Full-Stack Senior Developer Assignment

(Python FastAPI + React)

Overview

You are required to build a **full-stack application** consisting of:

- A **PostgreSQL** database
- A **REST API** built with **FastAPI (Python)**.
- A **React frontend** using **Vite** that consumes the API.
- A **User-Roles-Permissions** system.
- Full **CRUD** management for car data:
    - Brands
    - Models
    - Submodels
    - Generations
    - Base car specifications (year, engine, horsepower, etc.)

The objective is to demonstrate senior-level knowledge in architecture, clean code, security, and UI/UX.

Functional Requirements

User roles

The system must support three user roles:

Admin

- Has **all permissions** (full CRUD for users and car information).
- Can assign roles and permissions to other users.

CarSpec

- Can **create and update car information**.

- Can **delete only the items that *they* created**.

User

- Can **read all car information**.
- Can **add/delete cars to "My Cars" list**.

Authentication & Authorization

- Implement **JWT-based authentication**.
- Protect routes based on role and permissions.

List pages

All the list pages must have an:

- Sorting fields (can be sorted by more than one criteria)
- Filtering fields (with multiple options)
- Pagination

Technical Requirements

Use Docker compose for DB, backend and frontend

Backend

- PostgreSQL database
- FastAPI
- SQLAlchemy (async preferred)
- Alembic migrations
- Pydantic models
- JWT auth

Frontend

- React (using **Vite**)
- Use:
  - TypeScript
  - @tanstack/react-router (File-Based Routing)

    - @tanstack/react-form
    - @tanstack/react-query
    - Zod
    - axios
    - Tailwind
- Proper folder architecture
- Form validation
- Error handling and loading states

Bonus Points

- Use Entity(DB model) > Repository > Service > Controller; based pattern
- Using Swagger for the backend documentation with examples
- Write some UnitTests for some service
- Add Seed data

Deliverables

- GitHub repository containing:
    - Backend folder
    - Frontend folder
    - README with setup instructions
- SQL schema or migrations included
- Optional Postman collection

Evaluation Criteria

- Architecture & code quality
- Role/permission logic correctness
- API design quality
- Clean UI and UX
- Error handling
- Security awareness
- Maintainability & readability

- Senior-level reasoning in solution structure