

Looks finished. *Isn't.*

12 ways AI breaks your UI,
and how to catch them.

For designers who build with Claude, Cursor, Lovable, v0, Figma Make
or any other AI tool.

AI builds a screen in two minutes. *Reviewing it is your job.*

AI-built interfaces fail in surprisingly predictable ways. After a year of building real products with AI, and twelve years of designing products where "looks finished" was never enough, I keep finding the same twelve patterns. This guide names them, so you can spot them fast and fix them with one clear instruction.

It works with any AI tool. The fixes are plain-language prompts, and every pattern points to the standard behind it, for the moment a stakeholder asks "why?".

EVERY PATTERN PAGE HAS

What you see

How the problem shows up in a screen or a demo.

Catch it in 30 seconds

A quick test you can run without reading code.

Why AI does it

The reason behind it, so you can prevent it next time.

The one-line fix

A prompt you can paste into any AI tool.

HOW TO USE IT

Read it once. Then keep page 17, the 5-minute review, next to you whenever AI builds something for you.

AI is a fast junior who never says *"I don't know"*.

Give it a screenshot and a vibe, and it fills every gap with something plausible. A plausible component. A plausible error message. A plausible statistic.

Plausible is not *correct*.

That's not a reason to stop using AI. It's a reason to change how you work with it: decide more before it builds, and check more before you accept.

THREE QUESTIONS BEFORE YOU ACCEPT ANY SCREEN

- 1 Is it ours?** Our components, our tokens, our copy.
- 2 Is it real?** Real buttons, labels and focus, not pictures of them.
- 3 Does it behave?** Every state, every error, with real content.

CONTENTS

Twelve patterns, *four questions.*

IS IT OURS?

01	Components that only look like yours	p.05
02	Hardcoded values everywhere	p.06
03	Copy, data and features nobody approved	p.07

IS IT REAL?

04	Pictures of controls, not controls	p.08
05	Placeholder text as the label	p.09
06	Focus that disappears	p.10

DOES IT BEHAVE?

07	Only the happy path	p.11
08	Lost input and double submits	p.12
09	Breaks with real content	p.13

DOES IT COMMUNICATE?

10	The "AI-ish" look	p.14
11	Color as the only signal	p.15
12	Errors that don't help	p.16

+	The 5-minute review, feedback that works, and how to prevent all twelve	p.17
---	---	------

Components that only *look* like yours

WHAT YOU SEE

A button, card or dropdown that looks almost right, but isn't from your design system. Slightly different radius, padding or hover. It passes a quick glance and fails the next design review.

WHY AI DOES IT

It recreates what it sees instead of looking up what exists. Without a list of your real components, a convincing copy is the easiest path.

CATCH IT IN 30 SECONDS

Hover, focus and open every control. Compare one of them side by side with the real component in your library or Storybook.

BEFORE

Custom dropdown, close to yours

AFTER

Select from your design system

THE ONE-LINE FIX

"Use only components from our design system. If one is missing, use the closest existing one and mark it `TODO(design-gap)`. List every component you used."

STANDARD | ISO 9241-110 | conformity with user expectations | ISO 9241-112 | consistency

Hardcoded values *everywhere*

WHAT YOU SEE

Colors, spacing and font sizes typed in as raw numbers. It looks fine today and breaks the day you change a token, switch to dark mode or add a brand theme.

WHY AI DOES IT

Raw values are the shortest way to match a screenshot. Tokens require knowing your system, and nobody told it your system.

CATCH IT IN 30 SECONDS

Switch to dark mode or another theme, if you have one. Anything that doesn't change with it is hardcoded. Or ask the tool to list every raw color value in the screen.

BEFORE

```
color: #3b82f6; padding: 13px
```

AFTER

```
var(--color-primary); var(--space-3)
```

THE ONE-LINE FIX

"Replace every hardcoded color, spacing and font size in this screen with our design tokens. List any value that has no matching token."

STANDARD | ISO 9241-112 | consistency

Copy, data and features *nobody approved*

WHAT YOU SEE

A friendly hero headline, three made-up statistics, a testimonial from "Sarah, Product Lead" and a settings tab you never asked for. It looks generous. It's invented.

WHY AI DOES IT

Empty space reads as unfinished, so it fills it. Generating text and features costs it nothing, and it can't tell which content carries legal or support consequences.

CATCH IT IN 30 SECONDS

Trace every visible element back to your brief or spec. Anything you can't trace was invented.

BEFORE

"Trusted by 10,000+ teams"

AFTER

[COPY: what this text must say]

THE ONE-LINE FIX

"Use only the copy and features I gave you. Where text is missing, insert [COPY: what it must say] instead of writing it. Remove anything I didn't ask for."

STANDARD | ISO 9241-110 | suitability for the user's tasks | ISO 9241-112 | conciseness

Pictures of controls, *not controls*

WHAT YOU SEE

A "button" that's a styled div. A "menu" that's a stack of links. A "table" made of boxes. It looks right and works with a mouse. Keyboard and screen reader users can't use it.

WHY AI DOES IT

It builds from appearance. A div styled like a button looks identical in a screenshot, and nothing in a screenshot tells it what an element is.

CATCH IT IN 30 SECONDS

Press Tab through the screen without a mouse. If a button or link can't be reached or pressed with Enter or Space, it's a picture.

BEFORE

```
<div onclick>Save</div>
```

AFTER

```
<button>Save</button>
```

THE ONE-LINE FIX

"Use real HTML elements: button for actions, a link for navigation, nav with a list for menus, table with headers for data. No click handlers on divs."

STANDARD | WCAG 2.2 | 4.1.2 Name, Role, Value | 1.3.1 Info and Relationships | 2.1.1 Keyboard

Placeholder text *as the label*

WHAT YOU SEE

Clean, minimal forms where the field says "Email address" in grey, and the hint disappears the moment someone starts typing. Halfway through, nobody remembers which field was which.

WHY AI DOES IT

Placeholder-only fields look tidy in a screenshot, and tidy is what it optimises for. The problem only shows up while typing, which a screenshot never shows.

CATCH IT IN 30 SECONDS

Type into every field. If the field's name disappears, there's no real label.

BEFORE

```
placeholder="Email address"
```

AFTER

```
<label>Email address</label>  
+ field
```

THE ONE-LINE FIX

"Give every field a visible label that stays visible and is linked to the field. Keep placeholders only as optional hints."

STANDARD | WCAG 2.2 | 3.3.2 Labels or Instructions | 1.3.1 Info and Relationships

Focus that *disappears*

WHAT YOU SEE

Keyboard users press Tab and lose track of where they are. The focus outline has been removed for looks, or it's hidden behind a sticky header or a cookie banner. Sometimes they get stuck in a dialog.

WHY AI DOES IT

Default focus rings look "unpolished", so they get removed. Focus never appears in a screenshot, so nothing reminds it to put them back.

CATCH IT IN 30 SECONDS

Unplug your mouse and complete the whole flow. You must always see where you are, never get trapped, and be able to close dialogs with Escape.

BEFORE

outline: none

AFTER

Visible focus, 3:1 contrast, never hidden

THE ONE-LINE FIX

"Make the whole flow work with the keyboard alone, in a logical order, with a clearly visible focus indicator that is never hidden behind sticky elements."

STANDARD | WCAG 2.2 | 2.4.7 Focus Visible | 2.4.11 Focus Not Obscured | 2.1.2 No Keyboard Trap

Only the *happy path*

WHAT YOU SEE

The demo is perfect: data loaded, form valid, everything succeeds. Then the first real user hits an empty list, a slow network or a failed request, and gets a blank screen.

WHY AI DOES IT

You showed it one state, so it built one state. Empty, loading, error and no-permission states don't exist unless someone asks for them.

CATCH IT IN 30 SECONDS

Ask: "How do I trigger the empty, loading and error states?" If the answer is vague, they don't exist.

BEFORE

One state: everything works

AFTER

Empty, loading, error, success, no permission

THE ONE-LINE FIX

"Build every state of this screen: empty, loading, error, success, disabled and no permission. Tell me how to trigger each one."

STANDARD | ISO 9241-110 | self-descriptiveness, use error robustness | WCAG 2.2 | 4.1.3 Status Messages

Lost input and *double submits*

WHAT YOU SEE

Someone fills in a long form, one field fails validation, and the form clears. Or they click "Pay" twice on a slow connection and pay twice. Users remember this forever.

WHY AI DOES IT

Preserving input and preventing duplicates is invisible work. It doesn't show in a demo, so it doesn't get built unless it's a requirement.

CATCH IT IN 30 SECONDS

Make one field invalid and submit. Then double-click the submit button on a throttled connection. Then go back and forward.

BEFORE

Error clears the whole form

AFTER

Input kept, submit locked while sending

THE ONE-LINE FIX

"Keep everything the user typed after errors, failed requests and navigation. Disable the submit button while a request is in flight, so it can't be sent twice."

STANDARD | ISO 9241-110 | use error robustness, controllability | WCAG 2.2 | 3.3.4 Error Prevention | 3.3.7 Redundant Entry

Breaks with *real content*

WHAT YOU SEE

Built with "John Doe" and short English labels. Real users have long names, German translations, big numbers and phones in their hand. Buttons overflow, text gets cut, layouts break at 200% zoom.

WHY AI DOES IT

Demo data is short and tidy. It optimises for the example it was given, not for the range of content real life produces.

CATCH IT IN 30 SECONDS

Paste a 60-character name and a German translation, narrow the window to 320 pixels, then zoom to 200%.

BEFORE

"John Doe", English only

AFTER

Long names, translations, 320 px, 200% zoom

THE ONE-LINE FIX

"Test this screen with a 60-character name, German text, 320 px width and 200% zoom. Fix everything that breaks, overflows or gets cut off."

STANDARD | WCAG 2.2 | 1.4.10 Reflow | 1.4.4 Resize Text | 1.4.12 Text Spacing

The “*AI-ish*” look

WHAT YOU SEE

Purple gradients, soft glows, glass cards, a thick accent stripe on the left of every box, sparkle emoji and a hero line that could belong to any product. You can spot it from across the room.

WHY AI DOES IT

Without your brand and your decisions, it falls back to the average of what it has seen. Average is recognisable, and now it reads as “made by AI”.

CATCH IT IN 30 SECONDS

Remove one decorative element at a time. If the screen gets clearer, it didn't belong there.

BEFORE

Gradient, glow, ✨ emoji

AFTER

One accent, hierarchy through type

THE ONE-LINE FIX

“Remove everything decorative I didn't ask for: gradients, glows, accent stripes, emoji. Use one accent color and build hierarchy through type and spacing.”

STANDARD | ISO 9241-112 | freedom from distraction

Color as the *only signal*

WHAT YOU SEE

Errors shown only in red, status only as a colored dot, light grey text on white. It looks calm and modern. For many people it's unreadable or meaningless.

WHY AI DOES IT

Color is the quickest way to make a difference visible, and low contrast looks "elegant" in a screenshot. Contrast ratios aren't something it checks unless asked.

CATCH IT IN 30 SECONDS

Take a screenshot and make it greyscale. Can you still tell errors, status and actions apart? Then check text contrast with any contrast checker.

BEFORE

Red border only

AFTER

Red border + icon + "Enter a valid email"

THE ONE-LINE FIX

"Check contrast: at least 4.5:1 for text and 3:1 for icons, borders and focus. Never use color as the only signal; add text or an icon."

STANDARD | WCAG 2.2 | 1.4.1 Use of Color | 1.4.3 Contrast (Minimum) | 1.4.11 Non-text Contrast

Errors that *don't help*

WHAT YOU SEE

"Something went wrong." "Oops!" "Error 422." Users know something failed. They don't know what, and they don't know what to do next, so they call support or leave.

WHY AI DOES IT

Generic messages fit every case, so they're the safe default. Writing a helpful one needs knowledge of the rule behind it, which only you have.

CATCH IT IN 30 SECONDS

Trigger each error. Does the message say what happened, where, and how to fix it, in the user's words?

BEFORE

"Oops! Something went wrong."

AFTER

"Enter a date after today."

THE ONE-LINE FIX

"Rewrite every error so it says what happened and how to fix it, in plain words, next to the field it's about. No codes, no blame, no "Oops"."

STANDARD | WCAG 2.2 | 3.3.1 Error Identification | 3.3.3 Error Suggestion | ISO 9241-110 | use error robustness

Twelve checks before you say *"looks good"*.

Run it after every AI build step. Every unticked box becomes one instruction back to the AI.

IS IT OURS?

- ☐ Every component comes from our design system
- ☐ No raw colors or spacing; it survives a theme switch
- ☐ Every word and feature traces back to the brief or spec

IS IT REAL?

- ☐ Tab reaches and activates every control
- ☐ Every field keeps a visible label while typing
- ☐ Focus is always visible; Escape closes dialogs

DOES IT BEHAVE?

- ☐ Empty, loading, error and success states exist
- ☐ Input survives errors; submit can't fire twice
- ☐ Long names, translations, 320 px and 200% zoom work

DOES IT COMMUNICATE?

- ☐ Nothing decorative that nobody asked for
- ☐ Readable in greyscale; text contrast at least 4.5:1
- ☐ Every error says what happened and how to fix it

All twelve ticked? Then it doesn't just look finished.

p.05 to p.16

"Make it better" is not *feedback*.

AI does exactly what you say, so vague feedback gets a vague result, plus three new problems. One finding becomes one instruction: name the element, the rule and the result.

NOT THIS

"The form looks off. Fix it."

THIS

"The address step uses a styled div as a dropdown. Replace it with Select from our design system and keep the error message we defined for this field."

THREE HABITS

One finding per message. Several at once and something gets lost.

Name the rule, not your taste. "Labels must stay visible" lasts longer than "I don't like it".

Save before each fix. A commit or a version lets you undo a fix that made things worse.

Catching is good. *Preventing is cheaper.*

Most of these twelve patterns disappear when the AI knows three things before it builds: what exists in your design system, how the flow must behave in every state, and the rules a senior designer would give a new team member on day one.

SPEC FIRST

States, edge cases, errors and exact copy, written before anything is built.

RULES

Read by the AI at the start of every session, so you don't repeat yourself.

REVIEW

After every step, with findings you can paste back as instructions.

That's the process I packed into **Spec to Ship**, a kit for designers who build with Claude Code. Built on ISO 9241-210 and WCAG 2.2.

SPEC TO SHIP LITE | FREE

The 7-step process, the rules Claude reads and the senior review checklist.

uxpapa.gumroad.com/l/spec-to-ship-lite

SPEC TO SHIP PRO

Adds four Claude Code skills, three complete example specs, the WCAG 2.2 AA checklist and a 19-page field guide.

uxpapa.gumroad.com/l/spec-to-ship-pro

Claude does the typing. *You* make the call.

I'm Shiraz Tumasyan, a senior UX consultant with more than twelve years in products where "looks finished" was never enough: the Porsche 911 cockpit, MAN trucks (Red Dot Award), Siemens industrial software, and national e-government services. I hold an M.Sc. in Human Engineering and coach UX designers.

Today I build products with AI every week, and I teach designers to do it without losing what makes their work senior.

WEBSITE

tumasyan.com

INSTAGRAM

[@uxpapa](https://www.instagram.com/uxpapa)

LINKEDIN

[Shiraz Tumasyan](https://www.linkedin.com/in/shiraztumasyan)

TEAM WORKSHOPS AND QUESTIONS

info@tumasyan.com