Software Engineer Assessment Task

LLM-Based SOAP Note Generator & Evaluator

This assessment will help us understand your technical skills, ability to learn new systems, and comfort level working with modern AI tooling. Your task is to build a Next.js web application that uses models from OpenAI (GPT) and Google Gemini to generate medical SOAP notes from clinical transcripts and evaluate the generated notes against a reference. You are free to be creative in your design and architecture, but the features listed below must be implemented.

Project Overview

You will create a web-based tool where a user can:

1. Upload a clinical transcript (.txt)
2. Upload a reference SOAP note (.txt)
3. Select multiple models (GPT + Gemini)
4. Generate SOAP notes from the transcript using each selected model
5. Automatically evaluate the generated notes against the reference
6. View results in an interactive and visual format

Your final product should be a fully functional web application, either deployed on Vercel or runnable locally.

Functional Requirements

Model Usage (GPT + Gemini)

The user can choose:

- Any **three** OpenAI GPT models of varying capabilities. Examples (you may choose your own), Documentation: https://platform.openai.com/docs/api-reference
- Any **one** of the Google Gemini model Examples, https://ai.google.dev/gemini-api/docs

Each model should receive a prompt containing the uploaded transcript (*create your own prompt based on your best judgement; the below prompt is just a dummy prompt*):

```
Generate a SOAP note based on the following clinical transcript:

<TRANSCRIPT>
```

The model outputs should be displayed clearly in the interface.

Evaluation Metrics

After generating SOAP notes, your system should compare the model output with the reference SOAP note using any of the following evaluation matrices (ROUGE, BLEU, BERTscore). You may include additional metrics if you feel they improve the analysis.

User Interface Requirements (Next.js)

Your web interface should include:

- A clean, intuitive design
- File upload fields for: (Use the available transcript and notes from babylonhealth)
    - Transcript (.txt)
    - Reference SOAP note (.txt)
- Dropdowns to select:
    - Three GPT models
    - One Gemini model
- A "Generate Notes" action
- An "Evaluate" action
- A results section showing:
    - The generated SOAP notes
    - Evaluation metrics for each model
- An interactive graph comparing model performance (You may use Chart.js, Recharts, or Plotly)

Use any styling framework you prefer (Tailwind, Shadcn/UI, Chakra UI, etc.).

Deployment Requirements

Your application must be provided in one of the following ways:

Option A: Vercel Deployment (Preferred)

Deploy your Next.js project to Vercel: https://vercel.com/docs. Include instructions for setting environment variables, such as:

- OPENAI_API_KEY
- GEMINI_API_KEY

Option B: Local Execution

If deployment is not possible, provide:

- Complete source code in a version-controlled repository
- A README explaining setup and how to run locally
- A .env.example file

Deliverables

Please provide:

1. Full source code (GitHub or GitLab)
2. Deployed Vercel link **OR** local run instructions README.md containing:
    - Overview of the project

- Installation steps
- Environment variable setup
- API model choices
- Screenshots of your interface
- Any design considerations or limitations
- Notes about working within free-tier API constraints

Notes About API Usage

You are expected to work within the **free tier** of all platforms, including Vercel and any web hosting for backend deployment (you can explore free credits of GCP, Azure, or AWS).

OpenAI Free Tier (https://platform.openai.com/docs/guides/rate-limits)

- Limited requests per minute
- Some models may require billing

Gemini Free Tier (https://ai.google.dev/pricing)

- Token limits vary per region
- Some models may require enabling billing

Your application should gracefully handle:

- Rate limits
- Invalid API keys
- Temporary failures

6. Optional Enhancements (Completely Optional)

You may choose to add:

- Cost estimation per model
- Token usage summaries
- JSON export of results
- Downloadable comparison reports
- Caching of model outputs
- Improved data visualizations

These are not required, but can showcase your creativity.

BEST OF LUCK!