ML Engineer Test Project: Building an Agentic Worker

Objective

Design and implement a prototype for an agentic worker that can collaborate to complete complex tasks. This project will test your skills in AI/ML engineering, LLM applications, multi-agent systems, and full-stack development.

Project Components

1. Multi-Agent System Architecture

- Design a scalable architecture for a multi-agent system where agents can be dynamically created, assigned tasks, and collaborate.
- Implement a prototype of this system using Python, incorporating at least one LLM (e.g., GPT-3.5 or GPT-4) for agent decision-making.
- Develop a mechanism for agents to communicate and share information.
- Build to support at least one use case (more if possible). You can choose a use case of your choice or select from any of the following domains; marketing agent, sales agent, accounting agent, HR agent, Software Engineering agent.

2. Task Decomposition and Workflow Management

- Create a system that can break down complex tasks of your choice into smaller, manageable sub-tasks.
- Implement a workflow management system that can assign sub-tasks to appropriate agents based on their capabilities.
- Design and implement a simple UI, CLI/TUI or Gui of your choice for users to submit tasks and monitor progress. Ideally you would build a simple React UI, possibly using your agentic worker to create the code.

3. Agent Specialisation and Tool Integration

- Develop at least three specialised agent types with different capabilities (e.g., research agent, coding agent, writing agent). Organise them is a multi-agent squad where work
- Integrate at least two external tools or APIs that agents can use to complete tasks (e.g., web search API, code execution environment).
- Implement a Retrieval-Augmented Generation (RAG) system to provide relevant context to agents during task execution.

4. Performance Monitoring and Optimisation

- Design and implement a system to monitor agent performance and task completion metrics.
- Create a simple dashboard UI, CLI/TUI or Gui to visualise these metrics.
- Propose and implement at least one optimisation strategy based on the collected metrics.

5. Scalability and Cost Management

- Develop a plan for scaling the system to handle a large number of concurrent tasks and agents.
- Implement basic cost management strategies to optimise resource usage (e.g., agent pooling, task prioritisation).
- Provide a brief analysis of potential bottlenecks and proposed solutions.

Deliverables

1. Source code for the prototype system (shared via a GitHub repository).
2. A comprehensive README document explaining the system architecture, components, and how to run the prototype.
3. A brief presentation (5-10 slides) outlining your approach, key design decisions, and potential future improvements.
4. A short video demo (5-10 minutes) showcasing the working prototype.

Time Frame

You will have 7 days to complete this project. Please manage your time effectively and focus on demonstrating your skills across all required areas.

Notes

- Feel free to use open-source libraries and tools, but clearly document any external resources used.
- While a fully production-ready system is not expected, your prototype should demonstrate a clear path towards a scalable, production-quality solution.
- Be prepared to discuss your design decisions, trade-offs, and future improvement ideas during the follow-up interview.

Good luck! We're excited to see your innovative approach to building a multi-agent system for the use case of your choice.