CLAUDE CODE SERIES

CLAUDE.md

The Complete Setup Guide for **AI Code Quality**

Stop Prompting. Start Designing. Your AI Writes Better Code When You Onboard It Properly.

By Ehsan Khan | ehsankhanseo.expert

Applies to: Claude Code • Cursor • Zed • Any AI Coding Agent

📄 What Is CLAUDE.md?

The file most developers never set up properly

CLAUDE.md is a special markdown file that Claude Code reads automatically at the start of every single session. It holds persistent context — your stack, your rules, your workflow — so you never have to repeat yourself.

Think of it as onboarding your AI the same way you'd onboard a new developer. Without it, Claude starts every session knowing nothing about your project.

> 💡 **Without CLAUDE.md: Claude guesses your conventions, ignores your stack, and writes code that doesn't match your codebase. Every. Single. Session.**

Chat vs. CLAUDE.md — The Core Difference

	Without CLAUDE.md	With CLAUDE.md
Memory	Starts fresh every session	**Persistent context across every session**
Conventions	Guesses your code style	**Follows your exact rules**
Stack	Uses whatever it knows	**Knows your exact dependencies**
Workflow	Random approach	**Follows your defined process**

Quality	Inconsistent	**Consistent and on-brand**

📁 The 3 Scopes — Most People Miss This

Global, Project, and Folder — all three work together

CLAUDE.md isn't just one file. It has three levels, and Claude merges them all automatically. Most developers only use one and wonder why Claude ignores their rules.

01

Global Scope

~/.claude/CLAUDE.md — Lives in your home directory. Applies to every project on your machine.

Use for: your personal coding style, formatting preferences, universal rules you always want.

02

Project Scope

./CLAUDE.md — Lives in your project root. Applies to this project only. Commit it to version control so your whole team shares the same context.

Use for: stack details, architecture decisions, test commands, project-specific conventions.

03

Folder Scope

./src/CLAUDE.md or any subfolder — Applies only when Claude is working inside that folder.

Use for: microservices, monorepos with different rules per package, or isolated modules.

🔀 Merge Order: Global → Project → Folder. The last one wins on conflicts. Use this intentionally.

The WHAT / WHY / HOW Framework

🗂️ WHAT

Your tech stack, project structure, key dependencies, file layout. Give Claude a map of where everything lives.

🧠 WHY

Architecture decisions, patterns you follow, anti-patterns to avoid, reasons behind key technical choices.

⚙️ HOW

Commands, test workflows, build process, deploy flow, how Claude should verify its own changes. This is where Claude learns what 'done' looks like.

Skip any of the three = Claude guesses. And it guesses wrong.

✍️ How to Write a Great CLAUDE.md

The rules that separate good output from great output

Start With /init — Don't Write From Scratch

Run the /init command in your project directory. Claude analyses your codebase — build systems, test frameworks, code patterns — and generates a starter file automatically. Then delete what you don't need.

```
# In your terminal inside Claude Code:
/init

# Claude generates CLAUDE.md based on your actual project structure
# Then edit, prune, and refine it
```

Vague vs. Specific — The Single Biggest Mistake

❌ Vague — Claude Ignores	✅ Specific — Claude Follows
"Write clean code"	**"camelCase for variables, PascalCase for components"**
"Test everything"	**"80% coverage minimum, run: npm test --watch"**
"Use TypeScript"	**"TypeScript strict mode, no any types, always define interfaces"**
"Handle errors properly"	**"Always use try/catch in async functions, log to console.error"**
"Don't use old methods"	**"Do not use moment.js — use date-fns instead"**

The 5 Rules of a Good CLAUDE.md

1 **Run /init first**
Let Claude analyse your codebase before you write a single line. Delete what's obvious.

2 **Keep it under 500 lines**
Context is precious. Every line competes for attention. Ruthlessly prune.

3 **Expect ~70% compliance**

Not a bug — it's reality. Claude may deprioritise rules it deems irrelevant to the task.

4

Update it monthly
Ask Claude: "Review this CLAUDE.md and suggest improvements." Delete what's stale.

5

Reference configs, don't copy
Point to your .eslintrc or tsconfig.json. Don't copy their full contents into CLAUDE.md.

📋 A Real CLAUDE.md Example

What a properly structured file looks like

```
# Project: ShopFront
Next.js 14 e-commerce app — App Router, Stripe, Prisma ORM

## Stack
- Next.js 14 (App Router only — no Pages Router)
- TypeScript strict mode — no `any` types
- Prisma ORM — no raw SQL
- Zustand for state — no Redux
- date-fns — no moment.js

## Naming Conventions
- Variables: camelCase
- Components: PascalCase
- Custom hooks: prefixed with `use` (e.g. useCartItems)
- DB models: prefixed with `db` in variable names

## Testing
- Unit tests: Vitest
- Integration: Playwright
- 80% coverage minimum before merging
- Test files: live next to the file they test (*.test.ts)

## Commands
- Dev: npm run dev
- Test: npm test --watch
- Build: npm run build && npm run lint

## Rules
- Always explain architectural decisions in response
- Ask clarifying questions before implementing complex features
- NEVER modify /migrations directly
```

This file is under 30 lines. That's the goal — dense, specific, no filler.

Common Mistakes to Avoid

#	Mistake	Fix
1	File too long (500+ lines)	**Claude ignores buried rules. Ruthlessly prune to essentials only.**
2	Only one scope used	**Use all 3: global, project, folder. Each layer adds precision.**
3	Vague instructions	**Every rule must be specific and testable. If it's ambiguous, delete it.**

4	Never updated	**Monthly review with Claude keeps it clean and relevant.**
5	Copying full config files	**Reference the file. Don't paste its full contents.**
6	No workflow section	**Claude needs to know how to verify its own work — tests, builds, checks.**

💬 Final Thought

Top engineers aren't better at prompting. They're better at designing CLAUDE.md.

The developers who get 10x output from AI aren't writing longer prompts. They're investing 30 minutes upfront to properly onboard their AI — with the right stack context, the right rules, the right workflow.

Claude Code can follow ~150-200 instructions with strong consistency. That's not a limitation — that's a feature. Use it.

Set it up once. Update it monthly. Watch your AI code quality compound.

Don't just prompt. Design the system.

Want This Set Up For Your Business?

I help small businesses rank on Google and automate their workflows using AI + Semantic SEO.

Whether you need to show up on Google or save hours with AI automation — I can help you build the system.

Book a Free Strategy Call

ehsankhanseo.expert

Connect & Send a Direct Message

🔗 LinkedIn: Ehsan Khan — Semantic SEO

💬 WhatsApp (PK): +92 301 859 7809

💬 WhatsApp (BD): +880 1717-519345

Services: SEO Strategy | AI Workflow Automation | Google Rankings | Content Systems

Serving small businesses globally — from startups to established agencies.
