Test Project: Design an AI-Powered Voice System

Objective

Design a simple, scalable, and fault-tolerant architecture for an AI-powered voice system. The system should allow for **voice data collection**, perform **speech-to-text** conversion, and **analyze sentiment**. You do **not** need to implement the actual AI models or services; instead, focus on designing how these components would interact in a real-world application.

This project should take **no more than 8 hours** and can be completed without needing access to any third-party services or accounts.

🧱 Project Scope

1. **System Design**:
    - **Voice Data Collection**: Assume that the voice input will be captured through a generic telephony system (you can describe it as “Assume we are using a third-party service like Twilio or similar”). Focus on the architecture and not the integration with these systems.
    - **Speech-to-Text**: Design how the system would convert voice to text. This can be done using a theoretical speech-to-text service. You don’t need to specify which service, just explain how you would integrate it into the system.
    - **Sentiment Analysis**: The system should analyze the transcribed text to determine the sentiment of the conversation (positive, negative, neutral). Again, you do not need to specify or integrate an AI model, just explain where and how this step would occur.
2. **Scalability and Fault Tolerance**:
    - **Scalability**: Describe how you would design the system to handle an increasing number of concurrent voice calls. Consider the use of load balancers, auto-scaling, etc.
    - **Fault Tolerance**: Propose how the system would ensure reliability even when one of its components fails. Describe failover mechanisms and redundancy.
3. **Simple API Design**:

- **POST /voice-input**: This endpoint would accept the voice input and return a response indicating the status of the processing.
- **POST /transcribe**: This would accept the voice data, process it through a theoretical speech-to-text service, and return the transcribed text.
- **POST /analyze-sentiment**: This endpoint would accept the transcribed text and return sentiment analysis results (positive, neutral, negative).

4. **Basic Security Considerations**:
   - Outline **basic security practices** such as data encryption, API authentication, and secure data storage (e.g., how you would ensure voice data is protected when stored or transmitted).

🔑 Deliverables

1. **Architecture Diagram**:
   - A simple diagram showing the high-level architecture of the AI-powered voice system. This should include components for voice collection, transcription, sentiment analysis, and communication between them (using a theoretical service). Tools like Lucidchart, draw.io, or any diagramming tool can be used.
2. **Technical Design Document**:
   - A brief (1-2 pages) document outlining:
     - System components and their roles.
     - Description of API endpoints (POST /voice-input, POST /transcribe, POST /analyze-sentiment).
     - Basic design choices for scalability and fault tolerance.
     - A simple security strategy for handling sensitive voice data.

📊 Evaluation Criteria

The candidate will be evaluated on:

- **Clarity of Design**: How clearly the system components are defined and how they interact.
- **Scalability and Fault Tolerance**: The design's ability to handle a growing number of users and ensure high availability.
- **API Design**: Clear and simple API definitions that facilitate communication between system components.
- **Security Understanding**: Basic knowledge of securing sensitive data, even though no real implementation is required.

🚫 Prohibited

- Use of AI-generated code or assistance. Any evidence of AI involvement will result in immediate disqualification.

📦 Submission Guidelines

- Submit the **architecture diagram** (as a PNG, PDF, or other readable format).
- Submit the **technical design document** in a text-based format (e.g., DOCX, PDF).