Nathan Lee | JotPsych Intern Trial Project

What I Built

My harness is a CLI that scaffolds structured Claude Code workspaces from templates. It attempts to solve the issue of having repetitive rebuilds of the same Claude-assisted workflows across every clinic engagement — template training, payer extraction, review synthesis. Each time, someone rebuilds the setup from scratch. This harness makes that first setup shareable and repeatable.

You run one command, get a fully wired project: folder structure, CLAUDE.md with domain context, Claude agents, Notion sync config, tool permissions. This allows the team to have a standard structure while also having the capability of tailoring it to their clinic's needs.

The Core Design

Every harness project separates two things:

- Engine — the workflow scaffolding (CLAUDE.md, agents, configs, scripts). Version-controlled, shareable, lives in templates/ and goes to GitHub.
- Fuel — the actual data (patient notes, PDFs, outputs, API keys). Gitignored, stays local, never leaves the machine.

This is enforced at the filesystem level. I made it so people can have a standard templates folder tracked, while when you ask claude (or run the command yourself) to use an existing template it creates it in a new folder called projects/ which is in the .gitignore. That allows it so users are able to reuse templates all under the harness folder, export their current template (while cutting out all of the extra notes etc) all in the harness folder. The beauty of this architecture is that it allows users to only edit the templates directory (when they want to add their new project as a template) and that allows them to push to the repository for other people to use their templates. This will minimize merge conflicts since people will upload new templates only to the templates folder. On top of this, even the sensitive data inside each project — clinic notes, PDFs, outputs, API keys — is gitignored by default, so nothing private ever touches the repo.

Key Design Decisions

1. Templates are just directories

No special format, no registry, no build step. A template is a folder with a harness.json manifest and an engine/ subdirectory. Improving a template is a git commit. This kept the surface area small and the tool auditable.

2. Variable substitution is intentionally dumb

I made it so the {{clinicName}} gets replaced with a string. No conditionals, no loops. If two use cases are structurally different, they should be different templates, not one template with branching logic. Claude handles conditional reasoning at runtime via CLAUDE.md instructions.

3. MCP-only Notion integration

I initially considered a dual approach, a programmatic Notion API call at scaffold time plus MCP at runtime. I removed the programmatic layer and decided MCP-only means no key management, no CLI-level API calls, and Claude as the sole operator of Notion. The tradeoff is it only works when Claude is running the session — which for this workflow is always true. This decision also led to the idea of a notion-sync agent. Each template ships with a notion-sync agent that handles all Notion coordination. The agent is smart about hierarchy — it first searches the shared JotPsych workspace for an existing customer page matching the clinic name. If it finds one it adds the new harness page under it, if not it creates a new customer page first and then adds under that. This means two completely separate repos working on the same clinic will automatically share a Notion parent without any manual coordination — the agent figures it out by searching first. No duplicates, no extra setup.

4. Export with human-review checklist

harness export handles the 80% automatically — reads gitignore, copies engine files, reverses variable substitution, infers secrets. For the remaining 20% it prints an explicit checklist: what's empty, what's auto-generated, what still needs a human. The goal was a 5-minute review, not a hunt.

Challenges I Hit

1. Claude creating templates instead of projects

Early on, when I'd tell Claude "set up a harness for Valcourt," it would create a new directory in templates/ instead of scaffolding into projects/. The fix was adding an explicit intent guide to CLAUDE.md that disambiguates the two paths. Although it was a small change it made a big difference.

2. Making collaboration work without live infrastructure

The hardest design problem was coordination — multiple people working on the same customer from completely separate local repos with no shared database, no webhooks, no live sync. The naive solution would be a server that tracks state. Instead I used Notion as the coordination layer. The notion-sync agent searches for an existing customer page before creating anything — so if Jackson and Nate are both working on Valcourt from different machines, they automatically end up under the same Notion parent without ever talking to each other. The hierarchy builds itself. The tradeoff is it only works when Claude is running the session, but for this workflow that's always true.

3. Getting the Notion hierarchy right

The first pass had the agent creating harness pages directly under the workspace page. What we actually wanted was a workspace → customer → harness pages. The fix was making the agent explicitly capture a customerPageId from step 2 and use it as the parent in step 3 — with a hard note in the instructions: "NOT workspacePageId."

What I'd Build Next

If I had more time the first thing I'd add is a remote template registry — right now templates live in the same repo as the tool which means to get a new template you have to pull the whole repo. The better version is a separate repo (something like jotpsych/harness-templates) that harness can pull from with a simple command. That way the tool and the templates are on independent release cycles and anyone on the team can contribute a template without touching CLI code (although this would add a layer complexity that may not be needed). I'd also look at pulling Notion context into the scaffold automatically — so when you spin up a new harness for Valcourt, it goes and grabs whatever JotPsych already knows about Valcourt from Notion and drops it into the CLAUDE.md. Right now that's a manual step. That said, both of these would benefit from understanding how the team actually uses Notion internally — the right database structure, naming conventions, and page hierarchy would significantly shape the implementation. User feedback after real usage would drive most of these decisions more than anything I could design upfront. I think a lot of these things may work in my head but depending on the workflow of the team I would change to support their needs as I think perspective is everything.

How I'd Measure Success

Adoption is a big measure, along with how easy it is to use when collaborating with people. I did aim for developing from a collaboration standpoint and if people don't use it, nothing else matters. Specifically I would look for how many scaffolds happen per week, whether the same template gets reused across multiple clinics (so we don’t have to have like 1 million different templates and which means the engine/fuel separation is actually working), and whether non-engineers can get from zero to a working Claude session without asking for help.