

MINI PACK

# One Rules File for Every AI Tool

Keep AGENTS.md, CLAUDE.md, and Cursor rules in sync, short, and actually followed.

For Claude Code · Cursor · plus any tool that reads AGENTS.md

10 prompts · 4 sections

# Welcome! How to Use This Pack

You told Claude Code to use pnpm. You told Cursor to stay out of the migrations folder. Then you changed one file and forgot the other, and now your two tools follow two different sets of rules.

This pack fixes that. You'll find every instruction file in your project, merge them into one shared AGENTS.md, point your other tools at it, trim it down to the rules that matter, and keep it accurate as your app grows.

AI tools change how they read these files often. Every tool-specific tip in this pack comes from the official docs for Claude Code, Cursor, Lovable, Codex, and Windsurf as of October 2026. If something on your screen looks different, trust your tool's current docs.

## How each prompt is laid out

- **Title:** what the prompt is.
- **Use this when...:** one or two sentences on what the prompt is for and when to reach for it.
- **Fill in:** the blanks to replace before you paste.
- **The prompt:** the box you copy.
- **Tips to customize:** optional ways to change it.

Anything in **[BRACKETS IN CAPS]** is a blank to fill in.

**Works best with:** Claude Code or Cursor's Agent, because they can search your whole project, including hidden folders like `.cursor` and `.claude`. Prompt 4.1 works in any AI chat, like ChatGPT or Claude.

**The files, in one line each.** **AGENTS.md** is an open, plain-markdown rules file that many tools read, including Cursor, Codex, and Windsurf, and Lovable reads one at the root of your project.

**CLAUDE.md** is Claude Code's own rules file. **.cursor/rules** is a folder of Cursor rule files that end in `.mdc`.

**Using the free Claude Code and Cursor prompts on our site too?** Those help you write a first CLAUDE.md or set up Cursor rules. This pack is for when you have rules in more than one place and need them to agree.

## Quick Start: Find Your Problem

- **Not sure which rules files you even have?** Prompt 1.1.
- **Your tools act differently on the same task?** Prompt 1.2.
- **Same rules copied into several files?** Prompts 2.1 and 2.2.
- **Old files like .cursorrules lying around?** Prompt 2.3.
- **Your rules file is huge and the AI ignores parts of it?** Prompts 3.1 and 3.2.
- **Not sure a rule does anything?** Prompt 4.1.
- **The AI keeps making the same mistake?** Prompt 4.2.

**Golden rule:** Keep your rules in one file, and have every other rules file point to it instead of copying it.

## What's Inside

1. Find the Mess (2 prompts)
2. Make One Source of Truth (3 prompts)
3. Keep It Lean (2 prompts)
4. Keep It Working (3 prompts)

## SECTION 1

# Find the Mess

*You can't sync files you don't know about.*

### 1.1 Map Every Instruction File

Use this when you use more than one AI coding tool and aren't sure which instruction files you have or which tool reads which. It's for getting a plain map of every rules file in your project before you change anything.

**Fill in:** [MY AI TOOLS] (e.g. "Cursor and Claude Code"), [WHAT I'VE NOTICED] (e.g. "Cursor ignores my rules", "Claude keeps using npm instead of pnpm").

```
Don't change any files. I want a map of every AI instruction file in this project.
```

```
AI tools I use on this project: [MY AI TOOLS]
```

```
What I've noticed: [WHAT I'VE NOTICED]
```

```
Search the whole project, including hidden folders. Look for at least: AGENTS.md in any folder, CLAUDE.md, CLAUDE.local.md, .claude/CLAUDE.md, files in .claude/rules/, files in .cursor/rules/, .cursorrules, and any other file that gives instructions to an AI (for example a project brief the AI was told to read).
```

```
For each file, give me one row in a table with:
```

1. The path and its length in lines.
2. Which of my tools reads it, and when: every chat, only for certain files, or never. If you aren't sure, write "not sure" instead of guessing.
3. A one-line summary of what's in it.

```
Then tell me:
```

- Which rules appear in more than one file.
- Which files probably aren't being read at all.
- Which file is the best starting point for one shared rules file.



**Tips to customize:** Run this in Claude Code or in Cursor's Agent so it can search hidden folders. To see what actually loaded, run `/context` in Claude Code and look under Memory files. In Cursor, open Customize in the sidebar and go to Rules to see every rule and its status.

## 1.2 Find Rules That Fight Each Other

Use this when your AI tools behave differently on the same task, or follow a rule one day and ignore it the next. It's for finding rules that contradict each other, are out of date, or mention files and commands that no longer exist.

**Fill in:** [MY RULES FILES] (the paths from Prompt 1.1, e.g. "AGENTS.md, CLAUDE.md, .cursor/rules/style.mdc").

```
Don't change any files yet. Read these AI instruction files: [MY RULES FILES]
```

```
Then check them against the project itself (package.json, config files, and the folder structure) and report:
```

1. Contradictions: rules that say opposite things. Quote both lines and name their files.
2. Out-of-date rules: rules that mention commands, scripts, folders, files, or libraries this project doesn't have anymore.
3. Duplicates: the same rule worded differently in more than one file.
4. Vague rules that can't be checked, like "write clean code".

```
For each finding, suggest one fix: which line to keep, how to reword it, or which file it belongs in. List the findings most likely to explain why my tools act differently first.
```

 **Tips to customize:** Claude Code's docs warn that when two instructions contradict each other, Claude may pick one at random, so fix contradictions first. Newer versions of Claude Code also have `/doctor prompt-audit`, which checks your CLAUDE.md, CLAUDE.local.md, AGENTS.md, and the files in `.claude/` for conflicts and outdated content. It changes nothing until you ask, so it makes a good second opinion. If the command isn't there, update Claude Code or check its docs.

## SECTION 2

# Make One Source of Truth

---

*One file holds the rules. Everything else points to it.*

## 2.1 Merge Everything Into One AGENTS.md

Use this when the same rules are spread across several files and you're tired of updating each one. It's for combining them into one short AGENTS.md at the root of your project that your tools can share.

**Fill in:** [MY RULES FILES], [RULES THAT MUST STAY] (e.g. "always use pnpm", "never edit the migrations folder"), [TOOL-ONLY NOTES] (e.g. "Claude Code: use plan mode for database changes", or "none").

```
Combine my AI instruction files into one AGENTS.md at the project root. Show me the new file before you save it, and don't delete any old files yet.
```

```
Files to merge: [MY RULES FILES]
```

```
Rules that must stay: [RULES THAT MUST STAY]
```

```
Tool-only notes: [TOOL-ONLY NOTES]
```

```
Rules for the merge:
```

1. Use these sections: Project overview (2-3 sentences), Commands (install, run, build, and test, exactly as they work in this project), Code style, Project structure (key folders only), Rules for the AI, and Don't touch.
2. Write each rule as one short bullet that can be checked, like "Run npm test before saying a task is done".
3. Keep one copy of each rule. When two files disagree, list the conflict for me instead of picking a side.
4. Leave out anything you can figure out from the code, and anything only one tool needs. List those lines separately so I can decide where they go.
5. After the new file, list every old line you dropped and why.

 **Tips to customize:** Cursor reads an AGENTS.md in your project root, and so do Codex and Windsurf. Lovable reads a root-level AGENTS.md too. Claude Code needs one small extra step, which is Prompt 2.2. Keep your old files until you've tried the new one in each tool, then retire them with Prompt 2.3.

## 2.2 Make Claude Code Read the Same File

Use this when you have an AGENTS.md and also use Claude Code. It's for setting up a tiny CLAUDE.md that pulls in AGENTS.md, so Claude Code follows the same rules as your other tools.

**Fill in:** [MY CURRENT CLAUDE.MD] (paste it, or write "no CLAUDE.md yet"), [CLAUDE-ONLY NOTES] (e.g. "use plan mode for changes in src/billing/", or "none").

```
Set up CLAUDE.md so Claude Code reads my AGENTS.md. Show me the result before you save it.
```

```
My current CLAUDE.md: [MY CURRENT CLAUDE.MD]
```

```
Claude-only notes: [CLAUDE-ONLY NOTES]
```

```
Please:
```

1. Make the first line of CLAUDE.md exactly: @AGENTS.md
2. Below it, add a short "Claude Code" section with only my Claude-only notes. If there are none, the file is just that one line.
3. If my current CLAUDE.md has rules that aren't in AGENTS.md, list them and ask me whether each one belongs in AGENTS.md (for every tool) or stays here (Claude Code only).
4. Don't copy any AGENTS.md rules into CLAUDE.md. The import line already brings them in.



**Tips to customize:** Here's why this matters. Claude Code can read AGENTS.md on its own, but by default only when your project has no CLAUDE.md or CLAUDE.local.md. Once one of those exists, Claude Code reads it instead and skips AGENTS.md. The `@AGENTS.md` line imports the file, and Claude Code's docs say keeping that import never makes it read AGENTS.md twice. Start a new session afterward and run `/context` to confirm CLAUDE.md appears under Memory files. The docs also mention a symlink option, but they recommend the import line if anyone on the project uses Windows.

## 2.3 Retire Old Rules Files Safely

Use this when you've merged your rules but still have leftover files, like .cursorrules, an old project brief, or plain .md files in .cursor/rules. It's for retiring them without losing a rule or leaving a tool reading an outdated copy.

**Fill in:** [OLD FILES] (from Prompt 1.1), [MAIN RULES FILE] (usually AGENTS.md).

```
Help me retire my old AI instruction files safely. Don't delete or edit anything until I say OK.
```

```
Old files: [OLD FILES]
```

```
Main rules file: [MAIN RULES FILE]
```

For each old file:

1. Check it line by line. Tell me which rules are already in the main rules file, which were dropped on purpose, and which are missing.
2. Recommend one action: delete it, keep it (with a reason), or shrink it so it only points to the main rules file.
3. Tell me whether any of my tools would still read the old file after the change, so I don't end up with two versions of my rules.

Then list every change as one commit, suggest a commit message, and wait for my OK.

 **Tips to customize:** Do this on a new git branch so you can undo it in one step. Cursor ignores plain `.md` files inside `.cursor/rules` because they have no frontmatter, so anything in them isn't being used. Cursor's current rules docs cover project rules, user rules, team rules, and AGENTS.md, and they don't mention the older `.cursorrules` file, so move anything important out of it. A "pointer" only works as a real import in Claude Code (the `@AGENTS.md` line from Prompt 2.2). In a Cursor rule, mentioning a file tells the Agent where to look but doesn't load its contents, so for Cursor just rely on AGENTS.md.

## SECTION 3

# Keep It Lean

*Every line loads every time. Make each one count.*

### 3.1 Trim a Bloated Rules File

Use this when your rules file has grown long and the AI seems to ignore parts of it. It's for sorting every line into keep, move, or cut, so only rules that change the AI's behavior load in every chat.

**Fill in:** [RULES FILE] (e.g. AGENTS.md), [TARGET LENGTH] (e.g. "under 100 lines"), [MISTAKES THE AI STILL MAKES] (e.g. "adds new libraries without asking").

Don't change anything yet. Go through [RULES FILE] line by line and sort every rule into one group:

- KEEP: it changes what the AI does, and the AI couldn't work it out from the code.
- MOVE: it only matters for one folder or one kind of file, so it should load only there.
- CUT: it's obvious from the code, a general habit the AI already follows, out of date, too vague, or a duplicate.

Show me a table with each rule, its group, and a short reason. Then show me the trimmed file, aiming for [TARGET LENGTH].

Keep any rule that targets a mistake the AI still makes: [MISTAKES THE AI STILL MAKES]

If a rule is vague but important, rewrite it so it can be checked instead of cutting it.



**Tips to customize:** Claude Code's docs suggest keeping each CLAUDE.md under 200 lines, because longer files use more context and get followed less reliably. Cursor's docs suggest keeping each rule under 500 lines, and advise against pasting whole style guides (use a linter instead) or listing every common command. Splitting a file into imports doesn't make it lighter in Claude Code, since imported files load at the start too. Take your MOVE lines to Prompt 3.2.

## 3.2 Give a Folder Its Own Rules

Use this when some rules only matter in one part of your app, like the API folder or the database migrations. It's for moving those rules into a folder-level AGENTS.md, so both Cursor and Claude Code load them only when they work in that folder.

**Fill in:** [FOLDER] (e.g. src/api), [RULES FOR THAT FOLDER] (paste the MOVE lines from Prompt 3.1).

```
Move some rules out of my main rules file and into a rules file for one folder.
```

```
Folder: [FOLDER]
```

```
Rules for that folder: [RULES FOR THAT FOLDER]
```

```
Please:
```

1. Create an AGENTS.md inside [FOLDER] with just these rules as short bullets, under a one-line heading that says which part of the app it covers.
2. If I use Claude Code, also create a CLAUDE.md inside [FOLDER] that contains only the line @AGENTS.md, so Claude Code loads the same rules there.
3. Remove the moved rules from the main rules file.
4. Show me all the changed files before saving, and tell me how to check that each of my tools picks up the folder rules.

 **Tips to customize:** Cursor applies a folder's AGENTS.md automatically when it works with files in that folder, combined with the rules above it, and the more specific folder rules win. Claude Code loads a folder's CLAUDE.md once it reads or edits a file in that folder, and the @AGENTS.md line pulls in the same text. You can also use each tool's own format instead: Cursor rule files with a `globs` pattern, or Claude Code files in `.claude/rules/` with a `paths` pattern. That works well, but it gives you two copies to keep in sync again.

## SECTION 4

# Keep It Working

*Rules drift as your app grows. A few minutes now saves a mess later.*

### 4.1 Does This Rule Earn Its Spot?

Use this when you're not sure a rule does anything, or you want to slim down your file without guessing. It's for comparing the same small task done with and without one rule, then keeping the rule only if it changes the result.

**Fill in:** [THE RULE], [TEST TASK] (e.g. "add a delete button to the notes list"), [WHAT THE RULE SHOULD CHANGE] (e.g. "uses our Button component instead of a plain button"), [RESULT WITHOUT THE RULE], [RESULT WITH THE RULE] (paste the code changes from each run).

I tested one rule from my AI rules file by running the same task twice, each time in a fresh chat: once with the rule removed and once with it in place. Compare the two results and tell me whether the rule earns its spot.

The rule: [THE RULE]

Test task: [TEST TASK]

What the rule should change: [WHAT THE RULE SHOULD CHANGE]

Result without the rule: [RESULT WITHOUT THE RULE]

Result with the rule: [RESULT WITH THE RULE]

Please:

1. Tell me whether the results differ in the way the rule intends, and quote the lines that show it.
2. Give a verdict: keep it, reword it (with the new wording), or cut it.
3. If the task didn't really test the rule, suggest a better test task.

 **Tips to customize:** To run the test, commit your work first. Remove the rule, start a new chat (rules are read when a chat starts), run the task, and copy the changes it made (`git diff` shows them). Undo those changes, put the rule back, start another new chat, and run the same task again. Paste both results into a separate AI chat with this prompt. AI results vary from run to run, so one test is a hint, not proof. For an important rule, test it twice.

## 4.2 Turn a Repeated Mistake Into One Good Rule

Use this when the AI makes the same mistake for the second or third time. It's for writing one short, specific rule that prevents it and putting it in the right file, instead of piling on another vague line.

**Fill in:** [THE MISTAKE], [WHAT I WANTED INSTEAD], [EXAMPLE FILE] (a file that shows the right way, e.g. `src/components/Button.tsx`, or "none").

You keep making the same mistake. Help me write one rule that stops it. Don't edit any files until I approve the wording.

The mistake: [THE MISTAKE]

What I wanted instead: [WHAT I WANTED INSTEAD]

A file that shows the right way: [EXAMPLE FILE]

Please:

1. Check my rules files for a rule that already covers this. If one exists, tell me why it didn't work and rewrite it instead of adding a new one.
2. Otherwise, write one new rule: a single bullet, specific enough to check. If there's an example file, name its path in backticks without an @ in front.
3. Tell me where the rule belongs: the main AGENTS.md, a folder's rules file, or a file for one tool only, and why.
4. Show me the exact line and where it goes.



**Tips to customize:** Cursor's docs suggest starting simple and adding a rule only when you see the Agent make the same mistake more than once. The backticks matter in Claude Code: an @ in front of a path imports the whole file into every session, while a path in backticks is only a mention. Pointing to an example file keeps your rules short, and the AI reads the file when it needs it.

## 4.3 Keep My Personal Preferences Out of the Project Rules

Use this when your rules file mixes project rules with your own habits, like how short you want answers or how much you want things explained. It's for moving those personal preferences into each tool's personal settings, so the project file stays about the project.

**Fill in:** [RULES FILE], [MY AI TOOLS].

Don't change anything yet. Read [RULES FILE] and find the lines that are about me and how I like to work, not about this project. For example: reply style, how much to explain, or how I like to review changes.

AI tools I use: [MY AI TOOLS]

Please:

1. List each personal line and say why it's personal rather than about the project.

2. For each one, tell me where it should live instead in the tools I use. If you aren't sure where a tool keeps personal rules, say so and tell me to check its docs.
3. Show me the project rules file with those lines removed.

💡 **Tips to customize:** In Cursor, personal preferences go in User Rules (Customize → Rules), which apply across all your projects in Agent chats but not in Inline Edit or Tab. In Claude Code, `~/.claude/CLAUDE.md` holds personal preferences for every project on your computer. Claude Code also has `CLAUDE.local.md` for personal notes on one project. Add it to `.gitignore`, and keep the CLAUDE.md from Prompt 2.2, because a CLAUDE.local.md on its own stops Claude Code from reading AGENTS.md.

## Bonus: Magic Phrases for Rules Files

- "Don't change any files yet."
- "Quote the line and the file it came from."
- "If two rules disagree, ask me instead of picking one."
- "Keep one copy of each rule."
- "Write it so I can check it."
- "Which of my rules did you follow for this task?"

# One Rules File for Every AI Tool

That's the whole pack: **10 prompts across 4 sections**. Map your files, merge them into one AGENTS.md, point Claude Code at it, trim what doesn't earn its spot, and update your rules whenever the AI repeats a mistake.

**Happy building!**