Round 2 Project Documentation

Full-Stack Voice AI Pipeline Application

---

1. Project Overview

The objective of this project is to design and implement a **full-stack Voice AI application** that demonstrates your understanding of **speech processing, large language models, and system design**.

You will build a **voice pipeline** that allows a user to speak, processes that speech through multiple AI components, and responds back with synthesized voice output.

This project is intentionally open-ended to evaluate: - Core domain understanding - Architectural thinking - API integration skills - Backend–frontend coordination - Code quality and clarity

---

2. Core Voice Pipeline Requirements

Your system must implement a **complete voice-to-voice pipeline** consisting of the following components:

2.1 Speech-to-Text (STT)

- Capture user audio input
- Convert spoken audio into text
- Can use any STT provider or open-source model

2.2 Language Model (LLM)

- Take transcribed text as input
- Process it using a system prompt
- Generate a meaningful response
- Any LLM provider or open-source model is allowed

2.3 Text-to-Speech (TTS)

- Convert the LLM-generated response into audio
- Stream or return synthesized speech back to the user

2.4 Pipeline Flow

```
User Voice → STT → Text → LLM → Response Text → TTS → Voice Output
```

This pipeline **must be functional end-to-end**.

---

3. Full-Stack Application Requirements

You are required to build a **full-stack system**, not just a standalone pipeline.

3.1 User Authentication

The application must support: - User signup / login - Token-based authentication (JWT or similar) - Secure access to user-specific data

---

3.2 Agent Creation (Digital Agent)

After logging in, a user should be able to: - Create one or more **AI agents** - Define a **system prompt** for each agent - Manage (view, update, delete) created agents

Each agent represents a **custom personality or behavior** used by the LLM.

3.3 Provider Selection

For each agent, the user must be able to select:

- **STT Provider** (e.g., any cloud or open-source solution)
- **LLM Provider**
- **TTS Provider**

The selected providers should dynamically control how the voice pipeline behaves for that agent.

3.4 Voice Interaction

The user should be able to: - Select an agent - Speak into the application - Receive an audio response generated using the selected STT, LLM, and TTS configuration

---

4. Technology Stack (Mandatory)

You are expected to use the following stack:

Backend

- **Python** (FastAPI / Flask preferred)
- REST or WebSocket APIs

- Proper modular architecture

Frontend

- **React**
- Clean UI for login, agent creation, and voice interaction

Database

- **MongoDB** (preferred) or any equivalent NoSQL/SQL database
- Used to store:
- Users
- Agents
- Provider configurations

---

5. Expected Features (Minimum)

- User authentication
- Agent CRUD operations
- Configurable system prompt per agent
- Provider selection UI
- Functional voice pipeline
- Clear separation of frontend and backend

---

6. Evaluation Criteria

You will be evaluated on:

- **Correctness:** Does the pipeline work end-to-end?
- **Architecture:** Clean separation of concerns, scalability
- **Code Quality:** Readability, structure, naming conventions
- **API Design:** Clean and logical endpoints
- **Frontend UX:** Clarity and usability
- **Error Handling:** Graceful handling of failures
- **Documentation:** Clear setup and usage instructions

---

7. Bonus (Optional but Recommended)

- Streaming audio responses
- WebSocket-based voice interaction
- Multiple agents per user
- Environment-based configuration
- Deployment-ready setup (Docker)

8. Submission Guidelines

- Share a **GitHub repository** with:
- Backend code
- Frontend code
- README with setup instructions
- Include sample environment variables
- Clearly mention any assumptions made

---

9. Important Notes

- This project runs for **15–16 days**
- Focus on **clarity and correctness over complexity**
- You are free to choose libraries and providers
- Plagiarism or copy-paste projects will be disqualified

---

10. Final Goal

By the end of this project, you should have a **working Voice AI system** where:

> A user logs in → creates an agent → configures providers → speaks → and hears an AI-generated voice response.

Good luck, and happy building 🚀