Assignment: AI Orchestrator with Containers

Overview

In this 24-hour challenge, you'll build a small project that demonstrates:

- **LLM (Large Language Model) Integration**
- **Containerization** and **Orchestration**
- **Backend/Systems Design**

Task

Create a mini 'AI Orchestrator':

1. The user provides a high-level request (e.g., "Clean this dataset," "Analyze sentiment in this text").
2. Your system's LLM agent parses the request and selects one or more containerized tasks.
3. The orchestrator runs each container, collects the output, and returns results to the user.

If you don't have an LLM API key, consider **Groq** for free account access.

Requirements

1. **LLM Integration**
    - Connect to an LLM that decides which container(s) to run.
    - Show a simple prompt design or rule set for the agent's decision-making.
2. **Containerized Services**
    - Each data-processing step must be in a Docker container.
    - Demonstrate how you build, run, and pass data to/from these containers.
3. **System Design**
    - A central orchestrator service (could be a simple Python, Node, or any language backend) that:
        - Receives the user's request.
        - Calls the LLM.
        - Spins up containerized tasks in the correct order.
        - Collects final output and returns it.

4. **User Interface**
    - Provide a simple UI or CLI to:
        - Accept the request.
        - Display logs or final outputs.

Deliverables

Upload following to drive and share link at **career@bioquix.com**
Feel free to make reasonable assumptions wherever necessary, and ensure to document them clearly in your submission.

- **Code & Configuration**
    - Source code for the orchestrator.
    - Dockerfiles or Docker Compose for each container.
- **README**
    - Clear Setup steps.
    - Brief architecture explanation (can include a diagram).
- **Demo video**
    - Record screen to show at least one example input and the resulting output.

Time Limit

- You have **48 hours** to submit.
- Aim for a clean, working solution rather than a perfect production-level system.

Evaluation Criteria

- **System Clarity:** How clearly is the orchestrator and container architecture designed?
- **Functionality:** Does the system correctly run containerized tasks based on LLM decisions?
- **Code Quality:** Are the code and documentation understandable?
- **Efficiency & Simplicity:** A minimal, working solution is preferred over an overly complex one.