Memory Machines

Project Brief: The Live AI-Powered "Sentiment Aura"

The zoom-out:

The past 5 years of Artificial Intelligence have been all about the transformer architecture and LLMs. This is the core tech powering OpenAI, Gemini, and most other AI companies.

At Memory Machines, we have built a new kind of AI that goes past LLMs. A new vertical in AI. We're looking for a candidate who can build gorgeous, functional demos for our models to showcase the magic of our models.

1. The Core Challenge

Build a full-stack web application that performs real-time audio transcription and visualizes the speaker's emotional sentiment and key topics as a live, generative art display. It will be tested on a laptop.

When a user speaks, their words will appear on-screen. In parallel, the text will be streamed to a backend, which will use an AI model (calls to OpenAI/Gemini/Claude API) to extract **sentiment** and **keywords**. This data will then power a "gorgeous" (as per your taste) **Perlin noise visualization** that changes its color, energy, and form in real-time.

2. Technical Architecture: The Three-Part System

This project tests **full-stack orchestration**. The candidate must build and connect three distinct components.

1. **Frontend (React Application)**
    - **Role:** Captures audio, manages the real-time transcription connection, displays the UI, and renders the visualization.
    - **Tech:** React, react-p5 (or p5.js), Web Audio API, WebSocket client, axios.
2. **Backend (FastAPI/Python Proxy)**
    - **Role:** Do not host your own NLP models locally. The backend is to receive text from the frontend, securely call your favorite LLM API, and pass the structured response back.
    - **Tech:** FastAPI (or Node.js/Express) with an endpoint like /process_text.
3. **External 3rd-Party APIs**
    - **API 1 (Transcription):** A real-time transcription service (use Deepgram, your account will have $200 credits when you start. That will be sufficient). It accepts a WebSocket stream of audio.

- **API 2 (AI Model):** An AI service (e.g., OpenAI, Anthropic, Gemini) to detect sentiment and keywords.

3. Key Task: Backend Prompt Engineering

The core *backend* task is NOT prompt engineering. We don't care how good your prompt (and therefore detected keywords or sentiment are). Instead, we care about how good the backend API functions, and how well it fits with the front end. That is, Full Stack Engineering, NOT prompt engineering.

None

4. Detailed Data Flow

1. **Start:** User clicks "Start." React frontend requests mic access.
2. **Stream:** React opens a WebSocket to the **Transcription API** and begins streaming raw audio.
3. **Transcribe:** The Transcription API streams back JSON payloads with text (e.g., {"text": "Hello world", "is_final": true}).
4. **Display:** The React frontend displays this text in the <TranscriptDisplay>.
5. **Proxy Call:** When an "is_final": true transcript is received, the React app makes a POST request to **the backend** (/process_text) with the text.
6. **AI Call: the backend** receives the text, constructs the prompt (see section 3), and calls the **OpenAI API**.
7. **AI Response:** OpenAI returns a JSON (like in section 3) to **the backend**.
8. **Proxy Response: the backend** passes this clean JSON back to the React app.
9. **State Update:** The React app's main component updates its state: setSentiment(0.85) and setKeywords(["..."]).
10. **React (Render):** React passes these new state variables as props to the visualization components.

5. Visual & UI Specifications

This is where the "mind-blowing" part comes in. The visualization must be beautiful, fluid, creative.

A. The "Aura" Visualization (p5.js)

The p5.js canvas should run a Perlin noise field (https://sighack.com/post/getting-creative-with-perlin-noise-fields) in the background. It must react smoothly to the AI data. Think of how you would like to manipulate the color, form, and

shape of the field in response to the sentiment you receive from the LLM API call. In your demo, you can explain how you chose to modify these parameters to represent different aspects of the sentiment.

Perlin noise is powerful, and can be designed to make quite gorgeous displays. Search online for examples of how to play around with it.

B. The UI (React)

The UI should be a clean, modern overlay on top of the Aura.

- **<TranscriptDisplay>:** A semi-transparent panel showing the live, auto-scrolling transcript.
- **<KeywordsDisplay>:**
    - Displays the array of keywords returned from the AI.
    - **This is a key test of good UI:** Keywords should **fade in gracefully** one by one, not just "pop" in. They could float up from the bottom or fade into a tag cloud. This shows attention to detail.
- **<Controls>:** A simple "Start/Stop" button and a visualization of when recording/not recording.

6. What you will be assessed on/Rubric:

- **Full-Stack Orchestration:** Do your frontend, backend, and two different external APIs work well together in real time?
- **Data-Driven Visualization:** Can you map abstract data (sentiment type, score, other attributes) to *multiple* visual parameters (color, motion, text) in a way that is aesthetically pleasing and fluid?
- **Frontend Polish:** How do the colors transition, transcription and keywords shown on screen appear, and so on. We are looking for a “mind-blowing” demo.
- **Async Management & Error Handling:** How does the app behave if the OpenAI API is slow? Or if the WebSocket disconnects?

The past 5 years of Artificial Intelligence have been all about transformer models and LLMs. We’re building a new kind of AI that goes past LLMs. A new vertical in AI. We’re looking for a candidate who can build gorgeous, functional demos for our models to showcase the magic of our models.