MAI Network

Technical Architecture Document

Decentralized AI Inference Infrastructure on Solana using Proof of Useful Work (PoUW)

Version 1.0 | May 2026

miningmai.com

About This Document

This document describes the planned technical architecture for the MAI decentralized AI client — intended for our community, investors, and technical contributors. MAI is building a decentralized network where contributors share GPU/CPU power to process real AI inference requests and earn MAI tokens through Proof of Useful Work (PoUW). Every token earned represents real, useful computation.

This is not a final specification — it is a living technical roadmap that will evolve as development progresses. We commit to keeping the community updated as the architecture is refined and implemented.

What This Document Covers

Key Project Facts

Blockchain	Solana
Consensus	Proof of Useful Work (PoUW)
AI Engine	Python — Ollama / vLLM / HuggingFace
P2P Network	Rust + libp2p
Desktop Client	Tauri (Rust + JavaScript)

AI Models	Llama 4, Mistral, Qwen 2.5
Smart Contract	Pure Rust (Solana)
Storage	IPFS + Arweave
Verification	Zero-Knowledge Proofs (ZKP)
Dev Start	Q3 2026 (after presale)
Full Launch	Q3 2027 — Q1 2028

Table of Contents

MAI Network — Technical Architecture Document v1.0

This document will be updated as development progresses. Follow @miningmai_news on Telegram for the latest announcements.

Section 1

System Architecture Overview

The MAI network is a multi-layer decentralized system designed to route real AI inference requests from end users to miners who process them using open-source language models running on their own hardware. The architecture separates concerns into four distinct layers: the AI execution layer, the peer-to-peer communication layer, the verification and reward layer on Solana, and the user-facing desktop client. Each layer is independently developed and communicates with the others through well-defined interfaces.

Core Components

AI Inference Engine (Python)

Runs on each miner's machine. Loads and executes open-source models (Llama 4, Mistral, Qwen 2.5) via Ollama or vLLM. Receives task payloads from the P2P layer, processes them, and returns the result along with a cryptographic proof of execution.

P2P Network Layer (Rust + libp2p)

The communication backbone of the MAI network. Handles peer discovery via DHT (Distributed Hash Table), task broadcasting, result routing, and miner reputation tracking. Every node in the network is both a worker and a relay -- there is no central coordinator.

Solana PoUW Smart Contract (Rust)

The on-chain layer that governs the entire economy. After the presale, mint authority transfers from the presale contract to the new PoUW contract. This contract handles miner registration, on-chain task verification, MAI token minting as rewards, NFT bonus logic, and staking/DAO voting mechanisms.

Tauri Desktop Client (Rust + JavaScript)

The application miners install on their machines. Provides a graphical interface to configure resource allocation (Light / Balanced / Maximum / Flexible presets), monitor real-time earnings, view task history, manage the local AI model library, and connect the Solana wallet for reward claims.

IPFS + Arweave Storage

Decentralized storage used for AI model distribution and permanent logging. IPFS handles dynamic data such as task queues and cached model shards. Arweave provides permanent, immutable storage for AI model weights, historical task records, and audit logs.

Zero-Knowledge Proof (ZKP) Verification

Miners generate a cryptographic proof alongside each completed task -- a compact record proving that a real computation was performed on the correct input, without revealing the user's data. ZKP verification runs on-chain in the Solana contract, ensuring that rewards are only issued for genuine, verifiable work.

High-Level Data Flow

The following sequence describes how a single AI inference request travels through the MAI network from the moment a user submits it to the moment the miner receives a reward:

Step 1 -- Request Submission

A user submits an AI request (text, code, analysis) through the MAI platform interface. The request is encrypted and broadcast to the P2P network as a task payload.

Step 2 -- Task Assignment

The P2P layer evaluates available miners based on their registered hardware capabilities, current load, uptime score, and geographic proximity. The task is assigned to the most suitable miner node. If the primary miner fails to respond within a timeout window, the task is automatically reassigned to the next candidate.

Step 3 -- AI Execution

The assigned miner's Python AI engine loads the required model (or uses a cached version) and processes the inference request. The engine records the exact start time, end time, and compute units consumed during execution. This data is included in the result payload.

Step 4 -- Proof Generation

After execution, the miner's client generates a Zero-Knowledge Proof attesting that the computation was completed correctly. The proof, the result, and the execution metadata are signed with the miner's private key and submitted back to the P2P network.

Step 5 -- On-Chain Verification

The signed result and proof are submitted to the Solana PoUW smart contract. The contract verifies the proof, checks the miner's registration status, calculates the reward based on resources consumed and active bonuses (NFT tier, uptime), and mints the corresponding MAI tokens directly to the miner's registered wallet.

Step 6 -- Result Delivery

The processed AI response is delivered to the end user through the platform interface. The transaction is recorded permanently on-chain and in Arweave storage.

Note: All reward calculations are based exclusively on active task-processing time. A miner that is online but not processing tasks does not earn MAI. This design ensures that every token in circulation represents verified, useful computation.

Section 2

AI Inference Engine

The AI Inference Engine is the core computational component of each MAI miner node. It is written in Python and is responsible for downloading, managing, and executing open-source large language models on the miner's local hardware. The engine exposes a local API that the Rust P2P layer calls when a task is assigned to the node. The design prioritizes compatibility across a wide range of hardware — from consumer GPUs to CPU-only machines — ensuring that anyone can participate in the MAI network regardless of their hardware tier.

It is important to note that all AI models used by MAI are completely free and open-source. Anyone with technical knowledge can download and run Llama 4, Mistral, or Qwen locally using a single command via Ollama -- no API key, no subscription, no cost. This is the core competitive advantage of the MAI network: we do not build AI from scratch. Instead, we build the infrastructure that makes these powerful open models accessible to everyone -- routing requests from non-technical users to miners who run the models on their hardware, and rewarding miners fairly for doing so. This approach puts MAI in direct competition with paid AI services like OpenAI and Anthropic by offering equivalent open-model quality at a fraction of the cost, with full privacy, and without any centralized dependency.

Supported AI Models

MAI exclusively uses open-source, open-weight models that can be run locally without any API key or cloud dependency. The following model families are supported at launch:

Model Family	Developer	License	Best For
Llama 4 Scout / Maverick	Meta	Llama 4 Community	General chat, reasoning, long context
Mistral 7B / Small 4	Mistral AI	Apache 2.0	Efficiency, European data sovereignty
Qwen 2.5 (7B, 32B, 72B)	Alibaba	Apache 2.0	Multilingual, code, math

All models are served in quantized GGUF format (4-bit or 5-bit quantization) which reduces memory requirements by up to 75% while preserving 95-98% of full-precision quality. This makes it possible to run a 70B-parameter model on a consumer GPU with 40GB VRAM, or a 7B model on a machine with just 8GB RAM.

Inference Stack

The MAI AI engine supports three inference backends, selected automatically based on the miner's available hardware:

Ollama (primary for consumer hardware)

Ollama wraps llama.cpp with a clean REST API compatible with the OpenAI interface standard. It handles model downloading, GGUF quantization, and local serving automatically. Ollama is the default backend for miners with consumer GPUs (RTX 3080 and above) or CPU-only machines. It supports Llama 4, Mistral, and Qwen model families out of the box. The MAI client communicates with Ollama via HTTP on localhost, keeping the inference process isolated and secure.

vLLM (for high-throughput server nodes)

vLLM is a production-grade inference server optimized for maximum throughput using PagedAttention memory management. It is recommended for miners with data-center-grade GPUs (A100, H100) who want to

process multiple tasks in parallel. vLLM supports continuous batching, meaning it can serve several user requests simultaneously without waiting for each to complete before starting the next.Platform support: vLLM runs natively on Linux and macOS. On Windows, vLLM is supported via WSL2 (Windows Subsystem for Linux) with an NVIDIA GPU — the MAI client installer automates WSL2 setup. Windows miners without WSL2 capability automatically default to Ollama with no performance penalty for standard tasks.

HuggingFace Transformers (fallback and fine-tuning)

The HuggingFace Transformers library serves as the fallback backend for models not yet supported by Ollama or vLLM, and as the primary framework for any fine-tuning tasks that will be introduced in later network versions. It provides direct access to the full Hugging Face model hub and supports federated learning extensions.

Hardware Requirements

The MAI network is designed to be inclusive — miners at all hardware tiers can participate. Rewards scale proportionally with the compute power contributed:

Tier	Hardware Example	Recommended Models	Est. Throughput
Entry	CPU only / 8GB RAM	Qwen 2.5 3B, Mistral 7B (4-bit)	8-15 tokens/sec
Standard	RTX 3080 / 10GB VRAM	Llama 4 Scout, Mistral 7B	40-80 tokens/sec
High-End	RTX 4090 / 24GB VRAM	Qwen 2.5 32B, Llama 4 Maverick	80-150 tokens/sec
Server	A100 / H100 80GB	All models, full precision	200+ tokens/sec

Task Execution Pipeline

When a task arrives at the miner's AI engine from the P2P layer, it is processed through the following pipeline:

1. Task Deserialization

The task payload received from the Rust P2P module is deserialized and validated. The payload contains: the encrypted user prompt, the required model identifier, maximum token budget, task ID, and a timestamp. If validation fails, the task is rejected and the P2P layer is notified to reassign it.

2. Model Selection and Loading

The engine checks if the required model is already loaded in memory. If not, it loads the model from local storage. First-time downloads use HuggingFace Hub as the primary source for speed and reliability; IPFS/ Arweave serves as the decentralized verification and backup layer. Model loading time is not counted toward the task execution time for reward purposes..

3. Inference Execution

The prompt is passed to the active inference backend (Ollama, vLLM, or HuggingFace). The engine records the precise start timestamp, monitors GPU/CPU utilization during execution, and collects the number of tokens generated. All metrics are logged locally.

4. Execution Metadata Collection

Upon completion, the engine collects: total tokens generated, time elapsed in seconds, average GPU utilization percentage, peak RAM usage, and model identifier used. This metadata forms the basis for the PoUW reward calculation.

5. Result Packaging and Signing

The inference result and execution metadata are packaged into a signed result payload. The miner's Solana private key signs the payload, proving that this specific node performed the computation. The signed package is passed back to the Rust P2P module for proof generation and on-chain submission.

Privacy note: User prompts are encrypted end-to-end. The miner's AI engine receives and processes the decrypted prompt only within a sandboxed local process. Prompts are never stored on disk and are cleared from memory immediately after the result is returned. Zero-Knowledge Proofs ensure that verification of work does not require revealing the original input data to any third party.

Section 3

Proof of Useful Work (PoUW) Protocol

Proof of Useful Work (PoUW) is the consensus mechanism at the heart of the MAI network. Unlike traditional Proof of Work (Bitcoin mining) where computers solve meaningless mathematical puzzles to secure the blockchain, PoUW directs that same computational energy toward real, productive AI inference tasks. Every MAI token minted represents a verified AI computation that a real user requested and a real miner completed. There is no wasted energy in the MAI network -- every joule spent contributes directly to the decentralized AI economy.

Why PoUW Over Traditional PoW

Property	Traditional PoW (Bitcoin)	MAI PoUW
Purpose of compute	Solve arbitrary hash puzzles	Process real AI requests
Energy efficiency	Wasted on meaningless work	100% productive computation
Economic output	Block reward only	AI service + token reward
Scalability	Fixed difficulty target	Scales with AI demand
Verifiability	Hash output checked	ZKP proves correct AI execution
User benefit	None (miners vs users)	Users get AI, miners get MAI

Task Assignment

When a user submits an AI request, the PoUW protocol is responsible for finding the most suitable miner to process it. Task assignment is not random -- it follows a deterministic scoring algorithm that evaluates each available miner node:

Hardware Capability Score

Each miner registers their hardware profile on-chain at setup: GPU model, VRAM, CPU cores, RAM, and network speed. The protocol matches task complexity to available hardware -- a 70B model inference task will only be assigned to nodes with sufficient VRAM, while smaller tasks are distributed to entry-level miners to maximize network utilization.

Uptime and Reliability Score

The protocol tracks each miner's historical task completion rate, average response time, and number of failed or timed-out tasks. Miners with higher reliability scores are prioritized for assignment. A miner that consistently completes tasks on time accumulates a higher reputation score, leading to more frequent assignments and higher total earnings.

Current Load

The P2P layer monitors each node's real-time processing queue. Miners already executing tasks are deprioritized for new assignments until their queue clears. This prevents overloading individual nodes and ensures tasks are distributed evenly across the network.

Geographic Proximity

For latency-sensitive requests, the protocol prefers miners geographically closer to the requesting user. Proximity is approximated using IP geolocation data registered during miner onboarding. This ensures fast

response times for real-time AI interactions.

NFT Tier Bonus

Miners who hold Warrior NFTs receive a priority bonus in task assignment scoring. Platinum NFT holders (+20% mining bonus) and Gold NFT holders (+15%) are weighted higher in the assignment queue, rewarding early presale participants who demonstrated commitment to the network before it launched.

Work Verification

This is the most critical component of the PoUW protocol. Without reliable verification, malicious miners could submit fake results and claim rewards without doing real work. MAI uses a multi-layer verification approach:

Layer 1 -- Execution Metadata Validation

Every completed task must include a signed metadata record: task ID, model used, tokens generated, time elapsed, GPU utilization profile, and a hash of the output. The Solana contract checks that elapsed time and token count are consistent with the declared hardware tier. A miner claiming to generate 1,000 tokens in 0.1 seconds on a CPU-only machine will be flagged as fraudulent.

Layer 2 -- Zero-Knowledge Proof (ZKP)

The MAI network uses a multi-layer verification approach that evolves with available technology. Currently, execution metadata validation (Layer 1) combined with Spot-Check Sampling (Layer 3) provides robust anti-fraud protection — making fraud economically irrational regardless of ZKP availability. As zkML frameworks mature — particularly Lagrange DeepProve and EZKL which successfully proved GPT-2 inference in 2025 — MAI will integrate cryptographic proof verification for full on-chain verifiable compute. This roadmap ensures the network launches on schedule while adopting stronger verification as it becomes production-ready for large LLMs.

Layer 3 -- Spot-Check Sampling

Periodically, the protocol selects a random subset of completed tasks for re-execution by a second independent miner. If the outputs differ significantly (beyond acceptable model variance), both submissions are flagged for review and the original miner's reward is withheld. Repeated failures result in a temporary or permanent ban from the network. This sampling layer adds an additional deterrent against coordinated fraud.

Layer 4 -- Stake-Based Accountability

In later network versions, miners will be required to stake a minimum amount of MAI tokens to participate. If a miner is found to be submitting fraudulent results, a portion of their stake is slashed. This creates strong economic incentives for honest behavior -- cheating costs the miner more than it would earn.

Reward Calculation

Rewards are calculated and minted by the Solana PoUW smart contract immediately after a task is successfully verified. The base formula is:

Reward = (GPU_MHs x 5 + CPU_GHz x 2 + RAM_1000MHz x 0.2 + Storage_10GB x 0.1 + Net_Mbps x 0.1) x Active_Hours x NFT_Multiplier x Uptime_Bonus

Critically, Active_Hours counts only the hours during which the miner was actively processing tasks -- not simply connected to the network. A miner that is online for 24 hours but only processes tasks for 6 hours earns rewards for 6 hours only. This design prevents idle nodes from diluting rewards and ensures the token supply grows in direct proportion to real AI work performed.

Anti-Fraud Mechanisms Summary

Mechanism	Protects Against

ZKP verification	Fake computation results
Metadata consistency check	Implausible timing claims
Spot-check re-execution	Coordinated result manipulation
Stake slashing (v2)	Repeat fraudulent behavior
Active hours only reward	Idle nodes claiming rewards
Hardware registration	Overclaiming compute resources

The PoUW protocol is designed to be economically self-correcting: honest miners always earn more than it would cost to attempt fraud. As the network matures and stake-slashing is introduced, the cost of dishonest behavior increases further, making MAI one of the most fraud-resistant decentralized compute networks in the DePIN ecosystem.

Section 4

P2P Network Layer

The P2P (peer-to-peer) network layer is the communication backbone of the MAI network. It is built in Rust using the libp2p framework -- the same technology that powers IPFS, Ethereum 2.0, and Filecoin. There is no central server in the MAI network. Every miner node is simultaneously a worker, a router, and a relay. The network self-organizes, self-heals, and operates without any single point of failure.

Why Rust and libp2p

Rust was chosen for the P2P layer for three reasons: performance, memory safety, and ecosystem maturity. Rust programs run at near-C speed without a garbage collector, making them ideal for low-latency network operations. The language's ownership model eliminates entire classes of bugs (buffer overflows, race conditions, null pointer dereferences) at compile time -- critical for a financial network where bugs have real economic consequences. libp2p is the most battle-tested P2P framework available, with production deployments handling millions of nodes globally.

Peer Discovery

When a new miner installs the MAI desktop client and starts the node for the first time, it must discover other nodes in the network. MAI uses the Kademlia DHT (Distributed Hash Table) protocol from libp2p to handle peer discovery:

Bootstrap Nodes

The client connects to a small set of known bootstrap nodes operated by the MAI team during the early network phase. These nodes maintain a stable presence and help new miners join the network. Over time, as the network grows, bootstrap dependency decreases and the DHT becomes fully self-sustaining.

Kademlia DHT

Once connected, the node joins the Kademlia DHT -- a distributed routing table where each node stores information about a subset of other nodes. The DHT allows any node to find any other node in O(log n) steps, meaning a network of 1 million miners requires only ~20 routing hops to locate any specific peer. This scales infinitely without central coordination.

mDNS Local Discovery

On local networks (e.g., a data center with multiple MAI nodes), libp2p's mDNS protocol discovers peers automatically without going through the global DHT. This reduces latency for co-located nodes and improves task routing efficiency within the same geographic region.

Node Communication

All communication between MAI nodes uses encrypted channels. The network supports three types of message passing:

Direct Messaging (1-to-1)

Used for task assignment and result submission. When the network assigns a task to a specific miner, the task payload is sent directly to that node over an encrypted libp2p channel using the Noise protocol (a modern cryptographic handshake framework). The miner's response -- the signed result and ZKP -- travels back through the same direct channel.

GossipSub (1-to-many)

Used for network-wide announcements: new task availability, miner status updates, and protocol version broadcasts. GossipSub is a publish-subscribe protocol where nodes subscribe to topics and receive messages from all publishers on that topic. Messages propagate through the network in logarithmic time, ensuring all nodes stay synchronized without flooding the network with redundant traffic.

Request-Response Protocol

Used for miner capability queries and reputation checks. Before assigning a task, the network can query a target miner's current load, available VRAM, and uptime score via a lightweight request-response exchange. This ensures task assignment decisions are based on real-time hardware state, not stale cached data.

Network Resilience

The MAI P2P network is designed to remain operational under adverse conditions -- node failures, network partitions, and adversarial attacks:

Automatic Task Reassignment

If a miner fails to return a result within the configured timeout window (default: 60 seconds for standard tasks, 300 seconds for large model inference), the P2P layer automatically selects the next-best candidate from the assignment queue and reissues the task. The failed miner's reliability score is decremented accordingly.

Node Churn Tolerance

The Kademlia DHT is inherently tolerant of node churn -- nodes joining and leaving the network constantly. The routing table self-heals as nodes reconnect, and no task data is lost because tasks are not stored on individual nodes but routed dynamically through the network.

Sybil Attack Resistance

To prevent a single actor from controlling many nodes and manipulating task routing, miner registration requires an on-chain Solana transaction and hardware proof submission. In later versions, a minimum MAI stake will be required per node, making large-scale Sybil attacks economically prohibitive.

Network Partitioning

If a subset of nodes loses connectivity to the main network, they can continue processing tasks locally and sync results back to the main chain once connectivity is restored. The Solana contract accepts delayed result submissions within a configurable grace window.

The Rust P2P layer communicates with the Python AI engine via a local Unix socket or named pipe. This clean separation between the network layer (Rust) and the compute layer (Python) means each can be updated, replaced, or scaled independently without affecting the other. The Tauri desktop client connects to both layers through a unified local API.

Section 5

Solana Smart Contract & Mint Authority

The Solana smart contract is the on-chain backbone of the MAI economy. It governs every financial interaction in the network: token minting, miner registration, reward distribution, NFT bonus tracking, staking, and DAO governance. The contract is written in pure Rust using the Solana native program interface -- not Anchor framework -- giving us maximum control over compute unit consumption and transaction costs. It will be developed and deployed after the presale phase is complete, as a separate repository from the current presale contract.

Mint Authority Transfer

This is one of the most important events in the MAI network lifecycle. Currently, during the presale phase, the MAI token mint authority is held by the presale smart contract (Presale Program ID: C5JTCJ5K9uhCkQHqCzFLw5MHTdYjTqvU9F5kxZXjDWJ9). The MAI token mint address is 68ix24J9qbr2EJfirDUVDh75bnjpkY4V65rMRnm2CgCy. The presale contract mints tokens only when a user makes a purchase -- no tokens are created without a corresponding payment. After the presale ends, the following transition occurs:

Step 1 -- Presale Contract Freezes

Once the final presale stage closes, the presale contract's mint function is permanently disabled. No further purchases are accepted and no new tokens can be minted through the presale contract. The contract enters a read-only state used only for vesting schedule execution and claim processing.

Step 2 -- New PoUW Contract Deployed

The MAI development team deploys the new Proof of Useful Work smart contract to Solana mainnet. The contract code is open-source and verifiable on-chain. Before mint authority is transferred, the contract undergoes a security audit by an independent Solana security firm. The audit report will be published publicly for community review.

Step 3 -- Mint Authority Transferred

A multisig transaction signed by the MAI team transfers the MAI token mint authority from the presale contract to the new PoUW contract. From this point forward, the only way new MAI tokens can enter circulation is through verified mining work. The multisig requirement ensures no single team member can unilaterally control the token supply. This transfer is irreversible and publicly verifiable on-chain.

Step 4 -- Mining Goes Live

Once mint authority is transferred and the desktop client is released, miners can begin registering their nodes, processing AI tasks, and earning MAI rewards. The token supply from this point grows exclusively in proportion to real decentralized AI compute contributed to the network.

PoUW Contract Architecture

The new PoUW contract consists of five core modules, each handling a distinct aspect of the network economy:

Miner Registry Module

Handles miner onboarding and hardware profile management. When a miner first starts their node, the desktop client submits a registration transaction containing: the miner's Solana public key, hardware tier declaration (GPU model, VRAM, CPU cores, RAM), geographic region code, and client version. The contract

creates an on-chain miner account that tracks cumulative rewards earned, task completion rate, uptime score, active NFT bonuses, and stake balance.

Task Verification Module

The core of the PoUW mechanism. Receives signed result payloads from miners, validates the Zero-Knowledge Proof against the task parameters, checks execution metadata consistency (token count vs. elapsed time vs. hardware tier), and confirms the miner's signature matches their registered public key. If all checks pass, the module triggers the reward calculation and minting process. Failed verifications increment the miner's fraud attempt counter.

Reward Distribution Module

Calculates and mints MAI rewards for each verified task. Applies the base reward formula, NFT multipliers from the miner's registered NFT tier, uptime bonuses for miners with 80-100% weekly uptime, and the current halving schedule factor. Tokens are minted directly to the miner's registered Solana wallet in the same transaction that verifies the work -- atomic and instant.

NFT Bonus Registry

Tracks which Warrior NFTs are associated with which miner accounts. During mining client setup, miners link their NFT wallet to their miner account. The contract reads the NFT tier on-chain and applies the corresponding permanent bonus: Bronze +5%, Silver +10%, Gold +15%, Platinum +20%, Airdrop NFT +10%. Bonuses stack with other multipliers and apply for the lifetime of the network -- they never expire and cannot be revoked.

Staking and DAO Module

Manages the staking mechanism and decentralized governance. Miners and token holders can stake MAI to earn a discount on premium AI services and gain DAO voting rights. Minimum stake tiers: 100,000 MAI for Basic DAO access, 500,000 MAI for Standard, 2,000,000 MAI for Advanced. The 14-day unstaking cooldown period prevents flash-governance attacks. NFT holders gain exclusive DAO voting rights up to 12 months before general token holders, rewarding early community members who believed in the project before the network launched.

Why Solana

Property	Solana	Ethereum	BNB Chain
Transaction speed	**65,000 TPS**	15-30 TPS	2,000 TPS
Transaction cost	**$0.00025**	$1 - $50+	$0.05 - $0.20
Finality time	**~400ms**	12-64 sec	~3 sec
Smart contract language	**Rust (native)**	Solidity	Solidity
DePIN ecosystem	**Largest (IO.net, Helium)**	Emerging	Limited

Solana's 400ms block finality means miners receive their rewards within seconds of task verification -- not minutes or hours. At $0.00025 per transaction, the cost of submitting a task result on-chain is negligible even for small rewards. The existing Solana DePIN ecosystem (IO.net, Helium, Render Network) provides a proven playbook and a ready-made community of infrastructure investors familiar with the tokenomics model MAI is building.

Security note: The PoUW contract will be open-source from day one. The community is encouraged to review the code before the mint authority transfer occurs. A bug bounty program will be announced prior to mainnet deployment, rewarding security researchers who identify vulnerabilities before they can be exploited.

Section 6

Reward Mechanism

The MAI reward mechanism is designed to be transparent, predictable, and fair. Every miner knows exactly how much they will earn before they start processing tasks. Rewards are calculated based on the type and quantity of hardware contributed, the number of active processing hours logged, and any bonuses applied through NFT ownership or exceptional uptime performance. All rewards are minted on-chain by the PoUW smart contract immediately after task verification -- there is no manual reward distribution or team discretion involved.

Base Reward Rates by Resource Type

MAI rewards are calculated per active processing hour for each resource type contributed. The base rates are fixed at network launch and decrease by 10% every 6 months (halving schedule):

Resource Type	Unit	Base Rate	Notes
GPU Compute	per MH/s	5 MAI / hour	Primary reward driver
CPU Compute	per GHz	2 MAI / hour	For CPU-only miners
RAM	per 1000 MHz	0.2 MAI / hour	Supplementary reward
Storage	per 10 GB	0.1 MAI / hour	For model caching
Network	per Mbps	0.1 MAI / hour	Bandwidth contribution

IMPORTANT: Rewards are calculated exclusively for active task-processing hours. A miner that is connected to the network but idle (not processing any tasks) earns zero rewards for that period. This design ensures the token supply grows only in direct proportion to real AI compute delivered to the network.

Halving Schedule

To control the long-term token supply and maintain reward sustainability, base reward rates decrease by 10% every 6 months from the network launch date. This gradual reduction incentivizes early miners who joined the network before the ecosystem reached maturity:

Period	Schedule	GPU Rate (per MH/s/hr)	CPU Rate (per GHz/hr)
Launch	**Q3 2027**	5.00 MAI	2.00 MAI
Period 2	**Q1 2028**	4.50 MAI (-10%)	1.80 MAI (-10%)
Period 3	**Q3 2028**	4.05 MAI (-10%)	1.62 MAI (-10%)
Period 4	**Q1 2029**	3.65 MAI (-10%)	1.46 MAI (-10%)
Period 5	**Q3 2029**	3.28 MAI (-10%)	1.31 MAI (-10%)
Stabilization	**Q1 2030+**	Rate floor TBD by DAO	Rate floor TBD by DAO

NFT Warrior Bonuses

MAI Warrior NFTs were distributed during the presale to early supporters. These NFTs provide permanent, lifetime mining bonuses that multiply all base rewards earned by the holder. NFT bonuses are verified on-chain by the PoUW contract at the time of reward calculation and apply automatically -- no manual

claiming or activation required:

NFT Tier	Presale Threshold	Mining Bonus	Early Mining	Early DAO Voting
Bronze	$50 - $99	+5% forever	1 month	3 months
Silver	$100 - $199	+10% forever	2 months	6 months
Gold	$200 - $299	+15% forever	3 months	12 months
Platinum	$300+	+20% forever	3 months	12 months
Airdrop NFT	First 100 buyers / stage (10K+ MAI)	+10% forever	2 months	6 months

Uptime Bonus

Miners who maintain consistent uptime earn an additional weekly bonus on top of their base rewards and NFT multipliers. The uptime is calculated as average weekly time the miner stays online and active:

Weekly Uptime	Weekly Bonus Reward
80% - 100%	+5% on all rewards
50% - 79%	+2% on all rewards
49% and below	No bonus

Reward Calculation Example

The following example illustrates how rewards stack for a miner with an RTX 4090 GPU (150 MH/s), 4 GHz CPU, 1,000 Mbps network, Silver NFT, and 80-100% weekly uptime, running for 8 active processing hours:

Component	Calculation	Result
GPU base reward	150 MH/s x 5 MAI x 8 hours	6,000 MAI
CPU supplement	4 GHz x 2 MAI x 8 hours	64 MAI
Network supplement	1,000 Mbps x 0.1 MAI x 8 hours	800 MAI
Subtotal base		6,864 MAI
Silver NFT bonus (+10%)	6,864 x 1.10	7,550.4 MAI
Uptime bonus (+5%)	7,550.4 x 1.05	7,927.9 MAI
TOTAL EARNED	**8 hours of active processing**	**7,927.9 MAI**

Note: The example above uses launch-period base rates (Period 1). As the halving schedule progresses, base rates decrease by 10% every 6 months. However, as AI demand and MAI token value grow, the USD value of rewards is expected to remain competitive. Early miners with NFT bonuses are permanently protected from the full impact of halving through their fixed percentage multipliers.

Section 7

Desktop Client (Tauri)

The MAI Desktop Client is the application that miners install on their computers to participate in the network. It is built with Tauri -- a modern framework that combines a Rust backend with a JavaScript/HTML frontend to produce a lightweight, secure, cross-platform desktop application. Unlike Electron-based apps that bundle an entire Chromium browser, Tauri uses the operating system's native WebView, resulting in binaries that are 10-20x smaller and significantly more memory-efficient. The MAI client will be available for Windows, macOS, and Linux.

Why Tauri

Performance and Size

A typical Electron app weighs 150-300MB. A Tauri app performing the same function weighs 5-15MB. For miners who need a lightweight background process that does not interfere with their GPU's primary workload, this difference is significant. The Rust backend handles all computationally intensive operations -- P2P networking, cryptographic signing, local API communication -- while the JavaScript frontend handles only the user interface rendering.

Security Model

Tauri's security model is stricter than Electron's by default. The frontend JavaScript layer cannot access the operating system or file system directly -- all privileged operations must go through explicitly defined Rust commands. This prevents XSS attacks from escalating to full system compromise, which is critical for an application that manages a Solana wallet and handles cryptographic signing operations.

Cross-Platform Native Feel

Tauri uses the native WebView on each platform -- WebKit on macOS, WebView2 on Windows, and WebKitGTK on Linux. This means the app looks and feels native on each operating system without requiring separate codebases. A single Rust + JavaScript codebase compiles to native binaries for all three platforms.

Client Architecture

The MAI desktop client consists of three integrated layers that communicate through well-defined internal interfaces:

Rust Core (Backend)

The Rust core manages all system-level operations: starting and stopping the Python AI engine subprocess, maintaining the P2P network connection, handling Solana wallet interactions, signing task results with the miner's private key, and exposing a local HTTP API that the frontend consumes. The Rust core runs as a background service and can operate in headless mode (without the GUI) for server deployments.

JavaScript Frontend (UI)

The frontend is built with a lightweight JavaScript framework and communicates with the Rust core exclusively through Tauri's IPC (inter-process communication) bridge. It renders the miner dashboard, displays real-time statistics, handles user settings, and provides the wallet connection interface. The frontend has zero direct access to the file system, network, or cryptographic operations -- all such requests are routed through the Rust backend.

Python AI Engine (Subprocess)

The Python inference engine runs as a managed subprocess spawned by the Rust core. The Rust layer starts, monitors, and restarts the Python process as needed. Communication between Rust and Python happens via a local Unix socket (on macOS/Linux) or named pipe (on Windows), using a lightweight JSON-RPC protocol. This clean separation means the AI engine can be updated independently of the client without requiring a full application reinstall.

Miner Dashboard

The dashboard is the primary interface miners interact with. It provides real-time visibility into all aspects of the mining operation:

Live Earnings Tracker

Displays MAI tokens earned in the current session, today, this week, and all-time total. Updates in real time as each task is verified on-chain. Shows the current MAI/USD estimated value based on the presale price until listing, and market price post-listing.

Hardware Monitor

Real-time graphs of GPU utilization percentage, VRAM usage, CPU load, RAM consumption, and network throughput. Color-coded alerts trigger when any resource exceeds safe operating thresholds, helping miners avoid hardware damage from sustained overload.

Task History Log

A scrollable log of all tasks processed: task ID, model used, tokens generated, processing time, reward earned, and verification status (confirmed on-chain / pending / failed). Miners can export this log as CSV for tax or accounting purposes.

Network Status Panel

Shows the miner's current connection status to the P2P network, number of connected peers, latency to nearest bootstrap nodes, and current position in the task assignment queue. Displays the miner's reliability score and uptime percentage over the rolling 30-day window.

NFT Bonus Display

If the miner's wallet holds a Warrior NFT, the dashboard prominently displays the active NFT tier and bonus percentage. The bonus is shown as a multiplier applied to the live earnings tracker, making the value of the NFT immediately visible to the miner.

Wallet Integration

Supports connection via Phantom, Solflare, or direct private key import (stored encrypted in the local OS keychain, never transmitted). Shows the miner's current MAI balance, pending unclaimed rewards, and provides a one-click claim button that submits the reward claim transaction to the Solana network.

Resource Allocation Presets

Miners can choose how much of their hardware to dedicate to MAI mining using four built-in presets. These can be adjusted at any time and take effect immediately without restarting the client:

Preset	Allocated Power Share	Best For
Light	25%	Background mining while working
Balanced	50%	Mixed use — gaming + mining
Maximum	100%	Dedicated mining rig
Flexible	10% - 100% (step 10%)	Advanced users with custom needs

The MAI Desktop Client will be released as open-source software on GitHub prior to the mining launch. Community contributors are welcome to submit improvements, additional language translations, and UI themes. The client will support auto-update functionality -- miners will receive protocol updates automatically without manual reinstallation.

Section 8

Development Roadmap

The MAI client will be built in a strict sequential order -- each component depends on the one before it. This approach ensures that the foundation is solid before the next layer is added, and that the team can focus fully on one component at a time rather than building everything in parallel and introducing integration complexity too early. Development begins immediately after the presale phase concludes.

Build Order Overview

Phase	Component	Timeline	Status
Phase 1	Python AI Inference Engine	Q3 2026	*Planned*
Phase 2	Rust P2P Network Layer	Q4 2026 - Q1 2027	*Planned*
Phase 3	Solana PoUW Smart Contract	Q1 - Q2 2027	*Planned*
Phase 4	Tauri Desktop Client	Q2 - Q3 2027	*Planned*
Phase 5	Testnet Launch	Q3 2027	*Planned*
Phase 6	Mainnet + Mining Goes Live	Q3 - Q4 2027	*Planned*
Phase 7	Mobile App (lightweight tasks)	Q4 2028	*Future*

Phase 1 — Python AI Inference Engine (Q3 2026)

The first component to be built is the Python AI inference engine. This is the most critical piece of the stack because everything else depends on it: the P2P layer needs a working inference engine to route tasks to, the smart contract needs verified task outputs to reward, and the desktop client needs a running engine to monitor and control.

Core Inference API

Build a Python FastAPI service that wraps Ollama and vLLM, exposing a unified REST endpoint for task submission and result retrieval. The API must handle model loading, prompt preprocessing, inference execution, token counting, and execution metadata collection in a single request-response cycle.

Model Management System

Build a local model registry that tracks which models are installed, their disk usage, last-used timestamps, and performance benchmarks on the current hardware. Includes automatic model downloading from IPFS and version management.

Execution Metadata Logger

Implement the structured logging system that records all data required for PoUW verification: task ID, model, token count, elapsed time, GPU utilization profile, and output hash. This data feeds directly into the ZKP generation module built in Phase 2.

Hardware Benchmark Tool

A one-time benchmark that runs when the miner first installs the client, measuring actual GPU throughput, CPU performance, and RAM bandwidth. Results are used to populate the on-chain hardware registration profile and calibrate the task assignment algorithm.

Phase 2 — Rust P2P Network Layer (Q4 2026 - Q1 2027)

With a working inference engine in place, the second phase builds the networking layer that connects miners to each other and to task requesters. This phase produces the core Rust library that will later be embedded in the Tauri desktop client.

libp2p Node Implementation

Build the base P2P node using Rust and libp2p: Kademlia DHT for peer discovery, Noise protocol for encrypted channels, GossipSub for broadcast messaging, and the Request-Response protocol for direct node queries.

Task Router

Implement the task assignment algorithm that scores available miners and routes incoming tasks to the optimal node. Must handle timeouts, reassignment, and concurrent task queues without deadlocks or race conditions.

ZKP Generation Module

Integrate a Zero-Knowledge Proof library into the Rust layer. The ZKP is generated from the execution metadata produced by the Python engine and signed with the miner's Solana private key before submission to the network.

Local IPC Bridge

Build the Unix socket / named pipe interface that connects the Rust P2P layer to the Python AI engine. Define the JSON-RPC protocol for task handoff and result collection between the two processes.

Phase 3 — Solana PoUW Smart Contract (Q1 - Q2 2027)

With both the inference engine and P2P layer operational and tested together, Phase 3 builds the on-chain economic layer. This is the most security-critical component -- a bug here could result in financial loss for miners or token supply manipulation. The contract will be audited before mainnet deployment.

Miner Registry Contract

On-chain program for miner registration, hardware profile storage, NFT bonus association, stake management, and reputation score tracking.

Task Verification Program

The core PoUW verification logic: ZKP validation, metadata consistency checks, fraud detection, and reward calculation. This program runs entirely on-chain and must be highly compute-efficient to minimize transaction fees for miners.

Token Minting Logic

Integration with the MAI SPL token mint: atomic reward minting on successful task verification, halving schedule enforcement, and NFT/uptime multiplier application.

Security Audit

Independent audit by a professional Solana security firm before testnet deployment. All findings will be published publicly. Mint authority transfer only occurs after audit completion and community review period.

Phase 4 — Tauri Desktop Client (Q2 - Q3 2027)

The final development phase wraps all three backend components into a user-friendly desktop application. By this phase, all backend components are fully functional -- the Tauri client is the integration layer that makes them accessible to non-technical miners.

Rust Core Integration

Embed the P2P Rust library into the Tauri backend. Implement Tauri commands for starting/stopping mining, reading live stats, and submitting Solana transactions.

Dashboard UI

Build the miner dashboard: real-time earnings, hardware monitor, task log, network status, NFT bonus display, and wallet integration. Responsive design that works on all screen sizes.

Installer and Auto-Update

Package the client as native installers for Windows (.msi), macOS (.dmg), and Linux (.AppImage / .deb). Implement the auto-update system so miners receive protocol updates without manual reinstallation.

NFT Holder Early Access

NFT holders (Bronze/Silver/Gold/Platinum/Airdrop) receive the desktop client 1-3 months before general release, depending on their NFT tier. This is the first tangible reward for presale participants.

Phase 5-6 — Testnet and Mainnet Launch (Q3 - Q4 2027)

After all four components are integrated and NFT holders have tested the client during the early access period, the network opens to all token holders. Testnet runs for a minimum of 4-8 weeks before mainnet activation. During testnet, miners earn test tokens to validate the reward mechanism without real economic stakes. Any critical bugs discovered during testnet are fixed before mainnet goes live. Mainnet activation triggers the mint authority transfer from the presale contract to the PoUW contract -- marking the moment MAI transforms from a presale token into a live mining network.

All timelines are estimates subject to technical complexity and team capacity. The community will receive progress updates through @miningmai_news on Telegram and the official website. Development repositories will be made public on GitHub starting from Phase 1, allowing community developers to follow and contribute to the build process.

Conclusion

Building the Decentralized AI Future

The MAI network represents a fundamentally different approach to AI infrastructure. Rather than concentrating AI compute in the hands of a few large corporations, MAI distributes it across thousands of individual contributors who own the hardware and earn real rewards for real work. The open-source AI models that power the network -- Llama 4, Mistral, Qwen -- are already available to anyone with a computer. MAI builds the rails that make these models accessible to everyone, fairly compensating the miners who keep the network running.

Every component described in this document -- the Python inference engine, the Rust P2P layer, the Solana PoUW contract, and the Tauri desktop client -- is being built with the same principles: open-source, auditable, community-owned, and economically fair. The PoUW mechanism ensures that every MAI token in circulation represents verified, useful computation. The NFT bonus system ensures that early believers are permanently rewarded for their trust. The DAO governance system ensures that the community -- not the founding team -- controls the network's future direction.

This document will continue to evolve as development progresses. Technical decisions may change, timelines may shift, and new features may be added based on community feedback and ecosystem developments. What will not change is the core mission: build the most transparent, fair, and technically sound decentralized AI compute network on Solana.

Official Links

Resource	Link
Website	https://miningmai.com
Telegram News	https://t.me/miningmai_news
Telegram Chat	https://t.me/miningmai_chat
Twitter / X	https://x.com/miningmai_ai
Airdrop Bot	https://t.me/mai_verify_bot
Smart Contract (Solana)	68ix24J9qbr2EJfirDUVDh75bnjpkY4V65rMRnm2CgCy

MAI Network — Technical Architecture Document v1.0 | May 2026 | miningmai.com This document is provided for informational purposes only. It describes planned technical architecture and is subject to change. Mining is NOT yet active. Always do your own research before making investment decisions.