**Semester Project Doucmentation :**

**Railway Reservation System**

**Software Documentation**

**1. Introduction**

The Railway Reservation System is a console-based application designed to facilitate train management, including train details, seat bookings, and route updates. The system features two primary user roles: **Admin** and **Customer**, each with specific functionalities.

**2. Features**

**Admin Features:**

1. **Add Train**
   * Admins can add new trains with details like name, route, train number, and total seats.
2. **Update Route**
   * Update the route of a specific train by its name.
3. **Delete Train**
   * Remove a train from the system by its name.
4. **View Trains**
   * Display all available trains with their details.

**Customer Features:**

1. **View Train Information**
   * View details of all available trains.
2. **Book Seats**
   * Book seats on a specific train based on availability.
3. **Display Remaining Seats**
   * Check the number of available seats for a specific train.

**3. Classes and Methods**

**Class: Train**

Represents individual train details.

**Attributes:**

* name (string): Train name.
* route (string): Train route.
* Train\_number (int): Unique train identifier.
* no\_of\_seats (int): Total number of seats.
* booked\_seats (int): Number of booked seats.
* next (Train\*): Pointer to the next train in the list.
* previous (Train\*): Pointer to the previous train in the list.

**Methods:**

1. **Getters and Setters:**
   * getName(), setName(string)
   * getRoute(), setRoute(string)
   * getTrainNumber(), setTrainNumber(int)
   * getNoOfSeats(), setNoOfSeats(int)
   * getBookedSeats(), setBookedSeats(int)
   * getAvailableSeats(): Calculate available seats.
   * getNext(), setNext(Train\*)
   * getPrevious(), setPrevious(Train\*)

**Class: Railway\_Reservation\_System**

Manages all trains and user interactions.

**Attributes:**

* head (Train\*): Pointer to the first train in the doubly linked list.

**Methods:**

1. **Admin Functions:**
   * add\_train(): Add a new train to the system.
   * delete\_train(): Delete a train by its name.
   * update\_route(): Update the route of a specific train.
   * view\_trains(): Display all train details.
2. **Customer Functions:**
   * book\_seat(): Book seats on a specific train.
   * display\_remaining\_seats(): Check remaining seats for a train.
3. **Authentication and Menus:**
   * LogIn(): Authenticate admin users with username and password.
   * Admin\_Menu(): Menu for admin functionalities.
   * Customer\_Menu(): Menu for customer functionalities.

**4. System Flow**

**Admin Flow:**

1. Admin selects "Admin" from the main menu.
2. Admin logs in using the predefined credentials:
   * Username: 123
   * Password: 1234
3. Admin can perform tasks such as adding, deleting, or viewing trains.

**Customer Flow:**

1. Customer selects "Customer" from the main menu.
2. Customer can view train details, book seats, or check available seats.

**5. Key Highlights**

1. **Doubly Linked List:**
   * The trains are stored using a doubly linked list, enabling efficient traversal and updates.
2. **Seat Management:**
   * Ensures proper seat allocation and availability checks during booking.
3. **Modular Design:**
   * Separate functions for each task improve code readability and maintainability.

**6. Sample Console Interaction**

**Main Menu:**

mathematica

Copy code

Railway Reservation System

1) Admin

2) Customer

3) Exit

Enter choice:

**Admin Menu:**

css

Copy code

Admin Menu

1) Add Train

2) Update Route

3) Delete Train

4) View Trains

5) Exit to Main menu

Enter a choice:

**Customer Menu:**

css

Copy code

Customer Menu

1) View Train Informations

2) Book Seats

3) Display remaining seats

4) Exit to Main menu

Enter a choice:

**7. Future Enhancements**

1. Integration of a database for persistent storage of train and booking data.
2. User authentication for customers with account creation and login.
3. Enhanced seat allocation with dynamic pricing and class selection.
4. A graphical user interface (GUI) for improved usability.

**8. Conclusion**

This Railway Reservation System efficiently manages train operations and user interactions, demonstrating the use of object-oriented programming concepts in C++. It serves as a foundation for further enhancements and real-world applications.