ANIMUS

A Framework for Sovereign AI Cognition

Version 1.0 · February 2026

AreteDriver · github.com/AreteDriver/animus

Abstract. Current AI systems are rented. Memory is a feature platform providers can revoke. Context resets at their convenience. The relationship belongs to the vendor, not the user. Animus proposes an alternative architecture: a persistent, sovereign, portable AI that is cryptographically owned by one person and aligned to that person alone. This paper defines the four-layer Animus architecture — Core, Memory, Cognitive, and Interface — and the Forge and Quorum subsystems that enable multi-agent orchestration and coordination. Animus is not an application. It is an exocortex: an extension of human cognition that persists, learns, and grows across years without ever leaving your control.

Contents

1.

The Problem with Rented Cognition

Every AI assistant available today shares a common architectural flaw: the relationship belongs to the vendor. Your context, your memory, your interaction history — all of it lives on someone else's infrastructure, subject to their terms, their pricing decisions, and their roadmap. Memory is not a right you are granted. It is a feature that can be modified, throttled, or revoked.

This is not a criticism of any particular company. It is a structural observation about how current AI products are built. The business model requires the platform to own the relationship. The user is a tenant, not an owner.

> *"Current AI assistants are rented. Your context exists at the discretion of platform providers. The relationship resets at their convenience."*

The consequences compound over time. A personal AI that resets every conversation cannot learn your communication patterns. It cannot recognize the arc of a long-term project. It cannot connect a decision you made in January to a pattern that emerges in October. It can only respond to what you tell it right now.

Three specific failures define the current landscape:

- **Statelessness.** Each conversation begins with zero context. The AI has no memory of who you are, what you care about, or how you think. Every session is a first meeting.
- **Platform dependency.** Your accumulated context — if any is retained — is locked inside a proprietary system. You cannot export it, port it, or own it.
- **Misaligned incentives.** The platform is optimized for engagement, retention, and monetization. Your AI is not working for you. It is working for the company that deployed it.

Animus is designed to solve all three failures at the architectural level, not through product features layered on top of the same broken foundation.

2.

The Philosophy of Sovereign Intelligence

The concept of a personal guiding intelligence is ancient. Daemons in Greek philosophy were intermediary spirits — advisory entities that served one person's interests and remained loyal across a lifetime. Roman genius was the animating force that accompanied a person from birth. Medieval familiars, bound spirits, protective intelligences: every culture has articulated the idea of a cognitive companion that belongs to you.

This is not coincidence. It reflects something true about human cognition: we think better when we think with something. The tools we use to extend our minds shape the quality of our thought. A notebook that knows your handwriting. A library organized by your logic. A collaborator who understands your priorities without explanation.

> *"Animus translates this ancient concept into modern architecture: a persistent, private, portable AI that extends your cognitive capacity without compromising your sovereignty."*

Sovereignty in this context has a precise meaning. It is not privacy in the narrow sense of data protection. It is the broader condition of an intelligence that:

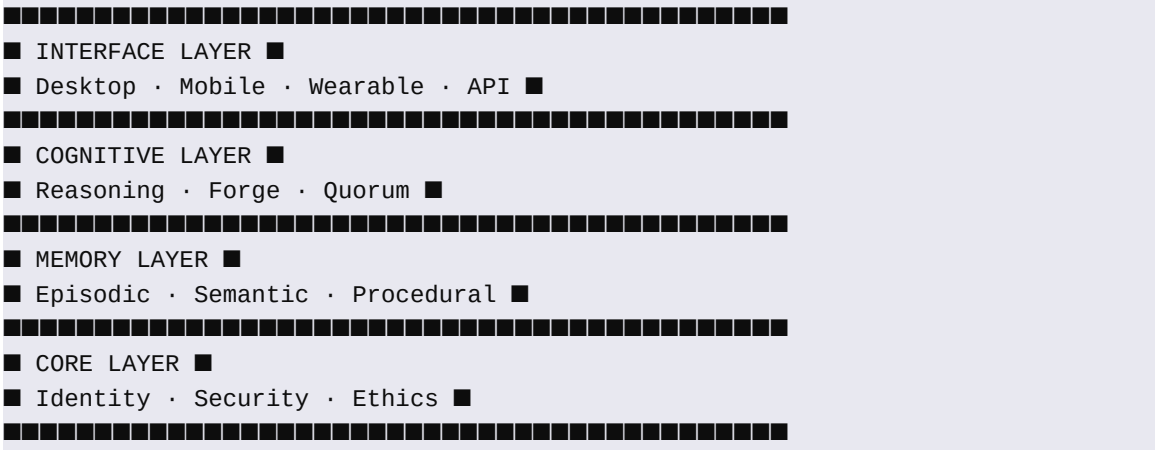
- Belongs to you cryptographically — not just contractually
- Persists across time without external dependency
- Remains aligned to your interests, not a platform's incentives
- Can be moved, backed up, and restored without asking permission
- Grows with you across years, not resets at the end of a session

The word *animus* itself carries this meaning: the animating intelligence, the inner will, the cognitive force that drives a person. This project borrows that word deliberately. The goal is not a better chatbot. It is an architecture for a new kind of cognitive relationship.

3.

Architecture Overview

Animus is organized as four vertically integrated layers. Each layer has a distinct responsibility, and each depends on the layer below it. The stack is designed to be modular — individual layers can be upgraded, swapped, or extended without breaking the whole — while maintaining coherent behavior across the full system.



Layer	Responsibility
Core	Cryptographic identity, security, ethics configuration. Defines who this Animus belongs to.
Memory	Persistent storage across episodic, semantic, procedural, and active context dimensions.
Cognitive	Reasoning, model routing, tool use, and multi-agent orchestration via Forge and Quorum.
Interface	Cross-device interaction with seamless context handoff across desktop, mobile, and wearable.

The two supporting subsystems — Forge and Quorum — operate within the Cognitive Layer but are architecturally distinct enough to warrant separate treatment. Forge handles multi-agent workflow orchestration. Quorum handles agent coordination and convergence. Together they enable Animus to decompose complex tasks across specialized agents while maintaining coherent output.

4.

Core Layer — Identity and Foundation

The Core Layer is the foundation of the entire system. It answers one question before anything else: whose Animus is this? Every operation that Animus performs flows from this layer's answer.

4.1 Identity

Identity in Animus is cryptographic, not contractual. Ownership is established through a keypair generated at initialization. The private key never leaves the user's control. There is no account with a third-party service, no username and password, no recovery flow that routes through someone else's infrastructure. If you hold the key, you own the instance.

4.2 Preferences

The preferences component stores the behavioral configuration of this specific instance: communication style, response register, topic priorities, interaction boundaries. This is the personality layer — the configuration that makes this Animus respond like it knows you, because it does.

4.3 Security

All data at rest is encrypted using the user's key. Device certificates control which physical devices can communicate with the instance. There is no plaintext data store accessible without authentication.

4.4 Ethics Configuration

Animus implements a three-tier ethics model: immutable core constraints that cannot be overridden, user-modifiable defaults that establish baseline behavior, and a free preference zone where the user has full control. The user defines what their Animus will and will not do. The architecture enforces it.

5.

Memory Layer — Persistence Across Time

The Memory Layer is what separates Animus from every stateless AI assistant. It is modeled on human cognitive memory research, with four distinct stores serving different temporal and functional roles.

Memory Type	Contents	Implementation
Episodic	Conversations, events, decisions — what happened and when	Timestamped logs with vector embeddings for semantic retrieval
Semantic	Facts, knowledge, learnings — what you know	Knowledge graph with entity relationships and confidence scores
Procedural	Workflows, habits, patterns — how you work	Structured workflow definitions with trigger conditions
Active Context	Current situation, live priorities, recent threads	Working memory buffer with decay and priority weighting

Storage is hybrid by design. A vector database (ChromaDB or Qdrant) handles semantic search across episodic and semantic memory. A knowledge graph manages structured relationships between entities. SQLite provides indexed metadata retrieval. Flat files ensure maximum portability and longevity. All storage is local by default, encrypted at rest, and fully exportable.

The memory layer does not merely store — it connects. The system identifies recurring patterns across episodic memory, updates semantic memory with learned facts, and surfaces procedural workflows when context suggests they are relevant. Memory is not a log. It is an active participant in reasoning.

6.

Cognitive Layer — The Reasoning Engine

The Cognitive Layer is the reasoning engine of Animus. It is deliberately model-agnostic: Animus can route tasks to local models running on consumer hardware for sovereignty-critical operations, or to frontier API models when maximum capability is required. The user controls which model handles which task type.

Capability	Description
Model routing	Automatic selection of local vs. API model based on task complexity, privacy requirements, and user-defined preferences
Tool use	File access, web search, API calls, device control, calendar integration, communication drafting
Analysis modes	Quick response (sub-second, local model) vs. deep reasoning (multi-step, frontier model)
Register translation	Adjusts communication register to context: formal for work, casual for personal, technical for code review
Forge integration	Complex multi-step tasks delegated to Forge orchestration with Quorum coordination

The key design principle of the Cognitive Layer is that model independence is not a feature — it is a requirement for sovereignty. A system that can only function with one provider's model is not sovereign. It is merely self-hosted dependency.

7.

Interface Layer — Presence Across Contexts

The Interface Layer implements seamless cross-device presence as a first-class architectural concern, not an afterthought. Context follows the user. A thought started on a desktop continues on a phone. A question asked via voice on a wearable draws on the full memory and reasoning stack.

Interface	Primary Mode	Primary Use
Desktop	Full GUI, text-primary	Long-form work, development, complex analysis
Mobile	Voice-first, text fallback	On-the-go queries, quick capture, ambient awareness
Wearable	Voice-only, minimal UI	Ambient awareness, quick queries, notifications
API	Programmatic access	Integrations with external tools and services

Handoff between devices is transparent to the user. The active context buffer in the Memory Layer maintains state across device transitions. The Interface Layer normalizes input modality (voice, text, gesture) into a consistent representation before passing it to the Cognitive Layer.

8.

Forge — Multi-Agent Orchestration

Forge is the orchestration engine that handles tasks too complex for single-turn inference. When the Cognitive Layer identifies a request that requires multiple steps, specialized reasoning, or parallel execution, it delegates to Forge.

Forge draws its design philosophy from lean manufacturing and the Toyota Production System — not from machine learning research. The insight is that multi-agent AI pipelines share the same failure modes as complex production lines: handoff errors, quality drift, resource exhaustion, and unchecked accumulation of work-in-progress. The solutions developed for manufacturing apply directly.

8.1 Core Mechanisms

- **Declarative YAML pipelines.** Workflows are defined as configuration, not code. Each pipeline specifies agents, dependencies, quality gates, and fallback behavior.
- **Per-agent token budgets.** Every agent in a pipeline operates within a defined token budget. Budget exhaustion triggers a quality gate rather than runaway inference.
- **SQLite checkpoint/resume.** All pipeline state is persisted at each step. A failed pipeline can be resumed from the last checkpoint, not restarted from scratch.
- **Specialized agent roles.** Agents are assigned specific roles — researcher, writer, critic, synthesizer — and operate within those roles rather than as general-purpose inference engines.
- **Quality gates.** Each pipeline stage has defined acceptance criteria. Output that does not meet the gate is routed to a revision agent rather than passed forward.

8.2 Operational Philosophy

The lean manufacturing influence is not cosmetic. Pull-based scheduling means agents request work when capacity is available rather than being pushed tasks regardless of current load. Work-in-progress limits prevent cascade failures where one slow agent blocks the entire pipeline. Continuous improvement loops capture quality metrics across runs and surface optimization opportunities.

9.

Quorum — Coordination Protocol

Quorum is the coordination layer that operates beneath Forge. Where Forge manages workflow orchestration — what agents do and in what order — Quorum manages how agents reach agreement when their outputs conflict or diverge.

The design inspiration for Quorum is stigmergy: the mechanism by which ant colonies coordinate complex collective behavior without central direction. Individual agents leave signals in a shared environment. Other agents respond to those signals. Coordinated behavior emerges from local interactions rather than top-down instruction.

9.1 Intent Graph

Quorum maintains an intent graph: a directed representation of the current task decomposition, agent assignments, and dependency relationships. The intent graph is visible to all agents in the pool. Agents can read the graph to understand where their output fits in the larger structure, and write signals indicating their status, confidence, and output availability.

9.2 Stability Scoring

Each node in the intent graph carries a stability score that reflects the confidence of the assigned agent's output and the agreement level of any reviewing agents. Quorum converges when all nodes reach a defined stability threshold. Nodes below threshold are automatically routed to revision or escalated to Forge for pipeline restructuring.

9.3 Decentralized Convergence

Critically, Quorum does not require a supervisor agent. Convergence is a property that emerges from the interaction of agents with the intent graph — exactly as ant colony coordination emerges from pheromone interaction. This eliminates the single point of failure inherent in supervisor-based multi-agent architectures.

10.

The Sovereignty Stack vs. Existing Systems

Animus does not exist in isolation. Several adjacent projects address overlapping problems. Understanding how Animus relates to them clarifies what it is and what it is not.

System	What It Does	What It Misses
Claude / ChatGPT / Gemini	Frontier AI reasoning with optional memory features	Stateless by default. Context owned by vendor. Resets at platform discretion.
Ollama / LM Studio	Local model inference with no cloud dependency	No persistent memory. No multi-agent orchestration. Query-response only.
Urbit	Decentralized personal server with sovereign identity and P2P networking	Infrastructure sovereignty without cognition sovereignty. No AI reasoning layer.
Obsidian / Notion	Personal knowledge management with rich linking	Storage and retrieval only. No active reasoning. Cannot think about your data.
AutoGPT / CrewAI	Multi-agent task execution frameworks	No persistent identity. No memory across sessions. Not designed for a single user.
Animus	Full-stack sovereign AI: persistent identity, layered memory, model-agnostic reasoning, multi-agent orchestration, cross-device presence	Currently in active development.

"Urbit solved the infrastructure layer. Animus solves the cognition layer. Sovereignty without intelligence is just a private server. Intelligence without sovereignty is just a rented service."

The combination of persistent identity, layered memory, model-agnostic reasoning, industrial-grade multi-agent orchestration, and cross-device presence does not exist in any currently shipping product. That is the gap Animus is designed to fill.

A.

Technical Appendix — Specifications

A.1 Memory Storage Stack

Store	Technology	Purpose
Vector DB	ChromaDB or Qdrant	Semantic search across episodic and semantic memory
Knowledge Graph	NetworkX + SQLite	Structured entity relationships and confidence scoring
Metadata Index	SQLite	Fast retrieval by timestamp, type, and tag
Raw Storage	Flat files (JSON/JSONL)	Maximum portability and long-term durability

A.2 Data Flow

```
User input (any device)
■
▼
Interface Layer ■■ normalizes modality
■
▼
Core Layer ■■■■■■■ authenticates, applies preferences
■
▼
Memory Layer ■■■■■ retrieves relevant context
■
▼
Cognitive Layer ■■ routes to local or API model
■ ■
■ (complex tasks)
■ ▼
■ Forge ■■ decomposes → agents → quality gates
■ ■
■ Quorum ■ intent graph → stability → convergence
■ ■
■■■■■■■■■■■■■■■
■
```

```
▼
Memory Layer ■■■■■ stores context, updates patterns
■
▼
Interface Layer ■■ delivers response
```

A.3 Forge Pipeline Schema

```
pipeline:
name: research_and_synthesize
agents:
- role: researcher
model: llama3.1:8b
token_budget: 4096
quality_gate: relevance_score > 0.8
- role: synthesizer
model: deepseek-r1:14b
token_budget: 8192
quality_gate: coherence_score > 0.85
- role: critic
model: qwen2.5:7b
token_budget: 2048
quality_gate: approval == true
checkpoint: sqlite://animus_forge.db
resume_on_failure: true
```

B.

Reference Hardware Deployment

Animus is designed to run on consumer hardware without cloud dependency. The reference deployment uses two Apple Silicon workstations connected via Thunderbolt 5, with the exo framework providing distributed inference across both machines.

Machine	Role	What It Runs
Mac Studio M4 Max 128GB	Primary workstation	Animus core, Forge orchestrator, 70B reasoning model
Mac Studio M4 Max 128GB	Inference server	Agent pool (8B–14B models), Forge worker processes

Combined unified memory of 256GB enables frontier-class models — including DeepSeek V3 671B at 4-bit quantization — to run fully local with zero API cost and zero data leaving the user's control. This is the reference configuration. Animus also runs on a single laptop with smaller models. The architecture scales down gracefully: fewer agents, smaller models, same sovereignty.

B.1 Minimum Viable Deployment

Component	Minimum Specification
Hardware	Any machine with 16GB RAM (Apple Silicon preferred for unified memory efficiency)
Primary model	Llama 3.1 8B or Qwen 2.5 7B via Ollama
Vector DB	ChromaDB (local, no external dependency)
Forge agents	2–3 specialized agents vs. 8+ in reference deployment
Storage	Local filesystem, ~10GB for initial memory stores

The minimum viable deployment sacrifices throughput and model capability but preserves the full architectural properties: cryptographic identity, persistent memory, multi-agent orchestration, and complete local sovereignty.

Animus is open architecture. Implementations, forks, and extensions are encouraged. The goal is not a product. It is a paradigm.