

GSD

The Beginner's Guide

*Build real projects with AI —
no coding experience required.*

For GSD v1.35.0

Tibsfox

*Released under CC BY-SA 4.0.
Authored by Tibsfox. GSD itself is MIT-licensed — see [gsd-build/get-shit-done](#).*

Contents

1 Who This Guide Is For	1
1.1 What Kind of Projects GSD Helps With	1
1.2 What You Don't Need to Know	1
1.3 What You'll Be Able to Do by the End	2
2 Setting Up	3
2.1 What You'll Need	3
2.2 Installing Node.js	3
2.3 Running the GSD Installer	4
2.4 What the Installer Asks	4
2.5 Verifying It Worked	4
3 What Is GSD?	5
3.1 The Problem: AI Forgets Mid-Conversation	5
3.2 The Solution: GSD Keeps the AI Organized	5
3.3 The Workflow, As a Story	6
3.4 What "Spec-Driven Development" Means	6
4 Your First Project — A Complete Walkthrough	7
4.1 Start a Project Folder	7
4.2 The First Command: <code>/gsd-new-project</code>	7
4.3 What GSD Asks You, and Why	8
4.4 What Gets Created	8
4.5 Understanding the Roadmap	9
4.6 Checking Where You Are	9
5 Building Your First Phase	10
5.1 What a "Phase" Is, Again	10
5.2 Step One: <code>/gsd-plan-phase</code>	10
5.3 Step Two: <code>/gsd-execute-phase</code>	11
5.4 Step Three: <code>/gsd-verify-work</code>	11
5.5 What to Do If Something Isn't Right	12
6 Navigating Your Project	13
6.1 <code>/gsd-progress</code> — "Where am I?"	13
6.2 <code>/gsd-next</code> — "What should I do next?"	13
6.3 <code>/gsd-help</code> — "What commands exist?"	13

6.4	The <i>.planning/</i> Directory	14
6.5	Pausing and Resuming	14
7	Quick Tasks	15
7.1	When to Use <i>/gsd-quick</i>	15
7.2	How to Use It	15
8	Tips for Success	17
8.1	Be Specific About What You Want	17
8.2	Let GSD Ask Questions	17
8.3	Don't Edit <i>.planning/</i> Files by Hand	17
8.4	Capture Ideas With <i>/gsd-add-todo</i>	18
8.5	Take Breaks	18
8.6	Celebrate Small Wins	18
9	What's Next?	19
9.1	Complete the Milestone	19
9.2	Start a New Milestone	19
9.3	When You're Ready for More	20
9.4	Getting Help	20
A	Troubleshooting	21
A.1	"Command Not Found"	21
A.2	"Something Went Wrong With the Project"	21
A.3	"I'm Completely Lost"	21
A.4	"The Phase Produced Something I Don't Want"	22
A.5	"Everything Is Fine, I'm Just Anxious"	22

Chapter 1

Who This Guide Is For

Welcome. If you've ever had an idea for a small tool, a personal website, a script that organizes your photos, or a little game—and you stopped because you thought “*I'd need to learn to code first*”—this guide is for you.

You don't need to learn to code. You need a good partner. GSD makes Claude, Anthropic's AI assistant, into that partner.

GSD stands for *Get Shit Done*. It's a small, free add-on for AI coding assistants that keeps the AI organized while it builds what you asked for. That's the whole pitch. You describe what you want in plain language. GSD helps the AI think carefully, plan the work, do the work, and check its own results. You stay in the driver's seat, but you don't have to be the mechanic.

1.1 What Kind of Projects GSD Helps With

GSD is happiest with software-shaped projects: websites, small apps, browser extensions, scripts that automate a chore, a tool that takes a spreadsheet and makes something useful out of it. It also does well with *creative* software: a little game, an interactive story, a digital zine, an art generator.

It can help you fix a bug in something you already have. It can help you read code someone else wrote and explain it to you. It can help you learn by doing.

It is *not* a magic wand that builds anything you can imagine with zero effort. You still need to know what you want, and you still need to answer questions the assistant asks you along the way. That part is good news: the better you describe your idea, the better the result.

1.2 What You Don't Need to Know

- You do not need to know how to program.
- You do not need to know what “git” is.

- You do not need to have used a terminal before.
- You do not need a computer science degree.
- You do not need to understand what’s happening “under the hood.”

You *will* be typing a few commands into a window, and I’ll walk you through every one of them. I’ll also tell you what each command does in plain English before you run it.

If something in this guide uses a word you don’t know, I’ll define it on the spot. If a chapter feels confusing, it’s the chapter’s fault, not yours. Skim ahead, come back, re-read. The commands are forgiving.

1.3 What You'll Be Able to Do by the End

By the time you reach the last page, you will have:

1. Installed GSD on your computer.
2. Started a real project with `/gsd-new-project`.
3. Watched the assistant build and check its own work.
4. Used `/gsd-progress` and `/gsd-next` to navigate the project.
5. Fixed things that weren’t quite right.
6. Finished a milestone and started a new one.

That’s a full loop. Once you’ve done it once, you’ve done it. Everything else is variations on the theme.

In the next chapter we’ll set everything up. It takes about fifteen minutes.

Chapter 2

Setting Up

You need four things. Three of them you probably already have.

2.1 What You'll Need

1. **A computer.** Mac, Windows, or Linux. GSD works on all three.
2. **An internet connection.** The assistant talks to Anthropic's servers to think.
3. **Claude Code.** This is Anthropic's coding assistant. It runs on your computer and is the thing GSD plugs into. You can get it at claude.com/claude-code. A free trial is usually available; a paid plan unlocks more.
4. **Node.js.** This is a free tool that lets your computer run the GSD installer. You only install it once.

2.2 Installing Node.js

Go to nodejs.org, click the big green "LTS" download button, run the installer, click through the defaults. That's it.

Node.js is a piece of software that runs small programs written in a language called JavaScript. GSD's installer is one of those small programs. You will not be writing any JavaScript yourself. Node.js is just the engine that starts the installer.

To check that it worked, open a terminal window.

- **Mac:** press `[Cmd]+[Space]`, type `terminal`, press `[Enter]`.
- **Windows:** open the Start menu, type `powershell`, press `[Enter]`.
- **Linux:** you already know.

A black (or light, if that's your taste) window opens with a blinking cursor. Type:

```
| node --version
```

Press [Enter]. You should see something like `v20.11.0`. If you do, Node.js is installed and you can close the terminal.

The exact numbers don't matter for this guide, as long as the first number is at least 18. If you see `v18`, `v20`, or `v22`, you're fine.

2.3 Running the GSD Installer

With Node.js installed, installing GSD is a single command. Open Claude Code (it gives you its own terminal prompt) and type:

```
| npx get-shit-done-cc@latest
```

Press [Enter]. The installer will ask you two questions.

2.4 What the Installer Asks

Question 1: Which runtime? The installer lists several AI coding assistants it supports. Pick **Claude Code**. (If you're using a different assistant, pick that one—the rest of this guide still applies; only the way you type commands changes very slightly.)

Question 2: Global or local? Pick **Global**. “Global” means GSD is available in every project on your computer. “Local” means only in the current folder. For a first project, global is easier.

The installer will print a few lines about copying files, then finish. That's it. GSD is installed.

2.5 Verifying It Worked

In Claude Code, type:

```
| /gsd-help
```

Press [Enter]. You should see a list of GSD commands scroll past. If you do, congratulations—GSD is working. If you don't, see Appendix A for troubleshooting.

The hardest part of this whole guide is now behind you. Everything from here on is just talking to your assistant.

Next up: a short chapter on *what* GSD is actually doing, so the commands in Chapter 4 feel like friends instead of strangers.

Chapter 3

What Is GSD?

You don't strictly need to read this chapter. You could skip to Chapter 4, run the commands, and have a project. But if you understand *why* GSD exists, the commands will feel obvious instead of arbitrary. It's a short chapter. Stick with me.

3.1 The Problem: AI Forgets Mid-Conversation

If you've used an AI chatbot for anything longer than a quick question, you've probably noticed something strange: it gets worse as the conversation gets longer. It forgets what you said earlier. It contradicts itself. It starts giving vaguer answers. People who use AI assistants for real work have a nickname for this: "*context rot*."

It happens because an AI assistant has a limited "conversation memory"—the amount of back-and-forth it can hold in its head at once. When the conversation fills up that memory, old things start getting squeezed out or blurred. Try to build a real project in one long chat and the assistant will lose the thread.

3.2 The Solution: GSD Keeps the AI Organized

GSD solves the memory problem in a clever way: instead of keeping one long conversation, it writes everything important down into files, and starts fresh conversations when it needs to. Each fresh conversation is given exactly the notes it needs and nothing more.

Think of it like this: imagine you hired a very skilled contractor to build a treehouse, but the contractor had a medical condition that made him forget everything at lunchtime. The only way to get the treehouse built is to:

1. Write down the plan in the morning while he still remembers the conversation.
2. After lunch, hand him the written plan and say "build exactly this."
3. Check what he built against the plan at the end of the day.

GSD is the clipboard. The assistant is the contractor. You are the homeowner who hired him. GSD keeps notes, hands out plans, and checks work, so the assistant can stay focused on one small task at a time and do it well.

3.3 The Workflow, As a Story

Here’s the whole GSD workflow in one paragraph:

You describe what you want. GSD asks you questions until it understands. It writes down everything it learned—what you want, how it’s going to build it, in what order. It breaks the work into phases. For each phase: it makes a plan, it does the work, it checks the work. When all phases are done, you ship a version. When you want to add more, you start a new version.

That’s it. That’s GSD. The commands in Chapter 4 each correspond to one step of that story.

3.4 What “Spec-Driven Development” Means

You might see the phrase *spec-driven development* in GSD’s documentation. It sounds jargony but the idea is homey: before you build something, write down what you’re building. The written description is called a “spec” (short for specification). Spec-driven development just means: write the spec first, then build from the spec.

GSD writes the spec *with you*, by asking questions. You don’t have to know how to write a spec. You just have to answer the questions honestly.

When GSD writes a spec, it lives in a folder called `.planning/` inside your project. You don’t normally open those files yourself, but they’re yours. They’re yours to read, yours to keep, yours to hand to another developer or another AI later.

Enough theory. Let’s build something.

Chapter 4

Your First Project --- A Complete Walkthrough

This is the longest chapter in the guide. Take your time. If you follow along on your own computer, you'll have a real project by the end.

For the walkthrough I'll pretend we're building a small personal website: a one-page site with a bio, a list of projects, and a contact link. Pick your own idea if you prefer—the steps are identical.

4.1 Start a Project Folder

Before you run any GSD commands, make a folder for your project. In the terminal:

```
mkdir my-website  
cd my-website
```

`mkdir` means “make directory” (a directory is the same thing as a folder). `cd` means “change directory”—it's how you walk into the folder you just made. After `cd my-website`, your terminal is “inside” that folder.

Use lowercase letters and dashes instead of spaces: `my-website`, `photo-tagger`, `birthday-quiz`. It makes everything easier later.

Now open Claude Code in this folder. (In most setups, Claude Code picks up the folder you started it in.)

4.2 The First Command: `/gsd-new-project`

This command is how you tell GSD “I have an idea and I want to start a project.” Type:

```
/gsd-new-project
```

Press [Enter]. GSD wakes up and starts asking questions.

4.3 What GSD Asks You, and Why

GSD's first job is to understand your idea well enough to build it. It does this by asking a series of questions. The exact questions vary, but they usually cover things like:

- What is the project? (One or two sentences is fine.)
- Who is it for? (Yourself? Your friends? Strangers on the internet?)
- What should it do when you're done? (Three or four bullet points.)
- Are there things that are explicitly *out of scope*? (Things you do *not* want, so the assistant doesn't gold-plate.)
- Is there anything it should look like or feel like? (A mood, a color, another site you admire.)

Answer each question the way you'd answer a friend. If you're not sure about an answer, say "I'm not sure, what do you recommend?"—the assistant will offer options.

The better you answer these questions, the better the final project. People who rush this step end up with projects that miss the mark. People who take their time end up surprised by how good the result is. Budget ten or fifteen minutes for the first question-and-answer session. You won't regret it.

4.4 What Gets Created

When the question session finishes, GSD will write a handful of files into a folder called `.planning/` inside your project. The important ones are:

- `.planning/PROJECT.md` — the summary of your idea. The thing you just described, written up cleanly.
- `.planning/REQUIREMENTS.md` — a more detailed breakdown of what "done" looks like.
- `.planning/ROADMAP.md` — the plan. A list of phases (numbered 1, 2, 3...) that together add up to your project.
- `.planning/STATE.md` — a living status file. GSD updates it as you progress.

You don't have to read any of these files. But if you're curious, open `PROJECT.md` in any text editor and read it. It should feel like your idea, written back to you.

If you want to change something in the plan—say you forgot to mention a feature—don't open `ROADMAP.md` and type into it. Instead, tell the assistant: *"I forgot to mention I want a dark mode toggle. Can you update the roadmap?"* It will update the files correctly and keep everything consistent.

4.5 Understanding the Roadmap

The roadmap is the most interesting of the files. It divides your project into **phases**. A phase is a bite-sized chunk of work that ends in something working. For a personal website, GSD might generate phases like:

1. **Phase 1:** Project skeleton and layout.
2. **Phase 2:** Bio and about section.
3. **Phase 3:** Projects list.
4. **Phase 4:** Contact section and polish.

Each phase is completed on its own. You finish Phase 1 (and *see* it working) before Phase 2 starts. This is how GSD keeps itself honest: small chunks, visible progress, no six-week silent marathon that ends in a pile of broken files.

If you start the first phase and realize the plan is wrong, you can stop and ask the assistant to revise the roadmap. Plans are meant to change.

4.6 Checking Where You Are

At any point, you can ask GSD to tell you where you are in the project:

```
| /gsd-progress
```

This prints a small status report: which phase you're on, what's done, what's next. It's a friendly "you are here" sign, and you'll use it constantly.

You now have a project, a plan, and a roadmap. Nothing has been *built* yet—that's what the next chapter is for.

Chapter 5

Building Your First Phase

GSD builds your project one phase at a time. In this chapter we'll walk through the three commands that make up a single phase: plan, execute, verify.

5.1 What a "Phase" Is, Again

A phase is a chunk of work with a clear beginning and end. For your first phase, GSD will pick something small enough to finish quickly but big enough that you can see real progress. In our website example, Phase 1 is usually just "set up the skeleton": a page that loads and shows "Hello, world" or similar.

Each phase goes through three steps:

1. **Plan** — GSD researches and writes a detailed plan for the phase.
2. **Execute** — GSD does the work in the plan.
3. **Verify** — GSD checks the work and asks you to double-check.

Let's go.

5.2 Step One: `/gsd-plan-phase`

Type:

```
| /gsd-plan-phase 1
```

Press [Enter]. (The 1 at the end means "phase 1." If you leave it off, GSD uses the next phase that needs planning, which is also fine for a first-time run.)

GSD now goes off and thinks. It may take a few minutes. It might ask you a clarifying question or two. When it's done, it will have written a detailed plan file for the phase into the `.planning/` folder. You don't have to read it, but if you do, you'll see something refreshingly clear: here's what we'll build, here's what each part does, here's how we'll know it worked.

Before running the next command, you can always ask the assistant: *“Can you summarize the plan for me in plain English?”* It will give you a short, readable summary. If anything sounds wrong, say so before moving on.

5.3 Step Two: `/gsd-execute-phase`

Once the plan exists, ask GSD to build it:

```
| /gsd-execute-phase 1
```

This is the fun part. The assistant will start creating files, writing code, and generally making things happen. You’ll see a running log of what it’s doing. At some points it will stop and ask for your permission before doing something potentially important—for example, before running a command that changes files in a significant way.

Claude Code will occasionally pause and ask you to approve an action. Always read the prompt before approving. The default behavior—where the assistant asks first—is the safe setting, and it is the correct setting for beginners. You should leave it alone. You’ll see suggestions online telling you to turn the prompts off for convenience. Don’t. Those prompts are your seat belt.

You may come across a command-line flag in other tutorials that completely disables permission prompts. Do not use it. It exists for experienced developers running automated pipelines in throwaway environments, and it bypasses every safety check the assistant has. For everything in this guide, the default interactive mode is what you want.

When the phase finishes, GSD will usually have created several new files in your project. Congratulations—something real exists now.

5.4 Step Three: `/gsd-verify-work`

Before moving on, ask GSD to check its own work:

```
| /gsd-verify-work 1
```

This runs a short self-review. GSD looks at what was built, compares it to what the plan said should happen, runs any tests that were written, and reports back. If everything passed, you’ll see a cheerful summary. If something is off, GSD will tell you exactly what and suggest a fix plan.

When `verify-work` reports success, the phase is done. Seriously—pause for a moment to notice that a real piece of software now exists on your computer that didn't exist an hour ago. You made that happen. It's a small thing, and also it isn't.

5.5 What to Do If Something Isn't Right

Sometimes the phase finishes and the result doesn't match what you had in your head. That's normal. Here's the order of things to try:

1. **Describe what's wrong.** Tell the assistant, in plain English, what you expected versus what you got. "The page loads, but the heading says 'Hello world' and I wanted it to say my name."
2. **Let it fix.** The assistant will usually propose a small fix and, with your permission, apply it.
3. **Re-run verify.** Type `/gsd-verify-work` again to confirm the fix worked.
4. **If you're stuck,** try `/gsd-debug` followed by a description: `/gsd-debug "Heading doesn't update when I change my name"`. GSD has a systematic debugging mode that walks through the problem step by step.

If you ever feel truly lost, Appendix A has more help.

You now know the full rhythm: plan, execute, verify, repeat for the next phase. The next chapter covers the little commands that help you find your way around a project in progress.

Chapter 6

Navigating Your Project

A GSD project grows as you work on it. After a few phases, you'll have a handful of plan files, some finished code, maybe a note or two. This chapter is about the short, reassuring commands that tell you where you are and what to do next.

6.1 `/gsd-progress` --- ``Where am I?''

Run this any time you want a bird's-eye view:

```
| /gsd-progress
```

You'll see a summary of the project, which phase is current, what's complete, and what's on deck. This is the command to run when you come back to the project after a break and have no idea what you were doing.

6.2 `/gsd-next` --- ``What should I do next?''

If `/gsd-progress` says "where am I," `/gsd-next` says "what next." Run it and GSD figures out the correct next step for your project and either does it or tells you the command to run:

```
| /gsd-next
```

If Phase 1 is planned but not executed, `/gsd-next` will execute it. If Phase 2 hasn't been planned, it will plan it. If every phase is done, it will suggest completing the milestone. It's the "just keep going" button.

6.3 `/gsd-help` --- ``What commands exist?''

If you forget which commands GSD offers:

```
| /gsd-help
```

You'll see the full list. You won't need most of them for a while, but it's nice to know they're there.

6.4 The *.planning/* Directory

I mentioned this folder earlier; it's worth a section of its own because you'll see it a lot. *.planning/* is where GSD keeps the project's brain: the spec, the roadmap, the plans for each phase, the status file, notes. It lives inside your project folder, right alongside the code GSD writes.

If you delete *.planning/*, GSD forgets everything about your project. Not the code—the code is safe—but everything about *why* the code is the way it is, what's planned next, and what's been decided. The folder is small (usually a few hundred kilobytes at most), so there's no reason to delete it.

If you're using version control (Git) with your project, the *.planning/* folder can be committed along with the code. It means the next person to open the project (including future-you on a different computer) gets the plan, not just the code.

6.5 Pausing and Resuming

If you have to stop mid-work (lunch, the end of the day, the cat pressed a key), GSD has two commands to make coming back painless:

```
/gsd-pause-work  
/gsd-resume-work
```

/gsd-pause-work writes a short handoff note summarizing where you are. */gsd-resume-work* reads it back and brings the assistant up to speed in a fresh session. Use them whenever you close your laptop.

That's navigation. The next chapter covers the “quick task” mode for when you don't need the full phase workflow.

Chapter 7

Quick Tasks

Not every task deserves a full roadmap. Sometimes you just want to add a footer to your website, or rename a button, or tweak a color. For those, GSD has a lighter command: `/gsd-quick`.

7.1 When to Use `/gsd-quick`

Use it for small, self-contained tasks that you could describe in one or two sentences. Things like:

- “Add a footer with my copyright to the bottom of every page.”
- “Change the background color to a warm beige.”
- “Fix this bug where the submit button does nothing.”
- “Add a favicon.”

For anything that feels bigger than a sentence or two—a new section, a new feature, a rethink of the layout—use the full planning workflow (Chapter 4 and 5) instead.

7.2 How to Use It

```
| /gsd-quick
```

GSD will ask you what you want done. Describe it in plain English. It will propose a short plan, ask your permission to proceed, and then do the work.

Even in quick mode, GSD still plans first and verifies afterward— just with less ceremony. Your work is still safe.

If you’d like GSD to be a little more thorough for a particular quick task, you can add flags:

- `--discuss` — have a short pre-planning discussion first.
- `--research` — let GSD look things up before planning.
- `--full` — run the full plan-check and verification steps.

For example: `/gsd-quick --full`.

The next chapter is a collection of small habits that make the whole experience smoother.

Chapter 8

Tips for Success

Everyone who uses GSD well does the same small number of things. They are not secrets and none of them are clever—they're just habits. Here they are.

8.1 Be Specific About What You Want

“Make me a website” gets you an average website. “Make me a one-page site for my grandmother’s 80th birthday, with her photo, a short tribute, a photo gallery of the grandkids, and a guestbook where people can leave messages” gets you *that* website. The more specific you are, the closer the result is to what you pictured.

This is the #1 habit. It outweighs every other piece of advice in this chapter combined.

8.2 Let GSD Ask Questions

When GSD asks a question—during `/gsd-new-project`, during planning, during execution—answer it. Don’t skip it, don’t say “whatever,” don’t type “idk.” Every question it asks is because it noticed a place where the result could go in two different directions and it wants you to pick. If you don’t pick, it picks for you, and you may not like its choice.

Say so: *“I don’t know, which do you recommend and why?”* GSD will give you a recommendation with reasons, and you can just agree if the reasoning sounds good.

8.3 Don't Edit `.planning/` Files by Hand

I said this earlier but it bears repeating. If you want to change the plan, talk to the assistant. Editing `.planning/` files directly is like rewriting the blueprints while the contractor is still reading them: confusing, and things will fall out of sync.

8.4 Capture Ideas With `/gsd-add-todo`

You'll get ideas while the assistant is working on something else. "Oh, I should add a search box." "It would be nice if the images had captions." Don't try to hold those in your head. Run:

```
| /gsd-add-todo "Add search box to the bio page"
```

GSD will save it to a list. Later, when you're ready, you can review the list:

```
| /gsd-check-todos
```

You'll see everything you captured and can pick one to work on.

8.5 Take Breaks

This is less about GSD and more about you. A fresh set of eyes catches things tired eyes miss. If something is going in circles, step away for twenty minutes. When you come back, run `/gsd-resume-work` and you'll be picked up exactly where you left off.

8.6 Celebrate Small Wins

The first time a phase completes successfully, stop and look at what you made. Open the thing in a browser, run the script, see the output. *Tell someone*. Building real things is a big deal, and the whole point of GSD is that you get to do it.

The final chapter covers what happens when you finish your first project and want to do more.

Chapter 9

What's Next?

You finished the last phase. `/gsd-verify-work` says everything checks out. What now?

9.1 Complete the Milestone

A “milestone” in GSD is a completed version of your project. When all phases of a milestone are done, run:

```
| /gsd-complete-milestone
```

GSD will archive the planning documents for this milestone, update a historical log of everything you’ve shipped, and (if you’re using version control) tag a release. It’s your graduation ceremony for that version of the project.

9.2 Start a New Milestone

When you’re ready to add more—new features, a bigger redesign, whatever—start a new milestone:

```
| /gsd-new-milestone
```

GSD will ask you what the new milestone is about, generate a new roadmap for it, and you’re back in the plan-execute-verify loop for this new version. The old milestone is still there, archived, so you can always look back.

Don’t be shy about calling something “done” and starting a new milestone. A project with five small completed milestones is healthier than a project with one never-finished milestone. Ship early, ship often, celebrate each one.

9.3 When You're Ready for More

This guide has covered the core workflow. GSD has many more commands for special situations: bringing GSD into an existing codebase, running phases in the background, reviewing code, generating tests, digging into bugs, and more. Run `/gsd-help` any time to see the full list.

Everything you learned here still applies. The extra commands are just variations on the same rhythm: describe, plan, do, check.

9.4 Getting Help

GSD has an active community on Discord. If you're stuck, have a question, or just want to share what you built, the people there are friendly and have probably run into the same thing you're running into. You can join from inside GSD:

```
| /gsd-join-discord
```

You started with an idea and you finished something real. That is the entire point. Everything else is just practice. The next project will be easier, the one after that easier still, and before long you'll be surprised at what you can make.

Appendix A

Troubleshooting

A short list of things that sometimes go wrong and what to do about them.

A.1 ``Command Not Found''

Symptom: you type `/gsd-help` (or any GSD command) and the assistant says it doesn't recognize it.

Fix: close Claude Code completely and open it again. GSD commands get picked up when the assistant starts, so a fresh start usually does it. If that doesn't work, re-run the installer (`npx get-shit-done-cc@latest`) and pick the same options as before.

A.2 ``Something Went Wrong With the Project''

Symptom: you get errors mentioning missing files, or `/gsd-progress` reports something confusing about the project state.

Fix: run the health check:

```
| /gsd-health
```

It will tell you what's wrong. If it offers to repair, let it:

```
| /gsd-health --repair
```

This fixes most common issues automatically.

A.3 ``I'm Completely Lost''

Symptom: you stepped away from a project for a week and you have no idea what you were doing.

Fix: two commands, in order:

```
| /gsd-resume-work  
| /gsd-progress
```

`/gsd-resume-work` brings the assistant up to speed; `/gsd-progress` reminds you. Between them you'll be re-oriented in under a minute.

A.4 ``The Phase Produced Something I Don't Want''

Symptom: `/gsd-execute-phase` finished, but the result is wrong in some way.

Fix: first, describe the problem to the assistant in your own words. Most issues resolve with a short conversation. If you'd rather undo the phase entirely and start over:

```
| /gsd-undo --phase 1
```

This rolls back the phase's changes safely. You can then re-plan and re-execute it.

A.5 ``Everything Is Fine, I'm Just Anxious''

Symptom: things are working but you're worried you're doing it wrong.

Fix: you're not. Run `/gsd-progress`, see that you're in a valid state, take a breath, and keep going. The workflow is forgiving. Mistakes are recoverable. You are allowed to be a beginner.

— *End of guide* —