AgentOps Studio – Technical Design & Build Guide

Version: 0.1\ **Date:** 17 Jun 2025\ **Authors:**

- **Shubhankar Tripathy** – Tech Lead / Full-Stack & LLM Integration
- **Chris** – Backend & Orchestration Lead
- **Neha** – Product, UI/UX & Design System Lead

Table of Contents

---

1 Vision & Value Proposition

AgentOps Studio is the first drag-and-drop DevOps platform for LLM agents. It lets developers:

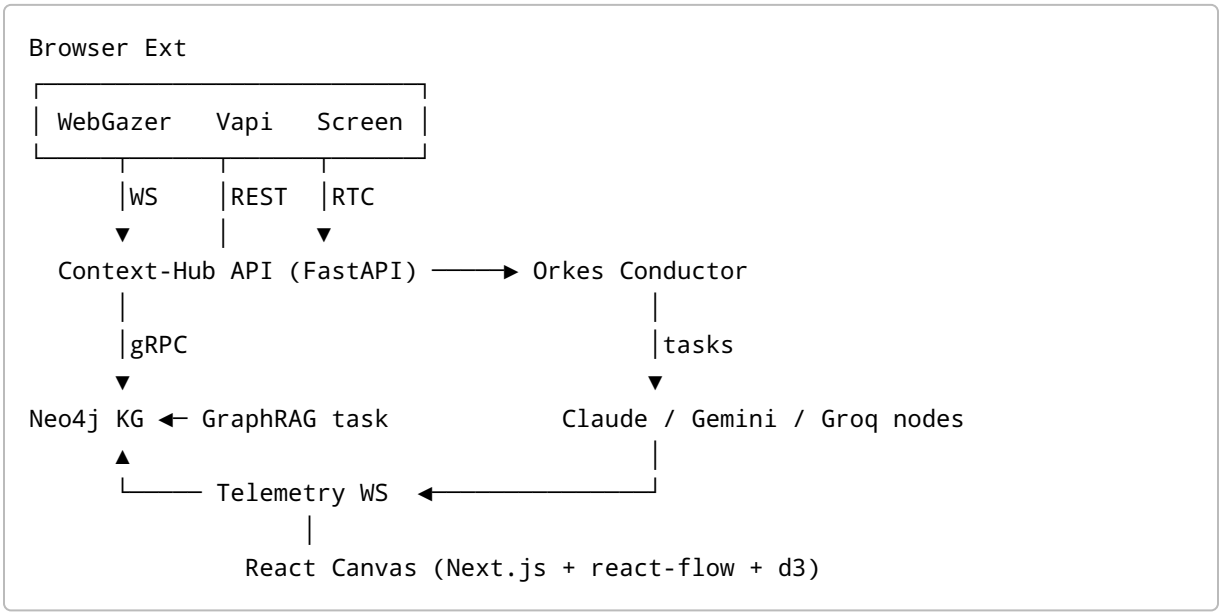
- Design inference pipelines visually like GitHub Actions for prompts.
- Observe cost & latency in real time.
- Ground outputs with a built-in Knowledge Graph (GraphRAG).
- Debug hands-free via a multimodal, context-aware Copilot (eye tracking + voice).

Target user: AI engineers & ML platform teams shipping GenAI features on multi-provider stacks.

Hackathon objective: Deliver a polished MVP that demonstrates technical depth *and* rapid user value to secure the Grand Prize for Dev Tools.

---

2 System Architecture Overview



High-level layers

1. **Front-end Canvas** – drag-and-drop editor, live metrics panel, KG visualiser.
2. **Context Hub** – auth, session mgmt, routes multimodal signals into Conductor.
3. **Orchestration** – Orkes Conductor executing each pipeline node as a task.
4. **LLM & KG services** – Claude, Gemini Vision, Groq LLM, Neo4j vector-graph.
5. **Observability** – Prometheus & Grafana sidecar + custom WS stream to UI.

---

3 Technology Stack

Layer	Tool / Library	Why chosen
FE	**Next.js 14** + React-Flow	File-system routing, SSR, drag graph DSL
FE UI	**Shadcn/UI**, TailwindCSS	Rapid consistent styling
Eye Tracking	**webgazer.js**	Lightweight, license-friendly
Voice I/O	**Vapi**	Sponsor API, duplex streaming
Screen Cap	**getDisplayMedia**	Native browser API

Layer	Tool / Library	Why chosen
BE API	**FastAPI**	Async Python, easy Pydantic models
Workflow	**Orkes Conductor**	Cloud-native workflow engine
Agents	**Fetch.ai uAgents**	Lightweight Python micro-services
LLM endpoints	**Claude-3** / **Gemini 1.5-Vision** / **Groq Llama-3**	Multimodal & low-latency
KG	**Neo4j Aura** + LangChain `Neo4jGraphRAGRetriever`	Unified Cypher + vector search
DB cache	Redis 7	Rate-limit & session store
DevOps	Docker-Compose, GitHub Actions, Vercel, Fly.io	Multi-env deploy
Observability	Prometheus, Grafana, Loki	Token & cost dashboards

4 Development Environment Setup

1. **Clone monorepo**

   ```
   git clone https://github.com/your-org/agentops-studio.git && cd
   agentops-studio
   ```

2. **Install pnpm workspaces** (front-end): `npm i -g pnpm && pnpm install`
3. **Python env** (backend & tasks):

   ```
   pyenv install 3.11.8; pyenv virtualenv 3.11.8 aostudio; pyenv local
   aostudio
   pip install -r backend/requirements.txt
   ```

4. **Docker Compose up** (Conductor + Neo4j + Redis)

   ```
   docker compose -f infra/local.yml up -d
   ```

5. **Env variables:** copy `.env.example` ➜ `.env` and fill:
6. `CLAUDE_API_KEY`
7. `GROQ_API_KEY`
8. `GEMINI_API_KEY`
9. `NEO4J_URI`, `NEO4J_USER`, `NEO4J_PASSWORD` …
10. **Run Dev servers**

```
pnpm --filter=web dev         # port 3000
uvicorn backend.main:app --reload  # port 8000
```

11. Visit **http://localhost:3000** – should load blank canvas + sidebar.

---

5 Core Modules

5.1 Front-end Canvas

- Built with **react-flow**; custom node types: `LLMNode`, `GraphRAGNode`, `CopilotNode`, `OutputNode`.
- State managed in Zustand; live run status via WebSocket events.
- Neha owns design system tokens & dark-mode theme.

5.2 Context Hub API (backend/)

- Auth via JWT (Clerk.dev for hackathon speed).
- `/events/eye`, `/events/voice`, `/events/screen` endpoints buffer → Redis.
- `/pipeline/run` → POST JSON spec → Conductor REST `/workflow`.

5.3 Orkes Conductor Workflows (workflows/definitions)

- Task types: `claude_call`, `gemini_vision`, `groq_call`, `graph_retrieve`, `focus_detect`, `voice_transcribe`, `apply_patch`.
- Temporal ID = `<user>-<timestamp>` for cost aggregation.

5.4 LLM Node Tasks

- **Claude**: text rewrite + chain-of-thought.
- **Gemini Vision**: screenshot caption & context embedding.
- **Groq**: final synthesis; params: `model=llama3-70b-instruct`, `stream=true`.

5.5 Knowledge Graph Layer

- Neo4j **schema**: `(:File {path})-[:CONTAINS]->(:Function {name})`, `CALLS`, `IMPORTS`, plus vector index on Function `embedding` property.
- Ingest script (`scripts/ingest_repo.py`) uses tree-sitter to parse AST and OpenAI function for embedding.
- Retriever: LangChain `Neo4jGraphRAGRetriever` hybrid search (Cypher + vector).

5.6 Context Copilot

- **webgazer.js** emits `<x,y>` focus 10 Hz → throttled.
- Screen frames JPEG at 1 fps → Gemini Vision.
- Voice transcribed by Vapi → Claude-rewrite.

5.7 Cost & Latency Telemetry

- Token counts parsed from LLM responses; price table cached in Redis.
- `@after_task` hook posts `{node, ms, $}` to `/ws/metrics`; UI shows spark-lines.

---

6 End-to-End Data Flow Walk-Through

1. **User drags nodes ➜** JSON workflow spec saved.
2. Hits **Run ➜** Context Hub POST to Conductor.
3. **Conductor** executes tasks; each emits WebSocket events.
4. LLM calls stream back tokens; cost tallied.
5. GraphRAG task queries Neo4j; returns subgraph JSON.
6. Cost, latency & graph visual update in UI; Groq streaming answer shown & optionally voiced back through Vapi.

---

7 Building From Scratch – Step-by-Step

Phase 0 Prereqs (½ day)

- Install Docker, Node 18+, Python 3.11, Redis, Make.

Phase 1 Hello-World Orchestrator (2 h)

1. `docker run -p 8080:8080 orkes/conductor-standalone`
2. Create `echo_workflow.json`; POST to `/api/metadata/workflow`.
3. Trigger via `/api/workflow` and verify UI.

Phase 2 Graph Canvas MVP (3 h)

- Scaffold Next.js app; add react-flow.
- Implement drag node → `onConnect` update JSON spec.
- Button `Run` POST to backend.

Phase 3 LLM Nodes (4 h)

- Create Python `@task` wrappers for Claude & Groq.
- Add metrics hook; stream tokens.

Phase 4 Knowledge Graph (4 h)

- Spin Neo4j Aura; run `scripts/ingest_repo.py --repo path/to/sample`.
- Implement `graph_retrieve` task.

Phase 5 Context Copilot Multimodal (6 h)

- Integrate webgazer & fallback cursor.
- Build Vapi voice widget; transcribe route.
- ScreenCapture jpeg stream.

Phase 6 Observability (2 h)

- Add Prometheus sidecar; `/metrics` exposed.
- Grafana dashboard import `grafana/dashboards/llm_cost.json`.

Phase 7 Packaging & Deploy (2 h)

- Docker Compose prod file (`infra/prod.yml`) packaging FastAPI + Conductor + NATS.
- Front-end on Vercel; backend on Fly.io.
- One-click `Deploy to Vercel` button in README.

---

8 Security & Compliance

Concern	Mitigation
API secrets in FE	All keys stored server-side; FE uses short-lived signed URLs
PII in screenshots	Sample repo only; blur faces via OpenCV prefilter
Rate limits / DoS	Redis token bucket, 100 req/min per IP
RBAC	Neo4j `reader` role for runtime, `writer` only for ingest uAgent

---

9 Testing & QA

- **Unit tests**: pytest + pytest-asyncio for every task.
- **Contract tests**: Pact between FE & API JSON spec.
- **E2E smoke**: Playwright script opens canvas, runs demo pipeline in CI.
- **Load test**: Locust hitting `/pipeline/run` 50 RPS, monitor < 250 ms P95.

---

10 Deployment & DevOps

Env	URL	Provider
Preview (PR)	`*.vercel.app`	Vercel
Staging	`staging.agentops.ai`	Fly.io + Aura Dev

Env	URL	Provider
Prod	`agentops.ai`	Fly.io (region sjc) + Neo4j Aura Prod

CI/CD via GitHub Actions:

- `push` ➜ lint + test + build.
- `main` ➜ deploy preview.
- `v* tag` ➜ promote to prod.

---

11 Scalability & Performance

- **Horizontal scaling** via Fly.io machines; Conductor is stateless when using Postgres persistence.
- **Cold-start**: keep 1 warm worker; Groq latency ≈ 50 ms.
- **KG**: use Neo4j GDS for batched embedding similarity.

---

12 Sponsor API Integration Playbook

1. **Groq** – set header `x-groq-api-key`; endpoint `https://api.groq.com/openai/v1/chat/completions`.
2. **Claude** – `https://api.anthropic.com/v1/messages` model `claude-3-opus-20240229`.
3. **Gemini Vision** – `POST https://generativelanguage.googleapis.com/v1/models/gemini-1.5-vision/latest:generateContent`.
4. **Vapi** – Websocket `wss://api.vapi.ai/v1/` – see `voice/handler.ts`.
5. **Fetch.ai uAgents** – install `uagents==0.10.3`, run inside `tasks/`.

---

13 Roadmap & Milestones

Date	Milestone	Owner
17 Jun	Documentation v0.1	Shubhankar
20 Jun	Canvas MVP running	Neha (FE), Chris (API)
23 Jun	LLM nodes + metrics	Shubhankar
25 Jun	Knowledge Graph online	Chris
27 Jun	Context Copilot demo	Shubhankar
29 Jun	Full dress-rehearsal pitch	All

---

14 Contribution Guide

1. Fork, create branch `feat/<module>`.
2. Run `make pre-commit` before PR.
3. PR template requires: description, checklist, screenshot/gif, linked issue.

Coding conventions:

- Python – black, isort, mypy strict.
- TS/JS – eslint, prettier, strictNullChecks.

---

15 Glossary

Term	Meaning
LLM Node	A task that wraps a call to an LLM provider.
GraphRAG	Retrieval-Augmented Generation using Knowledge Graph + Vector index.
Orkes	Managed Conductor SaaS used for workflows.
Copilot	Multimodal assistant triggered by eye/voice signals.

---

16 Appendix

16.1 `.env.example`

```
CLAUDE_API_KEY=
GROQ_API_KEY=
GEMINI_API_KEY=
VAPI_API_KEY=
ORKE
S_API_KEY=
NEO4J_URI=bolt+s://<id>.neo4jsandbox.com:7687
NEO4J_USER=neo4j
NEO4J_PASSWORD=<password>
REDIS_URL=redis://localhost:6379/0
```

16.2 Sample Orkes task definition

```
{
  "name": "groq_call",
  "retryCount": 1,
```

```
  "inputKeys": ["prompt"],
  "outputKeys": ["completion", "tokens", "cost", "latency"],
  "timeoutSeconds": 60
}
```

End of Document