## Interview Prep Bot: Voice & Text with RAG

### 1. Introduction

This document provides a comprehensive overview of the Interview Prep Bot, designed to offer realistic practice through **voice and text interactions** powered by Retrieval-Augmented Generation (RAG). It aims to be a valuable tool for users honing their interview skills.

### 2. High-Level Flow

The fundamental process can be summarized as:

**Voice/Text Input** → **Preprocessing** → **RAG (Retriever + Generator)** → **Q&A + Feedback** → **Voice/Text Output**

This architecture ensures a consistent and intelligent user experience regardless of the chosen input method.

### 3. Input Modes

The bot accommodates users through two distinct input channels:

* **📝 Text Mode:** Users directly input their questions or answers via text. This input is seamlessly passed to the RAG pipeline.
* **🎙 Voice Mode:** User's spoken input is captured and then converted into text using advanced Speech-to-Text (STT) technologies.

**Supported STT Technologies:**

* OpenAI Whisper
* Google Speech API
* Vosk

The transcribed text then follows the same RAG processing path as direct text input.

### 4. RAG Pipeline (Core Intelligence)

The RAG pipeline is central to the bot's ability to provide accurate and context-aware responses:

#### Retriever

This component is responsible for sourcing relevant information from a knowledge base.

* **Vector Database:** Stores embedding of interview-related content (technical, HR, company-specific). Options include FAISS, Pinecone, Weaviate.
* **Retrieval Mechanism:** Upon receiving a query, it identifies and retrieves the most semantically similar document chunks.

#### Generator (LLM)

Leveraging a Large Language Model (LLM), this component crafts the final output.

* **Synthesis:** Combines the user's input with the context provided by the Retriever.
* **Output Generation:** Produces contextual answers, behavioral questions, or personalized feedback based on the prompt.

**Core Prompt Template:**

You are an interview coach.  
Context: {retrieved\_docs}  
User Input: {user\_text}  
Task: Answer the question or evaluate the candidate’s response.  
If it’s a candidate answer, provide structured feedback with tips.

This template primes the LLM to act as an expert coach.

### 5. Output Modes

The bot delivers its responses through two primary channels:

* **📝 Text Output:** The LLM-generated text is displayed directly in the user interface.
* **🔊 Voice Output:** The text response is synthesized into speech using Text-to-Speech (TTS) technology and played back to the user.

**Supported TTS Technologies:**

* gTTS (Google Text-to-Speech)
* Amazon Polly
* OpenAI TTS

This flexibility enhances user experience and accessibility.

#### Voice Input Processing Flowchart (Mermaid Syntax)

graph TD  
 A[User Speaks] --> B{Capture Audio};  
 B --> C{Speech-to-Text (STT)};  
 C --> D[Convert to Text];  
 D --> E{RAG Pipeline};  
 E --> F[Generate Response (LLM)];  
 F --> G{Text-to-Speech (TTS)};  
 G --> H[Convert to Speech];  
 H --> I[Play Audio Output];

#### Text Input Processing Flowchart (Mermaid Syntax)

graph TD  
 A[User Types Text] --> B{RAG Pipeline};  
 B --> C[Generate Response (LLM)];  
 C --> D{Output Mode Selection};  
 D -- Text Output --> E[Display Text];  
 D -- Voice Output --> F{Text-to-Speech (TTS)};  
 F --> G[Convert to Speech];  
 G --> H[Play Audio Output];

#### RAG Pipeline Flowchart (Detailed Steps - Mermaid Syntax)

graph TD  
 A[User Query (Text)] --> B{Embed Query};  
 B --> C{Vector DB Search (Retriever)};  
 C --> D[Retrieve Relevant Chunks];  
 D --> E{LLM (Generator)};  
 E -- Uses --> F[Prompt Template + Retrieved Chunks + User Query];  
 F --> G[Generate Final Response];

### 6. Workflow Example

Let's illustrate the bot's operation with a common interview scenario:

1. **User (Voice):** “Tell me about a time you faced a conflict at work.”
2. **🎙 STT Conversion:** The user's audio is converted to the text: "Tell me about a time you faced a conflict at work".
3. **🔍 RAG Retrieval:** The RAG system retrieves relevant information, such as guides on answering behavioral questions or common conflict resolution strategies.
4. **🤖 LLM Generation:** The LLM, using the retrieved context, generates coaching tips and example answers, often suggesting frameworks like the STAR method.
5. **📝 Text Display:** The generated response is shown as text on the user's screen.
6. **🔊 Voice Playback:** The text response is converted to speech and played back to the user: “Here’s how you can structure your answer using the STAR method…”

Subsequently, if the student provides a voice reply with their answer, the RAG pipeline processes it to offer personalized feedback.

### 7. Tech Stack Recommendations

A robust tech stack is crucial for building this Interview Prep Bot. Here are recommended technologies:

**Frontend:** Streamlit, React, or Flutter (for web and mobile application development).

**Backend:** Python, with frameworks like FastAPI or Flask for building the API.

**Voice Processing:**

* STT: Whisper (OpenAI)
* TTS: gTTS, Amazon Polly, or OpenAI TTS

**RAG Framework:** LangChain or LlamaIndex (to streamline RAG pipeline development).

**Vector Database:** FAISS, Pinecone, or Weaviate (for efficient information retrieval).

### 8. Extra Features for Enhanced Practice

To further enrich the interview practice experience, consider implementing the following features:

* **“Mock Interview Mode”:** This mode could include a timer for responses and present a series of questions simulating a real interview.
* **Score Dashboard:** A feature that rates candidate answers based on criteria such as clarity, confidence, and relevance, providing objective feedback.
* **Customizable Question Banks:** Allow users to select specific industries, roles, or difficulty levels for their practice sessions.
* **Performance Analytics:** Track user progress over time, highlighting areas of strength and improvement.