Production-Worthy PDF RAG Chatbot

Your Objective:

To build a fully functioning production-worthy Retrieval-Augmented Generation (RAG) based AI Chatbot that can answer user queries using One or More PDF documents as knowledge sources.

This assignment is spilt into 5 levels of increasing difficulty starting from basic retrieval to advanced features like metadata filtering and agent-based orchestration. The more levels you can solve, the more brownie points you win during your selection process.

Our Expectations:

You are free to structure the code in any way you like. You are also free to use any resource to get the job done. The purpose of this assignment is to test your conceptual strength. Hence, we expect you to be able to:

- Explain every line of code you write. (Adding comments to your code is encouraged)
- Explain every decision you take. For example, choice of vector database, LLM, etc.
- Follow Pythonic way of coding.
- Make the code as dynamic as possible. Design should be such that we can easily substitute LLM API Keys with our Keys and your chatbot should work.
- Include `requirements.txt` or `pyproject.toml`.
- `README.md` with setup and usage instructions.
- Working demo via Streamlit or API call through Postman for ALL LEVELS.

Going Above and Beyond:

At RMgX, we encourage all our team members to go above and beyond and think beyond the available instructions. Again, win brownie points by thinking and adding more user-friendly valuable features to the chatbot you build by going beyond the 5 levels mentioned here.

Assignment

Level 1 – PDF RAG with Semantic Search

Goal: Build a working chatbot that answers questions based on a single/multiple PDF using semantic search and a language model.

Requirements:

- Parse text from a PDF file.
- Split text into chunks.
- Generate embeddings using a transformer model.
- Store embeddings in a vector database.
- Retrieve top-k relevant chunks for a user query.
- Use a pre-trained LLM to generate an answer based on retrieved context.
- CLI interface / Streamlit app / API call through Postman for demo.

Level 2 – Production-Ready RAG with LangChain or LlamaIndex

Goal: Refactor the pipeline into a modular, maintainable, production-grade system using LangChain or LlamaIndex.

Requirements:

- Use LangChain or LlamaIndex for document loading, text splitting, embedding, retrieval, and prompting.
- Modularize the pipeline into reusable components or chains.
- Provide a clean interface for uploading multiple PDFs.
- Support environment-based configuration and error handling.
- Create a Streamlit or API interface with robust UI/UX.

Level 3 – Conversational Memory Support

Goal: Enable multi-turn conversation capabilities with memory for a more natural chatbot experience.

Requirements:

- Add chat history using a memory mechanism.
- Maintain context across multiple user turns.

- Support both fresh sessions and continued conversations.

Level 4 – Metadata Tagging and Filtering

Goal: Enhance control and traceability with metadata-aware retrieval.

Requirements:

- Add metadata like:
  - Document name
  - Page number
  - Section titles (if extractable)
- Support retrieval filtering based on metadata (e.g., restrict to a specific document).
- Display source metadata alongside each answer.

Level 5 – Agent-based Chatbot with Tool Use

Goal: Build an intelligent chatbot using agent-like behavior for better orchestration and tool selection.

Requirements:

- Use LangChain agents or LlamaIndex agents to:
  - Select tools
  - Route complex queries
  - Chain multiple actions if required
- Allow the agent to dynamically decide between tasks like answering, summarizing, or document switching.
- Include error fallback strategies and logging.