Project Chimera: The Agentic Infrastructure Challenge

Role: Forward Deployed Engineer (FDE) Trainee

Mission: Architect the "Factory" that builds the "Autonomous Influencer."

Context: Spec-Driven Development, MLOps, & Agentic Orchestration.

Prerequisites: Completion of the "MCP Setup Challenge".

Business Objective

Project Chimera represents a pivot to building **Autonomous AI Influencers**—digital entities that research trends, generate content, and manage engagement without human intervention.

Your Role: You are **NOT** here to "vibe code" a quick prototype. You are here to act as the **Lead** Architect and **Governor**.

The Problem: Most AI projects fail because they rely on fragile prompts and messy codebases. When you try to scale them, they hallucinate or break.

The Solution: We need a robust engineering environment where **Intent** (Specs) is the source of truth, and **Infrastructure** (CI/CD, Tests, Docker) ensures reliability.

The Goal: By the end of Day 3, you must have a repository so well-architected, specified, and tooled that a swarm of AI agents could enter the codebase and build the final features with minimal human conflict.

Core Philosophies & Rules of Engagement

1. **Spec-Driven Development (SDD):** * We do not write implementation code until the Specification is ratified.
   - We use the **GitHub Spec Kit** framework.
   - *Why?* Because ambiguity is the enemy of AI. If your spec is vague, the Agent will hallucinate.
2. **Traceability (MCP):**
   - **Requirement:** You must keep the **Tenx MCP Sense** server connected to your IDE at all times. This is your "Black Box" flight recorder.
3. **Agentic "Skills" vs. "Tools":**
   - You must distinguish between **Skills** (reusable functions/scripts your agent calls, like download_video) and **MCP Servers** (external bridges, like a Database connector).
4. **Git Hygiene:**
   - Commit early, commit often (Minimum 2x/day).
   - Your commit history should tell a story of evolving complexity.

The 3-Day Roadmap

Task 1: The Strategist (Research & Foundation)

Focus: Domain Mastery, Environment Security & Future-Proofing.

- **Task** 1.1: Deep Research **& Reading (3 Hours)**
  - **Context:** You cannot build the future if you don't know where the market is going.
  - Reading List**:**
    - The Trillion Dollar AI Code Stack (a16z)
    - OpenClaw & The Agent Social Network
    - MoltBook: Social Media for Bots
  - Read the **Project Chimera SRS** document
  - **Analysis:** In your research notes, answer:
    - How does *Project Chimera* fit into the "Agent Social Network" (OpenClaw)?
    - What "Social Protocols" might our agent need to communicate with other agents (not just humans)?
- **Task 1.2: Domain Architecture Strategy (3 Hours)**
  - **Context:** Before we code, we plan.
  - **Action:** Create a research/architecture_strategy.md document.
  - **Requirements:**
    - **Agent Pattern:** Which pattern fits best? (e.g., Hierarchical Swarm vs. Sequential Chain).
    - **Human-in-the-Loop:** Where does the human approve the content? (Safety Layer).
    - **Database:** SQL vs NoSQL for storing high-velocity video metadata.
  - **Deliverable:** A detailed architectural document (Diagrams are encouraged, use Mermaid.js).
- **Task 1.3: The "Golden" Environment Setup (2 Hours)**
  - **Action:** Initialize your Git Repository.
  - **Requirement:** Connect **Tenx MCP Sense** to your IDE.
  - **Requirement:** Configure a professional Python environment (recommend using uv).
  - **Deliverable:** A confirmed connection log in MCP Sense and a pyproject.toml (or equivalent) in your repo.

Task 2: The Architect (Specification & Context Engineering)

Focus: Translating "Business Hopes" into "Executable Intent" and equipping the Agents.

- **Task** 2.1: The Master Specification **(4 Hours)**
  - **Context:** AI Agents cannot read your mind. They need precise instructions.
  - **Action:** Using the **GitHub Spec Kit** structure (create a specs/ directory), generate the full project blueprint.
  - **Required Specs:**
    - specs/_meta.md: The high-level vision and constraints.
    - specs/functional.md: User stories ("As an Agent, I need to fetch trends...").

        - specs/technical.md:
            - **API Contracts:** Define the JSON inputs/outputs for the agents.
            - **Database Schema:** The ERD for storing video metadata.
        - specs/openclaw_integration.md: **(optional)** detailed plan on how Chimera will publish its "Availability" or "Status" to the OpenClaw network.
- **Task** 2.2: Context Engineering & "The **Brain" (2 Hours)**
    - **Context:** You need to teach your IDE's AI Agent (your co-pilot) how to behave.
    - **Action:** Create a robust rules file (.cursor/rules or CLAUDE.md).
    - **Deliverable:** The rules file must explicitly contain:
        - **Project Context:** "This is Project Chimera, an autonomous influencer system."
        - **The Prime Directive:** "NEVER generate code without checking specs/ first."
        - **Traceability:** "Explain your plan before writing code."
- **Task** 2.3: Tooling & Skills **Strategy (2 Hours)**
    - **Context:** Agents are powerless without tools. You must define two categories of tools:
    - **Sub-Task A: Developer Tools (MCP):**
        - Select and configure MCP servers that help *YOU* develop (e.g., git-mcp for version control, filesystem-mcp for file editing). Document this in research/tooling_strategy.md.
    - **Sub-Task B: Agent Skills (Runtime):**
        - Define the **Skills** the *Chimera Agent* will use.
        - *Definition:* A "Skill" is a specific capability package (e.g., skill_download_youtube, skill_transcribe_audio).
        - **Action:** Create a skills/ directory and draft the README.md for at least 3 critical skills, defining their Input/Output contracts. *You do not need to implement the full logic yet, but the structure must be ready.*

Task 3: The Governor (Infrastructure & Governance)

Focus: Building the "Safety Net" for the AI Swarm.

- **Task 3.1: Test-Driven Development (TDD) (3 Hours)**
    - **Context:** How do we know the AI Agent built the right thing? We run tests.
    - **Action:** Based on your technical.md spec, write **Failing Tests**.
    - **Deliverable:** A tests/ folder containing:
        - test_trend_fetcher.py: Asserts that the trend data structure matches the API contract.
        - test_skills_interface.py: Asserts that your skills/ modules accept the correct parameters.
    - *Note:* These tests SHOULD fail when you run them. That is success. It defines the "Empty Slot" the AI must fill.
- **Task 3.2: Containerization & Automation (3 Hours)**
    - **Context:** "It works on my machine" is not acceptable in professional CloudOps.
    - **Action:** Create a Dockerfile that encapsulates your environment.
    - **Action:** Create a Makefile to standardise commands.
    - **Deliverable:**
        - make setup: Installs dependencies.
        - make test: Runs the (failing) tests in Docker.

        - make spec-check: A script that verifies if code aligns with specs (optional but recommended).
- **Task 3.3: CI/CD & AI Governance (2 Hours)**
    - **Context:** When an agent pushes code, who reviews it?
    - **Action:** Setup a GitHub Action (.github/workflows/main.yml) that runs your make test command on every push.
    - **Action:** Configure an **AI** Review **Policy**.
        - Setup (or simulate via config) a tool like **CodeRabbit**.
        - Create a .coderabbit.yaml (or equivalent) that instructs the reviewer to check for **Spec Alignment** and **Security Vulnerabilities**.

4. Submission Checklist

By the end of **today (February 4)** you should submit:-

1. A Google Drive link to a report (PDF, Google Doc, or Markdown) that is **accessible to anyone** with the link and includes:
    a. **Research Summary:** What key insights did you take from the reading materials (a16z article, OpenClaw, MoltBook, SRS)?
    b. **Architectural Approach:** What agent pattern and infrastructure decisions are you leaning toward, and why?

By the end of **Friday (February 6)**, you are **not** submitting a generated video. You are submitting the **Repository** that is ready to build it.

1. **Public GitHub Repository:**
    - Must contain specs/, tests/, skills/ (structure), Dockerfile, Makefile, .github/workflows/, and .cursor/rules.
2. **Loom** Video (Max 5 Mins):
    - Walk us through your **Spec Structure** and **OpenClaw Integration Plan**.
    - Show the **Failing Tests** running (proving the TDD approach).
    - Demonstrate your **IDE Agent's Context** (Ask it a question about the project and show that it answers using your rules).
3. **MCP Telemetry:**
    - Ensure **Tenx MCP Sense** was active. We will verify your "Thinking". Make sure you connect to MCP Sense with the same GitHub account you submit your project with.

5. Assessment Rubric: Velocity vs. Distance

We measure your performance on a matrix of **Velocity** (Speed) and **Distance** (Engineering Depth).

Dimension	**The FDE Trainee** **(1-3 Points)**	**The Orchestrator** **(4-5 Points)**

Spec Fidelity	Good text descriptions (.md files).	**Executable Specs:** API schemas, Database ERDs, and OpenClaw protocols are defined and linked.
Tooling & Skills	Basic MCP setup.	**Strategic Tooling:** Clear separation of Dev MCPs vs. Runtime Skills; interfaces are well-defined.
Testing Strategy	Basic unit tests.	**True TDD:** Failing tests exist *before* implementation, defining the agent's goal posts.
CI/CD	Basic build pipeline.	**Governance Pipeline:** Linting, Security Checks, and Testing run automatically in Docker.