THE COMPLETE CLAUDE AI BIBLE

BONUS 6 OF 7

COMPANION TOOLKIT

The Claude Code Project Starter Pack

Kickoff prompts · CLAUDE.md templates · 5 guided mini-projects

Everything you need to sit down at Claude Code and build a real, finished thing this week — even if you have never written a line of code.

A free companion toolkit for

The Complete Claude AI Bible (7-in-1)

David M. Patel

WHAT'S INSIDE

Your Toolkit, Section by Section

01 How to Use This Toolkit

02 The Mindset: Describe Outcomes, Not Code

03 Getting Set Up

04 The Kickoff Prompt Sequence

05 The CLAUDE.md Template

06 Five Guided Mini-Projects

07 Debugging With Claude

08 Good Habits & Safety

09 Your Next Step

SECTION 01

How to Use This Toolkit

You've just finished Book 6, where you learned the most liberating idea in this series: you don't need to understand code — you need to understand what you want. This toolkit turns that idea into ready-to-use tools you can keep beside your keyboard while you build.

Think of it as a builder's starter kit: the exact prompts to kick off any project, a fill-in-the-blanks template for the `CLAUDE.md` memory file, five complete mini-projects to copy and follow, and the habits that keep your building safe. Nothing here is theory — everything is something you can paste, adapt, and use.

Here is the gentle way to work through it:

- **Read Section 2 first.** It sets the mindset that makes all of this work. Five minutes, and everything else lands more easily.
- **Skim Section 3** to make sure you're set up — the desktop app and a project folder are all you need.
- **Keep Section 4 open while you build.** The kickoff prompt sequence is your reliable on-ramp for any project, large or small.
- **Pick one mini-project from Section 6 and actually do it.** Reading about building and building are different experiences; only one of them gives you something real at the end.

> **TIP**
> Don't try to read this cover to cover and then start. Read Sections 2 and 4, then jump straight into a mini-project. You'll learn far more by building one small thing than by studying ten.

The brackets you'll see in every prompt — like [your name] or [describe the look you want] — are for you to fill in with your own details. Replace them before you send the prompt. That's the only "coding" you'll ever do here: filling in your own words.

SECTION 02

The Mindset: Describe Outcomes, Not Code

Before any prompt, the mindset. Book 6 put it plainly: the people who thrive with Claude Code treat themselves as the director of the project, not the technician. They say what they want and what "good" looks like, and let Claude handle the how.

This is the single shift that makes building possible for someone who has never coded. A programmer thinks in the computer's language; you think in your own, about the result you're after. "A clean one-page site with my photo, a short bio, and a contact button" is a complete, professional instruction. You never translate it into code, because translating intention into working software is exactly Claude Code's job.

> **KEY IDEA**
>
> You don't need to understand what the code does — you need to understand what you want. Claude Code handles the translation from your intention to the working program. Keep your attention on the outcome — does the page look right, does the button work? — and let the loop carry you forward.

There are two traps that stop newcomers, and naming them helps you avoid them:

- **Freezing because you don't understand the technical details.** You don't have to. Just as you can drive a car without understanding the engine, you can build with Claude Code without reading the code. If something is wrong, you describe the problem; you don't fix it yourself.
- **Giving up because the first attempt wasn't perfect.** The first version is a starting point, not a verdict. Everything is refined in conversation, one turn at a time.

Hold the director's posture and three short rules carry you through almost anything:

- **Be the director:** say what you want and what good looks like.
- **Expect to iterate:** the first version is where you start, not where you stop.
- **Don't fear the code:** you can build something excellent without reading a single line of it.

SECTION 03

Getting Set Up

You need very little to begin, and the friendliest path is the one Book 6 walks you down: the redesigned Claude Code desktop app, released in 2026 to be approachable rather than intimidating. It gives you a clean, visual workspace instead of the bare text terminal that gave Claude Code its technical reputation.

What you need

- **At least a Claude Pro plan.** Claude Code isn't part of the free tier, so confirm you're on Pro (or higher) before you start.
- **The desktop app.** Go to claude.com/download, choose the Claude Code desktop app for your system (macOS or Windows), run the installer like any other program, and sign in with your Claude account.

That's the whole installation. There's also a way to install Claude Code in the terminal for people who prefer it, but you don't need that path and this toolkit never uses it. The desktop app does everything here and is far gentler.

The project folder idea

This is the one concept worth understanding clearly. Claude Code works inside a folder on your computer — a project folder — where it creates and edits the files that make up whatever you're building. Every project lives in its own folder.

So the rhythm of starting anything is always the same:

1. Make a new, empty folder somewhere easy to find, and give it a clear name (like `my-portfolio` or `tip-splitter`).
2. Open that folder in Claude Code.
3. Describe what you want in plain English in the message box.

From there, the describe-build-refine loop runs everything: you describe, Claude builds and shows you a preview, you look, you describe the next change — around and around until it's right.

TIP

Before your first real project, do the five-minute warm-up from Book 6. Make a folder called my-first-build, open it, and send: "Create a simple web page that says 'Hello, I'm [your name]' in large friendly text, centered, on a soft blue background." Watch it build, then ask for one change — a different colour, bigger text. Feeling the loop work once, on something trivial, teaches the rhythm better than any amount of reading.

SECTION 04

The Kickoff Prompt Sequence

Here is the reliable on-ramp for any build. Instead of dumping everything on Claude at once and hoping, you move through four short stages: plan, confirm, build step-by-step, review. This keeps you in the director's seat and gives you a working version at every stage rather than a half-finished tangle.

Copy these in order. Fill in the brackets with your own project.

Step 1 — Ask for a plan before building

You'll get a far better result if Claude thinks through the shape of the project before writing anything. This also lets you catch misunderstandings while they're cheap to fix.

PROMPT · STEP 1 — edit the brackets

```
I want to build [describe the thing — e.g. a one-page website for my photography, or a simple budget calculator].

Before you build anything, give me a short plan in plain language:
- What you'll create
- The main sections or features
- How I'll be able to see and try it

Keep it simple and don't write any code yet. I'll confirm before we start.
```

Step 2 — Confirm and adjust the plan

Read the plan. If it matches what you pictured, say so. If not, steer it — this is the cheapest possible moment to change direction.

PROMPT · STEP 2 — edit the brackets

```
That plan looks mostly right. Two changes:
- [change one — e.g. "make the contact section a simple button, not a form"]
- [change two — e.g. "add a section for testimonials"]

Once you've adjusted, go ahead and build the first version, then show me a preview.
```

Step 3 — Build step-by-step

For anything with more than one part — especially tools — ask Claude to build in clear stages so you can see and test each piece. This is the "chain the task" idea from Book 1, applied to building.

PROMPT · STEP 3 edit the brackets

```
Let's build this one piece at a time so I can follow along.

Start with just [the core piece — e.g. "the basic layout with my name and tagline"
or "the calculation that takes the three inputs and shows the result"].

Show me a preview of that first. Once I've checked it, we'll add the next piece.
```

Step 4 — Review what was built

After each preview, review like a director — judging the result, not the code. Describe what to keep and what to change.

PROMPT · STEP 4 edit the brackets

```
I've looked at the preview. Here's my review:
- This works well: [what you like]
- Please change: [specific, concrete change — e.g. "use a warmer colour, make the
heading bigger, add a little space around the edges"]
- Please add next: [the next piece]

Make those changes and show me the updated preview.
```

NOTE

"Specific requests get specific results" is the whole skill. "Make it nicer" gives Claude nothing to work with. "Use a warmer colour, make the heading bigger, and add a little space around the edges" gives it everything. The clearer your description, the better the build.

SECTION 05

The CLAUDE.md Template

As a project grows, it helps Claude Code to remember things about it without you repeating yourself each session. It does this with a simple file, named `CLAUDE.md` by convention, that lives in your project folder and acts as the project's memory. Every time you open the project, Claude reads this file and instantly knows the context.

You never have to write this file by hand — that wouldn't fit the no-code spirit of any of this. You describe what should be remembered, and you ask Claude to record it for you. The template below is what a good project memory contains; paste it into Claude with your own details filled in, and let Claude create the file.

CLAUDE.MD TEMPLATE — fill the brackets, then send

```
Please create a CLAUDE.md file for this project's memory, using these details:

PROJECT: [project name — e.g. "My Portfolio Site"]

WHAT THIS IS:
[One or two sentences. E.g. "A clean one-page personal portfolio website for my
freelance design work. The goal is to look professional and load fast."]

WHO IT'S FOR:
[Who will see or use this. E.g. "Potential freelance clients and recruiters."]

STYLE AND TONE:
[How you like things to look and feel. E.g. "Clean and minimal. Soft cream and dark
green colours. Calm, professional tone. No clutter."]

STANDING INSTRUCTIONS — ALWAYS KEEP IN MIND:
[Rules Claude should never forget. E.g.
- "Always use my name, [your name], in the footer."
- "Keep it working well on phones, not just computers."
- "Use British spelling."]

CONTENT I'VE GIVEN YOU:
[Things you've already provided. E.g. "My bio, my three project descriptions, and my
photo are in this folder. Use them rather than placeholder text."]

Please save this as CLAUDE.md in the project folder, and from now on follow it in
everything you build for me.
```

Here's what each part is doing, so you can adapt it confidently:

SECTION	WHAT IT'S FOR
What this is	The one-line purpose. Stops Claude from drifting off-target as the project grows.
Who it's for	The audience shapes a hundred small choices — tone, layout, what to emphasise.
Style and tone	Set your look and voice once, so you don't re-describe them every session.
Standing instructions	The "always" and "never" rules — your name in the footer, phone-friendly, your spelling.
Content I've given you	Points Claude at your real material so it stops using placeholders.

EXAMPLE

A simple one-liner you can send any time, even mid-project: "Remember that this is my personal portfolio site, keep the style clean and minimal, and always use my name in the footer." Claude records it for you. That's project memory in a single sentence — you don't have to use the full template if you don't want to.

TIP

For tiny builds you can ignore `CLAUDE.md` entirely. It earns its keep once a project has enough parts that re-explaining it each session would be tedious. When you notice yourself repeating the same context, that's the moment to ask Claude to remember it.

SECTION 06

Five Guided Mini-Projects

These are five complete, follow-along builds. Each one is small, useful, and finishable in an afternoon or less. For each you'll find who it's for, the prompt sequence to build it, and what to expect along the way. Pick the one that genuinely appeals to you — caring about the result is what carries you to the finish.

Mini-Project 1 — The One-Page Website

Who it's for: Anyone who wants a simple, shareable place online that represents them — for a job search, a side business, a creative pursuit, or just a polished link to send. This is the classic first build: quick, real, and instantly yours.

The lovely thing about a site is that you don't have to know how to brief it. Claude Code interviews you first, drawing the vision out one question at a time.

PROMPT · START — copy & send

```
Help me build a clean, modern one-page portfolio website.

Interview me first to understand what I want — ask me one question at a time about
what the site is for, who I am, what content I have, and how I'd like it to look.
Then build a first version and show me a preview.
```

Answer in plain language, as you would to a helpful designer — short answers are fine. Then refine by conversation, never touching the design tools or code:

PROMPT · REFINE — edit the brackets

```
The first version is a good start. Please make these changes:
- Use a warmer colour scheme — soft cream and dark green.
- Make my name bigger and add this tagline underneath: [your tagline].
- Move the projects section above the about section.
Then show me the updated preview.
```

Add your real content the same way — paste your actual bio, name your projects, point to a photo:

PROMPT · CONTENT — edit the brackets

```
Here's my real content to replace the placeholders:
- Bio: [paste your short bio]
```

```
- Projects: [list your projects or interests]
- Use this photo for the about section and round its corners: [point to the photo]
```

Finally, publish it for free:

PROMPT · PUBLISH copy & send

```
Help me publish this site for free and give me the link. Walk me through each step,
tell me exactly what to click, and do the technical parts yourself.
```

WHAT YOU GET BACK

- → You answer the interview questions in plain English.
- → Claude builds a first version and shows a preview in moments.
- → You refine the colours, layout, and text by describing changes.
- → You paste in your real bio, projects, and photo.
- → Claude guides you through a free host (GitHub Pages or Vercel), doing the technical parts.
- → You end with a real web address, like your-name.github.io, that you can send to anyone the same day.

NOTE

Publishing is the step that feels most technical, and it's the step where Claude Code helps the most. Both GitHub Pages and Vercel host simple sites for free. You ask, you follow along, and within minutes you have a live link.

Mini-Project 2 — A Simple Tool (Calculator or Tracker)

Who it's for: Anyone who wants to build something that does something — takes input from a person and gives back a useful result. A tip splitter, a savings-goal calculator, a habit tracker, a reading log. Pick one you'd actually use.

The trick with a tool is to describe behaviour as well as appearance: name what the user puts in and what they get out.

PROMPT · START copy & send

```
Build a simple, clean tip-splitting calculator as a web page.

The user enters the bill total, a tip percentage, and the number of people. When
they do, it shows the total tip, the grand total, and the amount each person owes.
Make it friendly and easy to use on a phone.
```

```
Show me a preview I can try.
```

Then build it up one feature at a time — testing after each, never asking for everything at once:

PROMPT · FEATURE copy & send

```
That works. Now add one feature: quick-pick buttons for common tip amounts — 15%,
18%, and 20%. Keep everything else the same and show me the preview.
```

PROMPT · FEATURE copy & send

```
Good. Next feature: if the bill or number of people is left blank, show a friendly
message instead of an error. Show me the preview when it's done.
```

WHAT YOU GET BACK

- → You type a bill total, tip %, and number of people.
- → The tool instantly shows total tip, grand total, and per-person amount.
- → You add quick-pick tip buttons (15% / 18% / 20%) — test, it works.
- → You add a friendly blank-field message — test, it works.
- → There's always a working version in front of you, never a half-built tangle.
- → You stop the moment it's good enough.

WATCH OUT

Resist the urge to pile on features. A tool that does one thing well beats one that does five things confusingly. For a tip splitter, splitting a bill clearly is the whole job. Build what serves the single purpose and leave the rest out.

Mini-Project 3 — A Small Automation

Who it's for: Anyone with a small, repetitive computer chore — renaming a batch of files, reformatting some data, tidying a spreadsheet. These little programs (scripts) quietly save you time, and Claude Code can build them by description.

Be concrete about the repetitive task and the rule for handling it:

PROMPT · START edit the brackets

```
I have a folder full of files I need to rename in bulk. Help me build a small script
that does this:
- Takes every file in the folder named [pattern, e.g. "IMG_1234.jpg"]
- Renames them to [the format you want, e.g. "holiday-2026-001.jpg", numbered in
order]

Before it runs, explain in plain language what it will do, and let me test it on a
copy of the folder first so nothing important is at risk.
```

Once you've seen it work safely on a copy, you can refine the rule:

PROMPT · REFINE copy & send

```
Almost right. Two adjustments:
- Keep the original file's date in the new name.
- Skip any file that already follows the new naming format.
Show me what the new names would be before actually renaming anything.
```

WHAT YOU GET BACK

- → Claude explains, in plain English, exactly what the script will do.
- → You run it on a COPY of your folder first.
- → You check the results — are the files named the way you wanted?
- → You refine the rule (keep dates, skip already-renamed files).
- → You preview the new names before committing.
- → Once it's right, you run it on the real folder, confident it's safe.

WATCH OUT

Automations change real files on your computer, which makes them the one project type where caution matters most. Always test on a copy first, and always have Claude show you what would happen before it actually happens. Back up the real folder before any bulk change.

Mini-Project 4 — Fix or Improve Something

Who it's for: Anyone who has already built something — a site or a tool — and wants to fix a bug or add a feature. A project is rarely finished the day you build it, and iterating is a normal, healthy part of making things.

Returning to a project is easy: open its folder in Claude Code and describe what you want different. Because Claude can see everything that's already there, it builds on your work rather than starting over.

First, lay down a safety net before any big change:

PROMPT · SAFETY NET copy & send

```
Before we make changes today, save a snapshot of the current working version, so we can go back to it if anything breaks.
```

Then describe the fix or the improvement in plain language:

PROMPT · CHANGE edit the brackets

```
Two things I'd like to change in this project:
1. Fix: the contact form isn't sending when someone clicks submit — please find the problem and fix it.
2. Improve: add a dark-mode toggle so visitors can switch between light and dark.

Make these changes and show me the updated preview so I can test them.
```

If a change goes wrong, you're not stuck:

PROMPT · RESTORE copy & send

```
That redesign isn't working the way I hoped. Please restore the version from before today's changes.
```

WHAT YOU GET BACK

- → Claude saves a snapshot of the working version (your safety net).
- → You describe the fix and the improvement in plain words.
- → Claude works on the existing project, changing it in place.
- → You test the contact form and the dark-mode toggle in the preview.
- → If anything broke, you ask to restore the earlier snapshot — and you're back to safe ground instantly.

KEY IDEA

You get version control — the professional habit of keeping saved snapshots so you can always return to a working version — without learning any of its machinery. You just ask, in plain English: "save a snapshot before this change," and "restore the version from before today." Claude manages the rest.

Mini-Project 5 — A Personal Dashboard

Who it's for: Anyone who wants a single page that pulls together the things they check often — a small "home base." This combines what you've learned: a site to look at and interactive pieces that do something. It's a satisfying step up, still well within reach.

Start with the plan-first approach, because a dashboard has a few parts:

PROMPT · PLAN — edit the brackets

```
I want to build a simple personal dashboard as a one-page website — a home base I
can open each morning.

Before building, give me a short plan in plain language. I'd like it to include:
- A friendly greeting with the current date
- A short to-do list I can add to and check off
- A space for [something you check often — e.g. "a few favourite links", "a note to
myself", "a simple habit tracker"]

Keep it clean and easy to read. Don't write code yet — show me the plan first.
```

Confirm the plan, then build it one piece at a time:

PROMPT · BUILD — copy & send

```
The plan looks good. Build it one section at a time. Start with just the greeting
and the date, and show me a preview. Once I've checked it, we'll add the to-do list,
then the rest.
```

Add the interactive piece and test it like a user:

PROMPT · INTERACTIVE — copy & send

```
Now add the to-do list. I should be able to type a task, press add, see it on the
list, and tick it off when it's done. Show me the preview so I can try it.
```

WHAT YOU GET BACK

- Claude shows you a plan; you confirm it.
- It builds the greeting and date first — you preview and approve.
- It adds the to-do list — you type a task, add it, tick it off, test it works.
- It adds your final section (links, note, or habit tracker).
- You refine the look until it feels like a calm, useful home base.
- Optionally, you publish it so it's one bookmark away every morning.

TIP

A dashboard is the perfect project to give a `CLAUDE.md` memory file (Section 5). Record what each section is for and your style preferences once, and every future tweak stays consistent without you re-explaining.

SECTION 07

Debugging With Claude

When something is wrong — a tool miscalculates, a button doesn't work, a page looks cramped on a phone — you don't fix it yourself. You report it to Claude in plain language, just as you'd tell a colleague. Claude diagnoses and repairs it, and you test again.

The whole skill of debugging here is describing the problem well. Three things make a report easy for Claude to act on:

- **What you did** — the action you took.
- **What you expected** — what should have happened.
- **What actually happened** — what went wrong instead.

You don't need technical words. Even pasting an error message you don't understand is helpful; Claude reads it for you.

PROMPT · BUG REPORT — edit the brackets

```
Something's not working. Here's what happened:
- What I did: I left the tip field blank and entered a $50 bill for 2 people.
- What I expected: it should treat the tip as zero.
- What actually happened: it shows "NaN" instead of an amount.

Please find the problem and fix it, then show me a preview so I can re-test.
```

PROMPT · DISPLAY ISSUE — copy & send

```
A display problem: on my phone, the buttons are too small to tap comfortably, and
the text runs off the edge of the screen. On my computer it looks fine. Please fix
it so it works well on a phone, and tell me when to re-check.
```

EXAMPLE

A complete, real-feeling bug report needs no jargon: "When I click 'calculate' with the number-of-people field empty, the whole page goes blank. I expected a gentle 'please enter the number of people' message. Please fix that." That's everything Claude needs.

After every fix, test again — the same way a user would. Round by round, this build-test-fix rhythm is how a rough draft becomes something you can trust.

SECTION 08

Good Habits & Safety

A few simple habits keep your building calm, safe, and steady. None of them are technical; all of them save you grief.

Review before you run

Before you let an automation or script loose on your real files, read Claude's plain-language explanation of what it will do, and run it on a copy first. You are the director — nothing happens that you haven't looked at and approved.

Back up before big changes

When you're about to change something that already works, ask Claude to save a snapshot first. This is the building equivalent of the backups you keep before any risky change. "Save the current version before we redesign the homepage" costs you one sentence and buys you the freedom to experiment fearlessly — if a change makes things worse, you simply ask to restore the earlier version.

Start small and grow

Don't try to build everything in one heroic leap. Start with the core, get it working, then add one feature at a time, testing after each. There's always a working version in front of you, and you can stop the moment it's good enough.

WATCH OUT

Know your limits — this is wisdom, not a restriction to resent. Claude Code is superb for personal sites, small tools, prototypes, and automations. It is not the right path for software that stores people's sensitive personal, financial, or health data at scale, serves thousands of paying customers, handles payments reliably, or runs a business's critical systems. Those still need professional developers and proper engineering. Recognising that line protects you and the people who'd rely on your work — and when you do reach a project that's too big, the working prototype and clear description you've built make you the informed client a developer wishes for. Ask Claude Code to help you prepare that hand-over; your work is never wasted.

SECTION 09

Your Next Step

Take a moment with what this toolkit puts in your hands. You can plan a build, kick it off with a reliable sequence of prompts, give a project a memory, follow five complete recipes, debug in plain language, and keep it all safe. You are, in a real and modern sense, a builder.

So here is your next step, and it's the one that makes everything real: pick one mini-project and actually finish it this week. Build it, refine it, and — if it's a site or a public tool — publish it and send the link to one person. The moment someone else opens something you built, you've crossed the line from learner to builder.

Then there's one final step in the journey. You've learned to build real things with Claude — now Book 7: The Claude Income & Career Playbook shows how to turn that skill into something that pays. Freelancing, digital products, consulting, and using Claude brilliantly at the job you already have: the building skills in this toolkit are exactly the raw material Book 7 turns into income and career leverage.

> **NOTE**
>
> For more templates, walkthroughs, and updates as Claude Code grows, visit the companion hub linked in your book. You began as someone curious about AI; you're ending this part of the journey as someone who can build with it — and the next book is where that starts working for you.