C. V. Raman Global University

Department of Computer Science and Engineering

Bhubaneswar, Odisha (752054)

C++ EXPERIENTIAL LEARNING

A CASE STUDY REPORT

ON

AIRLINE MANAGEMENT SYSTEM

GROUP 12

SUBMITTED BY:

Name	Regd No.
Sarthak Kumar	2201020017
Gourav Sinha	2201020581
Subrat Upadhyay	2201020953
Ayush Agrawal	2201020594
Adarsh Kumar	2201020445

Table of Contents

I. Problem Statement

- The management of passenger's records within airlines is a complex task that demands accuracy, efficiency, and security. Traditional manual methods are susceptible to errors, time-consuming, and lack scalability. The problem at hand is to develop a robust and user-friendly system that automates the process of passenger's record management, ensuring data accuracy, and providing a streamlined interface for users.

- The project aims to develop a comprehensive Airline Management System using C++. In modern and fast phase time, efficient management of airline records like passenger's details is crucial for travel monitoring. However, the current systems often prove to be time-consuming, error-prone, and inefficient.

- This project intends to address these challenges by creating a user-friendly software system that automates the process of booking ticket and providing favourable seats, if not occupied by others. The system will allow authorized users to input and update customer's details, including name, age, and other relevant data.

- Additionally, it provides functionalities for generating ticket details in two formats. One of them is seat number based and other one is without seat number based i.e., all the customer's details.

- The primary goal is to streamline the process of airline management system, reduce workload of the airline company, and ensure accuracy and accessibility of customer's record, ultimately enhancing the efficiency and effectiveness of the airline company/organisation's operations.

II. Introduction

A. BACKGROUND

- Historically, Airline Management System involved extensive paperwork, making it susceptible to errors and challenging to maintain. The need for a more efficient and accurate system led to the development of computerized solutions. The C++ program under consideration is a response to this need, aiming to simplify the process and reduce the burden on airline companies.

- The Airline Management System is a comprehensive software solution designed to streamline the process of managing and organizing passenger's data within airline companies. Developed using C++, this project aims to provide an efficient platform for user/admin to input, store, retrieve, and analyse passenger's information such as customer details, destination airport and other information.

- The system will feature a user-friendly interface allowing authorized personnel to easily navigate through functionalities like booking records, updating records, deleting records, view records. Robust data security measures will be implemented to ensure the confidentiality and integrity of sensitive customer details.

- The project's core objectives include enhancing administrative efficiency, facilitating accurate record-keeping.

B. CHALLENGES

Developing an Airline Management System in C++ presents several challenges:

- Firstly, ensuring efficient data management and storage of customer information while maintaining data integrity can be complex.

- Designing a user-friendly interface that admin/user to interact with the system seamlessly is another challenge.

- Implementing various functionalities like adding records, updating records, deleting records and etc., and handling different grading systems or scales adds complexity to the project.

- Additionally, incorporating features for security and access control to protect sensitive customer’s data from unauthorized access or manipulation is crucial.

- Lastly, testing the system thoroughly to identify and fix bugs while ensuring its scalability and performance under varying workloads poses a significant challenge in this project.

III. Proposed Solution

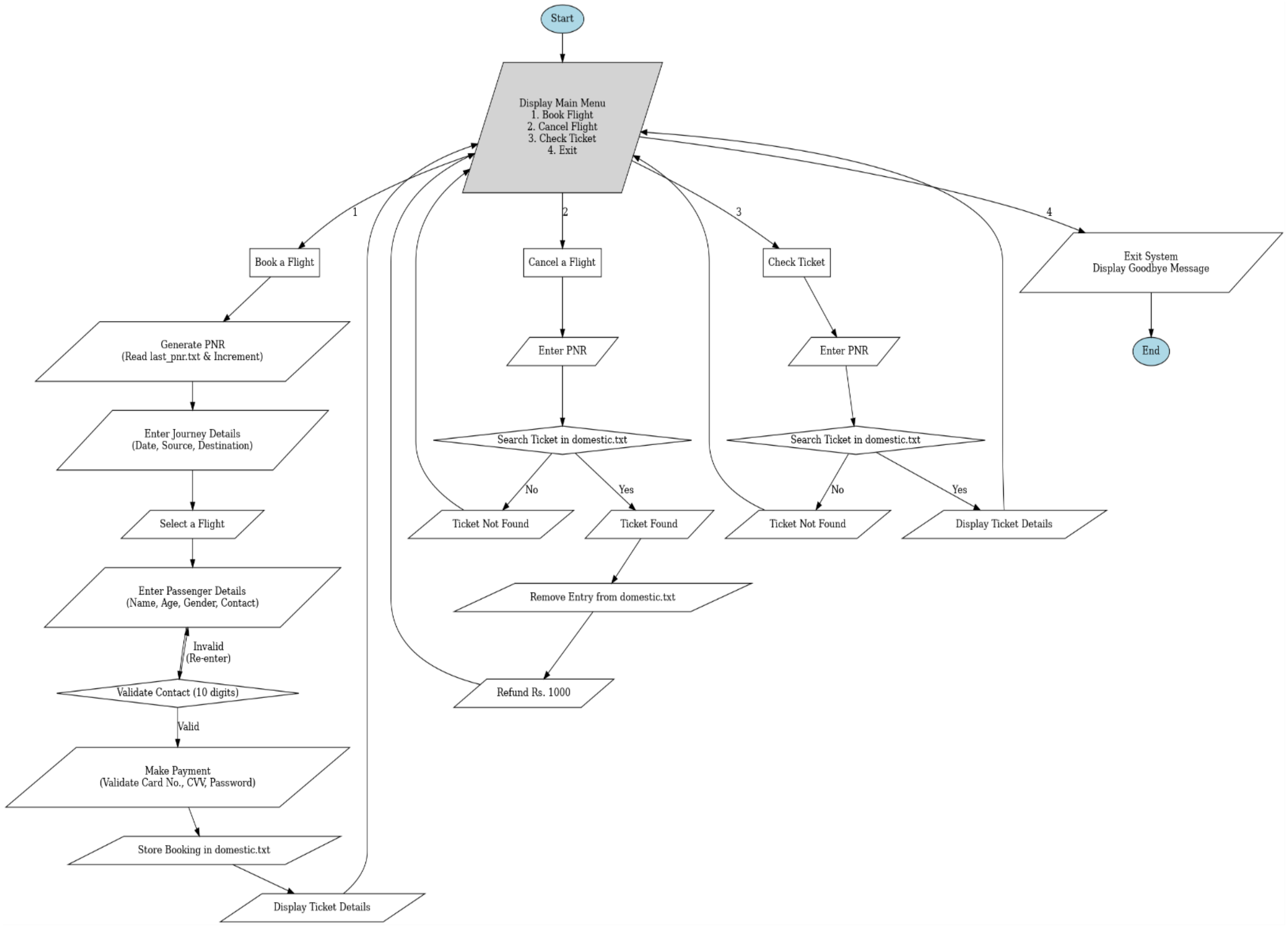
A. DESCRIPTION

- The proposed solution for the Airline Management System in C++ aims to streamline the process of recording, managing, and generating customer's records within airline companies. The system will be designed to efficiently store customer information, including personal details, source and destination details i.e., boarding airport name and destination airport name and additional relevant data.

- Utilizing C++'s object-oriented approach, the system will employ classes to represent customer's details, airport's information ensuring a structured and organized data management system. Through a user-friendly interface, admin/user will be able to input and update customer details, monitor the availability of seats, and generate comprehensive report about number of passengers or number of tickets booked. The system will incorporate features such as data validation (in some case), user authentication, and secure data storage to maintain confidentiality and accuracy.

- Overall, the Airline Management System in C++ aims to enhance efficiency, accuracy, and accessibility in managing customer records who are travelling within airline companies.

B. FLOW CHART



Start
Display Main Menu
1. Book Flight
2. Cancel Flight
3. Check Ticket
4. Exit
1
2
3
4
Book a Flight
Cancel a Flight
Check Ticket
Exit System
Display Goodbye Message
End
Generate PNR
(Read last_pnr.txt & Increment)
Enter Journey Details
(Date, Source, Destination)
Select a Flight
Enter Passenger Details
(Name, Age, Gender, Contact)
Invalid
(Re-enter)
Validate Contact (10 digits)
Valid
Make Payment
(Validate Card No., CVV, Password)
Store Booking in domestic.txt
Display Ticket Details
Enter PNR
Search Ticket in domestic.txt
No
Yes
Ticket Not Found
Ticket Found
Remove Entry from domestic.txt
Refund Rs. 1000
Enter PNR
Search Ticket in domestic.txt
No
Yes
Ticket Not Found
Display Ticket Details

C. ALGORITHM

Step 1: <u>Initialize the system</u>

1. **Display the main menu with options:**
   - ➢ Book a Flight
   - ➢ Cancel a Flight
   - ➢ Check Ticket
   - ➢ Exit
2. **Take user input for their choice.**

Step 2: <u>Book a Flight</u>

1. **Generate PNR:**
   - ➢ Read the last PNR number from the file (last_pnr.txt).
   - ➢ Increment the PNR and update the file.
2. **Enter Journey Details:**
   - ➢ Ask for date of journey, source, and destination.
   - ➢ Display available flights based on the route.
3. **Select a Flight:**
   - ➢ Take the user's choice and store flight details.
4. **Enter Passenger Details:**
   - ➢ Take input for first name, last name, age, gender, email, and contact number.
   - ➢ Validate the contact number (must be 10 digits).
5. **Make Payment:**
   - ➢ Ask for payment method (Debit Card, Credit Card, or Net Banking).
   - ➢ Validate card number (12 digits), CVV, and password.
   - ➢ Display "Transaction Successful" message.
6. **Store the Booking:**
   - ➢ Save the passenger object in the file (domestic.txt).
   - ➢ Display the ticket details.

Step 3: <u>Cancel a Flight</u>

1. **Ask for PNR number.**
2. **Open domestic.txt and search for the matching PNR.**
3. **If found:**
   - ➢ Display ticket details.
   - ➢ Remove the entry from domestic.txt.
   - ➢ Refund Rs. 1000.
4. **If not found:**
   - ➢ Display "Ticket not found".

Step 4: <u>Check Ticket</u>

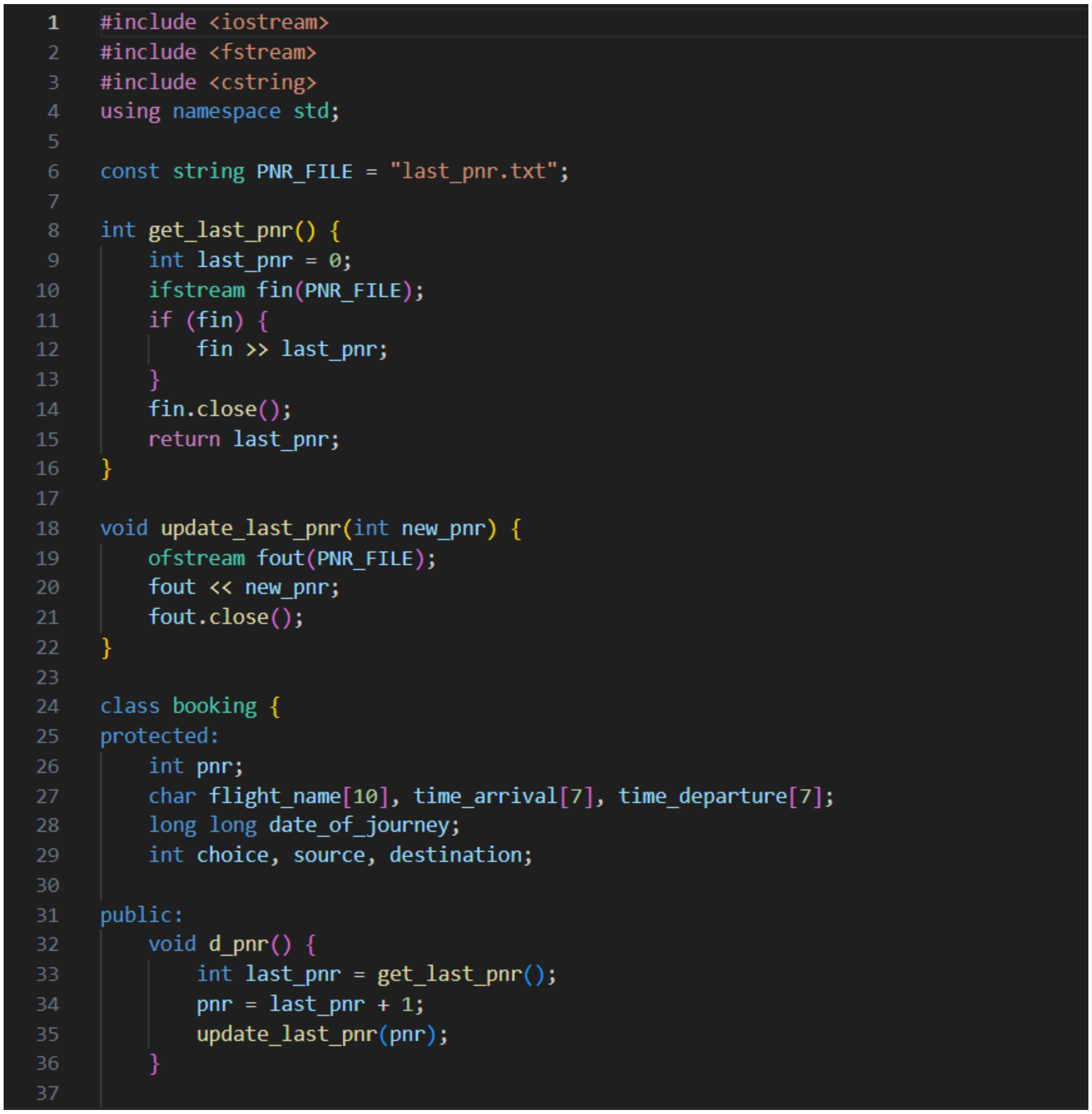
1. **Ask for PNR number.**
2. **Open domestic.txt and search for the PNR.**
3. **If found:**
   - ➢ Display ticket details.
4. **If not found:**
   - ➢ Display "Ticket not found".

Step 5: <u>Exit the System</u>

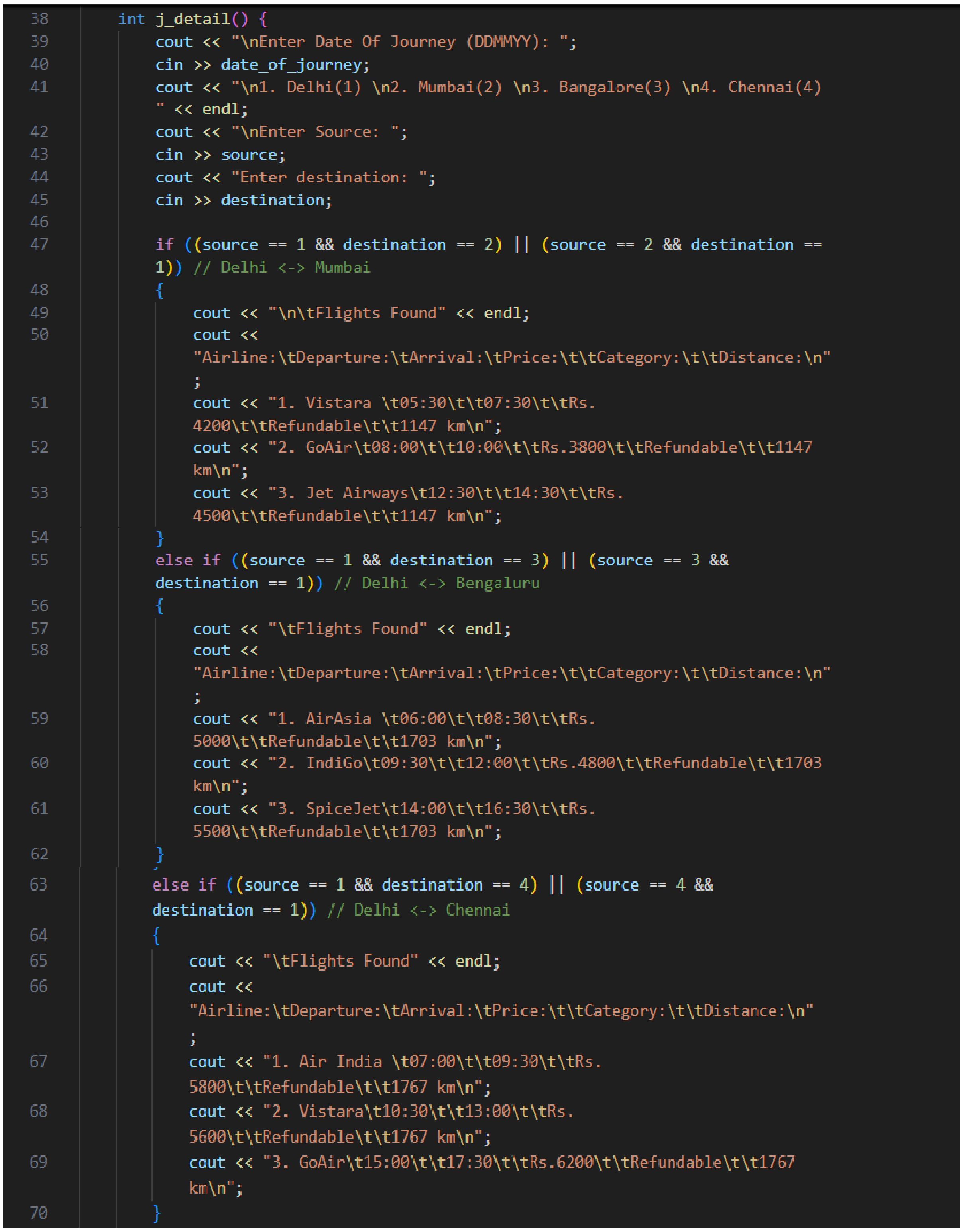
1. **If the user chooses Exit, display a goodbye message.**
2. **End the program.**

D. SOURCE CODE

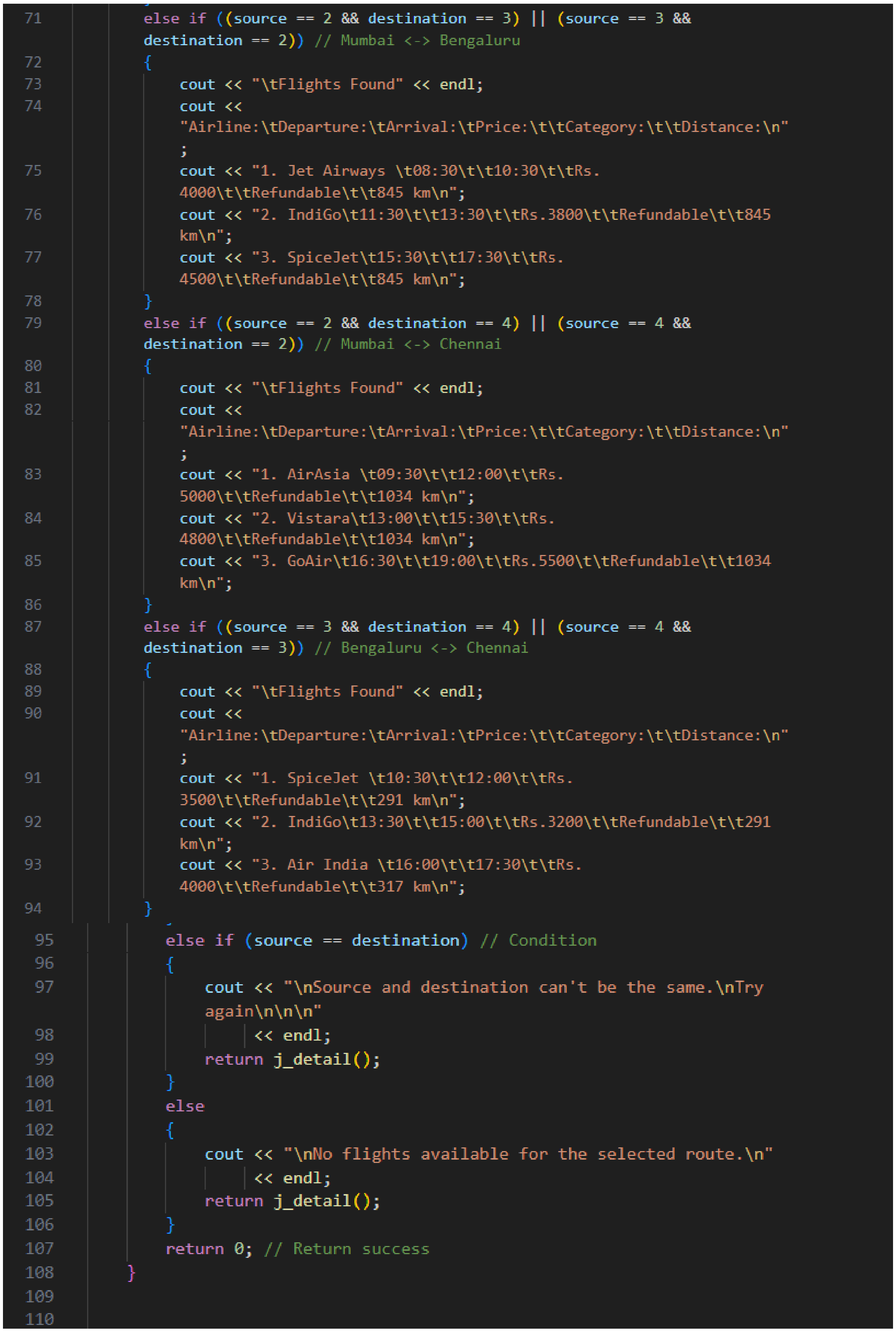
The source code follows best practices for C++ programming. The use of a student class demonstrates encapsulation and abstraction, ensuring a clear separation of concerns. The main function acts as the program's entry point, presenting users with a menu-driven interface.



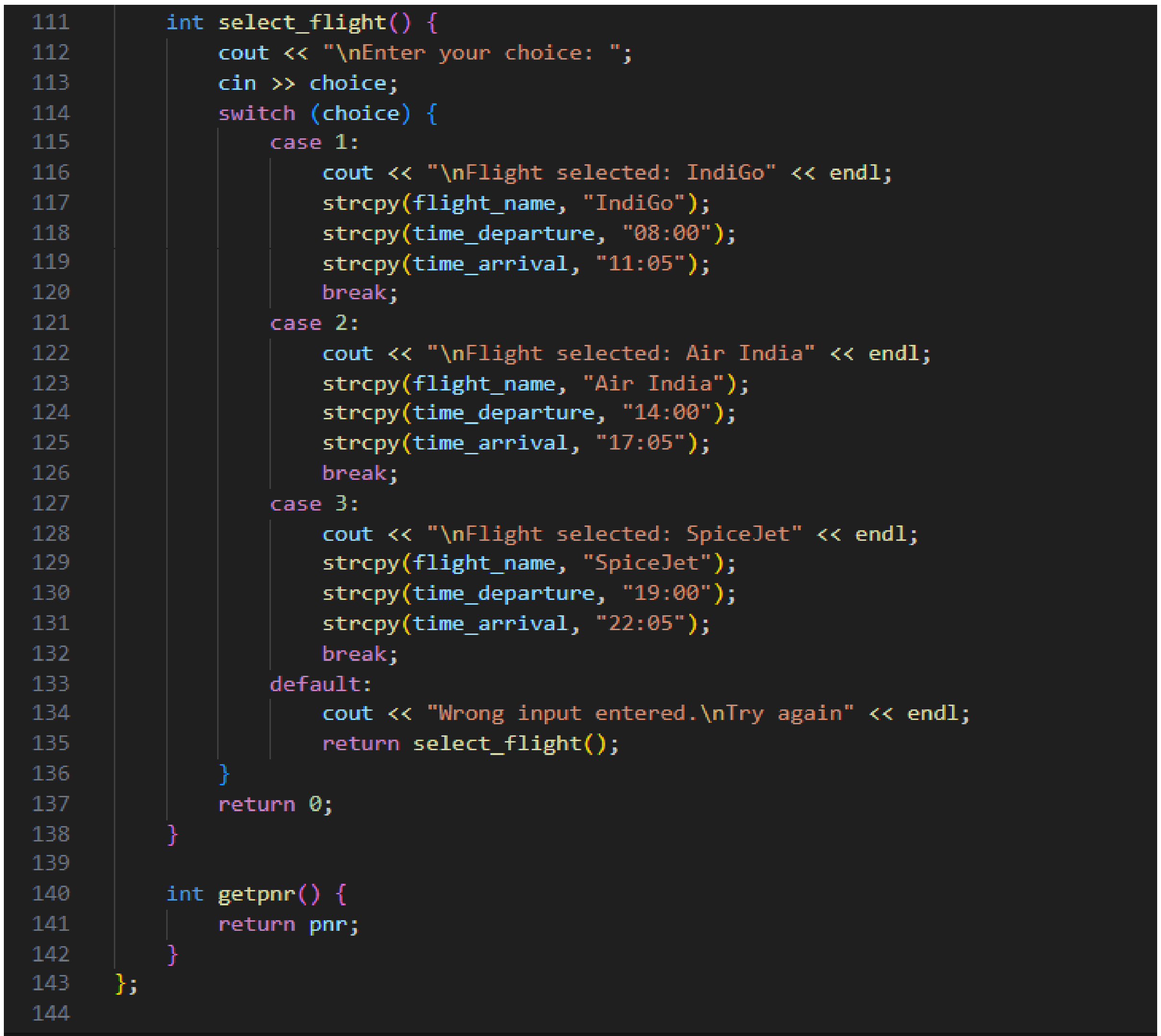
```cpp
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

const string PNR_FILE = "last_pnr.txt";

int get_last_pnr() {
    int last_pnr = 0;
    ifstream fin(PNR_FILE);
    if (fin) {
        fin >> last_pnr;
    }
    fin.close();
    return last_pnr;
}

void update_last_pnr(int new_pnr) {
    ofstream fout(PNR_FILE);
    fout << new_pnr;
    fout.close();
}

class booking {
protected:
    int pnr;
    char flight_name[10], time_arrival[7], time_departure[7];
    long long date_of_journey;
    int choice, source, destination;

public:
    void d_pnr() {
        int last_pnr = get_last_pnr();
        pnr = last_pnr + 1;
        update_last_pnr(pnr);
    }

```



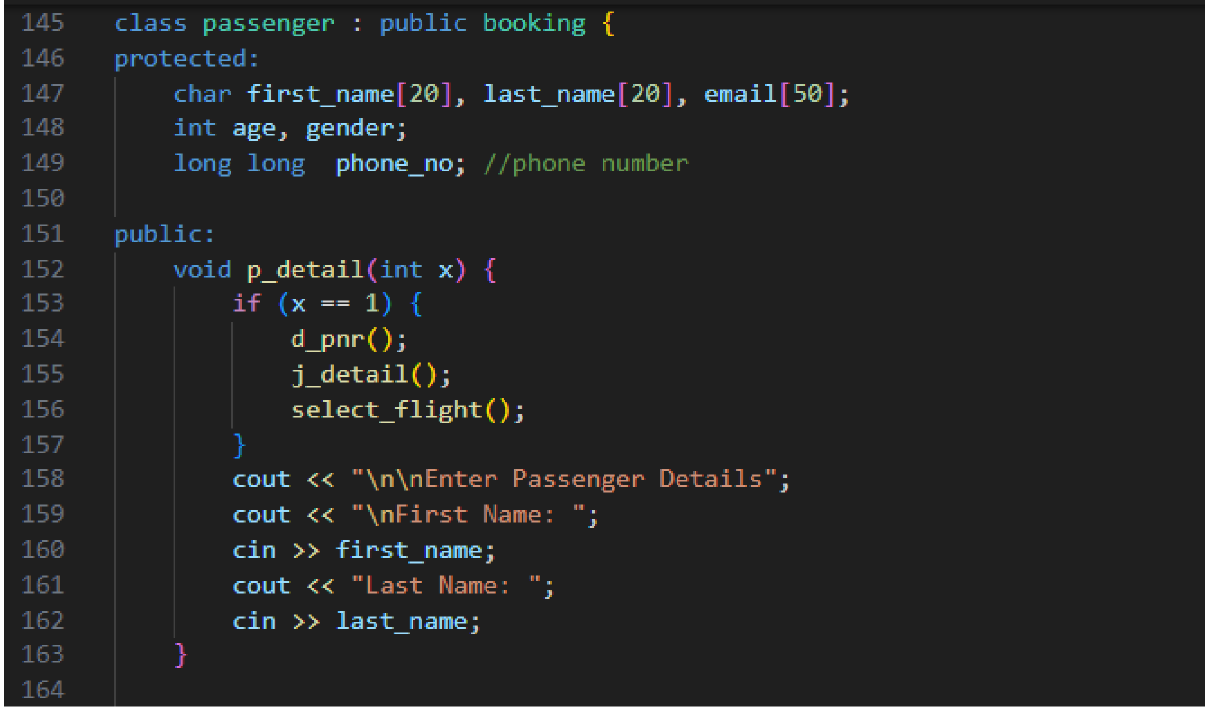
```
int j_detail() {
    cout << "\nEnter Date Of Journey (DDMMYY): ";
    cin >> date_of_journey;
    cout << "\n1. Delhi(1) \n2. Mumbai(2) \n3. Bangalore(3) \n4. Chennai(4)
    " << endl;
    cout << "\nEnter Source: ";
    cin >> source;
    cout << "Enter destination: ";
    cin >> destination;

    if ((source == 1 && destination == 2) || (source == 2 && destination ==
    1)) // Delhi <-> Mumbai
    {
        cout << "\n\tFlights Found" << endl;
        cout <<
        "Airline:\tDeparture:\tArrival:\tPrice:\t\tCategory:\t\tDistance:\n"
        ;
        cout << "1. Vistara \t05:30\t\t07:30\t\tRs.
        4200\t\tRefundable\t\t1147 km\n";
        cout << "2. GoAir\t08:00\t\t10:00\t\tRs.3800\t\tRefundable\t\t1147
        km\n";
        cout << "3. Jet Airways\t12:30\t\t14:30\t\tRs.
        4500\t\tRefundable\t\t1147 km\n";
    }
    else if ((source == 1 && destination == 3) || (source == 3 &&
    destination == 1)) // Delhi <-> Bengaluru
    {
        cout << "\tFlights Found" << endl;
        cout <<
        "Airline:\tDeparture:\tArrival:\tPrice:\t\tCategory:\t\tDistance:\n"
        ;
        cout << "1. AirAsia \t06:00\t\t08:30\t\tRs.
        5000\t\tRefundable\t\t1703 km\n";
        cout << "2. IndiGo\t09:30\t\t12:00\t\tRs.4800\t\tRefundable\t\t1703
        km\n";
        cout << "3. SpiceJet\t14:00\t\t16:30\t\tRs.
        5500\t\tRefundable\t\t1703 km\n";
    }
    else if ((source == 1 && destination == 4) || (source == 4 &&
    destination == 1)) // Delhi <-> Chennai
    {
        cout << "\tFlights Found" << endl;
        cout <<
        "Airline:\tDeparture:\tArrival:\tPrice:\t\tCategory:\t\tDistance:\n"
        ;
        cout << "1. Air India \t07:00\t\t09:30\t\tRs.
        5800\t\tRefundable\t\t1767 km\n";
        cout << "2. Vistara\t10:30\t\t13:00\t\tRs.
        5600\t\tRefundable\t\t1767 km\n";
        cout << "3. GoAir\t15:00\t\t17:30\t\tRs.6200\t\tRefundable\t\t1767
        km\n";
    }
```



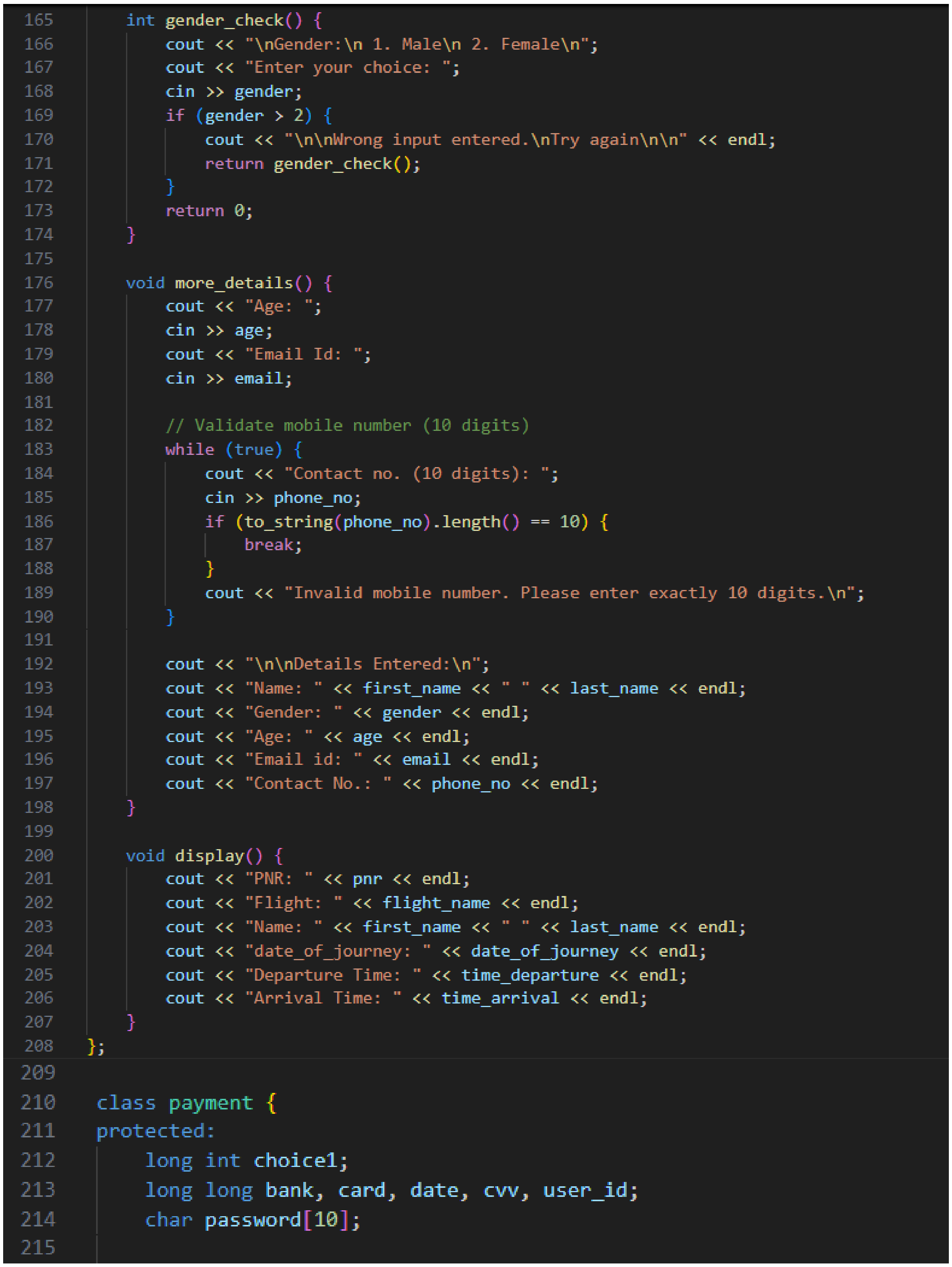
```
        else if ((source == 2 && destination == 3) || (source == 3 &&
        destination == 2)) // Mumbai <-> Bengaluru
        {
            cout << "\tFlights Found" << endl;
            cout <<
            "Airline:\tDeparture:\tArrival:\tPrice:\t\tCategory:\t\tDistance:\n"
            ;
            cout << "1. Jet Airways \t08:30\t\t10:30\t\tRs.
            4000\t\tRefundable\t\t845 km\n";
            cout << "2. IndiGo\t11:30\t\t13:30\t\tRs.3800\t\tRefundable\t\t845
            km\n";
            cout << "3. SpiceJet\t15:30\t\t17:30\t\tRs.
            4500\t\tRefundable\t\t845 km\n";
        }
        else if ((source == 2 && destination == 4) || (source == 4 &&
        destination == 2)) // Mumbai <-> Chennai
        {
            cout << "\tFlights Found" << endl;
            cout <<
            "Airline:\tDeparture:\tArrival:\tPrice:\t\tCategory:\t\tDistance:\n"
            ;
            cout << "1. AirAsia \t09:30\t\t12:00\t\tRs.
            5000\t\tRefundable\t\t1034 km\n";
            cout << "2. Vistara\t13:00\t\t15:30\t\tRs.
            4800\t\tRefundable\t\t1034 km\n";
            cout << "3. GoAir\t16:30\t\t19:00\t\tRs.5500\t\tRefundable\t\t1034
            km\n";
        }
        else if ((source == 3 && destination == 4) || (source == 4 &&
        destination == 3)) // Bengaluru <-> Chennai
        {
            cout << "\tFlights Found" << endl;
            cout <<
            "Airline:\tDeparture:\tArrival:\tPrice:\t\tCategory:\t\tDistance:\n"
            ;
            cout << "1. SpiceJet \t10:30\t\t12:00\t\tRs.
            3500\t\tRefundable\t\t291 km\n";
            cout << "2. IndiGo\t13:30\t\t15:00\t\tRs.3200\t\tRefundable\t\t291
            km\n";
            cout << "3. Air India \t16:00\t\t17:30\t\tRs.
            4000\t\tRefundable\t\t317 km\n";
        }
        else if (source == destination) // Condition
        {
            cout << "\nSource and destination can't be the same.\nTry
            again\n\n\n"
                 << endl;
            return j_detail();
        }
        else
        {
            cout << "\nNo flights available for the selected route.\n"
                 << endl;
            return j_detail();
        }
        return 0; // Return success
    }
```



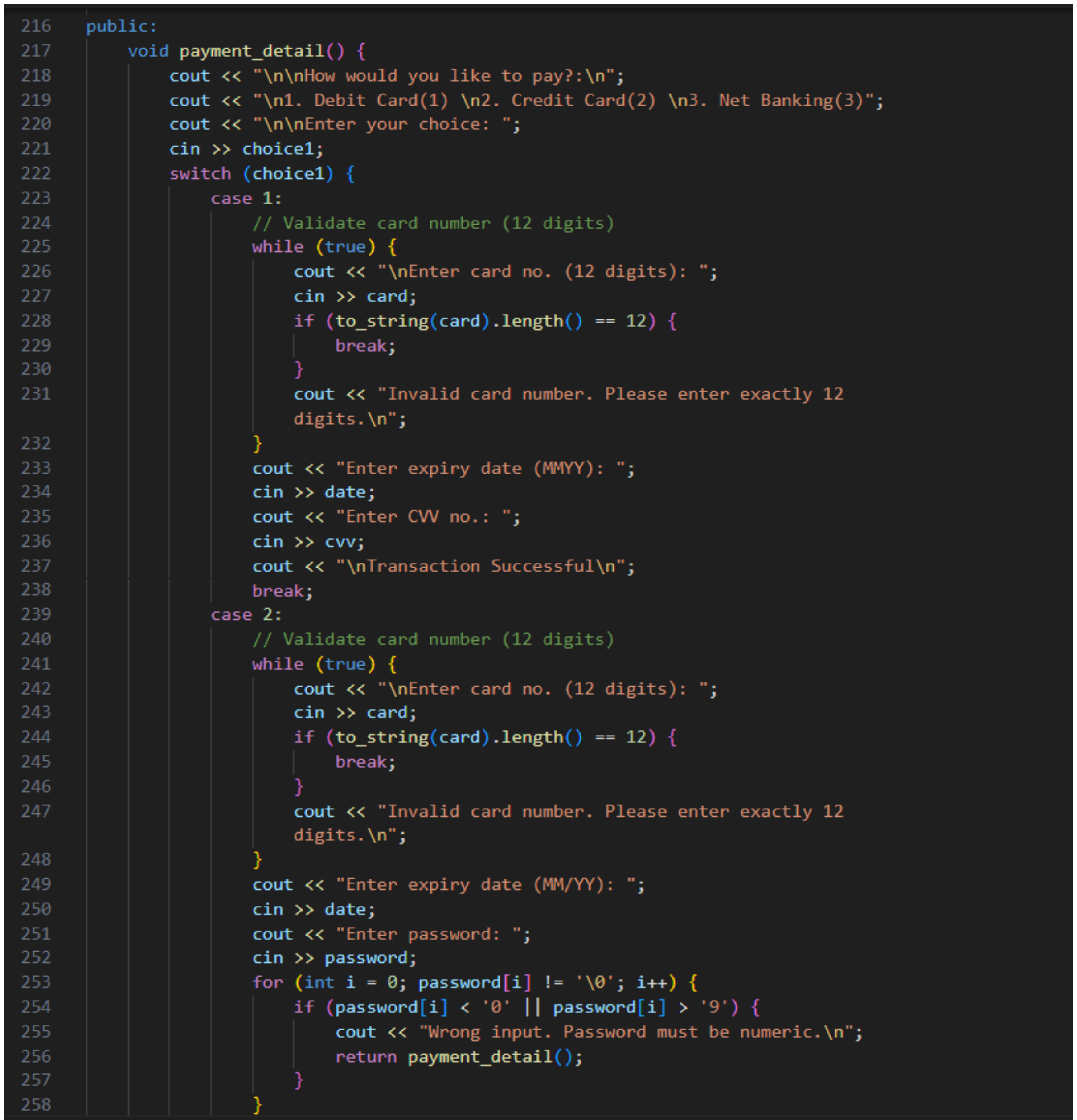
```cpp
    int select_flight() {
        cout << "\nEnter your choice: ";
        cin >> choice;
        switch (choice) {
            case 1:
                cout << "\nFlight selected: IndiGo" << endl;
                strcpy(flight_name, "IndiGo");
                strcpy(time_departure, "08:00");
                strcpy(time_arrival, "11:05");
                break;
            case 2:
                cout << "\nFlight selected: Air India" << endl;
                strcpy(flight_name, "Air India");
                strcpy(time_departure, "14:00");
                strcpy(time_arrival, "17:05");
                break;
            case 3:
                cout << "\nFlight selected: SpiceJet" << endl;
                strcpy(flight_name, "SpiceJet");
                strcpy(time_departure, "19:00");
                strcpy(time_arrival, "22:05");
                break;
            default:
                cout << "Wrong input entered.\nTry again" << endl;
                return select_flight();
        }
        return 0;
    }

    int getpnr() {
        return pnr;
    }
};

```



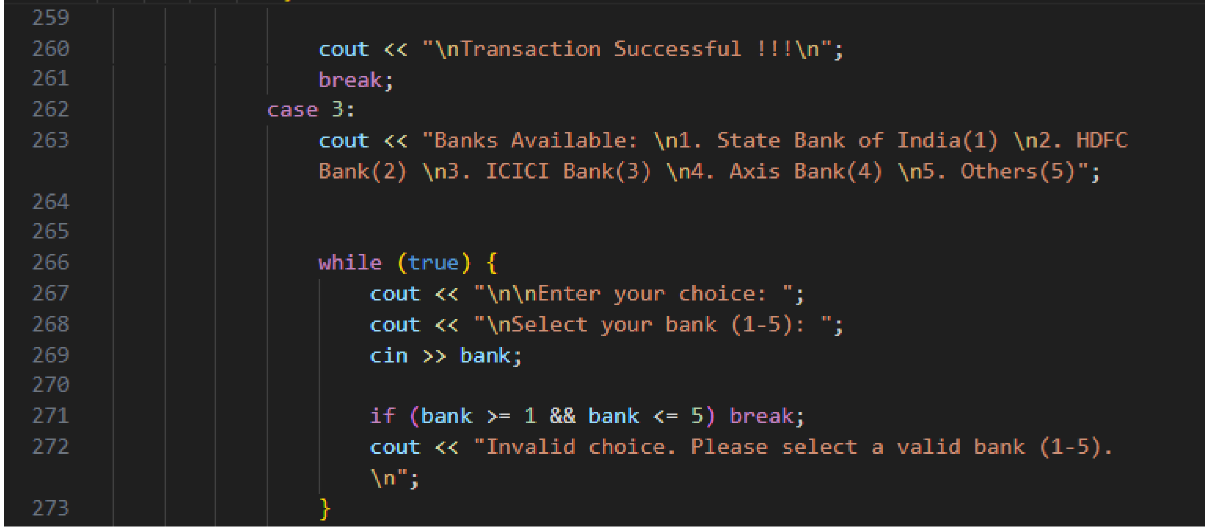
```cpp
class passenger : public booking {
protected:
    char first_name[20], last_name[20], email[50];
    int age, gender;
    long long  phone_no; //phone number

public:
    void p_detail(int x) {
        if (x == 1) {
            d_pnr();
            j_detail();
            select_flight();
        }
        cout << "\n\nEnter Passenger Details";
        cout << "\nFirst Name: ";
        cin >> first_name;
        cout << "Last Name: ";
        cin >> last_name;
    }

```



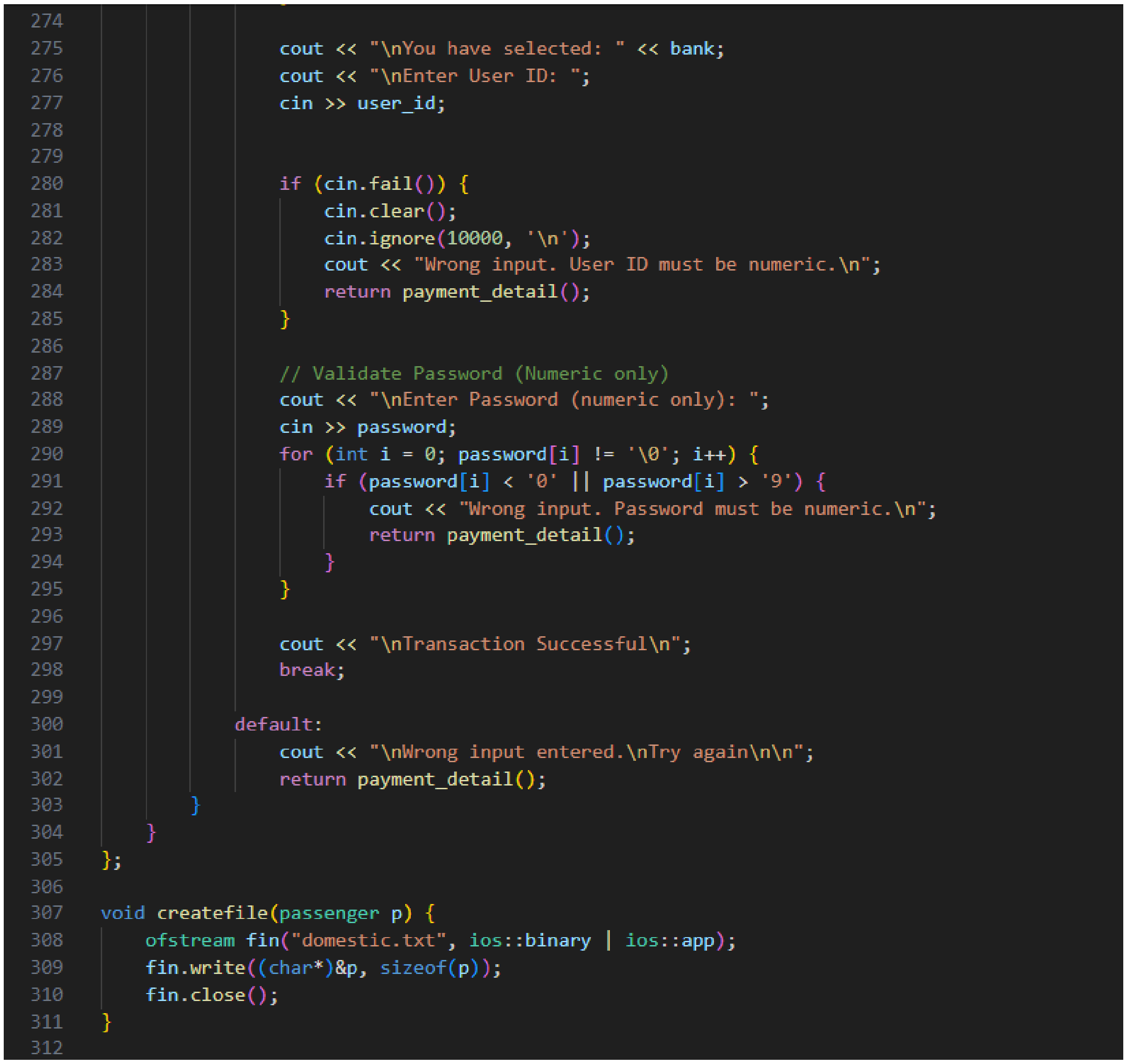
```cpp
    int gender_check() {
        cout << "\nGender:\n 1. Male\n 2. Female\n";
        cout << "Enter your choice: ";
        cin >> gender;
        if (gender > 2) {
            cout << "\n\nWrong input entered.\nTry again\n\n" << endl;
            return gender_check();
        }
        return 0;
    }

    void more_details() {
        cout << "Age: ";
        cin >> age;
        cout << "Email Id: ";
        cin >> email;

        // Validate mobile number (10 digits)
        while (true) {
            cout << "Contact no. (10 digits): ";
            cin >> phone_no;
            if (to_string(phone_no).length() == 10) {
                break;
            }
            cout << "Invalid mobile number. Please enter exactly 10 digits.\n";
        }

        cout << "\n\nDetails Entered:\n";
        cout << "Name: " << first_name << " " << last_name << endl;
        cout << "Gender: " << gender << endl;
        cout << "Age: " << age << endl;
        cout << "Email id: " << email << endl;
        cout << "Contact No.: " << phone_no << endl;
    }

    void display() {
        cout << "PNR: " << pnr << endl;
        cout << "Flight: " << flight_name << endl;
        cout << "Name: " << first_name << " " << last_name << endl;
        cout << "date_of_journey: " << date_of_journey << endl;
        cout << "Departure Time: " << time_departure << endl;
        cout << "Arrival Time: " << time_arrival << endl;
    }
};

class payment {
protected:
    long int choice1;
    long long bank, card, date, cvv, user_id;
    char password[10];

```



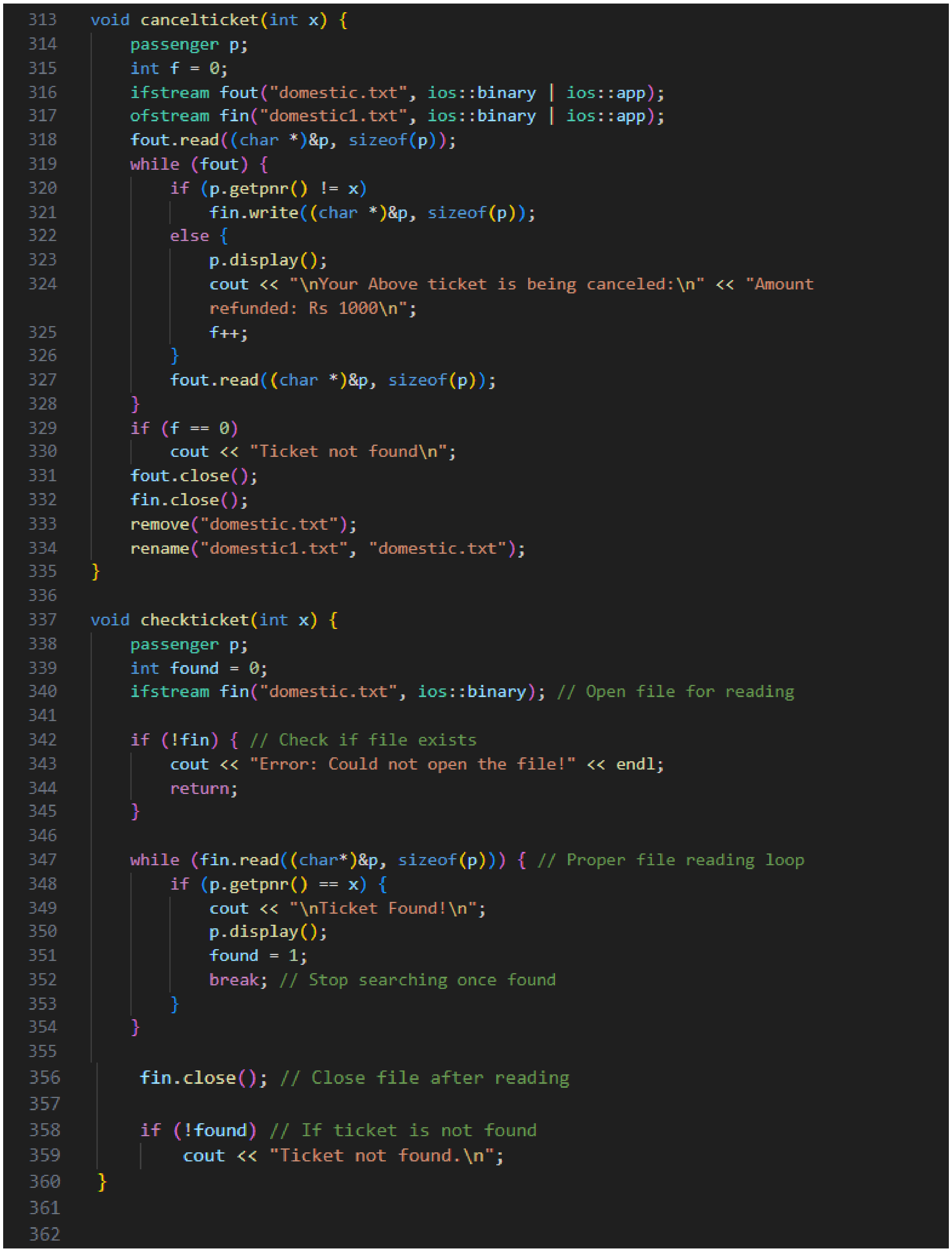
```cpp
public:
    void payment_detail() {
        cout << "\n\nHow would you like to pay?:\n";
        cout << "\n1. Debit Card(1) \n2. Credit Card(2) \n3. Net Banking(3)";
        cout << "\n\nEnter your choice: ";
        cin >> choice1;
        switch (choice1) {
            case 1:
                // Validate card number (12 digits)
                while (true) {
                    cout << "\nEnter card no. (12 digits): ";
                    cin >> card;
                    if (to_string(card).length() == 12) {
                        break;
                    }
                    cout << "Invalid card number. Please enter exactly 12 digits.\n";
                }
                cout << "Enter expiry date (MMYY): ";
                cin >> date;
                cout << "Enter CVV no.: ";
                cin >> cvv;
                cout << "\nTransaction Successful\n";
                break;
            case 2:
                // Validate card number (12 digits)
                while (true) {
                    cout << "\nEnter card no. (12 digits): ";
                    cin >> card;
                    if (to_string(card).length() == 12) {
                        break;
                    }
                    cout << "Invalid card number. Please enter exactly 12 digits.\n";
                }
                cout << "Enter expiry date (MM/YY): ";
                cin >> date;
                cout << "Enter password: ";
                cin >> password;
                for (int i = 0; password[i] != '\0'; i++) {
                    if (password[i] < '0' || password[i] > '9') {
                        cout << "Wrong input. Password must be numeric.\n";
                        return payment_detail();
                    }
                }
```



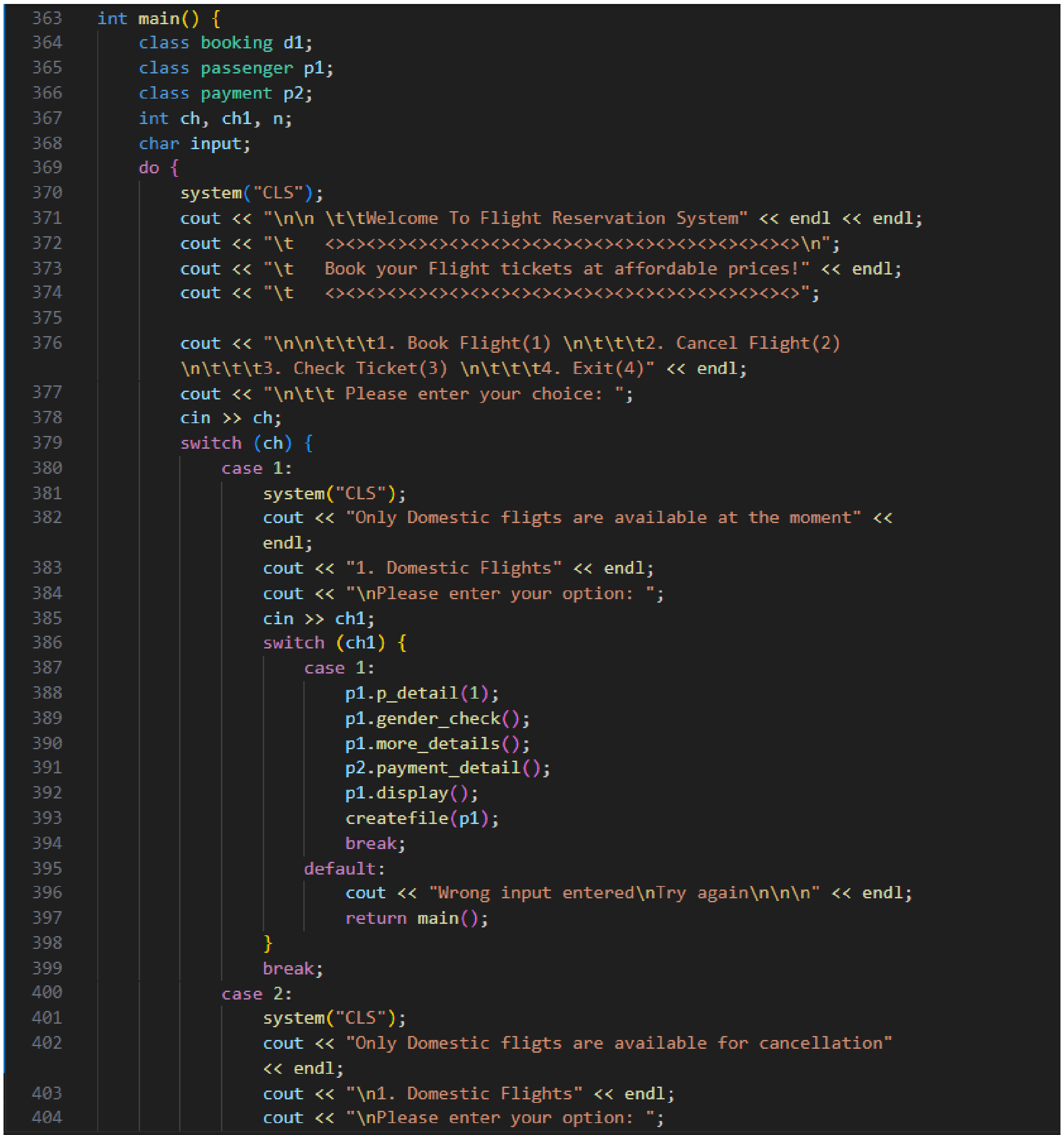
```cpp

                cout << "\nTransaction Successful !!!\n";
                break;
            case 3:
                cout << "Banks Available: \n1. State Bank of India(1) \n2. HDFC Bank(2) \n3. ICICI Bank(3) \n4. Axis Bank(4) \n5. Others(5)";


                while (true) {
                    cout << "\n\nEnter your choice: ";
                    cout << "\nSelect your bank (1-5): ";
                    cin >> bank;

                    if (bank >= 1 && bank <= 5) break;
                    cout << "Invalid choice. Please select a valid bank (1-5).\n";
                }
```



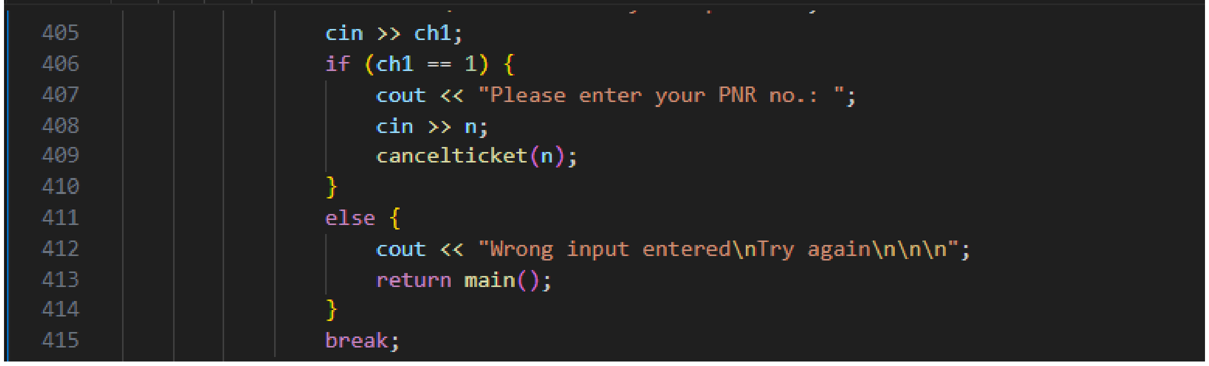
```
                    cout << "\nYou have selected: " << bank;
                    cout << "\nEnter User ID: ";
                    cin >> user_id;


                    if (cin.fail()) {
                        cin.clear();
                        cin.ignore(10000, '\n');
                        cout << "Wrong input. User ID must be numeric.\n";
                        return payment_detail();
                    }

                    // Validate Password (Numeric only)
                    cout << "\nEnter Password (numeric only): ";
                    cin >> password;
                    for (int i = 0; password[i] != '\0'; i++) {
                        if (password[i] < '0' || password[i] > '9') {
                            cout << "Wrong input. Password must be numeric.\n";
                            return payment_detail();
                        }
                    }

                    cout << "\nTransaction Successful\n";
                    break;

                default:
                    cout << "\nWrong input entered.\nTry again\n\n";
                    return payment_detail();
            }
        }
};

void createfile(passenger p) {
    ofstream fin("domestic.txt", ios::binary | ios::app);
    fin.write((char*)&p, sizeof(p));
    fin.close();
}
```



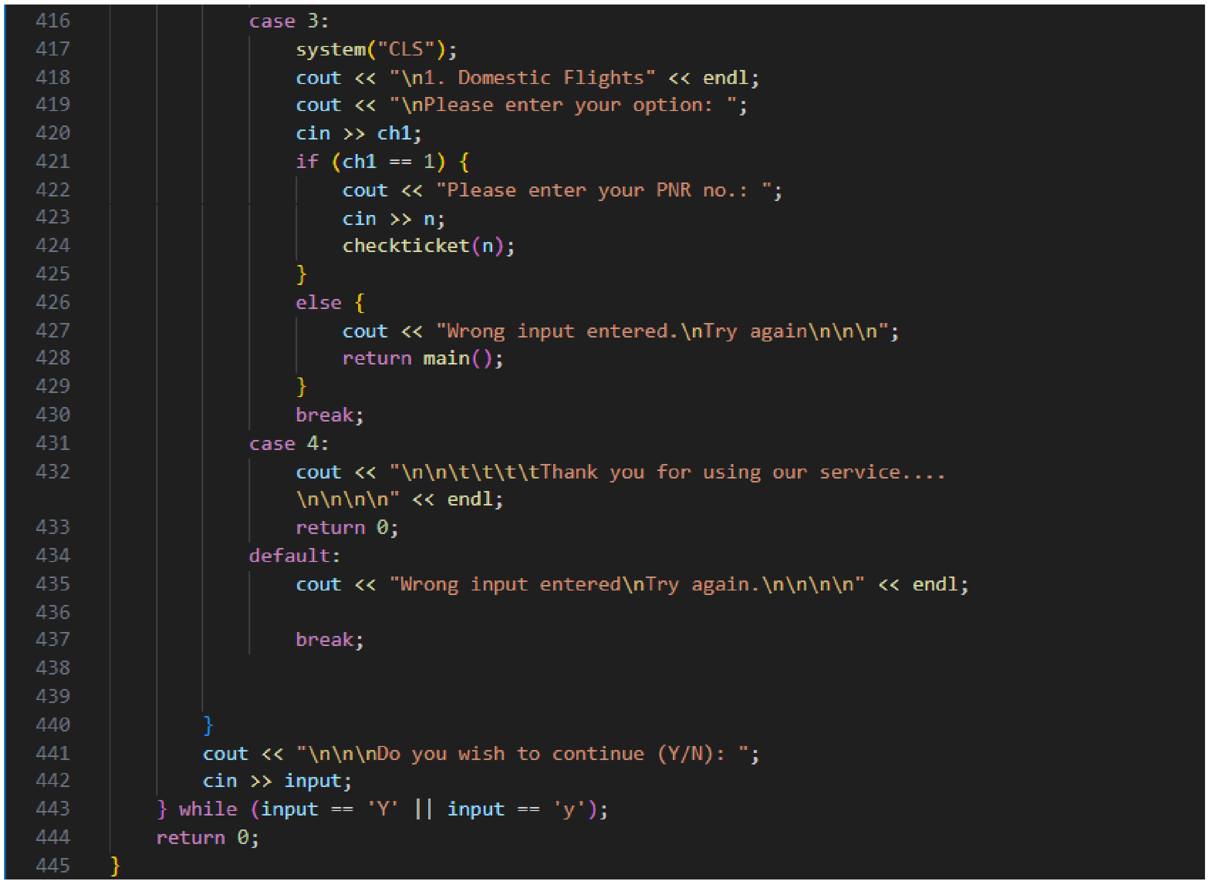
```cpp
void cancelticket(int x) {
    passenger p;
    int f = 0;
    ifstream fout("domestic.txt", ios::binary | ios::app);
    ofstream fin("domestic1.txt", ios::binary | ios::app);
    fout.read((char *)&p, sizeof(p));
    while (fout) {
        if (p.getpnr() != x)
            fin.write((char *)&p, sizeof(p));
        else {
            p.display();
            cout << "\nYour Above ticket is being canceled:\n" << "Amount
            refunded: Rs 1000\n";
            f++;
        }
        fout.read((char *)&p, sizeof(p));
    }
    if (f == 0)
        cout << "Ticket not found\n";
    fout.close();
    fin.close();
    remove("domestic.txt");
    rename("domestic1.txt", "domestic.txt");
}

void checkticket(int x) {
    passenger p;
    int found = 0;
    ifstream fin("domestic.txt", ios::binary); // Open file for reading

    if (!fin) { // Check if file exists
        cout << "Error: Could not open the file!" << endl;
        return;
    }

    while (fin.read((char*)&p, sizeof(p))) { // Proper file reading loop
        if (p.getpnr() == x) {
            cout << "\nTicket Found!\n";
            p.display();
            found = 1;
            break; // Stop searching once found
        }
    }

    fin.close(); // Close file after reading

    if (!found) // If ticket is not found
        cout << "Ticket not found.\n";
}
```



```cpp
int main() {
    class booking d1;
    class passenger p1;
    class payment p2;
    int ch, ch1, n;
    char input;
    do {
        system("CLS");
        cout << "\n\n \t\tWelcome To Flight Reservation System" << endl << endl;
        cout << "\t   <><><><><><><><><><><><><><><><><><><><><><><><><>\n";
        cout << "\t   Book your Flight tickets at affordable prices!" << endl;
        cout << "\t   <><><><><><><><><><><><><><><><><><><><><><><><><>";

        cout << "\n\n\t\t\t1. Book Flight(1) \n\t\t\t2. Cancel Flight(2) \n\t\t\t3. Check Ticket(3) \n\t\t\t4. Exit(4)" << endl;
        cout << "\n\t\t Please enter your choice: ";
        cin >> ch;
        switch (ch) {
            case 1:
                system("CLS");
                cout << "Only Domestic fligts are available at the moment" << endl;
                cout << "1. Domestic Flights" << endl;
                cout << "\nPlease enter your option: ";
                cin >> ch1;
                switch (ch1) {
                    case 1:
                        p1.p_detail(1);
                        p1.gender_check();
                        p1.more_details();
                        p2.payment_detail();
                        p1.display();
                        createfile(p1);
                        break;
                    default:
                        cout << "Wrong input entered\nTry again\n\n\n" << endl;
                        return main();
                }
                break;
            case 2:
                system("CLS");
                cout << "Only Domestic fligts are available for cancellation" << endl;
                cout << "\n1. Domestic Flights" << endl;
                cout << "\nPlease enter your option: ";
```



```cpp
                cin >> ch1;
                if (ch1 == 1) {
                    cout << "Please enter your PNR no.: ";
                    cin >> n;
                    cancelticket(n);
                }
                else {
                    cout << "Wrong input entered\nTry again\n\n\n";
                    return main();
                }
                break;
```



```
            case 3:
                system("CLS");
                cout << "\n1. Domestic Flights" << endl;
                cout << "\nPlease enter your option: ";
                cin >> ch1;
                if (ch1 == 1) {
                    cout << "Please enter your PNR no.: ";
                    cin >> n;
                    checkticket(n);
                }
                else {
                    cout << "Wrong input entered.\nTry again\n\n\n";
                    return main();
                }
                break;
            case 4:
                cout << "\n\n\t\t\t\tThank you for using our service....
                \n\n\n\n" << endl;
                return 0;
            default:
                cout << "Wrong input entered\nTry again.\n\n\n\n" << endl;

                break;


        }
        cout << "\n\n\nDo you wish to continue (Y/N): ";
        cin >> input;
    } while (input == 'Y' || input == 'y');
    return 0;
}
```

IV. Result & Analysis

A. EXPLANATION OF SOURCE CODE

The flight reservation system is implemented in **C++**, using **file handling and object-oriented programming (OOP)** concepts. Below is a breakdown of the **source code** with explanations.

1. File Handling in the Code

The program uses file handling to store and retrieve **PNR numbers and passenger booking details**.

- **last_pnr.txt** → Stores the last assigned PNR number.
  - ➢ **get_last_pnr()** → Reads the last PNR from the file.
  - ➢ **update_last_pnr(int new_pnr)** → Writes the updated PNR to the file.
- **domestic.txt** → Stores passenger booking details in **binary format**.
  - ➢ **createfile(passenger p)** → Saves passenger objects to the file.
  - ➢ **cancelticket(int x)** → Removes a booking by creating a new file and copying all other bookings.
  - ➢ **checkticket(int x)** → Searches for a ticket based on PNR and displays details.

2. Classes and Their Functions

- **booking (Base Class)**
  - ➢ Generates a unique **PNR number**.
  - ➢ Takes user input for **journey details (date, source, destination)**.
  - ➢ Displays **available flights** for selected routes.
- **passenger (Inherits booking)**
  - ➢ Stores **passenger details** (name, age, gender, email, contact number).
  - ➢ Handles **data validation** (e.g., ensuring a 10-digit phone number).
  - ➢ Displays ticket details.
- **payment**
  - ▪ **Debit Card**
  - ▪ **Credit Card**
  - ▪ **Net Banking**
  - ➢ Validates **card number (12 digits), CVV, and password**.
  - ➢ Displays "**Transaction Successful"** if valid.

3. <u>Booking Process Flow</u>

- User chooses to book a flight.
- The system **generates a new PNR** and **stores it in a file**.
- The user selects **source & destination** → Available flights are displayed.
- The user **selects a flight** and enters **personal details**.
- The system validates the **gender, age, contact number**.
- The user **makes a payment**.
- The ticket is **stored in domestic.txt** and displayed.

4. <u>Cancellation & Checking Process</u>

- **Cancel Ticket:**
    - User enters **PNR**.
    - The system searches for it in domestic.txt.
    - If found, the ticket is **removed**, and **Rs. 1000 is refunded**.
- **Check Ticket:**
    - User enters **PNR**.
    - The system searches for it and **displays ticket details** if found.

5. <u>Important Features Used</u>

- **Object-Oriented Programming (OOP)** → Classes, Inheritance (passenger inherits booking).
- **File Handling** → Binary file storage (ifstream & ofstream).
- **Data Validation** → Ensuring correct **PNR, card number, and contact number**.
- **Loop & Recursion** → Handling **incorrect inputs and re-entry requests**.

V. Conclusion

The Flight Reservation System implemented in C++ effectively demonstrates the application of object-oriented programming (OOP) concepts, file handling, and data validation techniques. Through encapsulation and inheritance, the system ensures a modular and structured approach to booking, canceling, and checking flight tickets.

The use of file handling enables persistent storage of passenger details and PNR numbers, ensuring data integrity across multiple sessions. Additionally, robust validation mechanisms enhance the accuracy and security of user inputs, particularly in payment and contact information.

This system provides a user-friendly menu-driven interface, allowing seamless navigation between different functionalities. By implementing essential features such as ticket booking, cancellation, and retrieval, the project successfully mimics real-world airline reservation systems.

In conclusion, the project serves as an excellent demonstration of software development principles, reinforcing key programming concepts while providing a practical solution for flight management. Future enhancements could include database integration, a graphical user interface (GUI), and additional security features for an improved user experience.