SMART ABES
Hackathon 2.0

Team Erastus

Presented By:

Samridhi Khanna & Parjanya Kumar Arya

/Parjanya-Kumar-Arya-github/SAH-2.0-Kinetic

Team ID: 2197f28d
Track: WEB-3
PS ID: PS-WEB3-02

KINETICS

Team ID: 2197f28d
Track: WEB-3
PS ID: PS-WEB3-02

Introduction

Overview of the Pitch

Addressing the Consumer Apps Problem Statement: Current consumer AI apps trap your personalised data inside centralised walled gardens. **Kinetic** revolutionises this by decoupling AI intelligence (Compute) from its memory (State). We utilise Solana Token-2022 extensions and IPFS to anchor encrypted AI memory to user-owned NFTs. This pitch outlines our hybrid architecture, the O(1) isolated vector retrieval system, our cryptographic "Lobotomy" mechanism for secure ownership transfer, and our roadmap to a fully sovereign AI ecosystem.

KINETICS

/Parjanya-Kumar-Arya-github/SAH-2.0-Kinetic

Team ID: 2197f28d
Track: WEB-3
PS ID: PS-WEB3-02

The Problem Gap

Current AI architectures treat user data as a platform commodity rather than a user-owned asset. This creates a systemic lock-in where high-value, personalised context cannot be securely or mathematically transferred across different applications.
.

/Parjanya-Kumar-Arya-github/SAH-2.0-Kinetic

Walled Gardens & State Lock-in

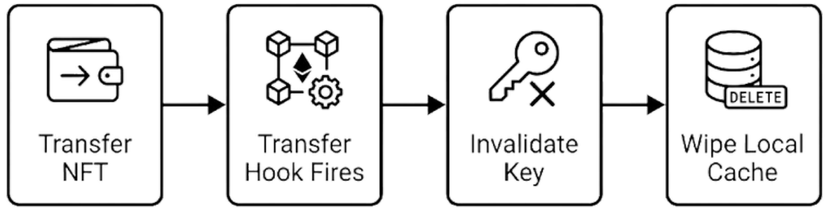
Users invest hundreds of hours training personalised AI agents, but the memory state is trapped on centralised servers. If a platform shuts down or alters its API, the user's trained context is destroyed with zero portability.

The Cost vs. Sovereignty Trade-off

Storing raw AI memory logs directly on a blockchain is prohibitively expensive and introduces unacceptable latency for real-time inference. Conversely, standard Web2 vector databases offer speed but strip away cryptographic ownership.

The "Double-Spend" of Knowledge

Safely decoupling compute (LLM) from memory (State) is a hard engineering problem. It requires a hybrid architecture to maintain chat-speed latency (<400ms) while cryptographically ensuring that transferring an AI instantly revokes the previous owner's access.



Team ID: 2197f28d
Track: WEB-3
PS ID: PS-WEB3-02

Novelty & Innovation

The 'Un-Obvious' Engineering Improvement: We built a system using "Sovereign State Injection." Rather than building an AI that holds a database of memories, our AI agent is completely stateless and "lobotomized" by default. It only gains intelligence and context at runtime when a user cryptographically signs a session, injecting their locally decrypted vector context into the LLM.

/Parjanya-Kumar-Arya-github/SAH-2.0-Kinetic

BEYOND A GENERIC WRAPPER

A standard generic wrapper simply takes a user query, saves it in a centralized PostgreSQL/Pinecone database, and sends it to OpenAI. The developer has full access to the user's data. Kinetic replaces the centralized database with an encrypted IPFS vector stream linked to a Solana NFT. The backend literally cannot read the user's data. Memory is decrypted entirely client-side using an AES-256 key derived from the user's Ed25519 wallet signature before hydrating the LangChain agent.

UNIQUE ENGINEERING INSIGHT

The Cryptographic "Lobotomy". How do you solve the "double-spend" problem for knowledge? If a user sells their trained AI, how do you ensure they don't keep a copy of the memories? Our unique insight leverages Solana Token-2022 Transfer Hooks. We programmed the token so that the moment the NFT is transferred to a new wallet, the smart contract triggers a client-side invalidation of the previous owner's decryption keys. Intelligence is truly moved, not just copied.

Team ID: 2197f28d
Track: WEB-3
PS ID: PS-WEB3-02

Optimization & Depth

Scaling personalized AI memory for millions of users introduces massive latency and cost bottlenecks. Brute-force searching a global vector database is inefficient. We bypassed this by engineering a system built on cryptographic partitioning and distributed state trees, ensuring that our search and storage complexities remain constant, no matter how large the platform grows.

/Parjanya-Kumar-Arya-github/SAH-2.0-Kinetic

Core Algorithms (DSA) Used

Hierarchical Navigable Small World (HNSW) Graphs: Used for O(log N) Approximate Nearest Neighbour (ANN) search across high-dimensional (1536d) vector embeddings during the RAG pipeline

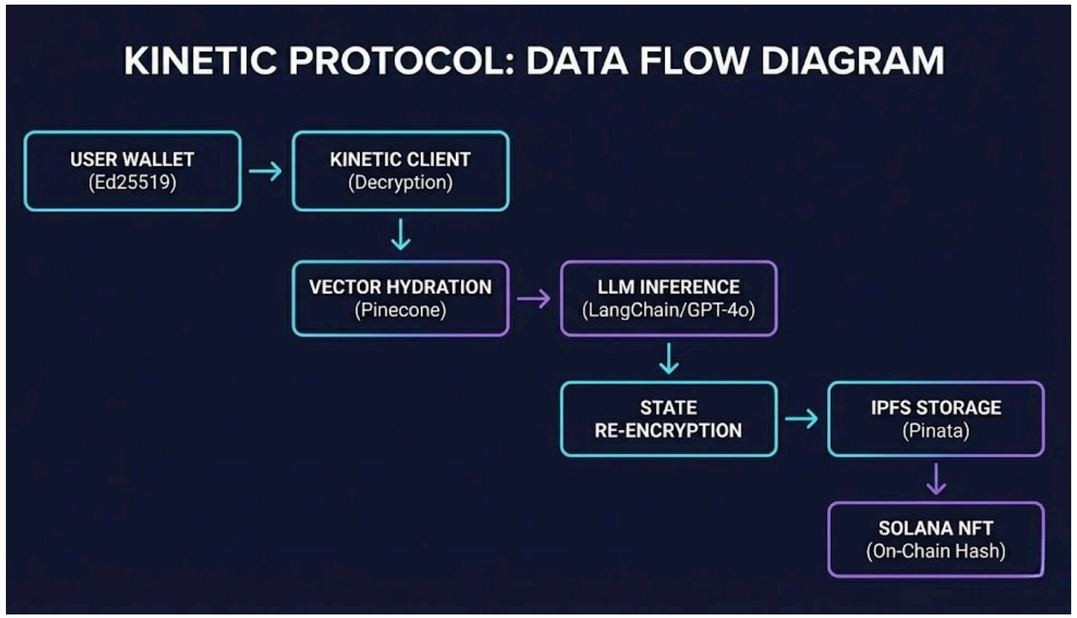
Merkle DAGs (Directed Acyclic Graphs): Used to serialise and store the AI's encrypted memory snapshots off-chain. Only the 32-byte Root Hash (CID) is committed to the Solana blockchain.

Complexity Analysis & Scaling Strategy

Time Complexity (O(1) Global Scaling): Standard HNSW performance degrades with multi-tenancy. We implemented Strict Namespace Isolation where $namespace = *NFT_Mint_Address*. This *O(1)* Hash Map lookup completely shields the user's vector search from global platform noise.
Space Complexity (Constant On-Chain Rent): Because we use Merkle DAGs, the on-chain storage requirement is strictly *O(1)*. Whether the AI has 10 memories or 10 million, the Solana transaction size and cost ($0.000005 SOL) remain the same.

Team ID: 2197f28d
Track: WEB-3
PS ID: PS-WEB3-02

Methodology



/Parjanya-Kumar-Arya-github/SAH-2.0-Kinetic

Auth & Decrypt (Wallet signature unlocks local AES-256 keys)

Vector Hydration (O(1) isolated RAG retrieval via Pinecone)

Stateless LLM (Injected context powers LangChain inference)

State Anchoring (Re-encrypt, pin to IPFS, commit to Solana)

The "Body & Soul" Hybrid Architecture:
Our methodology achieves Zero-Knowledge Web3 sovereignty without sacrificing Web2 chat latency. The AI agent is entirely stateless by default. When a user authenticates, their Ed25519 wallet signature derives a local AES decryption key. The client fetches the encrypted memory graph from IPFS, decrypts it locally, and temporarily "hydrates" the LLM. Post-inference, new conversation vectors are immediately re-encrypted, uploaded to Pinata, and the new state's Merkle Root hash is atomically updated on the Solana Token-2022 NFT contract. The backend never sees raw user data.

Team ID: 2197f28d
Track: WEB-3
PS ID: PS-WEB3-02

Articulation & Data Flow

Why is this architecture robust?
Our system fundamentally eliminates single points of failure by operating on a Zero-Knowledge framework combined with Atomic State Linking. The bridging server acting as the intermediary never sees raw user text—only encrypted vector blobs. By mathematically anchoring the AI memory's Merkle root to the Solana block height, we ensure the agent's state is cryptographically immutable, tamper-proof, and resilient against centralized server outages.

/Parjanya-Kumar-Arya-github/SAH-2.0-Kinetic

Input (Authentication)
The user connects their Web3 wallet (Ed25519). This cryptographic signature acts as the secure seed to locally derive AES-256 decryption keys, ensuring only the owner can unlock their AI's state.

Logic (Hydration & Bridge)
The Kinetic Client fetches the encrypted memory graph from IPFS. It decrypts the vectors entirely client-side, creating a temporary, isolated namespace to "hydrate" the LangChain LLM for high-speed, contextual inference.

Output (State Commit)
Post-inference, new conversation vectors are instantly re-encrypted on the device. The updated blob is pinned to Pinata, and the new root CID hash is atomically committed to the user's Solana NFT.

Feasibility & Roadmap

With our ideation phase and system architecture rigorously finalised on February 12th, we have completely de-risked the cryptographic logic of the protocol. We are now executing a highly parallelised, two-week development sprint to deliver a functional Web3/AI prototype by the February 27th deadline.

/Parjanya-Kumar-Arya-github/SAH-2.0-Kinetic

Current Status (Architecture Finalized)

The "Body & Soul" hybrid architecture is fully mapped. Smart contract schemas (Token-2022) and HNSW Vector DB isolation strategies are finalised. Local development environments and repos are initialized.

Execution Sprint (Feb 13 - Feb 27)

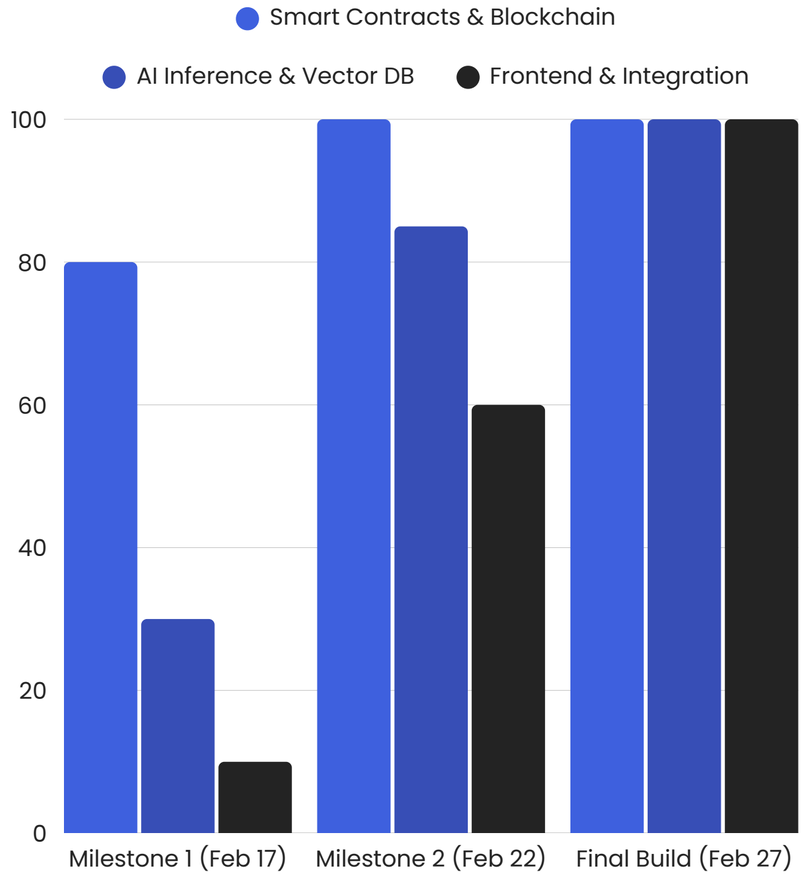
Feb 13-16: Deploy Anchor smart contracts to Solana Devnet. Feb 17-21: Build the LangChain + IPFS encrypted vector pipeline. Feb 22-26: Integrate Next.js UI, Wallet Auth, and "Lobotomy" Transfer Hooks.

Risk Mitigation

Aggressive 14-day development timeline. Mitigation: Strict modularity. We are parallelizing the Web3 contract development and the AI RAG pipeline, using mocked LLM responses during UI building to prevent deployment bottlenecks.

Team ID: 2197f28d
Track: WEB-3
PS ID: PS-WEB3-02

Projected Sprint Burn-Up (%)



Meet Our Team

We are Team Eractus. Bringing together deep expertise in decentralised infrastructure, full-stack engineering, and artificial intelligence, our mission is to fundamentally rewrite how users own and interact with their digital context. We don't just build wrappers; we engineer sovereign protocols.

/Parjanya-Kumar-Arya-github/SAH-2.0-Kinetic

Team ID: 2197f28d
Track: WEB-3
PS ID: PS-WEB3-02

Samridhi Khanna

- Web3 & Smart Contract Developer
- Frontend Developer & UI/UX

#samridhi18khanna@gmail.com

Parjanya Kumar Arya

- Lead System Architect & Full-Stack (React/NestJS)
- AI Integration & Vector Data Engineer

#parjanya789@gmail.com