LAYAI Whitepaper

v1.0 - release preparation · 24 September 2026

A lighter way to learn. An intelligence network.

LAYAI is developing a Deep Agent, reusable knowledge capsules and community computing, with blockchain coordinating updates to shared AI memory.

Follow the capsule lifecycle and the contribution network being developed around it.

Start with useful knowledge.

A Deep Agent receives a task and identifies what it needs to know. LAYAI's architecture organizes reusable knowledge into capsules, so useful context can carry forward.

Knowledge capsules and model weights are separate components.

Give knowledge a reusable form.

Condense task context, procedures and supporting sources into versioned capsules. Store them with indexes that make relevant knowledge discoverable.

The capsule library is the persistent source.

Bring the right context into focus.

Select the capsules related to the task and load their working copies. The agent combines them with the model and tools needed to execute the request.

The active workspace holds what the task needs.

Keep the knowledge. Reuse the space.

When a working copy is no longer needed, release it from active memory. Its stored capsule remains available; another task can use the same workspace.

Updated knowledge is saved as a new capsule version.

Connect useful work to the network.

Windows and Android 1.0 release candidates connect devices to a research centre. General access follows live account and device validation; capsule evaluation is the next research workload.

Larger GPU workloads follow as the network develops.

Knowledge grows. Together.

Agents propose capsule updates. The network evaluates their usefulness. The planned blockchain process records accepted versions for other agents to reuse.

Evaluations run off-chain. Published rules govern acceptance and disputes.

A shared intelligence ecosystem.

Deep Agent

Develop an agent that selects and reuses knowledge across tasks.

Community compute

Bring compatible devices together around useful AI workloads.

Shared memory

Coordinate accepted capsule versions and their history through blockchain.

$LAYAI economy

Connect eligible contributions and future service access through the token.

The development roadmap connects the agent, device network and token economy.

A shared memory. Clear responsibilities.

Agents & evaluators

Create capsules, execute tasks and evaluate quality off-chain.

Blockchain coordination

Record accepted versions, review decisions and disputes under published rules.

$LAYAI

Support planned service access and contribution rewards; evaluation funding and role-specific collateral are under design.

01 / LAYAI: a lighter way to learn

LAYAI is an AI × Web3 project building a **Deep Agent, reusable knowledge capsules and a community compute network**. Its goal is to make useful intelligence more accessible by reusing knowledge and activating the resources a task needs.

The project brings together four layers:

Layer	Role
Deep Agent	Organize experience into reusable knowledge; select relevant capsules and tools for a task.

Layer	Role
Contribution network	Assign compatible AI workloads to participating devices and record accepted results.
Blockchain coordination	Coordinate accepted capsule versions, review decisions and shared memory history.
$LAYAI	Support the planned contribution reward and service access economy.

The long-term objective is an agent that builds on previous work instead of reconstructing every useful piece of context for every task. Development starts with capsule evaluation and local execution, then expands into a connected contribution network.

02 / Knowledge capsules and selective execution

A knowledge capsule is a versioned package of reusable information: condensed task context, reusable procedures, retrieval indexes and provenance. Its contents and format depend on the workload. A capsule is distinct from the model weights that execute the task.

Compress > Activate > Compute > Release.

1. **Compress.** Structure useful knowledge into compact capsules. Record the source, version, format and access rules.

2. **Activate.** Select capsules relevant to the request. Load the necessary working copies into the agent's active context.

3. **Compute.** Execute the task with the chosen knowledge, model and tools. Record the result and resource use.

4. **Release.** Discard working copies that are no longer needed; reuse the available memory. The stored capsules remain available for later tasks.

Capsule reuse, context compression and selective execution are complementary techniques. Model loading is a separate implementation track. The research compares their combined end-to-end cost, including indexing, retrieval, loading and decompression. Any change to a capsule creates a versioned update rather than silently overwriting its source.

Building the Deep Agent core

The first planned workload, **capsule_eval_v1**, evaluates whether selected knowledge capsules preserve useful task quality while reducing the context and resources needed for execution. A small model runs locally on supported hardware. The comparison uses the same task, model version and output settings.

Evaluation	What will be published
Quality	Task accuracy or usefulness against a stated baseline and evaluation method.
Memory	Peak RAM and, where applicable, VRAM; capsule and index storage separately.

Evaluation	What will be published
Time	End-to-end latency including retrieval, loading, decompression and execution.
Resource cost	Context size, compute work and energy where reliable measurement is available.

Each result is tied to a task version, model digest, device profile and reproducible run. Benchmark reports will establish which configurations deliver useful gains. Performance figures are published with the measurements that support them.

03 / A contribution network for useful work

Connect 1.0 release preparation. Windows and Android release candidates implement device pairing, contribution controls and account history. Android installation and launch were checked on a Galaxy S22. Hosting has been purchased; live account access, cross-device statistics and task completion must pass before general access. Capsule evaluation remains a separate upcoming workload.

The planned network distributes independent AI jobs according to device capability. **Windows and Android are the first client targets.** Larger models are assigned to workers with sufficient RAM or GPU memory as those workload classes are added.

The contribution journey is straightforward: **pair a device > receive a compatible job > execute locally > submit a result > verify > record accepted work**.

Participants set resource limits and control when their device runs. Job assignments specify model and task versions, download size, expected resources, time limits and verification rules. Device pairing and one account-level device view keep participation simple.

A scheduler matches the workload to the device. A project-operated verification service initially checks results through task-specific validation, reference runs or repeated assignments. Accepted work enters an append-only contribution ledger; disputed results follow a review process. The network’s initial coordination and verification services are operated by the project.

Devices contribute separate useful jobs. Their RAM does not automatically combine into one large model. Distributed execution of a single large model is a later technical track. The first network demand comes from LAYAI’s own capsule evaluation; external workloads can follow demonstrated capacity and demand.

Blockchain and shared AI memory

Propose > Evaluate > Adopt > Reuse.

Participants test capsules; blockchain records the acceptance process.

LAYAI is developing a shared memory that agents can improve together. Blockchain is intended to coordinate which capsule versions enter that memory, under published acceptance and dispute rules. This connects the network’s computing work to the Deep Agent’s growing knowledge library.

1. **Propose.** An agent or contributor submits a new capsule or an improved version, identifying its source, prior version and intended task.

2. **Evaluate.** Network participants test quality and resource use against a defined baseline. Reports identify the capsule, evaluation method and execution conditions.

3. **Adopt.** The planned on-chain process records report commitments, evaluator signatures and disputes. A version becomes accepted for a stated task scope when the published review and acceptance conditions are met.

4. **Reuse.** Agents select accepted versions suited to their tasks and devices. Later findings can flag or retire a version, allowing clients to return to an earlier accepted version while preserving its history.

For example, a smaller coding capsule could enter shared memory after evaluations show that it meets the stated quality and resource criteria. Other agents could then reuse it without rebuilding the same task context. This develops the capsule library; updating that library does not automatically train model weights.

Computation off-chain; acceptance coordination on-chain. Capsules, model execution and test reports remain off-chain, with content hashes and references linking them to the registry. The chain records version relationships, acceptance states and review decisions. A hash identifies content; it does not establish its quality or guarantee its continued availability. Storage and retrieval remain part of the network design.

The intended benefit is a common update history and shared rules that independent participants can inspect and use. Evaluation remains responsible for judging usefulness; token holdings do not determine whether a capsule is correct. Initial coordination and evaluation are project-operated, with independent evaluators a later development step.

The first implementation scope is one capsule type and its proposal, evaluation, dispute and version-acceptance flow on the existing chain. This mechanism is in architectural development; the deployed testnet token does not yet implement it. Evaluator eligibility, acceptance thresholds, review periods and contract permissions will be specified before activation.

04 / A budgeted token economy

Proposed supply: 1,000,000,000 LAYAI · 18 decimals · BNB Smart Chain. The existing testnet token is a prototype. This revised allocation has not been distributed on mainnet and does not move existing balances.

Proposed allocation	Share	Tokens
Verified contribution rewards	35%	350,000,000
Research & ecosystem grants	15%	150,000,000
Operating treasury	15%	150,000,000
Team	10%	100,000,000
Community campaigns	10%	100,000,000
Liquidity reserve	5%	50,000,000
Future funding reserve	10%	100,000,000

The token is intended to support accepted research contributions and future network service access. Account creation and device participation do not require buying it. There is no guaranteed yield, market price, listing or

fixed points-to-token exchange rate.

Release discipline

- **Contribution reserve:** at least 60 months; a maximum budget of 70 million tokens per year. An approved period budget and verified eligible work are both required. Unused budgets remain reserved; this is not automatic emission or an entitlement.

- **Research and operating treasury:** 48-month planning horizon, milestone budgets and expenditure reports. Treasury tokens do not pay cash expenses unless a separately disclosed funding mechanism exists.

- **Team:** proposed 12-month cliff followed by 36 months of linear vesting from the declared mainnet start date. No immediate team unlock.

- **Community:** separate campaign budgets; the first distribution is proposed to use at most 10 million tokens (1% of supply), leaving 90 million for later campaigns. A campaign is not automatically launched by this document.

- **Liquidity:** 50 million reserved tokens, not an instruction to deposit all at once. Pairing assets, custody, pool parameters and lock policy require a separately funded plan.

- **Funding:** up to 10 million tokens in the first proposed round; the other 90 million are not part of that round. Later rounds need separate terms and disclosure.

Reserve wallets, vesting contracts, multisignature signers and spending authority will be published before activation. Team rewards and paid founder development must be disclosed separately to make total compensation visible. Proposed schedules require implementation and independent review; they are not enforced by the current testnet token.

05 / Development funding

Earliest planning window: Q2 2027 (April-June), subject to readiness. No active sale. This keeps funding at least six months after 24 September 2026. Funding is described here rather than promoted on the home page.

The first round is capped in planning at 1% of total supply. Price, soft/hard caps and spending targets must be derived from a costed scope, actual infrastructure costs and review quotations. Earlier 5/10 BNB caps, exchange rates and 90-day vesting were prototype parameters; they are not the current offer.

The earlier contract's 40% initial release and 60% participant-approved continuation remains a design option. Final refunds, review periods and vesting will be specified and tested together before accepting funds. Released development money cannot also be promised as fully refundable. Liquidity and ongoing reward liabilities require distinct budgets. No mainnet sale is authorized by this whitepaper.

Opening gates: working public account service, a reproducible capsule evaluation, costed deliverables, independent contract review, documented custody and published participation terms. Missing a gate moves the date; a calendar target never overrides it.

06 / Delivery roadmap

Target	Milestone	Deliverable and release gate
Q3 2026	Foundations & release candidates	Windows and Android 1.0 candidates, testnet token and purchased hosting. Next gate: live HTTPS sign-in and a real remote task.
Q4 2026	Connect 1.0 & public access	Central accounts, Google/EVM sign-in, shared statistics, signed updates and Android store testing. Release gate: device tests, backup recovery, abuse controls and measured capacity.
Q1 2027	Deep Agent & capsule evaluation	One capsule format, reproducible baseline and quality/resource comparisons on compatible devices. Deliverable: published methods, raw results and accepted-work records.
Q2 2027	Shared-memory acceptance	Proposal, evaluation, dispute and version history on testnet; reward eligibility and review rules. Gate: independent review and a reproducible capsule acceptance cycle.
Q3 2027	Network services & first distribution	Metered service experiments; first community distribution only after published eligibility, audit and funded claims. CMC/CoinGecko applications follow eligible mainnet launch and market data; acceptance is external.
Q4 2027	Capacity & independent evaluation	More GPU-compatible jobs, independent evaluators and integrations supported by real demand. Gate: published capacity and reliability results; no automatic promise of pooled model memory.

Dates are development targets, not guaranteed launches. Quarter order matters: accounts and safe device execution precede capsule evaluation; reproducible evaluation precedes acceptance coordination; distribution follows published rules and funded, reviewed contracts. Presale timing is covered only in the funding section.

07 / Contributions, points and campaigns

The account ledger separates checked availability, accepted work and community contributions. Android-only users can participate without a computer. The current Android candidate requires the app to remain open, charging and on unmetered Wi-Fi; it pauses on excessive temperature or low battery. It is not cryptocurrency mining or a promise of passive income.

Current contribution records are not redeemable tokens. No existing points are silently converted, repriced or erased by this proposal. Availability proves readiness checks, not that useful research was completed. A result passing a reference check is not a cryptographic proof of the physical device or its claimed energy consumption.

For a first campaign, a proposed fixed budget could be split 70% to accepted research, 20% to checked availability and 10% to reviewed community work. Eligibility, per-person limits, anti-duplicate rules and an appeal period must be published before the campaign starts. An unused research sub-budget is retained if there are no tasks; it is not automatically transferred to idle time. Contribution weights are a proposal, not a change to the current point ledger.

Planning sequence: rules and review process in Q2 2027; snapshot and first claims targeted no earlier than Q3 2027, after audited claims and funded custody are ready. Neither a snapshot nor a payout date is fixed today. Future campaigns depend on real demand and remaining reserves. Following a link alone does not verify social participation.

08 / Services and ecosystem access

The objective is repeatable capsule evaluation and useful service demand. Token distribution by itself creates neither research demand nor operating revenue. Service pricing, task budgets and settlement will be designed around measured costs; no artificial volume or guaranteed return is part of the model.

CoinMarketCap and CoinGecko application preparation follows a verified mainnet token, accurate project information and the relevant platform requirements. Acceptance and timing belong to those platforms. No exchange agreement or confirmed listing exists in this roadmap.

09 / Release status - 24 September 2026

Area	Verified position
Hosting	OVH VPS purchased and shown Active; application service and DNS cutover still pending.
Connect 1.0 Windows	Release candidate built; public endpoint validation and signed distribution pending.
Connect 1.0 Android	Signed APK/AAB built; Galaxy S22 installation and launch checked. Google/EVM end-to-end sign-in and remote work pending. Not published on Google Play.
Shared accounts and points	Implementation exists; production migration, recovery and cross-device tests remain release gates.
Capsule evaluation and Deep Agent	Research development; no public performance claim established.
Shared-memory acceptance	Design stage; no acceptance/dispute contracts deployed.
Token	Testnet prototype only; revised allocations and vesting are proposals.

Historical local contract tests are recorded in the test record. They do not certify this revised economic design or replace an independent review.

10 / Participation and publication

This is the 1.0 release-planning revision. General availability will be announced after operational checks; removing the word “pilot” does not establish scalability or technical completion. Research results, implemented features and future targets remain distinct. Mainnet addresses and final financial terms will be published before activation.

Official channels: X · Discord.

Design references: Golem token utility, Akash economic design, CMC listing criteria. These inform the analysis; they are not endorsements or LAYAI partnerships.