HEIDI AI / aiwebapp

Professional Implementation Plan (Short Version)

Document date: February 10, 2026

Purpose: Convert the attached idea into an actionable, phased implementation plan for a local-first, gamified AI “brain builder” that can later grow into a privacy-preserving, decentralized “World Seed” network.

1. Executive summary

HEIDI AI is a local-first web app that lets non-technical users “teach” a local AI like a game: drag-and-drop knowledge and tools into a visual brain canvas, earn XP, unlock skills, and see transparent reasoning traces. Under the hood, the app bridges the browser to a local model server (Ollama) through a Next.js API proxy, indexes user files into a local retrieval store, and updates the agent runtime configuration (e.g., opencode.json). In later phases, multiple HEIDI instances connect peer-to-peer (P2P) on LAN/Internet to exchange privacy-safe metadata/embeddings and to verify reasoning via a Proof of Contribution + human “Consensus Duel” UI.

2. Goals and success criteria

- **Usability:** “Training” feels like a game (drag & drop, animations, rewards).
- **Local-first privacy:** user data stays on-device by default; sharing is opt■in.
- **Trust:** reasoning trace + checkpoints before risky actions.
- **Extensibility:** skills/tools can be plugged in and unlocked over time.
- **Network readiness:** optional P2P discovery and verification layer.

3. Solution overview

Core modules

- **API Proxy (“Engine”):** Next.js route bridges UI → Ollama chat API (no direct browser-to-Ollama).
- **Cockpit UI:** model/temperature/context controls + run button + output console.
- **Brain Builder (Canvas):** React Flow-based node canvas with drag-and-drop inventory (knowledge, tools, personalities).
- **Teaching pipeline:** dropping a folder/doc triggers server actions: ingest → chunk → embed → store (local vector DB) and/or update agent instructions/context paths.
- **Gamification:** Zustand global store + Framer Motion aura/XP animations, skill tree unlocks, avatar progression.
- **Governance:** Consensus UI (“Logic Duel”) for humans to choose the best reasoning path, feeding Proof of Contribution.
- **Network layer (future):** libp2p + mDNS + encryption for discovery and privacy-safe exchange.

4. Reference architecture

Front-end (Next.js / React)
• Cockpit (settings + prompt + output)
• Brain Builder (nodes + edges + inventory)
• Gamified overlays (Aura, XP toast, avatar, memory jar, reasoning log)

Back-end (Next.js API routes + Server Actions)
• /api/ollama → local Ollama chat endpoint
• teach/ingest actions → embeddings + local vector store + config updates
• unlock-skill actions → inject tool configuration into opencode.json

Local services
• Ollama (model runtime)
• Vector store (e.g., ChromaDB or lightweight local storage)

Optional P2P layer (future)
• Discovery (LAN first), encrypted transport, signed ledger of contributions

5. Phased implementation plan

Phase	Outcome	Key deliverables
0 — Foundation	Local dev baseline ready	Next.js app shell; local Ollama running; Tailwind (optional); config scaffoldir
1 — Engine + Cockpit (MVP)	Chat works reliably via proxy	API proxy route; cockpit UI with model/temp/ctx; basic error handling
2 — Brain Builder (MVP)	Drag & drop teaching surface	React Flow canvas; inventory items; node types (Knowledge/Tool/Personal
3 — Teaching pipeline	Dropping data actually teaches	Ingest service; chunk/embed/store; retrieval hook; opencode.json sync
4 — Gamification + Trust	Game feel + safe execution	Aura/XP; skill tree; level up; checkpoints; reasoning log; memory jar
5 — P2P “World Seed” (Pilot)	LAN peer discovery and sharing	libp2p node; encrypted discovery; opt-in sharing portal; contribution ledger
6 — Consensus & Polish	Human-verified reasoning improves the system	Upvote/downvote; stake-weight voting; contribution scoring; dispute handling

6. Operating model and governance

- **Genesis identity:** create a “birth” record (timestamp + hash) that anchors reputation and later network contribution history.
- **Contribution accounting:** XP rewards for validated actions (teaching, verified fixes, consensus votes).
- **Opt-in sharing:** only content placed into a “Public/Community” node is eligible for exchange; default is private.
- **Safety gates:** checkpoint cards for risky actions (filesystem writes, commands).

7. Security, privacy, and compliance notes

- **Local-first:** keep raw files on device; store embeddings locally.
- **Encrypted transport:** for P2P use modern noise-based encryption and authenticated peers.
- **Minimize leakage:** share only metadata/embeddings; avoid raw weights/documents unless explicitly permitted.
- **Abuse resistance:** mitigate Byzantine peers with stake-weighted voting, rate limits, and audit logs.

8. Key risks and mitigations

- **Latency / coordination overhead:** start LAN-first, cache results, and use asynchronous verification queues.
- **Byzantine attacks:** require signatures, stake-weighted consensus, and anomaly detection.
- **Serverless filesystem constraints:** keep “write opencode.json” features for local deployments (or move to a writable service).
- **Model hallucinations:** prioritize retrieval grounding + consensus duels for disputed outputs.

Appendix: Minimal MVP checklist

- ■ Ollama running locally with CORS enabled; Next.js proxy route returns chat responses.
- ■ Cockpit UI can set model/temperature/context and run prompts.
- ■ Brain Builder canvas supports dropping a “Knowledge Folder” and triggers an ingest action.
- ■ Local embeddings store created; retrieval is wired into prompts (RAG).
- ■ XP increments + simple aura feedback when training runs.

Source: Idea attachment (Heidi-ai-AGI.txt).