/erica-layer

AI ON PURPOSE

A PLAYBOOK

The AI-Native Website Playbook

A start-to-finish guide to building your own website as an AI-native operator, with no engineering team, without hiring a developer or fighting a page builder. Ten steps, the real decision behind each one, the AI skill it builds, and the traps to avoid.

Erica Layer

layeradvisory.com

Dear reader,

I built this website myself, in Claude Code. Some people will think less of me for that. But I am extremely proud of this site, and I could not have done it on my own.

What made it possible is that the AI I work with already knows me. I have spent the last five months building an AI operating system that holds my context: who I am, how I work, what I care about, and what I am trying to build. It knows the projects I have running and the decisions I have made along the way. It has my positioning and my read on the market. It is also connected to the systems the site runs on, Netlify, Supabase, Kit, Stripe, Google Drive and Apps Script, Google AI Studio, and Cloudflare, so when something needed building it could go and build it instead of telling me how.

The work looked like a series of conversations. I would sit down with Claude, describe what I wanted, look at what came back, and send it back again. Claude wrote every line of code on this site. I wrote none of it.

The realistic alternative was paying a designer and a developer thousands of dollars, which I could not afford. Instead I have a site that belongs to me, and I can change it, extend it, or rebuild a page just by asking.

I have been part of a lot of website redesigns, and every one of them was a painful process. This was the first time I was building something that felt distinctly mine, that carried my voice, and that had a level of clarity I have never had in my career.

When I finished, I could see there was a lot in here that other people could use, so I wrote down what I actually did at each step, including the places I got it wrong.

My judgment is what made this good. It took days. I went in circles, threw a lot out, and started again more than once. You could hand this playbook to AI, ask it to build you a website, and have something back within the hour. It would look like everything else being churned out right now. The reason mine does not is that I made the decisions myself and put my taste into every part of it, and that is why I am happy to have my name on it.

I hope this shows you that you are capable of building something excellent. What I would ask is that you keep your own judgment front and center while you do it. Do not cut corners just because AI is now your partner. If you put the time in, I think you will end up with something you are just as proud of.

With gratitude,

Erica

Principal, Layer Advisory

`layeradvisory.com`

What this is

A start-to-finish guide to building your own website as an AI-native operator, with no engineering team, without hiring a developer or fighting a page builder. Ten steps, the real decision behind each one, the AI skill it builds, and the traps to avoid.

It is illustrated throughout with a real build as a worked example (layeradvisory.com, built exactly this way), so every step shows both the principle and how it played out in practice. Read it and build your own, or hand it to your AI and have it walk you through your version.

One thing to say up front, because it runs underneath every step that follows. This site was not built by a better prompt. It was built by an AI that already had my context: months of accumulated files about my positioning, my voice, my clients, and my real projects, written and maintained as an operating system I run my practice on. That is why the copy sounded like me on the second pass instead of the twentieth, and why the case studies almost wrote themselves. Wherever a step went unusually fast, the context layer is the reason. I flag it as it comes up, and I come back to it at the end.

Who it's for

Purpose-driven professionals and small organizations who want a real website that reflects who they are, built and owned by them, without hiring a developer or fighting a page builder. Non-technical, willing to work with AI step by step.

Before you start

Here is what you actually need, so you can tell whether you are ready or whether something else comes first.

For most of the playbook: an AI assistant you already use and trust, and a few hours spread over a couple of weeks. Steps 1 to 4 are thinking, not building. You can do all of them today, in a chat, with nothing installed.

For Steps 5 and 6, the build and the deploy: this worked example was built in Claude Code Desktop. That is a normal desktop app with a simple chat interface. You do not need the terminal, and you do not need to know anything about the terminal. You describe what you want, and it writes the files, runs whatever needs running, and shows you the result. The only requirement is a paid Claude plan, which starts at twenty dollars a month. For hosting, Netlify is free at this size, and you point your domain at it when you get there.

That is the whole list. No code, no terminal, no developer.

The part that actually decides how well this goes is context. The site in this worked example did not come out well because of the tool. It came out well because the AI doing the work already knew who I

case studies, the decisions I had already made and why. That is an AI operating system, and building one is a separate skill from building a website. It is also the skill that made this build fast, specific, and mine rather than generic.

You can absolutely follow this playbook without one. You will just do more of the explaining yourself, over and over, in every session. Where the operating system would have supplied the answer, you supply it. The steps still work.

If you are not ready to build yet, do the thinking now and the build later. Work through Steps 1 to 4 and come out with your positioning, your message, and a locked design system, written down. That is the part that makes a site look like you instead of like everyone else, and it is the part AI cannot do for you. When you are ready, your spec is waiting.

How to read it

Ten steps, in order, in three arcs: **Strategy** (1 to 3), **Build** (4 to 6), and **Grow** (7 to 10). Each step has the same shape:

- **Outcome:** what you have when the step is done.
- **Why it matters:** the principle underneath it.
- **In practice:** how it played out in the real build, with the specific decisions.
- **How you'd do it:** the reusable version, adapted to your situation.
- **The AI skill:** the capability you build, and where AI does the heavy lifting.
- **Watch for:** the real lessons and traps.

The principles that run through all ten steps

These are the through-line. Every step is an application of one or more of them.

- **Decide before you generate.** Lock the spec (positioning, offer, design) before any tool makes anything. This is the anti-slop discipline. It is the single biggest reason the site does not look AI-made.
- **The site is proof, not a brochure.** A brochure describes you. Proof demonstrates how you work. Build the second one.
- **Your work is your content.** When your real projects live in a structured, readable system (notes, PRDs, results), AI can turn them into case studies and posts with almost no effort. The context you already keep becomes the source for your proof.
- **Context beats prompting.** The quality of what AI makes for you is set mostly by how much it knows about you before you ask. An assistant working from your real files, your positioning, and your voice rules produces something specific on the first try. An assistant working from a blank slate produces the average of the internet, every time, no matter how good the prompt is.

- **Own your data.** Your analytics, your signups, your content live somewhere you control, not locked inside someone else's dashboard.
- **Privacy-light by default.** Collect the least you can. No cookies, no stored IPs, nothing you would not want to defend.
- **It never sounds like AI.** Every word passes the voice rules. If a sentence reads like a language model wrote it, it gets rewritten.
- **Verify by watching it work.** Load the real thing and watch the real event land before you trust a single number. Never assume.
- **Grade against a researched bar, and loop until it clears.** Do not judge your own work by eye. Have AI first research what good actually looks like for what you are making, then run a loop: grade the work against those criteria, fix the weak spots, and grade again, until it scores above a real threshold. Eight out of ten is a solid bar to hold to, applied at several points in the build.
- **A website is a system you operate, not a project you finish.** The last step is a loop, not a finish line.

The ten steps

Arc 1. Strategy

1. Decide to build it yourself
2. Find your positioning
3. Nail the offering and the message

Arc 2. Build

4. Lock the design system before generating anything
5. Build it in Claude Code as a static site
6. Deploy and put it on your domain

Arc 3. Grow

7. Make it findable by search and by AI
8. Wire the site to do the work
9. Build the analytics dashboard
10. Analyze, decide, iterate

ARC 1

Strategy

STEP 1

Decide to build it yourself

OUTCOME

A clear decision to build your own site, and a one-line brief for what it has to do for your practice.

WHY IT MATTERS

The decision to build it yourself is not a budget choice, it is a positioning choice. For an AI-native operator, building the site yourself IS the demonstration. The medium becomes part of the message. It also sets the whole rest of the path: if you are going to build it yourself, you build in a tool you control, not a page builder you fight.

IN PRACTICE

My old site was a generic fractional-COO site (Vercel-hosted, built in Lovable) that no longer told the truth about my work. I now build AI into how organizations operate, and the old site said none of that. I decided to rebuild it myself, in Claude Code, for three reasons: so it would lead with the AI offering rather than just "fractional COO," so it would speak to both of my audiences (purpose-driven organization leaders and individual practitioners), and so the site itself would be an example of the thing I sell. My one-line brief: the front door for the AI-offering pipeline, for both audiences, that proves the work by existing.

HOW YOU'D DO IT

Answer three questions before anything else.

1. What does this site actually have to do? Generate leads, establish credibility, be a place to send people, sell a specific thing? Name the primary job.
2. Who is it for? One audience, or two? Be honest. Two audiences is harder and shapes the whole structure.
3. Is building it yourself part of the proof? If you help people adopt AI, or build systems, or work in a modern way, then yes, and that changes your tooling choice downstream (Step 5).

THE AI SKILL

Using AI as a thinking partner to get honest about what the site is for, before falling in love with a design. You talk it through, it pushes back, you land on a brief you can hold everything else against.

WATCH FOR

Building a brochure by default. Most sites describe the person. If yours only describes you, you have a brochure. The whole playbook is about building the other thing: a site that shows how you work.

STEP 2

Find your positioning

OUTCOME

A sharp position (who you are, who it is for, the whitespace you own) that every later decision flows from.

WHY IT MATTERS

Positioning is upstream of copy, design, and offers. If it is vague, everything downstream is vague, and no amount of good design saves it. This is the step people skip and the one that decides whether the site lands.

IN PRACTICE

I positioned on the intersection of two worlds most people keep separate, operational systems and AI adoption. Not a consultant who knows about AI, not a technologist who learned operations. The person building both at the same time, in live client engagements. I did not invent that line. I mined it from a large body of research I had already built: a synthesis of around fourteen AI and operations podcasts (forty-plus episodes), a competitive read of dozens of other practitioners, and two detailed personas (the individual practitioner and the organizational leadership team, each with their real fears). AI did the synthesis. The position was already sitting in the material. I also got clear on who it is for and the problem I actually help with, without turning the site into a fear pitch.

HOW YOU'D DO IT

1. Gather your raw material: talks, posts, client calls, the language your competitors use, the questions people actually ask you.
2. Have AI synthesize it into candidate positions and, crucially, the whitespace: what is true about you that no one else is claiming.
3. Choose the position that is both true and uncontested. If it is not true, it will not hold up in a sales call. If it is contested, you are competing on someone else's ground.
4. Name your audience as one or two real people, specific and concrete, not "small businesses."
5. Get clear on the core problem you help with, and lead with that, not five things at once.

THE AI SKILL

Using AI to synthesize a mountain of source material into a sharp position. This is synthesis, not invention. The distinction matters: the position has to already be true about you; AI's job is to find it and sharpen the language, not to make something up that sounds good.

WATCH FOR

Two traps. First, letting AI invent a flattering position that is not actually yours, which collapses the moment a real prospect probes it. Second, trying to speak to every problem at once; lead with the core one, and let the site acknowledge the rest.

ARC 2

Build

STEP 3

Nail the offering and the message

OUTCOME

A small set of clear offers, a message that leads with the problem you actually solve, and copy in your own voice so the site never reads as AI.

WHY IT MATTERS

Positioning is the ground; the offering and the message are what the visitor actually reads. Fuzzy offers or copy that sounds machine-made will sink a sharp position. This is where most self-built sites quietly fail.

IN PRACTICE

I named three ways to work with me: Embed (fractional COO), Build (AI-native operations), and Teach (AI operating systems, including my TEMPO cohort). Landing on "Build" took several tries. I started with "AI systems build," which made me sound like a software developer shipping products, and reworked it until it named the real thing: rebuilding how an organization runs a recurring process, with AI in the loop. I did not type the copy myself. I directed it in detail, usually by voice, describing exactly what I wanted with a lot of specificity, and I held every line to hard voice rules (no em dashes, no AI buzzwords, first person, specific over abstract) until it sounded like me and not like a model. Then I put the whole site through graded loops. I had AI research what makes a site like mine work, clarity, credibility, and what moves a visitor to act, then run agentic loops that scored the site against those criteria, found the weak spots, fixed them, and scored again, repeating until it cleared eight out of ten. That work sharpened my core positioning, a fractional COO working in the space between operations and AI, and produced the line I lead with now: the difference between using AI as a thought partner in a chat window and building it into how you actually work.

The most convincing part, the case studies, was also the easiest. Because I run my practice on a structured operating system, a vault of markdown files with a PRD and build notes for every real project, the raw material was already there. AI read those files and turned real engagements into clear case studies with almost no effort. That is a quiet benefit of keeping your context in a structured system: your actual work becomes the source for your proof, instead of something you reconstruct from memory.

HOW YOU'D DO IT

1. Name your offers as a few clear "ways in" (two to four). For each, say what it is, who it is for, and the outcome, not the mechanics.

3. Direct the copy in detail rather than writing it yourself (I do mine by voice), then hold every line to a short list of hard voice rules until it sounds like you.
4. Have AI pressure-test the page as an outside reader, and fix what does not land.
5. For your proof, point AI at your real project records (notes, docs, results) and have it draft your case studies from what you actually did, instead of writing them from scratch.
6. Do not judge the result by eye alone. Have AI research the criteria for a site that works, then run a grade, fix, and re-grade loop against them until it clears a real bar. I used eight out of ten.

THE AI SKILL

Two of them, and the second is the one people skip. The first is directing AI to draft in your voice from specific, detailed instruction, then editing hard and using it as a review panel to catch anything that still reads like a machine. You direct, edit, and decide; you never accept the first draft. A written voice rulebook it can check every line against is worth building once. The second is running agentic loops. Instead of asking AI what it thinks of your page, you have it research what actually makes a site like yours work, turn that research into scoring criteria, then grade the page against those criteria, fix the weak spots, and grade again, on its own, until it clears a real bar. I hold to eight out of ten. Directing the draft is what makes the copy sound like you. The loop is what makes it good, and it runs without you standing over it.

> WATCH FOR
>
> Handing AI a vague brief and accepting what comes back, which always reads like a model; the work is in how specifically you direct it and how hard you edit. Also: describing the mechanics of your service instead of the outcome; leading with your product's name instead of what the reader came for.

STEP 4

Lock the design system before generating anything

OUTCOME

A one-page design spec, colors, type, spacing, components, and imagery rules, locked before any tool makes a single pixel.

WHY IT MATTERS

This is the anti-slop discipline made concrete, and it is the biggest reason the site does not look AI-made. Generate first and you inherit the generic AI-website look. Decide first and everything downstream inherits an identity you chose.

IN PRACTICE

I landed on a plum, teal, and cream palette with Space Grotesk, DM Sans, and DM Mono, retiring an old slate-and-gold palette I had never actually chosen on purpose. Worth being honest about how I got there, because it is not what people picture. I did not sit down and choose these myself, and I am not a designer. I asked AI to give me a huge variety, and I went through at least thirty different combinations of color and type, reacting to each one, before I hit the one that felt distinctly mine.

The reasons it is this and not something else are real ones. The plum reads as somewhat feminine, and I kept it for that reason: more women need to be building with AI, and the work should not look like it belongs to someone else. I did not want a black background, which is what most AI and programming work looks like right now. And I wanted something with a bit of computer typing and text in it. That mix is what says who I am without a word of copy.

Then I wrote a one-page spec before generating anything: the palette with strict usage rules (the accent goes on headlines, CTAs, and accent lines, never as a section-background fill), a dark structural color, and numbered editorial sections. The imagery rule was the important part: real photos of me and real screenshots of my systems only, no stock, no generic AI illustration. I made one scoped exception, a defined risograph house style for the Writing section, because a named artistic style you control and use consistently is a brand asset, not slop.

Most people have no idea how to actually produce an image they would want on their own site, so here is exactly what I do. I do not use Claude's own image generation. I use NanoBanana 2, through a Google AI Studio API key wired into Claude, so I can ask for an image in the same conversation where I am writing. The instructions for my house style are built into every single graphic it creates, so I never re-explain them and nothing drifts. I do not write the image description myself either: I point it at the piece of writing or name the idea I am trying to land, it writes the description from that, and it gives me three versions to react to. Reacting to three is far easier than starting from a blank prompt, and it is the whole reason the images stay consistent.

HOW YOU'D DO IT

Before you generate anything, write one page. Three to five colors with usage rules. A type system: heading, body, and one accent face. A spacing rhythm. Your core components. And a real decision about what your images will be. If you use AI imagery at all, define one deliberate style and use it consistently, or leave it out.

THE AI SKILL

Using AI to articulate and pressure-test a design spec, then holding every later generation to it. The taste and the decisions are yours; AI helps you write them down and enforce them.

> WATCH FOR
>
> "make me a website" with no spec, which is a guaranteed slop machine; using your accent color as a background fill; mixing stock photos, AI art, and screenshots into visual noise.

STEP 5

Build it in Claude Code as a static site

OUTCOME

The actual site, built page by page against your spec, in a tool you fully control.

WHY IT MATTERS

The build tool is a positioning choice too, decided back in Step 1. It also decides whether AI answer engines can read your site later (Step 7).

IN PRACTICE

I built the site as plain static HTML and CSS in Claude Code. I did not write any of the code myself. Claude wrote all of it from my direction, and I read each result and sent it back until it was right.

I have tried the alternatives, so this is a comparison and not a preference. I have built sites in tools like Squarespace, and I always end up frustrated in the same place: the moment it hands the design decisions back to me. I am a terrible designer, and a page builder is mostly a machine for asking you to make design decisions. I have also built in Lovable, which I still use for my other projects, and those sites come out fairly generic, with the exact output hard to control. It also produces a JavaScript-rendered app, which is a real problem for a marketing site, because AI answer engines do not run JavaScript and cannot read it (Step 7).

Claude Code was different in a way I did not expect. I can edit the site directly from the chat interface, describe the change in plain language, and deploy it live. By a distance, it was the simplest option, the one I have the most control over, and the one that came out most uniquely mine of anything I have ever used.

I built the homepage completely first, got it right, then applied the same system to every other page. Claude pulled in my actual photos from my Google Drive, so the images on the site are really me. For the product screenshots, I used the safe method: rather than screenshotting a client's real system, I rebuilt faithful mockups from the real app's own code with fake data, so the image is a true representation of the system with zero risk of exposing anyone's information.

HOW YOU'D DO IT

Pick a tool you control and can edit by conversation. I use Claude Code Desktop, which is a chat window, not a terminal; the site is plain static HTML and CSS, all of it written by AI from my direction. Give it your positioning and design spec before you ask for a single page, and if you keep your context in an operating system, point it there first. Build the homepage completely before you touch the other pages, then apply the system across them. Use real assets. If you need to show a system that holds sensitive data, rebuild a faithful mockup with fake data instead of screenshotting the real thing.

THE AI SKILL

Directing AI to build a real site file by file, reading each result, and iterating, without writing the code yourself. You direct and verify; AI writes the HTML and CSS.

WATCH FOR

Page builders that fight your spec; screenshotting real client or customer data; building every page before the homepage is actually right.

STEP 6

Deploy and put it on your domain

OUTCOME

The site live on your own domain, with HTTPS, and a one-command deploy so you ship changes yourself in seconds.

WHY IT MATTERS

Owning the deploy path means you never wait on an agency to change a comma. In practice it works like this: you type what you want changed into the same chat window you were already working in, Claude makes the change, and it goes live on the real site immediately. No files to move, no dashboard to log into, no one to email. That is worth setting up properly once, because it is the difference between a site you maintain and a site you quietly stop touching.

IN PRACTICE

I deployed to Netlify. I moved quickly from dragging a folder onto the dashboard to a proper deploy command, which Claude runs for me, so now the site builds a clean copy of itself and goes live from a chat. I pointed my custom domain through Cloudflare DNS with the exact records, set to DNS-only so Netlify could issue the HTTPS certificate, and I left my mail records untouched. The one real gotcha: Cloudflare's own bot protection blocks an automated browser, so when I needed to confirm the DNS had propagated, Claude verified it with a direct lookup instead of trying to load a dashboard.

HOW YOU'D DO IT

Pick a static host with a deploy command (I use Netlify) and get one-command deploy working early. You do not run that command yourself unless you want to. You ask your assistant to deploy, and it runs it. Point your domain's DNS at it. If your DNS sits behind Cloudflare, set the records to DNS-only so the host can issue the certificate, and do not touch your mail records. Confirm HTTPS is live.

THE AI SKILL

Standing up hosting, a deploy command, and DNS with AI walking you through each step, including reading the error when a record does not resolve and telling you what to change.

WATCH FOR

Editing your mail records by accident; leaving proxying on so the certificate will not issue; a bot-blocked dashboard, which your assistant gets around with a direct lookup instead.

ARC 3

Grow

STEP 7

Make it findable by search and by AI

OUTCOME

A site that search engines and, increasingly, AI answer engines can find and cite.

WHY IT MATTERS

More people than you would guess now ask an AI "who does X" instead of searching. AI answer engines do not run JavaScript, so the static HTML from Step 5 is already the right substrate. Most of the rest is clarity and specifics.

IN PRACTICE

I confirmed the research that AI answer engines do not execute JavaScript, which validated the static-HTML choice. I did the fundamentals: a sitemap, a robots file, structured data describing me and my practice, and Search Console (which auto-verified off my existing Cloudflare DNS, with the sitemap submitted and crawled). Then I leaned into what actually earns AI citations: clear claims and liftable summaries, real first-person experience and specifics, quotes and numbers, and a genuine Q&A page answering the questions people actually ask. I skipped the low-value fashions, an llms.txt file and over-investing in schema. And I captured signups on my own terms, a waitlist posting to my own script and sheet rather than a third party's form.

HOW YOU'D DO IT

Make sure your site is static HTML. Add a sitemap, a robots file, and basic structured data, and submit to Search Console. Then write for liftability: clear claims, real specifics and numbers, and an FAQ that answers the real questions. Own your signup capture.

THE AI SKILL

Using AI for the technical chores, and, more usefully, to pressure-test whether your writing is actually specific and liftable enough to be quoted.

> WATCH FOR
>
> Content rendered by JavaScript that AI crawlers never see; vague copy with no specifics or numbers; chasing llms.txt and heavy schema instead of writing clearly.

STEP 8

Wire the site to do the work

OUTCOME

A site that captures the people who arrive, emails them, books them, and takes payment, on infrastructure you own.

WHY IT MATTERS

A site that only describes you leaves you as the bottleneck for everything that happens next. Every registration, every question about which option to buy, every receipt is a job that lands back on you. Once the site can capture, email, schedule, and charge on its own, it stops being a brochure with a contact form and becomes the front end of how you actually operate. This is also where owning your data stops being a principle and starts being practical: the people who sign up are the most valuable thing the site produces, and they should land somewhere you control.

IN PRACTICE

I built this in pieces, one at a time, each one because I was tired of doing something by hand. None of it arrived as a plan.

The webinar registration is the fullest version. The form on the page posts to a Google Apps Script web app I own. That script writes the person into a Google Sheet, then emails them a confirmation from my own Gmail with a calendar invite attached, so the session goes straight into their calendar and their reply comes back to me rather than to a no-reply address. That last part is the reason I did not reach for a mail service: I already had Gmail, it costs nothing, and mail from a real person outperforms mail from a robot. One deliberate detail is worth copying: the page reads the script's response for real instead of firing and forgetting, so if Google is having a bad morning the visitor sees a visible, retryable error rather than a false "You're registered." A signup form that quietly loses people is worse than having no form at all.

The download gate on this playbook taught me that lesson the expensive way. It posts to a small serverless function that calls Kit's authenticated API, tags the subscriber by which asset they came for, adds them to my welcome sequence, and emails them the file as an attachment. It has to work that way. Posting straight from the browser to Kit's public form endpoint does not work: their spam guard quarantines programmatic submissions and returns a success response anyway. It looks like it worked, every time, while silently losing everyone. I only caught it by checking whether real subscribers were actually appearing on the list.

Two of those four steps I only added after checking. Creating a subscriber and tagging them does not put them in any sequence, because sequences are joined through forms and automations, and an API signup goes around both. So the people who had just read a whole playbook, the warmest people I had, were getting no follow-up at all. Separately, my thank-you message said I had emailed them a copy when nothing was sending one. Both were invisible from the outside: the form said success, the file downloaded, and the gap only showed up when I went and looked at what had actually happened to a real signup.

Selling runs on Stripe payment links, one per tier, which took about ten minutes and needed no code on my side. Behind them, a routine watches for the payment notification, updates my participants sheet, generates a branded receipt, and prepares the welcome email for me to review before it goes. I kept a human in that loop on purpose. Taking someone's money is the last place I want a silent automation.

The last piece answers the question I was answering by hand over and over: which option is right for me. I built a small assistant, embedded directly in the page, that interviews the visitor about their situation and gives an honest recommendation between the two tiers or waiting for the next cohort. It is connected to the payment step, so someone can go from unsure to enrolled without me in the middle. It recommends waiting when waiting is the right answer, which is the only reason it deserves to be on the page.

HOW YOU'D DO IT

Do not build a funnel. Build the one piece that removes the thing you are currently doing by hand most often, and let the rest follow. In order: capture the person somewhere you own (a sheet or a database, not a third party's form), send the confirmation yourself so you control what it says, put the calendar invite in that email if timing matters, and only then add payment. Use hosted payment links before you build any checkout. If you find yourself answering the same qualifying question repeatedly, that is a good candidate for a small assistant, and a bad candidate for a longer FAQ. Keep a human review step on anything involving money.

THE AI SKILL

Having AI stand up real integrations, a form endpoint, a mail sender, a list, a payment link, and wire them to each other, while you decide what should be automatic and what should stop for your review. The judgment is yours: what gets sent without you, and what waits.

> WATCH FOR
>
> Integrations that report success while doing nothing, which is the default failure mode here and the reason to watch a real signup land end to end before you trust the form; fire-and-forget submissions that cannot tell the visitor something went wrong; automating the money step completely; building the whole funnel before anyone has signed up for anything.

STEP 9

Build the analytics dashboard

OUTCOME

Your own analytics, in a store you control, with a dashboard and an "ask me how the site is doing" read.

WHY IT MATTERS

A site you built yourself is a site you can actually understand the behavior of. You can see whether visitors are doing the thing you hoped they would do: how long they stay, what they click, where they came from, and, just as useful, what they never look at.

You can do a lot of this in Google Analytics. I find it more helpful to build a dashboard tailored to exactly what I want to see, and with this kind of site you can write the tracking into the HTML yourself, so you capture precisely what you care about. There is a second payoff. If you build the dashboard somewhere with an MCP connector, like Lovable, you can query your site analytics directly with Claude: ask what the numbers mean, have it run the analysis, and get specific suggestions for what to change on the site next. The data stops being a chart you glance at and becomes something you can have a conversation with.

IN PRACTICE

I built the whole thing self-hosted, so I would own the data, on three pieces I wired together with AI. First, the store: my own Supabase project, which is a hosted Postgres database, with an `events` table that captures the few things that matter, page views, CTA clicks, and page exits, each with the path, referrer, time on page, and time since the visit started. Row-level security is set to anonymous-insert-only, so the public site can write events but nobody can read the raw table, and the dashboard reads only through a handful of aggregate database functions. Second, the tracker: a small piece of JavaScript on every page that sends those events, privacy-light by design, no cookies, with country resolved by a Netlify edge function that returns only the two-letter code. One real bug taught the "verify by watching it work" principle the hard way: the browser's sendBeacon call was silently dropping my events, so I switched to a fetch call and watched real events land before trusting a single number. Third, the dashboard: I built it in Lovable, an AI app builder, connected to Claude through its MCP integration, so I could describe what I wanted and have it built without hand-coding, and it wired up the Supabase connection and the keys for me. Lovable was the right choice here for the same reason it was the wrong one for the marketing site: this dashboard is a private tool, so its JavaScript rendering does not matter.

HOW YOU'D DO IT

Pick a data store you own (I use Supabase). Capture only the events that matter, insert-only, and read them back through aggregate functions rather than exposing the raw table. Keep it privacy-light: no cookies, minimal data. Build the dashboard with an AI app builder so you are not hand-coding it, and, the real payoff, point your AI at the data so you can just ask how the site is doing and get an answer with recommendations.

THE AI SKILL

Standing up a real analytics pipeline, schema, tracker, and dashboard, with AI, and then using AI to interpret the numbers rather than only display them.

> **WATCH FOR**
>
> Analytics that trap your data in a dashboard you cannot query; collecting more than you need; trusting a number before you have watched a real event land, which nearly shipped me a tracker that was quietly losing half its data.

STEP 10

Analyze, decide, iterate

OUTCOME

A running loop that turns site data and real feedback into specific improvements, on a cadence.

WHY IT MATTERS

A website is a system you operate, not a project you finish. The version you launch is the worst version it will ever be, and the loop is how it gets better.

IN PRACTICE

Almost everything good about the current site came out of this loop, not the launch. A strategic review sharpened the hero to own "AI-native fractional COO" and my positioning in the space between operations and AI. One reader's confusion drove a hero-sentence fix. A colleague's candid walk-through drove a whole pass on hierarchy, sizing, and visuals, including a brand-style diagram that replaced a wall of text. My own dissatisfaction with a flat line drove the "thought partner versus a colleague built into how I work" breakthrough. A mobile menu bug got caught and fixed. Each one followed the same shape: notice something, decide the change, ship it, verify it on the live site. I am formalizing this as a recurring review, the way my weekly review already works.

HOW YOU'D DO IT

Put the site on a cadence; monthly is plenty to start. Read your analytics and search data, gather honest reactions by sending it to a few people who will tell you the truth, list what is working and what is not, pick a small number of specific changes, ship them, and verify each one actually rendered and works. Then do it again.

THE AI SKILL

Using AI to read the analytics and the feedback, propose specific and prioritized changes, make them, and verify them, closing the loop instead of just reporting numbers.

> WATCH FOR
>
> Treating launch as the finish line; collecting feedback and never acting on it; changing something and never confirming it actually rendered and worked.

How to use this playbook

Two ways, and they are the whole point.

Read it and build it yourself. The ten steps are in order deliberately. Work through them. Each one gives you the outcome, the principle underneath, how it played out in the real build, how you would adapt it, the AI skill it builds, and the traps to avoid.

Or hand it to your AI and have it guide you. Give this whole document to a capable assistant and say: "Walk me through building my own version of this. Interview me about my situation first, one question at a time, then take me through it step by step." A good assistant uses the playbook as the method and adapts every step to you. That is the design: the playbook is the thinking, your AI is the hands.

Why it is a document, not a set of skills

You could imagine building this as a set of installable skills, a router and a guide for each step, the way a repeatable process like hiring gets encoded. For a website, a plain document is the better call, at least for now.

A repeatable process runs the same way every time, with the same forms and the same steps, so encoding it as skills pays off on every run. Building your website is the opposite. You do it once, and it is different for everyone: their positioning, their brand, their tools. That kind of one-time, high-judgment, high-variance work is exactly where a capable assistant with a good guide does better than a rigid script. A plain document is also more portable: anyone can read it or hand it to their AI without installing anything.

What made this actually work

If you take one thing from this playbook, take this.

Every step above is real, and you can follow all ten of them. But the reason this build went the way it did is not in the steps. It is that the AI doing the work already knew me. It had my positioning, written down and argued out. It had my voice rules, the specific ones, including the words I never use. It had my client projects as structured files, which is why the case studies took minutes instead of days. It had the decisions I had already made and the reasons behind them, so it never proposed something I had already rejected.

That is the difference between AI as a tool you prompt and AI as a colleague who knows your work. The first one gives you the average of the internet. The second one gives you something that sounds like you, because it has read everything you have decided.

I built the first version of my website with AI too, before any of that existed. It looked like every other AI site, and I never launched it. Same tools. Same person. What changed the second time was the context underneath.

So if this playbook feels like it describes a build that went suspiciously smoothly, that is the honest answer. The website was the easy part. The operating system underneath it is the real thing, and it is what made the website easy.

Learn to build the operating system

That system is what I teach.

TEMPO is my six-week cohort for non-technical people who want to build their own AI operating system: the context files, the daily routines, the knowledge base, the loops that hold AI to your standards, and the automations that run without you. You build it during the six weeks, on your own real work, and you leave owning it.

Everything in this playbook, the voice rules, the graded loops, the structured project files, the analytics dashboard, came out of that system. Build the system, and a website is one afternoon of what it can do.

layeradvisory.com/tempo

/erica-layer

AI ON PURPOSE

Build the operating system underneath.

The website was the easy part. What made it easy was an AI that already knew my positioning, my voice, and my real work, because I had built it a system to know them from.

TEMPO is my six-week cohort for non-technical people who want to build exactly that: the context files, the daily routines, the knowledge base, the loops that hold AI to your standards, and the automations that run without you. You build it on your own real work, and you leave owning it.

Everything in this playbook came out of that system.

layeradvisory.com/tempo

I'm Erica Layer, an AI-native fractional COO. I help purpose-driven organizations build the human and AI systems their work runs on, and I teach people to build their own.
layeradvisory.com · aionpurpose.substack.com