Chronoktonos System Implementation: Project Breakdown (v2 - Network-Aware)

Objective: To implement the Chronoktonos multi-agent system as a network-centric service under the chronoktonos.system domain. This involves establishing core API services, developing a network-aware agent framework, and then instantiating the specific, defined agents.

Phase 1: Foundational Network & Domain Setup (The "World")

This phase focuses on creating the foundational internet-facing infrastructure. It replaces the local file-system setup as the primary prerequisite.

- **Epic 1.1: Establish Domain and Network Presence**
  - **Task 1.1.1:** Register the domain chronoktonos.system.
  - **Task 1.1.2:** Configure DNS records.
    - *Details:* Set up subdomains for core services (e.g., registry.chronoktonos.system, api.chronoktonos.system) and potentially for individual agents.
  - **Task 1.1.3:** Provision server/hosting environment for the core API services.

Phase 2: Core Service Implementation (The "Laws of Physics")

This phase focuses on developing the backend APIs that will replace the file-system-based mechanisms.

- **Epic 2.1: Implement Central Registry Service**
  - **Task 2.1.1:** Design the Registry API specification.
    - *Details:* Define RESTful endpoints, e.g., POST /agents (register), GET /agents/{agent_id} (discover), PUT /agents/{agent_id}/status (heartbeat).
  - **Task 2.1.2:** Develop the backend application for the Registry Service.
  - **Task 2.1.3:** Deploy the Registry Service to its designated subdomain (e.g., registry.chronoktonos.system).
- **Epic 2.2: Implement Mailbox Service**
  - **Task 2.2.1:** Design the Mailbox API specification.
    - *Details:* Define endpoints for sending and receiving messages, e.g., POST /mailboxes/{recipient_id} (send), GET /mailboxes/{agent_id} (fetch).
  - **Task 2.2.2:** Develop the backend application for the Mailbox Service.
  - **Task 2.2.3:** Deploy the Mailbox Service to its designated subdomain (e.g., api.chronoktonos.system).

Phase 3: Agent Framework Development (The "DNA")

This phase adapts the generic agent to interact with the new network services instead *of the local filesystem.*

- **Epic 3.1: Develop the Generic Networked Agent Class/Script**
    - **Task 3.1.1: Implement Agent Initialization.** (Largely unchanged, still parses debriefing file).
    - **Task 3.1.2: Implement Registry Service Interaction.**
        - *Details:* Replace file-locking and writing with HTTP calls to the Registry Service API for registration, discovery, and heartbeats.
    - **Task 3.1.3: Implement Core Communication Logic.**
        - *Details:* Rework _send_message to make a POST request to the Mailbox Service. Rework "Active Listening" to periodically GET new messages from its own mailbox endpoint.
    - **Task 3.1.4: Implement Error Handling.** (Largely unchanged, but error messages are sent via API).

Phase 4: Agent Instantiation (The "Population")

This phase involves creating the specific agent instances that will use the new network infrastructure.

- **Epic 4.1: Instantiate ProjectManager_v1**
    - **Task 4.1.1:** Write the specific "Agent Debriefing Protocol" file, updating configuration to point to API endpoints.
    - **Task 4.1.2:** Implement the agent's unique logic, now interacting with a state file that could be local or stored in a cloud database.
- **Epic 4.2: Instantiate SystemArchitect_v4**
    - **Task 4.2.1:** Write the specific "Agent Debriefing Protocol" file with network configuration.
- **Epic 4.3: Instantiate Puppetmaster (Human Interface)**
    - **Task 4.3.1:** Write the debriefing file and register the Puppetmaster agent in the Registry Service.
    - **Task 4.3.2:** Develop a simple web-based client, CLI tool, or Postman collection for the human operator to interact with the system's APIs, replacing the need to manually manage files.