# TITLE : Railway Ticket Booking Simulation

## Abstract

This project simulates a simple Railway Ticket Booking System using basic Data Structures in C. It allows users to book and cancel tickets while maintaining a waiting list automatically. The system demonstrates how linked lists and queues can manage real-world problems efficiently.

## Introduction

The Railway Ticket Booking Simulation aims to provide a basic understanding of dynamic data handling. It manages limited seat bookings and automatically transfers waiting passengers when seats become available. This project uses linked lists for confirmed bookings and queues for managing waiting passengers.

## Existing System

The existing manual system for railway ticket booking involves human errors and inefficiency. There is no automated handling of waiting lists or real-time updates when cancellations occur.

## Proposed System

The proposed automated system efficiently handles bookings and waiting lists using data structures. It automatically allocates available seats to waiting passengers when cancellations occur, reducing manual work.

## Software Requirements

Operating System: Windows / Linux  
Compiler: GCC / Turbo C  
Language: C  
IDE: Code::Blocks or Dev C++

## Hardware Requirements

Processor: 1 GHz or higher  
RAM: 512 MB  
Storage: 10 MB free space  
Input Device: Keyboard  
Output Device: Monitor

## C Program Code

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX 5  // Maximum seats

typedef struct node {

    int id;

    char name[20];

    struct node \*next;

} Node;

Node \*booked = NULL, \*waitFront = NULL, \*waitRear = NULL;

int seatCount = 0, idCounter = 1;

// Function prototypes

void book();

void cancel();

void showBooked();

void showWaiting();

void searchPassenger();

void showSummary();

void freeAll();

int main() {

    int ch;

    do {

        printf("\n========= TICKET BOOKING SYSTEM =========\n");

        printf("1. Book Ticket\n");

        printf("2. Cancel Ticket\n");

        printf("3. Show Booked Passengers\n");

        printf("4. Show Waiting List\n");

        printf("5. Search Passenger\n");

        printf("6. Show Summary\n");

        printf("7. Exit\n");

        printf("Enter choice: ");

        scanf("%d", &ch);

        switch(ch) {

            case 1: book(); break;

            case 2: cancel(); break;

            case 3: showBooked(); break;

            case 4: showWaiting(); break;

            case 5: searchPassenger(); break;

            case 6: showSummary(); break;

            case 7: freeAll(); printf("Exiting... All memory freed.\n"); break;

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

        }

    } while(ch != 7);

    return 0;

}

void book() {

    char name[20];

    printf("Enter passenger name: ");

    scanf("%s", name);

    Node \*newNode = malloc(sizeof(Node));

    newNode->id = idCounter++;

    strcpy(newNode->name, name);

    newNode->next = NULL;

    if (seatCount < MAX) {

        newNode->next = booked;

        booked = newNode;

        seatCount++;

        printf("✅ Ticket booked successfully! ID: %d\n", newNode->id);

    } else {

        if (waitRear == NULL)

            waitFront = waitRear = newNode;

        else {

            waitRear->next = newNode;

            waitRear = newNode;

        }

        printf("⚠️  All seats full! Added to waiting list. ID: %d\n", newNode->id);

    }

}

void cancel() {

    if (!booked) { printf("❌ No bookings to cancel.\n"); return; }

    int id;

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

    scanf("%d", &id);

    Node \*temp = booked, \*prev = NULL;

    while (temp && temp->id != id) {

        prev = temp;

        temp = temp->next;

    }

    if (!temp) {

        printf("❌ ID not found in booked list.\n");

        return;

    }

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

    else booked = temp->next;

    free(temp);

    seatCount--;

    printf("✅ Ticket cancelled successfully!\n");

    // Move first waiting passenger to booked list

    if (waitFront) {

        Node \*moved = waitFront;

        waitFront = waitFront->next;

        if (!waitFront) waitRear = NULL;

        moved->next = booked;

        booked = moved;

        seatCount++;

        printf("🔁 Passenger '%s' (ID:%d) moved from waiting list to booked list.\n",

               moved->name, moved->id);

    }

}

void showBooked() {

    Node \*t = booked;

    if (!t) { printf("No booked passengers.\n"); return; }

    printf("\n🟩 Booked Passengers:\n");

    while (t) {

        printf("  ID: %-3d | Name: %-10s\n", t->id, t->name);

        t = t->next;

    }

}

void showWaiting() {

    Node \*t = waitFront;

    if (!t) { printf("No waiting passengers.\n"); return; }

    printf("\n🟨 Waiting List:\n");

    while (t) {

        printf("  ID: %-3d | Name: %-10s\n", t->id, t->name);

        t = t->next;

    }

}

void searchPassenger() {

    if (!booked && !waitFront) {

        printf("No passengers found.\n");

        return;

    }

    int id;

    printf("Enter passenger ID to search: ");

    scanf("%d", &id);

    Node \*t = booked;

    while (t) {

        if (t->id == id) {

            printf("✅ Passenger found in Booked List: %s (ID:%d)\n", t->name, t->id);

            return;

        }

        t = t->next;

    }

    t = waitFront;

    while (t) {

        if (t->id == id) {

            printf("⚠️ Passenger found in Waiting List: %s (ID:%d)\n", t->name, t->id);

            return;

        }

        t = t->next;

    }

    printf("❌ Passenger not found.\n");

}

void showSummary() {

    int waitCount = 0;

    Node \*t = waitFront;

    while (t) { waitCount++; t = t->next; }

    printf("\n===== SUMMARY =====\n");

    printf("Total Booked Seats: %d / %d\n", seatCount, MAX);

    printf("Total Waiting Passengers: %d\n", waitCount);

    if (waitFront)

        printf("Next Waiting Passenger: %s (ID:%d)\n", waitFront->name, waitFront->id);

    else

        printf("No one in waiting list.\n");

}

void freeAll() {

    Node \*temp;

    while (booked) {

        temp = booked;

        booked = booked->next;

        free(temp);

    }

    while (waitFront) {

        temp = waitFront;

        waitFront = waitFront->next;

        free(temp);

    }

}

**Sample output**

========= TICKET BOOKING SYSTEM =========

**1. Book Ticket**

**2. Cancel Ticket**

**3. Show Booked Passengers**

**4. Show Waiting List**

**5. Search Passenger**

**6. Show Summary**

**7. Exit**

**Enter choice: 1**

Enter passenger name: jyothika

Ticket booked successfully! ID: 1

**Enter choice: 1**

Enter passenger name: ananya

Ticket booked successfully! ID: 2

**Enter choice: 1**

Enter passenger name: harshitha

Ticket booked successfully! ID: 3

**Enter choice: 1**

Enter passenger name: sathvika

Ticket booked successfully! ID: 4

**Enter choice: 1**

Enter passenger name: bhanu

Ticket booked successfully! ID: 5

**Enter choice: 1**

Enter passenger name: sneha

All seats full! Added to waiting list. ID: 7

**Enter choice: 3**

Booked Passengers:

ID: 5 | Name: bhanu

ID: 4 | Name: sathvika

ID: 3 | Name: harshitha

ID: 2 | Name: ananya

ID: 1 | Name: jyothika

**Enter choice: 4**

Waiting List:

ID: 6 | Name: Sneha

**Enter choice: 6**

===== SUMMARY =====

Total Booked Seats: 5 / 5

Total Waiting Passengers: 1

Next Waiting Passenger: sneha(ID:6)

**Enter choice: 2**

Enter passenger ID to cancel: 1

Ticket cancelled successfully!

Passenger 'sneha' (ID:6) moved from waiting list to booked list.

**Enter choice: 7**

Exiting... All memory freed.

## Conclusion

This mini project successfully demonstrates the practical application of Data Structures using C. It efficiently manages booking, cancellation, and waiting lists using linked lists and queues.