

FREE · SHIP SAFE CHECKLIST

5 holes AI leaves in your app. *Check yours.*



Five quick tests for the most common holes in vibe-coded apps. Only run them against apps you own.

01

Secret keys in the browser

1. Open your **live** site → DevTools (F12) → **Sources** → search all files (Ctrl+Shift+F, Mac: ⌘⇧F).
2. Search, one at a time: `sk-` · `sk_live` · `sk-ant` · `service_role` · `sb_secret` · `secret` · `Bearer` .
3. Then search a local production build:

```
npm run build
grep -rEn "sk-[A-Za-z0-9_-]{20,}|sk_live_|sk-ant_|service_role|sb_secret_" .next/static dist build
2>/dev/null
```

YOU PASS IF

Nothing found, or only keys designed to be public:
Supabase *anon* / *sb_publishable_*, Stripe *pk_*,
Firebase web config.

YOU FAIL IF

Any OpenAI / Anthropic key, Stripe *sk_* or *rk_*, Supabase
service_role / *sb_secret_*, or a database URL with a
password.

02

Your database has no locks

1. Supabase → SQL editor. Tables with RLS off (expect **zero rows**): `select tablename from pg_tables where schemaname = 'public' and rowsecurity = false;`
2. Policies that allow everything: `select tablename, policyname, cmd from pg_policies where schemaname = 'public' and (qual = 'true' or with_check = 'true');`
3. Views skip RLS unless created with `security_invoker`: list them with `select table_name from information_schema.views where table_schema = 'public';`
4. Be the stranger. Logged out, using only the key from your own site:

```
curl "https://YOUR-PROJECT.supabase.co/rest/v1/profiles?select=*" \
-H "apikey: YOUR_PUBLIC_KEY" -H "Authorization: Bearer YOUR_PUBLIC_KEY"
```

YOU PASS IF

No tables without RLS. Every `true` policy is read-only and
on purpose (a public catalogue). The stranger gets []
from private tables.

YOU FAIL IF

Any private row readable while logged out, or any row
writable by every signed-in user.

03

Logged in ≠ allowed

1. Create two accounts, A and B. Use a normal window for A and a private window for B.
2. As B, create something (a note, invoice or project) and copy its URL and ID.
3. As A, open B's URL. Then replay A's own API calls with B's ID: DevTools → Network → right-click a request → **Copy as fetch** → paste into the Console, swap the ID, press Enter.
4. Try **read, edit and delete**, plus list endpoints with filters like `?userId=` .

YOU PASS IF

Account A gets `404 Not found` for every one of B's
records, on every method.

YOU FAIL IF

A sees, changes or deletes anything of B's, even partially
(a name, a count, a file URL).

08

Denial of wallet

1. Against **your own** app, fire 50 requests in a row at an expensive route:
2. Check every AI, email and SMS provider dashboard: is a **hard monthly spend limit** set?

```
for i in $(seq 1 50); do curl -s -o /dev/null -w "%{http_code} " \
-X POST https://YOUR-APP.com/api/chat -H "Content-Type: application/json" \
-d '{"message":"hi"}'; done
```

YOU PASS IF

You start seeing 429 within the limit you chose, and every paid provider has a hard cap.

YOU FAIL IF

50 × 200 (or 401 but the route still calls the provider before checking).

11

Packages that shouldn't exist

1. For every dependency in `package.json` you don't recognise:
2. Red flags: published in the last few weeks, almost no weekly downloads (see npmjs.com), no repository, a name one letter off a famous package.
3. Run `npm audit` and read the criticals (don't blindly `--force` them).

```
npm view PACKAGE-NAME time.created repository.url maintainers
```

YOU PASS IF

Every dependency is one you'd recognise or have checked, the lockfile is committed and there are no critical advisories.

YOU FAIL IF

Anything you can't explain: brand new, tiny, no repo, typo-like name.

Failed one? Fix it before you do anything else. The full kit has the copy-paste fix prompt and reference code for each of these, plus 10 more holes.

You checked 5.

There are 15.

The other ten are the ones that cost real money: fake payments, admin by mass assignment, Server Actions anyone can call, AI features that take orders from strangers.

01 Keys in the browser	02 Row-level security	03 Ownership checks (IDOR)
04 Mass assignment	05 UI-only checks	06 Server actions & routes
07 Unverified webhooks	08 Rate limits & spend caps	09 Storage & uploads
10 Secrets in Git	11 Hallucinated packages	12 Prompt injection
13 XSS in rendered content	14 SSRF in URL features	15 Headers, CORS & debug

THE FULL SHIP SAFE KIT · \$19

Stop reading about holes. Find yours in one command.

- **Ship Safe Scanner:** scans your code and live site, gives you a score and the exact file and line.
- **Supabase audit:** one SQL paste shows every RLS problem in your database.
- **Field guide:** 15 holes, each with a 5-minute test, a fix prompt, reference code and the mistakes AI makes fixing it.
- **Prompt pack + AI rules** for Cursor, Claude Code, Copilot, Windsurf and Lovable, so it stops writing holes.
- Lifetime updates. Use it on every app you build.

