

WHAT IS SPEC-DRIVEN DEVELOPMENT?

The super simple version. You write down what you want before anybody writes code, you cut it into numbered prompts, you feed them to the builder one at a time, and a watcher checks every step. That is the whole trick.

WHAT'S INSIDE

Most people talk to an AI coding agent like a slot machine: pull the lever, hope for a website. Spec-driven development is the opposite. You decide first, then the AI builds exactly what was decided, in small checked steps.

- 00 **The one-sentence version.** What it is and why it works
- 01 **The words.** Six terms, said once
- 02 **Write the spec.** Grok Bot interviews you and writes it down
- 03 **Cut it into prompts.** Numbered, one job each
- 04 **Feed them one at a time.** Grok Build builds, commits, reports
- 05 **Add the watchers.** A second AI checks every step
- 06 **When it breaks.** Roll back, fix, keep going
- 07 **The cheat sheet.** The whole loop on one page
- 08 **Links.** Every tool

HOW TO USE THIS GUIDE

Every grey box is a prompt: copy all of it. Text in [red brackets] is yours to fill in. You need two things: Grok Bot (the planner and checker) and Grok Build (the builder). Both are linked at the back.

REAL TALK This is how the AntiHero guides site, the Aura Homes site and The Hitchhiker's Guide to Web Design plan were built. Nothing fancy. Just very boring discipline that produces very un-boring results.

THE IDEA

00

Spec-driven development means the plan is the product. The code is just the plan, typed out.

“Write down what you want. Cut it into small numbered jobs. Do one job at a time. Check every job before the next.”

THE WHOLE METHOD

WHY IT WORKS

- **AI agents are brilliant builders with no memory and no taste.** A spec gives them both. Every prompt carries everything the builder needs, so a fresh session never has to guess.
- **Small jobs fail small.** If prompt 14 goes sideways you lose prompt 14, not the weekend.
- **You stay the boss.** You make the decisions once, in plain English, in the spec. After that you approve or reject. No arguing with a chatbot at 2am about why the button is purple.
- **Anyone can check it.** Because every job has a definition of done, a second AI (the watcher) can grade it. So can you.

VIBE CODING VS SPEC-DRIVEN

	VIBE CODING	SPEC-DRIVEN
How you start	"Build me a website"	A written spec: goal, audience, pages, rules
What the AI gets	One giant wish	One small job at a time, numbered
When it breaks	Start over, cry	Roll back one commit, re-run one prompt
Who decides	Whatever the AI felt like	You, once, in writing
Who checks	Nobody	A watcher, after every single step

THE WORDS

01

Six words. That is the entire vocabulary.

SPEC	The written plan. What you are building, for whom, what it must do, what it must never do. One document.
PROMPT PACK	The spec cut into numbered files: 001, 002, 003. Each one is a single job, written so a brand new AI session can do it with zero context.
GROK BOT	Your planner and checker. It interviews you, writes the spec and the prompt pack, and watches the build. x.ai/bot
GROK BUILD	xAI's coding agent. It runs in a terminal inside your project folder and actually writes the files. x.ai/cli
WATCHER	A second AI (usually Grok Bot) that reads what the builder did after every prompt and says pass, fail, or here is the fix.
COMMIT	A saved checkpoint in git. One prompt, one commit. Something breaks, you step back one.

WHAT A GOOD PROMPT FILE CONTAINS

Goal (one sentence). Files to read first. Files to create or change. Step-by-step actions. Acceptance criteria (how we know it is done). Verification commands. What not to touch.

FROM THE REAL THING The Hitchhiker's Guide plan has 209 prompt files built exactly like this, with a reviewer checkpoint after every three build prompts.

THE PLAN.

Nobody writes code in part one. You talk, the AI listens, and you end up with a spec and a numbered prompt pack.

WRITE THE SPEC

02

Open one long chat with Grok Bot so it remembers everything you decide. Attach anything you already have: brand brain, logo, old site, notes. Then let it interview you.

PROMPT 01 · INTERVIEW ME, THEN WRITE THE SPEC

You are a senior product lead. Interview me to write a spec for **[what you are building: a website, an app, a tool]**.

Ask one question at a time and wait for my answer. Push back if an answer is vague.

Cover: what it is for in one sentence, who uses it, the main things it must do, what it must never do, what already exists, what must not change, the look and feel, the tech (or pick it for me and explain why in one line), hard limits (time, money, hosting).

When you have enough, write SPEC.md:

1. Goal and audience
2. Features, each with a definition of done
3. Non-negotiable RULES (one paragraph I will paste on top of every build prompt)
4. Protected files and things nobody touches
5. Open questions

Then ask me what you got wrong.

PRO MOVE

Answer out loud with voice typing. You say better stuff than you type, and you stop editing yourself halfway through a thought.

RULE If you cannot explain a feature in one sentence, it is not ready to be a prompt. Keep talking until you can.

CUT IT INTO PROMPTS

03

Now Grok Bot turns the spec into a stack of numbered prompt files. Small, boring, self-contained. That is what makes them work.

PROMPT 02 · WRITE THE PROMPT PACK

Read SPEC.md. Cut it into numbered build prompts for Grok Build, one file per prompt: prompts/001-[short-name].md, 002-..., and so on, in the order they should be built. Setup and backup first. Structure before polish. Effects last.

Every prompt must be self-contained (a brand new session can run it) and include:

- the RULES paragraph on top
- Goal in one sentence
- Read first: the exact files to read before changing anything
- Change: the files to create or edit, and nothing else
- Steps
- Done when: acceptance criteria I can check
- Verify: the commands to run (build, test)
- Do not: push, deploy, delete, or touch protected files
- Finish with one git commit and a short report

After every 3 build prompts, add a REVIEW prompt that checks the last 3 against SPEC.md. Also write prompts/INDEX.md: a table of number, title, and effort (medium / high / xhigh).

SIZE THE JOB	EFFORT TAG	MODEL
Copy, a simple section, a config file	medium	Fast model is fine
A feature with logic, a page	high	Full model
3D, refactors, anything "architecture"	xhigh	Full model, extra effort

Check the INDEX before you run anything. If a prompt title has the word "and" in it twice, ask Grok Bot to split it.

PART TWO

THE

BUILD.

Now the builder builds, the watcher watches, and you do the most important job in the room: say yes or no.

THE LOOP

04

One prompt in, one commit out, one check. Then the next. That is the whole build.

INSTALL GROK BUILD, THEN OPEN IT IN YOUR PROJECT

```
# Mac or Linux
curl -fsSL https://x.ai/cli/install.sh | bash
# Windows, in PowerShell
irm https://x.ai/cli/install.ps1 | iex
# then, inside your project folder
grok
```

- 1 Paste prompt 001.** Let it work. Do not help. Do not touch the files while it works.
- 2 It commits.** That is your checkpoint.
- 3 The watcher checks.** Done or not, against the prompt's own "Done when".
- 4 You look.** Open it on localhost, hard refresh (Ctrl+Shift+R, Cmd+Shift+R on Mac).
- 5 Pass? Next number.** Fail? Paste the watcher's fix prompt, or roll back one commit.

FRESH SESSION, EVERY PROMPT

Start each numbered prompt in a fresh Grok Build session. Because each file is self-contained, the builder never drags yesterday's confusion into today's job.

POWER USER Grok Build can also take a prompt file directly, headless, so prompts can run back to back with nobody typing. Same rule: one file, one job, one commit, then a check. See the CLI reference at the back.

ADD THE WATCHERS

The builder grades its own homework a bit generously. The watcher does not. Give this to Grok Bot once, then paste each step's report into it.

PROMPT 03 · THE WATCHER

You are my build supervisor. Here is SPEC.md and the prompt pack: **[attach them]**.
Grok Build runs the prompts one at a time. After each one I will paste its final report and the output of ``git show --stat HEAD``.

For each step tell me:

1. Done or not done, against that prompt's "Done when".
 2. Did it touch any protected file? Name it.
 3. Did it break the RULES (deploys, deleted files, banned words, wrong colors)?
 4. What I should look at on localhost before moving on.
 5. If something is wrong: the exact fix prompt to paste back into Grok Build.
- Then tell me the next prompt number. Keep it short.

KINDS OF WATCHERS

- **Step watcher.** Checks every commit. The prompt above.
- **Review checkpoint.** Every three prompts, a REVIEW prompt compares the last three against the spec. Catches drift early.
- **Once-over.** At the very end, one extra-high-effort pass over the whole thing: missing features, broken links, ugly mobile.
- **You.** The only watcher that knows what it is supposed to feel like.

FULL AUTOPILOT

If Grok Bot can see the project folder directly, tell it to run `git log` and `git diff` itself after each step. Now the builder builds, the watcher checks, and you only approve.

ROLL BACK, DON'T PANIC

Something will break. That is not the method failing, that is the method working. You only ever lose one step.

SEE YOUR CHECKPOINTS

```
git log --oneline
# then tell Grok Build, in plain English:
Revert the last commit. Keep a backup branch called backup-[date].
```

THE THREE USUAL SUSPECTS

SYMPTOM	CAUSE	FIX
It did more than asked	Prompt too big	Split it into two numbered prompts
It forgot a rule	RULES not pasted on top	Re-run with the RULES paragraph first
It keeps failing the same step	Spec is vague there	Back to Grok Bot: clarify that feature, rewrite the prompt

GOLDEN RULES

One prompt, one job, one commit. Never two builders in the same folder. Never edit files while the agent works. Never let anything push or deploy without you saying so.

TRUTH A spec you wrote in 30 minutes saves about a week of "no, not like that".

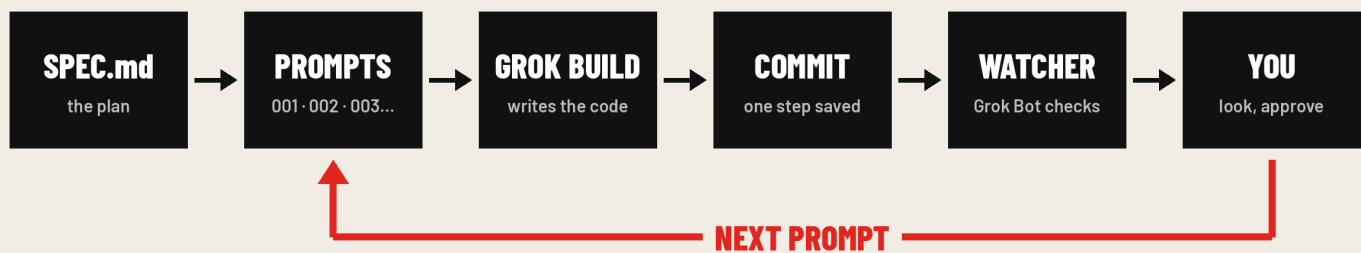
THE CHEAT SHEET

07

The whole method, start to finish.

- 1. Interview.** Grok Bot asks, you answer out loud.
- 2. Spec.** It writes SPEC.md. You fix what it got wrong.
- 3. Prompt pack.** It cuts the spec into numbered, self-contained prompts with review checkpoints.
- 4. Index.** You skim prompts/INDEX.md. Split anything too big.
- 5. Loop.** Prompt in, commit out, watcher checks, you look.
- 6. Fix or roll back.** Never lose more than one step.
- 7. Once-over.** One last careful pass over everything.
- 8. Ship.** Version 1 beats perfect. The version 2 list is very patient.

THE SPEC-DRIVEN LOOP



LINKS

08

Everything mentioned in this guide.

GROK

Grok Bot	x.ai/bot
Grok Build, install	x.ai/cli
Grok Build docs	docs.x.ai/build/overview
Grok Build CLI reference	docs.x.ai/build/cli/reference
Grok	grok.com

BASICS

Git	git-scm.com
Node.js	nodejs.org
GSD, the open spec-driven workflow behind the Hitchhiker plan	github.com/open-gsd

ANTIHERO

Websites on Autopilot guide (the full version of this loop)	facebook.com/groups/antiheroai
AntiHero AI group	facebook.com/groups/antiheroai

NOW GO

SPEC IT.

Homework: pick one small thing you have been meaning to build. Run the interview, get a spec, and post your prompts/INDEX.md in the group. Ten prompts is plenty for a first go. The AntiHero AI group is free, sells nothing, and will tell you honestly if prompt 4 is actually prompts 4 through 9.

[FACEBOOK.COM/GROUPS/ANTIHEROAI](https://facebook.com/groups/antiheroai)

