

Submitted by:

P.shashank

24J41A1250

Under the guidance of

M.Anusha Madam

**Abstract**

The Railway Reservation System is a console-based C program that manages train ticket bookings efficiently. It allows users to book and cancel tickets, view the passenger list, and maintain a waiting list when seats are full.

The system uses linked lists for booked passengers and queues for the waiting list, demonstrating practical use of data structures. It automates seat allocation and waiting list management, providing a simple and functional simulation of real-world railway booking operations.

**Objective**

To design and implement a simple railway reservation system using C that allows users to:

* Book tickets
* Cancel tickets
* View passenger list
* Manage waiting list

**Modules**

1. **Main Menu** – Displays options and handles user input.
2. **Book Ticket** – Adds passengers to booked list or waiting list if seats are full.
3. **Cancel Ticket** – Removes passenger and moves first waiting passenger to booked list.
4. **View Booked List** – Shows all booked passengers with details.
5. **View Waiting List** – Displays passengers in the waiting list.
6. **Queue Operations** – Handles enqueue and dequeue for waiting list.

**Programming Concepts Used**

* **Structures** – To store passenger details.
* **Linked List** – For booked passengers.
* **Queue** – For waiting list (FIFO).
* **Dynamic Memory Allocation** – Using malloc() and free().
* **File Handling (optional)** – To save/load data.
* **Basic I/O, Loops & Conditionals** – For user interaction and logic flow

source code:  
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_SEATS 5

struct Passenger {

int id;

char name[50];

int age;

char gender[10];

int seatNo;

struct Passenger \*next;

};

struct Passenger \*booked = NULL;

struct Passenger \*waiting = NULL;

int bookedCount = 0, nextID = 1;

// Function prototypes

void bookTicket();

void cancelTicket();

void viewBooked();

void viewWaiting();

void enqueue(struct Passenger \*p);

struct Passenger\* dequeue();

int main() {

int choice;

while (1) {

printf("\n=== RAILWAY RESERVATION SYSTEM ===\n");

printf("1. Book Ticket\n2. Cancel Ticket\n3. View Booked List\n4. View Waiting List\n5. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1: bookTicket(); break;

case 2: cancelTicket(); break;

case 3: viewBooked(); break;

case 4: viewWaiting(); break;

case 5: exit(0);

default: printf("Invalid choice!\n");

}

}

}

void bookTicket() {

struct Passenger \*p = (struct Passenger\*)malloc(sizeof(struct Passenger));

p->id = nextID++;

printf("Enter Name: "); scanf("%s", p->name);

printf("Enter Age: "); scanf("%d", &p->age);

printf("Enter Gender: "); scanf("%s", p->gender);

p->next = NULL;

if (bookedCount < MAX\_SEATS) {

p->seatNo = bookedCount + 1;

p->next = booked;

booked = p;

bookedCount++;

printf("Ticket Booked! Seat No: %d (Passenger ID: %d)\n", p->seatNo, p->id);

} else {

enqueue(p);

printf("All seats full! Added to waiting list.\n");

}

}

void cancelTicket() {

int id;

printf("Enter Passenger ID to cancel: ");

scanf("%d", &id);

struct Passenger \*temp = booked, \*prev = NULL;

while (temp != NULL && temp->id != id) {

prev = temp;

temp = temp->next;

}

if (temp == NULL) {

printf("Passenger not found!\n");

return;

}

if (prev == NULL) booked = temp->next;

else prev->next = temp->next;

printf("Ticket cancelled for %s (ID %d)\n", temp->name, temp->id);

free(temp);

bookedCount--;

// Move from waiting list if available

struct Passenger \*wait = dequeue();

if (wait != NULL) {

wait->seatNo = bookedCount + 1;

wait->next = booked;

booked = wait;

bookedCount++;

printf("Moved %s from waiting list to seat %d\n", wait->name, wait->seatNo);

}

}

void viewBooked() {

printf("\n--- Booked Passengers ---\n");

struct Passenger \*temp = booked;

while (temp != NULL) {

printf("ID:%d | Name:%s | Age:%d | Seat:%d\n", temp->id, temp->name, temp->age, temp->seatNo);

temp = temp->next;

}

}

void viewWaiting() {

printf("\n--- Waiting List ---\n");

struct Passenger \*temp = waiting;

while (temp != NULL) {

printf("ID:%d | Name:%s | Age:%d\n", temp->id, temp->name, temp->age);

temp = temp->next;

}

}

void enqueue(struct Passenger \*p) {

if (waiting == NULL) waiting = p;

else {

struct Passenger \*temp = waiting;

while (temp->next != NULL) temp = temp->next;

temp->next = p;

}

}

struct Passenger\* dequeue() {

if (waiting == NULL) return NULL;

struct Passenger \*temp = waiting;

waiting = waiting->next;

temp->next = NULL;

return temp;

}

**Conclusion**

The **Railway Reservation System** in C efficiently demonstrates the use of fundamental data structures like **linked lists** and **queues** to manage train ticket bookings and waiting lists. The system allows users to **book, cancel, and view tickets** easily, automating seat allocation and waiting list management.

This project provides practical exposure to **dynamic memory allocation, structures, and file handling**, while simulating a real-world railway reservation scenario. It enhances understanding of data structure concepts and their applications in solving real-life problems.