“JUBO” Project Requirements v1

1. Project Vision

1. Goal Statement

To build the first mass-market "AI Operating System" that runs entirely on the user's hardware. We aim to replace the "Cloud Rental" model (paying OpenAI for temporary intelligence) with an "Asset Ownership" model.
We are not just building a chatbot; we are building a permanent digital asset: a personalized "Vault" of context and intelligence that belongs to the user, resides on their device, and becomes more valuable (and harder to replicate) with every interaction.

2. Long-Term Vision

We envision a future where Intelligence is a utility, not a service.
In 5 years, we will be the standard interface for the digital world, a decentralized mesh of millions of devices forming the world's largest supercomputer. Users will rely on us not just because it is private, but because it holds a Fiduciary Duty to them. Unlike Cloud AI, which serves advertisers or shareholders, we serve only the user, creating a relationship of "Absolute Loyalty" that no centralized competitor can mathematically match.

3. The Convergence Thesis

Our product roadmap is predicated on the "Intersection Point" of two accelerating curves:

- Hardware Velocity: Mobile NPUs (Neural Engines) are doubling in TFLOPS performance every 18 months (Apple M-Series/A-Series, Qualcomm Snapdragon Elite).
- Algorithmic Compression: Research into Model Quantization, Distillation, and Sparse MoE (Mixture of Experts) is drastically reducing the parameter count required for "Human-Level Reasoning."

The Bet: While running a high-fidelity agent on a phone is a "stretch" today, we are building for the hardware of 2027. We expect a crossover moment within 18 months where sub-3B parameter models running on consumer NPUs achieve parity with GPT-3.5. By building the infrastructure now, we will be positioned to capture the market the moment the hardware catches up to our software vision.

2. Milestone Strategy

Phase 1: The Prototype (Proof of Concept)

Scope: iOS only. Runs on CPU. Text-based interaction.
Strategic Goal: Technical Feasibility. Prove that the core "No Data Center" thesis is valid, that a 3B parameter model can run on consumer hardware with acceptable latency, validating our unit economics.
Success Metrics: Latency < 1.5s; App binary < 200MB.

Phase 2: The Beta (Early Access / "The Butler")

Scope: Apple Ecosystem (GPU/Metal). Introduction of The Vault (RAG) and Vision Input.
Strategic Goal: Retention & Stickiness. This phase is about proving "Data Gravity." We need to show that users prefer a "dumb" local model that knows their secrets over a "smart" cloud model that knows nothing.
Success Metrics: Day-30 retention > 20%; "Context Density" (average facts stored per user) increasing week-over-week.

<u>Phase 3: v1 (Public Launch / "The Ecosystem")</u>

Scope: Expansion to Flagship Android (NPU/ExecuTorch). Launch of The Council (Marketplace).
Strategic Goal: Scale & Moat Construction. We prove the revenue model (arbitrage) and solidify the "Router Moat" using aggregate usage data to train a proprietary classifier that routes queries more efficiently than any copycat competitor.
Success Metrics: Free-to-Paid conversion > 3%; Battery drain < 5%/hour.

3. Model strategy

<u>1. The Decoupled Vault (Memory Persistence)</u>

Concept: The "Vault" (Knowledge Graph & Vector DB) is stored in a universal, model-agnostic format (standardized JSON/SQL schemas and open-standard embeddings like nomic-embed-text).
Benefit: This ensures that Memory is portable. If a user swaps their underlying model from Llama-3 to Phi-4, or if we pivot our default provider, the new model simply "mounts" the existing Vault and continues the conversation seamlessly. The "Brain" changes, but the "Memories" remain.

<u>2. The "Adapter" Approach (Fine-Tuning without Lock-in)</u>

Challenge: How do we optimize a new model (e.g., Mistral) to use our specific tools without expensive retraining?
Strategy: We utilize LoRA (Low-Rank Adaptation) Adapters or System Prompt Abstractions.
Instead of retraining the massive base model, we train lightweight "Adapter Layers" (small ~50MB files) that sit on top of the base model to teach it our specific JSON tool formats.
User Choice: If a user selects a custom model (e.g., "I want to use Google Gemma 2B"), we inject a standardized "System Super-Prompt" that aligns that generic model to our Vault's API, ensuring compatibility without requiring a dedicated fine-tune.

4. Feature & Technical Requirements

<u>Part 1: User-Facing Features</u>

Feature	POC	Beta	v1	Example Query / Use Case	Recommended Libraries / Links
Chat Interface	Single chat session. Clears on close	**Chat History** Saved sessions. Rename/Delete chats	**Multi-Chat Threads** with folder organization & search	Users can navigate to that conversation about ancient Greeks	Exyte/Chat (UI)
Memory (The Vault)	Session context only (Last 10> messages)	Can recall facts from current active chat logs	**The Vault:** Cross-session memory + storage/recall of user preferences	“I need you to help me write a followup email for that job I applied to yesterday”	GRDB.swift (Local SQLite)
Web Access	N/A (Offline only)	**Active Search:** Can trigger a search query if explicitly asked	“Intelligent” routing, model knows when to use own model and when to query external sources	"Did the Spurs win last night?"	Tavily API Brave API
Voice Mode	N/A (Chat only)	User can register a **Voice Note,** answer is text	**Real-time Conversation:** Voice-in / Voice-out (TTS) with interruption	"Read me this summary while I drive."	ASR : whisper.cpp TTS: SesameCSM
Camera / Vision	N/A	N/A	**Image Upload:** Send a photo, Take a picture with the camera	"Take a picture of this menu and translate it."	Apple Vision Framework
File Upload	Text copy-paste only.	N/A	**PDF/Doc Parsing:** Upload single file to chat context	"Summarize this PDF contract."	SwiftSoup (Parsing)
Phone APIs	N/A	**Calendar Read:** Access to see upcoming events	**Action Agent:** Email drafting, Calendar writing, Reminder setting	"Am I free next Tuesday at 2pm?"	EventKit

One-Shot Mode	N/A	**Incognito Toggle:** Chat without saving to DB	**Ephemeral VMs:** Sandboxed execution for risky queries	"How do I fix a virus? (Don't save this)."	N/A (Logic implementation)

Part 2: Technical Requirements

Requirement	Prototype (POC)	Beta (Early Access)	v1 (Public Launch)	Technical Implementation Note	Recommended Libraries / Links
On-Device Model	Qwen (~1.7B): Run on CPU. Focus on low memory footprint (~1.5GB RAM).	**Fine-tuned 3B:** Switch to a 3B model optimized for Metal (GPU).	**Model changes via updates** (OTA): We control the specific model version (e.g., Llama 3.2 + LoRA).	Strategy: Use LoRA Adapters to standardize tool usage. When we update the base model, we just push a new adapter, keeping the "Vault" compatible.	MLX Swift or Executorch
Compute Runtime (Cross-Platform)	Apple: Run the model on iOS/macOS using the CPU. Focus on logic validation, not battery efficiency.	**Apple GPU (Speed):** To increase inference speed and reduce battery usage	**Flagship Android Expansion + NPU (Efficiency):** 1. Port to Android (Vulkan). 2. Optimize Apple to CoreML (NPU) Goal: Cross-platform scale & efficiency.	Why v1 needs NPU: CPU/GPU inference drains battery (1% per minute). NPU inference is ~5x more efficient and allows background usage without overheating	Executorch: Meta's edge runtime that compiles specifically for CoreML and Android NNAPI/Qualcomm AI Stack.
Agentic Arbitration	N/A	**Heuristic Router:** If keyword "Search" found → Use Tool	**Semantic Classifier:** Small model decides: Local vs. Cloud vs. Tool	The "Brain" that decides cost vs. privacy	SwiftAI
Web Agent (Scraper)	N/A	**Headerless Fetch:** Get text from top 3 URLs	**Smart Scraper:** Parse JS-heavy sites, bypass paywalls (if auth'd)	Needs to strip ads/garbage before feeding to LLM	SwiftSoup

Context Engineering	Basic System Prompt ("You are helpful")	Vector Memory: Chunking chat history into a vector DB and retrieving top-k matches.	Knowledge Graph: A dynamic graph that extracts entities (People, Projects) and relationships to build a user-specific worldview.	Implement architecture: The model decides when to write to memory and when to read.	Core Logic: Letta The open-source standard for stateful agents Graph Storage: Mem0 - Specifically built for "Personal AI" memory; it automatically updates user profiles and facts.
Cloud Delegation	Hardcoded API call to OpenAI	**Privacy Relay:** Proxy server strips IP/User ID	**The Council Market:** API arbitrage (Route to cheapest provider)	The business model layer	Custom Relay (Node/Go)
Multi-Modal Support	Text only	Visual Analysis (Input): The model can "see." User sends an image, model generates a description or answers questions about it using a lightweight Vision-Language Model (VLM)	Image Generation (Output): The model can "create." User prompts for an image, the Orchestrator delegates to a quantized Stable Diffusion model to render it on-device.	Beta: Use Moondream2 (tiny 1.6B VLM) or LLaVA (4-bit). It projects image embeddings into the text model's token space. v1: Use SDXL Turbo. Orchestrator rewrites user prompt for visual clarity before sending to the SD pipeline.	Vision (Beta): Moondream (i) Generation (v1): Swift CoreML Diffusers
Sync & Backup	N/A	N/A	**P2P Sync:** Zero-knowledge sync between Mac and iPhone	CRDTs (Conflict-free Replicated Data Types)	Automerge
News Agent	N/A	**Scheduled Fetch:** Runs once a morning	**Proactive Push:** "Breaking news relevant to your portfolio."	Background task scheduling.	BackgroundTasks

Part 3: Notes

Note on the "Router" (Arbitration)
For the Prototype, simple Logic: If the prompt is < 10 words, run Local. If prompt contains "Search" or "Find", use Web Tool.
For Beta: This is where you implement a library to create a loop: User Prompt → Local Model Thoughts → Call Tool → Get Result → Final Answer.

Strategic Note on the "Vault" (Memory)
Prototype: We don't build a database. Just keep an array of strings in memory.
Beta: we'll Vector Search. When a user uploads a PDF, chunk it, create embeddings (using a local embedding model like nomic-embed-text), and store it in SQLite.
Library: GRDB.swift is a standard for local SQLite on iOS; it supports custom FTS5 (Full Text Search) which is often better/faster than vector search for simple queries.

5. Additional requirements

Operational requirements

Requirement	Description	Technical Strategy
Safety & Moderation	**Critical for App Store Approval.** Since we cannot filter server-side, we must filter on-device. The model must refuse to generate illegal content (CSAM, self-harm).	Implement Guardrails, ex. Swift-sensitive-content-analysis (Apple). UI must include an anonymized "Report Issue."
"Bring Your Own Key" (BYOK)	**Margin Protection**. Power users will burn through your API margins. We must allow them to input their own OpenAI/Anthropic keys. This shifts the cost to them and unlocks "Uncapped" usage.	Use iOS Keychain / Secure Enclave to encrypt user API keys. The "Router" logic must check: If User Key exists -> Direct Route. If not -> Use our own
OTA Model Updates	**Model Agnostic.** Ability to swap the underlying model (e.g., Llama 3 -> Llama 4) without a full App Store update.	Build a **Model Manifest System**. App checks server for latest model hash at launch and downloads delta-patches (to save bandwidth). Maybe offer a screen with different model options for user to pick then download?

Smart Onboarding	**The Big App Problem.** Users will not wait 10 minutes for a model download at a blank screen. **"Empty Vault" problem**. A "dumb" model feels useless on Day 1. We need to seed the memory without aggressive permission prompts (Contacts/Calendar) that scare users away.	**Tiered Loading:** 1. Ship app with "Tiny" model (immediate use). 2. Download "Pro" model in background. 3. "Calibration Mode": On first launch, the model initiates a 60-second interview. It asks 3 key questions (Who are you? What are you working on? How do you like answers formatted?). Goals: 1. Populate the “User_Profile” entity in the Vault immediately so the very first "real" query feels personalized, 2. Start the conversation right away to prevent users from wondering what to do with the app, 3. Keep users busy while the big model is getting downloaded in the background
Privacy Telemetry	**The Debugging Loop.** We need to know if the model is slow or crashing without reading user chats.	**Differential Privacy:** Collect anonymized metrics (Latency, Token Speed, Crash Rate, Thumbs Up/Down ratio). Zero text logging.
Power State Policy (Battery Protection)	**Critical UX/Hardware constraint**. Vectorizing PDFs and updating the Knowledge Graph is computationally heavy. Doing this on battery will cause the phone to overheat and drain rapidly.	**Queue System**: Lightweight inference runs immediately. Heavy tasks (indexing docs >3 pages, Graph reorganization) are queued for Background Execution. Logic: If (Battery < 20% OR Unplugged) -> Queue. Else -> Process. UI: Show "Processing in background..." for heavy files.

Cloud infrastructure requirements

Requirement	Beta (Simple Proxy)	v1	Long Term / Future	Technical Strategy
The Council (Relay & Router)	Search Proxy: A simple "Pass-through" server. It receives a search query, strips the IP, calls the Tavily/Brave API, and returns the JSON. No model routing yet.	Arbitrage Engine: 1. PII Scrubber: Redacts emails/names before sending to external APIs. 2. Cost Router: Routes prompts to the cheapest provider that meets the quality threshold (e.g., Haiku vs. GPT-4).	Unified Intelligence: Integrate with Web Agents (Perplexity-style deep search) and Decentralized Compute (routing tasks to other users' idle devices).	Stack: Go or Rust (High concurrency, low latency). Architecture: Stateless microservice. It must not store logs. API: Exposes a single standard endpoint to the phone (POST /ask), normalizing the chaotic output of different providers into one standard JSON format.
Secure Sync & Backup	N/A (Client-Side Only): Users rely on	Encrypted Blob Storage: A "dumb" storage relay. 1. Zero-Knowledge:	Mesh Signal Server: A real-time WebSocket	Architecture: "Host-Blind" Storage. Protocol: The server acts as a CRDT Relay

(The Blind Vault)	iCloud/Google Drive for manual backups. We host nothing.	We store the data blobs, but the decryption keys never leave the user's device. 2. Recovery: Implementation of a "Recovery Key" flow for lost phones.	relay that facilitates instant P2P syncing between devices (Phone <-> Laptop) without storing data permanently.	(using automerge-repo-sync-server). It merges updates from different devices but treats the actual content as opaque, encrypted binary blobs.
Model Registry & CDN (OTA)	Static Hosting: Host model config/weights on a standard bucket (S3/Hugging Face). App checks a static JSON URL for updates.	Differential Delivery: Custom CDN that serves Binary Diffs. If we update the 3B model, users only download the 200MB changed bytes, not the full 2GB. Includes Manifest Server to force critical updates.	P2P Model Sharing: Users on local WiFi can share model weights with each other to save bandwidth (like AirDrop for AI models).	Infrastructure: CloudFront / R2 (High bandwidth, low latency). Logic: bsdiff for generating delta patches between model versions to minimize user data usage.
Entitlements & Auth (The Bouncer)	N/A (Free): No gatekeeping.	Receipt Validator: Verifies Apple App Store / Google Play subscriptions. Issues a Short-Lived JWT (Token) that the app uses to authenticate with The Council.	Enterprise SSO: Support for corporate logins (Okta/AD) for B2B teams/Family plans.	Privacy Note: This system knows who paid (Apple ID), but it is decoupled from what they ask. The "Council" validates the Token but does not log the User ID against the Prompt.

6. The Distillery (Training & Fine-Tuning Pipeline)

Note: Below suggestion was generated from ML models as a reference, most sophisticated requirements will need to be written by a ML scientist.

Component	Description	Technical Strategy
Synthetic Data Generator	The Raw Material. We need 10,000+ examples of perfect interactions (e.g., User asks for a file -> Model outputs correct JSON). We cannot write these by hand.	Teacher-Student Loop: Use GPT-4o ("The Teacher") to generate synthetic conversation logs that strictly follow our Vault's JSON API schema. Output: A standardized training_data.jsonl file.
The LoRA Factory	The Machine. The script that takes a Base Model + Synthetic Data and outputs an Adapter.	Auto-Train Script: A Python pipeline (using Unsloth or Axolotl libraries) that runs on a rented Cloud GPU. Input: Base Model (e.g., Llama-3.2). Process: Train LoRA for 3 epochs on the synthetic data. Output: adapter.safetensors (50MB).
Automated Evaluation	The Quality Gate. A suite of "Unit Tests for Intelligence." We must verify that the	LLM-as-a-Judge: We run 500 test prompts through the new local model. We send the results to GPT-4 to grade them Pass/Fail. Metrics: 1. JSON Syntax Rate: Must be 100%

("The Gauntlet")	training didn't break the model's basic reasoning or tool usage capabilities.	compliant. 2. Refusal Accuracy: Did it correctly identify questions it can't answer? 3. Fact Retention: Did it remember the system prompt instructions?
Quantization Engine	The Compressor. Converting the heavy training weights (Float16) into a format the iPhone NPU can read (Int4) without losing intelligence.	ExecuTorch Export: Post-training, the model + adapter is fused and quantized to 4-bit. Final Check: Run "The Gauntlet" again on the quantized version to ensure compression didn't degrade performance (Perplexity Check).

7. Risk Assessment (& Mitigations)

1. The "Sherlock" Risk (Platform Risk)

Threat: Apple (Apple Intelligence) or Google (Gemini Nano) integrates basic on-device AI directly into the OS.
Mitigation: Cross-Platform Neutrality. Apple will never sync your context to Windows or Android. We win by being the "Switzerland" of AI, the only layer that works everywhere and connects to every model provider without ecosystem lock-in.

2. The "Commoditization" Risk (Copycat Risk)

Threat: Since the core technology (LLMs + Open Source Libraries) is accessible, a well-funded competitor copies our stack, creates a "Wrapper," and outspends us on marketing.
Mitigation: The "Data Gravity" Moat.
The Logic: Code is easy to copy; Context is not.
The Strategy: We prioritize the "Vault" (Knowledge Graph). Once we knows a user's relationships, project history, and preferences, that data exists only in our local graph. Switching to a competitor would mean "lobotomizing" their assistant. The longer a user stays, the higher the switching cost becomes (The "Evernote Effect").
The Router Asset: We use our data to build a proprietary "Router Model" that optimizes cost/quality better than any new entrant. A copycat will inefficiently burn cash on cloud costs, while our router maximizes free local compute.

3. The "Dumb Model" Risk (Quality Risk)

Threat: Cloud models (GPT-5/6) become so advanced that local models feel useless by comparison.
Mitigation: The Council + RAG. We do not fight GPT-5; we use it. If the local model struggles, we seamlessly route to the cloud. Furthermore, a "dumb" model with perfect access to your local data is functionally smarter than a genius model with zero context.

4. The "Battery" Risk (UX Risk)

Threat: Local inference drains battery, causing churn.
Mitigation: NPU Optimization (Bare Metal). We aggressively migrate from GPU (Beta) to NPU (v1) using ExecuTorch. This moves the workload to the most efficient chip on the phone, enabling "always-on" utility.

5. The "Incentive" Risk (Trust Risk)

Threat: Users are skeptical of another AI company promising privacy but eventually selling out.

Mitigation: Business Model Alignment. A VC-backed competitor needing 100x returns must eventually monetize data. We price like a utility (SaaS/Arbitrage). Our incentives are aligned with the user (Privacy + Efficiency). In this market, Trust is the product, and our architecture proves we cannot sell user data even if we wanted to.

6. The "Black Box" Risk (Optimization Risk)

Threat: Our "Zero-Knowledge" architecture creates a massive blind spot. Unlike OpenAI, we cannot read user logs to debug hallucinations, fix edge cases, or see why users are churning. We risk flying blind while competitors use user data to rapidly iterate.
Mitigation: Privacy-Preserving Telemetry.
The Strategy: We track metadata, not content. We implement Differential Privacy to aggregate signals like "Thumbs Down Ratio," "Latency," and "Crash Rate" without ever logging the prompt text.
The Signal: We use the Router as a Canary. If a specific user segment frequently rejects local answers and forces a "Council" (Cloud) route, that metadata signals exactly where the local model is weak (e.g., "Users route 80% of coding questions to Cloud"). This allows us to improve the model without reading the user's code.

8. Business Model & Tier Definition

We operate on a "SaaS Margins" model. The Free tier must have near-zero marginal cost for us. We monetize "Convenience" (Sync/Search) and "Brokerage" (Cloud Access), not the core utility of the AI itself.

Tier features table:

Feature	Tier 1: Local (Free)	Tier 2: Plus (Subscription)	Tier 3: Power (v2 / Future)
Core Intelligence	Unlimited Local. Uses the standard efficient model Fully offline.	Unlimited Local.	Model Selection. Ability to download larger/uncensored models (8B+) for high-end devices.
The Vault (Memory)	Device Only. Memories live and die on this specific phone. No backup.	Encrypted Cloud Backup. If you lose your phone, you restore your Vault from our secure relay.	Vault Explorer. Visual graph explorer; export Vault to JSON/Obsidian.
Connectivity	Offline.	Zero-Knowledge Sync. Real-time sync between iPhone, iPad, and (later) Mac/Android.	Same.
Cloud Access	BYOK (Bring Your Own Key). User inputs their own OpenAI/Anthropic API key. We route requests directly; user pays OpenAI.	Managed Brokerage. User pays us a flat fee. We handle the routing, API billing, and	Priority Routing. Access to "Reasoning" models (o1/Claude Opus) with higher caps.

		anonymization. Includes "Fair Use" monthly cap.	
Web Agents	None. (Can only read user-pasted URLs).	Live Web Search. "Did the Spurs win?" (We pay the Tavily/Brave API cost).	Deep Research. "Read these 10 websites and summarize."
News/Push	None.	Proactive Agents. "Wake up briefing" based on live news + user calendar.	Custom Agents. User scripts their own recurring tasks.

The "Fair Use" Algorithm (Margin Protection)

For the Plus subscription, we cannot offer "Unlimited GPT-4" or we will go bankrupt. The Router plays a financial role here:

- Tier 1 Check: Can the Local 3B model answer this? (Cost: $0)
- Tier 2 Check: Can a cheap cloud model (e.g., Haiku/Flash) answer this? (~Cost: $0.0001).
- Tier 3 Check: Does it require GPT-4? (~Cost: $0.03).

Mitigation: If a Plus user hits a "Soft Cap" (e.g., ~$5 of API cost), we degrade their Cloud Routing to cheaper models (Haiku/Flash) for the rest of the month, or prompt them to buy a "Top-Up Pack."

The BYOK Strategy (Free Tier)

Allowing "Bring Your Own Key" on the Free Tier is strategic, it kills the "Wrapper" accusation: "We aren't just reselling OpenAI; you can use your own key!" It also creates "Data Gravity": Users start building their Vault on the Free tier. When they want Sync or Web Search (which they can't do with just an API key), they upgrade to Plus.

Future "Flexible Billing" (Token Packs)

For users who need heavy cloud usage (e.g., generating 100 images or translating a book), we can sell "Compute Credits" (One-time purchases) on top of the Subscription. This avoids complicating the base monthly price.

Appendix. Glossary of Terms

Core Concepts

- **Asset Ownership Model:** The philosophy that AI should be a permanent digital asset owned by the user (like a house), rather than a temporary service rented from a corporation (like a hotel room).
- **No Data Center Thesis:** The economic premise that shifting compute costs from centralized servers to edge devices (User NPUs) enables a business model with near-zero marginal costs and infinite scalability.
- **The Intersection Point:** The projected moment (estimated 18 months out) where consumer hardware power intersects with model efficiency, allowing "GPT-3.5 level" intelligence to run locally on a standard smartphone.

System Components

- **The Vault:** The user's long-term memory. A decoupled storage system combining a **Vector Database** (for fuzzy search of history) and a **Knowledge Graph** (for structured facts about people/projects). It is model-agnostic, meaning the "memories" survive even if the underlying AI model is swapped.
- **The Orchestrator (Router):** The logic layer (or "Classifier Model") that sits between the user and the AI. It analyzes every prompt to decide if it should be answered Locally (Free/Fast) or routed to The Council (Paid/Smart). It is the primary mechanism for cost control.
- **The Council:** The marketplace of external, third-party cloud models (OpenAI, Anthropic, etc.). Used only when the local model is insufficient. We act as a broker/arbitrageur, selecting the best model for the specific task.
- **Privacy Relay:** An intermediate server owned by us that strips the user's IP address and identifiable metadata before forwarding a prompt to The Council. This ensures third-party providers (like OpenAI) cannot build a profile on our users.
- **LoRA Adapter (Low-Rank Adaptation):** A lightweight "plugin" (~50MB) applied to a generic base model. It forces the generic model to behave according to our specific needs (e.g., formatting JSON correctly for the Vault) without requiring a massive, expensive retraining of the whole model.

Hardware & Operations

- **NPU (Neural Processing Unit):** Specialized hardware on modern phones (Apple Neural Engine / Android TPU) designed solely for AI math. Utilizing this (via **Bare Metal** acceleration) is the only way to achieve "Always-on" functionality without draining the battery.
- **ExecuTorch:** The runtime library (by Meta/PyTorch) used to compile our AI models so they run efficiently on mobile NPUs.
- **BYOK (Bring Your Own Key):** A feature allowing users to input their own personal API keys for OpenAI/Anthropic. This shifts the compute cost to the user and allows for "Uncapped" usage of cloud models.
- **Data Gravity:** Our primary defensive moat. The concept that as the user adds more data to The Vault (files, history, relationships), the utility of the app increases, making the "switching cost" of moving to a competitor prohibitively high.