Train Reservation System

A Railway Management Project in C++ using OOPS

Project Members:
Vansh Khandelwal

Intro duction

This project implements a console-based Train Reservation System in C++ using Object-Oriented Programming (OOP). The system supports:

- User authentication (Standard User, Admin)
- Viewing available trains
- Booking and canceling tickets
- Managing customer profiles and ticket records

1. Classes

Base / Abstract / Interface Classes

- TrainDistance – Abstract base class for distance-related logic.
- DisplayClass – Abstract base class for standardized display methods.

Utility / Helper Classes

- RandomGenerator – Used for generating random train numbers, seats, destinations, etc.
- RolesAndPermissions – Manages admin and passenger authentication and registration.

Core Application Classes

- User – Main controller class (acts like AppController).
- Train – Represents a train and its schedule, passengers, etc.
- Customer – Represents a user/passenger with ticket info.
- TrainReservation – Handles booking logic and passenger-train interactions.

1.1. 1. Customer

(a) Customer.hpp

- Declares the Customer class with private members like User ID, name, email, phone number, password, address, age.
- Contains vectors for:
  - Train* pointers representing booked trains.
  - Integers representing ticket count per train.
- Provides methods for:
  - Getters and setters for customer data.
  - Managing train bookings.
  - Static handling of all customers through customerCollection.
  - Validation like isUniqueData() and display utilities.

(b) Customer.cpp

- Implements logic to add/edit/delete/search customers.
- Uses STL functions such as std::find if, std::remove if, etc.
- Handles many-to-many relationship between trains and customers.
- Maintains static storage: customerCollection.

1.2. 2. Train

(a) Train.hpp

- Inherits from TrainDistance abstract class.
- Contains data such as train number, route, schedule, seats, platform, passengers.
- Uses Customer.hpp and RandomGenerator.hpp.

- Maintains:
  - Static trainList and nextTrainDay.
  - Passengers list.
- Functions:
  - Schedule generation using trainScheduler().
  - CRUD-like methods.
  - Travel metrics calculation via calculateDistance() and calculateTravelTime().

(b) Train.cpp

- Implements train creation, passenger management, display, and scheduling.
- Uses std::vector<unique ptr<Train>> and Customer* lists.
- Employs Haversine formula for distance.
- Formats output using toString() and displayTrainSchedule().

1.3. 3. TrainReservation

(a) TrainReservation.hpp

- Inherits from DisplayClass.
- Uses Train and Customer.
- Memb er: trainIndexInCustomerList .
- Functions:
  - bookTrain() , cancelTrain() .
  - Helpermethodslike isTrainInCustomerList() , trainStatus().
  - Display methods to show train and user bookings.

(b) TrainReservation.cpp

- Uses global access to customer and train lists.
- Implements logic for booking, cancellation, and data display.
- Includes ASCII UI functions like displayArtWork().

1.4. 4. RolesAndPermissions

(a) RolesAndPermissions.hpp

- Manages authentication of admins and passengers.
- Members: adminAccounts stores all admin credentials.
- Metho ds:
  - isPrivilegedUserOrNot() — checks if admin credentials are valid.
  - registerAdmin() — registers a new admin.
  - isPassengerRegistered() — checks customer credentials from global data.

(b) RolesAndPermissions.cpp

- Implements login verification logic for both admins and passengers.
- Ensures safe access to system resources via secure login.

1.5. 5. RandomGenerator

(a) RandomGenerator.hpp

- Uses Mersenne Twister RNG for reproducibility.
- Metho ds:
  - randomTrainNumberGen() — generates alphanumeric train IDs.
  - randomNumOfSeats() — generates random seat numbers.
  - randomDestinations() — provides source/destination data.

(b) RandomGenerator.cpp

- Implements above randomization logic.
- Provides real-world feel with simulated station data and schedules.

1.6. 6. User

(a) User.hpp

- Provides static methods for user interface and app lifecycle control.
- Functions:
  - run(), displayMainMenu(), manualInstructions().
  - printArtWork() — to display headings and ASCII art.

(b) User.cpp

- Acts as the central controller for the application.
- Contains the main() function.
- Manages application flow:
  - Admin Login and CRUD
  - Passenger Login, Booking, and Cancellations
  - Registration and Manual
- Handles command-line interface and screen formatting.

2. Design Principles

The project applies the following OOP design principles:

- Encapsulation: Secure handling of data via access specifiers.
- Single Responsibility Principle: Each class performs a unique function.
- Modularity: Business logic, UI, and data access are separated.
- Reusability: Shared logic via utility and abstract base classes.

Conclusion

This group project demonstrates a practical implementation of Object-Oriented Program- ming in C++. It offers a modular and maintainable solution for a railway reservation system, integrating real-world logic and collaborative development.