AI Code:

Slop and Rot

Addressing the elephant in the pull request. Why bad code isn't a new problem, just a new medium.

> "AI generated code is bad, because AI generates slop and rot."

— Common Developer Sentiment

Defining the Terms

<Slop />

Plausible but low-quality code that sneaks past review. It looks correct at a glance but lacks robustness, edge-case handling, or efficiency.

<Rot />

The long-term structural decay caused by accumulating slop. It's the "death by a thousand cuts" that makes a codebase impossible to maintain or refactor.

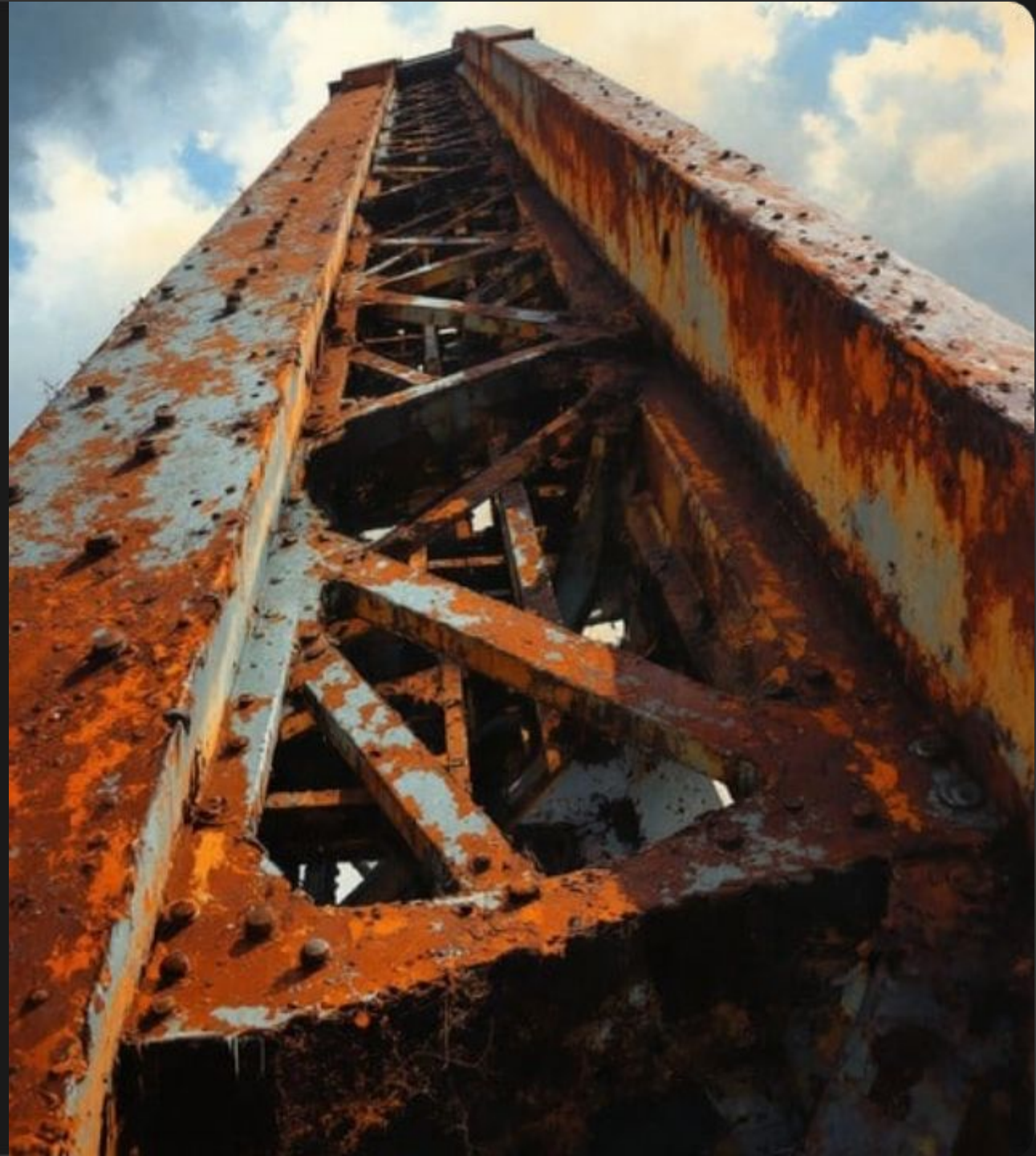
The Anatomy of Slop

- **Superficial Correctness:** It compiles, it runs, but it's fragile.
- **Lack of Context:** AI guesses the intent but misses the broader architectural constraints.
- **Subtle Bugs:** Errors that are hard to spot in a quick diff but crash production under load.
- **Review Fatigue:** Reviewers assume it's right because it looks authoritative.

The Long-Term Effect: Rot

When slop accumulates, it calcifies into Rot.

- System rigidity increases; changes become risky.
- Onboarding new developers becomes a nightmare.
- Refactoring is impossible because nobody understands the generated logic.
- The codebase becomes a legacy burden immediately after creation.



But doesn't that sound familiar?

Old Problems, New Names

AI Criticism	Human Equivalent	The Reality
Slop	Cargo Cult Code, Spaghetti Code	Copy-pasting without understanding.
Rot	Technical Debt	Choosing speed over quality.
Hallucinations	"It worked on my machine"	Assumptions vs. Specification.

We've Been Here Before

These problems existed long before LLMs.

- StackOverflow copy-paste driven development.
- Hacking away without specifications.
- Skipping code reviews to meet deadlines.
- Ignoring design patterns for "quick fixes".

Slop and Rot are not AI problems. They are workflow problems.

The Real Root Cause

No Standards

Accepting code—human or AI—without adhering to a strict style guide or architecture.

Weak Reviews

Rubber-stamping PRs without actually reading and understanding the logic.

Zero Spec

Generating code without a clear definition of what needs to be built.

The Enemy: "Vibe Coding"

Vibe Coding is the nemesis of professional engineering. It represents:

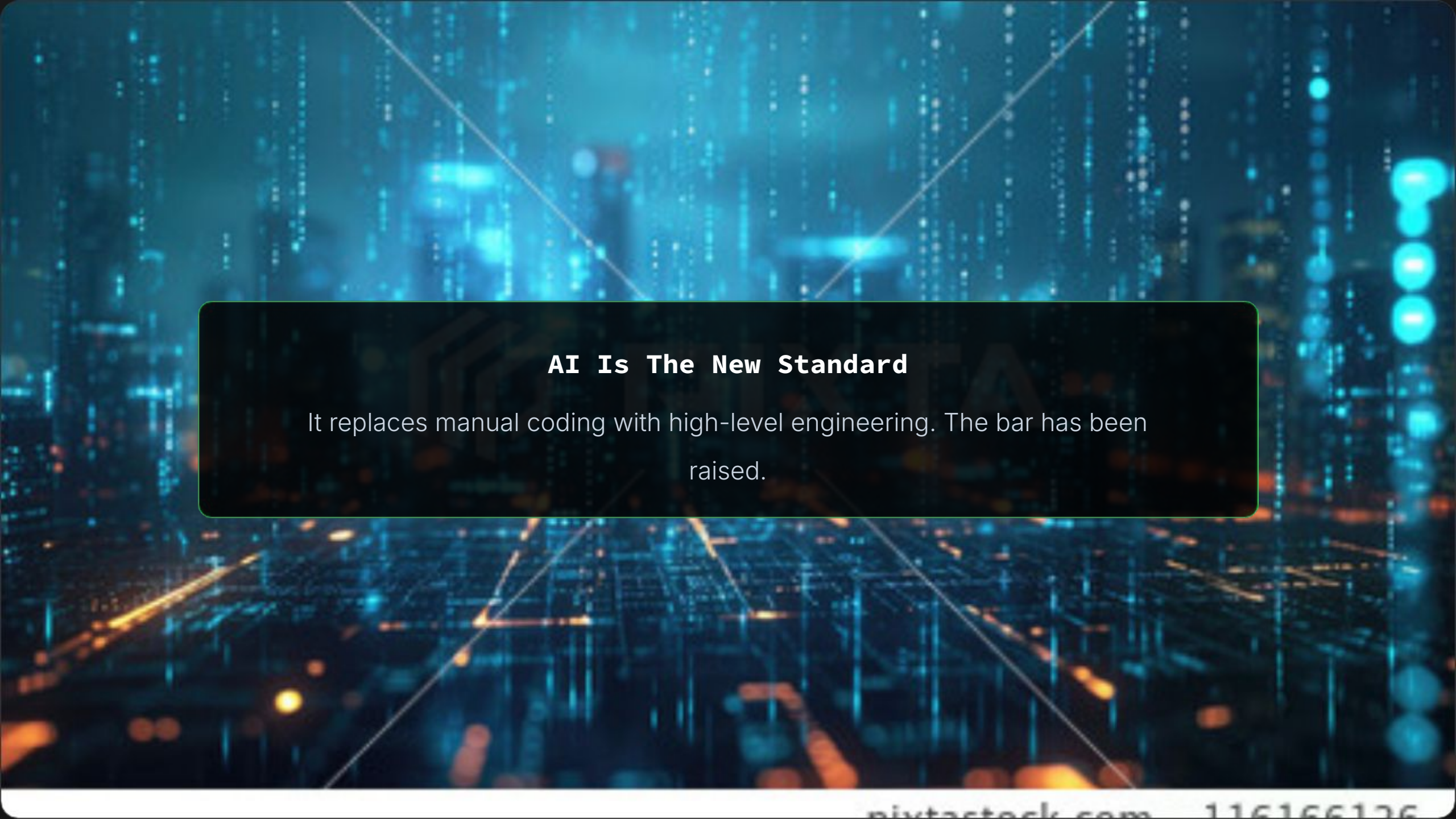
- Non-repeatable chaos
- Unmaintainable output
- Code that "almost" works

We need a process that turns AI into an engineer, not just a chaotic autocomplete.

The Solution: Mastery

With advanced techniques and context, AI performs as a **Senior Developer**.

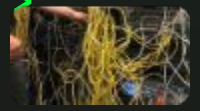
- **Strategic Review:** Shift focus from checking syntax to validating architectural fit.
- **High-Level Specs:** Providing deep context unlocks senior-level, production-ready output.
- **Automated Precision:** Use AI to enforce patterns and standards more rigorously than humans.
- **Orchestration:** We transition from writing code to directing intelligent systems.


AI Is The New Standard
It replaces manual coding with high-level engineering. The bar has been raised.

Q&A

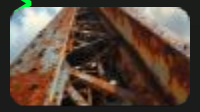
Let's discuss our workflow.

Image Sources



https://www.ashtonsolutions.com/wp-content/uploads/2019/05/CBG-Before-1.jpg

Source: www.ashtonsolutions.com

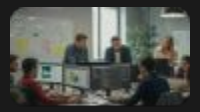


https://images.stockcake.com/public/1/b/0/1b0d5f66-7774-4761-9128-0d0d946c7750_large/rusty-bridge-detail-stockcake.jpg

Source: stockcake.com

https://i.redd.it/8j3u65w2ow191.png

Source: www.reddit.com



https://static.vecteezy.com/system/resources/thumbnails/071/530/832/small/software-developers-team-collaboration-meeting-in-modern-office-programming-code-review-and-discussion-photo.jpeg

Source: www.vecteezy.com

https://en.pimg.jp/116/166/126/1/116166126.jpg

Source: www.pixtastock.com