PROJECT REPORT

PROJECT TITLE: Ticket Reservation System using C Programming

SUBMITTED BY: Sahaj Parikh

SAP ID: 590026086

Course: Btech. Computer Science & Engineering

Course Code: CSEG1032

INDEX

Sr.No:	Name	Pg.No.:
1.	Abstract	3.
2.	Problem Definition	4.
3.	System Design	5.
4.	Implementation Details	11.
5.	Testing & Results	14.
6.	Applications	16.
7.	Conclusion & Future Work	17.
8.	References	18.
9.	Appendix	19.

ABSTRACT

The Railway Reservation System is simple computer-based application that streamlines the process of boking and managing train tickets. Created in C, the project showcases fundamental programming principles such as structures, file handling, loops and conditional statements. The system enables users to view train options, book multiple tickets, cancel tickets, and access passenger information using saved records.

Why do we need this system?

1. To reduce human error.
2. To provide fast and user-friendly ticket booking system.
3. To maintain organised records.
4. To ensure real time seat availability.

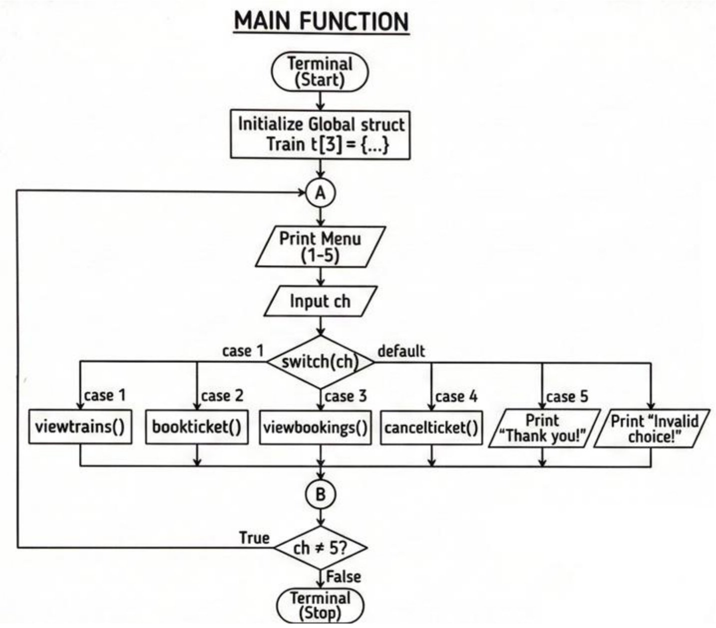
PROBLEM DEFINITION

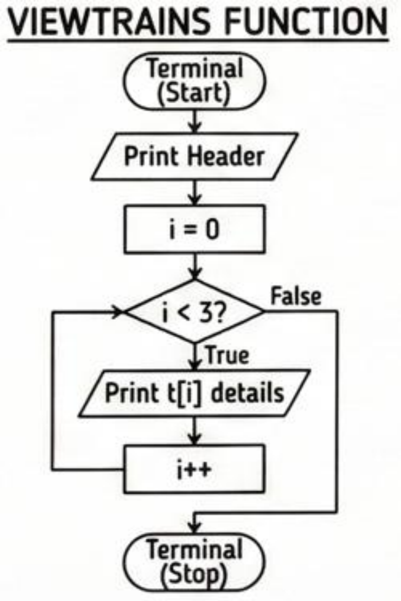
The objective is to create a system that:

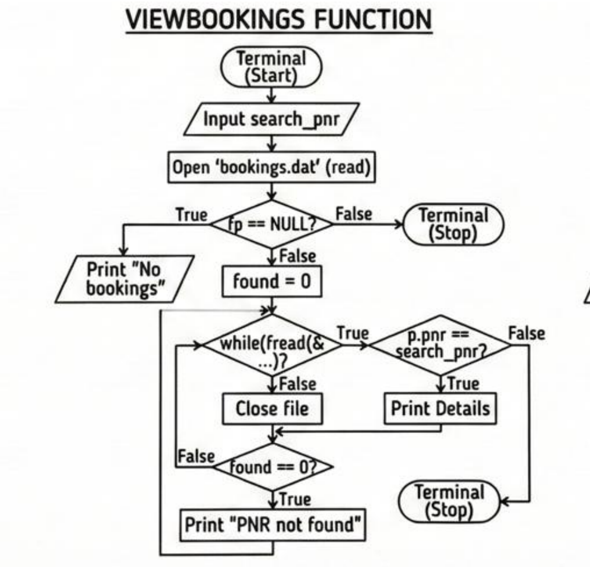
1. Shows the available trains.
2. Permits passengers to reserve seats.
3. Automatically refreshes the count of available seats.
4. Allows for ticket cancellations and reinstates the seat count.
5. Maintains records of booked tickets.

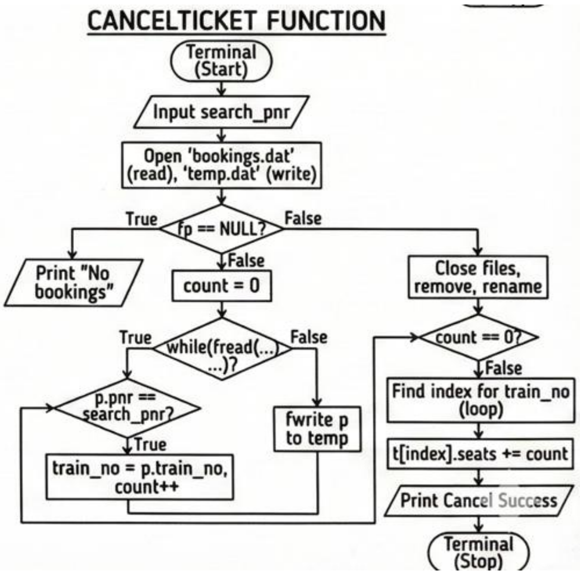
SYSTEM DESGIN

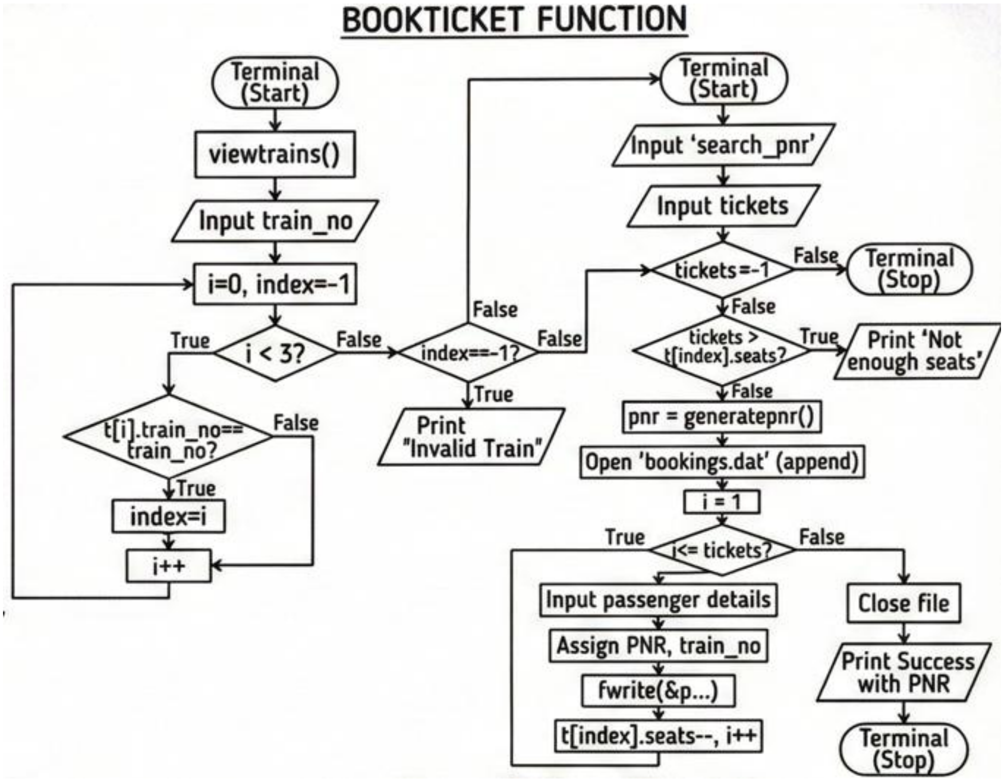
Flowcharts:











Algorithm: Global Definitions

1. **Define Structure Train:** Contains Train Number, Name, Source, Destination, and Seat Count.
2. **Define Structure Passenger:** Contains PNR, Name, Age, Gender, and Train Number.
3. **Initialize Data:** Create a global array of trains (t[3]) with pre-filled details (Rajdhani, Shatabdi, Duronto) and their seat capacities.

Main Function (Control Loop)

Step 1: Start.

Step 2: Initialize an integer variable ch (choice).

Step 3: (Start of Loop) Display the Main Menu:

1. View Available Trains
2. Book Ticket
3. View Bookings By PNR
4. Cancel Ticket
5. Exit

```
Step 4: Read the user's input into ch.
Step 5: Check ch using a Switch case:
    Case 1: Call viewtrains() function.
    Case 2: Call bookticket() function.
    Case 3: Call viewbookings() function.
    Case 4: Call cancelticket() function.
    Case 5: Display "Thank you" and exit the switch.
    Default: Display "Invalid Choice".
Step 6: Check condition: If ch is not equal to 5, go back to Step 3.
Step 7: If ch is equal to 5, Stop.
```

viewtrains() Function

```
Step 1: Start.
Step 2: Print the table header (Train No, Name, From, To, Seats).
Step 3: Initialize loop counter i = 0.
Step 4: Check if i < 3:
    If True: Print details of train t[i]. Increment i by 1. Repeat Step 4.
    If False: Exit the loop.
Step 5: Return to Main function.
```

generatepnr() Function

```
Step 1: Start.
Step 2: Declare a static integer pnr initialized to 1000.
Step 3: Increment pnr by 1.
Step 4: Return the value of pnr.
```

bookticket() Function

```
Step 1: Start.
Step 2: Call viewtrains() to display options to the user.
Step 3: Ask user for Train Number and read input.
Step 4: Search for the Train:
        Loop through the train array.
        If a train matches the input number, store its index.
        If loop finishes and no match is found, print "Invalid Train" and Return.
Step 5: Ask user for Number of Tickets and read input.
Step 6: Check Seat Availability:
        If tickets > t[index].seats, print "Not enough seats" and Return.
Step 7: Call generatepnr() and store the result in variable pnr.
Step 8: Open file "bookings.dat" in Append Binary mode.
Step 9: (Start Loop) Repeat tickets times:
        Input Passenger Name, Age, and Gender.
        Set passenger.pnr = pnr.
        Set passenger.train_no = selected train number.
        Write the passenger structure to the file.
        Decrease seat count in global array: t[index].seats = t[index].seats - 1.
Step 10: Close the file.
Step 11: Print "Booking Successful" and display the generated PNR.
Step 12: Return.
```

viewbookings() Function

```
Step 1: Start.
Step 2: Ask user for PNR to search.
Step 3: Open file "bookings.dat" in Read Binary mode.
Step 4: Check if File exists:
        If NULL, print "No bookings found" and Return.
Step 5: Set flag found = 0.
Step 6: (Start Loop) Read records from file one by one:
        If record.pnr == search_pnr:
        Set found = 1.
        Print Passenger Name, Age, Gender, and Train No.
Step 7: Close the file.
Step 8: If found == 0, print "PNR not found".
Step 9: Return.
```

canceltickeet() Function

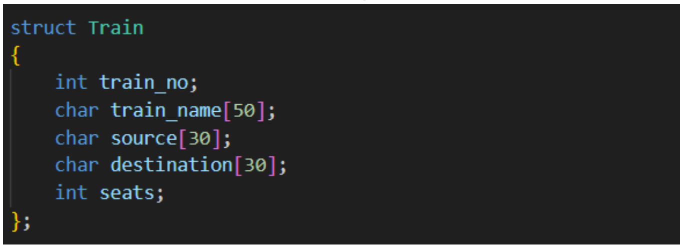
```
Step 1: Start.
Step 2: Ask user for PNR to cancel.
Step 3: Open "bookings.dat" in Read Binary mode.
Step 4: Open "temp.dat" in Write Binary mode.
Step 5: Check if input file exists. If not, print error and Return.
Step 6: Initialize count = 0.
Step 7: (Start Loop) Read records from "bookings.dat":
        If record.pnr == search_pnr:
        Store record.train_no into a variable.
        Increment count. (Do not write this record to temp file).
        Else (PNR does not match):
        Write record to "temp.dat".
Step 8: Close both files.
Step 9: Delete "bookings.dat" and Rename "temp.dat" to "bookings.dat".
Step 10: If count == 0:
    Print "PNR not found" and Return.
Step 11: Update Seat Count:
    Find the train in the global array using the stored train_no.
    Increase seats: t[index].seats = t[index].seats + count.
Step 12: Print "Cancellation Successful".
Step 13: Return.
```

IMPLEMENTATION DETAILS

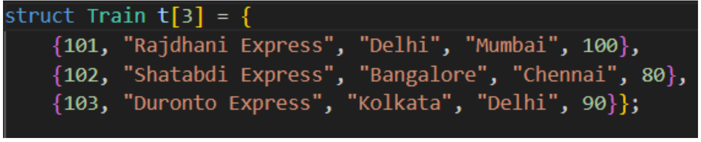
1. Viewing available trains:

   A static array was used to store the train details. The “viewtrains” function was used to display the available trains to the user.

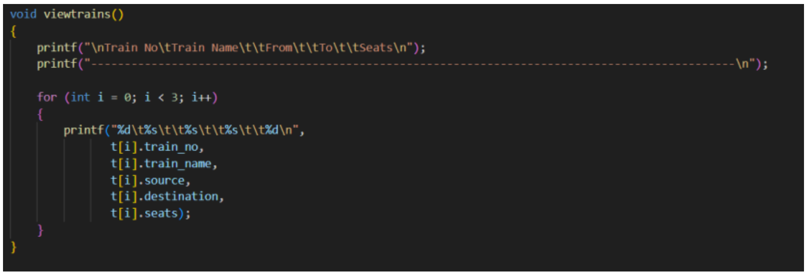
   Implementation using structures.



```c
struct Train
{
    int train_no;
    char train_name[50];
    char source[30];
    char destination[30];
    int seats;
};
```



```c
struct Train t[3] = {
    {101, "Rajdhani Express", "Delhi", "Mumbai", 100},
    {102, "Shatabdi Express", "Bangalore", "Chennai", 80},
    {103, "Duronto Express", "Kolkata", "Delhi", 90}};
```

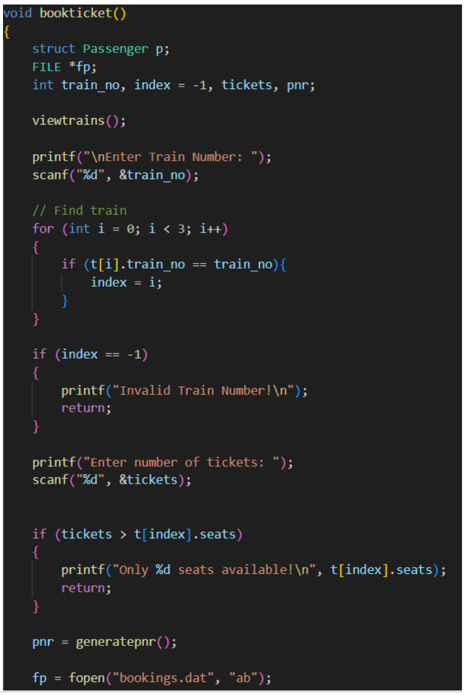


```c
void viewtrains()
{
    printf("\nTrain No\tTrain Name\t\tFrom\t\tTo\t\tSeats\n");
    printf("----------------------------------------------------------------------------------------------\n");

    for (int i = 0; i < 3; i++)
    {
        printf("%d\t%s\t\t%s\t\t%s\t\t%d\n",
                t[i].train_no,
                t[i].train_name,
                t[i].source,
                t[i].destination,
                t[i].seats);
    }
}
```

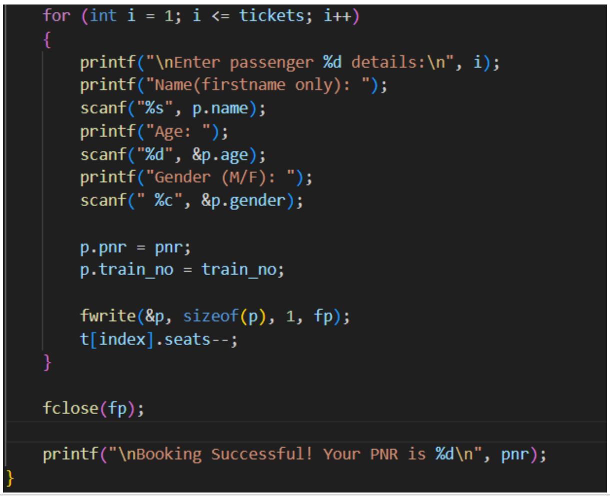
2. Booking Tickets:

   The “bookticket” function was used which accepts train number from the user, verifies if the train exists, asked the user for the number of seats to book, ensures seats are available, generates a single PNR for all passengers, stores the details of passengers in a binary file and decreases the seat count.

   Implementation using file handling, structures to store data and loops.



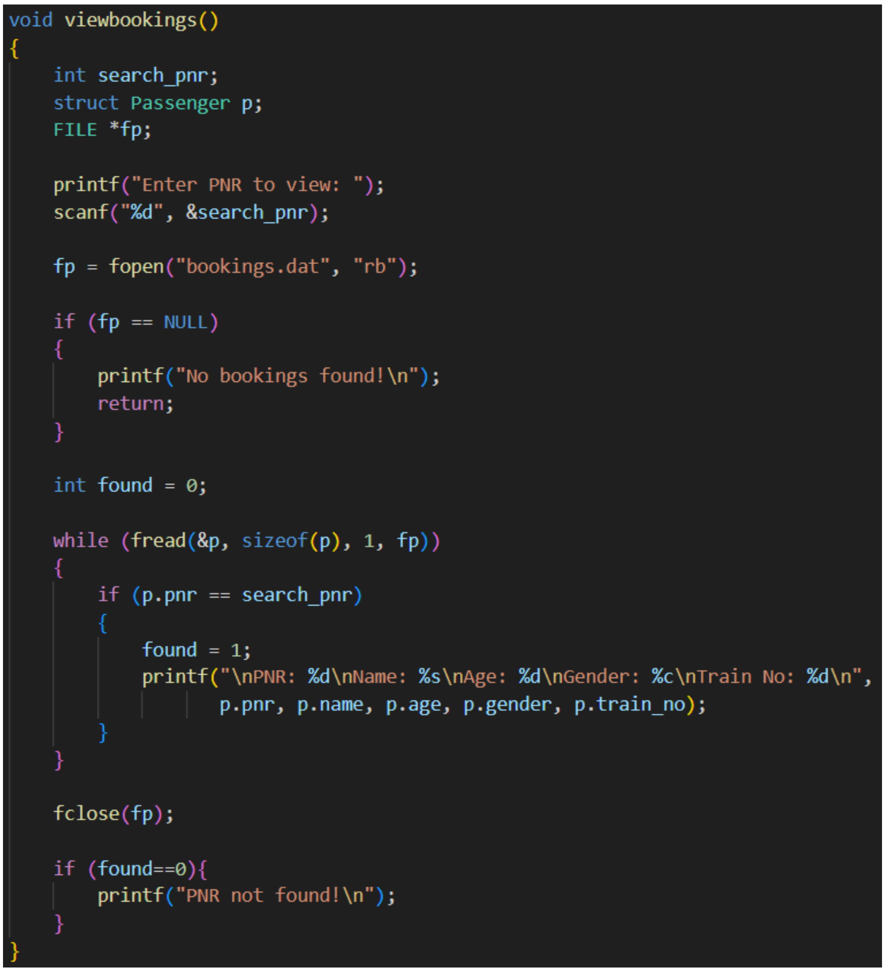
```c
void bookticket()
{
    struct Passenger p;
    FILE *fp;
    int train_no, index = -1, tickets, pnr;

    viewtrains();

    printf("\nEnter Train Number: ");
    scanf("%d", &train_no);

    // Find train
    for (int i = 0; i < 3; i++)
    {
        if (t[i].train_no == train_no){
            index = i;
        }
    }

    if (index == -1)
    {
        printf("Invalid Train Number!\n");
        return;
    }

    printf("Enter number of tickets: ");
    scanf("%d", &tickets);


    if (tickets > t[index].seats)
    {
        printf("Only %d seats available!\n", t[index].seats);
        return;
    }

    pnr = generatepnr();

    fp = fopen("bookings.dat", "ab");
```



```c
    for (int i = 1; i <= tickets; i++)
    {
        printf("\nEnter passenger %d details:\n", i);
        printf("Name(firstname only): ");
        scanf("%s", p.name);
        printf("Age: ");
        scanf("%d", &p.age);
        printf("Gender (M/F): ");
        scanf(" %c", &p.gender);

        p.pnr = pnr;
        p.train_no = train_no;

        fwrite(&p, sizeof(p), 1, fp);
        t[index].seats--;
    }

    fclose(fp);

    printf("\nBooking Successful! Your PNR is %d\n", pnr);
}
```

3. View Booking:

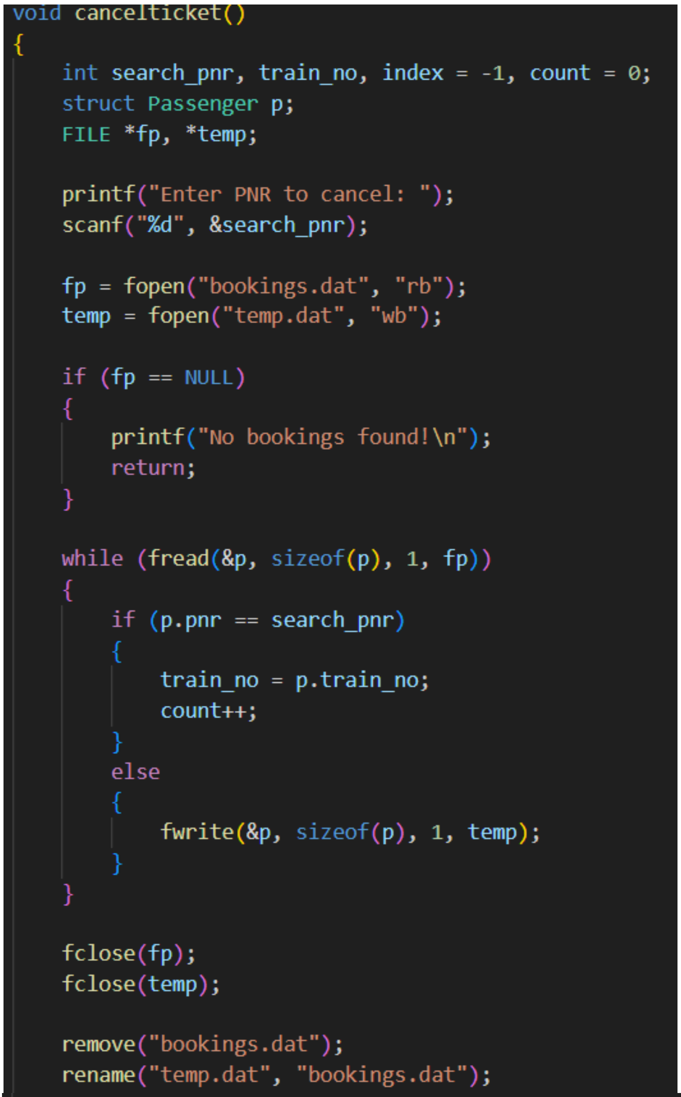
The “viewbookings” function was used to read all the records from the file and display the passengers matching the PNR entered by the user. Implemented using file operations and conditional statements.



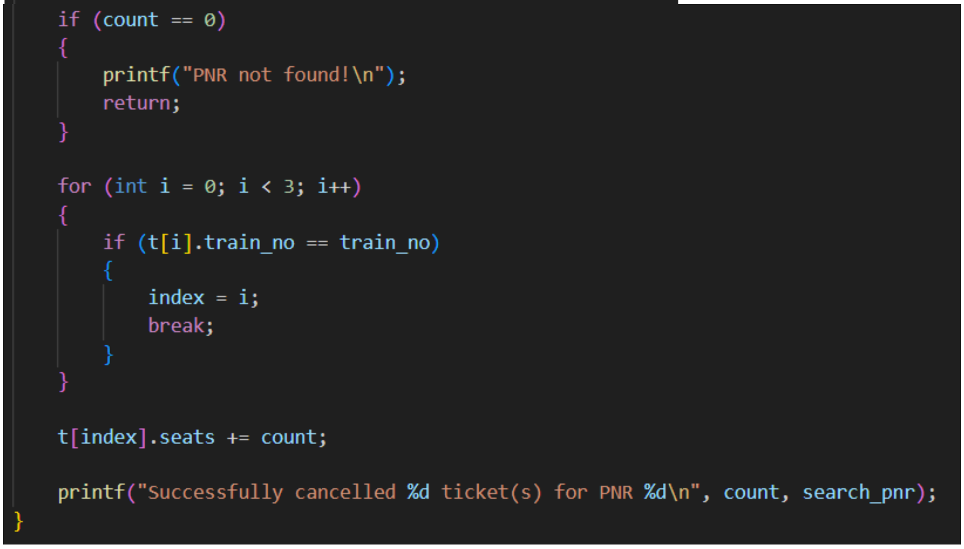
```c
void viewbookings()
{
    int search_pnr;
    struct Passenger p;
    FILE *fp;

    printf("Enter PNR to view: ");
    scanf("%d", &search_pnr);

    fp = fopen("bookings.dat", "rb");

    if (fp == NULL)
    {
        printf("No bookings found!\n");
        return;
    }

    int found = 0;

    while (fread(&p, sizeof(p), 1, fp))
    {
        if (p.pnr == search_pnr)
        {
            found = 1;
            printf("\nPNR: %d\nName: %s\nAge: %d\nGender: %c\nTrain No: %d\n",
                   p.pnr, p.name, p.age, p.gender, p.train_no);
        }
    }

    fclose(fp);

    if (found==0){
        printf("PNR not found!\n");
    }
}
```

4. Cancelation of Ticket:

The “canceltricket” function was used to search all the matching the PNR entered by the user, count how many tickets belong to that PNR, copies the rest od the record to a temporary file and replaced the old file. In the end it increases the number of seats by the number of seats which were cancelled.



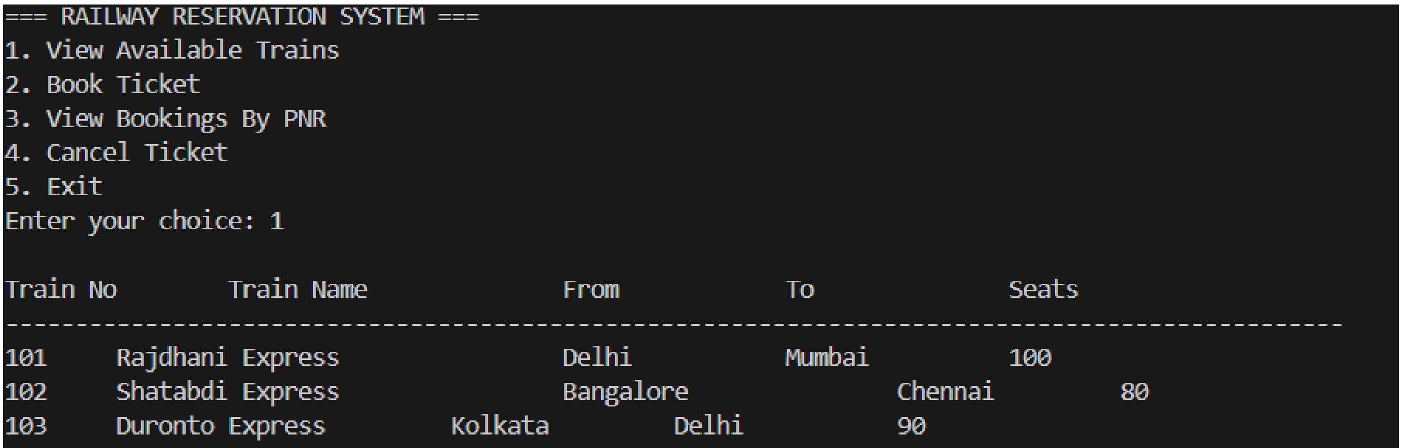
```c
void cancelticket()
{
    int search_pnr, train_no, index = -1, count = 0;
    struct Passenger p;
    FILE *fp, *temp;

    printf("Enter PNR to cancel: ");
    scanf("%d", &search_pnr);

    fp = fopen("bookings.dat", "rb");
    temp = fopen("temp.dat", "wb");

    if (fp == NULL)
    {
        printf("No bookings found!\n");
        return;
    }

    while (fread(&p, sizeof(p), 1, fp))
    {
        if (p.pnr == search_pnr)
        {
            train_no = p.train_no;
            count++;
        }
        else
        {
            fwrite(&p, sizeof(p), 1, temp);
        }
    }

    fclose(fp);
    fclose(temp);

    remove("bookings.dat");
    rename("temp.dat", "bookings.dat");
```



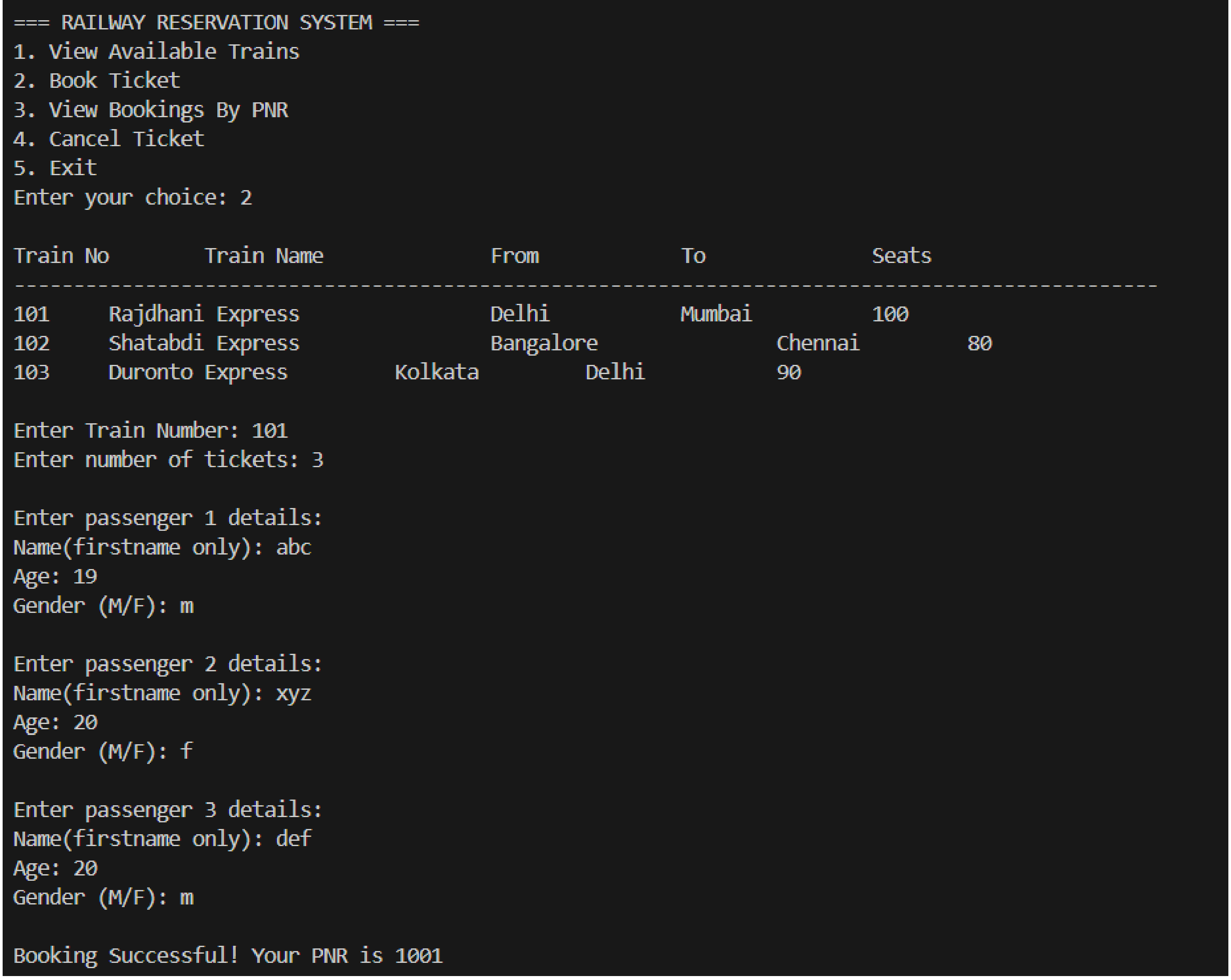
```c
    if (count == 0)
    {
        printf("PNR not found!\n");
        return;
    }

    for (int i = 0; i < 3; i++)
    {
        if (t[i].train_no == train_no)
        {
            index = i;
            break;
        }
    }

    t[index].seats += count;

    printf("Successfully cancelled %d ticket(s) for PNR %d\n", count, search_pnr);
}
```

Testing and Results

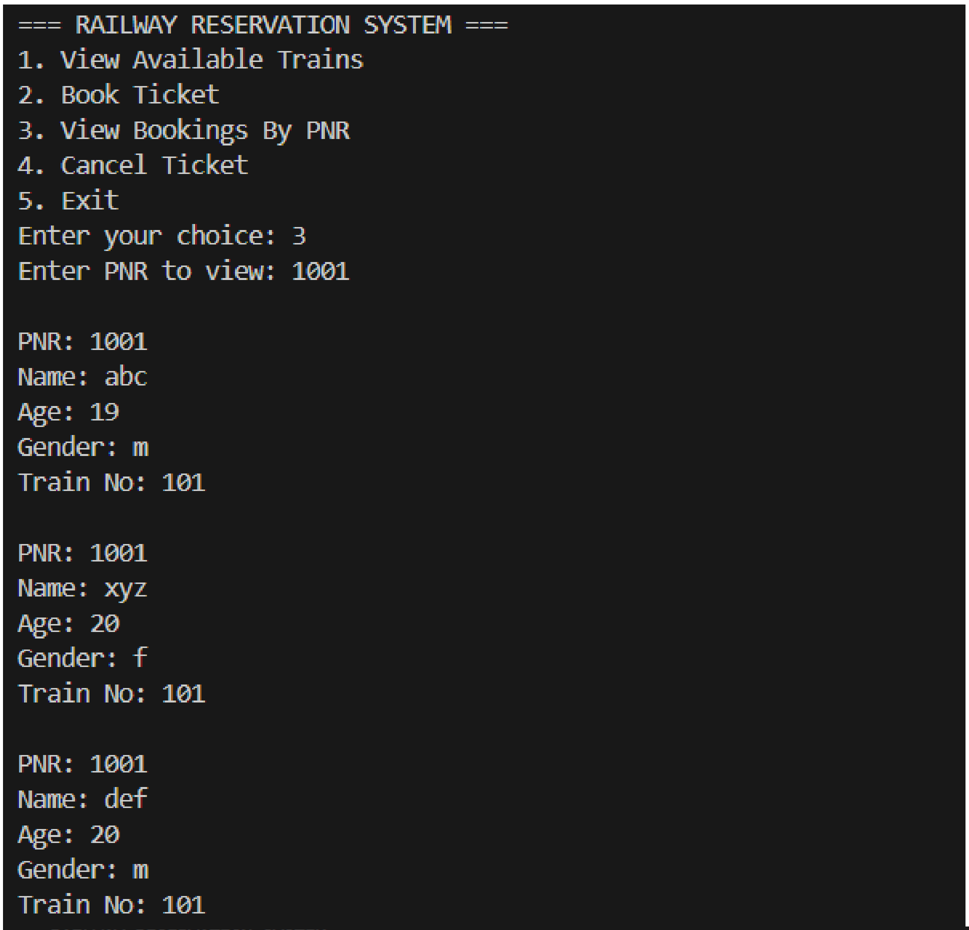
The code was test through various scenarios inputting valid and invalid enteries.



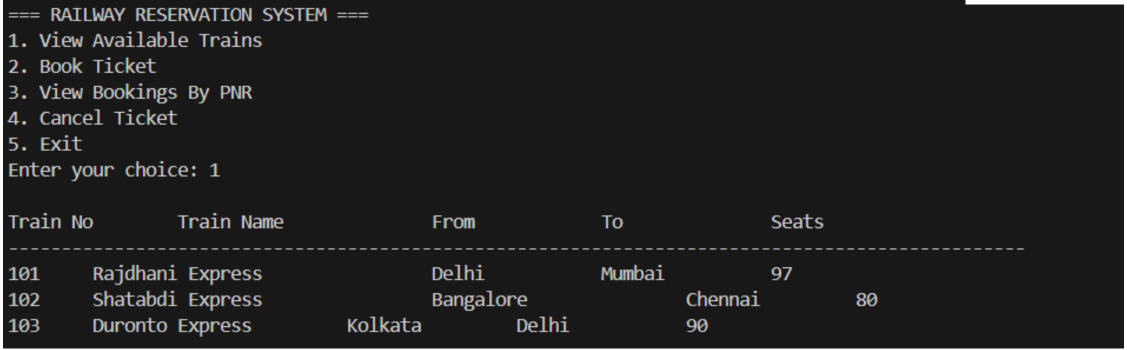
```
=== RAILWAY RESERVATION SYSTEM ===
1. View Available Trains
2. Book Ticket
3. View Bookings By PNR
4. Cancel Ticket
5. Exit
Enter your choice: 1

Train No        Train Name              From            To              Seats
----------------------------------------------------------------------------------------------------
101     Rajdhani Express                Delhi           Mumbai          100
102     Shatabdi Express                Bangalore               Chennai         80
103     Duronto Express         Kolkata         Delhi           90
```



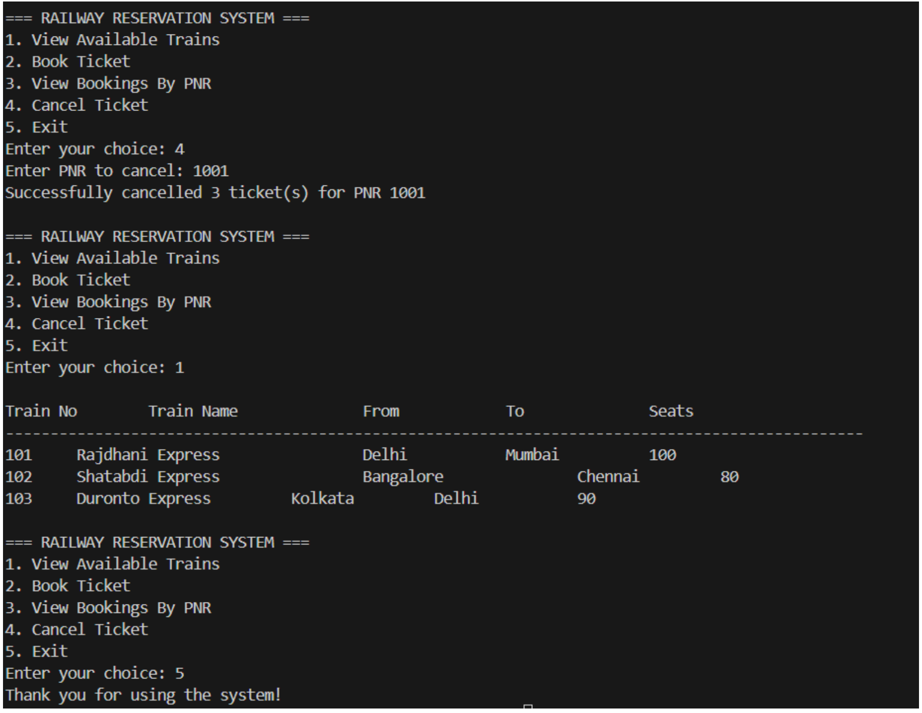
```
=== RAILWAY RESERVATION SYSTEM ===
1. View Available Trains
2. Book Ticket
3. View Bookings By PNR
4. Cancel Ticket
5. Exit
Enter your choice: 2

Train No        Train Name              From            To              Seats
----------------------------------------------------------------------------------------------------
101     Rajdhani Express                Delhi           Mumbai          100
102     Shatabdi Express                Bangalore               Chennai         80
103     Duronto Express         Kolkata         Delhi           90

Enter Train Number: 101
Enter number of tickets: 3

Enter passenger 1 details:
Name(firstname only): abc
Age: 19
Gender (M/F): m

Enter passenger 2 details:
Name(firstname only): xyz
Age: 20
Gender (M/F): f

Enter passenger 3 details:
Name(firstname only): def
Age: 20
Gender (M/F): m

Booking Successful! Your PNR is 1001
```



```
=== RAILWAY RESERVATION SYSTEM ===
1. View Available Trains
2. Book Ticket
3. View Bookings By PNR
4. Cancel Ticket
5. Exit
Enter your choice: 3
Enter PNR to view: 1001

PNR: 1001
Name: abc
Age: 19
Gender: m
Train No: 101

PNR: 1001
Name: xyz
Age: 20
Gender: f
Train No: 101

PNR: 1001
Name: def
Age: 20
Gender: m
Train No: 101
```



```
=== RAILWAY RESERVATION SYSTEM ===
1. View Available Trains
2. Book Ticket
3. View Bookings By PNR
4. Cancel Ticket
5. Exit
Enter your choice: 1

Train No        Train Name              From            To              Seats
------------------------------------------------------------------------------------------------
101     Rajdhani Express                Delhi           Mumbai          97
102     Shatabdi Express                Bangalore               Chennai         80
103     Duronto Express         Kolkata         Delhi           90
```



```
=== RAILWAY RESERVATION SYSTEM ===
1. View Available Trains
2. Book Ticket
3. View Bookings By PNR
4. Cancel Ticket
5. Exit
Enter your choice: 4
Enter PNR to cancel: 1001
Successfully cancelled 3 ticket(s) for PNR 1001

=== RAILWAY RESERVATION SYSTEM ===
1. View Available Trains
2. Book Ticket
3. View Bookings By PNR
4. Cancel Ticket
5. Exit
Enter your choice: 1

Train No        Train Name              From            To              Seats
------------------------------------------------------------------------------------------------
101     Rajdhani Express                Delhi           Mumbai          100
102     Shatabdi Express                Bangalore               Chennai         80
103     Duronto Express         Kolkata         Delhi           90

=== RAILWAY RESERVATION SYSTEM ===
1. View Available Trains
2. Book Ticket
3. View Bookings By PNR
4. Cancel Ticket
5. Exit
Enter your choice: 5
Thank you for using the system!
```

APPLICATIONS

- Similar systems can be made for bus, airline or any other transport system.
- Can also be used to develop cinema booking system
- Used to develop appointment booking system for hospitals.
- Can be used to make hotel and room reservation systems.
- Similar logic can be used in systems which require creating a unique booking ID.

Conclusion & Future Work

In conclusion, it has been a rewarding and enriching experience to complete the Major C Programming on Ticket Reservation System. Over the course of this project, we have learned a lot about C language and its functioning. Our understanding of the concepts we've learned in the classroom has been strengthened.

Future work may include:

- Larger train database.
- Allowing the user to enter both first name and surname.
- Payment integration.
- PNR generation with time stamp.
- GUI or web interface.

REFERENCES

1. Mr. Rahul Prasad Sir
2. Let us C by Yashwant Kanetkar
3. https://www.programiz.com/c-programming/online-compiler/
4. https://www.geeksforgeeks.org/c/c-programming-language/

APPENDIX

Source code:

```
#include <stdio.h>

struct Train
{
    int train_no;
    char train_name[50];
    char source[30];
    char destination[30];
    int seats;
};

struct Passenger
{
    int pnr;
    char name[50];
    int age;
    char gender;
    int train_no;
};

struct Train t[3] = {
    {101, "Rajdhani Express", "Delhi", "Mumbai", 100},
    {102, "Shatabdi Express", "Bangalore", "Chennai", 80},
    {103, "Duronto Express", "Kolkata", "Delhi", 90}};

int generatepnr();
void viewtrains();
void bookticket();
void viewbookings();
void cancelticket();

int main()
{
   int ch;

   do
   {
      printf("\n=== RAILWAY RESERVATION SYSTEM ===\n");
      printf("1. View Available Trains\n");
      printf("2. Book Ticket\n");
      printf("3. View Bookings By PNR\n");
      printf("4. Cancel Ticket\n");
      printf("5. Exit\n");
      printf("Enter your choice: ");
      scanf("%d", &ch);

      switch (ch)
      {
      case 1:
         viewtrains();
         break;
      case 2:
         bookticket();
         break;
      case 3:
         viewbookings();
         break;
      case 4:
         cancelticket();
```

```
            break;
        case 5:
            printf("Thank you for using the system!\n");
            break;
        default:
            printf("Invalid choice! Try again.\n");
        }
    } while (ch != 5);

    return 0;
}

void viewtrains()
{
    printf("\nTrain No\tTrain Name\t\tFrom\t\tTo\t\tSeats\n");
    printf("----------------------------------------------------------------------------------------------------\n");

    for (int i = 0; i < 3; i++)
    {
        printf("%d\t%s\t\t%s\t\t%s\t\t%d\n",
               t[i].train_no,
               t[i].train_name,
               t[i].source,
               t[i].destination,
               t[i].seats);
    }
}

int generatepnr()
{
    static int pnr = 1000;
    pnr++;
    return pnr;
}

void bookticket()
{
    struct Passenger p;
    FILE *fp;
    int train_no, index = -1, tickets, pnr;

    viewtrains();

    printf("\nEnter Train Number: ");
    scanf("%d", &train_no);

    for (int i = 0; i < 3; i++)
    {
        if (t[i].train_no == train_no){
            index = i;
        }
    }

    if (index == -1)
    {
        printf("Invalid Train Number!\n");
        return;
    }

    printf("Enter number of tickets: ");
    scanf("%d", &tickets);



    if (tickets > t[index].seats)
    {
        printf("Only %d seats available!\n", t[index].seats);
        return;
    }

    pnr = generatepnr();

    fp = fopen("bookings.dat", "ab");

    for (int i = 1; i <= tickets; i++)
    {
        printf("\nEnter passenger %d details:\n", i);
        printf("Name(firstname only): ");
        scanf("%s", p.name);
        printf("Age: ");
        scanf("%d", &p.age);
        printf("Gender (M/F): ");
        scanf(" %c", &p.gender);

        p.pnr = pnr;
        p.train_no = train_no;

        fwrite(&p, sizeof(p), 1, fp);
        t[index].seats--;
    }

    fclose(fp);

    printf("\nBooking Successful! Your PNR is %d\n", pnr);
}

void viewbookings()
```

```
{
    int search_pnr;
    struct Passenger p;
    FILE *fp;

    printf("Enter PNR to view: ");
    scanf("%d", &search_pnr);

    fp = fopen("bookings.dat", "rb");

    if (fp == NULL)
    {
        printf("No bookings found!\n");
        return;
    }

    int found = 0;

    while (fread(&p, sizeof(p), 1, fp))
    {
        if (p.pnr == search_pnr)
        {
            found = 1;
            printf("\nPNR: %d\nName: %s\nAge: %d\nGender: %c\nTrain No: %d\n",
                   p.pnr, p.name, p.age, p.gender, p.train_no);
        }
    }

    fclose(fp);

    if (found==0){
        printf("PNR not found!\n");
    }
}
void cancelticket()
{
    int search_pnr, train_no, index = -1, count = 0;
    struct Passenger p;
    FILE *fp, *temp;

    printf("Enter PNR to cancel: ");
    scanf("%d", &search_pnr);

    fp = fopen("bookings.dat", "rb");
    temp = fopen("temp.dat", "wb");

    if (fp == NULL)
    {
        printf("No bookings found!\n");
        return;
    }

    while (fread(&p, sizeof(p), 1, fp))
    {
        if (p.pnr == search_pnr)
        {
            train_no = p.train_no;
            count++;
        }
        else
        {
            fwrite(&p, sizeof(p), 1, temp);
        }
    }
    fclose(fp);
    fclose(temp);

    remove("bookings.dat");
    rename("temp.dat", "bookings.dat");

    if (count == 0)
    {
        printf("PNR not found!\n");
        return;
    }

    for (int i = 0; i < 3; i++)
    {
        if (t[i].train_no == train_no)
        {
            index = i;
            break;
        }
    }

    t[index].seats += count;

    printf("Successfully cancelled %d ticket(s) for PNR %d\n", count, search_pnr);
}
```