AUDIT

FIVE MISTAKES · FIVE FIXES · @DANIELWELSH_ROUTIQ

Built an app? Mistakes 101

If you built an app with AI, it probably works. And it's probably making a few of these. Nothing here is theory: **every mistake below broke my own app first.** Each one comes with what it looks like, the fix, and a 30-second check so you know you're actually safe.

01

GitHub, barely used.

LEARN IT!

One commit called **"updates"**, or no repo at all. Git is the undo button for AI mistakes: without it, one bad session can erase your working app and there's no yesterday to go back to.

THE FIX

- Create the repo **before** the first prompt
- Commit every working change: small commits, real messages
- Branch before experiments, merge when they work

The 30-second check

Run `git log --oneline`. You should see many small commits, not one giant "updates".

02

Secrets in the code.

USE ENVIRONMENT VARIABLES

API keys pasted straight into your files. **Anyone with dev tools can read them**. Scanners find hundreds of thousands of live keys on GitHub every month.

THE FIX

- `.env` + `.gitignore` before the first secret exists
- `.env.example` documents the shape: names, never values
- Secret keys stay server-side, never in the browser bundle

The 30-second check

Run `git ls-files | grep .env`. If it prints anything except `.env.example`, rotate those keys today.

03

Giant files?

BREAK THEM DOWN

The 2,000-line **page.tsx**. Claude keeps appending because adding in-place is easiest, until it can't safely edit its own file, and neither can you.

THE FIX

- Folder skeleton before features: `app/` `components/` `lib/` (one file per service)
- Set a 600-line budget per file. A rule, not a hope
- Ask Claude to split one extraction at a time, tests between

The 30-second check

Sort your files by length. Anything over **600 lines** is where your next bug is hiding.

04

Your database is probably public.

RLS ON

A 2026 audit found **88% of vibe-coded apps** had row-level security switched off. Any user could read every row. Nothing visibly breaks, so nobody notices until it's an incident.

THE FIX

- Turn RLS on the day you create each table, not "later"
- Write an ownership policy per table (`auth.uid() = user_id`)
- Ask the AI: **"show me the exact rule that enforces ownership"**. If it can't point to it, it doesn't exist

The 30-second check

The two-account test: sign up twice, try to read account A's data from account B. If you can, so can everyone.

05

Design is never an afterthought!

ONE PAGE FIRST

Two flavours of the same mistake: your app has **the same purple-gradient look as 80% of AI apps**, and screens got built before anyone decided what the app is, so the data model shifts weekly and everything breaks.

THE FIX

- Write one page first: what it does, who it's for, what "done" means
- Give Claude a design guide + screenshots of apps whose taste you want
- Data model before screens. Migrations are the truth

The 30-second check

Can a stranger say what your app does in one sentence? If not, the page isn't written yet.

WHY THIS MATTERS · THE RECEIPTS

88%
of vibe-coded apps had their database publicly readable. RLS disabled (2026 audit)

100k+
live API keys found on public GitHub every month by secret scanners

11%
of 20,000+ indie app launches exposed their database credentials in the frontend

2,535
lines in the single worst file of my own v1. Every rule here broke my app first

Fixing these five puts you ahead of most apps that ship. I'm rebuilding my own app, the one that answers the phone at real medical clinics, applying all five in public. Follow along: **@danielwelsh_routiq** on Instagram.

BONUS

THE BONUS ROUND · FIVE MORE

Fixed those five? Five more.

The first five keep your app safe. These five keep it alive once real people use it.

06

Testing by clicking around.

WRITE REAL TESTS

You change something, click through the app, and hope. That is a ritual, not a test suite. Ask Claude to write a test before each fix, and run them automatically on every push.

CHECK · Break something on purpose. Does a robot catch it before you do?

07

No idea when it breaks.

ADD ERROR TRACKING

Your app fails silently and you find out from a user screenshot days later. Wire an error tracker (Sentry has a free tier) the day you deploy, with alerts that reach your phone.

CHECK · Throw a test error. Did anything tell you?

08

The AI picks your dependencies.

ONE LIBRARY PER JOB

Three date libraries, two UI kits, npm install everything Claude suggests. Question every new install, keep one library per job, and prefer the platform when it is enough.

CHECK · Open package.json. Can you say why each line is there?

09

Testing in production.

USE PREVIEW DEPLOYS

Every change goes straight to the app your users are on. Preview deploys give every branch its own private URL (Vercel and Netlify do this free). Test there, then promote.

CHECK · Can you try a risky change with zero users seeing it?

10

Ignoring costs until the bill.

SET SPENDING CAPS

Pay-as-you-go keys everywhere, then a loop calls a paid API all night. Set spending caps and usage alerts on every paid service, and know what one active user costs you.

CHECK · Can you name your cost per user? If not, the bill will name it for you.

THE BOOKSHELF · WHERE THE FIXES COME FROM

The Pragmatic Programmer

HUNT & THOMAS

The one-page-first mindset. Decide what you are building before the AI does.

Clean Architecture

ROBERT C. MARTIN

One core, thin edges. Keep vendors at the door so you can swap them.

Don't Make Me Think

STEVE KRUG

The stranger test. Design that needs no manual, measured in seconds.

Continuous Delivery

HUMBLE & FARLEY

Gates before features. Why automatic checks make you faster, not slower.

Designing Data-Intensive Applications

MARTIN KLEPPMANN

Receipts and idempotency. Why nothing important should be able to happen twice.

Refactoring

MARTIN FOWLER

How to split the giant file safely, one small step at a time.