A blue and white logo

Description automatically generated

**Dynamic Railway Reservation System**

Submitted To: Sir Zahid Anwar

Subject: Data Structures

Submitted By: Muhammad Arabi

Reg # SP24-BSE-031

**Date: June 16, 2025**

Contents

[Tech Stack: 4](#_Toc200957108)

[Abstract 5](#_Toc200957109)

[INTRODUCTION 6](#_Toc200957110)

[Features: 7](#_Toc200957111)

[1.User Registration and Login System 7](#_Toc200957112)

[2.Train Management 7](#_Toc200957113)

[3.Train Search Functionality 7](#_Toc200957114)

[4.Ticket Booking System 7](#_Toc200957115)

[5.Registration queue 8](#_Toc200957116)

[6.Manually coded data structure 8](#_Toc200957117)

[Functionality 8](#_Toc200957118)

[1.User Class 8](#_Toc200957119)

[2.Booking Class 8](#_Toc200957120)

[3.Train Class 8](#_Toc200957121)

[4.MyStack<T> 8](#_Toc200957122)

[5.BookingManager Class 9](#_Toc200957123)

[6.TrainNode And TrainLinkedList 9](#_Toc200957124)

[7.MyQueue<T> 9](#_Toc200957125)

[8.UserManager Class 9](#_Toc200957126)

[9.Main Function 9](#_Toc200957127)

[Data Structures Used 9](#_Toc200957128)

[1.Stack 9](#_Toc200957129)

[2.Vector<int> 9](#_Toc200957130)

[3.Queue 9](#_Toc200957131)

[4.LinkedList 10](#_Toc200957132)

[5.Dynamic Array 10](#_Toc200957133)

[6.Unordered\_map 10](#_Toc200957134)

[Code Explanation 11](#_Toc200957135)

[1.User Class 11](#_Toc200957136)

[2.Booking Class 11](#_Toc200957137)

[3.Train Class 11](#_Toc200957138)

[4.MyStack 12](#_Toc200957139)

[5.BookingManager Class 12](#_Toc200957140)

[6.TrainNode and TrainLinkedList Classes 13](#_Toc200957141)

[7.TrainManager Class 13](#_Toc200957142)

[8.MyQueue Class 14](#_Toc200957143)

[9.UserManager Class 14](#_Toc200957144)

[UML Class Diagram 16](#_Toc200957145)

[SCOPE: 17](#_Toc200957146)

[LIMITATIONS 18](#_Toc200957147)

[CONCLUSION 19](#_Toc200957148)

# Tech Stack:

|  |  |  |
| --- | --- | --- |
| Category | Tool/Technology | Description |
| Programming Language | C++ | Language of program |
| IDE | Visual Studio Code | Used IDE: VS Code |
| Version Control | Git and GitHub | Version control: GitHub |
| UML Diagram | Draw.io | Drew UML Diagram |

**GitHub Repository Link:**

<https://github.com/arabi-1/A-Semester-Projects/tree/main/3rd%20Semester/DSA>

# Abstract

This dynamic railway reservation system is a software that involves the core functionalities of a railway reservation system. This software is based on CLI(Command Line Interface) and it is developed in C++ coding language. This system includes important data structures like LinkedList, stack, queue, vectors and hash maps, which are implemented manually. The properties of this system are Object Oriented Design, Train Management, Seat Availability System, User Account Management, Booking System, viewing recent bookings and CLI based interface.

# INTRODUCTION

The railway system is an important part of today’s fast paced transport system.

Due to lower fair costs and higher need of the transport, people try to prefer it more for long distance travels over airplanes. Thus managing this high traffic of people needs efficient management. For this purpose this software has been built to manage all the passengers so that they may know their relevant seats, bus numbers, travel timings, and many others details about their journey from one place to another. This is built using different algorithms and data structures.

This software gives the following features to it’s users:

* User Registration
* Train Management
* Train Search Functionality
* Seat Availability Checking
* Ticket Booking
* Ticket Cancellation
* View Booking History
* View Recent Bookings
* Custom Data Structures

The aim of this software is to make travel easier for travelers, and avoid any mis managements.

# Features:

## User Registration and Login System

New users can register with a username and password

User data is stored using HashMap

## Train Management

Trains are stored in a LinkedList

System can add new train with:

Train ID

Train Name

Source

Destination

Date

Total available seats

## Train Search Functionality

A typical user can search a train based on:

Source station

Destination station

Journey Date

The train that matches the data is displayed.

## Ticket Booking System

User can book a ticket by choosing:

Train ID

Seat Number

Date

A unique PNR number is generated for each booking

The booking details include:

User Name

Train ID

Seat Number

Journey Date

## Registration queue

A custom queue (MyQueue) stores new user registration for processing

Queue data structure is used in this feature

## Manually coded data structure

Custom stack implementation for booking history

Custom queue for registration processing

LinkedList for train management

# Functionality

## User Class

It represents a user who registers and books tickets

## Booking Class

Represents a single ticket booking

Enables a user to book a ticket

## Train Class

It represents a train in the system.

The details related to the seats, availability etc

## MyStack<T>

Here a custom stack is implemented

This stack is used to store booking history.

## BookingManager Class

Handles all booking-related operations

## TrainNode And TrainLinkedList

Custom singly linked list to manage trains

The TrainNode hold a train object and a pointer to the next node.

## MyQueue<T>

A custom queue is implemented

It is used for registration queue

Processing first-in, first-out operation

## UserManager Class

Handles user related operations like registration and login.

## Main Function

It controls the whole system and drives it

# Data Structures Used

## 1.Stack

This data structure is based on LIFO method

LIFO stands for last in first out

Used in Booking Manager

Stores the history of bookings

## Vector<int>

To store multiple bookings PNRs made by the user in the User class.

In the Train class it represents seat availability.

## Queue

Used in MyQueue<User>

Stores registration order of users

Used in main function, simulates processing users one by one from a queue

## LinkedList

Used in TrainLinkedList class, manages dynamic list of trains without fixed size.

TrainLinkedList is used in TrainManager to provide interface to manage the linked list of trains.

## Dynamic Array

T\* arr

Underlying array for queue operations in MyQueue class

Underlying array to implement stack operations

## Unordered\_map

Used for fast access (O(1) average time, using keys like usernames

# Code Explanation

## User Class

### Data Members

String username: Stores the user’s name

String password: Stores the user’s password

Vector<int> bookingPNRs: List of PNR numbers(bookings) associated with user

### Methods:

viewBookings(), Display all bookings made by the user

## Booking Class

### Data members

Int pnr: Booking reference number

String username: Username of the person who booked

Int trainID: ID of the booked train

Int seatNumber: Booked seat number

String bookingDate: Date of travel

### Method

printSummary():

prints all booking information(used for displaying bookings).

## Train Class

### Data Members

Int trainID: Unique ID of the train

String trainName: Name of the train

String source: Departure station

String destination: Arrival station

String date: Travel date;

Int totalSeats: Total number of seats in the train

Vector<boo> seats: Marks which seats are booked or available

### Methods

showAvailability(): Display the un booked seat numbers

bookSeat(int seatNumber): Books a seat if it’s available returns true on success

cancelSeat(int seatNumber):

Cancels a previously booked seat.

## MyStack

### Data Members

Int capacity: max size of the stack

Int topIndex: Tracks the top element

T\* arr: Dynamic array to store stack elements

### Methods

Push(T value): Adds elements to the top, checks for overflow

Pop(): Remove the top element. Check for underflow

Top(): Returns top element. Throws error if empty

Empty(): Returns true if stack is empty

Size():Returns number of elements in the stack

## BookingManager Class

### Data Members

Static int currentPNR: Counter to generate unique PNRs.

Unordered\_map(int, Booking> allBookings: Stores all bookings with PNR as key

MyStack<int> bookingHistory:Stack to store recent bookings

### Methods

bookTicket():Generate PNR, creates bookings, adds to map and stack

cancelTicket(pnr): Removes bookings by PNR if it exists

printBooking(pnr): Prints bookings info for a given PNR

showRecentBookings(count):Display last count bookings using the custom stack

## TrainNode and TrainLinkedList Classes

### TrainNode:

Train train: A train object

TrainNode\* next: Pointer to the next train node

### TrainLinkedList

TrainNode\* head: Head of the linked list

### Methods

addTrain(Train t)

Adds a new train node to the list

searchTrains(source, dest, date)

Searches for trains based on criteria

DisplayAllTrain():Prints all train details

getTrainByID(id): Finds and returns pointer to a train with given ID

## TrainManager Class

### Data Members

TrainLinkedList trainList: Composition relationship with TrainLinkedList

### Methods

addTrain(): Adda a train to the linked list

searchTrains(): Forwards request to TrainLinkedList

listAllTrain() Display all trains

getTrain(trainID):

Get a specific train object by ID

## MyQueue Class

Data Members

Int frontIndex, rearIndex, capacity

T\* arr: Dynamic array to store queue

### Methods:

Enqueu(value):

Adds item to queue

Dequeue(): Remove the front item

Front()Return front item

isEmpty():Check if queue is empty

size()returns number of items

## UserManager Class

**Data Members:**

unordered\_map<string, User> users: Map of username to user.

MyQueue<User> registrationQueue: Queue to store newly registered users.

### Methods:

**registerUser()**:  
Adds a new user to the map and registration queue.

**loginUser()**:  
Validates user credentials.

**showRegistrationQueue()**:  
Displays usernames in the queue.

**getUser(username)**:  
Returns pointer to a specific user.

# UML Class Diagram

A diagram of a train

AI-generated content may be incorrect.

# SCOPE:

The focus of this project is to develop a good to go software for managing the Railway system, efficiently managing the passengers, showing the available trains, their routes and the number and position of seats available. It’s main aim is to provide ease for both the passengers and the organizers or stakeholders of the service.

# LIMITATIONS

Absence of real time updates

For large data sets it will not be functional as long as a proper database is built or provided over cloud

Very less implementations of security services and privacy.

Due to the use of basic algorithms, only basic functionalities of the railway reservation system are implemented and not high-level functionalities are implemented.

# CONCLUSION

This project tries to demonstrate a railway reservation system with limited and basic functionalities. With basic data structures being implemented including LinkedList, Stack, Queues and unordered\_map, this indicates the understanding of these topics. This project lies foundation for further development as it can be extended to a real-world application by implementing a GUI and other functionalities that were mentioned in the limitations part of the document.