**REPORT OF MINI-PROJECT**

**Name of the sub-system:**

**Railway Ticket Booking**

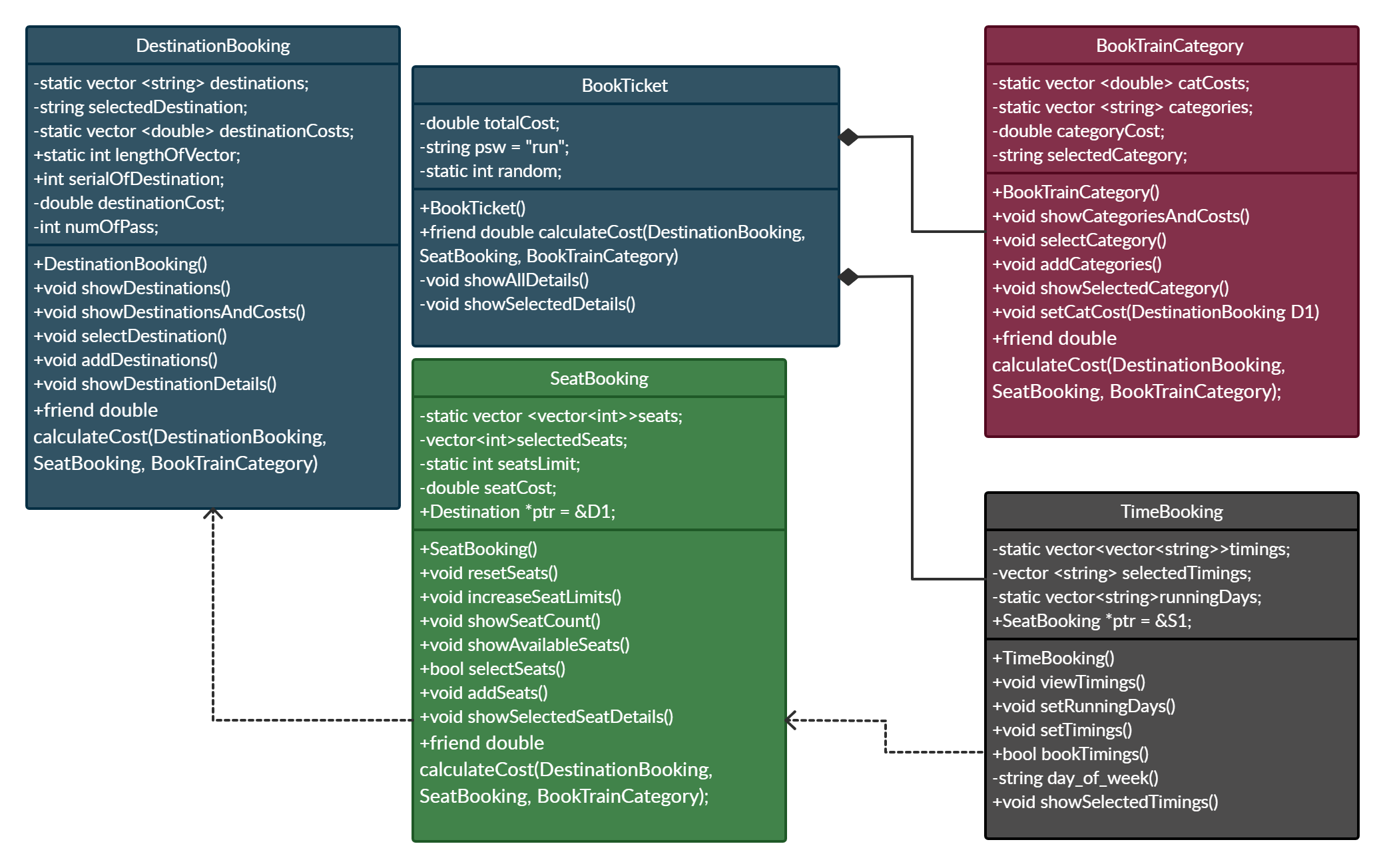
**Features/Objectives:**

To begin with, the objective of this sub-system, as defined by its name, is to let a user book a railway ticket for his required destination.

Following features have been included during this implementation:

* The user can view all the required details, and first of all, choose whatever destination he desires, if available in the given list.
* Based on the selected destination, the user is displayed with the number of seats available for the specific destination, the arrival, departure timings and the days on which trains to the selected destinations run, followed by inquiring details of selection from the user respectively.
* The user is just required to enter serial numbers of options he wants to select, without worrying about the entire words and their cases. The system will do it all for you.
* When the user is done with entering/selecting the details, different classes of the system interact and calculate the overall cost which is displayed alongwith other details.
* The system has been designed with validations at each input and a user-friendly environment (with time delays and neat presentation), so as to guide the user if he misunderstands and enters something wrong.
* Apart from the user access, the system also contains administrative access protected by the password ‘run’.
* Anyone with the admin access can add new destinations, increase seat limits, refresh seats, change timings etc.
* Because of the admin functionalities, the system has potential to adopt any additions that are desired by the administrator, without messing up with the code.

**Class Diagram:**



**Coding:**

#include <iostream>

#include <vector>

#include <iomanip>

#include <algorithm>

#include <sstream>

#include <unistd.h>

#include <ctime>

using namespace std;

class DestinationBooking;

class SeatBooking;

class BookTrainCategory;

class BookTicket;

class DestinationBooking {

    private:

        static vector <string> destinations;

        string selectedDestination;

        static vector <double> destinationCosts;

        double destinationCost;

        int numOfPass;

    public:

        static int lengthOfVector;

        int serialOfDestination;

        DestinationBooking() {

            selectedDestination = "";

            lengthOfVector = destinations.size();

        }

        void privateUse(int j) {

            cout << destinations[j];

        }

        void showDestinations(int i) {

            cout << i+1 << right << setw(19) << destinations[i];

        }

        void showDestinationsAndCosts() {

            for(int i=0; i < destinations.size(); i++) {

                showDestinations(i);

                cout << right << setw(10) << destinationCosts[i];

                cout << endl; };

        }

        void selectDestination() {

            cout << "\n" << right << setw(21) << "Destinations" << right << setw(12) << "Costs" "\n\n";

            showDestinationsAndCosts();

            bool error = true;

            while (error == true) {

                int serial;

                cout << "\nPlease Enter The Serial Number of Destination You Wish To Select >> "; cin >> serial;

                while(true) {

                    if(cin.fail()) {

                        cin.clear();

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

                        cout << "\nPlease enter a number only >> "; cin >> serial;

                    }

                    if(!cin.fail()) {

                        break;

                    }

                }

                if (serial >= 1 && serial <= destinations.size()) {

                    selectedDestination = destinations[serial-1];

                    serialOfDestination = serial - 1;

                    bool errorPass = true;

                    while (errorPass == true) {

                        cout << "\nPlease Enter the Number of Passengers >> "; cin >> numOfPass;

                        while(true) {

                            if(cin.fail()) {

                                cin.clear();

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

                                cout << "\nPlease enter a number only >> "; cin >> numOfPass;

                            }

                            if(!cin.fail()) {

                                break;

                            }

                        }

                        if (numOfPass > 0) {

                            destinationCost = numOfPass \* destinationCosts[serial-1];

                            errorPass = false;

                        }

                        else {

                            cout << "\nNumber of passengers must be greater than zero!\n";

                        }

                    }

                    error = false;

                }

                else {

                    cout << "\nPlease Enter a valid serial number!" << endl;

                }

            }

        }

        void addDestinations() {

            cout << "\nHow many destinations do you want to increase? >> "; int numOfDes; cin >> numOfDes;

            for (int i=0; i < numOfDes; i++) {

                cout << "\nPlease Enter the New Destination: "; string newDes; cin.ignore(); getline(cin, newDes);

                destinations.push\_back(newDes);

                cout << "\nPlease Enter the cost for " << newDes << " >> "; double cost; cin >> cost;

                destinationCosts.push\_back(cost);

            }

            lengthOfVector = destinations.size();

        }

        void showDestinationDetails() {

            cout << "\nYour Selected Destination is " << selectedDestination << endl;

        }

        friend class BookTrainCategory;

        friend double calculateCost(DestinationBooking, SeatBooking, BookTrainCategory);

};

vector<string> DestinationBooking::destinations={"Quetta", "Peshawar", "Hyderabad"};

vector<double> DestinationBooking::destinationCosts={4000, 6000, 9900};

int DestinationBooking:: lengthOfVector;

class BookTrainCategory {

    private:

        static vector <string> categories;

        string selectedCategory;

        static vector <double> categoryCosts;

        double categoryCost;

        int serial;

    public:

        BookTrainCategory() {

         selectedCategory = "";

        }

        void showCategoriesAndCosts() {

            cout << "\n" << right << setw(21) << "Categories" << right << setw(12) << "Costs" "\n\n";

            for(int i=0; i<categories.size(); i++) {

                cout << i+1 << right << setw(20) << categories[i] << right << setw(10) << categoryCosts[i] << endl;

            }

        }

        void setCatCost(DestinationBooking D1) {

            categoryCost = D1.numOfPass \* categoryCosts[serial-1];

        }

        void selectCategory() {

            showCategoriesAndCosts(); bool Caterror = true;

            while (Caterror == true) {

                cout << "\nPlease Enter The Category Number You Want to Select: "; cin >> serial;

                while(true) {

                    if(cin.fail()) {

                        cin.clear();

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

                        cout << "\nPlease enter a number only >> "; cin >> serial;

                    }

                    if(!cin.fail()) {

                        break;

                    }

                }

                if (serial >= 1 && serial <= categories.size()) {

                    selectedCategory = categories[serial-1];

                    Caterror = false;

                }

                else {

                    cout << "\nInvalid Category!\n";

                }

            }

        }

        void addCategories() {

            cout << "\nPlease enter number of categories you want to increase >> "; int numOfCat; cin >> numOfCat;

            for(int i=0; i<numOfCat; i++) {

                cout << "\nEnter the new category: "; string newCat; cin >> newCat;

                categories.push\_back(newCat);

                cout << "Enter the cost for the new category: "; double newCost; cin >> newCost;

                categoryCosts.push\_back(newCost);

            }

        }

        void showSelectedCategory() {

            cout << "Your Selected Category is " << selectedCategory << endl;

        }

        friend double calculateCost(DestinationBooking, SeatBooking, BookTrainCategory);

};

vector <string> BookTrainCategory::categories={"BusinessClass", "Economy"};

vector <double> BookTrainCategory::categoryCosts={5000, 2000};

class SeatBooking {

    private:

        static vector<vector<int>> seats;

        vector <int> selectedSeats;

        double seatCost;

        static int seatsLimit;

    public:

        SeatBooking() {

            selectedSeats = {};

        }

        DestinationBooking D1;

        DestinationBooking \*Dptr = &D1;

        void resetSeats(int i, int j) {

            for(j; j<=seatsLimit; j++) {

                seats[i].push\_back(j);

            }

        }

        void increaseSeatLimits() {

            cout << "\nPlease enter the number of seats you want to increase >> "; int seatsToIncrease; cin >> seatsToIncrease;

            for(int x=0; x<seats.size(); x++) {

                for(int y=seatsLimit+1; y<=seatsLimit+seatsToIncrease; y++) {

                    seats[x].push\_back(y);

                }

            }

            seatsLimit += seatsToIncrease;

        }

        void addSeats() {

            for(int i=seats.size(); i<Dptr->lengthOfVector; i++) {

                seats.push\_back({});

            }

            for(int i=0; i<Dptr->lengthOfVector; i++) {

                if(seats[i].size() == 0) {

                    resetSeats(i, 1);

                }

            }

        }

        void showSeatCount(int j) {

            cout << right << setw(10) << seats[j].size();

        }

        void showAvailableSeats() {

            cout << "\nAvailable Seats For Your Destination:";

            for(int i=0; i<seats[Dptr->serialOfDestination].size(); i++) {

                cout << right << setw(5) << seats[Dptr->serialOfDestination][i];

            }

        }

        bool selectSeats() {

            showAvailableSeats(); bool proceed = true; bool continued = false;

            vector<int>availSeats = seats[Dptr->serialOfDestination];

            while (proceed == true) {

                cout << "\n\nHow many seats do you want to book >> "; int numOfSeats; cin >> numOfSeats;

                while(true) {

                    if(cin.fail()) {

                        cin.clear();

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

                        cout << "\nPlease enter a number only >> "; cin >> numOfSeats;

                    }

                    if(!cin.fail()) {

                        break;

                    }

                }

                cout << "\n";

                if (numOfSeats > availSeats.size()) {

                    cout << "Required amount of Seats Not Available!\n" << endl;

                    cout << "Do you want to re-enter or exit?\n";

                    cout << "Enter R to re-enter and E to exit! >> "; char quest; cin >> quest;

                    if(quest ==  'e' || quest == 'E') {

                        proceed = false;

                        return proceed;

                    }

                }

                else {

                    for(int j=0; j<numOfSeats; j++) {

                        cout << "Please Enter Seat Number " << j+1 << " >> "; int selectedSeat; cin >> selectedSeat;

                        while(true) {

                            if(cin.fail()) {

                                cin.clear();

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

                                cout << "\nPlease enter a number only >> "; cin >> selectedSeat;

                            }

                            if(!cin.fail()) {

                                break;

                            }

                        }

                        vector<int>::iterator it = find(availSeats.begin(), availSeats.end(), selectedSeat);

                        if (it != availSeats.end()) {

                            int index = distance(availSeats.begin(), it);

                            selectedSeats.push\_back(selectedSeat);

                            availSeats.erase(availSeats.begin()+index);

                        }

                        else {cout << "\nInvalid Seat Number!\n"; j--;}

                        cout << endl;

                    }

                    seats[Dptr->serialOfDestination] = availSeats;

                    seatCost = numOfSeats \* 1000;

                    continued = true;

                    return continued;

                }

            }

            return 0;

        }

        void showSelectedSeatDetails() {

            cout << "You have selected the following seats: ";

            for(int i=0; i<selectedSeats.size(); i++) {

                cout << "\t" << selectedSeats[i];

            }

        }

        friend double calculateCost(DestinationBooking, SeatBooking, BookTrainCategory);

};

vector<vector<int>>SeatBooking::seats={{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}};

int SeatBooking::seatsLimit = 10;

class TimeBooking {

    private:

        static vector<vector <string>>timings;

        static vector<string> runningDays;

        vector <string> selectedTimings;

    public:

        SeatBooking S1;

        TimeBooking() {

            selectedTimings = {};

        }

        void setRunningDays() {

            int quest; int serial; char change = 'Y';

            cout << "\n1. Do you want to make changes in the existing running days?" << endl;

            cout << "2. Do you want to set running days for newly added destinations?" << endl;

            cout << "\nPlease Enter The Serial Number of Option You Would Like to Select >> "; cin >> quest;

            while (change != 'N') {

                if (quest == 1) {

                    cout << "\n";

                    for(int i=0; i<S1.Dptr->lengthOfVector; i++) {

                        S1.Dptr->showDestinations(i);

                        cout << endl;

                    }

                    cout << "\n";

                    cout << "Please enter the serial number of destination you want to make changes in >> "; cin >> serial;

                    cout << "\nRunning day for the selected destination is " << runningDays[serial-1];

                    cout << "\n\nPlease enter the new running day for selected destination: "; string newDay; cin >> newDay;

                    runningDays[serial-1] = newDay;

                }

                else if (quest == 2){

                    bool ran = false;

                    string day;

                    for (int j=runningDays.size(); j < S1.Dptr->lengthOfVector; j++) {

                        ran = true;

                        cout << "\nPlease enter the running day of train for destination ";S1.Dptr->privateUse(j); cout << " >> " ; cin >> day;

                        runningDays.push\_back(day);

                    }

                    if (ran == false) {

                        cout << "No new destinations have been added!\n";

                    }

                }

                cout << "\nDo you want to make more changes? (Y/N) >> "; cin >> change;

            }

        }

        void setTimings() {

            int quest; int serial; char change = 'Y';

            cout << "\n1. Do you want to make changes in the existing timings?" << endl;

            cout << "2. Do you want to set timings for newly added destinations?" << endl;

            cout << "\nPlease Enter The Serial Number of Option You Would Like to Select >> "; cin >> quest;

            while (change != 'N') {

                if (quest == 1) {

                    cout << "\n";

                    for(int i=0; i<S1.Dptr->lengthOfVector; i++) {

                        S1.Dptr->showDestinations(i);

                        cout << endl;

                    }

                    cout << "\n";

                    cout << "Please enter the serial number of destination you want to make changes in >> "; cin >> serial;

                    cout << "\nTimings for the selected destination are: " << endl;

                    cout << "\nArrival Time: " << timings[serial-1][0];

                    cout << "\nDeparture Time: " << timings[serial-1][1] << endl;

                    cout << "\nPlease enter the new arrival timings for selected destination(24-hour format): "; string newArrivalTime;

                    cin >> newArrivalTime; timings[serial-1][0] = newArrivalTime;

                    cout << "\nPlease enter the new departure timings for selected destination(24-hour format): "; string newDepartureTime;

                    cin >> newDepartureTime; timings[serial-1][1] = newDepartureTime;

                }

                else if(quest == 2){

                    cout << "\n";

                    for(int i=0; i<S1.Dptr->lengthOfVector; i++) {

                        S1.Dptr->showDestinations(i);

                        cout << endl;

                    }

                    cout << "\n";

                    string timeToAdd; bool ran = false;

                    for (int j=timings.size(); j < S1.Dptr->lengthOfVector; j++) {

                        timings.push\_back({});

                        ran = true;

                        cout << "\nPlease enter the arrival time of train for destination "; S1.Dptr->privateUse(j);

                        cout << " (24-hour format) >> "; cin >> timeToAdd; timings[j].push\_back(timeToAdd);

                        cout << "\nPlease enter the departure time of train for destination "; S1.Dptr->privateUse(j);

                        cout << " (24-hour format) >> "; cin >> timeToAdd; timings[j].push\_back(timeToAdd);

                    }

                    if (ran == false) {

                        cout << "No new destinations have been added!" << endl;

                    }

                }

                cout << "\nDo you want to make more changes? (Y/N) >> "; cin >> change;

            }

        }

        void viewTimings(int i) {

            cout << right << setw(18) << timings[i][0] << right << setw(22) << timings[i][1] << right << setw(26) << runningDays[i];

        }

    private:

        string day\_of\_week(int y, int m, int d) {

            static int t[] = {0, 3, 2, 5, 0, 3, 5, 1, 4, 6, 2, 4};

            y -= m < 3;

            int z;

            z = (y + y/4 - y/100 + y/400 + t[m-1] + d) % 7;

            switch(z){

                case 0: return "Sunday";

                case 1: return "Monday";

                case 2: return "Tuesday";

                case 3: return "Wednesday";

                case 4: return "Thursday";

                case 5: return "Friday";

                case 6: return "Saturday";

            }

            return 0;

        }

    public:

        bool bookTimings() {

            bool invalid = false, proceed = true;

            while(invalid == false && proceed == true) {

                int y,m,d;

                while(true) {

                    cout<<"\nEnter the date at which you want to depart (1-31) >> ";

                    cin>>d;

                    while(true) {

                        if(cin.fail()) {

                            cin.clear();

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

                            cout << "\nPlease enter a number only >> "; cin >> d;

                        }

                        if(!cin.fail()) {

                            break;

                        }

                    }

                    if (d >= 1 && d <= 31) {

                        break;

                    }

                    else {

                        cout << "\nDate must be within the given limit!Please try again..\n";

                    }

                }

                while(true) {

                    cout<<"\nEnter the month in which you want to depart (1-12) >> ";

                    cin>>m;

                    while(true) {

                        if(cin.fail()) {

                            cin.clear();

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

                            cout << "\nPlease enter a number only >> "; cin >> m;

                        }

                        if(!cin.fail()) {

                            break;

                        }

                    }

                    if (m >= 1 && m <= 12) {

                        break;

                    }

                    else {

                        cout << "\nMonth must be within the given limit!Please try again..\n";

                    }

                }

                time\_t now = time(0);

                tm \*ltm = localtime(&now);

                int currentYear = ltm->tm\_year + 1900;

                // int currentMonth = ltm->tm\_mon + 1;

                // int currentDate = ltm->tm\_mday;

                // stringstream cy, cm, cd;

                // cy << currentYear ; cm << currentMonth ; cd << currentDate;

                // string cyear, cmonth, cdate;

                // cy >> cyear; cm >> cmonth; cd >> cdate;

                // string currentDay = cdate + "-" + cmonth + "-" + cyear;

                while(true) {

                    cout<<"\nEnter the year in which you want to depart (yyyy) >> ";

                    cin>>y;

                    while(true) {

                        if(cin.fail()) {

                            cin.clear();

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

                            cout << "\nPlease enter a number only >> "; cin >> y;

                        }

                        if(!cin.fail()) {

                            break;

                        }

                    }

                    if(y >= currentYear) {

                        break;

                    }

                    else {

                        cout << "\nInvalid format or year!Please try again..\n";

                    }

                }

                string dayOfWeek = day\_of\_week(y, m, d);

                stringstream yy, mm, dd;

                yy << y ; mm << m ; dd << d;

                string year, month, date;

                yy >> year; mm >> month; dd >> date;

                string selectedDay = date + "-" + month + "-" + year;

                cout << "\nChecking availability";

                    for (int i=0; i<3; i++) {

                        cout << "."; usleep(1000000);

                    }

                    cout << "\n";

                if (dayOfWeek == runningDays[S1.Dptr->serialOfDestination]) {

                    selectedTimings.push\_back(timings[S1.Dptr->serialOfDestination][0]);

                    selectedTimings.push\_back(timings[S1.Dptr->serialOfDestination][1]);

                    selectedTimings.push\_back(selectedDay);

                    invalid = true;

                    return invalid;

                }

                else {

                    cout << "\nNo Train Available on the following date, do you wish to select new dates(enter Y) or abort(enter N)? >> ";

                    char ask; cin >> ask;

                    if(ask == 'N' || ask == 'n') {

                        cout << "\nThankyou for visiting us!\n\n";

                        proceed = false;

                        return proceed;

                    }

                }

            }

            return 0;

        }

        void showSelectedTimings() {

            cout << "\nYour arrival time is >> " << selectedTimings[0];usleep(1000000);

            cout << "\nYour departure time is >> " << selectedTimings[1];usleep(1000000);

            cout << "\nThe selected date is >> " << selectedTimings[2];usleep(1000000);

        }

};

vector<vector<string>>TimeBooking::timings = {{"06:00", "06:10"}, {"17:00", "17:20"}, {"20:00", "20:30"}};

vector<string>TimeBooking::runningDays = {"Sunday", "Monday", "Tuesday"};

class BookTicket {

    private:

        BookTrainCategory T1;

        TimeBooking B1;

        double totalCost;

        string psw = "run";

        static int random;

    public:

        BookTicket() {

            B1.S1.addSeats();

            cout << "\nDo you want to access the system as an administrator or a user?" << endl;

            cout << "Enter A for Admin Access, U for User Access, E to Exit the program! >> "; char adminOrUser; cin >> adminOrUser;

            if (adminOrUser == 'U' || adminOrUser == 'u') {

                cout << "\nDo you want to book a ticket or exit the system?" << endl;

                cout << "Enter B for Booking a Ticket and E for exitting the system >> ";

                char bookOrExit; cin >> bookOrExit;

                if(bookOrExit == 'B' || bookOrExit == 'b') {

                    cout << "\nPlease give us a moment to present details";

                    for (int i=0; i<3; i++) {

                        cout << "."; usleep(1000000);

                    }

                    cout << "\n";

                    showAllDetails();

                    cout << "\nProceeding";

                    for (int i=0; i<3; i++) {

                        cout << "."; usleep(1000000);

                    }

                    cout << "\n";

                    B1.S1.Dptr->selectDestination();

                    // B1.S1.Dptr->showDestinationDetails();

                    cout << "\nProceeding";

                    for (int i=0; i<3; i++) {

                        cout << "."; usleep(1000000);

                    }

                    cout << "\n";

                    T1.selectCategory();

                    // T1.showSelectedCategory();

                    T1.setCatCost(B1.S1.D1);

                    cout << "\nProceeding";

                    for (int i=0; i<3; i++) {

                        cout << "."; usleep(1000000);

                    }

                    cout << "\n";

                    bool seatsAcc = B1.S1.selectSeats();

                    if (seatsAcc == true) {

                        cout << "\nProceeding";

                        for (int i=0; i<3; i++) {

                            cout << "."; usleep(1000000);

                        }

                        cout << "\n";

                        bool timeAcc = B1.bookTimings();

                        if (timeAcc == true) {

                            cout << "\nTicket Successfully Booked!Following are your details:\n";

                            showSelectedDetails();

                        }

                    }

                }

                else if (bookOrExit == 'e' || bookOrExit == 'E'){

                    cout << "\nThankyou for visiting us!\n\n";

                }

                else {

                    cout << "\nInvalid Choice!Program Terminated..\n";

                }

            }

            else if(adminOrUser == 'A' || adminOrUser == 'a') {

                cout << "\n\nPassword required for administrative access >> "; string enteredPsw; cin >> enteredPsw;

                if(enteredPsw == psw) {

                    cout << "\nFollowing operations can be performed by the administrator: \n";

                    cout << "\n1. Add New Destinations" << endl;

                    cout << "2. Add New Train Categories" << endl;

                    cout << "3. Increase Seat Limit for each Destination" << endl;

                    cout << "4. Add/Reset Seats of all Destinations" << endl;

                    cout << "5. Add/Modify Timings" << endl;

                    cout << "6. Add/Modify Running Days" << endl; bool continued = true;

                    while (continued == true) {

                        cout << "\nPlease enter the serial number of operation you want to perform >> "; int serial; cin >> serial;

                        if (serial >= 1 && serial <= 6) {

                            if(serial == 1) {

                                B1.S1.Dptr->addDestinations();

                            }

                            else if(serial == 2) {

                                T1.addCategories();

                            }

                            else if(serial == 3) {

                                B1.S1.increaseSeatLimits();

                            }

                            else if(serial == 4) {

                                B1.S1.addSeats();

                            }

                            else if(serial == 5) {

                                B1.setTimings();

                            }

                            else {

                                B1.setRunningDays();

                            }

                            cout << "\nYou have successfully made changes!\n";

                            cout << "\nWould you like to continue making changes or exit?";

                            cout << "\nEnter C to continue, E to exit! >> "; char changeOrExit; cin >> changeOrExit;

                            if (changeOrExit == 'e' || changeOrExit == 'E') {

                                cout << "\nProgram successfully terminated!\n";

                                continued = false;

                            }

                        }

                        else {

                            cout << "\nInvalid Operation!\n";

                            cout << "Enter R to re-try and E to exit >> "; char tryOrExit; cin >> tryOrExit;

                            if (tryOrExit == 'e' || tryOrExit == 'E') {

                                cout << "\nProgram successfully terminated!\n";

                                continued = false;

                            }

                        }

                    }

                }

                else {

                    cout << "\nWrong Password, program terminated!\n" << endl;

                }

            }

            else if (adminOrUser == 'e' || adminOrUser == 'E'){

                cout << "\nThankyou for visiting us!\n\n";

            }

            else {

                cout << "\nInvalid Choice, Program Terminated.\n";

            }

        }

        void showAllDetails() {

            cout << "\n" << right << setw(20) << "Destinations" << right << setw(12) << "Seats" << right << setw(20) << "Arrival Times" << right << setw(22) << "Departure Times" << right << setw(22) << "Running Days";

            cout << "\n\n";

            for (int i=0; i<B1.S1.Dptr->lengthOfVector; i++) {

                B1.S1.Dptr->showDestinations(i);

                B1.S1.showSeatCount(i);

                B1.viewTimings(i);

                cout << "\n\n";

            }

        }

        void showSelectedDetails() {

            int ticketNum = rand() % 100 + random;

            random++;

            cout << "\nTicket Number >> " << ticketNum;

            B1.S1.Dptr->showDestinationDetails();

            usleep(1000000);

            T1.showSelectedCategory();

            usleep(1000000);

            B1.S1.showSelectedSeatDetails();

            usleep(1000000);

            B1.showSelectedTimings();

            usleep(1000000);

            cout << "\nThe overall cost is >> " << calculateCost(B1.S1.D1, B1.S1, T1) << "\n\n";

        }

        friend double calculateCost(DestinationBooking, SeatBooking, BookTrainCategory);

};

double calculateCost(DestinationBooking D1, SeatBooking S1, BookTrainCategory T1) {

    return (D1.destinationCost + S1.seatCost + T1.categoryCost);

}

int BookTicket::random = 0;

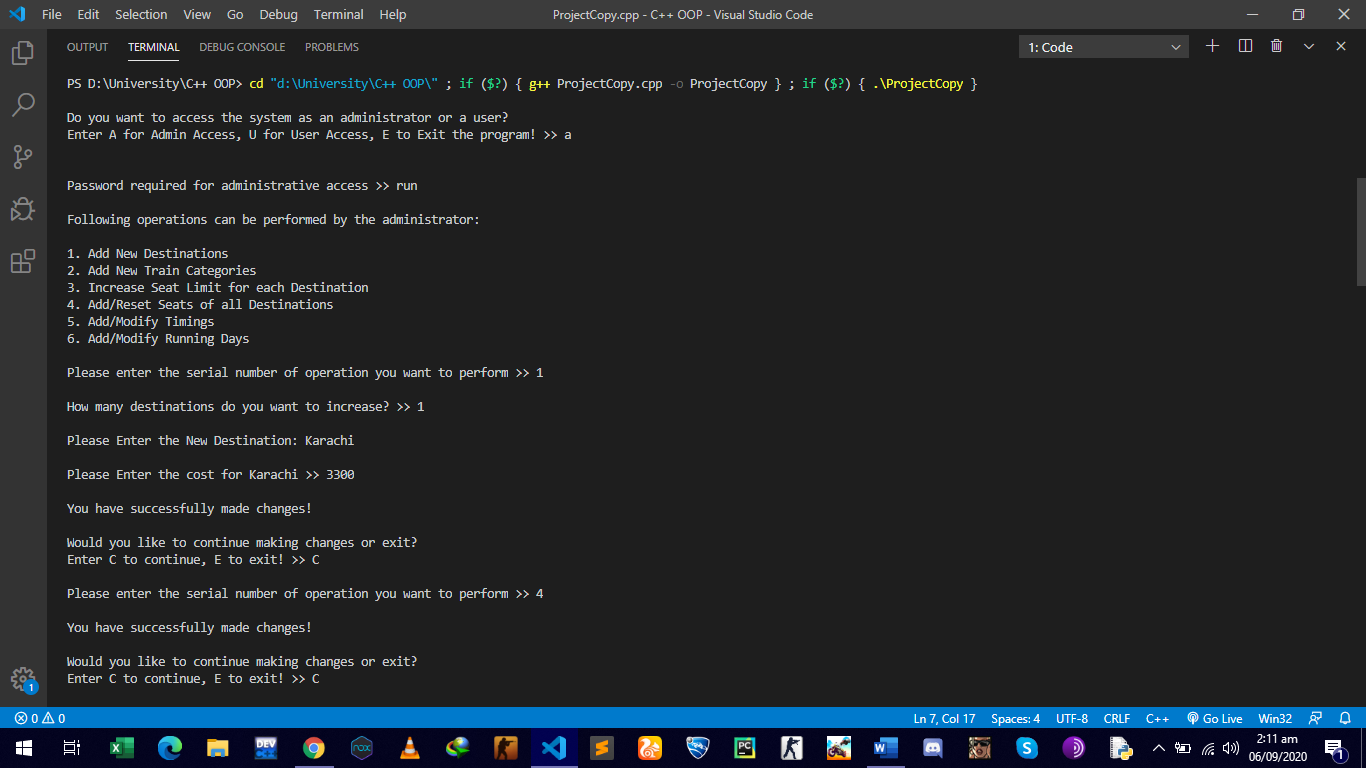
int main() {

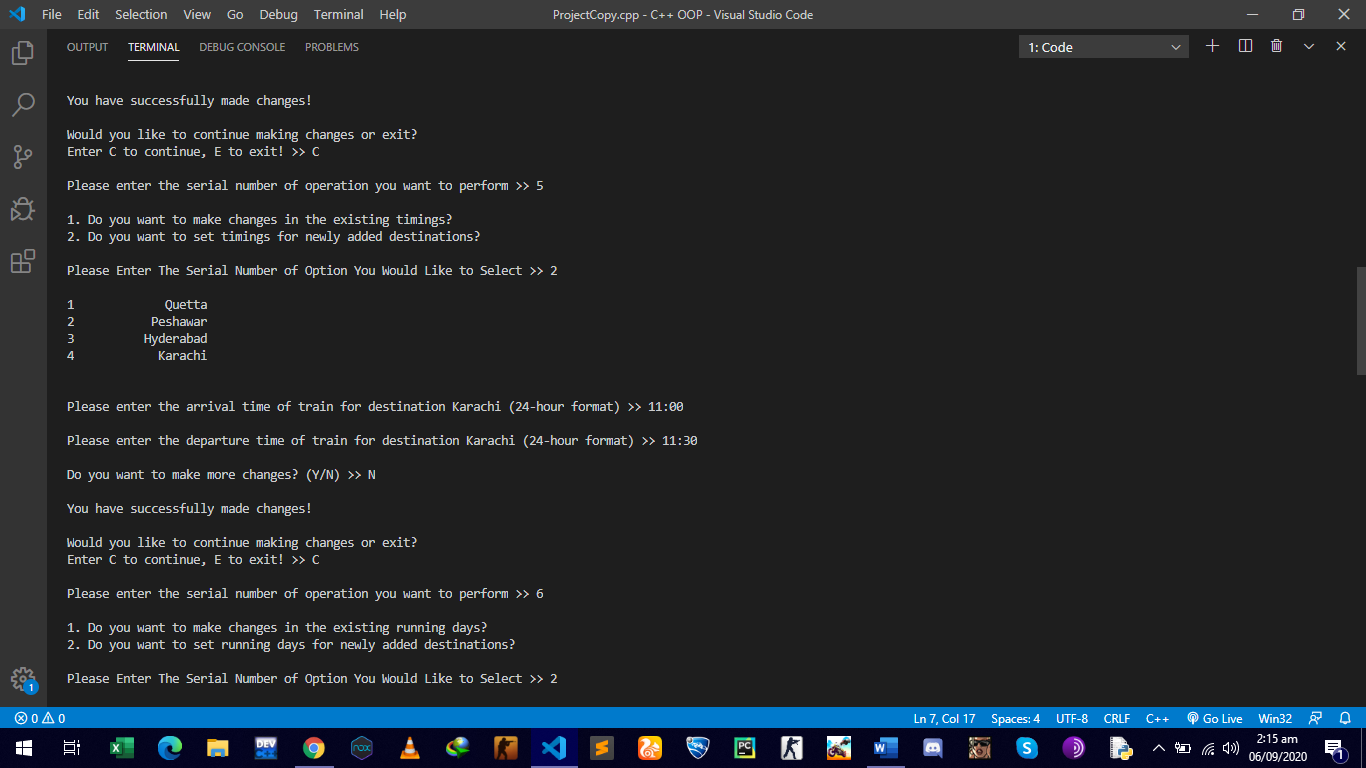
    BookTicket Ticket1;

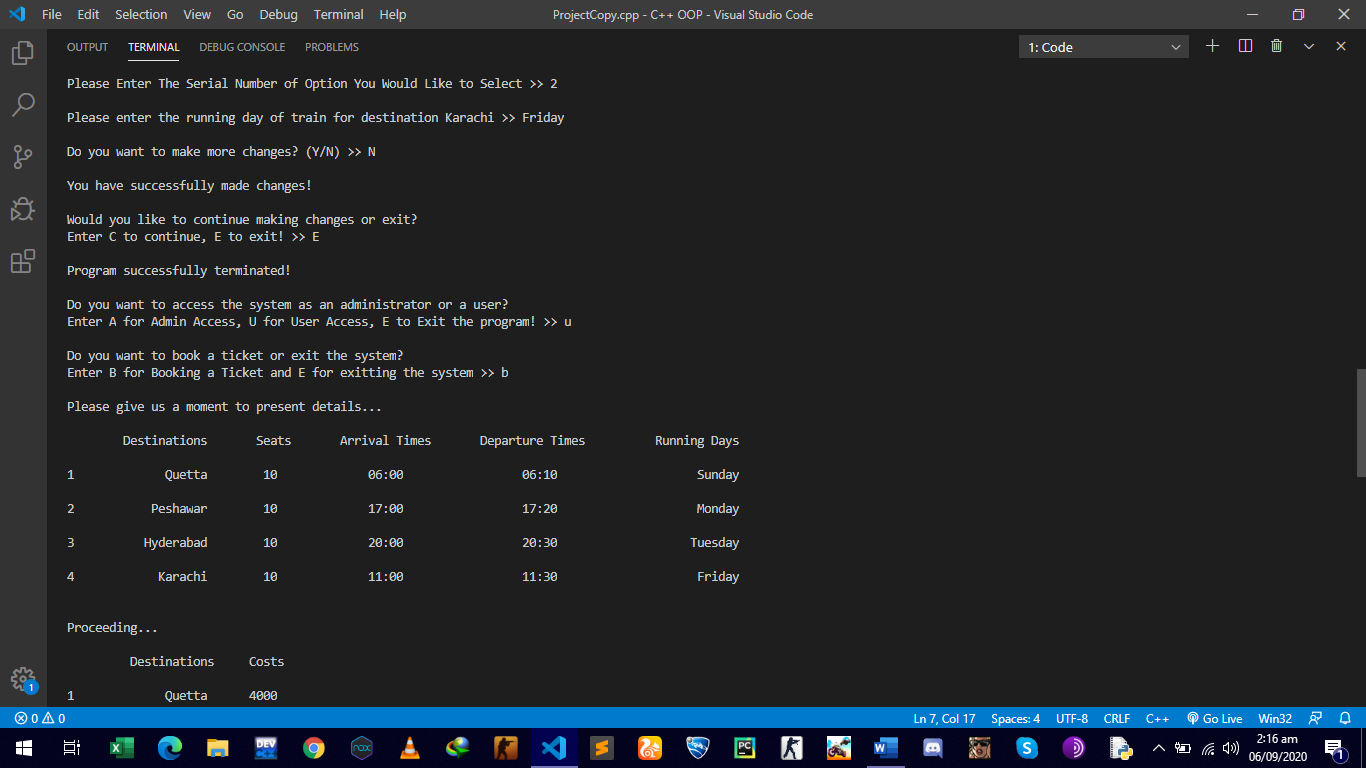
    BookTicket Ticket2;

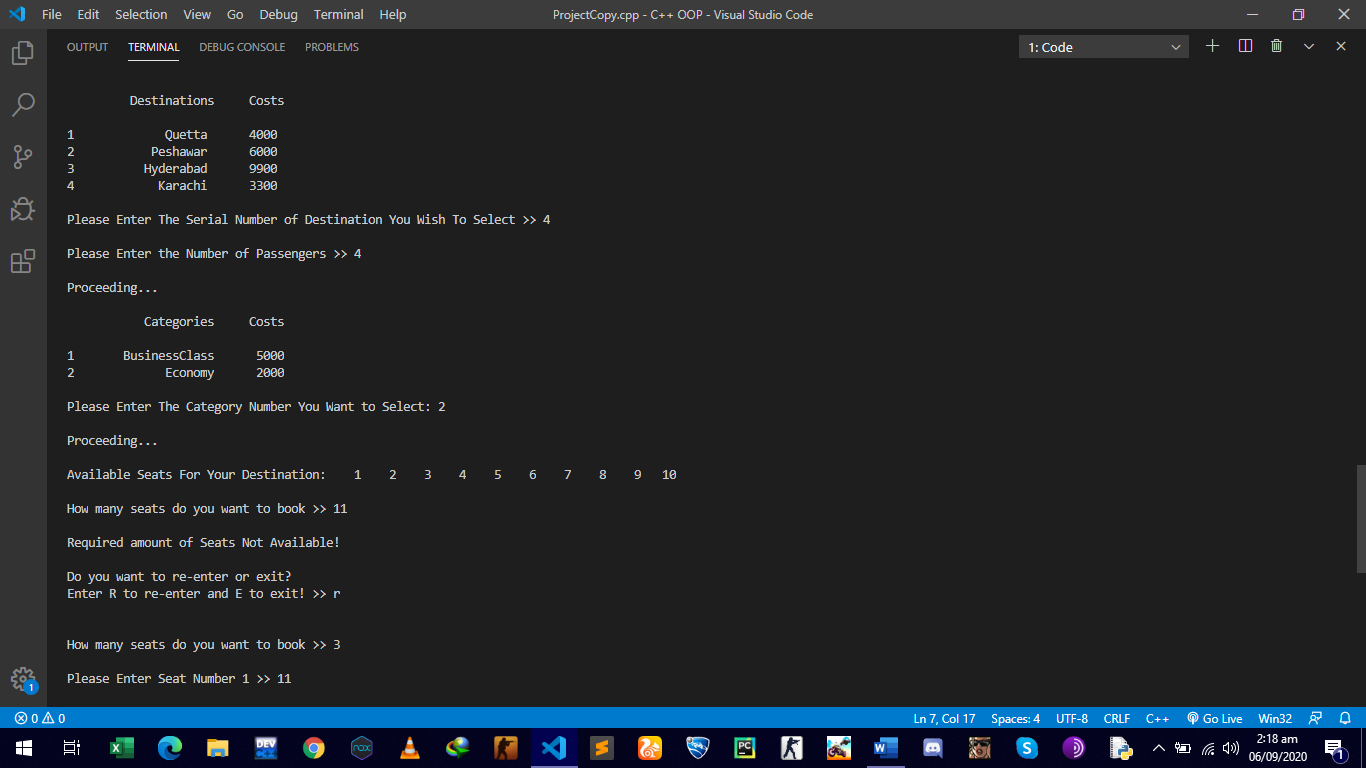
}

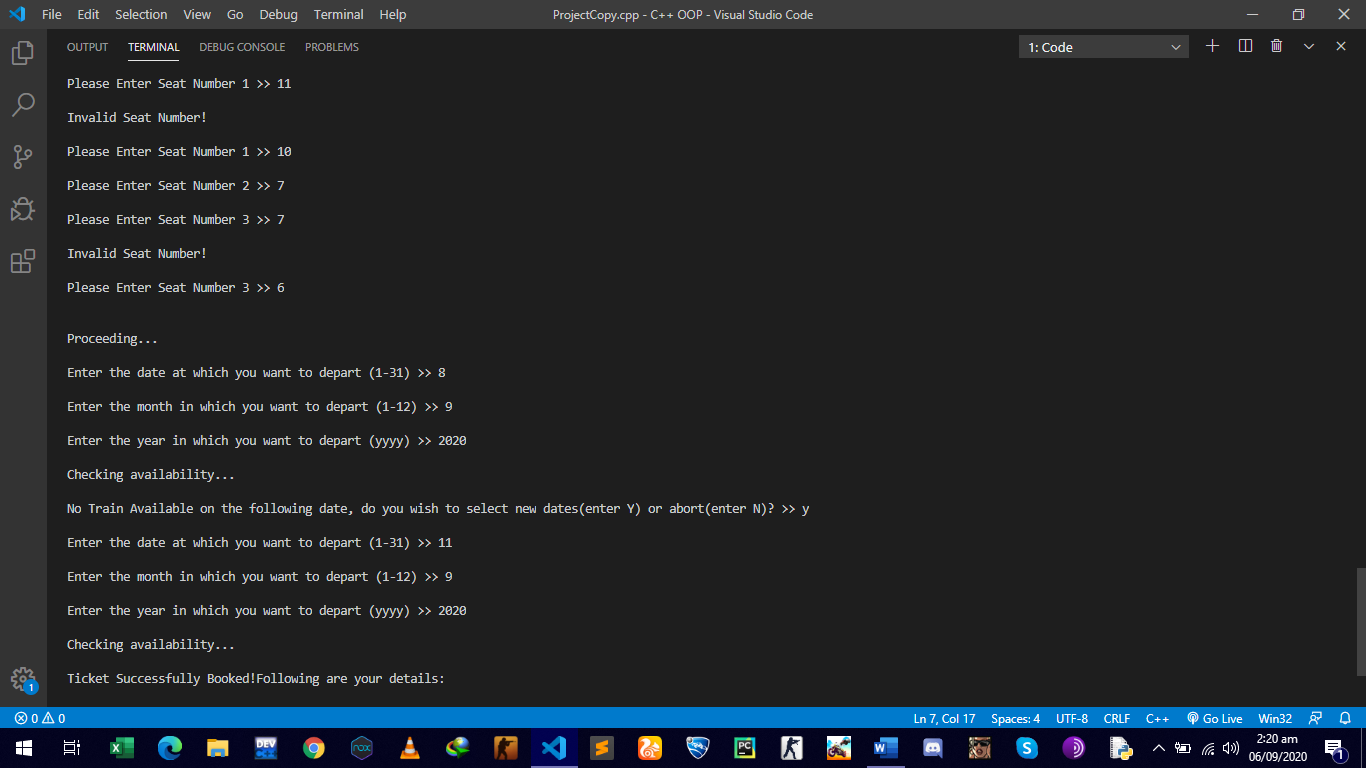
**Output Snaps:**

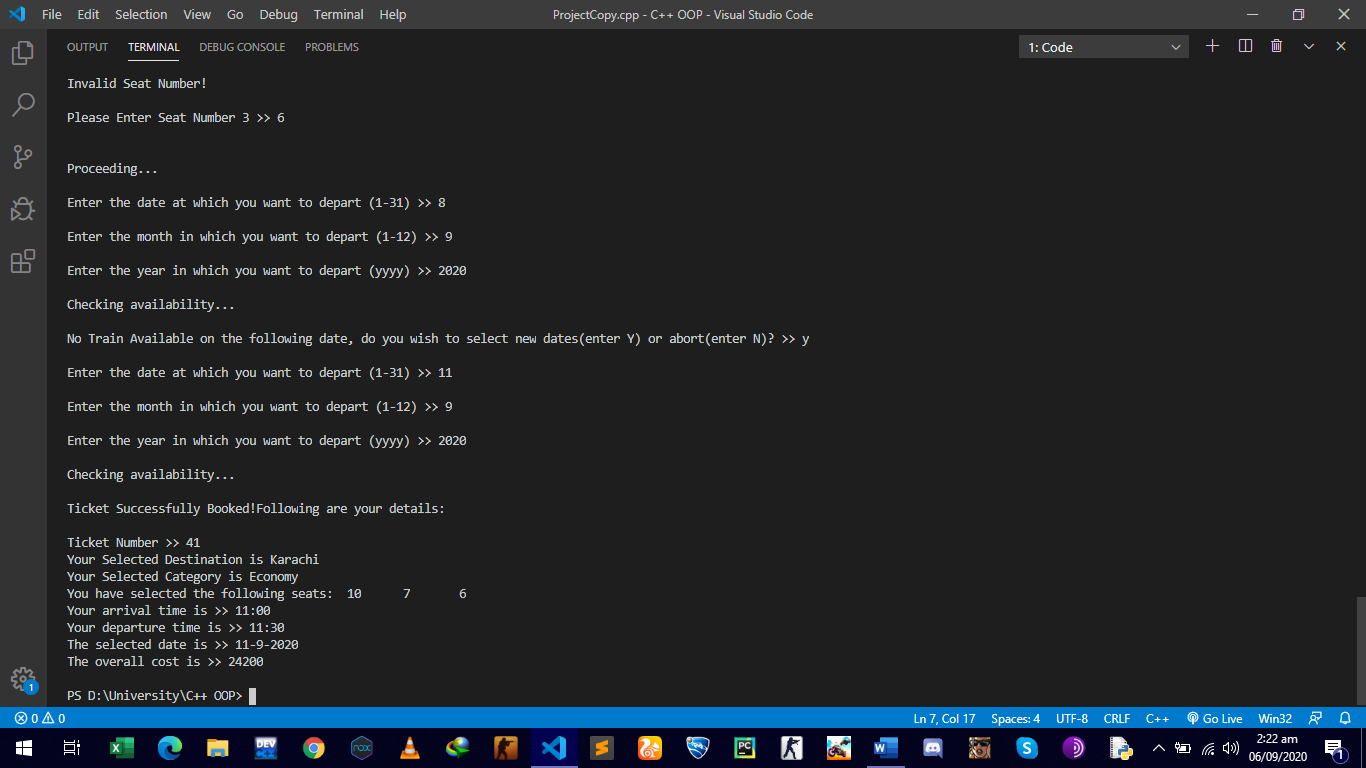
****

****

****

****

****

****

**Limitations:**

To be fair, the system works far better than what I expected while designing it. However, there are still some improvements I wished but couldn’t implement due to shortage of time. If the user wants to book the ticket for the same day he accesses the system, the system cannot check if the train has already departed on the following day or is yet to depart. Also, filing would have been a great option to hold the ticket records in. Apart from the mentioned limitations, the system works fine enough, and is open to any changes the administrator of the system desires to make.