UpSmith Agentic Challenge

Problem Statement

Your task is to build a chat "agent" that possesses basic intelligence. Building an agentic system represents the primary challenge we're working on at UpSmith, so it is the perfect reflection of a realistic challenge to understand your fit at UpSmith. To fit means you have the abilities to thrive here as well as being excited about the things we're working on.

Criteria

The term "agent" can be somewhat ambiguous. In the context of this specific exercise, we'll define an agent as an abstraction around LLMs that 1) leverages tool calls that are also powered by LLMs and 2) implements some degree of coordination to add "intelligence" on top of what the underlying LLM can already achieve. Beyond this, the criteria follows.

Requirements

Your solution must meet the following requirements:
- **Written in Python**: We work in Python at UpSmith and expect new hires to hit the ground running. Your solution must be in Python. It does not matter what version or framework you choose, if any.
- **Interactive**: We must be able to interact with the agent. It can be over a web UI or simply through the command line with a text-only "UI." We're not evaluating how pretty it is, but it should feel smooth.
- **LLM-powered tools**: Per our definition of an agent, you should implement at least one tool which is itself powered by an LLM. It is very likely you'll have more in order to meet the next requirement.

- **Coordination**: We will not be impressed if you simply leveraged a thinking model and maxed its thinking budget. We want to see that you recognize the limitations of LLMs and apply best practices in building applied AI apps to steer the outcomes.
- **Persistent "memory"**: Your agent should remember things across disparate sessions and use what it remembers to adapt how it responds to your messages. How you persist this data is up to you, whether in a text file or in a DB or something fancier; as long as it works and you have thought about how it might evolve.
- **Specialized workflow**: Beyond general interactions, your agent should have at least one specialized workflow it can execute. Leveraging its memory, it should be personalized. Feel free to use your imagination, but here are some examples to spark some ideas: restaurant recommendations, travel recommendations, calendar management, home improvement assistant.

Guidelines

Here are some guidelines and suggestions (not requirements) to help set you up for success.

- **Use Free Tools**: We don't want to burden you with unnecessary expenses just for interviewing with us. Several providers offer free API tiers like [Gemini](), [Groq Cloud](), and [OpenRouter](). You're welcome to spend money on a premium API, but we will be more impressed with how much you can do with very little (we are frugal).
- **Make it easy**: We'd love to play around with what you've built. Ideally, it's as easy as cloning a repo and typing `make run` or something similarly simple. The harder it is to get it up and running, the harder it will be for us to evaluate your submission. Obviously, this is helpful signal itself; developers who make things easy are highly valued.
- **Use AI**: This may already be assumed, but we'll reiterate that you are welcome to, and honestly expected to, use AI in building your solution. On the job, we all use AI heavily and treat it as an essential part of our toolkits. We don't consider it

cheating at all, and how well you use AI is also something that helps us evaluate your fit. Of course, we'll talk through your project live which will tell us how much you used AI as an aid vs a crutch.

- **Take your time**: There's no time limit, but we suggest budgeting 2-6 hours depending on your familiarity with build agents. If you find yourself spending more than that, feel free to send us a message to check in and we can provide guidance.
- **Have fun**: Most importantly, please have fun!

Resources

Just ask your favorite LLM!

When you're done, send your solution to your hiring manager.

Questions? Just email ktran@upsmith.com!