Assignment

Problem Based Learning

Problem Statement:

Bahria University Computing and Innovation Society (BUCIS) at Bahria University Karachi Campus organizes a variety of academic and recreational events every semester — including seminars, workshops, coding competitions, sports festivals, exhibitions, and guest speaker sessions. Currently, the event registration and ticket booking processes are handled manually using spreadsheets and handwritten lists. This has resulted in errors such as double bookings, loss of registration data, incomplete payment tracking, and difficulty managing seat availability.

To resolve these issues and professionalize event handling, the department has initiated the development of a BUCIS Event Management and Ticket Booking System (BETBS) — a consolebased C++ application. This system must automate student registration, event booking, seat management, cancellation handling, and ensure digital record-keeping for every event organized by BUCIS. The system must provide the following features:

	Functionality	Description
1	Customer Registration	Record customer name, CNIC/passport number, and contact. Ensure CNIC uniqueness.
2	Event Catalog Management	Maintain a list of events (Event ID, Event Name, Date, Venue, and Ticket Price).
3	Book Tickets	Allow a customer to book up to 5 tickets for an event. Check for seat availability before confirming.
4	Search Customer	Search by CNIC to view booked events and ticket details.
5	Search Event	Search by Event Name or ID to check available seats.
6	Overdue Payment Checking	Track customers who reserved tickets but delayed payment beyond a due date.
7	Cancel Booking + Fine Calculation	Allow cancellation with a cancellation fee (e.g., 20 AED per canceled ticket after deadline).
8	Sort Events	Sort events alphabetically by Event Name when displaying.
9	File Handling	Save all booking and customer data to files. Load data at startup.

Goal of the System:

- Build an efficient and simple event ticket management solution that:
- Avoids overbooking or double-booking
- Tracks payments and late cancellations
- Makes event searching and ticket booking smooth
- Keeps accurate digital records for all transactions

Proposed Solution:

1. Assumptions:

1. User must understand that how to enter data, like (XXXX) if this is the format displaying so customer must know that he/she have enter 4-digits.
2. Similarly in the case of date (DDMMYYYY) is interpreted as first two digits are of day, second two are for month and last four digits are for year and there should be no space as shown in format.
3. A maximum of 10 events can be entered.
4. System will only be able to store customer's and event's data in a file.
5. Total seats will 50 for all events, it is hardcoded. However it can be changed by changing is one variable 'totalSeats', in structure 'eventDetails'.
6. Customer can book 1-20 tickets at a time. I've done this to judge program quickly.

2. Code:

```
#include<iostream>
#include<string>
#include<fstream>
#include<iomanip> // to show details in a formatted way
using namespace std;

struct customerDetails{
    string name, paymentStatus;
    int cnic = 0, contact = 0, customerEventId = 0;
};

struct eventDetails{
    string name, venue;
    int id = 0, date = 0, totalSeats = 50;
    float ticketPrice = 0.00;
};

struct tickets{ // for ticket booking details
    eventDetails ed;
    int totalSeats = ed.totalSeats;
    int eventId = 0, availableSeats = ed.totalSeats;
    int toBook = 0, toCancel = 0, bookedTickets = 0;
};



bool findCnic(int targetCnic); // prototype here, because registerCustomer()
will check the uniqueness of CNIC
bool findEventId(int targetId); // prototype here, because registerCustomer()
will match event id which customer is attending
```

```
/*1. Function to register customer*/
void registerCustomer(){
    customerDetails c;
    bool cnicExists = false;
    bool eventIdExists = true;

    cin.ignore();
    cout << "Enter name: ";
    getline(cin, c.name);

    // to ensure cnic uniqueness
    do{
        cout << "Enter CNIC no. (XXXX): ";
        cin >> c.cnic;

        if(findCnic(c.cnic) == true){
            cnicExists = true;
            cout << endl;
            cout << "Please enter unique CNIC." << endl << endl;
        }
        else{
            cnicExists = false;
        }
    }while(cnicExists);

    cout << "Enter contact no. (XXXXXXXXX): ";
    cin >> c.contact;

    // to ensure it matches any event id
    do{
        cout << "Enter ID of event which customer wants to attend: ";
        cin >> c.customerEventId;
        cin.ignore();

        if(findEventId(c.customerEventId) != true){
            eventIdExists = false;
            cout << endl;
            cout << "Invaild event ID" << endl << endl;
        }
        else{
            eventIdExists = true;
        }
    }while(!eventIdExists);
    do{
        cout << endl;
        cout << "Enter payment status (paid/unpaid): ";
        getline(cin, c.paymentStatus);
```

```
        if(c.paymentStatus != "paid" && c.paymentStatus != "unpaid"){
            cout << endl;
            cout << "Invalid payment status!" << endl << endl;
        }

    }while(c.paymentStatus != "paid" && c.paymentStatus != "unpaid");

    //sending data to file
    ofstream fileOut("customerData.txt", ios::app);

    if(!fileOut.is_open()){
        cout << endl;
        cout << "Error in opening file to send customer data." << endl <<
endl;
    }
    else{
        fileOut<<c.cnic<<"|"<<c.name<<"|"<<c.contact<<"|"<<c.customerEventId<<
"|"<<c.paymentStatus<<endl;
        cout << endl;
        cout << "Customer data recorded successfully." << endl << endl;
        fileOut.close();
    }
}

/*2. To show all customers*/
void readCustomers(){
    int readCnic = 0, readContact = 0, readEvent = 0;
    string readName, readPayment, row;

    ifstream fileIn("customerData.txt", ios::in);

    if(!fileIn.is_open()){
        cout << "Error in opening file to read customer data." << endl;
    }


    else{
        int sNo = 1;
        cout << endl;
        cout << "S.No.   CNIC     Name                    Contact No.   Event
ID  Payment Status  "<< endl;
        for(int i = 1; i <= 81; i++) cout << "-";
        cout << endl;

        while(!fileIn.eof()){ //keep extracting lines until the last char came
            getline(fileIn, row);
            if(row.empty()) continue;
```

```
            int pos1 = row.find('|');
            int pos2 = row.find('|', pos1 + 1);
            int pos3 = row.find('|', pos2 + 1);
            int pos4 = row.find('|', pos3 + 1);

            readCnic = stoi(row.substr(0, pos1));
            readName = row.substr(pos1 + 1, (pos2 - pos1) - 1);
            readContact = stoi(row.substr(pos2 + 1, (pos3 - pos2) - 1));
            readEvent = stoi(row.substr(pos3 + 1, (pos4 - pos3) - 1));
            readPayment = row.substr(pos4 + 1);

            cout << left;
            cout<<setw(8)<<sNo<<setw(9)<<readCnic<<setw(24)<<readName<<setw(14
)<<readContact<<setw(10)<<readEvent<<setw(16)<<readPayment<<endl;
        }
        cout << endl;
        fileIn.close();
    }
}

/*3. To find customer with CNIC*/
//it is returning because registerCustomer() will also check the uniqueness...
bool findCnic(int targetCnic){
    int realCnic = 0, realContact = 0, realEvent = 0;
    string realName,realPayment, row;
    bool cnicFound = false;

    ifstream fileIn("customerData.txt", ios::in);

    if(!fileIn.is_open()){
        cout << "Error in opening file to find customer data." << endl;
    }

    else{

        while(!fileIn.eof()){
            getline(fileIn, row);
            if(row.empty()) continue;

            int pos1 = row.find('|');
            int pos2 = row.find('|', pos1 + 1);
            int pos3 = row.find('|', pos2 + 1);
            int pos4 = row.find('|', pos3 + 1);

            realCnic = stoi(row.substr(0, pos1));
            realName = row.substr(pos1 + 1, (pos2 - pos1) - 1);
            realContact = stoi(row.substr(pos2 + 1));
```

```
            realEvent = stoi(row.substr(pos3 + 1, (pos4 - pos3) - 1));
            realPayment = row.substr(pos4 + 1);

            if(realCnic == targetCnic){
              cnicFound = true;
              break;
            }
        }
        fileIn.close();

        if(cnicFound){
            cout << endl;
            cout << "Customer found. Details:" << endl << endl;
            int sNo = 1;
            cout << "S.No.   CNIC     Name                    Contact
No.   Event ID  Payment Status  "<< endl;
            for(int i = 1; i <= 81; i++) cout << "-";
            cout << endl;

            cout << left;
            cout<<setw(8)<<sNo<<setw(9)<<realCnic<<setw(24)<<realName<<setw(14
)<<realContact<<setw(10)<<realEvent<<setw(16)<<realPayment<<endl << endl;
        }
        else{
            cout << endl;
            cout << "CNIC is unique." << endl << endl;
        }
    }
    return cnicFound;
}

/*4. To find payment status*/
void findPaymentStatus(string targetPayment){
    int realCnic = 0, realContact = 0, realEvent = 0;
    string realName,realPayment, row;

    ifstream fileIn("customerData.txt", ios::in);

    if(!fileIn.is_open()){
        cout << "Error in opening file to find customer(s) with
"<<targetPayment<<" payment." << endl;
    }


    else{
        cout << endl;
        cout << "Customer(s) with "<<targetPayment<<" payment:" << endl <<
endl;
```

```
        for(int i = 1; i <= 81; i++) cout << "-";
        cout << endl;
        cout << "S.No.   CNIC     Name                    Contact No.   Event
ID  Payment Status  "<< endl;

        int sNo = 1;
        while(!fileIn.eof()){
            getline(fileIn, row);
            if(row.empty()) continue;

            int pos1 = row.find('|');
            int pos2 = row.find('|', pos1 + 1);
            int pos3 = row.find('|', pos2 + 1);
            int pos4 = row.find('|', pos3 + 1);

            realCnic = stoi(row.substr(0, pos1));
            realName = row.substr(pos1 + 1, (pos2 - pos1) - 1);
            realContact = stoi(row.substr(pos2 + 1));
            realEvent = stoi(row.substr(pos3 + 1, (pos4 - pos3) - 1));
            realPayment = row.substr(pos4 + 1);


            if(realPayment == targetPayment){
                cout << left;
                cout<<setw(8)<<sNo<<setw(9)<<realCnic<<setw(24)<<realName<<set
w(14)<<realContact<<setw(10)<<realEvent<<setw(16)<<realPayment<<endl;

                sNo++;
            }
        }
        cout << endl;
        fileIn.close();
    }
}


/*5. To register event*/
void registerEvent(){
    eventDetails e;
    bool uniqueEventId = true;

    //to ensure event id uniqueness
    do{
        cout << "Enter event ID (XXXX): ";
        cin >> e.id;

        if(findEventId(e.id) == true){
            uniqueEventId = false;
            cout << endl;
```

```
            cout << "Please enter unique event ID." << endl << endl;
        }
        else{
            uniqueEventId = true;
        }

    }while(!uniqueEventId);
    cin.ignore();

    cout << "Enter event name: ";
    getline(cin, e.name);
    cout << "Enter event date (DDMMYYYY): ";
    cin >> e.date;
    cin.ignore();
    cout << "Enter venue: ";
    getline(cin, e.venue);
    cout << "Enter tickect price(RS): ";
    cin >> e.ticketPrice;
    cin.ignore();

    ofstream fileOut("eventData.txt", ios::app);
    if(!fileOut.is_open()){
        cout << "Error in opening file to send event data." << endl;
    }
    else{
        fileOut<<e.id<<"|"<<e.name<<"|"<<e.date<<"|"<<e.venue<<"|"<<e.ticketPr
ice<<"|"<<e.totalSeats<<endl;
        cout << endl;
        cout << "Event data recorded successfully." << endl << endl;
        fileOut.close();
    }
}

/*6. To show all events (sorted)*/
void showSortedEvents(){
    int readId = 0, readTotalSeats = 0, readDate = 0;
    float readTicketPrice = 0.00;
    string readName, readVenue, row;

    int max = 10; // as we know only 10 events can be registered
    int id[max], totalSeats[max], date[max];
    float ticketPrice[max];
    string name[max], venue[max];
    int count = 0; // it will store how many indexes it have extracted
    ifstream fileIn("eventData.txt", ios::in);
    if(!fileIn.is_open()){
        cout << "Error in opening file to show sorted events." << endl;
    }
```

```
    else{
        while(!fileIn.eof()){
            getline(fileIn, row);
            if(row.empty()) continue;

            int pos1 = row.find('|');
            int pos2 = row.find('|', pos1 + 1);
            int pos3 = row.find('|', pos2 + 1);
            int pos4 = row.find('|', pos3 + 1);
            int pos5 = row.find('|', pos4 + 1);

            readId = stoi(row.substr(0, pos1));
            readName = row.substr(pos1 + 1, (pos2 - pos1) - 1);
            readDate = stoi(row.substr(pos2 + 1, (pos3 - pos2) - 1));
            readVenue = row.substr(pos3 + 1, (pos4 - pos3) - 1);
            readTicketPrice = stof(row.substr(pos4 + 1, (pos5 - pos4) - 1));
            readTotalSeats = stoi(row.substr(pos5 + 1));

            id[count] = readId;
            name[count] = readName;
            date[count] = readDate;
            venue[count] = readVenue;
            ticketPrice[count] = readTicketPrice;
            totalSeats[count] = readTotalSeats;

            count++;
        }
        fileIn.close();

        //now sorting array based on name
        for(int i = 0; i < count; i++){
            for(int j = 0; j < count-i-1; j++){

                if(name[j] > name[j+1]){

                    //swaping
                    int tempId = id[j];
                    id[j] = id[j+1];
                    id[j+1] = tempId;

                    string tempName = name[j];
                    name[j] = name[j+1];
                    name[j+1] = tempName;

                    int tempDate = date[j];
                    date[j] = date[j+1];
                    date[j+1] = tempDate;
```

```
                    string tempVenue = venue[j];
                    venue[j] = venue[j+1];
                    venue[j+1] = tempVenue;

                    float tempTicketPrice = ticketPrice[j];
                    ticketPrice[j] = ticketPrice[j+1];
                    ticketPrice[j+1] = tempTicketPrice;

                    int tempTotalSeats = totalSeats[j];
                    totalSeats[j] = totalSeats[j+1];
                    totalSeats[j+1] = tempTotalSeats;

                }
            }
        }

        //displaing sorted data
        cout << endl;
        cout << "Details of all events:" << endl << endl;
        cout << "S.No.   Event
ID  Name                    Date      Venue                    Ticket Price
(RS)  Total Seats" << endl;

        for(int i = 1; i <= 111; i++) cout << "-";
        cout << endl;

        cout << left;
        for(int i = 0, sNo = 1; i < count; i++, sNo++){
            cout<<setw(8)<<sNo<<setw(10)<<id[i]<<setw(24)<<name[i]<<setw(10)<<
date[i]<<setw(25)<<venue[i]<<setw(19)<<ticketPrice[i]<<setw(15)<<totalSeats[i]
<<endl;
        }
        cout << endl;
    }
}

/*7. To find targeted event ID*/
bool findEventId(int targetId){
    int realId = 0, realTotalSeats = 0, realDate = 0;
    float realTicketPrice = 0.00;
    string realName, realVenue, row;
    bool eventIdFound = false;

    ifstream fileIn("eventData.txt", ios::in);
    if(!fileIn.is_open()){
        cout << "Error in opening file to find tageted event id." << endl;
    }
    else{
```

```
        while(!fileIn.eof()){
            getline(fileIn, row);
            if(row.empty()) continue;

            int pos1 = row.find('|');
            int pos2 = row.find('|', pos1 + 1);
            int pos3 = row.find('|', pos2 + 1);
            int pos4 = row.find('|', pos3 + 1);
            int pos5 = row.find('|', pos4 + 1);

            realId = stoi(row.substr(0, pos1));
            realName = row.substr(pos1 + 1, (pos2 - pos1) - 1);
            realDate = stoi(row.substr(pos2 + 1, (pos3 - pos2) - 1));
            realVenue = row.substr(pos3 + 1, (pos4 - pos3) - 1);
            realTicketPrice = stof(row.substr(pos4 + 1, (pos5 - pos4) - 1));
            realTotalSeats = stoi(row.substr(pos5 + 1));

            if(targetId == realId){
                eventIdFound = true;
                break;
            }
        }
        if(eventIdFound){
            cout << endl;
            cout << "Event found. Details:" << endl << endl;
            int sNo = 1;

            cout << "S.No.   Event
ID  Name                    Date      Venue                    Ticket Price
(RS)  Total Seats" << endl;
            for(int i = 1; i <= 111; i++) cout << "-";
            cout << endl;
            cout << left;
            cout<<setw(8)<<sNo<<setw(10)<<realId<<setw(24)<<realName<<setw(10)
<<realDate<<setw(25)<<realVenue<<setw(19)<<realTicketPrice<<setw(15)<<realTota
lSeats<<endl;
        }
        else{
            cout << endl;
            cout << "Event ID is unique." << endl << endl;
        }
    }
    return eventIdFound;
}
/*8. To find event date*/
int findEventdate(int targetId){
    string realName, realVenue, row;
    int realId = 0, realDate = 0, realTotalSeats = 0;
```

```
    float realTicketPrice = 0.00;
    bool targetIdfound = false;

    ifstream fileIn("eventData.txt", ios::in);
    if(!fileIn.is_open()){
        cout << "Error in opening file to find tageted event id." << endl;
    }
    else{
        while(!fileIn.eof()){
            getline(fileIn, row);
            if(row.empty()) continue;

            int pos1 = row.find('|');
            int pos2 = row.find('|', pos1 + 1);
            int pos3 = row.find('|', pos2 + 1);
            int pos4 = row.find('|', pos3 + 1);
            int pos5 = row.find('|', pos4 + 1);

            realId = stoi(row.substr(0, pos1));
            realName = row.substr(pos1 + 1, (pos2 - pos1) - 1);
            realDate = stoi(row.substr(pos2 + 1, (pos3 - pos2) - 1));
            realVenue = row.substr(pos3 + 1, (pos4 - pos3) - 1);
            realTicketPrice = stof(row.substr(pos4 + 1, (pos5 - pos4) - 1));
            realTotalSeats = stoi(row.substr(pos5 + 1));

            if(targetId == realId){
                targetIdfound = true;
                break;
            }
        }
        if(targetIdfound){
            cout << endl;
            cout << "Event found. Details:" << endl << endl;
            int sNo = 1;

            cout << "S.No.   Event
ID  Name                    Date      Venue                    Ticket Price
(RS)  Total Seats" << endl;
            for(int i = 1; i <= 111; i++) cout << "-";
            cout << endl;



            cout << left;
            cout<<setw(8)<<sNo<<setw(10)<<realId<<setw(24)<<realName<<setw(10)
<<realDate<<setw(25)<<realVenue<<setw(19)<<realTicketPrice<<setw(15)<<realTota
lSeats<<endl;
```

```
        }

    }
    return realDate;
}




int main(){
    tickets bookTicket[10];
    int eventCount = 0; // to make sure events do not exced to 10
    int ticketBookCount = 0; //we will assigne an index number if a ticket is
booked for an event for the first time
    bool firstEventAdded = false; // we will not allow to regiter customer
until any event is registered
    int choice = 0;

    cout << "-----------------------------------------------------" <<endl;
    cout << "***BUCIS Event Management and Ticket Booking System***" << endl;
    cout << "-----------------------------------------------------" <<endl;
    do{
        cout << "\t---MENU---" << endl << endl;
        cout << "1. To register customer." << endl;
        cout << "2. To add event (upto 10 events allowed)." << endl;
        cout << "3. To see details of all customers." << endl;
        cout << "4. To search customer (with CNIC)." << endl;
        cout << "5. To search event (with ID)." << endl;
        cout << "6. To check payment status (paid/unpaid)." << endl;
        cout << "7. To see all events (sorted)." << endl;
        cout << "8. To book tickets for an event." << endl;
        cout << "9. To cancel tickets." << endl;
        cout << "0. To exit." << endl;
        cout << "Enter choice: ";
        cin >> choice;

        if(choice == 1){
            if(!firstEventAdded){ // making sure that at least one event is
registered
                cout << endl;
                cout << "Please add event first." << endl << endl;
            }
            else{
                cout << endl;
                registerCustomer();
                cout << endl;
```

```
        }
    }

    else if(choice == 2){
        if(eventCount > 10){
            cout << endl;
            cout << "Event registration limit reached!" << endl;
        }

        else{
            cout << endl;
            registerEvent();
            cout << endl;
            eventCount++;
            firstEventAdded = true;
        }
    }

    else if(choice == 3){
        cout << endl;
        readCustomers();
        cout << endl;
    }

    else if(choice == 4){
        int targetCnic = 0;
        cout << endl;
        cout << "Enter CNIC of customer (XXXX): ";
        cin >> targetCnic;
        findCnic(targetCnic);
        cout << endl;
    }

    else if(choice == 5){
        int targetId = 0;
        cout << endl;
        cout << "Enter event ID (XXXX): ";
        cin >> targetId;
        findEventId(targetId);
        cout << endl;
    }



    else if(choice == 6){
        cout << endl;
        string requiredStatus;
        cin.ignore();
```

```
            do{
                cout << "Enter payment status (paid/unpaid): ";
                getline(cin, requiredStatus);

                if(requiredStatus != "paid" && requiredStatus != "unpaid"){
                    cout << endl;
                    cout << "Invalid payment status!" << endl << endl;
                }
            }while(requiredStatus != "paid" && requiredStatus != "unpaid");
            findPaymentStatus(requiredStatus);
            cout << endl;
        }
        else if(choice == 7){
            cout << endl;
            showSortedEvents();
            cout << endl;
        }

        /*if user wants to book ticket*/
        else if(choice == 8){
            if(!firstEventAdded){ // making sure that any event is added
                cout << endl;
                cout << "Please add event first." << endl << endl;
            }
            else{
                int userEventId = 0;
                bool eventIdExists = true;
                int index = 0;
                do{
                    cout << endl;
                    cout<<"Enter ID of event for which you want to book
ticket:";
                    cin >> userEventId;
                    if(findEventId(userEventId) != true){ // first checking if
the user have eneter correct event id
                        eventIdExists = false;
                        cout << endl;
                        cout << "Invalid event Id." << endl << endl;
                    }



                    else{
                        eventIdExists = true;
                        bool found = false;;
```

```
                        for(int i = 0; i < 10; i++){ // here checking is there
ahny ticket booked for this event id
                            if(userEventId == bookTicket[i].eventId){
                                found = true;
                                index = i;
                                break;
                            }
                        }
                        if(!found){
                            bookTicket[ticketBookCount].eventId = userEventId;
                            index = ticketBookCount; // if not booked, so the
index will be based on ticket book count
                            ticketBookCount++;
                        }
                    }
                }while(!eventIdExists);
                do{
                    cout << endl;
                    cout << "Enter the number of tickets you want to book (1-
20): ";
                    cin >> bookTicket[index].toBook;
                    if(bookTicket[index].toBook <= 0 ||
bookTicket[index].toBook > 20){
                        cout << endl;
                        cout << "Invalid number of tickets to book." << endl
<< endl;
                        break; //if no. of tickets are invalid, so no action
should be done about booking of ticket
                    }
                    if(bookTicket[index].toBook >
bookTicket[index].availableSeats){
                        cout << "No enough seats are available." << endl <<
endl;
                    }
                    else{
                        bookTicket[index].availableSeats -=
bookTicket[index].toBook;
                        bookTicket[index].bookedTickets +=
bookTicket[index].toBook;
                        cout << endl;
                        cout << "Tickets booked successfully." << endl <<
endl;
                    }
                }while(bookTicket[index].toBook <= 0 ||
bookTicket[index].toBook > 20);
                cout << endl;
                cout << "All Booking Details" << endl << endl;
```

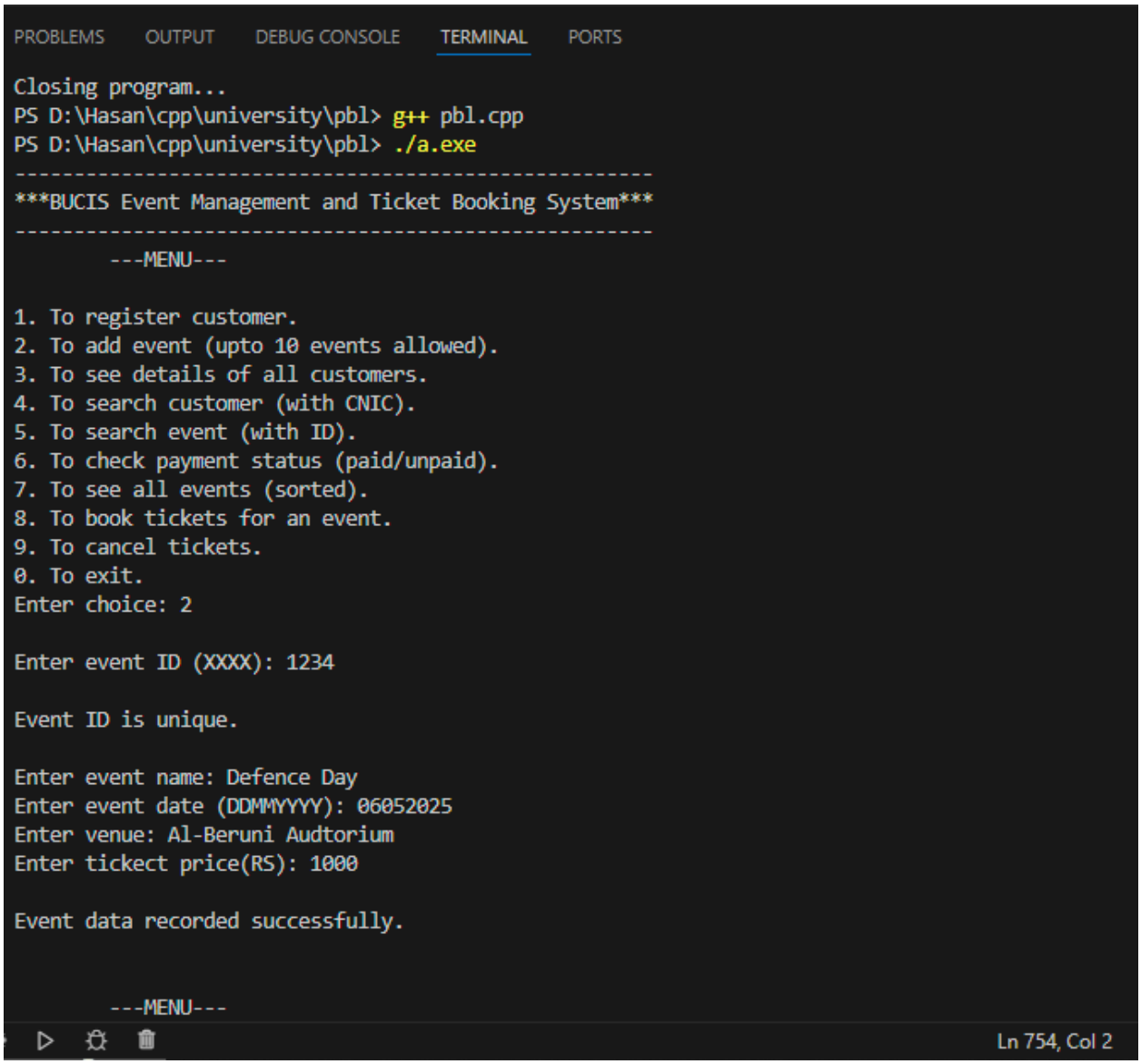
```
                cout << "S.No.   Event ID    Total Seats    Booked
Seats   Available Seats  " << endl;
                for(int i = 0; i < 68; i++) cout << "-";
                cout << endl;

                cout << left;
                for(int i = 0, sNo = 1; i < ticketBookCount; i++, sNo++){
                    cout<<setw(8)<<sNo<<setw(12)<<bookTicket[i].eventId<<setw(
15)<<bookTicket[i].totalSeats<<setw(16)<<bookTicket[i].bookedTickets<<setw(17)
<<bookTicket[i].availableSeats<<endl;
                }
            }

        }

        /*if user wants to cancel ticket*/
        else if(choice == 9){
            if(!firstEventAdded){ // making sure that any event is added
                cout << endl;
                cout << "Please add event first." << endl << endl;
            }
            else{
                int toCancel = 0;
                int ticketCanceldate = 0;
                int userEventId = 0;
                bool eventIdExists = true;
                bool ticketsCancelled = false;
                int index = 0;
                do{
                    cout << endl;
                    cout << "Enter ID of event for which you want to cancel
ticket: ";
                    cin >> userEventId;

                    if(findEventId(userEventId) != true){
                        eventIdExists = false;
                        cout << endl;
                        cout << "Invalid event Id." << endl << endl;
                    }
                    else{
                        eventIdExists = true;
                        bool found = false;
                        for(int i = 0; i < 10; i++){
                            if(userEventId == bookTicket[i].eventId){ //making
sure that a ticket is booked for this event id
                                found = true;
                                index = i;
                                break;
```

```
                        }
                    }
                    if(!found){
                        cout << endl;
                        cout << "No ticket is booked for this event." <<
endl << endl;
                    }

                    else{
                        cout << endl;
                        cout << "Enter no. of tickets you want to cancel:
";
                        cin >> toCancel;

                        if(toCancel > bookTicket[index].bookedTickets){
                            cout << endl;
                            cout << "No enough seats are booked." << endl
<< endl;
                        }
                        else{
                            bookTicket[index].toCancel = toCancel;
                            bookTicket[index].availableSeats +=
bookTicket[index].toCancel;
                            bookTicket[index].bookedTickets -=
bookTicket[index].toCancel;
                            cout << endl;
                            cout << "Tickets cancelled successfully!" <<
endl << endl;
                            ticketsCancelled = true;
                        }
                    }
                }

            }while(!eventIdExists);

            //now fine calculation
            if(ticketsCancelled){
                cout << "Enter date at the time of cancelling tickets
(DDMMYYYY): ";
                cin >> ticketCanceldate;

                if(ticketCanceldate > findEventdate(userEventId)){
                    cout << endl;
                    cout << "You are cancelling ticket after event date,
fine will be implemented (RS 20/ticket)."<< endl;
                    int fine = bookTicket[index].toCancel*20;
                    cout << "Fine is: " <<fine<< endl << endl;
                }
```

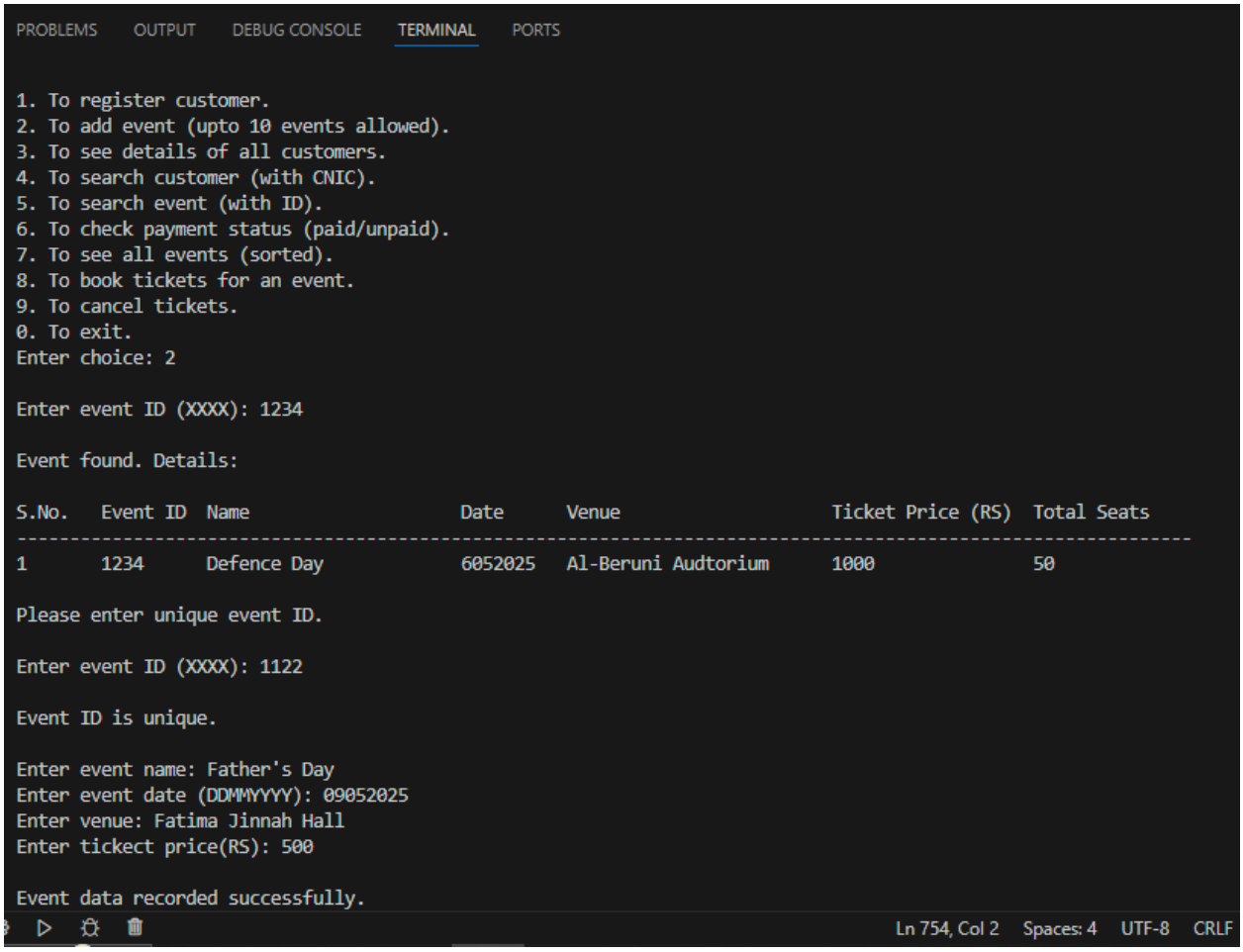
```
                    cout << endl;
                    cout << "All Booking Details" << endl;
                    cout << "S.No.   Event ID    Total Seats    Booked
Seats   Available Seats  " << endl;
                    for(int i = 0; i < 68; i++) cout << "-";
                    cout << endl;

                    cout << left;
                    for(int i = 0, sNo = 1; i < ticketBookCount; i++, sNo++){
                        cout<<setw(8)<<sNo<<setw(12)<<bookTicket[i].eventId<<s
etw(15)<<bookTicket[i].totalSeats<<setw(16)<<bookTicket[i].bookedTickets<<setw
(17)<<bookTicket[i].availableSeats<<endl;
                    }
                }
            }

        }

        else if(choice == 0){
            cout << endl;
            cout << "Closing program..." << endl;
        }
        else{
            cout << endl;
            cout << "Invalid choice!" << endl << endl;
        }

    }while(choice != 0);
}
```

3. Output:

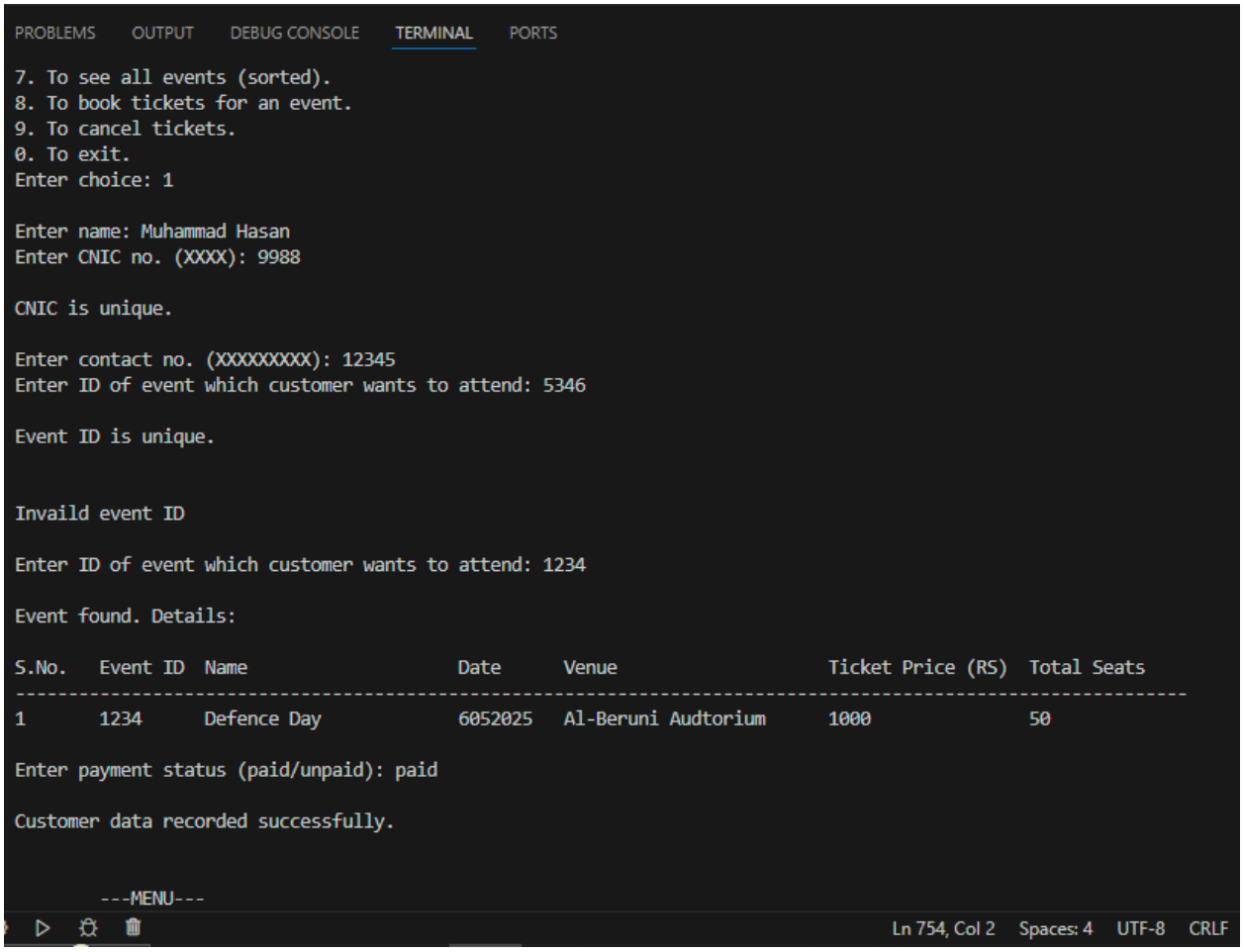
-Adding event.



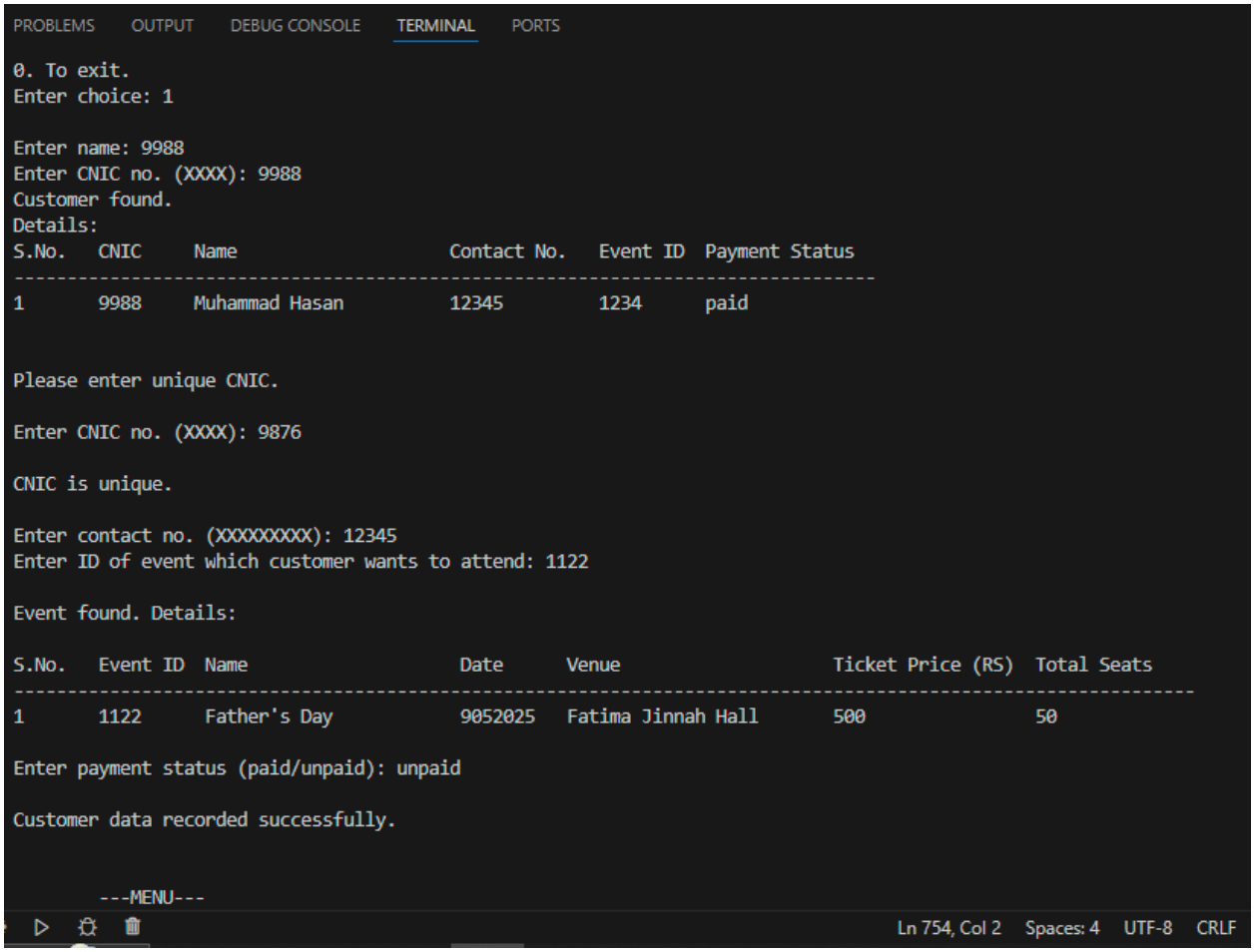
-Ensuring the uniqueness of Event ID while adding new event.



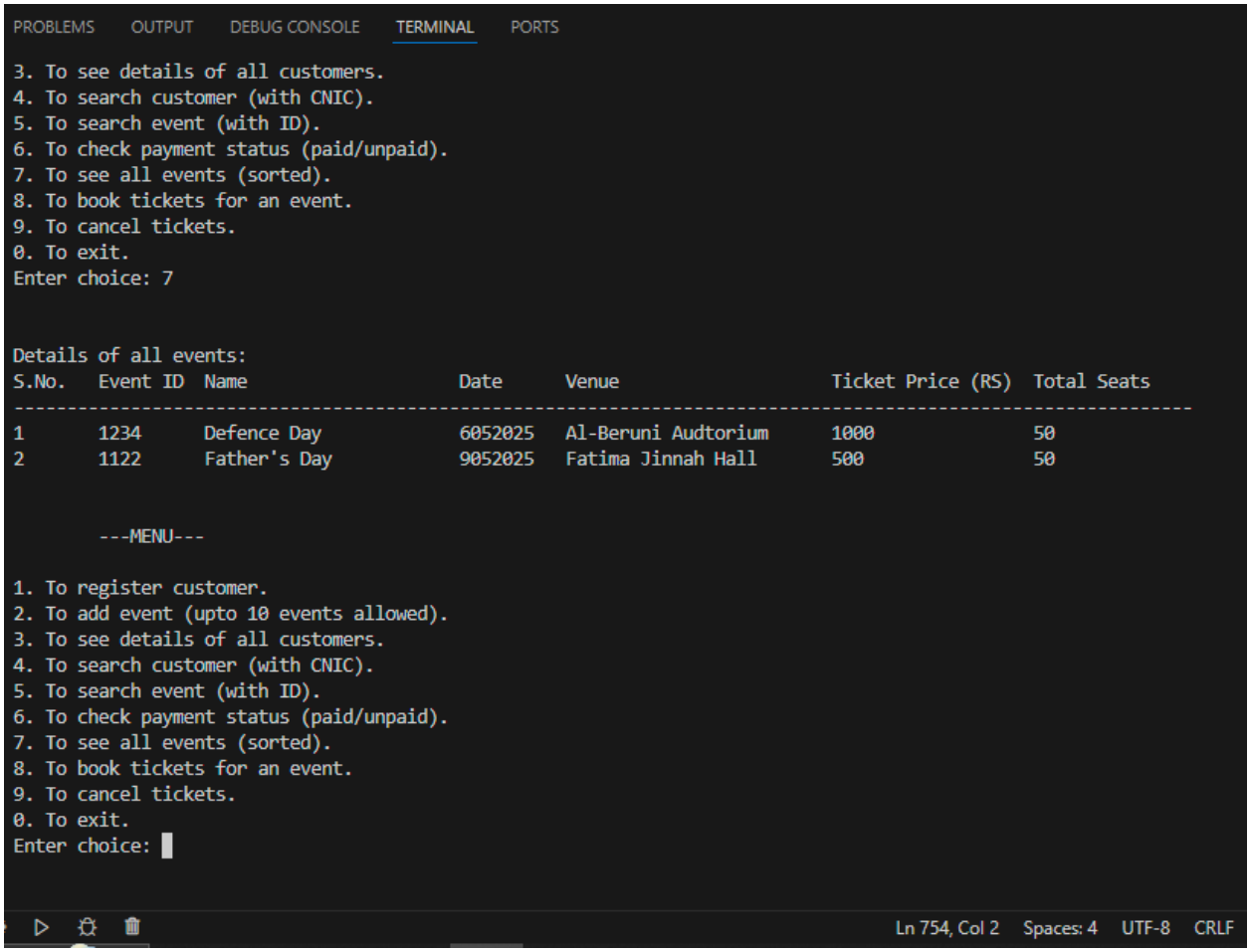
-Registering Customer.



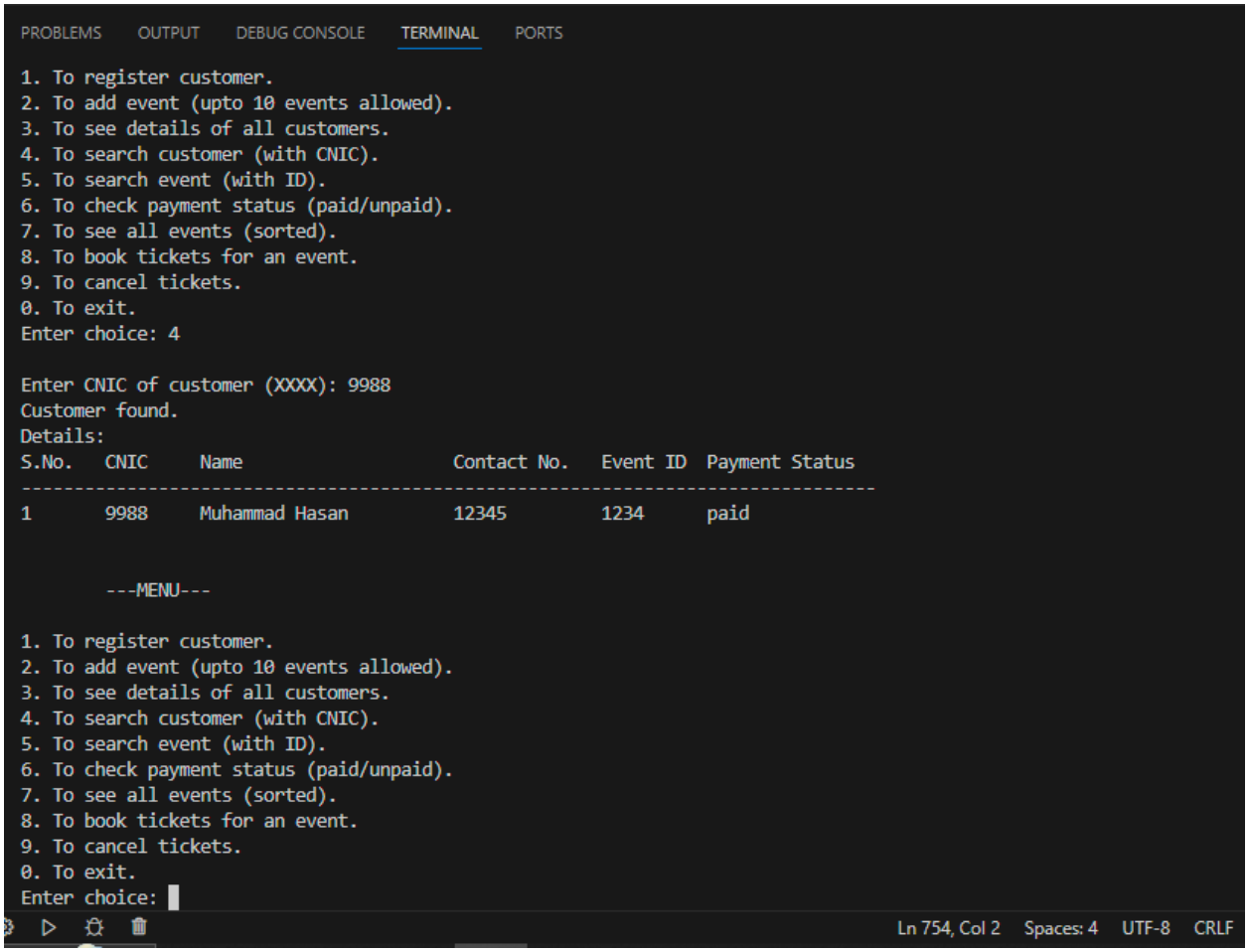
-Ensuring the uniqueness of CNIC while adding new customer.



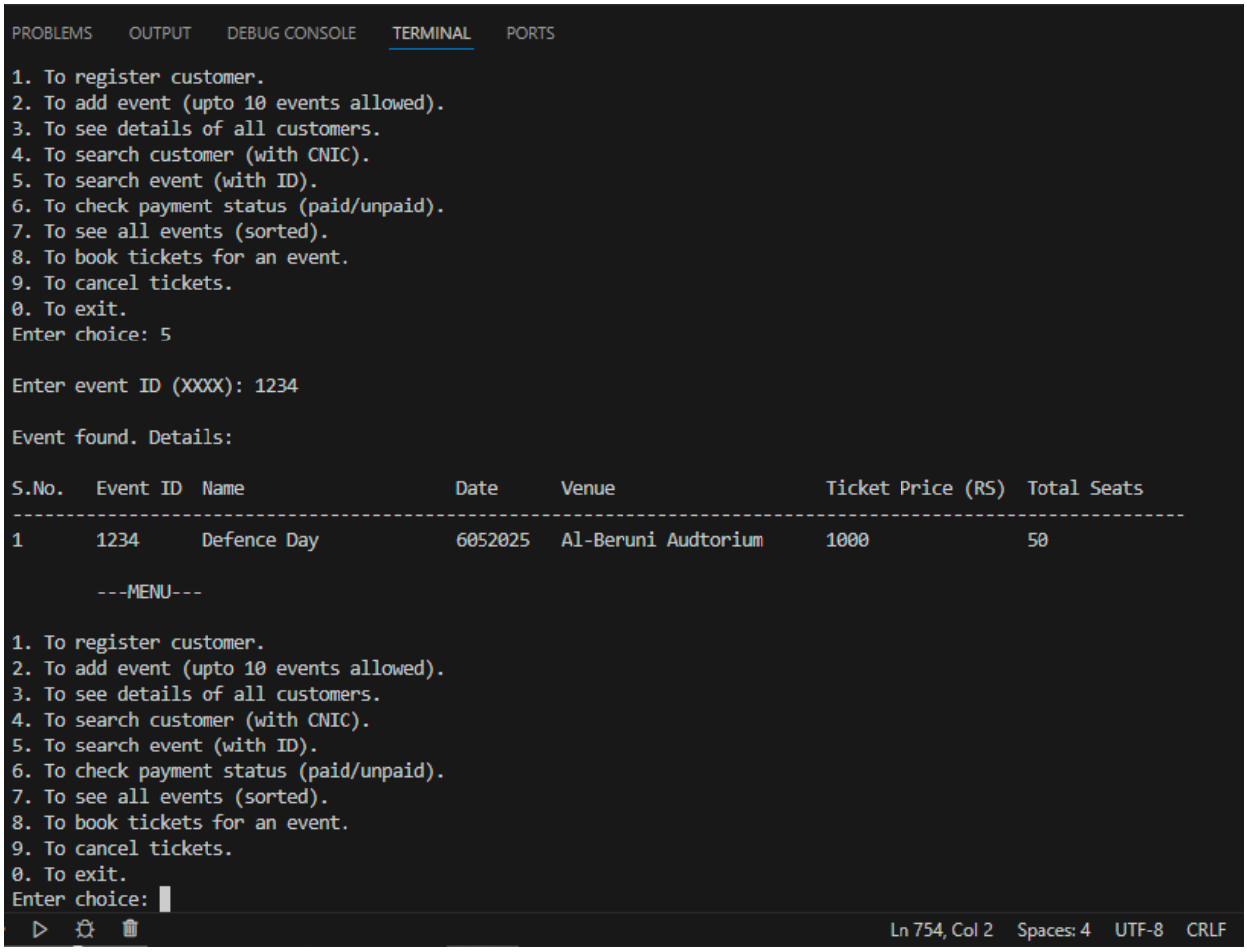
-Seeing all events.



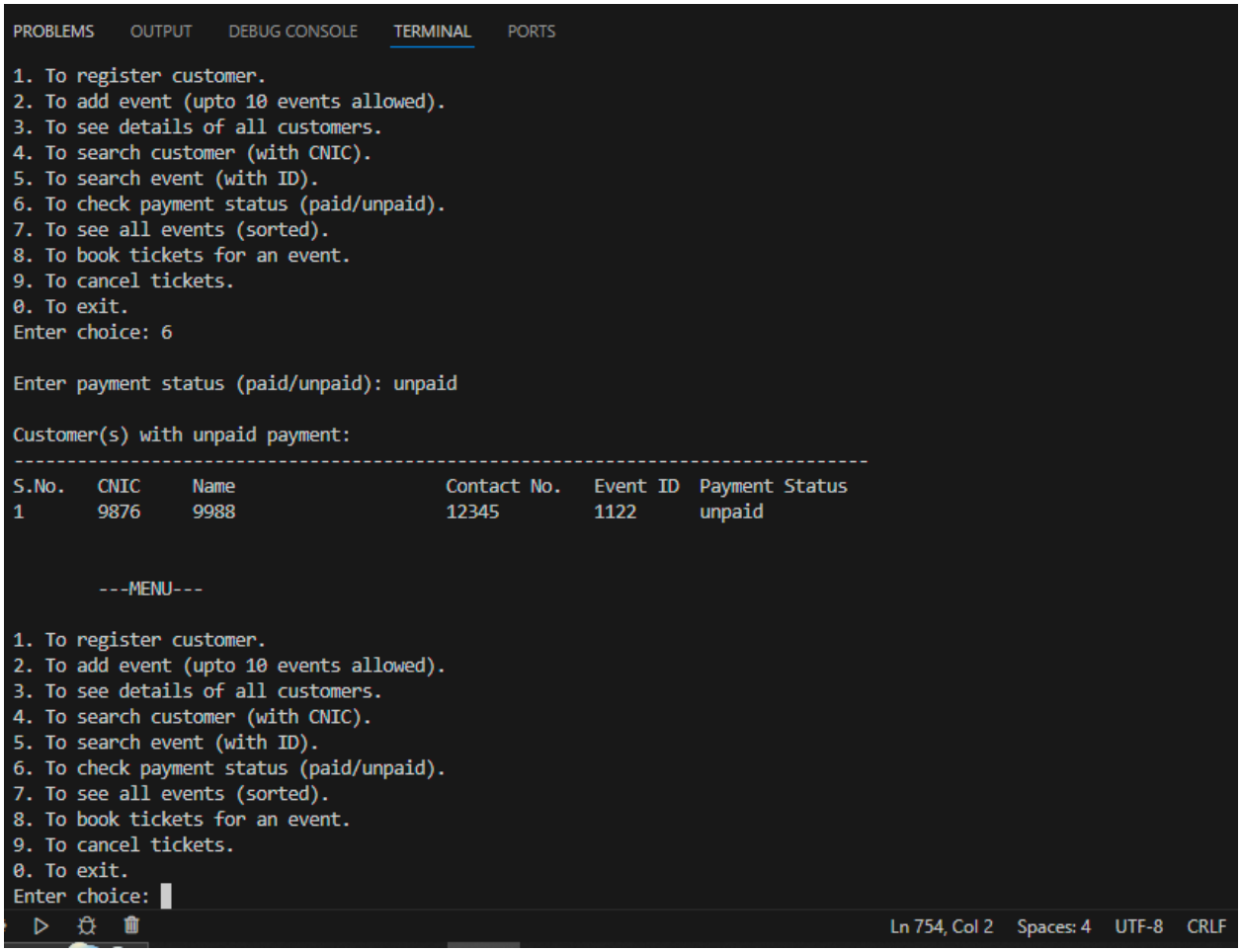
-Searching customer.



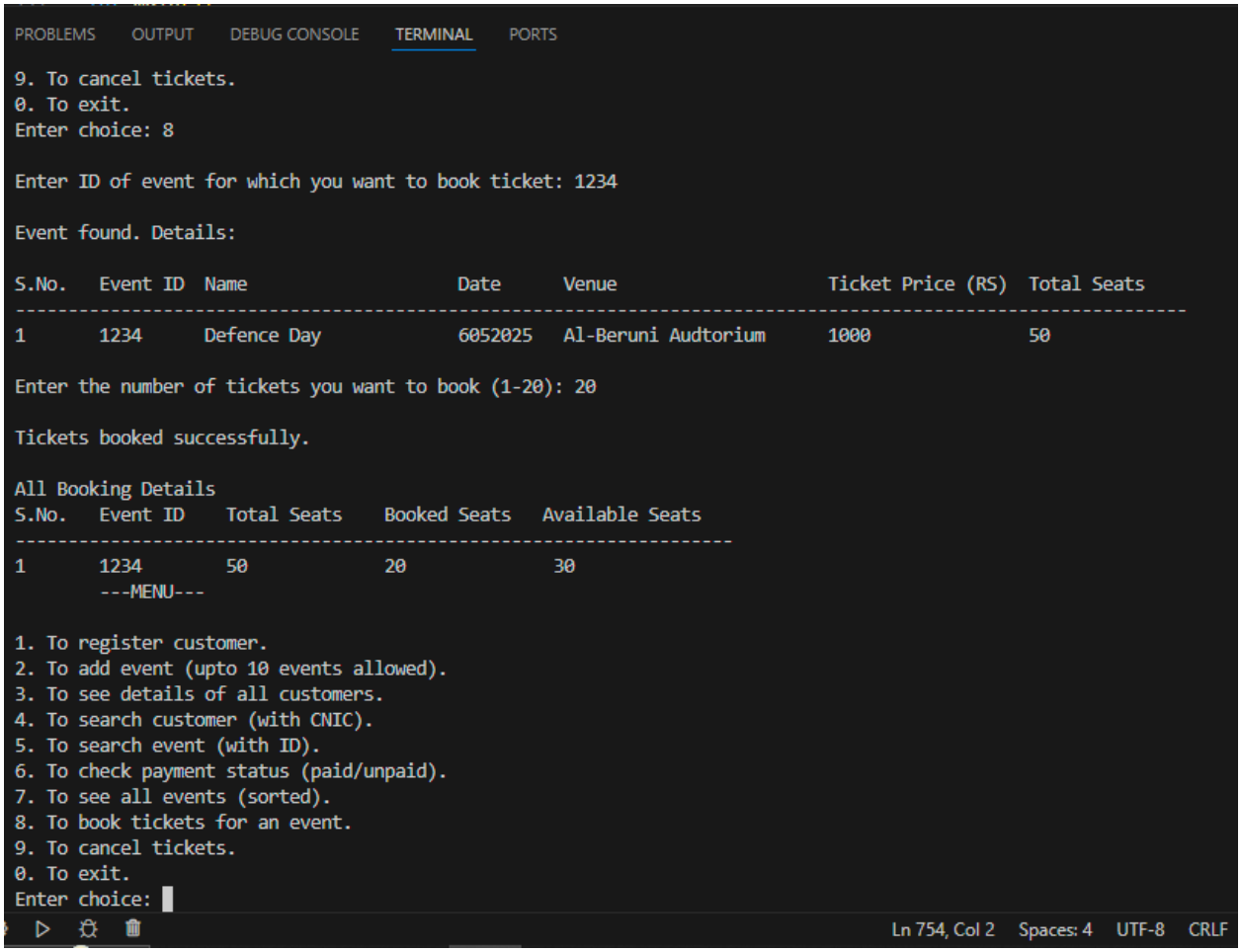
-Searching event with ID.



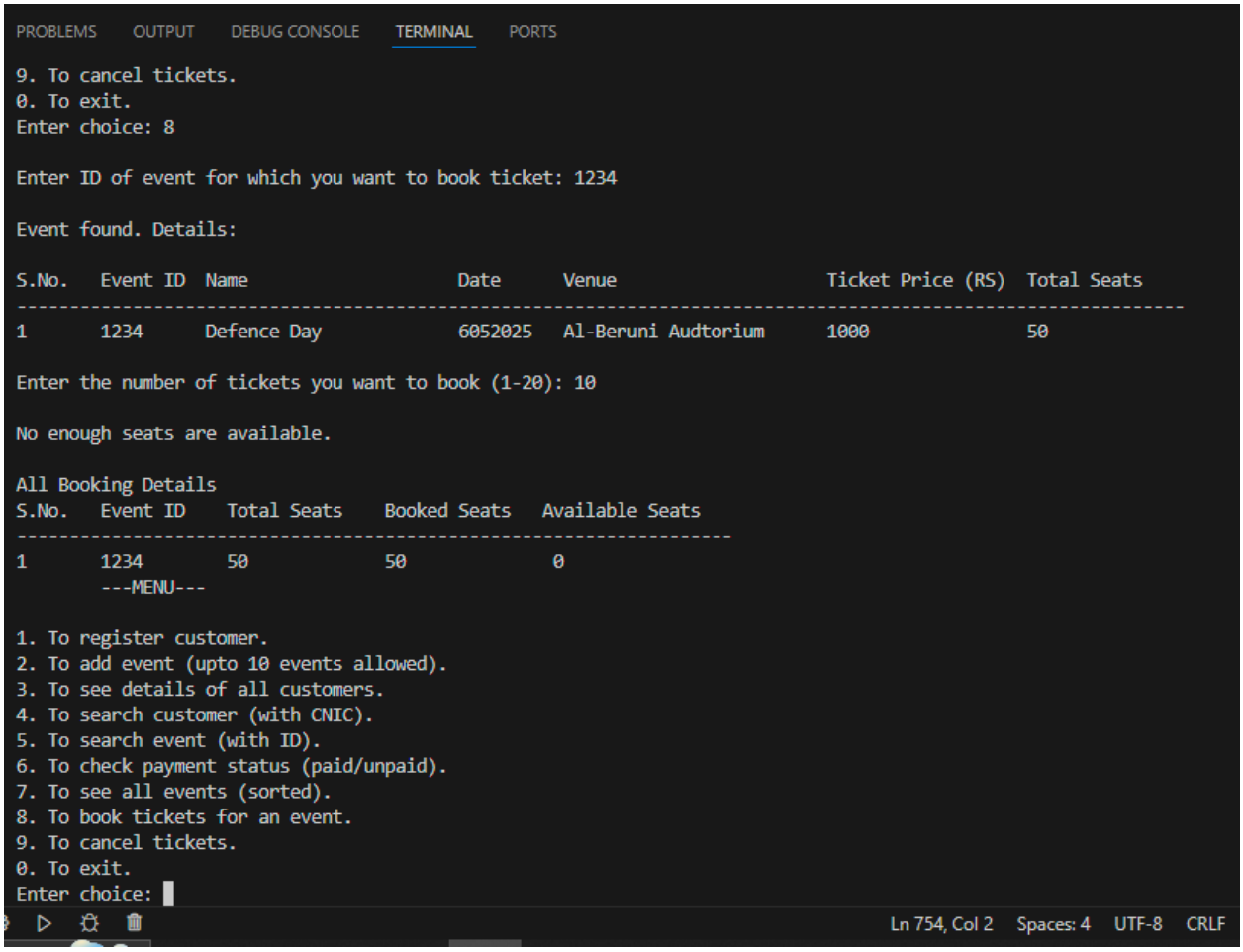
-Searching with payment status.



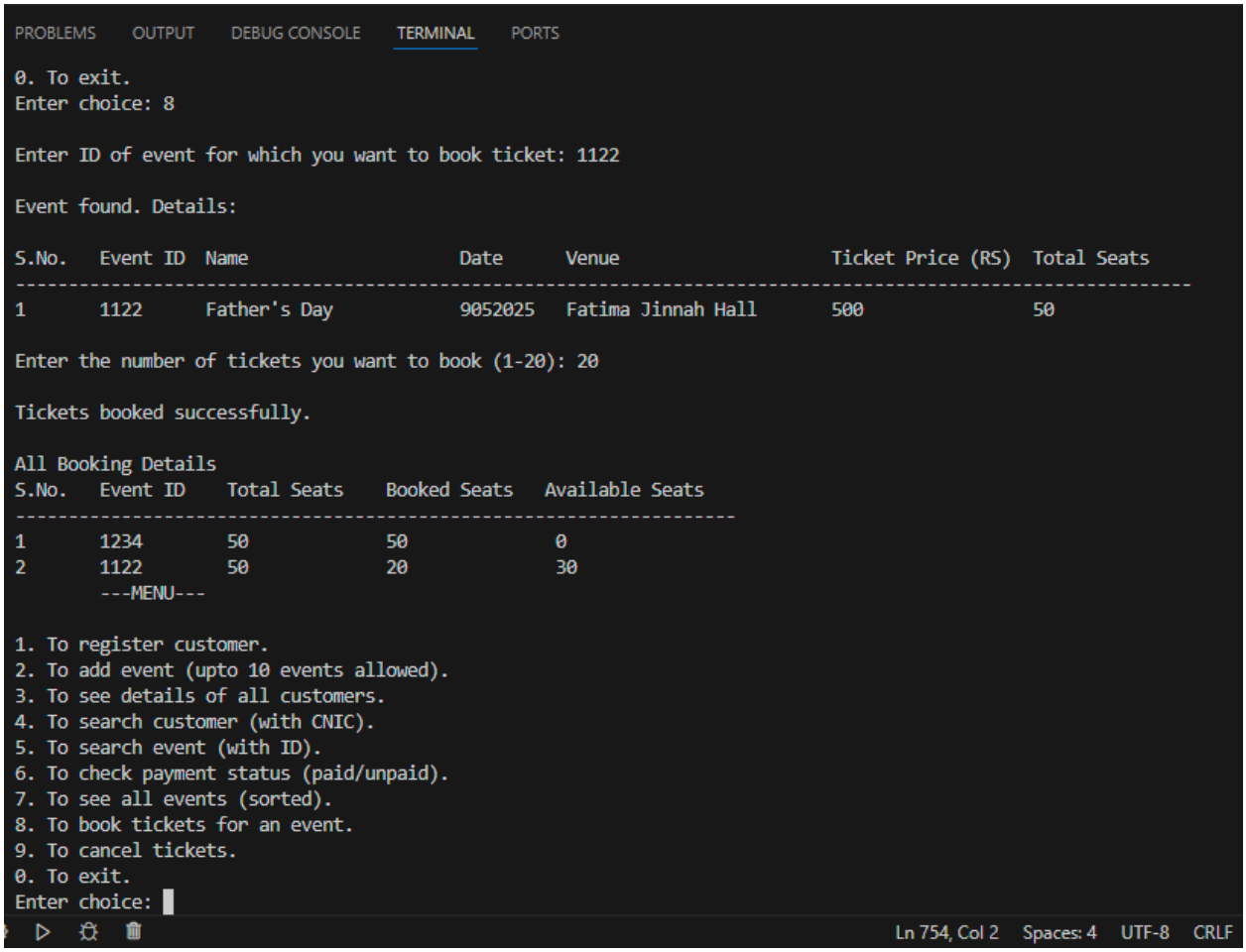
-Booking tickets.



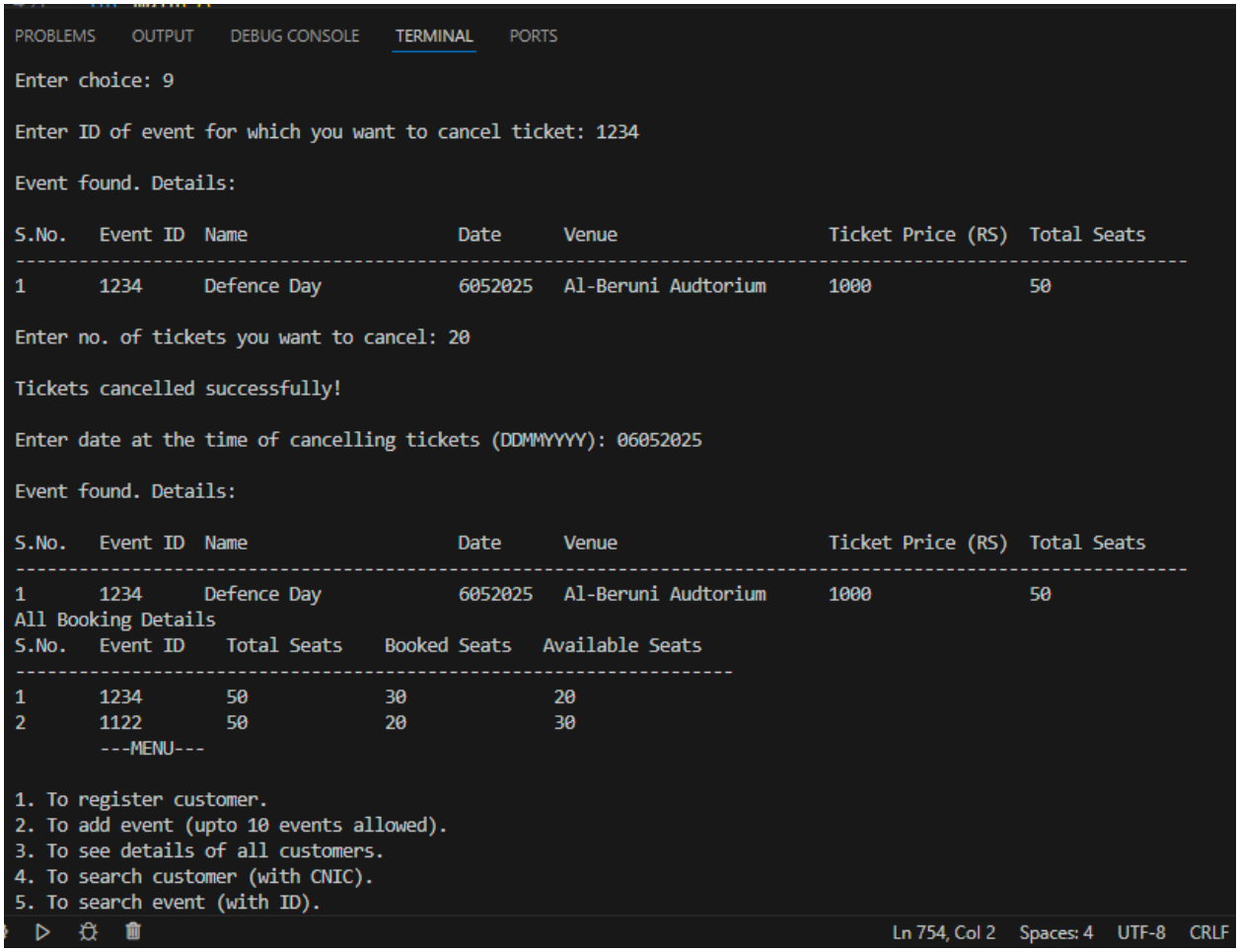
-Preventing to book ticket if limit reached.



-Booking ticket for another event.



-Cancelling ticket.



-Implementing fine after due date.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

9. To cancel tickets.
0. To exit.
Enter choice: 9

Enter ID of event for which you want to cancel ticket: 1122

Event found. Details:

S.No.   Event ID  Name                Date      Venue                Ticket Price (RS)  Total Seats
-------------------------------------------------------------------------------------------------
1       1122      Father's Day        9052025   Fatima Jinnah Hall   500                50

Enter no. of tickets you want to cancel: 10

Tickets cancelled successfully!

Enter date at the time of cancelling tickets (DDMMYYYY): 10052025

Event found. Details:

S.No.   Event ID  Name                Date      Venue                Ticket Price (RS)  Total Seats
-------------------------------------------------------------------------------------------------
1       1122      Father's Day        9052025   Fatima Jinnah Hall   500                50

You are cancelling ticket after event date, fine will be implemented (RS 20/ticket).
Fine is: 200

All Booking Details
S.No.   Event ID    Total Seats    Booked Seats   Available Seats
-----------------------------------------------------------------
1       1234        50             30             20
2       1122        50             10             40
        ---MENU---
```

Ln 754, Col 2 Spaces: 4 UTF-8 CRLF

-Data in files.

pbl.cpp | customerData.txt | eventData.txt

customerData.txt

```
1   9988|Muhammad Hasan|12345|1234|paid
2   9876|9988|12345|1122|unpaid
3
```

pbl.cpp | customerData.txt | eventData.txt

eventData.txt

```
1   1234|Defence Day|6052025|Al-Beruni Audtorium|1000|50
2   1122|Father's Day|9052025|Fatima Jinnah Hall|500|50
3
```