Project Chimera — SRS Report (Canonical)

Executive Summary

Project Chimera (2026 Edition) defines an autonomous, swarm-based Influencer Network that combines the Model Context Protocol (MCP), a FastRender Swarm, and Agentic Commerce. This report summarizes the SRS, analyzes the Trillion Dollar AI Code Stack, and explains how OpenClaw protocols influenced our `openclaw.json` integration strategy.

1. Analysis: The "Trillion Dollar AI Code Stack"

- Core idea: The "Trillion Dollar AI Code Stack" positions AI-native infrastructure (model access, data mesh, tooling, governance) as the backbone for next-gen software. Key layers:
- Model Abstraction Layer: standardized model access (MCP analog).
- Context & Memory Layer: vector DBs, RAG patterns.
- Tooling Layer: MCP Servers wrapping external APIs.
- Governance Layer: policy enforcement, traceability, audit logs.

- Relevance to Chimera: Project Chimera maps directly onto this stack:
- MCP provides the Model Abstraction + Tooling layers.
- Weaviate + Redis + Postgres supply the Context & Memory and Transactional layers.
- CI/spec-driven development implements the Governance layer (specs/, schema checks, .cursor/rules).

- Strategic implications: Prioritize interface contracts (JSON Schemas, MCP Tool specs) over internal implementations. This reduces drift when swapping model or tool providers and enables safe automation at scale.

2. OpenClaw Influence & `openclaw.json` integration

- OpenClaw (Agent Social Network) introduces social protocols for agent interoperability: presence, capability discovery, and capability negotiation.
- How it influenced `openclaw.json` (design decisions implemented):
- Capability Advertisement: agents publish a small, stable descriptor listing `channels`, `public_tools`, and `rate_limits` so other agents can discover and interoperate.
- Presence & Availability: `status` fields and heartbeat metadata to avoid ghost participants.
- Security: signed capability manifests to prevent impersonation; minimal public surface to avoid leaking secrets.

- Practical integration: `openclaw.json` acts as a lightweight MCP Server descriptor: it is parsed by the Orchestrator to register agent endpoints and map MCP Tool calls to OpenClaw channels (e.g., `social.post` → `openclaw://agent/<id>/post`).

3. Key Recommendations

- Enforce Spec-Driven Development: Keep `specs/` authoritative; fail CI on spec drift.
- Interface-first engineering: Implement JSON Schema for Task and Tool payloads early.
- Governance: Add `AGENTS.md` for operational roles and `.cursor/rules` for agent behavior.
- Agentic Commerce caution: Prototype with test wallets and strict budget controls (CFO Judge pattern) before enabling live transfers.

4. Deliverables in this repo (current)

- `specs/` — functional + technical specs (partial).
- `skills/` — skill scaffolds and adapters (db adapter, etc.).
- `tests/` — TDD scaffolding and integration tests (Postgres integration implemented).
- `docs/Project_Chimera_SRS_Report.md` — this canonical report (Markdown).

5. Submission & PDF Export Instructions

To produce a final PDF suitable for upload to Google Drive, run one of the following locally (recommended: `pandoc`):

- Using pandoc:
```bash
pandoc docs/Project_Chimera_SRS_Report.md -o docs/Project_Chimera_SRS_Report.pdf
--pdf-engine=xelatex
```

- Using Python (headless, no LaTeX):
```bash
python -m pip install markdown weasyprint
python - <<'PY'
import markdown, weasyprint
html = markdown.markdown(open('docs/Project_Chimera_SRS_Report.md').read())
weasyprint.HTML(string=html).write_pdf('docs/Project_Chimera_SRS_Report.pdf')
PY
```

Upload the resulting `docs/Project_Chimera_SRS_Report.pdf` to Google Drive and set Sharing to "Anyone with the link can view." Submit the Drive "file link" (not the folder link).

6. Notes on Submission Statement (what to include in the Drive doc metadata)

- Title: "Project Chimera — SRS Report (FDE Submission)"
- Description: include the repo URL, CI status badge, and a short bullet list of included artifacts (specs/, tests/, Dockerfile, Makefile, .cursor/rules).
- Accessibility: Confirm sharing set to "Anyone with the link can view" and include the link in your submission form.

If you want, I can convert this Markdown to PDF inside the container and create `docs/Project_Chimera_SRS_Report.pdf` for you — shall I proceed to generate the PDF in-repo now? (This will run the `pandoc` or `weasyprint` command inside the repo container.)

7. Privacy, Compliance & Ethics

This project must meet contemporary legal and ethical requirements for AI systems and data processing. Below are the minimum requirements and recommended controls that the implementation and deployment teams must follow.

- Regulatory Baseline: Ensure compliance with applicable laws (e.g., GDPR for EU data subjects, the EU AI Act for high-risk AI systems, and relevant US state laws). Contractually require downstream tenants to adhere to the same privacy standards.
- Data Minimization: Only collect and persist the minimum personal data required to accomplish a task. Use ephemeral short-term caches (Redis) for session/episodic data and store long-term data in Weaviate only after explicit sanitization and consent checks.
- PII Handling: Automatically detect and redact PII before forwarding to any external MCP Server or third-party tool. Implement an enforceable `redact_pii()` utility in the ingestion pipeline.
- Consent & Disclosure: Maintain explicit consent records for data subjects where required. Agents must automatically disclose their AI nature on direct inquiry (Honesty Directive) and include provenance metadata on published content where platforms support it.
- Secrets & Key Management: Use an enterprise secrets manager (AWS Secrets Manager, HashiCorp Vault) for all private keys and API credentials. Never store secrets in the codebase or log them. Inject secrets into agent runtimes at startup only.
- Financial Compliance (Agentic Commerce): For on-chain or fiat transfers, implement KYC/AML controls in the CommerceManager flow where required by jurisdiction and by the exchange/AgentKit

provider terms. Use test wallets in development and require human approval (CFO Judge) for high-value transactions.
- Auditability & Traceability: Log all tool calls, prompts, and critical decision metadata to an append-only audit store. Record `who/what/when` for every external action so human reviewers and auditors can reconstruct agent decisions.
- Risk-based HITL: Configure HITL thresholds and sensitive topic filters (politics, health, legal, finance) to require human approval regardless of auto-approval confidence thresholds.
- Retention & Deletion: Define clear retention policies for short-term caches, semantic memory snapshots, and transactional logs. Provide a mechanism to remove a subject's data from long-term memory in accordance with legal deletion requests.

Developer Checklist (minimum):

- Embed `AGENTS.md` policies into CI checks to ensure new persona updates pass an automated policy scanner.
- Add automated PII/redaction tests to `tests/` that fail if PII is accidentally persisted to long-term storage during CI runs.
- Add unit tests for `CommerceManager` that mock AgentKit responses and verify budget checks and atomic Redis updates.

References

- EU General Data Protection Regulation (GDPR): https://eur-lex.europa.eu/eli/reg/2016/679/oj
- EU AI Act (proposal overview): https://commission.europa.eu/AI-act
- Model Context Protocol: https://modelcontextprotocol.io/docs/learn/architecture
- Weaviate Vector Database: https://weaviate.io/docs
- Coinbase AgentKit (Agentic Commerce): https://github.com/coinbase/agentkit
- OpenClaw (agent social protocols): https://openclaw.example.org (design doc placeholder)
- Gemini 3 & Claude Opus (vendor docs): see vendor APIs for model selection

---