Aristotle

Take-Home Engineering Assessment

This assessment is meant to be completed asynchronously by the candidate in less than eight total hours and as few as four. We expect candidates to use frontier LLMs to assist with the assignment (We find Claude Code and OpenAI's Codex to be the strongest coding agents to assist in our workflows – please reach out for temporary credentials if you do not have your own access to these).

After completing this task, you will have implemented a text-based tutor and completed one data-driven experiment gauging its performance; you will be ready to explain your approach, findings, and challenges you encountered.

Task Description

Tutoring sessions with Aristotle often focus on homework problems the student has brought to the tutor. In order to properly guide a student through a homework problem, we have found that Aristotle should have a reference solution available in its context window.

Here are some (but not all) of the challenges involved in that process:

1. The homework problem may come in many formats, including but not limited to: PDF, screenshot, or HTML of a site containing the problem
2. The process of solving the problem may affect future tutor behavior
3. Difficult problems may incur significant latency to solve, straining the student's attention span

and many more.

Task Steps

Your task is a three-step assignment:

1. **Step One** (~1 hour): Design & implement a simple, clean approach to including a reference solution in a model's context window.
    a. *Success Criterion:* After **Step One**, you should have a codebase which enables a (text-only) conversation. In this conversation, the user uploads a problem, a problem solution is generated and entered into the model's context window, and the model then follows a specified system instruction to tutor the user.
    b. A CLI conversation works for this step, although a simple UI (We recommend using Streamlit) will make the next steps much easier.
    c. Use OpenRouter for your LLM gateway. We will provide you with an OpenRouter API key.
    d. This step is highly vibe-codeable. The challenge is to vibecode a solution (conversation interface, system instruction, and problem solution context insertion strategy) that *1)* works and *2)* provides a sound foundation for **Steps Two** and **Three**.
2. **Step Two** (1-2 hours): Experiment with your naive solution. If we were going to deploy this naive approach into production, what would break? When? Why?
    a. *Success Criterion:* After **Step Two**, you should have a collection of evidence that demonstrates your experiments. This evidence should be readable and clear – easy to make test cases out of in the future.
    b. A good format for evidence might be a JSON containing: Input problem, conversation example (back-and-forth user messages and tutor messages), and a quick annotation explaining the importance of this example (an evaluation and/or score of this example).

   c. You should also provide a very clear, concise document outlining the different errors you found. Note that we are not as interested in architecture reliability issues (prompt caching, backend design); we are more interested in model performance issues.

3. **Step Three** (2-3 hours): From your list of problems, pick one problem you find especially interesting. An example: When the user uploads a problem as a screenshot, the model doesn't properly understand the contents of the screenshot. Your task is to implement & test two or three different viable approaches to solving the problem.
   a. At the end of **Step Three**, you should have experimented with a variety of different approaches, and converged on two or three most promising options. We will ask you to explain how you arrived at these possible approaches and the pros and cons of each.
   b. When evaluating different approaches, consider cost and latency alongside performance.

If you're struggling or repeatedly hitting a wall, don't keep pushing. Success on this task is not lines of code written but rather evidence of systems thinking.

Even if you can't figure something out, you'll still have the chance to walk us through your steps and thinking during our debrief – that matters a lot to us. Do your best to document where you got stuck, what you tried, and what you'd explore next given more time.