**Project Report: Document-Aware Chatbot with FastAPI, Ollama, and LangChain**

**Project Overview**

In this project, I implemented a web-based chatbot capable of answering user questions using both general AI knowledge and content from specific company .docx documents. I also added the capability for the chatbot to display tables and images extracted from those documents. This approach enriches the user experience by providing contextually relevant answers that reference both the company’s documentation and external knowledge sources.

**How the Project Works**

1. **Document ingestion and indexing:** I built a pipeline to process all .docx files. The pipeline extracts text, tables, and images from each document. I then chunk the content, generate embeddings, and index everything using FAISS for efficient semantic retrieval.
2. **FastAPI backend:** I developed a FastAPI chat endpoint to handle incoming user requests. The endpoint retrieves relevant chunks of indexed content or falls back to general AI knowledge when needed. I also made sure it can render tables and images in the responses whenever they are relevant to the query.
3. **Frontend:** I built a browser-based chat UI that interacts with the backend and dynamically displays responses. The frontend presents answers along with any images and tables, providing an interactive user interface for chatting.

**How I Accomplished It**

* I set up the project structure with directories for docs, static, and images.
* I built an ingestion script to extract and index the content of .docx documents.
* I developed a FastAPI backend that integrates with LangChain and Ollama to handle language model queries.
* I created a frontend interface to display chat interactions, including images and tables from the documents.
* I iterated on the design and tested different configurations to improve answer accuracy and presentation quality.

**Tools and Libraries Used**

For this project, I utilized a variety of tools and libraries:

* **FastAPI:** Building the backend REST API for handling chat requests.
* **LangChain:** Orchestrating language model workflows and document retrieval logic.
* **Ollama:** Serving the local large language model for generating answers.
* **FAISS:** Powering the semantic vector search index for document retrieval.
* **HuggingFace Transformers:** Providing embedding models and NLP utilities.
* **HTML/JavaScript:** Implementing the responsive web-based frontend UI.
* **Additional Dependencies:** Including python-docx for parsing Word documents and other utility libraries as needed.

**Key Features**

I implemented the following key features:

* **Document-aware Q&A:** The chatbot answers questions using both the content of our documents and fallback to general AI knowledge.
* **Image and Table Display:** It automatically displays images and tables extracted from the documents in the chat responses.
* **Modern, Responsive UI:** The web interface is modern and mobile-responsive, providing an engaging chat experience.
* **Extensible Design:** The system is designed to be easily extensible, allowing new documents and features to be added with minimal effort.

**Summary**

In summary, I built a fully functional, document-aware chatbot that combines modern large language models with structured document retrieval. The chatbot integrates a document ingestion pipeline with advanced language models to enable context-aware question answering. Ultimately, this project provides enriched user experience through a simple, intuitive web interface, demonstrating an end-to-end solution for context-rich Q&A from both general knowledge and company-specific documents.