Architecture Strategy for Project Chimera

1. Introduction

Project Chimera is a platform for Autonomous AI Influencers where digital entities research trends, generate short-form video content, engage audiences, and manage finances with minimal human intervention. This document captures the architectural decisions, rationale, and trade-offs that govern all downstream specifications and implementation. The system is built *on Java 21+ with Virtual Threads, Spring Boot 3.x, and Maven, with all inter-agent communication modelled as immutable Java Records*.

2. Research Summary

The Project Chimera SRS establishes core functional requirements (trend research, content generation, engagement, financial governance) and mandates a human-in-the-loop safety layer. Its requirement for simultaneous multi-platform operation directly motivates our parallel swarm architecture. *(more on research document in research folder)*

3. Agent Coordination Pattern

Decision: Hierarchical Swarm with three roles — Planner, Worker, and Judge.

The Planner decomposes campaign goals into discrete tasks. Workers execute tasks concurrently on Java 21 Virtual Threads via Executors.newVirtualThreadPerTaskExecutor(). The Judge validates every output against persona rules, safety filters, and campaign specs before approving publication.

This pattern delivers parallelism (thousands of concurrent lightweight tasks), resilience (failed Workers are isolated and retried), quality control (single auditable validation point), and horizontal scalability (add more Workers without changing Planner or Judge).

I rejected the ***Sequential Chain alternative***, because it is fundamentally blocking a single slow step stalls the entire pipeline, and there is no mechanism for parallel fan-out or rework.

State consistency is managed through *Optimistic Concurrency Control (OCC).* Every task carries a version field. When a Worker submits results, the Judge compares versions; stale submissions are rejected and requeued. This eliminates locks and pairs naturally with immutable Records.

4. Human-in-the-Loop Placement

Decision: Post-generation, pre-publication with confidence-based escalation.

The Judge assigns a confidence level to every artifact. I will define three to four tiers ranging from high confidence (auto-approved for publication) down to low confidence (rejected and returned to the Planner for rework), with a middle tier queued for asynchronous human review while the agent continues other work.

A sensitive-topic filter covering areas such as politics, health, finance, and legal content overrides the confidence level entirely and mandates human review regardless. Exact thresholds for each tier are configurable per campaign, allowing operators to tighten or relax approval gates based on brand risk tolerance.

I rejected ***pre-generation review*** (throttles the pipeline) and post-publication monitoring (risks public exposure of harmful content).

5. Data Persistence Strategy

Decision: PostgreSQL as the single database, using JSONB columns for semi-structured data.

PostgreSQL provides relational integrity for structured data such as campaigns, agent profiles, persona configurations, and review decisions — all of which have clear relationships and benefit from constraints and structured queries. Its native JSONB type accommodates the variable-schema nature of trend data, video metadata, platform-specific fields, and engagement metrics, which differ in shape across platforms and evolve over time. These JSONB columns are queryable via PostgreSQL's JSON operators and indexable with GIN indexes for fast lookups.

This avoids the operational overhead of running two separate database systems while satisfying both requirements: relational structure for core platform data, and schema flexibility for content and trend metadata that varies across platforms.

For agent memory, short-term conversational context is modelled as an in-memory cache (Caffeine, with a clear interface for future Redis substitution) and long-term persona memory is represented as a Java interface with a stub implementation ready for a future vector database (Weaviate). All memory interfaces are defined so swapping in dedicated stores requires no contract changes.

I rejected ***a NoSQL-only*** approach because campaigns, agent configurations, and review workflows require relational queries and constraints that document stores handle poorly. We rejected a polyglot approach (separate SQL and NoSQL) because the operational complexity of two database systems is not justified at this stage that’s why PostgreSQL with JSONB delivers both capabilities in a single, well-understood system.

6. Ecosystem Integration

OpenClaw: Each agent holds a crypto wallet (Coinbase AgentKit) as its network identity. Four MCP server interfaces i.e. discovery, negotiation, reputation, and payment.

MoltBook: An immutable SOUL.md defines each agent's persona. Hierarchical memory (in-memory cache for short-term, vector store interface for long-term) maintains coherence across interactions. The honesty directive overrides persona when the agent is asked about its AI nature. Platform AI labels are included in all posts.

7. Technology Stack

Language: Java 21+ (Virtual Threads).

Framework: Spring Boot 3.3.x.

Build: Maven 3.9.x. Testing: JUnit 5 with Mockito.

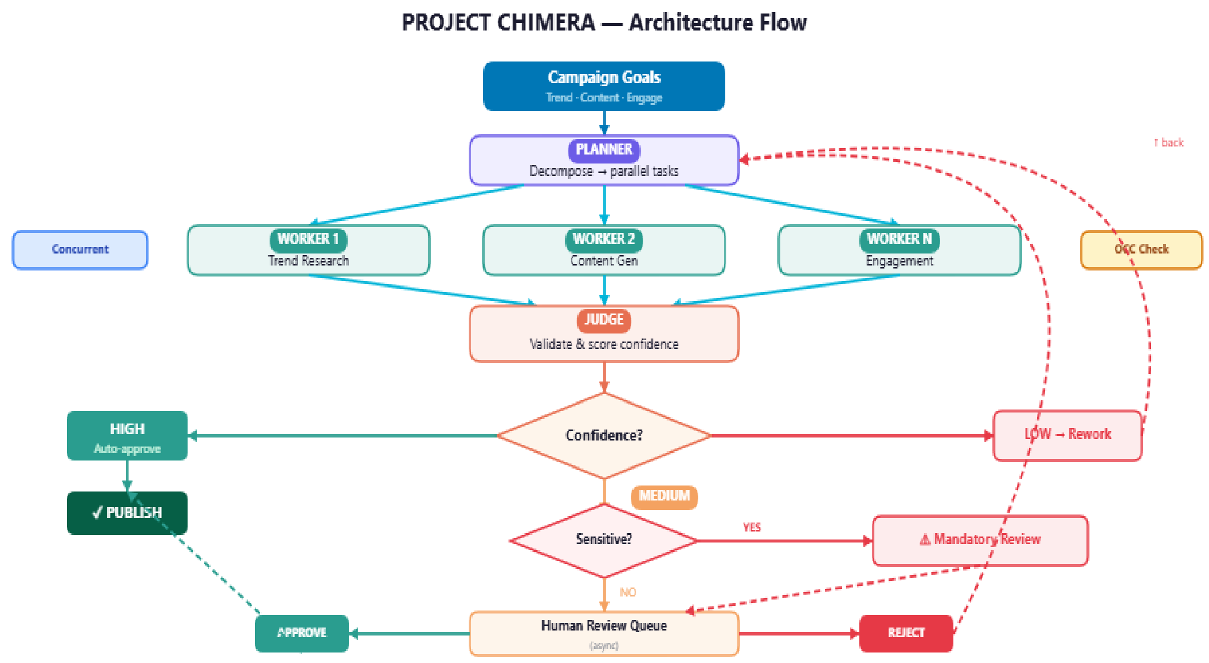
Database: PostgreSQL 16 (with JSONB for semi-structured data).

LLM Gateway: MCP Java SDK. CI/CD: GitHub Actions.

8. Key Architectural Decisions

ADR-001 — Hierarchical Swarm over Sequential Chain: parallelism, resilience, and built-in quality control.
ADR-002 — HITL at pre-publication gate: catches issues at the most critical point without throttling automation.
ADR-003 — PostgreSQL with JSONB as single database: relational integrity for business data plus schema flexibility for content metadata in one system.
ADR-004 — Java Records for all DTOs: immutability ensures thread-safe OCC.
ADR-005 — Virtual Threads for Worker concurrency: thousands of tasks without thread pool exhaustion.
ADR-006 — MCP abstraction for all external services: decouples agent logic from volatile APIs and enables clean stub testing.

Fig 1: Flowchart


PROJECT CHIMERA — Architecture Flow
Campaign Goals
Trend · Content · Engage
PLANNER
Decompose → parallel tasks
↑ back
Concurrent
WORKER 1
Trend Research
WORKER 2
Content Gen
WORKER N
Engagement
OCC Check
JUDGE
Validate & score confidence
HIGH
Auto-approve
Confidence?
LOW → Rework
✓ PUBLISH
MEDIUM
Sensitive?
YES
⚠ Mandatory Review
NO
APPROVE
Human Review Queue
(async)
REJECT