Multi-Agentic Conversational AI System

'Project Requirements & Submission Guidelines'

Objective:

Design and implement a Python-based RESTful API application that enables natural language conversation with a Large Language Model (LLM), integrated with Retrieval Augmented Generation (RAG) and a custom-built CRM system to capture, store, and retrieve user information throughout the chat interaction.

Core Requirements:

1. Conversational Chatbot with RAG

- Accept user messages through API.
- Use OpenAI GPT or similar cloud LLM APIs.
- Dynamically retrieve relevant documents using Retrieval-Augmented Generation from an internal knowledge base.
- Maintain full memory of prior conversation messages.
- Seamlessly blend RAG results with LLM prompts for contextual awareness.

2. CRM User Data Capture

- Build a lightweight CRM system (preferably using MongoDB or SQLite).
- Capture and store user information from conversational exchanges (e.g., name, company, email, preferences).
- Provide API endpoints to view, update, or delete CRM data.

3. CRM-Based Contextual Retrieval

- Save the full conversation history linked to each user (You can generate a Unique ID for each session which helps to identify an unique user).
- Enable LLM access to previous messages via CRM integration to provide richer, personalized responses.
- Expose endpoints to query past conversations and logs.
- CRM should support tagging or categorizing entries based on conversation topics(It could be an indicator for what category the user is chatting about, or it could be as Resolved, Unresolved, Inquiring).
- ***(Optional) Can you integrate calendar related events here as well. How would you go about this?***

---

4. RESTful API Design:

Endpoint	Method	Description
/chat	POST	Accepts user message, returns LLM response with optional RAG-enhanced context.
/upload_docs	POST	Upload docs (PDF/TXT/CSV/JSON) to populate the RAG base.
/crm/create_user	POST	Creates a new user profile with provided details.
/crm/update_user	PUT	Updates user information by user ID.
/crm/conversations/{user_id}	GET	Fetch full conversation history for a user.
/reset	POST	Clears conversation memory (optional: per user).

---

5. Submission Checklist:

- RESTful API app in Python (FastAPI/Flask preferred).
- GitHub repo with us added as collaborators (usernames will be shared).
- Working chatbot endpoint using APIs + RAG.
- Fully functional CRM module that logs user info and chat history.
- Document ingestion and indexing for RAG.
- Endpoints return relevant processing metadata (e.g., response time).
- API returns structured responses in JSON.

Note : You are always welcome to go beyond the requirements. Once the core requirements are met.

---

6. Add to your GitHub repo:

- README.md with clear usage and setup instructions.
- .env file with placeholder values (no real keys).
- CRM schema and architecture diagram using mermaid or similar tools.
- Sample conversation logs (if applicable).

---

7. API Contracts (.pdf file):

- Input formats
- Response schema
- Sample API calls and usage notes

"If you couldn't finish on time, don't be disappointed; submit what you have."
Try to have fun, Be Creative, Feel free to go beyond requirements, surprise us!
Good Luck! You've got this!