



RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

CS23333 OBJECT ORIENTED PROGRAMMING
USING JAVA LAB
BUS ROUTE MANAGEMENT SYSTEM
A MINI PROJECT REPORT

Submitted by

Surya Prajin S 231501179

Vasanth Kumar P 231501179

Vishnu Selvam A 231501186

In partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING IN
ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS) THANDALAM
CHENNAI-602105

2024 - 2025

BONAFIDE CERTIFICATE

Certified that this project report “**BUS ROUTE MANAGEMENT SYSTEM**” is the Bonafide work of “**Vasantha kumar(231501179), Surya Prajin (231501167), Vishnu selvam(231501186)**” who carried out the project work under my supervision.

Submitted for the Practical Examination held on _____

SIGNATURE

Mr.Devandra Rao,
Assistant Professor,
AIML,
Rajalakshmi Engineering College,
(Autonomous)
Thandalam, Chennai – 602 105

**INTERNAL EXAMINER
EXAMINER**

EXTERNAL

ABSTRACT

This paper presents the design and implementation of a Bus Route Management System, a web-based application that automates various bus operations. The system leverages a MySQL database to store and manage critical information, including user details, booking history, and the availability of routes. The user-friendly interface, built using JAVA, enables efficient user logins, Route selection, Booking of tickets, and updating of the bookings. By automating routine tasks and providing valuable insights, the system aims to streamline bus operations, improve user satisfaction, and enhance overall business efficiency.

Bus Route management system is a simple console application Mysql database for backend and JAVA for the frontend. The User can perform basic Bus Route management operations like Creating a account, Selecting various routes, Removing or Updating Booking Details.

Each User in the System has a unique Email id. The user can select the routes based on the given set of available routes provided by the admin and select the tickets based on the availability.

TABLE OF CONTENTS

1. INTRODUCTION	
1. INTRODUCTION	1
2. OBJECTIVES	2
3. MODULES	
2. SURVEY OF TECHNOLOGIES	
1. SOFTWARE DESCRIPTION	3
2. LANGUAGES	
2.2.1 JAVA	
2.2.2 MYSQL	
3. 3. REQUIREMENTS AND ANALYSIS	
3.1 REQUIREMENT SPECIFICATION	4
3.2 HARDWARE AND SOFTWARE REQUIREMENTS	
4. PROGRAM CODE	5
5. RESULT	45
6. TESTING	50
7. CONCLUSION	51
8. RESEARCH AND REFERENCE	52

CHAPTER 1

1.1 INTRODUCTION

The Bus Route and Booking Management System is an innovative platform designed to provide passengers with a seamless experience in planning and booking bus journeys. With the growing

reliance on public transportation, this system streamlines the process of searching for routes, reserving seats, and tracking trip details, all within a user-friendly interface.

This project focuses on enhancing the convenience and accessibility of bus services for passengers, allowing them to efficiently manage their travel plans through the following features:

- **User Registration and Login:**

Passengers can register securely and log in to access personalized booking features, including saving travel preferences and viewing booking history.

- **Dynamic Route Search:**

Users can explore available bus routes and schedules, filtering results by departure time, destination, and fare.

- **Interactive Seat Booking:**

Passengers can view seat layouts in real-time, select preferred seats, and add them to their booking cart. The system provides real-time seat availability to avoid conflicts.

- **Customizable Cart Management:**

Users can add multiple trips to their cart, adjust seat quantities, and update booking details before finalizing their purchase.

- **Booking History and Notifications:**

Passengers can track their trip details, including route, timing, and seat number, in a dedicated booking history section. Automatic email notifications provide updates on:

- o Successful account registration.
- o Booking confirmations.
- o Trip schedule changes or delays.
- o Trip Status Updates

By digitizing the traditional bus booking process, the Bus Route and Booking Management System offers unparalleled convenience, saving time and effort for travelers. This system

emphasizes reliability, real-time updates, and ease of use, making it an essential tool for modernizing public transportation services.

1.2 OBJECTIVE

The main objective of the Bus Route Management System is to manage the details of Users, Routes, Tickets. It manages all the information about Users, Routes, Tickets, Bookings Effectively using Mysql Database. The project is totally built at the user end and thus the user can get access by Signing up and login to the website.

1.3 MODULES

- Sign Up
- Login
- Route Details
- Booking Details
- Updating Booking Details

CHAPTER 2

2.1 SOFTWARE DESCRIPTION

Visual studio Code

Visual Studio Code combines the simplicity of a source code editor with powerful developer tooling, like IntelliSense code completion and debugging. First and foremost, it is an editor that gets out of your way. The delightfully frictionless edit-build-debug cycle means less time fiddling with your environment, and more time executing on your ideas.

2.2 LANGUAGES

2.2.1 JAVA

Java is a set of computer software and specifications that provides a software platform for developing application software and deploying it in a cross-platform computing environment. Java is used in a wide variety of computing platforms from embedded devices and mobile phones to enterprise servers and supercomputers. Java applets, which are less common than standalone Java applications, were commonly run in secure, sandboxed environments to provide many features of native applications through being embedded in HTML pages.

2.2.2 MySQL

Many of the world's largest and fastest-growing organizations including Facebook, Google, Adobe, Alcatel Lucent and Zappos rely on MySQL to save time and money powering their high-volume Web sites, business-critical systems and packaged software. Since then, the performance & scalability, reliability, and ease of use of the world's most popular open source database, characteristics that made MySQL the #1 choice for web applications, have relentlessly been improved.

CHAPTER 3

3.1 REQUIREMENTS SPECIFICATION

User Requirements

The system requirement in Bus route management focuses on the booking of tickets based on the available routes by the users.

System Requirements

There should be a database backup of the library management system. Operating system should be WindowsXP or a higher version of windows.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

Software Requirements

- Operating System Windows 10
- Front End JAVA
- Back End JAVA, MySQL

Hardware Requirements

- Desktop PC or a Laptop
- Operating System – Windows 10
- Intel® Core™ i3-6006U CPU @ 2.00GHz
- 4.00 GB RAM
- 64-bit operating system, x64 based processor
- 1024 x 768 monitor resolution
- Keyboard and Mouse

CHAPTER 4

PROGRAM CODE:

SIGNUP:

```
import java.awt.*;
```



```

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

public class signup {

    // Database connection details

    private static final String DB_URL = "jdbc:mysql://localhost:3306/bus"; // Replace 'bus' with
your database name

    private static final String DB_USER = "root"; // Replace with your MySQL username

    private static final String DB_PASSWORD = "moni2626"; // Replace with your MySQL
password

    public static void main(String[] args) {

        // Set the look and feel of the UI for better consistency across platforms

        try {

            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

        } catch (Exception e) {

            e.printStackTrace();

        }

        // Create and setup the signup frame

        JFrame frame = new JFrame("Signup Page");

        frame.setSize(450, 400);

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        frame.setLocationRelativeTo(null); // Center the frame on the screen

        frame.setResizable(false);

        // Create a JPanel with GridBagLayout for better control of component placement

        JPanel panel = new JPanel(new GridBagLayout());

        GridBagConstraints gbc = new GridBagConstraints();

        panel.setBorder(BorderFactory.createEmptyBorder(30, 30, 30, 30));

        // Add title label

```

```
JLabel titleLabel = new JLabel("Create Your Account");
titleLabel.setFont(new Font("Segoe UI", Font.BOLD, 24));
titleLabel.setForeground(new Color(0, 122, 204)); // A vibrant color for title
gbc.gridx = 0;
gbc.gridy = 0;
gbc.gridwidth = 2;
gbc.insets = new Insets(0, 0, 20, 0); // Spacing after the title
panel.add(titleLabel, gbc);

// Add Username label and text field
JLabel usernameLabel = new JLabel("Username:");
usernameLabel.setFont(new Font("Arial", Font.PLAIN, 14));
gbc.gridx = 0;
gbc.gridy = 1;
gbc.gridwidth = 1;
gbc.insets = new Insets(0, 0, 10, 0);
panel.add(usernameLabel, gbc);
JTextField usernameField = new JTextField(20);
usernameField.setFont(new Font("Arial", Font.PLAIN, 14));
gbc.gridx = 1;
gbc.gridy = 1;
panel.add(usernameField, gbc);

// Add Email label and text field
JLabel emailLabel = new JLabel("Email:");
emailLabel.setFont(new Font("Arial", Font.PLAIN, 14));
gbc.gridx = 0;
gbc.gridy = 2;
panel.add(emailLabel, gbc);
JTextField emailField = new JTextField(20);
emailField.setFont(new Font("Arial", Font.PLAIN, 14));
gbc.gridx = 1;
```

```

gbc.gridy = 2;
panel.add(emailField, gbc);
// Add Password label and field
JLabel passwordLabel = new JLabel("Password:");
passwordLabel.setFont(new Font("Arial", Font.PLAIN, 14));
gbc.gridx = 0;
gbc.gridy = 3;
panel.add(passwordLabel, gbc);
JPasswordField passwordField = new JPasswordField(20);
passwordField.setFont(new Font("Arial", Font.PLAIN, 14));
gbc.gridx = 1;
gbc.gridy = 3;
panel.add(passwordField, gbc);
// Add Sign Up button
JButton signUpButton = new JButton("Sign Up");
signUpButton.setFont(new Font("Arial", Font.BOLD, 16));
signUpButton.setBackground(new Color(0, 122, 204));
signUpButton.setForeground(Color.WHITE);
signUpButton.setBorderPainted(false);
signUpButton.setFocusPainted(false);
signUpButton.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
// Add a hover effect on the button
signUpButton.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseEntered(java.awt.event.MouseEvent evt) {
        signUpButton.setBackground(new Color(0, 150, 255)); // Lighter on hover
    }
    public void mouseExited(java.awt.event.MouseEvent evt) {
        signUpButton.setBackground(new Color(0, 122, 204)); // Default color
    }
});

```

```

gbc.gridx = 0;
gbc.gridy = 4;
gbc.gridwidth = 2;
gbc.insets = new Insets(20, 0, 10, 0);
panel.add(signupButton, gbc);
// Add Login button (Redirect to login page)
JButton loginButton = new JButton("Login Page");
loginButton.setFont(new Font("Arial", Font.PLAIN, 14));
loginButton.setBackground(new Color(255, 255, 255));
loginButton.setForeground(new Color(0, 122, 204));
loginButton.setBorder(BorderFactory.createLineBorder(new Color(0, 122, 204), 2));
loginButton.setFocusPainted(false);
loginButton.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
// Add a hover effect on the button
loginButton.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseEntered(java.awt.event.MouseEvent evt) {
        loginButton.setBackground(new Color(0, 122, 204)); // Lighter on hover
        loginButton.setForeground(Color.WHITE);
    }
    public void mouseExited(java.awt.event.MouseEvent evt) {
        loginButton.setBackground(Color.WHITE); // Default color
        loginButton.setForeground(new Color(0, 122, 204)); // Default text color
    }
});
gbc.gridy = 5;
panel.add(loginButton, gbc);
// Add status label for feedback
JLabel statusLabel = new JLabel("", JLabel.CENTER);
statusLabel.setFont(new Font("Arial", Font.PLAIN, 12));
statusLabel.setForeground(Color.RED);
gbc.gridx = 0;

```

```

gbc.gridy = 6;
panel.add(statusLabel, gbc);
// Add ActionListener for SignUp button
signupButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String username = usernameField.getText();
        String email = emailField.getText();
        String password = new String(passwordField.getPassword());
        if (username.isEmpty() || email.isEmpty() || password.isEmpty()) {
            statusLabel.setText("All fields are required!");
            return;
        }
        if (performSignup(username, email, password)) {
            statusLabel.setText("Signup successful! You can now log in.");
            statusLabel.setForeground(Color.GREEN);
            usernameField.setText("");
            emailField.setText("");
            passwordField.setText("");
        } else {
            statusLabel.setText("Signup failed. Try again.");
        }
    }
});
// Add ActionListener for Login Page button
loginButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        statusLabel.setText("Redirecting to login page...");
        frame.dispose(); // Close the current window
    }
});

```

```

        login.main(null); // Assuming you have a Login class
    }
});

// Show the frame
frame.add(panel);
frame.setVisible(true);
}

// Perform signup logic
private static boolean performSignup(String username, String email, String plainPassword) {
    // SQL INSERT query
    String sql = "INSERT INTO signup (username, email, password) VALUES (?, ?, ?)";
    try {
        // For security, hash the password (this is a placeholder, replace with a real hashing
        function)
        String hashedPassword = hashPassword(plainPassword);

        // Establish database connection and insert user data
        try (Connection conn = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
            PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, username);
            pstmt.setString(2, email);
            pstmt.setString(3, hashedPassword);
            int rowsInserted = pstmt.executeUpdate();
            return rowsInserted > 0;
        }
    } catch (Exception e) {
        System.err.println("Error during signup: " + e.getMessage());
        return false;
    }
}
}

```

```

// Placeholder method for hashing passwords
private static String hashPassword(String password) {
    // Implement a proper password hashing algorithm like BCrypt here
    return password; // Return plain text password for now (NOT secure)
}
}

```

LOGIN:

```

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

public class login {
    // Database connection details
    private static final String DB_URL = "jdbc:mysql://localhost:3306/bus";
    private static final String DB_USER = "root";
    private static final String DB_PASSWORD = "moni2626";
    public static void main(String[] args) {
        SwingUtilities.invokeLater(login::showLoginPage);
    }
    /**
     * Display the Login Page
     */
    public static void showLoginPage() {
        // Create and setup the login frame
        JFrame frame = new JFrame("Login Page");
        frame.setSize(450, 350);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLocationRelativeTo(null); // Center the frame
    }
}

```

```
frame.setResizable(false);
JPanel panel = new JPanel(new GridBagLayout());
GridBagConstraints gbc = new GridBagConstraints();
panel.setBorder(BorderFactory.createEmptyBorder(30, 30, 30, 30));
// Title Label
JLabel titleLabel = new JLabel("Login to Your Account");
titleLabel.setFont(new Font("Segoe UI", Font.BOLD, 24));
titleLabel.setForeground(new Color(0, 122, 204));
gbc.gridx = 0;
gbc.gridy = 0;
gbc.gridwidth = 2;
gbc.insets = new Insets(0, 0, 20, 0);
panel.add(titleLabel, gbc);
// Email Label and Field
JLabel emailLabel = new JLabel("Email:");
emailLabel.setFont(new Font("Arial", Font.PLAIN, 14));
gbc.gridx = 0;
gbc.gridy = 1;
gbc.gridwidth = 1;
panel.add(emailLabel, gbc);
JTextField emailField = new JTextField(20);
emailField.setFont(new Font("Arial", Font.PLAIN, 14));
gbc.gridx = 1;
gbc.gridy = 1;
panel.add(emailField, gbc);
// Password Label and Field
JLabel passwordLabel = new JLabel("Password:");
passwordLabel.setFont(new Font("Arial", Font.PLAIN, 14));
gbc.gridx = 0;
gbc.gridy = 2;
```



```
panel.add(passwordLabel, gbc);
JPasswordField passwordField = new JPasswordField(20);
passwordField.setFont(new Font("Arial", Font.PLAIN, 14));
gbc.gridx = 1;
gbc.gridy = 2;
panel.add(passwordField, gbc);

// Login Button
JButton loginButton = new JButton("Login");
loginButton.setFont(new Font("Arial", Font.BOLD, 16));
loginButton.setBackground(new Color(0, 122, 204));
loginButton.setForeground(Color.WHITE);
loginButton.setBorderPainted(false);
loginButton.setFocusPainted(false);
loginButton.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
gbc.gridx = 0;
gbc.gridy = 3;
gbc.gridwidth = 2;
gbc.insets = new Insets(20, 0, 10, 0);
panel.add(loginButton, gbc);

// Status Label
JLabel statusLabel = new JLabel("", JLabel.CENTER);
statusLabel.setFont(new Font("Arial", Font.PLAIN, 12));
statusLabel.setForeground(Color.RED);
gbc.gridx = 0;
gbc.gridy = 4;
panel.add(statusLabel, gbc);

// Login Button Action
loginButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
```

```

String email = emailField.getText();
String password = new String(passwordField.getPassword());
if (email.isEmpty() || password.isEmpty()) {
    statusLabel.setText("Both fields are required!");
} else if (authenticate(email, password)) {
    statusLabel.setText("Login successful!");
    statusLabel.setForeground(Color.GREEN);
    frame.dispose();
    PostLoginMenu.main(null);
} else {
    statusLabel.setText("Invalid email or password.");
}
}
});
// Add Panel to Frame and Display
frame.add(panel);
frame.setVisible(true);
}
/**
 * Authenticate user credentials against the database
 */
private static boolean authenticate(String email, String password) {
    String query = "SELECT * FROM signup WHERE email = ? AND password = ?";

    try (Connection connection = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
        PreparedStatement preparedStatement = connection.prepareStatement(query)) {
        preparedStatement.setString(1, email);
        preparedStatement.setString(2, password);
        ResultSet resultSet = preparedStatement.executeQuery();
        return resultSet.next();
    }
}

```

```

        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "Database error: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
        }
        return false;    }
    }

```

DASHBOARD:

```

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

public class PostLoginMenu {
    // Database connection details

    private static final String DB_URL = "jdbc:mysql://localhost:3306/bus"; // Replace with your
DB URL

    private static final String DB_USER = "root"; // Replace with your DB username
    private static final String DB_PASSWORD = "moni2626"; // Replace with your DB password

    public static void showMenu() {
        // Create the menu frame

        JFrame menuFrame = new JFrame("Main Menu");
        menuFrame.setSize(450, 400);
        menuFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        menuFrame.setLocationRelativeTo(null);

        JPanel panel = new JPanel(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        panel.setBorder(BorderFactory.createEmptyBorder(30, 30, 30, 30));

        // Title label

        JLabel menuTitleLabel = new JLabel("Welcome to the Main Menu", JLabel.CENTER);
        menuTitleLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));
        menuTitleLabel.setForeground(new Color(0, 122, 204));

        gbc.gridx = 0;
        gbc.gridy = 0;
    }
}

```

```
gbc.gridwidth = 2;
gbc.insets = new Insets(0, 0, 20, 0);
panel.add(menuTitleLabel, gbc);
// Buttons for options
JButton routeDetailsButton = createMenuButton("Route Details");
JButton bookTicketButton = createMenuButton("Book a Ticket");
JButton updateBookingButton = createMenuButton("Update Booking");
JButton viewBooking = createMenuButton("View Booking");
JButton deleteBooking = createMenuButton("Delete Booking");
JButton Check = createMenuButton("Members");
JButton update = createMenuButton("Update Details");
JButton logoutButton = createMenuButton("Logout");
// Arrange the buttons
gbc.gridx = 0;
gbc.gridy = 1;
gbc.gridwidth = 2;
panel.add(routeDetailsButton, gbc);
gbc.gridy = 2;
panel.add(bookTicketButton, gbc);
gbc.gridy = 3;
panel.add(viewBooking, gbc);

gbc.gridy = 4;
panel.add(deleteBooking, gbc);
gbc.gridy = 5;
panel.add(Check, gbc);
gbc.gridy = 6;
panel.add(update, gbc);
gbc.gridy = 7;
panel.add(logoutButton, gbc);
```

```

// Add button actions
routeDetailsButton.addActionListener(e -> {
    menuFrame.dispose();
    RouteDetails.open(); // Display route details from the database
});
bookTicketButton.addActionListener(e -> {
    menuFrame.dispose();
    // Replace with appropriate functionality
    TicketBooking.open();
});
viewBooking.addActionListener(e -> {
    menuFrame.dispose();
    ViewBookingDetails.open(); // Display route details from the database
});
deleteBooking.addActionListener(e -> {
    menuFrame.dispose();
    DeleteBooking.open(); // Display route details from the database
});
Check.addActionListener(e -> {
    menuFrame.dispose();
    PrintMembers.main(null); // Display route details from the database
});

update.addActionListener(e -> {
    menuFrame.dispose();
    updateDetail.main(null);
});
// ActionListener for the "Go to Login" button
logoutButton.addActionListener(e -> {
    menuFrame.dispose();

```

```

        login.main(null);
    });
    // Show the menu frame
    menuFrame.add(panel);
    menuFrame.setVisible(true);
}
/**
 * Utility function to create a menu button with styling
 */
private static JButton createMenuButton(String text) {
    JButton button = new JButton(text);
    button.setFont(new Font("Arial", Font.PLAIN, 14));
    button.setBackground(new Color(0, 122, 204));
    button.setForeground(Color.WHITE);
    button.setBorderPainted(false);
    button.setFocusPainted(false);
    button.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
    // Add hover effect
    button.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseEntered(java.awt.event.MouseEvent evt) {
            button.setBackground(new Color(0, 150, 255)); // Lighter on hover
        }
        public void mouseExited(java.awt.event.MouseEvent evt) {
            button.setBackground(new Color(0, 122, 204)); // Default color
        }
    });
    return button;
}
/**
 * Function to display route details from the database

```

```

*/
private static void displayRouteDetails() {
    try (Connection conn = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD)) {
        String query = "SELECT * FROM routes"; // Assuming a table named 'routes'
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(query);
        StringBuilder routesData = new StringBuilder("Route Details:\n\n");
        while (rs.next()) {
            routesData.append("Route ID: ").append(rs.getInt("route_id")).append("\n")
                .append("Start: ").append(rs.getString("start")).append("\n")
                .append("Destination: ").append(rs.getString("destination")).append("\n")
                .append("Fare: ").append(rs.getDouble("fare")).append("\n\n");
        }
        JOptionPane.showMessageDialog(null, routesData.toString());
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, "Error fetching route details: " + e.getMessage());
    }
}

public static void main(String[] args) {
    showMenu();
}
}

```

ROUTE DETAILS:

```

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

public class RouteDetails {
    // Database connection details

```

```

private static final String DB_URL = "jdbc:mysql://localhost:3306/bus";
private static final String DB_USER = "root";
private static final String DB_PASSWORD = "moni2626";
public static void open() {
    // Set the look and feel of the UI for better consistency across platforms
    try {
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    } catch (Exception e) {
        e.printStackTrace();
    }
    // Create and setup the frame for Route Details Finder
    JFrame frame = new JFrame("Route Details Finder");
    frame.setSize(900, 700); // Adjusted frame size for better content display
    frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    frame.setLocationRelativeTo(null); // Center the frame on the screen
    frame.setResizable(false);
    // Create a JPanel with GridBagLayout for better control of component placement
    JPanel panel = new JPanel(new GridBagLayout());
    GridBagConstraints gbc = new GridBagConstraints();
    panel.setBorder(BorderFactory.createEmptyBorder(30, 30, 30, 30)); // Added margin for
better spacing
    // Title Label
    JLabel titleLabel = new JLabel("Route Details Finder");
    titleLabel.setFont(new Font("Segoe UI", Font.BOLD, 28));
    titleLabel.setForeground(new Color(0, 122, 204)); // Vibrant color for title
    gbc.gridx = 0;
    gbc.gridy = 0;
    gbc.gridwidth = 2;
    gbc.insets = new Insets(0, 0, 30, 0); // Spacing after the title
    panel.add(titleLabel, gbc);
    // Route Name Label and TextField

```



```

JLabel routeLabel = new JLabel("Enter Route Name:");
routeLabel.setFont(new Font("Arial", Font.PLAIN, 16));
gbc.gridx = 0;
gbc.gridy = 1;
gbc.gridwidth = 1;
panel.add(routeLabel, gbc);

JTextField routeField = new JTextField(30); // Adjusted width of the text field
routeField.setFont(new Font("Arial", Font.PLAIN, 16));
routeField.setBorder(BorderFactory.createLineBorder(new Color(0, 122, 204), 2));
gbc.gridx = 1;
gbc.gridy = 1;
gbc.gridwidth = 2; // Allow the field to span two columns
panel.add(routeField, gbc);

// Search Button
JButton searchButton = new JButton("Search");
searchButton.setFont(new Font("Arial", Font.BOLD, 18));
searchButton.setBackground(new Color(0, 122, 204));
searchButton.setForeground(Color.WHITE);
searchButton.setBorderPainted(false);
searchButton.setFocusPainted(false);
searchButton.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));

// Hover effect for Search button
searchButton.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseEntered(java.awt.event.MouseEvent evt) {
        searchButton.setBackground(new Color(0, 150, 255)); // Lighter on hover
    }
    public void mouseExited(java.awt.event.MouseEvent evt) {
        searchButton.setBackground(new Color(0, 122, 204)); // Default color
    }
});

```

```

gbc.gridx = 0;
gbc.gridy = 2;
gbc.gridwidth = 3; // Button spans all columns
panel.add(searchButton, gbc);
// Result Area for displaying the route details
JTextArea resultArea = new JTextArea();
resultArea.setEditable(false); // Make the result area read-only
resultArea.setLineWrap(true);
resultArea.setWrapStyleWord(true);
resultArea.setFont(new Font("Arial", Font.PLAIN, 16));
resultArea.setBorder(BorderFactory.createLineBorder(new Color(0, 122, 204), 2));
JScrollPane scrollPane = new JScrollPane(resultArea); // Added scroll pane for better result
area visibility
scrollPane.setPreferredSize(new Dimension(700, 200)); // Adjusted size for result area
gbc.gridx = 0;
gbc.gridy = 3;
gbc.gridwidth = 3;
gbc.insets = new Insets(10, 0, 20, 0); // Added space between result area and buttons
panel.add(scrollPane, gbc);
// Buttons panel (Back and Login buttons)
JPanel buttonPanel = new JPanel();
buttonPanel.setLayout(new FlowLayout(FlowLayout.CENTER));
// Back Button
JButton backButton = new JButton("Back");
backButton.setFont(new Font("Arial", Font.PLAIN, 16));
backButton.setBackground(Color.WHITE);
backButton.setForeground(new Color(0, 122, 204));
backButton.setBorder(BorderFactory.createLineBorder(new Color(0, 122, 204), 2));
backButton.setFocusPainted(false);
backButton.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
// Hover effect for Back button

```

```

backButton.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseEntered(java.awt.event.MouseEvent evt) {
        backButton.setBackground(new Color(0, 122, 204)); // Lighter on hover
        backButton.setForeground(Color.WHITE);
    }
    public void mouseExited(java.awt.event.MouseEvent evt) {
        backButton.setBackground(Color.WHITE); // Default color
        backButton.setForeground(new Color(0, 122, 204)); // Default text color
    }
});

// Login Button
JButton loginButton = new JButton("Login");
loginButton.setFont(new Font("Arial", Font.PLAIN, 16));
loginButton.setBackground(new Color(255, 255, 255));
loginButton.setForeground(new Color(0, 122, 204));
loginButton.setBorder(BorderFactory.createLineBorder(new Color(0, 122, 204), 2));
loginButton.setFocusPainted(false);
loginButton.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));

// Hover effect for Login button
loginButton.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseEntered(java.awt.event.MouseEvent evt) {
        loginButton.setBackground(new Color(0, 122, 204)); // Lighter on hover
        loginButton.setForeground(Color.WHITE);
    }
    public void mouseExited(java.awt.event.MouseEvent evt) {
        loginButton.setBackground(Color.WHITE); // Default color
        loginButton.setForeground(new Color(0, 122, 204)); // Default text color
    }
});

buttonPanel.add(backButton);

```

```

buttonPanel.add(loginButton);

gbc.gridx = 0;
gbc.gridy = 4;
gbc.gridwidth = 3;
panel.add(buttonPanel, gbc);
// Add action listener for the search button
searchButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String routeName = routeField.getText().trim();
        if (routeName.isEmpty()) {
            resultArea.setText("Please enter a route name.");
        } else {
            String result = getRouteDetails(routeName);
            resultArea.setText(result);
        }
    }
});
// Add action listener for the back button
backButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        frame.dispose(); // Close the Route Details window
    }
});
// Add action listener for the login button
loginButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        frame.dispose(); // Close the Route Details window
    }
});

```

```

        login.main(null); // Assuming you have a Login class
    }
});
// Show the frame
frame.add(panel);
frame.setVisible(true);
}
/**
 * Method to retrieve route details from the database
 *
 * @param routeName The name of the route to search for
 * @return A string containing the route details or an error message
 */
private static String getRouteDetails(String routeName) {
    String query = "SELECT * FROM routes WHERE route_name = ?";
    try (Connection connection = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
        PreparedStatement preparedStatement = connection.prepareStatement(query)) {
        preparedStatement.setString(1, routeName);
        ResultSet resultSet = preparedStatement.executeQuery();
        if (resultSet.next()) {
            return String.format(
                "=== Route Details ===\n" +
                "Route ID: %d\n" +
                "Route Name: %s\n" +
                "Start Point: %s\n" +
                "End Point: %s\n" +
                "Distance (km): %.2f\n" +
                "Estimated Time: %s\n" +
                "Fare: %.2f\n",
                resultSet.getInt("route_id"),

```

```

        resultSet.getString("route_name"),
        resultSet.getString("start_point"),
        resultSet.getString("end_point"),
        resultSet.getDouble("distance_km"),
        resultSet.getTime("estimated_time"),
        resultSet.getDouble("fare")
    );
} else {
    return "No route found with the name: " + routeName;}
} catch (SQLException e) {
    return "Database error: " + e.getMessage();}
}
}

```

TICKET BOOKING:

```

import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import java.time.LocalDate;
import javax.swing.*;
import javax.swing.border.LineBorder;
public class TicketBooking {
    // Database connection details
    private static final String DB_URL = "jdbc:mysql://localhost:3306/bus";
    private static final String DB_USER = "root";
    private static final String DB_PASSWORD = "moni2626";
    public static void open() {
        // Set the look and feel of the UI for better consistency across platforms
        try {
            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

```

```
} catch (Exception e) {
    e.printStackTrace();
}

// Create the frame
JFrame frame = new JFrame("Ticket Booking");
frame.setSize(700, 600);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setLocationRelativeTo(null); // Center the window
frame.setResizable(false);

// Create JPanel with GridBagLayout for better component arrangement
JPanel panel = new JPanel(new GridBagLayout());
GridBagConstraints gbc = new GridBagConstraints();
panel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20)); //add margin to the panel

// Title Label
JLabel titleLabel = new JLabel("Ticket Booking");
titleLabel.setFont(new Font("Segoe UI", Font.BOLD, 28));
titleLabel.setForeground(new Color(0, 122, 204));
gbc.gridx = 0;
gbc.gridy = 0;
gbc.gridwidth = 2;
gbc.insets = new Insets(0, 0, 30, 0); // Add space after title
panel.add(titleLabel, gbc);

// Route Name
JLabel routeNameLabel = new JLabel("Route Name:");
routeNameLabel.setFont(new Font("Arial", Font.PLAIN, 16));
gbc.gridx = 0;
gbc.gridy = 1;
gbc.gridwidth = 1;
gbc.insets = new Insets(0, 0, 10, 0); // Add space between rows
panel.add(routeNameLabel, gbc);
```

```
JTextField routeNameField = new JTextField(30); // Wider field
routeNameField.setFont(new Font("Arial", Font.PLAIN, 16));
routeNameField.setBackground(Color.WHITE); // Set background color
routeNameField.setBorder(new LineBorder(new Color(0, 122, 204), 2)); // Set border color
gbc.gridx = 1;
gbc.gridy = 1;
panel.add(routeNameField, gbc);
// Start Point
JLabel startPointLabel = new JLabel("Start Point:");
startPointLabel.setFont(new Font("Arial", Font.PLAIN, 16));
gbc.gridx = 0;
gbc.gridy = 2;
panel.add(startPointLabel, gbc);
JTextField startPointField = new JTextField(30); // Wider field
startPointField.setFont(new Font("Arial", Font.PLAIN, 16));
startPointField.setBackground(Color.WHITE); // Set background color
startPointField.setBorder(new LineBorder(new Color(0, 122, 204), 2)); // Set border color
gbc.gridx = 1;
gbc.gridy = 2;
panel.add(startPointField, gbc);
// Travel Date
JLabel travelDateLabel = new JLabel("Travel Date (YYYY-MM-DD):");
travelDateLabel.setFont(new Font("Arial", Font.PLAIN, 16));
gbc.gridx = 0;
gbc.gridy = 3;
panel.add(travelDateLabel, gbc);
JTextField travelDateField = new JTextField(30); // Wider field
travelDateField.setFont(new Font("Arial", Font.PLAIN, 16));
travelDateField.setBackground(Color.WHITE); // Set background color
travelDateField.setBorder(new LineBorder(new Color(0, 122, 204), 2)); // Set border color
```



```
gbc.gridx = 1;
gbc.gridy = 3;
panel.add(travelDateField, gbc);
// User ID
JLabel userIdLabel = new JLabel("User ID:");
userIdLabel.setFont(new Font("Arial", Font.PLAIN, 16));
gbc.gridx = 0;
gbc.gridy = 4;
panel.add(userIdLabel, gbc);
JTextField userIdField = new JTextField(30); // Wider field
userIdField.setFont(new Font("Arial", Font.PLAIN, 16));
userIdField.setBackground(Color.WHITE); // Set background color
userIdField.setBorder(new LineBorder(new Color(0, 122, 204), 2)); // Set border color
gbc.gridx = 1;
gbc.gridy = 4;
panel.add(userIdField, gbc);
// Number of Seats
JLabel numberOfSeatsLabel = new JLabel("Number of Seats:");
numberOfSeatsLabel.setFont(new Font("Arial", Font.PLAIN, 16));
gbc.gridx = 0;
gbc.gridy = 5;
panel.add(numberOfSeatsLabel, gbc);
JTextField numberOfSeatsField = new JTextField(30); // Wider field
numberOfSeatsField.setFont(new Font("Arial", Font.PLAIN, 16));
numberOfSeatsField.setBackground(Color.WHITE); // Set background color
numberOfSeatsField.setBorder(new LineBorder(new Color(0, 122, 204), 2)); // Set border color
gbc.gridx = 1;
gbc.gridy = 5;
panel.add(numberOfSeatsField, gbc);
// Book Ticket Button
```

```
JButton bookTicketButton = new JButton("Book Ticket");
bookTicketButton.setFont(new Font("Arial", Font.BOLD, 18));
bookTicketButton.setBackground(new Color(0, 122, 204));
bookTicketButton.setForeground(Color.WHITE);
bookTicketButton.setBorderPainted(false);
bookTicketButton.setFocusPainted(false);
bookTicketButton.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));

gbc.gridx = 0;
gbc.gridy = 6;
gbc.gridwidth = 2;
panel.add(bookTicketButton, gbc);

// Message Label
JLabel messageLabel = new JLabel("Please enter the details above.");
messageLabel.setFont(new Font("Arial", Font.PLAIN, 16));
messageLabel.setForeground(Color.RED);

gbc.gridx = 0;
gbc.gridy = 7;
gbc.gridwidth = 2;
panel.add(messageLabel, gbc);

// Go to Login Page Button
JButton goToLoginButton = new JButton("Go to Login Page");
goToLoginButton.setFont(new Font("Arial", Font.PLAIN, 16));
goToLoginButton.setBackground(Color.WHITE);
goToLoginButton.setForeground(new Color(0, 122, 204));
goToLoginButton.setBorder(BorderFactory.createLineBorder(new Color(0, 122, 204), 2));
goToLoginButton.setFocusPainted(false);
goToLoginButton.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));

gbc.gridx = 0;
gbc.gridy = 8;
gbc.gridwidth = 2;
```

```

panel.add(goToLoginButton, gbc);
// Action listener for booking ticket
bookTicketButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // Get values from input fields
        String routeName = routeNameField.getText();
        String startPoint = startPointField.getText();
        String travelDate = travelDateField.getText();
        String userIdInput = userIdField.getText();
        String seatsInput = numberOfSeatsField.getText();
        if (routeName.isEmpty() || startPoint.isEmpty() || travelDate.isEmpty() ||
            userIdInput.isEmpty() || seatsInput.isEmpty()) {
            messageLabel.setText("All fields are required.");
            return;
        }
        if (!isValidDate(travelDate)) {
            messageLabel.setText("Invalid date format. Use YYYY-MM-DD.");
            return;
        }
        try {
            int userId = Integer.parseInt(userIdInput);
            int numberOfSeats = Integer.parseInt(seatsInput);
            int routeId = getRouteId(routeName, startPoint);
            if (routeId != -1) {
                double farePerSeat = getRouteFare(routeId);
                double totalFare = farePerSeat * numberOfSeats;
                insertBooking(userId, routeId, travelDate, numberOfSeats, totalFare,
messageLabel);
            } else {
                messageLabel.setText("Invalid route name or start point.");
            }
        }
    }
});

```

```

        } catch (NumberFormatException ex) {
            messageLabel.setText("Invalid input. Please enter valid numbers for User ID and
Seats.");
        }
    }
});

// Action listener for the 'Go to Login Page' button
goToLoginButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        frame.setVisible(false); // Hide the current window
        login.main(null); // Assuming you have a Login class
    }
});

// Show the frame
frame.add(panel);
frame.setVisible(true);
}

private static boolean isValidDate(String date) {
    return date.matches("\\d{4}-\\d{2}-\\d{2}");
}

private static void insertBooking(int userId, int routeId, String travelDate, int numberOfSeats,
double totalFare, JLabel messageLabel) {
    String query = "INSERT INTO booking (user_id, route_id, booking_date, travel_date,
number_of_seats, total_fare, status) VALUES (?, ?, ?, ?, ?, ?, ?)";

    try (Connection connection = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
        PreparedStatement preparedStatement = connection.prepareStatement(query)) {
        preparedStatement.setInt(1, userId);
        preparedStatement.setInt(2, routeId);
        preparedStatement.setDate(3, Date.valueOf(LocalDate.now())); // Current date
        preparedStatement.setDate(4, Date.valueOf(travelDate)); // Travel date
        preparedStatement.setInt(5, numberOfSeats);
    }
}

```

```

        preparedStatement.setDouble(6, totalFare);
        preparedStatement.setString(7, "Booked"); // Default status
        int rowsAffected = preparedStatement.executeUpdate();
        if (rowsAffected > 0) {
            messageLabel.setText("Booking successful! Total fare: ₹" + totalFare);
        } else {
            messageLabel.setText("Booking failed. Try again.");
        }
    } catch (SQLException e) {
        e.printStackTrace();
        messageLabel.setText("Database error: " + e.getMessage());
    }
}

private static int getRouteId(String routeName, String startPoint) {
    String query = "SELECT route_id FROM routes WHERE route_name = ? AND start_point = ?";

    try (Connection connection = DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD);
        PreparedStatement preparedStatement = connection.prepareStatement(query)) {
        preparedStatement.setString(1, routeName);
        preparedStatement.setString(2, startPoint);
        ResultSet resultSet = preparedStatement.executeQuery();
        if (resultSet.next()) {
            return resultSet.getInt("route_id");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }

    return -1; // Return -1 if no route found
}

private static double getRouteFare(int routeId) {
    String query = "SELECT fare FROM routes WHERE route_id = ?";

```

```

        try (Connection connection = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);

            PreparedStatement preparedStatement = connection.prepareStatement(query)) {
            preparedStatement.setInt(1, routeId);
            ResultSet resultSet = preparedStatement.executeQuery();
            if (resultSet.next()) {
                return resultSet.getDouble("fare");
            }
        } catch (SQLException e) {
            e.printStackTrace(); }
        return 0.0; // Return 0.0 if no fare found    }
    public static void main(String[] args) {
        open(); // Open the Ticket Booking GUI initially    }
}

```

VIEW BOOKING:

```

import java.awt.*;
import java.sql.*;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
public class ViewBookingDetails {
    // Database connection details
    private static final String DB_URL = "jdbc:mysql://localhost:3306/bus";
    private static final String DB_USER = "root";
    private static final String DB_PASSWORD = "moni2626";
    public static void open() {
        // Set up the frame
        JFrame frame = new JFrame("Booking Details");
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setSize(800, 500);
        frame.setLocationRelativeTo(null);
        // Create the panel
    }
}

```

```

JPanel panel = new JPanel(new BorderLayout());
JLabel titleLabel = new JLabel("Booking Details", SwingConstants.CENTER);
titleLabel.setFont(new Font("Segoe UI", Font.BOLD, 24));
titleLabel.setForeground(new Color(0, 122, 204));
panel.add(titleLabel, BorderLayout.NORTH);
// Table setup
DefaultTableModel model = new DefaultTableModel();
JTable table = new JTable(model);
JScrollPane scrollPane = new JScrollPane(table);
// Add columns to the table model
model.addColumn("Booking ID");
model.addColumn("User ID");
model.addColumn("Route ID");
model.addColumn("Booking Date");
model.addColumn("Travel Date");
model.addColumn("Seats");
model.addColumn("Total Fare");
model.addColumn("Status");
// Populate table with data from database
try (Connection connection = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
    Statement statement = connection.createStatement();
    ResultSet resultSet = statement.executeQuery("SELECT * FROM booking")) {
    while (resultSet.next()) {
        Object[] row = {
            resultSet.getInt("booking_id"),
            resultSet.getInt("user_id"),
            resultSet.getInt("route_id"),
            resultSet.getDate("booking_date"),
            resultSet.getDate("travel_date"),
            resultSet.getInt("number_of_seats"),

```

```

        resultSet.getDouble("total_fare"),
        resultSet.getString("status")
    };
    model.addRow(row);    }
} catch (SQLException e) {
    JOptionPane.showMessageDialog(frame, "Error fetching data: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);    }

// Style the table
table.setFont(new Font("Arial", Font.PLAIN, 14));
table.setRowHeight(25);
table.getTableHeader().setFont(new Font("Arial", Font.BOLD, 16));
table.getTableHeader().setBackground(new Color(0, 122, 204));
table.getTableHeader().setForeground(Color.WHITE);
panel.add(scrollPane, BorderLayout.CENTER);

// Back Button
JButton backButton = new JButton("Back");
backButton.setFont(new Font("Arial", Font.BOLD, 16));
backButton.setBackground(new Color(0, 122, 204));
backButton.setForeground(Color.WHITE);
backButton.setFocusPainted(false);
backButton.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
backButton.addActionListener(e -> {
    frame.dispose(); // Close the current window
    PostLoginMenu.main(null);
    System.out.println("Back button pressed. Implement logic to go back to the previous
screen.");
});

// Close Button
JButton closeButton = new JButton("Close");
closeButton.setFont(new Font("Arial", Font.BOLD, 16));
closeButton.setBackground(new Color(0, 122, 204));

```



```

        closeButton.setForeground(Color.WHITE);
        closeButton.setFocusPainted(false);
        closeButton.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
        closeButton.addActionListener(e -> frame.dispose());
        // Create a panel to hold both buttons
        JPanel buttonPanel = new JPanel();
        buttonPanel.setLayout(new FlowLayout(FlowLayout.CENTER));
        buttonPanel.add(backButton);
        buttonPanel.add(closeButton);
        panel.add(buttonPanel, BorderLayout.SOUTH);
        frame.add(panel);
        frame.setVisible(true);
    }
    public static void main(String[] args) {
        open();    }
}

```

DELETE BOOKING:

```

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

public class DeleteBooking {
    // Database connection details
    private static final String DB_URL = "jdbc:mysql://localhost:3306/bus";
    private static final String DB_USER = "root";
    private static final String DB_PASSWORD = "moni2626";
    public static void open() {
        // Set up the frame
        JFrame frame = new JFrame("Delete Booking");
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setSize(400, 400); // Adjusted size
    }
}

```

```

frame.setLocationRelativeTo(null);
frame.setUndecorated(true); // Removes window border for cleaner look
frame.getRootPane().setWindowDecorationStyle(JRootPane.PLAIN_DIALOG); // Adds
rounded edges

// Create the panel with custom background color
JPanel panel = new JPanel();
panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
panel.setBackground(new Color(255, 255, 255));
// Title label with stylish font and color
JLabel titleLabel = new JLabel("Delete Booking");
titleLabel.setFont(new Font("Segoe UI", Font.BOLD, 28));
titleLabel.setForeground(new Color(0, 122, 204));
titleLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
panel.add(titleLabel);
panel.add(Box.createVerticalStrut(20)); // Adds spacing
// Input field for booking_id
JLabel bookingIdLabel = new JLabel("Enter Booking ID:");
bookingIdLabel.setFont(new Font("Arial", Font.PLAIN, 16));
bookingIdLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
panel.add(bookingIdLabel);
panel.add(Box.createVerticalStrut(10)); // Adds spacing
JTextField bookingIdField = new JTextField();
bookingIdField.setFont(new Font("Arial", Font.PLAIN, 16));
bookingIdField.setPreferredSize(new Dimension(300, 30));
bookingIdField.setHorizontalAlignment(JTextField.CENTER);
bookingIdField.setBorder(BorderFactory.createLineBorder(new Color(0, 122, 204), 2)); //
Blue border
panel.add(bookingIdField);
panel.add(Box.createVerticalStrut(20)); // Adds spacing
// Message label to show status
JLabel messageLabel = new JLabel("");

```

```

messageLabel.setFont(new Font("Arial", Font.PLAIN, 14));
messageLabel.setForeground(Color.RED);
messageLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
panel.add(messageLabel);
panel.add(Box.createVerticalStrut(20)); // Adds spacing
// Delete button with modern look
JButton deleteButton = new JButton("Delete");
deleteButton.setFont(new Font("Arial", Font.BOLD, 16));
deleteButton.setBackground(new Color(0, 122, 204));
deleteButton.setForeground(Color.WHITE);
deleteButton.setFocusPainted(false);
deleteButton.setBorder(BorderFactory.createLineBorder(new Color(0, 122, 204), 2, true)); //
Rounded corners with LineBorder
deleteButton.setPreferredSize(new Dimension(200, 40));
deleteButton.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
deleteButton.setAlignmentX(Component.CENTER_ALIGNMENT);
deleteButton.addActionListener(e -> {
    String bookingIdInput = bookingIdField.getText();
    if (bookingIdInput.isEmpty()) {
        messageLabel.setText("Booking ID is required.");
        return;    }
    try {
        int bookingId = Integer.parseInt(bookingIdInput);
        if (deleteBookingFromDatabase(bookingId)) {
            messageLabel.setText("Booking deleted successfully.");
            messageLabel.setForeground(Color.GREEN); // Success message in green
        } else {
            messageLabel.setText("Booking ID not found.");    }
    } catch (NumberFormatException ex) {
        messageLabel.setText("Invalid Booking ID.");    }
});

```

```

panel.add(deleteButton);
panel.add(Box.createVerticalStrut(20)); // Adds spacing
// Back button with hover effect
JButton backButton = new JButton("Back");
backButton.setFont(new Font("Arial", Font.PLAIN, 16));
backButton.setBackground(Color.WHITE);
backButton.setForeground(new Color(0, 122, 204));
backButton.setAlignmentX(Component.CENTER_ALIGNMENT);
backButton.setFocusPainted(false);
backButton.setBorder(BorderFactory.createLineBorder(new Color(0, 122, 204), 2));
backButton.setPreferredSize(new Dimension(200, 40));
backButton.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
backButton.addActionListener(e -> {
    frame.setVisible(false); // Hide the current window
    PostLoginMenu.main(null); // Assuming `PostLoginMenu` is your previous menu
});
panel.add(backButton);
// Add the panel to the frame and show it
frame.add(panel);
frame.setVisible(true); }

private static boolean deleteBookingFromDatabase(int bookingId) {
    String query = "DELETE FROM booking WHERE booking_id = ?";
    try (Connection connection = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
        PreparedStatement preparedStatement = connection.prepareStatement(query)) {
        preparedStatement.setInt(1, bookingId);
        int rowsAffected = preparedStatement.executeUpdate();
        return rowsAffected > 0; // Return true if a row was deleted
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return false; // Return false if an error occurs or no rows were delete }

```

```
public static void main(String[] args) {  
    open(); }  
}
```

SQL QUERIES:

```
CREATE TABLE signup (  
    id INT AUTO_INCREMENT PRIMARY KEY, -- Unique ID for each user  
    username VARCHAR(50) NOT NULL UNIQUE, -- User's chosen username  
    email VARCHAR(100) NOT NULL UNIQUE, -- User's email address  
    password VARCHAR(255) NOT NULL -- Hashed password  
);
```

```
CREATE TABLE login (  
    login_id INT AUTO_INCREMENT PRIMARY KEY, -- Unique ID for each login entry  
    email VARCHAR(100) NOT NULL, -- Email used for login  
    password VARCHAR(255) NOT NULL, -- Password provided for login  
    FOREIGN KEY (email) REFERENCES signup(email) -- Links to the `signup` table  
);
```

```
select * from login;
```

```
SELECT  
    signup.id AS user_id,  
    signup.username,  
    signup.email,  
    signup.password AS signup_password,  
    login.login_id,  
    login.password AS login_password  
FROM  
    signup  
LEFT JOIN  
    login
```

ON

signup.email = login.email;

CREATE TABLE routes (

route_id INT AUTO_INCREMENT PRIMARY KEY, -- Unique ID for each route
route_name VARCHAR(100) NOT NULL, -- Descriptive name for the route
start_point VARCHAR(100) NOT NULL, -- Starting location of the route
end_point VARCHAR(100) NOT NULL, -- Ending location of the route
distance_km DECIMAL(10, 2) NOT NULL, -- Distance of the route in kilometers
estimated_time TIME NOT NULL, -- Estimated travel time
fare DECIMAL(10, 2) NOT NULL -- Fare for the route

);

SELECT* FROM routes;

CREATE TABLE booking (

booking_id INT AUTO_INCREMENT PRIMARY KEY, -- Unique ID for each booking
user_id INT NOT NULL, -- ID of the user making the booking
route_id INT NOT NULL, -- ID of the route for the booking
booking_date DATE NOT NULL, -- Date of the booking
travel_date DATE NOT NULL, -- Date of travel
number_of_seats INT NOT NULL, -- Number of seats booked
total_fare DECIMAL(10, 2) NOT NULL, -- Total fare for the booking
status ENUM('Booked', 'Cancelled') DEFAULT 'Booked', -- Status of the booking
FOREIGN KEY (user_id) REFERENCES signup(id), -- Links to the `signup` table
FOREIGN KEY (route_id) REFERENCES routes(route_id) -- Links to the `routes` table

);

SELECT

booking.booking_id,
booking.user_id,
booking.booking_date,

```
    booking.travel_date,
    booking.number_of_seats,
    booking.total_fare,
    booking.status,
    routes.route_name,
    routes.start_point,
    routes.end_point,
    routes.distance_km,
    routes.estimated_time,
    routes.fare
FROM
    booking
JOIN
    routes
ON
    booking.route_id = routes.route_id;

ALTER TABLE routes
ADD COLUMN seats_available INT NOT NULL DEFAULT 40; -- Assume 40 seats available
by default

START TRANSACTION; -- Start a transaction to ensure atomicity

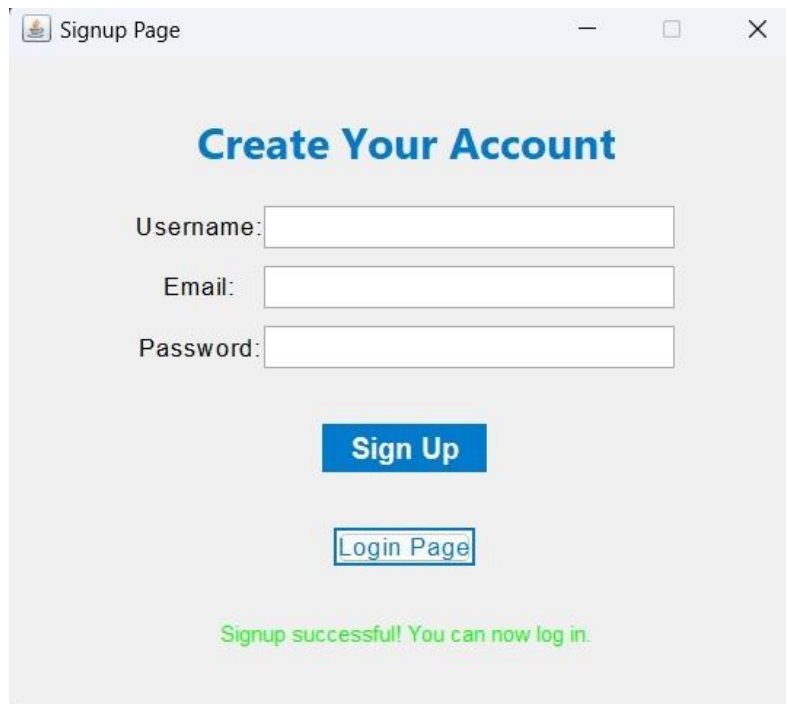
-- Update the available seats for the route
UPDATE routes
JOIN booking ON routes.route_id = booking.route_id
SET routes.seats_available = routes.seats_available - booking.number_of_seats
WHERE booking.booking_id = booking_id; -- Specify the booking_id of the booking to update

SELECT * FROM routes;
```

CHAPTER 5

RESULT:

SIGNUP PAGE:



Signup Page

Create Your Account

Username:

Email:

Password:

[Sign Up](#)

[Login Page](#)

Signup successful! You can now log in.

Signup Page

Create Your Account

Username:

xyz

Email:

xyz@gmail.com


Password:

●●●

Sign Up

Login Page

LOGIN:

 Login Page

Login to Your Account

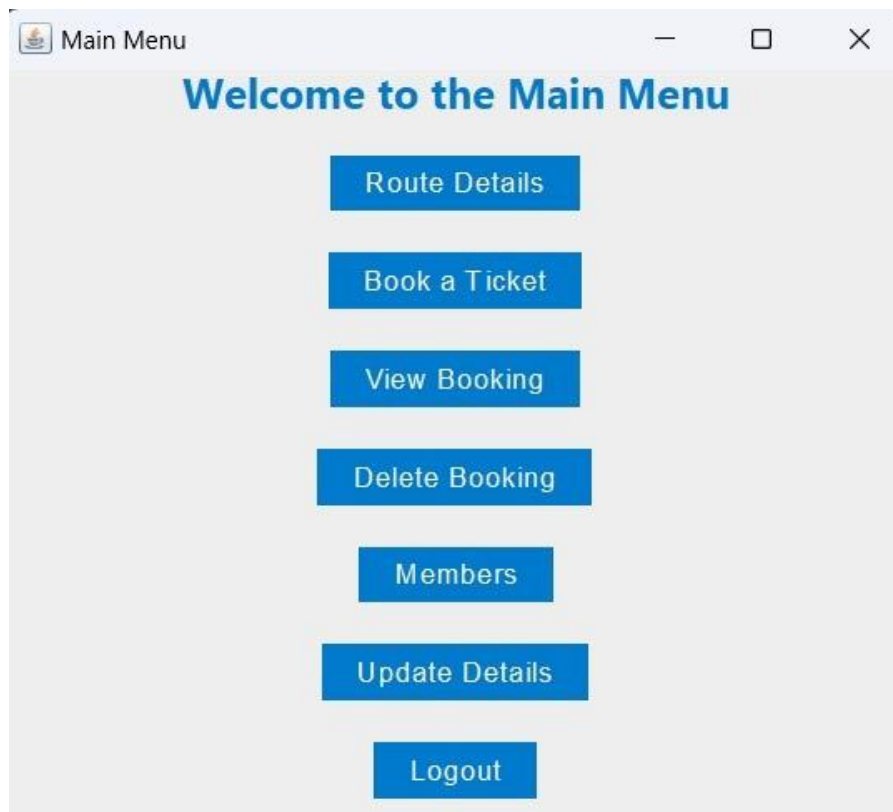
Email:

xyz@gmail.com

Password:

●●●

Login



DASHBOARD:

ROUTE DETAILS:

Route Details Finder

Enter Route Name:

Search

=== Route Details ===

Route ID: 3

Route Name: Beach Line

Start Point: Chennai Central

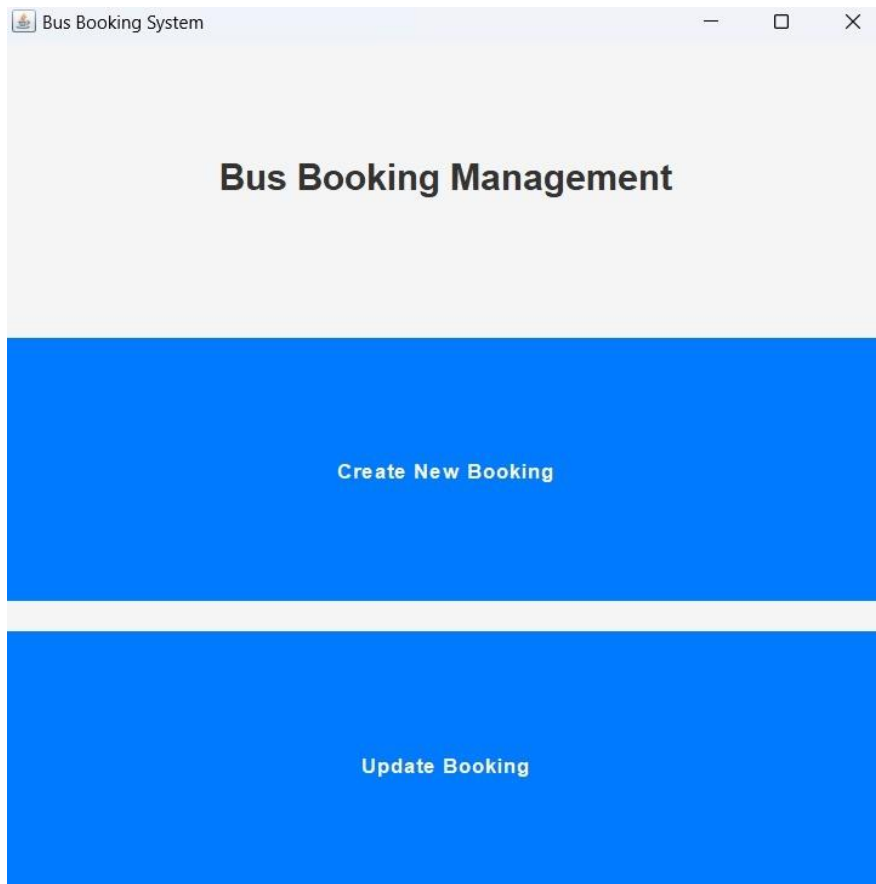
End Point: Marina Beach

Distance (km): 5.00

Estimated Time: 00:20:00

Fare: 10.00

[Login](#) [Back](#)



BOOKING PAGE:

Ticket Booking

Ticket Booking

Route Name:

Beach Line

Start Point:

Chennai Central

Travel Date (YYYY-MM-DD):

2024-12-12

User ID:

11

Number of Seats:

3

Book Ticket

Booking successful! Total fare: ₹30.0

Back

TICKET BOOKING:

BOOKING DETAILS:

Booking Details

-□×

Booking Details

1	1	10	2024-11-16	2024-12-01	2	120.0	Cancelled
2	2	14	2024-11-18	2024-12-10	2	120.0	Cancelled
3	2	3	2024-11-18	2025-01-01	2	20.0	Booked
4	2	3	2024-11-18	2025-01-01	2	20.0	Booked
5	5	8	2024-11-18	2024-12-10	2	70.0	Booked
8	2	4	2024-11-18	2024-12-12	2	50.0	Booked
9	2	1	2024-11-18	2024-12-14	1	30.0	Booked
16	11	3	2024-11-20	2024-12-12	3	30.0	Booked

UPDATE BOOKING:

Bus Booking System

Booking ID:

Field to Update:

Travel Date

New Value:

Update Booking

Back

Delete Booking

Enter Booking ID:

16

Booking deleted successfully.

Delete

Back

DELETE BOOKING:

6.1 Unit Testing

Unit testing is a testing technique in which modules are tested individually. Small individual units of source code are tested to determine whether it is fit to use or not. Different modules of games are put to test while the modules are being developed. Here modules refer to individual levels, players, scenes

6.2 Integration Testing

Integration testing is the technique in which individual components or modules are grouped together and tested. It occurs after testing. The inputs for the integrated testing are the modules that have already been unit tested.

6.3 System Testing

System testing is conducted on the entire system as a whole to check whether the system meets its requirements or not. software was installed on different systems and any errors or bugs that occurred were fixed.

6.4 Acceptance Testing

User Acceptance is defined as a type of testing performed by the Client to certify the system with respect to the requirements that was agreed upon. This testing happens in the final phase of testing before moving the software application to the Market or Production environment.

CHAPTER 7

7.1 CONCLUSION

After we have completed the project we are sure the problems in the existing system would overcome. The “**BUS ROUTE MANAGEMENT SYSTEM**” process made computerized to reduce human errors and to increase the efficiency. The main focus of this project is to lessen human efforts. The maintenance of the records is made efficient, as all the records are stored in the database, through which data can be retrieved easily. The navigation control is provided in all the forms to navigate through the large amount of records. If the numbers of records are very large then user has to just type in the search string and user gets the results immediately. The editing is also made simpler. The user has to just type in the required field and press the update button to update the desired field.

The Users are provide a particular unique Email id. So that they can access the routes correctly and without errors. Our main aim of the project is to get the correct booking information and tickets for the particular user on any available routes.

CHAPTER 8

RESEARCH AND REFERENCE

- <https://developer.ibm.com/languages/java/articles/>
- <https://dzone.com/java>
- <https://www.w3schools.com/java/>
- <https://www.geeksforgeeks.org/introduction-to-jdbc/>
- <https://github.com/s-a-c-h-i-n/Bus-Management-System>