OFFPEAK - Build document

A Codex execution contract for a capped inference pilot on Robinhood Chain

Version 1.0 · October 12, 2026 · Specification only; no application has been built or deployed

Companions: Product overview · Internal sourcing

1. Project summary

Build an inference service selling calls to one approved open-weight model running on directly operated or contractually authorized spare GPU capacity. Show prices that respond to spare capacity, authorize calls with wallet signatures, charge measured usage from withdrawable prepaid balances, and anchor private billing receipts to Robinhood Chain.

P0 is a complete, deliberately narrow service: non-streaming text requests, one model revision, one region, vetted workers, one settlement token, EOA wallets, custodial prepaid balances, and same-wallet withdrawals. "Pay per call" means each successful call incurs its own usage charge; it does not mean one blockchain transaction per call. Clients must initially fund a balance.

P1 can add direct x402 payments, streaming, tool calls and delegated wallet permissions. P2 can add a supplier marketplace, $OFPK utility and execution attestations. Do not implement these during P0. The first phase can start locally without private infrastructure; real GPU, asset and custody gates must pass before their dependent phases.

2. Requirement digest

ID	Requirement	Product mapping
FR-001	Return a versioned model catalog and expiring capacity-based quote	F-001-002 / AC-001-002
FR-002	Authenticate ownership and authorize each inference call without an OFFPEAK API key	F-003 / AC-003
FR-003	Credit finalized deposits once and reserve funds atomically	F-004 / AC-004
FR-004	Execute the exact quoted model, meter it, charge once and release unused reserves	F-005 / AC-005
FR-005	Return available funds to the owner wallet without double payment	F-006 / AC-006
FR-006	Issue signed receipts and verifiable batch commitments	F-007 / AC-007
FR-007	Provide the customer console, quickstart and local verifier	F-008 / AC-008
ADM-001	Permit only approved operators to change worker availability and incident controls	F-009 / AC-009
INT-001	Private GPU adapter with real hardware and license evidence	A-001-002, A-006
INT-002	Robinhood RPC, settlement token, treasury signer and finality adapter	A-004-005
NFR-001	Exact arithmetic, immutable accounting, idempotency and recovery	AC-004-006
NFR-002	Bounded latency, privacy retention, rate limits and tenant isolation	AC-008-009
OPS-001	Reconciled reserves, tested recovery and real deployment evidence	AC-009
OPS-002	Workload quality and positive contribution margin before scaling	AC-010

3. Repo/project assumptions

Create a new repository only when implementation begins; this document does not prescribe an existing path. Recommended layout:

```
apps/control/src/{api,domain,jobs,adapters,ops}/
apps/web/src/{routes,components,lib}/
```

```
packages/contracts/src/ReceiptRegistry.sol
packages/protocol/src/{authorization,receipt,amounts}/
packages/client/src/
packages/db/{schema,migrations}/
gpu/{adapter,deploy,benchmarks}/
tests/{unit,integration,e2e,fixtures,load}/
scripts/{reconcile,verify-receipt,smoke,deploy}/
docs/{decisions,runbooks,phase-reports}/
infra/compose.yaml
```

Use a modular application and a background process from the same codebase. PostgreSQL outbox jobs are sufficient; do not add a message broker. Keep GPU serving in its own private process because its hardware and dependency needs differ from the control app.

The candidate production asset is USDG; Robinhood's official token page lists it. Do not assume USDC availability or copy another chain's asset address. PH-05 must pin the actual token address, bytecode identity, decimals and transfer behavior. Token directory

4. Global architecture rules

G-01 - Supply and model identity

Only allow workers with recorded hardware/operator ownership or serving authorization, region, model license approval, container digest, weights revision, tokenizer hash, chat-template hash and benchmark report. "Open weights" does not automatically mean unrestricted commercial hosting.

The adapter provides `describe()`, `health()`, `capacity()`, `execute(requestId, request)`, `status(requestId)` and `cancel(requestId)`. A stable request ID makes worker submission idempotent. Model substitution is prohibited. Treat vendor telemetry as evidence to inspect, not trustless proof.

Start with a pinned vLLM image unless the team's existing engine satisfies the adapter. Keep every worker endpoint behind private networking and mTLS; the serving runtime's API-key flag alone is insufficient perimeter protection. vLLM server documentation

G-02 - Pricing and authorization ceiling

Let `u` be a 60-second smoothed fraction of occupied serving slots, clamped to [0,1]. GPU utilization is supplemental telemetry: low utilization alone does not guarantee schedulable capacity. Accept work only when a fresh worker reports a free slot and the scheduler reserves it atomically.

For each input/output rate, use `rate = max(cost_floor_rate, baseline_rate × (0.55 + 0.45u))`, rounded upward to integer token atoms per million model tokens. The 0.55 factor is a tunable pilot assumption. Floor rates must cover measured supplier charges, platform overhead, failure allowance and an explicit margin; document the allocation between input and output rates using a fixed benchmark mix. Do not imply every low-demand interval is cheaper if the floor binds.

Quote TTL is 30 seconds. A quote fixes model revision, tokenizer/template revision, billable input count, maximum generated tokens, both rates, asset, wallet, body digest, fee of zero in P0, and maximum charge. A quote reserves neither funds nor capacity until accepted; acceptance may return `capacity_unavailable`. Once accepted, its rates cannot change.

For integer token counts `I`, `O` and atom-denominated rates `Pi`, `Po` per million tokens:

```
actual_atoms = ceil((I × Pi + O × Po) / 1_000_000)
cap_atoms    = ceil((I × Pi + max_output_tokens × Po) / 1_000_000)
```

Round the combined sum once. Count input with the exact pinned chat template; count all generated tokens according to the published model profile, including any billed non-visible tokens. Disable cached-token discounts in P0. A model with ambiguous metering is ineligible. If observed counts exceed the authorized cap or disagree with the quote's input count, do not overcharge: quarantine the request, refund the hold and alert operations.

Illustrative fixture only: with a six-decimal mock asset, `Pi=200000`, `Po=600000`, `I=1000`, `max_output_tokens=500`, the cap is 500 atoms. If `O=200`, charge 320 atoms and release 180. These are test numbers, not advertised rates or a claim about USDG decimals.

G-03 - Identity and replay protection

Browser login uses a one-use wallet challenge bound to the web origin, chain and expiry. The HTTP-only session authorizes reads and requesting a quote; it cannot by itself spend or withdraw funds. P0 supports EOA signatures only and rejects unsupported smart-contract wallets explicitly.

Use EIP-712 typed authorizations. Domain: name `OFFPEAK`, version `1`, configured chain ID, and a salt derived from the canonical service origin plus deployment environment. Do not fabricate a verifying-contract address for this off-chain protocol.

Spend fields: `action=INFER`, owner, asset, quote digest, request digest, max charge, nonce, expiry, and idempotency-key digest. Withdraw fields use `action=WITHDRAW`, owner, asset, amount, same-owner destination, nonce, expiry and idempotency-key digest. Purposes are not interchangeable. Hash the exact validated request using a pinned JSON canonicalization specification and SHA-256; publish cross-language fixtures. Never persist unsalted prompt digests in public logs.

Enforce uniqueness of owner/purpose/nonce and owner/route/idempotency key. A retry with matching authenticated identity and payload returns the original status/result, even after acceptance-time expiry. New execution with an expired signature fails. A different payload on the same key returns 409. The SDK never loads a wallet secret from model output.

G-04 - Ledger and custody

Use a double-entry journal with a unique business-event ID. Accounts include treasury assets, customer available liabilities, customer reserved liabilities, pending withdrawal liabilities, earned platform revenue and operator expense accounts. Reserve transfers available liability to reserved liability; success converts the charge to revenue and releases the remainder; failed execution releases all. Deposits create treasury assets and customer liabilities. Successful withdrawals reduce both.

One database transaction accepts an intent, validates funds, reserves funds, claims capacity, creates the request, consumes its nonce and inserts its dispatch outbox entry. Use row locking in a fixed order or serializable transactions with bounded retries. Never authorize from a stale balance cache.

Custody is explicit: the operator controls pooled treasury assets and processes withdrawals. The receipt registry holds no funds. Customer money must not finance GPU bills; fund supplier expenses and gas from a separate operating reserve. Move earned revenue only after reconciliation and preservation of all outstanding customer liabilities.

Treasury reconciliation requires finalized token assets to cover available + request-reserved + withdrawal-pending customer liabilities. Also reconcile pending/in-flight transfers separately. A one-atom unexplained difference creates an incident. P0 permits no transferable platform credits and no promotional balances.

G-05 - Deposits, finality and reorganization

P0 accepts direct transfers from a connected EOA to the configured treasury on the configured chain and asset. Credit the ERC-20 `Transfer.from` owner, not a submitted transaction hash or arbitrary claimed beneficiary. Direct exchange, bridge-contract, or smart-wallet funding is unsupported; clearly explain this before displaying funding instructions. Unknown transfers enter an unmatched-deposit queue and are never treated as revenue.

Uniqueness key: `(chain_id, token_address, tx_hash, log_index)`. Record block hash and amount, verify token contract and destination, and mark pending until the transaction's L1 batch is Ethereum-finalized. A raw receipt or N L2 confirmations is insufficient. Contractually and technically validate the RPC finality mapping, using L1 batch verification if needed. Robinhood finality documentation

Before finality, a reorganization removes pending credit. On contradictory data after credited finality, stop new spending/withdrawal submission for affected accounts, open an incident and append compensating accounting only after investigation; never silently erase journals or allow negative spendable balances.

G-06 - Request recovery and charging boundary

```
quoted → reserved → dispatched → running → succeeded → charged
                                 □ failed/cancelled → released
                                 □ uncertain → reconciled success or released failure
```

Before returning success, persist the encrypted result, terminal meter record, charge journal, receipt and publication-outbox intent consistently. A caller disconnect does not undo a completed, durably retrievable response. The SDK can recover that response for 24 hours using the same intent. This is the charging boundary; delivery to the caller's screen cannot be proven.

P0 is non-streaming. Provider failure or timeout without a complete retrievable result is unbilled. Set a 120-second execution deadline. Attempt cancellation and reconcile for at most 5 additional minutes; if still uncertain, release the customer hold, fence the original request permanently against later billing, and absorb any late supplier cost. Do not dispatch a second execution merely because a worker acknowledgement was lost.

Use unique `(request_id, terminal_charge)` keys. A job lease expiry triggers status reconciliation, not blind re-execution. Record every transition with compare-and-swap state checks.

G-07 - Withdrawal state machine

```
requested → reserved → signed → broadcast → finalized
                       □ unresolved/replaced → same logical withdrawal
requested/reserved → rejected/cancelled → funds released
```

Persist signed transaction bytes/hash and the signer nonce before broadcast. Serialize treasury nonce allocation. Replacement transactions use the same nonce, recipient and amount; track every hash. If broadcast status is unknown, do not release the hold or create another transfer. Release only after a conclusively failed/reverted transfer or pre-signing cancellation. Retry broadcasting the same bytes safely. A finalized transfer reduces pending liability once.

Withdraw only available funds to the owner wallet. P0 charges no withdrawal fee; the operating budget pays gas. Publish expected processing time and incident status. Treasury signer failure stops new withdrawal submission but must not hide existing claims or statuses.

G-08 - Receipt semantics

Private receipt fields: schema version, random receipt ID, owner, request ID, quote digest, model/worker revision, metered counts, rates, charged atoms, asset, chain, final status, timestamps and operator signing-key ID. The receipt payload excludes prompts and responses. Generate a private 256-bit salt for each leaf.

Canonicalize the payload; compute its SHA-256 digest. Sign that digest with an EIP-712 `Receipt(bytes32 payloadHash)` message using a domain separated from spend authorization. Define the Merkle leaf as `keccak256(abi.encode(RECEIPT_DOMAIN, payloadHash, salt))`; internal nodes hash sorted pairs, and an odd node is promoted unchanged. Commit cross-language fixtures before implementation. The proof bundle contains the canonical payload, signature, salt, siblings, batch ID, registry address and chain transaction.

The minimal non-upgradeable registry records immutable batch IDs, root, leaf count, schema version, receipt signing-key address and commit time. Only an authorized publisher can append; a separate multisig admin can rotate the publisher but cannot rewrite old batches. The signing key must be uniform within a batch and its historic identity recorded. No custody, tokens or payouts live in this contract.

Publish a batch every 5 minutes when there are receipts. Retry idempotently until finalized. Warn after 15 minutes without finalized publication; stop accepting new paid work after 60 minutes of unresolved anchoring, leaving withdrawals available if otherwise healthy. Store proof bundles for 90 days and allow early download.

A valid proof demonstrates that the operator signed a record and committed it by a particular chain event. It does not prove the GPU performed the work, the counters are truthful, or the customer is independent of the operator. No "proof of compute" label in P0.

G-09 - Privacy and operations

P0 does not guarantee anonymity or unlinkability. Public token transfers expose addresses, amounts and timing; the backend associates wallet accounts with requests and billing. Hosting, RPC and network operators may observe network metadata. Receipt commitments do not conceal these payment and infrastructure links. Before launch, document which services observe IP addresses, their access controls and retention; do not advertise zero identity exposure.

No prompt logging. Disable inference-server request-body/debug logs and inspect reverse-proxy/error reporting. Retain encrypted result artifacts for 24 hours with owner-only access, then delete them. Retain metadata/proofs for 90 days by default, with a separately approved accounting retention policy before real funds. Secrets belong in a secret manager; GPU and treasury identities have separate permissions.

Operator roles: read-only support; capacity operator; treasury operator; deployment administrator. Financial corrections use journal entries, reason codes and dual review; never direct balance edits. A pause matrix separates new requests, deposits, withdrawal submission and receipt publication. A GPU outage must not automatically disable withdrawals.

5. Coding standards

Strict TypeScript, schema-validated API boundaries, integer money, UTC storage, explicit units in field names, and no untyped wallet payloads. Public monetary fields are decimal strings. Domain modules must be independent of HTTP and RPC clients. Use adapter fakes only at integration boundaries.

Store migrations, test fixtures, architecture decisions and OpenAPI definitions in version control. Do not update cryptographic schemas, price semantics or chain configuration silently. Keep secrets, production prompts and real customer data out of fixtures. Use additive database migrations; financial schema rollbacks require a reconciliation plan.

6. Stack expectations

Layer	Chosen baseline	Constraint
Runtime	Node.js LTS, TypeScript, pnpm workspace	Pin versions in PH-01
Web	React + Vite	Served by control application
HTTP	Fastify + schema validation	Bounded body sizes/timeouts
Persistence	PostgreSQL + Drizzle migrations	Real PG in accounting tests
Jobs	PostgreSQL outbox and leased runner	Reconcile uncertain side effects
GPU	Pinned vLLM container + private adapter	Real GPU validation required
Chain client	viem	Typed signatures; explicit chain/asset config
Contracts	Solidity + Foundry + vetted role-control library	Minimal non-upgradeable registry
Testing	Vitest, Playwright, Foundry, k6	Include fault injection
Deployment	Containers, managed PG, private object storage	Separate staging and production

These are proposed choices, not a statement that dependencies exist locally. Verify supported versions and pin them at setup. Robinhood currently documents mainnet 4663 and testnet 46630; validate `eth_chainId` before every deployment. Network configuration

7. Commands and operating contract

PH-01 must create these exact root scripts; later phases fill the relevant suites. A missing required suite is FAIL, not a skipped pass. The commands below are planned interfaces, not commands executed while writing this document.

```
pnpm install --frozen-lockfile
pnpm lint
pnpm typecheck
pnpm build
pnpm db:migrate
pnpm test:unit
pnpm test:integration -- --suite <suite-name>
pnpm test:e2e -- --project chromium
pnpm test:contracts
pnpm test:gpu
pnpm test:load
pnpm ops:reconcile -- --network <configured-network>
pnpm receipt:verify -- --bundle <bundle-path>
pnpm smoke -- --environment <staging-or-production>
```

Read applicable AGENTS.md instructions first. Build one phase, verify it, save `docs/phase-reports/PH-NN.md`, and stop. The report records commit ID, exact command results, artifacts, unresolved risks and PASS/FAIL. Do not advance on a failure or fill in external evidence with a mock.

8. Requirement traceability map

Requirement	Build phase	Required evidence
FR-001 / INT-001	PH-04	Hardware benchmark, pricing fixtures, stale-worker rejection
FR-002	PH-02, PH-06	Replay/domain tests and keyless call
FR-003 / NFR-001	PH-03, PH-05	Concurrent reservations and finalized deposit deduplication
FR-004	PH-06	Real GPU request, crash recovery and exact ledger charge
FR-005	PH-07	Replacement/retry test and testnet withdrawal
FR-006	PH-08	Finalized registry root and independent verifier
FR-007	PH-09	Browser flow and runnable client example
ADM-001 / NFR-002 / OPS-001	PH-10, PH-11	Isolation, recovery drill, reconciliation and deployment
OPS-002	PH-04, PH-11	Quality evaluation, measured cost sheet and pilot report

9. Phase map

Phase	Capability	Entry gate
PH-00	Freeze scope, units and evidence requirements	This document
PH-01	Running scaffold, CI and private staging baseline	PH-00 PASS
PH-02	Wallet identity and authorization protocol	PH-01 PASS
PH-03	Correct ledger and reservations	PH-02 PASS
PH-04	Real GPU, metering and capacity-based quote	PH-03 PASS + GPU/model evidence
PH-05	Finalized deposits	PH-04 PASS + verified asset/finality adapter
PH-06	Complete paid inference request	PH-05 PASS
PH-07	Reliable withdrawals	PH-06 PASS
PH-08	On-chain receipt verification	PH-07 PASS
PH-09	Usable console and agent client	PH-08 PASS
PH-10	Operational and security readiness	PH-09 PASS
PH-11	Capped public deployment and pilot	PH-10 PASS + real-money launch decisions

10. Full phase packets

All phases start **Not started**. Global failure procedure: stop; classify the failure; apply the smallest repair; rerun the phase verification; if unresolved, split the phase or revert its code. Preserve all journal and chain evidence. Each implementation prompt below incorporates G-01-G-09 and the command contract in §7.

PH-00 - Operating contract

Objective/outcome: eliminate ambiguous implementation choices before writing product code. **Requirements:** all, as definitions. **Why now:** money, identity and proof semantics must agree. **Entry:** this specification. **Scope/files:** `docs/decisions/`, `docs/phase-reports/`, `packages/protocol/fixtures/`. **Artifacts:** scope record, field dictionary, signing and receipt test vectors, dependency/evidence checklist. **Rules:** G-01-G-09.

Implementation prompt: "Freeze OFFPEAK P0 exactly as described in this build document. Write decision records for custody, token units, charge boundaries, wallet scope, finality and receipt semantics. Define canonical authorization and receipt fixtures and the external evidence required at PH-04, PH-05 and PH-11. Do not select an unverified model or asset address. Done when all subsequent interfaces have named inputs, outputs and errors. Stop after PH-00."

Verification prompt: "Review only the PH-00 records. Trace every requirement in §2 to a phase and every monetary field to an explicit unit. Check that no P1 feature is required by P0. Check cryptographic encodings for deterministic cross-language representation. Return PASS or FAIL with the smallest correction list. Do not build the application."

Commands/evidence: no runtime test yet; record a completed contract review in `PH-00.md`. **Acceptance/gate:** no undefined amount units, no claim of trustless execution, all external gates assigned. **Common failure:** pretending an unknown vendor integration exists. **Recovery/rollback:** correct the record; discard inconsistent fixtures before code depends on them. **Handoff:** approved field and fixture contract to PH-01. **Notes:** private GPU evidence is not required to begin this phase.

PH-01 - Scaffold and staging baseline

Objective/outcome: establish a reproducible app and CI. **Requirements:** NFR-002, OPS-001 foundation. **Why now:** later tests need a stable runtime. **Entry:** PH-00 PASS. **Scope/files:** workspace manifests, control health endpoint, web shell, compose, CI, root scripts. **Artifacts:** lockfile, local PG, private staging URL and deployment runbook. **Rules:** G-09 and §5-7.

Implementation prompt: "Create the repository layout in §3 and pin supported dependencies. Add health/readiness endpoints, a minimal React shell, local PostgreSQL, lint/typecheck/build scripts, and a private staging deployment with test-only configuration. Define every §7 command and fail clearly if a requested suite is not implemented. Add structured redacted logging. Do not implement billing or GPU routing. Done when a fresh checkout builds and the staging health check works. Stop after PH-01."

Verification prompt: "From a clean install run `pnpm install --frozen-lockfile`, `pnpm lint`, `pnpm typecheck`, `pnpm build`, and `pnpm smoke -- --environment staging`. Check readiness fails when PG is unavailable and no secret appears in web assets. Return PASS/FAIL; add no feature scope."

Evidence/gate: clean CI and staging health output. **Failure pattern:** environment works only on one laptop. **Recovery/rollback:** fix pinning or deployment configuration; remove the staging deployment if irreparable. **Handoff:** working CI and script contract to PH-02. **Notes:** staging must not accept real funds.

PH-02 - Wallet identity and signing

Objective/outcome: establish unambiguous account ownership and spend intent. **Requirements:** FR-002. **Entry:** PH-01 PASS. **Why now:** ledger ownership depends on it. **Scope/files:** protocol authorization, auth routes, session middleware, client signing helper, replay tests. **Artifacts:** EIP-712 fixtures and account/session migrations. **Rules:** G-03, G-09.

Implementation prompt: "Implement EOA login challenges, secure browser sessions, owner-only reads, and INFER/WITHDRAW signature verification with the fields in G-03. Build a TypeScript signing helper. Session access alone must never authorize spending. Test mainnet/testnet, origin and action separation. Do not debit funds yet. Done when positive fixtures pass and replay, tampering, wrong purpose, expiry and unsupported-wallet cases fail. Stop after PH-02."

Verification prompt: "Run `pnpm test:unit` and `pnpm test:integration -- --suite identity`. Verify the fixtures independently using a second encoding implementation. Attempt cross-wallet reads, nonce replay, modified body/cap, wrong network and CSRF. Return PASS/FAIL with evidence, without adding payment features."

Gate: AC-003 authorization subset passes. **Failure pattern:** accepting a session cookie as payment consent. **Recovery/rollback:** disable signed write routes and fix verification; revoke test sessions. **Handoff:** verified account and intent types to PH-03. **Notes:** wallet keys remain outside server/client logs.

PH-03 - Ledger and reservations

Objective/outcome: ensure a balance cannot be spent twice. **Requirements:** NFR-001, FR-003 foundation. **Entry:** PH-02 PASS. **Why now:** GPU dispatch must never outrun funds. **Scope/files:** journal/posting schemas, reservation domain, request state, outbox. **Artifacts:** migrations and concurrency suite. **Rules:** G-04, G-06.

Implementation prompt: "Implement balanced immutable journals, customer available/reserved balances, withdrawal-pending liabilities, exact atom arithmetic and transactional intent creation. Add unique business-event and idempotency constraints and append-only corrections. Use test funding entries only, clearly marked. Done when concurrent reservations never make available funds negative and all postings balance. Do not connect chain deposits. Stop after PH-03."

Verification prompt: "Run `pnpm db:migrate`, `pnpm test:unit`, and `pnpm test:integration -- --suite ledger` against real PostgreSQL. Race 100 reservations and withdrawal holds against a limited balance. Replay terminal events and crash between transaction steps. Verify conservation, uniqueness and exact rounding. Return PASS/FAIL."

Gate: zero imbalance and no overdraw under race tests. **Failure pattern:** reserving after dispatch or using floating point. **Recovery/rollback:** disable writes, repair transactions, rerun tests; reset only disposable test databases. **Handoff:** stable journal and reservation APIs to PH-04. **Notes:** ledger migrations cannot be destructive in production.

PH-04 - Real GPU and quote engine

Objective/outcome: demonstrate sellable, measurable compute. **Requirements:** FR-001, INT-001, OPS-002 foundation. **Entry:** PH-03 PASS plus authorized GPU access and candidate model/license evidence. **Why now:** pricing must come from actual serving economics. **Scope/files:** GPU adapter/deploy, model manifests, quote domain, benchmarks and fixtures. **Artifacts:** real benchmark and cost-floor calculation. **Rules:** G-01-G-02.

Implementation prompt: "Integrate the existing serving engine if it meets G-01; otherwise use pinned vLLM. Implement worker identity, health/capacity, idempotent execution/status/cancel, metering and quotes. Evaluate one model on a frozen representative workload; document hardware, concurrency, quality threshold and supplier terms. Compute provisional floor rates from measured throughput. Done when one real GPU can serve the pinned model and quote behavior matches G-02. Do not market mock performance. Stop after PH-04."

Verification prompt: "Run `pnpm test:integration -- --suite pricing` and `pnpm test:gpu`. Check stale capacity rejection, floor-bound rates, utilization changes, expired quotes, full worker slots, exact model revision, cancellation and token counters. Repeat the benchmark with recorded inputs. Return PASS/FAIL; real hardware evidence is mandatory."

Gate: AC-001-002 plus real metering evidence and a quality threshold pass. **Failure pattern:** good throughput on prompts unlike buyer workloads. **Recovery/rollback:** adjust model/profile or capacity policy and rerun; keep it unavailable until pass. **Handoff:** approved immutable model profile and rates to PH-05. **Notes:** no external finding establishes that the founder's existing engine already works.

PH-05 - Finalized deposits

Objective/outcome: fund the ledger from verifiable token transfers. **Requirements:** FR-003, INT-002. **Entry:** PH-04 PASS; chain/asset/finality evidence. **Why now:** paid calls need genuine funded balances. **Scope/files:** chain adapter, deposit indexer, config schema, deposit status routes. **Artifacts:** token config, event fixtures, finality proof and testnet transaction. **Rules:** G-04-G-05.

Implementation prompt: "Implement the direct-EOA deposit route in G-05 with pending/finalized status, duplicate suppression and restartable block cursors. Use a mock asset on testnet. Validate the candidate production USDG deployment and record contract identity, decimals and transfer behavior separately. Prove finality semantics against the chosen RPC. Done when duplicate/reorganized events cannot inflate balances and a finalized testnet deposit credits once. Stop after PH-05."

Verification prompt: "Run `pnpm test:integration -- --suite deposits` and `pnpm ops:reconcile -- --network robinhood-testnet`. Submit a test transfer, restart the indexer, replay logs, simulate reorgs and reject wrong-token/network/recipient events. Inspect credited finality evidence. Return PASS/FAIL; a mocked finality flag alone cannot pass."

Gate: AC-004 deposit subset passes. **Failure pattern:** trusting a user-submitted transaction hash or soft confirmation. **Recovery/rollback:** pause crediting, rebuild cursors from canonical events, append corrections in persistent environments. **Handoff:** funded accounts and reconciled chain adapter to PH-06. **Notes:** no real funds until PH-11.

PH-06 - Paid inference loop

Objective/outcome: complete one authorized, correctly charged model request. **Requirements:** FR-002, FR-004, NFR-001. **Entry:** PH-05 PASS. **Why now:** all core dependencies exist. **Scope/files:** chat API, request dispatcher, result artifacts, charge logic, receipt payload creation. **Artifacts:** paid-call integration transcript and failure matrix. **Rules:** G-02-G-04, G-06, G-09.

Implementation prompt: "Wire quote acceptance, signed intent, atomic funds/capacity reservation, durable dispatch and definitive result recovery. Persist a complete encrypted result before finalizing a successful charge. Generate the private receipt payload; chain publication comes later. Implement 120-second execution timeout and the five-minute uncertain-result resolution rule. Done when an agent call consumes the exact charge once and retries recover the same result. Stop after PH-06."

Verification prompt: "Run `pnpm test:integration -- --suite paid-inference` and `pnpm test:gpu`. Use the 500/320/180-atom fixture, race duplicate requests, kill the gateway and runner at each boundary, lose worker acknowledgement, disconnect the caller and simulate late success after refund. Assert no duplicate execution, no late charge and no excess charge. Return PASS/FAIL."

Gate: AC-003 and AC-005 pass on a real worker with test funds. **Failure pattern:** charging before durable result storage or retrying uncertain execution blindly. **Recovery/rollback:** stop admission, reconcile all in-flight intents, repair idempotency. **Handoff:** settled requests and private receipts to PH-07. **Notes:** this phase does not claim full OpenAI SDK drop-in compatibility.

PH-07 - Withdrawals

Objective/outcome: return unused funds reliably. **Requirements:** FR-005, INT-002, NFR-001. **Entry:** PH-06 PASS. **Why now:** withdrawability is part of the core promise. **Scope/files:** withdrawals API/domain, signer adapter, nonce coordinator, reconciliation. **Artifacts:** testnet receipt and recovery fixtures. **Rules:** G-03-G-05, G-07.

Implementation prompt: "Implement same-owner withdrawals from available funds, signed intent validation, atomic holds, durable signed-transaction persistence and idempotent broadcast/replacement handling. Separate gas funding from customer liabilities. Done when a testnet withdrawal finalizes once and all uncertain states preserve the hold safely. Stop after PH-07."

Verification prompt: "Run `pnpm test:integration -- --suite withdrawals` and `pnpm ops:reconcile -- --network robinhood-testnet`. Race withdrawal and inference, reject foreign destinations, inject crash-after-signing, lost RPC response, replacement, revert and reorg. Prove exactly one transfer or an explicitly unresolved hold per intent. Return PASS/FAIL."

Gate: AC-006 and reserve coverage pass. **Failure pattern:** refunding a hold while its transaction may still land. **Recovery/rollback:** stop new signing, reconcile signer nonces and chain hashes; do not erase pending transfers. **Handoff:** complete funding/spending/withdrawal cycle to PH-08. **Notes:** unresolved chain ambiguity is an incident, not permission to pay twice.

PH-08 - Receipt registry and verifier

Objective/outcome: make billing records independently checkable against chain commitments. **Requirements:** FR-006. **Entry:** PH-07 PASS. **Why now:** final charge records now exist. **Scope/files:** receipt codec, registry contract, batch worker, verifier CLI. **Artifacts:** verified testnet contract, receipt bundle, independent fixture implementation. **Rules:** G-08-G-09.

Implementation prompt: "Implement the precise receipt, signature and Merkle conventions in G-08, a minimal append-only registry, publisher rotation, batch retry and finality tracking. Build a verifier that reads the configured registry and validates the full bundle independently of the OFFPEAK API. Done when a testnet receipt verifies and tampering fails. Do not add fund custody or token issuance to the contract. Stop after PH-08."

Verification prompt: "Run `pnpm test:contracts`, `pnpm test:integration -- --suite receipts`, and `pnpm receipt:verify -- --bundle tests/fixtures/finalized-receipt.json`. Check odd leaf counts, one leaf, altered fields/salts, wrong chain/registry, unauthorized publisher, key rotation and duplicate batches. Confirm receipt-registry payloads contain no customer content or customer wallet addresses. Separately confirm the privacy disclosure explains that token transfers expose addresses, amounts and timing. Return PASS/FAIL."

Gate: AC-007 passes; historic roots cannot be overwritten. **Failure pattern:** treating root inclusion as validation of usage truth. **Recovery/rollback:** halt publication, preserve receipts, repair codec; deploy a new registry version if contract logic is faulty and clearly mark both histories. **Handoff:** verified bundles and publication status to PH-09. **Notes:** no upgrade proxy.

PH-09 - Customer console and agent client

Objective/outcome: make the complete product usable without operator assistance. **Requirements:** FR-007, NFR-002. **Entry:** PH-08 PASS. **Why now:** UI can display real domain states. **Scope/files:** RT-001-009 frontend, client package, developer docs, verifier UI. **Artifacts:** runnable quickstart and browser evidence. **Rules:** G-02-G-09.

Implementation prompt: "Build the routes/components in the overview with real API data and explicit pending/stale/error states. Provide a TypeScript example for quote→sign→call→retry→receipt and funding/withdrawal instructions. Run receipt validation locally in the browser without uploading the bundle. Explain custody, retention and proof limits at the relevant decisions. Done when a new user completes the full testnet journey. Stop after PH-09."

Verification prompt: "Run `pnpm test:e2e -- --project chromium`, `pnpm typecheck`, and `pnpm build`. Exercise wrong network, insufficient funds, quote expiry, worker failure, pending withdrawal and unpublished proof. Verify keyboard-only navigation and secret-free browser assets. Execute the documented client example from a clean environment. Return PASS/FAIL."

Gate: AC-008 passes and every route has real data or an honest empty state. **Failure pattern:** labeling pending transfers as available funds. **Recovery/rollback:** repair the affected UI/client path without changing accounting semantics. **Handoff:** complete staging experience to PH-10. **Notes:** no fabricated production statistics.

PH-10 - Operations and hardening

Objective/outcome: establish evidence that incidents preserve funds and privacy. **Requirements:** ADM-001, NFR-002, OPS-001. **Entry:** PH-09 PASS. **Why now:** full-system faults can be tested. **Scope/files:** ops roles, pause matrix, alerts, runbooks, retention jobs, load/fault suites. **Artifacts:** restore drill, reconciliation report, review findings and resolved release blockers. **Rules:** G-01-G-09.

Implementation prompt: "Implement role restrictions, immutable operator audit events, treasury coverage checks, separate pause controls, 24-hour result deletion and receipt-lag alerts. Drill RPC, GPU, database, signer and anchoring failure. Obtain focused review of payment authorization, ledger, withdrawal recovery and the registry before real funds. Done when release-blocking findings are repaired and recovery evidence exists. Stop after PH-10."

Verification prompt: "Run `pnpm lint`, `pnpm typecheck`, `pnpm test:unit`, all named integration suites from prior phases, `pnpm test:contracts`, `pnpm test:e2e -- --project chromium`, `pnpm test:load`, and testnet reconciliation. Restore a backup in isolation and reconcile it forward against chain events. Inspect actual logs for content leakage. Return PASS/FAIL."

Gate: AC-009, retention checks, admission controls and tested recovery pass; no unresolved critical/high payment or custody defect. **Failure pattern:** backup exists but cannot restore or replay safely. **Recovery/rollback:** keep real-money mode disabled, fix and repeat only affected evidence plus adjacent regressions. **Handoff:** signed-off release report to PH-11. **Notes:** measured SLOs replace provisional targets if justified and documented.

PH-11 - Capped deployment and pilot

Objective/outcome: release an operating product with measurable economics. **Requirements:** OPS-001-002, all end-to-end acceptance criteria. **Entry:** PH-10 PASS; operating entity/jurisdictions, applicable custody requirements, supplier rights, production token route, reviewed signing controls and operating reserve resolved. **Why now:** technical and operational gates exist. **Scope/files:** production config, deploy scripts, status page, launch and pilot reports. **Artifacts:** public URL, deployment manifest, production receipt/withdrawal evidence, seven-day pilot report. **Rules:** all.

Implementation prompt: "Deploy the reviewed release with separate production identities and a capped allowlisted paid pilot. Suggested initial caps are 100 token units per wallet and 2,000 aggregate customer liability, subject to the approved operating budget. The website/docs can be public while paid access is limited. Smoke-test using an operator-funded minimal amount: deposit, call, receipt verification and withdrawal. Run a seven-day supervised pilot and record quality, margin, failure, withdrawal and proof metrics. Done when the live loop works and the pilot exit criteria below are met. Stop after PH-11."

Verification prompt: "Run `pnpm smoke -- --environment production`, `pnpm ops:reconcile -- --network robinhood-mainnet`, and the independent verifier on a production bundle. Inspect chain ID/token/registry/signer manifest, finalized transfer receipts, monitoring, privacy pages and incident ownership. Review the seven-day cost/quality report. Return PASS only with live evidence and all pilot exit criteria satisfied; otherwise FAIL or blocked with the exact dependency."

Gate: all AC criteria pass; 1:1 liability backing holds; quality meets the frozen PH-04 threshold; contribution margin is positive on measured non-synthetic paid workload; no unexplained ledger differences. A seven-day clock alone does not establish demand or pass the economics gate.

Failure pattern: subsidized prices appear profitable because failed work and gas are omitted. **Recovery/rollback:** stop expansion; disable new deposits/paid admission when necessary; preserve withdrawal servicing, receipts and customer claims; repair or reprice transparently. **Handoff:** maintenance owner and hypercare record. **Notes:** deploying code is not a $OFPK issuance event.

11. Cross-phase decision log

Decision	Default	When it may change
D-001 Serving engine	Existing adapter-compatible tech or pinned vLLM	PH-04 benchmark evidence
D-002 Payment mode	Custodial wallet-signed prepaid calls	Separate P1 x402 design; no silent migration
D-003 Asset	Candidate USDG; mock token in tests	Verified production integration before PH-05/11
D-004 Token role	$OFPK deferred	Separate approved utility/economics specification
D-005 Inference scope	One non-streaming text model	P1 feature-specific metering/refund design
D-006 Proof level	Signed receipts + immutable roots	Independent execution attestation project
D-007 Retention	24h result artifacts / 90d metadata default	Documented privacy/accounting decision
D-008 Chain	Robinhood Chain	User-approved product change, not implementation convenience

12. Risk register

Risk	Detection	Mitigation / owner
Supply is not reliably idle or authorized	Contracts and workload contention	Supplier lead; narrow committed windows
Open model is too weak for buyers	Frozen workload quality set	Product/ML owner; limit supported workloads
Low utilization makes discount uneconomic	Cost per successful billed workload	Engineering/finance; adjust floor or stop offer
Stolen wallet/signing key	Signature anomalies and spend caps	Security; dedicated funded wallets, scoped signing service
Custodial liabilities exceed assets	Daily and event-driven reconciliation	Treasury; stop admission and investigate immediately
Ambiguous worker/chain side effect	Stuck state/nonce alerts	Engineering; reconcile, never blind retry
On-chain receipt overclaim	Copy review and verifier wording	Product; describe integrity rather than truth
Asset freeze, bridge or RPC disruption	Transfer probes and provider alerts	Treasury; status disclosure and controlled admission
Existing engine is unavailable	PH-04 integration evidence	Engineering; baseline adapter with honest readiness status

Economic worksheet: `contribution = paid inference revenue - supplier compute - serving/control infrastructure - failed/unbilled compute - payment/gas allocation`. Keep incentives and treasury transfers outside operating revenue. Allocate idle reserved-capacity costs across the whole purchase window, not only its busy minutes.

Illustrative 1,000-call scenario: revenue 2.00 token units; supplier 1.20; infrastructure 0.20; failed compute 0.10; payment/anchoring allocation 0.05; contribution 0.45 (22.5%). This is arithmetic, not a forecast. If revenue falls 30% with costs unchanged, contribution becomes -0.15. Replace every input with benchmark and pilot measurements before presenting margins.

13. Final launch checklist

[ ] One model has pinned weights/tokenizer/template, license and workload evaluation evidence.

[ ] Supply relationship permits third-party commercial inference and defines interruption terms.

[ ] Production chain/token/finality configuration has been independently verified.

[ ] Customer liability caps and separate operating/gas reserves are configured.

[ ] Identity, ledger, withdrawal and receipt reviews have no unresolved release blockers.

[ ] Custody, withdrawal, retention and model limitations are visible to customers.

[ ] A production deposit→call→receipt→withdrawal loop has finalized successfully.

[ ] Reconciliation, backup restore, incident contacts, pause controls and monitoring work.

[ ] No unsupported competitor accusation, guaranteed savings or proof-of-compute claim appears.

[ ] Public token material does not imply $OFPK issuance, redemption, yield or a Robinhood affiliation.

[ ] The paid pilot passes its quality/economic gates before limits are expanded.

14. Post-launch hypercare

For the first seven days, review treasury reconciliation at least daily and alert continuously on unexplained discrepancies, stuck withdrawal nonces, stale capacity and receipt lag. Inspect a sample of real, consented billing disputes using metadata, without routinely collecting prompts.

Hold a daily decision on capacity, price floor and customer caps. Keep a public status page current during incidents. After seven days, retain the capped pilot until there is enough non-synthetic workload to assess margin and quality; lack of volume is an unresolved commercial result, not a technical failure.

The next roadmap decision should follow measured demand: direct x402 if buyers resist prepayment; streaming/tool calls if compatibility blocks adoption; additional workers if capacity rejects useful demand. Evaluate $OFPK only against a specific coordination problem and write its economics separately.