Voice Agent Task

Position: AI Engineer / Voice Agent Developer

---

1. Background

We are developing a real-time voice-enabled AI agent that interacts naturally with users through voice, powered by LiveKit and Google Gemini Live API. The goal is to assess your ability to design and implement an end-to-end system that integrates a Live API-powered voice agent, a retrieval-augmented generation (RAG) component for knowledge grounding, and a simple web UI for interaction.

---

2. Task Objective

Build a working prototype of a LiveKit Voice Agent that uses Gemini Live API as its core intelligence and integrates a RAG system for context-aware responses. The system should provide a web interface for users to interact with the voice agent in real-time.

Key Objectives:

- Use LiveKit Agents as the framework for the voice agent.
- The voice pipeline must rely entirely on the Gemini Live API, which handles speech input, language understanding, and audio output.
- Integrate RAG so that the agent can retrieve context from a knowledge base before responding.
- Build a simple web page that connects to the LiveKit session, allowing users to speak to and hear the agent.

---

3. Requirements

3.1 Voice Agent (LiveKit + Gemini Live API)

- Use LiveKit Agents to host the voice agent. The agent must connect to a LiveKit Room for real-time audio interaction.
- Use Gemini Live API (free tier available) for live, streaming interaction — handling both voice input and voice output directly. (Docs)
- The Live API replaces the traditional STT → LLM → TTS pipeline with a unified streaming API that accepts and outputs audio in real time.
- The agent should:
    - Join a LiveKit Room.
    - Connect the audio stream to Gemini Live API.
    - Handle real-time voice conversation end-to-end.
    - Use retrieved RAG context (when applicable) to enhance responses.

3.2 RAG (Retrieval-Augmented Generation)

- Prepare a small knowledge base (e.g., FAQs or documentation snippets).
- Implement vector-based retrieval (e.g., FAISS, Qdrant, or in-memory embeddings).
- When the user asks a question, retrieve the top relevant snippets and pass them as context to Gemini Live API.
- Document how the retrieval works and show at least one example of the system using the knowledge base context to answer correctly.

3.3 Web Page Integration

- Build a simple web UI that connects to the LiveKit Room used by the agent.
- The user should be able to:
    - Join the LiveKit session.
    - Speak directly to the agent and hear its voice replies.
    - Optionally view live transcript or retrieved context.
- UI can be basic — focus on functionality. Provide clear setup instructions.

---

4. Recommended Stack

- **Voice Framework:** LiveKit Agents
- **Core AI:** Gemini Live API (Free Plan)
- **Retrieval:** FAISS, Qdrant, or local embeddings
- **Frontend:** React + LiveKit SDK
- **Backend:** Python

---

5. Deliverables

- GitHub Repository containing:
  - Full project code (LiveKit agent, RAG logic, frontend).
  - README with setup instructions (API keys, LiveKit setup, environment variables).
  - Explanation of architecture (how LiveKit connects to Gemini Live API and RAG).
  - Instructions on how to run locally.
- Demo video or script showing example interactions.
- Documentation describing how RAG retrieval integrates with Gemini Live API prompts.

Good Luck!
We look forward to seeing your implementation of a next-generation voice agent powered by LiveKit and Gemini Live API.