locations and systems, keeping information synchronized can be a massive challenge. This inconsistency can result in inaccurate reports and hinder effective decision-making.
- Limited Reporting Capabilities: Generating comprehensive reports that give insights into bus operations becomes complicated due to the lack of consolidated data. Management must piece together information from different systems, often resulting in incomplete or inaccurate assessments.
- Staff Training Challenges: New personnel may face a steep learning curve due to the complexity of using multiple systems, which can deter staff performance levels and adaptability.

2.2 PROPOSED MODEL

The Bus Ticket Booking System is envisioned as an integrated bus ticketing and management software solution that unifies all essential functionalities into a single platform. This cohesive approach aims to streamline operations and enhance overall efficiency.

- Unified System: All management functions are integrated within a singular application, allowing for seamless operation across passenger management, ticket bookings, payment processing, and route management.
- Centralized Data Management: The centralized architecture keeps all data synchronized and up-to-date, significantly reducing the risks of errors tied to outdated or incorrect information.
- Real-Time Updates: Real-time data access ensures that staff can instantly retrieve and modify passenger, ticket, and route information, providing current insights into bus operations.
- Streamlined Training and Usage: By minimizing the number of systems staff must learn, the training process becomes simpler and quicker. Staff can focus on delivering quality service rather than managing multiple platforms.
- Future Growth and Scalability: The software is designed with scalability in mind, allowing for the integration of additional features, such as online booking capabilities, mobile ticketing, and advanced reporting tools.
- Enhanced Operational Capabilities: By employing modern technological standards, the Bus Ticket Booking System not only improves current operational efficiencies but also positions bus operators to adapt to future changes and expectations of the industry.
- Improved Customer Service: With streamlined operations and immediate access to information, staff can provide faster and more personalized service to passengers, enhancing overall customer satisfaction.
- Increased Revenue: By reducing errors and improving operational efficiency, the Bus Ticket Booking System can help bus operators increase revenue and reduce costs.

CHAPTER 3

SYSTEM DESIGN

3.1 EVALUATION & SELECTION OF SPECIFICATIONS/FEATURES

The design of the Bus Ticket Booking System stemmed from a comprehensive evaluation of functional and non-functional requirements crucial for effective bus ticketing and management operations. Core to this design is the realization that an intuitive graphical user interface (GUI) is essential for enhancing user experience and minimizing training time for new staff.

Additionally, scalable architecture ensures that the application can handle an expanding number of passengers, bookings, and routes, thereby supporting bus operators of various sizes. The application is structured around core functionalities:

1. Passenger Management: This module allows for the addition, modification, and retrieval of passenger details, ensuring the data entered is validated to maintain accuracy.

2. Route Management: Enables the management of route schedules, fares, and bus assignments, providing real-time information on route availability and bus schedules.

3. Booking Management: This panel facilitates the creation of bookings where passengers can reserve seats on buses based on availability and desired travel dates. A close integration with the passenger and route modules ensures that passenger travel information reflects correctly.

4. Seat Management: The seat management panel allows for the allocation and management of seats on buses, ensuring that passengers can select their preferred seats and that seat availability is accurately reflected.

5. Payment Integration: The payment management panel processes passenger payments linked to their bookings, maintaining accurate financial records and enabling cash flow management.

6. Reporting Features: The system allows for the generation of statistical insights into bus operations, such as total bookings, passenger traffic, revenue generated, and route popularity, which assists in strategic decision-making.

3.2 Design Flow

The design flow of the Bus Ticket Booking System is organized to reflect the sequential processes involved in bus ticketing and management. Each interaction step is modularized, ensuring that user actions are logically routed to the correct functionality. This structure not only promotes clarity in usage but also enhances debugging and future enhancements.

Modules Interaction Flow:

1. Start: The application initializes and displays the login screen.
2. User Login: Users enter their credentials; upon authentication, they access the main menu.
3. Main Menu Navigation: Users select from options for managing passengers, routes, bookings, payments, or generating reports.
4. Passenger Management Module: Users can add, edit, or view passenger information. Validations are performed to ensure data integrity before entering the database.
5. Route Management Module: Users can view available routes, update schedules, and change fares. This module is linked to the booking process, reflecting real-time availability.
6. Booking Management Module: Users can create and manage bookings, ensuring that the booking system updates seat availability accordingly.
7. Seat Management Module: Users can view and manage seat allocations, ensuring that passengers can select their preferred seats.
8. Payment Module: Payment details are processed, ensuring accountability and logging transactions comprehensively.
9. Reporting: Used to generate summaries and financial reports for management review, including passenger traffic, revenue, and route popularity.
10. Exit: The user can logout or close the application; all sessions and data changes are saved.

3.2.1 Entity Relationship Diagram

ERDs help delineate the relationship between various entities in the system.

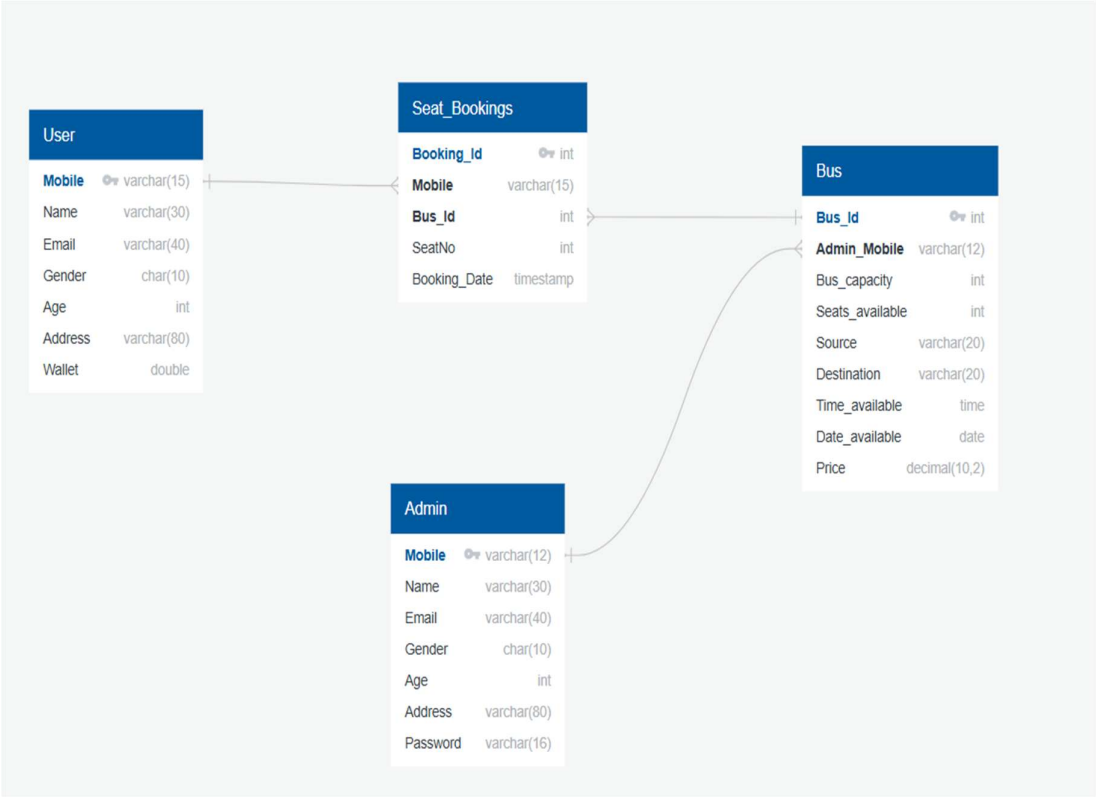


Fig 3.1 ER Diagram

3.2.2 Class Diagram

The Class Diagram outlines the structure of the classes in the system, focusing on the essential attributes and methods.

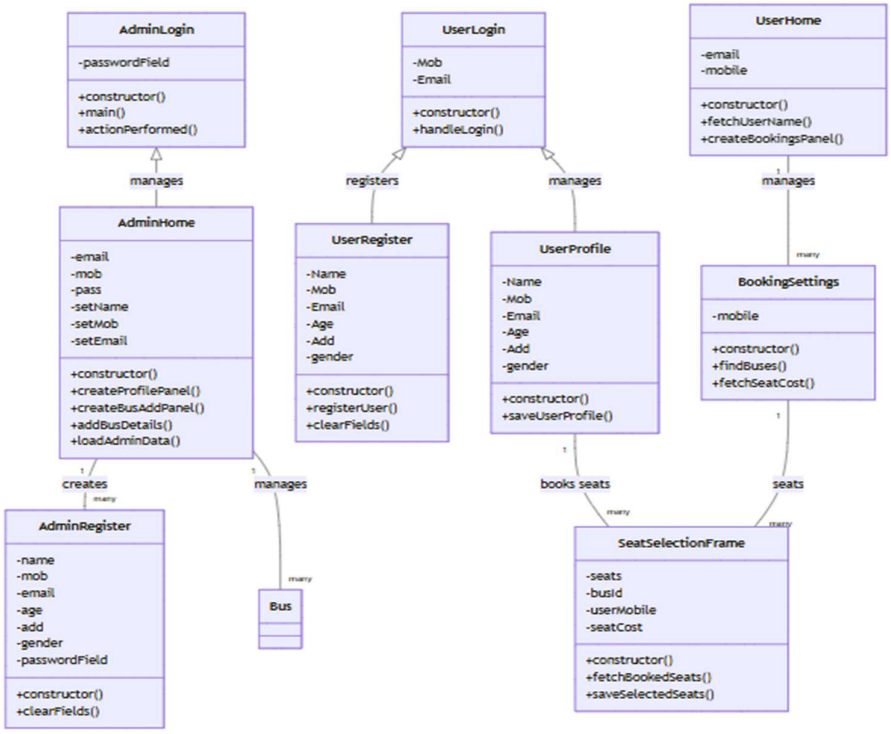


Fig 3.2 Class Diagram

3.2.3 Flowchart

The flow chart showcases the general workflow for processing a bus booking.

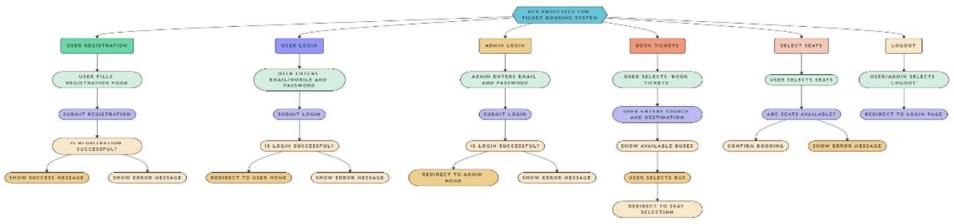


Fig 3.3 Flow Chart

3.2.4 Algorithm

Initialization

1. Start the application.
2. Initialize the connection to the MySQL database.
3. Display the login screen to the user.

User Login

1. Prompt the user to enter their username and password.
2. Validate the credentials:
    - If valid, grant access to the main menu.
    - If invalid, display an error message and prompt for credentials again. Once authenticated, navigate to the main menu.

Main Menu

Display the main menu with the following options:

- Passenger Management
- Route Management
- Booking Management
- Payment Management
- Bus Management
- Reporting
- Exit

Passenger Management

If "Passenger Management" is selected: Display options to add passenger information.

If adding a passenger:

- Prompt for passenger details (name, email, phone).
- Validate input (check email and phone format).
- Insert passenger information into the passenger database.
- Display a success message.

Route Management

If "Route Management" is selected: Display options to add, update, or view route information.

If adding a route:

- Prompt for route details (route number, origin, destination, fare).

- Validate input (ensure route number is unique).
- Insert route information into the route database.
- Display a success message.

Booking Management

If "Booking Management" is selected: Display options to create, update, or cancel bookings.

If creating a booking:

- Prompt for passenger ID, route ID, travel date, and seat selection.
- Validate seat availability for the selected route and travel date.
  If the seat is available:
  1. Create a new booking record in the booking database.
  2. Update the seat status to "Booked".
  3. Display booking confirmation.
     If the seat is not available: Display an error message.

Payment Management

If "Payment Management" is selected: Display options to process or view payments.

If processing a payment:

- Prompt for booking ID and payment amount.
- Validate the payment amount (ensure it is greater than zero).
- Create a new payment record linked to the booking in the payment database.
- Display payment confirmation.

Bus Management

If "Bus Management" is selected: Display options to add or update bus information.

If adding a bus:

- Prompt for bus details (bus number, capacity, route assignment).
- Store bus information in the bus database.
- Display success message.

Reporting

If "Reporting" is selected:

- Allow user to generate reports (e.g., total bookings, passenger traffic, revenue).
- Display generated reports to the user.

Exit Application

If "Exit" is selected:

- Close the database connection.
- End the application.

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 DEVELOPMENT ENVIRONMENT

The implementation of the Bus Ticket Booking System was performed utilizing the following environment settings:

- Java Development Kit (JDK): Version 8 or later should be installed to develop the application effectively.
- Integrated Development Environments (IDE): Tools like Eclipse are used for writing and testing code efficiently.
- Database: A MySQL database was selected for data storage, with the essential tables created to capture all necessary information about passengers, route details, bookings, payments, and buses.

4.2 CODE STRUCTURE

The source code is organized within packages to separate different functionalities and ensure cohesion and maintainability. Classes are classified based on their responsibility:

- BusTicketBookingApp: Acts as the entry point for the application, initializing the GUI and managing tabbed views.
- LoginFrame: Implements authentication features, enabling user access.
- DatabaseConnection: This class is responsible for establishing database connections and executing SQL queries.
- Management Panels: Each panel (passenger, route, booking, payment, bus) features its own class to handle relevant functionalities and UI.

4.3 DATABASE SETUP

The MySQL database schema was constructed using SQL commands to define the necessary tables for effective data management, including constraints to ensure referential integrity. Below is the schema of the bus_ticket_booking database:

4.3.1 SCHEMA

Field	Type	Null	Key	Default	Extra
Name	varchar(30)	YES	PRI	NULL	
Mobile	varchar(15)	NO		NULL	
Email	varchar(40)	YES		NULL	
Gender	char(10)	YES		NULL	
Age	int	YES		NULL	
Address	varchar(80)	YES		NULL	
Wallet	double	YES		0	

Table 4.1 Table User

Field	Type	Null	Key	Default	Extra
Name	varchar(30)	YES		NULL	
Mobile	varchar(12)	YES		NULL	
Email	varchar(40)	YES		NULL	
Gender	char(10)	YES		NULL	
Age	int	YES		NULL	
Address	varchar(80)	YES		NULL	
Password	varchar(16)	YES		NULL	

Table 4.2 Table Admin

Field	Type	Null	Key	Default	Extra
Bus_id	int	NO	PRI	NULL	
Bus_capacity	int	YES		NULL	
Seats_available	int	YES		NULL	
Source	varchar(20)	YES		NULL	
Destination	varchar(20)	YES		NULL	
Time_available	time	YES		NULL	
Price	decimal(10,2)	NO		0.00	

Table 4.3 Table Bus

Field	Type	Null	Key	Default	Extra
Booking_id	int	NO	PRI	NULL	auto_increment
Mobile	varchar(15)	NO	MUL	NULL	
Bus_id	Int	NO		NULL	
SeatNo	int	NO		NULL	
Booking_Date	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_ GENERATED

Table 4.4 Table Seat Booking

CHAPTER 5

RESULTS AND DISCUSSION

Several tests were conducted throughout the development process to ensure that every aspect of the Bus Ticket Booking System functions as intended.

5.1 USER TESTING

User testing played a crucial role in evaluating the usability and functionality of the application. Involving diverse bus operator staff representatives during testing provided valuable insights into real-world application performance. Key observations included simplicity in navigating through the system and capturing passenger information without extraneous steps. Feedback from users indicated that they appreciated features like immediate seat availability checks and booking confirmations.

5.2 PERFORMANCE EVALUATION

Performance testing subjected the application to various scenarios, simulating high user load and multiple concurrent operations. Average transaction times for bookings and payments were benchmarked, ensuring that response times remained below acceptable limits under stress. Results revealed that the application retained speed and efficiency, even with simultaneous inquiries across passenger data and booking processes, and maintained successful operational throughput with minimal downtime.

5.3 TEST CASES

Through rigorous testing, common functionalities were assessed through explicitly defined test cases:

- Adding valid passengers: Ensured that the application accurately records and displays the data.
- Introducing invalid data: (e.g., incorrectly formatted email addresses or invalid payment information) effectively triggered error messages, demonstrating robust input validation mechanisms.
- Booking operations: Were tested for accuracy, verifying if the system accurately updated seat statuses and reflected booking histories appropriately.

5.2 RESULTS

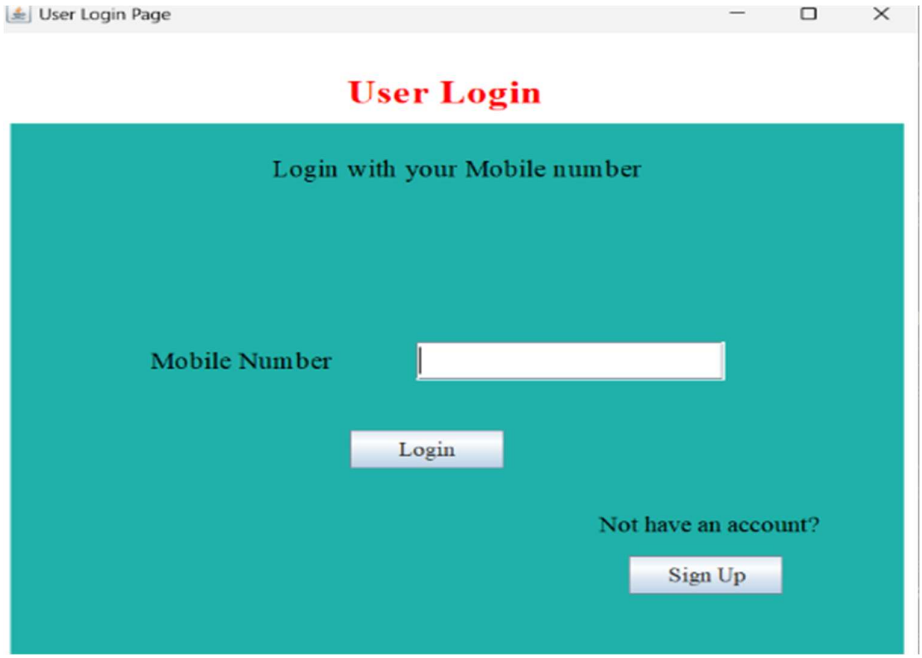


FIG 5.1 USER LOGIN

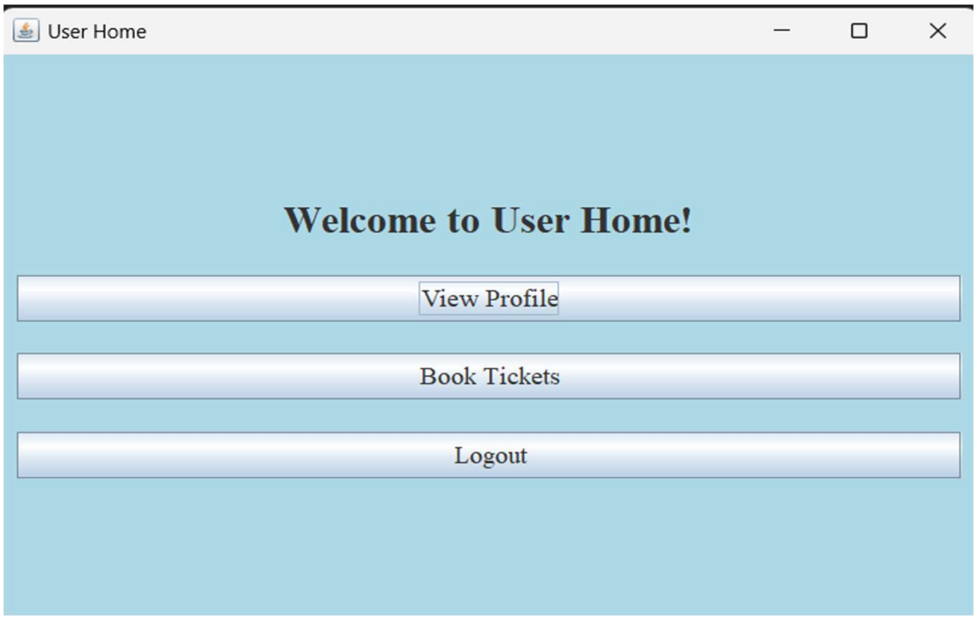


FIG 5.2 USER HOME PAGE

FIG 5.3 SEAT SELECTION

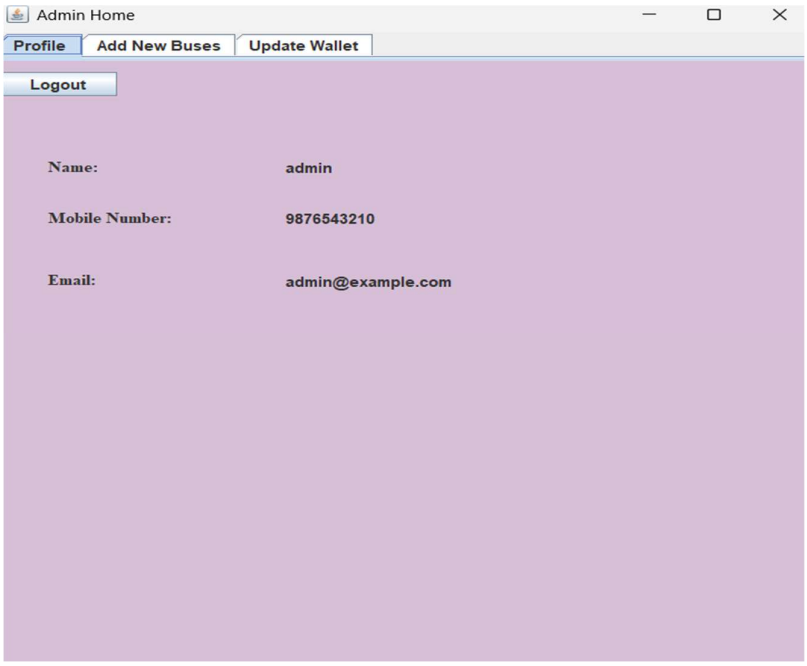


FIG 5.4 ADMIN HOME PAGE

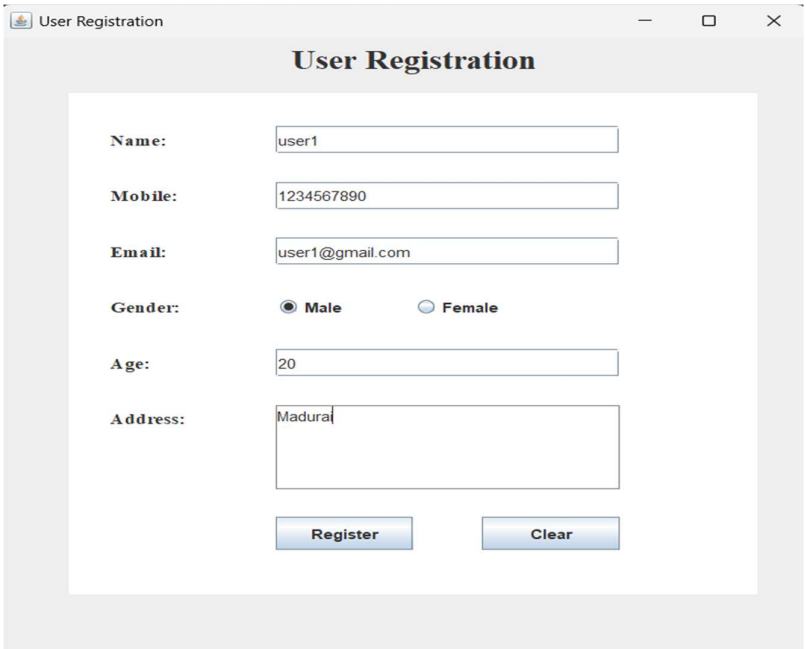


FIG 5.5 USER REGISTRATION

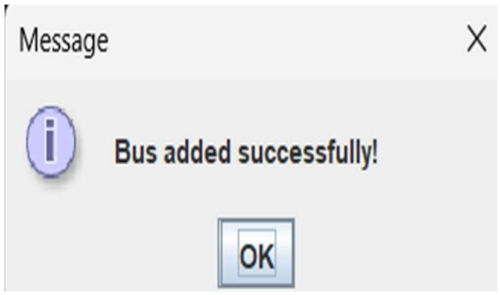


FIG 5.6 BUS ALLOCATION

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT(S)

In summary, the Bus Ticket Booking System provides a highly effective solution for managing bus ticketing and operations, integrating essential functionalities into a single platform. The successful implementation of the application highlights the advantages of incorporating technology into everyday business processes, improving both efficiency and passenger experience.

CONCLUSION

The project's initial objectives have been met, with features that offer real-time management of passenger data and bus services, significantly reducing manual tasks and human error. User feedback reflects a positive reception, particularly noting the simplicity and effectiveness of the application in streamlining bus ticketing and operations.

FUTURE ENHANCEMENTS

The Bus Ticket Booking System is built to evolve, anticipating future requirements and technological advancements:

- Online Booking Capabilities: Allowing passengers to initiate bookings remotely via a web interface or mobile application.
- Enhanced Analytics: Implementing a more robust reporting tool that could provide insights into trends related to passenger preferences, route popularity, and historical bookings.
- Integration with Third-Party Services: Various third-party tools could be considered for aspects such as payment gateways, marketing automation, or passenger engagement solutions.
- User Role Management: Incorporating security features that would permit differentiated access levels, ensuring sensitive data remains secure while still being easily accessible to authorized personnel.
- Mobile Ticketing: Enabling passengers to access and manage their tickets using their mobile devices, reducing the need for physical tickets and improving the overall travel experience.

APPENDIX – A

SOURCE CODE

```
import java.sql.*;

import java.awt.Font;

import javax.swing.*;

import java.awt.Color;


public class AdminHome extends JFrame {

    private static final long serialVersionUID = 1L;

    public static String email;

    public static String mob;

    public static String pass;

    private JLabel setName, setMob, setEmail;

    AdminHome(String email, String mob, String pass) {

        AdminHome.email = email;

        AdminHome.mob = mob;

        AdminHome.pass = pass;

        setTitle("Admin Home");

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        setSize(600, 500);

        JTabbedPane tabPanel = new JTabbedPane();

        tabPanel.setBackground(Color.WHITE);
```

```
    JPanel page1 = createProfilePanel();
    tabPanel.addTab("Profile", page1);
    JPanel page2 = createBusAddPanel();
    tabPanel.addTab("Add New Buses", page2);
    getContentPane().add(tabPanel);
    setVisible(true);
    loadAdminData();
}
private JPanel createProfilePanel() {
    JPanel panel = new JPanel();
    panel.setBackground(new Color(216, 191, 216));
    panel.setLayout(null);
    JLabel lblName = new JLabel("Name:");
    lblName.setFont(new Font("Times New Roman", Font.BOLD, 13));
    lblName.setBounds(35, 82, 131, 21);
    panel.add(lblName);
    JLabel lblMobileNumber = new JLabel("Mobile Number:");
    lblMobileNumber.setFont(new Font("Times New Roman", Font.BOLD, 13));
    lblMobileNumber.setBounds(35, 126, 131, 21);
    panel.add(lblMobileNumber);
    JLabel lblEmail = new JLabel("Email:");
    lblEmail.setFont(new Font("Times New Roman", Font.BOLD, 13));
    lblEmail.setBounds(35, 180, 131, 21);
```

```
        panel.add(lblEmail);
        setName = new JLabel();
        setName.setBounds(205, 82, 131, 21);
        panel.add(setName);
        setMob = new JLabel();
        setMob.setBounds(205, 126, 131, 21);
        panel.add(setMob);
        setEmail = new JLabel();
        setEmail.setBounds(205, 180, 131, 21);
        panel.add(setEmail);
        JButton btnLogout = new JButton("Logout");
        btnLogout.addActionListener(e -> {
            AdminLogin ul = new AdminLogin();
            setVisible(false);
            ul.setVisible(true);
        });
        btnLogout.setBounds(0, 10, 85, 21);
        panel.add(btnLogout);
        return panel;
    }
    private JPanel createBusAddPanel() {
        JPanel panel = new JPanel();
        panel.setBackground(new Color(216, 191, 216));
```

```
panel.setLayout(null);
JLabel lblBusId = new JLabel("Bus ID:");
lblBusId.setFont(new Font("Times New Roman", Font.BOLD, 13));
lblBusId.setBounds(30, 50, 100, 25);
panel.add(lblBusId);
JTextField busIdField = new JTextField();
busIdField.setBounds(140, 50, 150, 25);
panel.add(busIdField);
JLabel lblSource = new JLabel("Source:");
lblSource.setFont(new Font("Times New Roman", Font.BOLD, 13));
lblSource.setBounds(30, 100, 100, 25);
panel.add(lblSource);
JTextField sourceField = new JTextField();
sourceField.setBounds(140, 100, 150, 25);
panel.add(sourceField);
JLabel lblDestination = new JLabel("Destination:");
lblDestination.setFont(new Font("Times New Roman", Font.BOLD, 13));
lblDestination.setBounds(30, 150, 100, 25);
panel.add(lblDestination);
JTextField destinationField = new JTextField();
destinationField.setBounds(140, 150, 150, 25);
panel.add(destinationField);
JLabel lblPrice = new JLabel("Price:");
```

```
lblPrice.setFont(new Font("Times New Roman", Font.BOLD, 13));
lblPrice.setBounds(30, 200, 100, 25);
panel.add(lblPrice);
JTextField priceField = new JTextField();
priceField.setBounds(140, 200, 150, 25);
panel.add(priceField);
JLabel lblSeats = new JLabel("Seats Available:");
lblSeats.setFont(new Font("Times New Roman", Font.BOLD, 13));
lblSeats.setBounds(30, 250, 150, 25);
panel.add(lblSeats);
JTextField seatsField = new JTextField();
seatsField.setBounds(180, 250, 150, 25);
panel.add(seatsField);
JButton btnAddBus = new JButton("Add Bus");
btnAddBus.setBounds(100, 300, 150, 25);
btnAddBus.addActionListener(e -> {
    String busId = busIdField.getText().trim();
    String source = sourceField.getText().trim();
    String destination = destinationField.getText().trim();
    String price = priceField.getText().trim();
    String seats = seatsField.getText().trim();
```

```
        if (busId.isEmpty() || source.isEmpty() || destination.isEmpty() || price.isEmpty() ||
seats.isEmpty()) {
            JOptionPane.showMessageDialog(panel, "All fields are required.", "Error",
JOptionPane.ERROR_MESSAGE);
            return;
        }
        try {
            double priceValue = Double.parseDouble(price);
            int seatsValue = Integer.parseInt(seats);
            if (addBusDetails(busId, source, destination, priceValue, seatsValue)) {
                JOptionPane.showMessageDialog(panel, "Bus added successfully!");
            } else {
                JOptionPane.showMessageDialog(panel, "Failed to add bus. Check the
details.", "Error", JOptionPane.ERROR_MESSAGE);
            }
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(panel, "Invalid price or seats. Please enter
valid numbers.", "Error", JOptionPane.ERROR_MESSAGE);
        }
    });
    panel.add(btnAddBus);
    return panel;
  }
```

```
    private boolean addBusDetails(String busId, String source, String destination, double
price, int seats) {
        try (Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/bus_ticket", "root",
"Jac@231005");
             PreparedStatement ps = con.prepareStatement("INSERT INTO bus (Bus_Id,
Source, Destination, Price, Seats_Available) VALUES (?, ?, ?, ?, ?)")) {
            ps.setString(1, busId);
            ps.setString(2, source);
            ps.setString(3, destination);
            ps.setDouble(4, price);
            ps.setInt(5, seats);
            return ps.executeUpdate() > 0;
        } catch (SQLException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(this, "Database error: " + e.getMessage(),
"Error", JOptionPane.ERROR_MESSAGE);
        }
        return false;
    }
    private void loadAdminData() {
        try (Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/bus_ticket", "root",
"Jac@231005");
```

```
        PreparedStatement ps = con.prepareStatement("SELECT * FROM admin
WHERE email = ? OR mobile = ?")) {
            ps.setString(1, email);
            ps.setString(2, mob);
            try (ResultSet rs = ps.executeQuery()) {
                if (rs.next()) {
                    setName.setText(rs.getString("name"));
                    setMob.setText(rs.getString("mobile"));
                    setEmail.setText(rs.getString("email"));
                } else {
                    setName.setText("Not found");
                    setMob.setText("Not found");
                    setEmail.setText("Not found");
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(this, "Error loading admin data: " +
e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
    public static void main(String[] args) {
        AdminHome adminh = new AdminHome("admin@example.com", "9876543210",
"admin1234");
```

```
        adminh.setVisible(true);
    }
}
import javax.swing.*;
import java.awt.Font;
import java.awt.Color;
import java.sql.*;
public class UserLogin extends JFrame {
    private static final long serialVersionUID = 1L;
    private JTextField txtMob;
    private JTextField txtEmail;
    public UserLogin() {
        // Setting JFrame properties
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 613, 532);
        setTitle("User Login Page");
        getContentPane().setLayout(null);
        JLabel head = new JLabel("User Login", JLabel.CENTER);
        head.setFont(new Font("Times New Roman", Font.BOLD, 26));
        head.setForeground(Color.RED);
        head.setBounds(216, 10, 150, 30);
        getContentPane().add(head);
        getContentPane().setBackground(new Color(255, 255, 255));
```

```
JPanel panel = new JPanel();
panel.setBackground(new Color(32, 178, 170));
panel.setBounds(10, 59, 579, 426);
getContentPane().add(panel);
panel.setLayout(null);
JLabel lblEmail = new JLabel("Email");
lblEmail.setBounds(61, 52, 135, 30);
panel.add(lblEmail);
lblEmail.setHorizontalAlignment(SwingConstants.CENTER);
lblEmail.setFont(new Font("Times New Roman", Font.PLAIN, 18));
txtEmail = new JTextField(20);
txtEmail.setBounds(263, 53, 200, 30);
panel.add(txtEmail);
txtEmail.setFont(new Font("Times New Roman", Font.PLAIN, 18));
JLabel l1 = new JLabel("OR");
l1.setBounds(188, 113, 150, 30);
panel.add(l1);
l1.setHorizontalAlignment(SwingConstants.CENTER);
l1.setFont(new Font("Times New Roman", Font.PLAIN, 19));
JLabel l2 = new JLabel("Mobile Number");
l2.setBounds(83, 168, 135, 30);
panel.add(l2);
l2.setHorizontalAlignment(SwingConstants.CENTER);
```

```
l2.setFont(new Font("Times New Roman", Font.PLAIN, 18));
txtMob = new JTextField(20);
txtMob.setFont(new Font("Times New Roman", Font.PLAIN, 18));
txtMob.setBounds(263, 168, 200, 30);
panel.add(txtMob);
JButton btnLogin = new JButton("Login");
btnLogin.setBounds(220, 236, 100, 30);
panel.add(btnLogin);
btnLogin.setFont(new Font("Times New Roman", Font.PLAIN, 16));
JLabel l3 = new JLabel("Not have an account?");
l3.setBounds(381, 293, 155, 30);
panel.add(l3);
l3.setFont(new Font("Times New Roman", Font.PLAIN, 17));
JButton btnSignUp = new JButton("Sign Up");
btnSignUp.setBounds(401, 333, 100, 30);
panel.add(btnSignUp);
btnSignUp.setFont(new Font("Times New Roman", Font.PLAIN, 16));
// Action Listener for Sign Up button
btnSignUp.addActionListener(e -> {
    UserRegister userRegister = new UserRegister();
    userRegister.setVisible(true);
    setVisible(false);
});
```

```
        // Action Listener for Login button
        btnLogin.addActionListener(e -> handleLogin());
    }
    private void handleLogin() {
        String email = txtEmail.getText().trim();
        String mobile = txtMob.getText().trim();
        // Validate inputs
        if (email.isEmpty() && mobile.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Please enter either your Email or Mobile Number.", "Input Error", JOptionPane.ERROR_MESSAGE);
            return;
        }
        try {
            // Ensure the driver is loaded
            Class.forName("com.mysql.cj.jdbc.Driver");
            // Establish the database connection
            Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/bus_ticket", "root", "Jac@231005");
            String sql = "SELECT * FROM user WHERE email = ? OR mobile = ?";
            PreparedStatement ps = con.prepareStatement(sql);
            ps.setString(1, email);
            ps.setString(2, mobile);
```

```
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                // Login successful
                JOptionPane.showMessageDialog(this, "Login successful!", "Success",
JOptionPane.INFORMATION_MESSAGE);
                // Navigate to UserHome1
                UserHome1 userHome = new UserHome1(email, mobile);
                userHome.setVisible(true);
                setVisible(false);
            } else {
                JOptionPane.showMessageDialog(this, "Invalid login credentials.", "Login
Failed", JOptionPane.ERROR_MESSAGE);
            }
            con.close();
        } catch (ClassNotFoundException cnfe) {
            JOptionPane.showMessageDialog(this, "MySQL JDBC Driver not found.",
"Driver Error", JOptionPane.ERROR_MESSAGE);
            cnfe.printStackTrace();
        } catch (SQLException sqle) {
            JOptionPane.showMessageDialog(this, "Database error: " + sqle.getMessage(),
"Database Error", JOptionPane.ERROR_MESSAGE);
            sqle.printStackTrace();
        } catch (Exception ex) {
```

```
            JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
            ex.printStackTrace();
        }
    }
    public static void main(String[] args) {
        // Run the UserLogin application
        SwingUtilities.invokeLater(() -> {
            UserLogin userLogin = new UserLogin();
            userLogin.setVisible(true);
        });
    }
}
import javax.swing.*;
import java.awt.*;
import java.sql.*;
public class BookingSettings extends JFrame {
    private static final long serialVersionUID = 1L;
    public BookingSettings(String mobile) {
        // Set JFrame properties
        setTitle("Book Tickets");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(600, 400);
```

```
setLocationRelativeTo(null);

// Create panel for booking functionality

JPanel panel = new JPanel();

panel.setLayout(null);

panel.setBackground(new Color(240, 230, 140));

JLabel lblSource = new JLabel("Source:");

lblSource.setBounds(30, 20, 100, 25);

panel.add(lblSource);

JTextField sourceField = new JTextField();

sourceField.setBounds(140, 20, 150, 25);

panel.add(sourceField);

JLabel lblDestination = new JLabel("Destination:");

lblDestination.setBounds(30, 60, 100, 25);

panel.add(lblDestination);

JTextField destinationField = new JTextField();

destinationField.setBounds(140, 60, 150, 25);

panel.add(destinationField);

JLabel lblBus = new JLabel("Available Buses:");

lblBus.setBounds(30, 100, 100, 25);

panel.add(lblBus);

JComboBox<String> busComboBox = new JComboBox<>();

busComboBox.setBounds(140, 100, 300, 25);

panel.add(busComboBox);
```

```
        JButton btnSearch = new JButton("Search");
        btnSearch.setBounds(320, 40, 100, 25);
        btnSearch.addActionListener(e -> {
            String source = sourceField.getText().trim();
            String destination = destinationField.getText().trim();
            if (source.isEmpty() || destination.isEmpty()) {
                JOptionPane.showMessageDialog(panel, "Please enter both source and
destination.", "Error", JOptionPane.ERROR_MESSAGE);
                return;
            }
            String buses = findBuses(source, destination);
            if (buses == null || buses.isEmpty()) {
                JOptionPane.showMessageDialog(panel, "No buses available for the selected
route.", "Info", JOptionPane.INFORMATION_MESSAGE);
                return;
            }
            busComboBox.removeAllItems();
            for (String bus : buses.split("\n")) {
                busComboBox.addItem(bus);
            }
        });
        panel.add(btnSearch);
```

```
        JButton btnBookTickets = new JButton("Book Tickets");
        btnBookTickets.setBounds(200, 140, 150, 30);
        btnBookTickets.setEnabled(false); // Initially disabled
        btnBookTickets.addActionListener(e -> {
            String selectedBus = (String) busComboBox.getSelectedItem();
            if (selectedBus != null) {
                try {
                    String[] busDetails = selectedBus.split(", ");
                    String busId = busDetails[0]; // Assuming Bus ID is the first value
                    double seatCost = fetchSeatCost(busId); // Fetch seat cost from database
                    JFrame seatFrame = new JFrame("Seat Selection");
                    seatFrame.setSize(600, 600);
                    seatFrame.add(new SeatSelectionFrame(busId, mobile, seatCost));
                    seatFrame.setVisible(true);
                } catch (Exception ex) {
                    JOptionPane.showMessageDialog(panel, "Error processing seat selection: " +
ex.getMessage(),
                            "Error", JOptionPane.ERROR_MESSAGE);
                    ex.printStackTrace();
                }
            } else {
                JOptionPane.showMessageDialog(panel, "Please select a bus first.", "Error",
JOptionPane.ERROR_MESSAGE);
```

```
            }
        });
        panel.add(btnBookTickets);
        // Enable the "Book Tickets" button when a bus is selected
        busComboBox.addActionListener(e -> {
            btnBookTickets.setEnabled(busComboBox.getSelectedItem() != null);
        });
        add(panel);
    }
    private String findBuses(String source, String destination) {
        StringBuilder result = new StringBuilder();
        try (Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/bus_ticket", "root",
"Jac@231005");
             PreparedStatement ps = con.prepareStatement("SELECT * FROM bus WHERE
Source = ? AND Destination = ? AND Seats_Available > 0")) {
            ps.setString(1, source);
            ps.setString(2, destination);
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                result.append(rs.getInt("Bus_Id")).append(", ");
                result.append(rs.getString("Source")).append(", ");
                result.append(rs.getString("Destination")).append(", ");
                result.append(rs.getString("Time_Available")).append(", ");
```

```
                result.append(rs.getDate("Date_Available")).append("\n");
            }
        } catch (SQLException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(null, "Error fetching buses: " + e.getMessage(),
"Error", JOptionPane.ERROR_MESSAGE);
        }
        return result.toString().trim();
    }
    private double fetchSeatCost(String busId) {
        try (Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/bus_ticket", "root",
"Jac@231005");
             PreparedStatement ps = con.prepareStatement("SELECT Price FROM bus
WHERE Bus_Id = ?")) {
            ps.setString(1, busId);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                return rs.getDouble("Price");
            } else {
                throw new SQLException("Price not found for Bus ID: " + busId);
            }
        } catch (SQLException e) {
            e.printStackTrace();
```

```
            JOptionPane.showMessageDialog(this, "Error fetching seat cost: " +
e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
            return 0.0;
        }
    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            BookingSettings bookingSettings = new BookingSettings("1234567890");
            bookingSettings.setVisible(true);
        });
    }
}
```

REFERENCES

- GitHub Topics: Bus Ticket Booking System Java
- JavaTpoint - Bus Ticket Booking System Implementation
- Lucidchart - Create System Diagrams
- W3Schools SQL Tutorial
- Database Design Best Practices by Vertabelo