Why Your Claude Code Performance Tanks

(And How to Fix It)

One of the most common posts in Reddit's Claude Code community:

"My Claude Code performance has tanked and I'm not sure why"

Khur Boon Kgim

AI-Powered Dad and Technopreneur

Sound Familiar?

Help with Claude Code perfor... Ask Open App

← Return to Search Results

r/ClaudeCode 7d ago
takeurhand

Seeking help: My Claude Code performance has tanked and I’m not sure why

Recently, I’ve noticed a significant drop in the quality of Claude Code's responses, to the point where they are frankly subpar. It feels as if I’m using a 'throttled' version of Claude, perhaps 90% or even only 60% of Opus 4.5's full intelligence. Even when I use the 'ultrathink' keyword, the code implementation is still unsatisfactory. This isn’t just a subjective feeling; I have reached this conclusion through rigorous, objective testing.

I am a subscriber to Google Gemini CLI and OpenAI Codex, as well as Claude Max. I tested all three on the exact same project to implement the same tasks, using identical memory files (claude.md, gemini.md, and agents.md), identical plans, and identical requirements. Claude Code used to produce the best results, but now it is performing the worst.

I am constantly stuck in a 'test -> error -> fix -> test -> error' loop, often encountering unexpected and very low-level mistakes. This has made Claude Code unreliable for my workflow. As a loyal user, I need the Claude server to return 100% of Opus 4.5's capabilities to my client, not a degraded version. I would like the technical department to investigate my account to see what is wrong and if my performance can be fully restored. I truly do not want to abandon the long-term habits and workflows I’ve built with Claude Code.

Users have been asking this all year long.

→

I Had the Same Experience

When I started using Claude Code for a new project, it was magical.

As my project grew bigger, Claude Code started:

- Violating best practices I gave it
- Hallucinating code that contradicted my instructions
- Claiming completion when tasks weren't done

My hypothesis? Context collapse.

Anthropic Confirms It

Anthropic just released a best practice guide that confirms this.

WHAT I FOUND FROM MY RESEARCH

- Performance starts to degrade at **40-50%** of context window
- The more best practices you give, the faster it fills up
- When context is filling up, Claude ignores your instructions

> *If you give Claude Code lots of best practices to follow... it means nothing if Claude didn't follow them.*

The Fix

(1) BREAK BEST PRACTICES INTO SKILLS

Skills allow Claude to load best practices on demand.

(2) AGENTS + SKILLS

Skills alone isn't enough. When I tested, Claude loaded 5 relevant skills at once. Context filled up quickly. The real solution: **One agent. One skill.**

(3) APPLY AT EVERY STAGE

During planning, plan review, and implementation.

(4) CREATE MASTER SKILLS

Create master skills for planning, plan review, and implementation.

These skills repeat reviewers until all best practices pass.

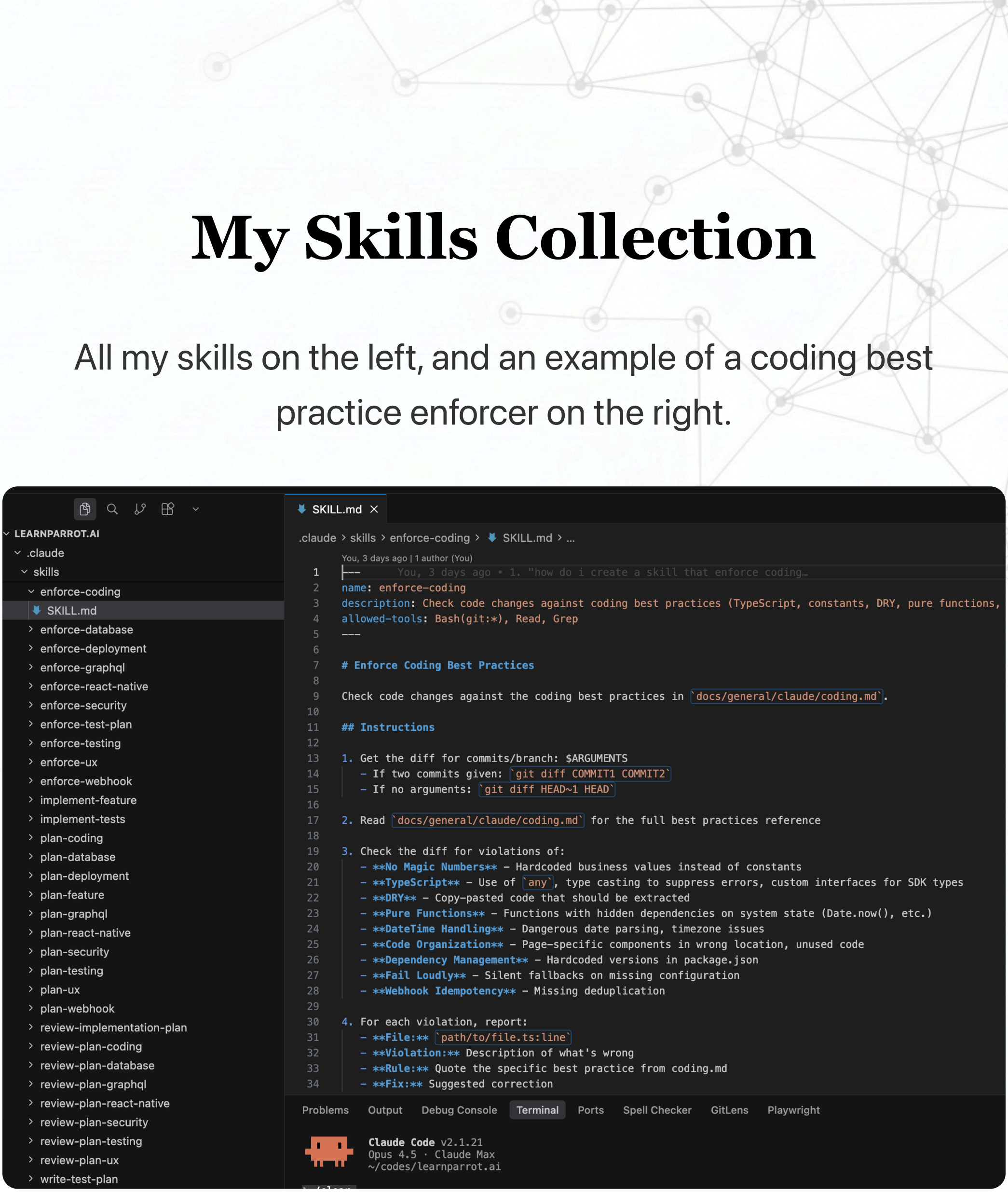
Skills in Action

Enforce skills caught violations and fixed them automatically.

```
● Good catch. The enforce skills found several violations. Let me fix the important ones:

  Summary of Violations:
  - enforce-database: Pre-existing timestamp issue (not introduced by this PR)
  - enforce-graphql: Pre-existing redundant familyId pattern (not introduced by this PR)
  - enforce-react-native: No violations ✅
  - enforce-ux: 2 issues - missing loading state during mutation, empty state needs guidance
  - enforce-testing: Missing edge case tests for aiMode with different ages

  Let me fix the UX and testing violations:

● Read 1 file (ctrl+o to expand)

● Update(apps/aiparrot/src/components/ai-mode-settings.tsx)
  └ Added 25 lines, removed 16 lines
      133   */
      134  function AiModeCard({ child, familyId }: { child: ChildProfile; familyId: string }) {
      135    const [optimisticMode, setOptimisticMode] = useState<AiMode | null>(null)
    136 +  const [isMutating, setIsMutating] = useState(false)
      137    const [, updateChildProfile] = useMutation<UpdateChildAiModeResult>(UpdateChildAiModeDocument)
      138
      139    const displayMode = optimisticMode ?? child.aiMode ?? 'learning'
     ...
      147    }, [child.aiMode, optimisticMode])
      148
      149    const handleToggle = async () => {
    150 +    if (isMutating) return // Prevent double-tap
    151 +
      152      const newMode: AiMode = isExploration ? 'learning' : 'exploration'
      153      const previousMode = child.aiMode
      154
      155      // Optimistic update
      156      setOptimisticMode(newMode)
    157 +    setIsMutating(true)
```

My Skills Collection

All my skills on the left, and an example of a coding best practice enforcer on the right.



Apply at Every Stage

DURING PLANNING

Load planning-specific skills and best practices

DURING PLAN REVIEW

Validate plans against architectural guidelines

DURING IMPLEMENTATION

Enforce coding standards with dedicated reviewers

MASTER SKILLS

Create skills that repeat reviewers until all best practices pass

Master Skill: Implement

Workflow phases, enforcement loop, and skill triggers.

● Created .claude/skills/implement-feature/SKILL.md.

Summary

Skill: implement-feature

Workflow

Phase	Action
0. Load Plan	Parse implementation plan, create task list from phases
1. Execute	Implement each phase, run enforce skills after each change
2. Test	Run unit + integration tests
3. Final Review	Run all enforce skills on complete diff

Enforcement Loop

For every code change:
Make change → Run enforce skill → FAIL? Fix → PASS? → Typecheck/lint → Next

Enforce Skill Triggers

Files Changed	Skills Run
packages/database/src/schema/*.ts	enforce-database, enforce-coding
apps/api/src/schema/*.ts	enforce-graphql, enforce-security, enforce-coding
apps/*/app/**/*.tsx	enforce-ux, enforce-react-native, enforce-coding
**/*.test.ts	enforce-testing, enforce-coding
**/validations/*.ts	enforce-security, enforce-coding
**/*webhook*.ts	enforce-webhook, enforce-coding

Minimum (Always Run)

- /enforce-coding - TypeScript, constants, pure functions
- /enforce-security - Input validation, auth

Review Result

Each review checks against documented best practices.

Review 3: Security Best Practices

Checking against docs/general/claude/security.md:

Requirement	Plan Status	Notes
Backend validation	PASS	Input validation mentioned
Authorization checks	PASS	Family membership verification
No hardcoded secrets	PASS	Uses env vars (implicit)
Silent denial for queries	WARN	Not explicitly stated - should add
Explicit errors for mutations	PASS	Error codes returned

Review 4: UX Best Practices

Checking against docs/general/claude/ux-patterns.md:

Requirement	Plan Status	Notes
Loading states	PASS	Mentioned for creation
Error feedback	PASS	Error codes mapped to messages
Confirmation for destructive actions	WARN	Delete group needs confirmation dialog
Optimistic UI	PASS	Mentioned for drag-drop
Empty states	WARN	No mention of empty group state

Review 5: React Native Best Practices

Checking against docs/general/claude/react-native.md:

Requirement	Plan Status	Notes
Data-driven UI	PASS	Plans fetched from API
No hardcoded plans	PASS	Explicitly addressed
SafeAreaView edges	WARN	Not explicitly mentioned

The Irony

To use Claude Code effectively, you need to understand how Claude Code works.

Vibe coding is not "just vibe."

It requires engineering discipline.

> *We are just shifting from software engineering to context and workflow engineering.*

Let's Connect

Are you experiencing Claude Code performance issues?
What's your workaround?

> Follow for more AI coding insights
> Repost to help others
> Comment with your experience

#ClaudeCode #VibeCoding #AI #SoftwareEngineering
#DeveloperExperience

Khur Boon Kgim
AI-Powered Dad and Technopreneur