# KRISHNA PRASAD Railway reservation system 2541032

## 1.PROJECT TITLE

Digital Train Reservation System using C Programming

## 2. PROJECT DESCRIPTION

The Railway Reservation System in C is a console-based mini project that simulates the process of train ticket booking and management. It provides users with options to display train details, book tickets, view passenger list, and cancel tickets in a simple and user-friendly menu-driven format.

The project is implemented using structures, arrays, and functions to store and manage train as well as passenger information. Each booking generates a unique PNR number and assigns a seat number automatically while reducing the available seat count of the selected train. In case of cancellation, the ticket is removed from the passenger list and the seat availability of the respective train is updated.

This system can handle up to 5 trains and 100 passengers at a time. It is mainly developed for learning purposes and to demonstrate the use of C programming concepts such as input/output handling, loops, conditional statements, arrays, and modular programming. Though it does not store data permanently, it provides a strong foundation for building more advanced reservation systems with features like file storage, waiting list management, and payment options.

## 3. RESEARCH & BACKGROUND STUDY

✓ Railway Reservation in Real Life

Earlier reservation was done manually through registers.

Manual system was slow, error-prone, and difficult to manage for many passengers.

Today's railway systems use computers and online platforms for faster, accurate, and reliable booking.

✓Need for Computerized System

To manage seat availability efficiently.

To generate unique PNR numbers for each passenger.

To handle booking and cancellation without confusion.

✓ Role of C Programming in this Project

Structures: store train and passenger details.

Arrays: manage multiple trains and passengers.

Functions: perform booking, cancellation, and display operations.

Menu-driven interface: makes the system user-friendly.

✓Outcome of Study

Project shows how real-world reservation systems work in a simplified way.

Provides foundation for advanced systems with file handling and databases.

## 4. SYSTEM DESIGN & IMPLEMENTATION

### ✓ Algorithm: Railway Reservation System

1. Start

2. Initialize train details (train no, name, source, destination, seats).

3. Show Main Menu:

1. Display Trains

2. Book Ticket

3. View Passengers

4. Cancel Ticket

5. Exit

## ✦ Implementation

1. Start Program → Load Train Data.

```
// Structure for train details
struct Train {
    int train_no;
    char name[50];
    char source[30];
    char destination[30];
    int seats_available;
};
```

2. Show Menu (Display, Book, View, Cancel, Exit).

```
while (1) {
    printf("\n===== Railway Reservation
        System =====\n");
    printf("1. Display Train Details\n");
    printf("2. Book Ticket\n");
    printf("3. View Passengers\n");
    printf("4. Cancel Ticket\n");
    printf("5. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);
```

3. Book Ticket → Search Train → Validate Seats → Input Passenger → Generate PNR → Save Data (File).

```c
// Book ticket
void bookTicket() {
    int tno;
    printf("\nEnter Train Number to book
        ticket: ");
    scanf("%d", &tno);

    // Search for train
    int found = -1;
    for (int i = 0; i < MAX_TRAINS; i++) {
        if (trains[i].train_no == tno) {
            found = i;
            break;
        }
    }

    if (found == -1) {
        printf("Train not found!\n");
        return;
    }

    if (trains[found].seats_available <= 0) {
        printf("No seats available on this
            train!\n");
        return;
    }
```

4. Cancel Ticket → Input PNR → Search → Delete Passenger → Update Seats → Save Changes.

```
158  // Cancel ticket
159  void cancelTicket() {
160      int pnr;
161      printf("\nEnter PNR to cancel ticket: ");
162      scanf("%d", &pnr);
163
164      int found = -1;
165      for (int i = 0; i < passenger_count; i++)
             {
166          if (passengers[i].pnr == pnr) {
167              found = i;
168              break;
169          }
170      }
171
172      if (found == -1) {
173          printf("Ticket with PNR %d not
                 found!\n", pnr);
174          return;
175      }
```

5. View → Print Passenger List.

```c
// View passenger details
void viewPassengers() {
    if (passenger_count == 0) {
        printf("\nNo passengers booked yet.\n"
            );
        return;
    }
    printf("\nPassenger List:\n");
    printf("PNR\tName\t\tAge\tTrain No\tSeat
        No\n");
    for (int i = 0; i < passenger_count; i++)
        {
        printf("%d\t%-15s%d\t%d\t\t%d\n",
                passengers[i].pnr,
                passengers[i].name,
                passengers[i].age,
                passengers[i].train_no,
                passengers[i].seat_no);
    }
}
```

6. Exit → Show "Thank You".

- **c codes**

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

#define MAX_TRAINS 5
```

```
#define MAX_SEATS 50
#define MAX_PASSENGERS 100

// Structure for train details
struct Train {
    int train_no;
    char name[50];
    char source[30];
    char destination[30];
    int seats_available;
};
// Structure for passenger details
struct Passenger {
    int pnr;
    char name[50];
    int age;
    int train_no;
    int seat_no;
};
```

```
// Global arrays
struct Train trains[MAX_TRAINS];
struct Passenger passengers[MAX_PASSENGERS];
int passenger_count = 0;
int pnr_counter = 1000;

// Function prototypes
void initializeTrains();
void displayTrains();
void bookTicket();
void viewPassengers();
void cancelTicket();

int main() {
    int choice;
    initializeTrains();

    while (1) {
```

```
        printf("\n===== Railway Reservation System
=====\n");
        printf("1. Display Train Details\n");
        printf("2. Book Ticket\n");
        printf("3. View Passengers\n");
        printf("4. Cancel Ticket\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                displayTrains();
                break;
            case 2:
                bookTicket();
                break;
            case 3:
                viewPassengers();
```

```
            break;
        case 4:
            cancelTicket();
            break;
        case 5:
            printf("Thank you for using Railway
Reservation System!\n");
            exit(0);
        default:
            printf("Invalid choice! Try again.\n");
      }
   }
   return 0;
}

// Initialize train details
void initializeTrains() {
   trains[0] = (struct Train){101, "Deccan Express",
"Mumbai", "Pune", MAX_SEATS};
```

```
    trains[1] = (struct Train){102, "Shatabdi Express", "Mumbai", "Delhi", MAX_SEATS};
    trains[2] = (struct Train){103, "Konkan Express", "Mumbai", "Goa", MAX_SEATS};
    trains[3] = (struct Train){104, "Rajdhani Express", "Delhi", "Kolkata", MAX_SEATS};
    trains[4] = (struct Train){105, "Duronto Express", "Pune", "Nagpur", MAX_SEATS};
}

// Display train details
void displayTrains() {
    printf("\nAvailable Trains:\n");
    printf("Train No\tName\t\t\tSource\t\tDestination\tSeats Available\n");
    for (int i = 0; i < MAX_TRAINS; i++) {
        printf("%d\t\t%-20s%-15s%-15s%d\n",
               trains[i].train_no,
               trains[i].name,
```

```
            trains[i].source,
            trains[i].destination,
            trains[i].seats_available);
    }
}

// Book ticket
void bookTicket() {
    int tno;
    printf("\nEnter Train Number to book ticket: ");
    scanf("%d", &tno);

    // Search for train
    int found = -1;
    for (int i = 0; i < MAX_TRAINS; i++) {
        if (trains[i].train_no == tno) {
            found = i;
            break;
        }
```

```
    }

    if (found == -1) {
        printf("Train not found!\n");
        return;
    }

    if (trains[found].seats_available <= 0) {
        printf("No seats available on this train!\n");
        return;
    }

    // Passenger details
    struct Passenger p;
    p.pnr = pnr_counter++;
    p.train_no = tno;
    p.seat_no = MAX_SEATS -
trains[found].seats_available + 1;
```

```
    printf("Enter passenger name: ");
    scanf(" %[^\n]", p.name);
    printf("Enter age: ");
    scanf("%d", &p.age);

    passengers[passenger_count++] = p;
    trains[found].seats_available--;

    printf("Ticket booked successfully! PNR: %d, Seat
No: %d\n", p.pnr, p.seat_no);
}

// View passenger details
void viewPassengers() {
    if (passenger_count == 0) {
        printf("\nNo passengers booked yet.\n");
        return;
    }
    printf("\nPassenger List:\n");
```

```
    printf("PNR\tName\t\tAge\tTrain No\tSeat No\n");
    for (int i = 0; i < passenger_count; i++) {
        printf("%d\t%-15s%d\t%d\t\t%d\n",
               passengers[i].pnr,
               passengers[i].name,
               passengers[i].age,
               passengers[i].train_no,
               passengers[i].seat_no);
    }
}

// Cancel ticket
void cancelTicket() {
    int pnr;
    printf("\nEnter PNR to cancel ticket: ");
    scanf("%d", &pnr);

    int found = -1;
```

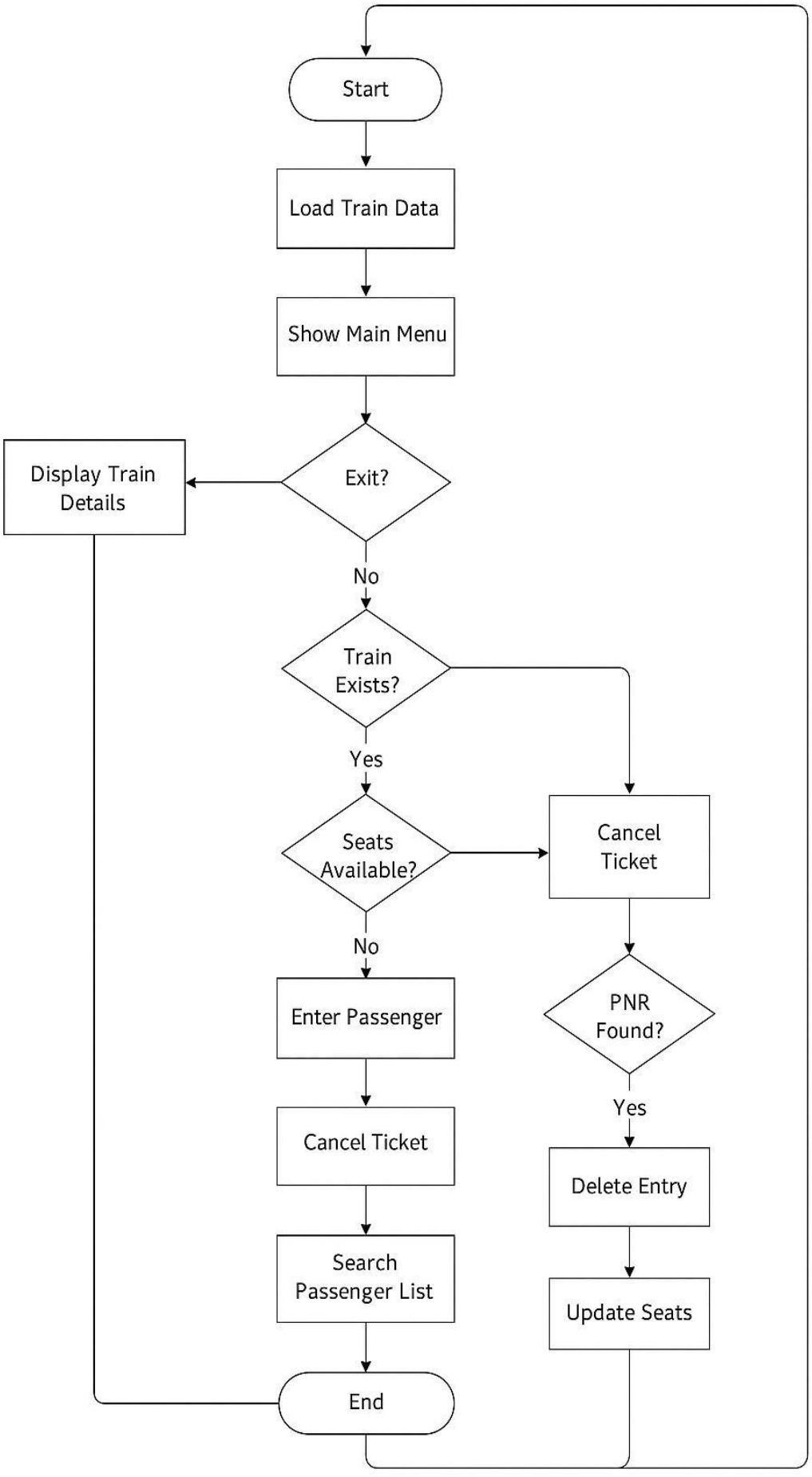
```
    for (int i = 0; i < passenger_count; i++) {
        if (passengers[i].pnr == pnr) {
            found = i;
            break;
        }
    }

    if (found == -1) {
        printf("Ticket with PNR %d not found!\n", pnr);
        return;
    }

    // Update train seat availability
    for (int i = 0; i < MAX_TRAINS; i++) {
        if (trains[i].train_no ==
passengers[found].train_no) {
            trains[i].seats_available++;
            break;
        }
```

```
    }

    printf("Ticket for %s (PNR: %d) cancelled
successfully.\n",
           passengers[found].name,
passengers[found].pnr);

    // Remove passenger
    for (int i = found; i < passenger_count - 1; i++) {
        passengers[i] = passengers[i + 1];
    }
    passenger_count--;
}
```

✓ **Testing:---**

```
===== Railway Reservation System =====
1. Display Train Details
2. Book Ticket
3. View Passengers
4. Cancel Ticket
5. Exit
Enter your choice:
```

✓ **Flow Chart:----**


Start
Load Train Data
Show Main Menu
Exit?
Display Train Details
No
Train Exists?
Yes
Seats Available?
No
Enter Passenger
Cancel Ticket
Search Passenger List
End
Cancel Ticket
PNR Found?
Yes
Delete Entry
Update Seats

## Conclusion:

The Railway Reservation System project in C language provides a simple and effective way to manage train ticket booking. It allows users to reserve tickets, cancel them, and check the availability of seats in an organized manner. The system helps in reducing manual work, errors, and confusion by using a structured program-based approach. Through this project, we also understand the use of important C concepts like structures, functions, arrays, and file handling. It improves efficiency and provides a user-friendly environment for railway ticket management. Overall, this project demonstrates how programming can be used to solve real-life problems in an efficient way.