PROJECT REPORT

Computer Programming Lab (CSL-113)

Smart Ticketing System – Railway Based Ticketing System

BS (IT) – 1A

Group Members

Name	Enrollment
1. MUHAMMAD HASAN	02-135251-040
2. MUHAMMAD ARHAM	02-135251-016
3. MUHAMMAD MURTAZA KHAN	02-135251-018

Submitted to:

Sir Muhammad Bilal

BAHRIA UNIVERSITY KARACHI CAMPUS

Department of Computer Science

ABSTRACT

The project **Smart Ticketing System** is developed by using C++ that allows users to book and cancel railway tickets, while admins can manage train data and customer records. It uses **file handling** to store and retrieve information, **input validation** to prevent errors, and **structured programming techniques** to ensure clean and modular code. The system provides a basic but functional simulation of a railway reservation interface.

INTRODUCTION:

The Smart Ticketing System is designed to provide a user-friendly and efficient way to manage railway ticket reservations. It was created as a part of our coursework, covering core programming concepts such as **loops, conditions, arrays, file handling, functions, and structures**.

PROBLEM STATEMENT:

Manual ticketing is time-consuming and error-prone. It lacks data storage, validation, and flexibility. This project aims to solve these problems by creating a console-based C++ application that simulates a smart ticketing environment where users can book, cancel, and view train and customer records reliably.

METHODOLOGY:

The system is built using **C++** with the following components:

- **Structures** to manage train and customer data.
- **Functions** for modularity and code reuse.
- **File handling** to store and update train and customer data persistently.
- **Validation** using `cin.fail()` for robust input handling.
- **Loops & conditions** for user interaction and logic control.
- **Menu-driven interface** for both users and admins.

PROJECT SCOPE:

- Register multiple trains with all relevant details
- Book tickets using CNIC
- Cancel tickets and update seat availability
- Admin features (password protected)
- View all customers
- Search customer by CNIC
- Fully file-based, no database or GUI required
- Designed for small-to-medium scale train management simulations

CODE:

```
#include<iostream>
#include<string>
#include<fstream>
#include<vector>
#include<iomanip>
using namespace std;

struct train{
    int id = 0, depTime = 0, arrivalTime = 0, depDate = 0, totalSeats = 0,
availableSeats = 0;
    string name, fromTo;
    float ticketPrice = 0.00;
};

struct customer{
    int cnic = 0, trainId = 0, reservedSeats = 0;
    string name, paymentStatus;
};

// some prototypes
bool searchTrainId(int targetId);
bool searchCnic(int targetCnic);
void showSummary(int cnic, string name, int trainId, string trainName, string
fromTo, int depTime, int arrivalTime, int depDate, int reservedSeats, float
payableAmount, string status);
vector<string> split(string row, char delimiter);
void getValidInt(string prompt, int &var);
void getValidFloat(string prompt, float &var);


/*1. To add trains*/
void addTrain(){
    train t;
    bool trainIdExists = false;

    // ensuring train id uniqueness
    do{
        getValidInt("Enter train ID (XXXX): ", t.id);

        if(searchTrainId(t.id) == true){
            trainIdExists = true;
            cout << "\nPlease enter unique train ID." << endl << endl;
        }
        else{
```

```
            trainIdExists = false;
        }
    }while(trainIdExists);
    cin.ignore();

    cout << endl;
    cout << "Enter train name: ";
    getline(cin, t.name);
    cout << "Enter departure and arrival destination (From/To): ";
    getline(cin, t.fromTo);

    getValidInt("Enter departure time (24hr): ", t.depTime);
    getValidInt("Enter arrival time (24hr): ", t.arrivalTime);
    getValidInt("Enter departure date (DDMMYYYY): ", t.depDate);

    getValidFloat("Enter ticket price (RS): ", t.ticketPrice);

    getValidInt("Enter total of seats: ", t.totalSeats);
    getValidInt("Enter available seats: ", t.availableSeats);

    //sending data to file
    ofstream fileOut("trainData.txt", ios::app);
    if(!fileOut.is_open()) cout << "Error in opening file to send train data." <<
endl;
    else{
        fileOut<<t.id<<"|"<<t.name<<"|"<<t.fromTo<<"|"<<t.depTime<<"|"<<t.arrival
Time<<"|"<<t.depDate<<"|"<<t.ticketPrice<<"|"<<t.totalSeats<<"|"<<t.availableSeat
s<<endl;
        cout << "\nTrain added successfully!" << endl << endl;
        fileOut.close();
    }
}

/*2. To view all trains*/
void viewAllTrains(){
    train t;
    string row;

    ifstream fileIn("trainData.txt", ios::in);
    if(!fileIn.is_open()) cout << "Error in opening file to get train data." <<
endl << endl;
    else{
        int sNo = 1;
```

```
        cout << "S.No. Train ID  Name              From/To             Dep.
Time   Arrival Time  Dep. Date  Ticket Price  Total Seats  Avaiable Seats " <<
endl;
        for(int i = 0; i < 133; i++) cout << "-";
        cout << endl;

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

            vector<string> word = split(row, '|');

            t.id = stoi(word[0]);
            t.name = word[1];
            t.fromTo = word[2];
            t.depTime = stoi(word[3]);
            t.arrivalTime = stoi(word[4]);
            t.depDate = stoi(word[5]);
            t.ticketPrice = stof(word[6]);
            t.totalSeats = stoi(word[7]);
            t.availableSeats =  stoi(word[8]);

            cout << left;
            cout<<setw(6)<<sNo<<setw(10)<<t.id<<setw(18)<<t.name<<setw(20)<<t.fro
mTo<<setw(12)<<t.depTime<<setw(14)<<t.arrivalTime<<setw(11)<<t.depDate<<setw(14)<
<t.ticketPrice<<setw(13)<<t.totalSeats<<setw(15)<<t.availableSeats<<endl;

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

/*3. To search train with ID*/
bool searchTrainId(int targetId){
    train t;
    string row;
    bool trainFound = false;

    ifstream fileIn("trainData.txt", ios::in);
    if(!fileIn.is_open()) cout << "Error in opening file to search train with
ID." << endl << endl;
    else{
        while(!fileIn.eof()){
```

```
            getline(fileIn, row);
            if(row.empty()) continue;

            vector<string> word = split(row, '|');
            t.id = stoi(word[0]);
            t.name = word[1];
            t.fromTo = word[2];
            t.depTime = stoi(word[3]);
            t.arrivalTime = stoi(word[4]);
            t.depDate = stoi(word[5]);
            t.ticketPrice = stof(word[6]);
            t.totalSeats = stoi(word[7]);
            t.availableSeats =  stoi(word[8]);

            if(targetId == t.id){
                trainFound = true;
                break;
            }
        }
        fileIn.close();

        if(trainFound){
            cout << endl;
            cout << "Train found! Details:" << endl;
            int sNo = 1;

            cout << "S.No. Train ID  Name              From/To             Dep.
Time   Arrival Time  Dep. Date  Ticket Price  Total Seats  Avaiable Seats " <<
endl;
            for(int i = 0; i < 133; i++) cout << "-";
            cout << endl;

            cout << left;
            cout<<setw(6)<<sNo<<setw(10)<<t.id<<setw(18)<<t.name<<setw(20)<<t.fro
mTo<<setw(12)<<t.depTime<<setw(14)<<t.arrivalTime<<setw(11)<<t.depDate<<setw(14)<
<t.ticketPrice<<setw(13)<<t.totalSeats<<setw(15)<<t.availableSeats<<endl;
        }
        else{
            cout <<"\nTrain ID is unique!" << endl;
        }
    }
    return trainFound;
}

/*4. To add customer*/
```

```
void addCustomer(){
    customer c;
    // declaring these variables because when customer will enter a train id so i
will get the all details of that train for the customer, inshort these variables
will store the details of customer's train...
    int cusTrainId = 0, cusDepTime = 0, cusArrivalTime = 0, cusDepDate = 0,
cusTotalSeats = 0, cusAvailSeats = 0;
    float cusTicketPrice = 0.00;
    string cusTrainName, cusFromTo;

    bool cnicExists = false, trainIdExists = false, seatsReserved = false;
    int toReserve = 0;

    // input no. 1: CNIC
    do{ // ensuring the uniqueness of cnic
        getValidInt("Enter your CNIC (XXXX): ", c.cnic);

        if(searchCnic(c.cnic) == true){
            cnicExists = true;
            cout << "\nPlease enter unique CNIC." << endl << endl;
        }
        else{
            cnicExists = false;
        }
    }while(cnicExists);
    cin.ignore();

    //input no. 2: Name
    cout << "Enter your name: ";
    getline(cin, c.name);

    // input no. 3: Train ID
    do{ // ensuring that it matches any with train id
        getValidInt("Enter ID of train of which you want to reserve seats: ",
c.trainId);

        if(searchTrainId(c.trainId) == false){
            trainIdExists = false;
            cout << "\nInvalid train ID." << endl << endl;
        }
        else{
            trainIdExists = true;
        }
    }while(!trainIdExists);
```

```
    // getting the data of train for which customer is booking ticket, and
storing them in customer's train variables, because i will do changes with it
    string row;
    ifstream fileIn("trainData.txt", ios::in);
    if(!fileIn.is_open()) cout << "Error in opening file to get train data." <<
endl << endl;
    else{
        while(!fileIn.eof()){
            getline(fileIn, row);
            if(row.empty()) continue;

            vector<string> word = split(row, '|');

            cusTrainId = stoi(word[0]);
            cusTrainName = word[1];
            cusFromTo = word[2];
            cusDepTime = stoi(word[3]);
            cusArrivalTime = stoi(word[4]);
            cusDepDate = stoi(word[5]);
            cusTicketPrice = stof(word[6]);
            cusTotalSeats = stoi(word[7]);
            cusAvailSeats =  stoi(word[8]);

            if(c.trainId == cusTrainId){
                break;
            }
        }
        fileIn.close();
    }

    //input no. 4: No. of seats
    do{
        getValidInt("\nEnter no. of seats you want reserve: ", toReserve);

        if(toReserve < 0){
            seatsReserved = false;
            cout << "\nInvalid no. of seats to reserve!" << endl << endl;
        }
        else if(toReserve > cusAvailSeats){
            seatsReserved = false;
            cout << "\nNo enough seats are available. If there are '0' seats
available, then enter '0'." << endl << endl;
        }
        else{
```

```
            seatsReserved = true;
            c.reservedSeats = toReserve;
        }
    }while(toReserve > cusAvailSeats || toReserve < 0);

    //if seatsReserved is true, so updating data of train in trainData.txt
    if(seatsReserved){
        cusAvailSeats -= toReserve; // updating the seats of train of which
customer has booked seats, because in next step i will be updating the train data
in file

        train t;
        string row;

        ofstream tempOut("temp.txt", ios::app);

        ifstream fileIn("trainData.txt", ios::in);
        if(!fileIn.is_open()) cout << "Error in opening file to get train data."
<< endl << endl;
        else{
            while(!fileIn.eof()){

                getline(fileIn, row);
                if(row.empty()) continue;

                vector<string> word = split(row, '|');

                t.id = stoi(word[0]);
                t.name = word[1];
                t.fromTo = word[2];
                t.depTime = stoi(word[3]);
                t.arrivalTime = stoi(word[4]);
                t.depDate = stoi(word[5]);
                t.ticketPrice = stof(word[6]);
                t.totalSeats = stoi(word[7]);
                t.availableSeats =  stoi(word[8]);

                if(cusTrainId != t.id){
                    tempOut<<t.id<<"|"<<t.name<<"|"<<t.fromTo<<"|"<<t.depTime<<"|
"<<t.arrivalTime<<"|"<<t.depDate<<"|"<<t.ticketPrice<<"|"<<t.totalSeats<<"|"<<t.a
vailableSeats<<endl;
                }
                //sending customer's updated train detail
                else{
```

```
                    tempOut<<cusTrainId<<"|"<<cusTrainName<<"|"<<cusFromTo<<"|"<<
cusDepTime<<"|"<<cusArrivalTime<<"|"<<cusDepDate<<"|"<<cusTicketPrice<<"|"<<cusTo
talSeats<<"|"<<cusAvailSeats<<endl;
                }
            }
            tempOut.close();
            fileIn.close();

            remove("trainData.txt");
            rename("temp.txt", "trainData.txt");
        }
    }

    // now calculating customer's payable amount
    float payableAmount = 0.00;
    int choice = 0; // for payment choice
    if(seatsReserved){
        payableAmount = toReserve*cusTicketPrice;
        cout << "\nYour payable amount is: " << payableAmount << endl;

        do{
            cout << "Select payment status. Enter '1' for paid or Enter '0' for
unpaid: ";
            cin >> choice;

            if(choice == 1) c.paymentStatus = "paid";
            else if(choice == 0) c.paymentStatus = "unpaid";
            else cout << "\nInvalid payment status!" << endl;

        }while(choice != 1 && choice != 0);
    }

    // if seatsReserved is true, it means customer has reserved some seats, so
sending customer data to file
    if(seatsReserved){
        ofstream fileOut("customerData.txt", ios::app);
        if(!fileOut.is_open()) cout << "Error in opening file to send customer
data." << endl << endl;
        else{
            fileOut<<c.cnic<<"|"<<c.name<<"|"<<c.trainId<<"|"<<c.reservedSeats<<"
|"<<c.paymentStatus<<endl;
            cout << "\nTickets booked successfully!" << endl << endl;
            fileOut.close();
        }
    }
```

```
    //calling the summary function
    showSummary (c.cnic, c.name, cusTrainId, cusTrainName, cusFromTo, cusDepTime,
cusArrivalTime, cusDepDate, toReserve, payableAmount, c.paymentStatus);
}

/*5. To show all customers*/
void showAllCustomers(){
    customer c;
    string row;

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

    if(!fileIn.is_open()) cout << "Error in opening file to get customer data!"
<< endl;
    else{
        int sNo = 1;
        cout << "S.No.  CNIC   Name              Train ID  Reserved Seats Payment
Status " << endl;
        for(int i = 0; i < 72; i++) cout << "-";
        cout << endl;

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

            vector<string> word = split(row, '|');

            c.cnic = stoi(word[0]);
            c.name = word[1];
            c.trainId = stoi(word[2]);
            c.reservedSeats = stoi(word[3]);
            c.paymentStatus = word[4];

            cout << left;
            cout<<setw(7)<<sNo<<setw(7)<<c.cnic<<setw(18)<<c.name<<setw(10)<<c.tr
ainId<<setw(15)<<c.reservedSeats<<setw(15)<<c.paymentStatus<<endl;

            sNo++;
        }
        fileIn.close();
    }
}
```

```
/*6. To search customer with cnic*/
bool searchCnic(int targetCnic){
    customer c;
    string row;
    bool cnicFound = false;

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

    if(!fileIn.is_open()) cout << "Error in opening file to search customer with
CNIC." << endl;
    else{
        while(!fileIn.eof()){
            getline(fileIn, row);
            if(row.empty()) continue;

            vector<string> word = split(row, '|');

            c.cnic = stoi(word[0]);
            c.name = word[1];
            c.trainId = stoi(word[2]);
            c.reservedSeats = stoi(word[3]);
            c.paymentStatus = word[4];

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

        if(cnicFound){
            int sNo = 1;
            cout << endl;
            cout << "Customer found! Details:" << endl;
            cout << "S.No.  CNIC   Name              Train ID  Reserved Seats
Payment Status " << endl;
            for(int i = 0; i < 72; i++) cout << "-";
            cout << endl;

            cout << left;
            cout<<setw(7)<<sNo<<setw(7)<<c.cnic<<setw(18)<<c.name<<setw(10)<<c.tr
ainId<<setw(15)<<c.reservedSeats<<setw(15)<<c.paymentStatus<<endl;
        }
        else{
            cout << "\nCNIC is unique!" << endl << endl;
```

```
        }
    }
    return cnicFound;
}

/*7. To show ticket summary*/
void showSummary(int cnic, string name, int trainId, string trainName, string
fromTo, int depTime, int arrivalTime, int depDate, int reservedSeats, float
payableAmount, string status){

    cout << "\n\t\t\t\t===Your Ticket===" << endl << endl;

    cout<<left;
    cout<<"Customer's CNIC:
"<<setw(18)<<cnic<<"From/To:      "<<setw(20)<<fromTo<<"Reeserved Seats:
"<<setw(15)<<reservedSeats<<endl;
    cout<<"Customer's Name: "<<setw(18)<<name<<"Dep
Time:     "<<setw(20)<<depTime<<"Payable
Amount:  "<<setw(15)<<payableAmount<<endl;
    cout<<"Train ID:        "<<setw(18)<<trainId<<"Arrival Time:
"<<setw(20)<<arrivalTime<<"Payment Status:  "<<setw(15)<<status<<endl;
    cout<<"Train Name:      "<<setw(18)<<trainName<<"Dep
Date:     "<<setw(20)<<depDate<<endl;
}

/*8. To cancel ticket*/
void cancelTicket(){

    // these are the intials variables which will hold unupdated values of
customer details
    int cusCnic = 0, cusTrainId = 0, cusReservedSeats = 0;
    string cusName, cusPaymentStatus;

    // these are initial variables which will store unupdated values of
customer's train details
    int cusBookedTrainId = 0, cusBookedDepTime = 0, cusBookedArrivalTime = 0,
cusBookedDepDate = 0, cusBookedTotalSeats = 0, cusBookedAvailSeats = 0;
    float cusBookedTicketPrice = 0.00;
    string cusBookedTrainName, cusBookedFromTo;

    int oldCustomerReserve = 0; // this is because i will store the initial
reseved seats by customer in this variable. later i will be using it calculate
payable amount by this formula, (old reserve seats - to cancel seats)*ticket
price
```

```
    int targetCnic = 0, toCancel = 0;
    bool cnicExists = false, seatsCancelled = false;

    do{
        getValidInt("Enter CNIC of customer (XXXX): ", targetCnic);

        if(searchCnic(targetCnic) == false){
            cnicExists = false;
            cout << "\nNo ticket is booked with this CNIC." << endl << endl;
        }
        else cnicExists = true;
    }while(!cnicExists);

    // now getting all the details customer according to valid cnic
    string customerRow;

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

    if(!customerFileIn.is_open()) cout << "Error in opening file to get customer
data!" << endl;
    else{
        while(!customerFileIn.eof()){
            getline(customerFileIn, customerRow);
            if(customerRow.empty()) continue;

            vector<string> word = split(customerRow, '|');

            cusCnic = stoi(word[0]);
            cusName = word[1];
            cusTrainId = stoi(word[2]);
            cusReservedSeats = stoi(word[3]);
            cusPaymentStatus = word[4];

            if(targetCnic == cusCnic){
                break;
                oldCustomerReserve = cusReservedSeats; // i will be his initial
reserved in the calculation of payable amount
            }
        }
        customerFileIn.close();
    }

    // also getting the data of train for which customer has booked seats
    string trainRow;
    ifstream trainFileIn("trainData.txt", ios::in);
```

```
    if(!trainFileIn.is_open()) cout << "Error in opening file to get train data."
<< endl << endl;
    else{
        while(!trainFileIn.eof()){
            getline(trainFileIn, trainRow);
            if(trainRow.empty()) continue;

            vector<string> word = split(trainRow, '|');

            cusBookedTrainId = stoi(word[0]);
            cusBookedTrainName = word[1];
            cusBookedFromTo = word[2];
            cusBookedDepTime = stoi(word[3]);
            cusBookedArrivalTime = stoi(word[4]);
            cusBookedDepDate = stoi(word[5]);
            cusBookedTicketPrice = stof(word[6]);
            cusBookedTotalSeats = stoi(word[7]);
            cusBookedAvailSeats =  stoi(word[8]);

            if(cusTrainId == cusBookedTrainId){
                break;
            }
        }
        trainFileIn.close();
    }

    // asking about seats to cancel
    do{
        getValidInt("\nEnter the number of seats you want to cancel: ",
toCancel);

        if(toCancel > cusReservedSeats){
            seatsCancelled = false;
            cout << "\nNo enough seats are booked by this customer!" << endl <<
endl;
        }
        else{
            seatsCancelled = true;
            cout << "\nSeats cancelled successfully!" << endl << endl;
        }
    }while(!seatsCancelled);
```

```
    //if seatsCancelled got true, updating variables, because we will be sending
them in file
    if(seatsCancelled){
        cusReservedSeats -= toCancel; // updating data of customer's reserved
seats
        cusBookedAvailSeats += toCancel; // updating data of customer's booked
train's available seats
    }

    // if seatsCancelled got true, so updating customer data in file...
    if(seatsCancelled){

        customer c;
        string row;

        ofstream customerTempOut("temp.txt", ios::app);
        ifstream oldCustomerIn("customerData.txt", ios::in);

        if(!oldCustomerIn.is_open()) cout << "Error in opening file to get
customer data!" << endl;
        else{
            while(!oldCustomerIn.eof()){
                getline(oldCustomerIn, row);
                if(row.empty()) continue;

                vector<string> word = split(row, '|');

                c.cnic = stoi(word[0]);
                c.name = word[1];
                c.trainId = stoi(word[2]);
                c.reservedSeats = stoi(word[3]);
                c.paymentStatus = word[4];

                // do not send data in temp file of a customer of which we have
cnic
                if(targetCnic != c.cnic){
                   customerTempOut<<c.cnic<<"|"<<c.name<<"|"<<c.trainId<<"|"<<c.r
eservedSeats<<"|"<<c.paymentStatus<<endl;
                }
                else{
                    customerTempOut<<c.cnic<<"|"<<c.name<<"|"<<c.trainId<<"|"<<cu
sReservedSeats<<"|"<<c.paymentStatus<<endl;
                }
            }
            customerTempOut.close();
```

```
            oldCustomerIn.close();

            remove("customerData.txt");
            rename("temp.txt", "customerData.txt");
        }
    }

    // now updating data of train file
    if(seatsCancelled){
        train t;
        string row;

        ofstream trainTempOut("temp.txt", ios::app);

        ifstream oldTrainIn("trainData.txt", ios::in);
        if(!oldTrainIn.is_open()) cout << "Error in opening file to get train
data." << endl << endl;
        else{
            while(!oldTrainIn.eof()){

                getline(oldTrainIn, row);
                if(row.empty()) continue;

                vector<string> word = split(row, '|');

                t.id = stoi(word[0]);
                t.name = word[1];
                t.fromTo = word[2];
                t.depTime = stoi(word[3]);
                t.arrivalTime = stoi(word[4]);
                t.depDate = stoi(word[5]);
                t.ticketPrice = stof(word[6]);
                t.totalSeats = stoi(word[7]);
                t.availableSeats =  stoi(word[8]);

                if(cusBookedTrainId != t.id){
                    trainTempOut
<<t.id<<"|"<<t.name<<"|"<<t.fromTo<<"|"<<t.depTime<<"|"<<t.arrivalTime<<"|"<<t.de
pDate<<"|"<<t.ticketPrice<<"|"<<t.totalSeats<<"|"<<t.availableSeats<<endl;
                }
```

```
                //sending customer's updated train detail
                else{
                    trainTempOut
<<t.id<<"|"<<t.name<<"|"<<t.fromTo<<"|"<<t.depTime<<"|"<<t.arrivalTime<<"|"<<t.de
pDate<<"|"<<t.ticketPrice<<"|"<<t.totalSeats<<"|"<<cusBookedAvailSeats<<endl;
                }
            }
            trainTempOut.close();
            oldTrainIn.close();

            remove("trainData.txt");
            rename("temp.txt", "trainData.txt");
        }
    }

    // now doing calculations for updated ticket
    float payableAmount = 0.00;

    if(seatsCancelled){
        payableAmount = (oldCustomerReserve - toCancel)*cusBookedTicketPrice;
        cout << "\nNow your payable amount is: " << payableAmount << endl <<
endl;
    }

    // calling show summary function
    showSummary (cusCnic, cusName, cusTrainId, cusBookedTrainName,
cusBookedFromTo, cusBookedDepTime, cusBookedArrivalTime, cusBookedDepDate,
cusReservedSeats, payableAmount, cusPaymentStatus);
}

/*9. To check if the train data file is empty or not*/
bool checkTrainFile(){
    bool hasData = false;
    string row;
    ifstream fileIn("trainData.txt", ios::in);

    hasData = (fileIn.is_open() && getline(fileIn, row)); // if it get succesfull
to get line from file, it will return true
    return hasData;
}

/*10. To check customer file if empty or not*/
bool checkCustomerFile(){
    bool hasData = false;
    string row;
```

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

    hasData = (fileIn.is_open() && getline(fileIn, row));
    return hasData;
}

/*11. Extract words from row*/
vector<string> split(string row, char delimiter){
    vector<string> word;
    string temp = ""; // initializing it empty, so that it should not hold any
garbage value.

    for(char ch : row){ // it will run through char by char of row
        if(ch == delimiter){
            word.push_back(temp); // it means that a word has been completed
            temp = "";
        }
        else{
            temp += ch; // building the word through concatination
        }
    }
    word.push_back(temp);

    return word;
}

/*12. To get valid input from user while entering integer*/
void getValidInt(string prompt, int &var){
    cout << prompt;
    cin >> var;

    while(cin.fail()){
        cout << "Inavalid input! Try again..." << endl;
        cin.clear(); // clearing the flag of cin.fail() (making it false..)
        cin.ignore(1000, '\n'); //keep ignoring the characters until the \n
comes..(limit is 1000 characters)

        cout << prompt;
        cin >> var;
    }
}

/*13. To get valid float*/
void getValidFloat(string prompt, float &var){
```

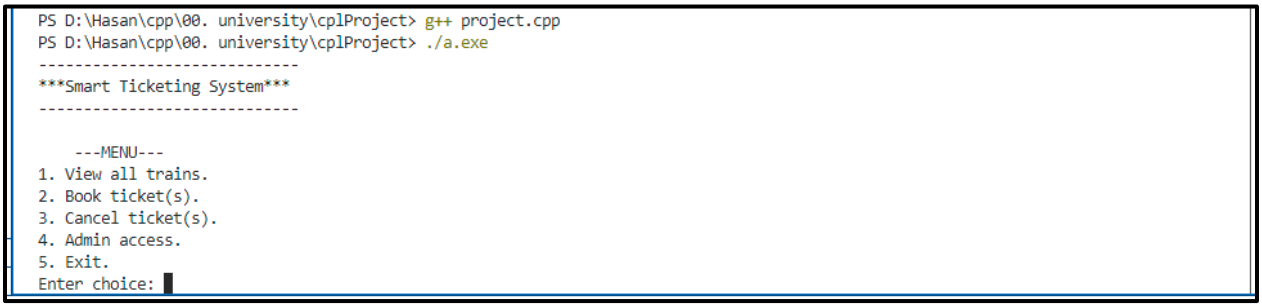
```
    cout << prompt;
    cin >> var;

    while(cin.fail()){
        cout << "Invalid input! Try again..." << endl;
        cin.clear();
        cin.ignore(1000, '\n');
        cout << prompt;
        cin >> var;
    }
}


int main(){
    int userChoice = 0, adminChoice = 0;
    const int adminPassword = 1234;
    int adminInput = 0;

    cout << "----------------------------" << endl;
    cout << "***Smart Ticketing System***" << endl;
    cout << "----------------------------" << endl;
    do{
        cout << endl;
        cout << "    ---MENU---     " << endl;
        cout << "1. View all trains." << endl;
        cout << "2. Book ticket(s)." << endl;
        cout << "3. Cancel ticket(s)." << endl;
        cout << "4. Admin access." << endl;
        cout << "5. Exit." << endl;
        getValidInt("Enter choice: ", userChoice);
        cin.ignore();

        // to view all trains
        if(userChoice == 1){
            cout << endl;
            viewAllTrains();
            cout << endl;
        }
        // to book tickets
        else if(userChoice == 2){

            bool firstTrainAdded = checkTrainFile();
            if(!firstTrainAdded){
                cout << "\nPlease add train first." << endl << endl;
            }
```

```
        else{
            cout << endl;
            addCustomer();
            cout << endl;
        }
    }

    // to cancel tickets
    else if(userChoice == 3){
        bool firstTicketBooked = checkCustomerFile();

        if(!firstTicketBooked){
            cout << "\nPlease book any ticket first." << endl << endl;
        }
        else{
            cout << endl;
            cancelTicket();
            cout << endl;
        }
    }

    // admin access
    else if(userChoice == 4){
        int count = 0;
        bool limitReached = false;
        do{
            getValidInt("\nEnter password: ", adminInput);
            count++;
            if(adminInput != adminPassword){
                cout << "\nIncorrect password!" << endl;
                if(count >= 3){
                    cout << "\nLimit reached! Try again later." << endl;
                    limitReached = true;
                }
            }
            if(limitReached) break;
        }while(adminInput != adminPassword);

        if(!limitReached){
            do{
                cout << endl;
                cout << "-----------------------------" << endl;
                cout << "       ---Admin Menu---      " << endl;
                cout << "1. Add train." << endl;
                cout << "2. See all customers." << endl;
```

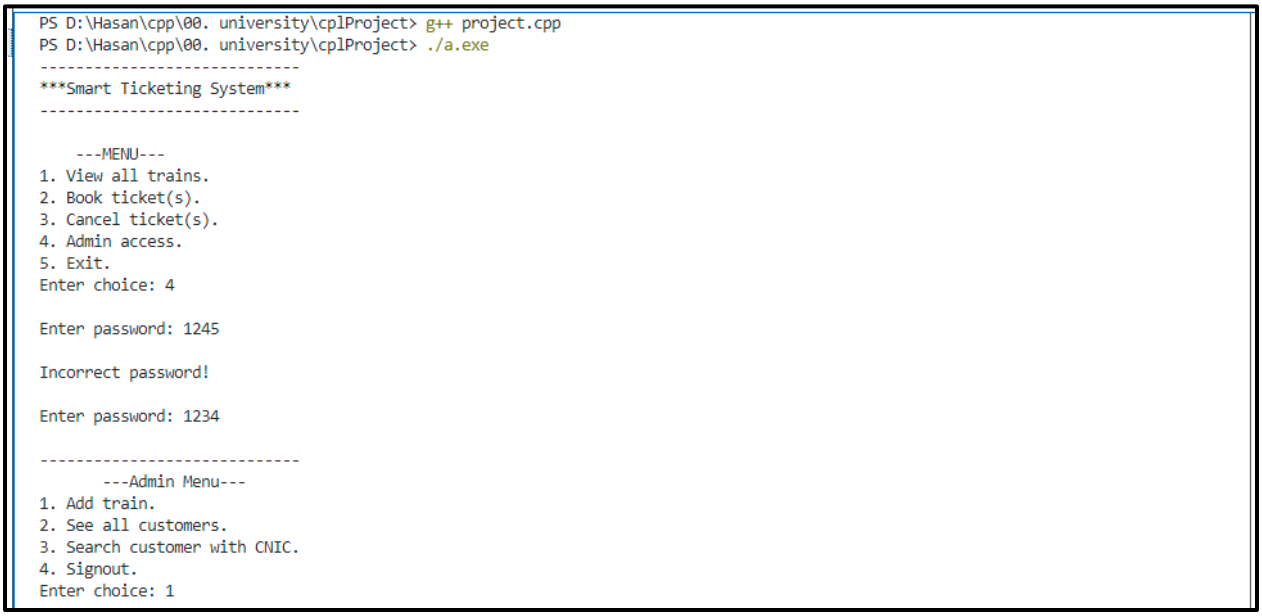
```
                    cout << "3. Search customer with CNIC." << endl;
                    cout << "4. Signout." << endl;
                    getValidInt("Enter choice: ", adminChoice);
                    cin.ignore();
                    // add train
                    if(adminChoice == 1){
                        cout << endl;
                        addTrain();
                        cout << endl;
                    }

                    // see all customers
                    else if(adminChoice == 2){
                        cout << endl;
                        showAllCustomers();
                        cout << endl;
                    }

                    //search customer with cnic
                    else if(adminChoice == 3){
                        int targetCnic = 0;
                        getValidInt("\nEnter CNIC of customer (XXXX): ",
targetCnic);
                        cin.ignore();
                        cout << endl;
                        searchCnic(targetCnic);
                        cout << endl;
                    }
                    // signout
                    else if(adminChoice == 4){
                        cout << "\nSigning out..." << endl << endl;
                    }
                    else cout << "\nInvalid choice!" << endl << endl;

                }while(adminChoice != 4);
            }
        }
        // exit
        else if(userChoice == 5){
            cout << "\nClosing program..." << endl << endl;
        }
        else cout << "\nInvalid choice!" << endl << endl;
    }while(userChoice != 5);
}
```

OUTPUT:

1. Main menu



2. Admin menu



3. Add train (Admin)

```
        ---Admin Menu---
1. Add train.
2. See all customers.
3. Search customer with CNIC.
4. Signout.
Enter choice: 1

Enter train ID (XXXX): 1234

Train ID is unique!

Enter train name: Train A
Enter departure and arrival destination (From/To): Karachi/Lahore
Enter departure time (24hr): 1800
Enter arrival time (24hr): er
Inavalid input! Try again...
Enter arrival time (24hr): 1100
Enter departure date (DDMMYYYY): 03062025
Enter ticket price (RS): 100
Enter total of seats: 50
Enter available seats: 50

Train added successfully!
```

4. View all trains

```
     ---MENU---
1. View all trains.
2. Book ticket(s).
3. Cancel ticket(s).
4. Admin access.
5. Exit.
Enter choice: 1

S.No. Train ID  Name            From/To           Dep. Time   Arrival Time  Dep. Date  Ticket Price  Total Seats  Avaiable Seats
-------------------------------------------------------------------------------------------------------------------------------
1     1234      Train A         Karachi/Lahore    1800        1100          3062025    100           50           50
2     9876      Train B         Multan/Sialkot    1600        800           4062025    500           100          100



    ---MENU---
1. View all trains.
2. Book ticket(s).
3. Cancel ticket(s).
4. Admin access.
5. Exit.
Enter choice:
```

5. Book ticket (with validation)

```
    ---MENU---
1. View all trains.
2. Book ticket(s).
3. Cancel ticket(s).
4. Admin access.
5. Exit.
Enter choice: 2

Enter your CNIC (XXXX): 2345

Customer found! Details:
S.No.  CNIC   Name             Train ID  Reserved Seats Payment Status
-------------------------------------------------------------------------
1      2345   Muhammad Arham   1234      10             paid

Please enter unique CNIC.

Enter your CNIC (XXXX): 1234

CNIC is unique!

Enter your name: Muhammad Hasan
Enter ID of train of which you want to reserve seats: 4556

Train ID is unique!

Invalid train ID.

Enter ID of train of which you want to reserve seats: 1234
```

```
Train ID is unique!

Invalid train ID.

Enter ID of train of which you want to reserve seats: 1234

Train found! Details:
S.No. Train ID  Name             From/To            Dep. Time   Arrival Time  Dep. Date  Ticket Price  Total Seats  Avaiable Seats
-------------------------------------------------------------------------------------------------------------------------------------
1     1234      Train A          Karachi/Lahore     1800        1100          3062025    100           50           40

Enter no. of seats you want reserve: 90

No enough seats are available. If there are '0' seats available, then enter '0'.


Enter no. of seats you want reserve: 10

Your payable amount is: 1000
Select payment status. Enter '1' for paid or Enter '0' for unpaid: 0

Tickets booked successfully!


                              ===Your Ticket===

Customer's CNIC: 1234             From/To:      Karachi/Lahore      Reeserved Seats: 10
Customer's Name: Muhammad Hasan   Dep Time:     1800                Payable Amount:  1000
Train ID:        1234             Arrival Time: 1100                Payment Status:  unpaid
Train Name:      Train A          Dep Date:     3062025
```

6. Cancel ticket

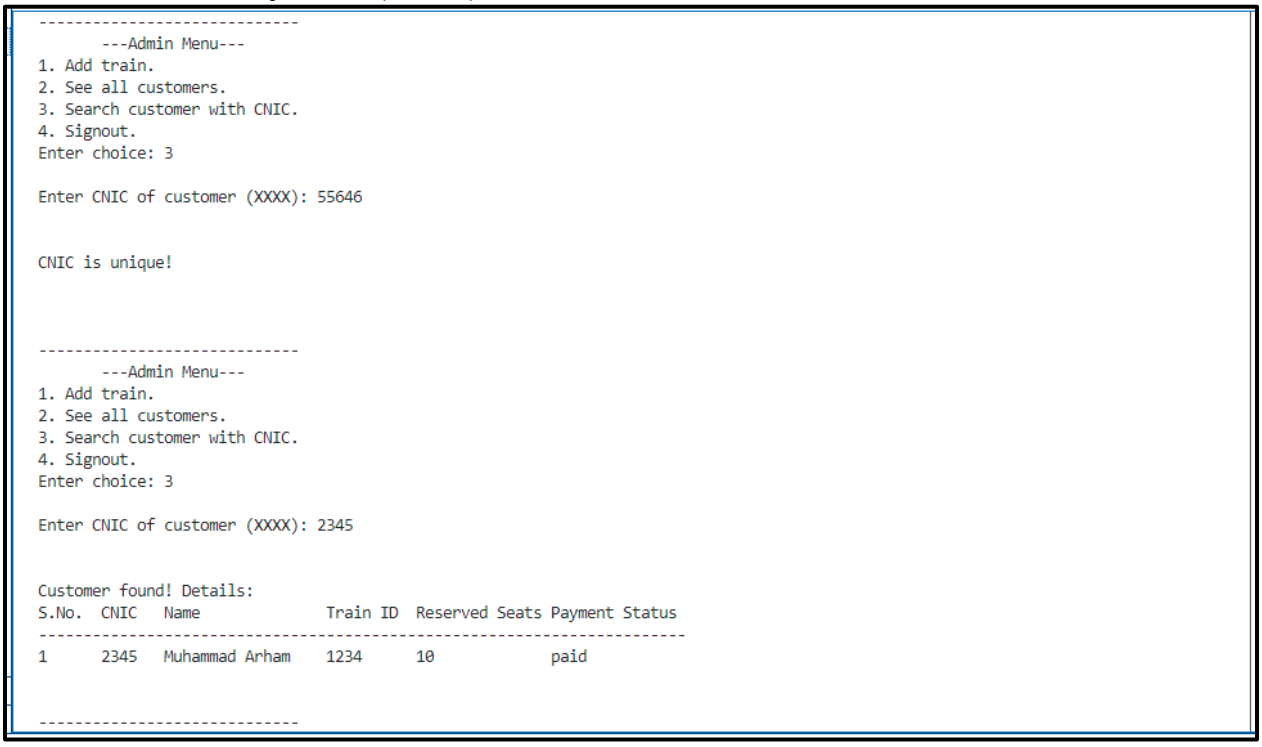
```
Enter CNIC of customer (XXXX): 6788

CNIC is unique!


No ticket is booked with this CNIC.

Enter CNIC of customer (XXXX): 1234

Customer found! Details:
S.No.  CNIC   Name              Train ID  Reserved Seats Payment Status
-------------------------------------------------------------------------
1      1234   Muhammad Hasan    1234      10             unpaid

Enter the number of seats you want to cancel: 23

No enough seats are booked by this customer!


Enter the number of seats you want to cancel: 5

Seats cancelled successfully!
```

```
Seats cancelled successfully!


Now your payable amount is: -500


                              ===Your Ticket===

Customer's CNIC: 1234              From/To:      Karachi/Lahore      Reeserved Seats: 5
Customer's Name: Muhammad Hasan    Dep Time:     1800                Payable Amount:  -500
Train ID:        1234              Arrival Time: 1100                Payment Status:  unpaid
Train Name:      Train A           Dep Date:     3062025
```

7. View all customers (Admin)

```
    ---MENU---
1. View all trains.
2. Book ticket(s).
3. Cancel ticket(s).
4. Admin access.
5. Exit.
Enter choice: 4

Enter password: 1234

-----------------------------
       ---Admin Menu---
1. Add train.
2. See all customers.
3. Search customer with CNIC.
4. Signout.
Enter choice: 2

S.No.  CNIC   Name              Train ID  Reserved Seats Payment Status
-------------------------------------------------------------------------
1      2345   Muhammad Arham    1234      10             paid
2      1234   Muhammad Hasan    1234      5              unpaid
```

8. Search customer by CNIC (Admin)



```
-----------------------------
       ---Admin Menu---
1. Add train.
2. See all customers.
3. Search customer with CNIC.
4. Signout.
Enter choice: 3

Enter CNIC of customer (XXXX): 55646


CNIC is unique!



-----------------------------
       ---Admin Menu---
1. Add train.
2. See all customers.
3. Search customer with CNIC.
4. Signout.
Enter choice: 3

Enter CNIC of customer (XXXX): 2345


Customer found! Details:
S.No.  CNIC   Name              Train ID  Reserved Seats Payment Status
-------------------------------------------------------------------------
1      2345   Muhammad Arham    1234      10            paid


-----------------------------
```

FUTURE DEVELOPMENT:

Some improvements that can be added later:

- GUI using C++ libraries or conversion to a web app using PHP/ASP.NET
- CNIC length validation (format: XXXXX-XXXXXXX-X)
- Date & time format checks
- Train deletion or customer edit options
- Sort/search filters by name, date, price, etc.

CONCLUSION:

This project successfully demonstrates the core concepts of C++ programming through a real-world simulation. It covers essential features like data management, validation, file handling, and structured programming. The system is easy to use, secure, and expandable. It can be enhanced further but already achieves its intended purpose efficiently.