AI Engineer Technical Test

Background

In a busy clinic, healthcare professionals spend countless hours reviewing clinical progress notes to extract HCC-relevant conditions and ensure proper compliant documentation. This tedious process leaves room for errors, which can result in missed reimbursements or compliance issues.
Your task? Revolutionize this workflow. By leveraging cutting-edge AI, including Python, LLMs, NLP, and LangGraph, you'll create a solution that transforms a cumbersome task into a seamless process, giving clinicians the tools to focus on what truly matters—patient care.

The Task

Create an AI pipeline that performs the following:

1. Extract conditions and their associated codes from provided clinical progress notes.
2. Determine which of the extracted conditions are HCC-relevant, referencing the provided CSV file.

Your pipeline must:

- Leverage **Vertex AI Gemini 1.5 Flash** for LLM capabilities.
- Utilize the **LangGraph** framework to structure and optimize your solution.

Deliverables

1. **A working AI pipeline implemented in Python.**

2. **A Dockerfile** that:
    - Builds an environment containing all necessary dependencies.
    - Automatically processes all given progress notes upon running the container.
    - Generates the output and stores it in a mounted volume, making it easily accessible within the documented directory structure.
3. **A README file** with:
    - Setup instructions.
    - Detailed explanation of your solution.
    - Steps to run the pipeline on test files.
    - Documentation on how to start the LangGraph development web app.
    - Explanation of how to use the provided Dockerfile.
    - Clear instructions to access the processed output inside the mounted volume.
4. **Well-documented, clear, and maintainable code.**
5. **A structured project directory** following best practices.
6. **Unit tests** to ensure the system works as expected.
7. The code must be delivered as a public GitHub repository with clear and structured commits.

Guidelines

- Use **Poetry** for project management.
- Follow a layered architecture, separating condition extraction from HCC relevance evaluation.
- Include robust error handling with appropriate logging or fallbacks.
- Adhere to best practices in Python development, ensuring your code is elegant and simple.
- Provide clear documentation and meaningful comments in your code.
- Ensure the LangGraph development web app can be easily started using `langgraph dev` in the root directory.

- Ensure that Google credentials for accessing Vertex AI Gemini 1.5 Flash are managed securely using a service account JSON file. Include instructions in the README for configuring and using the service account file.

Hints

- **Think scalability:** Design your solution to handle larger datasets in the future.
- **The section of the progress note that lists the conditions is always under "assessment/plan".**
- Use modular functions to separate logic and improve maintainability.
- Optimize your solution for performance, especially when working with LLMs.
- Don't hesitate to ask questions for clarification.

Provided Data

1. **Clinical Progress Notes:** A set of test files containing real-world examples to analyze.
2. **CSV File:** A list of codes to track and evaluate HCC relevance.

Time Estimate

This task typically requires **5–6 hours** to complete, but please be open to take your time and deliver great code.

We're excited to see how you approach this challenge and make HCC documentation smarter and faster. Best of luck! 🚀