
The Articulation Method

How I built a personalised AI diagnostic tool
in seven days — without writing code.

JACK HARRIS

jackharris.ai · 2026

The Shift

Why Domain Knowledge Is the Leverage

Here's what nobody tells you about working with AI: the most valuable skill isn't prompt engineering or knowing how to code. It's knowing your domain deeply enough to describe what needs to exist.

Every operator carries years of accumulated knowledge. Pricing decisions, customer patterns, workflow bottlenecks, edge cases you've handled so many times they feel obvious. Most of it is so intuitive you'd never think to write it down. But a machine doesn't have your intuition. It needs all of it made explicit. And the process of making it explicit — through conversation, through dictation, through being pushed with clarifying questions — is where the real breakthroughs happen.

A developer with no domain knowledge builds generic software. An operator who can articulate their domain builds something that solves their actual problem.

Beyond Prompting

If you've been thinking about AI in terms of "prompt engineering," you're operating at the first layer of a five-layer hierarchy. Prompts matter in the same way that knowing how to type matters — it's a baseline, not a differentiator.

THE FIVE-LAYER HIERARCHY

- 1 Prompt Craft**
Writing clear instructions. Most people stop here.
- 2 Context Engineering**
Giving the AI your specific business context, constraints, and history.
- 3 Intent Engineering**
Describing outcomes, not tasks. Shifting from instructions to objectives.
- 4 Specification Engineering**
Documents precise enough that a machine can execute without further briefing.
- 5 System Design**
Orchestrating multiple AI capabilities into a coherent operation.

Most operators never get past layer two. The Articulation Method is designed to get you to layer four — where you can build anything.

The Method

01 Capture

Dictate, don't type. Use voice-to-text and brain dump without editing. Go off on tangents. The goal is to get the idea out of your head with zero friction between thinking and capturing.

The gap between what you know and what you can type is enormous. When you type, you self-edit, simplify, leave out nuance. When you dictate, you circle around an idea, come back to it, add layers. AI is sophisticated enough to tease out structure from a verbose, wandering answer.

02 Distil

Let AI find the structure. Paste the raw dictation into a conversation with your project context loaded. The AI identifies what you're actually asking for, pushes back with clarifying questions, and distills the mess into a structured brief. This is a conversation, not a single prompt. You go back and forth. You argue. You refine. This is where 80% of the value is created.

03 Spec

Write the map, not the code. The output is a sprint spec: a document detailed enough that a fresh AI instance — or a human developer — could execute it without further briefing. The spec is the work. The build is the output.

04 Build

The AI executes the spec. If the spec is good, the build is mechanical. When the spec is precise, the build takes minutes for work that would traditionally take hours.

05 Ship

Deploy. Test. Fix what only becomes visible when real people use your tool. Loop back to Stage 1. The gap between 'built' and 'shipped' has collapsed to a single command.

The Case Study

Building the AI Readiness Scorecard — Seven Days, Start to Finish

The Operator AI Readiness Score at jackharris.ai/score is a diagnostic tool for tour operators. Twelve questions. Five minutes. A personalised assessment matched to an operator archetype, with specific first moves per category. That's the surface. Underneath, it's doing significantly more.

What's Actually Happening — The Pipeline

ON EMAIL BLUR — 0 SECONDS INTO THE QUIZ

The moment you type your company email and tab to the next field, the application triggers a parallel pipeline. Your website is being scraped while you answer questions.

The front-end detects a company email domain (Gmail and free providers are filtered out). It fires an API call that triggers two things in parallel: **1)** Firecrawl scrapes your website — homepage first, with a raw-fetch fallback, then your /about page if detected. **2)** Claude Opus analyses the scrape, extracting a structured business profile: type, products, pricing tier, distribution channels, tech stack, team signals, review signals, and a confidence rating.

BY QUESTION 4 — ~90 SECONDS IN

The personalisation pipeline has returned. Questions 4-10 are now rewritten for your specific business — not by inserting product names (that feels invasive), but using operational language that reflects your world. A bike tour operator sees "managing guides" and "seasonal scheduling." A cooking class operator sees "kitchen logistics." The questions feel like they were written for you because, at that point, they were.

ON QUIZ COMPLETION — ~5 MINUTES IN

Your answers hit six analysis engines: **Scoring engine** — 12 questions, four categories, custom scoring functions per question (the SOP question scores differently at 0, 1-5, 6-15, and 16+). **Archetype matching** — six operator archetypes defined by midpoint vectors, matched via Euclidean distance with confidence rating. **Layer assessment** — five-layer hierarchy mapping answers to operational maturity (rewritten three times; final version uses weighted scoring with 0.6 threshold). **Structured website scraper** — homepage plus three internal pages, extracting booking widgets (FareHarbor, Bokun, TicketingHub, Peek, Rezdy, Xola), OTA presence, reviews, tech stack. **AI insight generation** — all data assembled into a 4,000+ word system prompt for Claude Opus, returning personalised diagnostics, cross-category patterns, and specific first moves. **PDF and email** — results render on screen and generate as a PDF with custom React Email templates, attached and sent via Resend.

Graceful fallbacks everywhere. No website scrape? Section hides. Gmail address? Same. Fewer than five submissions? Percentile hidden. LLM fails? Static recommendations. Every section degrades gracefully.

The Timeline — From the Git Log

Thu 20 Feb, 22:45

First commit. Create Next App. The scaffolding.

Tue 25 Feb, 00:30

Scorecard build begins. Quiz engine, five question formats (numeric, button grid, icon select, slider, statement-agree), email capture. Twenty commits this day.

Tue 25 Feb, 23:15

Sprint 1 complete: archetype matching live. Six archetypes with midpoint vectors.

Tue 25 Feb, 23:45

Committed viewport-locked quiz layout. Looked correct in development.

Tue 25 Feb, 23:49

Reverted. Four minutes later. iOS Safari: CSS dvh + keyboard = layout jumping on every keystroke.

Wed 26 Feb, 00:05

Fix: JS-measured viewport height, auto-focus gated behind desktop check. Three commits for a one-line problem.

Wed 26 Feb, 09:12

All Anthropic calls switched to Opus 4.6 fast mode. Latency matters with two Claude calls per submission.

Thu 27 Feb, 00:28

Results page overhaul. Layer map, archetype visualisation, numbered next steps. Score readout becomes diagnostic report.

Thu 27 Feb, 00:37

Calculating screen: deadpan cycling lines, progress bar racing to 70% then crawling (AI genuinely takes 10-15s).

Thu 27 Feb, 00:58

Layer assessment rewrite #3. Weighted scoring replaces pass/fail. Most-iterated component in the build.

Thu 27 Feb, 01:07

Competitive percentile, website reality check, revenue risk, peer benchmark radar. All with graceful fallbacks.

Thu 27 Feb, 22:16

Final commits: PDF null checks, email buffer encoding fix. Edge cases from real data.

Total: 7 days. ~90 commits. 6,300+ lines of application code. Zero lines written by hand.

The Messy Middle

The iOS Safari viewport bug. CSS dvh units for a viewport-locked layout. Perfect in Chrome, catastrophic on Safari. The mobile keyboard triggered layout recalculation on every keystroke. Three commits to fix what looked trivial.

The layer assessment rewrites. Three versions. V1 used strict pass/fail: one weak signal blocked everything above. Strong operators showed as stuck at layer 1 from a single documentation gap. The insight: operational maturity isn't a strict hierarchy in practice. Weighted scoring was the fix. This is domain knowledge at work — no technical skill alone would have caught it.

The OG image loop. Four versions in one hour. Each looked fine in the editor, wrong in the social preview. Fonts render at ~600px in social feeds — all type sizes needed roughly double.

The email encoding bug. PDF attachments arrived corrupted. Sent as base64, Resend expected a Buffer. One line fix, two commits to diagnose.

The Decision Framework

Let it run when functionality is clearly defined, scope is bounded, and there's no live data at risk. A component, a utility, a page layout.

Check in frequently when the build involves design decisions, touches live data, or the scope is ambiguous enough the AI might optimise for the wrong thing.

Stay hands-on when you're working through an idea that isn't fully formed. You can't delegate clarity.

The honest answer: I'm still refining this. The calculus changes as the tools improve. What doesn't change: the quality of the output is always a direct reflection of the quality of the specification.

The spec is the work. Everything else is execution.

The Stack

You don't need to become a developer. But you should understand the building blocks well enough to have an opinion about what you're building on.

Next.js	The framework. Website, app, and API in one place.
Vercel	Deployment. Push code, it's live. No servers.
GitHub	Version control. Every change tracked, reversible.
Supabase	The database. Submissions, scores, scrape results.
Resend	Transactional email with React Email templates.
Firecrawl	Website scraping. Homepage + internal pages.
Claude	AI model. Personalisation, analysis, insights.
Loops	Email marketing. Segmentation and nurture.

The specific tools matter less than the principle: every layer should be programmable. If you can't wire it into something else, you'll hit a ceiling fast.

What This Means for You

Once you've built one thing with AI, the second is faster. Not because the tools improve (though they do), but because your ability to describe what you want improves. You learn what level of detail the machine needs. You develop an instinct for specification.

The scorecard was the vehicle. The ability to articulate, spec, build, and ship — that's the asset. It applies to everything you build from now on.

Take the Score

The AI Readiness Score is live at jackharris.ai/score. Twelve questions, five minutes. Now you know what's happening behind the scenes.

Build With Me

The Build Sprint, AI agent deployment, or a conversation about what fits your operation: jackharris.ai/work-with-me

Jack Harris runs Tally Ho, a premium tour company in London, and builds AI tools for experience operators. Running tours since 2011, building with AI since 2024, documenting both at jackharris.ai.