The Right Software Assignment Document

Overview

This assignment consists of **2 main parts**:

1. Building a Retrieval-Augmented Generation (RAG) system with custom similarity and chunking.
2. Implementing a basic voice-to-voice chatbot with open-source components and researching methods to optimize latency.

You **must use GitHub** as an integral part of your workflow, maintaining **feature branches** and clear commit history.

Part 1 – Retrieval-Augmented Generation (RAG) System

Objectives:

- Implement a RAG pipeline that:
    - **Uses a custom similarity function** (e.g., weighted Jaccard, Manhattan distance, or learned similarity metric).
    - **Applies chunking** to long documents to improve retrieval accuracy.
    - Leverages embeddings for vector search.

Expected Deliverables:

- Code that:
    - Ingests textual documents.
    - Chunks them (by sentence, paragraph, or token length).
    - Generates embeddings for each chunk.
    - Implements and applies your custom similarity function.
    - Retrieves top-k most relevant chunks.
    - Feeds retrieved context to an LLM to generate responses.
- Clear documentation and instructions to run the system.

Notes:

- You can use:
    - **FAISS** or **Annoy** for vector indexing.
    - **SentenceTransformers** or **Hugging Face Transformers** for embeddings.

- The implementation must be modular, with separate retrieval and generation components.

Part 2 – Voice-to-Voice Chatbot

This part has **two sub-tasks**:

2a – Basic Voice-Voice Chatbot Prototype

Objectives:

- Build an end-to-end pipeline that:
    1. Converts **speech to text**.
    2. Feeds the transcribed text into a retrieval-augmented or plain LLM.
    3. Converts the generated text back into **speech**.

Requirements:

- Use **open-source models** for:
    - **Speech-to-Text** (e.g., Whisper, Vosk).
    - **Retrieval (RAG)** (you may reuse Part 1).
    - **LLM** (e.g., LLaMA, Mistral, OpenHermes).
    - **Text-to-Speech** (e.g., Coqui TTS, Piper).
- Latency is **not a constraint** for this prototype.

Expected Deliverables:

- Working code demonstrating:
    - Audio input capture.
    - Speech transcription.
    - Text generation.
    - Speech synthesis.
- Clear instructions for setup and usage.

2b – Research on Latency Optimization

Objectives:

- Research and document:
    - Available **open-source Speech-to-Text, Text-to-Speech, and LLM** models.
    - Protocols and techniques to **reduce latency** (e.g., streaming inference, quantization, batching).

        - Cloud or on-premise services that could improve response times.

Expected Deliverables:

- A **written report** (Markdown or PDF) containing:
    - A **comparison of models**, including size, performance, and latency considerations.
    - Protocols evaluation (REST, WebSockets, gRPC) for efficient component communication.
    - Recommendations for production optimization.
    - Citations and references.

GitHub Workflow Requirements

- **Use GitHub as the central repository.**
- **Create a dedicated repository** for the assignment.
- Follow a **feature branch workflow**:
    - Main branch (`main` or `master`) must always contain stable, working code.
    - Each task or feature must be developed in its own **feature branch** (e.g., `feature/rag-retrieval`, `feature/voice-chat`).
    - Use **pull requests** to merge feature branches back into `main`.
    - Include **meaningful commit messages** describing each change.
- Keep your GitHub repository **well organized** with:
    - A clear README.
    - Setup instructions.
    - Links to any datasets or model checkpoints.
    - Example commands to run the code.

Submission Checklist

- ☑ RAG system code with custom similarity and chunking
- ☑ Voice-to-voice chatbot working prototype
- ☑ Research report on latency improvements
- ☑ GitHub repository with feature branches and pull requests
- ☑ Instructions for setup and running deliverables

Useful Resources

- Hugging Face Model Hub
- FAISS Documentation
- Coqui TTS
- OpenAI Whisper
- Gradio
- Git Feature Branch Workflow