Public Transportation Station Management System - Documentation

1. Object-Oriented Analysis (OOA)

In this project, the problem domain is a city's public transportation network (buses and express buses). We identify the following key objects and their attributes/methods: - Vehicle: route, capacity, status, book seats, cancel seats, calculate travel time. - ExpressBus (inherits from Vehicle): additional speed attribute, overrides travel time calculation. - Station: name, location, list of vehicles; can add/display vehicles. - BusStation (inherits from Station): manages ExpressBuses specifically. - Passenger: name, ID, can book/cancel vehicles, stores booking history. - Schedule: time, type (arrival/departure), associated vehicle. Relationships: - Inheritance: Vehicle -> ExpressBus, Station -> BusStation. - Composition: Station contains Vehicles, Passenger contains booked Vehicles, Schedule references Vehicle.

2. Class Design and Inheritance

The class design demonstrates inheritance and polymorphism: - Vehicle is the base class. ExpressBus inherits from Vehicle and overrides calculateTravelTime() to adjust travel time with higher speed and efficiency. - Station is the base class. BusStation inherits from Station to handle ExpressBus separately. - Encapsulation is applied with private/protected attributes and getter methods. - Passenger interacts with Vehicle objects by booking/canceling seats. - Schedule links time and type to a Vehicle object. Inheritance is used to avoid duplication and promote reusability. Polymorphism allows different types of vehicles to calculate travel time differently.

3. Code Walkthrough

Key parts of the C++ implementation: - Vehicle class: stores route, capacity, booked seats; provides bookSeat() and cancelSeat() methods. - ExpressBus class: adds speed attribute and overrides calculateTravelTime(). - Station and BusStation: manage vehicles and provide display functionality. - Passenger: can book and cancel vehicles; ensures capacity is not exceeded. - Schedule: associates a vehicle with a departure or arrival time. The main() function demonstrates usage by creating stations, vehicles, passengers, schedules, and running booking and cancellation operations.

4. Test Results

```
Sample outputs include:

- Booking until capacity is full triggers a 'Vehicle is full!' message.
- Canceling a non-existent booking notifies the user.
- Passengers can book multiple different vehicles.
- Multiple schedules display correct vehicle and booking data.
- Travel time differs depending on the type of vehicle (normal bus vs. express bus).

Example output snippet:

Vehicles at Central Bus Station:
- Route: Bus 101, Capacity: 3, Booked: 2
- Route: Express 201, Capacity: 2, Booked: 1

Alice booked a seat on Bus 101
Bob booked a seat on Bus 101
Vehicle Bus 101 is full!
Alice canceled booking on Bus 101
```

5. LLM Usage

I used ChatGPT (an LLM AI model) as an assistant in the design phase of this project. Specifically, I asked the model to suggest inheritance hierarchies for vehicles in a transportation system and to provide feedback on my C++ code structure. For example, one of my prompts was: "Suggest inheritance hierarchies for vehicles in a transportation system." ChatGPT responded with an idea to create a base Vehicle class and derive Bus, Train, and ExpressBus. I adapted these suggestions to my project context and implemented my own code. The LLM was used to brainstorm ideas and improve clarity, but the final C++ implementation, testing, and debugging were completed by me.