FROM THE APOLLO LENS DOWNLOAD PAGE

AI-Assisted Development Practices

Twenty-one working rules for building software with an AI coding assistant. Short by design — read it at the start of a session, not once.

APOLLO LENS SOLUTIONS

RULES 1–7

Set up before you build

1. **Start with a plan.** Write the spec first. The model builds exactly what you describe, so decide up front.
2. **Maintain `CLAUDE.md` / `AGENTS.md`.** Stack, conventions, off-limits files. It's the briefing every session starts from.
3. **Use skills.** Pre-built for common formats; custom ones for domain rules (survey standards, DXF handling) so they apply consistently.
   Install only from a source you trust and watch for look-alike names; read a skill before installing it; never run a "prerequisite" command unread.
4. **Match the model to the task.** Cheap and fast for boilerplate and edits; the strongest model for architecture, debugging, and security.
5. **Keep secrets out of prompts.** Credentials live in `.env` (git-ignored). The model only needs to know the variable *name*.
6. **Prefer boring tech.** Postgres, Python, React. Models are most reliable where the training data is deepest.
7. **Build modular.** Small files, one job each, clear interfaces between them. The model reasons well about one piece at a time and poorly about a 2,000-line file. A mistake stays contained, and a piece can be swapped without touching the rest.

RULES 8–15

While building

8. **Small, verifiable steps.** One feature or fix per request. Large outputs hide bugs.
9. **Commit before every AI edit.** `git checkout` is the undo button.
10. **Explain before build.** Ask "how would you approach this?" first. A cheap way to catch wrong assumptions.
11. **Give real inputs.** Sample files, actual error text, screenshots — not descriptions.
12. **Tests are the referee.** Write them; make the AI run them. "Make the tests pass" beats "looks right."
13. **Read the diff.** The summary and the actual change can differ. Skim every changed line.
14. **Fix root causes.** Ask "why did this break?" before accepting a patch.
15. **Ship a thin slice end to end** before adding breadth. Integration problems surface early and cheap.

RULES 16–21

Over the long haul

16. **Use GitHub for everything.** History, reversibility, and the ability to resume months later.
17. **Keep `DECISIONS.md`.** "We chose X because Y." Stops relitigating, and stops the AI reversing deliberate choices.
18. **Fresh sessions often.** Long chats drift. New session + `CLAUDE.md` = clean context.
19. **Second opinion on architecture.** A different model or a fresh session critiques the plan.
20. **Ask for the plain-English version.** If you can't explain the code, you can't maintain it.
21. **Learn one thing per session.** Pick something the AI did and understand it fully.

WHY THESE WORK

Every rule points the same direction: **the AI does the typing; you own the decisions.** Spec, tests, diffs, and decisions are the parts a professional signs off on — the model produces them, you approve them.

IN SURVEY TERMS

The spec is the scope letter. Tests are the closure check. The diff is the redline review. `DECISIONS.md` is the project notes you wish the last surveyor had left you.

MAKE IT STICK

Paste rules 1–21 into the bottom of your `CLAUDE.md` under a heading like `## Working rules`. The AI then reads them every session — and so do you.

Good practices are cheaper than good debugging. — Questions, or want these adapted to your firm's workflow? **info@apollolens.io**