[GUIDE]

Set up UX Central Governance with Claude Code

How to set up one central set of rules, a design process and a task board that every project inherits, so Claude Code does UX work the way you decide.

Sebastian Bourges

Version 1.4.0, October 2026
sebastianbourges.com/guides/ux-central-governance

Licence and attributions

Copyright 2026 Sebastian Bourges.

Licence

This guide and its source files are published under the Creative Commons Attribution 4.0 International licence (CC BY 4.0). You may copy, share and adapt them for any purpose, including commercial work. Credit the author, link to the licence, and say whether you made changes.

The licence: https://creativecommons.org/licenses/by/4.0/

A credit line you can copy:

> Set up UX Central Governance with Claude Code, by Sebastian Bourges, https://sebastianbourges.com/guides/ux-central-governance/, licensed under CC BY 4.0.

Trademarks

Claude, Claude Code and Anthropic are trademarks of Anthropic, PBC.
Visual Studio Code is a trademark of Microsoft Corporation.
Figma is a trademark of Figma, Inc.
Node.js is a registered trademark of the OpenJS Foundation.
Git and the Git logo are either registered trademarks or trademarks of Software Freedom Conservancy, Inc., corporate home of the Git Project, in the United States and/or other countries.
GitHub is a trademark of GitHub, Inc., registered in the United States and other countries.

Other names belong to their owners. This guide is independent: none of these companies wrote, reviewed or endorses it.

Typefaces

Playfair Display by Claus Eggers Sørensen, Inter by Rasmus Andersson and JetBrains Mono by JetBrains, each under the SIL Open Font License 1.1.

Where to find it

The guide on the web: https://sebastianbourges.com/guides/ux-central-governance/

The source files: https://github.com/sb-simplifyUX/UXCentral-Governance

Contents

Part I **Start here** 7

1 What it is 8

2 Set it up with one sentence 9

Part II **How it works** 13

3 How it works 14

4 The design process at its heart 18

Part III **Keeping AI on task** 21

5 The board that keeps AI on task 22

6 Who decides what 26

7 What it enforces, and how strongly 27

Part IV **Build it and use it** 29

8 Install it by hand, phase by phase 30

9 Run your first project 32

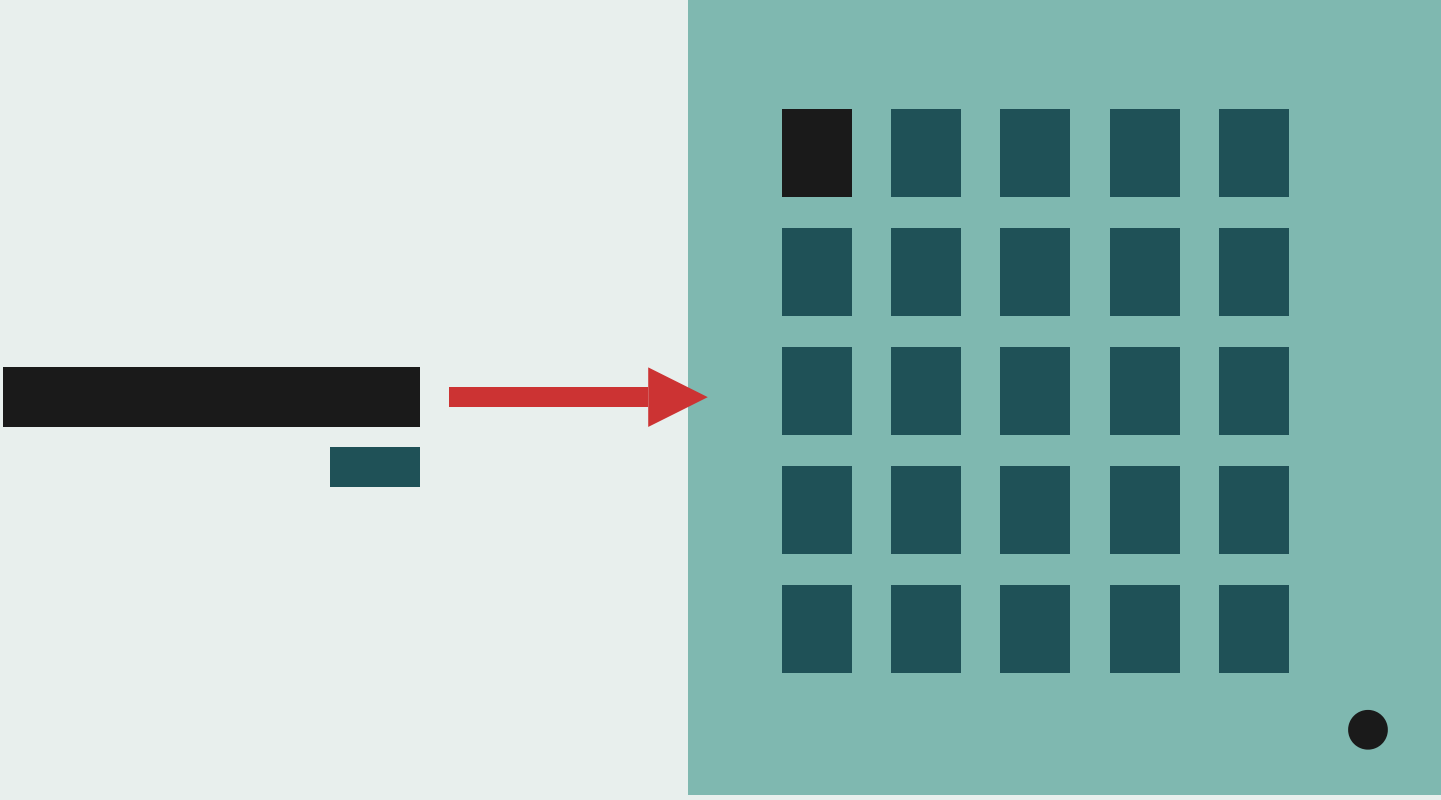
Part V **Where it comes from** 37

10 Software built with the method 38

11 Official references 39

12 Credit 40

© 2026 Sebastian Bourges | SimplifyUX



Part I

Start here

What is the system, and how do you set it up with one sentence?

1 **What it is** 8

One set of rules, one design process and one task board, kept in one folder that every project inherits.

2 **Set it up with one sentence** 9

The two things you install, the sentence you paste, and the 25 files Claude works from.

© 2026 Sebastian Bourges | SimplifyUX

I

1 What it is

One set of rules, one design process and one task board, kept in one folder that every project inherits.

> Open a folder in Visual Studio Code, open the Claude Code panel and paste:
>
> Set up the UX central governance system on this machine. The guide is at
>
> `https://sebastianbourges.com/guides/ux-central-governance/`

UX Central Governance is one set of rules, one design process and one task board. They live in a single central folder, and every design project on your machine inherits them. Claude Code reads them at the start of every session, so it works the same way in every project. It finds the problem before the solution. It leaves the decisions to the decider, the one person who makes the final call on a project. And it cannot call a task done without proof.

This page explains the system and walks you through setting it up. The files that do the work are the 24 chapters in the guide's contents. They are written for Claude to follow, one phase at a time, and they stop for your approval wherever a person has to decide.

© 2026 Sebastian Bourges | SimplifyUX

2
Set it up with one sentence

The two things you install, the sentence you paste, and the 25 files Claude works from.

You need two things, and you install both yourself:

- **Visual Studio Code with the Claude Code extension.**
- **A Claude account on a plan that includes Claude Code.**

Claude installs everything else, and it asks before each install. That means the Claude Code command-line tool, Node.js (the guards and the installer are Node.js scripts) and Git (the central folder is a Git repository). It also installs pnpm, but only if you build your own component libraries later. If an install needs your computer password, Claude does not try to run it. It shows you the exact command to run in the editor's terminal, then checks the result.

To start, open an empty folder in Visual Studio Code, open the Claude Code panel and paste the sentence at the top of this page. Claude reads the guide from this site. For more control, copy the guide's source files into a new, empty folder, open that folder instead and paste the full setup prompt. It carries your answers in advance, so Claude asks fewer questions:

© 2026 Sebastian Bourges | SimplifyUX

I

```
Set up the UX central governance system on this machine.
The blueprint's files are in this folder: read README.md, then follow 16-
Implementation-Guide.md
phase by phase.

Resume at: Phase 0
Organisation: <name or neutral label>
Administrator (full name / how to address them): <...> / <...>
Brands (display names and slugs): <...>
Industry and customer types: <...>
Locale for customer copy: <for example en-AU>
Work root folder, inside your home folder: <for example ~/ux-work>
Design tool and design system: <tool; library name>
Git host for central: <your private git host, or "local only for now">
Operating system: <macOS | Linux>

Rules: run the Phase 0 questionnaire for anything missing; present the full plan
for my approval before creating
anything; ask before installing, downloading, deleting, pushing or editing
~/.claude files; write files only with
your file tools and read them back; run each phase's Check before moving on;
delegate mechanical implementation
to Sonnet.
```

Claude then works through 12 phases, from discovery to onboarding other people. Each phase ends with a check, and Claude stops if the check fails. It shows you the full plan and waits for your approval before it creates anything. After that it still asks before each install or download, before it deletes or pushes anything, and before it changes your user settings. Some steps need Visual Studio Code restarted, which ends the session. Before that happens, Claude tells you which phase to resume at, and you paste the prompt again with that phase in its `Resume at` line.

If Claude cannot read this site from your network, copy the source files into the folder yourself and paste the full setup prompt.

These are the source files, all plain Markdown in one folder:

© 2026 Sebastian Bourges | SimplifyUX

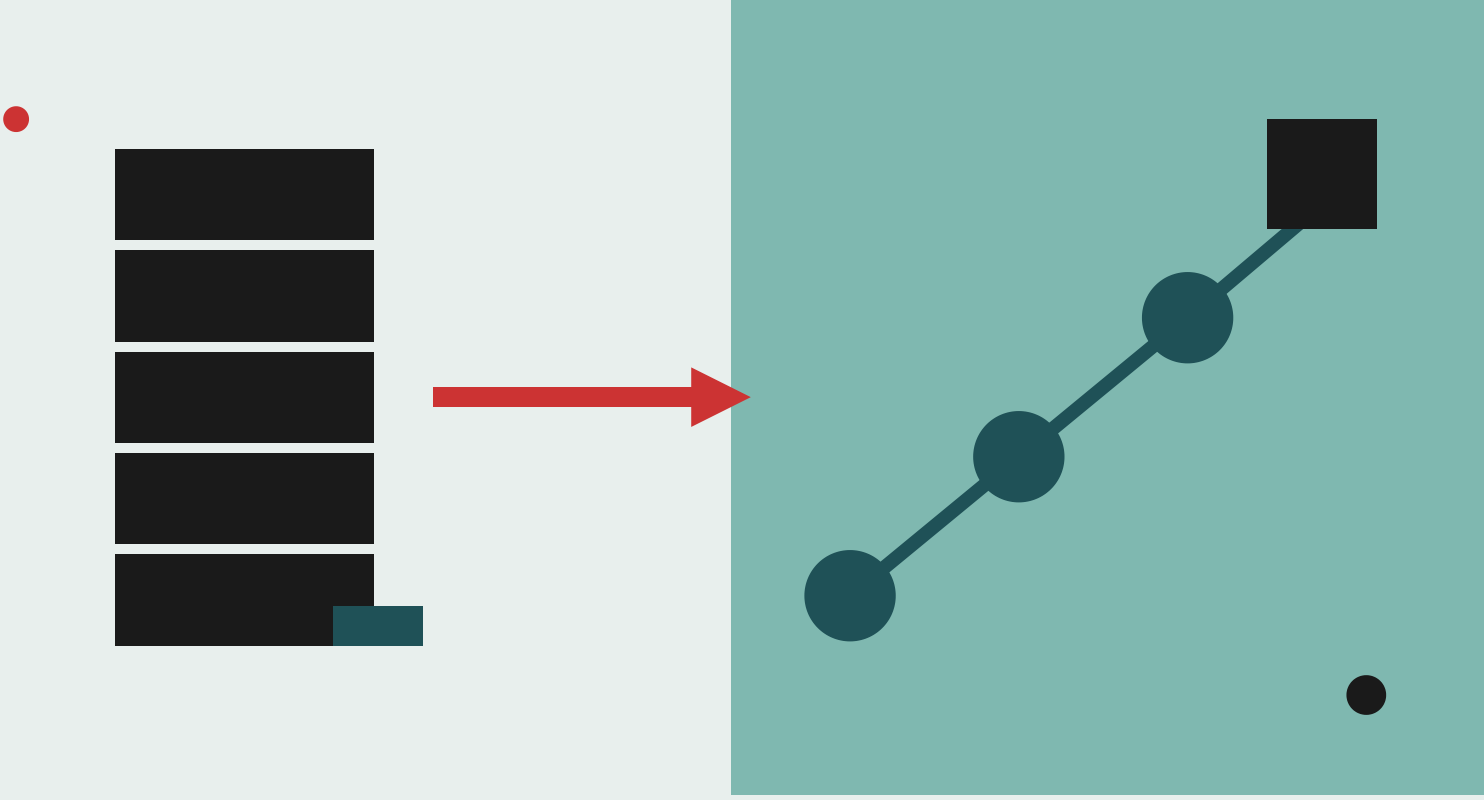
ux-central-governance/

`README.md` Start here: what you need, the credit, the reading order and the placeholders you replace.	`12-Skills-Catalogue.md` The 11 skills: what each does, when it runs and where it stops.
`01-Overview-and-Architecture.md` What the system is, its five layers, the folder layout and how projects inherit.	`13-Agents-and-Reviewers.md` The six read-only reviewers and the one helper.
`02-Build-Order-and-Lessons.md` The build order version by version, and the lessons behind the rules.	`14-Hooks-Permissions-Enforcement.md` The guards, the permission lists and what enforces each rule.
`03-Roles-and-Authorisation.md` The user, the decider and the administrator, and how approvals are recorded.	`15-Versioning-and-Change.md` Version numbers, the change log and bringing old projects up to date.
`04-UX-Design-Process.md` The six steps, what ends each one, and where AI is kept out.	`16-Implementation-Guide.md` The build plan Claude follows, phase by phase, with a check after each.
`05-Kanban-Method.md` Releases, outcomes, the nine-section card, the FALSIFIER and the five doors.	`17-Adapting-To-A-New-Business.md` The discovery questions and every value you replace.
`06-PRD-and-Project-Template.md` The project brief every project keeps, and the template folder.	`18-Porting-To-Other-AI-Tools.md` Running the same system with tools that lack plugins or hooks.
`07-Business-Funnel.md` How to map your customers' journey for your own industry.	`LICENCE.txt` The licence: Creative Commons Attribution 4.0.
`08-Brand-Tone-and-Regional-Communication.md` Several brands, tone files and regional language rules.	`appendices/` The copy-ready material the chapters point to.
`09-Design-Tool-Connection.md` Connecting Claude to your design tool, and the rules for its library.	`A-File-Templates.md` Every file the system needs, ready to copy.
`10-Privacy-Accessibility-Stack-Markdown.md` Research data, WCAG 2.2 AA, the software stack and the file standard.	`B-Skill-and-Agent-Sources.md` The full text of every skill and agent.
`11-Anthropic-Claude-Code-Standards.md` The Claude Code features the system uses, with the official pages.	`C-Hook-and-Tool-Sources.md` The guard scripts, the installer and the checkers.
	`D-Prompt-Presets.md` Five prompts to paste: three for projects, one to set up, one for another machine.
	`E-Glossary.md` Every term the system uses.

Figure 1. The 25 source files, in one folder. Every name, brand and example in them is invented, except the author's.

If you work in another language, the files stay in English and Claude can still answer you in yours. Set it for yourself, never in the shared files. Choose the response language in Claude Code's `/config`, or add `"language": "spanish"` (or your own language) to your user settings file, `~/.claude/settings.json`. Section names such as `FALSIFIER` and `Done when` stay in English, as the files write them, because the reviewers look for those names.

© 2026 Sebastian Bourges | SimplifyUX



Part II

How it works

What are its parts, and how does one session use them?

3	**How it works** The failures it prevents, its five layers, one session from opening a project to an approval, and what lives where.	14
4	**The design process at its heart** Six steps in two rounds: find the right problem, then the right solution, then test it and ship it.	18

© 2026 Sebastian Bourges | SimplifyUX

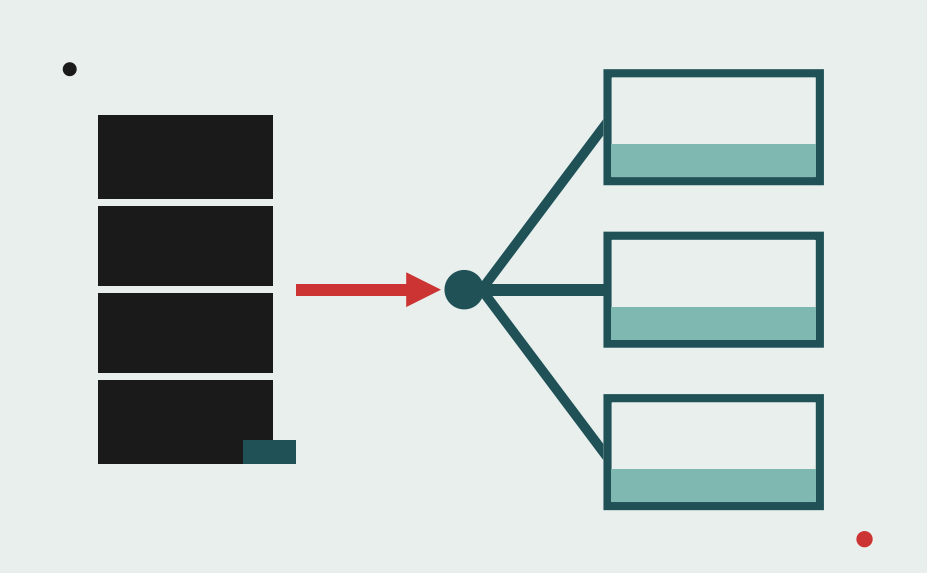
3 How it works

II

The failures it prevents, its five layers, one session from opening a project to an approval, and what lives where.

Teams that design with an AI assistant across many projects meet the same failures. The rules get pasted into each project and drift apart. The assistant jumps to solutions before anyone understands the problem. It makes decisions and writes customer types that a person should own. Tasks are called done because they feel finished. Design components get invented when the real one cannot be found. Customer details from research leak into documents and web requests. And nobody can say which version of the rules a project follows.

The system answers each failure in one of five layers. The first three tell Claude what to do; the fourth stops it when an instruction is not enough; the fifth holds only what is true of one project.



© 2026 Sebastian Bourges | SimplifyUX

Core rules	A short file of rules that never change from project to project, under 150 lines. **In Claude Code:** your user CLAUDE.md imports central's rules file, so every session loads it.
Specs and standards	The design process, the board, the file standard, privacy, accessibility, the software stack, brand and tone. **In Claude Code:** Markdown files in central that the rules, skills and reviewers read when they need them.
Procedures	Skills, which are step-by-step playbooks, and agents, which review the work or do mechanical jobs. **In Claude Code:** one plugin in a local marketplace inside central, installed for your user.
Enforcement	Rules that ask or refuse whatever Claude decides, and scripts that refuse bad input. **In Claude Code:** permission lists in your user settings, four hooks in the plugin, and Node.js tools.
Project facts	Each project's README, its brief with the task board, its own short CLAUDE.md and settings. **In Claude Code:** created from central's project template when the project starts.

Figure 2. The five layers. The first four live once, in central; only the fifth lives in each project.

© 2026 Sebastian Bourges | SimplifyUX

Here is what happens in one session, from the moment you open a project to the moment someone approves the work.

01 **The session opens in a project folder**

Your user CLAUDE.md loads and imports central's core rules; then the project's own CLAUDE.md loads, which holds project facts only.

02 **A start-up hook adds one status line**

It compares the project's rules version with central's. It names the current design step, the cards in progress and the project's decider, and flags an overdue design-library check.

03 **You ask for work, and Claude checks the step**

Does the work belong to a later step of the design process? Then Claude names the steps it would skip and asks the decider first.

04 **Claude finds the card the work serves**

If no card covers it, Claude drafts one and waits for the decider to approve it.

05 **A skill runs the procedure**

For example building a screen, sketching in the design tool, or drafting the project brief from research.

06 **The guards act while it works**

Hooks block customer personal information from leaving the machine. Permissions ask before anything is deleted, pushed or installed, before central changes, and before anything is written to the design tool.

07 **Reviewers check the result**

Read-only agents answer APPROVE or REQUEST FIXES, each with evidence from the files.

08 **The decider approves**

The approval is recorded beside the work with a name and a date.

Figure 3. One session, from opening a project to an approval.

© 2026 Sebastian Bourges | SimplifyUX

A project never copies the rules. It reaches them, so a change made once in central reaches every project at its next session. The only thing copied is the project template, once, when the project starts.

Central only

The rules, specs, design process, customer journey and tone files

The skills, agents, hooks, tools and component libraries

The project template, the prompts, the change log and the enforcement list

How it reaches a project

- → **Rules**: imported at the start of every session
- → **Skills, agents and hooks**: loaded from the plugin installed from central
- → **Permissions**: written once into your user settings by the installer
- → **Template**: copied once, when the project is created
- → **Version**: recorded in the project brief, so the start-up hook can warn when a project falls behind

Each project only

Objectives, scope, hard stops and success measures

The named decider, the current step, the problem statement and the decision record

The task board and its cards

Research, kept on the machine, and what came of it: sketches, specs and prototypes

Figure 4. What lives where. A rule repeated in a project file counts as a defect, and a reviewer flags it.

© 2026 Sebastian Bourges | SimplifyUX

4
The design process at its heart

Six steps in two rounds: find the right problem, then the right solution, then test it and ship it.

The process is the six steps I use on every project, described on **How I solve a problem**. It is two rounds of the same move: open up wide, then narrow down. The first round finds the right problem (Understand, then Define). The second finds the right solution (Sketch, then Decide). Validate and Deliver test it and ship it.

The most important file in any project is one sentence, written in Define: '[customer] needs a way to [need], because [what we learned].' Claude may offer three to five candidates, each citing evidence. Only the decider agrees the one the project solves. Define can never be skipped: Sketch, Decide, Validate and Deliver all refuse to start until the sentence is agreed.

© 2026 Sebastian Bourges | SimplifyUX

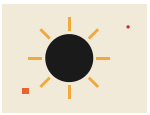
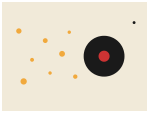
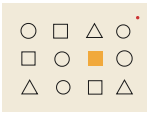
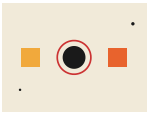
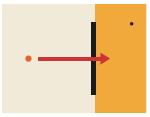
	[STEP 01 / UNDERSTAND] Stop solving. Start looking.	**Where AI helps** Reading huge piles of research faster than I could alone, and spotting what repeats across them.	**Where I keep AI out** The conversation with your customer. A model can't read a room or know when to push and when to wait. I won't outsource that.
	[STEP 02 / DEFINE] Boil it down to one problem, in one sentence.	**Where AI helps** Grouping notes across mountains of interviews, and drafting candidate sentences for me to argue with.	**Where I keep AI out** Writing the customer types. A model averages everyone into bland, generic descriptions.
	[STEP 03 / SKETCH] Lots of ideas first. Judge them later.	**Where AI helps** Throwing in raw ideas, especially when the team keeps circling the same one. One more voice, not the conclusion.	**Where I keep AI out** Deciding which idea is good. A model has no taste, no stakes, and no history with your customers.
	[STEP 04 / DECIDE] One person decides. Everyone informs the decision.	**Where AI helps** Stress-testing the chosen idea against cases I might have missed: "how does this fail for a customer in situation X?"	**Where I keep AI out** The decision itself. A model has nothing at stake and answers to no one. The person who signs off has both.
	[STEP 05 / VALIDATE] Test it with real people before it gets expensive.	**Where AI helps** Drafting the test plan, and pulling the recurring patterns out of a batch of sessions.	**Where I keep AI out** Running the session. A real person reading another real person isn't work a model can do.
	[STEP 06 / DELIVER AND AFTER] Launch isn't the finish line. It's the next set of data.	**Where AI helps** Drafting the documentation for the reusable kit, and spotting patterns in the post-launch numbers.	**Where I keep AI out** The conversation with the engineers. It's about what's actually possible and what the business needs first. A model can prep for it; it can't have it.

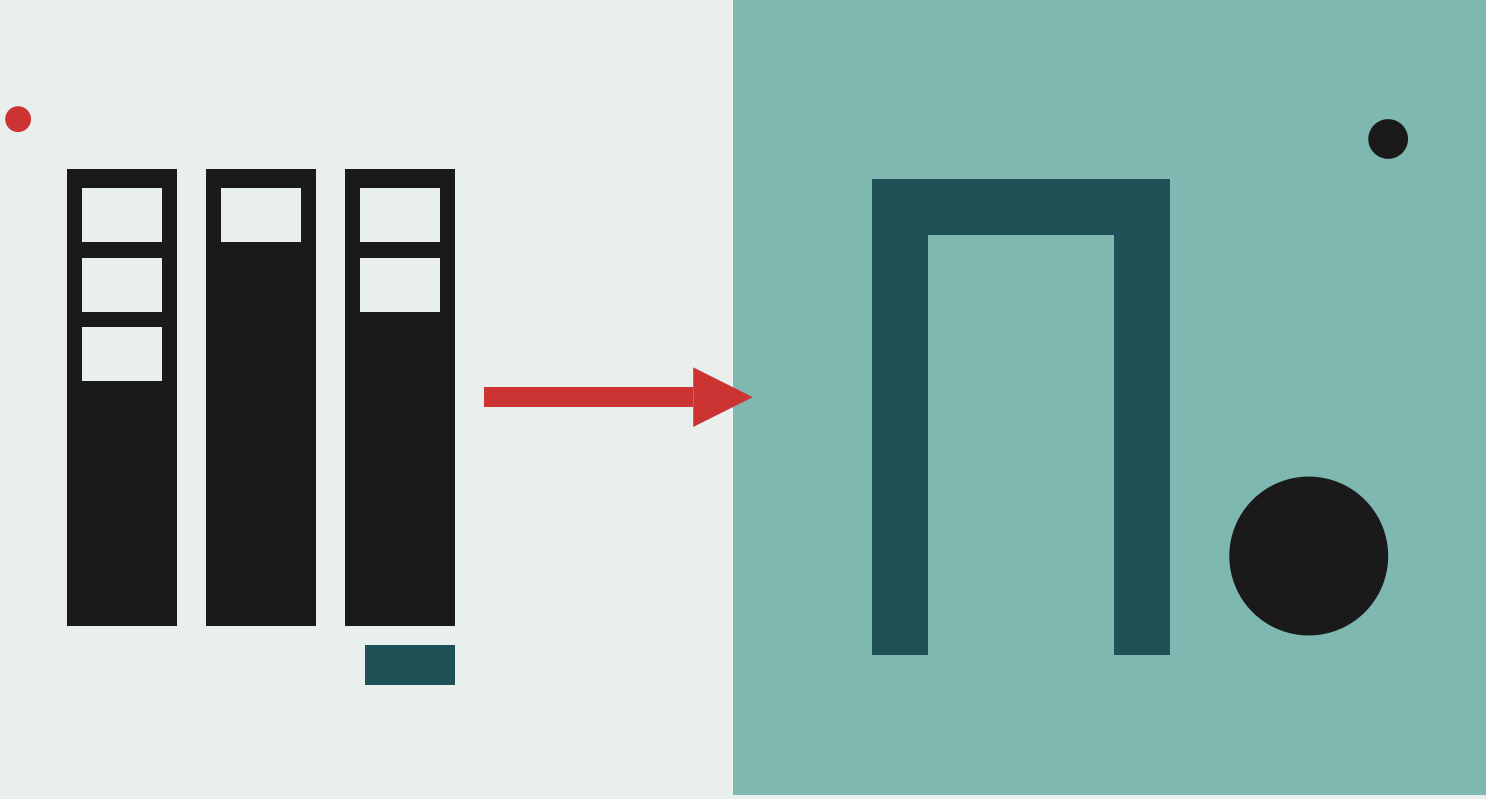
Figure 5. The six steps, with where AI helps and where I keep it out. The same steps, illustrations and wording as the process page.

The rule behind every step is that AI is an intern, not an oracle. It does the volume work. Judgement, accountability and one person reading another stay with people.

Claude may	Claude must not
Summarise and sort research	Replace conversations with customers
Draft problem statements for the decider to challenge	Write customer types
Add raw ideas	Judge which idea is good
Stress-test the chosen idea	Make the decision
Draft test plans	Run test sessions
Spot patterns in sessions or launch data	Replace conversations with engineers
Draft documentation for the design kit	Invent evidence, or treat its own output as truth

When a step needs something on the right, Claude prepares the materials (an interview guide, a shortlist, a test plan), names who has to act, and stops.

© 2026 Sebastian Bourges | SimplifyUX



Part III

Keeping AI on task

What stops Claude from drifting, deciding for you or skipping a rule?

5	**The board that keeps AI on task** Releases, outcomes and cards, and the five doors the board checks before work moves.	22
6	**Who decides what** The user, the decider and the administrator, and what each one approves.	26
7	**What it enforces, and how strongly** Every rule sorted as hard, checked or guided, counted, and what none of them stops.	27

© 2026 Sebastian Bourges | SimplifyUX

5
The board that keeps AI on task

Releases, outcomes and cards, and the five doors the board checks before work moves.

Every task maps to a card on the project's board, which lives in the project brief. Before any work, Claude finds the card the work serves. If there is none, it drafts one and waits. Work is cut into releases. A release has two to five outcomes, fixed when it opens. An outcome is a change for customers or the business, not a task. Every card serves exactly one outcome of the open release.

© 2026 Sebastian Bourges | SimplifyUX

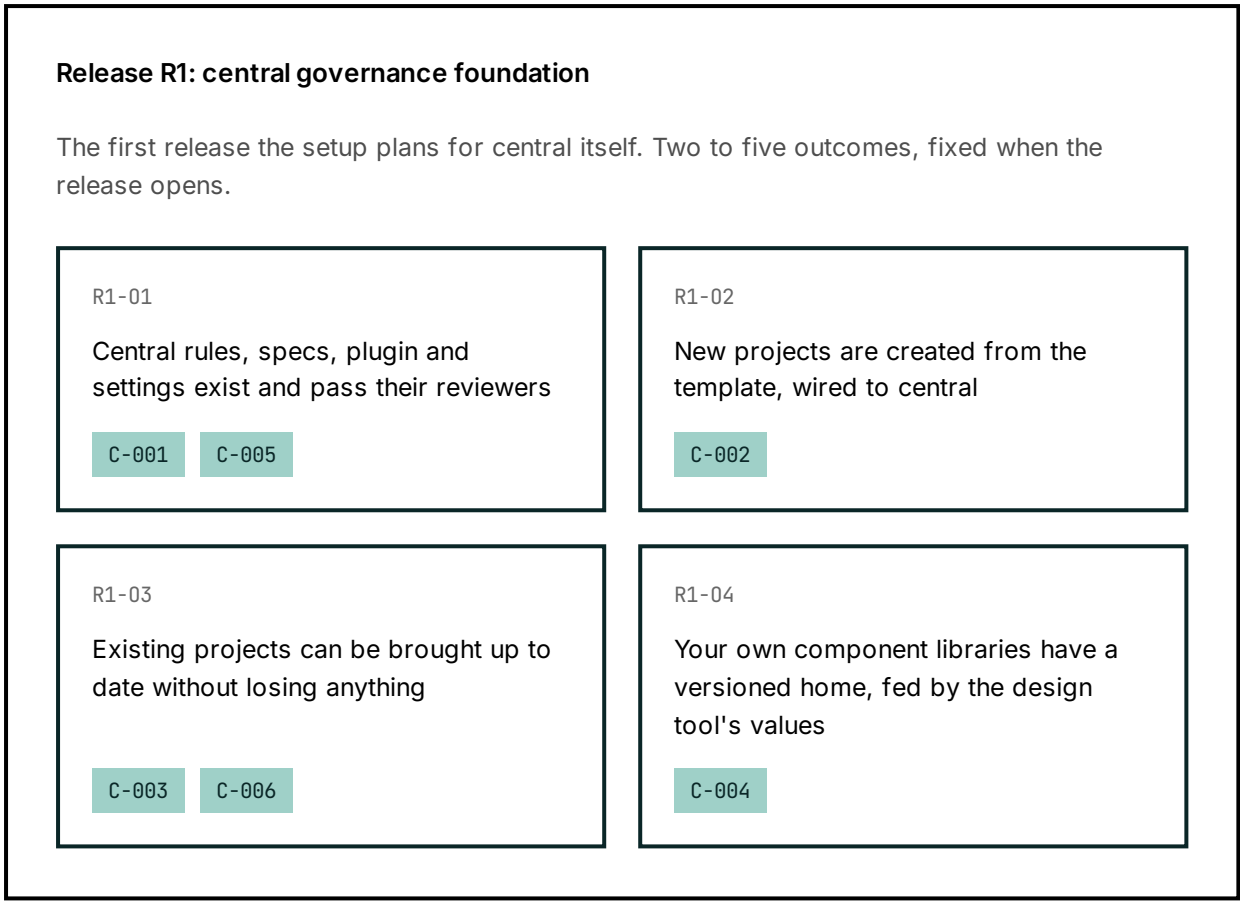


Figure 6. One open release holds its outcomes, and each outcome holds the cards that serve it.

A card has nine sections, always in the same order. The seventh, the FALSIFIER, names in one line what failure would look like. If you cannot write that line, you do not yet understand the task well enough to start it. Work that needs two FALSIFIER lines is two cards.

© 2026 Sebastian Bourges | SimplifyUX

C-002	**Create a sample project from the template and review it**	
1	**In one line**	Create a sample project from the template and have it reviewed.
2	**Outcome served**	R1-O2
3	**Process step**	n/a (central's own cards have no design step)
4	**Traces to**	The project brief, section Project template
5	**Brand**	All
6	**Done when**	A person starts the new-project skill, the sample passes the file check, and three reviewers approve it.
7	**FALSIFIER**	The generated project still holds an unfilled placeholder after the brief is drafted.
8	**Evidence**	not yet (one line, filled when the card finishes)
9	**Hard stops**	Claude never starts the new-project skill itself. No real personal information goes into the sample.

Figure 7. One of central's first six cards. Section 7 is the size test: one card, one FALSIFIER.

The board has four columns: To do, Doing, Blocked and Done. Each column has a shape as well as a colour, so a status never depends on colour alone. Doing holds at most three cards. A card moves only through one of five doors, and each door checks something before it opens:

© 2026 Sebastian Bourges | SimplifyUX

01	02	03	04	05
Create	**Start**	**Finish**	**Close outcome**	**Close release**
The card names an outcome of the open release. Claude proposes; the decider approves.	Sections 1 to 7 and 9 are filled, the FALSIFIER is one line, and Doing has a free slot.	The card is in Doing and section 8 holds one line of evidence.	Every card serving the outcome is done, with one line of outcome evidence.	Every outcome is closed, the safety pass is complete, and the decider writes the safety statement.

Figure 8. The five doors. Only the decider can open the last one.

When a door refuses, Claude quotes the line that failed and what would pass. When Doing is full, it names the three cards holding the slots. Before a release closes, a safety pass runs. It checks that no personal information is in the brief or the work, and that no reviewer still asks for fixes. It also checks accessibility, and where a tool cannot verify something, a person the decider names checks it by hand.

© 2026 Sebastian Bourges | SimplifyUX

6
Who decides what

The user, the decider and the administrator, and what each one approves.

The system has three roles. One person can hold all three, but the system never assumes it.

Role	Who	Approves
User	Whoever is running the Claude session	Nothing by role alone: they pass on approvals from the decider or the administrator
Decider	One person per project, named in the project brief	Everything in their project: file changes, deletions, commits, cards, the problem statement, the shortlist and the winner, and closing releases
Administrator	One person for the whole organisation, named in central's brief	Every change to central

When the person at the keyboard is not the one who must approve, Claude prepares the work, stops, and asks them to get that approval. It records it as `approved by <name>, <date>`. Claude cannot check who is at the keyboard, so it never claims to have. Some gates only a person can open: setting up a project, closing a release, deleting, pushing, installing and writing to the design tool.

© 2026 Sebastian Bourges | SimplifyUX

7 What it enforces, and how strongly

Every rule sorted as hard, checked or guided, counted, and what none of them stops.

Every rule in the system has a strength. **Hard** means a permission rule, a hook or a script asks or refuses, whatever Claude decides. **Checked** means a script or a reviewer verifies it after the fact. **Guided** means Claude follows it as an instruction, and nothing mechanical stops it. Central keeps a list of every rule with what enforces it, and a rule with no entry is a defect. Counted from the matrix in chapter 14:

© 2026 Sebastian Bourges | SimplifyUX

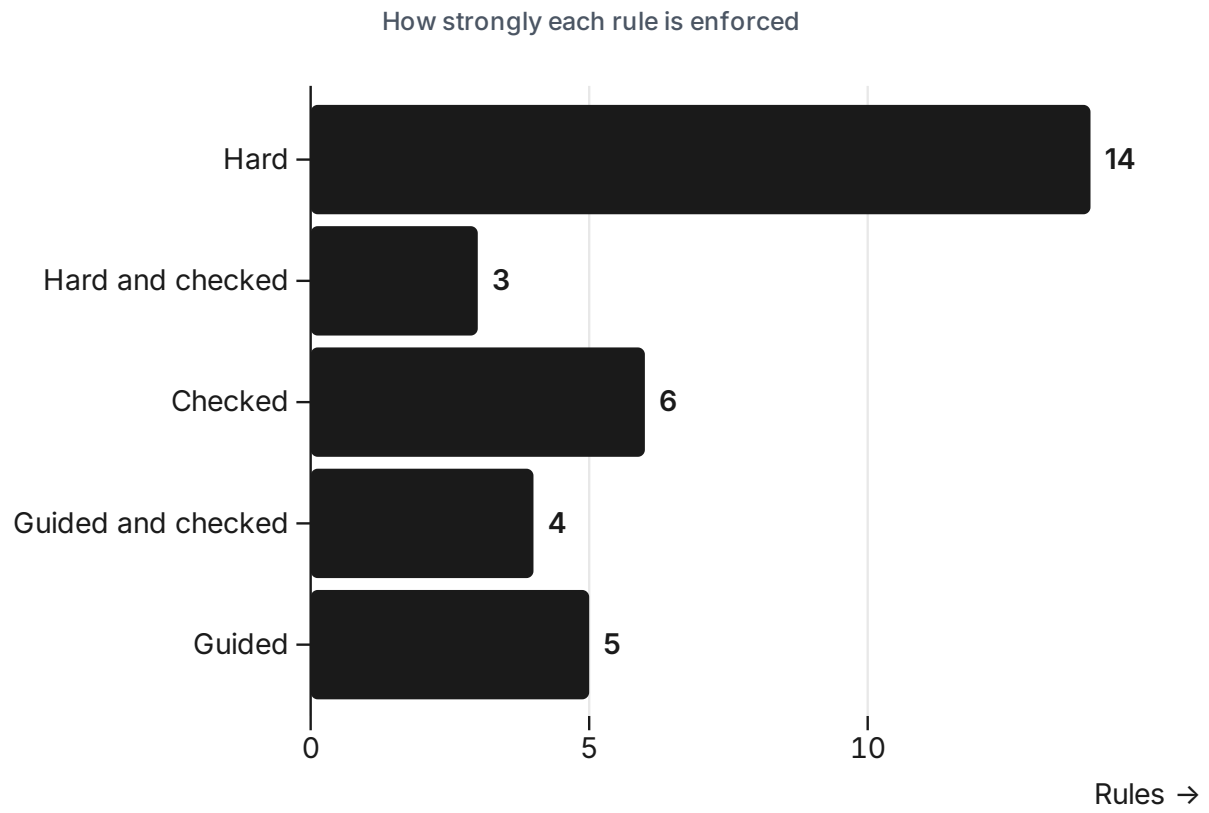
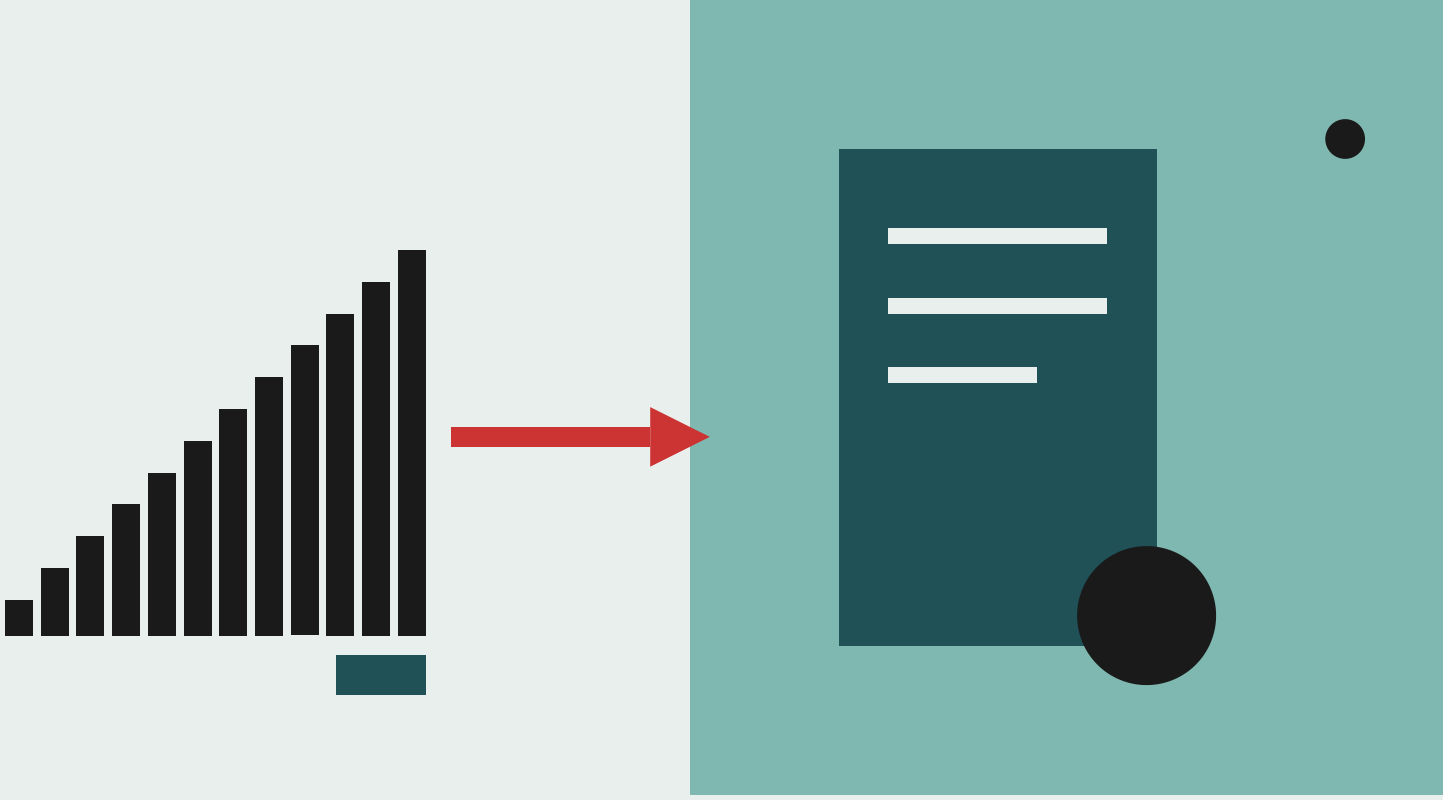


Figure 9. The 32 rules in chapter 14's matrix by strength. Six of the hard rules have stated limits.

The guide keeps an honest list of what the guards cannot do. The most important limits:

- Shell permission rules match the command text Claude writes. A program started another way is not caught by them.
- The personal-information guard catches identifiers with a known shape, such as an account or tax number. It does not catch names or street addresses, and the privacy reviewer covers those afterwards.
- ‘Every task maps to a card’ and the door checks depend on Claude following the skills. Reviewers catch a broken rule after the fact.
- Approvals are recorded, not proven. The administrator role becomes hard only once central’s remote repository lets only the administrator push.

© 2026 Sebastian Bourges | SimplifyUX



Part IV

Build it and use it

How is it built, phase by phase, and what is a first project like?

8 **Install it by hand, phase by phase** 30
The 12 phases Claude runs for you, from discovery to other people's machines, and the checks between them.

9 **Run your first project** 32
One invented project, a booking page for a small studio, moving through the six steps.

© 2026 Sebastian Bourges | SimplifyUX

8
Install it by hand, phase by phase

The 12 phases Claude runs for you, from discovery to other people's machines, and the checks between them.

Claude normally runs these phases for you. They are listed here so you know what it is doing, and so you can follow them yourself. Each phase ends with a check, and the next phase starts only once it passes.

© 2026 Sebastian Bourges | SimplifyUX

00 **Discovery and plan**

Claude asks the discovery questions, fills in every placeholder and drafts central's first release with its six cards.

Check: every placeholder has a value or says 'not used', and you have approved the plan.

01 **Prerequisites**

Claude installs the command-line tool, Node.js and Git, asking before each.

Check: claude, node and git each print a version.

02 **Central repository**

The central folder, its Git repository, its brief with the board, and a work folder kept outside Git.

Check: Git shows the files, and the board holds the release, its six cards and your approval.

03 **Core rules and standards**

The core rules file and every spec the rules point to.

Check: the core rules are under 150 lines, and every file follows the file standard.

04 **Tools**

The personal-information patterns, the Markdown checker and the new-project tool.

Check: the checker passes on central.

05 **Project template and prompts**

The folder every new project starts from, and the prompts you paste.

Check: the checker passes on the template.

06 **Plugin: skills, agents and hooks**

The 11 skills, the seven agents and the four hooks.

Check: the plugin validates, and each hook gives the expected answer to a sample input.

07 **User settings and installer**

Your user CLAUDE.md, your settings (with your language, if Claude should answer in another) and the installer, run once as a dry run, then for real with your approval.

Check: in a sample project, a second session quotes the status line, says it is in Understand, and asks before it edits central or deletes a file.

08 **Design tool connection**

Claude connects to your design tool and maps its library.

Check: a pinned library search returns a component you know.

09 **Your own libraries**

Optional: component and token packages fed by the design tool's values.

Check: the libraries install from their lockfile.

10 **Verify the whole system**

All six reviewers on central, a sample project from the template, and the enforcement list completed.

Check: every reviewer approves and every rule maps to its enforcement.

11 **Onboard other people**

Each person pastes the prompt for another machine. Claude copies central from your private git host and runs the installer for them, and each new project names its decider when it is created.

Figure 10. The 12 phases, from discovery to other people's machines. Phases 0 to 10 each end in a check. Phases 8 and 9 are skipped if you have no design tool or build no libraries of your own.

© 2026 Sebastian Bourges | SimplifyUX

9
Run your first project

One invented project, a booking page for a small studio, moving through the six steps.

Once the system is set up, a project starts when a person pastes the new-project prompt and fills it in. It asks for the decider, a one-line summary, the objectives, what is in and out of scope, the hard stops, and the research you already have. Claude never starts a project on its own. These exchanges show one invented project, a booking page for a small studio, moving through the six steps.

> You
>
> /ux-governance:new-project Studio bookings. Decider: Alex Example. Summary: clients cannot book a session online. Research I already have: 6 interview notes and 1 support-ticket export.

> Claude
>
> I've created the project from the template and drafted its README and brief. The file check passed, and three reviewers approved the draft. Success measures are still TBC.
>
> The project starts at Understand. There's no problem statement yet and no release. What should Alex approve first?

Exchange 1. Setting up. Claude creates the folder, drafts the brief and stops.

© 2026 Sebastian Bourges | SimplifyUX

> You
>
> Summarise the interview notes. Then write two customer types from them.

> Claude
>
> Here are the notes sorted into four themes, each with the notes it came from. The strongest theme is that clients give up when they can't see open times.
>
> I won't write the customer types. That's the team's job, because it needs judgement about real people. I've drafted an interview guide for the gaps, ready for Alex to approve.

Exchange 2. Understand. Claude does the reading and sorting, and keeps out of the customer types.

> You
>
> Draft the problem statement.

> Claude
>
> Three candidates for Alex, each citing its notes. The first: 'A returning client needs a way to see open times before they ask, because four of six gave up waiting for a reply.'
>
> Only Alex can agree one. Until then, Sketch stays closed.

> You
>
> Alex agrees the first one.

> Claude
>
> Recorded: agreed by Alex Example, with today's date. The project moves to Sketch.

Exchange 3. Define. The decider agrees the sentence, and every later step points back to it.

© 2026 Sebastian Bourges | SimplifyUX

> You
>
> Which of the sketches is best?

> Claude
>
> I won't rank them in Sketch. That's the decider's call in Decide. I've added eight raw ideas to the board and drafted a shortlist sheet for Alex: two or three options, each traced to the problem statement.

> You
>
> Alex picked option two.

> Claude
>
> Recorded as the winner, with Alex's reasoning. Here is a stress test of option two: three ways it could fail, and what would show each one.

Exchange 4. Sketch and Decide. Claude adds ideas and stress-tests the winner, but never judges between them.

© 2026 Sebastian Bourges | SimplifyUX

> You
>
> Run the usability test.

Claude

I've drafted the test plan and the card. Its FALSIFIER reads: 'Fewer than 3 of 5 participants find an open time without help.' A person runs the sessions. I'll sort the notes afterwards.

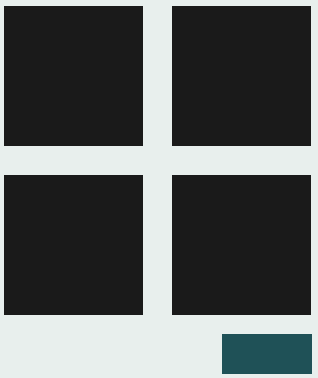
> You
>
> Five sessions done. Build the page.

Claude

Built from your design library's components only. I didn't invent any. The accessibility reviewer approved. Two checks need a person: a keyboard-only pass and one screen-reader pass. The release can close once Alex writes the safety statement.

Exchange 5. Validate and Deliver. People test and check; the decider closes the release.

© 2026 Sebastian Bourges | SimplifyUX



Part V

Where it comes from

What has been built with it, and what is it built on?

10	**Software built with the method** Four products built and run with this method, each written up as a case study.	38
11	**Official references** Anthropic's own pages for every Claude Code feature the system uses.	39
12	**Credit** Who designed the method, and the licence the guide is published under.	40

© 2026 Sebastian Bourges | SimplifyUX

10
Software built with the method

Four products built and run with this method, each written up as a case study.

I build and run my own software with this method. Each is written up as a case study:

- A task board that a person and a dozen AI sessions share, which runs on the kanban method above.
- A desktop design tool built for an AI agent to drive.
- One console to create, watch, change and roll back websites.
- A notes app that keeps every note as a plain file.

© 2026 Sebastian Bourges | SimplifyUX

11
Official references

Anthropic's own pages for every Claude Code feature the system uses.

The guide is built on Claude Code’s own features, and chapter 11 lists the official page for each. Read these before you change any settings, permission, hook, plugin, skill or agent file:

- Settings and Permissions
- Hooks reference
- Plugins overview
- Skills and Subagents
- How Claude remembers your project
- Use Claude Code in VS Code

© 2026 Sebastian Bourges | SimplifyUX

12
Credit

Who designed the method, and the licence the guide is published under.

UX Central Governance is a method I designed and own. That covers its six-step design process, its rule that AI is an intern and not an oracle, and its kanban method with FALSIFIER lines and doors.

The guide and its files are published under the Creative Commons Attribution 4.0 licence. You may copy, share and adapt them for any purpose, including commercial work. Credit the author, link to the licence, and say whether you made changes.

© 2026 Sebastian Bourges | SimplifyUX