

PROGRAMMING DESK

Programming Desk: seven agent seats that work under path ownership and gates

Programming Desk is a seven-seat agent system for software development, commissioned by Omega Labs. Its gates and gateway are built and under test on a branch that is not yet merged, and its upstream integrations are configuration-gated.

Published 21 October 2026 · Updated 21 October 2026 · spectrumweb.co/case-studies/programming-desk

01 / 09

The Problem

A coding agent can write plausible code, explain why it works, and report success without running it. Several agents working in one repository also need one clear owner for each path.

02 / 09

What is the Programming Desk?

The Programming Desk is a seven-seat agent system for software development across backend, web, Android, iOS, and infrastructure. It was commissioned by Omega Labs. Each seat maps to one bot identity:

- lead: bot-00-programming-lead
- systems: bot-01-systems-backend
- web: bot-02-web-edge
- android: bot-03-android
- ios: bot-04-ios
- infra: bot-05-infrastructure
- quality: bot-06-quality-security

The lead seat takes intake from the queue. The five build seats implement changes in the paths they own. The quality seat reviews the work, and it maintains the gates and contracts.

03 / 09

What problem does it solve?

The desk starts from one stated failure mode: a coding agent reports success without executing the code. Every completion claim has to be backed by a verification receipt that records the commands actually run and their exit codes. The lead seat consolidates the build seats' receipts and the quality seat's review before it reports anything as done.

The second problem is ownership. Several agents in one repository need one clear owner for each path. The ownership.yaml manifest maps every path to exactly one bot, and a path that matches nothing is a build failure.

04 / 09

How are changes checked?

The gate runner, `ci/gates/run_all.py`, runs seven gates on every change:

- G-1, path ownership. No bot modifies a path it does not own, and no path is unowned.
- G-2, verification receipt. No completion claim is accepted without a receipt.
- G-3, committed secrets. No secret is committed.
- G-4, contract-first changes. A breaking contract change needs a version bump, a migration, and consumer acknowledgements.
- G-5 and G-6, rollback and destructive operations. A deploy needs a rollback plan that names the command, the recovery time, and the data implications, and a destructive operation must carry an approval identifier that the gate checks for.
- G-7, desk integrity. This gate reads the repository as it stands, not a diff, so it runs on every change.

A change is checked with the bot name, a base reference, and a receipt file, for example `python3 ci/gates/run_all.py --bot bot-01-systems-backend --base origin/main --receipt .receipts/bot-01-systems-backend/fix-api.json`.

05 / 09

How does the gateway work?

The desk gateway serves one MCP endpoint per seat, at `/mcp/<seat>`. It maps an OAuth token to one seat and serves only that seat's tool roster. Each roster holds the eight core tools that all seats share, plus that seat's own tools. Bots never join the tailnet and never hold an upstream credential. The gateway makes every upstream call for them.

The gateway applies four checks to each tool call:

- The arguments are validated against the roster's input schema, which rejects any field the schema does not declare.
- Arguments that have the shape of a credential are refused.
- Each call has a 20-second deadline.
- Each call is written to an audit trail as a redacted tool.call event.

Gated write tools carry the gate tags g5 or g6. A g5 tool requires the `rollback_plan` and `approval_id` fields. A g6 tool requires the `approval_id` field. Both fail closed when their upstream is not configured. Read tools fail open and return `not_configured` with a reason.

06 / 09

Current status

The Programming Desk is built and under test on a branch that has not been merged. Omega Labs commissioned the desk, and the status below is the one the repository supports.

Built

- Seven seats, mapped to bot identities in the gateway settings.
- A desk gateway with one MCP endpoint per seat, an OAuth flow that maps a token to one seat, a 20-second deadline, credential refusal, and an audit trail.
- The gate runner with gates G-1 to G-7.
- Contracts for the tool rosters that each seat is served.
- Test suites for the gates and for the gateway. This page does not report their results.

Configuration-gated

- Upstream adapters for Substrate, the agent bus, Greptime, Dragonfly, Hindsight, RAGFlow, Railway, Vercel, Greptile, GitHub, Play Console, and App Store Connect. Each returns `not_configured` until its environment variables are set.

Not yet built, or not on the desk path

- Solana devnet anchoring is local hashing. It makes no RPC call, so its records are not yet on-chain.
- The intake endpoint has a canary split. On the legacy route it answers with the state `legacy_stub` and HTTP 202, and it does not route the request to the canary gateway.
- The research and quantum modules are not the desk path.

07 / 09

Source notes

Paths are relative to the programming-desk repository.

- README.md
-

services/desk-gateway/README.md

- services/desk-gateway/src/desk_gateway/config.py
- services/desk-gateway/src/desk_gateway/upstreams.py
- services/desk-gateway/src/desk_gateway/server.py
- services/desk-gateway/src/desk_gateway/quantum_cv_anchoring.py
- ci/gates/run_all.py
- contracts/tool-rosters/lead.yaml

08 / 09

Frequently Asked Questions

Is the Programming Desk in use today?

It is built and under test on a branch that has not been merged. Its upstream integrations return `not_configured` until their environment variables are set, so no upstream service is connected by default.

Can one seat change files that another seat owns?

The path-ownership gate, G-1, rejects a change to a path that the bot does not own. A path that matches no owner in `ownership.yaml` fails the build.

Does the desk anchor its records on the Solana chain?

Not yet. The quantum memory module computes a local hash and makes no RPC call, so its records are not yet on-chain. That module is research code and is not the desk path.

How does a seat reach its tools?

Each seat has its own MCP endpoint. The gateway maps an OAuth token to one seat and serves only that seat's tool roster, which holds the eight core tools plus that seat's tools.

09 / 09

Sources

Programming Desk README
`programming-desk, README.md`

Desk gateway README
`programming-desk, services/desk-gateway/README.md`

Gateway settings and seat map
`programming-desk, services/desk-gateway/src/desk_gateway/config.py`

Gate runner
`programming-desk, ci/gates/run_all.py`

LEAD tool roster

programming-desk, contracts/tool-rosters/lead.yaml