Public Transportation Station Management System

1. Object-Oriented Analysis (OOA) :

The OOA model for the Public Transportation Station Management System follows a 4-step process:

a. ***Identify Objects***: Station, Vehicle, ExpressBus, Passenger, Schedule.
b. ***Identify Attributes***:
- Station: name, location, type, schedules (vector).
- Vehicle: route, capacity, booked, onTime, speed.
- ExpressBus: Inherits Vehicle attributes, adds speedMult, stops.
- Passenger: name, id, bookedRoutes (vector of strings).
- Schedule: time, vehicleRoute, isArrival.

c. ***Identify Methods***:
- Station: addSchedule(), removeSchedule(), displayInfo().
- Vehicle: calcTravelTime(), bookPassenger(), cancelPassenger(), displayInfo().
- ExpressBus: Overrides calcTravelTime(), displayInfo().
- Passenger: bookTicket(), cancelTicket(), displayInfo().
- Schedule: display(),

d. ***Identify Inheritance:*** ExpressBus inherits from Vehicle, enabling polymorphism for calcTravelTime() and display().

2. Class Design

The system uses five classes:
- **Vehicle**: Base class with attributes (route, capacity, booked, onTime, speed) and virtual methods (calcTravelTime(), display()) for polymorphism.
- **ExpressBus**: Inherits from Vehicle, adds speedMult and stops, overrides calcTravelTime() (20% faster) and display() to show additional details.
- **Station**: Manages schedules (vector, max 10) with methods to add/remove/display schedules.
- **Passenger**: Tracks booked rides (by route strings) with book/cancel methods.
- **Schedule**: Stores time, vehicleRoute, and isArrival for scheduling.

- > Inheritance is used to avoid duplicating code in ExpressBus, reusing Vehicle's attributes and methods while customizing travel time and display. A static vector in Vehicle tracks all vehicles, replacing pointers in Schedule/Passenger for memory safety and to meet the requirement of avoiding pointers.

3. Code Walkthrough

The C++ code (TransStation.cpp) implements the system:

- **Vehicle Class**: Stores route, capacity, and speed; provides virtual calcTravelTime() (distance/speed) and bookPassenger() with capacity checks. A static vector tracks all vehicles, allowing lookup by route.
- **ExpressBus Class**: Extends Vehicle, overrides calcTravelTime() for 20% faster travel using speedMult.
- **Schedule Class:** Stores vehicleRoute (string) instead of pointers, links to Vehicle via static findVehicle().
- **Station Class**: Uses vector for schedules, enforces max 10 schedules, checks vehicle existence before adding.
- **Passenger Class**: Manages bookedRoutes (vector of strings), uses findVehicle() for booking/canceling.
- **Main Function**: Creates stations, vehicles, passengers; tests scheduling, booking, canceling, and displays results.

Key design choice: Avoided pointers in Schedule/Passenger by using route strings and a static Vehicle vector, ensuring no memory leaks while maintaining polymorphism.

4. Test Results

Schedule added at station Central Station
Station: Central Station (Bus), Location: Downtown
Schedules:
Arrival at 12:00 -> Vehicle Route: Bus Route 1, Capacity: 2, Booked: 0, Status: On-time
Departure at 17:00 -> Express Bus Route: Express Route A, Capacity: 2, Booked: 0, Status: On-time, Speed: 100 km/h

==========
Passenger TaiHuynh booked ticket for route: Bus Route 1
Booking successful! Seats booked: 1/2
Passenger Xuan Phung booked ticket for route: Bus Route 1
Booking successful! Seats booked: 2/2
Booking failed. Vehicle is full!

==========
Vehicle Route: Bus Route 1, Capacity: 2, Booked: 2, Status: On-time
Express Bus Route: Express Route A, Capacity: 2, Booked: 0, Status: On-time,

Speed: 100 km/h
Normal bus travel time for 100 km: 2 hours
Express bus travel time for 100 km: 1 hours
==========

Passenger ID: 101, Name: TaiHuynh
Booked tickets: Bus Route 1
Passenger ID: 102, Name: Xuan Phung
Booked tickets: Bus Route 1
Passenger Alice canceled ticket for route: Bus Route 1
Passenger ID: 101, Name: TaiHuynh
Booked tickets: None

- > This demonstrates:
+ Scheduling with arrival/departure details.
+ Booking tickets with capacity checks (error when full).
+ Faster travel time for ExpressBus (0.8 vs. 2 hours).
+ Canceling tickets and updating passenger/vehicle states.

5. LLM Usage:

I used Grok and ChatGPT to assist with code optimization and error checking. Prompt: "Suggest ways to remove pointers from Schedule and Passenger classes while maintaining polymorphism." LLM proposed using route strings and a static vector in Vehicle, which I adapted to avoid object slicing and ensure memory safety. I also asked Grok to check the C++ code for compilation errors and suggest a concise documentation structure. All code was written by me, with LLM providing suggestions for refinement.