
Task Document: The "Agentic Architect" Sprint

Role: AI Engineer (Agentic Systems)

Time Limit: 5 Days from receipt.

Tech Stack: Python (LangGraph) + React (TypeScript) + MCP (Model Context Protocol).

---

1. The Mission

At Cerina, we don't just build chatbots; we build autonomous systems that act as clinical foundries. Your task is to architect and build the **"Cerina Protocol Foundry"**—an intelligent multi-agent system that autonomously designs, critiques, and refines CBT (Cognitive Behavioral Therapy) exercises.

The Twist: We are not giving you a blueprint. You must act as the **System Architect**. You decide the agent topology. You decide how they collaborate.

Constraint: This scope is deliberately massive for 3 days. You are **expected** to use AI coding assistants (Cursor, Claude, Copilot) to scaffold, generate, and debug code rapidly. We are evaluating your ability to orchestrate complex systems, not your typing speed.

---

2. The Requirements

A. The Backend (The "Brain")

Framework: Python & LangGraph.

Database: You must use a persistent backend (SQLite or Postgres) with LangGraph Checkpointers.

1. The Agent Architecture (Your Choice):

We need a system that mimics a rigorous clinical review board. A simple linear chain (A → B → C) is insufficient. We want to see autonomy and complex reasoning.

- *The Goal:* Produce a safe, empathetic, and structured CBT exercise based on a user intent (e.g., *"Create an exposure hierarchy for agoraphobia"*).
- *The Team:* You decide the roster. However, a robust solution likely needs agents acting as **Draftsmen**, **Safety Guardians** (checking for self-harm/medical advice), **Clinical Critics** (judging tone/empathy), and perhaps a **Supervisor/Manager** to route tasks and decide when a draft is "good enough."
- *The Pattern:* Choose an architecture that best solves this (e.g., **Supervisor-Worker**, **Hierarchical Teams**, or **Network/Swarm**).
- *Autonomy:* The system should be able to loop, self-correct, and debate internally *before* disturbing the human.

2. Deep State Management ("The Blackboard"):

The agents must share a rich, structured state. It shouldn't just be a list of messages. Think of it as a shared project workspace.

- **Context:** Detailed scratchpads where agents can leave notes for each other (e.g., *"Safety Agent flagged line 3; Drafter needs to revise"*).
- **Versions:** Ability to track previous drafts vs. current drafts.
- **Metadata:** Iteration counts, safety scores, empathy metrics.

3. Persistence & Memory:

- **Checkpointing:** Every step of the graph must be check-pointed to the database. If the server crashes, it should resume exactly where it left off.
- **History:** The system must retain a log of all past queries and generated protocols in the database.

B. The Interfaces (The "Body")

1. Interface A: The React Dashboard (Human-in-the-Loop)

- **Visualization:** Build a UI that makes the "Black Box" transparent. We want to see the agents working in real-time (streaming thoughts/actions).
- **The "Halt" Mechanism:** The graph must **interrupt** execution before finalizing.
    - The UI must fetch the current state from the checkpoint.
    - It must present the generated draft to the Human User.
    - The Human can **Edit** the text or **Approve** it.
    - Only upon approval does the graph resume and save the final artifact.

2. Interface B: The MCP Server (Machine-to-Machine)

- Implement the **Model Context Protocol (MCP)** using the mcp-python SDK.
- Expose your complex LangGraph workflow as a single **Tool** (resource) to the MCP ecosystem.
- *Use Case:* A user on an MCP Client (like Claude Desktop) should be able to prompt: *"Ask Cerina Foundry to create a sleep hygiene protocol."* This triggers your backend, runs the agents, and returns the result—bypassing the React UI but using the same underlying logic.

---

3. Recommended Resources

- **Architecture:** *Building Effective Agents* (Anthropic), *LangGraph Multi-Agent Supervisor* tutorials.
- **Protocol:** *modelcontextprotocol.io* (MCP Documentation).

---

4. Submission Guidelines

Deadline: 5 days.

Deliverables:

1. **Code Repo:** Modular, clean, and well-documented.
2. **Architecture Diagram:** A visual representation of your chosen agent topology.
3. **Loom Video (Critical - Max 5 mins):**
    - **The React UI:** Show the agents debating/refining a draft, the "Human-in-the-Loop" interruption, and the final approval.
    - **The MCP Demo:** Connect your server to a client (e.g., Claude Desktop) and trigger the workflow remotely.
    - **The Code:** Briefly explain your state definition and checkpointer logic.

---

5. Evaluation Criteria

1. **Architectural Ambition:** Did you build a trivial chain, or did you design a robust, self-correcting system?
2. **State Hygiene:** How effectively did you use the shared state/scratchpad?
3. **Persistence:** Does the Human-in-the-Loop flow work reliably using database checkpoints?
4. **MCP Integration:** Did you successfully implement the new interoperability standard?
5. **AI Leverage:** Did you use AI coding tools to deliver a "weeks worth of work" in 5 days?

Good luck. Show us what you can build.