AICI Challenge: Conversational Q&A System

Objective

The objective of this challenge is to design and implement a hybrid Retrieval-Augmented Generation(RAG) system that can answer user questions by combining two distinct sources of information:

- Persistent textual knowledge, provided as documents (e.g. PDFs, regulations, technical texts) and stored as embeddings in a vector database.
- Ephemeral, session-based structured data, represented as a list of drawing-related objects provided in JSON format, which may change during the course of a user session.

The system should support interactive question answering where responses are derived from both the fixed document knowledge and the current state of the session-specific object list, which may be modified between queries.

To achieve this, the system should be implemented in a modular architecture, comprising:

- A web frontend for submitting questions and providing the object list
- A backend API responsible for user authentication, session management, and service coordination
- An AI Agent service that implements the hybrid RAG logic and synthesizes answers using both persistent and ephemeral inputs

The primary focus of this challenge is the AI Agent service and its ability to correctly reason over persistent text embeddings and dynamic session data. The frontend and backend should be implemented in a minimal but functional manner to enable end-to-end interaction with the system.

Deliverables

The submission should consist of a modular, functional system comprising the following components:

1. Web Frontend

- Suggested Framework: React or Vue.js
- Functionalities:
    - Interface for submitting natural-language questions

    - Input area (e.g., a textarea) for providing and updating the JSON-based object list
    - Display answers returned from the AI Agent service
- Minimal styling is sufficient; focus is on usability

2. Backend API

- Suggested Framework: FastAPI or Express.js
- Functionalities:
    - User management with authentication (JWT)
    - Session management for ephemeral object lists
    - Communication with the AI Agent service

3. AI Agent Service

- Suggested Frameworks: LangChain, LangGraph
- Functionalities:
    - Hybrid RAG pipeline combining:
        - Persistent textual knowledge from a vector database (e.g., Pinecone, ChromaDB, FAISS)
        - Session-specific JSON object lists
    - Reasoning over both sources to generate accurate answers
    - API endpoint to receive queries from the backend and return responses
- Any suitable LLM may be utilized, including OpenAI API, Anthropic API, or HuggingFace models

4. Containerization & Documentation

- Dockerfile to package the backend and AI Agent service for deployment
- Instructions to build and run the system locally, including Docker commands
- Steps to test queries and update ephemeral object lists
- Overview of the system architecture and design decisions

Implementation Details

The following details provide guidance on how the system should be implemented. These instructions clarify expectations without prescribing exact approaches.

1. Ephemeral Object Handling

- The object list is session-specific and may change between queries.
- Objects are provided in JSON format.

- The system must always use the most current version of the object list when generating answers.
- Ephemeral objects should not be stored in the vector database; they should remain in-memory for the duration of the session.
- Users must be able to add, remove, or update objects between queries.

2. Persistent Text Embeddings

- Documents (e.g., regulations, manuals, PDFs) are pre-embedded in a vector database such as FAISS, ChromaDB, or Pinecone.
- Relevant text chunks should be retrieved for each query using similarity search.
- These embeddings remain fixed and unchanged throughout the session.

3. Hybrid RAG Logic

- Responses must be derived by combining retrieved text embeddings with the ephemeral object list.
- The large language model should receive both sources in a single reasoning step.
- Example operations include counting objects, analyzing properties, or cross-referencing objects against rules in the text embeddings.

4. Backend Service

- The backend must handle user authentication, such as JWT-based login and registration.
- Session management must ensure that each user's ephemeral object list is maintained correctly.
- Connection management between the frontend and the AI Agent service should guarantee that queries are processed with the appropriate session state.
- Connection management between the frontend and the AI Agent service should support real-time communication, ensuring that queries are processed with the appropriate session state
- The backend should provide sufficient functionality for the frontend to:
    - Submit user queries
    - Upload or update ephemeral object lists
    - Authenticate and track users

5. Prompt Construction

- The system should implement system prompts to define agent behavior and provide context for the large language model.
- Query prompts should include:
    - The user's question
    - Relevant text retrieved from the vector database
    - The current ephemeral object list
- Proper construction of prompts is essential to ensure accurate reasoning over both persistent and ephemeral knowledge sources.

Plus/Optional Features

In addition to the core requirements, the system may be enhanced with one or more optional features, which will be evaluated positively but are not mandatory for completion:

1. Advanced PDF Preprocessing

- Implement techniques to extract content from PDF documents with greater precision.
- Text and images may be captured separately, allowing improved embeddings for textual knowledge and potential future extensions involving image data.
- Enhanced extraction contributes to higher-quality embeddings in the vector database, improving the relevance and accuracy of retrieved information.

2. Agentic AI Functionality

- Develop an agentic AI workflow, where the system exhibits higher-level reasoning, planning, or autonomous behavior beyond a standard conversational agent.
- The agent may autonomously plan actions, verify object lists, or guide users based on both persistent text embeddings and session-specific ephemeral data.

3. Verification of Object Lists

- Implement functionality to verify whether objects in the session-specific list comply with information in the text embeddings.
- The system may identify and flag objects that do not conform to rules or constraints derived from the textual knowledge base, enabling proactive detection of inconsistencies or errors

Evaluation

- The AI Agent must provide accurate and consistent answers based on both the persistent text embeddings and the session-specific ephemeral object list, correctly reflecting any updates made between queries.
- The hybrid RAG implementation should effectively integrate persistent textual knowledge with ephemeral session data, ensuring proper retrieval and prompt construction for reasoning.
- The system should demonstrate functional end-to-end behavior, including a frontend for submitting queries and updating object lists, a backend managing users and sessions, and an AI Agent service that synthesizes answers correctly.
- The codebase should be well-structured, readable, and maintainable, with a clear separation of concerns among the frontend, backend, and AI Agent service.
- Error handling and robustness will be considered, including correct handling of invalid object lists or malformed queries.
- Optional / Plus features, such as advanced PDF preprocessing (capturing text and images separately), agentic AI capabilities, or verification of object lists against text embeddings, will be evaluated positively

Submission

- Submissions must include a Git repository containing all necessary files to build and run the system, including Dockerfiles, environment configuration files (if any), and clear documentation describing setup and usage.
- The challenge duration is **7 days** from the date of assignment.
- If necessary, participants may request an extension, subject to approval.