#include <iostream>

#include <fstream>

#include <string>

using namespace std;

class User

{

private:

string username;

string password;

public:

//constructors

User()

{

username[0] = '\0';

password[0] = '\0';

}

User(string newName, string newPassword)

{

username = newName;

password = newPassword;

}

User(const User& newUser)

{

username = newUser.username;

password = newUser.password;

}

//setters

void setUsername(string newName)

{

username = newName;

}

void setPassword(string newPassword)

{

password = newPassword;

}

//getters

string getUsername()

{

return username;

}

string getPassword()

{

return password;

}

bool operator == (const User& newUser)

{

if (username == newUser.username && password == newUser.password)

{

return true;

}

else

return false;

}

bool operator !=(const User& newUser)

{

if (username != newUser.username && password != newUser.password)

{

return true;

}

else

return false;

}

// Print Function

};

class Passenger :public User

{

private:

string passengerName;

string nationality;

long long int passportNo;

int ticketsBooked;

int\* ticketId;

public:

//Constructors

Passenger()

{

passengerName[0] = '\0';

nationality[0] = '\0';

passportNo = 0;

ticketsBooked = 0;

ticketId = 0;

}

Passenger(string newName, string newPassword, string newPassengerName, string newNationality,

long long int newPassportNo, int totaltickets, int\* newticketid) : User(newName, newPassword)

{

passengerName = newPassengerName;

nationality = newNationality;

passportNo = newPassportNo;

ticketsBooked = totaltickets;

ticketId = new int[ticketsBooked];

for (int i = 0; i < ticketsBooked; i++)

{

ticketId[i] = newticketid[i];

}

}

Passenger(const User& newUser, const Passenger& newPassenger) :User(newUser)

{

passengerName = newPassenger.passengerName;

nationality = newPassenger.nationality;

passportNo = newPassenger.passportNo;

ticketsBooked = newPassenger.ticketsBooked;

ticketId = new int[ticketsBooked];

for (int i = 0; i < ticketsBooked; i++)

{

ticketId[i] = newPassenger.ticketId[i];

}

}

//Setters

void SetPassengerName(string newPassengerName)

{

passengerName = newPassengerName;

}

void SetNationality(string newNationality)

{

nationality = newNationality;

}

void SetPassportNo(long long int newPassportNo)

{

passportNo = newPassportNo;

}

void SetTicketsBooked(int totaltickets)

{

ticketsBooked = totaltickets;

}

void SetTicketIds(int\* newticketid)

{

ticketId = new int[ticketsBooked];

for (int i = 0; i < ticketsBooked; i++)

{

ticketId[i] = newticketid[i];

}

}

//Getters

string GetPassengerName()

{

return passengerName;

}

string GetNationality()

{

return nationality;

}

long long int GetPassportNo()

{

return passportNo;

}

int GetTicketsBooked()

{

return ticketsBooked;

}

int\* GetTicketIds()

{

return ticketId;

}

void DeleteTicketIds()

{

delete[] ticketId;

ticketId = nullptr;

}

//operator overloading

friend ostream& operator << (ostream& out, const Passenger& newPasenger);

friend ostream& operator << (ostream& out, const Passenger& newPasenger);

//friend istream& operator >> (istream& in, const Passenger& newPasenger);

};

ostream& operator << (ostream& out, Passenger& newPassenger)

{

cout << newPassenger.getUsername() << ", ";

cout << newPassenger.getPassword() << ", ";

cout << newPassenger.GetPassengerName() << ", ";

cout << newPassenger.GetNationality() << ", ";

cout << newPassenger.GetPassportNo() << ", ";

cout << newPassenger.GetTicketsBooked() << ",";

int\* tickets = newPassenger.GetTicketIds();

for (int j = 0; j < newPassenger.GetTicketsBooked(); j++)

{

cout << " " << tickets[j];

}

cout << ".";

return out;

}

class Admin : public User

{

private:

string adminName;

public:

//constructors

Admin()

{

adminName[0] = '\0';

}

Admin(string newName, string newPassword, string newAdminName) :User(newName, newPassword)

{

adminName = newAdminName;

}

Admin(const User& newUser, const Admin& newAdmin) :User(newUser)

{

adminName = newAdmin.adminName;

}

//setter

void SetAdminName(string newAdminName)

{

adminName = newAdminName;

}

//getter

string GetAdminName()

{

return adminName;

}

};

class Ticket

{

private:

int ticketID;

int passportNumber;

public:

Ticket()

{

ticketID = 0;

passportNumber = 0;

}

Ticket(int newID, int newPass)

{

ticketID = newID;

passportNumber = newPass;

}

void setID(int newID)

{

ticketID = newID;

}

void setPassportNumber(int newPass)

{

passportNumber = newPass;

}

int getID()

{

return ticketID;

}

int getPassportNumber()

{

return passportNumber;

}

};

class Date

{

public:

int day;

int month;

int year;

Date()

{

day = 0;

month = 0;

year = 0;

}

Date(int newDay, int newMonth, int newYear)

{

day = newDay;

month = newMonth;

year = newYear;

}

Date(const Date& copyObj)

{

day = copyObj.day;

month = copyObj.month;

year = copyObj.year;

}

};

class Time

{

public:

int hour;

int minutes;

int seconds;

Time()

{

hour = 0;

minutes = 0;

seconds = 0;

}

Time(int newHours, int newMinutes, int newSeconds)

{

hour = newHours;

minutes = newMinutes;

seconds = newSeconds;

}

Time(const Time& newObj)

{

hour = newObj.hour;

minutes = newObj.minutes;

seconds = newObj.seconds;

}

};

class Airline

{

private:

string airlineName;

int airlineID;

public:

Airline()

{

airlineName = '\0';

airlineID = -1;

}

Airline(string newName, int newID)

{

airlineName = newName;

airlineID = newID;

}

void setAirlineName(string newName)

{

airlineName = newName;

}

void setAirlineID(int newID)

{

airlineID = newID;

}

string getAirlineName()

{

return airlineName;

}

int getAirlineID()

{

return airlineID;

}

};

class InternationalFlight :virtual public Airline

{

private:

string country;

public:

InternationalFlight() : Airline()

{

country = '\0';

}

InternationalFlight(string newName, int newID, string newCountry = "Nothing set") : Airline(newName, newID)

{

country = newCountry;

}

void setCountry(string newCountry)

{

country = newCountry;

}

string getCountry()

{

return country;

}

};

class LocalFlight : virtual public Airline

{

private:

string city;

public:

LocalFlight() : Airline()

{

city = '\0';

}

LocalFlight(string newName, int newID, string newCity = "Nothing set") : Airline(newName, newID)

{

city = newCity;

}

void setCity(string newCity)

{

city = newCity;

}

string getCity()

{

return city;

}

};

class Plane : public InternationalFlight, public LocalFlight

{

private:

Ticket\* ticketArray;

char statusOfFlight; //O for ontime, D for delayed, M for Missed, C for completed

int availableTickets;

int bookedTickets;

int flightNumber;

Time departureTime;

Time arrivalTime;

Date flightDate;

public:

Plane() :InternationalFlight(), LocalFlight()

{

ticketArray = NULL;

statusOfFlight = '\0';

availableTickets = 0;

bookedTickets = 0;

flightNumber = 0;

flightDate.day = 0;

flightDate.month = 0;

flightDate.year = 0;

departureTime.hour = 0;

departureTime.minutes = 0;

departureTime.seconds = 0;

arrivalTime.hour = 0;

arrivalTime.minutes = 0;

arrivalTime.seconds = 0;

}

Plane(Ticket\* newArr, char newStatus, int newTickets, int newBookedTickets, int newFlightNumber, Time newDeptTime, Time newArrTime, Date newDate, string newName,

int newID) : InternationalFlight(newName, newID), LocalFlight(newName, newID)

{

ticketArray = newArr;

statusOfFlight = newStatus;

availableTickets = newTickets;

bookedTickets = newBookedTickets;

flightNumber = newFlightNumber;

flightDate.day = newDate.day;

flightDate.month = newDate.month;

flightDate.year = newDate.year;

departureTime.hour = newDeptTime.hour;

departureTime.minutes = newDeptTime.minutes;

departureTime.seconds = newDeptTime.seconds;

arrivalTime.hour = newArrTime.hour;

arrivalTime.minutes = newArrTime.minutes;

arrivalTime.seconds = newArrTime.seconds;

}

Plane(const Plane& obj)

{

ticketArray = obj.ticketArray;

statusOfFlight = obj.statusOfFlight;

availableTickets = obj.availableTickets;

bookedTickets = obj.bookedTickets;

flightNumber = obj.flightNumber;

flightDate = obj.flightDate;

departureTime = obj.departureTime;

arrivalTime = obj.arrivalTime;

}

// Setters

void setTicketArr(Ticket\* newArr)

{

ticketArray = newArr;

}

void setStatusOfFlight(char newStatus)

{

statusOfFlight = newStatus;

}

void setAvailableTickets(int newAvailableTickets)

{

availableTickets = newAvailableTickets;

}

void setBookedTickets(int newBookedTickets)

{

bookedTickets = newBookedTickets;

}

void setFlightNumber(int newFlight)

{

flightNumber = newFlight;

}

void setDepartureTime(Time newDeptTime)

{

departureTime.hour = newDeptTime.hour;

departureTime.minutes = newDeptTime.minutes;

departureTime.seconds = newDeptTime.seconds;

}

void setArrivalTime(Time newArrivalTime)

{

arrivalTime.hour = newArrivalTime.hour;

arrivalTime.minutes = newArrivalTime.minutes;

arrivalTime.seconds = newArrivalTime.seconds;

}

void setDate(Date newDate)

{

flightDate.day = newDate.day;

flightDate.month = newDate.month;

flightDate.year = newDate.year;

}

// Getters

Ticket\* getTickets()

{

return ticketArray;

}

char getStatusOfFlight()

{

return statusOfFlight;

}

int getAvailableTickets()

{

return availableTickets;

}

int getBookedTickets()

{

return bookedTickets;

}

int getUnbookedTickets()

{

return availableTickets - bookedTickets;

}

int getFlightNumber()

{

return flightNumber;

}

Time getDepartureTime()

{

return departureTime;

}

Time getArrivalTime()

{

return arrivalTime;

}

Date getDate()

{

return flightDate;

}

~Plane()

{

delete[] ticketArray;

}

};

////////////////////////////////////////////Menu Functions Start Here////////////////////////////////

int getChoice()

{

int choice = 0;

cin >> choice;

while (!cin.good())

{

cout << "Invalid Input (Wrong Data Type)\n";

cin.clear();

cin.ignore(std::numeric\_limits<int>::max(), '\n');

cout << "=>";

cin >> choice;

}

return choice;

}

int MainMenu()

{

int choice = 0;

system("CLS");

cout << "Welcome to the Flight Management System!!" << endl;

cout << endl;

cout << "Choose one of the options down below!" << endl;

cout << "1. For Admin" << endl;

cout << "2. For Passenger" << endl;

cout << "0. To exit the program" << endl;

cout << "=> ";

choice = getChoice();

return choice;

}

int SignInMenu()

{

int choice = 0;

system("CLS");

cout << "Choose whatever you would like to do:" << endl;

cout << "1. Log In" << endl;

cout << "2. Sign Up" << endl;

cout << "0. To exit the program" << endl;

cout << "=> ";

choice = getChoice();

return choice;

}

int AdminMenu()

{

int choice = 0;

system("CLS");

cout << "Choose whatever you would like to do:" << endl;

cout << "1. Search through flights" << endl;

cout << "2. Add new flights to the database" << endl;

cout << "3. Remove flights from the database" << endl;

cout << "4. Edit details of a flight" << endl;

cout << "5. Display Passengers (Based on flight number, nationality or passport number)" << endl;

cout << "6. To exit the Admin Menu" << endl;

cout << "=> ";

choice = getChoice();

return choice;

}

int PassengerMenu()

{

int choice = 0;

system("CLS");

cout << "Choose whatever you would like to do:" << endl;

cout << "1. Display available flights" << endl;

cout << "2. Book a flight" << endl;

cout << "3. Edit your details (Passenger details)" << endl;

cout << "4. Search through flights" << endl;

cout << "5. Exit the menu" << endl;

cout << "=> ";

choice = getChoice();

return choice;

}

int EditTicketMenu()

{

int choice = 0;

system("CLS");

cout << "Choose whatever you would like to do:" << endl;

cout << "1. Add Ticket" << endl;

cout << "2. Delete Ticket" << endl;

cout << "=> ";

choice = getChoice();

return choice;

}

int DisplayPassengerMenu()

{

int choice = 0;

system("CLS");

cout << "Choose whatever you would like to do:" << endl;

cout << "1. Display passengers based on flight ID" << endl;

cout << "2. Display passengers based on Nationality" << endl;

cout << "3. Display passenger data based on Passport Number" << endl;

cout << "4. To exit the Menu" << endl;

cout << "=> ";

choice = getChoice();

return choice;

}

/////////////////////////////////////////////////////Menu Functions ENd here/////////////////////////////////////////////////////////

//////////////////////////////////////////////////////Sarmad's Input Functions Start here///////////////////////////////////////////////

void PassengerFileName(string& passengerFile)

{

cout << "Enter the name of file containing passenger records (filename should have its file type at the end e.g passengers.dat): ";

getline(cin, passengerFile, '\n');

}

void PlaneFileName(string& PassengerFile)

{

cout << "Enter the name of file containing plane records (filename should have its file type at the end e.g planes.dat): ";

getline(cin, PassengerFile, '\n');

}

void InputPassengerData(Passenger\*& PassengerData, int& totalPassengers, string PassengerFile)

{

ifstream fin;

totalPassengers = 0;

if (!PassengerData)

{

delete[] PassengerData;

}

char c;

fin.open(PassengerFile);

while (!(fin.eof()))

{

fin >> c;

if (c == '.')

{

totalPassengers++;

}

}

fin.close();

totalPassengers--;

PassengerData = new Passenger[totalPassengers];

string username, password, passengername, nationality;

int TotalTickets;

long long int passportNo;

int\* ticketids;

fin.open(PassengerFile);

for (int c = 0; c < totalPassengers; c++)

{

if (c > 0)

{

fin.ignore();

}

getline(fin, username, ',');

fin.ignore();

getline(fin, password, ',');

fin.ignore();

getline(fin, passengername, ',');

fin.ignore();

getline(fin, nationality, ',');

fin.ignore();

fin >> passportNo;

fin.ignore();

fin >> TotalTickets;

fin.ignore();

ticketids = new int[TotalTickets];

for (int i = 0; i < TotalTickets; i++)

{

fin >> ticketids[i];

}

fin.ignore();

PassengerData[c].setUsername(username);

PassengerData[c].setPassword(password);

PassengerData[c].SetPassengerName(passengername);

PassengerData[c].SetNationality(nationality);

PassengerData[c].SetPassportNo(passportNo);

PassengerData[c].SetTicketsBooked(TotalTickets);

PassengerData[c].SetTicketIds(ticketids);

}

fin.close();

}

void InputPlaneData(Plane\*& PlaneData, int& totalPlanes, string PlaneFile)

{

delete[] PlaneData;

ifstream fin;

totalPlanes = 0;

char c;

fin.open(PlaneFile);

while (!(fin.eof()))

{

fin >> c;

if (c == '.')

{

totalPlanes++;

}

}

fin.close();

totalPlanes--;

PlaneData = new Plane[totalPlanes];

string airlineName, country, city;

char status;

Time departure, arrival;

Date flightDate;

int airlineID = 0, availableTickets = 0, flightNo = 0, bookedTickets = 0;

fin.open(PlaneFile);

for (int i = 0; i < totalPlanes; i++)

{

getline(fin, airlineName, ',');

fin.ignore();

fin >> airlineID;

fin.ignore(2);

getline(fin, city, ',');

fin.ignore();

getline(fin, country, ',');

fin.ignore();

fin >> status;

fin.ignore();

fin >> availableTickets;

fin.ignore();

fin >> bookedTickets;

fin.ignore();

fin >> flightNo;

fin.ignore();

fin >> arrival.hour;

fin.ignore();

fin >> arrival.minutes;

fin.ignore();

fin >> arrival.seconds;

fin.ignore();

fin >> departure.hour;

fin.ignore();

fin >> departure.minutes;

fin.ignore();

fin >> departure.seconds;

fin.ignore();

fin >> flightDate.day;

fin.ignore();

fin >> flightDate.month;

fin.ignore();

fin >> flightDate.year;

fin.ignore();

Ticket\* tickets = new Ticket[availableTickets];

int ticketID = 0, passportNo = 0;

for (int j = 0; j < bookedTickets; j++)

{

fin >> ticketID;

fin >> passportNo;

tickets[j].setID(ticketID);

tickets[j].setPassportNumber(passportNo);

fin.ignore();

}

if (bookedTickets == 0)

{

fin >> ticketID;

fin >> passportNo;

fin.ignore();

}

fin.ignore();

PlaneData[i].setAirlineName(airlineName);

PlaneData[i].setAirlineID(airlineID);

PlaneData[i].setCountry(country);

PlaneData[i].setCity(city);

PlaneData[i].setDate(flightDate);

PlaneData[i].setArrivalTime(arrival);

PlaneData[i].setDepartureTime(departure);

PlaneData[i].setAvailableTickets(availableTickets);

PlaneData[i].setBookedTickets(bookedTickets);

PlaneData[i].setFlightNumber(flightNo);

PlaneData[i].setStatusOfFlight(status);

PlaneData[i].setTicketArr(tickets);

}

fin.close();

}

void OutputPlaneData(Plane\* PlaneData, const int totalPlanes, const string PlaneFile)

{

ofstream fout;

fout.open(PlaneFile);

for (int i = 0; i < totalPlanes; i++)

{

Time arrival = PlaneData[i].getArrivalTime();

Time departure = PlaneData[i].getDepartureTime();

Date dateflight = PlaneData[i].getDate();

Ticket\* tickets = PlaneData[i].getTickets();

fout << PlaneData[i].getAirlineName() << ", ";

fout << PlaneData[i].getAirlineID() << ", ";

fout << PlaneData[i].getCity() << ", ";

fout << PlaneData[i].getCountry() << ", ";

fout << PlaneData[i].getStatusOfFlight() << ", ";

fout << PlaneData[i].getAvailableTickets() << ", ";

fout << PlaneData[i].getBookedTickets() << ", ";

fout << PlaneData[i].getFlightNumber() << ", ";

fout << arrival.hour << ":" << arrival.minutes << ":" << arrival.seconds << ", ";

fout << departure.hour << ":" << departure.minutes << ":" << departure.seconds << ", ";

fout << dateflight.day << "/" << dateflight.month << "/" << dateflight.year << ", ";

for (int j = 0; j < (PlaneData[i].getBookedTickets()); j++)

{

fout << tickets[j].getID() << " " << tickets[j].getPassportNumber();

if (j == (PlaneData[i].getBookedTickets()) - 1)

{

fout << ".";

}

else

{

fout << ", ";

}

}

if (PlaneData[i].getBookedTickets() == 0)

{

fout << "0 0.";

}

fout << endl;

}

fout.close();

}

void AddFlights(Plane\*& PlaneData, int& totalPlanes, const string PlaneFile)

{

totalPlanes++;

Plane\* temp = new Plane[totalPlanes];

Ticket\* ticket = nullptr;

for (int i = 0; i < totalPlanes - 1; i++)

{

temp[i].setTicketArr(PlaneData[i].getTickets());

PlaneData[i].setTicketArr(ticket);

temp[i].setAirlineName(PlaneData[i].getAirlineName());

temp[i].setCountry(PlaneData[i].getCountry());

temp[i].setCity(PlaneData[i].getCity());

temp[i].setStatusOfFlight(PlaneData[i].getStatusOfFlight());

temp[i].setAirlineID(PlaneData[i].getAirlineID());

temp[i].setArrivalTime(PlaneData[i].getArrivalTime());

temp[i].setAvailableTickets(PlaneData[i].getAvailableTickets());

temp[i].setDepartureTime(PlaneData[i].getDepartureTime());

temp[i].setBookedTickets(PlaneData[i].getBookedTickets());

temp[i].setDate(PlaneData[i].getDate());

temp[i].setFlightNumber(PlaneData[i].getFlightNumber());

}

string airlineName, country, city;

char status;

Time departure, arrival;

Date flightDate;

int airlineID = 0, availableTickets = 0, flightID = 0, bookedTickets = 0;

cout << "Input the information for the new flight: ";

cout << "\nName of Airline: ";

getline(cin, airlineName, '\n');

cout << "Enter the ID of the airline: ";

cin >> airlineID;

cin.ignore();

cout << "Country: ";

getline(cin, country, '\n');

cout << "City: ";

getline(cin, city, '\n');

cout << "Status Of Flight: ";

cin >> status;

status = toupper(status);

while (status != 'C' && status != 'O' && status != 'M' && status != 'D')

{

cout << "Invalid status. Please enter again: ";

cin >> status;

status = toupper(status);

}

cout << "Enter the available number tickets of the flight: ";

cin >> availableTickets;

while (availableTickets < 0)

{

cout << "Invalid Airline ID. Please enter again: ";

cin >> availableTickets;

}

cout << "Enter the flight ID: ";

cin >> flightID;

while (flightID < 0)

{

cout << "Invalid Airline ID. Please enter again: ";

cin >> flightID;

}

for (int counter = 0; counter < totalPlanes - 1; counter++)

{

if (flightID == PlaneData[counter].getFlightNumber())

{

cout << "Same Airline ID. Please enter again: ";

cin >> flightID;

counter--;

}

}

cout << "Enter the Date\n";

cout << "Enter Day: ";

cin >> flightDate.day;

while (flightDate.day <= 0 || flightDate.day > 31)

{

cout << "Invalid day entered. Please enter again: ";

cin >> flightDate.day;

}

cout << "Enter Month: ";

cin >> flightDate.month;

while (flightDate.month <= 0 || flightDate.month > 12)

{

cout << "Invalid day entered. Please enter again: ";

cin >> flightDate.month;

}

cout << "Enter year: ";

cin >> flightDate.year;

while (flightDate.year < 2022)

{

cout << "Invalid day entered. Please enter again: ";

cin >> flightDate.year;

}

cout << "Enter the Arrival Time\n";

cout << "Enter hours: ";

cin >> arrival.hour;

while (arrival.hour > 24 || arrival.hour < 0)

{

cout << "Invalid hours. Try again: ";

cin >> arrival.hour;

}

cout << "Enter minutes: ";

cin >> arrival.minutes;

while (arrival.minutes > 60 || arrival.minutes < 0)

{

cout << "Invalid minutes. Try again: ";

cin >> arrival.minutes;

}

cout << "Enter seconds: ";

cin >> arrival.seconds;

while (arrival.seconds > 60 || arrival.hour < 0)

{

cout << "Invalid seconds. Try again: ";

cin >> arrival.seconds;

}

cout << "Enter the Departure Time\n";

cout << "Enter hours: ";

cin >> departure.hour;

while (departure.hour > 24 || departure.hour < 0)

{

cout << "Invalid hours. Try again: ";

cin >> departure.hour;

}

cout << "Enter minutes: ";

cin >> departure.minutes;

while (departure.minutes > 60 || departure.minutes < 0)

{

cout << "Invalid minutes. Try again: ";

cin >> departure.minutes;

}

cout << "Enter seconds: ";

cin >> departure.seconds;

while (departure.seconds > 60 || departure.hour < 0)

{

cout << "Invalid seconds. Try again: ";

cin >> departure.seconds;

}

Ticket\* newArr = new Ticket[availableTickets];

temp[totalPlanes - 1].setAirlineID(airlineID);

temp[totalPlanes - 1].setAirlineName(airlineName);

temp[totalPlanes - 1].setCity(city);

temp[totalPlanes - 1].setCountry(country);

temp[totalPlanes - 1].setStatusOfFlight(status);

temp[totalPlanes - 1].setAvailableTickets(availableTickets);

temp[totalPlanes - 1].setBookedTickets(bookedTickets);

temp[totalPlanes - 1].setFlightNumber(flightID);

temp[totalPlanes - 1].setDate(flightDate);

temp[totalPlanes - 1].setArrivalTime(arrival);

temp[totalPlanes - 1].setDepartureTime(departure);

temp[totalPlanes - 1].setTicketArr(newArr);

PlaneData = temp;

temp = nullptr;

OutputPlaneData(PlaneData, totalPlanes, PlaneFile);

cout << "Flight has been added." << endl << endl;

system("pause");

}

void InsertPassenger(Passenger\* passenger, int& totalPassengers, string PassengerFile)

{

ofstream fout;

fout.open(PassengerFile);

for (int i = 0; i < totalPassengers; i++)

{

fout << passenger[i].getUsername() << ", ";

fout << passenger[i].getPassword() << ", ";

fout << passenger[i].GetPassengerName() << ", ";

fout << passenger[i].GetNationality() << ", ";

fout << passenger[i].GetPassportNo() << ", ";

fout << passenger[i].GetTicketsBooked() << ",";

int\* tickets = passenger[i].GetTicketIds();

for (int j = 0; j < passenger[i].GetTicketsBooked(); j++)

{

fout << " " << tickets[j];

}

fout << ".\n";

}

fout.close();

}

void RemoveFlight(Plane\*& PlaneData, int& totalPlanes, Passenger\*& PassengerData, int totalPassengers, string PlaneFile, string PassengerFile)

{

int ID = 0, IDindex = -1;

bool IDfound = false;

cout << "Enter the Flight ID of the flight you want to remove: ";

cin >> ID;

for (int i = 0; i < totalPlanes; i++)

{

if (PlaneData[i].getFlightNumber() == ID)

{

IDfound = true;

IDindex = i;

break;

}

}

if (!IDfound)

{

cout << "\nNo flight with such ID exists!\n\n";

system("pause");

return;

}

totalPlanes--;

Plane\* temp = new Plane[totalPlanes];

Ticket\* ticket = nullptr;

Ticket\* RemoveTickets = PlaneData[IDindex].getTickets();

for (int j = 0; j < PlaneData[IDindex].getBookedTickets(); j++)

{

for (int i = 0; i < totalPassengers; i++)

{

for (int k = 0; k < PassengerData[i].GetTicketsBooked(); k++)

{

bool TicketFound = false;

int\* bookedTickets = PassengerData[i].GetTicketIds();

if (RemoveTickets[j].getID() == bookedTickets[k])

{

TicketFound = true;

}

if (TicketFound)

{

PassengerData[i].SetTicketsBooked(PassengerData[i].GetTicketsBooked() - 1);

int\* temp = new int[PassengerData[i].GetTicketsBooked()];

for (int l = 0, m = 0; l < PassengerData[i].GetTicketsBooked() + 1 && m < PassengerData[i].GetTicketsBooked(); l++)

{

if (bookedTickets[l] == RemoveTickets[j].getID())

{

continue;

}

temp[m] = bookedTickets[l];

m++;

}

PassengerData[i].SetTicketIds(temp);

temp = nullptr;

k = 0;

}

}

}

}

InsertPassenger(PassengerData, totalPassengers, PassengerFile);

for (int i = 0, j = 0; i < totalPlanes + 1; i++)

{

if (PlaneData[i].getFlightNumber() == ID)

{

continue;

}

temp[j].setTicketArr(PlaneData[i].getTickets());

PlaneData[j].setTicketArr(ticket);

temp[j].setAirlineName(PlaneData[i].getAirlineName());

temp[j].setCountry(PlaneData[i].getCountry());

temp[j].setCity(PlaneData[i].getCity());

temp[j].setStatusOfFlight(PlaneData[i].getStatusOfFlight());

temp[j].setAirlineID(PlaneData[i].getAirlineID());

temp[j].setArrivalTime(PlaneData[i].getArrivalTime());

temp[j].setAvailableTickets(PlaneData[i].getAvailableTickets());

temp[j].setDepartureTime(PlaneData[i].getDepartureTime());

temp[j].setBookedTickets(PlaneData[i].getBookedTickets());

temp[j].setDate(PlaneData[i].getDate());

temp[j].setFlightNumber(PlaneData[i].getFlightNumber());

j++;

}

delete[]PlaneData;

PlaneData = temp;

temp = nullptr;

OutputPlaneData(PlaneData, totalPlanes, PlaneFile);

cout << "Flight has been removed." << endl << endl;

system("pause");

}

///////////////////////////////////////////////////////////SArmad's Functions end here///////////////////////////////////////////////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////Haseeb's functions Start Here///////////////////////////////////////////////////////////////////////////////////////////

int returnFlight(Plane\* obj, const int totalPlanes, const int searchNum) // If flight number same, returns the object with same flight number

{

for (int counter = 0; counter < totalPlanes; counter++)

{

if (obj[counter].getFlightNumber() == searchNum)

{

cout << "Flight Found!" << endl;

return counter;

}

}

return -1;

}

void FlightDisplay(Plane\* obj, const int position) // Displays specific flight sent as parameter

{

Time tempTime = obj[position].getDepartureTime();

Date tempDate = obj[position].getDate();

cout << "Available number of booked Tickets: " << obj[position].getBookedTickets() << endl;

cout << "Available number of unbooked Tickets: " << obj[position].getUnbookedTickets() << endl;

cout << "Flight Number: " << obj[position].getFlightNumber() << endl;

cout << "Flight Status: " << obj[position].getStatusOfFlight() << endl;

cout << "Flight City: " << obj[position].getCity() << endl;

cout << "Flight Country: " << obj[position].getCountry() << endl;

cout << "Flight Date: " << tempDate.day << "/" << tempDate.month << "/" << tempDate.year << endl;

cout << "Departure Time: " << tempTime.hour << ":" << tempTime.minutes << ":" << tempTime.seconds << endl;

tempTime = obj[position].getArrivalTime();

cout << "Arrival Time: " << tempTime.hour << ":" << tempTime.minutes << ":" << tempTime.seconds << endl << endl;

}

void InputFlightDetails(Plane\*& obj, int totalPlanes, const int position)

{

string airlineName, city, country;

int airlineID, flightID, ticketID, passportNum;

char status;

Time arrivalTime, departureTime;

Date newDate;

cin.ignore();

cout << "Enter the Airline Name: ";

getline(cin, airlineName);

cout << "Enter the Airline ID: ";

cin >> airlineID;

while (airlineID < 0)

{

cout << "Invalid Airline ID. Please enter again: ";

cin >> airlineID;

}

for (int counter = 0; counter < totalPlanes; counter++)

{

if (airlineID == obj[counter].getAirlineID())

{

cout << "Same Airline ID. Please enter again: ";

cin >> airlineID;

counter--;

}

}

cin.ignore();

cout << "Enter the city of the flight: ";

getline(cin, city);

cin.ignore();

cout << "Enter the country of the flight: ";

getline(cin, country);

cout << "Enter the status of the flight: ";

cin >> status;

status = toupper(status);

while (status != 'C' && status != 'O' && status != 'M' && status != 'D')

{

cout << "Invalid status. Please enter again: ";

cin >> status;

status = toupper(status);

}

cout << "Enter the flight ID: ";

cin >> flightID;

while (flightID < 0)

{

cout << "Invalid Airline ID. Please enter again: ";

cin >> flightID;

}

for (int counter = 0; counter < totalPlanes; counter++)

{

if (flightID == obj[counter].getFlightNumber())

{

cout << "Same Airline ID. Please enter again: ";

cin >> flightID;

counter--;

}

}

cout << "Enter the Date\n";

cout << "Enter Day: ";

cin >> newDate.day;

while (newDate.day <= 0 || newDate.day > 31)

{

cout << "Invalid day entered. Please enter again: ";

cin >> newDate.day;

}

cout << "Enter Month: ";

cin >> newDate.month;

while (newDate.month <= 0 || newDate.month > 12)

{

cout << "Invalid day entered. Please enter again: ";

cin >> newDate.month;

}

cout << "Enter year: ";

cin >> newDate.year;

while (newDate.year < 2022)

{

cout << "Invalid day entered. Please enter again: ";

cin >> newDate.year;

}

cout << "Enter the Arrival Time\n";

cout << "Enter hours: ";

cin >> arrivalTime.hour;

while (arrivalTime.hour > 24 || arrivalTime.hour < 0)

{

cout << "Invalid hours. Try again: ";

cin >> arrivalTime.hour;

}

cout << "Enter minutes: ";

cin >> arrivalTime.minutes;

while (arrivalTime.minutes > 60 || arrivalTime.minutes < 0)

{

cout << "Invalid minutes. Try again: ";

cin >> arrivalTime.minutes;

}

cout << "Enter seconds: ";

cin >> arrivalTime.seconds;

while (arrivalTime.seconds > 60 || arrivalTime.hour < 0)

{

cout << "Invalid seconds. Try again: ";

cin >> arrivalTime.seconds;

}

cout << "Enter the Departure Time\n";

cout << "Enter hours: ";

cin >> departureTime.hour;

while (departureTime.hour > 24 || departureTime.hour < 0)

{

cout << "Invalid hours. Try again: ";

cin >> departureTime.hour;

}

cout << "Enter minutes: ";

cin >> departureTime.minutes;

while (departureTime.minutes > 60 || departureTime.minutes < 0)

{

cout << "Invalid minutes. Try again: ";

cin >> departureTime.minutes;

}

cout << "Enter seconds: ";

cin >> departureTime.seconds;

while (departureTime.seconds > 60 || departureTime.hour < 0)

{

cout << "Invalid seconds. Try again: ";

cin >> departureTime.seconds;

}

obj[position].setAirlineID(airlineID);

obj[position].setAirlineName(airlineName);

obj[position].setCity(city);

obj[position].setCountry(country);

obj[position].setStatusOfFlight(status);

obj[position].setFlightNumber(flightID);

obj[position].setDate(newDate);

obj[position].setArrivalTime(arrivalTime);

obj[position].setDepartureTime(departureTime);

}

int CompletedFlights(Plane\* obj, const int totalPlanes)

{

int num = 0;

for (int counter = 0; counter < totalPlanes; counter++)

{

if (toupper(obj[counter].getStatusOfFlight()) == 'C')

{

FlightDisplay(obj, counter);

num = 1;

}

}

return num;

}

void DisplayFlights(Plane\* obj, const int totalPlanes)

{

for (int counter = 0; counter < totalPlanes; counter++)

{

Time tempTime = obj[counter].getDepartureTime();

Date tempDate = obj[counter].getDate();

cout << "Available number of booked Tickets: " << obj[counter].getBookedTickets() << endl;

cout << "Available number of unbooked Tickets: " << obj[counter].getUnbookedTickets() << endl;

cout << "Flight Status: " << obj[counter].getStatusOfFlight() << endl;

cout << "Flight Number: " << obj[counter].getFlightNumber() << endl;

cout << "Flight Date: " << tempDate.day << "/" << tempDate.month << "/" << tempDate.year << endl;

cout << "Departure Time: " << tempTime.hour << ":" << tempTime.minutes << ":" << tempTime.seconds << endl;

tempTime = obj[counter].getArrivalTime();

cout << "Arrival Time: " << tempTime.hour << ":" << tempTime.minutes << ":" << tempTime.seconds << endl << endl;

}

}

void DisplayAvailableFlights(Plane\* obj, const int totalPlanes)

{

system("CLS");

bool Found = false;

for (int counter = 0; counter < totalPlanes; counter++)

{

if (toupper(obj[counter].getStatusOfFlight()) != 'C' && obj[counter].getBookedTickets() < obj[counter].getAvailableTickets())

{

Time tempTime = obj[counter].getDepartureTime();

Date tempDate = obj[counter].getDate();

cout << "Available number of booked Tickets: " << obj[counter].getBookedTickets() << endl;

cout << "Available number of unbooked Tickets: " << obj[counter].getUnbookedTickets() << endl;

cout << "Flight Status: " << obj[counter].getStatusOfFlight() << endl;

cout << "Flight Number: " << obj[counter].getFlightNumber() << endl;

cout << "Flight Date: " << tempDate.day << "/" << tempDate.month << "/" << tempDate.year << endl;

cout << "Departure Time: " << tempTime.hour << ":" << tempTime.minutes << ":" << tempTime.seconds << endl;

tempTime = obj[counter].getArrivalTime();

cout << "Arrival Time: " << tempTime.hour << ":" << tempTime.minutes << ":" << tempTime.seconds << endl << endl;

Found = true;

}

}

if (!Found)

{

cout << "No available flights!" << endl;

}

system("pause");

}

void SearchFlights(Plane\* obj, const int totalPlanes)

{

system("CLS");

Plane tempObj;

int option = 0;

int tempChoice = 0;

int flightNum = 0;

bool Found = false;

string searchTemp;

cout << "What would you like to search by? " << endl;

cout << "1. By Flight Number" << endl;

cout << "2. By Location" << endl;

cout << "3. By Time" << endl;

cout << "4. By Airline" << endl;

cout << "5. By completed flights" << endl;

cout << "0. Exit Search Flights Menu" << endl;

cout << "=> ";

option = getChoice();

cout << endl;

system("CLS");

if (option == 1)

{

int foundorNot = -1;

cout << "Enter the Flight Number: ";

cin >> flightNum;

while (flightNum < 0)

{

cout << "Invalid Flight Number has been entered. Please try again: ";

cin >> flightNum;

}

foundorNot = returnFlight(obj, totalPlanes, flightNum);

if (foundorNot != -1) {

FlightDisplay(obj, foundorNot);

Found = true;

}

}

else if (option == 2)

{

cout << "Enter choice" << endl;

cout << "1. Search by City" << endl;

cout << "2. Search by Country\n=> ";

tempChoice = getChoice();

while (tempChoice <= 0 || tempChoice > 2)

{

cout << "Invalid choice. Please input again\n=> ";

tempChoice = getChoice();

}

system("CLS");

cin.ignore();

cout << "Enter search value: ";

getline(cin, searchTemp);

if (tempChoice == 1)

{

for (int counter = 0; counter < totalPlanes; counter++)

{

if (searchTemp.compare(obj[counter].getCity()) == 0)

FlightDisplay(obj, counter);

}

}

else

{

for (int counter = 0; counter < totalPlanes; counter++)

{

if (searchTemp.compare(obj[counter].getCountry()) == 0)

{

FlightDisplay(obj, counter);

Found = true;

}

}

}

}

else if (option == 3)

{

Time objTime;

Time objTime2;

cout << "Enter choice" << endl;

cout << "1. Search by Arrival Time" << endl;

cout << "2. Search by Departure Time\n=> ";

tempChoice = getChoice();

while (tempChoice <= 0 || tempChoice > 2)

{

cout << "Invalid choice. Please input again\n=> ";

tempChoice = getChoice();

}

system("CLS");

cout << "Enter hours: ";

cin >> objTime.hour;

while (objTime.hour > 24 || objTime.hour < 0)

{

cout << "Invalid hours. Try again: ";

cin >> objTime.hour;

}

cout << "Enter minutes: ";

cin >> objTime.minutes;

while (objTime.minutes > 60 || objTime.minutes < 0)

{

cout << "Invalid minutes. Try again: ";

cin >> objTime.minutes;

}

cout << "Enter seconds: ";

cin >> objTime.seconds;

while (objTime.seconds > 60 || objTime.hour < 0)

{

cout << "Invalid seconds. Try again: ";

cin >> objTime.seconds;

}

if (tempChoice == 1)

{

for (int counter = 0; counter < totalPlanes; counter++)

{

objTime2 = obj[counter].getArrivalTime();

if (objTime2.hour == objTime.hour && objTime2.minutes == objTime.minutes && objTime2.seconds == objTime.seconds)

{

FlightDisplay(obj, counter);

Found = true;

}

}

}

else

{

for (int counter = 0; counter < totalPlanes; counter++)

{

objTime2 = obj[counter].getDepartureTime();

if (objTime2.hour == objTime.hour && objTime2.minutes == objTime.minutes && objTime2.seconds == objTime.seconds)

{

FlightDisplay(obj, counter);

Found = true;

}

}

}

}

else if (option == 4)

{

cout << "Enter choice: " << endl;

cout << "1. Search by Airline Name" << endl;

cout << "2. Search by Airline ID\n=> ";

tempChoice = getChoice();

while (tempChoice < 0 || tempChoice>2)

{

cout << "Invalid Choice. Please try again\n=> ";

tempChoice = getChoice();

}

if (tempChoice == 1)

{

system("CLS");

string airlineName;

cin.ignore();

cout << "Enter the Airline Name: ";

getline(cin, airlineName);

for (int counter = 0; counter < totalPlanes; counter++)

{

if (airlineName.compare(obj[counter].getAirlineName()) == 0)

{

FlightDisplay(obj, counter);

Found = true;

}

}

}

else

{

system("CLS");

int searchID;

cout << "Enter the Airline ID: ";

cin >> searchID;

while (searchID < 0)

{

cout << "Invalid Input. Please try again: ";

cin >> searchID;

}

for (int counter = 0; counter < totalPlanes; counter++)

{

if (searchID == obj[counter].getAirlineID())

{

FlightDisplay(obj, counter);

Found = true;

}

}

}

}

else if (option == 5)

{

Found = CompletedFlights(obj, totalPlanes);

}

if (!Found)

{

cout << "No flight was found!" << endl;

}

cout << endl;

system("pause");

}

void EditFlightDetails(Plane\*& obj, int totalPlanes, const string fileName)

{

system("CLS");

int flightID;

bool Found = false;

cout << "First search for the specific flight\n";

cout << "Enter the flight ID: ";

cin >> flightID;

while (flightID < 0)

{

cout << "Invalid flight ID. Please enter again: ";

cin >> flightID;

}

for (int counter = 0; counter < totalPlanes && !Found; counter++)

{

if (obj[counter].getFlightNumber() == flightID)

{

Found = true;

InputFlightDetails(obj, totalPlanes, counter);

OutputPlaneData(obj, totalPlanes, fileName);

InputPlaneData(obj, totalPlanes, fileName);

FlightDisplay(obj, counter);

}

}

if (!Found)

cout << "The Flight was not found!" << endl;

else

cout << "The Flight was found and edited!" << endl;

cout << endl;

system("pause");

}

Ticket\* SearchTicketIDs(Ticket\* arrTicket, const int availableTickets, const int passportNum)

{

Ticket\* tempTicket = new Ticket[availableTickets];

bool atleastOne = false;

int count = 0;

for (int counter = 0; counter < availableTickets; counter++)

{

if (arrTicket[counter].getPassportNumber() == passportNum)

{

tempTicket[count].setID(arrTicket[counter].getID());

atleastOne = true;

count++;

}

}

return tempTicket;

}

//////////////////////////////////////////Haseeb's Functions end here///////////////////////////////////////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////Abdul Raheem's Functions's start here////////////////////////////////////////////////////////////////////////////////////////////////////////

bool CredentialsExist(Passenger\* passenger, int& totalPassengers, User tempUser, int& idx)

{

bool credentialsExist = false;

for (int i = 0; i < totalPassengers; i++)

{

if (tempUser == passenger[i])

{

credentialsExist = true;

idx = i;

break;

}

}

return credentialsExist;

}

bool CredentialsExist(Passenger\* passenger, int& totalPassengers, string username)

{

bool credentialsExist = false;

for (int i = 0; i < totalPassengers; i++)

{

if (username == passenger[i].getUsername())

{

credentialsExist = true;

break;

}

}

return credentialsExist;

}

bool CredentialsExist(Passenger\* passenger, int& totalPassengers, long long int passportNo)

{

bool credentialsExist = false;

for (int i = 0; i < totalPassengers; i++)

{

if (passportNo == passenger[i].GetPassportNo())

{

credentialsExist = true;

break;

}

}

return credentialsExist;

}

bool LoginPortal(Admin admin)

{

User tempUser;

string tempUsername;

string tempPassword;

char choice;

bool isAdmin = false;

do

{

system("CLS");

cout << "Welcome to Admin Login Portal \n\n\n";

cout << "Username: ";

cin >> tempUsername;

tempUser.setUsername(tempUsername);

cout << "\nPassword: ";

cin >> tempPassword;

tempUser.setPassword(tempPassword);

if (tempUser == admin)

{

isAdmin = true;

cout << admin.GetAdminName() << " Welcome to Flight Management System \n";

system("pause");

break;

}

else

{

cout << "\nInvalid Username or Password \n\n";

cout << "Press y to retry \t Press any key to exit\n";

cin >> choice;

if (choice != 'y')

{

break;

}

else

{

continue;

}

}

} while (tempUser != admin);

return isAdmin;

}

bool LoginPortal(Passenger\* passenger, int totalPassengers, int& index)

{

Passenger tempUser;

string tempUsername;

string tempPassword;

bool isPassenger = false;

do

{

system("CLS");

cout << "Welcome to Passenger Log In Portal \n\n\n";

cout << "Username: ";

cin >> tempUsername;

cout << "\nPassword: ";

cin >> tempPassword;

tempUser.setUsername(tempUsername);

tempUser.setPassword(tempPassword);

if (CredentialsExist(passenger, totalPassengers, tempUser, index))

break;

else if (!CredentialsExist(passenger, totalPassengers, tempUser, index))

{

cout << "The username or password is incorrect\n";

cout << "Kindly re enter \n";

system("pause");

continue;

}

} while (!CredentialsExist(passenger, totalPassengers, tempUser, index));

if (CredentialsExist(passenger, totalPassengers, tempUser, index))

{

isPassenger = true;

tempUser.SetPassengerName(passenger[index].GetPassengerName());

system("CLS");

cout << tempUser.GetPassengerName() << " Welcome to Flight Management System \n";

system("pause");

}

return isPassenger;

}

void InsertPassenger(Passenger\* passenger, int& totalPassengers, string PassengerFile, Passenger newPassenger)

{

ofstream fout;

fout.open(PassengerFile);

for (int i = 0; i < totalPassengers; i++)

{

fout << passenger[i].getUsername() << ", ";

fout << passenger[i].getPassword() << ", ";

fout << passenger[i].GetPassengerName() << ", ";

fout << passenger[i].GetNationality() << ", ";

fout << passenger[i].GetPassportNo() << ", ";

fout << passenger[i].GetTicketsBooked() << ",";

int\* tickets = passenger[i].GetTicketIds();

for (int j = 0; j < passenger[i].GetTicketsBooked(); j++)

{

fout << " " << tickets[j];

}

fout << ".\n";

}

fout << newPassenger.getUsername() << ", ";

fout << newPassenger.getPassword() << ", ";

fout << newPassenger.GetPassengerName() << ", ";

fout << newPassenger.GetNationality() << ", ";

fout << newPassenger.GetPassportNo() << ", ";

fout << newPassenger.GetTicketsBooked() << ",";

int\* tickets = newPassenger.GetTicketIds();

for (int j = 0; j < newPassenger.GetTicketsBooked(); j++)

{

fout << " " << tickets[j];

}

fout << ".\n";

fout.close();

}

Passenger SignUp(Passenger\* passenger, int& totalPassengers, string PassengerFile)

{

Passenger newPassenger;

string tempUsername;

string tempPassword;

string tempPassengerName;

string tempNationality;

long long int tempPassportNo;

char choice;

cin.ignore();

do

{

system("CLS");

cout << "Welcome to Passenger Sign Up Portal" << endl << endl << endl;

cout << "Username: ";

getline(cin, tempUsername, '\n');

if (CredentialsExist(passenger, totalPassengers, tempUsername))

{

cout << "The username is already taken \n";

cout << "Kindly enter another username \n";

system("pause");

}

else

break;

} while (CredentialsExist(passenger, totalPassengers, tempUsername));

if (!CredentialsExist(passenger, totalPassengers, tempUsername))

{

newPassenger.setUsername(tempUsername);

}

cout << "\nPassword: ";

getline(cin, tempPassword, '\n');

newPassenger.setPassword(tempPassword);

cout << "\nPassenger Name: ";

getline(cin, tempPassengerName, '\n');

newPassenger.SetPassengerName(tempPassengerName);

cout << "\nNationality: ";

getline(cin, tempNationality, '\n');

newPassenger.SetNationality(tempNationality);

do

{

cout << "\nPassport No: ";

cin >> tempPassportNo;

if (CredentialsExist(passenger, totalPassengers, tempPassportNo))

{

cout << "\nPassport No already exists \n";

cout << "Check your Passport No and re enter \n";

system("pause");

}

} while (CredentialsExist(passenger, totalPassengers, tempPassportNo));

if (!CredentialsExist(passenger, totalPassengers, tempPassportNo))

{

newPassenger.SetPassportNo(tempPassportNo);

}

return newPassenger;

}

void EditUsername(Passenger\* passenger, int& totalPassengers, int index)

{

string tempUsername;

cin.ignore();

do {

cout << "Enter New Username\n";

cout << "=>";

getline(cin, tempUsername, '\n');

if (!CredentialsExist(passenger, totalPassengers, tempUsername))

{

passenger[index].setUsername(tempUsername);

break;

}

else if (CredentialsExist(passenger, totalPassengers, tempUsername))

{

cout << "The username is unavailable\n";

continue;

}

} while (CredentialsExist(passenger, totalPassengers, tempUsername));

cout << "New Information:\n" << passenger[index] << endl;

}

void EditPassword(Passenger\* passenger, int& totalPassengers, int index)

{

string tempPassword;

cin.ignore();

cout << "Enter New Password\n";

cout << "=>";

getline(cin, tempPassword, '\n');

passenger[index].setPassword(tempPassword);

cout << "New Information:\n" << passenger[index] << endl;

}

void EditPassengerName(Passenger\* passenger, int& totalPassengers, int index)

{

string tempPassengerName;

cin.ignore();

cout << "Enter New PassengerName\n";

cout << "=>";

getline(cin, tempPassengerName, '\n');

passenger[index].SetPassengerName(tempPassengerName);

cout << "New Information:\n" << passenger[index] << endl;

}

void EditNationality(Passenger\* passenger, int& totalPassengers, int index)

{

string tempNationality;

cin.ignore();

cout << "Enter New Nationality\n";

cout << "=>";

getline(cin, tempNationality, '\n');

passenger[index].SetNationality(tempNationality);

cout << "New Information:\n" << passenger[index] << endl;

}

void EditPassengerDetails(Passenger\* passenger, int& totalPassengers, int index)

{

int choice = 0;

cout << "User information:\n";

cout << passenger[index];

cout << endl;

system("pause");

system("CLS");

cout << "What do you want to edit?\n";

cout << "1. Username" << endl;

cout << "2. Password" << endl;

cout << "3. Passenger Name" << endl;

cout << "4. Nationality" << endl;

cout << "=> ";

choice = getChoice();

if (choice == 1)

{

EditUsername(passenger, totalPassengers, index);

}

else if (choice == 2)

{

EditPassword(passenger, totalPassengers, index);

}

else if (choice == 3)

{

EditPassengerName(passenger, totalPassengers, index);

}

else if (choice == 4)

{

EditNationality(passenger, totalPassengers, index);

}

system("pause");

}

bool TicketAvailable(Plane\*& obj, int FlightNum, int ticketID)

{

bool ticketAvailable = true;

Ticket\* tempTickets = obj[FlightNum].getTickets();

for (int i = 0; i < obj[FlightNum].getBookedTickets(); i++)

{

if (ticketID == tempTickets[i].getID())

{

ticketAvailable = false;

break;

}

}

return ticketAvailable;

}

int returnFlightIndex(Plane\* obj, const int totalPlanes, int tempFlightID) // If flight number same, returns the object with same flight number

{

int index = -1;

for (int counter = 0; counter < totalPlanes; counter++)

{

if (obj[counter].getFlightNumber() == tempFlightID)

{

cout << "Flight Found!" << endl;

index = counter;

break;

}

else

{

index = -1;

}

}

return index;

}

void InputFlightDetails(Plane\*& obj, const int FlightNum, long long int passportno)

{

int avaiableTickets, bookedTickets, ticketID, passportNum;

Ticket tempTicket;

bookedTickets = obj[FlightNum].getBookedTickets();

bookedTickets++;

obj[FlightNum].setBookedTickets(bookedTickets);

cout << "Enter Ticket ID\n";

ticketID = getChoice();

while (!TicketAvailable(obj, FlightNum, ticketID))

{

cout << "Ticket ID already exists\n";

cout << "Enter Ticket ID\n";

ticketID = getChoice();

}

if (TicketAvailable(obj, FlightNum, ticketID))

{

tempTicket.setID(ticketID);

}

tempTicket.setPassportNumber(passportno);

Ticket\* newArr = obj[FlightNum].getTickets();

newArr[bookedTickets - 1] = tempTicket;

obj[FlightNum].setTicketArr(newArr);

cout << "\nCongratulations your flight is booked\n";

}

bool FlightExists(Plane\*& obj, int& totalPlanes, int flightNum)

{

bool flightExist = false;

for (int i = 0; i < totalPlanes; i++)

{

if (flightNum == obj[i].getFlightNumber())

{

flightExist = true;

break;

}

}

return flightExist;

}

int GetFlightID(Plane\*& obj, int& totalPlanes)

{

int flightNum = 0;

cout << "Enter Flight Num:";

flightNum = getChoice();

while (!FlightExists(obj, totalPlanes, flightNum))

{

if (FlightExists(obj, totalPlanes, flightNum))

{

return flightNum;

}

else if (!FlightExists(obj, totalPlanes, flightNum))

{

cout << "Invalid Flight Num: \n";

cout << "Enter Flight Num:";

flightNum = getChoice();

}

}

return flightNum;

}

void BookTickets(Plane\*& obj, int& totalPlanes, const string fileName, long long int passportno)

{

int flightNum = 0, flightIndex = -1;

DisplayAvailableFlights(obj, totalPlanes);

flightNum = GetFlightID(obj, totalPlanes);

flightIndex = returnFlightIndex(obj, totalPlanes, flightNum);

while (obj[flightIndex].getBookedTickets() == obj[flightIndex].getAvailableTickets())

{

if (obj[flightIndex].getBookedTickets() == obj[flightIndex].getAvailableTickets())

{

cout << "The Flight is full \n";

DisplayFlights(obj, totalPlanes);

flightNum = GetFlightID(obj, totalPlanes);

flightIndex = returnFlightIndex(obj, totalPlanes, flightNum);

}

}

InputFlightDetails(obj, flightIndex, passportno);

system("pause");

}

int GetPassengerTickets(Plane\*& obj, const int totalPlanes)

{

int passengerTickets = 0;

int totalbookedtickets = 0;

for (int i = 0; i < totalPlanes; i++)

{

totalbookedtickets += obj[i].getBookedTickets();

}

return totalbookedtickets;

}

int PassengerTickets(Ticket\* arrTicket, const int availableTickets, const int passportNum)

{

bool atleastOne = false;

int count = 0;

for (int counter = 0; counter < availableTickets; counter++)

{

if (arrTicket[counter].getPassportNumber() == passportNum)

{

atleastOne = true;

count++;

}

}

return count;

}

void EditPassengerTickets(Plane\*& obj, const int totalPlanes, Passenger\*& PassengerData, int index)

{

int totalbookedTickets = 0;

for (int i = 0; i < totalPlanes; i++)

{

totalbookedTickets += PassengerTickets(obj[i].getTickets(), obj[i].getBookedTickets(), PassengerData[index].GetPassportNo());

}

int\* passengersticket = new int[totalbookedTickets];

Ticket\* tempTicket = 0;

for (int i = 0; i < totalbookedTickets;)

{

for (int j = 0; j < totalPlanes; j++)

{

{

tempTicket = SearchTicketIDs(obj[j].getTickets(), obj[j].getBookedTickets(), PassengerData[index].GetPassportNo());

}

for (int k = 0; k < obj[j].getBookedTickets(); k++)

{

if (tempTicket[k].getID() != 0)

{

passengersticket[i] = tempTicket[k].getID();

i++;

}

}

}

}

PassengerData[index].SetTicketsBooked(totalbookedTickets);

PassengerData[index].SetTicketIds(passengersticket);

}

void EditPassengerRemoveTickets(Plane\*& obj, const int totalPlanes, Passenger\*& PassengerData, int index)

{

int totalbookedTickets = 0;

for (int i = 0; i < totalPlanes; i++)

{

totalbookedTickets += PassengerTickets(obj[i].getTickets(), obj[i].getBookedTickets(), PassengerData[index].GetPassportNo());

}

int\* passengersticket = new int[totalbookedTickets];

Ticket\* tempTicket = 0;

for (int i = 0; i < totalbookedTickets;)

{

for (int j = 0; j < totalPlanes; j++)

{

{

tempTicket = SearchTicketIDs(obj[j].getTickets(), obj[j].getBookedTickets(), PassengerData[index].GetPassportNo());

}

for (int k = 0; k < obj[j].getBookedTickets(); k++)

{

if (tempTicket[k].getID() != 0)

{

passengersticket[i] = tempTicket[k].getID();

i++;

}

}

}

}

PassengerData[index].SetTicketsBooked(0);

PassengerData[index].SetTicketIds(0);

}

void DisplayPassengerDetails(Passenger\* passenger, int& totalPassengers)

{

for (int i = 0; i < totalPassengers; i++)

{

cout << passenger[i];

}

}

void DisplayPassengerDetails(Passenger\* passenger, int& totalPassengers, string tempNationality)

{

for (int i = 0; i < totalPassengers; i++)

{

if (passenger[i].GetNationality() == tempNationality)

cout << passenger[i] << endl;

}

system("pause");

}

long long int getvalidpptno()

{

int choice = 0;

cin >> choice;

while (!cin.good())

{

cout << "Invalid Input (Wrong Data Type)\n";

cin.clear();

cin.ignore(std::numeric\_limits<long long int>::max(), '\n');

cout << "=>";

cin >> choice;

}

return choice;

}

long long int GetPassportNo()

{

long long int tempPassportno = 0;

cout << "Enter Passort Number: \n";

tempPassportno = getvalidpptno();

return tempPassportno;

}

void DisplayPassengerDetails(Passenger\* passenger, int& totalPassengers, long long int tempPassportNo)

{

for (int i = 0; i < totalPassengers; i++)

{

if (passenger[i].GetPassportNo() == tempPassportNo)

{

cout << passenger[i] << endl;

break;

}

}

system("pause");

}

void DisplayPassengerDetails(Plane\*& obj, int& totalPlanes, Passenger\* passenger, int& totalPassengers)

{

int flightNum = 0, flightIndex = -1, tempTotalTickets = 0;

Ticket\* tempTickets = 0;

DisplayFlights(obj, totalPlanes);

flightNum = GetFlightID(obj, totalPlanes);

flightIndex = returnFlightIndex(obj, totalPlanes, flightNum);

tempTotalTickets = obj[flightIndex].getBookedTickets();

tempTickets = obj[flightIndex].getTickets();

for (int i = 0; i < tempTotalTickets; i++)

{

DisplayPassengerDetails(passenger, totalPassengers, tempTickets[i].getPassportNumber());

}

system("pause");

}

int main()

{

bool isAdmin = false;

bool isPassenger = false;

string passengerFile, PlaneFile;

int totalPassengers = 0, totalPlanes = 0;

int userchoice = 0, signInChoice = 0, passengerChoice = 0, adminChoice = 0, displayPassengerChoice = 0;

Passenger newPassenger;

int index = -1;

Passenger\* PassengerData;

Plane\* PlaneData = nullptr;

Admin Admin1("Admin1", "FMS123", "Mr. Admin1");

PlaneFileName(PlaneFile);

PassengerFileName(passengerFile);

InputPassengerData(PassengerData, totalPassengers, passengerFile);

InputPlaneData(PlaneData, totalPlanes, PlaneFile);

// Testing everything

userchoice = MainMenu();

while (userchoice != 0)

{

if (userchoice == 1)

{

isAdmin = LoginPortal(Admin1);

if (isAdmin)

{

adminChoice = AdminMenu();

while (adminChoice >= 1 && adminChoice <= 5)

{

cin.ignore();

if (adminChoice == 6)

break;

else

{

if (adminChoice == 1)

SearchFlights(PlaneData, totalPlanes);

else if (adminChoice == 2)

{

AddFlights(PlaneData, totalPlanes, PlaneFile);

}

else if (adminChoice == 3)

{

RemoveFlight(PlaneData, totalPlanes, PassengerData, totalPassengers, PlaneFile, passengerFile);

}

else if (adminChoice == 4)

EditFlightDetails(PlaneData, totalPlanes, PlaneFile);

else

{

displayPassengerChoice = DisplayPassengerMenu();

while (displayPassengerChoice >= 1 && displayPassengerChoice <= 3)

{

if (displayPassengerChoice == 4)

break;

else

{

if (displayPassengerChoice == 1)

DisplayPassengerDetails(PlaneData, totalPlanes, PassengerData, totalPassengers);

else if (displayPassengerChoice == 2)

{

string tempInput;

cin.ignore();

cout << "Enter nationality: ";

getline(cin, tempInput, '\n');

DisplayPassengerDetails(PassengerData, totalPassengers, tempInput);

}

else

{

long long int tempPPT = 0;

tempPPT = GetPassportNo();

DisplayPassengerDetails(PassengerData, totalPassengers, tempPPT);

}

}

displayPassengerChoice = DisplayPassengerMenu();

}

}

}

adminChoice = AdminMenu();

}

}

}

else if (userchoice == 2)

{

signInChoice = SignInMenu();

while (signInChoice >= 1 && signInChoice <= 2)

{

if (signInChoice == 1)

{

isPassenger = LoginPortal(PassengerData, totalPassengers, index);

if (isPassenger)

{

passengerChoice = PassengerMenu();

while (passengerChoice >= 1 && passengerChoice <= 4)

{

if (passengerChoice == 1)

DisplayAvailableFlights(PlaneData, totalPlanes);

else if (passengerChoice == 2)

{

long long int tempPassportNo = PassengerData[index].GetPassportNo();

BookTickets(PlaneData, totalPlanes, PlaneFile, tempPassportNo);

OutputPlaneData(PlaneData, totalPlanes, PlaneFile);

InputPlaneData(PlaneData, totalPlanes, PlaneFile);

EditPassengerTickets(PlaneData, totalPlanes, PassengerData, index);

InsertPassenger(PassengerData, totalPassengers, passengerFile);

InputPassengerData(PassengerData, totalPassengers, passengerFile);

}

else if (passengerChoice == 3)

{

EditPassengerDetails(PassengerData, totalPassengers, index);

InsertPassenger(PassengerData, totalPassengers, passengerFile);

InputPassengerData(PassengerData, totalPassengers, passengerFile);

}

else

SearchFlights(PlaneData, totalPlanes);

passengerChoice = PassengerMenu();

}

}

}

else

{

newPassenger = SignUp(PassengerData, totalPassengers, passengerFile);

InsertPassenger(PassengerData, totalPassengers, passengerFile, newPassenger);

InputPassengerData(PassengerData, totalPassengers, passengerFile);

}

signInChoice = SignInMenu();

}

}

userchoice = MainMenu();

}

return 0;

}