The Neatlogs Activation Gap

Context

On 27 May 2026 we ran a structured first-impression study on `dev.neatlogs.com`. A fresh email signed up, completed the four-step onboarding, and toured every reachable surface in the empty-state app. The session was captured turn-by-turn — 26 screenshots with live commentary written in two passes (a "design system" pass and a "real user" pass), plus an outside-in investigation by a separate AI agent that examined only the public docs and GitHub.

This case study summarizes the single largest issue we found, the smaller issues that compound it, and a concrete proposal for a default first-run state. It is an artifact intended for the Neatlogs product team — short, specific, actionable.

The headline finding

Every product surface in Neatlogs is gated behind a Project. Onboarding never creates one.

The numbers tell it cleanly: the sidebar has eight navigation items; six of them render the same empty state — `No project selected. Select a project from the sidebar to view traces.` — for a user who has no projects. The seventh, `Investigate`, renders an AI search composer with three suggested prompts that all assume data the user does not have ("Find traces with high latency"). The eighth is `Settings`, which itself contains a `Project API keys` page whose body reads `No projects found.`

The four-step onboarding ends without asking the user to create a project. Step 4 is `Invite your team` with a primary CTA labelled `Skip` — a direct admission that the most prominent post-account action is one most users will dismiss. There is no in-app quickstart card, no SDK snippet, no link to documentation, no surfacing of the API key the user will eventually need. The post-onboarding URL is `/`, which routes to `Investigate` rather than to a "connect your first app" surface.

The copy itself works against the user. *Select a project from the sidebar* implies a list of projects the user can pick from — but the sidebar contains

no project list. The only project-adjacent element is a `Create project` button at the top, presented with the same visual weight as ordinary nav items. A reasonable user who has not been told otherwise spends thirty seconds clicking nav items, sees the same dead-end message six times, and concludes the product is broken.

Why this matters more than it looks

A typical activation funnel for a developer tool is: sign up → connect first project → emit first trace → "first value" moment. Neatlogs has the first step polished and the third step well-supported by the SDK, but the second step — the moment between account-exists and project-exists — is left to the user to discover unaided. That gap is exactly where activation drops are largest. It is also where the cost of fixing the issue is lowest, because the fix is mostly a UX rearrangement, not new infrastructure.

The outside-in agent investigation reinforces this. From the public surface, the data model exposed by the SDK is `api_key` → `workflow_name` → `trace` → `span`. The "workflow" concept in the SDK is what the UI now calls a "project." The Org and Workspace entities the UI asks the user to name during onboarding are invisible to the data plane — they are administrative scopes around an API key. From the user's first-five-minutes perspective, the entity that *does the work* is the project; the entities the onboarding spent two steps configuring are the ones that *don't*. The activation flow has its priorities inverted.

Smaller issues that compound the headline

These are not headline-worthy on their own, but each is a paper cut for a user already unsure they're in the right place.

The sidebar item **Activity** is collapsible but does nothing visible when clicked. A user who tries it cannot tell whether the feature is broken, gated, or simply uninhabited.

Investigate as the landing page is an opinionated choice — "ask, don't browse" — that fails in the empty state. The three suggested prompts are static and assume traces, detections, and a checkout flow that does not exist for a fresh user. The send button is correctly disabled, but the suggestions remain

clickable. A more state-aware empty state would either swap suggestions for setup actions, or hide the composer entirely until first data lands.

Workspace name is invisible after onboarding. The user typed `Production` as a workspace name during step 2; nowhere in the post-onboarding app is that label surfaced — no breadcrumb, no header, no setting. Either workspace is a real entity (and we should show it) or it isn't (and we shouldn't ask for it).

Terminology drift between SDK and UI. The SDK speaks `workflow_name`; the UI speaks `Project`. Both will continue to exist in the user's mental model — the SDK on day one, the UI on day two. Reconciling the names, or at least documenting the equivalence prominently, prevents a class of confusion that scales with audience.

A minor copy bug worth flagging: `Organization name` **and** `Organisation URL` sit next to each other in step 2 of onboarding, en-US and en-GB respectively. Easy fix, but it's the kind of detail that makes a careful reader doubt the rest.

A default first-run state, proposed

The smallest possible change that closes the activation gap is to **create a project automatically** at the moment the email is verified, and route the user to a first-run dashboard that shows that project, its API key, and a copy-pasteable SDK snippet — with the step-4 invite-team flow demoted to a dismissible card on the same screen.

Concretely, after step 3 (role qualifier) we would:

1. **Auto-create a default project** named after the user's role and workspace, e.g. `colas-default` or `production-default`. Make the name editable in the same view, but commit it eagerly so the user has something real from second one.
2. **Generate the project API key** and display it with a one-click copy.
3. **Render an SDK snippet** matching the role and language inferred where possible (Python today; explicit language picker until other SDKs ship). The snippet should be runnable verbatim — `pip install neatlogs && python -c " … "` — and produce a real trace in the user's project.

4. **Stream first-trace detection live** on the same page. When the SDK sends its first span, the page transitions from "waiting" to "first trace received" without a refresh. This is the activation milestone.
5. **Demote** `Invite team` to a dismissible card alongside the SDK snippet. Teammates can come at any time; data cannot.

Under this proposal, the user reaches a real "first value" moment within five minutes of email verification, with no further configuration choices. The IA the product already has — Traces, Detections, Evals, Analytics, Triage — fills with one row of meaningful content (the user's own trace), and the empty states across the rest of the app become invitations to do more, not dead ends.

Beyond the default project — what else belongs in the default state?

Some options worth weighing. We are not advocating all of these — they are listed so the conversation is concrete.

A **demo workspace toggle** for users who want to evaluate Neatlogs without instrumenting anything. A pre-populated read-only project of synthetic traces that demonstrates Detections, Evals, and Analytics with real-looking data. The trade-off is upkeep: demo data goes stale and starts to misrepresent the product if not curated.

A **quickstart card** on the landing page that survives the first-trace moment and converts into a checklist (`✓ Project created · ○ First trace received · ○ First detection configured · ○ First teammate invited`). When complete, it archived — but reachable from a permanent "Setup" entry in the sidebar.

State-aware suggested prompts in `Investigate`. When traces exist, the existing prompts are fine. When none do, swap to setup-oriented suggestions ("Show me how to instrument my first agent", "Open the SDK quickstart").

A **first-trace celebration** — a small but unmistakable moment when the first trace lands. Something between a toast and a confetti animation; enough to feel earned, not enough to feel infantilizing.

Role-conditional defaults. The role qualifier in step 3 already collects intent — `AI Engineer`, `ML Ops`, `Team Lead`, `Other`. The default project, the default detection rules, the default eval scorecard, and the dashboard tile

composition can each be tuned to the role rather than being identical for everyone. AI Engineers want to debug; ML Ops want to monitor; Team Leads want a digest.

An **in-app doc surface**. Even a single permanent ? button in the topbar that opens the Quickstart in a slideover would close the worst gap — the one where a confused user has nowhere to turn except back to Google.

A **public ingest API and language-agnostic SDK story**. This is bigger than the default state, but it changes what's possible inside the default state. Until JS/TS exists, "AI Engineer" onboarding is implicitly Python-only without saying so.

What we'd like to discuss next

Two questions for the product team. First — is the Org/Workspace/Project hierarchy the right model, or is it an over-fit of a future need? If projects are the only entity the data plane cares about today, every other layer is administrative overhead the user pays during onboarding for no immediate return. Second — what does Neatlogs want the first five minutes to feel like? Today the implicit answer is "look at our beautiful navigation"; the better answer might be "you sent your first trace, and here is the comment your future teammate will leave on it."

Both are answerable. We have the artifacts to back the conversation — the screenshots, the running commentary, and the outside-in agent's view — and we are happy to dig further on any thread the team wants to pull.