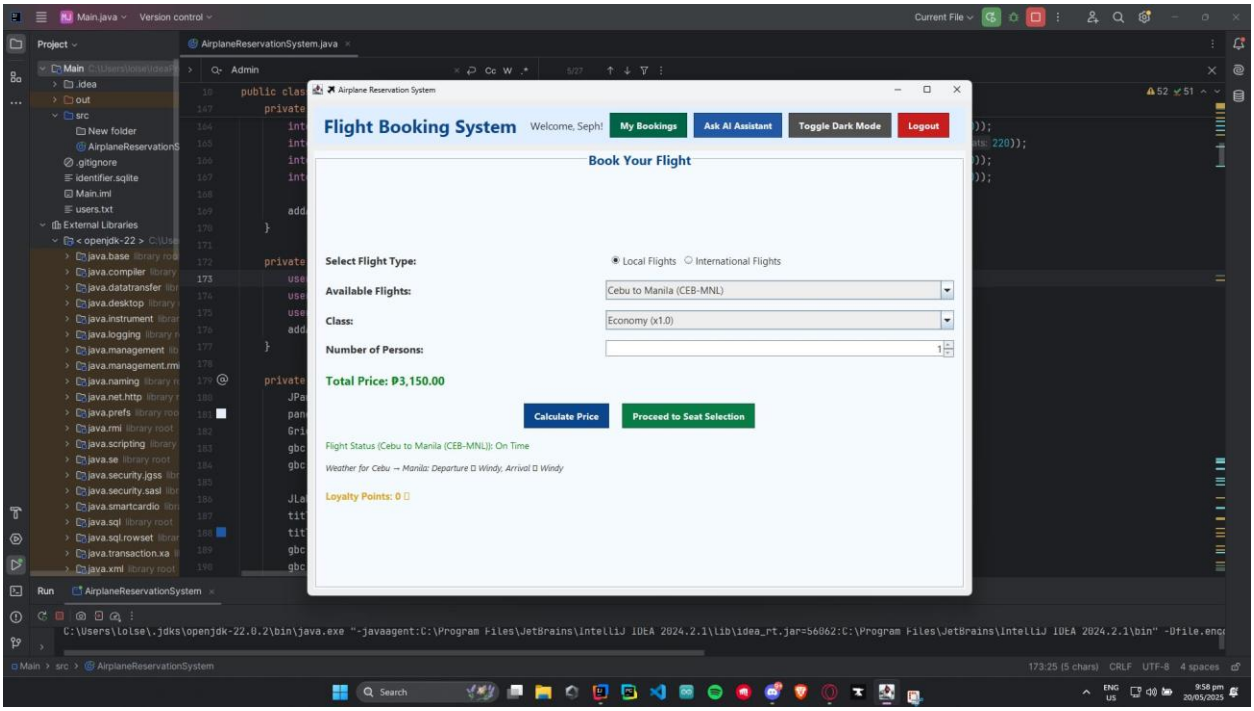
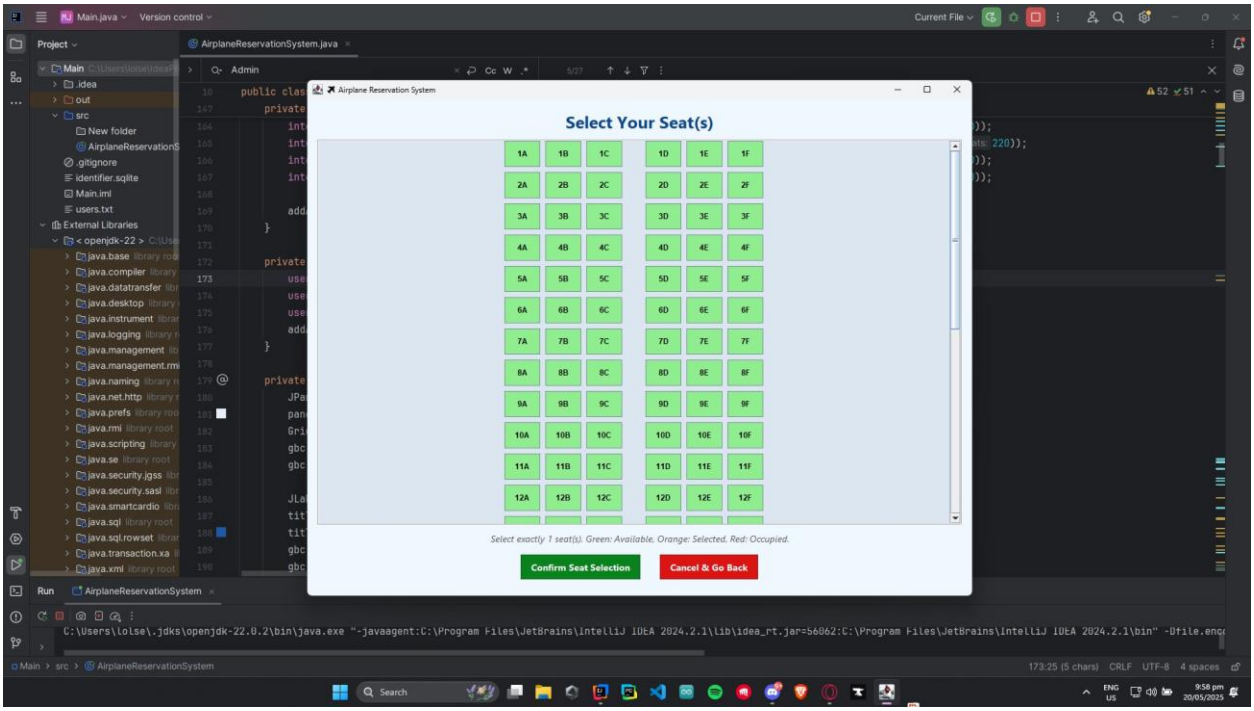
Brief discussion of your project

The development of this **Airplane Reservation System** provided a practical application of object-oriented programming principles and GUI design using Java Swing. Key learnings included mastering event handling for dynamic user interfaces, effectively managing state transitions across multiple panels via CardLayout, and structuring data for efficient retrieval and updates. While the current system offers robust core functionalities like user authentication, flight booking, and administrative oversight, future enhancements could involve integrating real-time external APIs for live flight data and weather, implementing a more sophisticated database for persistent storage, and expanding the chatbot's capabilities with advanced natural language processing. This project has significantly strengthened our proficiency in software development and problem-solving within a Java environment.

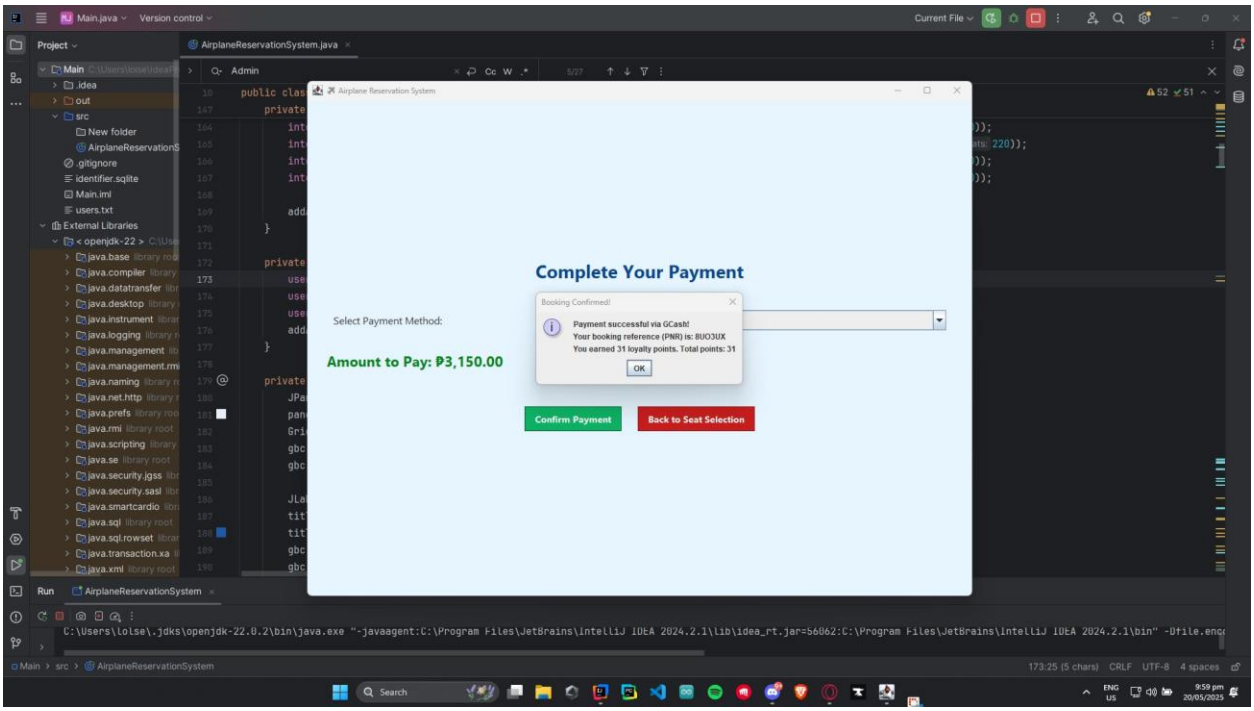
SS of outputs with explanation



The Airplane Reservation System stands as a robust demonstration of GUI development, event handling, and data management within the Java Swing framework. It showcases the application of object-oriented principles in creating an interactive and functional system, encompassing user authentication, complex transactional workflows, and administrative oversight. This project provides a strong foundation for understanding the intricacies of building comprehensive desktop applications.



It demonstrates the application's ability to render interactive seat maps, manage seat availability, and visually communicate seat status, contributing to an intuitive and efficient booking experience.



The payment functionality within the Airplane Reservation System is the crucial final step that transforms selected flight and seat preferences into a confirmed booking. It seamlessly integrates the calculated total price from previous steps, validates the user's payment input, generates a unique PNR, and updates the system's booking records, potentially rewarding the user with loyalty points, thereby finalizing the reservation process.

SS of codes with comments

```java
import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.util.List; // Keep this import, might be used later or was intended.
import java.util.TimerTask;
import java.text.SimpleDateFormat;

public class AirplaneReservationSystem {

    private JFrame frame;
    private CardLayout cardLayout = new CardLayout();
    private JPanel mainPanel;

    // Users map username -> User
    private Map<String, User> users = new HashMap<>();
    private User loggedInUser = null;

    // Flights separated into local and international
    private Map<String, Flight> localFlights = new LinkedHashMap<>();
    private Map<String, Flight> internationalFlights = new LinkedHashMap<>();

    private Map<String, FlightStatus> flightStatuses = new HashMap<>();

    // Bookings by PNR
    private Map<String, Booking> bookings = new HashMap<>();

    private Set<String> selectedSeats = new HashSet<>();
    private Map<String, JButton> seatButtons = new HashMap<>();

    // Temp booking info
    private Flight tempFlight = null;
    private Fare tempFare = null;
    private int tempPersons = 1;
    private double tempTotalPrice = 0.0;

    // Components for UI references to update
    private JComboBox<String> flightSearchCombo;
    private JComboBox<String> classComboBox;
```

```java
    private JComboBox<String> classComboBox;
    private JLabel priceLabel;
    private JSpinner personSpinner;
    private JLabel flightStatusLabel;
    private JLabel weatherLabel;
    private JTextArea chatbotArea;
    private JTextField chatbotInput;
    private JRadioButton radioLocal;
    private JRadioButton radioInternational;

    private JPanel seatPanel;

    private JLabel loyaltyPointsLabel;
    private DefaultListModel<String> bookingListModel;
    private JList<String> bookingList;

    private boolean isDarkMode = false; // Added for dark mode toggle

    private java.util.Timer uiTimer;

    private JLabel totalBookingsLabel;
    private JLabel revenueLabel;
    private JLabel occupancyLabel;
    private JLabel checkInCountLabel;
    private DefaultListModel<String> auditLogModel;
    private JList<String> auditLogList;

    private Random rand = new Random();

    // FlightStatus enum
    private enum FlightStatus { ON_TIME, DELAYED, CANCELED }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new AirplaneReservationSystem().initUI());
    }

    private void initUI() {
        frame = new JFrame("✈ Airplane Reservation System");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(1000, 800); // Increased height slightly for chatbot and admin log
```

```java
        frame.setSize(1000, 800); // Increased height slightly for chatbot and admin log
        frame.setLocationRelativeTo(null);
        // frame.setResizable(false); // Allowing resize might be better for different screen sizes

        mainPanel = new JPanel(cardLayout);
        auditLogModel = new DefaultListModel<>(); // Initialize auditLogModel

        initFlights();
        initUsers();
        addAuditLog("System Initialized.");

        mainPanel.add(createLoginPanel(), "Login");
        mainPanel.add(createRegisterPanel(), "Register");
        mainPanel.add(createCustomerPanel(), "CustomerPanel");
        mainPanel.add(createSeatSelectionPanel(), "SeatSelection");
        mainPanel.add(createPaymentPanel(), "Payment");
        mainPanel.add(createBookingListPanel(), "Bookings");
        mainPanel.add(createAdminPanel(), "AdminPanel");
        mainPanel.add(createChatbotPanel(), "Chatbot"); // This was the missing link

        frame.add(mainPanel);
        frame.setVisible(true);

        cardLayout.show(mainPanel, "Login");
        startUITimer();
    }

    private void startUITimer() {
        uiTimer = new java.util.Timer();
        uiTimer.scheduleAtFixedRate(new TimerTask() {
            @Override
            public void run() {
                SwingUtilities.invokeLater(() -> {
                    updateFlightStatuses();
                    updateAdminDashboard();
                    animateSeatSelection(); // Added call
                });
            }
        }, 0, 5000); // Update every 5 seconds
    }
```

```java
    }

    // Placeholder for seat animation
    private void animateSeatSelection() {
        if (seatPanel != null && seatPanel.isVisible()) {
            for (JButton btn : seatButtons.values()) {
                if (btn.isEnabled() && !selectedSeats.contains(btn.getText())) { // Animate only available, non-selected seats
                    // Simple animation: slightly change color or size
                    // This is a very basic example. More complex animations would require more effort.
                    Color originalColor = new Color(144, 238, 144); // Light Green
                    Color animatedColor = new Color(173, 255, 173); // Lighter Green
                    if (btn.getBackground().equals(originalColor)) {
                        // btn.setBackground(animatedColor); // This might be too flashy if toggling quickly
                    } else {
                        // btn.setBackground(originalColor);
                    }
                }
            }
        }
    }


    private void stopUITimer() {
        if (uiTimer != null) {
            uiTimer.cancel();
            uiTimer = null;
        }
    }

    private void initFlights() {
        // Local flights
        localFlights.put("Manila to Cebu (MNL-CEB)", new Flight("Manila to Cebu (MNL-CEB)", 3200, 150));
        localFlights.put("Manila to Davao (MNL-DVO)", new Flight("Manila to Davao (MNL-DVO)", 3500, 160));
        localFlights.put("Manila to Iloilo (MNL-ILO)", new Flight("Manila to Iloilo (MNL-ILO)", 3100, 140));
        localFlights.put("Manila to Bacolod (MNL-BCD)", new Flight("Manila to Bacolod (MNL-BCD)", 3300, 130));
        localFlights.put("Manila to Bohol (MNL-TAG)", new Flight("Manila to Bohol (MNL-TAG)", 3600, 150));
        localFlights.put("Manila to Zamboanga (MNL-ZAM)", new Flight("Manila to Zamboanga (MNL-ZAM)", 3500, 140));
        localFlights.put("Cebu to Manila (CEB-MNL)", new Flight("Cebu to Manila (CEB-MNL)", 3150, 150));
        localFlights.put("Davao to Manila (DVO-MNL)", new Flight("Davao to Manila (DVO-MNL)", 3450, 160));
```

```java
        // International flights
        internationalFlights.put("Manila to Tokyo (MNL-NRT)", new Flight("Manila to Tokyo (MNL-NRT)", 12000, 200));
        internationalFlights.put("Manila to Seoul (MNL-ICN)", new Flight("Manila to Seoul (MNL-ICN)", 11000, 180));
        internationalFlights.put("Manila to Singapore (MNL-SIN)", new Flight("Manila to Singapore (MNL-SIN)", 10000, 180));
        internationalFlights.put("Manila to Dubai (MNL-DXB)", new Flight("Manila to Dubai (MNL-DXB)", 18000, 200));
        internationalFlights.put("Manila to Paris (MNL-CDG)", new Flight("Manila to Paris (MNL-CDG)", 19000, 210));
        internationalFlights.put("Manila to New York (MNL-JFK)", new Flight("Manila to New York (MNL-JFK)", 20000, 220));
        internationalFlights.put("Tokyo to Manila (NRT-MNL)", new Flight("Tokyo to Manila (NRT-MNL)", 11800, 200));
        internationalFlights.put("Seoul to Manila (ICN-MNL)", new Flight("Seoul to Manila (ICN-MNL)", 10800, 180));

        addAuditLog("Flight data initialized.");
    }

    private void initUsers() {
        users.put("admin", new User("admin", "admin123", Role.ADMIN)); // Changed admin password for slight security
        users.put("user1", new User("user1", "pass1", Role.CUSTOMER));
        users.put("john", new User("john", "doe", Role.CUSTOMER));
        addAuditLog("User data initialized.");
    }

    private JPanel createLoginPanel() {
        JPanel panel = new JPanel(new GridBagLayout());
        panel.setBackground(new Color(230, 240, 250)); // Light blue-ish
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(10, 15, 10, 15); // Increased insets
        gbc.fill = GridBagConstraints.HORIZONTAL;

        JLabel title = new JLabel("✈ Airplane Reservation System", JLabel.CENTER);
        title.setFont(new Font("Segoe UI", Font.BOLD, 32)); // Larger font
        title.setForeground(new Color(33, 89, 166)); // Darker blue
        gbc.gridx = 0; gbc.gridy = 0; gbc.gridwidth = 2;
        gbc.insets = new Insets(30, 10, 40, 10); // More top/bottom margin for title
        panel.add(title, gbc);

        gbc.insets = new Insets(10, 15, 10, 15); // Reset insets for other components

        JLabel userLbl = new JLabel("Username:");
        userLbl.setFont(new Font("Segoe UI", Font.PLAIN, 16));
        gbc.gridwidth = 1;
```

```java
        gbc.gridwidth = 1;
        gbc.gridx = 0; gbc.gridy = 1;
        panel.add(userLbl, gbc);

        JTextField userField = new JTextField(20);
        userField.setFont(new Font("Segoe UI", Font.PLAIN, 16));
        gbc.gridx = 1; gbc.gridy = 1;
        panel.add(userField, gbc);

        JLabel passLbl = new JLabel("Password:");
        passLbl.setFont(new Font("Segoe UI", Font.PLAIN, 16));
        gbc.gridx = 0; gbc.gridy = 2;
        panel.add(passLbl, gbc);

        JPasswordField passField = new JPasswordField(20);
        passField.setFont(new Font("Segoe UI", Font.PLAIN, 16));
        gbc.gridx = 1; gbc.gridy = 2;
        panel.add(passField, gbc);

        JPanel btnPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 25, 15)); // Increased gaps
        btnPanel.setBackground(panel.getBackground()); // Match panel background

        JButton loginBtn = createStyledButton("Login", new Color(33, 89, 166), Color.WHITE);
        JButton toRegisterBtn = createStyledButton("Register", new Color(100, 149, 237), Color.WHITE);

        btnPanel.add(loginBtn);
        btnPanel.add(toRegisterBtn);

        gbc.gridx = 0; gbc.gridy = 3; gbc.gridwidth = 2;
        gbc.insets = new Insets(20, 10, 20, 10); // More margin for button panel
        panel.add(btnPanel, gbc);

        // Login action
        ActionListener loginAction = e -> {
            String user = userField.getText().trim();
            String pass = new String(passField.getPassword());

            if (user.isEmpty() || pass.isEmpty()) {
                JOptionPane.showMessageDialog(frame, "Please enter username and password.", "Input Error", JOptionPane.WARNING_MESSAGE);
                return;
```

```java
                return;
            }
            if (users.containsKey(user) && users.get(user).checkPassword(pass)) {
                loggedInUser = users.get(user);
                addAuditLog("User logged in: " + loggedInUser.getUsername());
                userField.setText("");
                passField.setText("");
                if (loggedInUser.getRole() == Role.ADMIN) {
                    updateAdminDashboard(); // Update dashboard on admin login
                    cardLayout.show(mainPanel, "AdminPanel");
                } else {
                    updateCustomerPanelWelcome(); // Update welcome message
                    updateLoyaltyPoints();
                    cardLayout.show(mainPanel, "CustomerPanel");
                }
            } else {
                addAuditLog("Failed login attempt for username: " + user);
                JOptionPane.showMessageDialog(frame, "Invalid username or password.", "Authentication Failed", JOptionPane.ERROR_MESSAGE);
            }
        };

        loginBtn.addActionListener(loginAction);
        passField.addActionListener(loginAction); // Allow login on Enter in password field
        userField.addActionListener(loginAction); // Allow login on Enter in user field


        toRegisterBtn.addActionListener(e -> {
            userField.setText("");
            passField.setText("");
            cardLayout.show(mainPanel, "Register");
        });

        return panel;
    }

    private JLabel customerWelcomeLabel; // Declare at class level to update it

    private void updateCustomerPanelWelcome() {
        if (customerWelcomeLabel != null && loggedInUser != null) {
            customerWelcomeLabel.setText("Welcome, " + loggedInUser.getUsername() + "!");
```

```java
            customerWelcomeLabel.setText("Welcome, " + loggedInUser.getUsername() + "!");
        }
    }


    private JPanel createRegisterPanel() {
        JPanel panel = new JPanel(new GridBagLayout());
        panel.setBackground(new Color(250, 240, 230)); // Light orange-ish
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(10, 15, 10, 15);
        gbc.fill = GridBagConstraints.HORIZONTAL;

        JLabel title = new JLabel("Create New Account", JLabel.CENTER);
        title.setFont(new Font("Segoe UI", Font.BOLD, 28));
        title.setForeground(new Color(165, 110, 60)); // Darker orange
        gbc.gridx = 0; gbc.gridy = 0; gbc.gridwidth = 2;
        gbc.insets = new Insets(25, 10, 30, 10);
        panel.add(title, gbc);

        gbc.insets = new Insets(10, 15, 10, 15);

        Font labelFont = new Font("Segoe UI", Font.PLAIN, 16);
        Font fieldFont = new Font("Segoe UI", Font.PLAIN, 16);

        JLabel userLbl = new JLabel("Username:");
        userLbl.setFont(labelFont);
        gbc.gridwidth = 1;
        gbc.gridx = 0; gbc.gridy = 1;
        panel.add(userLbl, gbc);

        JTextField userField = new JTextField(20);
        userField.setFont(fieldFont);
        gbc.gridx = 1; gbc.gridy = 1;
        panel.add(userField, gbc);

        JLabel passLbl = new JLabel("Password:");
        passLbl.setFont(labelFont);
        gbc.gridx = 0; gbc.gridy = 2;
        panel.add(passLbl, gbc);
```

```java
        gbc.gridx = 1; gbc.gridy = 2;
        panel.add(passField, gbc);

        JLabel passConfirmLbl = new JLabel("Confirm Password:");
        passConfirmLbl.setFont(labelFont);
        gbc.gridx = 0; gbc.gridy = 3;
        panel.add(passConfirmLbl, gbc);

        JPasswordField passConfirmField = new JPasswordField(20);
        passConfirmField.setFont(fieldFont);
        gbc.gridx = 1; gbc.gridy = 3;
        panel.add(passConfirmField, gbc);

        JLabel roleLbl = new JLabel("Role:");
        roleLbl.setFont(labelFont);
        gbc.gridx = 0; gbc.gridy = 4;
        panel.add(roleLbl, gbc);

        JTextField roleField = new JTextField("Customer");
        roleField.setFont(fieldFont);
        roleField.setEditable(false);
        roleField.setBackground(Color.LIGHT_GRAY); // Indicate non-editable
        gbc.gridx = 1; gbc.gridy = 4;
        panel.add(roleField, gbc);

        JPanel btnPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 25, 15));
        btnPanel.setBackground(panel.getBackground());

        JButton registerBtn = createStyledButton("Register", new Color(165, 110, 60), Color.WHITE);
        JButton backBtn = createStyledButton("Back to Login", new Color(205, 133, 63), Color.WHITE);

        btnPanel.add(registerBtn);
        btnPanel.add(backBtn);

        gbc.gridx = 0; gbc.gridy = 5; gbc.gridwidth = 2;
        gbc.insets = new Insets(20, 10, 20, 10);
        panel.add(btnPanel, gbc);

        registerBtn.addActionListener(e -> {
            String user = userField.getText().trim();
```

```java
            String pass = new String(passField.getPassword());
            String passConfirm = new String(passConfirmField.getPassword());

            if (user.isEmpty() || pass.isEmpty() || passConfirm.isEmpty()) {
                JOptionPane.showMessageDialog(frame, "Please fill in all fields.", "Input Error", JOptionPane.WARNING_MESSAGE);
                return;
            }
            if (!user.matches("^[a-zA-Z0-9_]{3,15}$")) { // Username length 3-15
                JOptionPane.showMessageDialog(frame, "Username must be alphanumeric (3-15 characters).", "Input Error", JOptionPane.WARNING_MESSAGE);
                return;
            }
            if (users.containsKey(user)) {
                JOptionPane.showMessageDialog(frame, "Username already exists. Please choose another.", "Registration Error", JOptionPane.ERROR_MESSAGE);
                return;
            }
            if (!pass.equals(passConfirm)) {
                JOptionPane.showMessageDialog(frame, "Passwords do not match.", "Registration Error", JOptionPane.ERROR_MESSAGE);
                return;
            }
            if (pass.length() < 6) { // Increased password min length
                JOptionPane.showMessageDialog(frame, "Password must be at least 6 characters.", "Registration Error", JOptionPane.ERROR_MESSAGE);
                return;
            }

            User newUser = new User(user, pass, Role.CUSTOMER);
            users.put(user, newUser);
            addAuditLog("New user registered: " + user);
            JOptionPane.showMessageDialog(frame, "Registered successfully! Please login.", "Registration Complete", JOptionPane.INFORMATION_MESSAGE);
            userField.setText("");
            passField.setText("");
            passConfirmField.setText("");
            cardLayout.show(mainPanel, "Login");
        });

        backBtn.addActionListener(e -> {
            userField.setText("");
            passField.setText("");
            passConfirmField.setText("");
            cardLayout.show(mainPanel, "Login");
        });
```

```java
        return panel;
    }

    private JPanel createCustomerPanel() {
        JPanel panel = new JPanel(new BorderLayout(0, 10)); // Added vertical gap
        panel.setBorder(new EmptyBorder(10, 10, 10, 10)); // Overall padding

        // Top panel with title and user info + buttons
        JPanel topPanel = new JPanel(new BorderLayout());
        topPanel.setBackground(new Color(224, 239, 255)); // Light blue
        topPanel.setBorder(new EmptyBorder(10, 15, 10, 15));

        JLabel title = new JLabel("Flight Booking System");
        title.setFont(new Font("Segoe UI", Font.BOLD, 28));
        title.setForeground(new Color(10, 70, 140)); // Dark blue
        topPanel.add(title, BorderLayout.WEST);

        JPanel userActionsPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT, 10, 0)); // Reduced gap
        userActionsPanel.setOpaque(false); // Transparent background

        customerWelcomeLabel = new JLabel("Welcome!"); // Initial text
        customerWelcomeLabel.setFont(new Font("Segoe UI", Font.PLAIN, 16));
        customerWelcomeLabel.setForeground(new Color(50, 50, 50));

        JButton bookingsBtn = createStyledButton("My Bookings", new Color(0, 100, 70), Color.WHITE);
        JButton chatbotBtn = createStyledButton("Ask AI Assistant", new Color(33, 89, 166), Color.WHITE); // Renamed
        JButton darkModeToggleBtn = createStyledButton("Toggle Dark Mode", new Color(80, 80, 80), Color.WHITE);
        JButton logoutBtn = createStyledButton("Logout", new Color(200, 30, 30), Color.WHITE);

        userActionsPanel.add(customerWelcomeLabel);
        userActionsPanel.add(bookingsBtn);
        userActionsPanel.add(chatbotBtn);
        userActionsPanel.add(darkModeToggleBtn);
        userActionsPanel.add(logoutBtn);
        topPanel.add(userActionsPanel, BorderLayout.EAST);

        panel.add(topPanel, BorderLayout.NORTH);

        // Center panel for flight selection
        JPanel centerPanel = new JPanel(new GridBagLayout());
```

```java
        centerPanel.setBorder(BorderFactory.createTitledBorder(
                BorderFactory.createEtchedBorder(), "Book Your Flight",
                TitledBorder.CENTER, TitledBorder.TOP,
                new Font("Segoe UI", Font.BOLD, 20), new Color(10, 70, 140)));
        centerPanel.setBackground(new Color(245, 250, 255)); // Very light blue
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(8, 12, 8, 12); // Adjusted insets
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.weightx = 1.0; // Allow components to expand horizontally

        // Flight Type Radio Buttons
        JLabel lblSelectType = new JLabel("Select Flight Type:");
        lblSelectType.setFont(new Font("Segoe UI", Font.BOLD, 16));
        gbc.gridx = 0; gbc.gridy = 0; gbc.gridwidth = 1; gbc.anchor = GridBagConstraints.WEST;
        centerPanel.add(lblSelectType, gbc);

        radioLocal = new JRadioButton("Local Flights");
        radioLocal.setFont(new Font("Segoe UI", Font.PLAIN, 15));
        radioLocal.setSelected(true);
        radioLocal.setOpaque(false);

        radioInternational = new JRadioButton("International Flights");
        radioInternational.setFont(new Font("Segoe UI", Font.PLAIN, 15));
        radioInternational.setOpaque(false);

        ButtonGroup group = new ButtonGroup();
        group.add(radioLocal);
        group.add(radioInternational);

        JPanel radioPanel = new JPanel(new FlowLayout(FlowLayout.LEFT, 5, 0));
        radioPanel.setOpaque(false);
        radioPanel.add(radioLocal);
        radioPanel.add(radioInternational);
        gbc.gridx = 1; gbc.gridy = 0; gbc.gridwidth = 2; gbc.anchor = GridBagConstraints.WEST;
        centerPanel.add(radioPanel, gbc);

        // Available Flights
        JLabel flightLbl = new JLabel("Available Flights:");
        flightLbl.setFont(new Font("Segoe UI", Font.BOLD, 16));
        gbc.gridx = 0; gbc.gridy = 1; gbc.gridwidth = 1; gbc.anchor = GridBagConstraints.WEST;
```

```java
        gbc.gridx = 0; gbc.gridy = 1; gbc.gridwidth = 1; gbc.anchor = GridBagConstraints.WEST;
        centerPanel.add(flightLbl, gbc);

        flightSearchCombo = new JComboBox<>();
        flightSearchCombo.setFont(new Font("Segoe UI", Font.PLAIN, 15));
        gbc.gridx = 1; gbc.gridy = 1; gbc.gridwidth = 2;
        centerPanel.add(flightSearchCombo, gbc);

        // Class Selection
        JLabel classLbl = new JLabel("Class:");
        classLbl.setFont(new Font("Segoe UI", Font.BOLD, 16));
        gbc.gridx = 0; gbc.gridy = 2; gbc.gridwidth = 1; gbc.anchor = GridBagConstraints.WEST;
        centerPanel.add(classLbl, gbc);

        String[] classes = { "Economy (x1.0)", "Economy Plus (x1.5)", "Business Class (x2.0)", "First Class (x3.0)" }; // Added First Class
        classComboBox = new JComboBox<>(classes);
        classComboBox.setFont(new Font("Segoe UI", Font.PLAIN, 15));
        gbc.gridx = 1; gbc.gridy = 2; gbc.gridwidth = 2;
        centerPanel.add(classComboBox, gbc);

        // Number of Persons
        JLabel personsLbl = new JLabel("Number of Persons:");
        personsLbl.setFont(new Font("Segoe UI", Font.BOLD, 16));
        gbc.gridx = 0; gbc.gridy = 3; gbc.gridwidth = 1; gbc.anchor = GridBagConstraints.WEST;
        centerPanel.add(personsLbl, gbc);

        SpinnerNumberModel spinnerModel = new SpinnerNumberModel(1, 1, 9, 1); // Min 1, Max 9
        personSpinner = new JSpinner(spinnerModel);
        personSpinner.setFont(new Font("Segoe UI", Font.PLAIN, 15));
        JFormattedTextField spinnerText = ((JSpinner.DefaultEditor) personSpinner.getEditor()).getTextField();
        spinnerText.setEditable(false); // Make spinner text field non-editable directly
        spinnerText.setBackground(Color.WHITE); // Ensure background is white
        gbc.gridx = 1; gbc.gridy = 3; gbc.gridwidth = 1; // Spinner takes less space
        gbc.anchor = GridBagConstraints.WEST;
        centerPanel.add(personSpinner, gbc);

        // Price Label
        priceLabel = new JLabel("Total Price: ₱0.00");
        priceLabel.setFont(new Font("Segoe UI", Font.BOLD, 18));
        priceLabel.setForeground(new Color(10, 130, 30)); // Green for price
        gbc.gridx = 0; gbc.gridy = 4; gbc.gridwidth = 3; gbc.anchor = GridBagConstraints.CENTER;
```

```java
        centerPanel.add(priceLabel, gbc);
        gbc.insets = new Insets(8, 12, 8, 12); // Reset insets

        // Buttons: Calculate and Book
        JPanel actionButtonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 20, 0));
        actionButtonPanel.setOpaque(false);
        JButton calcPriceBtn = createStyledButton("Calculate Price", new Color(10, 70, 140), Color.WHITE);
        JButton bookBtn = createStyledButton("Proceed to Seat Selection", new Color(10, 125, 70), Color.WHITE);
        actionButtonPanel.add(calcPriceBtn);
        actionButtonPanel.add(bookBtn);
        gbc.gridx = 0; gbc.gridy = 5; gbc.gridwidth = 3; gbc.anchor = GridBagConstraints.CENTER;
        centerPanel.add(actionButtonPanel, gbc);

        // Flight Status and Weather
        flightStatusLabel = new JLabel("Flight Status: Select a flight");
        flightStatusLabel.setFont(new Font("Segoe UI", Font.PLAIN, 14));
        gbc.gridx = 0; gbc.gridy = 6; gbc.gridwidth = 3; gbc.anchor = GridBagConstraints.CENTER;
        centerPanel.add(flightStatusLabel, gbc);

        weatherLabel = new JLabel("Weather: -");
        weatherLabel.setFont(new Font("Segoe UI", Font.ITALIC, 13));
        gbc.gridx = 0; gbc.gridy = 7; gbc.gridwidth = 3; gbc.anchor = GridBagConstraints.CENTER;
        centerPanel.add(weatherLabel, gbc);

        // Loyalty Points
        loyaltyPointsLabel = new JLabel("Loyalty Points: 0");
        loyaltyPointsLabel.setFont(new Font("Segoe UI", Font.BOLD, 15));
        loyaltyPointsLabel.setForeground(new Color(218, 165, 32)); // Gold color for points
        gbc.gridx = 0; gbc.gridy = 8; gbc.gridwidth = 3; gbc.anchor = GridBagConstraints.CENTER;
        gbc.insets = new Insets(15, 12, 10, 12);
        centerPanel.add(loyaltyPointsLabel, gbc);

        panel.add(centerPanel, BorderLayout.CENTER);

        // Event Listeners
        populateFlights(true); // Initial population

        ItemListener flightTypeListener = e -> populateFlights(radioLocal.isSelected());
        radioLocal.addItemListener(flightTypeListener);
        radioInternational.addItemListener(flightTypeListener);
```

```java
        // Update price and status when flight, class, or persons change
        ItemListener updateListener = e -> { if (e.getStateChange() == ItemEvent.SELECTED) calculatePrice(); };
        flightSearchCombo.addItemListener(updateListener);
        classComboBox.addItemListener(updateListener);
        personSpinner.addChangeListener(e -> calculatePrice());


        calcPriceBtn.addActionListener(e -> calculatePrice());
        bookBtn.addActionListener(e -> proceedToSeatSelection());

        bookingsBtn.addActionListener(e -> {
            loadUserBookings();
            cardLayout.show(mainPanel, "Bookings");
        });
        chatbotBtn.addActionListener(e -> cardLayout.show(mainPanel, "Chatbot"));
        logoutBtn.addActionListener(e -> logoutUser());
        darkModeToggleBtn.addActionListener(e -> toggleDarkMode());

        // Update welcome message and loyalty points when panel is shown
        panel.addComponentListener(new ComponentAdapter() {
            @Override
            public void componentShown(ComponentEvent e) {
                updateCustomerPanelWelcome();
                updateLoyaltyPoints();
                populateFlights(radioLocal.isSelected()); // Refresh flight list
            }
        });

        return panel;
    }

    private void populateFlights(boolean local) {
        flightSearchCombo.removeAllItems();
        Map<String, Flight> sourceMap = local ? localFlights : internationalFlights;

        if (sourceMap.isEmpty()) {
            flightSearchCombo.addItem("No flights available for this type.");
            flightSearchCombo.setEnabled(false);
        } else {
            sourceMap.keySet().stream().sorted().forEach(flightSearchCombo::addItem); // Sort flight names
```

```java
        // Reset dependent fields if no flight is selected or available
        if (flightSearchCombo.getItemCount() == 0 || !flightSearchCombo.isEnabled()) {
            priceLabel.setText("Total Price: ₱0.00");
            flightStatusLabel.setText("Flight Status: -");
            weatherLabel.setText("Weather: -");
            tempFlight = null;
        } else {
            flightSearchCombo.setSelectedIndex(0); // Select first flight by default
            calculatePrice(); // Calculate price for the default selected flight
        }
    }


    private void calculatePrice() {
        String selectedFlightName = (String) flightSearchCombo.getSelectedItem();
        String selectedClassStr = (String) classComboBox.getSelectedItem();
        int persons = (int) personSpinner.getValue();

        if (selectedFlightName == null || selectedFlightName.startsWith("No flights available") || selectedClassStr == null) {
            priceLabel.setText("Total Price: ₱0.00");
            flightStatusLabel.setText("Flight Status: Please select a flight.");
            weatherLabel.setText("Weather: -");
            tempFlight = null; // Reset tempFlight if selection is invalid
            return;
        }

        Flight flight = (radioLocal.isSelected() ? localFlights : internationalFlights).get(selectedFlightName);

        if (flight == null) {
            priceLabel.setText("Error: Flight data not found.");
            tempFlight = null;
            return;
        }

        double multiplier = 1.0;
        if (selectedClassStr.contains("Economy Plus (x1.5)")) multiplier = 1.5;
        else if (selectedClassStr.contains("Business Class (x2.0)")) multiplier = 2.0;
        else if (selectedClassStr.contains("First Class (x3.0)")) multiplier = 3.0;
```

```java
        double totalPrice = flight.baseFare * multiplier * persons;
        priceLabel.setText(String.format("Total Price: ₱%,.2f", totalPrice));

        tempFlight = flight;
        tempFare = new Fare(selectedClassStr.substring(0, selectedClassStr.indexOf(" (x")), multiplier); // Get class name only
        tempPersons = persons;
        tempTotalPrice = totalPrice;

        updateFlightStatusDisplay(flight.route); // Pass route for specific status
        updateWeatherInfo(flight.route);
    }

    private void updateFlightStatusDisplay(String flightRoute) {
        if (flightRoute == null || flightStatusLabel == null) return;

        FlightStatus status = flightStatuses.getOrDefault(flightRoute, FlightStatus.ON_TIME); // Get status for specific flight
        String statusText = "Flight Status (" + flightRoute + "): ";
        Color statusColor = Color.BLACK;

        switch (status) {
            case ON_TIME:
                statusText += "On Time";
                statusColor = new Color(0, 128, 0); // Green
                break;
            case DELAYED:
                statusText += "Delayed";
                statusColor = new Color(255, 165, 0); // Orange
                break;
            case CANCELED:
                statusText += "Canceled";
                statusColor = Color.RED;
                break;
        }
        flightStatusLabel.setText(statusText);
        flightStatusLabel.setForeground(statusColor);
    }


    private void updateFlightStatuses() {
        // Initialize statuses if not present
```

```java
        // Initialize statuses if not present
        localFlights.keySet().forEach(f -> flightStatuses.putIfAbsent(f, FlightStatus.ON_TIME));
        internationalFlights.keySet().forEach(f -> flightStatuses.putIfAbsent(f, FlightStatus.ON_TIME));

        // Randomly change a few statuses for simulation
        List<String> allFlightKeys = new ArrayList<>(flightStatuses.keySet());
        if (allFlightKeys.isEmpty()) return;

        int changes = rand.nextInt(3) + 1; // Change 1 to 3 statuses
        for (int i = 0; i < changes; i++) {
            String randomFlightKey = allFlightKeys.get(rand.nextInt(allFlightKeys.size()));
            int r = rand.nextInt(100);
            FlightStatus newStatus = FlightStatus.ON_TIME;
            if (r < 5) newStatus = FlightStatus.CANCELED;       // 5% chance canceled
            else if (r < 20) newStatus = FlightStatus.DELAYED;  // 15% chance delayed
            flightStatuses.put(randomFlightKey, newStatus);
        }

        // If a flight is currently selected in the customer panel, update its displayed status
        if (tempFlight != null && flightStatusLabel != null && flightStatusLabel.isVisible()) {
            updateFlightStatusDisplay(tempFlight.route);
        }
    }


    private void updateWeatherInfo(String route) {
        if (route == null || weatherLabel == null) return;
        String[] weathers = {"☀ Sunny", "☁ Cloudy", "🌧 Light Rain", "🌧 Rainy", "⛈ Stormy", "🌫 Foggy", "💨 Windy"};
        String depWeather = weathers[rand.nextInt(weathers.length)];
        String destWeather = weathers[rand.nextInt(weathers.length)];
        weatherLabel.setText("Weather for " + route.split(" ")[0] + " → " + route.split(" ")[2] + ": Departure " + depWeather + ", Arrival " + destWeather);
    }

    private void proceedToSeatSelection() {
        if (loggedInUser == null) {
            JOptionPane.showMessageDialog(frame, "Please login first to book a flight.", "Not Logged In", JOptionPane.WARNING_MESSAGE);
            return;
        }
        if (tempFlight == null || tempFare == null || tempTotalPrice <= 0) {
            JOptionPane.showMessageDialog(frame, "Please select a flight and calculate the price before proceeding.", "Booking Error", JOptionPane.WARNING_MESSAGE
```

```java
        FlightStatus currentStatus = flightStatuses.getOrDefault(tempFlight.route, FlightStatus.ON_TIME);
        if (currentStatus == FlightStatus.CANCELED) {
            JOptionPane.showMessageDialog(frame, "This flight (" + tempFlight.route + ") is currently CANCELED and cannot be booked.", "Flight Canceled", JOption
            return;
        }

        selectedSeats.clear();
        buildSeatMap(tempFlight); // Build or rebuild seat map
        // Update the info label in seat selection panel
        if (seatInfoLabel != null) {
            seatInfoLabel.setText("Select exactly " + tempPersons + " seat(s). Green: Available, Orange: Selected, Red: Occupied.");
        }
        cardLayout.show(mainPanel, "SeatSelection");
    }

    private JLabel seatInfoLabel; // Declare at class level

    private JPanel createSeatSelectionPanel() {
        JPanel panel = new JPanel(new BorderLayout(10,10));
        panel.setBorder(new EmptyBorder(15,15,15,15));
        panel.setBackground(new Color(240, 248, 255)); // Alice Blue

        JLabel title = new JLabel("Select Your Seat(s)", JLabel.CENTER);
        title.setFont(new Font("Segoe UI", Font.BOLD, 26));
        title.setForeground(new Color(10, 70, 140));
        panel.add(title, BorderLayout.NORTH);

        seatPanel = new JPanel(); // Using FlowLayout for dynamic sizing of rows
        seatPanel.setLayout(new BoxLayout(seatPanel, BoxLayout.Y_AXIS)); // Vertical layout for rows of seats
        seatPanel.setBackground(new Color(220, 230, 240)); // Light greyish blue for seat area background

        JScrollPane scrollPane = new JScrollPane(seatPanel);
        scrollPane.setBorder(BorderFactory.createEtchedBorder());
        scrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);
        scrollPane.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
        panel.add(scrollPane, BorderLayout.CENTER);

        seatInfoLabel = new JLabel("Select seats. Green: Available, Orange: Selected, Red: Occupied.", JLabel.CENTER);
        seatInfoLabel.setFont(new Font("Segoe UI", Font.ITALIC, 14));
        seatInfoLabel.setBorder(new EmptyBorder(8, 0, 8, 0));
```

```java
        JPanel bottomPanel = new JPanel(new BorderLayout(10,5));
        bottomPanel.setOpaque(false);

        seatInfoLabel = new JLabel("Select exactly " + tempPersons + " seat(s).", JLabel.CENTER); // Initial text
        seatInfoLabel.setFont(new Font("Segoe UI", Font.ITALIC, 14));
        seatInfoLabel.setForeground(new Color(80, 80, 80));
        bottomPanel.add(seatInfoLabel, BorderLayout.NORTH);


        JPanel btnPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 30, 10));
        btnPanel.setOpaque(false);

        JButton confirmBtn = createStyledButton("Confirm Seat Selection", new Color(10, 130, 30), Color.WHITE);
        JButton cancelBtn = createStyledButton("Cancel & Go Back", new Color(220, 20, 20), Color.WHITE);

        btnPanel.add(confirmBtn);
        btnPanel.add(cancelBtn);
        bottomPanel.add(btnPanel, BorderLayout.CENTER);

        panel.add(bottomPanel, BorderLayout.SOUTH);

        confirmBtn.addActionListener(e -> {
            if (selectedSeats.size() != tempPersons) {
                JOptionPane.showMessageDialog(frame, "You must select exactly " + tempPersons + " seat(s). You have selected " + selectedSeats.size() + ".", "Seat
                return;
            }
            // Double check availability before proceeding (though UI should prevent selecting reserved)
            if (!tempFlight.areSeatsAvailable(selectedSeats)) {
                JOptionPane.showMessageDialog(frame, "Some selected seats are no longer available. Please re-select.", "Seat Availability Error", JOptionPane.ERRO
                buildSeatMap(tempFlight); // Rebuild to show current status
                return;
            }
            // No need to reserve here, will be done upon payment.
            // tempFlight.reserveSeats(selectedSeats); // This should happen after payment.
            updatePaymentPanelInfo();
            cardLayout.show(mainPanel, "Payment");
        });

        cancelBtn.addActionListener(e -> {
            selectedSeats.clear(); // Clear selections
            // No need to release here as they weren't formally reserved yet
```

```java

        // Update seat info label when panel is shown
        panel.addComponentListener(new ComponentAdapter() {
            @Override
            public void componentShown(ComponentEvent e) {
                if (seatInfoLabel != null) {
                    seatInfoLabel.setText("Select exactly " + tempPersons + " seat(s). Green: Available, Orange: Selected, Red: Occupied.");
                }
            }
        });

        return panel;
    }

    private void resetSeatButtonsVisuals() { // Renamed for clarity
        seatButtons.forEach((id, btn) -> {
            if (tempFlight != null && tempFlight.isSeatReserved(id)) {
                btn.setBackground(new Color(220, 20, 20)); // Red for occupied
                btn.setEnabled(false);
            } else {
                btn.setBackground(new Color(144, 238, 144)); // Green for available
                btn.setEnabled(true);
            }
        });
        selectedSeats.clear(); // Clear the temporary selection set
    }

    private void buildSeatMap(Flight flight) {
        seatPanel.removeAll(); // Clear previous seat map
        seatButtons.clear();   // Clear button references

        int totalSeats = flight.totalSeats;
        int seatsPerRow = 6; // Standard 3-3 configuration (A-F)
        int aisleAfter = 3;  // Aisle after C
        int numRows = (int) Math.ceil((double) totalSeats / seatsPerRow);

        int seatWidth = 55;
        int seatHeight = 40;
        int hGap = 8; // Horizontal gap
        int vGap = 8; // Vertical gap
        int aisleGap = 20; // Wider gap for aisle
```

```java
                JButton seatBtn = new JButton(seatId);
                seatBtn.setFont(seatFont);
                seatBtn.setPreferredSize(new Dimension(seatWidth, seatHeight));
                seatBtn.setFocusPainted(false);
                seatBtn.setMargin(new Insets(2,2,2,2));

                if (flight.isSeatReserved(seatId)) {
                    seatBtn.setBackground(new Color(220, 20, 20)); // Red - Occupied
                    seatBtn.setForeground(Color.WHITE);
                    seatBtn.setToolTipText("Occupied");
                    seatBtn.setEnabled(false);
                } else {
                    seatBtn.setBackground(new Color(144, 238, 144)); // Green - Available
                    seatBtn.setForeground(Color.BLACK);
                    seatBtn.setToolTipText("Available");
                    seatBtn.setEnabled(true);
                }

                seatBtn.addActionListener(evt -> {
                    JButton clickedButton = (JButton) evt.getSource();
                    String currentSeatId = clickedButton.getText();

                    if (selectedSeats.contains(currentSeatId)) {
                        selectedSeats.remove(currentSeatId);
                        clickedButton.setBackground(new Color(144, 238, 144)); // Back to Green
                    } else {
                        if (selectedSeats.size() >= tempPersons) {
                            JOptionPane.showMessageDialog(frame,
                                    "You can select only " + tempPersons + " seat(s). Deselect a seat first if you want to change.",
                                    "Seat Selection Limit Reached", JOptionPane.WARNING_MESSAGE);
                            return;
                        }
                        selectedSeats.add(currentSeatId);
                        clickedButton.setBackground(Color.ORANGE); // Orange - Selected
                    }
                });
                rowPanel.add(seatBtn);
                seatButtons.put(seatId, seatBtn);
```

```java
        JPanel panel = new JPanel(new GridBagLayout());
        panel.setBorder(new EmptyBorder(20, 20, 20, 20));
        panel.setBackground(new Color(230, 245, 255)); // Light cyan-blue
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(12, 20, 12, 20);
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.weightx = 1.0;

        JLabel title = new JLabel("Complete Your Payment", JLabel.CENTER);
        title.setFont(new Font("Segoe UI", Font.BOLD, 28));
        title.setForeground(new Color(10, 70, 140));
        gbc.gridx = 0; gbc.gridy = 0; gbc.gridwidth = 2;
        gbc.insets = new Insets(10, 10, 30, 10);
        panel.add(title, gbc);

        gbc.insets = new Insets(12, 20, 12, 20); // Reset insets

        // Payment Method
        JLabel paymentLbl = new JLabel("Select Payment Method:");
        paymentLbl.setFont(new Font("Segoe UI", Font.PLAIN, 16));
        gbc.gridx = 0; gbc.gridy = 1; gbc.gridwidth = 1; gbc.anchor = GridBagConstraints.EAST;
        panel.add(paymentLbl, gbc);

        JComboBox<String> paymentCombo = new JComboBox<>(new String[] { "Credit/Debit Card", "GCash", "Maya", "PayPal", "Bank Transfer" });
        paymentCombo.setFont(new Font("Segoe UI", Font.PLAIN, 16));
        gbc.gridx = 1; gbc.gridy = 1; gbc.anchor = GridBagConstraints.WEST;
        panel.add(paymentCombo, gbc);

        // Amount to Pay
        paymentAmountLabel = new JLabel(String.format("Amount to Pay: ₱%,.2f", tempTotalPrice));
        paymentAmountLabel.setFont(new Font("Segoe UI", Font.BOLD, 22));
        paymentAmountLabel.setForeground(new Color(10, 130, 30)); // Green for amount
        gbc.gridx = 0; gbc.gridy = 2; gbc.gridwidth = 2; gbc.anchor = GridBagConstraints.CENTER;
        gbc.insets = new Insets(20, 10, 25, 10);
        panel.add(paymentAmountLabel, gbc);

        // Placeholder for card details (could be dynamically shown based on selection)
        // For simplicity, we are skipping actual card input fields.
```

```java
                Thread.sleep(2500); // Simulate 2.5 seconds processing
            } catch (InterruptedException ie) { Thread.currentThread().interrupt(); }
            SwingUtilities.invokeLater(() -> {
                processingDialog.dispose();
                completeBooking(paymentMethod);
            });
        }).start();

        processingDialog.setVisible(true);
    });

    backBtn.addActionListener(e -> cardLayout.show(mainPanel, "SeatSelection"));

    // Update payment amount when panel is shown
    panel.addComponentListener(new ComponentAdapter() {
        @Override
        public void componentShown(ComponentEvent e) {
            updatePaymentPanelInfo();
        }
    });

    return panel;
}

private void completeBooking(String paymentMethod) {
    // Actual reservation of seats happens here
    if (!tempFlight.areSeatsAvailable(selectedSeats)) {
        JOptionPane.showMessageDialog(frame, "Unfortunately, some selected seats became unavailable during payment. Please try selecting seats again.", "Sea
        buildSeatMap(tempFlight); // Rebuild to show current status
        cardLayout.show(mainPanel, "SeatSelection");
        return;
    }
    tempFlight.reserveSeats(selectedSeats); // Formally reserve the seats now

    String pnr = generatePNR();
    Date bookingDate = new Date(); // Current date and time for booking

    Booking newBooking = new Booking(pnr, loggedInUser, tempFlight, tempFare, tempTotalPrice, new HashSet<>(selectedSeats), false, bookingDate);
    bookings.put(pnr, newBooking);
    loggedInUser.addBooking(pnr);
```

```java
    private void updateLoyaltyPoints() {
        if (loggedInUser != null && loyaltyPointsLabel != null) {
            loyaltyPointsLabel.setText("Loyalty Points: " + loggedInUser.getLoyaltyPoints() + " ✨");
        }
    }

    private String generatePNR() {
        String chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
        StringBuilder pnr;
        do {
            pnr = new StringBuilder();
            for (int i = 0; i < 6; i++) { // 6-character PNR
                pnr.append(chars.charAt(rand.nextInt(chars.length())));
            }
        } while (bookings.containsKey(pnr.toString())); // Ensure PNR is unique
        return pnr.toString();
    }

    private JPanel createBookingListPanel() {
        JPanel panel = new JPanel(new BorderLayout(15, 15));
        panel.setBorder(new EmptyBorder(20, 25, 25, 25));
        panel.setBackground(new Color(230, 245, 240)); // Light green-blue

        JLabel title = new JLabel("My Bookings", JLabel.CENTER);
        title.setFont(new Font("Segoe UI", Font.BOLD, 26));
        title.setForeground(new Color(0, 100, 70)); // Dark green
        panel.add(title, BorderLayout.NORTH);

        bookingListModel = new DefaultListModel<>();
        bookingList = new JList<>(bookingListModel);
        bookingList.setFont(new Font("Monospaced", Font.PLAIN, 14)); // Monospaced for better alignment
        bookingList.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        bookingList.setCellRenderer(new BookingListRenderer()); // Custom renderer for better display

        JScrollPane scrollPane = new JScrollPane(bookingList);
        scrollPane.setBorder(BorderFactory.createEtchedBorder());
        panel.add(scrollPane, BorderLayout.CENTER);

        JPanel btnPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 20, 12)); // Adjusted gaps
        btnPanel.setBackground(panel.getBackground());
```

```java
            }

            // Check cancellation policy (e.g., cannot cancel if too close to flight)
            // For simplicity, allowing cancellation anytime for now.
            int confirm = JOptionPane.showConfirmDialog(frame,
                    "Are you sure you want to cancel booking PNR: " + pnr + " for flight " + bookingToCancel.flight.route + "?\n" +
                            "A cancellation fee might apply (not implemented).",
                    "Confirm Cancellation", JOptionPane.YES_NO_OPTION, JOptionPane.WARNING_MESSAGE);

            if (confirm != JOptionPane.YES_OPTION) return;

            bookings.remove(pnr);
            loggedInUser.removeBooking(pnr);
            bookingToCancel.flight.releaseSeats(bookingToCancel.getSelectedSeats()); // Release seats
            addAuditLog("Booking canceled. PNR: " + pnr + ", User: " + loggedInUser.getUsername());

            // Refund logic could be added here (e.g., partial refund to loyalty points or original method)
            // For now, just a message.
            loggedInUser.addLoyaltyPoints((int)(-bookingToCancel.totalPrice / 200)); // Example: Small penalty on points

            JOptionPane.showMessageDialog(frame, "Booking PNR: " + pnr + " canceled successfully.", "Cancellation Confirmed", JOptionPane.INFORMATION_MESSAGE);
            loadUserBookings(); // Refresh list
            updateLoyaltyPoints();
        });

        checkInBtn.addActionListener(e -> {
            int idx = bookingList.getSelectedIndex();
            if (idx == -1) {
                JOptionPane.showMessageDialog(frame, "Please select a booking to check in.", "Selection Error", JOptionPane.WARNING_MESSAGE);
                return;
            }
            String selectedValue = bookingList.getSelectedValue();
            String pnr = extractPNRFromBookingString(selectedValue);
            if (pnr == null) return;

            Booking booking = bookings.get(pnr);
            if (booking == null) return;

            if (booking.isCheckedIn) {
                JOptionPane.showMessageDialog(frame, "You have already checked in for booking PNR: " + pnr + ".", "Already Checked In", JOptionPane.INFORMATION_M
                return;
```

```java
    private String extractPNRFromBookingString(String bookingStr) {
        if (bookingStr == null) return null;
        // Updated to work with HTML formatted string from renderer
        String pnrPrefix = "PNR: ";
        int pnrStartIndex = bookingStr.indexOf(pnrPrefix);
        if (pnrStartIndex == -1) return null;

        pnrStartIndex += pnrPrefix.length();
        int pnrEndIndex = bookingStr.indexOf("</b>", pnrStartIndex); // PNR is bolded
        if (pnrEndIndex == -1) { // Fallback if not bolded for some reason
            pnrEndIndex = bookingStr.indexOf(" ", pnrStartIndex); // Find next space
            if (pnrEndIndex == -1) pnrEndIndex = bookingStr.indexOf("<br/>", pnrStartIndex); // Or next line break
            if (pnrEndIndex == -1) pnrEndIndex = bookingStr.length();
        }

        try {
            return bookingStr.substring(pnrStartIndex, pnrEndIndex).trim();
        } catch (Exception e) {
            System.err.println("Error extracting PNR from: " + bookingStr + " - " + e.getMessage());
            return null;
        }
    }

    private void loadUserBookings() {
        bookingListModel.clear();
        if (loggedInUser == null) { // Should not happen if panel is shown correctly
            bookingListModel.addElement("Error: Not logged in.");
            return;
        }
        if (loggedInUser.getBookings().isEmpty()) {
            bookingListModel.addElement("No bookings found. Time to plan a trip!");
            bookingList.setEnabled(false); // Disable list if empty
            return;
        }

        bookingList.setEnabled(true);
        List<String> pnrsSorted = new ArrayList<>(loggedInUser.getBookings());
        // Sort by booking date, most recent first
        pnrsSorted.sort((pnr1, pnr2) -> {
            Booking b1 = bookings.get(pnr1);
```

```java
    // -------- Admin panel -------
    private JPanel createAdminPanel() {
        JPanel panel = new JPanel(new BorderLayout(10,10));
        panel.setBorder(new EmptyBorder(10,15,15,15));
        panel.setBackground(new Color(240, 245, 250)); // Light grey-blue

        // Top Panel: Title and Logout
        JPanel topPanel = new JPanel(new BorderLayout());
        topPanel.setOpaque(false);
        JLabel title = new JLabel("Administrator Dashboard", JLabel.CENTER);
        title.setFont(new Font("Segoe UI", Font.BOLD, 28));
        title.setForeground(new Color(10, 70, 140));
        topPanel.add(title, BorderLayout.CENTER);

        JButton logoutBtn = createStyledButton("Logout Admin", new Color(200, 30, 30), Color.WHITE);
        logoutBtn.addActionListener(e -> logoutUser()); // Reusing logoutUser
        JPanel logoutPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT)); // To align button right
        logoutPanel.setOpaque(false);
        logoutPanel.add(logoutBtn);
        topPanel.add(logoutPanel, BorderLayout.EAST);
        panel.add(topPanel, BorderLayout.NORTH);

        // Center: Stats and Logs
        JSplitPane splitPane = new JSplitPane(JSplitPane.VERTICAL_SPLIT);
        splitPane.setOpaque(false);
        splitPane.setBorder(null); // Remove border from split pane itself

        // Stats Panel
        JPanel dashboardPanel = new JPanel(new GridLayout(2, 2, 20, 20)); // 2x2 grid for stats
        dashboardPanel.setOpaque(false);
        dashboardPanel.setBorder(BorderFactory.createTitledBorder(
                BorderFactory.createEtchedBorder(), "System Statistics",
                TitledBorder.LEFT, TitledBorder.TOP,
                new Font("Segoe UI", Font.BOLD, 18), new Color(10, 70, 140)));

        totalBookingsLabel = createAdminStatLabel("Total Bookings: 0");
        revenueLabel = createAdminStatLabel("Total Revenue: ₱0.00");
        occupancyLabel = createAdminStatLabel("Overall Occupancy: 0%");
        checkInCountLabel = createAdminStatLabel("Passengers Checked In: 0");
```

```java
    private void updateAdminDashboard() {
        if (totalBookingsLabel == null) return; // Panel not initialized yet

        totalBookingsLabel.setText("Total Bookings: " + bookings.size());

        double totalRevenue = bookings.values().stream().mapToDouble(b -> b.totalPrice).sum();
        revenueLabel.setText(String.format("Total Revenue: ₱%,.2f", totalRevenue));

        long totalCapacitySeats = localFlights.values().stream().mapToLong(f -> f.totalSeats).sum() +
                internationalFlights.values().stream().mapToLong(f -> f.totalSeats).sum();
        long totalBookedSeats = 0;
        for(Flight f : localFlights.values()) totalBookedSeats += f.reservedSeats.size();
        for(Flight f : internationalFlights.values()) totalBookedSeats += f.reservedSeats.size();


        int occupancy = totalCapacitySeats == 0 ? 0 : (int) ((totalBookedSeats * 100.0) / totalCapacitySeats);
        occupancyLabel.setText("Overall Occupancy: " + occupancy + "%");

        long checkedInCount = bookings.values().stream().filter(b -> b.isCheckedIn).count();
        checkInCountLabel.setText("Passengers Checked In: " + checkedInCount);
    }

    private void addAuditLog(String message) {
        if (auditLogModel != null) {
            String timestamp = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(new Date());
            auditLogModel.addElement(timestamp + " - " + message);
            if (auditLogList != null && auditLogModel.getSize() > 0) { // Auto-scroll
                auditLogList.ensureIndexIsVisible(auditLogModel.getSize() - 1);
            }
        }
    }

    // -------- Chatbot Panel (FIXED) -------
    private JPanel createChatbotPanel() {
        JPanel panel = new JPanel(new BorderLayout(10, 10));
        panel.setBorder(new EmptyBorder(15, 15, 15, 15));
        panel.setBackground(new Color(240, 248, 255)); // AliceBlue

        JLabel title = new JLabel("AI Flight Assistant", JLabel.CENTER);
        title.setFont(new Font("Segoe UI", Font.BOLD, 26));
```

```java
        panel.add(southWrapperPanel, BorderLayout.SOUTH);

        ActionListener sendAction = e -> processChatbotInput();
        sendBtn.addActionListener(sendAction);
        chatbotInput.addActionListener(sendAction); // Allow Enter to send

        backBtn.addActionListener(e -> {
            if (loggedInUser != null) {
                if (loggedInUser.getRole() == Role.ADMIN) {
                    cardLayout.show(mainPanel, "AdminPanel");
                } else {
                    cardLayout.show(mainPanel, "CustomerPanel");
                }
            } else {
                cardLayout.show(mainPanel, "Login"); // Should not happen if chatbot accessed after login
            }
        });

        // Set initial focus to input field when panel is shown
        panel.addComponentListener(new ComponentAdapter() {
            @Override
            public void componentShown(ComponentEvent e) {
                chatbotInput.requestFocusInWindow();
            }
        });

        return panel;
    }

    private void processChatbotInput() {
        String userInput = chatbotInput.getText().trim().toLowerCase();
        if (userInput.isEmpty()) return;

        chatbotArea.append("👤 You: " + chatbotInput.getText() + "\n");
        chatbotInput.setText(""); // Clear input field

        String botResponse = "🤖 AI Assistant: ";
        if (userInput.startsWith("hello") || userInput.startsWith("hi")) {
            botResponse += "Hello there! How can I help you today?";
        } else if (userInput.contains("flights to")) {
```

```java
    // Helper for styled buttons
    private JButton createStyledButton(String text, Color bgColor, Color fgColor) {
        JButton button = new JButton(text);
        button.setBackground(bgColor);
        button.setForeground(fgColor);
        button.setFont(new Font("Segoe UI", Font.BOLD, 14));
        button.setFocusPainted(false);
        button.setBorder(BorderFactory.createCompoundBorder(
                BorderFactory.createLineBorder(bgColor.darker(), 1), // Subtle border
                new EmptyBorder(8, 15, 8, 15) // Padding
        ));
        // Hover effect (simple)
        button.addMouseListener(new MouseAdapter() {
            Color originalBg = button.getBackground();
            public void mouseEntered(MouseEvent evt) {
                button.setBackground(originalBg.brighter());
            }
            public void mouseExited(MouseEvent evt) {
                button.setBackground(originalBg);
            }
        });
        return button;
    }

    // Dark Mode Toggle
    private void toggleDarkMode() {
        isDarkMode = !isDarkMode;
        addAuditLog("Dark mode " + (isDarkMode ? "enabled" : "disabled") + " by " + (loggedInUser != null ? loggedInUser.getUsername() : "System"));
        if (isDarkMode) {
            // Apply dark theme colors
            UIManager.put("Panel.background", Color.DARK_GRAY);
            UIManager.put("Label.foreground", Color.WHITE);
            UIManager.put("Button.background", Color.GRAY);
            UIManager.put("Button.foreground", Color.WHITE);
            UIManager.put("TextField.background", Color.LIGHT_GRAY);
            UIManager.put("TextField.foreground", Color.BLACK);
            UIManager.put("TextArea.background", Color.LIGHT_GRAY);
            UIManager.put("TextArea.foreground", Color.BLACK);
            UIManager.put("ComboBox.background", Color.GRAY);
            UIManager.put("ComboBox.foreground", Color.WHITE);
```

```java
    // --- Data classes ---

    private enum Role { ADMIN, CUSTOMER }

    private static class User {
        String username;
        String password; // In a real app, hash this!
        Role role;
        Set<String> bookingPNRs = new HashSet<>();
        int loyaltyPoints = 0;

        User(String u, String p, Role r) {
            username = u; password = p; role = r;
        }
        boolean checkPassword(String pass) { return password.equals(pass); } // Case-sensitive
        String getUsername() { return username; }
        Role getRole() { return role; }
        void addBooking(String pnr) { bookingPNRs.add(pnr); }
        void removeBooking(String pnr) { bookingPNRs.remove(pnr); }
        Set<String> getBookings() { return Collections.unmodifiableSet(bookingPNRs); } // Return unmodifiable view
        void addLoyaltyPoints(int pts) {
            this.loyaltyPoints += pts;
            if (this.loyaltyPoints < 0) this.loyaltyPoints = 0; // Prevent negative points
        }
        int getLoyaltyPoints() { return loyaltyPoints; }
    }

    private static class Flight {
        String route; // e.g., "Manila to Cebu"
        double baseFare;
        int totalSeats;
        Set<String> reservedSeats = new HashSet<>(); // Stores seat IDs like "1A", "12F"

        Flight(String r, double fare, int seats) {
            route = r; baseFare = fare; totalSeats = seats;
        }

        boolean isSeatReserved(String seatId) { return reservedSeats.contains(seatId); }

        boolean areSeatsAvailable(Set<String> seatsToCheck) {
```

```java
        void reserveSeats(Set<String> seatsToReserve) { reservedSeats.addAll(seatsToReserve); }
        void releaseSeats(Set<String> seatsToRelease) { reservedSeats.removeAll(seatsToRelease); }
        int getAvailableSeats() { return totalSeats - reservedSeats.size(); }
    }

    private static class Fare {
        String classType; // e.g., "Economy", "Business"
        double multiplier; // e.g., 1.0, 1.5, 2.0
        Fare(String ct, double m) { classType=ct; multiplier=m; }
    }

    private static class Booking {
        String pnr;
        User user; // Reference to the user who made the booking
        Flight flight; // Reference to the flight
        Fare fare; // Fare class chosen
        double totalPrice;
        Set<String> selectedSeats; // Set of seat IDs
        boolean isCheckedIn;
        Date bookingDate; // When the booking was made

        // Corrected constructor call was: new Booking(pnr, loggedInUser, tempFlight, tempFare, tempTotalPrice, new HashSet<>(selectedSeats), false,
        Booking(String p, User u, Flight f, Fare fr, double price, Set<String> seats, boolean checkedIn, Date bd) {
            pnr = p;
            user = u;
            flight = f;
            fare = fr;
            totalPrice = price;
            selectedSeats = new HashSet<>(seats); // Store a copy
            isCheckedIn = checkedIn;
            bookingDate = bd;
        }
        // Getters (optional, but good practice)
        String getPnr() { return pnr; }
        Set<String> getSelectedSeats() { return Collections.unmodifiableSet(selectedSeats); }
        Flight getFlight() { return flight; }
        User getUser() { return user; }
        Date getBookingDate() { return bookingDate; }
        boolean isCheckedIn() { return isCheckedIn; }
    }
```

This Java code defines a complete **AirplaneReservationSystem**, a desktop application built using the Swing graphical user interface (GUI) toolkit. It simulates a flight booking platform with various functionalities for both customers and administrators.

Let's break down the code's structure and key components:

1. Overall Application Structure (AirplaneReservationSystem class)

- **JFrame frame**: The main window of the application.
- **CardLayout cardLayout & JPanel mainPanel**: These are crucial for navigation. mainPanel uses CardLayout to switch between different screens (panels) like login, customer dashboard, seat selection, etc., allowing only one panel to be visible at a time.
- **Data Storage (Maps & Sets)**:
    - Map<String, User> users: Stores user accounts (username to User object).
    - Map<String, Flight> localFlights, internationalFlights: Store available flights, categorized by type. LinkedHashMap is used to maintain insertion order (though sorted() is used when populating the combo box).
    - Map<String, FlightStatus> flightStatuses: Stores the current status (ON_TIME, DELAYED, CANCELED) for each flight route.

    - Map<String, Booking> bookings: Stores all confirmed bookings, indexed by their PNR.
    - Set<String> selectedSeats: Temporarily holds the seats chosen by the user during the booking process.
    - Map<String, JButton> seatButtons: Stores references to the seat buttons for easy manipulation (e.g., changing color/state).
- **Temporary Booking Information**:
    - Flight tempFlight, Fare tempFare, int tempPersons, double tempTotalPrice: These variables temporarily store the details of the flight, fare class, number of passengers, and total price as the user progresses through the booking flow, before the booking is finalized.
- **UI Component References**: Many JLabel, JRadioButton, JComboBox, JSpinner, JTextArea, and JTextField objects are declared as class members to allow different methods to access and update their content (e.g., priceLabel, flightSearchCombo, chatbotArea).
- **boolean isDarkMode**: A flag to control the application's theme.
- **java.util.Timer uiTimer**: Used to schedule periodic tasks, like updating flight statuses and animating seat selection.
- **Admin Dashboard Labels**: totalBookingsLabel, revenueLabel, occupancyLabel, checkInCountLabel, auditLogModel, auditLogList are used to display real-time statistics and a log for administrators.
- **Random rand**: For simulating random flight statuses and weather.
- **FlightStatus Enum**: Defines the possible states of a flight.

2. Core Data Models (Inner Static Classes)

- **User**: Represents a user with a username, password (simplified, in a real app this would be hashed), role (CUSTOMER or ADMIN), and loyaltyPoints.
- **Role (Enum)**: Defines CUSTOMER and ADMIN roles.
- **Flight**: Represents a flight with a route, baseFare, capacity, and a Set<String> occupiedSeats to track which seats are taken.
- **Fare**: Represents a fare class with a className (e.g., "Economy") and a multiplier for the base fare.

- **Booking**: Represents a confirmed booking with a pnr (Passenger Name Record), username, flightRoute, fareClass, numberOfPersons, seats, totalPrice, actualPaidAmount (after points), paymentMethod, and checkedIn status.

3. Key Methods and Functionalities

- **main(String[] args)**: The entry point of the application, which schedules the GUI creation on the Event Dispatch Thread (EDT).
- **initUI()**: Initializes the main JFrame, mainPanel, sets up the CardLayout, initializes flight and user data, adds all the different UI panels to mainPanel, and makes the frame visible. It also starts the uiTimer.
- **startUITimer() / stopUITimer()**: Manages a timer that periodically calls updateFlightStatuses(), updateAdminDashboard(), and animateSeatSelection().
- **initFlights() / initUsers()**: Populates the initial flight and user data into their respective maps.
- **createLoginPanel()**: Builds the login screen with username, password fields, and login/register buttons. Handles authentication logic.
- **createRegisterPanel()**: Builds the user registration screen, including input fields and validation for creating new customer accounts.
- **createCustomerPanel()**: This is the main customer interface.
    - Allows selection of local/international flights via JRadioButtons.
    - Uses JComboBox for flight routes and fare classes.
    - Uses JSpinner for the number of persons.
    - **calculatePrice()**: Dynamically updates the priceLabel based on selected flight, class, and persons.
    - **populateFlights()**: Updates the flightSearchCombo based on whether local or international flights are selected.
    - Displays flightStatusLabel and weatherLabel (updated by updateFlightStatusDisplay() and updateWeatherInfo()).
    - Displays loyaltyPointsLabel.
    - Buttons to navigate to "My Bookings," "Chatbot," "Toggle Dark Mode," and "Logout."
- **updateFlightStatusDisplay() / updateFlightStatuses()**: Simulates real-time flight status changes (ON_TIME, DELAYED, CANCELED) and updates the display.

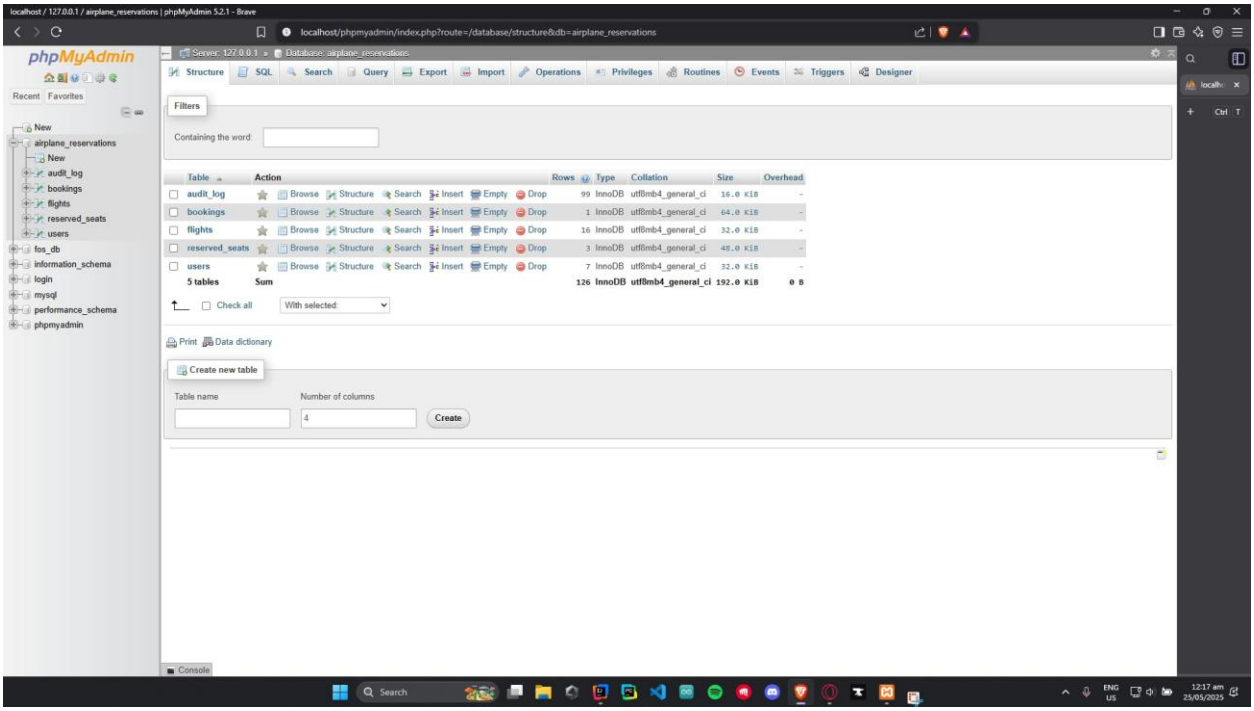
- **updateWeatherInfo()**: Simulates weather conditions for departure and destination cities.
- **proceedToSeatSelection()**: Validates booking details and transitions to the seat selection panel.
- **createSeatSelectionPanel()**: Builds the interactive seat map.
    - **populateSeats()**: Dynamically creates JButtons for each seat, color-coding them as available (green), selected (blue), or occupied (red). It also simulates initially occupied seats.
    - **toggleSeatSelection()**: Handles the logic for selecting/deselecting seats, enforcing the tempPersons limit.
    - **confirmSeatSelection()**: Validates the number of selected seats and proceeds to the payment panel.
- **createPaymentPanel()**: Handles the final booking confirmation and payment.
    - Displays a summary of the selected flight, class, persons, and seats.
    - Shows the totalAmountLbl and finalPriceLabel (which can be affected by loyalty points).
    - Allows selection of paymentMethod (including "Loyalty Points").
    - **Payment Logic**: On "Pay Now," it generates a PNR, creates a Booking object, adds it to bookings, updates loggedInUser's loyalty points (earning or deducting), marks seats as occupied in the Flight object, and displays a confirmation message.
    - **generatePNR()**: Generates a random PNR string.
    - **updateLoyaltyPoints()**: Updates the loyalty points display.
- **createBookingListPanel()**: Displays a list of the loggedInUser's bookings.
    - **loadUserBookings()**: Populates the bookingListModel with formatted booking strings.
    - Provides "Check-In" and "Cancel Booking" buttons with associated logic (checkInBooking(), cancelBooking()).
- **createAdminPanel()**: Provides an overview for administrators.
    - Displays system statistics: totalBookingsLabel, revenueLabel, occupancyLabel, checkInCountLabel.
    - **updateAdminDashboard()**: Calculates and updates these statistics.
    - Includes an auditLogList to display system events.

    - **addAuditLog()**: Adds new messages to the audit log.
- **createChatbotPanel()**: Implements a simple AI assistant.
    - chatbotArea for displaying conversation, chatbotInput for user input.
    - **processChatbotInput()**: Handles user input and calls generateChatbotResponse().
    - **generateChatbotResponse()**: Provides predefined responses based on keywords in the user's input.
- **createStyledButton()**: A helper method to create visually consistent buttons with hover effects.
- **logoutUser()**: Resets the loggedInUser and returns to the login screen.
- **toggleDarkMode() / applyDarkModeToComponent()**: Manages switching between light and dark themes by recursively applying color changes to UI components.

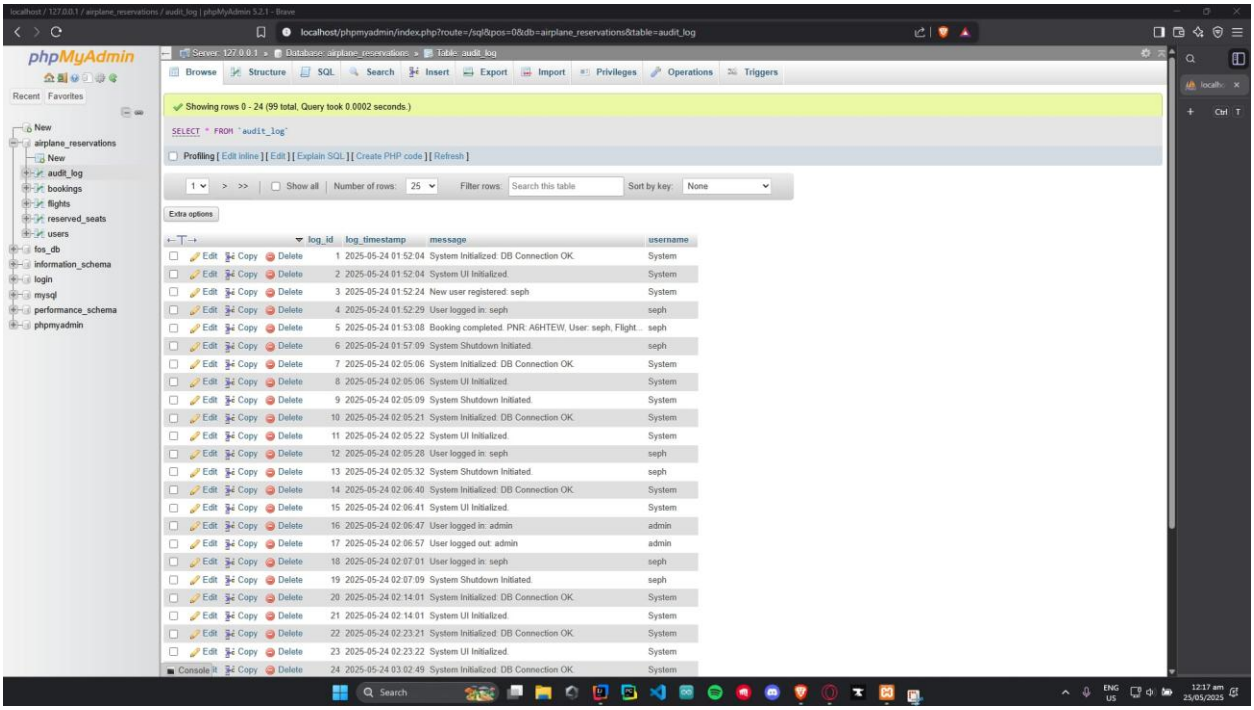
Conclusion

This AirplaneReservationSystem code provides a robust and well-structured example of a Java Swing application. It demonstrates key GUI programming concepts, including layout managers, event handling, custom components, and data management. The system effectively simulates a real-world scenario, offering a comprehensive user experience from account creation and flight booking to administrative monitoring and basic AI assistance.

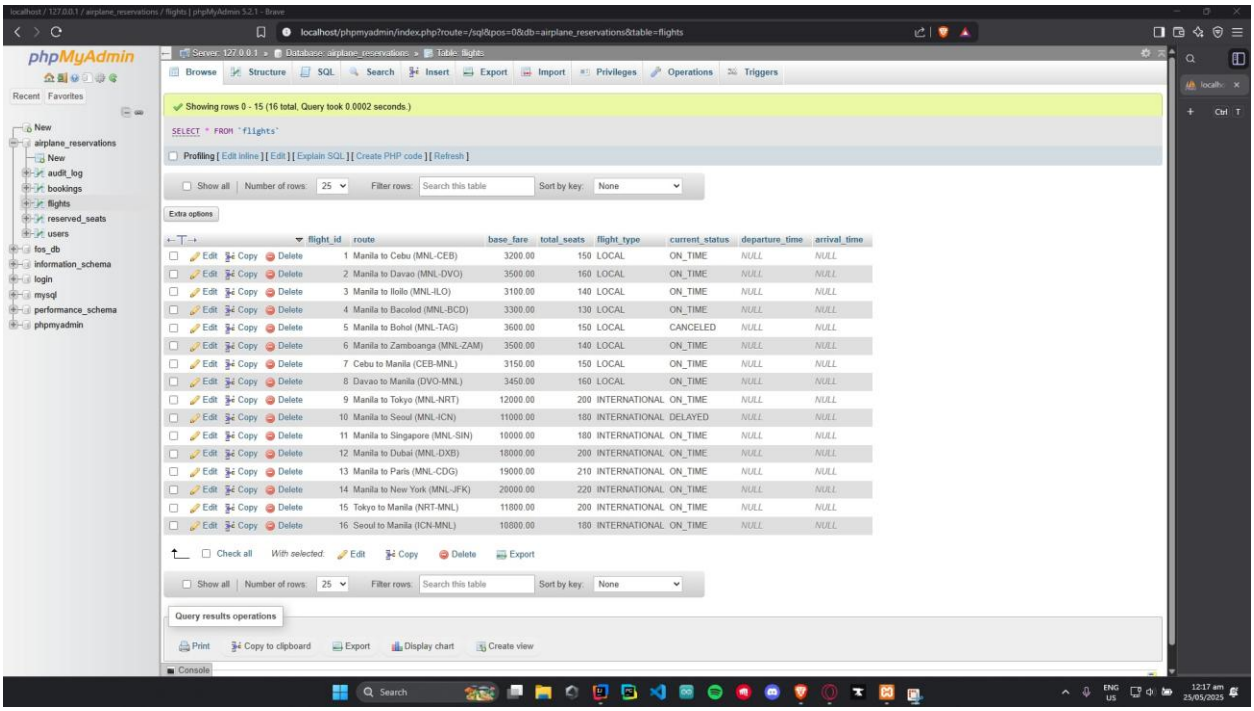
SS of database schema and tables with description



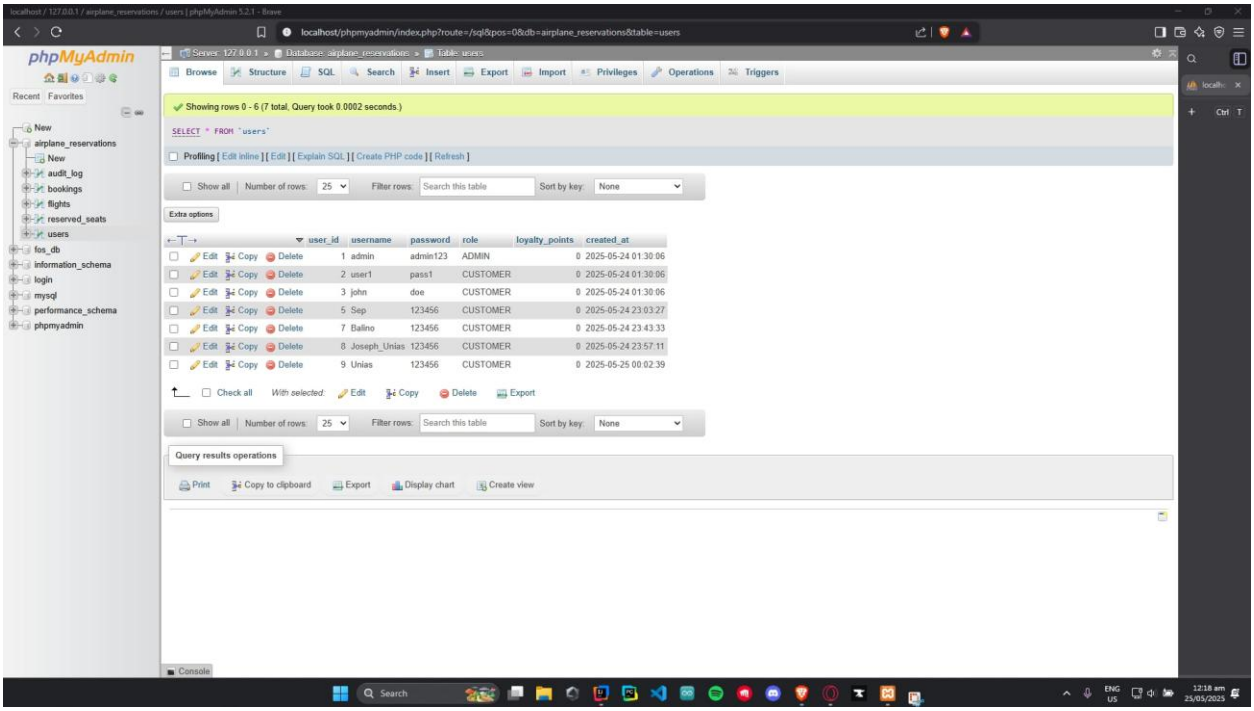
The image provides a clear view of the airplane_reservatons database, listing its tables, their row counts, storage engine, and size, along with various options for managing these tables and creating new ones. It's a typical administration view for someone working with a database application, likely related to an "airplane reservations" system.



This view displayed the actual log entries, including log_id, log_timestamp, message, and username, providing a clear record of system and user activities related to the "airplane reservations" system. The options to edit, copy, or delete individual rows were also visible, underscoring phpMyAdmin's utility for direct data manipulation.



This image specifically shows the operational data for various flights. It includes details like flight routes, base fares, flight types, and their current operational status. The NULL values in departure_time and arrival_time suggest that these time-specific details might be stored in a different, related table, or are simply not populated in this dataset.



This image displays the user data, including usernames, roles, and creation timestamps, within the airplane_reservations database. While it shows the functionality of phpMyAdmin in Browse user records, it also highlights a critical security vulnerability in how user passwords are being stored.