End of term project : Airline Management System Using Java Swing and MySQL

School of Digital Engineering and Artificial Intelligence: 3rd Year Big Data

School year : 2024-2025

DONE BY :

- **HANAE TALEBI**
- **IMANE MALIKI**

I.Introduction :

In a world where managing flights and passengers is becoming increasingly complex, computer systems play a vital role in automating and simplifying these processes. The **Airline Management System** project is a desktop application designed to address these specific needs. Developed using **Java Swing** for the user interface and **MySQL** for data management, this application provides a comprehensive solution for airlines, enabling them to efficiently handle their daily operations.

Project Objectives:

The primary objective of this project is to provide a user-friendly and efficient interface that allows:

1. **Passenger Management**: Add, update, and search for passenger information.
2. **Flight Management**: Organize flight schedules, routes, and availability.
3. **Ticket Booking**: Facilitate quick and secure ticket reservations for passengers.
4. **Ticket Cancellation**: Simplify the processes of ticket cancellation and refund.
5. **Secure Access**: Ensure that only authorized personnel can access the system through an authentication module.

Technologies Used :

- **Java Swing:** Provides an interactive and ergonomic graphical user interface for the application.
- **MySQL:** Manages the data related to users, passengers, flights, and bookings.
- **JDBC (Java Database Connectivity):** Serves as the interface between the application and the data .

Significance of the Project :

The proposed system automates processes that would otherwise require significant time and effort if done manually. This automation enables:

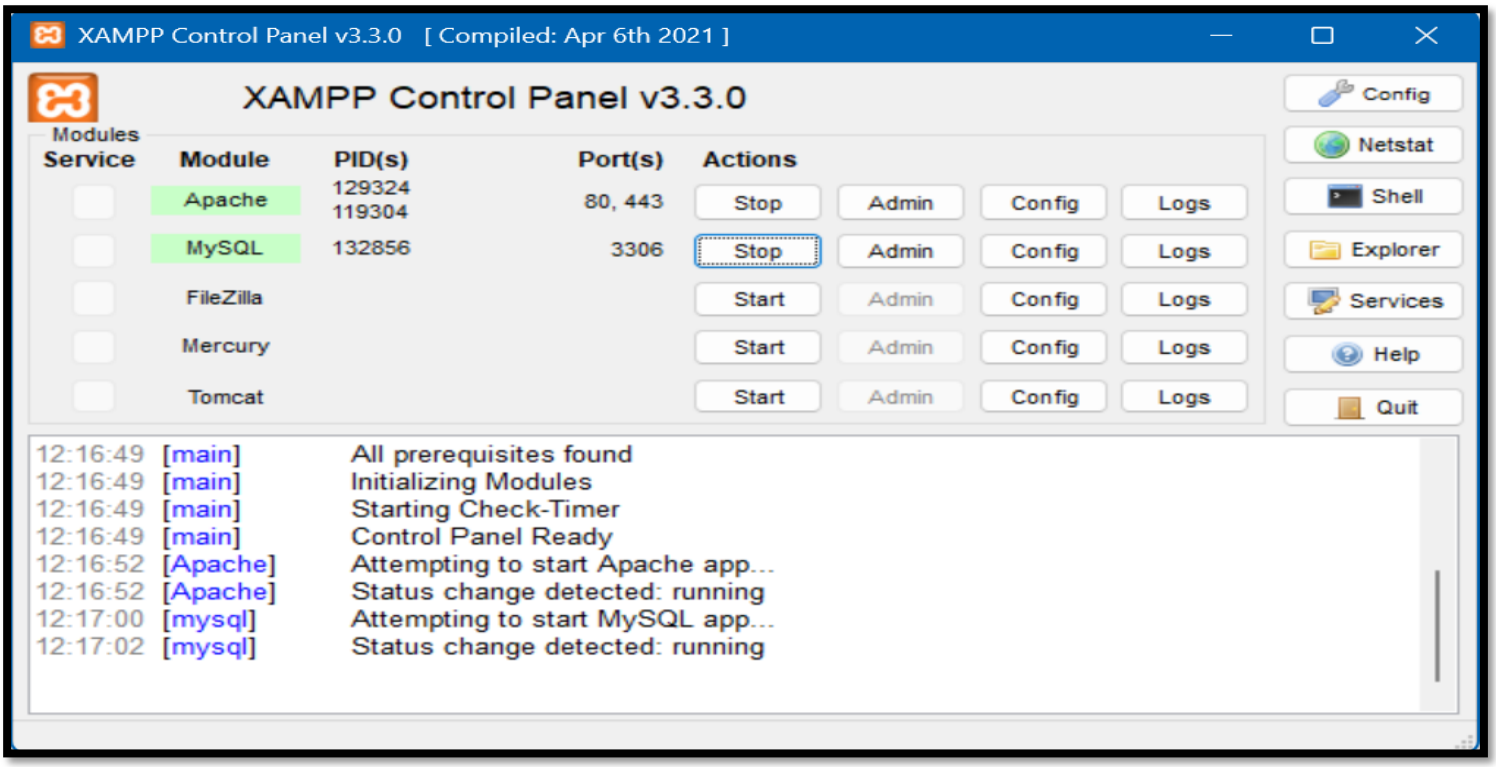
- Better organization of data.
- Reduction of human errors in bookings and cancellations.
- Improved user experience for airline staff.

This report delves into each part of the application in detail, illustrating its core functionalities with screenshots and in-depth explanations. The following sections describe the individual components of the application, their roles in the overall system, and how they contribute to the efficiency of airline management processes .

Table of Contents :

II. Database Structure and Setup :

- The backbone of the Airline Management System is the database, which stores all essential information in a structured format. Using **MySQL**, the database is designed with tables that represent key entities in the airline system, such as passengers, flights, and bookings, and hosted locally through the **XAMPP** platform , which is an open-source cross-platform web server solution that bundles Apache, MySQL, PHP, and Perl, making it easy to set up and manage and role the databases for local development.



Serveur : 127.0.0.1 » Base de données : airlinedb

Structure | SQL | Rechercher | Requête | Exporter | Importer | Opérations | Privilèges | Procédures stockées | Plus

Filtres

Contenant le mot :

Table	Action	Lignes	Type	Interclassement	Taille	Perte
bookingtbl	Parcourir Structure Rechercher Insérer Vider Supprimer	1	InnoDB	utf8mb4_general_ci	16,0 kio	-
cancellationtbl	Parcourir Structure Rechercher Insérer Vider Supprimer	6	InnoDB	utf8mb4_general_ci	16,0 kio	-
flighttbl	Parcourir Structure Rechercher Insérer Vider Supprimer	4	InnoDB	utf8mb4_general_ci	16,0 kio	-
passengerstbl	Parcourir Structure Rechercher Insérer Vider Supprimer	4	InnoDB	utf8mb4_general_ci	16,0 kio	-
4 tables	**Somme**	**15**	**InnoDB**	**utf8mb4_general_ci**	**64,0 kio**	**0 o**

Tout cocher Avec la sélection :

Key Database Tables and Their Roles:

1. Passengers Table:

- Maintains records of passenger information.
- Includes details such as name, contact information, and identification.
- Columns: Pid, PName, PNat , PGen , Pphone, PAdd.

Serveur : 127.0.0.1 » Base de données : airlinedb » Table : passengerstbl

Parcourir | Structure | SQL | Rechercher | Insérer | Exporter | Importer | Privilèges | Opérations | Suivi | Plus

Affichage des lignes 0 - 3 (total de 4, traitement en 0,0004 seconde(s).)

```
SELECT * FROM `passengerstbl`
```

Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]

Tout afficher | Nombre de lignes : 25 Filtrer les lignes: Chercher dans cette table Trier par clé : Aucun(e)

Options supplémentaires

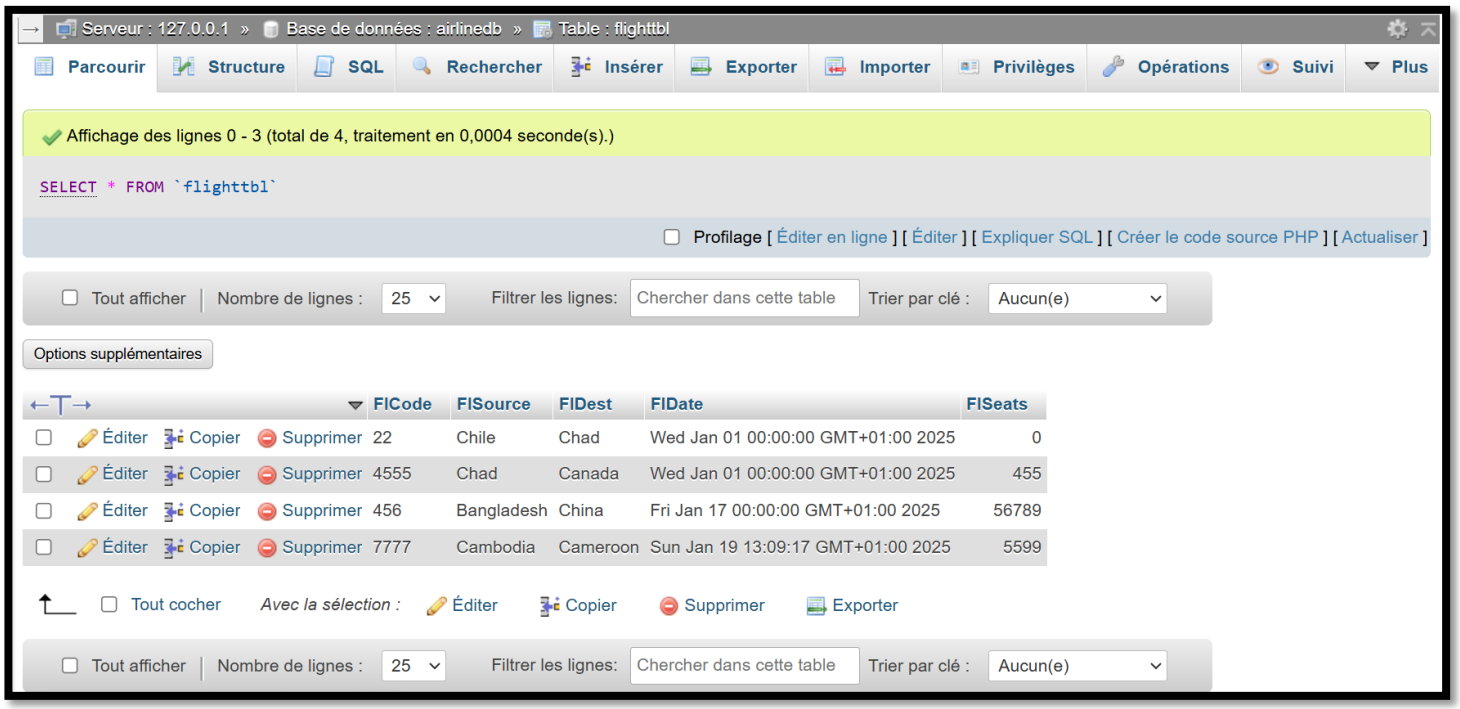
	Pid	PName	PNat	PGen	PPass	PAdd	Pphone
Éditer Copier Supprimer	1	mounia	Afghan	Male	1234	FFF	06353542
Éditer Copier Supprimer	3	KOKO	Argentine	Female	16666	GG	06444
Éditer Copier Supprimer	4	QSER	Bolivian	Female	3333	MIKRN	0645454
Éditer Copier Supprimer	5	MO	Afghan	Female	8999	MEKNES	0699232144

Tout cocher *Avec la sélection :* Éditer Copier Supprimer Exporter

Tout afficher | Nombre de lignes : 25 Filtrer les lignes: Chercher dans cette table Trier par clé : Aucun(e)

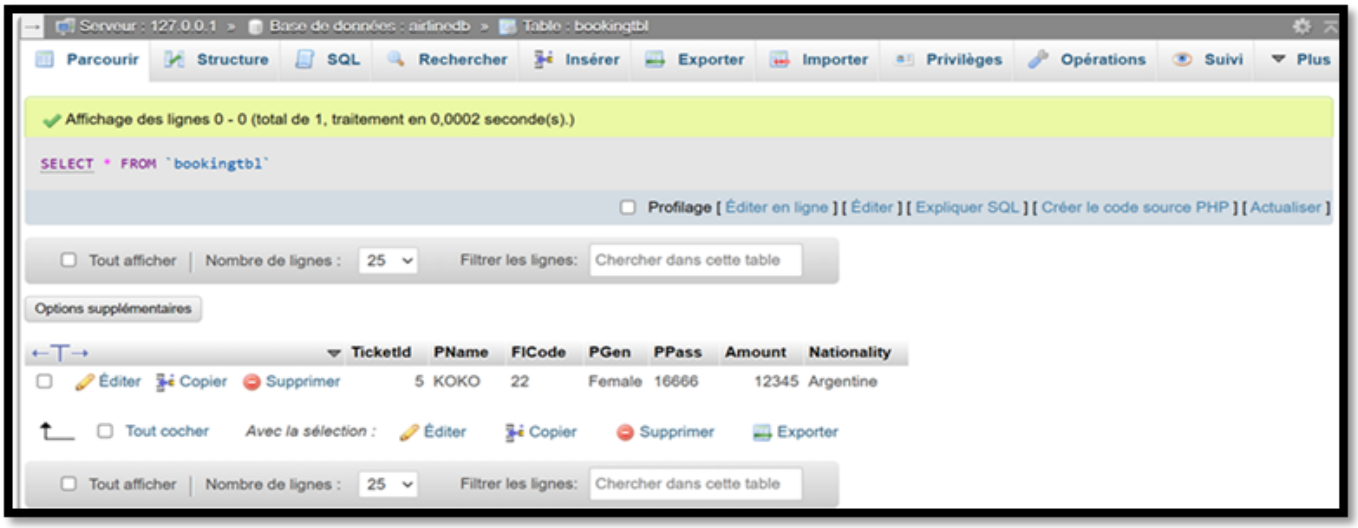
2. Flights Table:

- Contains flight schedules and details such as routes, timings, and availability.
- Helps in searching and managing flight information.
- Columns: FICode , FISource, FIDest, FIDate , FISeats, .



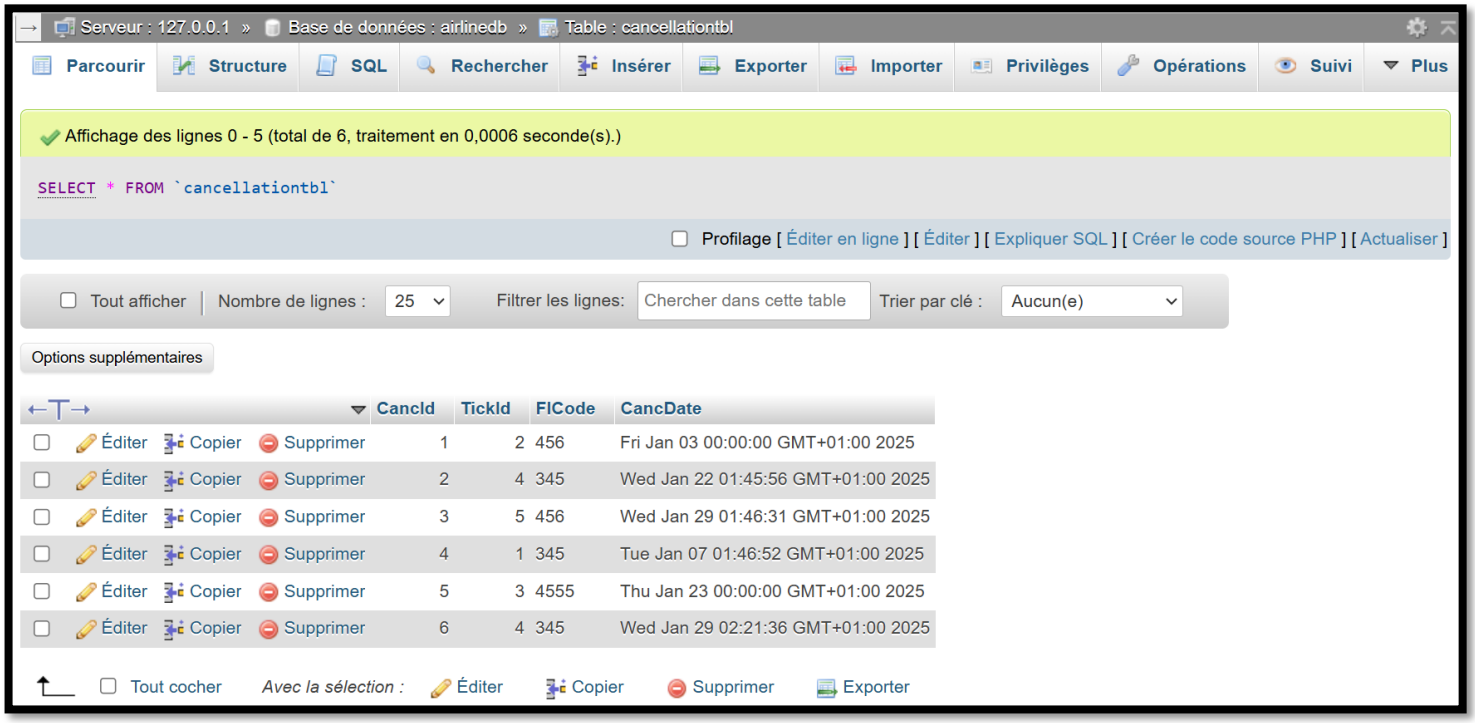
3. Bookings Table:

- Links passengers with flights through ticket reservations.
- Tracks booking status, payment details, and ticket references.
- Columns: booking_id, passenger_id, flight_id, date, status, payment_amount.



4. **Cancellations Table** :

- Tracks tickets that have been canceled.
- Helps in maintaining accurate records for refunds or cancellations.
- Columns: CanclId, Tickid, FICode, CancDate .



Integration with Java :

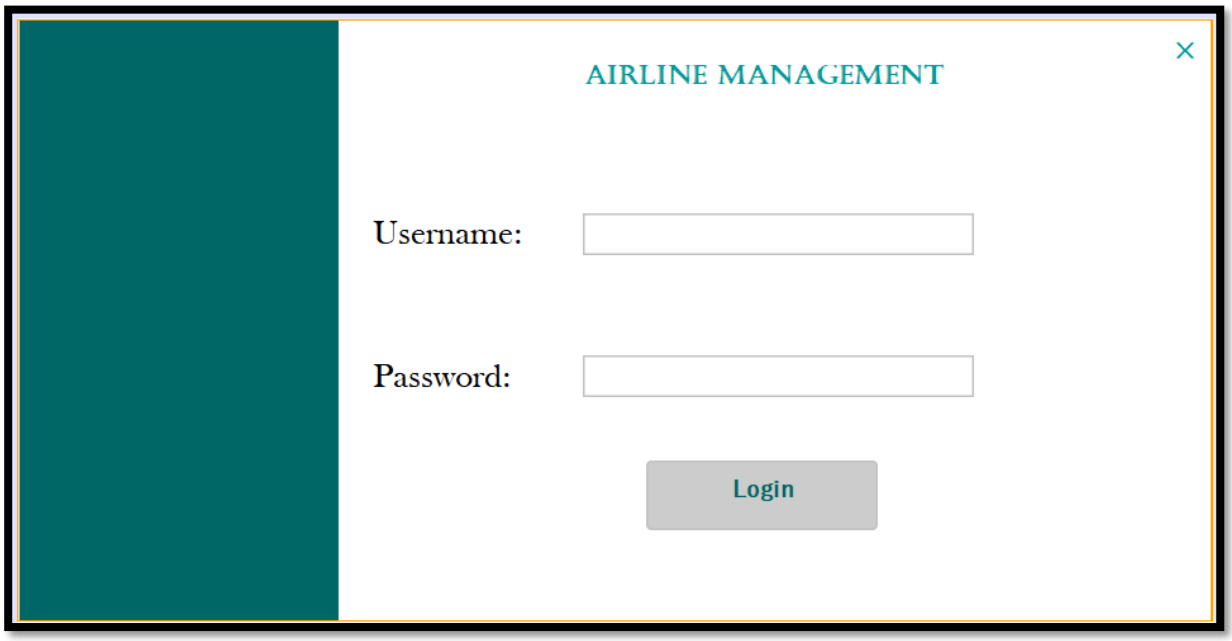
- **JDBC (Java Database Connectivity)**:

- The application uses JDBC to connect Java Swing forms with the MySQL database.
- Allows dynamic queries for operations such as login validation, booking tickets, and fetching flight schedules.
- Example: A SQL query like SELECT * FROM flights WHERE origin=? AND destination=? fetches available flights dynamically based on user input.

III. Login Page :

The **Login Page** is a critical module of the Airline Management System, ensuring that only authorized users can access the application. This page acts as the first layer of security, validating user credentials against stored records in the database. It is designed to protect sensitive data and restrict unauthorized actions, making it an essential component of the system.

User Interface Overview :



1. **Username Field**: Allows users to input their username.
2. **Password Field**: Provides a secure input box for passwords, masking the entered characters.
3. **Login Button**: Initiates the authentication process.
4. **Error Messages**: Displays appropriate feedback for invalid login attempts, such as "Wrong username and password."

Code :

```java
package com.mycompany.project;
import javax.swing.JOptionPane;

public class login extends javax.swing.JFrame {

    public login() {
        initComponents();
    }
    @SuppressWarnings("unchecked")
    Generated Code

    private void UnameTbActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void LoginBtnMouseClicked(java.awt.event.MouseEvent evt) {
        if(UnameTb.getText().isEmpty() || PasswordTb.getText().isEmpty())
        {
            JOptionPane.showMessageDialog(this,"enter username and password");
        }else if(UnameTb.getText().equals("Admin") && PasswordTb.getText().equals("12345"))
        {
            new mainform().setVisible(true);
            this.dispose();

        }else{
            JOptionPane.showMessageDialog(this,"wrong username and password");
            UnameTb.setText(null);
            PasswordTb.setText(null);
        }
    }
```

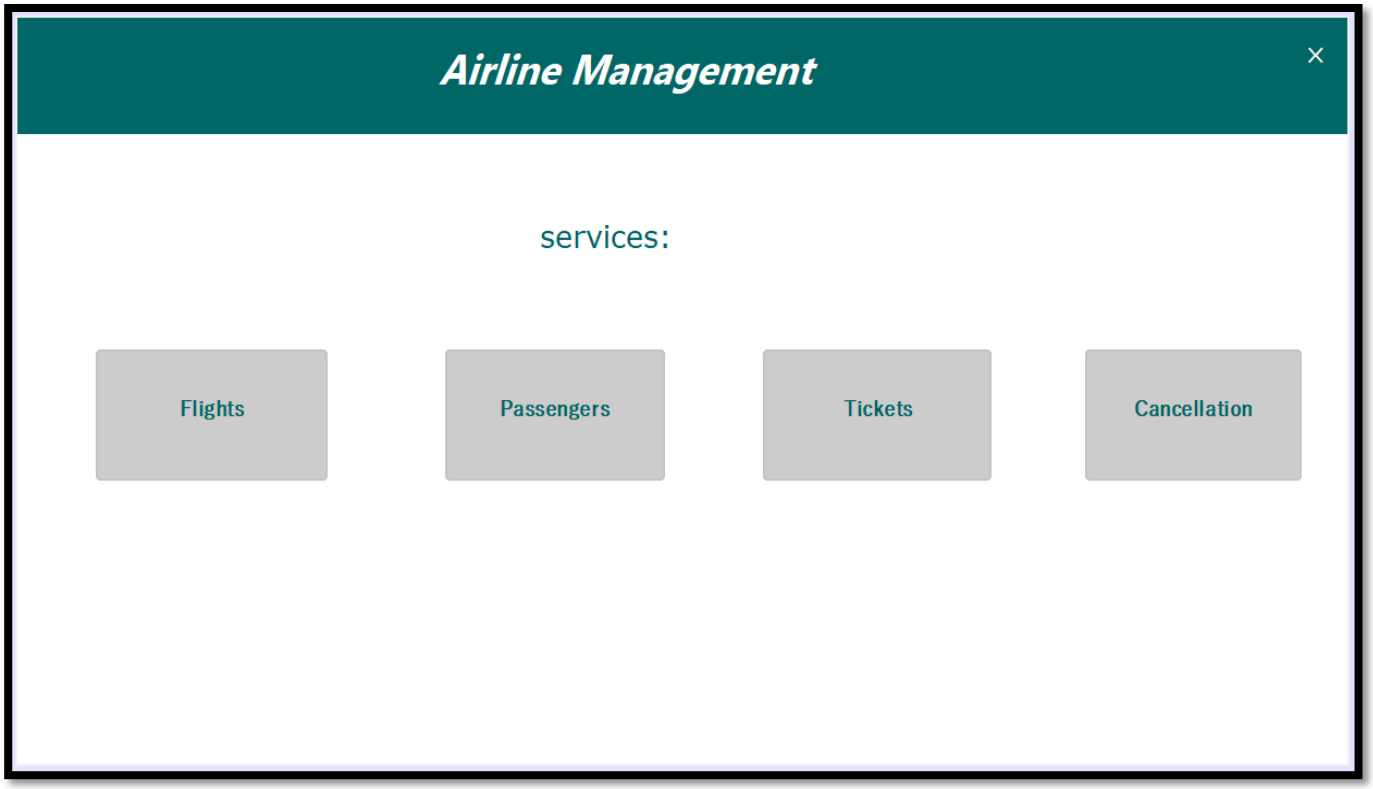
```java
    private void jButton3MouseClicked(java.awt.event.MouseEvent evt) {
        new ticketbooking().setVisible(true);
        this.dispose();
    }

    private void jButton4MouseClicked(java.awt.event.MouseEvent evt) {
        new Cancelation().setVisible(true);
        this.dispose();
    }

    private void jLabel8MouseClicked(java.awt.event.MouseEvent evt) {
        System.exit(0);
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        Look and feel setting code (optional)
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new mainform().setVisible(true);
            }
        });
    }
    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton4;
    private javax.swing.JLabel jLabel2;
```

IV. Main Dashboard :

- The **Main Dashboard** serves as the central hub of the Airline Management System. It provides an organized and intuitive interface, enabling users to navigate between various modules of the application. Acting as the first screen after login, the dashboard offers a high-level overview of the system and serves as the gateway to all critical features such as passenger management, flight scheduling, ticket booking, and cancellations.

User Interface Overview :

- Buttons or menu items for accessing modules like:
  - **Passengers**
  - **Flights**
  - **Tickets**
  - **Cancellation**



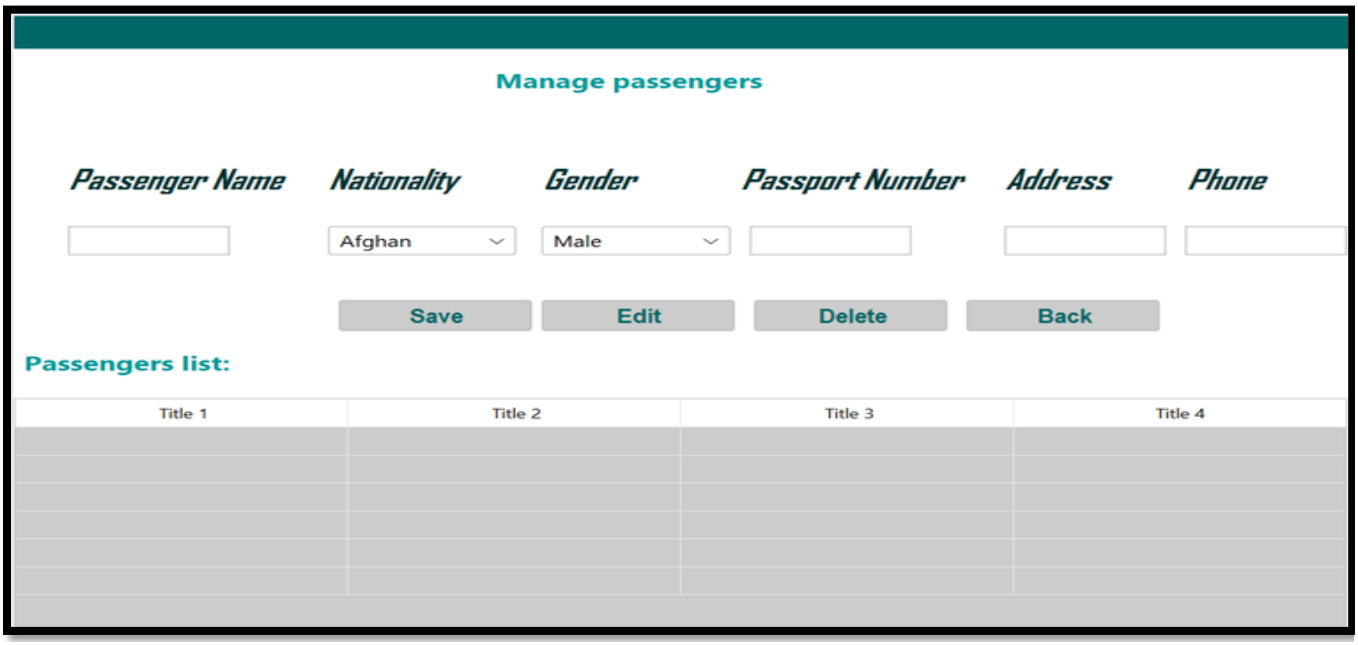
The Code :

```
package com.mycompany.project;
public class mainform extends javax.swing.JFrame {
    public mainform() {
        initComponents();
    }
    @SuppressWarnings("unchecked")
    Generated Code

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jButton2MouseClicked(java.awt.event.MouseEvent evt) {
        new Passengers().setVisible(true);
        this.dispose();
    }

    private void jButton1MouseClicked(java.awt.event.MouseEvent evt) {
        new flights().setVisible(true);
        this.dispose();
    }

    private void jButton3MouseClicked(java.awt.event.MouseEvent evt) {
```

```java
    private void jButton3MouseClicked(java.awt.event.MouseEvent evt) {
        new ticketbooking().setVisible(true);
        this.dispose();
    }

    private void jButton4MouseClicked(java.awt.event.MouseEvent evt) {
        new Cancelation().setVisible(true);
        this.dispose();
    }

    private void jLabel8MouseClicked(java.awt.event.MouseEvent evt) {
        System.exit(0);
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        Look and feel setting code (optional)
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new mainform().setVisible(true);
            }
        });
    }
    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton4;
    private javax.swing.JLabel jLabel2;
```

V. Passenger Management Page :

The **Passenger Management Page** is a key module in the Airline Management System, designed to handle passenger-related data efficiently. It provides an interface for managing passenger records, including adding, updating, deleting, and searching for passengers. This functionality ensures that the system maintains up-to-date and accurate passenger information, which is critical for smooth operations such as ticket bookings and flight scheduling.



User Interface Overview:

The Passenger Management Page is built using **Java Swing**, featuring a clean and user-friendly design. Key interface components include:

1. **Form Fields**: Text fields for entering or editing passenger details such as name, nationality, phone number, address ,gender.
2. **Table Display**: A dynamic table that lists all passenger records
3. **Buttons**:
    - **Add**: For creating new passenger records.
    - **Edit**: For modifying existing records.
    - **Delete**: For removing a passenger record.
    - **Back**: Go back to the pervious page .
4. **Error and Confirmation Messages**: Displays feedback for user actions, such as "Passenger added successfully" or "Record not found."

Code :

```java
package com.mycompany.project;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import net.proteanit.sql.DbUtils;
public class Passengers extends javax.swing.JFrame {
    public Passengers() {
        initComponents();
        DisplayPassengers();
    }
    @SuppressWarnings("unchecked")
    Generated Code

    private void PNameTbActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void PAddressTbActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void PassNumTbActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void PPhoneTbActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
```

```java
Connection Con= null;
PreparedStatement pst=null;
ResultSet Rs = null,Rs1=null;
Statement St =null,St1=null;
private void DisplayPassengers()
{
    try{
        Con= DriverManager.getConnection("jdbc:mysql://localhost:3306/airlinedb?zeroDateTimeBehavior=CONVERT_TO_NULL","HANAE","");
        St=Con.createStatement();
        Rs =St.executeQuery("select * from PassengersTbl");
        PassengersTable.setModel(DbUtils.resultSetToTableModel(Rs));
    }catch (Exception e){
    }
}
int PassId=0;
private void CountPassengers()
{
    try{
        St1=Con.createStatement();
        Rs1 =St1.executeQuery("select Max(PId) from PassengersTbl");
        Rs1.next();
        PassId=Rs1.getInt(1)+1;
    }catch(Exception e){
    }
}
private void Clear()
{
    PNameTb.setText("");
    PassNumTb.setText("");
    PAddressTb.setText("");
    PPhoneTb.setText("");
```

```java
}
    private void SaveBtnMouseClicked(java.awt.event.MouseEvent evt) {
        if(PNameTb.getText().isEmpty()  || PassNumTb.getText().isEmpty() || PAddressTb.getText().isEmpty() || PPhoneTb.getText().isEmpty()
        {
            JOptionPane.showConfirmDialog(this, "Mission information");
        }else{
            try {
                CountPassengers();
                Con= DriverManager.getConnection("jdbc:mysql://localhost:3306/airlinedb?zeroDateTimeBehavior=CONVERT_TO_NULL","HANAE","");
                PreparedStatement Add = Con.prepareStatement("insert into PassengersTbl values(?,?,?,?,?,?,?)");
                Add.setInt(1,PassId);
                Add.setString(2, PNameTb.getText());
                Add.setString(3, NatCb.getSelectedItem().toString());
                Add.setString(4, GenCb.getSelectedItem().toString());
                Add.setString(5, PassNumTb.getText());
                Add.setString(6, PAddressTb.getText());
                Add.setString(7, PPhoneTb.getText());
                int row=Add.executeUpdate();
                JOptionPane.showMessageDialog(this,"Passenger added");
                Con.close();
                DisplayPassengers();
                Clear();
            }catch (Exception e){
                JOptionPane.showMessageDialog(this,e);
        }
    }
}
    private void DeleteBtnMouseClicked(java.awt.event.MouseEvent evt) {
        if (Key==0){
            JOptionPane.showMessageDialog(this,"select a passenger");
        }else{
```

```java
            try{
                Con= DriverManager.getConnection("jdbc:mysql://localhost:3306/airlinedb?zeroDateTimeBehavior=CONVERT_TO_NULL","HANAE","")
                String Query="delete from passengersTbl where PId="+Key;
                Statement Del =Con.createStatement();
                Del.executeUpdate(Query);
                JOptionPane.showMessageDialog(this,"Passenger deleted");
                DisplayPassengers();
            }catch (Exception e){
                JOptionPane.showMessageDialog(this,e);
            }
        }
    }
    int Key = 0 ;
    private void PassengersTableMouseClicked(java.awt.event.MouseEvent evt) {
        DefaultTableModel model =(DefaultTableModel)PassengersTable.getModel();
        int MyIndex=PassengersTable.getSelectedRow();
        Key=Integer.valueOf(model.getValueAt(MyIndex,0).toString());
        PNameTb.setText(model.getValueAt(MyIndex,1).toString());
        NatCb.setSelectedItem(model.getValueAt(MyIndex,2).toString());
        GenCb.setSelectedItem(model.getValueAt(MyIndex,3).toString());
        PassNumTb.setText(model.getValueAt(MyIndex,4).toString());
        PAddressTb.setText(model.getValueAt(MyIndex,5).toString());
        PPhoneTb.setText(model.getValueAt(MyIndex,6).toString());
    }

    private void BackBtnMouseClicked(java.awt.event.MouseEvent evt) {
        new mainform().setVisible(true);
        this.dispose();
    }

    private void EditDtnMouseClicked(java.awt.event.MouseEvent evt) {
```

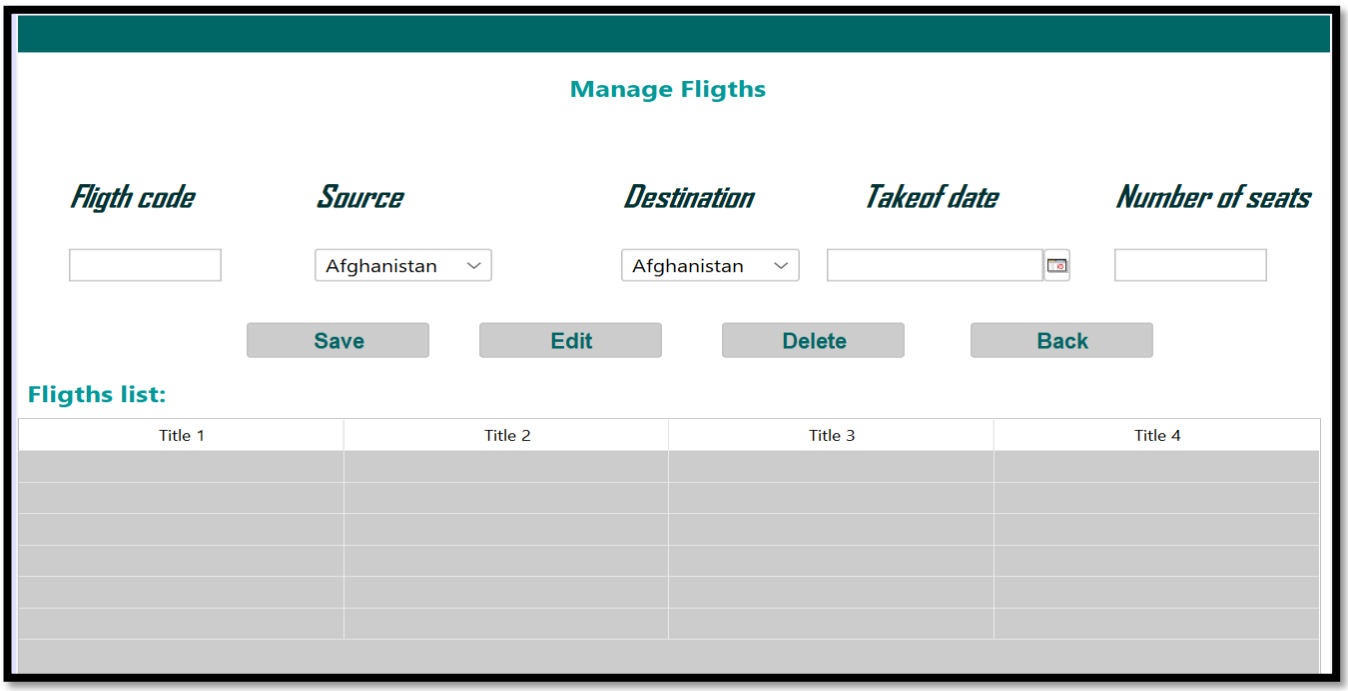
```java
private void EditDtnMouseClicked(java.awt.event.MouseEvent evt) {
    if(Key==0)
    {
        JOptionPane.showConfirmDialog(this, "select a passenger");
    }else{
        try {
            //CountPassengers();
            Con= DriverManager.getConnection("jdbc:mysql://localhost:3306/airlinedb?zeroDateTimeBehavior=CONVERT_TO_NULL","HANAE","");
            String Query="Update PassengersTbl set PName=?,PNat=?,PGen=?,PPass=?,PAdd=?,Pphone=? where PId=?";
            PreparedStatement Add = Con.prepareStatement(Query);
            Add.setInt(7,Key);
            Add.setString(1, PNameTb.getText());
            Add.setString(2, NatCb.getSelectedItem().toString());
            Add.setString(3, GenCb.getSelectedItem().toString());
            Add.setString(4, PassNumTb.getText());
            Add.setString(5, PAddressTb.getText());
            Add.setString(6, PPhoneTb.getText());
            int row=Add.executeUpdate();
            JOptionPane.showMessageDialog(this,"Passenger updated");
            Con.close();
            DisplayPassengers();
            Clear();
        }catch (Exception e){
            JOptionPane.showMessageDialog(this,e);
        }
    }
}
}
```

VI.Flights Management Page :

The **Flights Management Page** is a pivotal module in the Airline Management System, responsible for organizing and managing flight schedules. This interface allows users to handle tasks such as creating new flights, updating flight details, deleting flights, and searching for specific flights. By providing a centralized system for managing flight data, it ensures the smooth coordination of airline operations, from scheduling to passenger booking.

User Interface Overview:

- The Flights Management Page is built using **Java Swing**, featuring a user-friendly interface designed to facilitate efficient workflows. Key components of the UI include :



1. **Form Section:**
    - Fields for entering flight details, such as:
        - Flight code .
        - Source.
        - Destination .
        - Takeof Date .
        - Number of seats.
2. **Dynamic Table**:
    - Displays all registered flights in a tabular format with sortable columns for flight details.
3. **Action Buttons:**
    - **Save** : For saving a flight records.
    - **Edit**: For modifying existing records.
    - **Delete**: For removing a Flight record.
    - **Back**: Go back to the pervious page .
4. **Feedback Mechanisms:**
    - Error messages for invalid inputs.
    - Confirmation messages for successful operations.

Code :

```java
package com.mycompany.project;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import net.proteanit.sql.DbUtils;

/**
 *
 * @author HANAE
 */
public class flights extends javax.swing.JFrame {

    /**
     * Creates new form flights
     */
    public flights() {
        initComponents();
        DisplayFlight();
    }
```

```java
    private void FCodeTbActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void SeatsTbActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void BackBtnMouseClicked(java.awt.event.MouseEvent evt) {
        new mainform().setVisible(true);
        this.dispose();
    }

    private void FCodeTbMouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
    }

    private void FSourceCbActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }
Connection Con= null;
PreparedStatement pst=null;
ResultSet Rs = null,Rsl=null;
Statement St =null,Stl=null;
private void DisplayFlight(){
    try{
        Con= DriverManager.getConnection("jdbc:mysql://localhost:3306/airlinedb?zeroDateTimeBehavior=CONVERT_TO_NULL","HANAE","");
        St=Con.createStatement();
        Rs =St.executeQuery("select * from FlightTbl");
        FlightsTable.setModel(DbUtils.resultSetToTableModel(Rs));
```

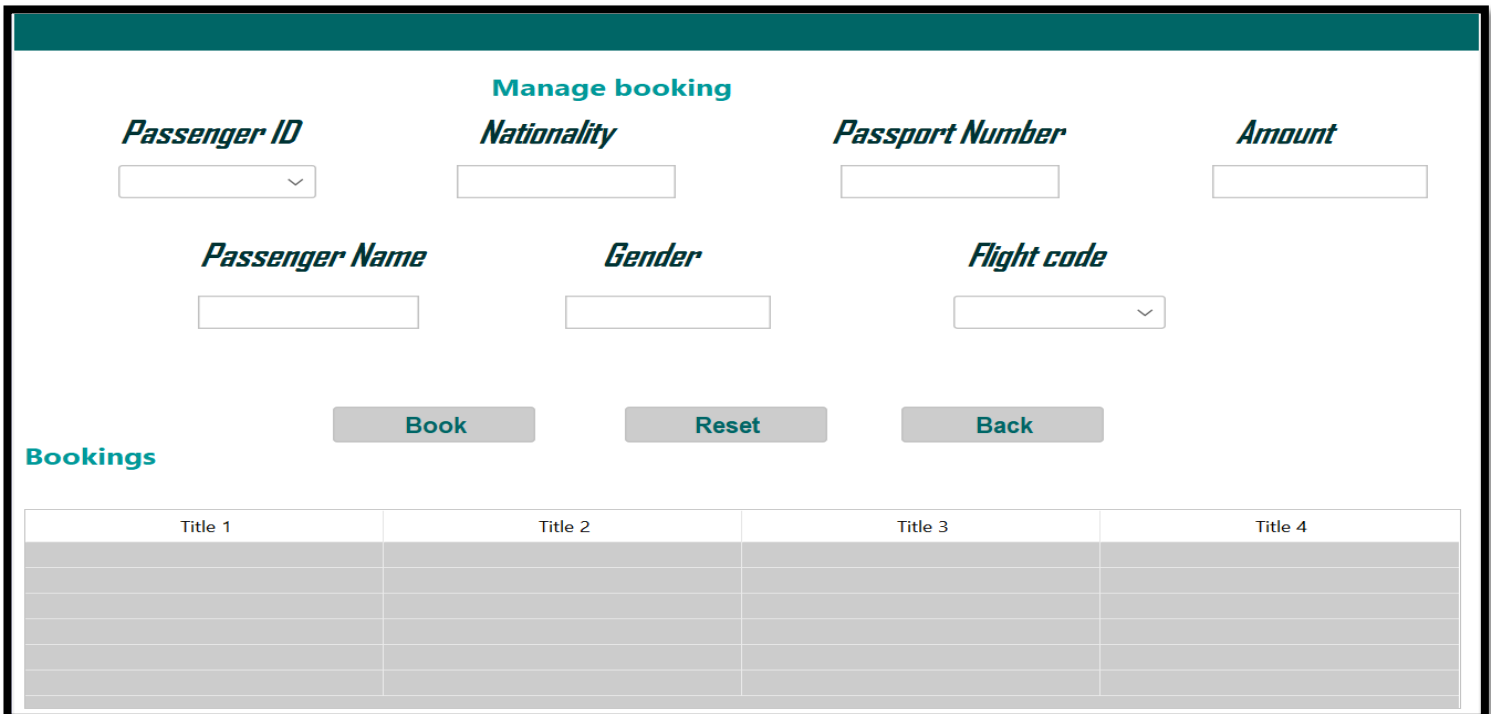
```java
    }catch (Exception e){
    }
}
private void Clear()
{
    FCodeTb.setText("");
    SeatsTb.setText("");

}
    private void SaveBtnMouseClicked(java.awt.event.MouseEvent evt) {
        if(FCodeTb.getText().isEmpty()  || FSourceCb.getSelectedIndex() == -1 || FDestCb.getSelectedIndex() == -1 ||  SeatsTb.getText().isEmpty(
        {
            JOptionPane.showConfirmDialog(this, "Mission information");
        }else{
            try {

                Con= DriverManager.getConnection("jdbc:mysql://localhost:3306/airlinedb?zeroDateTimeBehavior=CONVERT_TO_NULL","HANAE","");
                PreparedStatement Add = Con.prepareStatement("insert into FlightTbl values(?,?,?,?,?)");
                Add.setString(1, FCodeTb.getText());
                Add.setString(2, FSourceCb.getSelectedItem().toString());
                Add.setString(3, FDestCb.getSelectedItem().toString());
                Add.setString(4, FDate.getDate().toString());
                Add.setInt(5, Integer.valueOf(SeatsTb.getText()));
                int row=Add.executeUpdate();
                JOptionPane.showMessageDialog(this,"Flight added");
                Con.close();
                DisplayFlight();
                Clear();
            }catch (Exception e){
                JOptionPane.showMessageDialog(this,e);
```

```java
    private void DeleteBtnMouseClicked(java.awt.event.MouseEvent evt) {
        if (Key==""){
            JOptionPane.showMessageDialog(this,"select a Flight");
        }else{
            try{
                Con= DriverManager.getConnection("jdbc:mysql://localhost:3306/airlinedb?zeroDateTimeBehavior=CONVERT_TO_NULL","HANAE","");
                String Query="delete from FlightTbl where FIcode='"+Key+"'";
                Statement Del =Con.createStatement();
                Del.executeUpdate(Query);
                JOptionPane.showMessageDialog(this,"Flight deleted");
                DisplayFlight();
            }catch (Exception e){
                JOptionPane.showMessageDialog(this,e);
            }
        }
    }
String Key ="";
    private void FlightsTableMouseClicked(java.awt.event.MouseEvent evt) {
        DefaultTableModel model =(DefaultTableModel)FlightsTable.getModel();
        int MyIndex=FlightsTable.getSelectedRow();
        Key=model.getValueAt(MyIndex,0).toString();
        FSourceCb.setSelectedItem(model.getValueAt(MyIndex,1).toString());
        FDestCb.setSelectedItem(model.getValueAt(MyIndex,2).toString());
        SeatsTb.setText(model.getValueAt(MyIndex,4).toString());
    }

    private void EditBtnMouseClicked(java.awt.event.MouseEvent evt) {
        if(Key==="")
        {
            JOptionPane.showConfirmDialog(this, "select a passenger");
```

```java
        }else{
            try {
                //CountPassengers();
                Con= DriverManager.getConnection("jdbc:mysql://localhost:3306/airlinedb?zeroDateTimeBehavior=CONVE
                String Query="Update flighttbl set FISource=?,FIDest=?,FIDate=?,FISeats=? where FICode=?";
                PreparedStatement Add = Con.prepareStatement(Query);
                Add.setString(5,Key);
                Add.setString(1, FSourceCb.getSelectedItem().toString());
                Add.setString(2, FDestCb.getSelectedItem().toString());
                Add.setString(3, FDate.getDate().toString());
                Add.setString(4, SeatsTb.getText());
                int row=Add.executeUpdate();
                JOptionPane.showMessageDialog(this,"Flight updated");
                Con.close();
                DisplayFlight();
                Clear();
            }catch (Exception e){
                JOptionPane.showMessageDialog(this,e);
            }
    }
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        Look and feel setting code (optional)

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
```

VII. Ticket Booking Page :

- The **Ticket Booking Page** is a critical component of the Airline Management System, designed to handle the reservation of flight tickets for passengers. This module ensures that customers can seamlessly book tickets by selecting available flights, specifying passenger details, and finalizing bookings. The system verifies seat availability, calculates ticket prices, and securely stores booking data in the database.



User Interface Overview:

- The **Ticket Booking Page** is built using **Java Swing**, designed to make ticket reservation intuitive and user-friendly. It features the following components:

1. Passenger Information Form:

- Includes fields to input or select the following details:
  - **Passenger ID**: A dropdown menu for selecting the passenger's unique identifier.
  - **Passenger Name**: A text field for entering the passenger's full name.
  - **Nationality**: A text field for specifying the passenger's country of origin.
  - **Gender**: A text field or selection for identifying the passenger's gender.
  - **Passport Number**: A text field for entering the passenger's passport number for identification.
  - **Flight Code**: A dropdown menu for selecting the desired flight code.
  - **Amount**: A text field to specify or display the total ticket price.

2. Action Buttons:

- **Book**: Confirms the booking process and stores the information in the database.
- **Reset**: Clears all the fields to allow for fresh data entry.
- **Back**: Navigates back to the previous page or menu.

3. Booking Records Table:

- Displays a tabular view of all existing bookings, with columns such as passenger name, flight code, and booking details (currently placeholders are labeled as "Title 1," "Title 2," etc.).
- Provides an overview of the bookings, ensuring the user can verify and manage records efficiently.

4. Feedback and Notifications:

- Error prompts are triggered for invalid inputs (e.g., missing passenger ID or flight code).
- Success messages confirm the booking completion.

- **Code :**

```java
package com.mycompany.project;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.swing.JOptionPane;
import net.proteanit.sql.DbUtils;
public class ticketbooking extends javax.swing.JFrame {
    public ticketbooking() {
        initComponents();
        GetPassenger();
        NationalityTb.setEditable(false);
        PassNameTb.setEditable(false);
        PassNumTb.setEditable(false);
        GenTb.setEditable(false);
        GetFlights();
        DisplayBooking();
    }
    @SuppressWarnings("unchecked")
    Generated Code

    private void PassNameTbActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void AmountTbActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }
```

```java
    private void PassNumTbActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void GenTbActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }
Connection Con= null;
PreparedStatement pst=null;
ResultSet Rs = null,Rs1=null;
Statement St =null,St1=null;
private void GetPassenger()
{
    try{
        Con= DriverManager.getConnection("jdbc:mysql://localhost:3306/airlinedb?zeroDateTimeBehavior=CONVERT_TO_NULL
        St=Con.createStatement();
        String Query ="select* from PassengersTbl ";
        Rs=St.executeQuery(Query);
        while(Rs.next())
        {
            String PId=String.valueOf(Rs.getInt("PId"));
            PassIdCb.addItem(PId);
        }
    }catch(Exception e){

    }
    }
private void GetFlights()
{
    try{
        Con= DriverManager.getConnection("jdbc:mysql://localhost:3306/airlinedb?zeroDateTimeBehavior=CONVERT_TO_NULL
```

```java
        St=Con.createStatement();
        String Query ="select* from FlightTbl ";
        Rs=St.executeQuery(Query);
        while(Rs.next())
        {
            String FCode=Rs.getString("FICode");
            FCodeCb.addItem(FCode);
        }
    }catch(Exception e){

    }
    }
private void GetPassengerData()
{
    String Query="select* from PassengersTbl where PId ="+PassIdCb.getSelectedItem().toString();
    Statement St;
    ResultSet Rs;
    try{
       Con= DriverManager.getConnection("jdbc:mysql://localhost:3306/airlinedb?zeroDateTimeBehavior=CONVERT_TO_NULL","HANAE","");
       St =Con.createStatement();
       Rs=St.executeQuery(Query);
       if(Rs.next())
        {
            PassNameTb.setText(Rs.getString("PName"));
            GenTb.setText(Rs.getString("PGen"));
            PassNumTb.setText(Rs.getString("PPass"));
            NationalityTb.setText(Rs.getString("PNat"));
        }
    }catch (Exception e){
```

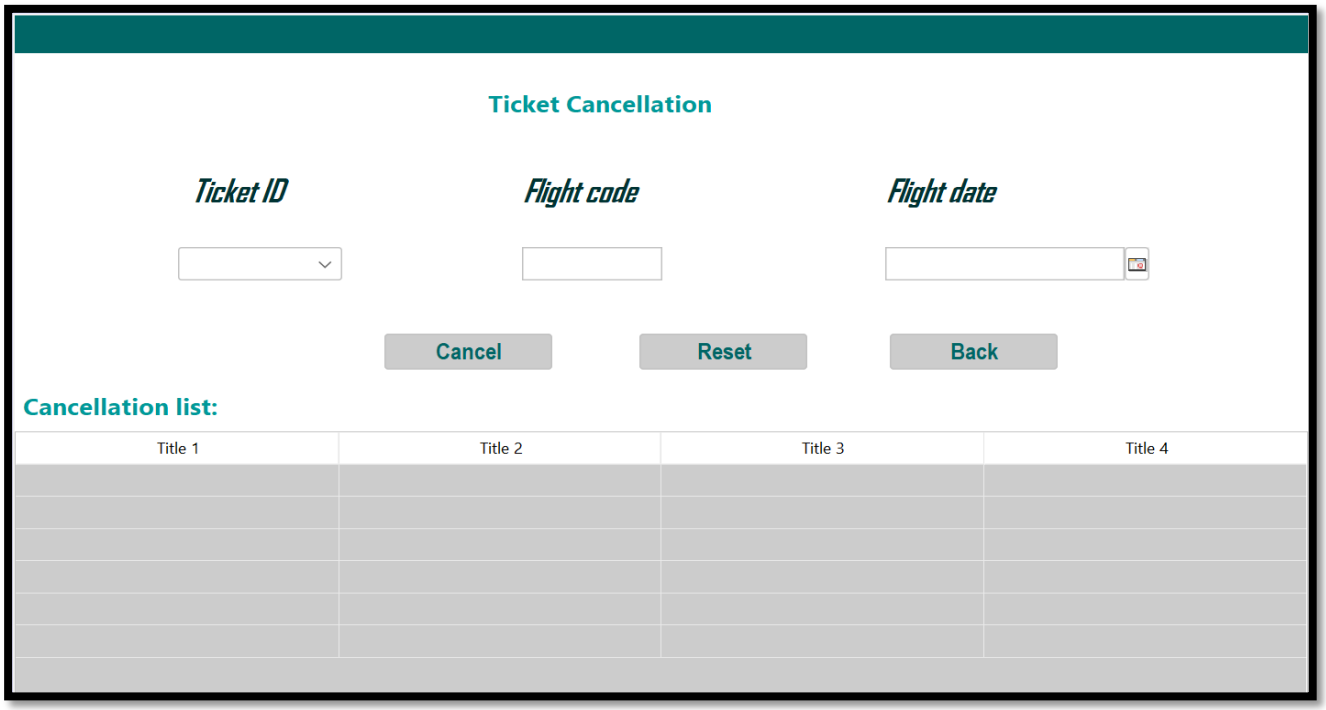
```java
private void DisplayBooking()
{
    try{
        Con= DriverManager.getConnection("jdbc:mysql://localhost:3306/airlinedb?zeroDateTimeBehavior=CONVERT_TO_NULL","HANAE","");
        St=Con.createStatement();
        Rs =St.executeQuery("select * from bookingTbl");
        BookingTable.setModel(DbUtils.resultSetToTableModel(Rs));
    }catch (Exception e){
    }
}
int TId=0;
private void CountFlights()
{
    try{
        Stl=Con.createStatement();
        Rsl =Stl.executeQuery("select Max(TicketId) from bookingTbl");
        Rsl.next();
        TId=Rsl.getInt(1)+1;
    }catch(Exception e){
    }
}
    private void BookBtnMouseClicked(java.awt.event.MouseEvent evt) {
        if(PassIdCb.getSelectedIndex()==-1  || FCodeCb.getSelectedIndex()==-1 || AmountTb.getText().isEmpty())
        {
            JOptionPane.showConfirmDialog(this, "Mission information");
        }else{
            try {
                CountFlights();
                Con= DriverManager.getConnection("jdbc:mysql://localhost:3306/airlinedb?zeroDateTimeBehavior=CONVERT_TO_NULL","HANA
                PreparedStatement Add = Con.prepareStatement("insert into BookingTbl values(?,?,?,?,?,?,?)");
                Add.setInt(1,TId);
```

```
                Add.setString(2, PassNameTb.getText());
                Add.setString(3, FCodeCb.getSelectedItem().toString());
                Add.setString(4, GenTb.getText());
                Add.setString(5, PassNumTb.getText());
                Add.setInt(6, Integer.valueOf(AmountTb.getText()));
                Add.setString(7, NationalityTb.getText());
                int row=Add.executeUpdate();
                JOptionPane.showMessageDialog(this,"ticket booked");
                Con.close();
                DisplayBooking();
                Clear();
            }catch (Exception e){
                JOptionPane.showMessageDialog(this,e);
            }
        }
        }

        private void PassIdCbMouseClicked(java.awt.event.MouseEvent evt) {

        }

        private void PassIdCbActionPerformed(java.awt.event.ActionEvent evt) {
            GetPassengerData();
        }

        private void NationalityTbActionPerformed(java.awt.event.ActionEvent evt) {
            // TODO add your handling code here:
        }

        private void BackBtnMouseClicked(java.awt.event.MouseEvent evt) {
            new mainform().setVisible(true);
            this.dispose();
```

```
    private void BackBtnMouseClicked(java.awt.event.MouseEvent evt) {
        new mainform().setVisible(true);
        this.dispose();
    }
private void Clear()
{
        FCodeCb.setSelectedIndex(-1);
        //PassIdCb.setSelectedIndex(-1);
        PassNameTb.setText("");
        PassNumTb.setText("");
        GenTb.setText("");
        NationalityTb.setText("");
}
    private void ResetBtnMouseClicked(java.awt.event.MouseEvent evt) {
        Clear();

    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        Look and feel setting code (optional)

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new ticketbooking().setVisible(true);
            }
        });
```

VIII. Ticket Cancellation Page :

- The **Ticket Cancellation Page** is an essential module in the Airline Management System. This interface enables users to cancel previously booked tickets. It simplifies the cancellation process while ensuring that all changes are seamlessly updated in the database, such as seat availability for flights. Designed using **Java Swing** and integrated with **MySQL**, this page ensures reliability and user-friendliness.



User Interface Overview:

- The Ticket Cancellation Page is divided into the following sections:

  1. **Cancellation Form:**

     - **Ticket ID**:
       A dropdown menu to select the unique ID of the Flight that needs to be canceled.
     - **Flight Code**:
       A dropdown menu or text field showing the flight code associated with the ticket ID. This helps in identifying the specific flight for which the ticket is being canceled.
     - **Flight Date** :
       A dropdown menu or text field showing the flight date associated with the ticket ID. This helps in identifying the specific flight for which the ticket is being canceled

2. **Action Buttons:**
   - **Cancel Ticket**:

     This button executes the cancellation process, which involves removing the ticket record from the database and updating the flight's available seats.
   - **Reset**:

     Clears all input fields, allowing users to start fresh in case of errors or new inputs.
   - **Back**:

     Navigates the user to the previous menu or main dashboard for further actions.
3. **Cancellation Records Table**:
   - A tabular section at the bottom displays a list of all canceled tickets. It provides details such as passenger name, flight code, cancellation date, and any refund status (if applicable).
4. **Feedback and Notifications**:
   - Success or error messages are displayed after performing actions (e.g., “Ticket canceled successfully” or “Error: Ticket ID not found”).

Code :

```java
package com.mycompany.project;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import javax.swing.JOptionPane;
import net.proteanit.sql.DbUtils;
public class Cancelation extends javax.swing.JFrame {
    public Cancelation() {
        initComponents();
        GetTicket();
        FcodeTb.setEditable(false);
        DisplayCan();
    }
    @SuppressWarnings("unchecked")
    Generated Code

    private void FcodeTbActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void ResetBtnActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void BookBtnMouseClicked(java.awt.event.MouseEvent evt) {
          if(FcodeTb.getText().isEmpty())
          {
              JOptionPane.showConfirmDialog(this, "Mission information");
          }else{
```

```
        }else{
            try {
                CountCanc();
                Con= DriverManager.getConnection("jdbc:mysql://localhost:3306/airlinedb?zeroDateTimeBehavior=CONVERT_TO_NULL","HANAE","");
                PreparedStatement Add = Con.prepareStatement("INSERT INTO CancellationTbl (TickId, FICode, CancDate) VALUES (?, ?, ?)");
                Add.setInt(1, Integer.valueOf(TIdCb.getSelectedItem().toString()));
                Add.setString(2, FcodeTb.getText());
                Add.setString(3, CDate.getDate().toString());
                int row=Add.executeUpdate();
                JOptionPane.showMessageDialog(this,"Ticket cancelled");
                Con.close();
                Cancel();
                DisplayCan();
                GetTicket();
                //Clear();
            }catch (Exception e){
                JOptionPane.showMessageDialog(this,e);
        }
    }
    }

    private void BackBtnMouseClicked(java.awt.event.MouseEvent evt) {
        new mainform().setVisible(true);
        this.dispose();

    }
Connection Con= null;
PreparedStatement pst=null;
ResultSet Rs = null,Rs1=null;
Statement St =null,St1=null;
    private void GetTicket()
```

```
    private void GetTicket()
{
    try{
        Con= DriverManager.getConnection("jdbc:mysql://localhost:3306/airlinedb?zeroDateTimeBehavior=CONVERT_TO_NULL","HA
        St=Con.createStatement();
        String Query ="select * from BookingTbl ";
        Rs = St.executeQuery(Query);
        while(Rs.next())
        {
            int T = Rs.getInt("TicketId");
            TIdCb.addItem(String.valueOf(T));
        }
    }catch(Exception e){
    }
    }
private void GetFCode()
{
    String Query="select * from BookingTbl where TicketId ="+TIdCb.getSelectedItem().toString();
    Statement St;
    ResultSet Rs;
    try{
       Con= DriverManager.getConnection("jdbc:mysql://localhost:3306/airlinedb?zeroDateTimeBehavior=CONVERT_TO_NULL","HAN
       St =Con.createStatement();
       Rs=St.executeQuery(Query);
       if(Rs.next())
        {
            FcodeTb.setText(Rs.getString("FICode"));
        }
    }catch (Exception e){
    }
}
```

```java
    private void DisplayCan()
    {
        try{
            Con= DriverManager.getConnection("jdbc:mysql://localhost:3306/airlinedb?zeroDateTimeBehavior=CON
            St=Con.createStatement();
            Rs =St.executeQuery("select * from cancellationTbl");
            cancelationtable.setModel(DbUtils.resultSetToTableModel(Rs));


        }catch (Exception e){
        }
    }
    int CId=0;
    private void CountCanc()
    {
        try{
            Stl=Con.createStatement();
            Rsl =Stl.executeQuery("select Max(CancId) from CanceilationTbl");
            if (Rsl.next()) {
                CId = Rsl.getInt(1) + 1;
            } else {
                CId = 1; // If there are no records, start with 1
            }
        }catch(Exception e){
        }
    }
        private void ResetBtnMouseClicked(java.awt.event.MouseEvent evt) {
            FcodeTb.setText("");
        }

        private void TIdCbActionPerformed(java.awt.event.ActionEvent evt) {
```

```java
        private void TIdCbActionPerformed(java.awt.event.ActionEvent evt) {
            GetFCode();
        }
    private void Cancel()
    {
        try{
                Con= DriverManager.getConnection("jdbc:mysql://localhost:3306/airlinedb?zeroDateTimeBe
                String Query="delete from BookingTbl where TicketId="+TIdCb.getSelectedItem();
                Statement Del =Con.createStatement();
                Del.executeUpdate(Query);
                //JOptionPane.showMessageDialog(this,"Flight deleted");
                //DisplayFlight();
            }catch (Exception e){
                JOptionPane.showMessageDialog(this,e);
            }
    }
        /**
         * @param args the command line arguments
         */
        public static void main(String args[]) {
            /* Set the Nimbus look and feel */
            Look and feel setting code (optional)

            /* Create and display the form */
            java.awt.EventQueue.invokeLater(new Runnable() {
                public void run() {
                    new Cancelation().setVisible(true);
                }
            });
        }
```

IX. Conclusion :

- In conclusion, the Airline Management System is a scalable, efficient, and user-friendly tool that addresses key challenges in the airline industry. By adopting modern software development principles and integrating future-oriented features, the system has the potential to evolve into a powerful platform that not only meets current requirements but also anticipates future demands.