Project Documentation

Project Title : Flight Reservation System in C++

Student Name : Hamza Butt.

Course: Programming Fundamentals.

Course Instructor: Dr. Sheeraz, Mr. Mansoor.

University : Salim Habib University.

Semester/Section : Fall 2024,

BS Computer Science,

Semester 1.

GitHub Repository Link

26 Jan, 2025

Introduction

Project Overview:

The Flight Reservation System is a console-based C++ program that enables users to:

- Book a flight
- Cancel a flight
- View their bookings

The system utilizes file handling to store and manage flights and customer bookings.

Purpose:

This project was chosen to enhance understanding of file handling, exception handling, and fundamental C++ programming concepts while building a real-world application.

Objectives

- To develop a functional flight reservation system using C++.
- To implement file handling for data storage and retrieval.
- To practice exception handling for better error management.
- To reinforce knowledge of functions, loops, and conditional statements.

Tools and Technologies

- **Programming Language** : C++
- **IDE** : Dev-C++
- **File Handling** : Text-based storage (flight.txt , bookings.txt)
- **Concepts Used** : Loops, Functions, File Handling, Exception Handling

Project Design

Workflow:

1. The user selects an option: Book Flight, Cancel Flight, or View Bookings.
2. If booking a flight:
    - The system asks for departure and arrival locations.
    - It searches flight.txt for matching flights.
    - The user selects a flight and class.
    - The booking is stored in bookings.txt .
3. If canceling a flight:
    - The user provides booking details.
    - The system removes the booking from bookings.txt .
4. If viewing bookings:
    - The system retrieves and displays bookings from bookings.txt .

Implementation

- The project consists of a single source file (flight.cpp).
- **File Handling** : ifstream and ofstream are used for reading and writing data.
- **Functions** :
    - bookFlight() : Handles flight booking.
    - cancelFlight() : Manages booking cancellations.
    - viewBookings() : Displays user bookings.
- **Error Handling** : Try-catch blocks handle file I/O errors and invalid inputs.

Testing

Test Cases:

Input	Expected Output	Actual Output
Book flight (LHE -> KHI)	Flight options displayed, booking confirmed	Same
Cancel booking (Valid ID)	Booking successfully canceled	Same
Cancel booking (Invalid ID)	Error: Booking not found	Same
View bookings	List of bookings displayed	Same

Results

The program successfully allows users to book, cancel, and view flights while maintaining data using file handling. The system performs well and meets all requirements.

```
Available Flights:

   Airline      Departure -> Arrival    Date    Time
-------------------------------------------------------------------------
1.  Emirates,  Dubai -> Pakistan,   25-05-2025 18:56
2.  PIA,        Karachi -> Doha,   25-08-2025
3.  ETIHAD,     Islamabad -> Sharjah, 25-01-2025 22:56
4.  FlyJinnah,  Karachi -> Seattle,     26-02-2025 14:00
```

Conclusion

This project helped in understanding:

- File handling and exception management in C++.
- The importance of structured code using functions.
- Real-world applications of C++ programming concepts.

The project can be further enhanced by integrating a GUI and database support in the future.

Source Code

```
#include <iostream>
#include <iomanip>
#include<conio.h>  // For getch()
#include <fstream>
#include<windows.h>  // For system("cls") and Sleep()
#include <string>
#include <ctime>  // For generating unique booking IDs
#include <sstream> // For converting data to strings
#include <stdexcept> // Include for exceptions
using namespace std;



// Constants for rows and columns in each class
const int FIRST_ROWS = 2, FIRST_COLS = 3;
const int BUSINESS_ROWS = 4, BUSINESS_COLS = 6;
const int ECONOMY_ROWS = 10, ECONOMY_COLS = 6;



// Function declarations
void seats(char seats[][10], int rows, int cols);
void displaySeats(char seats[][10], int rows, int cols, const string &className, int aislePos1, int aislePos2);
bool bookSeat(char seats[][10], int rows, int cols, const string &className, int flightIndex);
bool cancelSeat(char seats[][10], int rows, int cols, const string &className);
void displayFlights();
void admin();
string generateBookingID();




int main(){

system("color 71");

    // Seats for each class
    char firstClassSeats[FIRST_ROWS][10];
    char businessClassSeats[BUSINESS_ROWS][10];
    char economyClassSeats[ECONOMY_ROWS][10];
```

```
login();
system("cls");

    // Initialize seats to available ('O')
    seats(firstClassSeats, FIRST_ROWS, FIRST_COLS);
    seats(businessClassSeats, BUSINESS_ROWS, BUSINESS_COLS);
    seats(economyClassSeats, ECONOMY_ROWS, ECONOMY_COLS);

    int choice;
    do{

        // Prompt user for action
        cout << "\nWhat would you like to do?" << endl;
        cout << "1. View Flight Schedules" << endl;
        cout << "2. Book a Seat" << endl;
        cout << "3. Cancel a Seat" << endl;
        cout << "4. Admin Menu" << endl;
        cout << "5. Exit" << endl;
        cout << "\nEnter your choice (1-5): ";
        cin >> choice;

        switch(choice){
            case 1:{
                    system("cls");
            try{
                    displayFlights();
                            }
                            catch (const exception& e){
                                    cout << "Error: " << e.what() << endl;
                            }
                            break;
                    }

           case 2:{

               // Booking a seat

               // Display seat availability
               try{

                       system("cls");
                       displayFlights();
```

```
        int flightIndex;
cout << "\nEnter the Flight Number you want to book (enter index): ";
cin >> flightIndex;

if (cin.fail() || flightIndex <= 0) {
   throw runtime_error("Invalid flight index input.");
}

ifstream flightFile("flights.txt");
int numFlights = 0;
string line;
while(getline(flightFile, line))
   numFlights++;
flightFile.close();

if (flightIndex > numFlights) {

   cout << "\n\tInvalid flight index. There are only " << numFlights << " flight(s) available." << endl;
   cout << "\n\t\t\t\t\tPress any key to continue...";
   getch();
   system("cls");
   break;
}

        displaySeats(firstClassSeats, FIRST_ROWS, FIRST_COLS, "First Class", 1, 2);
        displaySeats(businessClassSeats, BUSINESS_ROWS, BUSINESS_COLS, "Business Class", 2, 4);
        displaySeats(economyClassSeats, ECONOMY_ROWS, ECONOMY_COLS, "Economy Class", 3, -1);

        int classChoice;
        cout << "\nWhich class do you want to book a seat in?" << endl;
        cout << "1. First Class" << endl;
        cout << "2. Business Class" << endl;
        cout << "3. Economy Class" << endl;
        cout << "\nEnter your choice (1/2/3): ";
        cin >> classChoice;


        if(classChoice == 1){
   bookSeat(firstClassSeats, FIRST_ROWS, FIRST_COLS, "First Class", flightIndex);
}
                        else if(classChoice == 2){
   bookSeat(businessClassSeats, BUSINESS_ROWS, BUSINESS_COLS, "Business Class", flightIndex);
}
                        else if(classChoice == 3){
```

```
            bookSeat(economyClassSeats, ECONOMY_ROWS, ECONOMY_COLS, "Economy Class", flightIndex);
        }
                                    else{
            cout << "Invalid choice! Please try again." << endl;
        }
    }
                                catch (const exception& e) {
        cout << "Error: " << e.what() << endl;
    }
    break;

}

case 3:{
    // Canceling a seat

    system("cls");
    displayFlights();

    int flightIndex;
        cout << "\nEnter the Flight Number you want to book (enter index): ";
        cin >> flightIndex;

        if(cin.fail() || flightIndex <= 0) {
            throw runtime_error("Invalid flight index input.");
        }

        ifstream flightFile("flights.txt");
        int numFlights = 0;
        string line;
        while(getline(flightFile, line))
            numFlights++;
        flightFile.close();

        if (flightIndex > numFlights){

            cout << "\n\tInvalid flight index. There are only " << numFlights << " flight(s) available." << endl;
            cout << "\n\t\t\t\t\tPress any key to continue...";
            getch();
            system("cls");
            break;
        }
```

```
    // Display seat availability
    displaySeats(firstClassSeats, FIRST_ROWS, FIRST_COLS, "First Class", 1, 2);
    displaySeats(businessClassSeats, BUSINESS_ROWS, BUSINESS_COLS, "Business Class", 2, 4);
    displaySeats(economyClassSeats, ECONOMY_ROWS, ECONOMY_COLS, "Economy Class", 3, -1);
    int classChoice;
    cout << "\nWhich class do you want to cancel a seat in?" << endl;
    cout << "1. First Class" << endl;
    cout << "2. Business Class" << endl;
    cout << "3. Economy Class" << endl;
    cout << "\nEnter your choice (1/2/3): ";
    cin >> classChoice;


    if(classChoice == 1){
       cancelSeat(firstClassSeats, FIRST_ROWS, FIRST_COLS, "First Class");
    }
                            else if(classChoice == 2){
       cancelSeat(businessClassSeats, BUSINESS_ROWS, BUSINESS_COLS, "Business Class");
    }
                            else if(classChoice == 3){
       cancelSeat(economyClassSeats, ECONOMY_ROWS, ECONOMY_COLS, "Economy Class");
    }
                            else{
       cout << "Invalid choice! Please try again." << endl;
    }
    break;
}
case 4:
    system("cls");
    try{
                admin();
                            }
                            catch (const exception& e){
                                        cout << "Error: " << e.what() << endl;
                            }

    break;


                case 5:
    cout << "Exiting the program. ThankYou for using our Flight Reservation System. Goodbye!" << endl;
    return 0;
```

```
            default:
                cout << "Invalid choice! Please try again." << endl;
        }

}while(choice!=5);


    return 0;
}

// Initialize seats to 'O' (available)
void seats(char seats[][10], int rows, int cols) {
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            seats[i][j] = 'O';
        }
    }
}

// Seats display
void displaySeats(char seats[][10], int rows, int cols, const string &className, int aislePos1, int aislePos2){
    cout << "\n\n\n\n\t\t\t\t\t\t\t\t";
    cout << className << "\n"<<  endl;
    for (int i = 0; i < rows; ++i) {
        cout << "\t\t\t\t\t\t\t";
        cout << "Row " << setw(2) << i + 1 << ":   ";
        for (int j = 0; j < cols; ++j) {
            if (j == aislePos1 || j == aislePos2) cout << "| "; // Add aisle
            cout << seats[i][j] << " ";
        }
        cout << endl;
    }
}

// Function to generate a unique booking ID
string generateBookingID(){
    srand(time(0));
    int randomID = rand() % 100000;  // Generates a 5-digit random number
    stringstream ss;
    ss << "BK" << randomID;
    return ss.str();
}
```

```
// Booking a Seat
bool bookSeat(char seats[][10], int rows, int cols, const string& className, int flightIndex){
    int row, col;
    string name, cnic, contact;
    cout << "\nEnter the row (1-" << rows << ") and column (1-" << cols << ") to book: ";
    cin >> row >> col;

    if(row <= 0 || row > rows || col <= 0 || col > cols) {
        throw out_of_range("Invalid seat selection. Row and column must be within valid range.");
    }

    if(seats[row - 1][col - 1] == 'X'){
        throw runtime_error("Seat already booked! Please choose another seat.");
    }

    cin.ignore();
    cout << "Enter your name: ";
    getline(cin, name);
    cout << "Enter your CNIC number: ";
    getline(cin, cnic);
    cout << "Enter your contact number: ";
    getline(cin, contact);

    string bookingID = generateBookingID();

    ofstream bookingFile("bookings.txt", ios::app);
    if (!bookingFile) {
        throw runtime_error("Unable to open booking file for writing.");
    }

    bookingFile << bookingID << "," << name << "," << cnic << "," << contact
        << "," << className << "," << row << "," << col << "," << flightIndex << endl;
    bookingFile.close();

    seats[row - 1][col - 1] = 'X';

    system("cls");
    displaySeats(seats, rows, cols, className, (className == "First Class") ? 1 : (className == "Business Class") ? 2 : 3,
(className == "First Class") ? 2 : (className == "Business Class") ? 4 : -1);
    cout << "\n\n\n\t\t\t\t\t\tSeat booked successfully in " << className << "!" << endl;
    cout << "\t\t\t\t\tYour Booking ID: " << bookingID << endl;
    cout << "\t\t\t\t\tThank you for booking with us!\n\n\n" << endl;
    cout << "\n\t\t\t\t\tPress any key to continue...";
    getch();
```

```
    system("cls");

    return true;
}


// Canceling a Seat
bool cancelSeat(char seats[][10], int rows, int cols, const string &className) {
    string bookingID, name, cnic;
    int flightIndexToCancel;
    bool bookingFound = false;

    cout << "Enter your Booking ID: ";
    cin >> bookingID;
    cout << "Enter the Flight Index of the booking to cancel: ";
    cin >> flightIndexToCancel;
    cout << "Enter your name: ";
    getline(cin, name);
    cout << "Enter your CNIC number: ";
    getline(cin, cnic);

    ifstream bookingFile("bookings.txt");
    ofstream tempFile("temp.txt");


    if(!bookingFile || !tempFile) {
        cout << "Error accessing booking file!" << endl;
        return false;
    }
    system("cls");

    string line;
    while (getline(bookingFile, line)) {
        stringstream ss(line);
        string id, bookedName, bookedCNIC, contact, bookedClass;
        int bookedRow, bookedCol, bookedFlightIndex;

        getline(ss, id, ',');
        getline(ss, bookedName, ',');
        getline(ss, bookedCNIC, ',');
        getline(ss, contact, ',');
        getline(ss, bookedClass, ',');
        ss >> flightIndexToCancel >> bookedRow >> bookedCol;
```

```
        if(id == bookingID && bookedName == name && bookedCNIC == cnic && bookedClass == className &&
bookedFlightIndex == flightIndexToCancel){
            bookingFound = true;
            if(className == bookedClass) {
                seats[bookedRow - 1][bookedCol - 1] = 'O'; // Mark seat as available
                cout << "\n\n\n\t\t\t\t\t\tBooking successfully canceled!" << endl;
            }
                            else{
                tempFile << line << endl; // Keep the line if the class doesn't match
            }
        }
                else {
            tempFile << line << endl; // Copy non-matching lines
        }
    }

    bookingFile.close();
    tempFile.close();

    if(bookingFound){
        remove("bookings.txt");
        rename("temp.txt", "bookings.txt");
    }
else{
        remove("temp.txt");
        cout << "Booking not found or details incorrect!" << endl;
    }

    cout << "\n\n\t\tPress any key to continue...";
    getch();
    system("cls");

    return bookingFound;
}

// Display flight schedules
void displayFlights(){
    ifstream flightFile("flights.txt");
    if(!flightFile) {
        throw runtime_error("Unable to open flights file.");
    }

    cout << "\n\tAvailable Flights:" << endl;
    cout << "\n\t   Airline\tDeparture -> Arrival\tDate\tTime" << endl;
```

```
     cout << "\t-------------------------------------------------------------------------" << endl;

    string line;
    int flightNumber = 1;
    while (getline(flightFile, line)) {
        cout << "\t" << flightNumber << ".  ";
        cout << line << endl;
        flightNumber++;
    }


    flightFile.close();
}



// Admin menu to view bookings
void admin(){
    const string ADMIN_USERNAME = "admin123";
    const string ADMIN_PASSWORD = "password456";

    string username, password;
    cout << "\nEnter admin username: ";
    cin >> username;
    cout << "Enter admin password: ";
    cin >> password;

    if(username != ADMIN_USERNAME || password != ADMIN_PASSWORD) {
        cout << "\nInvalid credentials! Returning to main menu." << endl;
        return;
    }

    int adminChoice;
    while(true){
        cout << "\nAdmin Menu:" << endl;
        cout << "1. Add Flight" << endl;
        cout << "2. Remove Flight" << endl;
        cout << "3. View Bookings" << endl;
        cout << "4. Back to Main Menu" << endl;
        cout << "\nEnter your choice (1-4): ";
        cin >> adminChoice;

        switch(adminChoice){
            case 1:{
```

```
        // Add a new flight
    ofstream flightFile("flights.txt", ios::app);
    if(!flightFile) {
        throw runtime_error("Unable to open flights file for writing.");
    }

    string airline, departure, arrival, dateTime;
    cout << "\nEnter Airline: ";
    cin.ignore();
    getline(cin, airline);

    cout << "Enter Departure Airport: ";
    getline(cin, departure);

    cout << "Enter Arrival Airport: ";
    getline(cin, arrival);


    cout << "Enter Date and Time (DD-MM-YYYY HH:MM): ";
    getline(cin, dateTime);



    flightFile << airline << ",\t" << departure << " -> " << arrival << ",\t" << dateTime << endl;
    cout << "Flight added successfully!" << endl;

    flightFile.close();
    break;
}
case 2:{
    // Remove a flight
    ifstream flightFile("flights.txt");
    if(!flightFile) {
        throw runtime_error("Unable to create temporary file.");
    }

    ofstream tempFile("temp.txt");
    if(!tempFile) {
        throw runtime_error("Unable to create temporary file.");
    }

    displayFlights();

    string line, flightToRemove;
```

```
    cout << "\nEnter details of the flight to remove: ";
    cin.ignore();
    getline(cin, flightToRemove);

    bool found = false;

    while(getline(flightFile, line)) {
        if (line == flightToRemove) {
            found = true;
        }
                                        else{
            tempFile << line << endl;
        }
    }

    flightFile.close();
    tempFile.close();

    if(found){
        remove("flights.txt");
        rename("temp.txt", "flights.txt");
        cout << "Flight removed successfully!" << endl;
    }
                                else{
        remove("temp.txt");
        cout << "Flight not found!" << endl;
    }

    break;
}
case 3:{
    ifstream bookingFile("bookings.txt");

    if(!bookingFile) {
        throw runtime_error("Unable to open booking file.");
    }

    cout << "\nBookings List:" << endl;
    string line;
    while (getline(bookingFile, line)) {
        cout << line << endl;
    }

    bookingFile.close();
```

```
            break;
        }
        case 4:
            system("cls");
            return;


        default:
            cout << "Invalid choice! Please try again." << endl;
    }
  }
}
```