Sovereign Personal AI Assistant

A local assistant with the polish of a modern chat app and encrypted phone access, scaling consumer → enterprise on one architecture

● Design & build plan – consumer tier buildable now; enterprise tier a hardware step, not a rewrite

Abstract

Cloud assistants are polished but place every prompt, response, and history on someone else's server; local assistants keep data on owned hardware but have historically been desktop-bound. This design closes the gap: a sovereign assistant with the comfort of a modern chat interface – chat history, projects, artifacts, tool use – reachable from a phone, while the model and every conversation stay on hardware the owner controls. The core mechanism is a host that serves an open-weight model behind an OpenAI-compatible endpoint, bridged to a thin mobile client by an end-to-end-encrypted mesh that opens no ports. The design is deliberately a consumer product first, and the same architecture scales to a team by adding hardware for compute and a governance layer for isolation. This document specifies the architecture, situates it in the 2026 local-inference and remote-access landscape, states an honest hardware envelope with real models that fit the card, and – because it is a build plan rather than a measured deployment – expresses its central claims as falsifiable acceptance criteria, enumerates the threat model, and designs the enterprise multi-tenant path rather than asserting it.

Keywords: local inference · data sovereignty · OpenAI-compatible API · encrypted mesh · open-weight models · MoE · multi-tenant · consumer→enterprise scaling

§01

The idea

Cloud assistants are polished but put every prompt, response and history on someone else's server. Local assistants keep the data local but have been desktop-bound. This project closes the gap: a sovereign assistant with the comfort of a modern chat interface – chat history, projects, artifacts, tool use – reached from a phone, while the model and every conversation stay on owned hardware.

It is deliberately a consumer product first – not evaluation infrastructure, not a professional operator workstation, not research. The design that works at home is the same one that scales to a team; the compute variable is hardware, and the governance variable is a software layer (§9).

Two forces make the timing right in 2026. The first is privacy: the only architecture that can offer a genuine "data never leaves the device" guarantee is one where inference and storage are both local – a property cloud providers cannot match by construction. The second is economics: once the hardware is in place, self-hosted inference runs at a marginal cost dominated by electricity, and at sustained usage it crosses below commercial-API spend. Sovereignty is no longer a trade of convenience or money against privacy; increasingly it is cheaper and more private, provided the model fits the card (§4).

§02

Remote architecture (the differentiator)

The host runs the model behind an OpenAI-compatible endpoint; the phone is a thin client; the two are bridged by an end-to-end-encrypted mesh that opens no ports and exposes nothing to the public internet. Inference runs on the host, chat history stays on the devices, and the only thing that reaches the vendor's backend is the device-discovery list used to pair the machines (Figure 1).

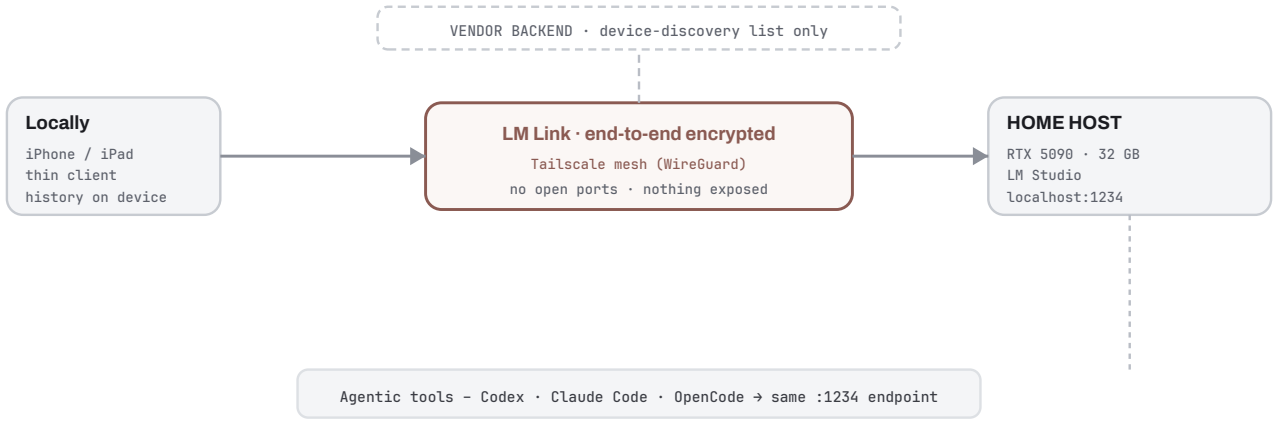
Concretely this maps to a shipping consumer product. On 4 June 2026, LM Studio 0.4.16 shipped Locally – a first-party iPhone/iPad app (the former standalone Locally AI app, acquired in April 2026) – together with LM Link, a remote-access layer that bridges desktop models to the phone over an end-to-end-encrypted Tailscale mesh [1, 2]. The two are distinct: Locally is the client app; LM Link is the transport. The security property is the point: the devices communicate over a mesh VPN (WireGuard-based, via `tsnet`) with no ports opened to the internet, chats remain on each device, and only the device-discovery list touches the vendor's servers [3]. Because the host still serves the ordinary OpenAI-compatible API on `localhost:1234`, the same remote surface is reachable by any tool that already targets that endpoint.

The property is general, not tied to one vendor. Alternatives implement the same "reach owned hardware over an encrypted tunnel" pattern differently: Off Grid is an open-source client for iOS and Android that auto-discovers LM Studio/Ollama servers and additionally runs a smaller model on-device as an offline fallback; 3sparks Chat builds the remote link on Tailscale directly; and browser front-ends such as Open-WebUI point at the same local endpoint [4]. What is invariant across all of them is the architecture, not the app: reach capable owned hardware, from anywhere, over an encrypted tunnel nobody else can read.

Honest current limits of the first-party path: the mobile client is iPhone/iPad only (Android unannounced), both ends run the same app, pairing is account-gated, and phone sessions default to a shorter context (8k) [1]. These are convenience-layer constraints, not architectural ones – the open-source alternatives above relax several of them.

The remote architecture (the differentiator)

Inference and chat history stay on owned hardware; the phone is a thin client over an encrypted tunnel that opens no ports.



§03

Related work: the local-inference and remote-access landscape

The design sits on top of a mature 2026 local-inference stack. Backends – Ollama, vLLM, llama.cpp, and LM Studio – all expose the same OpenAI-compatible surface, which is what makes the assistant model-agnostic and tool-compatible. OpenAI's own open-weight release, gpt-oss (August 2025, Apache 2.0), was designed for exactly this class of deployment and shipped with native MXFP4 quantization so the smaller model runs in ~16 GB and the larger fits a single 80 GB card [5, 6]. The remote-access question – how to reach a home model from a phone without exposing it – is the newer frontier, and 2026 produced a cluster of answers built on the same encrypted-mesh idea (LM Link, Off Grid, 3sparks, Tailscale-based tunnels) rather than on port forwarding, dynamic DNS, or reverse proxies [1, 3, 4]. This project's contribution is not a new backend or a new tunnel, but the composition: a polished, phone-reachable, sovereign assistant whose consumer build and enterprise build differ only in hardware for compute and a governance layer for isolation, assembled from these commodity parts under one discipline – the same local-inference discipline as the evaluation and workstation projects it sits beside.

Honest hardware envelope

Sovereignty is only real if the model fits the card (Figure 2).

Consumer – 1× RTX 5090 (32 GB). Runs a model that fits with headroom:

Model	Footprint	Note
gpt-oss-20b	~13 GB (MXFP4)	fast daily driver; 20.9B total / 3.6B active
Gemma4-31B	~19 GB	dense, strong reasoning
Qwen3.6-35B-A3B	~24 GB	MoE, ~3B active; reported ~73 on SWE-bench Verified

These footprints are accurate to the shipping weights: the gpt-oss-20b MXFP4 checkpoint is 12.8 GiB [6], and the RTX 5090's 32 GB runs gpt-oss-20b at the full 128k context without offloading. Qwen3.6-35B-A3B is notable as evidence that the consumer tier is not a toy: a ~3B-active MoE reported (by the Qwen team) in the range of far larger closed models on SWE-bench Verified.

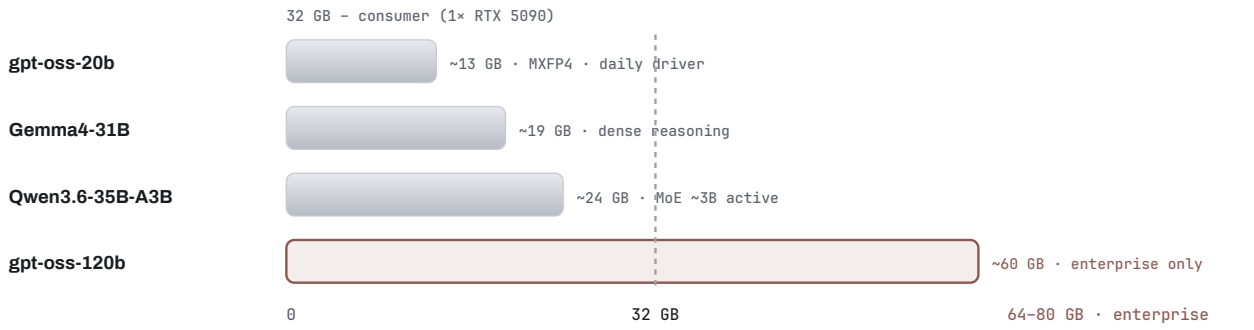
Enterprise – 2× 5090 (64 GB) or an 80 GB host. Adds the model the consumer card cannot hold:

Model	Footprint	Why not consumer
gpt-oss-120b	~60 GB (MXFP4)	116.8B total / 5.1B active; needs 64–80 GB. On 32 GB it offloads ~half the weights to system RAM and hits the bandwidth cliff.

The 120B is not a consumer claim – it is exactly what the enterprise tier adds when the hardware arrives [6]. Two honest technical notes underlie the table. First, **the bandwidth cliff is real and quantifiable**: the 5090's ~1.8 TB/s of GDDR7 bandwidth is one to two orders of magnitude above the tens of GB/s of system DDR5, so the moment weights spill to host RAM, throughput collapses – which is why "fits in VRAM" is the binding constraint, not raw model size. Second, **MoE saves compute per token, not resident memory**: all experts must stay loaded, so a 35B-A3B model occupies memory like a 35B model even though it computes like a ~3B one [6]. A quality caveat consistent with our evaluation work: 4-bit formats (MXFP4/NVFP4) hold within roughly 2–4 percentage points on most tasks but are most likely to lose ground on complex multi-step reasoning, so the choice of quant is a quality decision, not only a memory one [8]. This caveat is made testable as acceptance criterion C5 below.

Honest hardware envelope – what fits the card

Sovereignty is only real if the model fits; the consumer→enterprise step is hardware, not a rewrite.



§05

The stack

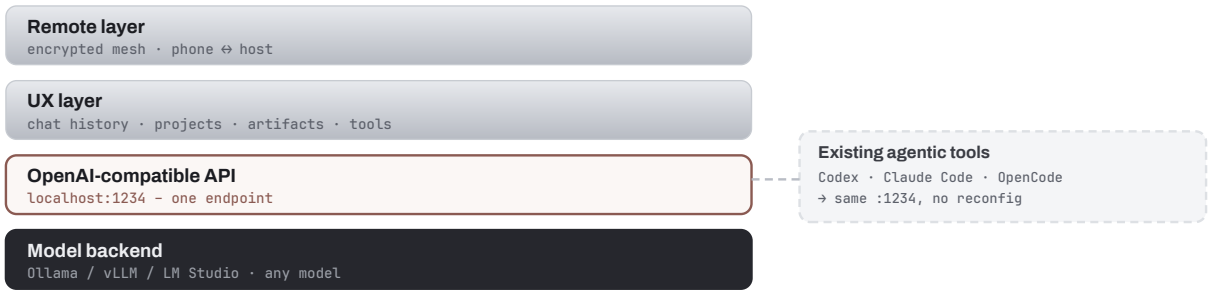
Because the server is OpenAI-compatible, the assistant is model-agnostic and tool-compatible (Figure 3):

- **Model backend** – Ollama / vLLM / LM Studio, any open-weight model.
- **OpenAI-compatible API** – `localhost:1234`, one endpoint.
- **UX layer** – chat history, projects, artifacts, tool-use.
- **Remote layer** – encrypted mesh, phone ↔ host, same endpoint.

Existing agentic tools (e.g. Codex CLI, Claude Code, OpenCode) target the same endpoint and keep working, locally or remotely, with no reconfiguration [3]. The same decoupling is what lets the enterprise tier reuse the layers above the API unchanged: swapping the backend for a serving engine that does continuous batching (vLLM) turns the single-user assistant into a multi-user one for throughput [7] – but multi-tenancy (who may see what) is a separate concern, designed in §9 rather than assumed.

The stack – model-agnostic, tool-compatible, one endpoint

An OpenAI-compatible endpoint decouples everything above the API from the model below it.



§06

Claims and acceptance criteria (evaluation plan)

This is a design & build plan, not a measured deployment, so its central claims are stated here as falsifiable acceptance criteria, each with a defined instrument and a pass/fail threshold, to be evaluated on the built system. Every criterion is marked predicted until measured – the same measured/predicted discipline used across the CTC papers. Thresholds marked [proposed] are defaults offered for confirmation.

The workstream-level hypothesis (W5) is: a local-plus-remote assistant preserves data sovereignty (zero content egress) at usable latency, on one architecture that scales consumer → enterprise by adding hardware for compute and a governance layer for isolation. It decomposes into five criteria.

- **C1 – Content egress is zero.** The defining claim, and the most directly measurable one. Instrument: packet capture at both the host NIC and the network gateway during a defined session (chat, tool use, document attach), with traffic classified by destination and payload. Pass: the only traffic leaving the owned network is device-discovery / mesh-coordination metadata; no prompt, response, or history bytes reach any third-party endpoint. Falsification: any content-bearing egress to a non-owned destination. (predicted)
- **C2 – Usable latency over the mesh.** Metric: time-to-first-token (TTFT) and sustained tokens/s, measured local-baseline vs. mesh across home Wi-Fi, cellular, and a CGNAT / hotel-Wi-Fi path. Pass [proposed]: TTFT ≤ 2 s and sustained throughput ≥ 15 tok/s (comfortably above reading speed) on the consumer model at 8k context, with mesh overhead ≤ 20% versus the local baseline. Falsification: throughput below reading speed, or TTFT beyond threshold, on any tested network. (predicted)
- **C3 – One architecture, hardware-only scaling of compute.** Claim: consumer → enterprise changes nothing above the API for compute. Instrument: the identical client and identical OpenAI-compatible endpoint

exercised against a 32 GB single-user host and a 64–80 GB multi-user host, plus a concurrency curve (tok/s per user at c = 1…N under continuous batching) [7]. Pass: no client or endpoint change required, and per-user throughput stays above the C2 usability floor up to the target seat count. Falsification: any code/endpoint change needed to move tiers, or per-user throughput collapsing below the floor at target concurrency. (predicted)

- **C4 – Break-even against cloud.** Metric: total cost of ownership (hardware amortization + electricity) versus equivalent cloud token spend at the owner's measured usage, with an explicit crossover month. Pass: a defined break-even point under stated assumptions (usage, electricity price, hardware cost) – the owner's own numbers rather than a generic study. Falsification: no crossover within the hardware's useful life at realistic usage. (predicted)
- **C5 – Judgment quality under quantization.** Claim: the chosen quantized assistant model is good enough for daily use. Instrument: score the MXFP4 assistant model against a full-weight or cloud reference on a small, fixed task set, using the frozen-rubric judge from the evaluation pipeline (W1) – closing the loop between the two workstreams. Pass [proposed]: task-success / agreement within a pre-set margin of the full-weight reference (e.g. ≤ 3 pp degradation on the set). Falsification: degradation beyond the margin on reasoning-heavy tasks [8]. (predicted)

None of these require frontier-scale compute; all are runnable on the consumer build with a router, a laptop, and the eval pipeline already in the program – which is the point. C1 in particular converts the paper's headline promise from an assertion into a test.

§07

Threat model and the sovereignty boundary

"Sovereign" should be a precise claim, not a slogan. Under normal operation, inference runs entirely on the owned host, chat history is stored only on owned devices, and no prompt, response, or document is sent to any model provider; the encrypted mesh opens no inbound ports, so there is no attack surface from port forwarding, reverse proxies, or dynamic DNS [3]. The guarantee is bounded, however, and the honest statement of it is an enumeration of what it does and does not cover.

- **Compromised host.** If the host itself is compromised, local inference and history are exposed. Sovereignty defends against third-party cloud exposure, not against endpoint compromise. Mitigation: host hardening, full-disk encryption, and the same hardened-sandbox discipline used in the evaluation stack for any untrusted tool execution.
- **Compromised or malicious client.** The mobile client holds chat history; a malicious app or a stolen, unlocked phone exposes it. Mitigation: device encryption, trusted app sources, biometric lock, minimal on-device retention.
- **Device-discovery / account-system trust.** Pairing is authenticated through the vendor's account system, and the mesh's coordination plane brokers which devices may find each other. A compromise of that account could let an attacker's device attempt to pair. This is the single residual third-party touchpoint – the device list, not the conversation, is what leaves [1, 3]. Mitigation: strong account auth with 2FA, or self-hosting the coordination plane to remove the dependency entirely.
- **Backend / supply chain.** The inference backend, the model weights, and the mesh client are third-party software; a poisoned weight file or a backdoored binary is a genuine vector. Mitigation: pinned versions, checksum/signature verification of weights, and a preference for open-weight, open-source components that can be audited.
- **On-path network adversary.** A hostile network (hotel Wi-Fi, CGNAT) sees only ciphertext; the mesh is end-to-end encrypted and exposes no ports [3]. This is the case the architecture handles best.

The defensible claim is therefore precise: content never leaves owned hardware under normal operation, bounded by host and client integrity and by the small device-discovery dependency – not "no third-party software is ever involved." Criterion C1 verifies the content-egress half of that claim empirically.

§08

Economics: when local pays for itself

The privacy case used to carry a cost penalty; in 2026 it often does not. Once the hardware is in place, self-hosted inference on a consumer Blackwell card runs at a marginal cost dominated by electricity, and at sustained usage the amortised cost per token falls below budget-tier cloud APIs. For a privacy-conscious professional the calculus is therefore doubly favourable: the confidential material never leaves the machine, and the per-token cost drops below cloud once the card is paid off. Rather than rest on a generic industry figure, criterion C4 instantiates this as the owner's own break-even calculation on measured usage, electricity price, and hardware cost. For the enterprise tier the same hardware serves several seats via continuous batching, so the per-seat economics improve rather than degrade as the deployment grows – the consumer→enterprise path is an efficiency gain, not just a capability one.

§09

Enterprise path: multi-tenancy, auth, and data governance

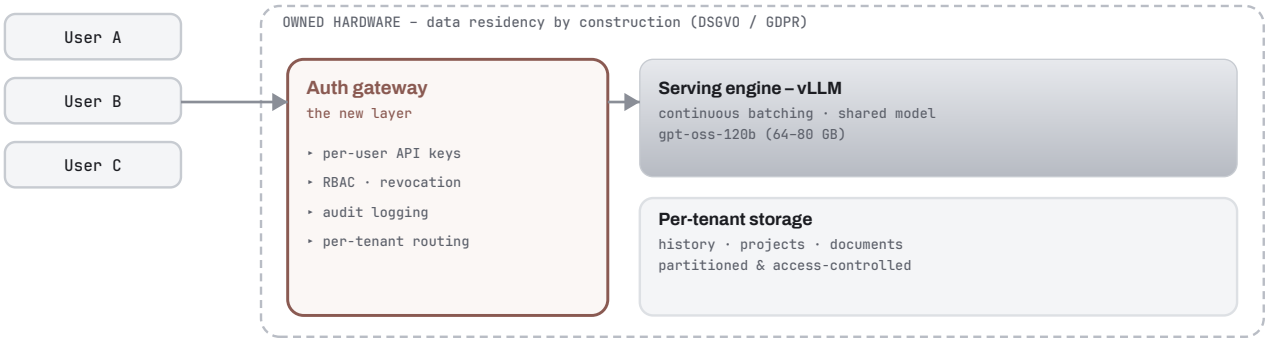
"The same stack serves the team" is true for compute and misleading for governance, and the paper should be precise about the difference (Figure 4). Continuous batching (vLLM) lets one shared model serve many concurrent users [7], and local hosting satisfies data residency by construction – a genuine advantage for DACH mid-market clients under the DSGVO / GDPR. But a single-user assistant does not become multi-tenant for free; a governance layer must be built above the API.

- **Authentication.** The OpenAI-compatible endpoint needs per-user API keys behind an auth gateway so users are distinguishable, rate-limitable, and revocable – the single-user design has no notion of "who" is calling.
- **Per-tenant isolation.** Chat history, projects, and attached documents must be partitioned per user/tenant with access control; a shared model does not imply shared data. Isolation lives at the gateway and storage layers, not in the model.
- **Governance and audit.** For a firm's confidential material, retention policy, audit logging, RBAC, and data-subject deletion must exist even though the data never leaves the premises. Local residency removes the cross-border transfer problem but not the record-keeping and access-control obligations – this is exactly the secure-by-design surface that CTC Advisory addresses.

The honest framing, then: **compute and residency scale by adding a card; multi-tenant governance is a software layer that must be built.** It is not a rewrite of the assistant – the client and endpoint are unchanged (C3) – but it is not nothing, and it is the part a co-founder or a first enterprise customer will interrogate before anything else.

Enterprise path – what changes beyond hardware

Compute scales by adding a card; multi-tenant governance is a software layer that must be built.



§10

Practical use case (STAR)

- **Situation.** A privacy-conscious professional – and later their small firm – wants a capable assistant with the comfort of a modern chat interface including phone access, but confidential material cannot go to a third-party cloud, and desktop-bound local setups are useless away from the desk.
- **Task.** Stand up a sovereign assistant that is genuinely usable day to day, reachable from a phone anywhere, and runs a model that actually fits the hardware – with a clean path to a team that doesn't require rebuilding the core.
- **Action.** On the home RTX 5090, serve a 32 GB-fitting model (gpt-oss-20b or Qwen3.6-35B-A3B) through the OpenAI-compatible endpoint; wrap it in a polished chat UX; reach it from the phone over the encrypted mesh – no open ports, inference and history on owned hardware.
- **Result.** A personal assistant the owner uses from their phone with nothing leaving their control – verified by C1 – and a documented enterprise path: add a second 5090 (or an 80 GB host) for compute and the auth/isolation layer of §9 for governance, and the same client and endpoint serve a 120B-class model to the whole team over the same encrypted mesh, still with no public exposure.

§11

Status

Buildable now at the consumer tier with real, fitting models; the enterprise tier is a hardware step for compute and a defined software step for governance, not a re-architecture. The core property holds at every scale: inference and history stay on owned hardware – the whole reason to build locally – and the acceptance criteria in §6 make that property testable rather than merely asserted. Designed under CTC AI Operations, on the same local-inference and falsifiable-claim discipline as the evaluation and workstation projects it sits beside.

References

[1] LM Studio. (2026). Run (your largest) local models from your iPhone – introducing Locally + LM Link. LM Studio Blog, 4 June 2026. lmstudio.ai/link. [2] LM Studio. (2026). Locally AI joins LM Studio. LM Studio Blog, 8 April 2026. [3] Tailscale. (2026). LM Link: encrypted access to remote LLMs on hardware you own. tailscale.com (Tailscale mesh; no open ports; device-discovery only). [4] Community tooling for local-model remote access: Off Grid (open-source, iOS/Android, network discovery + on-device fallback); 3sparks Chat (Tailscale); Open-WebUI (browser front-end for OpenAI-compatible endpoints). [5] OpenAI. (2025). Introducing gpt-oss. Apache 2.0 open-weight models; native MXFP4; 128k context. [6] OpenAI. (2025). gpt-oss-120b & gpt-oss-20b Model Card. arXiv:2508.10925 (checkpoint sizes 60.8 GiB / 12.8 GiB; 116.8B/5.1B and 20.9B/3.6B). [7] Kwon, W., Li, Z., Zhuang, S., et al. (2023). Efficient Memory Management for LLM Serving with PagedAttention. SOSP. arXiv:2309.06180. [8] Li, Z., Su, Y., Yang, R., et al. (2025). Quantization Meets Reasoning: Exploring LLM Low-Bit Quantization Degradation for Mathematical Reasoning. arXiv:2501.03035.