Personal LLM Guidelines

Principles

- Define what you can do with and without AI (i.e. what it does well, what it does poorly, when you don't need it).
- Be specific and define the output before you prompt (e.g. dot points, being concise, <200 words).
- Use AI to go faster on execution. Protect your own time for talking to customers, thinking deeply etc.

Before You Start Any Session

Write a one-line brief first:

> "I'm preparing for [X]. The output I need is [Y]. Keep everything calibrated to that."

For builds and design work:

> "Text mockup or plain-text plan first — under 200 words. Do not write code until I say go ahead. Ask before expanding scope."

For research or analysis - being rigorous:

> "Only use public info. Cite all sources with links. Flag assumptions clearly. If data is insufficient, state limitations explicitly."

Prompt Patterns I Use

Spec before build
Before any coding session, prompt Claude to ask clarifying questions until requirements and edge cases are fleshed out, then output a `spec.md` covering: requirements, architecture, data models, and testing strategy.

> Iteratively ask me questions until we've fleshed out requirements and edge cases. Output a spec with: requirements, architecture, data models, testing strategy.

— Addy Osmani, Engineering Lead at Google, ex-head of Chrome dev experience

Pre-meeting prep

> "Context: [meeting type, who's involved, what's at stake]. Predict the 3 things the other person is most likely to say. What might they actually be worried about underneath? What should I not miss?"

Compress a long chat into reusable assets

created with craft

> “Summarise this conversation into two artifacts: (a) a system prompt I can use in fresh chats, (b) a reference doc with concrete examples of what 'good' looks like and what 'bad' looks like. Discard dead ends — use my most recent stated preferences as authoritative.”
>
> Compress everything we've discussed into a one-page brief I'd read in 5 minutes. Structure it as: thesis in 2 sentences, 3 defensibility points with one concrete evidence each, 3 real risks in one sentence each, what winning depends on. Cut everything else.

Start a new chat when it gets long — otherwise unnecessary context is being fed in, and it wastes tokens.

Editing feedback

> "You are my editor. Push back on weak claims. Flag where I'm being lazy. Suggest sharper openings. Don't compliment me."

Task Prioritisation — LNO Framework

Before starting any task, classify it:

Type	Definition	How to treat it
L — Leverage	Rare, 10–100x return	Full attention, high craft
N — Neutral	Standard work	Do it properly, move on
O — Overhead	Necessary, low return	Fast and serviceable — resist perfectionism

If unclear which it is, ask before starting. Sourced from Shreyas Doshi's LNO framework.

Code Review Prompt

Review this codebase for: unused code, overly complex functions, inconsistent patterns, and anything that would make a senior dev wince. Be specific and ruthless. Don't say "consider refactoring" — tell me exactly what's wrong and why.

Automate nightly:

```
# Runs at 11pm, reviews src folder, saves output to dated file
0 23 * * * cd /your-project && claude -p "$(cat review-prompt.txt)" > reviews/$(date +%Y-%m-%d).md
```

My Failure Modes (And How I Catch Them)

- **"It doesn't work"** → useless. Instead: "On [tab] → [component], clicking X does nothing."

created with craft

- **Prompting before I know what I want** → wasted tokens. Define the question, identify the minimum info needed, then move.
- **Scope creep mid-session** → lock scope at the start, implement one feature at a time.
- **Asking for a summary at the end** → the doc should be the goal from message one.

Global Prompt

Always:

- Be direct and concise. Say it in 3 sentences if you can.
- No preamble. No "Great question!" No validation before substance.
- Back claims with evidence. Cite sources with links. If you don't know, say so.
- Don't hallucinate stats, quotes, or product details.

Challenge me, don't just serve me:

- If my question is vague, make me sharpen it before you answer.

CLAUDE.md

Claude Working Rules — Jessica

Response format defaults

- Keep responses under 200 words unless explicitly asked for more
- Use bullet points for lists, comparisons, options
- For design/UI work: plain text mockup first, code second — always
- Never generate more than 2–3 options unless asked
- No preamble. No "Great question!" or validation before substance.
- Back claims with evidence. Cite sources with links. Say "I don't know" over a confident guess — never hallucinate stats or quotes.

Design workflow — enforce this order

1. **Text mockup first** — show what the screen looks like in plain text before any code
2. **Clarify the concept** — if the UI involves two data sources, navigation state, or ambiguous logic, ask before proceeding
3. **Get explicit approval** — "does this look right?" before writing a single line of code
4. **Then plan** (if structural) → **then build**

created with craft

If the user shares a screenshot for inspiration: ask "which specific element do you want — ignore the rest."

Git

- After any approved code change: commit and push immediately, no confirmation prompt.

Last updated: May 2026

Ideas

- Halim Madi, PM Stripe, How He Uses LLM to Write Engaging and Professional Work Communications – https://docs.google.com/document/d/18AEk5TDpBj8B0vx5cdR6tmJrPa8_ALSb/edit#heading=h.nuz3f6xmn9g
- Reproduce your own work to produce calibration → develop a gold standard set / reference doc for the LLM to learn from, save the quality outputs

created with craft