

# Build Software With AI

From idea to paying users without a dev team. One tool-independent method for shipping web apps, internal tools, SaaS and automations with Claude Code, Cursor, Copilot, Codex, Lovable, Bolt, v0 or Replit: specs, prompts, architecture, debugging, testing, security, payments, deployment, costs and maintenance. With a 90-Day Build Plan and templates.

**Manthan Bhendkar** · Founder, Asendify  
[asendify.co](https://asendify.co)

# Read this first

What AI coding tools can and cannot build for you in 2026, and the one method this book teaches so that what you build actually ships, gets paid for and survives.

## Who this is for

You want software to exist and you do not have a development team. You might be a founder with a product idea, an operator who needs an internal tool that no off-the-shelf app quite does, a freelancer or agency owner whose clients keep asking for "a portal", or someone who has been typing prompts into an AI builder for weeks and keeps hitting the same walls. You do not need to be a programmer. You do need to be willing to read, to test what you are given, and to own the result.

Read the book by your goal:

| Your goal                                | Read first      | Then                      |
|------------------------------------------|-----------------|---------------------------|
| A SaaS (software people pay for monthly) | Parts 1–8       | 10, 13, 14, 15, 16, 19    |
| An internal tool for your own business   | Parts 1–3, 5–8  | 10, 11, 18                |
| A client project (freelancer or agency)  | Parts 1–5, 7–8  | 12, 13, 15, 18 (handover) |
| An automation (a report, a sync, a bot)  | Parts 1–3, 5, 8 | 11, 17, 18                |

Whatever your goal, Parts 1, 2, 5 and 8 are the spine. Skip them and the rest will not hold.

## The honest picture

AI coding tools are very good at a specific kind of work: producing plausible code quickly from a clear description, in popular languages and frameworks, for problems thousands of other people have already solved. Log-in pages, dashboards, forms, database tables, payment checkouts, email sending, the glue between two services. That is most of what a small business needs, which is why this moment is real.

They are also unreliable in specific, predictable ways. They invent functions, settings and whole software packages that do not exist. They agree with you when you are wrong. They forget what they did an hour ago. They "fix" one thing by quietly breaking another. They say a feature is finished when it only looks finished. Replit's pricing page says its Agent's behaviour "is probabilistic" and it "may occasionally make mistakes"; Claude Code's documentation says you are "responsible for reviewing proposed code and commands for safety before approval". Every serious vendor says a version of this. Believe them.

So "it works on my screen" is not shipped. Shipped means a stranger can use it on their phone, with their own account, pay for it, see nobody else's data, and come back next week to find it still running. The gap between the demo and that sentence is where most AI-built projects die. This book is about crossing it.

## Vibe coding versus building

"Vibe coding" (a phrase coined by Andrej Karpathy in February 2025, and Collins Dictionary's Word of the Year for 2025) means describing what you want, accepting whatever the AI produces, and pushing on by feel. It is a fine way to make a throwaway prototype or to learn. It is a poor way to make something people pay for, because three things are missing:

- **A spec.** A written description of what the software must do and must not do. Without it, every prompt is a new guess and there is nothing to test against.
- **Verification.** Checking that the thing does what the spec says, under the conditions real users create: two people at once, a double click, a refund, an expired card.
- **Ownership.** Your code in your own repository, your data exportable, your decisions written down, your secrets kept secret. If a tool changes its pricing or shuts down, you carry on.

### THE ONE IDEA

You are not learning to code. You are learning to run a small software project in which the AI does the typing. Your job is to decide, describe, check and own. The method below is how.

## The Build Loop and the four project files

Everything in this book hangs on one repeating cycle, which we call the **Build Loop**:

1. **Spec** — write down what you are building and how you will know it works.
2. **Plan** — turn the spec into an ordered list of tasks.
3. **Slice** — cut the next task small enough for one AI session and one change you can review.
4. **Build** — give the AI that one slice, with the context it needs.
5. **Verify** — test the result against the "done when" line, yourself and with automated checks.
6. **Commit** — save that working state in version control so you can always go back to it.
7. **Ship** — put it on a preview address, then on production, when a slice is worth releasing.

Then back to the top. The loop is deliberately boring; boring is what works when you are tired and the AI is confidently wrong. It runs on **four project files**, all plain text, kept in the project folder and saved with the code:

END OF THE FREE PREVIEW

# Build Software With AI

You've read the first 2 pages. The full book has 109 pages, with the complete method, the plan and every worksheet.

**Get the full book**

[buildai.asendify.co](https://buildai.asendify.co)