NITTE (Deemed to be University) | NMAM INSTITUTE OF TECHNOLOGY

Nitte (DU) established under Section 3 of UGC Act 1956 | Accredited with 'A+' Grade by NAAC

**Department of Computer Science & Engineering**

Report on Mini Project

# Railway Ticket Booking

Course Code : CS3651-1
Course Name : Programming with C++

Semester: VSEM Batch:

A1

**Submitted To:**
Dr. Raju K
Associate Professor
Department of Computer Science and Engineering

**Submitted By:**
Agam R (NNM22CS011)
Ansh Vashisht
(NNM22CS029) Charith (
NNM22CS046)
Deeksha Ramachandra (NNM22CS056)

**Date of submission:**
11/11/2024

**Signature of Course Instructor**

## ABSTRACT

It outlines the design and implementation of a train management and ticket booking system using C++. The system is structured into multiple classes that handle various functionalities, including managing train details, admin and user authentication, and ticket bookings. It enables administrators to add new trains, view available trains and manage user accounts, while customers can book tickets and select seat classes securely. The program ensures data integrity by employing file handling to store train and user information, preventing duplicates and enabling data persistence across sessions. Additional features includes to implement include the ability for users to change passwords, the inclusion of validations to ensure correct input, and the use of a menu-driven interface to enhance user experience. The system is designed to be scalable, user-friendly, and efficient, catering to the needs of both administrators and regular users in managing and booking train tickets.

# ACKNOWLEDGEMENT

It is with great satisfaction and enthusiasm that we are submitting the Mini Project Report on "**Railway Ticket Booking**". We have completed it as a part of the V semester **Programming with C++** (CS3651-1) of Bachelor of Engineering in **Computer Science & Engineering** of Nitte University, Deralakatte.

I take this opportunity to express my gratitude and regards to my project guide ,**Dr. Raju K,** Associate Professor ,Department of Computer Science Engineering for his guidance ,monitoring and encouragement throughout this Mini project.

I would like to express my deepest gratitude to **Dr. Jyothi Shetty**, Head of the Department of Computer Science and Engineering, for her invaluable guidance, unwavering support, and encouragement throughout my academic journey.

My sincere thanks to our beloved principal**, Dr. Niranjan N Chiplunkar**, and vice principal **Dr. I.R.Mithanthaya** for permitting us to carry out this Mini project at our college and providing us with all needed facilities.

I thank my parents and friends for the immense support, honest opinion, and suggestions throughout the mini project.

# TABLE OF CONTENTS

# INTRODUCTION

This report presents the design and implementation of a Train Management and Ticket Booking System developed in C++. This system aims to streamline the process of train scheduling, user account management, and ticket booking, providing a centralized solution that meets the needs of both administrators and customers. With increasing demand for efficient and reliable transportation management systems, this application addresses the need for secure, easy-to-use, and accessible ticket booking and train management services. The system combines secure authentication, organized data handling, and intuitive user interaction, ensuring users can book tickets and manage travel plans with ease while administrators maintain control over train schedules and booking records.

### Purpose

The purpose of this project is to develop a functional and user-friendly application for managing train schedules and booking tickets. It aims to improve the efficiency of ticket booking, reduce errors in scheduling, and offer secure user authentication. By automating these tasks, the system reduces manual work, streamlines operations, and enhances the convenience for both customers and administrator.

## Scope

The Train Management and Ticket Booking System has been developed to support multiple essential functions within a single application. The system includes capabilities for administrators to manage trains, schedules, and booking records, while customers can search for available trains, book tickets, and view their booking history. Additionally, the system uses file handling for data persistence and is built using an object-oriented approach in C++, allowing future scalability and customization for additional features.

## Overview

The report covers the design principles, functionality, and technical implementation of the Train Management and Ticket Booking System. It includes the following sections:

- **System Design** – Outlines the object-oriented structure, key classes (trains, users, tickets), and relationships within the system.

- **Functionality** – Details the main features available for both administrators and users, including account management, train scheduling, ticket booking, and cancellation.

- **Data Management** – Discusses the use of file handling for persistent data storage and retrieval, ensuring data integrity.

- **Security and Validation** – Describes user authentication mechanisms and input validation processes to maintain data accuracy and secure user interaction.

This structure provides a comprehensive look at the system's capabilities and underlying architecture, supporting both user convenience and administrative control.

# BASIC CONCEPTS OF OOPs

Object-Oriented Programming (OOP) is a programming paradigm that organizes software design around data, or objects, rather than functions and logic. In OOP, the focus is on defining classes, which serve as blueprints for objects, and establishing relationships between them through concepts like encapsulation, inheritance, abstraction, and polymorphism. This approach provides modular, reusable, and efficient code that is easier to maintain and extend. Below are the fundamental concepts of OOP, including explanations of classes and objects with examples related to the concepts discussed.

## Class and Object

A class in OOP is a blueprint for creating objects, encapsulating data for the object (attributes) and methods to manipulate that data (functions). An object, on the other hand, is an instance of a class that holds specific values for the attributes defined in the class. For example we could define a class called **User** , **Train** with attributes such as **username**, **ADMIN_PASSWORD** and **trainid**. Then, each individual attribute would be an object with its own specific values for these attributes.

## Encapsulation

Encapsulation is the concept of bundling data (attributes) and methods (functions) that operate on that data into a single unit, known as a class. In this system, classes like Train, User, and Ticket encapsulate related data and methods, hiding their internal workings from outside access. For instance, the Train class may contain private attributes like **trainNumber** and **departureTime**, and public methods such as **getTrainInfo()** to provide controlled access. Encapsulation enhances code maintainability by keeping data secure and reducing interdependencies.

## Abstraction

Abstraction simplifies complex systems by exposing only essential features while hiding implementation details. In this system, abstraction is achieved by defining functions that provide necessary operations, like **bookTicket()** and **cancelBooking(),** without revealing the internal workings of these processes. Users interact with straightforward methods, which internally handle data validation, file operations, and other complexities, thus enhancing usability and focusing on core functionalities while hiding implementation details.

## Inheritance

Inheritance allows a new class to inherit properties and behaviors from an existing class, fostering code reuse and logical hierarchy. In this system, for example, a class like User might serve as a base class, with specialized classes like **AdminUser** and **CustomerUser** inheriting from it. Each subclass can extend or override functionalities as needed, enabling tailored features for each user type, such as administrative privileges for **AdminUser** or booking capabilities for **CustomerUser**.

## Polymorphism

Polymorphism allows objects to be treated as instances of their parent class, enabling flexibility in function usage. In this system, polymorphism can be implemented with functions like **manageAccount()** that might perform differently based on the user type **(AdminUser or CustomerUser).** Through function overriding or operator overloading, polymorphism lets the system dynamically determine and execute the appropriate version of a function, allowing flexible interaction while ensuring consistency across different user types.

## PROBLEM STATEMENT

Developing a train reservation system in C++ that enables users to manage trains, tickets, and accounts effectively. The system should allow adding and viewing train details (train number, name, origin, destination, and seat availability), booking tickets for specific trains, and handling user accounts for sign- up, login, and password management. The program must include input validation to ensure unique train numbers and usernames, secure passwords, and correct ticket details for each reservation. It should store data persistently, so train and user information is retained across sessions.

The solution consists of three main classes: **Train**, **Ticket**, and **User**. The **Train** class allows users to add trains with unique details, check seat availability, and update records when reservations are made. The **Ticket** class manages ticket bookings, where each ticket is assigned a unique reservation number and booking status. The **User** class handles account functionalities, including sign-up, login, account deletion, and password changes, using file operations for data persistence. Input validation checks for empty or duplicate fields, and secure password management is enforced to ensure a robust user experience. Binary files are used to store train and user information, supporting persistent data management and making the system efficient and reliable for basic train reservation tasks

## OBJECTIVES

The primary objective of this train reservation system code is to simulate the functionalities of a simple train booking system through object-oriented programming. By creating classes for **Train**, **User** and **Ticket**. The code organizes the main aspects of the reservation process, including adding train details, managing user accounts, and handling reservations. This structure provides a clear, modular framework to ensure that each component functions independently while contributing to the overall operation of the reservation system.

The code also aims to facilitate easy data access and management for users. For example, users can search for train availability, create accounts, and book tickets through straightforward commands. Although this code does not include backend functionality or data persistence, it provides a foundation for how a reservation system could work, emphasizing logical flows and interaction between classes. Through methods like validating user credentials, checking seat availability, and booking tickets, the code focuses on essential tasks in a train reservation system without involving databases or server connections.

Finally, this code serves as a learning tool for building a train reservation system in a basic, offline environment. By simulating real-world actions, it introduces the user to programming concepts like class inheritance, encapsulation, and object interactions in a practical context. This makes it a valuable exercise for understanding how larger, more complex systems operate and prepares the foundation for further development, such as integrating a backend or enhancing the code with a graphical user interface.

In addition to simulating a train reservation system, the code also focuses on providing a seamless and user-friendly experience for interactions. It enables the user to easily access functionalities such as viewing available trains, booking tickets, and managing user profiles. This simple interaction flow makes it easy to understand the essential operations of a train booking system while also providing a starting point for adding more advanced features in the future. The code's clear structure, through well-defined classes and methods, emphasizes simplicity, making it easy for developers to extend or modify as needed.
Moreover, the system is designed to handle basic booking operations in a streamlined manner, ensuring that users can check for available trains, book tickets, and manage their bookings with minimal hassle

By simulating real-world booking operations—like checking availability and booking tickets—the system also gives an insight into how complex data, like seat availability and user management, can be handled. Though limited to a console-based interaction, this approach provides a clear understanding of the flow of data and the sequence of operations involved in the reservation process.

Lastly, while the system is not connected to a live backend or a database, it lays the groundwork for future enhancements. For example, developers can integrate a database to store train schedules, user details, and booking records, providing persistent data storage and retrieval. Furthermore, future versions could include real-time updates, such as train delays or seat availability changes. As it stands, this code can easily be expanded to a more robust system with additional features, such as payment integration, notifications, and advanced search filters, to make it a comprehensive train reservation system.

# METHODOLOGY

The methodology for developing the Railway Ticket Booking System revolved around object-oriented programming principles in C++. The key phases include:

Requirements Gathering and Problem Definition: Understanding the primary objectives of the system, such as user account management, train schedule handling, ticket booking, and the need for secure data handling.

System Design: The system was architected using an object-oriented approach. The main components include:

Class Definitions: Classes were defined for Train, Ticket, and User, encapsulating their respective data attributes and methods.

Object Relationships: A clear relationship between classes was established to represent real-world interactions between users, trains, and ticket reservations.

Data Persistence: File handling mechanisms were used to ensure data such as train details and user credentials were stored persistently.

Implementation Strategy:

A menu-driven console interface was developed for user interaction.
Input validation was prioritized to prevent invalid entries and ensure data integrity.

Secure authentication mechanisms were implemented for user accounts.

File handling ensured persistence across user sessions for train schedules and user information

# IMPLEMENTATION

The system implementation in C++ revolves around several key classes that handle different functionalities such as train scheduling, user management, and ticket booking. The **Train** class is responsible for managing train schedules and seat availability. It includes data members such as **train_number, train_name, origin, destination, and seats_available**. The class provides functions like **add_train()** to add new trains, **view_schedule()** to display all available trains, and **update_seat_availability()** to modify seat counts when tickets are booked. Additionally, the class can update seat availability when a booking is made or a ticket is canceled, and all train data is stored using file handling to ensure persistence.

The User class is designed to handle user registration, login, and account management. It serves as a base class for two derived classes: **Admin** and **Customer**. The base class includes essential data members such as **username**, **password**, **email**, as well as functions like **register_user()** to handle **user registration, login() for authentication, and update_profile()** to allow users to modify their details. The Admin subclass provides additional capabilities such as managing train schedules and viewing all user profiles, while the Customer subclass focuses on ticket booking and management. User data is saved in a binary file to maintain persistent records across sessions.

The **Ticket** class manages the booking and cancellation of tickets. It contains data members such as **ticket_number, train_number, customer_username, and status**. Key functions in this class include **book_ticket()**, which handles the reservation process by assigning a unique ticket number and updating seat availability, and **cancel_ticket(),** which allows customers to cancel their bookings, thus freeing up seats. It also includes a **view_ticket()** function that displays the details of a specific booking. Like other classes, ticket details are stored using file operations to maintain consistency and retrieve data efficiently.

File handling is a crucial component in the system, ensuring that all user credentials, train details, and ticket information are stored in binary files. This approach ensures efficient data persistence and retrieval, as well as easy updates and maintenance of the system's records. The use of binary files helps in efficiently managing large volumes of data and makes the system more robust.

# RESULTS AND DISCUSSIONS

The Railway Ticket Booking System underwent extensive testing to ensure both functionality and a positive user experience. Key results and observations from the testing process are as follows:

**User Experience:**

The system featured a user-friendly, menu-driven interface, providing a straightforward and intuitive platform for users to book tickets and manage their accounts. The clear and organized structure of the interface made navigation easy, ensuring that even users with minimal technical experience could operate the system without difficulty. Input validation was implemented to prevent erroneous data entries, enhancing the accuracy and reliability of user input.

**System Performance:**

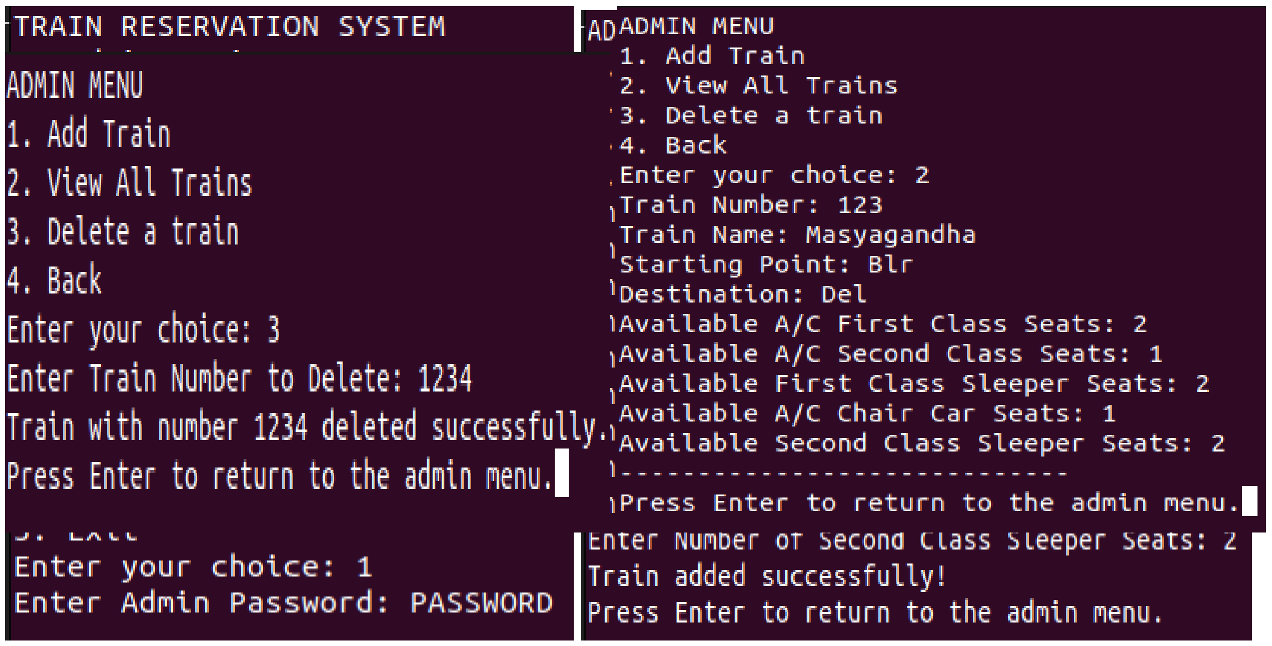
The system's use of file handling for persistent data storage and retrieval was effective, ensuring smooth operations across multiple sessions. This approach allowed for efficient management of train schedules, user profiles, and ticket bookings, even when dealing with a moderate volume of data. Ticket booking and cancellation processes were executed seamlessly, ensuring quick and accurate updates to train seat availability without any noticeable lag or errors.

**Security Features:**

The system’s secure user login and account management functionalities demonstrated a high level of security for handling sensitive user credentials. Passwords were validated and stored in a secure manner, protecting user information from unauthorized access. Additionally, the system employed robust validation checks for train entries and ticket bookings, ensuring data integrity and preventing inconsistent or fraudulent entries. These security measures significantly reduced the risk of data corruption or unauthorized access, contributing to a trustworthy and reliable system.

Overall, the system performed well in terms of usability, performance, and security, ensuring a seamless and secure experience for all users.

## OUTPUTS



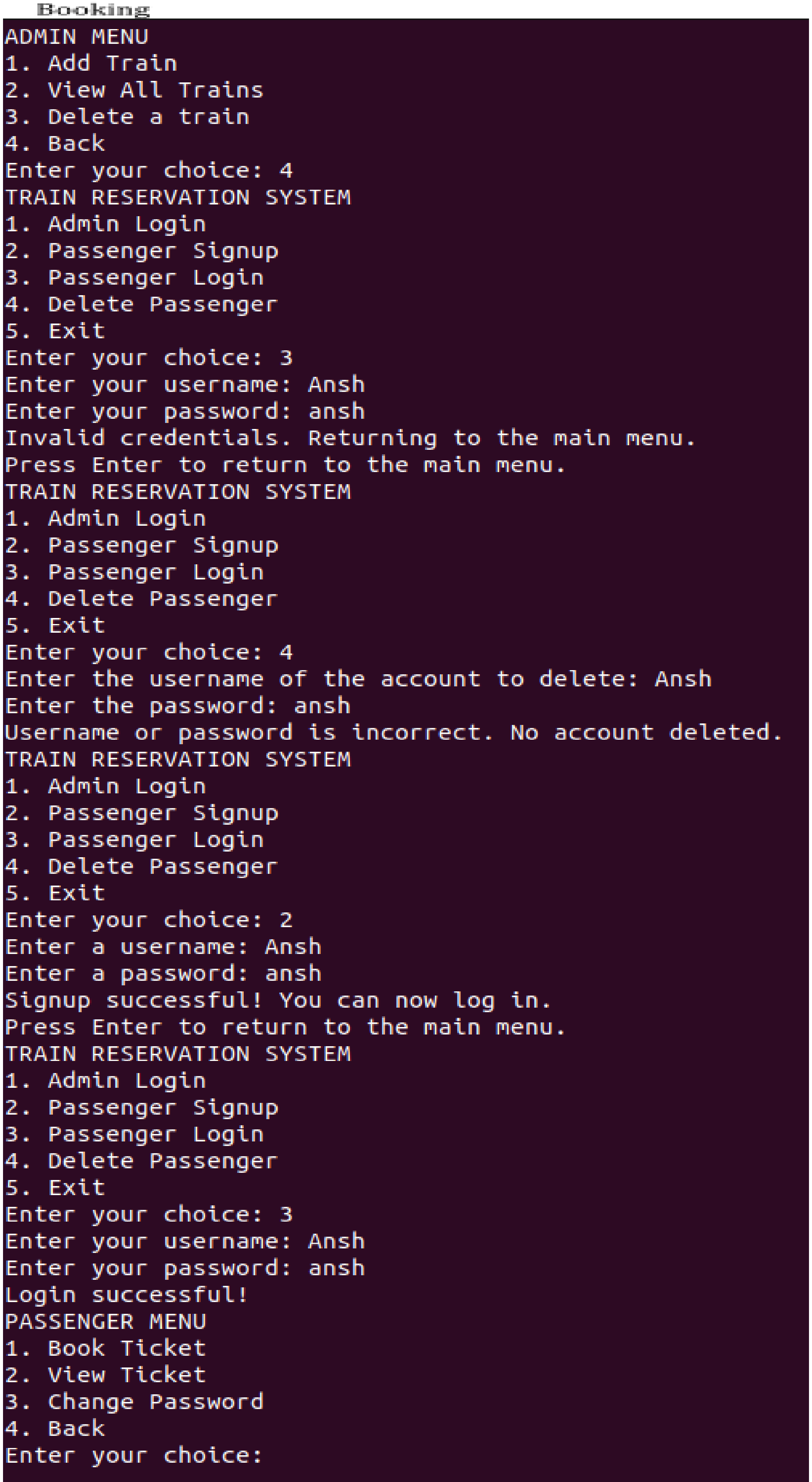
```
TRAIN RESERVATION SYSTEM
```

```
ADMIN MENU
1. Add Train
2. View All Trains
3. Delete a train
4. Back
Enter your choice: 3
Enter Train Number to Delete: 1234
Train with number 1234 deleted successfully.
Press Enter to return to the admin menu.
```

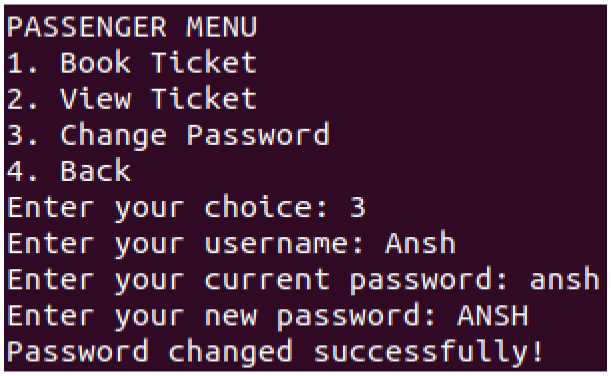
```
Enter your choice: 1
Enter Admin Password: PASSWORD
```

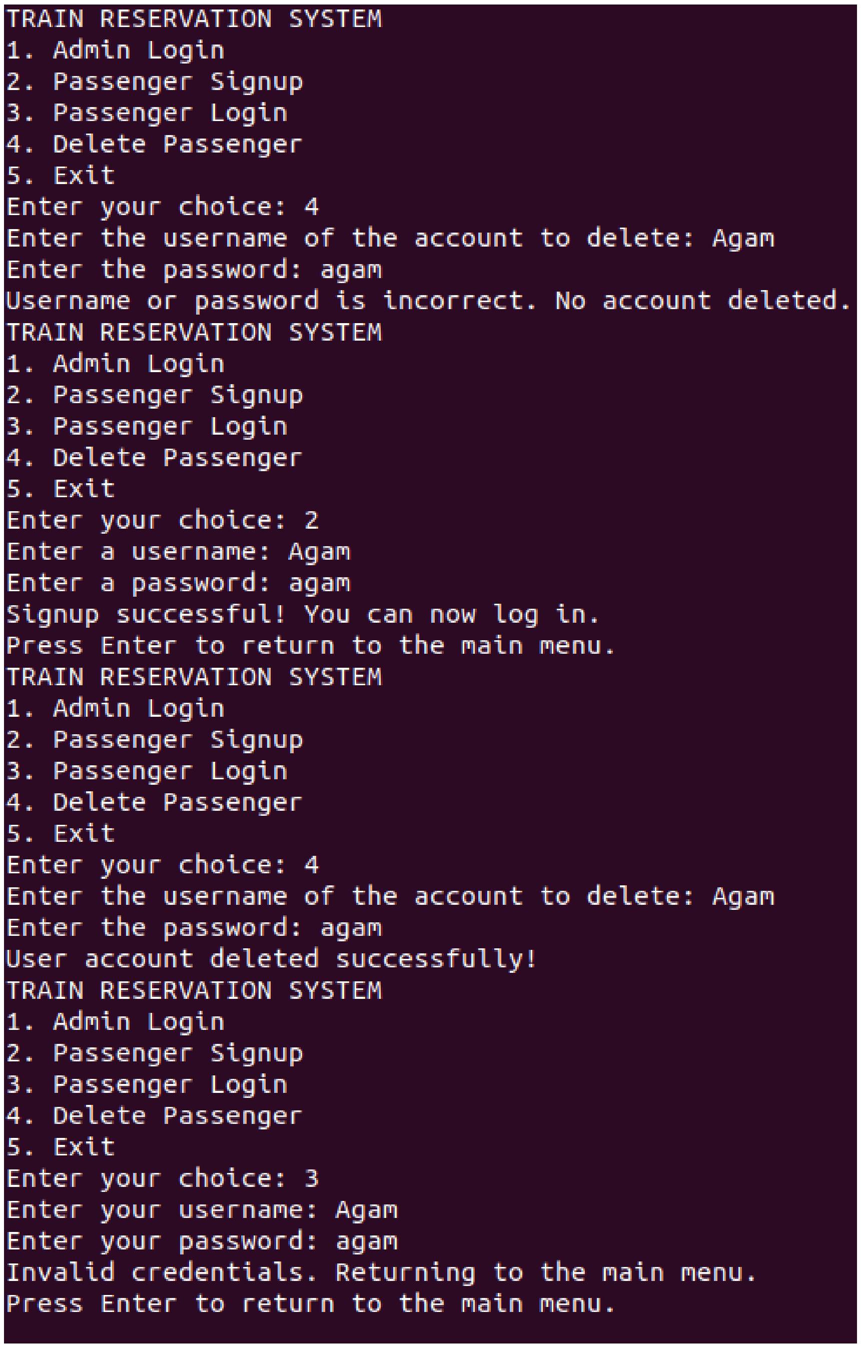
```
ADMIN MENU
1. Add Train
2. View All Trains
3. Delete a train
4. Back
Enter your choice: 2
Train Number: 123
Train Name: Masyagandha
Starting Point: Blr
Destination: Del
Available A/C First Class Seats: 2
Available A/C Second Class Seats: 1
Available First Class Sleeper Seats: 2
Available A/C Chair Car Seats: 1
Available Second Class Sleeper Seats: 2
------------------------------
Press Enter to return to the admin menu.
```

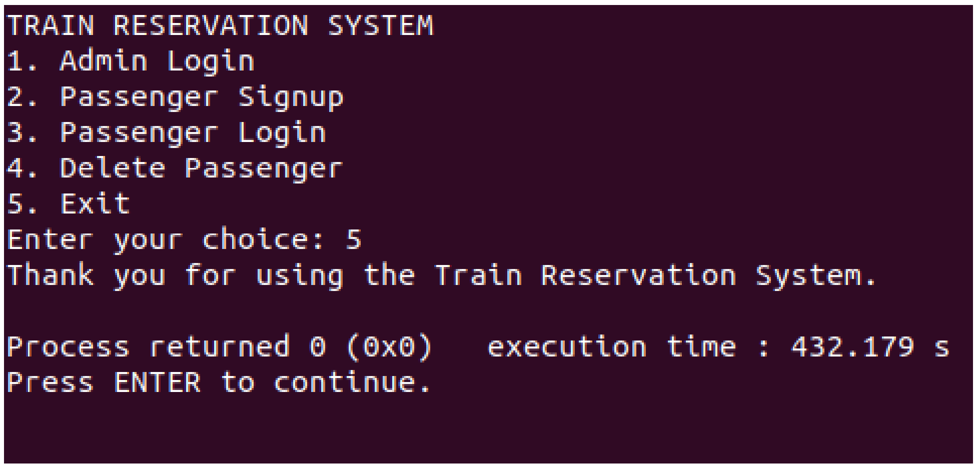
```
Enter Number of Second Class Sleeper Seats: 2
Train added successfully!
Press Enter to return to the admin menu.
```



```
ADMIN MENU
1. Add Train
2. View All Trains
3. Delete a train
4. Back
Enter your choice: 4
TRAIN RESERVATION SYSTEM
1. Admin Login
2. Passenger Signup
3. Passenger Login
4. Delete Passenger
5. Exit
Enter your choice: 3
Enter your username: Ansh
Enter your password: ansh
Invalid credentials. Returning to the main menu.
Press Enter to return to the main menu.
TRAIN RESERVATION SYSTEM
1. Admin Login
2. Passenger Signup
3. Passenger Login
4. Delete Passenger
5. Exit
Enter your choice: 4
Enter the username of the account to delete: Ansh
Enter the password: ansh
Username or password is incorrect. No account deleted.
TRAIN RESERVATION SYSTEM
1. Admin Login
2. Passenger Signup
3. Passenger Login
4. Delete Passenger
5. Exit
Enter your choice: 2
Enter a username: Ansh
Enter a password: ansh
Signup successful! You can now log in.
Press Enter to return to the main menu.
TRAIN RESERVATION SYSTEM
1. Admin Login
2. Passenger Signup
3. Passenger Login
4. Delete Passenger
5. Exit
Enter your choice: 3
Enter your username: Ansh
Enter your password: ansh
Login successful!
PASSENGER MENU
1. Book Ticket
2. View Ticket
3. Change Password
4. Back
Enter your choice:
```


```
PASSENGER MENU
1. Book Ticket
2. View Ticket
3. Change Password
4. Back
Enter your choice: 3
Enter your username: Ansh
Enter your current password: ansh
Enter your new password: ANSH
Password changed successfully!
```


```
TRAIN RESERVATION SYSTEM
1. Admin Login
2. Passenger Signup
3. Passenger Login
4. Delete Passenger
5. Exit
Enter your choice: 4
Enter the username of the account to delete: Agam
Enter the password: agam
Username or password is incorrect. No account deleted.
TRAIN RESERVATION SYSTEM
1. Admin Login
2. Passenger Signup
3. Passenger Login
4. Delete Passenger
5. Exit
Enter your choice: 2
Enter a username: Agam
Enter a password: agam
Signup successful! You can now log in.
Press Enter to return to the main menu.
TRAIN RESERVATION SYSTEM
1. Admin Login
2. Passenger Signup
3. Passenger Login
4. Delete Passenger
5. Exit
Enter your choice: 4
Enter the username of the account to delete: Agam
Enter the password: agam
User account deleted successfully!
TRAIN RESERVATION SYSTEM
1. Admin Login
2. Passenger Signup
3. Passenger Login
4. Delete Passenger
5. Exit
Enter your choice: 3
Enter your username: Agam
Enter your password: agam
Invalid credentials. Returning to the main menu.
Press Enter to return to the main menu.
```



```
TRAIN RESERVATION SYSTEM
1. Admin Login
2. Passenger Signup
3. Passenger Login
4. Delete Passenger
5. Exit
Enter your choice: 5
Thank you for using the Train Reservation System.

Process returned 0 (0x0)   execution time : 432.179 s
Press ENTER to continue.
```

## CONCLUSION

The train reservation system presented in the code provides a fundamental implementation of a train ticket booking system and a clone of real life booking of railway tickets. It successfully simulates essential functionalities, such as viewing train schedules, booking tickets, and managing user details like login and registration. The code structure ensures a user-friendly interface while maintaining clarity and simplicity. Although the system currently operates as a standalone application without a backend, it serves as a good starting point for developers to understand the core processes involved in ticket booking. The system also showcases basic functionalities such as login and ticket booking, providing a solid foundation for further development.

## FUTURE SCOPE

While the current implementation serves as a basic prototype, there are numerous opportunities to expand and improve the system. A key area for enhancement is the integration of a backend database to handle user profiles, train schedules, and booking data persistently. This would allow users to check their booking history, and admins could manage train schedules and seat availability in real-time. Additionally, incorporating features like seat reservations and dynamic pricing based on demand could enhance the system's functionality and provide users with a more comprehensive experience.

To further improve the system, it could be extended to include features such as an automatic email confirmation system and payment gateway integration, enabling users to make payments securely and receive tickets electronically. A more refined graphical user interface (GUI) or web-based interface could be introduced for a better user experience, especially for mobile or tablet users. Furthermore, adding real-time tracking for train locations, departure delays, and notifications for users regarding changes in their bookings would make the system more dynamic and responsive to real-world conditions. Finally, the system could incorporate multi-language support to cater to a broader range of users, increasing accessibility and usability in diverse geographical regions.

## REFERENCES

[1] **GeekforGeeks** [https://www.geeksforgeeks.org/c-plus-plus/]

[2] **JavatPoint** [https://www.javatpoint.com/cpp-tutorial]

[3] **Chatgpt** [https://openai.com/index/chatgpt/]