



A FREE GUIDE

Before You Prompt

What Most People Get Wrong About AI —
and the Three-Part Framework Builders Use Instead

BY DAMIAN MARTINEZ

WHO THIS IS FOR

This guide is for anyone who has spent real time with AI tools and still feels like they are not getting what they came for. You don't need a technical background to close the gap between chatting with AI and actually building something real. You need a different mental model.

You're using AI like a search engine.

When you trained yourself to use Google, you trained yourself to strip a question down to a few keywords and fire. That behavior is automatic now. You bring it directly into AI tools without realizing it.

Search engines are designed for one-off questions with no memory and no context. This works for looking things up. **It does not work for building.**

Builders use AI differently. They don't fire questions. They give instructions. They provide context before they ask for anything. They define what the AI is not allowed to do. They tell the AI how to verify its own work. And then they ask.

The output is completely different — not because they're using a different tool, but because they're communicating in a way that actually works.

"As long as you treat AI like a search engine with better grammar, a generic conversation is all you will ever get."

AI is not a vending machine.

Most people treat AI transactionally. Put in a request, get back an output, evaluate, try again. The relationship is transactional. The results reflect that.

Builders treat AI as a new team member on their first day. Think about that for a moment. If you hired someone capable, intelligent, and eager to help — but they knew nothing about your project, your rules, or what already exists — what would happen if you handed them a task with zero context?

They'd do something. It might even be impressive. But it would be wrong. Not because they're bad at their job. Because they had no information.

That's exactly what happens when you prompt AI without preparation. It does something. It does it confidently. And it's often wrong in ways you don't immediately catch.

"Every session with AI is an onboarding. The quality of what you get is directly tied to the preparation you bring in."

Context. Constraints. Verification.

Everything that separates a builder's output from a chatter's output comes down to three things, applied in order, **before the actual request is ever made.**

These aren't three separate tactics. They're three layers of a single system. Remove any one of them and the system breaks down — not every time, but reliably over time.

01

CONTEXT

What you tell AI before asking for anything. What the project is, what exists, what this session is for.

02

CONSTRAINTS

What AI is NOT allowed to do. What not to break, rename, or reorganize. The rules before the request.

03

VERIFICATION

How AI checks its own work. Before finalizing, verify what changed and what still works.

Context

THE MOST SKIPPED STEP. THE MOST CONSEQUENTIAL.

Before you ask AI to do anything, it needs to know what world it's operating in. Not in a vague, general way — specifically. What is this project? What already exists? What role is AI playing right now? What is this particular session for?

Without context, AI guesses. It produces something plausible based on your request alone — and plausible is the enemy of accurate. Plausible means it looks right. It reads right. It might even feel right. But it doesn't fit your project, because AI had none of that information when it built the output.

This is why two people can ask AI the exact same question and get dramatically different results. The difference isn't in the question. **It's in what came before the question.**

Builders build context first. They describe the project, what already exists, what this session is specifically intended to accomplish. They hand AI the blueprint before they ask it to swing a hammer.

Constraints

ONCE AI HAS CONTEXT, IT NEEDS RULES.

Constraints are instructions about what AI is **not allowed to do**. Most people never think about this layer because they're focused on telling AI what to build, not what to leave alone.

Here's why it matters: **AI doesn't edit. It regenerates.** When you ask AI to add something or fix something, it doesn't make a surgical change — it rebuilds the output. Anything not explicitly protected is fair game for change.

This creates a specific and extremely frustrating problem. You ask AI to add one feature. It adds the feature. It also quietly reorganizes three other sections, renames a variable, and removes something it considered unnecessary. The output looks like it addressed your request. But the underlying work shifted in ways you didn't ask for.

"Don't modify what already works. Don't rename. Don't reorganize. Your job is to add this one thing without touching anything else."

Verification

HOW YOU CLOSE THE LOOP.

After AI produces output, the natural instinct is to evaluate it yourself. This works at small scale. It breaks down as complexity grows. Builders add a different layer: **they make AI responsible for checking its own work** before they ever review it.

This isn't about trusting AI to catch everything. It won't. It's about building a habit that filters out the most common and costly errors before they reach you — and that forces AI to account for its own actions.

Verification instructions tell AI what to check before finalizing:

- "Did you change anything I didn't explicitly ask you to change?"
- "Does the thing I asked you to build actually work as described?"
- "List anything you modified that I didn't request."

When these questions are built into the session before the actual work, the output quality improves substantially. Verification is the layer that makes everything else stick.

Why all three together.

Each pillar helps on its own. None of them is sufficient alone.

CONTEXT WITHOUT CONSTRAINTS

AI understands your project — and will still quietly reorganize, rename, or "improve" things you didn't ask it to touch.

CONSTRAINTS WITHOUT CONTEXT

AI has rules but nothing to apply them to. It knows what not to do — but it's operating blind on everything else.

CONTEXT + CONSTRAINTS WITHOUT VERIFICATION

Better sessions, but all quality control rests on you. At scale, you miss things too.

This is a sequence, not a checklist. You build it into every session before the actual work begins.

PROOF

Non-coder. Real platform. Real users.

I am not a developer. I have no computer science background and I don't write code in the traditional sense. Despite that, I built and launched a live software platform with real paying users.

The platform runs. People use it. That is the proof. The outcome is the direct result of following this process.

- Payment processing
- Report generation
- API integrations
- User accounts & subscriptions
- Automated email delivery

The gap between having this framework and not having it is the difference between building something and talking about building something.

What this guide does not cover.

This guide provides the mental model. The implementation layer — the tools that make this executable — is reserved for the course.

— TEMPLATES

Copy-paste session starters and context structures. The exact language builders use.

— WORKED EXAMPLES

Real before-and-after prompts showing each layer applied to actual projects.

— GUARDRAIL SYSTEM

Backup protocols, regression detection, and step-by-step recovery procedures.

— BUILD SEQUENCE

What to build first, what to defer, what to skip — the order that prevents rework.

— FAILURE STORIES

The specific mistakes that cost weeks, and the exact adjustments that fixed them.

— GUARDRAIL CHECKLIST

The session checklist that runs before and after every AI modification.

This guide is the **"why."** The execution layer is the **"how."**

Understanding vs. building.

Understanding a framework and being able to use it are two different things.

You now know that builders use Context, Constraints, and Verification before every session. You understand why each layer exists. You can already see how it would change the quality of what you get from AI.

That understanding is real. It matters. It's also not enough to build with.

The gap between understanding the framework and applying it consistently — across different types of projects, different levels of complexity — is the execution layer. It's the templates, the worked examples, the failure cases, the checklists, and the sequence decisions that turn a concept into a repeatable system.

That layer takes longer to explain. It requires examples that go deep enough to be genuinely useful rather than just illustrative. It includes the things that go wrong and the exact adjustments that fix them.

"Knowing the framework is the first step. Bridging the gap to action requires the execution layer."

The map from here.

1

THE FREE TRAINING

The free training walks through the execution layer — how a real build unfolds from idea to working product, how the framework looks in practice, and what separates the people who finish from the people who stay stuck. About six minutes.

damianmartinez.com/free-training

2

FROM PROMPT TO PRODUCT — \$17

The full communication framework with copy-paste templates, worked examples, the guardrail system, the build sequence, and the five most costly mistakes — with the exact prevention for each. The execution layer behind everything in this guide.

damianmartinez.com/before-you-prompt

Start with the free training. Then decide.