RAILWAY/BUS RESERVATION SYSTEM

A PROJECT REPORT

Submitted by

SIVARAJ T **(113224071O91)**

In the partial fulfillment of the award of the degree

Of

BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY

VELAMMAL
ENGINEERING COLLEGE
The Wheel of Knowledge rolls on!

DEPARTMENT OF INFORMATION TECHNOLOGY

VELAMMAL ENGINEERING COLLEGE,CHENNAI-66.
(An Autonomous Institution,Affiliated to Anna University,chennai)

ANNA UNIVERSITY
PROGRESS THROUGH KNOWLEDGE

ANNA UNIVERSITY, CHENNAI - 600025
NOV 2025

VELAMMAL ENGINEERING COLLEGE DEPARTMENT, OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

certified that this project report titled "**RAILWAY/BUS RESERVATION SYSTEM**" is the bonafide work of **SIVARAJ T (113224071091),** Who carried out the Mini project work under my supervision.

SIGNATURE

INTERNAL GUIDE,

Mrs.S.Priyadharshini,

Assistant Professor ,

Department Of Information Technology ,

Velammal Engineering College,

Ambattur-Redhills road, Chennai - 600066.

SIGNATURE

HEAD OF THE DEPARTMENT,

Dr. Jeevaa Katiravan

Professor and Head,

Department of Information Technology,

Velammal Engineering College,

Ambattur-Redhills road,

Chennai - 600066.

ABSTRACT

This project is a Railway/Bus Reservation System developed in the C programming language that manages basic ticket booking operations for a limited number of seats. It uses an array to keep track of seat availability and a linked list to store passenger details such as seat number, name, and age. The system allows users to book tickets by automatically assigning the first available seat, cancel bookings either by seat number or passenger name, view the current status of all seats, and display the list of passengers with their details. Through these features, the program demonstrates the practical application of fundamental concepts in data structures, dynamic memory allocation, and console-based input/output. Although designed as a simple mini project, it can be further extended to handle larger real-world reservation systems with additional functionalities like file storage, duplicate name handling, or advanced search options.

KEYWORDS:

C Programming, Reservation System, Railway/Bus Booking, Passenger Management, Seat Allocation, Linked List, Array, Dynamic Memory Allocation, Ticket Cancellation, Data Structures.

ACKNOWLEDGEMENT

Behind every achievement lies an unfathomable sea of gratitude to those who achieved it, without whom it would ever have come into existence. To them we express our words of gratitude.We give all the glory and thanks to our almighty **GOD** for showering upon, the necessary wisdom and grace for accomplishing this project . We express our gratitudeand thanks to **Our Parents** first for giving health and sound mind for completing this project . First of all, we would like to express our deep gratitude to our beloved and respectable **Chairman Thiru M.V. Muthuramalingam and our Chief Executive Officer Thiru M.V.M. Velmurugan** for thier kind encouragement . We express our deep gratitude to **Dr.Sathish Kumar, Principal** , Velammal Engineering College for his helpful attitude towards this project . We wish to express our sincere thanks and gratitude to **Dr.Jeevaa Katiravan** , Professor and Head of the Department, Department of Information technology for motivating anencouraging in every part of our project.We express our sincere gratitude to **the Project Coordinator name Mrs.S.Priyadharshini,** Assistant Professor, Department of Information technology for their invaluable guidance in shaping of this project without them this project would not have been possible.Our thanks to all other **Faculty and Non- Teaching Staff members** of our department for their support and peers for having stood by me and helped me to complete this project.

TABLE OF CONTENTS

1.INTRODUCTION

The Railway/Bus Reservation System is a simple console-based application developed in the C programming language to manage the process of booking and cancelling tickets for passengers. The project is designed to handle a limited number of seats and provides essential functionalities such as booking tickets, cancelling reservations, viewing available seats, and displaying the passenger list. It uses arrays to track seat availability and linked lists to store passenger details dynamically, ensuring efficient memory management and easy data handling. This project serves as a practical implementation of core data structure concepts like arrays and linked lists, while also demonstrating the use of dynamic memory allocation and structured programming in C. Although developed as a mini project, it lays the foundation for understanding larger and more complex reservation systems used in real-world applications such as railway booking, bus ticketing, and even cinema hall management.

1.1 MOTIVATION

The main motivation behind developing this project is to understand how real-world reservation systems work and to implement their basic functionalities using the C programming language. In today's world, ticket booking for buses, trains, and other services pla ys an essential role in transportation management. By creating a simplified version of such a system, this project helps in learning the practical application of data structures like arrays and linked lists, as well as concepts of dynamic memory allocation and structured programming. The project also motivates learners to think about extending small-scale programs into larger real-time applications by adding features such as file handling, user authentication, and database connectivity. Ultimately, this project serves as a stepping stone for students to bridge the gap between theoretical knowledge and practical system development.

1.2 PROBLEMS:

1. **Manual ticket booking is time-consuming and error-prone.**

Handling bookings by hand often leads to mistakes in recording passenger details or seat numbers.

2. **Difficulty in tracking available and booked seats.**

Without an automated system, it becomes hard to quickly identify which seats are free or already reserved.

3. **Cancellations are difficult to manage without proper records.**

Manual cancellation processes can cause confusion and lead to incomplete updates of seat availability.

4. **Passenger details are not properly organized.**

Maintaining passenger information in notebooks or registers results in data inconsistency and retrieval issues.

5. **Risk of double-booking or over-booking seats.**

When multiple bookings are handled simultaneously, there is no mechanism to prevent assigning the same seat to multiple passengers.

6. **Lack of an automated system for small-scale transport services.**

Small bus or railway services often lack a simple digital platform, leading to inefficiency and poor customer satisfaction.

1.3 PROPOSED SYSTEM:

1. **Automated Ticket Booking:**
   The system allows passengers to book tickets quickly by entering their details, automatically assigning the next available seat.

2. **Real-Time Seat Availability:**
   Users can view which seats are booked and which are available instantly, eliminating confusion and manual checking.

3. **Easy Ticket Cancellation:**
   Passengers can cancel their booked tickets easily by providing either their seat number or name, and the system updates availability immediately.

4. **Organized Passenger Records:**
   Passenger details such as name, age, and seat number are stored systematically, allowing easy retrieval and management

5. **Error-Free Booking Process:**
   Since the system is automated, the chances of duplicate or incorrect bookings are minimized.

6. **Efficient Seat Management:**
   The system maintains an updated list of available and occupied seats, ensuring smooth operation and clear tracking of bookings.

2.SOFTWARE USED:

1. **C – Main Programming Language:**
It is used to implement the core functionalities such as ticket booking, cancellation, seat management, and passenger data handling

2. **IDE (Integrated Development Environment) – Dev-C++:**
It provides an easy-to-use interface for code editing, debugging, and program execution

3. **C Compiler (Built into Dev-C++):**
The compiler translates the C source code into executable machine code, allowing the program to run successfully.

4. **Console / Terminal:**
The program runs in a console-based interface where the user can interact through menu options — such as booking tickets, cancelling bookings, and viewing seat availability.

2.1 PLATFORM USED

1. **Operating System – Windows 11:**

The Ticket Reservation System is developed, compiled, and tested on the Windows 11 operating system, which provides a stable and user-friendly environment for software development.

2. **IDE – Dev-C++:**

Dev-C++ is used as the Integrated Development Environment for writing, compiling, and executing the C program. It provides features such as syntax highlighting, debugging, and easy program execution.

3. **C Compiler (Built into Dev-C++):**

The built-in C compiler is used to compile the source code into an executable format, ensuring smooth and efficient program execution.

4. **Console / Terminal:**

The program runs through a console-based interface, allowing the user to interact with the system via menu options. Users can book or cancel tickets, view available seats, and display the list of passengers directly through the command-line interface.

2.2 TOOLS USED:

Programming Tools:

1. **C – Main Programming Language:**

Used to develop the Ticket Reservation System.It handles ticket booking, cancellation, seat management, and passenger record operations through structured programming concepts.

Development Tools:

1. **Dev-C++ – IDE:**

Used for writing, compiling, and executing the C program. It provides an integrated environment with features like debugging, syntax highlighting, and code execution support.

2. **C Compiler (Built into Dev-C++):**

Compiles the C source code into executable form efficiently, ensuring smooth and error-free program execution.

3. **Console / Terminal:**

Used for running the program and interacting with the user through a simple, menu-driven interface.It displays options to book or cancel tickets, view available seats, and show the passenger list.

3. SYSTEM IMPLEMENTATION

3.1 C LANGUAGE:

C is the main programming language used for developing the Ticket Reservation System. It is chosen for its simplicity, efficiency, and structured programming features. The language enables implementation of core functionalities such as ticket booking, cancellation, seat allocation, and passenger data management.Features like arrays, structures, pointers, and linked lists are used to store passenger details, manage seats, and ensure smooth operation of the system.C provides fast execution and flexibility, making it suitable for building reliable console-based applications.

3.2 DEV-C++:

Dev-C++ serves as the primary development environment for the project. It is used to write, compile, and debug the C program in an organized and user-friendly interface.The IDE provides features like syntax highlighting, code completion, error detection, and integrated compilation, which make the development process faster and easier.Dev-C++ also allows executing the program directly within the console window, simplifying testing and debugging during development.

3.3 C COMPILER (Built-in Dev-C++ Compiler):

The C compiler translates the written source code into machine-readable instructions, ensuring that the program executes correctly and efficiently. The built-in compiler in Dev-C++ provides reliable and optimized compilation, helping the Ticket Reservation System run smoothly without external dependencies.

3.4 CONSOLE / TERMINAL INTERFACE:

The console or terminal acts as the user interface for the Ticket Reservation System. It allows users to interact with the application through a menu-driven interface, offering options such as:

- Booking a ticket
- Cancelling a ticket
- Viewing available seats
- Displaying the passenger list

The console provides a simple and efficient way for users to enter passenger details, manage bookings, and receive immediate responses about seat availability and booking confirmations.

4. ALGORITHM

Step 1: Start the Program

- Begin execution of the Ticket Reservation System.
- Initialize all necessary variables and data structures.
- Set all seat positions in the array seats[] to 0 (indicating all seats are free).
- Set the linked list head = NULL to indicate no passengers are booked yet.

Step 2: Display Menu

Show the user the main menu options:

1. Book Ticket
2. Cancel Ticket
3. View Available Seats
4. View Passenger List
5. Exit

Prompt the user to enter their choice.

Step 3: Book Ticket

- Search for the first available seat in the array seats[].
- If no seats are available, display: "All seats are booked."
- Otherwise:
    1. Prompt the user to enter Passenger Name and Age.
    2. Mark the selected seat as booked (seats[i] = 1).
    3. Create a new Passenger Node containing:
        - 3.1. Seat Number
        - 3.2. Name
        - 3.3. Age
    4. Insert the new passenger node at the end of the linked list.
    5. Display a success message showing the assigned seat number.

Step 4: Cancel Ticket

- Ask the user whether they want to cancel by Seat Number or Passenger Name.
- Traverse the linked list to find the matching passenger record.
- If found:
  1. Remove the passenger node from the linked list.
  2. Update the seat status in seats[] to 0 (available).
  3. Display: "Ticket cancelled successfully."
- If not found:
  1. Display: "Passenger not found."

Step 5: View Available Seats

- Traverse the seats[] array.
- For each seat:
  1. If value = 0 → Display "Available"
  2. If value = 1 → Display "Booked"
- This helps users see current seat status in real time.

Step 6: View Passenger List

- Check if the linked list is empty.
- If no passengers are booked, display: "No passengers booked yet."
- Otherwise, traverse the linked list and display details for each passenger:
  1. Seat Number
  2. Name
  3. Age

Step 7: Exit the Program

- When the user selects Exit:
  1. Free all dynamically allocated memory (linked list nodes).
  2. End program execution gracefully.
  3. Display: "Thank you for using the Ticket Reservation System."

5.SOURCE CODE

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

#define MAX_SEATS 20

struct Passenger {
    int seatNo;
    char name[50];
    int age;
    struct Passenger *next;
};

struct Passenger *head = NULL;
int seats[MAX_SEATS];

void bookTicket() {
    int i, seatNo;
    char name[50];
    int age;

    seatNo = -1;
    for (i = 0; i < MAX_SEATS; i++) {
        if (seats[i] == 0) {
            seatNo = i + 1;
            break;
        }
    }
```

```
if (seatNo == -1) {
        printf("\nAll seats are booked! No seats available.\n");
        return;
    }

    printf("\nEnter Passenger Name: ");
    scanf(" %[^\n]", name);
    printf("Enter Age: ");
    scanf("%d", &age);

    seats[seatNo - 1] = 1;

    struct Passenger *newP = (struct Passenger *)malloc(sizeof(struct
Passenger));
    newP->seatNo = seatNo;
    strcpy(newP->name, name);
    newP->age = age;
    newP->next = NULL;

    if (head == NULL) {
        head = newP;
    } else {
        struct Passenger *temp = head;
        while (temp->next != NULL)
            temp = temp->next;
        temp->next = newP;
    }

    printf("\nTicket booked successfully! Seat No: %d\n", seatNo);
}
```

```
void cancelTicket() {
    if (head == NULL) {
        printf("\nNo bookings yet!\n");
        return;
    }

    int option;
    printf("\nCancel Ticket By:\n1. Seat Number\n2. Passenger Name\nEnter
choice: ");
    scanf("%d", &option);

    struct Passenger *temp = head, *prev = NULL;

    if (option == 1) {
        int seatNo;
        printf("\nEnter Seat Number to Cancel: ");
        scanf("%d", &seatNo);

        if (seatNo < 1 || seatNo > MAX_SEATS || seats[seatNo - 1] == 0) {
            printf("\nInvalid Seat Number or Seat not booked.\n");
            return;
        }

        while (temp != NULL && temp->seatNo != seatNo) {
            prev = temp;
            temp = temp->next;
        }
    }
    else if (option == 2) {
        char name[50];
        printf("\nEnter Passenger Name: ");
        scanf(" %[^\n]", name);
```

```
        while (temp != NULL && strcmp(temp->name, name) != 0) {
            prev = temp;
            temp = temp->next;
        }
    }
    else {
        printf("\nInvalid option.\n");
        return;
    }

    if (temp == NULL) {
        printf("\nPassenger not found!\n");
        return;
    }

    if (prev == NULL) {
        head = temp->next;
    } else {
        prev->next = temp->next;
    }

    seats[temp->seatNo - 1] = 0;
    printf("\nTicket for Seat No %d (Passenger: %s) cancelled successfully!\n",
temp->seatNo, temp->name);

    free(temp);
}
```

```
void viewPassengerList() {
    if (head == NULL) {
        printf("\nNo passengers booked yet!\n");
        return;
    }

    printf("\n=== Passenger List ===\n");
    struct Passenger *temp = head;
    while (temp != NULL) {
        printf("Seat No: %2d | Name: %-20s | Age: %d\n",
               temp->seatNo, temp->name, temp->age);
        temp = temp->next;
    }
}

int main() {
    int choice;

    for (int i = 0; i < MAX_SEATS; i++)
        seats[i] = 0;

    while (1) {
        printf("\n====== Railway/Bus Reservation System ======\n");
        printf("1. Book Ticket\n");
        printf("2. Cancel Ticket\n");
        printf("3. View Available Seats\n");
        printf("4. View Passenger List\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
```

```
switch (choice) {
            case 1: bookTicket(); break;
            case 2: cancelTicket(); break;
            case 3: viewAvailableSeats(); break;
            case 4: viewPassengerList(); break;
            case 5: exit(0);
            default: printf("\nInvalid choice! Try again.\n");
        }
    }

    return 0;
}
```

6. OUTPUT

```
====== Railway/Bus Reservation System ======
1. Book Ticket
2. Cancel Ticket
3. View Available Seats
4. View Passenger List
5. Exit
Enter your choice: 1

Enter Passenger Name: SIVARAJ T
Enter Age: 18

? Ticket booked successfully! Seat No: 1

====== Railway/Bus Reservation System ======
1. Book Ticket
2. Cancel Ticket
3. View Available Seats
4. View Passenger List
5. Exit
Enter your choice: 1

Enter Passenger Name: THARUN T
Enter Age: 13

? Ticket booked successfully! Seat No: 2

====== Railway/Bus Reservation System ======
1. Book Ticket
2. Cancel Ticket
3. View Available Seats
4. View Passenger List
5. Exit
Enter your choice: 1

Enter Passenger Name: KISHORE R
Enter Age: 18

? Ticket booked successfully! Seat No: 3
```

```
====== Railway/Bus Reservation System ======
1. Book Ticket
2. Cancel Ticket
3. View Available Seats
4. View Passenger List
5. Exit
Enter your choice: 1

Enter Passenger Name: PRAVIN S
Enter Age: 16

? Ticket booked successfully! Seat No: 4

====== Railway/Bus Reservation System ======
1. Book Ticket
2. Cancel Ticket
3. View Available Seats
4. View Passenger List
5. Exit
Enter your choice: 1

Enter Passenger Name: ASHWIN
Enter Age: 12

? Ticket booked successfully! Seat No: 5

====== Railway/Bus Reservation System ======
1. Book Ticket
2. Cancel Ticket
3. View Available Seats
4. View Passenger List
5. Exit
Enter your choice: 2

Cancel Ticket By:
1. Seat Number
2. Passenger Name
Enter choice: 1

Enter Seat Number to Cancel: 3

? Ticket for Seat No 3 (Passenger: KISHORE R) cancelled successfully!
```

```
====== Railway/Bus Reservation System ======
1. Book Ticket
2. Cancel Ticket
3. View Available Seats
4. View Passenger List
5. Exit
Enter your choice: 3

=== Available Seats ===
Seat  1 [Booked]
Seat  2 [Booked]
Seat  3 [Available]
Seat  4 [Booked]
Seat  5 [Booked]
Seat  6 [Available]
Seat  7 [Available]
Seat  8 [Available]
Seat  9 [Available]
Seat 10 [Available]
Seat 11 [Available]
Seat 12 [Available]
Seat 13 [Available]
Seat 14 [Available]
Seat 15 [Available]
Seat 16 [Available]
Seat 17 [Available]
Seat 18 [Available]
Seat 19 [Available]
Seat 20 [Available]
```

```
====== Railway/Bus Reservation System ======
1. Book Ticket
2. Cancel Ticket
3. View Available Seats
4. View Passenger List
5. Exit
Enter your choice: 4

=== Passenger List ===
Seat No:  1 | Name: SIVARAJ T            | Age: 18
Seat No:  2 | Name: THARUN T             | Age: 13
Seat No:  4 | Name: PRAVIN S             | Age: 16
Seat No:  5 | Name: ASHWIN               | Age: 12
```

7. DRAWBACKS

1. **Manual Data Entry:**
   - Passenger details such as name and age must be entered manually through the console.
   - This makes the process time-consuming, especially when handling multiple bookings.
2. **Limited User Interface:**
   - The system operates through a simple text-based (console) interface.
   - It lacks a graphical user interface (GUI) that could make ticket booking more user-friendly.
3. **No Online or Real-Time Integration:**
   - The system cannot connect to online databases or real-time seat availability updates.
   - All data exists only during program execution and is lost when the program closes.
4. **No Data Storage or Persistence:**
   - Booked passenger information is not saved permanently.
   - Once the program is terminated, all data is erased, limiting long-term usability.
5. **Limited Scalability:**
   - The system handles only a fixed number of seats (e.g., 20).
   - Increasing capacity would require modifying the source code manually.
6. **No Payment or Confirmation Feature:**
   - The system does not include payment gateways, ticket receipts, or booking confirmations.
   - Users must rely solely on console messages for booking status.
7. **Error Handling and Validation Limitations:**
   - The system assumes correct input formats.
   - Invalid data or unexpected inputs may cause errors or improper functioning.

8. FUTURE ENHANCEMENT

1. **Graphical User Interface (GUI):**
   - Implement a GUI using frameworks like Qt, GTK, or Tkinter for a more interactive and user-friendly interface.
   - A visual interface will allow users to easily select seats, view bookings, and navigate the system efficiently, instead of relying on a text-based console.
2. **Online Booking & Real-Time Integration:**
   - Connect the system to an online database to enable real-time booking and seat availability updates.
   - Users could book tickets remotely, check availability instantly, and reduce manual effort.
3. **Support for Additional Transportation Types:**
   - Extend the system to include multiple bus routes, trains, or other transport types.
   - This makes the system scalable for larger transport networks.
4. **Automated Reports:**
   - Generate reports on ticket bookings, cancellations, seat utilization, and passenger demographics.
   - These reports can help transport operators in planning, resource allocation, and decision-making.
5. **Scalability & Optimization:**
   - Optimize the system to handle a larger number of seats and passengers efficiently.
   - Use advanced data structures or databases to maintain performance even with high booking volumes.
6. **Ticket Confirmation & Payment Integration:**
   - Integrate payment gateways for online ticket payments.
   - Generate printable or digital tickets with confirmation numbers for passengers.
7. **Enhanced Error Handling & Validation:**
   - Implement robust input validation to avoid incorrect or incomplete data entry.
   - This will reduce errors and improve reliability for users and operators.

9. CONCLUSION

The Ticket Reservation System successfully demonstrates how bus or railway ticket booking can be managed efficiently using C and structured programming concepts. The system provides a systematic approach to booking, cancelling, and tracking tickets, ensuring that seat allocation is organized, reliable, and minimizes errors.

One of the major achievements of this project is its ability to reduce manual effort significantly. The automated booking and cancellation process improves operational efficiency, prevents double booking, and ensures that seat management is accurate and up-to-date.

The system provides a comprehensive view of seat availability and passenger information, allowing administrators and operators to monitor which seats are free and which are booked, as well as maintain a record of all passengers. This clarity is critical for effective management, helping avoid confusion, optimize seat utilization, and provide better service to passengers.

The menu-driven console interface ensures that users can easily interact with the system without requiring technical expertise, making it accessible for operators and staff at all levels.

Potential improvements include integrating a graphical user interface (GUI) for a more interactive experience, enabling online booking and real-time seat availability, adding payment integration for ticket confirmation, and implementing advanced reporting for bookings, cancellations, and seat utilization. These enhancements would further improve efficiency, scalability, and usability of the system for a wider range of transport services.

10. REFERENCE

1. Kernighan, B.W., & Ritchie, D.M., The C Programming Language, 2nd Edition, Prentice Hall, 1988.
2. Schildt, H., C: The Complete Reference, 4th Edition, McGraw-Hill Education, 2000.
3. Hanly, J.R., Data Structures Using C, 2nd Edition, McGraw-Hill, 2012.
4. Tanenbaum, A.S., Structured Computer Organization, 6th Edition, Pearson, 2016.
5. GeeksforGeeks, C Programming Language, https://www.geeksforgeeks.org/c-programming-language/
6. TutorialsPoint, C Programming Language
https://www.tutorialspoint.com/cprogramming/index.htm
7. Reema Thareja, Data Structures Using C, Oxford University Press, 2017.
8. Cprogramming.com, C Programming
https://www.cprogramming.com/tutorial/c-tutorial.html