Airline Management System

A MINI-PROJECT BY:

Kashif Nazir 230701144

in partial fulfillment of the award of the degree

OF

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

NOVEMBER 2024

BONAFIDE CERTIFICATE

Certified that this project report "**Airline Management System"** is a Bonafide work of "**Kashif Nazir (230701144)"** who carried out the project work under my supervision**.**

Submitted for the Practical Examination held on _____

Signature

Ms. Dharshini B S

Assistant Professor,

Computer Science and Engineering,

Rajalakshmi Engineering College (Autonomous),

Thandalam, Chennai-602 105

Internal Examiner **External Examiner**

ABSTRACT:

The **Airline Management System** is a comprehensive platform designed to streamline and optimize the management of airline operations, including booking, scheduling, passenger management, flight tracking, and more. This system facilitates efficient handling of flight reservations, ticketing, customer service, and flight operations, providing a centralized solution for airline administrators and staff to manage daily activities.

By offering real-time updates on flight status, booking information, and customer details, the system ensures seamless communication between passengers, travel agents, and airline staff. It aids in simplifying the flight booking process, managing seat availability, handling customer queries, and improving overall operational efficiency.

Through its user-friendly interface, the **Airline Management System** allows users to book flights, manage reservations, view flight schedules, and track luggage, all while ensuring smooth operations from check-in to takeoff. The system includes an admin dashboard, enabling administrators to oversee and manage flight operations, monitor booking trends, and ensure timely departures and arrivals.

With robust security features and an efficient database management system, the **Airline Management System** ensures the accuracy, safety, and confidentiality of passenger and flight data. It supports both online and offline operations, ensuring operational continuity under all circumstances.

Overall, the **Airline Management System** enhances the efficiency of airline operations, empowers airline staff to provide better service to passengers, and helps passengers with a convenient and transparent booking experience.

TABLE OF CONTENTS

Introduction

Creating an **Airline Management System** using **Java as the front-end** and **SQL as the back-end** involves developing a software application that manages the various operations of an airline, such as flight bookings, ticket reservations, customer details, flight schedules, etc.

Here's a basic introduction to how you could approach this project:

1. System Overview

The system will be divided into two major parts:

- **Front-End (Java)**: Java is used to develop the user interface (UI) and handle the business logic of the airline management system.
- **Back-End (SQL Database)**: SQL is used to manage and store the data, such as customer information, flight schedules, bookings, etc.

2. Key Components

- **Front-End (Java):**
  - **Swing or JavaFX for UI**: Java Swing or JavaFX libraries can be used to build the graphical user interface (GUI) for the airline system.
  - **Java Classes and Methods**: Java classes will handle the logic for user actions (e.g., booking a flight, checking flight status).
  - **JDBC (Java Database Connectivity)**: JDBC is used to connect the front-end Java application to the SQL database, enabling operations like querying the database, inserting records, etc.
- **Back-End (SQL Database):**
  - **SQL Server / MySQL / PostgreSQL**: The back-end will store all data regarding the airline, such as passengers, flights, reservations, etc.
  - **Tables**:
    - **Flights**: Stores details of flights (flight number, origin, destination, departure time, arrival time).
    - **Passengers**: Stores details about the passengers (name, contact, ID).
    - **Reservations**: Stores information on flight reservations (passenger ID, flight ID, booking status, etc.).
  - **SQL Queries**: Used to manipulate the data stored in the database (SELECT, INSERT, UPDATE, DELETE).

3. Example Workflow

1. **User Login**:
    a. A user logs into the system (either an admin or customer).
    b. Java front-end captures the credentials and checks them against the user data stored in the SQL database.
2. **Flight Search**:
    a. After logging in, the user searches for available flights by inputting the origin, destination, and date.
    b. Java queries the SQL database to return matching flights and displays the results on the UI.
3. **Booking a Flight**:
    a. The user selects a flight, and a booking form is displayed.
    b. The front-end collects the passenger details (name, contact) and sends this data to the SQL database to create a new reservation.
4. **Flight Information**:
    a. Users can view current flights, including flight status (delayed, on-time), available seats, etc., all of which are pulled from the database.
5. **Selecting a Booking to Cancel:**

- a. Action: The customer selects the booking they wish to cancel.
- **Details**:
    - From the list of bookings, the user clicks on the booking they wish to cancel.
    - Java displays a **Booking Details Page** where the customer can see detailed information about the selected flight (including flight status, available seats, etc.).
    - A **Cancel Booking** button or option is provided.

6. **Initiating Cancellation:**
   Action: The customer clicks on the **Cancel Booking** button
   Details:
    a. Java prompts the user with a **confirmation dialog**: "Are you sure you want to cancel your booking?"
    b. If the customer confirms, the system proceeds with the cancellation process.
    c. Java then sends a **cancellation request** to the **SQL database** to update the booking status to "Cancelled."

Chapter 2: SURVEY OF TECHNOLOGIES

2.1 SOFTWARE DESCRIPTION

Building an Airline Management System (AMS) requires a combination of various technologies to handle user interfaces, data management, security, and backend operations. Below is an overview of the key technologies that can be used in the development of an airline management system, segmented by front-end, back-end, database, and additional technologies.

Java (Swing/JavaFX)

- **Overview**: Java is a powerful, widely used programming language for building cross-platform applications. In an Airline Management System, Java can be used for building desktop-based applications with rich graphical user interfaces.
    - **Swing**: A part of the Java Foundation Classes (JFC), it allows developers to build GUI-based applications.
    - **JavaFX**: A more modern and feature-rich alternative to Swing, JavaFX provides a better look-and-feel and includes tools for handling multimedia content.
- **Advantages**:
    - Platform independence (can run on any system with a JVM).
    - Strong community and support.
    - Ability to create complex user interfaces.

MYSQL

- **Role**: MySQL is used as the relational database for storing and managing all student-related information.
- **Usage**:
    - Stores student profiles, attendance, grades, and course information.
    - Efficient SQL queries enable quick retrieval and management of large datasets.
- **Advantages**:
    - Reliable, open-source database management.
    - Ensures data integrity and supports complex queries for academic reporting.

Chapter 3: REQUIREMENTS AND ANALYSIS

3.1 REQUIREMENT SPECIFICATION

The functional requirements define the core functionalities the system must support, from user interactions to backend operations.

1. User Management

- **Login/Logout**:
  - Users must authenticate themselves through a secure login mechanism (username and password).
  - Different roles (Admin, Customer, Staff) should have appropriate access permissions.
  - Users should be able to log out of the system.
- **Registration**:
  - New users (customers) should be able to register by providing personal details (name, contact, address, etc.).
  - Administrators should be able to register new airline staff and assign appropriate roles.
- **Profile Management**:
  - Users should be able to manage their personal information, including contact details, address, and payment information.
  - Customers should be able to view their booking history and cancellation records.

2. Flight Management

- **Flight Scheduling**:
  - Admin users should be able to add, update, or delete flight schedules, including details like flight number, origin, destination, departure/arrival times, and available seats.
- **Flight Availability**:
  - The system should display available flights based on search criteria such as date, origin, and destination.
  - The user should be able to see flight details, including available seats and ticket prices.
- **Flight Status**:
  - The system must track and display the real-time status of flights (on-time, delayed, cancelled).
-

3. Ticket Management

- **Ticket Booking**:
  - Customers should be able to book tickets by entering necessary information such as passenger details, payment details, and travel preferences.
  - The system should validate the entered data and ensure the flight is available before confirming the ticket.
- **Ticket Cancellation**:
  - Customers should be able to cancel a booking.
  - The system should allow for refunds based on the airline's cancellation policy.

HARDWARE AND SOFTWARE REQUIREMENTS

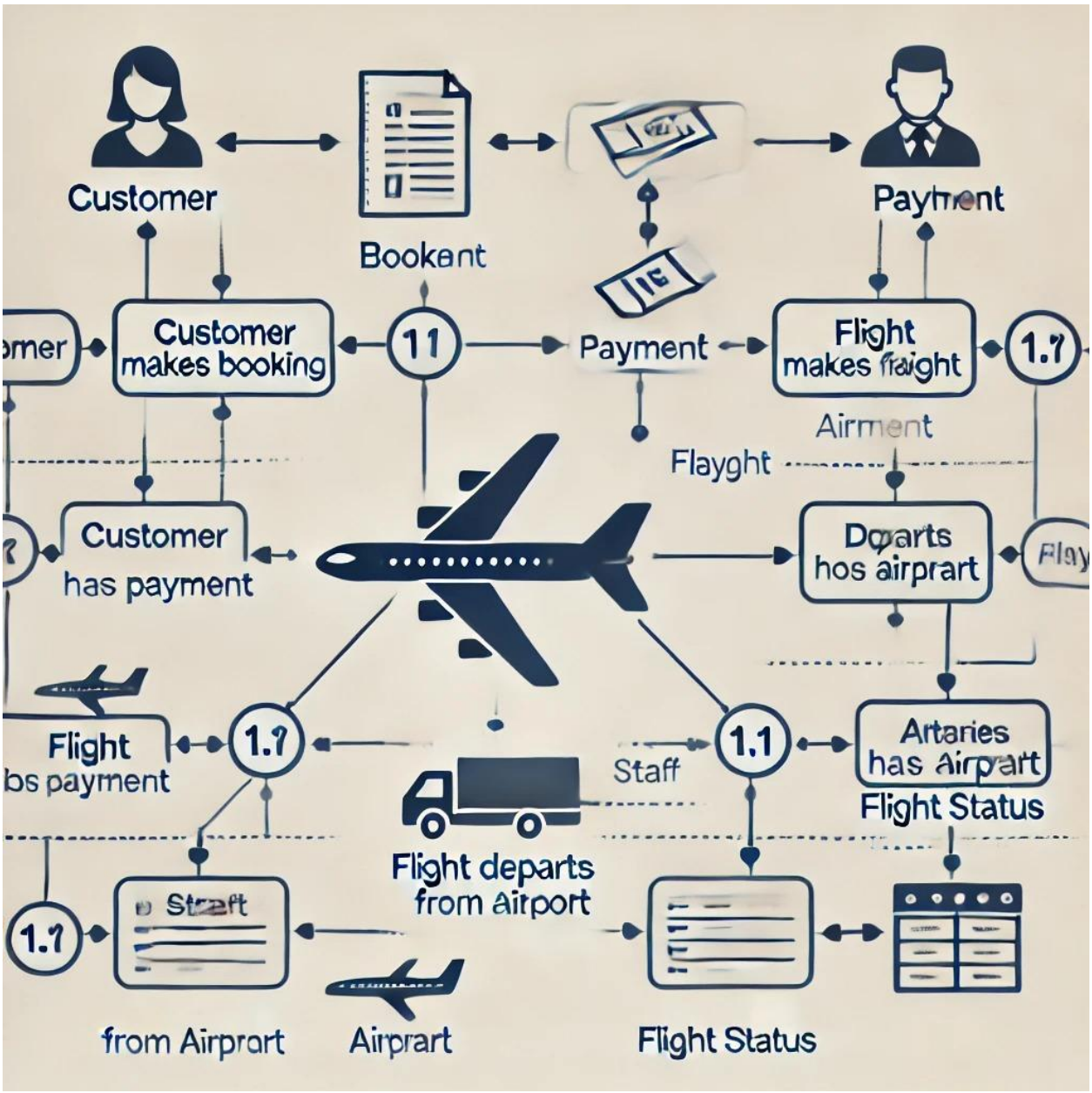
Hardware Requirement

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

Software Requirement

- **Operating System**: Windows 10 or higher.
- **Frontend**: Java Swing (JFrame-based interface).
- **Backend**: MySQL for database management.
- **IDE**: NetBeans for development.
- **Version Control**: Git for code versioning and collaboration.

3.3 ER DIAGRAM


Customer
Bookant
Payment
Customer makes booking
1.1
Payment
Flight makes flight
1.7
Airment
Flayght
Customer has payment
Departs hos airprart
Flight bs payment
1.7
1.1
Staff
Artaries has Airprart
Flight Status
Flight departs from Airport
1.7
Staft
from Airprart
Airprart
Flight Status

Chapter 4: JAVA PROGRAM CODE

Login And Signup Page

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

public class Login extends JFrame {
    private JTextField usernameField;
    private JPasswordField passwordField;
    private JButton loginButton;
    private JButton signupButton;

    public Login() {
        // Set title and window properties
        setTitle("Login");
        setSize(300, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(null);

        // Define the dark green color (same as the HomePage background)
        Color darkGreen = new Color(0, 51, 38);

        // Set background color to dark green (same as the HomePage)
        getContentPane().setBackground(darkGreen);

        // Username label and text field
        JLabel usernameLabel = new JLabel("Username:");
        usernameLabel.setBounds(20, 30, 80, 25);
        usernameLabel.setForeground(Color.WHITE); // White text for the label
        add(usernameLabel);

        usernameField = new JTextField();
        usernameField.setBounds(100, 30, 150, 25);
        usernameField.setForeground(darkGreen); // Dark green text color (same as
background)
        usernameField.setBackground(Color.WHITE); // Set white background to make
text visible
        add(usernameField);
```

```
        // Password label and text field
        JLabel passwordLabel = new JLabel("Password:");
        passwordLabel.setBounds(20, 70, 80, 25);
        passwordLabel.setForeground(Color.WHITE); // White text for the label
        add(passwordLabel);

        passwordField = new JPasswordField();
        passwordField.setBounds(100, 70, 150, 25);
        passwordField.setForeground(darkGreen); // Dark green text color (same as
background)
        passwordField.setBackground(Color.WHITE); // Set white background to make
text visible
        add(passwordField);

        // Login button styling
        loginButton = new JButton("Login");
        loginButton.setBounds(100, 110, 150, 25);
        loginButton.setBackground(new Color(245, 210, 115)); // Soft Gold
(matching Etihad)
        loginButton.setForeground(Color.BLACK); // Black text
        loginButton.setFocusPainted(false); // Remove focus border
        add(loginButton);

        // Sign Up button styling
        signupButton = new JButton("Sign Up");
        signupButton.setBounds(100, 140, 150, 25);
        signupButton.setBackground(new Color(245, 210, 115)); // Soft Gold
(matching Etihad)
        signupButton.setForeground(Color.BLACK); // Black text
        signupButton.setFocusPainted(false); // Remove focus border
        add(signupButton);

        // Action Listener for Login button
        loginButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String username = usernameField.getText();
                String password = new String(passwordField.getPassword());
                loginUser(username, password);
            }
        });

        // Action Listener for Sign Up button
```

```
        signupButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                new Signup();
                dispose(); // Close the login window when signing up
            }
        });

        setVisible(true);
    }

    private void loginUser(String username, String password) {
        // Check for admin login (both username and password are "admin")
        if (username.equals("admin") && password.equals("admin")) {
            // If login is successful, go to HomePage
            JOptionPane.showMessageDialog(this, "Login successful!");
            new HomePage(); // Show HomePage
            dispose(); // Close the Login window
        }
        // Check if the username exists in the in-memory "database" (from Signup
class)
        else if (Signup.userDatabase.containsKey(username)) {
            if (Signup.userDatabase.get(username).equals(password)) {
                // If login is successful, go to HomePage
                JOptionPane.showMessageDialog(this, "Login successful!");
                new HomePage(); // Show HomePage
                dispose(); // Close the Login window
            } else {
                JOptionPane.showMessageDialog(this, "Invalid password.");
            }
        } else {
            JOptionPane.showMessageDialog(this, "Invalid username.");
        }
    }

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

1.2-Signup Page

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.HashMap;
import java.util.Map;

public class Signup extends JFrame {
    private JTextField usernameField;
    private JPasswordField passwordField;
    private JPasswordField confirmPasswordField;
    private JButton signUpButton;
    private JButton backButton;

    // Static in-memory storage for users (username -> password)
    public static Map<String, String> userDatabase = new HashMap<>();

    // Color Definitions (matching Login page)
    private static final Color DARK_GREEN = new Color(0, 51, 38); // Dark green
background
    private static final Color SOFT_GOLD = new Color(245, 210, 115); // Soft Gold
for buttons
    private static final Color WHITE = Color.WHITE; // White for text fields and
labels

    public Signup() {
        // Add the default admin user to the database
        if (!userDatabase.containsKey("admin")) {
            userDatabase.put("admin", "admin");
        }

        setTitle("Sign Up");
        setSize(300, 250);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(null);
        getContentPane().setBackground(DARK_GREEN); // Set background to dark
green

        // Username label and text field
        JLabel usernameLabel = new JLabel("Username:");
        usernameLabel.setBounds(20, 30, 80, 25);
        usernameLabel.setForeground(WHITE); // White text for the label
```

```
        add(usernameLabel);

        usernameField = new JTextField();
        usernameField.setBounds(100, 30, 150, 25);
        usernameField.setForeground(DARK_GREEN); // Dark green text color
        usernameField.setBackground(WHITE); // Set white background to make text
visible
        add(usernameField);

        // Password label and text field
        JLabel passwordLabel = new JLabel("Password:");
        passwordLabel.setBounds(20, 70, 80, 25);
        passwordLabel.setForeground(WHITE); // White text for the label
        add(passwordLabel);

        passwordField = new JPasswordField();
        passwordField.setBounds(100, 70, 150, 25);
        passwordField.setForeground(DARK_GREEN); // Dark green text color
        passwordField.setBackground(WHITE); // Set white background to make text
visible
        add(passwordField);

        // Confirm Password label and text field
        JLabel confirmPasswordLabel = new JLabel("Confirm Password:");
        confirmPasswordLabel.setBounds(20, 110, 120, 25);
        confirmPasswordLabel.setForeground(WHITE); // White text for the label
        add(confirmPasswordLabel);

        confirmPasswordField = new JPasswordField();
        confirmPasswordField.setBounds(140, 110, 150, 25);
        confirmPasswordField.setForeground(DARK_GREEN); // Dark green text color
        confirmPasswordField.setBackground(WHITE); // Set white background to
make text visible
        add(confirmPasswordField);

        // Sign Up button styling
        signUpButton = new JButton("Sign Up");
        signUpButton.setBounds(100, 150, 150, 25);
        signUpButton.setBackground(SOFT_GOLD);  // Soft Gold for the button
        signUpButton.setForeground(Color.BLACK); // Black text
        signUpButton.setFocusPainted(false); // Remove focus border
        add(signUpButton);
```

```
        // Back button styling
        backButton = new JButton("Back");
        backButton.setBounds(100, 180, 150, 25);
        backButton.setBackground(SOFT_GOLD);  // Soft Gold for the button
        backButton.setForeground(Color.BLACK); // Black text
        backButton.setFocusPainted(false); // Remove focus border
        add(backButton);

        // Action Listener for Sign Up button
        signUpButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String username = usernameField.getText();
                String password = new String(passwordField.getPassword());
                String confirmPassword = new
String(confirmPasswordField.getPassword());
                signUpUser(username, password, confirmPassword);
            }
        });

        // Action Listener for Back button
        backButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                new Login();
                dispose(); // Close the Sign Up window
            }
        });

        setVisible(true);
    }

    private void signUpUser(String username, String password, String
confirmPassword) {
        if (username.equals("admin")) {
            JOptionPane.showMessageDialog(this, "Username 'admin' is reserved.");
            return;
        }

        if (!password.equals(confirmPassword)) {
            JOptionPane.showMessageDialog(this, "Passwords do not match.");
            return;
        }
```

```
        if (userDatabase.containsKey(username)) {
            JOptionPane.showMessageDialog(this, "Username already exists. Choose
another username.");
        } else {
            userDatabase.put(username, password);
            JOptionPane.showMessageDialog(this, "User registered successfully!");
        }
    }

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

2.HOME PAGE

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

public class HomePage extends JFrame {

    public HomePage() {
        setTitle("Home Page");
        setSize(600, 500); // Increase window height to accommodate larger image
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        // Create a panel for the buttons with FlowLayout for horizontal
arrangement
        JPanel buttonPanel = new JPanel();
        buttonPanel.setLayout(new FlowLayout(FlowLayout.CENTER, 10, 10)); //
Centered buttons with spacing between
        buttonPanel.setBackground(new Color(230, 215, 180)); // Beige background
to match Etihad style

        // Create the buttons with custom colors and smaller sizes
        JButton bookTicketsButton = new JButton("Book Tickets");
        JButton cancelTicketsButton = new JButton("Cancel Tickets");
        JButton bookedTicketsButton = new JButton("Booked Tickets");
        JButton availableTicketsButton = new JButton("Available Tickets"); //
Renamed button

        // Set custom button sizes (smaller buttons)
        Dimension buttonSize = new Dimension(120, 25); // Smaller width and
height for buttons
        bookTicketsButton.setPreferredSize(buttonSize);
        cancelTicketsButton.setPreferredSize(buttonSize);
        bookedTicketsButton.setPreferredSize(buttonSize);
        availableTicketsButton.setPreferredSize(buttonSize);

        // Set Etihad-inspired colors for buttons
        bookTicketsButton.setBackground(new Color(102, 85, 75)); // Dark brown
tone
        cancelTicketsButton.setBackground(new Color(190, 140, 30)); // Gold for
cancellation
```

```
        bookedTicketsButton.setBackground(new Color(110, 90, 80)); // Warm brown
tone
        availableTicketsButton.setBackground(new Color(225, 190, 130)); // Light
gold tone

        // Set text color for better readability
        bookTicketsButton.setForeground(Color.WHITE);
        cancelTicketsButton.setForeground(Color.BLACK);
        bookedTicketsButton.setForeground(Color.WHITE);
        availableTicketsButton.setForeground(Color.BLACK);

        // Add buttons to the panel
        buttonPanel.add(bookTicketsButton);
        buttonPanel.add(cancelTicketsButton);
        buttonPanel.add(bookedTicketsButton);
        buttonPanel.add(availableTicketsButton); // Added renamed button

        // Add the buttons panel to the top of the frame (NORTH)
        add(buttonPanel, BorderLayout.NORTH);

        // Set background color for main content panel and add a larger image to
the center (CENTER)
        JPanel imagePanel = new JPanel();
        imagePanel.setBackground(new Color(230, 215, 180)); // Beige background
for central panel
        ImageIcon originalIcon = new
ImageIcon("C:\\Users\\Dell\\OneDrive\\Desktop\\java project\\images.png"); //
Update with your image path
        Image scaledImage = originalIcon.getImage().getScaledInstance(600, 300,
Image.SCALE_SMOOTH); // Resize to 600x300 pixels
        ImageIcon scaledIcon = new ImageIcon(scaledImage);
        JLabel imageLabel = new JLabel(scaledIcon, JLabel.CENTER);
        imagePanel.add(imageLabel);
        add(imagePanel, BorderLayout.CENTER);

        // Action Listeners for buttons
        bookTicketsButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                new BookTickets(); // Opens the BookTickets window
            }
        });
```

```
        cancelTicketsButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                new CancelTicket(); // Opens the CancelTicket window
            }
        });

        bookedTicketsButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                new BookedTickets(); // Opens the BookedTickets window
            }
        });

        availableTicketsButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                new AvailableTickets(); // Opens the AvailableTickets window
(renamed action)
            }
        });

        setVisible(true);
    }

    public static void main(String[] args) {
        new HomePage(); // Launch the HomePage
    }
}
```

3.Available tickets

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.List;

public class AvailableTickets extends JFrame {

    private JTable availableFlightsTable;
    private JScrollPane scrollPane;
    private JTextField searchField;  // Search field for user input
    private JButton backButton;      // Back button to return to HomePage

    // List to hold available flights (instead of fetching from DB)
    private List<Flight> availableFlights;

    // Etihad Gold Color (approximation)
    private static final Color ETIHAD_GOLD = new Color(252, 205, 79);

    public AvailableTickets() {
        setTitle("Available Tickets");  // Changed title here
        setSize(600, 500);  // Adjusted window size for the table and buttons
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setLayout(new BorderLayout());

        // Initialize available flights data
        availableFlights = new ArrayList<>();
        populateAvailableFlights();  // Populate the list with sample data

        // Title label
        JLabel titleLabel = new JLabel("Available Tickets",
SwingConstants.CENTER);  // Changed title here
        titleLabel.setFont(new Font("Arial", Font.BOLD, 20));
        add(titleLabel, BorderLayout.NORTH);

        // Search Panel (for search field and button)
        JPanel searchPanel = new JPanel();
        searchPanel.setLayout(new BorderLayout());
```

```
        // Search Field
        searchField = new JTextField();
        searchPanel.add(searchField, BorderLayout.CENTER);

        // Search Button
        JButton searchButton = new JButton("Search");
        searchButton.setBackground(ETIHAD_GOLD);  // Set button background to
Etihad Gold
        searchButton.setForeground(Color.BLACK);  // Set button text color to
black
        searchPanel.add(searchButton, BorderLayout.EAST);

        // Add search panel to the top
        add(searchPanel, BorderLayout.NORTH);

        // Panel for table
        JPanel tablePanel = new JPanel();
        tablePanel.setLayout(new BorderLayout());

        // Create a table with column names
        String[] columnNames = {"Airline Name", "Flight Number", "Origin",
"Destination", "Travel Date", "Available Seats"};

        // Create a table model
        DefaultTableModel model = new DefaultTableModel();
        model.setColumnIdentifiers(columnNames);

        // Fetch data and add to table model (initially show all available
flights)
        addFlightsToTable(model, availableFlights);

        // Create the JTable using the model
        availableFlightsTable = new JTable(model);
        scrollPane = new JScrollPane(availableFlightsTable);  // Adding scroll
functionality to the table
        tablePanel.add(scrollPane, BorderLayout.CENTER);

        // Add the table panel to the frame
        add(tablePanel, BorderLayout.CENTER);

        // Back Button Panel
        JPanel backPanel = new JPanel();
```

```
        backPanel.setLayout(new BorderLayout());

        backButton = new JButton("Back");
        backButton.setBackground(ETIHAD_GOLD);  // Set button background to
Etihad Gold
        backButton.setForeground(Color.BLACK);  // Set button text color to black
        backPanel.add(backButton, BorderLayout.CENTER);

        // Add Back Button at the bottom
        add(backPanel, BorderLayout.SOUTH);

        // Action Listener for Search Button
        searchButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String searchQuery = searchField.getText().trim();
                List<Flight> filteredFlights = searchFlights(searchQuery);
                addFlightsToTable(model, filteredFlights);  // Update the table
with filtered results
            }
        });

        // Action Listener for Back Button
        backButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                dispose();  // Close the current window (AvailableTickets)
                new HomePage();  // Open the HomePage
            }
        });

        setVisible(true);
    }

    // Populate the list of available flights with sample data
    private void populateAvailableFlights() {
        availableFlights.add(new Flight("Airline A", "AA123", "New York", "Los
Angeles", "2024-12-01", 100));
        availableFlights.add(new Flight("Airline B", "BB456", "Chicago", "San
Francisco", "2024-12-05", 50));
        availableFlights.add(new Flight("Airline C", "CC789", "Dallas", "Miami",
"2024-12-10", 30));
```

```
        availableFlights.add(new Flight("Airline A", "AA101", "Los Angeles", "New
York", "2024-12-15", 75));
        availableFlights.add(new Flight("Airline D", "DD202", "San Francisco",
"Chicago", "2024-12-20", 200));

        availableFlights.add(new Flight("Airline E", "EE303", "Houston",
"Seattle", "2024-12-21", 120));
        availableFlights.add(new Flight("Airline F", "FF404", "Phoenix",
"Denver", "2024-12-22", 95));
        availableFlights.add(new Flight("Airline G", "GG505", "Boston",
"Orlando", "2024-12-23", 60));
        availableFlights.add(new Flight("Airline H", "HH606", "Atlanta", "Las
Vegas", "2024-12-24", 110));
        availableFlights.add(new Flight("Airline I", "II707", "San Diego",
"Portland", "2024-12-25", 70));

        availableFlights.add(new Flight("Airline J", "JJ808", "Washington", "Salt
Lake City", "2024-12-26", 85));
        availableFlights.add(new Flight("Airline K", "KK909", "Philadelphia",
"Minneapolis", "2024-12-27", 95));
        availableFlights.add(new Flight("Airline L", "LL1010", "San Antonio",
"Detroit", "2024-12-28", 65));
        availableFlights.add(new Flight("Airline M", "MM1111", "Jacksonville",
"Charlotte", "2024-12-29", 80));
        availableFlights.add(new Flight("Airline N", "NN1212", "Columbus",
"Indianapolis", "2024-12-30", 55));

        availableFlights.add(new Flight("Airline O", "OO1313", "Fort Worth", "El
Paso", "2024-12-31", 150));
        availableFlights.add(new Flight("Airline P", "PP1414", "Nashville",
"Memphis", "2024-12-31", 60));
        availableFlights.add(new Flight("Airline Q", "QQ1515", "Louisville",
"Baltimore", "2025-01-01", 75));
        availableFlights.add(new Flight("Airline R", "RR1616", "Milwaukee",
"Albuquerque", "2025-01-02", 95));
        availableFlights.add(new Flight("Airline S", "SS1717", "Tucson",
"Fresno", "2025-01-03", 85));

        availableFlights.add(new Flight("Airline T", "TT1818", "Sacramento",
"Mesa", "2025-01-04", 55));
        availableFlights.add(new Flight("Airline U", "UU1919", "Kansas City",
"Omaha", "2025-01-05", 60));
```

```
        availableFlights.add(new Flight("Airline V", "VV2020", "Long Beach",
"Virginia Beach", "2025-01-06", 45));
        availableFlights.add(new Flight("Airline W", "WW2121", "Miami",
"Atlanta", "2025-01-07", 70));
        availableFlights.add(new Flight("Airline X", "XX2222", "Denver",
"Houston", "2025-01-08", 90));

        availableFlights.add(new Flight("Airline Y", "YY2323", "Los Angeles",
"San Francisco", "2025-01-09", 50));
        availableFlights.add(new Flight("Airline Z", "ZZ2424", "New York",
"Chicago", "2025-01-10", 125));
        availableFlights.add(new Flight("Airline A1", "A12525", "Boston",
"Washington", "2025-01-11", 60));
        availableFlights.add(new Flight("Airline B1", "B12626", "Seattle", "Salt
Lake City", "2025-01-12", 40));
        availableFlights.add(new Flight("Airline C1", "C12727", "Orlando",
"Dallas", "2025-01-13", 115));

        availableFlights.add(new Flight("Airline D1", "D12828", "Minneapolis",
"Phoenix", "2025-01-14", 90));
        availableFlights.add(new Flight("Airline E1", "E12929", "San Jose",
"Sacramento", "2025-01-15", 30));
        availableFlights.add(new Flight("Airline F1", "F13030", "Austin",
"Indianapolis", "2025-01-16", 85));
        availableFlights.add(new Flight("Airline G1", "G13131", "Charlotte",
"Milwaukee", "2025-01-17", 65));
        availableFlights.add(new Flight("Airline H1", "H13232", "El Paso",
"Jacksonville", "2025-01-18", 55));

        availableFlights.add(new Flight("Airline I1", "I13333", "Las Vegas",
"Columbus", "2025-01-19", 90));
        availableFlights.add(new Flight("Airline J1", "J13434", "Salt Lake City",
"Nashville", "2025-01-20", 50));
        availableFlights.add(new Flight("Airline K1", "K13535", "Detroit", "Fort
Worth", "2025-01-21", 150));
        availableFlights.add(new Flight("Airline L1", "L13636", "Baltimore", "San
Antonio", "2025-01-22", 120));
        availableFlights.add(new Flight("Airline M1", "M13737", "Chicago", "Long
Beach", "2025-01-23", 60));
    }

    // Add flights to the table model
```

```
    private void addFlightsToTable(DefaultTableModel model, List<Flight> flights)
{
        model.setRowCount(0);  // Clear the existing table data
        for (Flight flight : flights) {
            model.addRow(new Object[]{flight.getAirlineName(),
flight.getFlightNumber(),
                    flight.getOrigin(), flight.getDestination(),
flight.getTravelDate(), flight.getAvailableSeats()});
        }
    }

    // Search flights based on user input
    private List<Flight> searchFlights(String query) {
        List<Flight> filteredFlights = new ArrayList<>();
        for (Flight flight : availableFlights) {
            if
(flight.getAirlineName().toLowerCase().contains(query.toLowerCase()) ||

flight.getFlightNumber().toLowerCase().contains(query.toLowerCase()) ||

flight.getOrigin().toLowerCase().contains(query.toLowerCase()) ||

flight.getDestination().toLowerCase().contains(query.toLowerCase())) {
                filteredFlights.add(flight);
            }
        }
        return filteredFlights;
    }

    // Flight class for holding flight details
    public class Flight {
        private String airlineName;
        private String flightNumber;
        private String origin;
        private String destination;
        private String travelDate;
        private int availableSeats;

        public Flight(String airlineName, String flightNumber, String origin,
String destination, String travelDate, int availableSeats) {
            this.airlineName = airlineName;
            this.flightNumber = flightNumber;
            this.origin = origin;
```

```
            this.destination = destination;
            this.travelDate = travelDate;
            this.availableSeats = availableSeats;
        }

        public String getAirlineName() {
            return airlineName;
        }

        public String getFlightNumber() {
            return flightNumber;
        }

        public String getOrigin() {
            return origin;
        }

        public String getDestination() {
            return destination;
        }

        public String getTravelDate() {
            return travelDate;
        }

        public int getAvailableSeats() {
            return availableSeats;
        }
    }

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

4.Booked TIckets

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

public class BookedTickets extends JFrame {

    private JTable bookedTicketsTable;
    private JScrollPane scrollPane;
    private JTextField searchField;  // Search field for user input
    private JButton backButton;      // Back button to return to HomePage

    // Database connection parameters
    private static final String DB_URL = "jdbc:mysql://localhost:3306/Airline";
// Update with your DB URL
    private static final String DB_USER = "root";  // Update with your DB
username
    private static final String DB_PASSWORD = "1234";  // Update with your DB
password

    // Etihad Gold Color (approximation)
    private static final Color ETIHAD_GOLD = new Color(252, 205, 79);

    public BookedTickets() {
        setTitle("Booked Tickets");
        setSize(600, 500);  // Adjusted window size for the table and buttons
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setLayout(new BorderLayout());

        // Define the light gold color
        Color lightGold = new Color(255, 239, 153);  // Light Gold color

        // Set the background color to light gold
        getContentPane().setBackground(lightGold);

        // Title label
        JLabel titleLabel = new JLabel("Booked Tickets", SwingConstants.CENTER);
        titleLabel.setFont(new Font("Arial", Font.BOLD, 20));
        titleLabel.setForeground(Color.BLACK);  // Set the text color to black
```

```
        add(titleLabel, BorderLayout.NORTH);

        // Search Panel (for search field and button)
        JPanel searchPanel = new JPanel();
        searchPanel.setLayout(new BorderLayout());
        searchPanel.setBackground(lightGold);  // Set background color to light
gold

        // Search Field
        searchField = new JTextField();
        searchPanel.add(searchField, BorderLayout.CENTER);

        // Search Button
        JButton searchButton = new JButton("Search");
        searchButton.setBackground(ETIHAD_GOLD);  // Set button background to
Etihad Gold
        searchButton.setForeground(Color.BLACK);  // Set button text color to
black
        searchPanel.add(searchButton, BorderLayout.EAST);

        // Add search panel to the top
        add(searchPanel, BorderLayout.NORTH);

        // Panel for table
        JPanel tablePanel = new JPanel();
        tablePanel.setLayout(new BorderLayout());
        tablePanel.setBackground(lightGold);  // Set background color to light
gold

        // Create a table with column names (Change these based on your database
columns)
        String[] columnNames = {"Name", "Age", "Airline Number", "Airline Name",
"Origin", "Destination", "Phone Number", "Travel Date"};

        // Create a table model
        DefaultTableModel model = new DefaultTableModel();
        model.setColumnIdentifiers(columnNames);

        // Fetch data from the database and add to the table model
        fetchAndDisplayBookedTickets(model, "");

        // Create the JTable using the model
```

```
        bookedTicketsTable = new JTable(model);
        bookedTicketsTable.setBackground(Color.WHITE);  // Set table background
to white
        bookedTicketsTable.setForeground(Color.BLACK);  // Set table text color
to black

        scrollPane = new JScrollPane(bookedTicketsTable);  // Adding scroll
functionality to the table
        tablePanel.add(scrollPane, BorderLayout.CENTER);

        // Add the table panel to the frame
        add(tablePanel, BorderLayout.CENTER);

        // Back Button Panel
        JPanel backPanel = new JPanel();
        backPanel.setLayout(new BorderLayout());
        backPanel.setBackground(lightGold);  // Set background color to light
gold

        backButton = new JButton("Back");
        backButton.setBackground(ETIHAD_GOLD);  // Set button background to
Etihad Gold
        backButton.setForeground(Color.BLACK);  // Set button text color to black
        backPanel.add(backButton, BorderLayout.CENTER);

        // Add Back Button at the bottom
        add(backPanel, BorderLayout.SOUTH);

        // Action Listener for Search Button
        searchButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String searchQuery = searchField.getText().trim();
                fetchAndDisplayBookedTickets(model, searchQuery);  // Fetch and
update the table based on the search query
            }
        });

        // Action Listener for Back Button
        backButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                dispose();  // Close the current window (BookedTickets)
```

```
                new HomePage();  // Open the HomePage
            }
        });

        setVisible(true);
    }

    // Method to fetch and display booked tickets from the database with an
optional search query
    private void fetchAndDisplayBookedTickets(DefaultTableModel model, String
searchQuery) {
        // Clear the existing table data before adding new rows
        model.setRowCount(0);

        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;

        try {
            // Establish the database connection
            conn = DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD);

            // SQL query to fetch all records from the Airline table (or search
if a query is provided)
            String sql = "SELECT * FROM Airline WHERE name LIKE ? OR
airline_number LIKE ? OR phone_number LIKE ?";

            // Prepare the statement to avoid SQL injection
            PreparedStatement preparedStatement = conn.prepareStatement(sql);
            String searchPattern = "%" + searchQuery + "%";  // Add wildcards for
LIKE search
            preparedStatement.setString(1, searchPattern);
            preparedStatement.setString(2, searchPattern);
            preparedStatement.setString(3, searchPattern);

            // Execute the query
            rs = preparedStatement.executeQuery();

            // Loop through the result set and add each row to the table model
            while (rs.next()) {
                String name = rs.getString("name");
                String age = rs.getString("age");
                String airlineNumber = rs.getString("airline_number");
```

```
                String airlineName = rs.getString("airline_name");
                String origin = rs.getString("origin");
                String destination = rs.getString("destination");
                String phoneNumber = rs.getString("phone_number");
                String travelDate = rs.getString("travel_date");

                // Add the row data to the table model
                model.addRow(new Object[]{name, age, airlineNumber, airlineName,
origin, destination, phoneNumber, travelDate});
            }

        } catch (SQLException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(this, "Error while fetching booked
tickets.", "Database Error", JOptionPane.ERROR_MESSAGE);
        } finally {
            try {
                if (rs != null) rs.close();
                if (stmt != null) stmt.close();
                if (conn != null) conn.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

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

5.Book Tickets

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

public class BookTickets extends JFrame {

    private JTextField nameField, ageField, airlineNumberField, airlineNameField,
originField, destinationField, phoneNumberField, travelDateField;
    private JButton submitButton, backButton;

    // Database connection parameters
    private static final String DB_URL = "jdbc:mysql://localhost:3306/Airline";
// Update with your DB URL
    private static final String DB_USER = "root";  // Update with your DB
username
    private static final String DB_PASSWORD = "1234";  // Update with your DB
password

    public BookTickets() {
        setTitle("Book Tickets");
        setSize(400, 500);  // Increase the height to allow space for buttons at
the bottom
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setLayout(new BorderLayout());

        // Define Etihad Airways colors
        Color etihadGreen = new Color(0, 51, 38); // Dark Green
        Color etihadGold = new Color(245, 210, 115); // Soft Gold
        Color whiteColor = Color.WHITE;

        // Set background color to Etihad's green
        getContentPane().setBackground(etihadGreen);

        // Title label
        JLabel titleLabel = new JLabel("Book Your Ticket",
SwingConstants.CENTER);
        titleLabel.setFont(new Font("Arial", Font.BOLD, 20));
        titleLabel.setForeground(whiteColor); // Set title color to white
```

```
        add(titleLabel, BorderLayout.NORTH);

        // Panel for form fields
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(9, 2, 15, 15)); // Increased space between
rows and columns
        panel.setBackground(etihadGreen); // Set panel background to Etihad green

        // Passenger Name
        JLabel nameLabel = new JLabel("Name:");
        nameLabel.setForeground(whiteColor); // Set text color to white
        panel.add(nameLabel);
        nameField = new JTextField();
        nameField.setBackground(whiteColor); // White background for text field
        nameField.setForeground(etihadGreen); // Dark green text color
        panel.add(nameField);

        // Age
        JLabel ageLabel = new JLabel("Age:");
        ageLabel.setForeground(whiteColor); // Set text color to white
        panel.add(ageLabel);
        ageField = new JTextField();
        ageField.setBackground(whiteColor);
        ageField.setForeground(etihadGreen);
        panel.add(ageField);

        // Airline Number
        JLabel airlineNumberLabel = new JLabel("Airline Number:");
        airlineNumberLabel.setForeground(whiteColor); // Set text color to white
        panel.add(airlineNumberLabel);
        airlineNumberField = new JTextField();
        airlineNumberField.setBackground(whiteColor);
        airlineNumberField.setForeground(etihadGreen);
        panel.add(airlineNumberField);

        // Airline Name
        JLabel airlineNameLabel = new JLabel("Airline Name:");
        airlineNameLabel.setForeground(whiteColor); // Set text color to white
        panel.add(airlineNameLabel);
        airlineNameField = new JTextField();
        airlineNameField.setBackground(whiteColor);
        airlineNameField.setForeground(etihadGreen);
        panel.add(airlineNameField);
```

```
        // Origin
        JLabel originLabel = new JLabel("Origin:");
        originLabel.setForeground(whiteColor); // Set text color to white
        panel.add(originLabel);
        originField = new JTextField();
        originField.setBackground(whiteColor);
        originField.setForeground(etihadGreen);
        panel.add(originField);

        // Destination
        JLabel destinationLabel = new JLabel("Destination:");
        destinationLabel.setForeground(whiteColor); // Set text color to white
        panel.add(destinationLabel);
        destinationField = new JTextField();
        destinationField.setBackground(whiteColor);
        destinationField.setForeground(etihadGreen);
        panel.add(destinationField);

        // Phone Number
        JLabel phoneNumberLabel = new JLabel("Phone Number:");
        phoneNumberLabel.setForeground(whiteColor); // Set text color to white
        panel.add(phoneNumberLabel);
        phoneNumberField = new JTextField();
        phoneNumberField.setBackground(whiteColor);
        phoneNumberField.setForeground(etihadGreen);
        panel.add(phoneNumberField);

        // Travel Date
        JLabel travelDateLabel = new JLabel("Travel Date (yyyy-mm-dd):");
        travelDateLabel.setForeground(whiteColor); // Set text color to white
        panel.add(travelDateLabel);
        travelDateField = new JTextField();
        travelDateField.setBackground(whiteColor);
        travelDateField.setForeground(etihadGreen);
        panel.add(travelDateField);

        // Add the panel to the frame
        add(panel, BorderLayout.CENTER);

        // Panel for buttons
        JPanel buttonPanel = new JPanel();
```

```
        buttonPanel.setLayout(new FlowLayout(FlowLayout.CENTER));  // Align
buttons at the center horizontally
        buttonPanel.setBackground(etihadGreen);  // Ensure buttons have the same
background color

        // Submit Button
        submitButton = new JButton("Submit");
        submitButton.setBackground(etihadGold); // Soft gold background
        submitButton.setForeground(Color.BLACK); // Black text
        submitButton.setFocusPainted(false); // Remove focus border
        buttonPanel.add(submitButton); // Add submit button to button panel

        // Back Button
        backButton = new JButton("Back");
        backButton.setBackground(etihadGold); // Soft gold background
        backButton.setForeground(Color.BLACK); // Black text
        backButton.setFocusPainted(false); // Remove focus border
        buttonPanel.add(backButton); // Add back button to button panel

        // Add the button panel to the bottom of the frame
        add(buttonPanel, BorderLayout.SOUTH);

        // Action Listener for Submit Button
        submitButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Collect the information entered by the user
                String name = nameField.getText();
                String age = ageField.getText();
                String airlineNumber = airlineNumberField.getText();
                String airlineName = airlineNameField.getText();
                String origin = originField.getText();
                String destination = destinationField.getText();
                String phoneNumber = phoneNumberField.getText();
                String travelDate = travelDateField.getText();

                // Store the information in the database
                storeTicketData(name, age, airlineNumber, airlineName, origin,
destination, phoneNumber, travelDate);

                // Display the entered information in a message dialog
                JOptionPane.showMessageDialog(BookTickets.this,
                        "Ticket Booked Successfully!\n" +
```

```
                        "Name: " + name + "\n" +
                        "Age: " + age + "\n" +
                        "Airline Number: " + airlineNumber + "\n" +
                        "Airline Name: " + airlineName + "\n" +
                        "Origin: " + origin + "\n" +
                        "Destination: " + destination + "\n" +
                        "Phone Number: " + phoneNumber + "\n" +
                        "Travel Date: " + travelDate,
                        "Booking Confirmation",
                        JOptionPane.INFORMATION_MESSAGE);
            }
        });

        // Action Listener for Back Button
        backButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // When Back button is clicked, close the current window and open
the HomePage
                dispose(); // Close the BookTickets window
                new HomePage(); // Open the HomePage
            }
        });

        setVisible(true);
    }

    private void storeTicketData(String name, String age, String airlineNumber,
String airlineName, String origin, String destination, String phoneNumber, String
travelDate) {
        Connection conn = null;
        PreparedStatement stmt = null;

        try {
            // Establish the database connection
            conn = DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD);

            // Prepare the SQL query
            String sql = "INSERT INTO Airline (name, age, airline_number,
airline_name, origin, destination, phone_number, travel_date) VALUES
(?, ?, ?, ?, ?, ?, ?, ?)";
            stmt = conn.prepareStatement(sql);
```

```
            // Set the parameters in the query
            stmt.setString(1, name);
            stmt.setString(2, age);
            stmt.setString(3, airlineNumber);
            stmt.setString(4, airlineName);
            stmt.setString(5, origin);
            stmt.setString(6, destination);
            stmt.setString(7, phoneNumber);
            stmt.setString(8, travelDate);

            // Execute the query to insert the data into the database
            stmt.executeUpdate();

        } catch (SQLException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(this, "Error while storing ticket
data.", "Database Error", JOptionPane.ERROR_MESSAGE);
        } finally {
            try {
                if (stmt != null) stmt.close();
                if (conn != null) conn.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

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

6.Cancle Tickets

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

public class CancelTicket extends JFrame {

    private JTextField nameField, phoneNumberField, flightNumberField;
    private JButton deleteButton, backButton;

    // Database connection parameters
    private static final String DB_URL = "jdbc:mysql://localhost:3306/Airline";
// Update with your DB URL
    private static final String DB_USER = "root";  // Update with your DB
username
    private static final String DB_PASSWORD = "1234";  // Update with your DB
password

    public CancelTicket() {
        setTitle("Cancel Ticket");
        setSize(400, 250);  // Adjusted window size
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setLayout(new BorderLayout());

        // Define Etihad Airways colors
        Color etihadGreen = new Color(0, 51, 38); // Dark Green
        Color etihadGold = new Color(245, 210, 115); // Soft Gold
        Color whiteColor = Color.WHITE;

        // Set background color to Etihad's green
        getContentPane().setBackground(etihadGreen);

        // Title label
        JLabel titleLabel = new JLabel("Cancel Booked Ticket",
SwingConstants.CENTER);
        titleLabel.setFont(new Font("Arial", Font.BOLD, 20));
        titleLabel.setForeground(whiteColor); // Set title color to white
        add(titleLabel, BorderLayout.NORTH);

        // Panel for search fields (Name, Phone Number, Flight Number)
```

```
        JPanel searchPanel = new JPanel();
        searchPanel.setLayout(new GridLayout(4, 2, 10, 10)); // 4 rows, 2 columns
        searchPanel.setBackground(etihadGreen); // Set panel background to Etihad
green

        // Name
        JLabel nameLabel = new JLabel("Name:");
        nameLabel.setForeground(whiteColor); // Set text color to white for Name
label
        searchPanel.add(nameLabel);
        nameField = new JTextField();
        nameField.setBackground(whiteColor); // White background for text field
        nameField.setForeground(Color.BLACK); // White text color
        searchPanel.add(nameField);

        // Phone Number
        JLabel phoneNumberLabel = new JLabel("Phone Number:");
        phoneNumberLabel.setForeground(whiteColor); // Set text color to white
for Phone Number label
        searchPanel.add(phoneNumberLabel);
        phoneNumberField = new JTextField();
        phoneNumberField.setBackground(whiteColor); // White background for text
field
        phoneNumberField.setForeground(Color.BLACK); // White text color
        searchPanel.add(phoneNumberField);

        // Flight Number
        JLabel flightNumberLabel = new JLabel("Flight Number:");
        flightNumberLabel.setForeground(whiteColor); // Set text color to white
for Flight Number label
        searchPanel.add(flightNumberLabel);
        flightNumberField = new JTextField();
        flightNumberField.setBackground(whiteColor); // White background for text
field
        flightNumberField.setForeground(Color.BLACK); // White text color
        searchPanel.add(flightNumberField);

        // Add the search panel to the frame
        add(searchPanel, BorderLayout.CENTER);

        // Panel for Buttons (Back and Delete)
        JPanel buttonPanel = new JPanel();
        buttonPanel.setLayout(new FlowLayout(FlowLayout.CENTER));
```

```
        buttonPanel.setBackground(etihadGreen);  // Set panel background to
Etihad green

        // Back Button
        backButton = new JButton("Back");
        backButton.setBackground(etihadGold); // Soft gold background
        backButton.setForeground(Color.BLACK); // White text color
        backButton.setFocusPainted(false); // Remove focus border
        buttonPanel.add(backButton);

        // Delete Button
        deleteButton = new JButton("Delete");
        deleteButton.setBackground(etihadGold); // Soft gold background
        deleteButton.setForeground(Color.BLACK); // White text color
        deleteButton.setFocusPainted(false); // Remove focus border
        buttonPanel.add(deleteButton);

        add(buttonPanel, BorderLayout.SOUTH);

        // Action listener for Back button
        backButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Close the current window and return to the HomePage
                dispose(); // Close the CancelTicket window
                new HomePage(); // Open the HomePage
            }
        });

        // Action listener for Delete button
        deleteButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String name = nameField.getText().trim();
                String phoneNumber = phoneNumberField.getText().trim();
                String flightNumber = flightNumberField.getText().trim();

                // Check if all fields are filled
                if (name.isEmpty() || phoneNumber.isEmpty() ||
flightNumber.isEmpty()) {
                    JOptionPane.showMessageDialog(CancelTicket.this, "Please
enter all search fields.",
```

```
                                                "Incomplete Fields",
JOptionPane.WARNING_MESSAGE);
                } else {
                    deleteTicket(name, phoneNumber, flightNumber);  // Delete the
ticket from the database
                }
            }
        });

        setVisible(true);
    }

    // Method to delete a ticket from the database using Name, Phone Number, and
Flight Number
    private void deleteTicket(String name, String phoneNumber, String
flightNumber) {
        Connection conn = null;
        PreparedStatement stmt = null;

        try {
            // Establish the database connection
            conn = DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD);

            // SQL query to delete the record based on Name, Phone Number, and
Flight Number
            String sql = "DELETE FROM Airline WHERE name = ? AND phone_number = ?
AND airline_number = ?";
            stmt = conn.prepareStatement(sql);
            stmt.setString(1, name);
            stmt.setString(2, phoneNumber);
            stmt.setString(3, flightNumber);

            // Execute the query
            int rowsAffected = stmt.executeUpdate();

            if (rowsAffected > 0) {
                JOptionPane.showMessageDialog(this, "Ticket deleted
successfully.", "Deletion Success", JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(this, "No matching ticket found.",
"Deletion Error", JOptionPane.ERROR_MESSAGE);
            }
```

```
        } catch (SQLException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(this, "Error while deleting ticket.",
"Database Error", JOptionPane.ERROR_MESSAGE);
        } finally {
            try {
                if (stmt != null) stmt.close();
                if (conn != null) conn.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

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

7.Search Flights

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

public class SearchFlights extends JFrame {

    private JTextField flightNumberField;
    private JTextArea resultsArea;
    private JButton searchButton, backButton;

    // Simulating the flight data (you can replace this with actual data
retrieval logic)
    private List<Flight> availableFlights;

    public SearchFlights() {
        setTitle("Search Flights by Flight Number");
        setSize(500, 500);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setLayout(new BorderLayout());

        // Sample data of available flights (you can remove this when integrating
with a database or real data)
        availableFlights = new ArrayList<>();
        availableFlights.add(new Flight("AI202", "Delhi", "Mumbai", "2024-11-25",
"08:30 AM"));
        availableFlights.add(new Flight("AI203", "Mumbai", "Chennai", "2024-11-
26", "10:30 AM"));
        availableFlights.add(new Flight("AI204", "Delhi", "Chennai", "2024-11-
27", "07:30 AM"));

        // Title label
        JLabel titleLabel = new JLabel("Search Flights by Flight Number",
SwingConstants.CENTER);
        titleLabel.setFont(new Font("Arial", Font.BOLD, 20));
        add(titleLabel, BorderLayout.NORTH);

        // Panel for search input
        JPanel inputPanel = new JPanel(new GridLayout(2, 2, 10, 10));
```

```
        // Flight Number input
        inputPanel.add(new JLabel("Flight Number:"));
        flightNumberField = new JTextField();
        inputPanel.add(flightNumberField);

        // Search button
        searchButton = new JButton("Search");
        searchButton.setPreferredSize(new Dimension(100, 30));  // Set smaller
button size
        inputPanel.add(searchButton);
        inputPanel.add(new JLabel("")); // Placeholder for layout

        add(inputPanel, BorderLayout.CENTER);

        // Results area
        resultsArea = new JTextArea();
        resultsArea.setEditable(false);
        add(new JScrollPane(resultsArea), BorderLayout.CENTER);

        // Panel for back button (at the bottom)
        JPanel bottomPanel = new JPanel();
        bottomPanel.setLayout(new FlowLayout(FlowLayout.CENTER, 10, 10));

        // Back button
        backButton = new JButton("Back");
        backButton.setPreferredSize(new Dimension(100, 30));  // Set smaller
button size
        bottomPanel.add(backButton);

        add(bottomPanel, BorderLayout.SOUTH);

        // Action Listener for Search Button
        searchButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String flightNumber = flightNumberField.getText().trim();
                searchFlightByNumber(flightNumber);
            }
        });

        // Action Listener for Back Button
```

```
        backButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Close the current window (SearchFlights) and go back to the
HomePage
                dispose(); // Close the SearchFlights window
                new HomePage(); // Open the HomePage (or another screen)
            }
        });

        setVisible(true);
    }

    // Method to search flights based on flight number
    private void searchFlightByNumber(String flightNumber) {
        StringBuilder results = new StringBuilder("Flight Information:\n\n");
        boolean found = false;

        for (Flight flight : availableFlights) {
            if (flight.getFlightNumber().equalsIgnoreCase(flightNumber)) {
                results.append(flight.toString()).append("\n");
                found = true;
                break;
            }
        }

        if (!found) {
            results.append("No flight available with the specified flight
number.");
        }

        resultsArea.setText(results.toString());
    }

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

    // Inner class to represent flight data (replace with actual database or data
source)
    public static class Flight {
        private String flightNumber;
        private String origin;
```

```
        private String destination;
        private String date;
        private String time;

        public Flight(String flightNumber, String origin, String destination,
String date, String time) {
            this.flightNumber = flightNumber;
            this.origin = origin;
            this.destination = destination;
            this.date = date;
            this.time = time;
        }

        public String getFlightNumber() {
            return flightNumber;
        }

        public String getOrigin() {
            return origin;
        }

        public String getDestination() {
            return destination;
        }

        public String getDate() {
            return date;
        }

        public String getTime() {
            return time;
        }

        @Override
        public String toString() {
            return "Flight Number: " + flightNumber + "\n" +
                   "Origin: " + origin + "\n" +
                   "Destination: " + destination + "\n" +
                   "Date: " + date + "\n" +
                   "Time: " + time;
        }
    }
}
```

8.APP

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

public class App {
    public static void main(String[] args) {
        System.out.println("Hello, world! This is the main class.");

        // Example usage of DBConnection
        try (Connection connection = DBConnection.getConnection()) {
            if (connection != null) {
                System.out.println("Connected to the database!");

                // Example SQL query
                String query = "SELECT * FROM your_table_name"; // Replace with
your table name
                Statement statement = connection.createStatement();
                ResultSet resultSet = statement.executeQuery(query);

                // Process and display the query result
                while (resultSet.next()) {
                    System.out.println("Column1: " +
resultSet.getString("column1")); // Update column names as needed
                }

                // Close the statement and result set
                resultSet.close();
                statement.close();
            } else {
                System.out.println("Failed to connect to the database.");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

9.DB connection

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

public class DBConnection {

    // Database URL, username, and password
    private static final String URL = "jdbc:mysql://localhost:3306/Airline"; //
Update with your database name
    private static final String USER = "root"; // Update with your MySQL username
    private static final String PASSWORD = "1234"; // Update with your MySQL
password

    // Private constructor to prevent instantiation
    private DBConnection() {}

    // Method to get a connection to the database
    public static Connection getConnection() {
        Connection connection = null;
        try {
            // Establish the connection
            connection = DriverManager.getConnection(URL, USER, PASSWORD);
            System.out.println("Database connection established.");
        } catch (SQLException e) {
            e.printStackTrace();
            System.out.println("Failed to connect to the database.");
        }
        return connection;
    }

    // Method to close the connection
    public static void closeConnection(Connection connection) {
        if (connection != null) {
            try {
                connection.close();
                System.out.println("Database connection closed.");
            } catch (SQLException e) {
                e.printStackTrace();
                System.out.println("Failed to close the database connection.");
            }
```

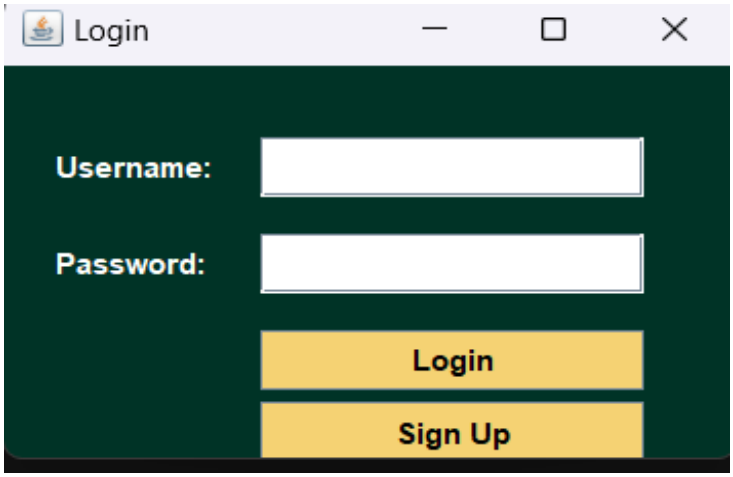
```
        }
    }}
```

Chapter 4.1: SQL CODE

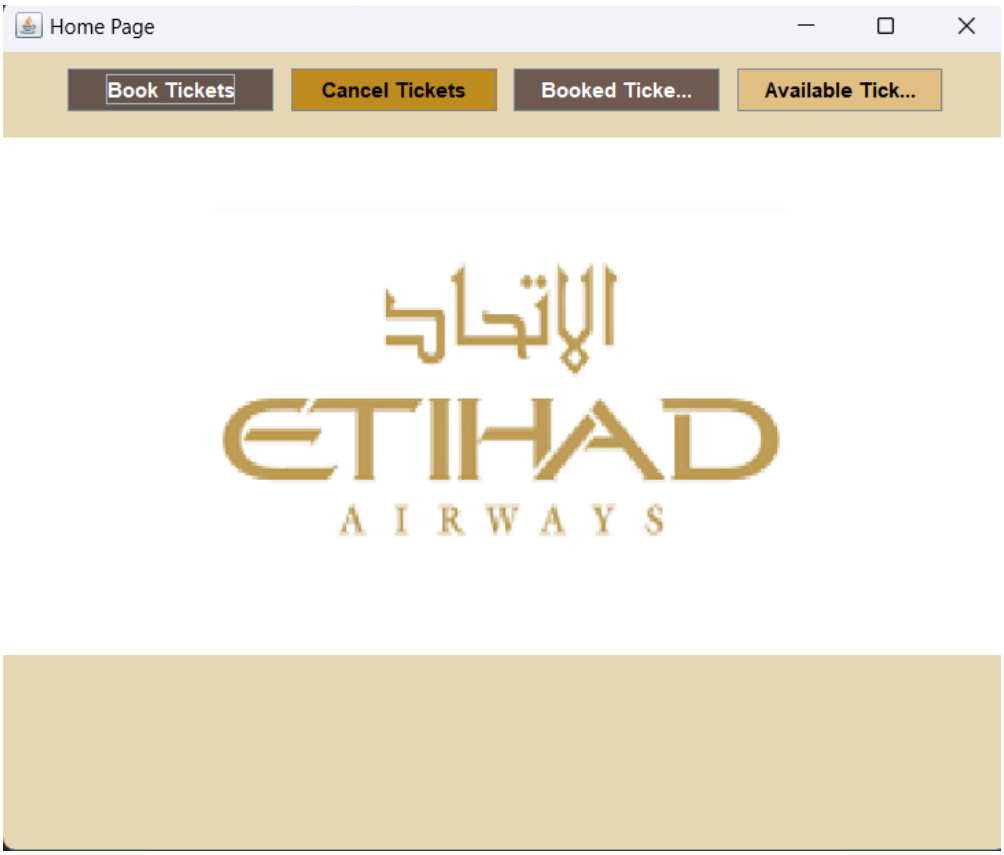
```
create database Airline;
USE Airline;
CREATE TABLE Airline (
   id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR (100),
   age INT,
   airline_number VARCHAR (50),
   airline_name VARCHAR (100),
   origin VARCHAR (100),
   destination VARCHAR (100),
   phone_number VARCHAR (15),
   travel_date DATE
);
Select * from Airline;
```

Chapter 5: RESULT AND DISCUSSION

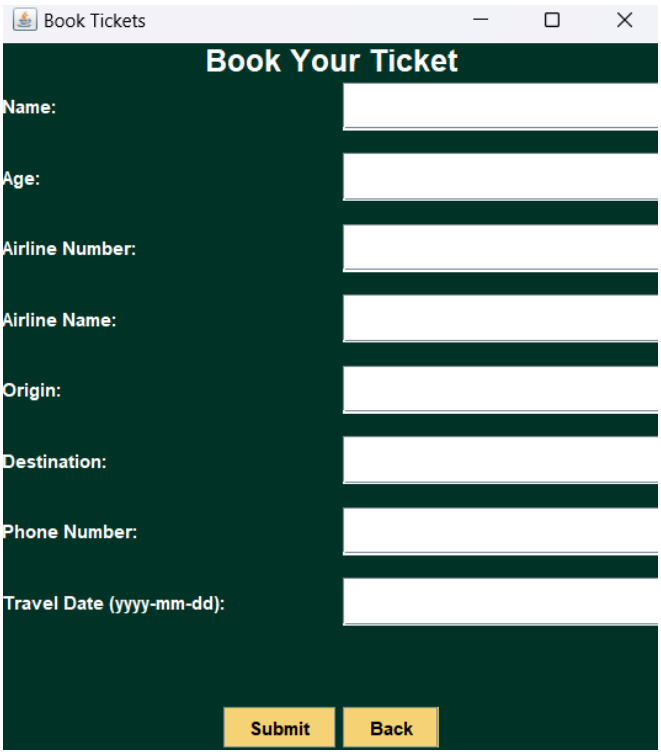
1. **Login Page:**



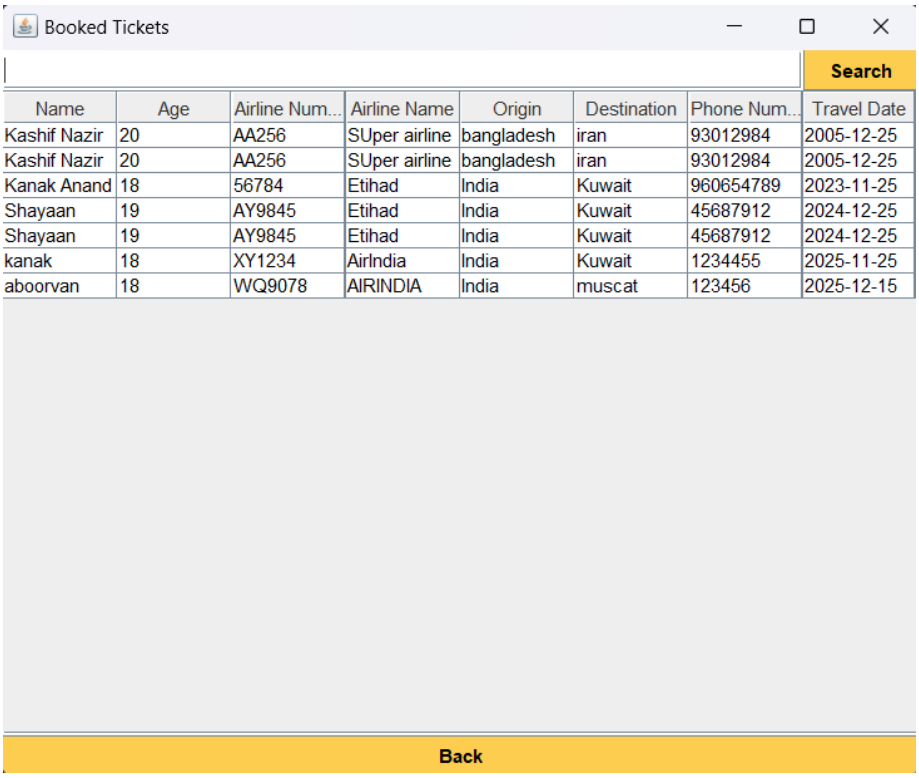
2. **Home Page**



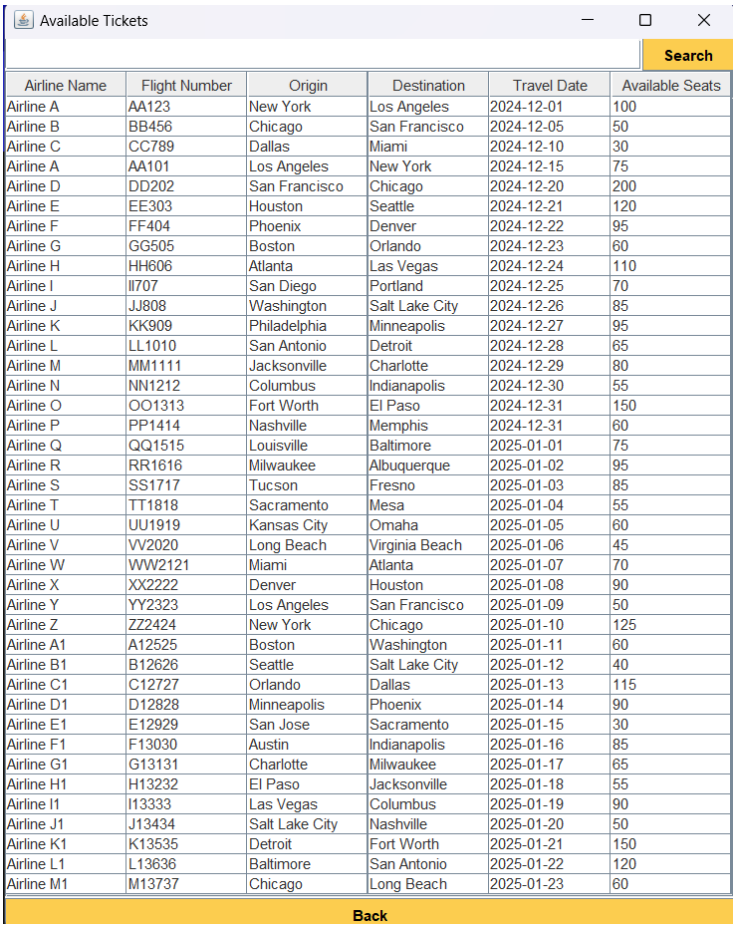
3.Booking Page



4. Booked Ticket Page

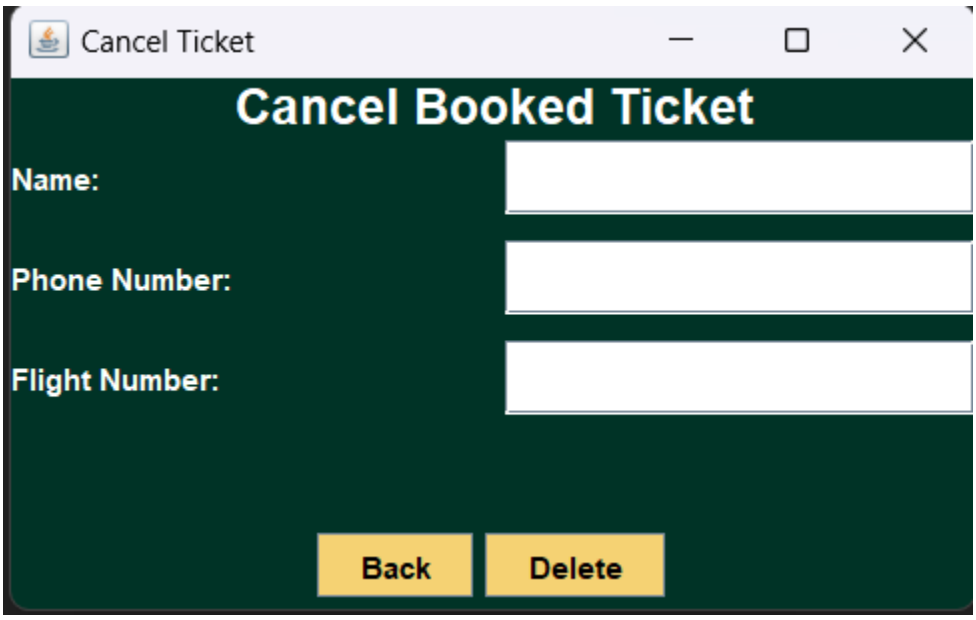


5. Available Ticket



Airline Name	Flight Number	Origin	Destination	Travel Date	Available Seats
Airline A	AA123	New York	Los Angeles	2024-12-01	100
Airline B	BB456	Chicago	San Francisco	2024-12-05	50
Airline C	CC789	Dallas	Miami	2024-12-10	30
Airline A	AA101	Los Angeles	New York	2024-12-15	75
Airline D	DD202	San Francisco	Chicago	2024-12-20	200
Airline E	EE303	Houston	Seattle	2024-12-21	120
Airline F	FF404	Phoenix	Denver	2024-12-22	95
Airline G	GG505	Boston	Orlando	2024-12-23	60
Airline H	HH606	Atlanta	Las Vegas	2024-12-24	110
Airline I	II707	San Diego	Portland	2024-12-25	70
Airline J	JJ808	Washington	Salt Lake City	2024-12-26	85
Airline K	KK909	Philadelphia	Minneapolis	2024-12-27	95
Airline L	LL1010	San Antonio	Detroit	2024-12-28	65
Airline M	MM1111	Jacksonville	Charlotte	2024-12-29	80
Airline N	NN1212	Columbus	Indianapolis	2024-12-30	55
Airline O	OO1313	Fort Worth	El Paso	2024-12-31	150
Airline P	PP1414	Nashville	Memphis	2024-12-31	60
Airline Q	QQ1515	Louisville	Baltimore	2025-01-01	75
Airline R	RR1616	Milwaukee	Albuquerque	2025-01-02	95
Airline S	SS1717	Tucson	Fresno	2025-01-03	85
Airline T	TT1818	Sacramento	Mesa	2025-01-04	55
Airline U	UU1919	Kansas City	Omaha	2025-01-05	60
Airline V	VV2020	Long Beach	Virginia Beach	2025-01-06	45
Airline W	WW2121	Miami	Atlanta	2025-01-07	70
Airline X	XX2222	Denver	Houston	2025-01-08	90
Airline Y	YY2323	Los Angeles	San Francisco	2025-01-09	50
Airline Z	ZZ2424	New York	Chicago	2025-01-10	125
Airline A1	A12525	Boston	Washington	2025-01-11	60
Airline B1	B12626	Seattle	Salt Lake City	2025-01-12	40
Airline C1	C12727	Orlando	Dallas	2025-01-13	115
Airline D1	D12828	Minneapolis	Phoenix	2025-01-14	90
Airline E1	E12929	San Jose	Sacramento	2025-01-15	30
Airline F1	F13030	Austin	Indianapolis	2025-01-16	85
Airline G1	G13131	Charlotte	Milwaukee	2025-01-17	65
Airline H1	H13232	El Paso	Jacksonville	2025-01-18	55
Airline I1	I13333	Las Vegas	Columbus	2025-01-19	90
Airline J1	J13434	Salt Lake City	Nashville	2025-01-20	50
Airline K1	K13535	Detroit	Fort Worth	2025-01-21	150
Airline L1	L13636	Baltimore	San Antonio	2025-01-22	120
Airline M1	M13737	Chicago	Long Beach	2025-01-23	60

6. Cancel Ticket



Chapter 6: Conclusion

An effective airline management system is crucial for ensuring smooth operations, enhancing customer satisfaction, and maintaining profitability in the highly competitive aviation industry. By integrating key components such as booking, payment processing, flight scheduling, staff management, and real-time status updates, airlines can deliver a seamless travel experience.

Additionally, leveraging technology like data analytics, automation, and modern user interfaces helps optimize resources, reduce costs, and improve operational efficiency. Such systems also support compliance with aviation regulations and adapt to dynamic industry demands.

In conclusion, a robust airline management system not only improves customer satisfaction but also fosters long-term business sustainability in a rapidly evolving market.

Chapter 7: REFERENCE

7.1 REFERENCES

[1] https://stackoverflow.com

[2] https://www.youtube.com/watch?v=OGP2R29vzAw

[3] https://www.youtube.com/watch?v=jHSBrX8lLWk