Passing by Default: A Guided Vibe-Coding Process, Audited

A case study on what a working test suite didn't catch.

An anonymized client case study • 2026

The Situation

A client came to us running a live RV classifieds marketplace, built mostly through AI-assisted ("vibe-coded") development. The site's senior developer put real guardrails around it: strict type-checking from day one, real unit tests for at least one core module, more process discipline than most AI-assisted builds we see.

We ran a full production audit across code, infrastructure, and the delivery pipeline: the same eight-discipline pass we run on every engagement, covering security, code quality, test coverage, dependencies, performance, accessibility, SEO, and the data layer.

What We Found

The test suite had never run a test.

From the outside, the project's automated tests looked exactly like a working safety net: a test command, checked into continuous integration, reporting success on every run. When we ran it directly against the actual test file, it worked: a real set of tests, passing. But the command wired into CI matched files by a pattern that never matched anything else in the project. It had been reporting "0 tests, 0 failures" as a pass since the day it was written. The team never noticed, because a build that reports success and a build that ran nothing look identical on a dashboard.

That gap mattered more than a code-quality footnote. Our audit surfaced multiple critical-severity findings elsewhere in the codebase: exactly the kind a real test suite exists to catch. They'd been shipping the whole time behind a CI check that always showed green.

Two more findings, from different disciplines, made the same point again.

On the accessibility side, one full step of the site's core conversion flow, uploading a photo when creating a listing, was unreachable by keyboard. A user tabbing through the form skipped straight past it. It worked with a mouse. It didn't work for anyone navigating by keyboard alone, screen reader users

included.

On the SEO side, the site served search crawlers a near-empty page while showing real visitors the full one. The gap was wide enough that most major crawlers, including several AI-crawler agents, indexed almost nothing about the site's actual listings.

None of the three threw an error or showed up in a manual click-through. All three stayed invisible to exactly the kind of testing a busy team, or a green CI check, relies on.

Why It Happened

This came from a process that looked complete from the inside: type safety, tests, review discipline. What it lacked was the one check none of those controls can run on themselves, an outside read of the finished system, done by someone with no stake in the outcome. The team trusted the test suite and never went back to confirm it was running the right files.

The Takeaway

Guardrails reduce risk. They don't remove the need for someone outside the process to confirm the guardrails are still doing their job. We find the riskiest gaps most often in projects disciplined enough to believe their process already had this covered.

This case study is anonymized at the client's request. Findings are described at the level of impact and category; reproduction-level detail is omitted.