TRAIN TICKET RESERVATION SYSTEM

C Programming Project Report

Course Code: CSEG1041

School of Computer Science

University of Petroleum and Energy Studies

Submitted By: Shorya Verma

SAP ID: 590027164

Instructor: Dr. Prashant Trivedi

Date: December 01,2025

TABLE OF CONTENTS

1. ABSTRACT

The Train Ticket Reservation System is a console-based application developed in C programming language that simulates a real-world.

Railway reservation system . The system manages passengers booking, maintains confirmed and waiting lists, handles cancellation with automatic promotion from waiting list, and generates detailed ticket with completes train information.

The project demonstrates the practical application of fundamental C programming concepts including structures, arrays, enumeration & Modular Programming.

The system provides an efficient solution for managing limited seat availability while maintaining data integrity and user-friendly interaction.

Key Features:

- **Seat availability checking with real-time updates**
- **Automatic waiting list management**
- **Unique PNR generation**
- **Detailed Ticket printing with ticket information**
- **Automatic promotion from waiting list upon cancellation**
- **Input validation**

2. PROBLEM DEFINITION

Problem Statement:-

Railway reservation system need to efficiently manage limited seating capacity while handling multiple booking requests, cancellation, and maintaining waiting lists. Manual management of such system prone to errors, double bookings, and poor customer experience.

Objectives:-

1. **Automatic Booking Process:** creates a system that automatically manages seat allocation.
2. **Manages Waiting Lists**: Implement a queue based waiting list that automatically promotes passengers.
3. **Generate unique identifiers:** Creates unique PNR number for tracking.
4. **Provide Real-time Information**: Display current seat availability and booking status.
5. **Handle cancellations:** process cancellation and update waiting list accordingly.
6. **Store Complete Information**: Maintain comprehensive train and train and passenger details.

Scope

The system handles:-

- Maximum 5 confirmed seats
- Maximum 3 waiting list positions
- Train details: number, name, station from, station to, & Journey date
- Passenger details: name, PNR, and booking status
- Operations: booking, cancelation, ticket printing, and availability checking

Limitations

- Data is stored in memory (not persistent across program runs)
- Single train management per session.
- No seat class differentiation (all seats are equal)
- No payment processing simulation

3. SYSTEM DESIGN

Main Algorithm:

START

1. Initialize counters (confirmedCount = 0, waitingCount = 0)
2. Set starting PNR = 1001
3. REPEAT
   - Display menu
   - Get user choice
   - switch (choice)

     CASE 1: Display seat availability

     CASE 2: Book ticket

     CASE 3: Cancel Ticket

     CASE 4: Print ticket

     CASE 5: Exit program

   - END switch
4. UNTIL user choose exit

END

Booking algorithm

Function Book_Ticket(passenger name)

1. Generate new PNR
2. Get passenger name
3. Get train details (number, name, from, to, date)
4. IF confirmed seats available THEN
   - Add to confirmed list
   - Increment confirmedCount
   - Display “confirmed” message

5. ELSE IF waiting list available THEN
   - Add to waiting list
   - Increment waitingCount
   - Display “Waiting” message
6. ELSE
   - Display “No seat available”
7. Display PNR number

END FUNCTION

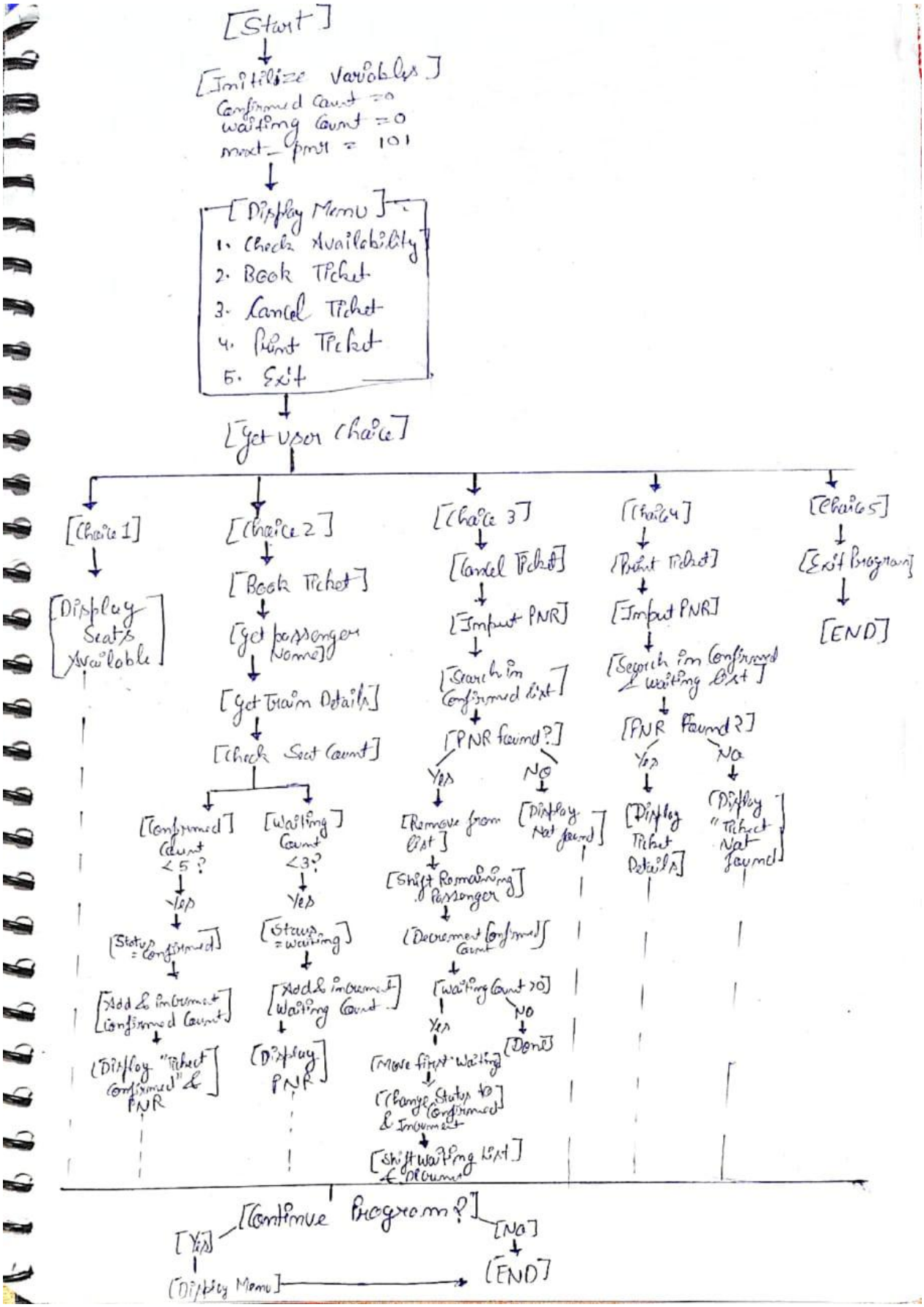
Cancellation Algorithm

FUNCTION cancelTicket(pnr)

1. Search for PNR in confirmed list
2. IF found
   - Remove passenger from confirmed list
   - Shift remaining passengers forward
   - Decrement confirmedCount
   - If waiting list not empty THEN
   - Move first waiting passenger to confirmed list
   - Update status to CONFIRMED
   - Shift remaining waiting passengers
   - Decrement waitingCount
   - Display promotion message
   - END IF
3. ELSE
   - Display “PNR” not found

END FUNCTION

FLOWCHART


[Start]
[Initialize Variables]
Confirmed Count = 0
Waiting Count = 0
next_pnr = 101
[Display Menu]
1. Check Availability
2. Book Ticket
3. Cancel Ticket
4. Print Ticket
5. Exit
[Get User Choice]
[Choice 1]
[Display Seats Available]
[Choice 2]
[Book Ticket]
[Get passenger Name]
[Get Train Details]
[Check Seat Count]
[Confirmed Count < 5 ?]
Yes
[Status = Confirmed]
[Add & Increment Confirmed Count]
[Display "Ticket Confirmed" & PNR]
[Waiting Count < 3?]
Yes
[Status = Waiting]
[Add & Increment Waiting Count]
[Display PNR]
[Choice 3]
[Cancel Ticket]
[Input PNR]
[Search in Confirmed List]
[PNR found?]
Yes
No
[Remove from List]
[Shift Remaining Passenger]
[Decrement Confirmed Count]
[Waiting Count > 0]
Yes
No
[Done]
[Move first Waiting]
[Change Status to Confirmed & Increment]
[Shift Waiting List & Decrement]
[Display Not found]
[Choice 4]
[Print Ticket]
[Input PNR]
[Search in Confirmed & Waiting List]
[PNR found?]
Yes
No
[Display Ticket Details]
[Display "Ticket Not found"]
[Choice 5]
[Exit Program]
[END]
[Continue Program?]
[Yes]
[Display Menu]
[NO]
[END]

Data structures:

Enumeration:

```
enum  Status { CONFIRMED, WAITING };
```

Train Structure:

```
typedef struct{
  int train_number;
  char train_name[50];
  char from[30];
  char to[30];
  char date[15];
} Train;
```

Passenger structure:

```
typedef struct {
   int pnr;
   char name[50];
   enum Status status;
   Train train_info;
} Passenger;
```

Global Arrays:

```
Passenger confirmed[MAX_SEATS];
Passenger waiting[MAX_WAITING];
```

4. IMPLEMENTATION DETAILS

FUNCTIONS:-

PNR Generation

```
// function for genrating unique PNR for each passenger
int gentrate_PNR(){
    return next_pnr++;
}
```

Purpose: generates unique PNR starting from 1001

Logic: Uses a static counter that increments with each call

Return: Integer PNR value

Seat Availability Check

```
// function for checking  availability of seats
int seatsAvailable() {
    return MAX_SEATS - confirmedCount;
}
```

Purpose: Calculates remaining confirmed seats

Logic: subtracts current count from maximum capacity

Return: number of available seats

Train Details Input

```c
// function for  taking input of train details
void Train_details(Train *train){
    printf("Enter Train Number: ");
    scanf("%d", &train->train_number);
    getchar();

    printf("Enter the Train Name: ");
    fgets(train->train_name, 50, stdin);
    train->train_name[strcspn(train->train_name, "\n")] = 0;

    printf("Enter the station from you take the train: ");
    fgets(train->from, 50, stdin);
    train->from[strcspn(train->from, "\n")] = 0;

    printf("Enter the station where you want to reach: ");
    fgets(train->to, 50, stdin);
    train->to[strcspn(train->to, "\n")] = 0;

    printf("Enter the Journey Date(DD/MM/YYYY): ");
    scanf("%s", train->date);

}
```

Purpose: Collect all train-related information

Buffer Handling: Uses getchar() after scanf to clear newline

String Input: Uses fgets() to allow spaces in names

Booking Function

```c
//function to book the tickets
void Book_Tickect(char name[]){
    Passenger p;
    p.pnr = gentrate_PNR();
    strcpy(p.name, name);

    Train_details(&p.train_info); // input of train details

    if(confirmedCount < MAX_SEATS){
        p.status = CONFIRMED;
        confirmed[confirmedCount++] = p;
        printf("Ticket Booked Successfully!\n");
    }
    else if(waitingCount < MAX_WAITING){
        p.status = WAITING;
        waiting[waitingCount++] = p;
        printf("Train Full .Added to waiting list.\n");
    }
    else{
        printf("Sorry! No seat and waiting list also full.\n");
        return;
    }
    printf("Your PNR: %d\n", p.pnr);
}
```

Logic Flow:

1. Generates unique PNR
2. Copy passenger name
3. Get train details
4. Heck confirmed seat availiabilty
5. If full, add to waiting list
6. If waiting list full, reject booking

Cancellation with auto-promotion

```c
// function to cancel the tickets
void cancelTicket(int pnr){
    int found=0;
    for(int i =0; i< confirmedCount; i++){
        if(confirmed[i].pnr == pnr){
            found=1;
            printf("Ticket with PNR %d cancelled.\n", pnr);

            for(int j = i; j < confirmedCount - 1; j++){   // shifting remaining pasengers
                confirmed[j] = confirmed[j+1];
            }
            confirmedCount--;

            // shifting the paasenger from waiting list to confirmed
            if(waitingCount > 0){
                confirmed[confirmedCount++] = waiting[0];
                confirmed[confirmedCount-1].status = CONFIRMED;

                // shifting of waiting list
                for(int j=0; j<waitingCount -1; j++){
                waiting[j] = waiting[j+1];
                }
                waitingCount--;
                printf("Seat confirmed for the passenger from waiting list");

            }
            break;
        }
    }
    if(!found){
        printf("passenger with PNR %d not found in confirmed list\n");
    }
}
```

Key Features:

- Linear search for PNR
- Array shifting to remove elements
- Automatic waiting list promotion
- Queue-based for waiting passengers

Structures

- Nested structures (Train inside passenger)
- Structure pointers for efficient passing
- Structure arrays for data storage

Arrays

- Fixed-size arrays for passengers storage
- Array manipulation (Insertion, deletion, shifting)
- Parallel arrays for confirmed and waiting list

String Handling

- strcpy() for string copying
- fgets() for space-inclusive input
- strcspn() for newline removal
- Buffer management with getchar()

Enumeration

- Status enumeration for type safety
- Cleaner code than integer flags

Modular programming

- Separate function for each operation
- Single Responsibility Principle
- Function reusability

Input Validation

- Buffer clearing after scanf operations
- Newline character removal from fgets input
- Boundary checking for array access
- PNR existence verification before operations

5. Testing & Results

Test case 1: Normal Booking

```
--------Train ticket Reservation System-----------
1. Check Seat Availability
2. Book Ticket
3. Cancel Ticket
4. Print Ticket
5. Exit
Enter choice: 2
Enter Passenger Name: John
Enter Train Number: 12345
Enter the Train Name: Rajdhani Express
Enter the station from you take the train: Delhi
Enter the station where you want to reach: Mumbai
Enter the Journey Date(DD/MM/YYYY): 15/12/2025
Ticket Booked Successfully!
Your PNR: 1001
```

Result: Passed

Test case 2: Waiting list Scenario (already 5 confirmed and book 6th ticket)

```
--------Train ticket Reservation System-----------
1. Check Seat Availability
2. Book Ticket
3. Cancel Ticket
4. Print Ticket
5. Exit
Enter choice: 2
Enter Passenger Name: Shiva
Enter Train Number: 1435
Enter the Train Name: Janta Express
Enter the station from you take the train: LUcknow
Enter the station where you want to reach: Dehradun
Enter the Journey Date(DD/MM/YYYY): 12/12/2025
Train Full .Added to waiting list.
Your PNR: 1006
```

Result: Passed

Test case 3: Train Full & waiting list also full after that passenger trying to book the ticket

```
--------Train ticket Reservation System-----------
1. Check Seat Availability
2. Book Ticket
3. Cancel Ticket
4. Print Ticket
5. Exit
Enter choice: 2
Enter Passenger Name: Shreya
Enter Train Number: 12001
Enter the Train Name: Shatabdi Express
Enter the station from you take the train: Delhi
Enter the station where you want to reach: Roorkee
Enter the Journey Date(DD/MM/YYYY): 12/12/2025
Sorry! No seat and waiting list also full.
```

Result: Passed

Test case 4: Cancellation and Promotion

```
--------Train ticket Reservation System-----------
1. Check Seat Availability
2. Book Ticket
3. Cancel Ticket
4. Print Ticket
5. Exit
Enter choice: 3
Enter PNR to cancel: 1003
Ticket with PNR 1003 cancelled.
Seat confirmed for the passenger from waiting list
```

Result: Passed

Test case 5: Ticket Printing

```
--------Train ticket Reservation System-----------
1. Check Seat Availability
2. Book Ticket
3. Cancel Ticket
4. Print Ticket
5. Exit
Enter choice: 4
Enter PNR to print: 1001

----------Train Ticket--------------
PNR: 1001
Name: John
Status: CONFIRMED
Train No.: 12345
Train Name: Rajdhani Express
From: Delhi
To: Mumbai
Date: 15/12/2025

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

Result: Passed

Test Case 6: Invalid PNR

```
--------Train ticket Reservation System-----------
1. Check Seat Availability
2. Book Ticket
3. Cancel Ticket
4. Print Ticket
5. Exit
Enter choice: 4
Enter PNR to print: 9999
No ticket found with this PNR: 9999
```

Result: Passed

Test Case 7: Exit

```
--------Train ticket Reservation System-----------
1. Check Seat Availability
2. Book Ticket
3. Cancel Ticket
4. Print Ticket
5. Exit
Enter choice: 5
Exiting...!
```

Result: Passed

6.Conclusion & Future Work

Achievements

The Train Reservation System Successfully demonstrates:

1. Core C Programming Concepts:

    - Complex data structure with nested structures
    - Efficient array manipulation and management
    - String handling with proper buffer management
    - Modular function design

2. Real world Problem Solving:
    - Simulated actual railway reservation logic
    - Handles edges cases and invalid inputs
    - Provides user friendly console interface
    -
3. Algorithm Implementation
    - Queue based waiting list
    - Dynamic seat allocation
    - Automatic status updates

Learning Outcomes

Through this project, I gained hands on experience in:

- Designing scalable data structures
- Managing memory efficient with fixed size arrays
- Handling user input with various C functions
- Debugging buffer related issues
- Writing modular , maintained code
- Testing and validation cpnplex logic flows

Future Enhancements

Short term Improvements: -

- **File persistence:** Save booking to file for data retention
- **Search functionality:** Search by name, train number, or date

Long term Improvements: -

- **Seat Classes:** Add AC/Non-AC, Sleeper/seater distinctions
- **Dynamic Pricing:** Implement pricing based on distance and class.
- **Birth allocation:** Assign specific seat/birth numbers
- **Memory Optimization:** Use dynamic memory allocation **(kind of Technical Improvements)**

Challenges Faced

1. Buffer Management: Handling newline characters after scanf required careful getchar()
2. Array Shifting: Implementing efficient element removal while maintaing order
3. Status Tracking: Maintaining consistency between confirmed and waiting lists

Conclusion

The Train Ticket Reservation system project successfully demonstrates the practical application of C programming fundamentals in solving real world problems. The system efficiently manages limited resources, handles multiple scenarios, and established a solid framework for future development. This project has strengthened my understanding of data structures and algorithm/ flowcharts.

REFERENCES

1. C Programming Language (2nd Edition)
   Brian W.Kernighan and Dennis M. Ritchie
   Prentice Hall,1988
2. GeeksforGeeks - C Programming
   https://www.geeksforgeeks.org/c/c-programming-language/

3. Indian Railways Reservation System (for domain understanding)
   https://www.irctc.co.in/nget/train-search

APPENDIX

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_SEATS 5       // maximum available seats for confirmation
#define MAX_WAITING 3     // maximum number of seats for waiting list

// status either confirmed or waiting for each passenger
enum Status { CONFIRMED, WAITING };

// structure for train information
typedef struct{
   int train_number;
   char train_name[50];
   char from[30];
   char to[30];
   char date[15];
} Train;

// to store passenger details we use  -> structure
typedef struct {
    int pnr;
    char name[50];
    enum Status status;
    Train train_info;
} Passenger;

Passenger confirmed[MAX_SEATS];   // array which stores confirmed passengers
Passenger waiting[MAX_WAITING];   // array which stores waiting passengers

// counters to check how many seats are filled
int confirmedCount = 0;
int waitingCount = 0;

int next_pnr = 1001;  // variable to genrate pnr

// function for genrating unique PNR for each passenger
int gentrate_PNR(){
    return next_pnr++;
}

// function for checking  availability of seats
int seatsAvailable() {
    return MAX_SEATS - confirmedCount;
}
```

```c
// function for  taking input of train details
void Train_details(Train *train){
    printf("Enter Train Number: ");
    scanf("%d", &train->train_number);
    getchar();

    printf("Enter the Train Name: ");
    fgets(train->train_name, 50, stdin);
    train->train_name[strcspn(train->train_name, "\n")] = 0;

    printf("Enter the station from you take the train: ");
    fgets(train->from, 50, stdin);
    train->from[strcspn(train->from, "\n")] = 0;

    printf("Enter the station where you want to reach: ");
    fgets(train->to, 50, stdin);
    train->to[strcspn(train->to, "\n")] = 0;

    printf("Enter the Journey Date(DD/MM/YYYY): ");
    scanf("%s", train->date);

}

//function to book the tickets
void Book_Tickect(char name[]){
    Passenger p;
    p.pnr = gentrate_PNR();
    strcpy(p.name, name);

    Train_details(&p.train_info); // input of train details

    if(confirmedCount < MAX_SEATS){
        p.status = CONFIRMED;
        confirmed[confirmedCount++] = p;
        printf("Ticket Booked Successfully!\n");
    }
    else if(waitingCount < MAX_WAITING){
        p.status = WAITING;
        waiting[waitingCount++] = p;
        printf("Train Full .Added to waiting list.\n");
    }
    else{
        printf("Sorry! No seat and waiting list also full.\n");
        return;
    }
    printf("Your PNR: %d\n", p.pnr);
}
```

```c
// function for printing the the ticket
void printTicket(int pnr){
    for(int i =0; i < confirmedCount; i++){
        if(confirmed[i].pnr == pnr){
            printf("\n----------Train Ticket--------------\n");
            printf("PNR: %d\n", confirmed[i].pnr);
            printf("Name: %s\n", confirmed[i].name);
            printf("Status: CONFIRMED\n");
            printf("Train No.: %d\n", confirmed[i].train_info.train_number);
            printf("Train Name: %s\n", confirmed[i].train_info.train_name);
            printf("From: %s\n", confirmed[i].train_info.from);
            printf("To: %s\n", confirmed[i].train_info.to);
            printf("Date: %s\n", confirmed[i].train_info.date);
            printf("\n-----------------------------------\n");
            return;
        }
    }
    // // for(int i =0; i < waitingCount; i++){
        if(waiting[i].pnr == pnr){
            printf("\n----------Train Ticket--------------\n");
            printf("PNR: %d\n", waiting[i].pnr);
            printf("Name: %s\n", waiting[i].name);
            printf("Status: WAITING\n");
            printf("Train No.: %d\n", waiting[i].train_info.train_number);
            printf("Train Name: %s\n", waiting[i].train_info.train_name);
            printf("From: %s\n", waiting[i].train_info.from);
            printf("To: %s\n", waiting[i].train_info.to);
            printf("Date: %s\n", waiting[i].train_info.date);
            printf("\n-----------------------------------\n");
            return;
        }
    }
    printf("No ticket found with this PNR: %d\n", pnr);

}

// function to cancel the tickets
void cancelTicket(int pnr){
    int found=0;
    for(int i =0; i< confirmedCount; i++){
        if(confirmed[i].pnr == pnr){
            found=1;
            printf("Ticket with PNR %d cancelled.\n", pnr);

            for(int j = i; j < confirmedCount - 1; j++){   // shifting
remaining pasengers
                confirmed[j] = confirmed[j+1];
            }
```

```
            confirmedCount--;

            // shifting the passenger from waiting list to confirmed
            if(waitingCount > 0){
                confirmed[confirmedCount++] = waiting[0];
                confirmed[confirmedCount-1].status = CONFIRMED;

                // shifting of waiting list
                for(int j=0; j<waitingCount -1; j++){
                waiting[j] = waiting[j+1];
                }
                waitingCount--;
                printf("Seat confirmed for the passenger from waiting list");

            }
            break;
        }
    }
    if(!found){
        printf("passenger with PNR %d not found in confirmed list\n");
    }
}

int main(){
    int choice, pnr;
    char name[60];

    while(1){
        printf("\n--------Train ticket Reservation System-----------\n");
        printf("1. Check Seat Availability\n");
        printf("2. Book Ticket\n");
        printf("3. Cancel Ticket\n");
        printf("4. Print Ticket\n");
        printf("5. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch(choice){
            case 1 :
                printf("Seat Available: %d\n", seatsAvailable());
                break;
            case 2 :
                printf("Enter Passenger Name: ");
                scanf("%s", name);
                Book_Tickect(name);
                break;
            case 3 :
                printf("Enter PNR to cancel: ");
```

```
                scanf("%d", &pnr);
                cancelTicket(pnr);
                break;

            case 4 :
                printf("Enter PNR to print: ");
                scanf("%d", &pnr);
                printTicket(pnr);
                break;
            case 5 :
                printf("Exiting...!\n");
                exit(0);
            default:
                printf("Invalid Choice!\n");

        }
    }
    return 0;
}
```