Full-Stack Engineering Internship Assignment

Objective

Develop a No-Code/Low-Code web application that enables users to visually create and interact with intelligent workflows. The application should allow users to configure a flow of components that handle user input, extract knowledge from documents, interact with language models, and return answers through a chat interface.

Once a user builds a valid workflow, they should be able to ask questions. The system should process the query using the defined components and return a final response.

Tech Stack Requirements

- **Frontend**: React.js
- **Backend**: FastAPI
- **Database**: PostgreSQL
- **Drag & Drop Library**: React Flow
- **Vector Store**: ChromaDB (or similar)
- **Embedding Model**: OpenAI Embeddings
- **LLM**: OpenAI GPT , Gemini
- **Web Search Tool**: SerpAPI | Brave
- **Text Extraction**: PyMuPDF or similar

Core Components

The application must allow users to build workflows using the following four components:

1. User Query Component

- Accepts user queries via a simple interface.
- Serves as the entry point for the workflow.
- Sends the query forward to the next connected component.

2. KnowledgeBase Component

- Allows uploading and processing of documents (e.g., PDFs).
- Extracts text from files (using PyMuPDF or similar).
- Generates embeddings from the text using OpenAI Embeddings , gemini embedding models etc.
- Stores embeddings in a vector store (e.g., ChromaDB).
- Retrieves relevant context based on the user query.
- **Context passing to the LLM Engine is optional.**

3. LLM Engine Component

- Accepts:
    - Query from the User Query Component
    - **Optional** context from the KnowledgeBase Component
    - **Optional** custom prompt
- Sends request to an LLM (e.g., OpenAI GPT , Gemini) to generate a response.
- Optionally uses **SerpAPI** to retrieve information from the web.
- Outputs a response and sends it to the Output Component.

4. Output Component

- Displays the final response to the user.
- Should function as a chat interface.
- Follow-up questions should re-run the workflow using the same logic.

Workflow Execution

Build Stack

- Users connect components in a logical order to define the workflow.
- The application validates the workflow for correctness and configuration.

Chat with Stack

- Users can enter queries in a chat interface.
- Each query is passed through the workflow:
  User Query → (Optional) KnowledgeBase → LLM Engine → Output
- The final response is shown in the chat.

Frontend Specification

Component Library Panel

- Lists all four available components.
- Components can be dragged onto the canvas.

Workspace Panel

- Visual canvas for building workflows using React Flow.
- Should support:

    - Drag-and-drop
    - Connection lines with arrows
    - Zoom and pan
    - (Optional) Snap-to-grid for better alignment

Component Configuration Panel

- Dynamically shows configuration options based on the selected component.
- Should support appropriate inputs (text fields, dropdowns, toggles).
- Optional: Tooltips or help for complex options
- Optional: Import predefined configurations

Execution Controls

- **Build Stack**: Validates and prepares the workflow for execution.
- **Chat with Stack**: Opens the chat modal for query interaction.
- Optional: Real-time logs or progress indicators for workflow execution

Figma Design

Use the following Figma design as reference:
🔗 Figma Design URL

Backend Specification

- Use FastAPI to expose endpoints for:
    - Uploading and processing documents
    - Storing and retrieving embeddings
    - Running the workflow based on connections

    - LLM and SerpAPI interaction
- Orchestrate the component logic in the order defined by the user’s workflow

Database Specification

- Use PostgreSQL to:
    - Store document metadata
    - Store workflow definitions (optional)
    - Store chat logs (optional)

Assignment Deliverables

- ✅ Full source code (frontend + backend)
- ✅ README with setup and run instructions
- ✅ Clear component structure and modular design
- ✅ Video demo or screen recording (optional but preferred)
- ✅ (Optional) Architecture diagram or simple flowchart

Evaluation Criteria

- Functional correctness and adherence to requirements
- UI/UX quality and usability of the workflow builder
- Backend architecture and API design
- Code clarity, organization, and documentation
- Correct use of tools (LLM, embeddings, vector DB, web search)

- Extensibility and modularity

Optional Features

- Workflow saving/loading from the database
- Chat history persistence
- Execution logs
- User authentication

Note: Share source code using Github only.