

Phillip Jan Moran

AI Engineer at Boost Capital. Full-stack across TypeScript and Python, building agent tooling, MCP servers and the products around them.

Screenshots are from the running apps. Work screens use sample data, with internal names, accounts and people removed; anything that holds client data is described, not shown.

`moran.phillipjan@gmail.com`
`github.com/sipelepis`
`linkedin.com/in/lepmoran`
`lepskie-dev-black.vercel.app`

PROFESSIONAL — BOOST CAPITAL

VerifyIQ Portal	02
Boost Library	03
Boost Modules	04
Bogart the Taskmaster	05
Bogart, the character	06
Tyr	07

PERSONAL PROJECTS

Viz Cancer	08
rag-pet	09
Buhat	10
Roadmaps	11
Bayanihan	12
Liwaliw	13
Gastador	14
More: Jobtailor, Laro Tayo, tradedesk, Scribe, Kalawakan, Portfolio site	15

BOOST CAPITAL · AI ENGINEER · MAR 2026 — PRESENT

VerifyIQ Portal

Multi-tenant provisioning and management for an AI document-fraud detection product sold to financial-services tenants.

I co-architected the platform that takes a new tenant from cloud infrastructure to per-tenant feature configuration, then spent most of my time making it cheaper to run, faster to ship and harder to break.

- Led security hardening across the stack: dependency CVEs, PII redaction in error monitoring, endpoint access control, API-layer feature flags and SAST remediation.
- Repaired the tamper-evident audit log and the admin session lifecycle.
- Shipped core tenant flows — application intake, document upload, PDF preview — that feed the LLM fraud pipeline, plus the component library both frontends share.
- 866 commits since April 2026.

NestJS React TypeScript Nx Cloud SQL Cloud Run
Terraform GitHub Actions Sentry

No screenshots: the portal holds client and applicant data.

BY THE NUMBERS

18

live tenant backends moved to shared, region-matched Cloud SQL tiers with zero downtime

20 + 33

database servers retired, and orphaned secrets removed

4 → 1

repositories folded into one Nx monorepo: 8 apps, 10 libraries, shared auth and NestJS libs

26 → 12 min

PR pipeline time; deploys now fail loudly on migration errors

BOOST CAPITAL · AI ENGINEER · INTERNAL PLATFORM

Boost Library

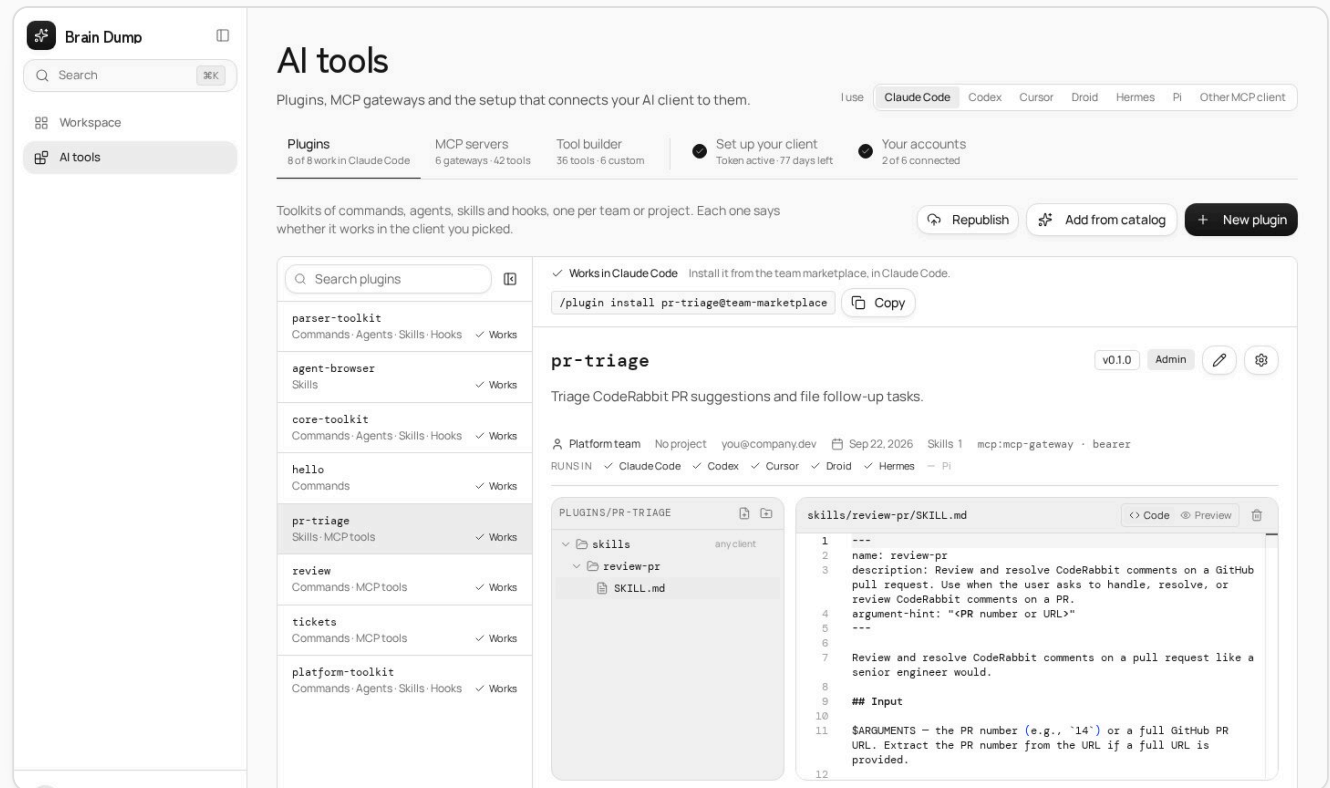
The internal hub that connects Claude Code and other AI clients to company systems: plugin marketplace, MCP gateway, tool builder and knowledge hub.

Engineers install team plugins from a catalog, connect their own ClickUp, GitHub and Google accounts once, and call company tools through a hosted MCP gateway where every action runs as the person who asked. I'm the primary author (394 of 433 commits).

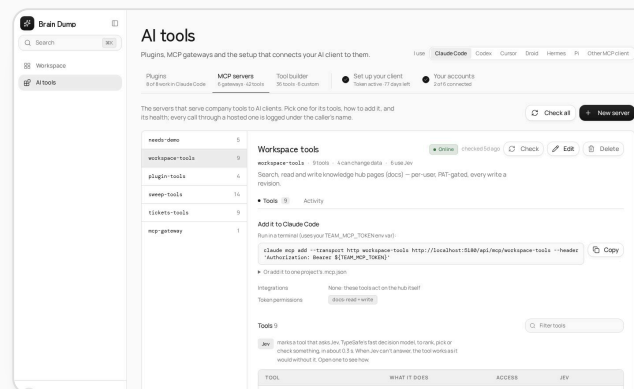
- Database-backed plugin marketplace (commands, agents, skills, hooks) that publishes itself as a Claude Code marketplace, and says per plugin and per file whether it runs in Claude Code, Codex, Cursor and others.
- Hosted MCP gateway on NestJS: personal access tokens with read/write scopes; each tool call uses the caller's own credential and is logged under their name.
- Tool builder that turns any internal HTTP endpoint into a typed MCP tool from a declarative spec.
- One-click OAuth (with refresh tokens) for ClickUp, GitHub and Google; credentials live in Secret Manager, never in the database.
- Knowledge hub replacing ClickUp Docs and Confluence: spaces, page trees with revisions, per-space roles and an activity log.

React 19 NestJS 11 Prisma PostgreSQL MCP
OAuth 2.0 Nx Cloud Run Terraform

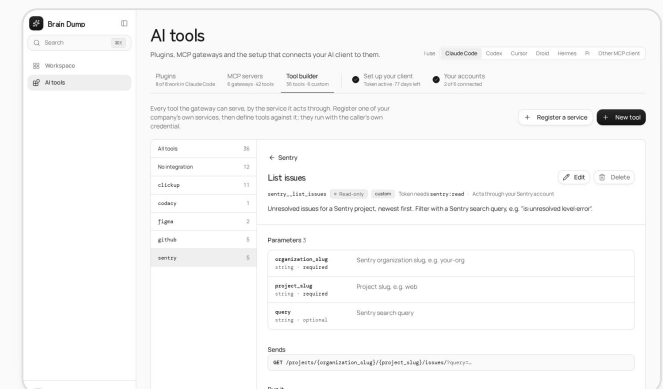
Local dev instance (in-app name: Brain Dump). Internal names and accounts are generalised in these screens.



Plugin marketplace: a plugin's files, which clients it runs in, and its install command



An MCP gateway server: install command, token permissions and its tools



Tool builder: an HTTP spec with typed parameters becomes an MCP tool

BOOST CAPITAL · AI ENGINEER · SHARED PACKAGES

Boost Modules

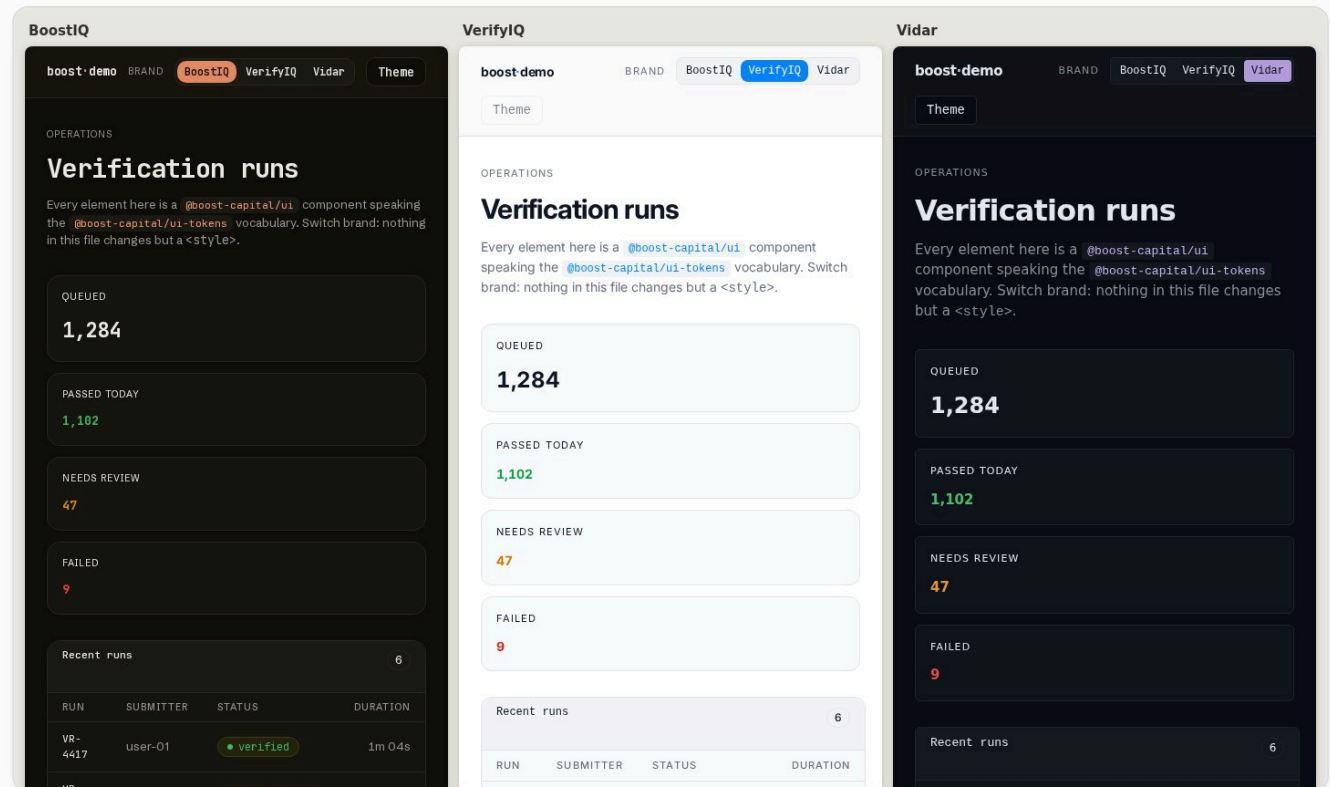
One design-token package with three brand themes and a 27-component React library, published as private @boost-capital npm packages.

Boost's apps had each been hand-copying the same UI code until it drifted. I unified two forks into versioned packages: components speak only the token vocabulary, so swapping one stylesheet re-brands the whole app.

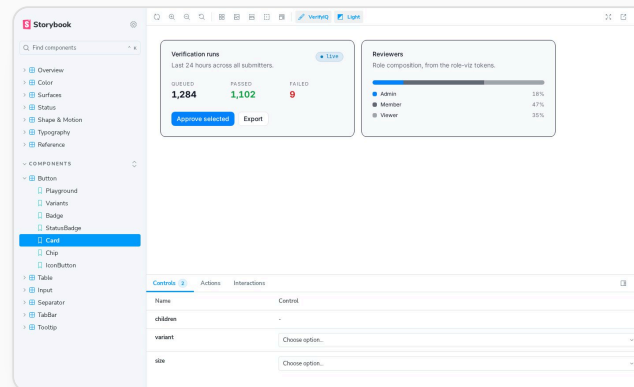
- Three-layer tokens: ~30 primitives per brand, a derived layer built with OKLCH color-mix, and a Tailwind v4 @theme mapping.
- 27 components (Button, Dialog, Select, Table, Tooltip...) documented in Storybook with brand and light/dark switches.
- Nx module boundaries enforced on scope and type — verified by adding forbidden imports and watching lint reject them.
- Released per package by GitHub Actions with npm trusted publishing (OIDC, no long-lived token), plus an install-and-render canary and a consumer app that fails if a package resolves from anywhere but the registry.

React 19 Tailwind CSS v4 OKLCH Radix UI

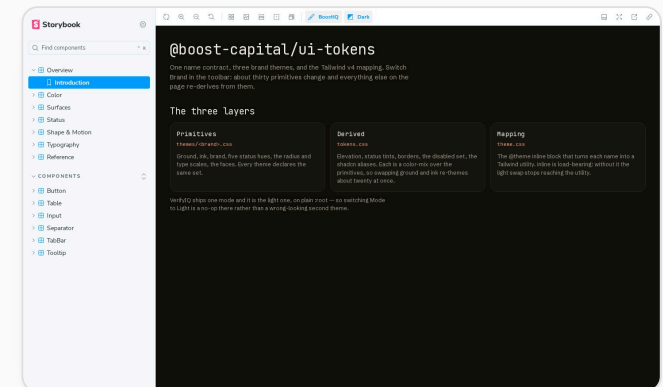
Storybook 9 Nx npm GitHub Actions



The same consumer screen in three brands — only the theme stylesheet changes



Storybook: Card in the VerifyIQ theme with live controls



Storybook: the token package's three layers

BOOST CAPITAL · AI ENGINEER · DESKTOP QA AGENT

Bogart the Taskmaster

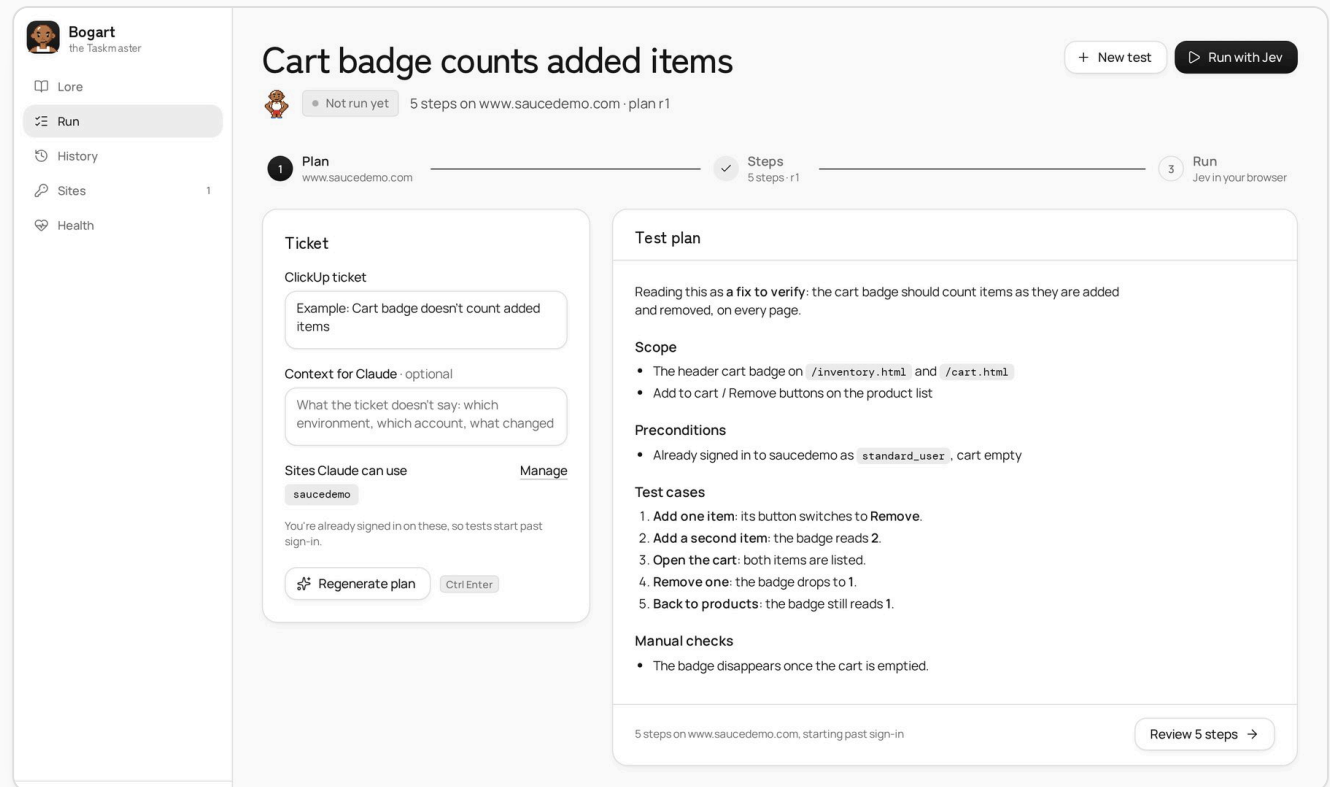
A desktop QA agent: give it a ClickUp ticket, Claude writes the test plan, and a fast decision model runs it in your own signed-in browser.

An Electron app for the step between “fixed” and “sent to QA”. It reads the ticket, its comments, the commits and PRs tagged with it and earlier runs, then drives each step over the Chrome DevTools Protocol and gives every check a verdict, a probability and a screenshot.

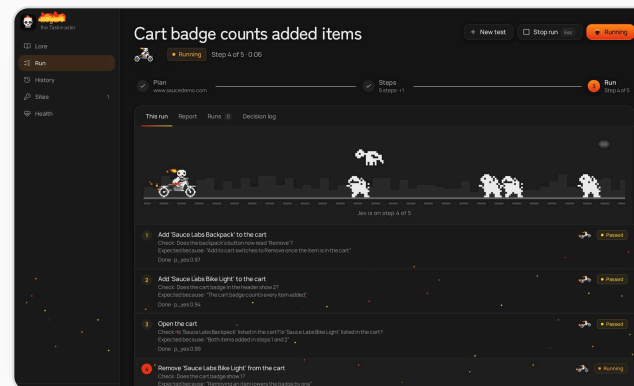
- Pipeline: ClickUp ticket → `claude -p` with a JSON schema writes a structured plan → Jev (TypeSafe’s ~300 ms decision model) runs each step in the user’s real browser.
- Secrets stay on the machine: saved logins are encrypted with Electron safeStorage, and the models only ever see `{{site.password}}` placeholders.
- Safe on real accounts: runs hand off to the user for 2FA, SSO and irreversible actions, and a failed run can continue from step N.
- Full history in Postgres — content-hashed plan revisions with step diffs, per-step evidence, and a run report built without a model call.

Electron Node.js Claude Code CLI Playwright / CDP
PostgreSQL ClickUp API WebGL Canvas

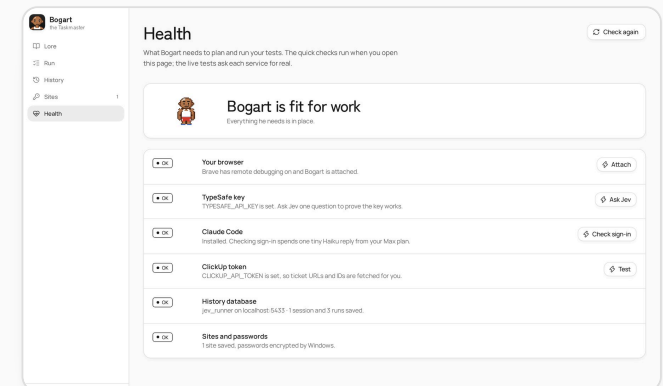
Shown with a sample test against the public demo shop saucedemo.com, not a real ticket.



Plan stage: the ticket on the left, Claude's test plan on the right



Overdrive mode mid-run: each check scored live



Health: setup checks for every dependency

BOOST CAPITAL · AI ENGINEER · BOGART, CONTINUED

Bogart, the character

The app's mascot is drawn entirely in code.

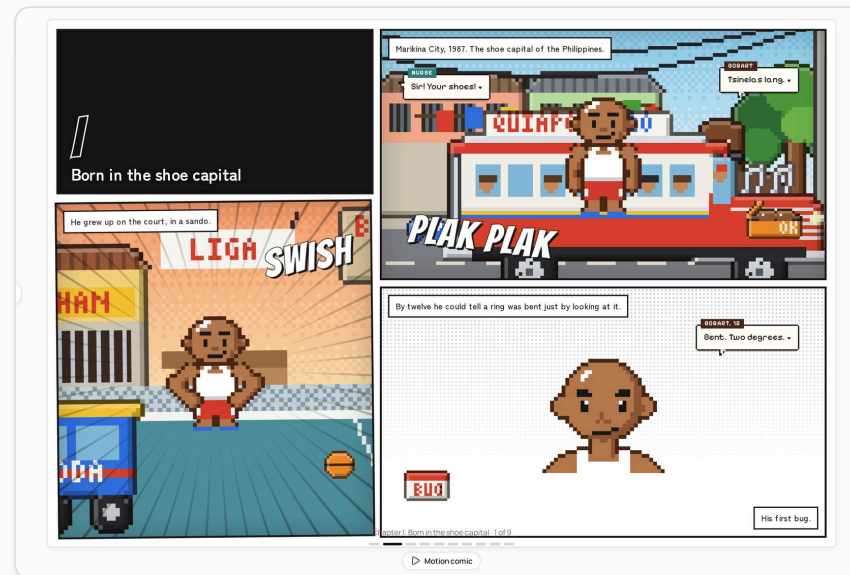
Bogart is a 32×32 pixel-art character from a procedural sprite pipeline (~1,400 lines): bodies are hand-drawn grids, arms are posed from shoulder, elbow and hand joints, and a build step emits each distinct frame once with generated CSS keyframes. He acts out every app state — thinking while Claude plans, dancing while the browser runs, a verdict pose at the end.

- A ~4-second intro, a WebGL pixel-fire effect, and a canvas fight scene that plays while you wait.
- An eight-chapter manga “Lore” page with a live canvas street-corner cover that changes between morning and night.
- Designed with the Akashi design system under a strict CSP: no inline script or style, no build step.

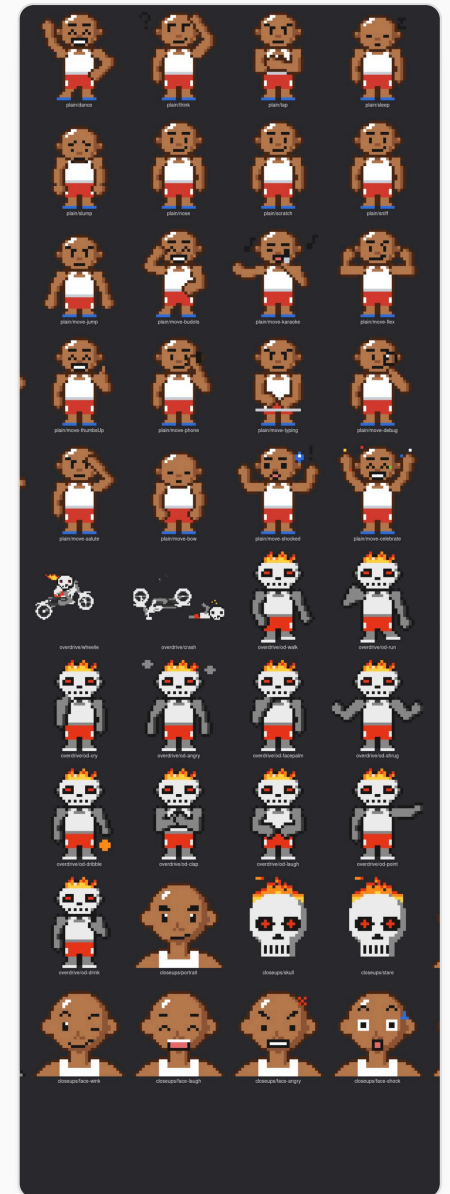
Canvas WebGL CSS keyframes ImageMagick Node.js



The launch intro



Lore, chapter I



Sprite contact sheet exported from the pipeline

BOOST CAPITAL · AI ENGINEER · AI HARNESS FRONT-END

Tyr

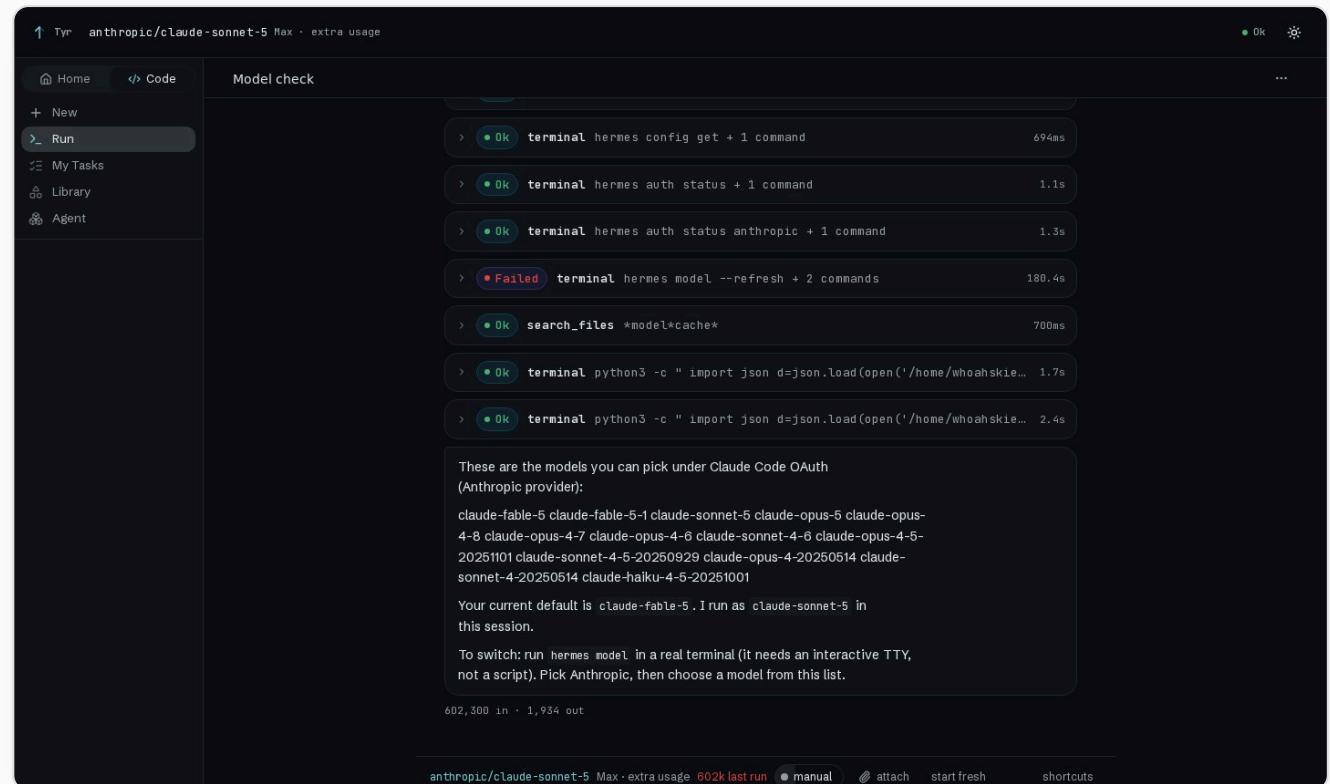
The operator console for an autonomous coding harness built on the Hermes agent: streamed runs, readable tool calls, and a human approval gate before the agent touches anything real.

Hermes supplies the agent loop, tools and API. Tyr is what a person needs to supervise it on real ticketed work: a live transcript with tool-call cards, an approval gate that ranks and explains risky commands, and pages for the agent's capabilities, memory, artifacts and scheduled jobs.

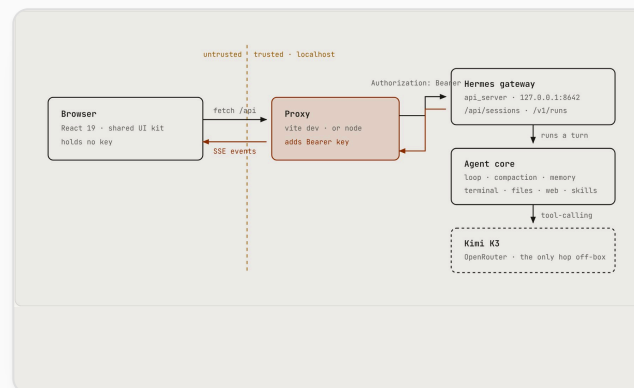
- Approval gate that ranks each request: independent detectors must agree before low and medium signals escalate, while trusted rules can fire alone — so no model can talk a dangerous command down.
- Security boundary: the gateway key authorises a shell-capable agent, so it lives only in a dev-server proxy; verified with curl (401 direct, 200 through the proxy).
- Cut the per-turn context floor by 31% (28,792 → 19,773 tokens) by trimming unused tool schemas and skills, measured with the same probe before and after.
- Accessibility pass: pointer targets under the WCAG 2.2 24 px minimum went from 218 in one view to zero across all seven.

TypeScript React 19 Tailwind CSS v4 TanStack Query

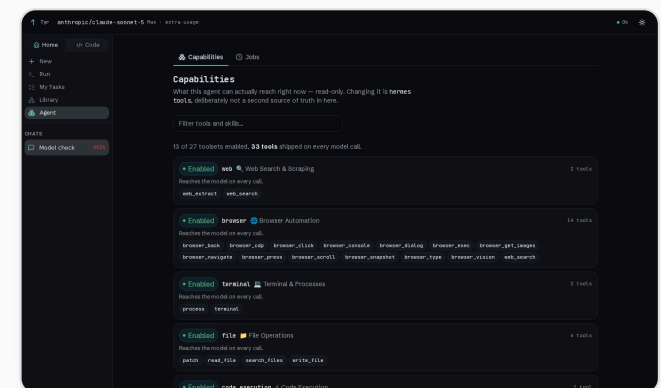
SSE node:sqlite Hermes Agent Nx



A run: every tool call is a card with its status, command and duration



From the design doc: only the proxy holds the agent's key



Capabilities, read live from the gateway

PERSONAL PROJECT · LIVE

Viz Cancer

Open-data visualisation of cancer in the Philippines, where every figure is checked against the publisher before it ships.

An interactive 3D anatomical body map for women and men shades each organ by registered deaths, estimated new cases or survival — 77,504 cancer deaths in 2024 across 18 regions.

- Reproducible Python pipeline that snapshots 5 official sources (PSA, IARC GLOBOCAN, peer-reviewed studies) into 4 clean tables of ~58,000 rows.
- 44,551 automated ground-truth checks against each publisher's own tables; every file is SHA-256 fingerprinted and a failing check blocks the deploy. Each chart shows its verification status.
- An 8-tool MCP server so AI assistants query the same verified data, deployed as a Vercel function beside the site.
- Accessible 3D viewer: full keyboard control, screen-reader announcements, reduced-motion support.

React 19

Three.js

TypeScript

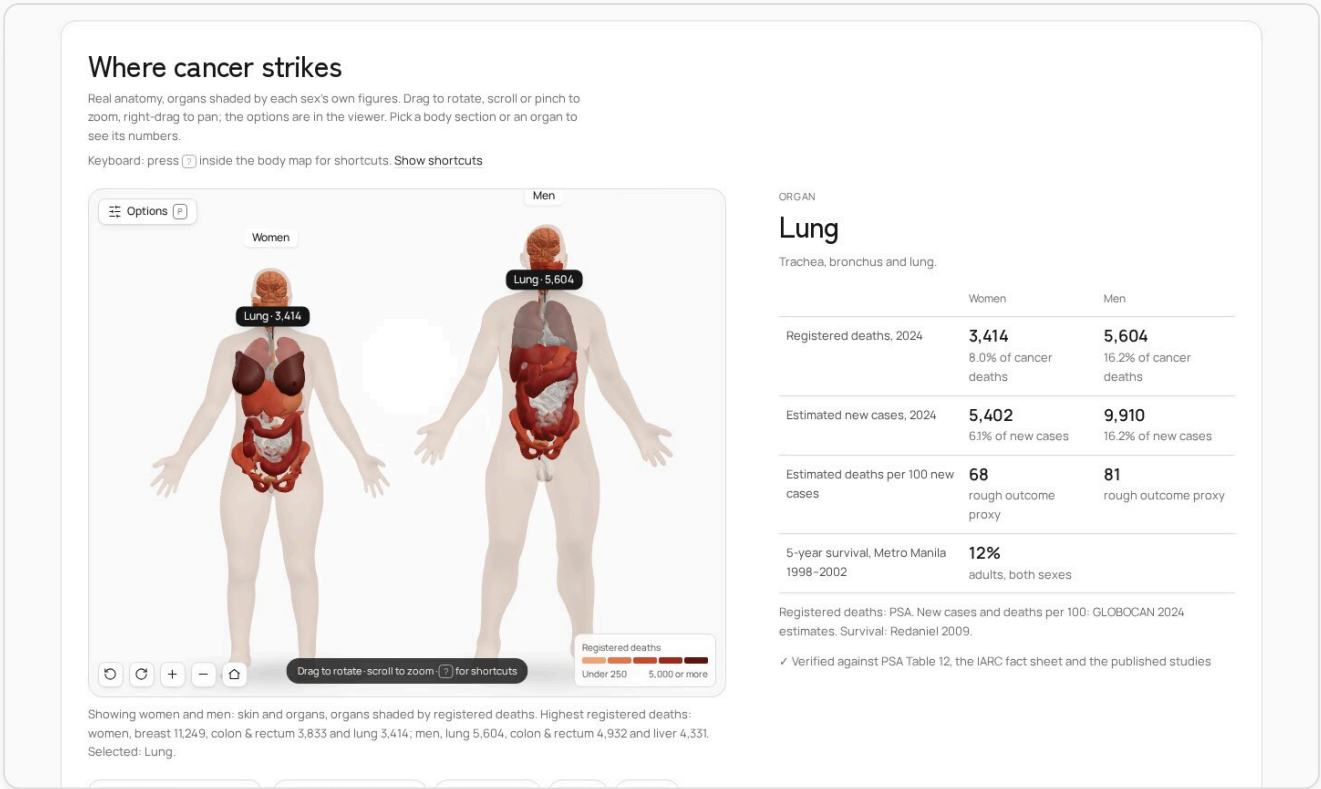
Python

MCP

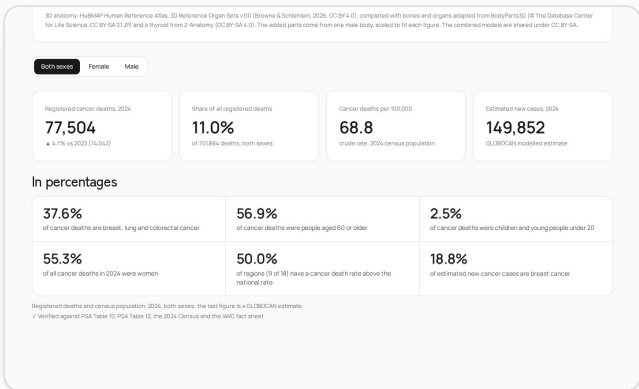
Nx

Vercel

viz-cancer.vercel.app github.com/sipelepis/viz



3D body map: organs shaded by each sex's registered deaths



Headline figures, each with its verification line



Deaths by site and death rate by region

PERSONAL PROJECT · LIVE

rag-pet

Retrieval-augmented generation over your own documents, with the retrieval shown instead of described.

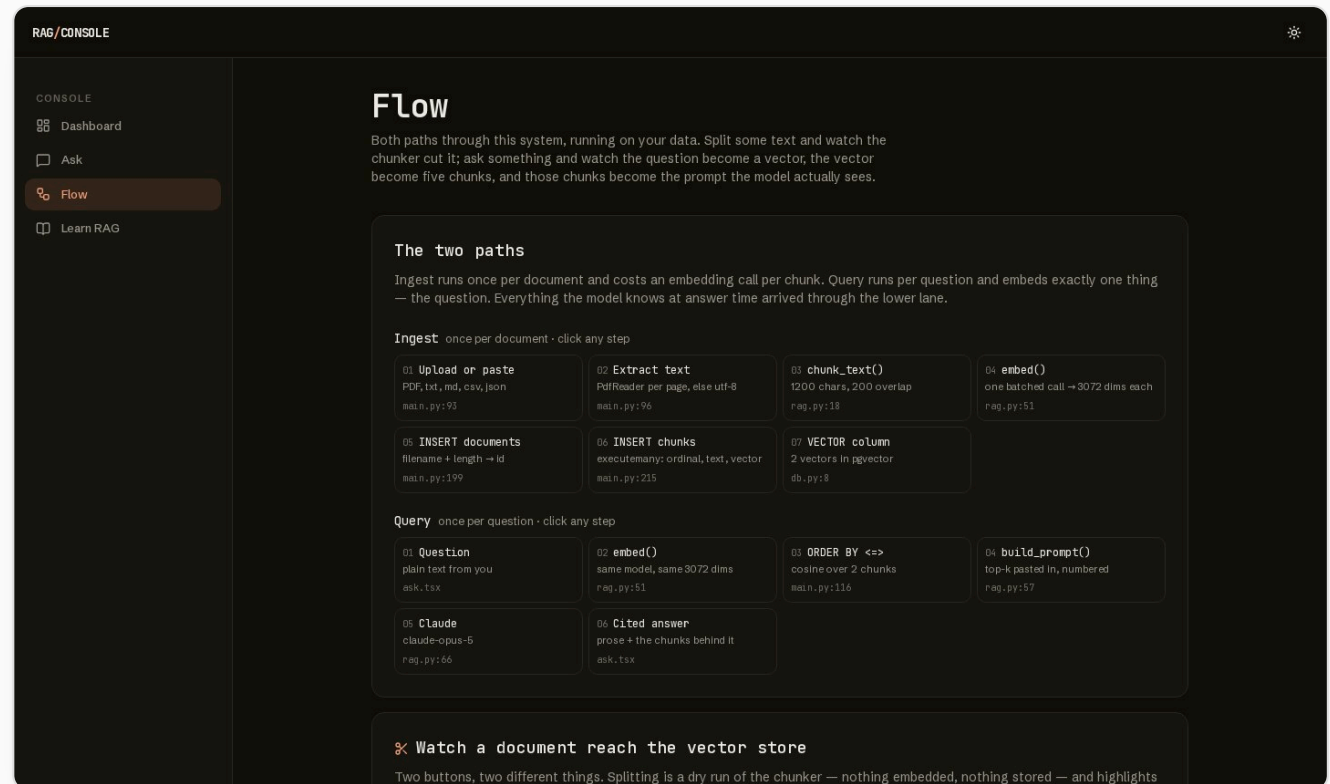
A FastAPI service does ingest and retrieval against Postgres + pgvector; a React console indexes sources, answers questions and lists the exact chunks each answer was built from, so you can tell bad retrieval from bad generation.

- A Flow page that runs both pipelines live — the chunker with its overlap highlighted, a traced query down to the embedding, the exact prompt, and where the milliseconds went.
- Pydantic models are the only API contract: codegen turns FastAPI's OpenAPI schema into the frontend types through the Nx graph, so a stale contract cannot compile.
- Every step links to the code that does it and to a Learn article on what goes wrong there.

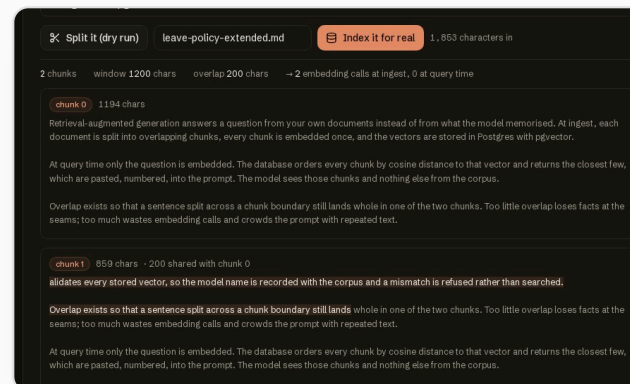
Python 3.13 FastAPI pgvector React 19 Vite

Tailwind CSS 4 Nx Fly.io

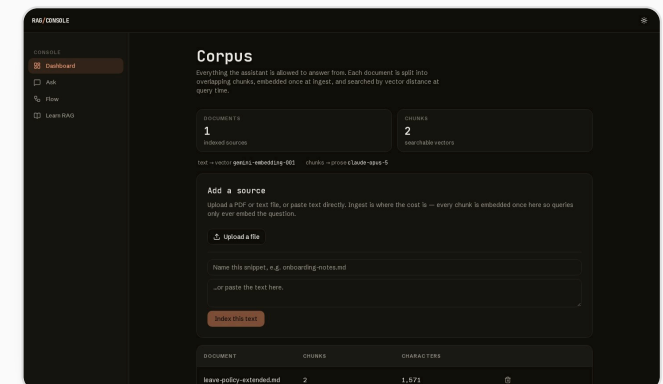
[rag-pet.fly.dev](#)



Flow: both paths, every step linked to the code that runs it



The chunker, live: overlap between chunks highlighted



The corpus dashboard

PERSONAL PROJECT · LIVE

Buhat

A strength and nutrition tracker that replaced my coach's spreadsheet.

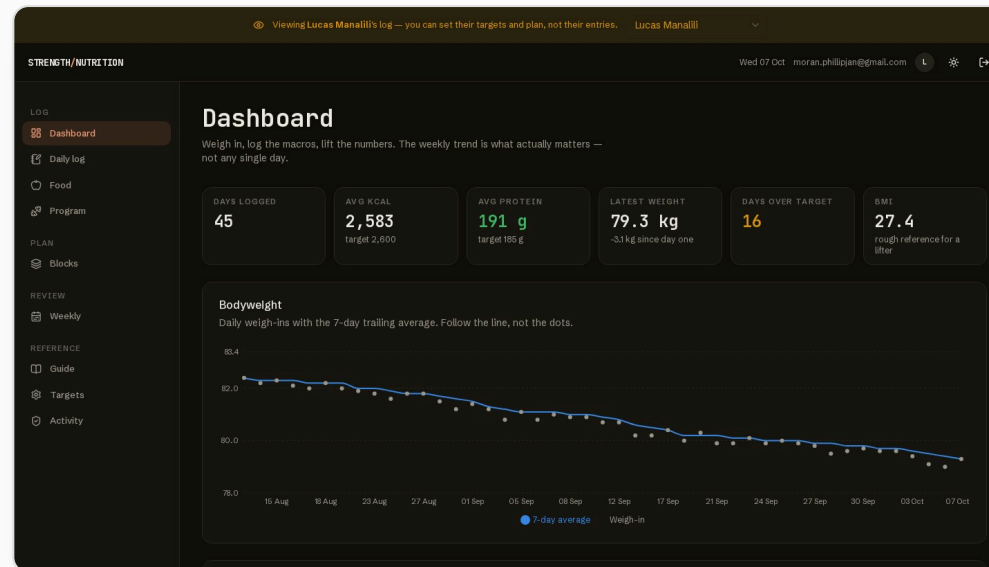
Log bar work between sets on a phone, macros at home, and read the weekly trend rather than any single day. Three surfaces — tracker, coach app and admin console — share one SQLite database.

- Google sign-in with server-enforced write scopes: a coach can set the plan and targets but never edit the athlete's log.
- One shared maths library (calories from macros, estimated 1RM, rolling averages) read by both API and browser, so they cannot disagree.
- Owner-controlled audit trail, and a seed script that builds a mock gym — coaches, trainees, months of training — for local demos.

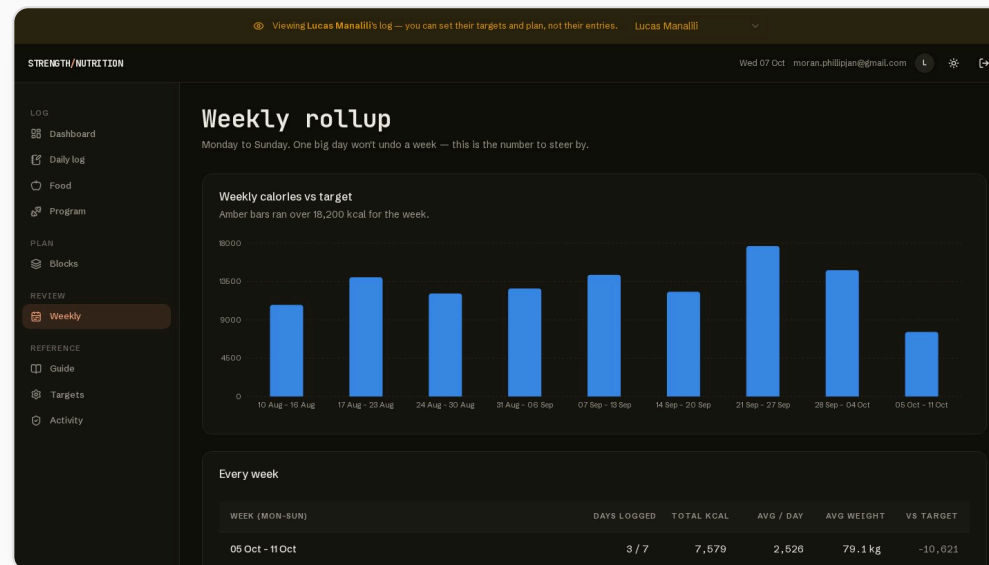
NestJS node:sqlite React 19 Vite recharts
Tailwind CSS 4 Nx Fly.io

buhat-site.fly.dev

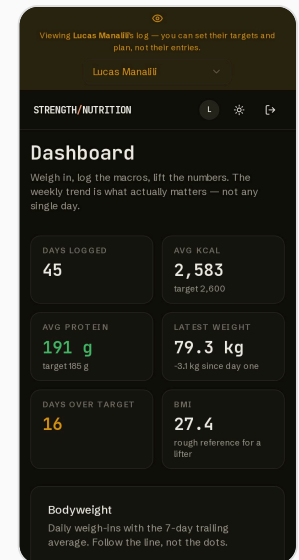
Screens use the seeded mock gym, not real athletes.



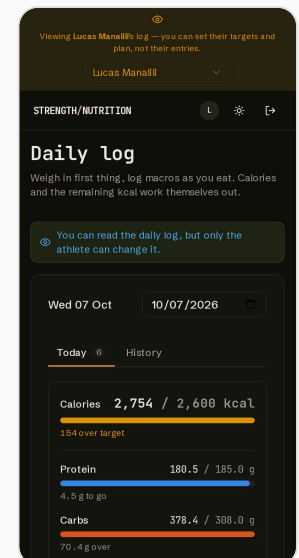
Dashboard, viewed by a coach: 7-day trailing bodyweight



Weekly rollup



Phone



Read-only for the coach

PERSONAL PROJECT · LIVE

Roadmaps

Nine self-hosted learning platforms: 174 modules and 749 exercises that run in the browser, with no install.

One shell, nine curricula — TypeScript, Python, RAG, Go, Rust, Kotlin, Swift, Android and iOS. Modules are laid out as a dependency graph, each with an article, a live playground and exercises with tests.

- Real code execution per language: Monaco's TypeScript worker, CPython on WebAssembly (Pyodide), and the official Go, Rust and Kotlin playgrounds; Swift compiles on a real swiftc via Compiler Explorer.
- Android and iOS tracks split testable logic exercises from build tasks with a checklist and a reference solution.
- One Turborepo; each roadmap deploys to Vercel on its own.

TypeScript Vite Monaco Pyodide WebAssembly
Turborepo Vercel

ts-roadmap-puce.vercel.app ↗
github.com/sipelepis/roadmaps ↗

The screenshot shows the 'Roadmaps' application interface. The top navigation bar includes 'Roadmap', 'Learn', 'Exercises', and 'Reference'. The left sidebar lists modules: 'MODULES', 'What is TypeScript?', 'Basic types', 'Functions' (selected), 'Objects & interfaces', 'Arrays & tuples', 'Unions & literals', 'Enums', 'Narrowing', 'Classes', 'Generics', 'keyof, typeof, T[K]', 'Utility types', 'Modules', 'Async & Promises', 'Conditional types', 'Mapped types', 'Template literal types', 'Advanced patterns', 'Validating unknown data', and 'Type-level puzzles'. The main content area displays an exercise titled '2. Optional last name' under the 'Functions' module. The exercise description states: 'fullName takes a required first name and an optional last name. It returns "First Last" when both are given and just "First" otherwise.' The code editor shows a TypeScript function:

```
function fullName(first: string, last: string): string {
  return `${first} ${last}`;
}
```

. Below the code editor are buttons for 'Run tests', 'Reset', and 'Show tests'. The bottom of the interface shows navigation arrows and labels for '1. Apply twice Functions' and '3. Average of any number of values Functions'.

An exercise: editor, hints and in-browser tests

The four screenshots show the 'Learn' tab for different technologies: RAG, Rust, Android, and a Module article. Each screenshot shows a 'Learn' page with a title, a brief introduction, and a 'Start the learning' button. The RAG page is titled 'Learn RAG from the page to the answer.' The Rust page is titled 'Learn Rust the way it fits together.' The Android page is titled 'Learn Android the way it fits together.' The Module article page is titled 'Basic types'.

PERSONAL PROJECT · IN DEVELOPMENT

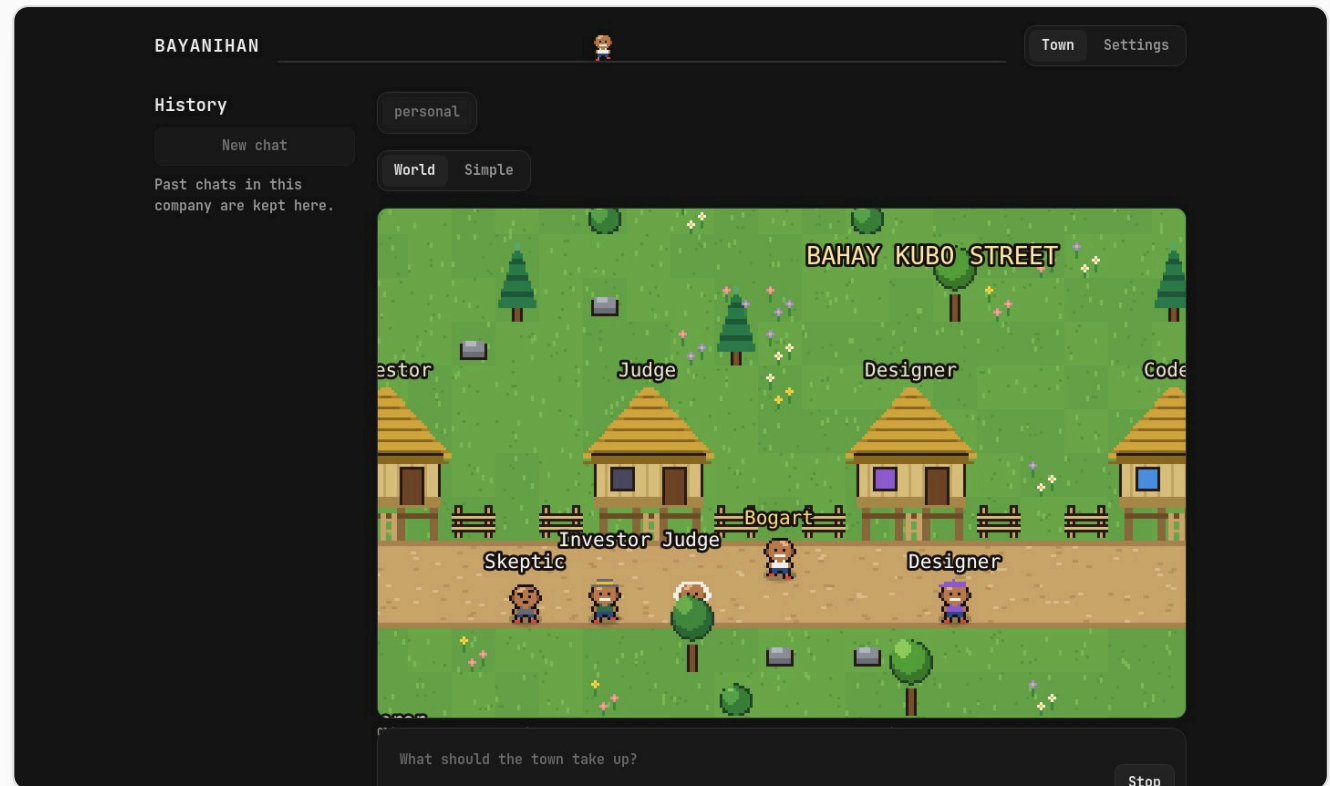
Bayanihan

A multi-agent harness drawn as a town: each company is a map, agents live in houses, and a router decides who to call for each prompt.

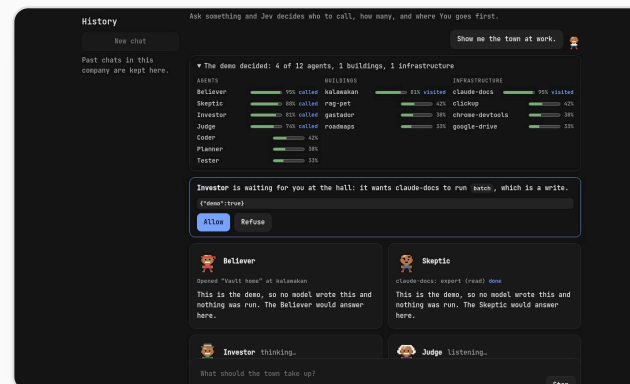
You walk the town as a character. For each prompt the router picks which of 12 role agents to call, which project buildings to visit and which MCP servers to use — and any tool call that writes waits at the town hall for your Allow or Refuse.

- Router output shown as percentages per agent, building and server, with the round's route drawn on the map.
- Human approval gate for write tool calls, shown in the world and in the chat.
- 12 editable agent roles, each with its own prompt and pixel character; a scripted demo round runs with no model calls.

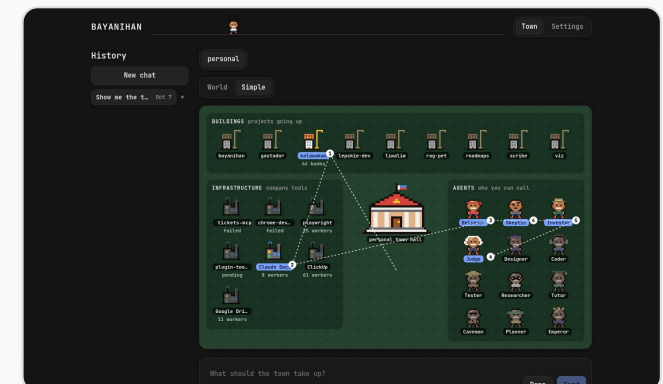
TypeScript Phaser React Node 24 MCP Nx pnpm



World view: agents leaving their houses on Bahay Kubo Street



The router's decision, and a write held for approval



Simple view: projects, MCP servers, agents and the route

PERSONAL PROJECT · IN DEVELOPMENT

Liwaliw

Every organised hike in the Philippines, in one place.

A mobile app for hikers (“joiners”) to find a group hike that fits their date, budget and fitness and book a slot with the organiser, replacing a hunt across scattered Facebook groups.

- React Native 0.79 app built from 11Nx libraries (hikes, booking, requests, inbox, activity, auth over shared UI and data-access layers).
- Supabase backend in 10 SQL migrations: schema, row-level security, chat RPCs, notifications and organiser access.
- Native ports under way: Kotlin + Jetpack Compose for Android and SwiftUI for iOS.
- Design rules written down first: WCAG AA as the floor, one-handed use in sunlight at the trailhead.

React Native

TypeScript

Supabase

Nx

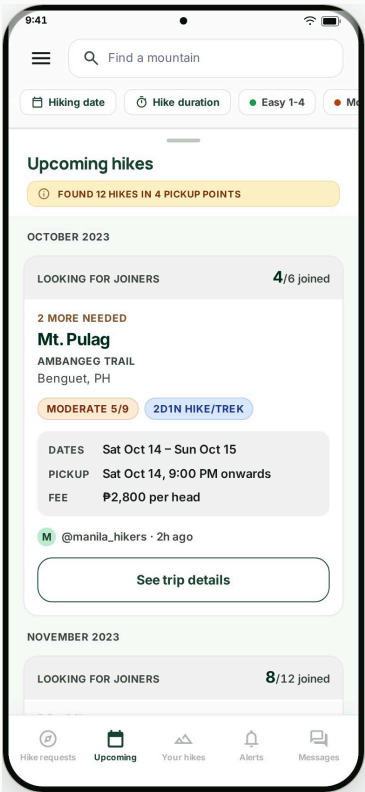
TanStack Query

Kotlin

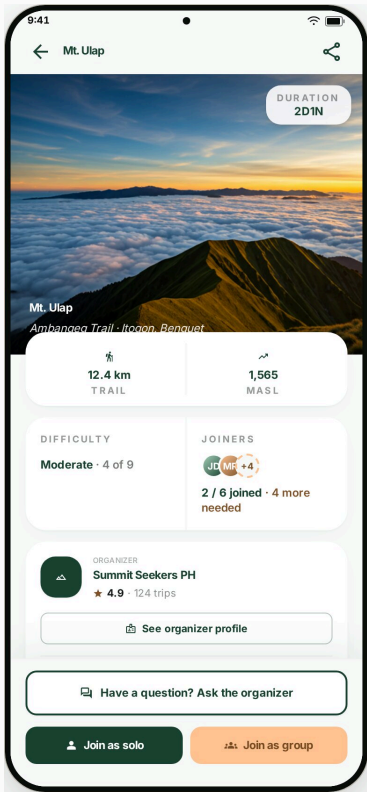
Jetpack Compose

SwiftUI

From the clickable UI prototype, with sample data.



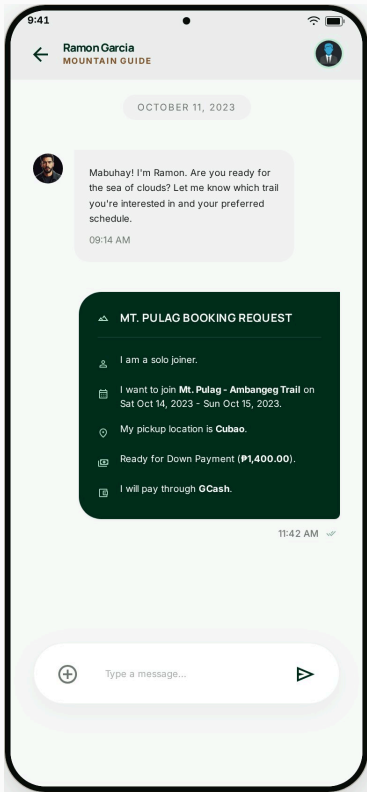
Upcoming hikes



Hike details



Hiking-dates heatmap



Booking request as a chat card

PERSONAL PROJECT · LIVE

Gastador

A peso budget tracker in one HTML file: no framework, no build step, no backend.

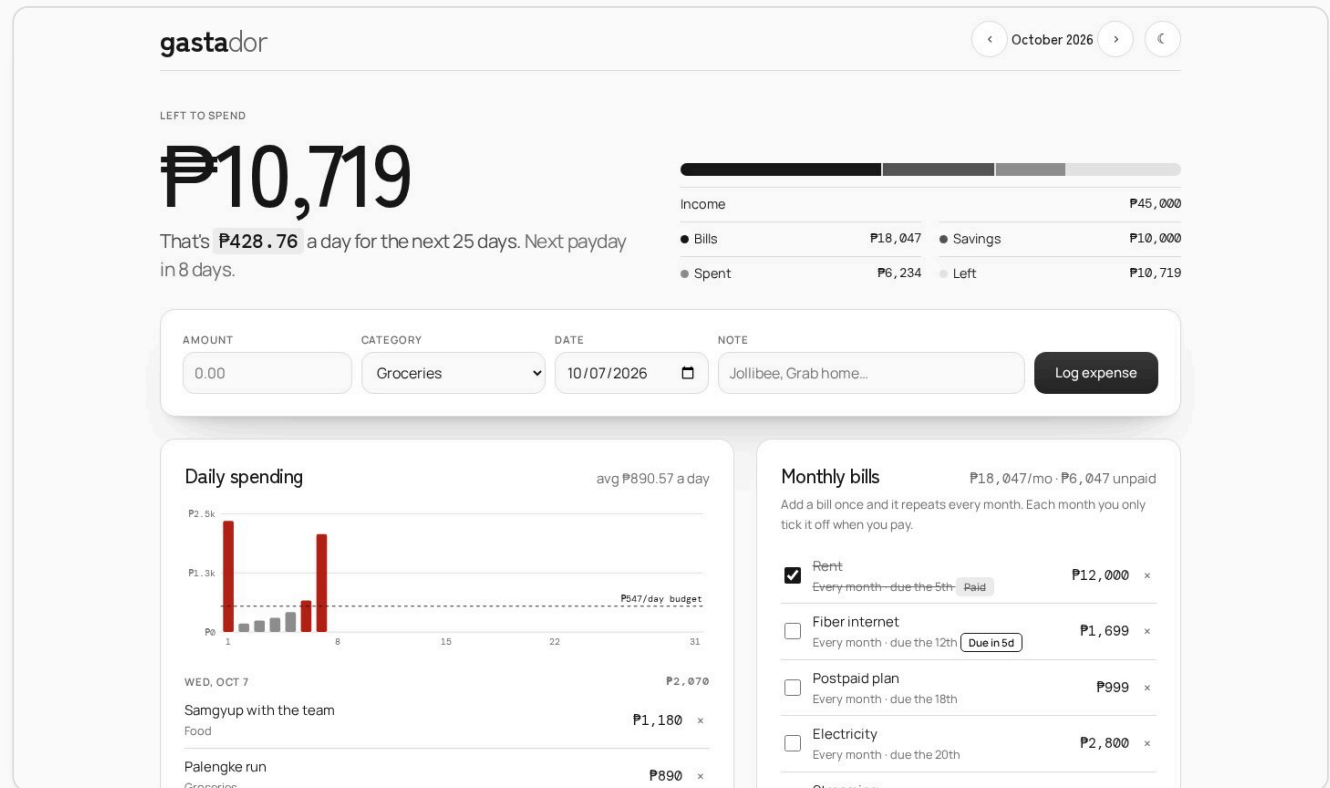
It answers one question — how much can I still spend this month? Income minus bills, savings deposits and expenses, shown as a daily allowance until payday.

- About 40 KB of HTML, CSS and vanilla JS; a self-check asserts the month maths on load.
- Hand-drawn SVG spending chart, category breakdown, recurring bills and savings goals that show the monthly amount needed.
- JSON backup and per-dataset CSV export; data stays in the browser.

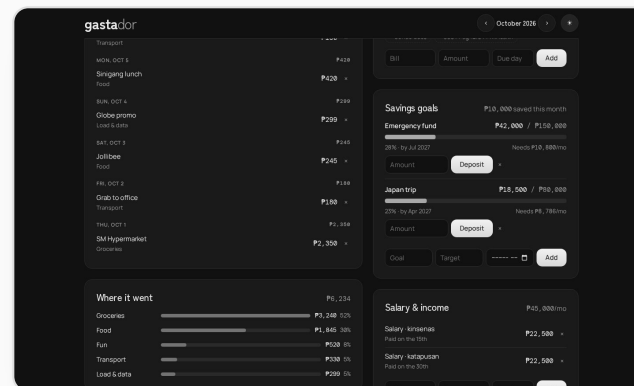
HTML CSS Vanilla JS SVG Vercel

gastador-six.vercel.app

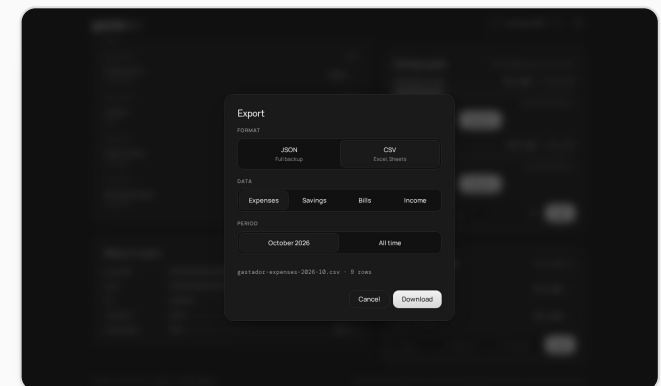
github.com/sipelepis/gastador



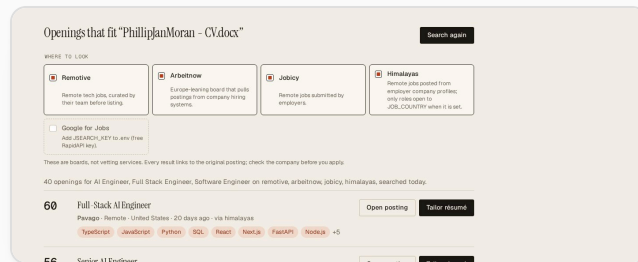
Month view: left to spend and the daily allowance



Dark theme



Export dialog



Openings from four job boards, ranked by skill overlap

PRIVATE

Jobtailor

Upload a résumé, find openings that fit, and get a version tailored to one posting.

- Three LLM providers behind one interface — Ollama locally for free, the Claude Code CLI or the Anthropic API — and scraped postings sanitised against prompt injection.

TypeScript React Express node:sqlite Ollama



The job queue, and an SRT it produced

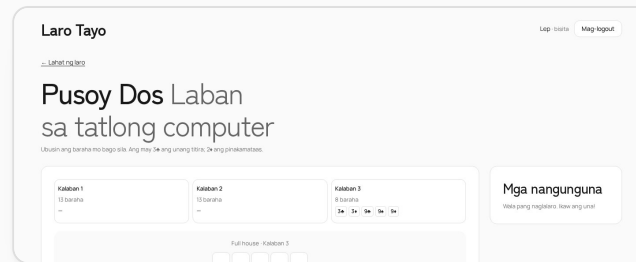
PRIVATE

Scribe

Long videos to .txt and .srt transcripts on your own GPU, with no API keys.

- Streams multi-GB uploads in 4 MB chunks, caches transcripts by content hash, and falls back from CUDA to CPU.

Python FastAPI faster-whisper CUDA



Pusoy Dos against three computer players

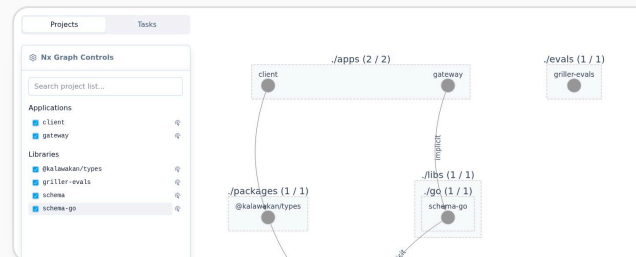
IN DEVELOPMENT

Laro Tayo

Traditional Filipino games — Sungka, Dama, Pusoy Dos, Jak-en-poy — against the computer.

- Rule engines as pure, tested functions; computer players use alpha-beta search, with a 150 ms budget for Dama.

TypeScript React 19 NestJS node:sqlite Nx



Nx project graph

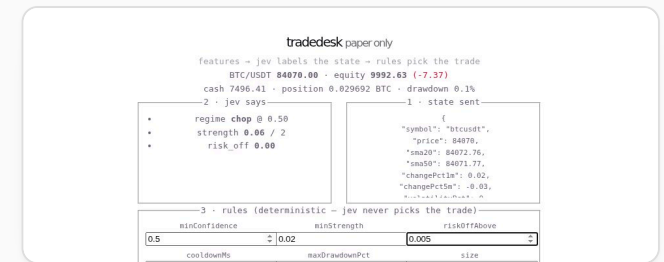
IN PROGRESS

Kalawakan

A 3D colony sim that will be the control surface for a multi-agent harness.

- JSON Schema generates the TypeScript and Go types; three spikes with written verdicts, including a headless `claude -p` job contract.

Go TypeScript JSON Schema Nx



Live BTC/USDT; each fill names the rule that fired

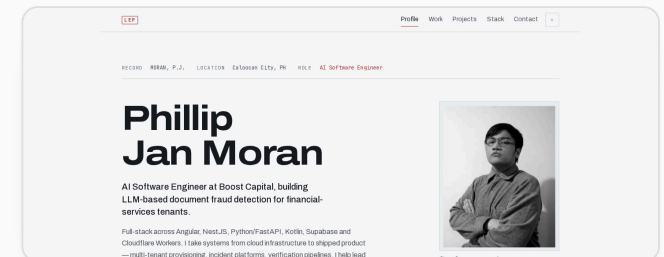
IN DEVELOPMENT

tradedesk

Paper trading where a model labels the market and deterministic code picks the trade.

- The model only answers typed questions; an ordered rule table decides, with a drawdown halt and cooldown above it.

TypeScript React 19 WebSocket TypeSafe API



Home

LIVE

Portfolio site

Angular 20 prerendered to static HTML, with a NestJS contact API.

- Nx monorepo with types shared between site and API; spartan-ng components; Supabase.

Angular 20 NestJS Nx Supabase

lepskie-dev-black.vercel.app