University of Calgary

ENSF 337 : Programming Fundamentals for Software and Computer

Final Project

Submitted by Group 11:

Simon Ryabinkin

Abdelrahman ElSayed

Maxime Boucher

Date of Submission: December 1, 2025

Header Files

Airline.h
| ```cpp
// include/classes/Airline.h
#ifndef AIRLINE_H
#define AIRLINE_H

#include <vector>
#include <string>
#include "Flight.h"
using std::string;
using std::vector;

class Airline {
private:
    string name;
    vector<Flight> flights;
public:
    Airline(string name);
    void set_name(string name);
    string get_name() const;
    const vector<Flight>& get_flights() const;
    Flight& get_flight(int index);
    const Flight& get_flight(int index) const;
    void addFlight(Flight& flight);
};

#endif // AIRLINE_H
``` |

Flight.h

```
// include/classes/Flight.h
#ifndef FLIGHT_H
#define FLIGHT_H

#include <vector>
#include <string>
#include "Passenger.h"
#include "Seat.h"
#include "Route.h"
using std::string;
using std::vector;

class Flight {
private:
    vector<Seat> seats;
    vector<Passenger> passengers;
    int number_of_rows;
    int number_of_seats_per_row;
    Route route;
    string flight_id;
public:
    Flight(string flight_id, int number_of_rows, int number_of_seats_per_row, Route route);
    /* REQUIRES
 *   flight_id
     number_of_rows > 0
     number_of_seats_per_row > 0
     route is a valid Route object
 *
 * PROMISES
 *   creates a Flight object with the given flight ID, number of rows, number of seats per row, and route.
 */

    void addPassenger(Passenger& passenger);
    /*REQUIRES
 *   passenger is a valid Passenger object
     * PROMISES
     *  adds the given Passenger object to the flight's vector of passengers.

     */

    void addSeat(Seat& seat);
    /*REQUIRES
 *   seat  is a valid Seat object
     * PROMISES
     *  adds the given Saet object to the flight's vector of seats.

     */
```

```cpp
    const vector<Seat>& get_seats() const;

    int get_number_of_rows() const;
    int get_number_of_seats_per_row() const;
    Route get_route() const;
    string get_flight_id() const;
    const vector<Passenger>& get_passengers() const;
    void removePassenger(const string& passenger_id);
    /*REQUIRES
    passenger id is a string
    PROMISES
    removed passenger and a corresponding seat
    */
};

#endif // FLIGHT_H
```

Passenger.h

```cpp
// include/classes/Passenger.h
#ifndef PASSENGER_H
#define PASSENGER_H

#include <string>
using std::string;

class Passenger {
private:
    string passenger_id;
    string first_name;
    string last_name;
    string phone_number;
public:
    Passenger(string passenger_id, string f_name, string l_name, string p_number);
    string get_passenger_id() const;
    string get_first_name() const;
    string get_last_name() const;
    string get_phone_number() const;
};

#endif // PASSENGER_H
```

Route.h

```
// include/classes/Route.h
#ifndef ROUTE_H
#define ROUTE_H

#include <string>
using std::string;

class Route {
private:
   string source;
   string destination;
public:
   Route(string source, string destination);
   string get_source() const;
   string get_destination() const;
};

#endif // ROUTE_H
```

Seat.h
| ```cpp
// include/classes/Seat.h
#ifndef SEAT_H
#define SEAT_H

#include <string>
using std::string;

class Seat {
private:
   int row_number;
   char seat_character;
   string passenger_id;
public:
   Seat(string seat_id, string passenger_id);
   Seat(int row_number, char seat_character, string passenger_id);
   string get_passenger_id() const;
   int get_row_number() const;
   char get_seat_character() const;
};

#endif // SEAT_H
``` |

data_io.h

```
// include/utils/data_io.h
#ifndef DATA_IO_H
#define DATA_IO_H

#include <vector>
#include "../classes/Airline.h"
#include "../classes/Flight.h"
using std::vector;

Airline* createAirline(const string& airline_name);
//REQUIRES:
// airline_name is a valid string
// PROMISES:
// Creates a new Airline instance with the provided name
// Populates Airline with Flights, Routes, Passengers and Seats information by reading data files
// Prints error messages if input files can't be opened
// Returns a pointer to the created airline instance

void saveData(vector<Flight> flights);
//REQUIRES:
// flights is a valid vector of Flight objects
// PROMISES:
// Prompts user on if they would like to save the passengers data to a file
// Reprompts on invalid input
// Prints error message if output file can't be opened
// Stores all passengers information in a file "passengers.txt"

#endif // DATA_IO_H
```

flight_operations.h

```cpp
// include/utils/flight_operations.h
#ifndef FLIGHT_OPERATIONS_H
#define FLIGHT_OPERATIONS_H

#include <vector>
#include "../classes/Flight.h"
using std::vector;

int selectFlight(const vector<Flight>& flights);
// REQUIRES:
// flights is a valid vector of Flight objects
// PROMISES:
// Displays a numbered list of flights
// Prompts the user to select a flight from the displayed list
// Reprompts the user on invalid inputs
// Displays the user's choice after they pick

void displaySeatMap(Flight& flight);
// REQUIRES:
// flight is a valid Flight object
// PROMISES:
// Displays a seat map for the selected flight indicating whether a seat is empty or occupied

void displayPassengerInfo(Flight& flight);
// REQUIRES:
// flight is a valid Flight object
// PROMISES:
// Displays info for all Passengers on the selected flight

void addNewPassenger(Flight& flight);
// REQUIRES:
// flight is a valid Flight object
// PROMISES:
// Prompts user for new passenger and seat information to be added to the selected flight
// Checks that passenger doesn't already exist through the passenger ID
// Reprompts on invalid inputs
// Inserts new Passenger and Seat instances if successful

void removeExistingPassenger(Flight& flight);
// REQUIRES:
// flight is a valid Flight object
// PROMISES:
// Prompts the user for the ID of a passenger to be removed from the selected flight
// Checks if the passenger exists
// Displays a confirmation message that the passenger was removed

#endif // FLIGHT_OPERATIONS_H
```

ui_utils.h

// include/utils/ui_utils.h
#ifndef UI_UTILS_H
#define UI_UTILS_H
 	
void cleanStandardInputStream(void);
// PROMISES:
// Clears the standard inpuit stream
 	
void clearScreen(void);
// PROMISES:
// Clears the terminal/screen for new output
 	
void pressEnter();
// PROMISES:
// Prompts the user to press enter before continuing with the program
 	
void displayHeader();
// PROMISES:
// Displays the start screen for the program
 	
void printChoicePrompt();
// PROMISES:
// Displays the options for the user to select from
 	
int menu();
// PROMISES:
// Returns user menu choice as an int
// Reprompts user on invalid menu inputs
 	
#endif // UI_UTILS_H

.cpp Files

Airline.cpp

```cpp
// src/classes/Airline.cpp
#include "../../include/classes/Airline.h"
#include <stdexcept>
#include <string>
using namespace std;

Airline::Airline(string name) {
    set_name(name);
    flights.clear();
}

void Airline::set_name(string input_name) {
    name = input_name;
}

string Airline::get_name() const {
    return name;
}

const vector<Flight>& Airline::get_flights() const {
    return flights;
}

Flight& Airline::get_flight(int index) {
    if (index < 0 || index >= flights.size()) {
        throw out_of_range("Flight index " + to_string(index) + " is out of range.");
    }
    return flights[index];
}

void Airline::addFlight(Flight& flight) {
    flights.push_back(flight);
    return;
}
```

```cpp
// src/classes/Flight.cpp
#include "../../include/classes/Flight.h"
#include <stdexcept>
using namespace std;

Flight::Flight(string flight_id, int number_of_rows, int number_of_seats_per_row, Route route)
   : flight_id(flight_id), number_of_rows(number_of_rows),
number_of_seats_per_row(number_of_seats_per_row), route(route)
{}

void Flight::addPassenger(Passenger& passenger) {
   passengers.push_back(passenger);
}

void Flight::removePassenger(const string& passenger_id) {
   for (size_t i = 0; i < passengers.size(); i++) {
      if (passengers[i].get_passenger_id() == passenger_id) {
         passengers.erase(passengers.begin() + i);
         // Remove corresponding seat
         for (size_t j = 0; j < seats.size(); j++) {
            if (seats[j].get_passenger_id() == passenger_id) {
               seats.erase(seats.begin() + j);
               break;
            }
         }
         return;
      }
   }
}

void Flight::addSeat(Seat& seat) {
   seats.push_back(seat);
}

const vector<Seat>& Flight::get_seats() const {
   return seats;
}

int Flight::get_number_of_rows() const {
   return number_of_rows;
}

int Flight::get_number_of_seats_per_row() const {
   return number_of_seats_per_row;
}

Route Flight::get_route() const {
```

```
    return route;
}

string Flight::get_flight_id() const {
    return flight_id;
}

const vector<Passenger>& Flight::get_passengers() const {
    return passengers;
}
```

Passenger.cpp

```cpp
// src/classes/Passenger.cpp
#include "../../include/classes/Passenger.h"
using namespace std;

Passenger::Passenger(string passenger_id, string f_name, string l_name, string p_number)
    : passenger_id(passenger_id), first_name(f_name), last_name(l_name), phone_number(p_number)
{}

string Passenger::get_passenger_id() const {
    return passenger_id;
}

string Passenger::get_first_name() const {
    return first_name;
}

string Passenger::get_last_name() const {
    return last_name;
}

string Passenger::get_phone_number() const {
    return phone_number;
}
```

Route.cpp

```cpp
// src/classes/Route.cpp
#include "../../include/classes/Route.h"
using namespace std;

Route::Route(string source, string destination)
   : source(source), destination(destination)
{}

string Route::get_source() const {
   return source;
}

string Route::get_destination() const {
   return destination;
}
```

Seat.cpp

```cpp
// src/classes/Seat.cpp
#include "../../include/classes/Seat.h"
#include <string>
using namespace std;

Seat::Seat(string seat_id, string passenger_id)
  : passenger_id(passenger_id)
{
  // Convert seat ID to row number and seat character
  // Find where the numeric part ends
  int num_end = 0;
  while (num_end < seat_id.length() && seat_id[num_end] >= '0' && seat_id[num_end] <= '9') {
    num_end++;
  }

  // Get row number
  if (num_end > 0) {
    row_number = stoi(seat_id.substr(0, num_end));
  } else {
    row_number = 0;
  }

  // Get seat character
  if (num_end < seat_id.length()) {
    seat_character = seat_id[num_end];
  } else {
    seat_character = '\0';
  }
}

Seat::Seat(int row_number, char seat_character, string passenger_id)
  : row_number(row_number), seat_character(seat_character), passenger_id(passenger_id)
{}

string Seat::get_passenger_id() const {
  return passenger_id;
}

int Seat::get_row_number() const {
  return row_number;
}

char Seat::get_seat_character() const {
  return seat_character;
}
```

data_io.cpp

```cpp
// src/utils/data_io.cpp
#include "../../include/utils/data_io.h"
#include "../../include/classes/Passenger.h"
#include "../../include/classes/Seat.h"
#include "../../include/classes/Route.h"
#include "../../include/classes/Flight.h"
#include "../../include/classes/Airline.h"
#include "../../include/utils/ui_utils.h"
#include <iostream>
#include <fstream>
#include <sstream>
#include <iomanip>
#include <string>
#include <vector>
using namespace std;

Airline* createAirline(const string& airline_name) {
    Airline* airline = new Airline(airline_name);

    // Open data files
    ifstream flightFile("data/flights.txt");
    if (!flightFile) {
        cerr << "Error opening flights.txt file." << endl;
        exit(1);
    }
    ifstream passengerFile("data/passengers.txt");
    if (!passengerFile) {
        cerr << "Error opening passengers.txt file." << endl;
        exit(1);
    }

    string line;
    while (getline(flightFile, line)) {
        int number_of_rows, number_of_seats_per_row;
        string flight_id, flight_departure, flight_destination;
        stringstream iss(line);
        iss >> flight_id >> flight_departure >> flight_destination >> number_of_rows >>
number_of_seats_per_row;
        Route route = Route(flight_departure, flight_destination);
        Flight flight = Flight(flight_id, number_of_rows, number_of_seats_per_row, route);

        // Add passengers and seats to flight
        // Reset passenger file pointer to beginning for each flight
        passengerFile.clear();
        passengerFile.seekg(0);
        while (getline(passengerFile, line)) {
            string passenger_flight_id, first_name, last_name, phone_number, seat_id, passenger_id;
```

```cpp
        stringstream iss(line);
        iss >> passenger_flight_id >> first_name >> last_name >> phone_number >> seat_id >>
passenger_id;
        Passenger passenger = Passenger(passenger_id, first_name, last_name, phone_number);
        Seat seat = Seat(seat_id, passenger_id);
        if (passenger_flight_id == flight_id) {
            flight.addPassenger(passenger);
            flight.addSeat(seat);
        }
    }

    airline->addFlight(flight);
  }
  flightFile.close();
  passengerFile.close();
  return airline;
}

void saveData(vector<Flight> flights) {
  char answer;
  while (true) {
    cout << "Do you want to save the data to \"data/passengers.txt\"? Please answer <Y or N>: ";
    cin >> answer;
    cleanStandardInputStream();
    if (answer == 'Y') break;
    else if (answer == 'N') return;
    else {
      cout << "\nInvalid answer. Please try again.\n" << endl;
    }
  }

  ofstream passengerFile("data/passengers.txt");
  if (!passengerFile) {
    cerr << "Error opening passengers.txt file." << endl;
    exit(1);
  }

  // Iterate through all flights and save passengers from each flight
  bool first_passenger = true;
  for (const Flight& flight : flights) {
    vector<Passenger> passengers = flight.get_passengers();
    vector<Seat> seats = flight.get_seats();

    for (const Passenger& passenger : passengers) {
      // Find the passenger's seat
      string seat_id = "";
      for (const Seat& seat : seats) {
        if (seat.get_passenger_id() == passenger.get_passenger_id()) {
```

```cpp
                seat_id = to_string(seat.get_row_number()) + seat.get_seat_character();
                break;
            }
        }

        if (!first_passenger) {
            passengerFile << endl;
        }
        first_passenger = false;

        passengerFile << left << setw(8) << flight.get_flight_id()
                << setw(8) << passenger.get_first_name()
                << setw(10) << passenger.get_last_name()
                << setw(16) << passenger.get_phone_number()
                << setw(6) << seat_id
                << passenger.get_passenger_id();
        }
    }
    passengerFile.close();

    cout << "\nAll the data in the passenger list were saved." << endl;
}
```

```cpp
// src/utils/flight_operations.cpp
#include "../../include/utils/flight_operations.h"
#include "../../include/classes/Flight.h"
#include "../../include/classes/Passenger.h"
#include "../../include/classes/Seat.h"
#include "../../include/utils/ui_utils.h"
#include <iostream>
#include <iomanip>
#include <vector>
#include <stdexcept>
using namespace std;

int selectFlight(const vector<Flight>& flights) {
    cout << "Here is the list of available flights. Please select one: \n" << endl;

    for (int i = 0; i < flights.size(); i++) {
        cout << i + 1 << ". " << left << setw(8) << flights[i].get_flight_id()
            << setw(10) << flights[i].get_route().get_source()
            << setw(10) << flights[i].get_route().get_destination()
            << setw(4) << flights[i].get_number_of_rows()
            << flights[i].get_number_of_seats_per_row() << endl;
    }

    int selected_flight_index;
    while (true) {
        cout << "\nEnter your choice: ";
        cin >> selected_flight_index;

        if (!cin.fail() && selected_flight_index >= 1 && selected_flight_index <= flights.size()) {
            cleanStandardInputStream();
            break;
        }

        cin.clear();
        cleanStandardInputStream();
        cout << "\nInvalid choice. Please enter a number between 1 and " << flights.size() << "." << endl;
    }

    cout << "You have selected flight " << flights[selected_flight_index - 1].get_flight_id()
        << " from " << flights[selected_flight_index - 1].get_route().get_source()
        << " to " << flights[selected_flight_index - 1].get_route().get_destination() << "." << endl;
    return selected_flight_index - 1;
}

void displaySeatMap(Flight& flight) {
    vector<Seat> seats = flight.get_seats();
    int num_rows = flight.get_number_of_rows();
```

```cpp
    int num_cols = flight.get_number_of_seats_per_row();

    // Create a seatMap matrix to access seats by row and column
    vector<vector<bool>> seatMap(num_rows, vector<bool>(num_cols, false));
    for (const Seat& seat : seats) {
        int seat_row = seat.get_row_number();
        char col_char = seat.get_seat_character();
        int map_row = seat_row - 1; // Convert for indexing
        if (map_row >= 0 && map_row < num_rows && col_char >= 'A' && col_char < 'A' + num_cols) {
            int map_col = col_char - 'A';
            seatMap[map_row][map_col] = !seat.get_passenger_id().empty();
        }
    }

    cout << "Aircraft Seat Map for flight " << flight.get_flight_id() << "\n" << endl;

    // Print column headers
    cout << setw(3) << "";
    for (int c = 0; c < num_cols; c++) {
        cout << "  " << (char)('A' + c) << " ";
    }
    cout << endl;

    // Print top border
    cout << right << setw(4) << "+";
    for (int c = 0; c < num_cols; c++) {
        cout << "---+";
    }
    cout << endl;

    // Print seat rows
    for (int r = 0; r < num_rows; r++) {
        cout << left << setw(2) << (r + 1) << " |";
        for (int c = 0; c < num_cols; c++) {
            if (seatMap[r][c]) {
                cout << " X |";
            } else {
                cout << "   |";
            }
        }
        cout << endl;
        cout << right << setw(4) << "+";
        for (int c = 0; c < num_cols; c++) {
            cout << "---+";
        }
        cout << endl;
    }
}
```

```cpp
void displayPassengerInfo(Flight& flight) {
  vector<Passenger> passengers = flight.get_passengers();
  vector<Seat> seats = flight.get_seats();

  cout << "Passenger List (Flight: " << flight.get_flight_id()
    << " from " << flight.get_route().get_source()
    << " to " << flight.get_route().get_destination() << ")\n" << endl;

  cout << left << setw(15) << "First Name" << setw(15) << "Last Name"
    << setw(18) << "Phone" << setw(8) << "Row" << setw(8) << "Seat"
    << setw(5) << "ID" << endl;
  cout << "---------------------------------------------------------------------" << endl;
  for (int i = 0; i < passengers.size(); i++) {
    Passenger& passenger = passengers[i];
    Seat& seat = seats[i];
    cout << left << setw(15) << passenger.get_first_name()
      << setw(15) << passenger.get_last_name()
      << setw(18) << passenger.get_phone_number()
      << setw(8) << seat.get_row_number()
      << setw(8) << seat.get_seat_character()
      << setw(5) << passenger.get_passenger_id() << endl;
    cout << "---------------------------------------------------------------" << endl;
  }
}

void addNewPassenger(Flight& flight) {
  string passenger_id, first_name, last_name, phone_number;
  int row_number;
  char seat_character;

  // Get valid passenger ID
  while (true) {
    cout << "Please enter the passenger id or press return to quit: ";
    cin >> passenger_id;
    if (passenger_id == ""){break;}
    cleanStandardInputStream();

    // Check if passenger ID already exists
    const vector<Passenger>& passengers = flight.get_passengers();
    bool id_exists = false;
    for (const Passenger& passenger : passengers) {
      if (passenger.get_passenger_id() == passenger_id) {
        id_exists = true;
        break;
      }
    }
```

```cpp
      if (!id_exists) break;
      cout << "\nPassenger ID already exists. Please enter a different ID.\n" << endl;
  }

  cout << "Please enter the passenger first name: ";
  cin >> first_name;
  cout << "Please enter the passenger last name: ";
  cin >> last_name;
  cout << "Please enter the passenger phone number: ";
  cin >> phone_number;

  // Get valid seat
  while (true) {
    // Get valid row number
    int max_rows = flight.get_number_of_rows();
    while (true) {
      cout << "\nPlease enter the row number (1-" << max_rows << "): ";
      cin >> row_number;

      if (!cin.fail() && row_number >= 1 && row_number <= max_rows) {
        cleanStandardInputStream();
        break;
      }

      cin.clear();
      cleanStandardInputStream();
      cout << "\nInvalid row number. Please enter a number between 1 and " << max_rows << "." <<
endl;
    }

    // Get valid seat character
    int max_seats = flight.get_number_of_seats_per_row();
    char max_seat_char = 'A' + max_seats - 1;
    while (true) {
      cout << "Please enter the seat character (A-" << max_seat_char << "): ";
      cin >> seat_character;
      cleanStandardInputStream();

      if (seat_character >= 'A' && seat_character <= max_seat_char) {
        break;
      }

      cout << "\nInvalid seat character. Please enter a letter between A and " << max_seat_char << ".\n"
<< endl;
    }

    // Check if seat is already taken
    const vector<Seat>& seats = flight.get_seats();
```

```cpp
      bool seat_taken = false;
      for (const Seat& seat : seats) {
        if (seat.get_row_number() == row_number && seat.get_seat_character() == seat_character) {
          seat_taken = true;
          break;
        }
      }

      if (seat_taken) {
        cout << "\nSeat " << row_number << seat_character << " is already taken. Please choose a
different seat." << endl;
        continue;  // Go back to start of outer loop to ask for new seat
      }

      // Seat is available, break out of outer loop
      break;
    }

    Passenger passenger = Passenger(passenger_id, first_name, last_name, phone_number);
    Seat seat = Seat(row_number, seat_character, passenger_id);
    flight.addPassenger(passenger);
    flight.addSeat(seat);
    cout << "\nPassenger " << passenger.get_first_name() << " " << passenger.get_last_name() << " added
successfully to flight " << flight.get_flight_id() << "." << endl;
}

void removeExistingPassenger(Flight& flight) {
    string passenger_id;
    cout << "Please enter the id of the passenger that needs to be removed: ";
    cin >> passenger_id;
    cleanStandardInputStream();

    // Check if passenger exists
    const vector<Passenger>& passengers = flight.get_passengers();
    bool passenger_exists = false;
    string passenger_name;
    for (const Passenger& passenger : passengers) {
      if (passenger.get_passenger_id() == passenger_id) {
        passenger_exists = true;
        passenger_name = passenger.get_first_name() + " " + passenger.get_last_name();
        break;
      }
    }

    if (!passenger_exists) {
      cout << "\nPassenger with ID \"" << passenger_id << "\" not found." << endl;
      return;
    }
```

```
    flight.removePassenger(passenger_id);
    cout << "\nPassenger " << passenger_name << " was successfully removed from flight " <<
flight.get_flight_id() << "." << endl;
}
```

ui_utils.cpp

```cpp
// src/utils/ui_utils.cpp
#include "../../include/utils/ui_utils.h"
#include <iostream>
using namespace std;

void cleanStandardInputStream(void) {
    int leftover;
    do {
        leftover = cin.get();
    } while (leftover != '\n' && leftover != EOF);
}

void clearScreen(void) {
    cout << "\033[2J\033[1;1H"; // ANSI escape sequence to clear the screen (cross-platform compatible)
}

void pressEnter() {
    cout << "\n<<< Press Return to Continue >>>" << endl;
    cin.get();
}

void displayHeader() {
    clearScreen();
    cout << "FMAS Version 1.0" << endl;
    cout << "Term Project - Flight Management Application System" << endl;
    cout << "Produced by Group#: 11" << endl;
    cout << "Names: Ryabinkin, Simon; ElSayed, Abdelrahman; Boucher, Maxime" << endl;
    pressEnter();
    cleanStandardInputStream();
}

void printChoicePrompt() {
    cout << "Please select one the following options:\n" << endl;
    cout << "1. Select a flight" << endl;
    cout << "2. Display Flight Seat Map." << endl;
    cout << "3. Display Passengers Information." << endl;
    cout << "4. Add a New Passenger." << endl;
    cout << "5. Remove an Existing Passenger" << endl;
    cout << "6. Save data" << endl;
    cout << "7. Quit." << endl;
    cout << "\nEnter your choice: (1, 2, 3, 4, 5, 6, or 7) ";
}

int menu() {
    int choice = -1;
    clearScreen();
```

```
    while (true) {
        printChoicePrompt();

        if (!(cin >> choice)) {
            cin.clear();
            cout << "\nInvalid choice. Please enter a number between 1 and 7.\n" << endl;
            cleanStandardInputStream();
            continue;
        }
        cleanStandardInputStream();
        break;
    }
    return choice;
}
```

main.cpp

```cpp
// main.cpp
#include <iostream>
#include "include/classes/Airline.h"
#include "include/classes/Flight.h"
#include "include/utils/ui_utils.h"
#include "include/utils/flight_operations.h"
#include "include/utils/data_io.h"
using namespace std;
const string default_airline = "WestJet";

int main() {
  displayHeader();
  int menu_choice;

  // Initialize airline with flights
  Airline* selectedAirline = createAirline(default_airline);

  int selected_flight_index = 0;

  while((menu_choice = menu())) {
    clearScreen();
    Flight& selected_flight = selectedAirline -> get_flight(selected_flight_index);
    switch(menu_choice) {
      case 1:
        selected_flight_index = selectFlight(selectedAirline -> get_flights());
        pressEnter();
        break;
      case 2:
        displaySeatMap(selected_flight);
        pressEnter();
        break;
      case 3:
        displayPassengerInfo(selected_flight);
        pressEnter();
        break;
      case 4:
        addNewPassenger(selected_flight);
        pressEnter();
        break;
      case 5:
        removeExistingPassenger(selected_flight);
        pressEnter();
        break;
      case 6:
        saveData(selectedAirline -> get_flights());
        pressEnter();
        break;
```

```cpp
            case 7:
                cout << "Program terminated." << endl;
                delete selectedAirline;
                return 0;
            default:
                cout << "Invalid choice. Please try again." << endl;
                pressEnter();
                break;
        }
    }
    delete selectedAirline;
    return 0;
}
```