

# 27-Point Pre-Launch Security Checklist for Vibe Coders

Oct 31, 2026 · @shakir c

Run these 27 checks before you launch any app you built with AI. They cover the holes AI-generated code leaves open most often: leaked keys, open databases, missing server checks and unsafe defaults.

Tick each box as you go. If a check doesn't apply to your stack, mark it done and move on. At the end there's a ready-made Claude prompt that audits your whole codebase against this list.

By Shakir Cheppali · [@shakirc.ai](https://shakirc.ai)

## 1. Secrets and API keys (checks 1-4)

- ☐ **1. Hide your API keys.** Keep secret keys on the server in environment variables. Anything in frontend code or a `NEXT_PUBLIC_` variable is visible to everyone.
- ☐ **2. Keep .env out of Git.** Add `.env` to `.gitignore` and commit only a `.env.example` with empty values.
- ☐ **3. Scan your Git history for leaked secrets.** Deleting a key in a later commit doesn't remove it from history. Run gitleaks or trufflehog, or turn on GitHub secret scanning.
- ☐ **4. Revoke and rotate any exposed key.** If a key was ever committed or shipped to the browser, treat it as stolen: create a new one and disable the old one.

## 2. Authentication and access (checks 5-10)

- ☐ **5. Lock your database.** Turn on Row Level Security (or equivalent rules) on every table, so each user can read and write only their own rows.
- ☐ **6. Check auth on every API route.** Verify the session on the server for each request. Hiding a button in the UI is not protection.
- ☐ **7. Stop ID guessing.** Changing `/orders/18` to `/orders/22` must not show someone else's data. Check that the record belongs to the logged-in user.
- ☐ **8. Lock down admin routes.** Check the user's role on the server for every admin page and admin API, not just in the menu.
- ☐ **9. Store passwords safely.** Use a trusted auth provider, or hash passwords with bcrypt or argon2. Never store plain text.

- ☐ **10. Make password-reset links expire.** One use only, valid for 15–60 minutes, and invalidated after the password changes.

### 3. Data and user input (checks 11–17)

- ☐ **11. Validate every form on the server.** Browser checks are easy to bypass. Validate type, length and range again on the server (for example with zod).
- ☐ **12. Use parameterised queries.** Use your ORM or query builder. Never build SQL by joining strings with user input.
- ☐ **13. Prevent XSS.** Escape or sanitise anything users can post (comments, names, bios) before showing it. Avoid `dangerouslySetInnerHTML` with user content.
- ☐ **14. Restrict file uploads.** Allow only the file types you need, set a size limit, rename files, and store them in private storage.
- ☐ **15. Verify prices and payments on the server.** Never trust a price or total sent from the browser. Check payment webhooks with the provider's signature.
- ☐ **16. Rate-limit sensitive endpoints.** Limit login, signup, OTP, password reset and AI calls per user and per IP to stop brute force and surprise bills.
- ☐ **17. Return only the data you need.** API responses shouldn't include password hashes, emails of other users or internal fields.

### 4. HTTPS, headers and production config (checks 18–23)

- ☐ **18. Enforce HTTPS.** Redirect all HTTP traffic to HTTPS and turn on HSTS.
- ☐ **19. Restrict CORS.** Allow only your own domains. Never use `*` on APIs that use cookies or tokens.
- ☐ **20. Add security headers.** Set Content-Security-Policy, X-Frame-Options, X-Content-Type-Options and Referrer-Policy.
- ☐ **21. Secure your cookies.** Mark session cookies HttpOnly, Secure and SameSite, and protect forms against CSRF.
- ☐ **22. Turn off debug mode in production.** Show users a generic error page and keep stack traces in your logs only.
- ☐ **23. Remove leftovers.** Delete test accounts, seed data, exposed `.git` folders, backups and public source maps from the live site.

### 5. Dependencies, monitoring and testing (checks 24–27)

- ☐ **24. Update vulnerable packages.** Run `npm audit` (or your package manager's equivalent) and update anything flagged high or critical.

- ☐ **25. Remove unused packages.** Every extra dependency is extra attack surface. Uninstall what the app no longer uses.
- ☐ **26. Log and monitor suspicious activity.** Log failed logins, permission errors and unusual spikes, and set an alert. Keep automatic database backups on.
- ☐ **27. Run a full security test.** Ask Claude to audit the codebase against this list (prompt below), fix every finding, then test the main flows again.