

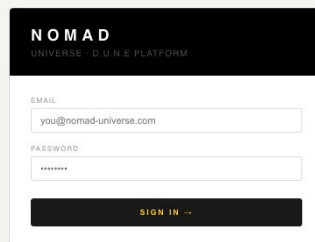
D.U.N.E: eight specialised AI agents working one project

D.U.N.E stands for Digital Universe for Nomadic Excellence. It is an internal AI workspace for the Nomad Universe group, and it is not a chatbot. A user opens a project and assigns work to a team of eight named, role-specialised...

Role: Founder, sole architect and developer

Stack: Next.js (client), Express and MongoDB (server), Anthropic Claude via the official SDK (Claude Opus 4.5 by default), Vercel

Live: dune-self.vercel.app (sign-in required; the platform is internal to Nomad Universe)



LIVE SIGN-IN, DUNE-SELF.VERCEL.APP (INTERNAL TOOL)

Context

D.U.N.E stands for Digital Universe for Nomadic Excellence. It is an internal AI workspace for the Nomad Universe group, and it is not a chatbot. A user opens a project and assigns work to a team of eight named, role-specialised agents: Taskmaster, Strategist, Planner, Researcher, Producer, Comms, Chief of Staff and Creative Director. Each agent has its own responsibilities and its own way of working, and the system is built so that a human can see exactly what each agent did, what it cost, and where it needed help.

I designed and built it from scratch, without an agent framework, because I wanted to own the decisions about specialisation, observability and escalation rather than inherit someone else's.

What I built

The agent layer

Each of the eight agents is a dedicated route on the Express server with its own system prompt defining its role, its rules and what it must not do. All eight call one shared service that wraps the Anthropic SDK directly. Model, maximum tokens and temperature are configurable per call, so a Researcher can run cooler and longer than a Creative Director. Agents work in two modes: attached to a project, where they share its context and tasks, or standalone.

The domain model

Projects, tasks, concepts and users are first-class MongoDB models. So is the agent log: every single agent interaction is persisted with the agent's name, the project it belonged to, the prompt, the response, the token count and the duration in milliseconds. That gives per-agent cost and latency observability from the first day, not bolted on after a surprising bill.

The workspace

A Next.js client with dedicated project and agent views: a new-project form, project cards, an "ask an agent" panel for direct work, and an escalation block. The escalation block is the piece I care most about. When an agent cannot resolve something with confidence, that is a designed state with its own UI that hands the decision to a person. The alternative, an agent that silently guesses, is how these systems lose trust.

Supporting services

A brand-fetcher service gathers brand and context material for the Creative Director and Comms agents, and an email service handles notifications.

Why it is built this way

- ◆ **Role decomposition over one general assistant.** Narrow system prompts produce more reliable behaviour than one agent asked to do everything, and they make failures legible: you know which agent got it wrong.
- ◆ **Observability as data, not dashboards.** Because every call is a row with tokens and milliseconds, cost per project, per agent and per week is a query, and slow or expensive agents show up immediately.
- ◆ **Escalation as a feature.** A human hand-off path is designed in, so the product degrades gracefully instead of hallucinating a resolution.

Outcomes and evidence

- ◆ Eight working agents behind a sign-in gate at dune-self.vercel.app, used inside the Nomad Universe group.
- ◆ Full per-call logging of tokens and latency across all agents.
- ◆ Built directly on the Anthropic SDK with no framework dependency, so the architecture is entirely in the codebase and entirely mine to change.

The hub-and-spoke diagram on asadaliawan.com is drawn from this codebase's route structure.