Technical Task

Situation

Hello, our future Data & AI Engineer. The following exercise has been designed to test your API design, retrieval system development and data processing skills using **Python**, **LangChain** and **FastAPI**.

Resources Provided

~$100 USD worth of OpenAI API credits

API key:

`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`

Your task

The task will require you to build a simple Retrieval-Augmented Generation (RAG) system that can answer questions about Oxylabs' developer documentation from https://developers.oxylabs.io/.

Step 1: Data Collection

1. Visit https://developers.oxylabs.io/ and choose one product section;
2. Manually copy content from 5-10 key pages into separate .txt files;
3. Include main content only (explanations, code examples, parameters).

Step 2: RAG System Implementation

Using **LangChain**, implement:

1. **Document Processing**:
    - Load your text files using **LangChain** document loaders;
    - Split into chunks (aim for 500-1000 chars per chunk).
2. **Vector Storage**:

    - Use **ChromaDB** with **LangChain** integration;
    - Create embeddings using **OpenAI's** embedding model.
3. **RAG Chain**:
    - Build a retrieval chain for similarity search;
    - Use **OpenAI's** chat model for generation.

Step 3: FastAPI Backend

Create a minimal **FastAPI** app with:

1. **POST /query**:
    - Input: {"question": "your question here"}
    - Output: {"answer": "generated answer", "sources": ["source chunks"]}
2. **Basic error handling** and request validation using Pydantic models.

Step 4: Containerization

Provide your solution in a containerized format:

- Create a Dockerfile;
- Include a docker-compose.yml (optional but preferred);
- Include a requirements.txt;
- Add basic setup instructions in README.

(Later) Step 5: Presentation (~20 mins)

1. **Showcase** your system with a couple of relevant questions about Oxylabs products and services (running the application *locally*).
2. **Present limitations** of your current implementation and areas where it might fail.
3. (Important) **Propose improvements** for a production-ready version, considering aspects like:
    - Scalability and performance;
    - Data freshness and update mechanisms;
    - Security and authentication;
    - Monitoring and observability;
    - Cost optimization.

Note: You don't need a production-ready system. Focus on demonstrating your understanding of RAG concepts, clean code practices, and system design thinking. Feel free to use any existing *tutorials, examples, or frameworks as starting points. Likewise, it is okay (or even advisable) to use AI Code editors (e.g. Cursor, Copilot) or agentic coding tools (e.g. Claude Code, Gemini CLI) to complete the exercise.*

Good luck!