Liatrio Lead AI Software Engineer Exercise

Directions of how to approach this Demo:

You will receive this exercise 2-3 days ahead of the scheduled demo time with the team in order to work and prepare for the demonstration.

We want you to complete the exercise but with the intent of showcasing the workflow on how to build and to talk through technical architecture that “could be” possible.

We want you to explain your approach to AI-native development and give a short 5-6 slide presentation on the “why” adopting a new workflow is going to be important

You should be ready to talk about technology, but most importantly, give us a 15-20 minute overview of the importance of how AI will shape change for enterprise technology. (You can expand on any aspect of the change; technology, agentic development, organizational change/change management)

Parking Garage Management Web API

Exercise Overview/Technical Overview:

You are tasked with building the initial version of a **Parking Garage Management System**. Your client owns a garage and needs a basic **Web API** to manage parking spots and track cars as they enter and exit.

Be prepared to discuss your design decisions, trade-offs, and potential extension opportunities in a follow-up conversation.

We encourage you to use AI tools to assist with your work — effective use of these tools is part of what we’re evaluating.

Project Requirements:

Core Features

1. **Your API should support:**

- **Garage Layout**:
    - Manage *floors* and *bays* (areas within a floor).
    - Define and manage *parking spots* with unique identifiers.

- **Parking Spot Management**:
    - List all parking spots with their availability status (available/occupied).
    - Retrieve only available spots.
    - Ability to mark spots as occupied or available.
- **Car Tracking**:
    - Check a car *in*: Assign the car to an available spot.
    - Check a car *out*: Free up the spot.
    - Track check-in and check-out times.

2. **Minimum Data Fields**

2a. For Parking Spots:

- Spot ID
- Floor
- Bay
- Spot number
- Status (available/occupied)

2b. For Cars:

- License plate number
- Assigned spot ID
- Check-in timestamp

---

3. Stretch Goals:

If you complete the basics or as time permits:

- **Search**:
  - Look up a car by license plate.
- **Spot Types**:
  - Introduce different spot sizes (compact, standard, oversized) and enforce compatibility with car sizes.
- **Rate Calculation**:
  - Implement basic billing: calculate parking fees based on check-in and check-out times (e.g., an hourly rate).
- **Spot Features**:
  - Add support for special spot types (e.g., EV charging stations with different pricing).

General Expectations

- **In-memory data storage** (no database required).
- **Web API** (RESTful) — JSON-based input and output.
- No authentication or authorization required.
- Focus on code clarity, structure, and functionality.

- You are free to choose any programming language and framework (e.g.,
- Python/FastAPI, Node.js/Express, Java/Spring Boot, etc.).

What We'll Discuss Afterwards

Be prepared to:

- **Walk us through** your design choices and trade-offs.
- **Discuss extension opportunities**: how your design could be adapted or expanded for additional features or business needs.

  Share any creative ideas: if there are improvements or extensions you think would be valuable, we'd love to hear about them.

Creativity Encouraged:

Feel free to implement additional functionality or propose ideas that you believe would improve the system. Clear communication of your ideas and decision-making process is just as important as the code itself.

✅ **Goal**: Get as far as you can from now until Friday and be ready to dive deeper during the follow-up discussion/ presentation!