Build a Website: Phase One, Design It

A step by step guide to designing a website before you build it, even if you are not a developer. This is Phase One of the process I used to build my own site: planning, research, designing the pages, and locking a design system you can build on.

Every step that involves talking to an AI tool includes the prompt to use. Copy it, fill in the parts in square brackets, and send it.

What you end up with: a fast, custom website on your own domain, that you can update yourself, hosted free, with every change going live automatically when you save it.

What it costs: the domain, roughly 10 to 20 US dollars a year. Everything else in this guide is free.

How long it takes: a weekend for a simple site. A few weeks if you have a lot of pages or writing to produce. Most of that time is content, not code.

What it works for: any site that is mostly pages and content. A personal or portfolio site, a freelance or consultant site, a small business or agency site, a product or service landing page, a blog or publication, a community or event site, a nonprofit, documentation. If your site needs user accounts, payments, or a live database, this process still gets you the front of it, but you will need more than this guide for the rest.

Who it is for: anyone who wants a site that is genuinely theirs rather than a template. You do not need to know how to code, but you do need to be willing to read what the tools give you and ask for changes.

The tools

Job	What I used	Alternatives
Design the pages	Claude	Figma, any AI design tool
Turn the design into a real project	Claude Code	Cursor, Antigravity, Codex
Store the code	GitHub	GitLab
Host the site	Cloudflare Pages	Netlify, Vercel
Buy the domain	Cloudflare Registrar	Namecheap, GoDaddy

The process matters more than the tools. Any coding assistant and any host will work. What matters is the order you do things in.

How to prompt well

Five rules that make everything below work better.

1. **Say what you want, not how to code it.** Describe the outcome and the feeling. The tool knows the code, it does not know your intent.
2. **Give constraints, not just requests.** "Change the typography, leave the layout and colour alone" gets a far better result than "make it nicer".
3. **Ask to see before it changes.** For audits and big rewrites, add "show me what you find before changing anything". You stay in control.
4. **One change at a time.** If you change four things and it gets worse, you cannot tell which one did it.
5. **Ask why.** When something is done that you do not understand, ask. You end up understanding your own site, which matters when it breaks in six months.

The whole process

Phase 1, design it. Plan, research, design the pages, lock the design system, export it, convert it into a real project.

Phase 2, ship it. Buy the domain, put the code on GitHub, connect it to hosting, get a live URL, build the rest of the pages.

Phase 3, build it out. Add your writing and anything a static site cannot do on its own.

Phase 4, finish it. Make it findable, make it fast, audit it, launch.

The single most important thing in this guide: **get the site live in Phase 2, while it is still ugly and half finished.** Everything after that is calm, because there is no scary launch day. You have been live the whole time.

PHASE 1: DESIGN IT

Step 1. Plan

No tools yet. Open a notes app. Half an hour here saves a week later.

Answer three questions.

What is the site for? Be specific, and pick one main job. "A website for my business" is not an answer. "Get people to book a consultation" or "get people onto my mailing list" or "prove I can do this work so I get hired" is an answer. Everything you build later either serves this or gets cut.

Who is it for, and what do they need to see? Write down the one visitor you care about most and what they are trying to find out. Most sites fail because they answer the owner's questions instead of the visitor's.

What is the site map? List every page. This becomes your folder structure, so getting it right now saves renaming later. Some common shapes:

- **Personal or portfolio:** Home, About, Work index, one page per project, Contact
- **Service business or freelance:** Home, Services, About, Pricing, Contact
- **Agency:** Home, Services, Work, About, Team, Contact
- **Product or software:** Home, Features, Pricing, Docs, Contact
- **Blog or publication:** Home, Articles index, one page per article, About, Newsletter
- **Local business:** Home, Services, About, Location and hours, Contact
- **Event or community:** Home, Programme, Speakers, Venue, Register

Nearly every site also needs a Privacy page and a 404 page.

What content actually exists? Be honest. Count the real things you have: finished projects, service descriptions, product photos, testimonials, writing. If you offer three services, you need three pages, not six. This tells you the real size of the job.

Common mistake: planning what the site looks like at this stage. That comes next. Right now you are only deciding structure and purpose.

If you want help thinking it through, prompt:

```
I'm planning a website for [what you do or sell].

The main thing I want it to achieve: [e.g. get people to book a call,
sell a product, get hired, grow a mailing list]

The visitor I care about most: [who they are and what they want to know]

What I already have:
- [content, photos, projects, services, writing, testimonials]

Ask me questions until you understand it properly, then propose a site map
with a one line purpose for every page. Tell me which pages I do not need.
```

Step 2. Research

Collect references before you design anything.

Gather reference sites. Find five to ten sites you genuinely like. Include a few from outside your industry, since copying your competitors is how everything ends up looking the same. For each one, write down exactly what you like. Be specific about layout and structure, "the way the pricing table puts the recommended option in the middle and larger", rather than "the vibe". Layout is the useful part. Colour you

will decide yourself.

Decide your starting point. There are three ways in:

- Starting from nothing, which is what most people do
- You already have a design in Figma, so you are converting rather than inventing
- You are redesigning an existing site, so you already know what works and what does not

Collect your assets. Photos, logo, product images, brand colours if you have them, and any writing that already exists. Knowing what you have shapes what you design. There is no point designing a section around a video you do not have.

To pull the patterns out of your references, prompt:

```
Here are screenshots of [number] websites I like.

For each one, tell me what it is actually doing structurally: the layout,
the spacing, the type hierarchy, how it uses motion, and how it guides the
visitor toward an action.

Then tell me what they have in common, because that is probably what I
actually like. Ignore the colours, I will choose those myself.
```

Step 3. Design the pages

This is where the look gets decided. You describe what you want and get back a real, working, coded page you can scroll and hover, rather than a flat mockup you have to imagine in motion.

Start with the home page only. Everything else inherits its type, colour, and spacing. Do not design eight pages before you know the first one is right.

Describe your intent, not the CSS. Say what each section does and how it should feel.

Put your references in. Paste the screenshots you collected and say what you want from each.

Ask for variations. When you are unsure, ask for three versions. Choosing between things you can see is far quicker than describing something you cannot.

Iterate in passes, not all at once. Structure first, then typography, then colour, then motion.

The first design prompt:

```
Design the home page for [describe the site: e.g. a small architecture
studio, a freelance copywriter, a project management app, a food blog].

What the page needs to achieve: [the one action you want visitors to take]

Sections, in order:
1. [e.g. a hero with a clear headline saying what this is and who it is for]
2. [e.g. a short paragraph explaining the offer]
3. [e.g. the three services, or the main features, or recent articles]
4. [e.g. proof: testimonials, logos, results, or reviews]
5. [e.g. a closing call to action and footer]

How it should feel: [e.g. calm and editorial, or bold and energetic, or
clean and technical. Say what it should NOT feel like too]

I've attached screenshots of sites I like. From the first I want [specific
thing]. From the second I want [specific thing].

Build it as one working page I can scroll, with realistic placeholder
content rather than lorem ipsum.
```

Then iterate, one pass at a time:

```
Good base. Now change only the typography. Make the headings [bigger and
tighter], increase the body line height, and set a clearer size difference
between headings and body. Leave the layout and colours exactly as they are.
```

```
Now the colour pass. Use [describe your palette, e.g. a warm near-black,
a soft off-white background, and one accent colour]. One accent only.
Leave the layout and type alone.
```

```
Now motion. Add [e.g. a gentle rise on the headline when the page loads,
and sections fading up as they enter view]. Keep it subtle. Make sure it
all respects prefers-reduced-motion.
```

When you are unsure, prompt:

```
Show me three variations of just the [hero / pricing table / navigation].
Keep everything else identical so I can compare them fairly.
```

Then the other pages:

```
Using exactly the same design system as the home page, design the [Services]
page. Do not introduce new colours, fonts, or type sizes.

Sections: [list them]
```

Step 4. Lock the design system

Do not skip this. It is the reason a finished site looks like one site instead of forty separate pages.

Once the home page looks right, write the design system down as a real document.

What goes in it:

- Every colour, with a name, a hex code, and what it is for
- The type scale: which fonts, which weights, which sizes, and their jobs
- Spacing and layout rules
- Component rules, so a button looks the same everywhere

Keep it small. A good system is restrictive. One text colour, one background tint, one accent, two or three fonts with clear jobs. That is usually enough for an entire site.

Prompt:

```
Now write the design system for this design as a Markdown file called
DESIGN_SYSTEM.md.

Include:
- Every colour, with a token name, hex value, and exactly what it is used for
- The type scale: each font, its weights, its sizes, and its job
- The spacing scale
- Component rules for buttons, cards, forms, and links
- The breakpoints

At the very top, add this rule: no new colours, fonts, or type sizes may be
added without updating this file first.

Keep it restrictive. If I have used five greys, collapse them to two and tell
me which ones I lost. Fewer tokens is better.
```

That last instruction matters. A first design usually has more colours than it needs, and this is the moment to cut them.

Where this leaves you

At the end of Step 4 you have the three things that make the rest of a website build straightforward:

1. **A plan.** You know what the site is for, who it is for, and every page it needs.
2. **A design you can actually see and scroll**, not a flat mockup you have to imagine.
3. **A locked design system**, so everything you add later already knows what it should look like.

That is the whole of the design phase. What follows in the full process is turning that design into a real project, buying a domain, putting the code on GitHub, connecting free hosting so every change deploys itself, building out the rest of the pages, then making the site findable, fast, and ready to launch.

Take your time over Step 4. A restrictive design system is the difference between a site that looks designed and a site that looks assembled.

This guide describes the process behind tamkeen.studio. If you build something with it, I would genuinely like to see it.