

**Submitted to : Sir Shahid Bhatti**

**Section : A**

**Theory Assignment : 03**

**Submitted by : Mehrunisa Rafique(SP24-BSE135)**

**Hiba Akram (SP24-BSE-043)**

**Saba Ahmed (SP24-BSE-136)**

**Airline App:**

package com.example.projectapp;  
  
import javafx.application.Application;  
import javafx.geometry.Insets;  
import javafx.geometry.Pos;  
import javafx.scene.Scene;  
import javafx.scene.control.\*;  
import javafx.scene.image.Image;  
import javafx.scene.layout.\*;  
import javafx.scene.paint.Color;  
import javafx.scene.text.Font;  
import javafx.scene.text.TextAlignment;  
import javafx.stage.Stage;  
  
import java.util.Objects;  
  
public class AirlineApp extends Application {  
  
 @Override  
 public void start(Stage primaryStage) {  
  
 BorderPane root = new BorderPane();  
 BackgroundImage backgroundImage = new BackgroundImage(  
 new Image("backgroundimage.jpg"),  
 BackgroundRepeat.*NO\_REPEAT*,  
 BackgroundRepeat.*NO\_REPEAT*,  
 BackgroundPosition.*CENTER*,  
 new BackgroundSize(BackgroundSize.*AUTO*, BackgroundSize.*AUTO*, false, false, true, true)  
 );  
 root.setBackground(new Background(backgroundImage));  
  
  
 Label heading = new Label("Welcome to AirJet Booking System");  
 heading.setFont(Font.*font*("Arial", 36));  
 heading.setStyle("-fx-textFill: #34495e;-fx-font-weight: bold;");  
 heading.setAlignment(Pos.*CENTER*);  
  
  
 Label quote = new Label("\" Air travel is nature's way of making you look like your passport photo.\"");  
 quote.setFont(Font.*font*("Arial", 18));  
 quote.setStyle("-fx-textFill: #34495e; -fx-font-weight: bold;");  
 quote.setTextAlignment(TextAlignment.*CENTER*);  
 quote.setWrapText(true);  
  
 VBox centerVBox = new VBox(10, heading, quote);  
 centerVBox.setAlignment(Pos.*CENTER*);  
 root.setCenter(centerVBox);  
  
 HBox navbar = new HBox(30);  
 navbar.setPadding(new Insets(10));  
 navbar.setAlignment(Pos.*TOP\_RIGHT*);  
 navbar.setStyle("-fx-background-color: #34495e;");  
  
  
 Label navbarText = new Label("AirJet Booking");  
 navbarText.setFont(Font.*font*("Arial", 18));  
 navbarText.setTextFill(Color.*WHITE*);  
 navbarText.setAlignment(Pos.*TOP\_LEFT*);  
  
  
 Button loginButton = new Button("LOGIN");  
 Button signUpButton = new Button("Sign Up");  
  
 Button flightSearchButton = new Button("Flight Search");  
 Button flightBookingButton = new Button("Book Flights");  
 Button paymentFormButton = new Button("Payment Info");  
 Button bookingConfirmationButton = new Button("Booking Confirmation");  
  
  
 String buttonStyle = "-fx-font-weight: bold; -fx-background-color: #87cefa; -fx-font-size: 14px;";  
 loginButton.setStyle(buttonStyle);  
 signUpButton.setStyle(buttonStyle);  
  
 flightSearchButton.setStyle(buttonStyle);  
 flightBookingButton.setStyle(buttonStyle);  
 paymentFormButton.setStyle(buttonStyle);  
 bookingConfirmationButton.setStyle(buttonStyle);  
  
  
 loginButton.setOnAction(e -> NavigationHelper.*openLoginWindow*(primaryStage)  
 );  
 signUpButton.setOnAction(e -> NavigationHelper.*openSignUpWindow*(primaryStage));  
  
 flightSearchButton.setOnAction(e -> {  
 NavigationHelper.*checkLoginBeforeAction*(() -> {  
 FlightSearch.*openFlightSearchForm*(primaryStage);  
 });  
 });  
  
 flightBookingButton.setOnAction(e -> {  
 NavigationHelper.*checkLoginBeforeAction*(() -> {  
 Flight selectedFlight = FlightSearch.*getSelectedFlight*();  
 if (selectedFlight == null) {  
 Alert alert = new Alert(Alert.AlertType.*WARNING*, "Please select a flight from the table before booking.");  
 alert.showAndWait();  
 } else {  
 FlightBooking.*openBookingForm*(primaryStage, selectedFlight);  
 }  
 });  
 });  
  
 paymentFormButton.setOnAction(e -> {  
 NavigationHelper.*checkLoginBeforeAction*(() -> {  
 NavigationHelper.*openPaymentInfoForm*(primaryStage);  
 });  
 });  
  
 bookingConfirmationButton.setOnAction(e -> {  
 NavigationHelper.*checkLoginBeforeAction*(() -> {  
 NavigationHelper.*openBookingConfirmationWindow*(primaryStage);  
 });  
 });  
  
 navbar.getChildren().addAll( navbarText,loginButton, signUpButton, //forgotPasswordButton,  
 flightSearchButton, flightBookingButton, paymentFormButton, bookingConfirmationButton);  
  
 root.setTop(navbar);  
  
 primaryStage.setMaximized(true);  
 primaryStage.setResizable(true);  
  
 Scene scene = new Scene(root);  
 primaryStage.setTitle("Airline Booking System");  
 primaryStage.setScene(scene);  
 primaryStage.show();  
 }  
  
 public static void main(String[] args) {  
 *launch*();  
}  
}

**Create Account:**

package com.example.projectapp;  
  
import javafx.geometry.Insets;  
import javafx.geometry.Pos;  
import javafx.scene.Scene;  
import javafx.scene.control.\*;  
import javafx.scene.image.Image;  
import javafx.scene.layout.\*;  
import javafx.scene.paint.Color;  
import javafx.scene.text.Font;  
import javafx.stage.Stage;  
  
public class createaccount {  
 public void createaccount(){  
 Stage newstage = new Stage();  
 newstage.setTitle("Create new Password");  
  
 BackgroundFill backgroundFill = new BackgroundFill(Color.*LIGHTBLUE*,null,null);  
 VBox newlayout1 = new VBox(20);  
 newlayout1.setAlignment(Pos.*CENTER*);  
 newlayout1.setMaxHeight(400);  
 newlayout1.setMaxWidth(400);  
 newlayout1.setPadding(new Insets(20));  
  
 Label createaccountlabel = new Label("Create Account");  
 createaccountlabel.setStyle("-fx-font-weight:bold;");  
 createaccountlabel.setTextFill(Color.*GRAY*);  
 createaccountlabel.setFont(new Font("Times New Roman",20));  
 createaccountlabel.setAlignment(Pos.*CENTER*);  
  
 Label name1label = new Label("First Name");  
 name1label.setStyle("-fx-font-weight:bold; -fx-text-fill: black;");  
 name1label.setFont(new Font(16));  
 TextField name1field= new TextField();  
 name1field.setPromptText("First name");  
  
 Label lastnamelabel = new Label("Last Name");  
 lastnamelabel.setStyle("-fx-font-weight:bold; -fx-text-fill: black;");  
 lastnamelabel.setFont(new Font(16));  
 TextField lastnamefield = new TextField();  
 lastnamefield.setPromptText("Last name");  
  
 Label emaillabel = new Label("Email Address");  
 emaillabel.setStyle("-fx-font-weight:bold; -fx-text-fill: black;");  
 emaillabel.setFont(new Font(16));  
 TextField emailtextfield = new TextField();  
 emailtextfield.setPromptText("Email address");  
  
 Label passwordlabel = new Label("Password");  
 passwordlabel.setStyle("-fx-font-weight:bold; -fx-text-fill: black;");  
 passwordlabel.setFont(new Font(16));  
 PasswordField passwordField= new PasswordField();  
 passwordField.setPromptText("Password");  
 Label passwordtextlabel = new Label("Use at least 8 characters, including both letters and symbols");  
  
 Button createbutton = new Button("Create");  
 createbutton.setAlignment(Pos.*CENTER*);  
 createbutton.setPrefWidth(250);  
 createbutton.setStyle(  
 "-fx-background-color: skyblue;"+  
 "-fx-text-fill: white;"+  
 "-fx-border-radius: 5px;"+  
 "-fx-text-weight: bold;"  
 );  
 createbutton.setOnAction(e->{  
 String name = name1field.getText();  
 String email = emailtextfield.getText();  
  
  
 if (name.isEmpty() || email.isEmpty()) {  
 Alert alert = new Alert(Alert.AlertType.*WARNING*, "Please fill all the text fields");  
 alert.showAndWait();  
 } else {  
 System.*out*.println("Login successful! Name: " + name + ", Email: " + email);  
  
 }  
  
 });  
  
 Button cancelbutton = new Button("Cancel");  
 cancelbutton.setAlignment(Pos.*CENTER*);  
 cancelbutton.setPrefWidth(250);  
 cancelbutton.setOnAction(e-> System.*exit*(0));  
 cancelbutton.setStyle(  
 "-fx-background-color: skyblue;"+  
 "-fx-text-fill: white;"+  
 "-fx-border-radius: 5px;"+  
 "-fx-text-weight: bold;"  
 );  
 CheckBox acceptTerms = new CheckBox("I accept the Privacy Policy and Terms of Use.");  
  
 newlayout1.setStyle(  
 "-fx-background-color: white;"+  
 "-fx-border-radius: 15;"+  
 "-fx-background-radius:20;"+  
 "-fx-padding: 20;"  
  
 );  
  
 newlayout1.getChildren().addAll(createaccountlabel,name1label,  
 name1field,lastnamelabel,lastnamefield,  
 emaillabel,emailtextfield,passwordlabel,passwordField,passwordtextlabel,  
 acceptTerms,createbutton,cancelbutton);  
  
 StackPane root = new StackPane();  
 root.setBackground(new Background(backgroundFill));  
 root.getChildren().addAll(newlayout1);  
  
 Scene scene = new Scene(root,550,550);  
 newstage.setScene(scene);  
 newstage.show();  
  
 }  
  
}

**Flight Booking:**

package com.example.projectapp;  
  
import javafx.geometry.Insets;  
import javafx.geometry.Pos;  
import javafx.scene.Scene;  
import javafx.scene.control.\*;  
import javafx.scene.layout.GridPane;  
import javafx.scene.layout.StackPane;  
import javafx.scene.layout.VBox;  
import javafx.scene.paint.Color;  
import javafx.scene.text.Font;  
import javafx.stage.Screen;  
import javafx.stage.Stage;  
  
import java.io.BufferedWriter;  
import java.io.FileWriter;  
import java.io.IOException;  
  
public class FlightBooking {  
  
 public static void openBookingForm(Stage primaryStage, Flight selectedFlight) {  
 if (selectedFlight == null) {  
 Alert alert = new Alert(Alert.AlertType.*WARNING*);  
 alert.setTitle("Booking Warning");  
 alert.setHeaderText(null);  
 alert.setContentText("Please select a flight before proceeding with the booking.");  
 alert.showAndWait();  
 return;  
 }  
  
  
 Stage bookingStage = new Stage();  
  
 bookingStage.setTitle("Flight Booking");  
  
  
  
 VBox layout = new VBox(20);  
 layout.setStyle("-fx-background-color: #2c3e50;-fx-font-weight: bold;");  
 layout.setPadding(new Insets(20));  
 layout.setAlignment(Pos.*TOP\_CENTER*);  
  
 Label titleLabel = new Label("Booking for Flight: " + selectedFlight.getFlightID());  
 titleLabel.setTextFill(Color.*WHITE*);  
 titleLabel.setStyle("-fx-font-size: 24px; -fx-font-weight: bold;");  
  
 Label detailsLabel = new Label(selectedFlight.toString());  
 detailsLabel.setTextFill(Color.*WHITE*);  
 detailsLabel.setStyle("-fx-font-size: 16px;");  
  
 Label passengerNameLabel = new Label("Passenger Name:");  
 passengerNameLabel.setAlignment(Pos.*CENTER*);  
 passengerNameLabel.setTextFill(Color.*WHITE*);  
 TextField passengerNameField = new TextField();  
 passengerNameLabel.setPrefWidth(100);  
  
 Label contactLabel = new Label("Contact Information:");  
 contactLabel.setTextFill(Color.*WHITE*);  
 TextField contactField = new TextField();  
 contactField.setPrefWidth(300);  
  
 Label adultLabel = new Label("Number of Adults:");  
 adultLabel.setTextFill(Color.*WHITE*);  
 Spinner<Integer> adultSpinner = new Spinner<>(1, 10, 1);  
 adultSpinner.setPrefWidth(150);  
  
 Label childLabel = new Label("Number of Children:");  
 childLabel.setTextFill(Color.*WHITE*);  
  
 Spinner<Integer> childSpinner = new Spinner<>(0, 10, 0);  
 childSpinner.setPrefWidth(150);  
  
 Label infantLabel = new Label("Number of Infants:");  
 infantLabel.setTextFill(Color.*WHITE*);  
 Spinner<Integer> infantSpinner = new Spinner<>(0, 10, 0);  
 infantSpinner.setPrefWidth(150);  
  
 Label priceLabel = new Label("Total Price: PKR: 0");  
 priceLabel.setTextFill(Color.*WHITE*);  
 priceLabel.setStyle("-fx-font-size: 18px;");  
 Button calculateButton = new Button("Calculate Price");  
 calculateButton.setOnAction(e -> {  
 int numAdults = adultSpinner.getValue();  
 int numChildren = childSpinner.getValue();  
 int numInfants = infantSpinner.getValue();  
  
 double totalPrice = *calculateTotalPrice*(selectedFlight, numAdults, numChildren, numInfants);  
 priceLabel.setText("Total Price: PKR" + totalPrice);  
 });  
  
 Button confirmBookingButton = new Button("Confirm Booking");  
 confirmBookingButton.setOnAction(e -> {  
 String passengerName = passengerNameField.getText();  
 String contactInfo = contactField.getText();  
  
 if (passengerName.isEmpty() || contactInfo.isEmpty()) {  
 Alert alert = new Alert(Alert.AlertType.*WARNING*, "Please fill in all fields.");  
 alert.showAndWait();  
 } else {  
 *saveBookingDetails*(selectedFlight, passengerName, contactInfo, adultSpinner.getValue(), childSpinner.getValue(), infantSpinner.getValue());  
 Alert successAlert = new Alert(Alert.AlertType.*INFORMATION*, "Booking confirmed!");  
 successAlert.showAndWait();  
 bookingStage.close();  
 }  
 });  
  
 layout.getChildren().addAll(titleLabel, detailsLabel, passengerNameLabel, passengerNameField, contactLabel, contactField,  
 adultLabel, adultSpinner, childLabel, childSpinner, infantLabel, infantSpinner, calculateButton, priceLabel, confirmBookingButton);  
  
 Scene scene = new Scene(layout, 800, 700);  
 primaryStage.setFullScreen(true);  
 bookingStage.setScene(scene);  
 bookingStage.show();  
 }  
  
 private static double calculateTotalPrice(Flight flight, int numAdults, int numChildren, int numInfants) {  
 double adultPrice = flight.getPrice();  
 double childDiscount = 0.5;  
 double infantDiscount = 0.1;  
  
 double totalPrice = (numAdults \* adultPrice) + (numChildren \* adultPrice \* childDiscount) + (numInfants \* adultPrice \* infantDiscount);  
 return totalPrice;  
 }  
  
 private static void saveBookingDetails(Flight flight, String passengerName, String contactInfo, int numAdults, int numChildren, int numInfants) {  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter("bookings.txt", true))) {  
 writer.write("Flight ID: " + flight.getFlightID());  
 writer.newLine();  
 writer.write("Passenger Name: " + passengerName);  
 writer.newLine();  
 writer.write("Contact Info: " + contactInfo);  
 writer.newLine();  
 writer.write("Number of Adults: " + numAdults);  
 writer.newLine();  
 writer.write("Number of Children: " + numChildren);  
 writer.newLine();  
 writer.write("Number of Infants: " + numInfants);  
 writer.newLine();  
 writer.write("Total Price: " + *calculateTotalPrice*(flight, numAdults, numChildren, numInfants));  
 writer.newLine();  
 writer.write("---------------------------------------");  
 writer.newLine();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
}  
}

**User File Handler:**

package com.example.projectapp;  
  
  
import javafx.scene.control.Alert;  
  
import java.io.\*;  
import java.security.MessageDigest;  
import java.security.NoSuchAlgorithmException;  
  
public class UserFileHandler {  
  
 private static final String *FILE\_PATH* = "users.txt";  
  
  
 public static void saveUserInfo(String name, String email, String password) {  
 String encryptedPassword = *encryptPassword*(password);  
  
 String userInfo = name + "," + email + "," + encryptedPassword;  
  
  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter(*FILE\_PATH*, true))) {  
 writer.write(userInfo);  
 writer.newLine();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public static boolean emailExists(String email) {  
 try (BufferedReader reader = new BufferedReader(new FileReader(*FILE\_PATH*))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 String[] userData = line.split(",");  
 if (userData.length >= 2 && userData[1].equals(email)) {  
 return true;  
 }  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return false;  
 }  
  
  
 public boolean validateCredentials(String email, String password) {  
 String encryptedPassword = *encryptPassword*(password);  
 try (BufferedReader reader = new BufferedReader(new FileReader(*FILE\_PATH*))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 String[] parts = line.split(",");  
 if (parts.length >= 3) {  
 String storedEmail = parts[1];  
 String storedPassword = parts[2];  
 if (storedEmail.equals(email) && storedPassword.equals(encryptedPassword)) {  
 return true;  
 }  
 }  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return false;  
 }  
  
 private static String encryptPassword(String password) {  
 try {  
 MessageDigest md = MessageDigest.*getInstance*("SHA-256");  
 byte[] hashBytes = md.digest(password.getBytes());  
 StringBuilder hexString = new StringBuilder();  
 for (byte b : hashBytes) {  
 hexString.append(String.*format*("%02x", b));  
 }  
 return hexString.toString();  
 } catch (NoSuchAlgorithmException e) {  
 e.printStackTrace();  
 return password;  
 }  
 }  
 public static boolean resetPassword(String email, String newPassword) {  
 File file = new File(*FILE\_PATH*);  
 File tempFile = new File("temp\_users.txt");  
 boolean emailFound = false;  
  
 try (BufferedReader reader = new BufferedReader(new FileReader(file));  
 BufferedWriter writer = new BufferedWriter(new FileWriter(tempFile))) {  
  
 String line;  
 while ((line = reader.readLine()) != null) {  
 String[] userData = line.split(",");  
 if (userData.length >= 3 && userData[1].equals(email)) {  
 emailFound = true;  
 String encryptedPassword = *encryptPassword*(newPassword);  
 writer.write(userData[0] + "," + userData[1] + "," + encryptedPassword);  
 } else {  
 writer.write(line);  
 }  
 writer.newLine();  
 }  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 return false;  
 }  
  
  
 if (emailFound) {  
 if (!file.delete() || !tempFile.renameTo(file)) {  
 System.*out*.println("Error updating the password file.");  
 return false;  
 }  
 } else {  
 tempFile.delete();  
 }  
  
 return emailFound;  
 }  
  
 static void savePaymentInfoToFile(String cardNumber, String expiryDate, String cvv) {  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter("paymentHistory.txt", true))) {  
  
 writer.write("Card Number: " + cardNumber);  
 writer.newLine();  
 writer.write("Expiry Date: " + expiryDate);  
 writer.newLine();  
 writer.write("CVV: " + cvv);  
 writer.newLine();  
 writer.write("---------");  
 writer.newLine();  
  
  
 Alert alert = new Alert(Alert.AlertType.*INFORMATION*, "Payment info saved successfully.");  
 alert.showAndWait();  
 } catch (IOException e) {  
 e.printStackTrace();  
 Alert alert = new Alert(Alert.AlertType.*ERROR*, "Error saving payment info.");  
 alert.showAndWait();  
 }  
  
 }  
}

**Flight Search:**

package com.example.projectapp;  
  
import javafx.collections.FXCollections;  
import javafx.collections.ObservableList;  
import javafx.geometry.Insets;  
import javafx.geometry.Pos;  
import javafx.scene.Scene;  
import javafx.scene.control.\*;  
import javafx.scene.layout.\*;  
import javafx.scene.text.Font;  
import javafx.stage.Stage;  
import java.io.BufferedWriter;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.time.LocalDate;  
import java.util.stream.Collectors;  
  
public class FlightSearch {  
 private static TableView<Flight> *flightTable* = new TableView<>(); // Initialize flightTable  
  
 public static Flight getSelectedFlight() {  
 return *flightTable*.getSelectionModel().getSelectedItem();  
 }  
  
 private static ObservableList<Flight> *allFlights* = FXCollections.*observableArrayList*(  
 new Flight("FL123", "New York", "London", "Economy", "2024-12-15", 500),  
 new Flight("FL124", "Los Angeles", "Paris", "Business", "2024-12-16", 1200),  
 new Flight("FL125", "Chicago", "Tokyo", "Economy", "2024-12-20", 800),  
 new Flight("FL126", "San Francisco", "Rome", "Business", "2024-12-22", 1500)  
 );  
  
 public static void openFlightSearchForm(Stage primaryStage) {  
 Stage newStage = new Stage();  
 newStage.setTitle("Flight Search");  
  
 GridPane layout = *createFlightSearchLayout*(newStage);  
 layout.setStyle("-fx-background-color: #2c3e50;-fx-font-weight: bold;");  
  
 StackPane root = new StackPane();  
 root.getChildren().add(layout);  
  
 Scene scene = new Scene(root, 800, 700);  
 primaryStage.setFullScreen(true);  
 newStage.setScene(scene);  
 newStage.show();  
 }  
  
 private static GridPane createFlightSearchLayout(Stage newStage) {  
 GridPane layout = new GridPane();  
 layout.setAlignment(Pos.*CENTER*);  
 layout.setVgap(15);  
 layout.setHgap(15);  
 layout.setPadding(new Insets(20));  
  
 Label flightSearchLabel = new Label("Flight Search");  
 flightSearchLabel.setFont(new Font("Arial", 22));  
 flightSearchLabel.setStyle("-fx-font-weight: bold; -fx-text-fill: white;");  
  
 Label flightTypeLabel = new Label("Flight Type:");  
 flightTypeLabel.setStyle("-fx-font-weight: bold; -fx-text-fill: white;");  
 RadioButton oneWayRadio = new RadioButton("One-Way");  
 RadioButton roundTripRadio = new RadioButton("Round-Trip");  
 ToggleGroup flightTypeGroup = new ToggleGroup();  
 oneWayRadio.setToggleGroup(flightTypeGroup);  
 roundTripRadio.setToggleGroup(flightTypeGroup);  
 oneWayRadio.setSelected(true);  
 oneWayRadio.setStyle("-fx-font-weight: bold; -fx-text-fill: white;");  
 roundTripRadio.setStyle("-fx-font-weight: bold; -fx-text-fill: white;");  
  
 Label departureLabel = new Label("Departure City:");  
 departureLabel.setStyle("-fx-font-weight: bold; -fx-text-fill: white;");  
 TextField departureField = new TextField();  
 departureField.setPromptText("Enter Departure City");  
  
 Label destinationLabel = new Label("Destination City:");  
 destinationLabel.setStyle("-fx-font-weight: bold; -fx-text-fill: white;");  
 TextField destinationField = new TextField();  
 destinationField.setPromptText("Enter Destination City");  
  
 Label departureDateLabel = new Label("Departure Date:");  
 departureDateLabel.setStyle("-fx-font-weight: bold; -fx-text-fill: white;");  
 DatePicker departureDatePicker = new DatePicker();  
  
 Label returnDateLabel = new Label("Return Date:");  
 returnDateLabel.setStyle("-fx-font-weight: bold; -fx-text-fill: white;");  
 DatePicker returnDatePicker = new DatePicker();  
 returnDatePicker.setDisable(true);  
  
 roundTripRadio.setOnAction(e -> returnDatePicker.setDisable(false));  
 oneWayRadio.setOnAction(e -> returnDatePicker.setDisable(true));  
  
 Label classLabel = new Label("Class:");  
 classLabel.setStyle("-fx-font-weight: bold; -fx-text-fill: white;");  
 ComboBox<String> classComboBox = new ComboBox<>();  
 classComboBox.getItems().addAll("Economy", "Business");  
  
 Label travelersLabel = new Label("Number of Travelers:");  
 travelersLabel.setStyle("-fx-font-weight: bold; -fx-text-fill: white;");  
 Spinner<Integer> travelersSpinner = new Spinner<>(1, 10, 1);  
  
 *flightTable* = *createFlightTable*();  
  
 Button searchButton = new Button("Search Flights");  
 searchButton.setStyle("-fx-background-color: #4A90E2; -fx-text-fill: white;");  
 searchButton.setOnAction(e -> *handleSearchButtonClick*(departureField, destinationField, classComboBox,  
 departureDatePicker, returnDatePicker,  
 *flightTable*, oneWayRadio, roundTripRadio));  
  
  
 Button favoritesButton = new Button("Select Flight");  
 favoritesButton.setStyle("-fx-background-color: #FFC107; -fx-text-fill: black;");  
 favoritesButton.setOnAction(e -> *saveSelectedFlight*(*flightTable*.getSelectionModel().getSelectedItem()));  
  
  
 Button cancelButton = new Button("Cancel");  
 cancelButton.setStyle("-fx-background-color: #FF3D00; -fx-text-fill: white;");  
 cancelButton.setOnAction(e -> newStage.close());  
  
 layout.add(flightSearchLabel, 0, 0, 2, 1);  
 layout.add(flightTypeLabel, 0, 1);  
 layout.add(oneWayRadio, 1, 1);  
 layout.add(roundTripRadio, 1, 2);  
 layout.add(departureLabel, 0, 3);  
 layout.add(departureField, 1, 3);  
 layout.add(destinationLabel, 0, 4);  
 layout.add(destinationField, 1, 4);  
 layout.add(departureDateLabel, 0, 5);  
 layout.add(departureDatePicker, 1, 5);  
 layout.add(returnDateLabel, 0, 6);  
 layout.add(returnDatePicker, 1, 6);  
 layout.add(travelersLabel, 0, 7);  
 layout.add(travelersSpinner, 1, 7);  
 layout.add(classLabel, 0, 8);  
 layout.add(classComboBox, 1, 8);  
 layout.add(*flightTable*, 0, 9, 2, 1);  
 layout.add(searchButton, 0, 10);  
 layout.add(favoritesButton, 1, 10);  
 layout.add(cancelButton, 1, 11);  
  
 return layout;  
 }  
  
 private static TableView<Flight> createFlightTable() {  
 TableView<Flight> table = new TableView<>();  
 table.setPrefWidth(600);  
  
 TableColumn<Flight, String> flightIDColumn = new TableColumn<>("Flight ID");  
 flightIDColumn.setCellValueFactory(cellData -> cellData.getValue().flightIDProperty());  
  
 TableColumn<Flight, String> departureCityColumn = new TableColumn<>("Departure City");  
 departureCityColumn.setCellValueFactory(cellData -> cellData.getValue().departureCityProperty());  
  
 TableColumn<Flight, String> destinationCityColumn = new TableColumn<>("Destination City");  
 destinationCityColumn.setCellValueFactory(cellData -> cellData.getValue().destinationCityProperty());  
  
 TableColumn<Flight, String> flightClassColumn = new TableColumn<>("Class");  
 flightClassColumn.setCellValueFactory(cellData -> cellData.getValue().flightClassProperty());  
  
 TableColumn<Flight, String> dateColumn = new TableColumn<>("Date");  
 dateColumn.setCellValueFactory(cellData -> cellData.getValue().dateProperty());  
  
 TableColumn<Flight, Double> priceColumn = new TableColumn<>("Price (PKR)");  
 priceColumn.setCellValueFactory(cellData -> cellData.getValue().priceProperty().asObject());  
  
 table.getColumns().addAll(flightIDColumn, departureCityColumn, destinationCityColumn, flightClassColumn, dateColumn, priceColumn);  
 table.setItems(*allFlights*);  
  
 return table;  
 }  
  
 private static void handleSearchButtonClick(TextField departureField, TextField destinationField,  
 ComboBox<String> classComboBox, DatePicker departureDatePicker,  
 DatePicker returnDatePicker, TableView<Flight> flightTable,  
 RadioButton oneWayRadio, RadioButton roundTripRadio) {  
 String departure = departureField.getText();  
 String destination = destinationField.getText();  
 String selectedClass = classComboBox.getValue();  
 LocalDate departureDate = departureDatePicker.getValue();  
 LocalDate returnDate = returnDatePicker.getValue();  
  
 if (departure.isEmpty() || destination.isEmpty() || selectedClass == null || departureDate == null) {  
 Alert alert = new Alert(Alert.AlertType.*WARNING*, "Please fill all required fields.");  
 alert.showAndWait();  
 return;  
 }  
  
 if (roundTripRadio.isSelected() && returnDate == null) {  
 Alert alert = new Alert(Alert.AlertType.*WARNING*, "Please select a return date for Round-Trip flights.");  
 alert.showAndWait();  
 return;  
 }  
  
 ObservableList<Flight> filteredFlights = *allFlights*.stream()  
 .filter(flight -> flight.getDepartureCity().equalsIgnoreCase(departure))  
 .filter(flight -> flight.getDestinationCity().equalsIgnoreCase(destination))  
 .filter(flight -> flight.getFlightClass().equalsIgnoreCase(selectedClass))  
 .filter(flight -> flight.getDate().equals(departureDate.toString()) ||  
 (roundTripRadio.isSelected() && flight.getDate().equals(returnDate.toString())))  
 .collect(Collectors.*toCollection*(FXCollections::*observableArrayList*));  
  
 if (filteredFlights.isEmpty()) {  
 Alert alert = new Alert(Alert.AlertType.*INFORMATION*, "No flights match the search criteria.");  
 alert.showAndWait();  
 } else {  
 flightTable.setItems(filteredFlights);  
 }  
 }  
  
 private static void saveSelectedFlight(Flight selectedFlight) {  
 if (selectedFlight == null) {  
 Alert alert = new Alert(Alert.AlertType.*WARNING*, "Please select a flight to save.");  
 alert.showAndWait();  
 return;  
 }  
  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter("favorites.txt", true))) {  
 writer.write(selectedFlight.toString());  
 writer.newLine();  
 Alert alert = new Alert(Alert.AlertType.*INFORMATION*, "Your Flight is saved.");  
 alert.showAndWait();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
}  
}

**Navigation Helper:**

package com.example.projectapp;  
  
import javafx.geometry.Insets;  
import javafx.geometry.Pos;  
import javafx.scene.Scene;  
import javafx.scene.control.\*;  
import javafx.scene.layout.GridPane;  
import javafx.scene.layout.VBox;  
import javafx.scene.paint.Color;  
import javafx.scene.text.Font;  
import javafx.stage.Stage;  
  
public class NavigationHelper {  
  
 private static boolean *isLoggedIn* = false;  
 private static String *loggedInUser* = "";  
 private static Stage *parentStage*;  
  
  
 public static void openSignUpWindow(Stage parentStage) {  
  
 Stage signUpStage = new Stage();  
 VBox signUpBox = new VBox(15);  
 signUpBox.setAlignment(Pos.*CENTER*);  
 signUpBox.setPadding(new Insets(20));  
 signUpBox.setStyle("-fx-background-color: #2c3e50;-fx-font-weight: bold;");  
  
 TextField nameField = new TextField();  
 nameField.setPromptText("Enter your name");  
 nameField.setPrefHeight(40);  
 nameField.setStyle("-fx-background-color: #3A3A3A; -fx-border-color: #1976D2; -fx-border-radius: 5px; -fx-text-fill: white; -fx-font-weight: bold;");  
  
 TextField emailField = new TextField();  
 emailField.setPromptText("Enter your email");  
 emailField.setPrefHeight(40);  
 emailField.setStyle("-fx-background-color: #3A3A3A; -fx-border-color: #1976D2; -fx-border-radius: 5px; -fx-text-fill: white; -fx-font-weight: bold;");  
 Label signupLabel=new Label("Sign up");  
 signupLabel.setFont(Font.*font*("Arial", 36));  
 signupLabel.setStyle("-fx-text-fill: white;");  
 PasswordField passwordField = new PasswordField();  
 passwordField.setPromptText("Enter your password");  
 passwordField.setPrefHeight(40);  
 passwordField.setStyle("-fx-background-color: #3A3A3A; -fx-border-color: #1976D2; -fx-border-radius: 5px; -fx-text-fill: white; -fx-font-weight: bold;");  
  
 Button signUpButton = new Button("Sign Up");  
 signUpButton.setPrefHeight(35);  
 signUpButton.setStyle("-fx-background-color: #1976D2; -fx-text-fill: white; -fx-font-weight: bold;");  
 signUpButton.setPrefWidth(100);  
 signUpButton.setOnAction(e -> {  
 String name = nameField.getText();  
 String email = emailField.getText();  
 String password = passwordField.getText();  
  
 if (name.isEmpty() || email.isEmpty() || password.isEmpty()) {  
 *showAlert*(Alert.AlertType.*WARNING*, "Validation Error", "Please fill in all fields.");  
 return;  
 }  
  
  
 if (!email.matches("^[a-zA-Z0-9\_+&-]+(?:\\.[a-zA-Z0-9\_+&-]+)\*@(?:[a-zA-Z0-9-]+\\.)+[a-zA-Z]{2,7}$")) {  
 *showAlert*(Alert.AlertType.*WARNING*, "Invalid Email", "Please enter a valid email address.");  
 return;  
 }  
  
 if (password.length() < 6) {  
 *showAlert*(Alert.AlertType.*WARNING*, "Weak Password", "Password must be at least 6 characters long.");  
 return;  
 }  
  
 if (UserFileHandler.*emailExists*(email)) {  
 *showAlert*(Alert.AlertType.*WARNING*, "Email Already Registered", "This email is already registered.");  
 return;  
 }  
  
  
 UserFileHandler.*saveUserInfo*(name, email, password);  
  
 *showAlert*(Alert.AlertType.*INFORMATION*, "Sign-Up Successful", "You have successfully signed up!");  
 signUpStage.close();  
 });  
  
 signUpBox.getChildren().addAll(signupLabel, nameField, emailField, passwordField, signUpButton);  
  
  
 Scene signUpScene = new Scene(signUpBox, 400, 300);  
 signUpStage.setTitle("Sign Up");  
 signUpStage.setScene(signUpScene);  
 signUpStage.show();  
 }  
  
 public static void openLoginWindow(Stage primaryStage) {  
 Stage loginStage = new Stage();  
 loginStage.setTitle("Login");  
  
 VBox layout = new VBox(10);  
 layout.setStyle("-fx-padding: 20; -fx-alignment: center;-fx-background-color: #2c3e50;");  
 Label loginLabel=new Label("Login to your account");  
 loginLabel.setStyle("-fx-text-fill: white;-fx-text-fill: white;");  
 loginLabel.setFont(Font.*font*("Arial", 20));  
 TextField emailField = new TextField();  
 emailField.setPromptText("Enter your email");  
 emailField.setPrefHeight(35);  
 emailField.setStyle("-fx-background-color: #3A3A3A; -fx-border-color: #1976D2; -fx-border-radius: 5px; -fx-text-fill: white; -fx-font-weight: bold;");  
  
 PasswordField passwordField = new PasswordField();  
 passwordField.setPrefHeight(35);  
 passwordField.setPromptText("Enter your password");  
 passwordField.setStyle("-fx-background-color: #3A3A3A; -fx-border-color: #1976D2; -fx-border-radius: 5px; -fx-text-fill: white; -fx-font-weight: bold;");  
  
  
 Button loginButton = new Button("Login");  
 loginButton.setPrefWidth(100);  
 loginButton.setStyle("-fx-background-color: #1976D2; -fx-text-fill: white; -fx-font-weight: bold;");  
 loginButton.setOnAction(e -> *handleLogin*(emailField.getText(), passwordField.getText(), loginStage));  
  
 Button cancelButton = new Button("Cancel");  
 cancelButton.setPrefWidth(100);  
 Button forgotPassword= new Button("Forgot Password");  
 forgotPassword.setOnAction(e->{  
 NavigationHelper.*openForgotPasswordWindow*();  
 });  
 cancelButton.setStyle("-fx-background-color: #1976D2; -fx-text-fill: white;-fx-font-weight: bold;");  
 cancelButton.setOnAction(e -> loginStage.close());  
 forgotPassword.setStyle("-fx-background-color: red; -fx-text-fill: white;-fx-font-weight: bold;");  
  
 layout.getChildren().addAll(loginLabel,  
 emailField,  
 passwordField,  
 loginButton,  
 cancelButton, forgotPassword  
 );  
  
 Scene loginScene = new Scene(layout, 400, 250);  
 loginStage.setScene(loginScene);  
 loginStage.show();  
 }  
  
 private static void handleLogin(String email, String password, Stage loginStage) {  
 UserFileHandler userFileHandler = new UserFileHandler();  
  
 if (userFileHandler.validateCredentials(email, password)) {  
 *isLoggedIn* = true;  
 *loggedInUser* = email;  
 *showAlert*(Alert.AlertType.*INFORMATION*, "Login Success", "Welcome, " + email + "!");  
 loginStage.close();  
 } else {  
 *showAlert*(Alert.AlertType.*ERROR*, "Login Failed", "Invalid email or password. Please try again.");  
 }  
 }  
  
 public static void checkLoginBeforeAction(Runnable action) {  
 if (!*isLoggedIn*) {  
 Alert alert = new Alert(Alert.AlertType.*WARNING*);  
 alert.setTitle("Authentication Required");  
 alert.setHeaderText(null);  
 alert.setContentText("You need to log in to perform this action.");  
 alert.showAndWait();  
 } else {  
 action.run();  
 }  
 }  
  
  
 private static void showAlert(Alert.AlertType type, String title, String content) {  
 Alert alert = new Alert(type);  
 alert.setTitle(title);  
 alert.setHeaderText(null);  
 alert.setContentText(content);  
 alert.showAndWait();  
 }  
  
 public static void openPaymentInfoForm(Stage primaryStage) {  
 Stage paymentStage = new Stage();  
 paymentStage.setTitle("Payment Information");  
 GridPane layout = new GridPane();  
 layout.setStyle("-fx-background-color: #2c3e50;-fx-font-weight: bold;");  
 layout.setAlignment(Pos.*CENTER*);  
 layout.setVgap(15);  
 layout.setHgap(15);  
 layout.setPadding(new Insets(20));  
  
  
 Label cardNumberLabel = new Label("Card Number:");  
 cardNumberLabel.setTextFill(Color.*WHITE*);  
 TextField cardNumberField = new TextField();  
 cardNumberField.setPromptText("Enter your card number");  
 cardNumberField.setStyle("-fx-background-color: #3A3A3A; -fx-border-color: #1976D2; -fx-border-radius: 5px; -fx-text-fill: white; -fx-font-weight: bold;");  
  
 Label expiryDateLabel = new Label("Expiry Date (MM/YY):");  
 expiryDateLabel.setTextFill(Color.*WHITE*);  
 TextField expiryDateField = new TextField();  
 expiryDateField.setPromptText("Enter expiry date");  
 expiryDateField.setStyle("-fx-background-color: #3A3A3A; -fx-border-color: #1976D2; -fx-border-radius: 5px; -fx-text-fill: white; -fx-font-weight: bold;");  
  
 Label cvvLabel = new Label("CVV:");  
 cvvLabel.setTextFill(Color.*WHITE*);  
 TextField cvvField = new TextField();  
 cvvField.setPromptText("Enter CVV");  
 cvvField.setStyle("-fx-background-color: #3A3A3A; -fx-border-color: #1976D2; -fx-border-radius: 5px; -fx-text-fill: white; -fx-font-weight: bold;");  
  
 Button submitButton = new Button("Submit Payment Info");  
 submitButton.setStyle("-fx-background-color: #1976D2; -fx-text-fill: white; -fx-font-weight: bold;");  
 submitButton.setOnAction(e -> {  
 String cardNumber = cardNumberField.getText();  
 String expiryDate = expiryDateField.getText();  
 String cvv = cvvField.getText();  
  
 if (cardNumber.isEmpty() || expiryDate.isEmpty() || cvv.isEmpty()) {  
  
 Alert alert = new Alert(Alert.AlertType.*WARNING*, "Please fill in all payment details.");  
 alert.showAndWait();  
 } else {  
  
 UserFileHandler.*savePaymentInfoToFile*(cardNumber, expiryDate, cvv);  
 paymentStage.close();  
 }  
 });  
  
  
 Button cancelButton = new Button("Cancel");  
 cancelButton.setStyle("-fx-background-color: #FF3D00; -fx-text-fill: white;");  
 cancelButton.setOnAction(e -> paymentStage.close());  
  
  
 layout.add(cardNumberLabel, 0, 0);  
 layout.add(cardNumberField, 1, 0);  
 layout.add(expiryDateLabel, 0, 1);  
 layout.add(expiryDateField, 1, 1);  
 layout.add(cvvLabel, 0, 2);  
 layout.add(cvvField, 1, 2);  
 layout.add(submitButton, 0, 3);  
 layout.add(cancelButton, 1, 3);  
  
 Scene scene = new Scene(layout, 400, 400);  
 paymentStage.setScene(scene);  
 paymentStage.show();  
  
 }  
  
 public static void openBookingConfirmationWindow(Stage primaryStage) {  
 }  
  
 public static void openForgotPasswordWindow() {  
 Stage forgotStage = new Stage();  
 forgotStage.setTitle("Forgot Password");  
  
 VBox layout = new VBox(10);  
 layout.setPadding(new Insets(20));  
 layout.setAlignment(Pos.*CENTER*);  
 layout.setStyle("-fx-background-color: #2c3e50;-fx-font-weight: bold;");  
 Label resetLabel=new Label("Reset your password");  
 resetLabel.setStyle("-fx-text-fill: white;-fx-text-fill: white;");  
 resetLabel.setFont(Font.*font*("Arial", 20));  
 TextField emailField = new TextField();  
 emailField.setPrefHeight(35);  
 emailField.setStyle("-fx-border-radius: 5px;-fx-border-color: #1976D2;-fx-background-color: #3A3A3A;");  
 emailField.setPromptText("Enter your registered email");  
  
 PasswordField newPasswordField = new PasswordField();  
 newPasswordField.setPrefHeight(35);  
 newPasswordField.setPromptText("Enter new password");  
 newPasswordField.setStyle("-fx-border-radius: 5px;-fx-border-color: #1976D2;-fx-background-color: #3A3A3A;");  
  
 Button resetButton = new Button("Reset Password");  
 resetButton.setStyle("-fx-background-color: #1976D2; -fx-text-fill: white; -fx-font-weight: bold;-fx-border-radius: 5px;");  
 resetButton.setOnAction(e -> {  
 String email = emailField.getText();  
 String newPassword = newPasswordField.getText();  
  
 if (email.isEmpty() || newPassword.isEmpty()) {  
 *showAlert*(Alert.AlertType.*WARNING*, "Validation Error", "Please fill in all fields.");  
 return;  
 }  
  
 if (!newPassword.matches(".{6,}")) { // Simple password strength check  
 *showAlert*(Alert.AlertType.*WARNING*, "Weak Password", "Password must be at least 6 characters long.");  
 return;  
 }  
  
 if (UserFileHandler.*resetPassword*(email, newPassword)) {  
 *showAlert*(Alert.AlertType.*INFORMATION*, "Password Reset", "Your password has been reset successfully.");  
 forgotStage.close();  
 } else {  
 *showAlert*(Alert.AlertType.*ERROR*, "Reset Failed", "Email not found. Please try again.");  
 }  
 });  
  
 Button cancelButton = new Button("Cancel");  
 cancelButton.setPrefHeight(20);  
 cancelButton.setPrefWidth(100);  
 cancelButton.setStyle("-fx-background-color: #1976D2; -fx-text-fill: white; -fx-font-weight: bold;-fx-border-radius: 5px;");  
 cancelButton.setOnAction(e -> forgotStage.close());  
  
 layout.getChildren().addAll(resetLabel,  
 emailField,  
 newPasswordField,  
 resetButton,  
 cancelButton  
 );  
  
 Scene forgotScene = new Scene(layout, 350, 300);  
 forgotStage.setScene(forgotScene);  
 forgotStage.show();  
}  
}

**Flight:**

package com.example.projectapp;  
  
import javafx.beans.property.\*;  
  
public class Flight {  
  
 private final StringProperty flightID;  
 private final StringProperty departureCity;  
 private final StringProperty destinationCity;  
 private final StringProperty flightClass;  
 private final StringProperty date;  
 private final DoubleProperty price;  
  
 public Flight(String flightID, String departureCity, String destinationCity, String flightClass, String date, double price) {  
 this.flightID = new SimpleStringProperty(flightID);  
 this.departureCity = new SimpleStringProperty(departureCity);  
 this.destinationCity = new SimpleStringProperty(destinationCity);  
 this.flightClass = new SimpleStringProperty(flightClass);  
 this.date = new SimpleStringProperty(date);  
 this.price = new SimpleDoubleProperty(price);  
 }  
  
 public String getFlightID() {  
 return flightID.get();  
 }  
  
 public void setFlightID(String flightID) {  
 this.flightID.set(flightID);  
 }  
  
 public StringProperty flightIDProperty() {  
 return flightID;  
 }  
  
 public String getDepartureCity() {  
 return departureCity.get();  
 }  
  
 public void setDepartureCity(String departureCity) {  
 this.departureCity.set(departureCity);  
 }  
  
 public StringProperty departureCityProperty() {  
 return departureCity;  
 }  
  
 public String getDestinationCity() {  
 return destinationCity.get();  
 }  
  
 public void setDestinationCity(String destinationCity) {  
 this.destinationCity.set(destinationCity);  
 }  
  
 public StringProperty destinationCityProperty() {  
 return destinationCity;  
 }  
  
 public String getFlightClass() {  
 return flightClass.get();  
 }  
  
 public void setFlightClass(String flightClass) {  
 this.flightClass.set(flightClass);  
 }  
  
 public StringProperty flightClassProperty() {  
 return flightClass;  
 }  
  
 public String getDate() {  
 return date.get();  
 }  
  
 public void setDate(String date) {  
 this.date.set(date);  
 }  
  
 public StringProperty dateProperty() {  
 return date;  
 }  
  
 public double getPrice() {  
 return price.get();  
 }  
  
 public void setPrice(double price) {  
 this.price.set(price);  
 }  
  
 public DoubleProperty priceProperty() {  
 return price;  
 }  
  
 @Override  
 public String toString() {  
 return "Flight ID: " + flightID.get() +  
 ", Departure: " + departureCity.get() +  
 ", Destination: " + destinationCity.get() +  
 ", Class: " + flightClass.get() +  
 ", Date: " + date.get() +  
 ", Price: $" + price.get();  
}  
}