The launch-safety checklist

Five things to check before you share an app built with Lovable, Bolt, Cursor, v0 or Replit. No coding background needed. Each step has a quick way to check it and a prompt you can paste into your builder to fix it. About 30 minutes.

1 Keep secret keys out of the code your visitors download

Everything your app sends to the browser can be read by anyone who opens it. Public keys (like a Supabase anon or publishable key) are made for that. Secret keys (OpenAI, Stripe sk_live, Supabase service_role) are not, and they belong on the server only.

HOW TO CHECK

- ☐ Open your live site, right-click, choose View Page Source, and search for sk_, service_role, and secret.
- ☐ Look at your environment variables. Anything starting with VITE_ or NEXT_PUBLIC_ is copied into the page, so no secret should have that prefix.
- ☐ If a secret key was ever in the page or on GitHub, rotate it (make a new one and delete the old one) in the provider's dashboard.

PROMPT TO PASTE INTO YOUR BUILDER

```
Find every API key used in this project. Move any secret key (anything that isn't meant to be public) to server-side code or an edge function, read it from a server-only environment variable, and make sure no secret key is shipped in the browser bundle.
```

2 Turn on row-level security for every database table

With Supabase (and similar tools), the browser talks to your database directly. Row-level security (RLS) is the set of rules that decides who can read or change each row. Without it, the public key in your page can read the whole table.

HOW TO CHECK

- ☐ In Supabase, open Table Editor and check that every table shows RLS enabled.
- ☐ Make sure no policy simply says true for reading or writing private data.
- ☐ Make two test accounts. Create something as account A, then log in as B and confirm B can't see it.

PROMPT TO PASTE INTO YOUR BUILDER

```
List every table in my database. Enable row-level security on each one and add policies so users can only read and change their own rows (match on auth.uid()). Show me any table or policy that still lets anyone read or write everything.
```

3 Require a login on every private API route

Buttons hidden in the interface don't protect anything. Anyone can call your API routes directly. Each route that reads or changes private data needs to check who is asking, on the server, every time.

HOW TO CHECK

- [ ] List your API routes or edge functions. Next to each, write who should be allowed to call it.
- [ ] Log out, then open a private API route's address in your browser. It should refuse.
- [ ] Check that admin-only actions verify the user's role on the server, not just in the page.

PROMPT TO PASTE INTO YOUR BUILDER

```
Go through every API route and server function in this project. For each one that reads or changes user data, check the logged-in user on the server and return 401 if there isn't one. Make sure users can only act on their own records, and admin actions check the role on the server.
```

4 Add the basic security headers

A few response headers tell browsers to block common tricks, like loading your site inside someone else's page or downgrading to plain HTTP. They take a few minutes to add and most hosts support them.

HOW TO CHECK

- [ ] Run a free scan at throttlebird.com, or check your site in securityheaders.com.
- [ ] Look for Strict-Transport-Security, Content-Security-Policy, X-Frame-Options (or frame-ancestors), X-Content-Type-Options and Referrer-Policy.
- [ ] Add them in your host's config (netlify.toml, vercel.json, or next.config) and re-check after you deploy.

PROMPT TO PASTE INTO YOUR BUILDER

```
Add these security headers to every response for this site: Strict-Transport-Security, X-Content-Type-Options: nosniff, Referrer-Policy: strict-origin-when-cross-origin, X-Frame-Options: DENY (or CSP frame-ancestors 'none'), and a Content-Security-Policy that allows only the domains this app actually uses. Use my hosting provider's config file.
```

5 Put rate limits on sign-up, login and anything that costs you money

Without a limit, one script can try thousands of passwords, create fake accounts, or run up your AI or email bill overnight. A simple per-IP or per-user limit on the busy routes keeps that in check.

HOW TO CHECK

- ☐ List the routes that send email, call a paid API (like OpenAI), or handle login and sign-up.
- ☐ Check that each one has a limit, for example 5 tries a minute per IP or per account.
- ☐ Set a spending cap or usage alert in each paid provider's dashboard as a backstop.

PROMPT TO PASTE INTO YOUR BUILDER

```
Add rate limiting to the login, sign-up, password reset, and any route that sends email
or calls a paid API. Limit by IP and by user (for example 5 requests per minute), return
429 when the limit is hit, and show a friendly message in the app.
```

Check the technical half for free. Paste your live URL at throttlebird.com for a plain-English list of what a stranger can see, with a fix prompt for each finding. It only reads your public site and takes under a minute.

Questions? Email support@throttlebird.com.
Jordan Reyes, Throttle Bird, throttlebird.com · Updated 2026-10-30