Transport Management System (TMS) - Backend Developer Assignment

🎯 Objective

Build a backend system for a **Transport Management System (TMS)** using **Spring Boot** and **PostgreSQL**. This assignment tests your ability to handle real-world logistics challenges with complex business rules.

📦 Core Entities

1. Load Entity

```
{
  "loadId": "UUID",
  "shipperId": "String",
  "loadingCity": "String",
  "unloadingCity": "String",
  "loadingDate": "Timestamp",
  "productType": "String",
  "weight": "double",
  "weightUnit": "KG | TON",
  "truckType": "String",
  "noOfTrucks": "int",
  "status": "POSTED | OPEN_FOR_BIDS | BOOKED | CANCELLED",
```

```
  "datePosted": "Timestamp"
}
```

2. Transporter Entity

```
{
  "transporterId": "UUID",
  "companyName": "String",
  "rating": "double (1-5)",
  "availableTrucks": {
    "truckType": "String",
    "count": "int"
  }[]
}
```

3. Bid Entity

```
{
  "bidId": "UUID",
  "loadId": "UUID",
  "transporterId": "UUID",
  "proposedRate": "double",
  "trucksOffered": "int",
  "status": "PENDING | ACCEPTED | REJECTED",
  "submittedAt": "Timestamp"
}
```

4. Booking Entity

```
{
  "bookingId": "UUID",
  "loadId": "UUID",
  "bidId": "UUID",
  "transporterId": "UUID",
  "allocatedTrucks": "int",
  "finalRate": "double",
  "status": "CONFIRMED | COMPLETED | CANCELLED",
  "bookedAt": "Timestamp"
}
```

🔥 Critical Business Rules (Must Implement)

Rule 1: Capacity Validation

- ✅ Transporter can only bid if trucksOffered ≤ availableTrucks for that truck type
- ✅ When booking is confirmed, deduct allocatedTrucks from transporter's availableTrucks
- ✅ When booking is cancelled, restore trucks to available pool

Rule 2: Load Status Transitions

```
POSTED → OPEN_FOR_BIDS (when first bid received)

OPEN_FOR_BIDS → BOOKED (when bid accepted)

BOOKED → CANCELLED (if booking cancelled)
```

- ❌ Cannot bid on CANCELLED or BOOKED loads
- ❌ Cannot cancel load that's already BOOKED

Rule 3: Multi-Truck Allocation

- If noOfTrucks > 1, allow multiple bookings until fully allocated
- Track: remainingTrucks = noOfTrucks - SUM(allocatedTrucks)
- Load becomes BOOKED only when remainingTrucks == 0

Rule 4: Concurrent Booking Prevention

- Use **optimistic locking** (@Version) to prevent double-booking
- If two transporters accept simultaneously, first transaction wins
- Second should fail with ConflictException

Rule 5: Best Bid Calculation

Implement **GET /load/{loadId}/best-bids** sorted by:

score = (1 / proposedRate) * 0.7 + (rating / 5) * 0.3

Higher score = better bid

🔹 Required APIs (15 Total)

Load APIs (5)

1. **POST /load** → Create load (status = POSTED)
2. **GET /load?shipperId=&status=&page=0&size=10** → List with pagination
3. **GET /load/{loadId}** → Get load with active bids
4. **PATCH /load/{loadId}/cancel** → Cancel load (validate status)
5. **GET /load/{loadId}/best-bids** → Get sorted bid suggestions

Transporter APIs (3)

1. **POST /transporter** → Register transporter with truck capacity
2. **GET /transporter/{transporterId}** → Get details
3. **PUT /transporter/{transporterId}/trucks** → Update available trucks

Bid APIs (4)

1. **POST /bid** → Submit bid (validate capacity & load status)
2. **GET /bid?loadId=&transporterId=&status=** → Filter bids
3. **GET /bid/{bidId}** → Get bid details
4. **PATCH /bid/{bidId}/reject** → Reject bid

Booking APIs (3)

1. **POST /booking** → Accept bid & create booking (deduct trucks, handle concurrency)
2. **GET /booking/{bookingId}** → Get booking details
3. **PATCH /booking/{bookingId}/cancel** → Cancel booking (restore trucks, update load status)

🛠️ Technical Requirements

Stack (Mandatory)

- Spring Boot 3.2+, Java 17+
- Spring Data JPA with PostgreSQL
- Global Exception Handling (@ControllerAdvice)

Architecture

Controller → DTO → Service (Business Logic) → Repository → Entity

Database Design Must Have:

- Foreign key constraints with proper cascading
- Unique constraint: One ACCEPTED bid per load
- Index on: loadId, transporterId, status
- Optimistic locking: @Version column in Load entity

Exception Handling

- InvalidStatusTransitionException

- InsufficientCapacityException

- LoadAlreadyBookedException (OptimisticLockException)

- ResourceNotFoundException

📩 Submission Guidelines

Repository Structure

```
├── src/
│   ├── main/java/com/yourname/tms/
│   │   ├── controller/
│   │   ├── service/
│   │   ├── repository/
│   │   ├── entity/
│   │   ├── dto/
│   │   └── exception/
│   └── test/
├── README.md (Setup + API docs + Schema diagram)
```

README.md Must Include:

1. **Database Schema Diagram** (draw.io or image)
2. **API Documentation** (Swagger URL or Postman collection link)
3. **Test Coverage Screenshot**

Email to: careers@cargopro.ai

Subject: YourName_Backend_TMS_Assignment
Attach: Resume + GitHub link

📊 Evaluation Criteria

Criteria	What We Check
Business Logic	Capacity validation, status transitions, concurrency
Code Quality	SOLID principles, clean code, no hardcoded values
DB Design	Normalization, constraints, indexing
Documentation	README clarity, API docs

🚀 What We're Looking For

We want candidates who can:

- ✅ Understand complex business requirements
- ✅ Handle edge cases without being told
- ✅ Write testable, maintainable code
- ✅ Make smart trade-off decisions
- ✅ Explain their design choices clearly

AI tools are allowed, but you must understand every line of code. In the interview, we'll ask you to:

- Modify code live

- Explain why you chose certain approaches
- Debug issues on the spot

Questions? Email careers@cargopro.ai with subject: Question_TMS_Assignment_YourName