OKADA&CO.

Voice Conversational Agentic AI

'Project Requirements & Submission Guidelines'

Objective:

Design and implement a Python-based RESTful API application that enables bi-directional voice conversations with a Large Language Model (LLM), enhanced with Retrieval Augmented Generation (RAG) using proprietary/internal documents.

Core Requirements:

1. Voice Input & Transcription

- Capture real-time voice input from the microphone.
- Convert speech to text using cloud-based STT APIs such as OpenAI Whisper API, Google Cloud Speech-to-Text or Azure Cognitive Services Speech API.
- Return transcription time in seconds.

2. LLM Text Processing

- Use OpenAI GPT or similar cloud LLM APIs.
- Previous memory should be retained. (LLM should have the context of the convo going on).
- Support parameter injection of RAG context alongside message.

3. RAG Agent Integration

- Accept proprietary/internal documents through a dedicated upload endpoint.
- Index documents using vector stores such as FAISS, Pinecone or Weaviate.
- Retrieve relevant documents dynamically. So the text to speech can use relevant information from the uploaded document.

4. Text-to-Speech (TTS)

- Convert LLM output text into audio using cloud APIs like OpenAI TTS, Google Cloud TTS or Amazon Polly
- Play the generated speech through the speaker.
- Report TTS generation time.

---

5. RESTful API Design:

Endpoint	Method	Description
/transcribe	POST	Accepts audio, returns text + STT time
/chat	POST	Accepts conversation+new-message+context, returns response
/speak	POST	Accepts text, returns audio + TTS time
/converse	POST	End-to-end pipeline (voice LLM + RAG audio)
/reset	POST	Clears the conversation memory
/upload_rag_docs	POST	Uploads RAG knowledge base documents (PDF, TXT, CSV, JSON)

---

6. Submission Checklist:

- RESTful API app in Python (FastAPI/Flask preferred).
- Github repo link adding us as collaborators (usernames will be shared).
- Voice-to-text LLM (with RAG) text-to-speech pipeline.
- Functional upload and indexing of documents via /upload_rag_docs.
- All endpoints return processing durations.

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

---

7. Add to your GitHub repo:

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

---

8. 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!