100 Winged Words of Developers (and some Managers) — 2025 Edition

A century of excuses, confessions, and corporate poetry from the codebase we call life.

4 min read · Just now

Gunnar Östberg

By **Gunnar Östberg**

Introduction

Developers have their own language. It's one part English, one part Stack Overflow, and one part existential crisis.

After forty years in tech — from assembly code and rotary phones to AI-driven everything — I've learned one universal truth:

> No matter the technology, the excuses stay the same.

What follows is a field guide to what we *say* when systems fail, deadlines crumble, or agile ceremonies mutate into standstill meetings.
Some of these phrases are older than UNIX; others were born last month in a Slack thread full of panic and caffeine.

Grab your favorite debugging beverage (Coke or coffee?) and enjoy this anthology of the world's most relatable developer quotes.

The full 100 are listed in the **Appendix.**

Part A — The classics

“Weird.”
The shortest bug report in history.

“But it worked before!”
Said moments before it doesn’t.

“Just a few small things left to fix.”
Developer time ranges from 15 minutes to 3 days.

“How could that even happen?”
A question rarely answered honestly.

“Must be a hardware issue.”
Translation: I have no idea what’s going on.

“You must be running an old version.”
Timeless scapegoat.

“It’s just a cosmetic issue.”
A phrase that always ends in “…and now nothing works.”

“This can’t possibly affect that.”
The most dangerous sentence in programming.

“Aside from not working, how do you like it?”
Design always comes first.

“We’re just making sure it’s stable.”
Code for “It’s crashing right now.”

“It actually works fine. It just doesn’t look like it.”
Philosophy meets debugging.

Part B — Modern excuses (2000–2025)

“Works on my machine.”
Still undefeated after 25 years.

“The API changed — again.”
Every integration developer’s cry for help.

“Probably the cache.”
Because it’s always the cache.

“Just waiting for the CI/CD pipeline.”
A modern lullaby.

“Let’s roll back and pretend it never happened.”
Corporate damage control in one line.

“The AI model hallucinated, not my code.”
2025’s version of “Not my fault.”

“It’s only broken in production.”
Otherwise known as Friday afternoon.

“It works in Docker.”
These words are engraved on the gates of hell.

“We just need to reindex the vector DB.”
Because even data needs therapy.

“Must be a version mismatch.”
Always was, always will be.

“I swear it worked yesterday.”
Last words before disaster.

Part C — Consultant poetry

“We just need to align our backlog with the roadmap.”
Modern haiku for people who haven’t coded in years.

“Let me check Confluence.”
Famous last words before a two-hour search.

“We have CI/CD, but I deploy manually.”
The dev equivalent of “I trust the autopilot, but I still hold the wheel.”

“Stack Overflow was down, so I had to think for myself.”
Courage under fire.

“It works, but I don’t know why.”
Faith-based engineering.

“It doesn’t work, but I know why.”
Pride in understanding failure.

“We probably need to bring in DevOps.”
The consultant’s 911 call.

“We really should automate this.”
The eternal sigh of modern labor.

Part D — Management mantras

“We need to be more Agile.”
Usually said during a three-hour PowerPoint.

“How do we maximize business value?”
Translation: We have no idea what the customer wants.

“Let’s push that to next quarter.”
Corporate procrastination, rebranded.

“We need a holistic solution.”
Nobody agrees on what the problem is.

“We have a POC for that.”
A demo that will never reach production.

“We must deliver customer value.”
And by “customer,” they mean “stakeholder with a budget.”

“Can we reprioritize the backlog?”
Also known as “Let’s move this pain somewhere else.”

“We need to drive this change.”
Because saying it feels like progress.

Closing thoughts

Every era of tech has its own poetry.

In the ’80s, we blamed hardware. In the ’90s, the user. In the 2000s, the network.

Today, we blame the AI.

Behind every excuse hides something noble: curiosity, optimism, and the eternal belief that the *next* build will finally work.

So when someone says, *“We just need to clear the cache,”* don’t roll your eyes. Smile. You’re witnessing the oldest ritual in software development — the art of hopeful denial.

Appendix — The Complete Collection

No	Quote	Comment
1	Weird.	Shortest bug report ever.
2	I have no idea what that is.	After the code has run fine for years.
3	But it worked before!	Uttered before it stops working.
4	Just a few small things left to fix.	A “15-minute” fix that takes days.
5	How could that even happen?	Asked, never answered.
6	Must be a hardware issue.	When logic has collapsed.
7	Maybe they changed the release.	Classic blame-shifting.
8	You must’ve done something wrong.	User ≠ system.
9	There must be something wrong with the input.	Standard defense.
10	But I didn’t change anything in that module.	Programmer’s oath.
11	Yeah, it’ll be ready before then.	The optimistic lie.
12	You must be running an old version.	Comforting excuse.
13	It’s just a cosmetic issue.	Affects looks … and logic.
14	I’m almost done.	See #4.
15	Sure, once I add the final tweaks.	Each “final” adds five more.
16	Won’t take any time at all.	Second most dangerous phrase.
17	We’re just making sure it’s stable.	Means “it’s crashing.”
18	Probably just a coincidence.	Magical thinking.
19	You can’t test everything.	So we tested nothing.
20	This can’t possibly affect that.	The deadliest phrase.
21	But I thought I fixed that.	Mental rollback.
22	It’s included — just not tested.	Half-working promise.
23	It actually works fine, it just doesn’t look like it.	Philosophy meets code.
24	Aside from not working, how do you like it?	Design > function.
25	Works on my machine.	Always accurate, never helpful.
26	It’s not a bug, it’s a feature.	Ancient defense.
27	Probably the cache.	99 % of problems.
28	Let’s try clearing the cache.	99 % of solutions.
29	Just waiting for the CI/CD pipeline.	Modern patience.
30	Must be a version mismatch.	Forever.
31	The API changed again.	Of course it did.
32	We need a clean rebuild.	Because hope.
33	Works in dev, fails in prod.	Classic.
34	Not sure why it works, but I’m not touching it.	If it ain’t broke …
35	Let’s roll back and pretend it never happened.	Corporate memory wipe.
36	I swear it worked yesterday.	Famous last words.
37	The logs show nothing.	The logs lie.
38	It’s a dependency issue.	Always someone else’s code.
39	I’ll fix it after the sprint demo.	Never fixed.
40	The bug disappeared when I added a print statement.	Observer effect.
41	We really should add tests for that.	Someday.
42	That’s an edge case.	Edge = user.
43	It’s an intermittent issue.	Unfixable.
44	It’s only broken in production.	So who cares?
45	Let’s restart the container.	Universal cure.
46	That’s not my code.	Blameless culture ™.

47	Must be the network.	Blame the wires.
48	Let's open a Jira ticket.	Problem deferred.
49	Out of scope for this sprint.	Never to be done.
50	Framework limitation.	Sacred excuse.
51	We need to upgrade the library.	Because release notes.
52	It's the cloud provider's fault.	Always the cloud.
53	The LLM misunderstood the prompt.	New age defense.
54	We need to reindex the vector database.	AI meets ritual.
55	That's a data-drift issue.	Trendy way to say bug.
56	The AI model hallucinated, not my code.	Modern classic.
57	Let's push it to staging and see what happens.	YOLO deployment.
58	Works in Docker.	Sure it does.
59	It's a race condition in async code.	Ghost in the machine.
60	The microservice isn't responding — again.	Of course not.
61	We need to align our backlog with the roadmap.	Buzzword alignment.
62	There's probably a Jira ticket for that.	There always is.
63	Let me check Confluence.	Lost forever.
64	We should've had an architect in that meeting.	Post-mortem wisdom.
65	We'll handle that next sprint.	Infinite loop.
66	It's a known issue.	Therefore, ignored.
67	We have a workaround.	Half solution.
68	We need to refactor a bit first.	And never finish.
69	We really should automate this.	See #68.
70	Not a bug, just an edge case.	Comforting lie.
71	It works in the cloud.	Somewhere.
72	We have CI/CD, but I deploy manually.	Trust issues.
73	AI assistants should handle this soon.	Wishful thinking.
74	Just a tiny syntax error — that breaks everything.	Butterfly effect.
75	I pushed the code but forgot to commit.	Temporal paradox.
76	I think Git froze.	Blame Git.
77	Stack Overflow was down, so I had to think for myself.	Heroic act.
78	It works, but I don't know why.	Faith.
79	It doesn't work, but I know why.	Control.
80	We need to bring in DevOps.	911.
81	We need to be more Agile.	Endless mantra.
82	We need some synergy here.	Buzzword alchemy.
83	How do we maximize business value?	Translation: Help.
84	We have to innovate and disrupt the market.	MBA gospel.
85	We need a growth mindset.	Poster material.
86	Can we deliver that in the MVP?	Scope denial.
87	We need to streamline the processes.	More meetings.
88	That's a quick win we need to capture.	Mythical win.
89	Can we reprioritize the backlog?	See #61.
90	We need a deep dive on this.	Meaningless depth.
91	It's essential we secure delivery.	Empty assurance.
92	We need to decouple the architecture.	Architectural zen.
93	We need to onboard a consultant.	Budget burn.
94	We need a holistic solution.	No one knows what it means.

94	We need a holistic solution.	No one knows what it means.
95	We need to build bridges between teams.	Forced friendship.
96	We need to iterate on this solution.	Perpetual motion.
97	Can we push that to next quarter?	Delay mode.
98	Let's park that for now.	Forever.
99	We have a POC for that.	Never finished.
100	Do we have an exit strategy?	Never.

About the Author

Gunnar Östberg is a Swedish engineer, management & IT consultant, and lifelong student of human systems — technical and otherwise.
He writes about technology, AI, and the absurd beauty of modern work.

Programming | Humor | Software Development | Developer Life

Edit profile

Written by Gunnar Östberg

180 followers · 127 following

CEO Business Clarity. Management Consulting, Product Management, Business Analysis, Business Architecture, Information Systems, Information Technology.

No responses yet

Gunnar Östberg

What are your thoughts?