VIBE CODING · BEGINNER · 12 MIN READ

Debugging with AI: when it breaks

Read the error, reproduce it, give the AI the right clues, and undo safely. The skill that separates finishing from giving up.

Pass the check (4 of 5) online to earn the “Bug Hunter” badge:
https://digitalbuilders757.com/university/study/debugging-with-ai

Everything breaks. That's normal.

Every builder, beginner or senior, spends a big part of their time fixing things. Vibe coding doesn't remove bugs; it makes them faster to fix, if you hand the AI good evidence instead of 'it doesn't work'.

Debugging is detective work: find out exactly what happens, compare it to what should happen, then narrow down where the difference starts.

Step 1: Reproduce it

Write down the exact steps that cause the problem: 'Sign in as a new user, click Save on an empty form, the page goes blank.' If you can make it happen on purpose, you can prove it's fixed. If it only happens sometimes, note what's different when it does (phone vs laptop, signed in vs out, first visit vs later).

Step 2: Find the error message

The computer usually tells you what went wrong; it just whispers. Look in these places:

- On the page: red text, a toast, or a blank screen with a message.
- The browser console: right-click, Inspect, then the Console tab. Red lines are errors.
- The Network tab: a failed request shows a status like 401 (not signed in), 403 (not allowed), 404 (not found) or 500 (server error).
- Build logs and runtime logs on your host (Vercel, your builder's log panel).
- Your database dashboard's logs (Supabase has logs for the API, auth and database).

Step 3: Brief the AI like a detective

A great debugging prompt has five parts:

- What you did (the reproduce steps).
- What you expected to happen.
- What actually happened.

- The exact error text, copied, not paraphrased, plus where you found it.
- What changed recently ('it worked before I added the photo upload').

Step 4: Fix small, test, and keep an exit

Ask the AI to explain the cause before it changes anything, then to make the smallest fix. Test your reproduce steps again. If a fix makes things worse, don't stack fixes on fixes: go back to your last working version (a Git commit or your builder's history) and try a different approach.

Stuck in a loop where the AI keeps 'fixing' and nothing changes? Start a fresh chat with a short summary and the exact error, or ask it to add logging so you both can see what's really happening.

Key terms

Bug
A mistake that makes software behave wrongly.

Debugging
Finding and fixing bugs, often by reading the exact error message.

Error message
The computer's description of what went wrong. Copy it exactly when asking for help.

Browser console
The developer-tools panel (Inspect, then Console) that shows a page's errors and logs.

HTTP status code
A number a server sends back: 200 OK, 401 not signed in, 403 not allowed, 404 not found, 500 server error.

Reproduce (a bug)
Make a bug happen on purpose with exact steps, so you can prove a fix works.

Logs
Records of what happened during a build or while the app ran, including errors.

Revert
Undo a change by going back to an earlier working version, often a Git commit.

Try it: Break it on purpose

1. In any small project, ask the AI to deliberately introduce one bug and not tell you where.
2. Reproduce it, find the error in the console or logs, and write a five-part debugging prompt.
3. Fix it with the AI, then confirm your reproduce steps now pass.
4. Commit the working version with a clear message.

KNOWLEDGE CHECK

Debugging with AI: when it breaks

5 questions. Circle one answer for each; 4 or more correct passes. Answers are at the back.

1. What's the most useful thing to give an AI when asking it to fix a bug?

A. 'It doesn't work, fix it'
B. A long, polite message explaining how important the deadline is to you
C. The exact error text, steps to reproduce, and what you expected
D. Your password

2. Where do you usually find JavaScript errors from a web page?

A. The page's HTML source (View Source)
B. Your email inbox
C. The domain registrar
D. The browser console (Inspect, then Console)

3. A request fails with status 401. What does that usually mean?

A. Not signed in, or the sign-in isn't being sent
B. The page is too big to send, so the request was cut off
C. The server crashed while it was handling your request
D. Everything worked

4. The AI's last three fixes made the app worse. Best move?

A. Ask for a fourth fix on top, with more detail about what went wrong
B. Go back to the last working version
C. Delete the whole project and rebuild it from scratch with a new prompt
D. Ignore the bug for now and keep adding features

5. Why reproduce a bug before fixing it?

A. It's a formality
B. To make the bug worse
C. So you can prove the fix actually worked by repeating the same steps
D. Reproducing is only needed for bugs that paying customers have reported to you

Official sites

Chrome DevTools: Console overview: https://developer.chrome.com/docs/devtools/console
MDN: HTTP response status codes: https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status

Last reviewed 2026-11-25. AI tools change fast; check each tool's own site for current features, plans and prices.

ANSWER KEY

Debugging with AI: when it breaks

1. C: The exact error text, steps to reproduce, and what you expected
Exact errors and reproduce steps let the AI find the real cause instead of guessing.

2. D: The browser console (Inspect, then Console)
The console shows errors and logs from the page's code.

3. A: Not signed in, or the sign-in isn't being sent
401 means unauthenticated. 403 means signed in but not allowed; 404 not found; 500 a server error.

4. B: Go back to the last working version
Return to known-good (Git or builder history) instead of stacking fixes.

5. C: So you can prove the fix actually worked by repeating the same steps
If you can trigger it on purpose, you can confirm it's gone.