
19 things I do differently when building an AI agent.

A working discipline, not a listicle — from someone shipping one right now.

THE PROBLEM

Prompting an agent straight into existence creates technical debt on day one.

"Build me an agent that does X" skips every decision that determines whether it survives contact with real data and real users. **Speed at the start becomes rework later.**

Describe the problem before you write a single line.

Use your platform's plan mode — Claude, ChatGPT, whichever chatbot. **It will ask clarifying questions.** Answer them. That's the step most builders skip.

NOT THIS

"Create me an agent that does X and Y."

THIS

Describe the problem in plan mode → let it interrogate the requirements → agree on scope before any code.

One model's architecture is an opinion. Two disagreeing models is a review.

STEP 1

Build the architecture with Claude.

STEP 2

Get it critiqued by a different model (e.g. ChatGPT) — full context included.

STEP 3

Feed the critique back to Claude, revise, and repeat once more before writing code.

Get a day-by-day plan. Start with the folder structure.

```
/agent-project
├─ agents/ ← orchestrator + subagents
│   ├─ orchestrator.py
│   └─ subagents/
├─ skills/ ← one skill, one file, <200 lines
├─ config/
├─ tests/
├─ docs/
│   └─ decision-log.md ← updated daily
├─ .env.example ← no real secrets, ever
├─ .gitignore
└─ README.md
```

After every file, make the AI explain it back to you.

Section by section. Before the next file gets built. **Skip this and you don't have an agent — you have cognitive debt:** a system you can't explain because you never had to understand it.

Tomorrow's session shouldn't start blind.

END OF EVERY DAY

Log every decision made and everything implemented — it's tomorrow's reference for testing and debugging.

ALWAYS

A real README — not a placeholder.

NEVER

Real credentials (PAT, usernames, passwords) committed to a .env file.

Cost and structure aren't cleanup tasks. They're inputs.

COST

Think about cost when you start building — not after. Bolt it on last and you end up rearchitecting the whole thing.

SKILLS

Keep skill files under 200 lines. Past that, models start missing their own instructions. Split into smaller skills.

AGENTS

One orchestrator, subagents reporting to it, dispatchers to manage the handoffs. Divide tasks as granularly as the problem allows.

THE TAKEAWAY

None of this is about moving slowly for its own sake.

It's about knowing exactly what got built — instead of finding out in production.

A prompt gets you a demo.

A planned, cross-examined architecture gets you something that survives contact with real users.

YOUR TURN

What's the one step in your AI-agent build process you skip when you're in a hurry — and has it ever cost you later?

Follow [Jashan Sadioura](#) for more on building AI agents that survive production.