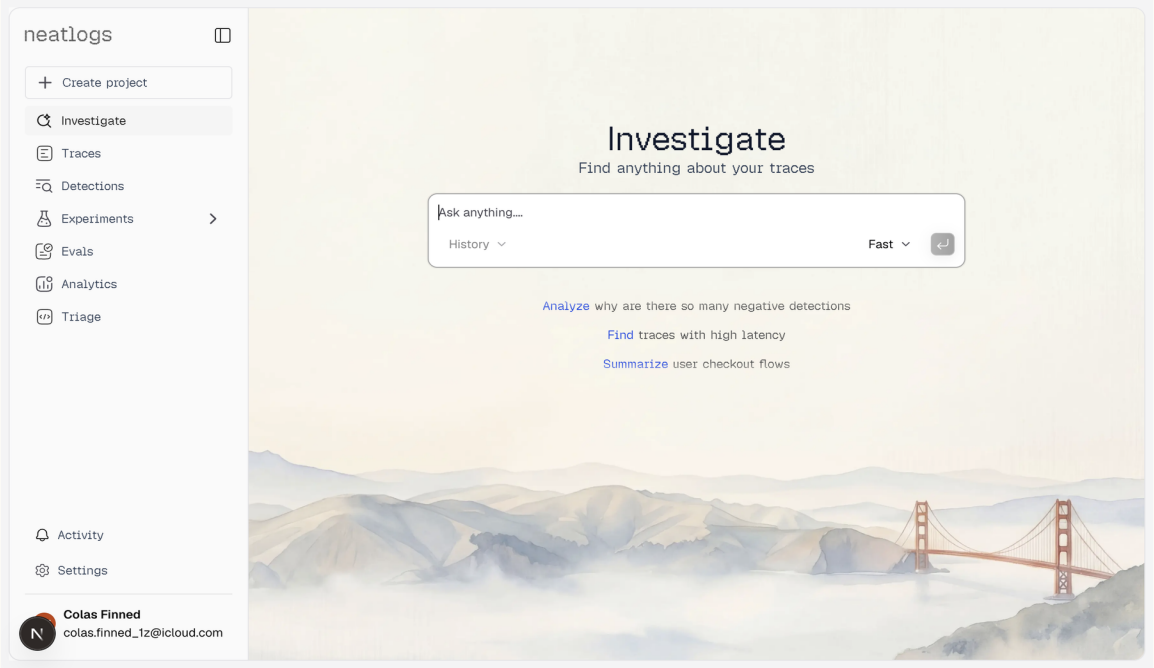
Build journal — feat/first-run-experience

> Same format as `signup-flow.md`: each entry is written **before** the next move, with screenshots and live thoughts. Retrospective entries are marked.

Setup (retrospective — captured after the fact)

- Cloned `neatlogs/neatlogs-app#dev` into `/Users/kasa/Downloads/Projects/neatlogs-app` via `gh repo clone`.
- Copied `.env.local` from `/Users/kasa/Downloads/neatlogs-app-dev-clone/.env.local` — points DATABASE/BACKEND at staging RDS (`neatlogs-postgres-staging.ctw4iwwyka5g.us-west-1.rds.amazonaws.com`) and Express backend `http://52.53.40.222:4100`). No docker, no local services.
- `pnpm install` clean (10s).
- Started `pnpm dev` in background (id `b0ldnxfby`); port 3000 was occupied by another node process so Next.js auto-fell-back to `http://localhost:3001`.
- Cut branch `feat/first-run-experience` from `dev`.
- Set local git identity to `neatsks <saurabh@neatlogs.com>` (this repo only — `--local`, not `--global`). Email already verified on the `neatsks` GitHub account so commit avatars will resolve.
- Logged into the local app with `colas.finned_1z@icloud.com` / `Neat-Test!2026`. Landed on `/` which rendered `Investigate` with no project selected — **identical activation gap reproduced locally against staging.**



signed-in empty state, local

Code map (live — 2026-05-27, before any change)

> Goal of this pass: locate everything I'd need to touch for the auto-create-default-project fix, **without changing anything yet**.

Onboarding lives in two places

There are **two** onboarding entry points, and the wizard `page.tsx` is not where the API calls happen:

1. `src/app/onboarding/page.tsx` (448 lines) — pure UI / view-state machine. Three flows `ftux` `invitec` `domain`) and a separate auth la `sign-in`, `forgot-*`, `signup-otp`). **No API calls.** When the FTUX user hits the Invite step's primary CTA or Skip, it just calls `goTour = () ⇒ router.push("/")`.
2. `src/app/onboarding/setup/_components/onboarding-setup-view.tsx` (330 lines) — this is where the work *commits*. Three `workspace` → `role` → `invite`. It calls:
   - `POST /api/organization` (idempotent — returns existing org if user already has one)

- `POST /api/team` (best-effort — 409 recovery via `GET /api/team` + name lookup; failures swallowed)
- `POST /api/team/invite` (best-effort, parallel `Promise.allSettled`)
- On `finish()` → `window.location.replace("/")`.

So the gap-fix needs to land **in `onboarding-setup-view`'s `createWorkspace` mutation** — right after team creation succeeds. That's the moment we have `organizationId + teamId` and a verified user.

Project create API (the call we need)

Endpoint: `POST /api/project` → proxied via `next.config.ts` rewrites to `${BACKEND_URL}/api/project` → handled in `backend/src/routes/project/route.ts:234` (`postHandler`).

Required body fields:

- `name` (string, trimmed, non-empty) — 400 if missing
- `teamId` (string) — 400 if missing

Optional:

- `seedDetections` (defaults to `true`; triggers fire-and-forget seed of default detection groups)

What the handler does:

- Generates a 32-char `nanoid` as `projectKey` (this is the API key the user will need).
- Inserts via `addProject( … )`.
- Fire-and-forget: copies LLM integrations from `SANDBOX_PROJECT_ID` if set.
- Fire-and-forget: seeds default detection groups + detectors (`seedWithRetry`).
- Returns `201` with the project minus `projectKey` (`stripProjectKey`).

Important: the response **strips the API key**. To get it back, we'd need a separate `GET /api/project/key?projectId=…` (line 356) which requires `requireAuth` only. So we either:

- Read the key from the create response *before* the strip (means a server-side change), or
- Create the project and immediately `GET /api/project/key` for it, or
- Surface the key on the first-run dashboard via the existing `Project API keys` settings page route.

Cleanest is probably option 2 (no backend contract change), but worth flagging.

Where the redirect target lives

After onboarding finishes, the user is se t to `/`. That s `src/app/(app)/page.tsx`, which is just:

```typescript
export { default } from "./ai-search/page";
```

— so `/` literally re-exports the Investigate page. This is *the* design choice that turns the empty-state into a dead end. Two options to fix:

- **A.** Change `(app)/page.tsx` to a router that pick `Investigate` if the user has data, else a `FirstRunDashboard`. Keeps Investigate as the "hot path" for returning users.
- **B.** Send the post-onboarding user to a dedicate `/welcome` route, leav `/` alone. New users would land on `/welcome` once, and from there always land on `/`. Lower risk (no behaviour change for existing users), but means a new persisted "has-finished-first-run" flag.

Leaning **B** for v1 — smaller blast radius, easier to revert, no risk of breaking returning users' muscle memory.

Empty-state copy that's bleeding

Six pages render the `No project selected` empty state — they all live under `src/app/(app)/`:

- `traces/page.tsx`
- `triage/page.tsx`
- `detections/page.tsx`
- `prompts/page.tsx`
- `analytics/page.tsx`
- `dashboard/_components/sidebar/nav-users.tsx` (sidebar account row, separate concern)

Plus th `Select a project from the sidebar` varia `traces` and `prompts`. None of these need to change as part of the v1 fix — once the user has a default project, they all light up — but they're the surfaces a `feat/state-aware-empty-states` follow-up branch would touch.

What I will NOT touch in this branch

- The wizard UI in `onboarding/page.tsx` — copy/animation only, fix lives in `setup/`.
- The Express backend project route — no contract changes if we use the `GET /key` workaround.
- Empty-state copy on the six pages — separate branch.
- Terminology drift (`workflow_name` ↔ `Project`) — separate branch, touches SDK.

Plan, before I write any code

The minimum-viable activation-gap fix for this branch:

1. **In `onboarding-setup-view.tsx`, inside the `createWorkspace` mutation**, after team creation succeeds and we have `{ organizationId, teamId }`:
   - `POST /api/project` with `{ name: "<draft.orgName>-default", teamId }`. Best-effort: log + continue on failure (mirrors how team-create handles failures today). Stash `defaultProjectId` in component state.
   - `GET /api/project/key?projectId=<defaultProjectId>` to recover the projectKey. Stash it.
2. **Replace step 4 (Invite)** with a new `FirstRunDashboard` step that shows:
   - Project name (editable, posts a PATCH on commit) — single field.
   - API key with copy-to-clipboard button.
   - Python SDK snippet, role-conditional language label (Python-only today).
   - "Waiting for your first trace…" placeholder.
   - Demoted `Invite team` card (collapsed by default).
3. **Stream first-trace state**: poll a "do you have any traces" endpoint every ~3s until one shows up, then transition the page state. Out of scope to ship this branch — stub it as a TODO with a manual refresh button as v0.

For v1 of this branch I'll ship items 1+2 and leave the live-stream as a follow-up. Items 3+ from the case study (demo-workspace toggle, persistent quickstart card, role-conditional defaults, in-app `?` help) all stay parked.

Decisions locked (2026-05-27)

Decision	Choice
Default project name	`<orgName>-default` (lowercase, hyphenated). Inline-editable later.
Post-onboarding landing	New `/welcome` route. `/` left untouched.
API-key recovery	Read directly from `POST /api/project` response. **Corrected 2026-05-27 after end-to-end test:** the server strips `projectKey` on the create path too (`backend/ ... /project/route.ts:283 — stripProjectKey(newProject)`). Use the follow-up `GET /api/project/key?projectId=` round-trip; it requires only auth + team membership. The existing `create-project-dialog.tsx` `data.projectKey` check was effectively dead on this branch.
First-run-done flag	**localStorage for now**; long-term home is the user record on the server. Must be flagged in: (a) a code comment at the localStorage call site, (b) the commit message, (c) the PR description.

Implementation plan, locked (revised after pattern scan)

Key revision: the existing onboarding wizard (account → verify → workspace → role → invite) is **untouched**. The Invite step keeps working — Skip is no longer scary because users land on `/welcome`, not on Investigate. The `/welcome` page doubles as a persistent **"Setup"** sidebar entry, not a one-shot screen.

1. **Auto-create the default proje** `onboarding-setup-view.tsx`, inside `createWorkspace`'s mutation, immediately after team creation. Best-effort v `try/catch` like team-create — failures don't block onboarding. Name: `<orgSlug>-default`.
2. **Read the API key** `GET /api/project/k` `POST /api/project` *strips* the key (server-side, on every path). End-to-end test confirmed: create returns 201 with `projectKey`. We follow up with `GET /api/project/key?projectId=` (dedicated key-fetch endpoint, auth + team membership). One extra round-trip, no backend change. The earlier note that "create response carries the key" was wrong — the existin `create-project-dialog.t` `data.projectKey` check is dead code on this branch.
3. **New** `/welcome` **route** under `src/app/(app)/welcome/page.tsx`. Surfaces:
   - Project name (rename inline → PATCH) + API key with one-click copy.

- SDK snippet panel: Python tab active, JS tab "coming soon" disabled.
- **"Send a test trace" button** — present but disabled with "coming soon" tooltip in v1. Wired in v1.1 (Sentry's `firstEventFooter` pattern: pulsing "Waiting..." → green "Received", button copy swaps to "Take me to my trace").
- Activation checklist (4 ite `Project created · First trace received · First detection configured · First teammate invited`. Each open item links to the relevant flow.
- "Need help?" footer: Quickstart link, Talk-to-us, `?` shortcut hint.

4. **Sidebar entry** `Setup` wit `1/4`-style badge (completed/total — Sentry progress-ring style). Visible until checklist complete; collapses into Settings → Setup when done.
5. **Modify** `finish()` **in onboarding-setup-view.tsx** — inste of `window.location.replace("/")`, push t `/welcome`, **gated by localStorage fla** `neatlogs:firstRun:<userId>`. Returning users still hit `/`. **TODO comment at flag site:** "First-run flag in localStorage — long-term move to user record server-side; tracked in PR description."
6. The `/welcome` route itself is **not** first-run-gated — anyone can revisit via the sidebar. localStorage flag only governs the *auto-redirect* on completion of onboarding.

Pattern sources (from `setup-pattern-scan.md`):

- Sentry → test-event button + status indicator + sidebar progress ring.
- PostHog → number-badge style on sidebar items, hogfetti-on-completion as a future-celebration cue.
- Stripe → flagged as anti-pattern (browser-only state, 8-item slog, API key punted to settings).

Out of scope for this branch (parked, but tracked):

Inside the v1 dashboard but stubbed:

- *Live first-trace stream* — page transitions from "waiting" to "first trace received" without refresh. v1 ships with a manual refresh button instead.

Separate branches, sized small to medium:

- *State-aware empty states* on the six gated pages (Traces, Triage, Detections, Prompts, Analytics, Dashboard sidebar). Once a project always exists, these stop bleeding — but the copy `Select a project from the sidebar`

- `to view traces` should still be replaced with setup-oriented copy until the first trace lands.
- *State-aware suggested prompts in Investigate* — when no traces exist, swap the three "Find traces with high latency" prompts for setup-oriented ones (`Show me how to instrument my first agent`, `Open the SDK quickstart`).
- *In-app `?` doc surface* — single permanent topbar button opening Quickstart in a slideover. Closes the "user has nowhere to turn but Google" gap.
- *First-trace celebration* — between a toast and confetti. Earned, not infantilizing.
- *Locale typo* — en-US `Organization name` next to en-GB `Organisation URL` on workspace step. Tiny PR.
- *Terminology drift* — SDK speaks `workflow_name`, UI speaks `Project`. Either reconcile names or document the equivalence prominently.

Bigger initiatives (separate epics, not single branches):

- *Persistent quickstart card / checklist* on the landing page — `□ Project created · □ First trace · □ First detection · □ First teammate.` Vanishes when complete, archived but reachable from a permanent `Setup` sidebar entry.
- *Role-conditional defaults* — the role qualifi `AI Engineer / ML Ops / Team Lead / Other`) already collects intent. The default project, default detection rules, default eval scorecard, and dashboard tile composition should each be tuned to the role. AI Engineers want to debug, ML Ops want to monitor, Team Leads want a digest.
- *Demo workspace toggle* — pre-populated read-only project of synthetic traces demonstrating Detections, Evals, Analytics with real-looking data. Trade-off: upkeep, demo data goes stale.
- *Public ingest API + language-agnostic SDK* — bigger than the default state but changes what's *possible* inside it. Until JS/TS exists, "AI Engineer" onboarding is implicitly Python-only without saying so. Touches SDK roadmap, not just the app.

For each parked item, when we open its branch, the case study and usability report are the source-of-truth briefs.

Pre-flight: security + repo hygiene check (2026-05-27)

Before any code change, audited risk surfaces:

Repo hygiene:

- Org-owned repo, so artifacts from this investigation (journal, screenshots, PDFs, pattern scan) **stay outside** in `nl-setup/`. None enter `neatlogs-app/`.
- No new top-level docs, no scratch files, no commented-out blocks. Match existing Biome/ultracite formatting; let the linter do the work.
- `.env.local` already in `.gitignore`; no `.DS_Store` in commits.
- One logical change per commit. Surgical diffs.

Security review:

Surface	Risk	Mitigation
`POST /api/project` from client	Auth, idempotency	Endpoint already requires `requireAuth + requireTeamMembership + denyIfSuperUser`. Existing `workspaceCreateInFlightRef` guards double-submit.
API key in client memory	Logged, leaked, persisted accidentally	Read from create response; keep in component state only; never log; never put in URL/query/toast/error message; never sync to global store.
API key display	Shoulder-surf during demos	**Reuse existing** `getMaskedKey()` **helper** from `create-project-dialog.tsx:195`. Show last-4, eye icon to reveal full key.
`/welcome` route	Unauthenticated access	Lives under `src/app/(app)/`, inherits the existing session-required layout.
URL params	Secret leakage	No project ID or key in URL. Derived from session/store.
localStorage flag	Cross-account contamination on shared browser	Key namespaced as `neatlogs:firstRun:<userId>`. Value is just `1`. No PII, no token.
Test-trace button (v1.1)	SSRF, log injection via client-supplied content	Server constructs the trace; client only triggers. Rate-limit per project server-side.
Proxy boundary	Cookies/CSRF/auth-header drift	All traffic stays through `/api/*` rewrite. No direct `BACKEND_URL` fetches from browser.

Useful findings from pre-flight grep:

1. **No third-party analytics SDK** is wired (`package.json` has no PostHog / Amplitude / Sentry / Datadog / Mixpanel / Segment / Heap / Rudderstack). One less leak vector to worry about.
2. `POST /api/project` **returns** `projectKey` **on the create path** — verified in `src/components/overlay/create-project-dialog.tsx:52-` (`if (data.projectKey) setProjectKey(data.projectKey)`). The `stripProjectKey` server-side helper only applies to list/get responses. **Plan revised: drop the pla** `GET /api/project/key` **round-trip.**
3. `getMaskedKey()` **helper already exists** at `create-project-dialog.tsx:195`. We reuse rather than reinvent.
4. **Server-side** `fetchProjectsForApiKeys` explicitly comments "Return only necessary data, excluding the actual keys" — the team treats keys carefully. We follow that posture.

—

End-to-end test (2026-05-27, post-commit)

Signed up `paradox_agates8z@icloud.com` against the local app on `:3001` (the working repo), walked the full sign-up flow (account → OTP → workspace → role → invite/skip), landed on `/`. The signup correctly used `/sign-up` (the new flow), not `/onboarding/setup` — surfaced a bug: my first commit only patched the legacy `onboarding-setup-view.tsx`, so the new flow still had the activation gap.

Findings while testing:

- Two onboarding flows exist in the codebase:
  - `/sign-up` — `src/app/(auth)/sign-up/_components/sign-up-flow-view.tsx` (the live flow new users hit)
  - `/onboarding/setup` — `src/app/onboarding/setup/_components/onboarding-setup-view.tsx` (legacy / sign-in-no-org branch)
  - Both call POST `/api/organization` `/api/team` and do best-effort recovery; both needed the default-project block.

- **Correction to "decisions locke** `POST /api/project` returns 201 *without* `projectKey`. Confirmed by direct API call in the browser console while signed in as Paradox:
  - `POST /api/project { name, teamId }` → 201, body ha `_id` but no `projectKey`.
  - `GET /api/project/key?projectId=…` → 200 `{ key: "<32 chars>" }`.
  - Server `stripProjectKey` is applied on the POST path too (line 283 of `backend/src/routes/project/route.ts`), not just list/get as previously claimed.
- The `existin create-project-dialog.tsx:52` code path that reads `data.projectKey` from POST response is effectively dead on this branch — explains why the create dialog doesn't surface the key today.
- Wrong-port hazard: a stale `pnpm dev` from the reference clone was still listening on `:3000`. NextAuth cookies are scoped to host (not port), so a session created on `:3000` was visible on `:3001` and vice-versa. Worth noting if I ever go back to a clean DB test.

Resulting commits:

1. `2f2fd564 feat(onboarding): auto-create default project on workspace setup` — patched only the legacy view; assumed key-on-create.
2. `f7d00587 fix(onboarding): recover projectKey via GET /api/project/key` — corrected key recovery + applied the same block to the new sign-up flow.

—

Next entry: implementation of `/welcome` *+ sidebar Setup, written as I go.*

Live-tail + first-trace celebration — parked (2026-05-28)

Decided to ship `/welcome` *without* the live "first trace received" transition for v1, but locked in the design so the next branch can drop straight in.

What "live tail" means here: the moment the user runs the SDK snippet on their machine, th `/welcome` page should flip the *First trace received* checklist row to done, swap the disabled "Send a test trace" CTA for "Take me to my trace →", and play a small celebration. No reload, no manual refresh.

Audit of what exists in the repo today:

- Trace fetching is `useQuery` only — no `refetchInterval`, no SSE on the trace path.
- An SSE channel exists at `/api/v1/notifications/sse` (used by the in-app notifications context). It's not currently used to push trace events.
- Trace ingest goes SDK → Express backend → Postgres + ClickHouse. No publish hook on first-trace-per-project today.

Three options, in increasing order of cost:

1. **Polling (recommended for v1.1)** — On `/welcome`, while `trace.done` `===` `false`, run a tight `GET /api/traces/v3?projectId=…&limit=1` on a 3s `refetchInterval`. Stop when count > 0. No backend change. Sentry's "Waiting for your first event" works exactly like this; users tolerate 3s in this context because they're staring at the page.
2. **Reuse notifications SSE** — Backend emits `first-trace-received` event when a project's first trace lands. Frontend subscribes. Instant, but needs a small backend change and a project-scoped filter on the existing SSE.
3. **Dedicated traces SSE** — New `/api/traces/stream?projectId=` endpoint. Real backend work. Worth doing later anyway because a live-tail in `/traces` is a feature several competitors have, but out of scope here.

Celebration treatment: not canvas-confetti — too cartoony for an observability product. Instead: a small motion overlay (we already have `motion/react`) that fades in the words *"first trace received"*, the trace's `traceId`, and a primary CTA *"Take me to my trace →"* linking to `/trace/:id`. The CTA on the snippet panel becomes the same link. Same easing as the existing onboarding step transitions.

State surface to drive this: the `useActivationChecklist` hook already exposes `defaultProject.numOfTraces`. Next branch just needs to add a polling option to the project query (the hook is shared with the sidebar so the sidebar `1/4` badge will tick to `2/4` automatically). The celebration is a one-shot React effect keyed off the `false → true` transition of `trace.done`.

Tracking: to be done in a separate branch (`feat/welcome-live-tail`) once the activation-gap PR merges, so reviewers can read the two changes independently.