

Project Title: Train Ticket Booking

|  |
| --- |
| Name: Naveen Kumar M |
| College Reg No: 620819106056 |
| Batche NO: 101 |
| Cranes Reg. No: R2025040484510482 |
| Project Title: Train Ticket Booking |
| S/w Used: C Programming (Visual Studio Code) |
| Data Structures Used: Linked List |
| Trainer Name: Sandeep |

# Acknowledgment

I would like to express my sincere gratitude to my trainer Sandeep for the valuable guidance and support throughout the project.

I am also thankful to Cranes Varsity and its faculty members for providing the infrastructure and learning environment to carry out this work.

I would also like to thank my classmates and family members for their encouragement and moral support.

# Abstract

The Train Ticket Reservation System is a mini-project developed in the C programming language that simulates the core functionalities of a railway ticket booking system. The project utilizes a

linked list data structure to dynamically manage and store the booking information, allowing for efficient addition and removal of tickets without fixed memory limitations. This system enables users to perform several key operations such as viewing the available trains, booking tickets by entering passenger details, displaying all current bookings, and cancelling previously booked tickets.

The choice of a linked list as the underlying data structure provides flexibility in handling a varying number of bookings, as memory is allocated and freed dynamically based on the user’s actions. This prevents wastage of memory and enhances performance compared to static arrays. Furthermore, the project incorporates essential programming concepts like file handling to save booking details persistently, memory management to avoid leaks, and modular programming to organize the code into manageable functions for better readability and maintenance.

Through this project, users and developers gain practical experience in implementing data structures and handling real-world scenarios such as managing train schedules and passenger bookings. It serves as an educational tool to understand how complex systems can be modelled using fundamental programming techniques. The system can be further improved by integrating features like user authentication for security, support for multiple classes of tickets, and a graphical user interface (GUI) for easier interaction. Additionally, incorporating database connectivity could allow for better scalability and data management in larger systems.

Overall, the Train Ticket Reservation System is a foundational project that demonstrates the application of linked lists in real-life problems while also reinforcing core concepts of C programming. It lays the groundwork for more advanced ticketing systems used in the transportation industry today.

# Introduction to Project

India has one of the largest rail networks in the world, serving millions of people daily. With the growing need for digital solutions, an efficient and user-friendly train ticket reservation system becomes essential. This project simulates a basic train reservation system built using the C programming language. It allows users to book and manage their tickets dynamically using a linked list data structure.

The system provides four main functionalities: displaying available trains, booking a ticket, viewing booked tickets, and canceling bookings. This project also aids learners in understanding core concepts of data structures, modular programming, and memory management in C.

# Objective and Scope

The main objective of this project is to create a mini-application that simulates a train ticket booking system using the C language.

Objectives:

* To understand the working of linked lists in managing dynamic data.
* To apply modular programming practices in real-world problem solving.
* To simulate a ticket reservation system using basic C concepts.

Scope:

* This system serves as a learning tool and base model for advanced ticketing systems.
* Can be extended to integrate features like database storage, GUI interface, fare calculation, and schedule management.

# System Requirements

Hardware Requirements:

* A system with minimum 2GB RAM
* Intel Pentium or higher processor
* 100MB free disk space

Software Requirements:

* Operating System: Ubuntu/Linux or Windows
* GCC Compiler
* Code Editor: VS Code / Code::Blocks / Turbo C (Windows)
* Basic terminal or command prompt access

# Data Structures Used

The primary data structure used in this project is a singly linked list. Structure Definition:

struct BookingNode { char name[50];

int age; int trainN;

int seat\_no; char date[20]; char time[20];

struct BookingNode \*next;

};

Each node in the list holds booking details for a passenger and points to the next booking. This structure allows dynamic addition and deletion of tickets as required

# Flowcharts

**Main Menu Flowchart**

flowchart TD

A[Start] --> B{Show Menu} B -->|1| C[Display Trains]

B -->|2| D [Book Ticket]

B -->|3| E [View Booked Tickets] B -->|4| F [Cancel Ticket]

B -->|5| G[Exit]

C --> B

D --> B

E --> B

F --> B

G --> H[End]

# Booking Process Flowchart

flowchart TD

A [Start Booking] --> B [Input Name, Age, Train No]

B --> C [Check Valid Train]

C -->|Invalid| B

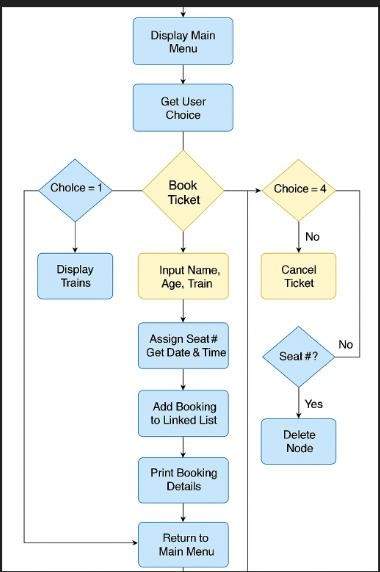
C -->|Valid| D [Assign Seat Number]

D --> E [Get Date & Time]

E --> F [Add to Linked List]

F --> G [Confirm Booking]

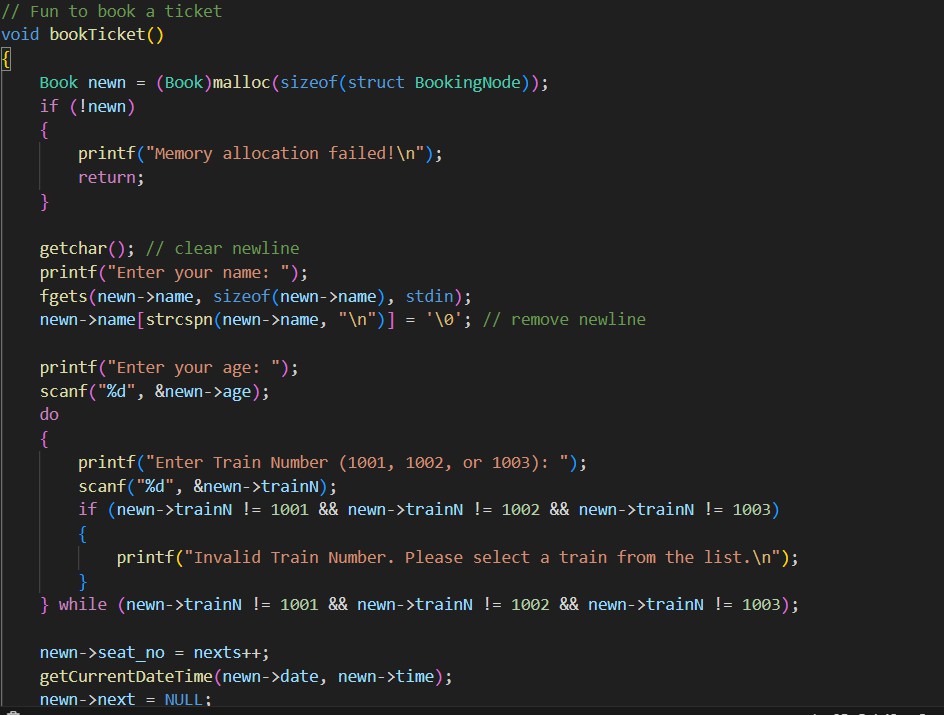
G --> H[End]

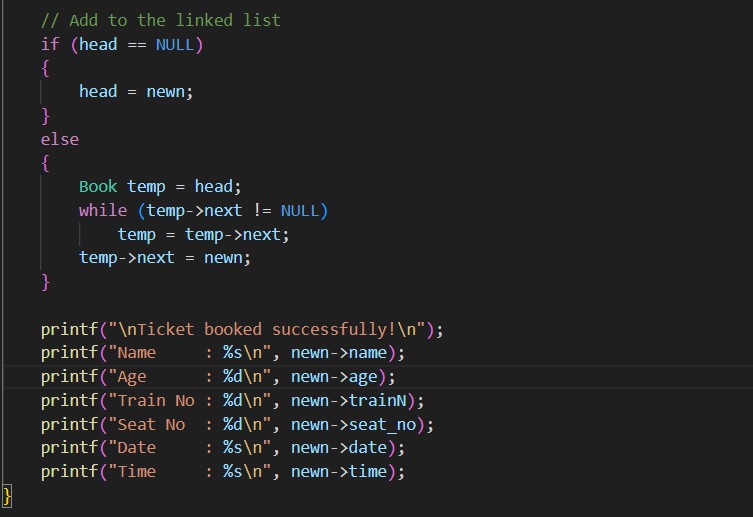


# Code Snippets and Explanations

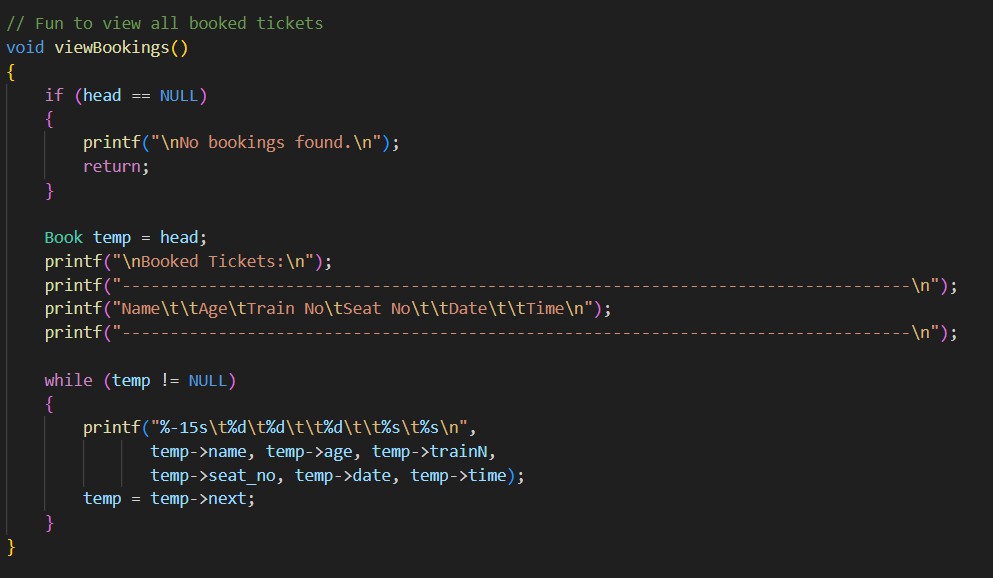
The core functionalities are implemented using C programming. Below are some key functions:

1. bookTicket():
   * Dynamically allocates memory for a new booking.
   * Takes input: name, age, train number.
   * Assigns seat number, date, and time.
   * Adds the node to the linked list.

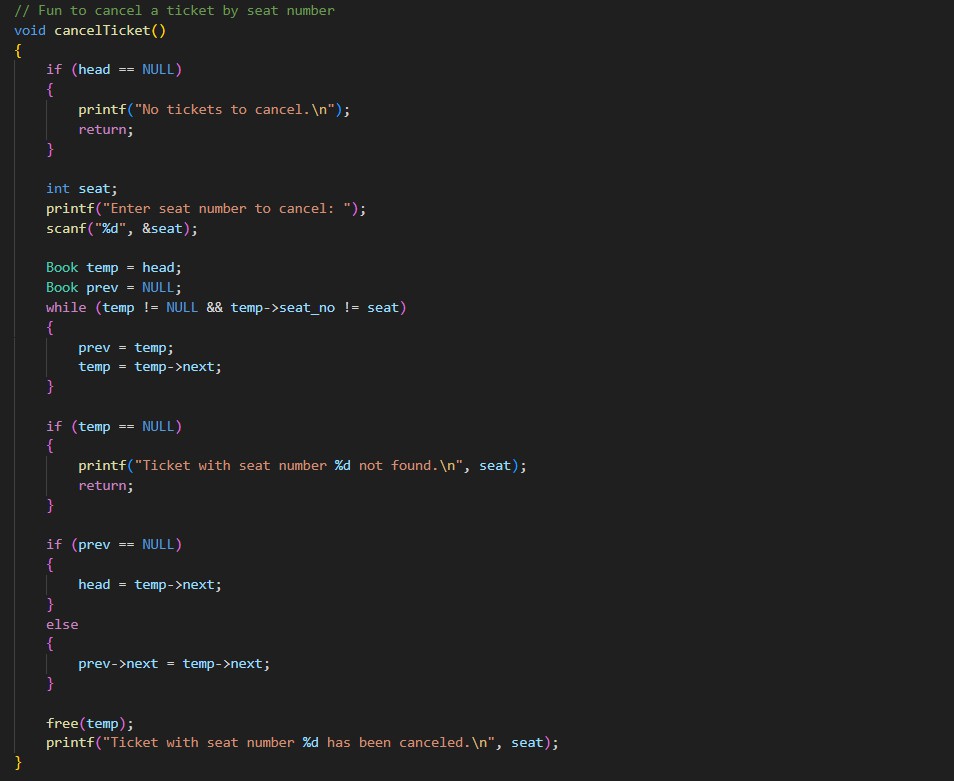




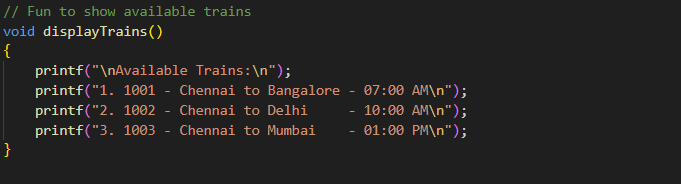
1. viewBookings():
   * Traverses the linked list.
   * Displays booking details for each node.



1. cancelTicket():
   * Accepts a seat number to be canceled.
   * Searches and deletes the corresponding node.



1. displayTrains():
   * Lists all available trains and basic information.



Structure:

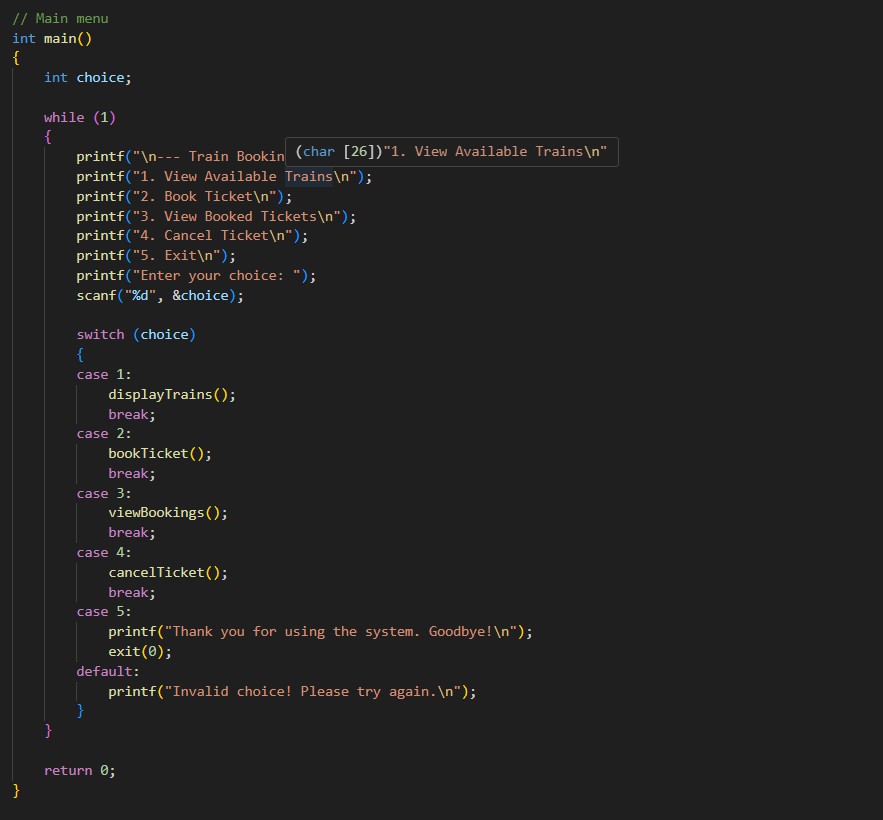
struct BookingNode { char name[50];

int age; int trainN;

int seat\_no; char date[20]; char time[20];

struct BookingNode \*next;

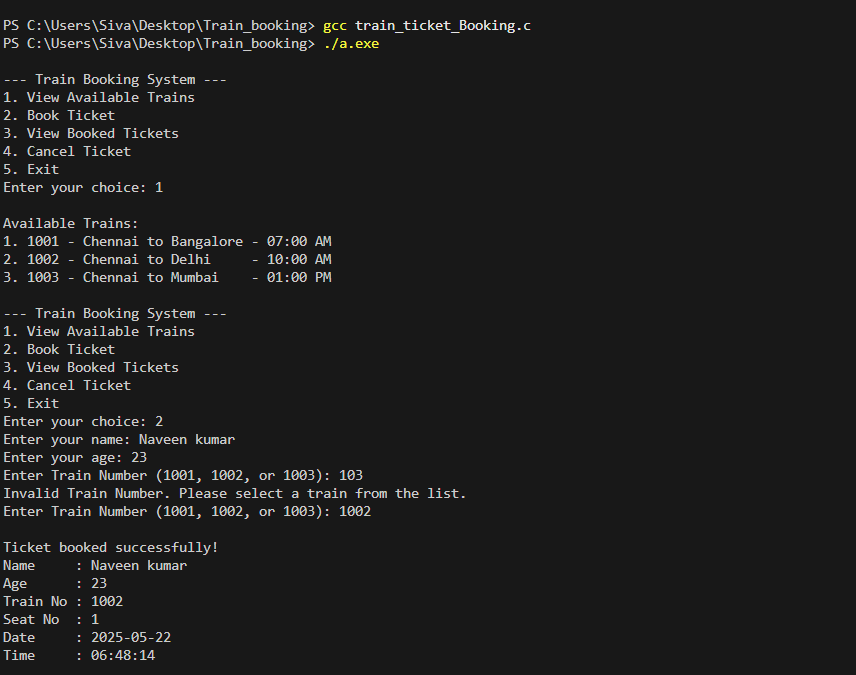
};

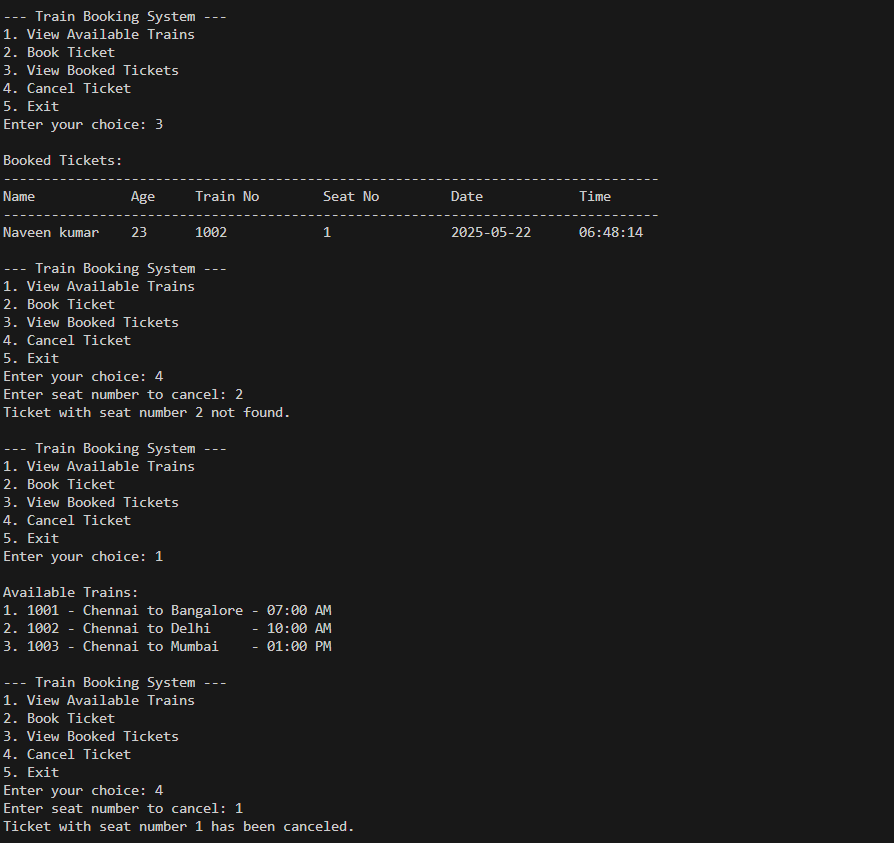


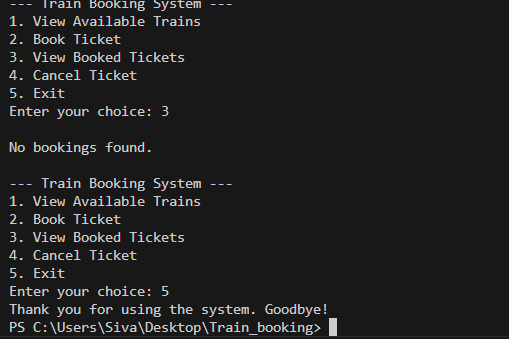
.

# Key Functions Used

* + - **displayTrains()**: Displays the available trains.
    - **getCurrentDateTime()**: Gets system date and time.
    - **bookTicket()**: Books a new ticket and adds to the linked list.
    - **viewBookings()**: Displays all tickets stored in the linked list.
    - **cancelTicket()**: Cancels a ticket based on seat number.







# Conclusion and Future Enhancements

This project successfully demonstrates a basic train ticket reservation system using the C language and linked list data structure. It dynamically stores booking details and supports real- time operations like booking, viewing, and canceling tickets.

# Future Enhancements:

* + - Adding file handling to store bookings persistently.
    - Introducing a GUI using C++ or Python.
    - Including train timings, class options, and price.
    - Adding admin and user login features.

# Conclusion and Future Enhancements

This project effectively demonstrates the application of linked lists and structures in building a simple yet functional **Train Ticket Reservation System**. Developed in the C programming language, the system helps simulate core features such as booking tickets, viewing passenger data, and canceling reservations. The use of **singly linked lists** provides dynamic memory management and showcases how data structures can be applied in real-world applications.

From a learning perspective, this project offers an engaging opportunity to understand key programming concepts such as:

* + - Modular programming through function separation
    - Dynamic memory allocation using pointers and malloc()
    - Struct-based record handling
    - Command-line interaction and real-time input processing

These concepts are crucial for anyone aspiring to become a proficient programmer, particularly in systems or embedded software development.

# Future Enhancements

To extend the usability and practical application of this project, several improvements can be made:

* + - **Persistent File Storage:** Use file handling to save bookings and restore them across sessions.
    - **GUI Integration:** Develop a user interface using C++ (Qt) or Python (Tkinter) to make it user-friendly.
    - **Train Timetables and Pricing:** Add support for different routes, train timings, class types, and fare calculations.
    - **Authentication System:** Include login and role-based access (Admin and User).
    - **Database Connectivity:** Integrate with SQL or NoSQL databases for large-scale storage and retrieval.

With these features, the project can evolve from an academic model to a more comprehensive real-world system usable by institutions, travel agencies, or railway networks.

**References**

* + - Kernighan, B. W., & Ritchie, D. M. (1988). *The C Programming Language* (2nd ed.). Prentice Hall.
    - Balagurusamy, E. (2019). *Programming in ANSI C* (8th ed.). McGraw Hill Education.
    - GeeksforGeeks. (n.d.). *Linked List in Data Structures*. Retrieved from <https://www.geeksforgeeks.org/linked-list-data-structure/>
    - Tutorialspoint. (n.d.). *C Programming Examples and Linked List*. Retrieved from https:/[/www.tutorialspoint.com/cprogramming/index.htm](http://www.tutorialspoint.com/cprogramming/index.htm)
    - Stack Overflow. (n.d.). *Dynamic Memory Allocation & C Data Structures*. Retrieved from [https://stackoverflow.com](https://stackoverflow.com/)