

A Synapse-Like Architecture for Continuous Cognitive Copiloting

LivePair AI

livepairai.com · v0.2 · public draft

ABSTRACT

LivePair AI is a real-time cognitive copilot for live work: meetings, coding sessions, presentations, and hardware builds. Rather than answering prompts, the system maintains a persistent internal model of the session — beliefs, goals, uncertainty, and a model of its own competence — and proactively surfaces what matters while it still matters. We describe the session architecture at the level of structure and guarantees, the metering and points ledgers that account for it economically, and the fair-launch token design we intend to grow into. Implementation details (thresholds, prompts, provider internals) are deliberately omitted.

KEYWORDS — cognitive copilot · proactive agents · persistent state · human-computer interaction · usage metering · fair-launch tokenomics

1. INTRODUCTION: THE INTERACTION MODEL IS WRONG

Prompt-driven assistants assume the human will stop working, formulate a question, and wait. In live work that assumption fails: the moment a question becomes necessary, the moment it was needed has already passed. The useful unit is not the answer — it is the *intervention at the right time*. A copilot that only speaks when spoken to is a search box with extra latency.

LivePair is built around presence. During a live session the agent is continuously situated: it observes the screen and camera, listens to the room, and maintains state across the whole session. Output is a stream of judgments about what deserves the human's attention — rendered on a heads-up display, not a chat log. This note formalizes that loop (§3), the economic layer that meters it (§4), and the network design target (§5).

2. POSITION: AUGMENTATION, NOT AUTOMATION

The design target is a copilot in the HCI sense: the human remains the decision-maker, and the machine acts on the human's *information state* rather than on the world. Three commitments follow:

- **Visibility of machinery.** Confidence and uncertainty are part of the output contract — a guess is rendered as a guess.
- **No covert steering.** Suggestions are visible and labeled; the agent never nudges toward an outcome the human did not ask for.
- **Data minimality.** Sessions are not sold, mined for advertising, or used to train models for other customers.

These are load-bearing constraints, not marketing: an agent that fabricates confidence or steers covertly is a worse copilot, because the human can no longer calibrate how much to trust it.

3.1 State

Each live session maintains a persistent internal state we write as

$$S_t = \langle B_t, G_t, U_t, M_t \rangle$$

where B_t is the belief state (what the agent currently holds to be true about the session), G_t the active goal stack, U_t the explicit uncertainty map over B_t , and M_t a self-model — the agent's own estimate of where it is competent in this session. Because S_t persists, the agent never starts cold; "as I said earlier" is a real reference, not a retrieved transcript line.

3.2 Perception and appraisal

Perception streams — screen frames, camera, microphone — arrive in short chunks and update S_t continuously. A separate appraisal loop scores candidate interventions. An intervention i is surfaced only when its expected value clears both a relevance threshold and the cost of interrupting the human:

$$\text{surface}(i) \iff p_i \cdot v_i > \tau + c_{\text{int}}$$

where p_i is the probability the intervention is currently relevant, v_i its value if relevant, τ the surfacing threshold, and c_{int} the estimated interruption cost of pulling the human's attention. A copilot that speaks whenever $p_i > 0$ is spam; a copilot that never speaks is useless. The appraisal loop exists to sit between those failure modes — quiet most of the time, loud exactly when it matters.

3.3 Affect dynamics

Alongside S_t the session carries a continuous affect vector $\mathbf{a}_t$ over seven axes — urgency, uncertainty, novelty, progress, social pressure, stress, confidence. Fast axes relax toward their baselines as leaky integrators with per-axis time constants:

$$a_k(t+dt) = a_k(t) + (a_k^0 - a_k(t))(1 - e^{-dt/\tau_k})$$

so arousal decays exponentially rather than snapping — an agent that spikes and instantly forgets behaves like a search box, not a copilot. Slow axes (stress, confidence) are evidence-driven rather than time-driven, and the learning rate itself adapts to the situation:

$$\eta_t = \eta_0 (1 + \alpha u_t + \beta n_t)$$

where u_t is current uncertainty and n_t current novelty — the system updates its own appraisal faster precisely when the situation is least predictable. Evidence moves the slow axes only beyond a dead-zone δ :

$$\Delta a_{\text{slow}} \propto \eta_t \cdot \max(0, |d| - \delta) \cdot \text{sgn}(d)$$

so noise below δ is ignored and only genuine prediction errors retrain the agent's sense of the session. Axis constants ($\tau_k, \eta_0, \alpha, \beta, \delta$) are tuned deployment parameters and intentionally unpublished.

3.4 Authority and durability

Session state lives in a single-writer authority; media arrives in chunks, state transitions are serialized, and the durable record survives reconnects. A dropped phone does not lose the session. Output is written to the HUD: interventions carry the confidence from U_t that produced them.

```
perception (screen · camera · mic)
```



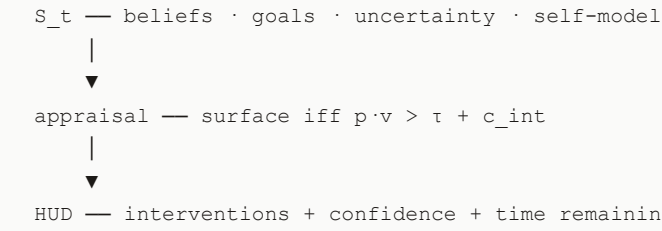


Figure 1 — the session loop. State persists; surfacing is selective.

4. METERING AND THE ECONOMIC LAYER

LivePair is funded by usage, not attention. Two meters exist, cleanly separated: prepaid **credits** — a micro-USD ledger charged at provider cost plus a disclosed markup — and **session time**, a pool of minutes drawn atomically when a session starts.

4.1 Charge rule

For a metered call with provider cost c , the customer debit is

$$\text{charged} = \lceil c \cdot (1 + m) \rceil, \quad m = 0.30$$

Every provider call is also written to an append-only usage ledger — what the provider burned is recorded separately from what the user was charged, so pricing can be recomputed and audited. The ledger, not the bill, is the source of truth.

4.2 Points ledger

Every account accrues **points**: an off-chain score recording who built this with us — signups, purchases, usage. A user's balance is the sum of signed events,

$$\text{pts}(u) = \sum_{e \in E_u} e_{\text{points}}$$

with each event idempotent on a unique reference id, so webhook redelivery can never double-award. Points have no cash value, are not transferable, and promise nothing. They are, however, the natural allocation table for a network if one ever exists.

4.3 Solvency invariant

Outstanding prepaid balance is deferred revenue. The system's reported liability is

$$\text{liability} = \text{balance}_{\text{outstanding}} \cdot \frac{\sum c_{\text{provider}}}{\sum \text{charged}}$$

— the realized cost-per-dollar-charged applied to what remains unconsumed. Hard platform caps bound the burn rate independently of user balances: a monthly provider-spend ceiling and a concurrency cap, both operator-adjustable without deploy.

5. INTENDED NETWORK TOKEN — DESIGN TARGET

Should LivePair launch a token, the design target is a fair launch in the spirit of Venice (VVV). Genesis supply is fixed:

$$S = 10^8 \text{ units, no mint function, zero emissions}$$

5.1 Allocation

TRANCHE	SHARE	DESTINATION
Community airdrop	50%	Points-proportional claim, per-wallet cap, sybil-filtered on verified identity + real usage history
Treasury	30%	Multisig-held; gradual market distribution funds operations — there is no presale
Team	10%	Vesting contract, 4-yr vest / 1-yr cliff
Liquidity	5%	DEX seed pool (Base)
Ecosystem	5%	Multisig-held incentives

Table 1 — allocation vector $A = [0.50, 0.30, 0.10, 0.05, 0.05] \cdot S$.

5.2 Staking: margin-backed compute pool

Stakers do not receive printed yield. Each epoch, a compute-credit pool is sized as a fixed share of *net margin* — settled revenue minus provider cost and payment fees — and distributed pro-rata by staked weight:

$$\text{pool}_t = \alpha \cdot \text{margin}_t, \quad \text{yield}_i = \text{pool}_t \cdot \frac{s_i}{\sum_j s_j}$$

with α a design-target constant (0.25). Because the pool keys off margin rather than a fixed */unitrate*, *stakeryieldisbackedbyrealproductrevenueandcanneverexceedit* : *margin=0 \rightarrow pool=0*\$. Token demand is therefore product demand — a staked unit is a claim on future compute, priced by the same ledger that meters it.

5.3 Buy & burn

A second share of net margin routes to a burn reserve that programmatically buys the token on-market and sends it to a burn address:

$$\text{burn}_t = \beta \cdot \text{margin}_t, \quad \beta = 0.25 \text{ (design target)}$$

Epoch-settled and verifiable on-chain. Growth tightens the asset instead of diluting it: as margin rises, more supply is permanently removed while the claims on capacity stay constant. The company retains $(1 - \alpha - \beta)$ of margin — the token commits a fixed fraction of success, never more than success produces.

6. GOVERNANCE AND ACCOUNTABILITY

LivePair is stewarded by its operating company (a US entity). Operating limits — capacity caps, spend ceilings, rate limits — are published policy, not silent throttles. The base model powering live sessions is named publicly and upgraded in the open. There is no token governance and no vote on safety rules; changes that affect users are made in daylight.

7. SCOPE OF THIS DOCUMENT

This note describes structure and guarantees, deliberately not implementation. Thresholds (τ , c_{int} estimation), prompts, provider internals, and the appraisal scoring function stay private. The architecture is open in intent, closed in detail.

8 . REFERENCES

- [1] Venice.ai — VVV token documentation: fair-launch allocation, no presale, staking-to-compute (DIEM-style) utility.
- [2] F. Battaglia et al. — belief-state and self-model literature underlying persistent agent state.
- [3] ERC-20 token standard (EIP-20); OpenZeppelin reference implementation.
- [4] LivePair AI — /manifesto: the public charter this note accompanies.

Nothing in this document is an offer, solicitation, or promise. No token exists today; points are an in-app score with no cash value. The token-economics section is a design target subject to change. © LivePair AI.