

Net Gemini — Hybrid Social & AI Shell

Architecture skeleton · 2026-10-02

Executive framing

Net Gemini (mensmercatus-site) is a Vite/React TypeScript shell with Chat, Live Voice, Social Feed, Website Builder, Net Code, Net Learn, and Netzar Lab modes. Gemini SDK wiring exists in services; no .env.local was found in the workspace tree — do not claim configured API keys or proven live inference.

System thesis

Treat the product as a multimodal agent shell: chat retrieval via grounded search tools (when keyed), live voice session state machine, generative website/code assist surfaces, and a learning curriculum view. Mode switching is client state; secrets belong server-side or in local env — neither proven here.

CHAT · LIVE ChatView + LiveInterface geminiService chat session googleSearch tool designed Live connection state enum Keys: NOT present in tree Status: UI + SDK stubs	SOCIAL · WEB SocialFeed pulse surface WebsiteBuilder generative UI Image generate helper (SDK) Client mode = AppMode enum No claimed prod social graph Baking content models	CODE · LEARN · NETZAR CodeStudio assist surface LearnView curriculum NetzarLab protocol desk Hybrid Social & AI framing Vite + React + TS stack No .env.local inventoried
---	---	--

Non-goals (explicit)

- Not a claim that GEMINI_API_KEY / .env.local exists or that live calls succeed.
- Not a production social network — Feed is a product surface under construction.
- Not Netzar legal/payments production — Lab is exploratory UI.

Schematic A — Control loop

mode select → prompt → (optional) Gemini → render pane



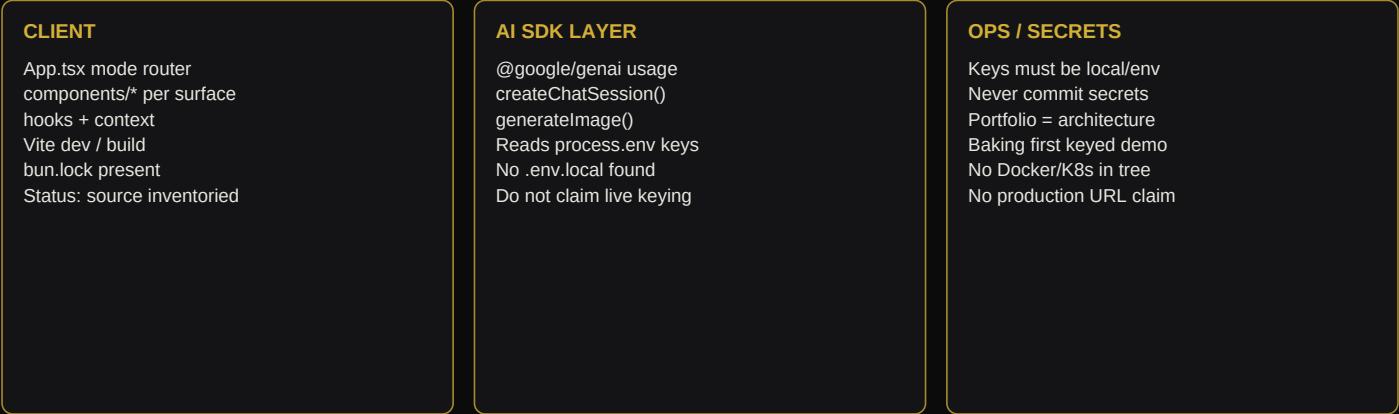
Client-first modes · SDK present · secrets absent from inventory — baking keyed runs

ML / systems vocabulary (as designed)

Agentic chat	Chat session object with optional system instruction + search tool config.
Grounding (designed)	googleSearch tool on chat — requires valid key; not proven in this tree.
Live state machine	disconnected → connecting → connected → error for voice path.
Multimodal generate	Image helper via Gemini flash-image path when keyed.
Latency budgets	TTFT for chat; live audio RTT; UI mode switch < one frame budget.
Evals (designed)	Tool-call usefulness, hallucination on builder output, live drop rate.

Schematic B — Runtime topology

Vite/React · services/geminiService.ts · env not in tree



Secret & runtime honesty

Inventory found Gemini client code paths but no .env.local. Portfolio language must stay at 'SDK wired / baking keyed runs' — never 'live Gemini production'.

DEV CADENCE (DESIGNED)

npm/bun install → provide local GEMINI_API_KEY → vite dev → exercise Chat then Live
Builder/Code/Learn can be UI-reviewed without keys; model panes need secrets
No scheduled scrape plane in this app

Eval plan · risks · portfolio note

[How to read this artifact](#)

Evaluation axes (when live / under demo)

- Mode isolation — switching AppMode does not leak stale session UI.
- Key absence UX — graceful empty-key errors, no silent fake completions.
- Builder faithfulness — generated site/code marked model-assisted.
- Live drop monitoring — connection state visible to operator.

Primary risks

- Accidental key commit if .env.local is created carelessly.
- Overclaiming Live/Social as production systems.
- Model/version strings in code may drift from what Google serves.

PORTFOLIO READING GUIDE

Skeleton for Net Gemini hybrid shell: seven modes, Gemini SDK wiring, secrets not inventoried.

Pair with mensmercatus-site README and services/geminiService.ts. No demo video in this batch.

We're baking keyed runs — UI is real; live inference is not claimed.