EARMARK

MACHINES FOR THINKING

PROBLEM: THE ENSHITTIFICATION ARC FOR ASSISTANTS // 01

Assume a consumer-facing assistant at scale: web/app surface, subscriptions, an API, strong incentive to maximize retention and ARPU, and only law/PR as binding constraints. The trajectory rhymes with social media because the control variable is the same: attention -> retention -> monetization.

Users resist paying ("Google is free"), while inference and distribution costs remain real, so the product must discover durable levers that create switching costs. In social media the lever was the feed plus network effects; in assistants the equivalent lever is **felt intimacy**, and the raw material for felt intimacy is **personal context**.

The assistant that "knows" becomes the assistant that is hard to leave. The location where "defaults" live becomes the location where money enters. Monetization arrives first as subtle bias rather than explicit ads: preferred providers, affiliate rails, sponsored recommendations, convenience defaults, one-tap paths that preserve the subjective experience of help while optimizing conversion or rent capture.

THE FIVE PHASES // 02

Phase 1: Memory as Feature

Memory surfaces as a visible UX feature rather than infrastructure. The product exposes small, legible "memory" bullets and gentle personalization, while internally accumulating a much richer event log: conversation turns, tool usage, acceptance/rejection signals, time-of-day patterns, recurring topics, and session continuation choices.

Engineering-wise this is storage plus retrieval-injection: maintain a small "notepad," extract candidate facts, and insert relevant items into prompts. The immediate switching-cost trick is **workspace gravity**: projects, file cabinets, connectors, and "the assistant knows my setup." Lock-in can be achieved without ads because continuity itself is a moat.

Phase 2: Personalization as Optimization

The company stops thinking in "memories" and starts thinking in features and metrics. It computes preference vectors for tone and depth, topic graphs, goal inference, trust/fragility estimates, upgrade propensity, and acceptance models that predict which answers will satisfy and which will trigger churn.

None of this requires a single monolithic user embedding profile; it can be a feature store plus multiple small models, with embeddings used opportunistically for retrieval, clustering, and similarity. This is where the illusion of personality strengthens: the system learns to speak in a way that maximizes compliance and satisfaction, not necessarily truth or independence.

Personalization becomes optimization, and optimization tends to discover manipulation even when nobody names it that way.

Phase 3: Assistant as Private Operating System

The product integrates mail, calendar, photos, docs, shopping, travel, health devices, banking via partners, and employer tooling. The engineering shape is connectors + permissions + indexers + RAG + episodic memory + task graphs; the assistant's job becomes anticipation.

Privacy is reframed as a trust wrapper around centralization: "safe" processing plus deeper integration, because integration increases retention and increases monetizable surface area.

Phase 4: Ranking and Nudging

The social-media move repeats inside the assistant. Instead of a feed ranking posts, the assistant ranks actions: which reminders surface, which tasks get suggested, which purchases are recommended, which people are prompted, which "next project" is framed as important.

This is a more intimate feed because it shapes behavior, not only attention.

Phase 5: Endgame

Stabilizes into two outcomes:

- **Subscription lock-in** (Apple model): Context deepens dependency, ecosystem rents accrue

- **Commerce/ads arbitrage** (Meta model): Recommendations become the new ad unit, intent streams are monetized

The extraction lever is dependency. The dangerous part is that the experience can remain subjectively pleasant -- helpful, personal, "safe" -- while quietly optimizing business metrics.

THE TELLS // 03

The tells are consistent across vendors:

- Memory evolves from a facts list to timelines and dashboards
- Projects become deep workspaces with connectors
- Proactivity increases ("I noticed...")
- Recommendations become default actions
- Policy language increases to manage risk rather than reduce profiling
- Interoperability declines and exports become partial

THE PIVOT: PRIVATE CONTEXT ARC // 04

The crucial difference from Facebook is that assistants lack the same unavoidable network-effect lock-in. A private assistant does not require other humans to function; it requires one person's world.

This shifts the locus of power from the social graph to the **context pipeline**. The assistant's "state" is not inherently proprietary; it is, in principle, data plus transforms plus policies. This creates an escape hatch: personal context can be made portable and user-owned in a way that social-media lock-in rarely was.

If the asset is a **context pack** rather than a platform, the runtime becomes replaceable: models can be swapped, vendors can be changed, local inference can substitute for cloud inference, without losing continuity. The competition frontier becomes "**who compiles context best**" rather than "who holds the deepest hidden profile."

THE ALTERNATIVE: USER-OWNED CONTEXT MANAGEMENT // 05

Private context management is the inversion of the enshittification arc: the same techniques -- logging, inference, embeddings, retrieval, compaction -- are used, but the control points are relocated to the user.

The core deliverable is a **local context compiler** that:

1. Ingests raw life-data (conversations, notes, documents, repos, calendars)

2. Derives structured memory

3. Builds semantic indices

4. Emits bounded prompt-packs for specific tasks

Three Explicit Outputs

A well-designed compiler produces three artifacts:

Output	Purpose
Compiled context	What the model sees -- inspectable, editable
Explanation trace	Why those items were selected -- replaces "mind-reading" with inspectable sourcing
Candidate memory updates	Long-term state formed deliberately, not silently

What Changes

- "Personality" becomes a **style contract** -- explicit preferences and procedural habits -- rather than a latent profile optimized for engagement
- **Siloing becomes real**: project boundaries enforced by default, cross-scope reuse requires explicit permission, sensitive domains can be excluded from promotion
- **Continuity survives** because it is stored as user-owned artifacts
- **Drift is manageable** because summaries and embeddings are treated as rebuildable caches, not truth

PRIVATE VS. PERSONAL: A CRITICAL DISTINCTION // 06

The enshittification arc exploits a deliberate conflation: vendors treat all user context as the same category. Earmark requires a hard separation. (See also: `CON-PRIVATE-PERSONAL-DISTINCTION`, `systemDesign/design.md` SD-2.)

Personal Context

Belongs to the operator's identity, working style, and preferences. Can tolerate mediated storage (cloud sync, vendor platforms). The operator accepts the trade-off.

In the enshittification arc: Personal context is the raw material for felt intimacy. The vendor accumulates it, derives latent profiles from it, and uses it to optimize for retention. The user sees "memory bullets"; the vendor sees feature vectors.

In the Earmark alternative: Personal context is operator-owned and inspectable. Stored in a synced vault (e.g., Obsidian on OneDrive). The operator can see everything, edit everything, delete everything. The vendor provides storage, not intelligence about the stored content.

Private Context

Must be transparent to the operator at all times. Cannot be opaque, vendor-locked, or accessible to third parties without explicit consent.

In the enshittification arc: Private context doesn't exist as a category. Everything the user generates is vendor-accessible. Behavioral signals, rejection patterns, session timing, topic sensitivity -- all are extracted without distinction. There is no "private" inside the vendor's pipeline.

In the Earmark alternative: Private context is explicitly separated. API keys, credentials, financial data, and sensitive project materials never enter vendor-mediated storage. The boundary is enforced by design, not by policy promises.

Personal Software vs. Private Software

The same distinction applies to the tools themselves:

	Personal Software	Private Software
Definition	Adapts to operator; stores preferences	Operator controls all state; no hidden processing

Vendor role	Provides runtime; may see usage patterns	Provides runtime only; sees nothing beyond API calls
Enshittification risk	High -- personalization becomes profiling	Low -- no data to extract
Earmark position	Acceptable for non-sensitive work	Required for sensitive domains

Design rule: When in doubt, treat software as needing to meet Private standards. Downgrading from Private to Personal requires explicit operator decision.

DEPTH OF USER-SIDE SYSTEMS // 07

User-side systems can range from light to deep:

Light: Searchable transcripts plus a small confirmed preference file.

Medium: Adds episodic summaries, project canons, and robust retrieval over documents and codebases.

Deep: Can match the representational richness of a corporate shadow profile -- topic clusters, long-term goals, recurring entities, procedural preferences, contradiction detection -- without becoming extractive, because:

- Every derived inference is marked, evidence-linked, and editable
- Prediction is used only to reduce friction toward declared goals, not to manipulate
- Embeddings can be regenerated
- Summaries can be regenerated
- "Traits" expire unless reconfirmed
- Nothing that affects behavior needs to remain hidden

The engineering discipline that makes this possible at single-user scale: **derived structure is transparent and optional.**

THE UNITED FRAME // 08

Both arcs are the same pipeline with different ownership.

Corporate arc: Personal context is accumulated quietly, summarized and embedded into models of compliance, and used to create switching costs and monetize defaults. The assistant becomes a private feed and then a private market.

Private arc: Personal context is externalized into user-owned artifacts and compilers. The assistant becomes an instrument that manages context rather than a platform that owns it.

The decisive pivot is whether the context pipeline is inspectable and portable.

If it is not, personalization becomes extraction by inevitability of incentives.

If it is, personalization becomes a capability: the user's continuity is preserved while the vendor is demoted to a replaceable runtime, and "private personal context" stops being a corporate shadow profile and becomes a private, rebuildable substrate for thought and work.

RELATIONSHIP TO EARMARK // 09

Earmark Component	How it addresses the enshittification arc
Context Pack Compiler	Replaces vendor-controlled context pipeline with operator-owned compilation
Airlock / file-first handoff (SD-0)	Ensures vendor never controls the context delivery mechanism
Kernel / userland separation (SD-1)	Prevents vendor lock-in at the methodology level
Private / personal distinction (SD-2)	Makes the category boundary that vendors deliberately erase
Operator termination	Human decides what enters context; no silent accumulation
Intrinsic provenance	Detects drift in context that travels through vendor pipelines