Interview Questions and Answers

Q: How do you handle dependency injection in FastAPI?

A: Use FastAPI's Depends system. Define a function that returns the dependency and FastAPI injects it into path functions.

Q: What are Pydantic models and how are they used in FastAPI?

A: Pydantic models validate and serialize data. They're used to define request/response schemas.

Q: Explain the difference between synchronous and asynchronous routes in FastAPI.

A: Synchronous (def) blocks the thread. Asynchronous (async def) is non-blocking and better for I/O-bound operations.

Q: How do you implement RBAC in a FastAPI application?

A: Use Depends to extract user role and apply conditional access control logic in each route.

Q: How would you handle file uploads and downloads using FastAPI?

A: Use UploadFile for uploads and FileResponse for downloads.

Q: What is a RunnableSequence in LangChain and how does it compare to LLMChain?

A: RunnableSequence is a modular approach introduced in LangChain v0.1+. LLMChain is older and less flexible.

Q: How would you implement a SQL agent with LangChain?

A: Use SQLDatabaseToolkit and an agent executor to convert natural language to SQL using an embedded database schema.

Q: Explain how you would optimize a SQL query generated by an LLM.

A: Chain the raw query into another LLM prompt asking it to optimize for performance.

Q: How do you ensure that the LLM answers only based on retrieved documents (RAG)?

A: Retrieve top-k documents from a vector DB and include them in the prompt. Instruct the LLM to answer only using the context.

Q: How do you load and use prompt templates from external files in LangChain?

A: Read prompt from .txt file and pass to PromptTemplate with input variables.

Q: How does RAG architecture improve response accuracy in LLM-based applications?

A: It grounds the LLM's response in retrieved context, reducing hallucination.

Q: Describe how you implemented a RAG pipeline using ChromaDB.

A: Split documents, embed them, store in ChromaDB, retrieve top-k docs and use them with the LLM.

Q: What are the pros and cons of using website content directly for RAG instead of scraping?

A: Pros: simplicity. Cons: lack of control, dynamic content may be missed.

Q: How do you convert CSV data into SQL-like queries using LangChain?

A: Load CSV to pandas DataFrame and use LangChain agents or tools to interpret natural language queries.

Q: How do you determine the correct chart type based on user input?

A: Use prompt classification or an LLM chain to match query to chart type.

Q: How do you convert charting results into JSON format for use in frontend apps?

A: Structure the output into a dictionary with keys like 'chart_type', 'labels', and 'values'.

Q: How do you Dockerize a FastAPI app?

A: Use a Dockerfile with python base image, copy code, install dependencies, and start using uvicorn.

Q: Describe your CI/CD pipeline for deploying FastAPI apps on Azure Web App.

A: Use GitHub Actions to build and push Docker image to Azure Container Registry and deploy to Azure App Service.

Q: What are the benefits of using Poetry with Python projects?

A: It simplifies dependency management, environment isolation, and produces a lock file for reproducibility.

Q: How do you handle version control and package dependencies?

A: Use Git for versioning, requirements.txt or poetry.lock for dependency locking, and .env for secrets.

Q: How would you enforce access control on a per-role basis in an LLM-driven app?

A: Use JWT/session metadata to determine roles and selectively run prompts or SQL logic based on roles.

Q: How do you prevent prompt injection attacks in your LangChain pipelines?

A: Avoid direct string concatenation, validate user inputs, and use structured inputs.

Q: How do you structure prompts to guide LLM behavior (e.g., for classification, filtering)?

A: Use clear task instructions and few-shot examples for consistent classification output.

Q: What's your approach to debugging prompt-based issues in an LLM pipeline?

A: Log inputs/outputs, iterate prompt design, and test various edge cases. Use tools like LangSmith.