

VIBE-CODED WEBSITE PRODUCTION CHECKLIST

25 Checks + 5 Bonus Tips + AI Optimization Prompt

A production-readiness operational guide for vibe coders, AI developers, indie hackers, and startup founders using Claude, Cursor, OpenAI Codex, Gemini, Lovable, Bolt, and Replit. **The Central Launch Rule:**

"Your vibe-coded website may look finished. That doesn't mean it's ready to launch."

AI prompts get you to 90% visually in minutes. The remaining 10%—error states, SEO hygiene, legal compliance, accessibility, security, and mobile quirks—determines whether your project succeeds or silently fails in production.

1. THE 25-POINT PRODUCTION CHECKLIST

01. Custom 404 Page

- **What:** A branded, helpful error page displayed whenever visitors hit a broken link or nonexistent route.
- **Why:** Default server or framework 404 pages leak technical stack details and cause immediate visitor bounces.
- **Checks:**
 - a. Navigate to an invalid route like `/this-page-does-not-exist` and confirm your custom layout renders.
 - b. Verify clear navigation links back to the home page or primary app entry point.
 - c. Ensure the HTTP response code returned is strictly 404 Not Found (not 200 OK).
- **Status:** ☐ Not checked ☐ Needs work ☐ Passed

02. Meta Title Per Page

- **What:** Unique, descriptive title tags tailored for every public route across the site.
- **Why:** Crucial for search indexing and click-through rates; avoids every tab displaying generic placeholders like "Vite + React".
- **Checks:**
 - a. Inspect the title tag across home, pricing, blog, and auth pages.
 - b. Confirm titles follow the formula: Page Name | Brand or Value Proposition (under 60 characters).
- **Status:** ☐ Not checked ☐ Needs work ☐ Passed

03. Meta Description Per Page

- **What:** A tailored 140–160 character summary embedded in the meta description tag for each page.
- **Why:** Directly drives organic click-through rates from search engine result snippets.
- **Checks:**
 - a. Verify each route contains a distinct, non-duplicate meta description.
 - b. Check that descriptions clearly explain the value proposition and include a soft call-to-action.
- **Status:** ☐ Not checked ☐ Needs work ☐ Passed

04. Call-to-Action (CTA) Above the Fold

- **What:** The primary action button (e.g., "Start Free Trial", "Book Demo") visible without scrolling on standard screens.
- **Why:** Over 50% of first-time visitors never scroll below the initial viewport.
- **Checks:**
 - a. View the landing page on standard laptop (1366×768 and 1920×1080) screens.
 - b. Confirm high-contrast visual hierarchy and that the CTA is immediately clickable without scrolling.
- **Status:** ☐ Not checked ☐ Needs work ☐ Passed

05. Favicon & App Icons

- **What:** Custom browser tab icons in .ico, .png, and SVG formats, including apple-touch-icon.
- **Why:** Prevents embarrassing default icons (Vercel, Vite, Next.js, Netlify) from appearing in customer browser tabs and bookmarks.
- **Checks:**
 - a. Verify favicon renders cleanly in light mode, dark mode, and mobile bookmarks.
 - b. Check link rel icon and apple-touch-icon tags in the HTML head.
- **Status:** ☐ Not checked ☐ Needs work ☐ Passed

06. robots.txt

- **What:** A plain text file located at `/robots.txt` directing search engine crawlers.
- **Why:** Ensures search engines crawl public pages while blocking private staging, internal APIs, and auth routes.
- **Checks:**
 - a. Navigate to `/robots.txt` in your browser.
 - b. Confirm User-agent: *, Allow: /, disallow rules for sensitive paths, and the exact sitemap location.
- **Status:** ☐ Not checked ☐ Needs work ☐ Passed

07. sitemap.xml

- **What:** An XML document at `/sitemap.xml` listing all indexable URLs on your website.
- **Why:** Accelerates search engine discovery and indexation of new pages and updates.
- **Checks:**
 - a. Open `/sitemap.xml` and verify all URLs use canonical production HTTPS links.
 - b. Ensure dynamic and draft/admin pages are excluded.
- **Status:** ☐ Not checked ☐ Needs work ☐ Passed

08. Open Graph (OG) Image & Social Cards

- **What:** Dedicated preview image (1200×630px) and tags (`og:image`, `og:title`, `twitter:card`).
- **Why:** Links shared on Twitter/X, LinkedIn, Slack, and messaging apps look trustworthy and get higher click engagement.
- **Checks:**
 - a. Test your URL in a card validator or open graph debugger.
 - b. Ensure image text is legible at thumbnail size and uses an absolute HTTPS URL.
- **Status:** ☐ Not checked ☐ Needs work ☐ Passed

09. Alt Text for Images

- **What:** Descriptive alt attributes on every non-decorative `img` element.
- **Why:** Essential for screen-reader accessibility and image search indexing.
- **Checks:**
 - a. Verify informative images describe their context and meaning.
 - b. Verify pure background or decorative icons use empty alt attributes or `aria-hidden`.
- **Status:** ☐ Not checked ☐ Needs work ☐ Passed

10. Mobile Responsive Layout

- **What:** Fluid design layout that adjusts smoothly across viewports from 360px to 4K displays.
- **Why:** More than 60% of web traffic originates on mobile devices; horizontal scrolling destroys conversions.
- **Checks:**
 - a. Test at 360px, 390px, 768px, and 1024px widths.
 - b. Confirm no horizontal overflow and text is legible without zooming.
- **Status:** ☐ Not checked ☐ Needs work ☐ Passed

11. Mobile Call-to-Action (Sticky / Visible CTA)

- **What:** An accessible, easy-to-tap CTA optimized specifically for touch devices (minimum 44×44px target).
- **Why:** Desktop CTA buttons often get buried below deep mobile fold heights or become clumsy to tap.
- **Checks:**
 - a. Verify the primary CTA appears cleanly in the mobile viewport or as a sticky bottom bar.
 - b. Tap with one thumb to verify comfort and immediate response.
- **Status:** ☐ Not checked ☐ Needs work ☐ Passed

12. Loading States & Feedback

- **What:** Visual spinners, skeleton screens, or disabled button states during network requests.
- **Why:** Prevents confused users from clicking buttons multiple times and submitting duplicate requests or orders.
- **Checks:**
 - a. Throttle network to Slow 3G in DevTools.
 - b. Submit forms or trigger data loads: ensure buttons disable and show a clear loading indicator.
- **Status:** ☐ Not checked ☐ Needs work ☐ Passed

13. Form Error States

- **What:** Clear, contextual error messages when form inputs are missing, invalid, or server requests fail.
- **Why:** AI coding tools often forget backend error handling or leave generic unhelpful alerts.
- **Checks:**
 - a. Submit empty required inputs and enter invalid email formats.
 - b. Simulate network disconnection to confirm graceful error message presentation.
- **Status:** ☐ Not checked ☐ Needs work ☐ Passed

14. Thank-You / Success Page

- **What:** A confirmed success screen or dedicated URL after a form submission or purchase.
- **Why:** Confirms receipt to the user, outlines immediate next steps, and provides a target for analytics conversion pixels.
- **Checks:**
 - a. Submit every contact, lead, and payment form on the site.

- b. Verify confirmation message or redirect to `/thank-you` works with tracking firing.
- **Status:** ☐ Not checked ☐ Needs work ☐ Passed

15. Privacy Policy

- **What:** A comprehensive, publicly accessible document explaining data collection, storage, and processing.
- **Why:** Legally mandatory under GDPR, CCPA, and required by payment gateways (Stripe) and ad networks.
- **Checks:**
 - a. Confirm footer link connects to an updated Privacy Policy with your legal entity name and contact email.
 - b. Detail third-party tracking tools (Analytics, PostHog, Stripe, Crisp).
- **Status:** ☐ Not checked ☐ Needs work ☐ Passed

16. Terms & Conditions

- **What:** Legal agreement governing the terms under which users access your software, content, and services.
- **Why:** Limits your liability, establishes intellectual property rights, and defines refund and cancellation rules.
- **Checks:**
 - a. Link Terms & Conditions in the footer and sign-up flows.
 - b. Ensure liability limitations, governing jurisdiction, and refund terms are explicit.
- **Status:** ☐ Not checked ☐ Needs work ☐ Passed

17. Cookie Consent (Where Applicable)

- **What:** A compliant cookie banner giving visitors control over non-essential tracking cookies.
- **Why:** Required by EU/UK GDPR and ePrivacy directives when utilizing analytics or marketing trackers.
- **Checks:**
 - a. Ensure non-essential scripts (analytics, marketing pixels) do not fire before consent is granted.
 - b. Confirm preferences can be reviewed and adjusted by users at any time.
- **Status:** ☐ Not checked ☐ Needs work ☐ Passed

18. Analytics & Event Tracking

- **What:** Web analytics setup (Plausible, PostHog, Google Analytics, Umami) to track traffic and goal events.

- **Why:** Without analytics, you cannot gauge visitor acquisition channels, drop-offs, or conversion success.
- **Checks:**
 - a. Verify script loads without console errors and registers pageviews in real-time reporting.
 - b. Test that key conversion events (signup click, form submit) trigger successfully.
- **Status:** ☐ Not checked ☐ Needs work ☐ Passed

19. Real Contact Information

- **What:** Legitimate contact channels: working support email, physical/virtual business address, or active contact form.
- **Why:** Establishes credibility, prevents customer frustration, and is mandatory for payment merchant verification.
- **Checks:**
 - a. Send a test message to your support email or form and verify inbox delivery.
 - b. Ensure no placeholder text (such as support@example.com or 123 Main St) remains.
- **Status:** ☐ Not checked ☐ Needs work ☐ Passed

20. Compressed & Modern Images

- **What:** Image assets optimized using modern formats (WebP, AVIF) and properly dimensioned.
- **Why:** Large AI-generated PNGs (3MB–8MB) ruin mobile loading speed and plummet Core Web Vitals scores.
- **Checks:**
 - a. Verify hero images and graphics are under 200KB where possible.
 - b. Check that width and height attributes are declared to prevent Cumulative Layout Shift (CLS).
- **Status:** ☐ Not checked ☐ Needs work ☐ Passed

21. Semantic HTML

- **What:** Proper structural tags (header, nav, main, section, article, footer, h1–h6).
- **Why:** Empowers accessibility screen readers and clarifies content hierarchy for search crawlers.
- **Checks:**
 - a. Confirm there is exactly one h1 per page reflecting the core topic.
 - b. Replace generic clickable div elements with standard button or anchor tags.
- **Status:** ☐ Not checked ☐ Needs work ☐ Passed

22. Keyboard Accessibility

- **What:** Full site navigability using only the keyboard (Tab, Shift+Tab, Enter, Space, Escape).
- **Why:** Ensures users with disabilities can navigate, while providing visible focus indicators for power users.
- **Checks:**
 - a. Navigate the entire site using only Tab and Shift+Tab.
 - b. Ensure focused elements have visible focus rings and modal dialogs close with Escape.
- **Status:** ☐ Not checked ☐ Needs work ☐ Passed

23. Performance & Core Web Vitals

- **What:** Fast page metrics: Largest Contentful Paint (LCP < 2.5s), Cumulative Layout Shift (CLS < 0.1), and Interaction to Next Paint (INP < 200ms).
- **Why:** Directly impacts Google ranking and prevents user churn caused by laggy web interactions.
- **Checks:**
 - a. Run Lighthouse / PageSpeed audit on production build.
 - b. Confirm mobile performance score is at least 80+ and no layout jumping occurs during load.
- **Status:** ☐ Not checked ☐ Needs work ☐ Passed

24. Secure Forms & Input Validation

- **What:** Both client-side and strict server-side validation, rate limiting, and spam protection.
- **Why:** Client-side checks can be bypassed in seconds, exposing backend endpoints to bot spam, injection, and denial of service.
- **Checks:**
 - a. Test submitting oversized payloads, malicious script tags, and rapid repeated requests.
 - b. Verify bot protection (such as Cloudflare Turnstile or CAPTCHA) is active.
- **Status:** ☐ Not checked ☐ Needs work ☐ Passed

25. Cross-Browser & Cross-Device Testing

- **What:** Verifying layout, styling, and interactivity across Chrome, Safari, Firefox, and Edge on iOS, Android, macOS, and Windows.
- **Why:** Safari on iOS handles CSS flexbox, 100vh viewport units, and form controls differently than Chromium browsers.
- **Checks:**
 - a. Test UI and form interactions specifically on iOS Safari and mobile Chrome.

- b. Verify flexbox and sticky elements render without overlap or clipping.
- **Status:** ☐ Not checked ☐ Needs work ☐ Passed

2. 5 BONUS CHECKS

26. Canonical URLs

- **Explanation:** Specifying `rel="canonical"` tags tells search engines the master copy of a page, preventing duplicate content penalties caused by query parameters or HTTP/HTTPS variants.
- **Practical Check:** Inspect source code on key pages and verify the canonical URL points strictly to the preferred production HTTPS domain without trailing slash inconsistencies.

27. Structured Data / Schema Markup

- **Explanation:** Embedding JSON-LD schema (e.g., Organization, SoftwareApplication, Product, FAQPage) gives search engines rich semantic context to generate search rich snippets.
- **Practical Check:** Test your production URL in Google's Rich Results Test tool to ensure valid, error-free JSON-LD schema.

28. Security Headers

- **Explanation:** HTTP response headers (Content-Security-Policy, X-Frame-Options, X-Content-Type-Options, Strict-Transport-Security, Referrer-Policy) protect against clickjacking, MIME sniffing, and cross-site attacks.
- **Practical Check:** Scan your domain using an online security header scanner and aim for an A or B rating.

29. Custom Social Sharing Metadata

- **Explanation:** Setting specific Twitter card styles (`summary_large_image`) and Open Graph tags guarantees maximum visual impact across Twitter/X, LinkedIn, Slack, and WhatsApp.
- **Practical Check:** Send test links to a private Slack channel and messaging app to verify title, description, and high-resolution thumbnail display properly.

30. HTTPS & Custom Domain Verification

- **Explanation:** Forcing HTTPS everywhere, setting up automatic HTTP-to-HTTPS redirects, and verifying www to non-www redirects ensures seamless encryption and prevents certificate errors.

- **Practical Check:** Type both the HTTP and non-secure versions of your domain into your browser bar; confirm both automatically redirect seamlessly to HTTPS with a valid SSL padlock.

3. QUICK PRE-LAUNCH CHECK

(One-Page Pre-Flight Checklist — Check off 15 minutes before hitting launch)

- ☐ **Mobile works** — No horizontal scrolling; typography and layouts scale cleanly at 375px+
- ☐ **Navigation works** — All navbar, dropdown, and footer links direct to valid pages without dead ends
- ☐ **Buttons work** — Every primary and secondary CTA triggers expected actions with clear visual states
- ☐ **Forms work** — Submissions deliver payloads to backend/inbox and trigger confirmation UI
- ☐ **Error states work** — Invalid inputs trigger clear, readable, non-technical feedback messages
- ☐ **Favicon works** — Custom icon displays crisply across light mode, dark mode, and mobile tabs
- ☐ **Metadata exists** — Unique page title and description tags configured on all indexable routes
- ☐ **Social preview works** — Open Graph image (1200×630px) and preview cards show up on Slack and X
- ☐ **robots.txt exists** — Accessible at </robots.txt> with correct crawl rules and sitemap pointer
- ☐ **sitemap exists** — Accessible at </sitemap.xml> with production HTTPS URLs
- ☐ **Images are optimized** — Formatted as WebP/AVIF, compressed under 200KB, explicit dimensions set
- ☐ **No obvious console errors** — Browser Console shows zero uncaught exceptions or failed network calls
- ☐ **HTTPS works** — Valid SSL certificate active with strict automatic redirect from HTTP
- ☐ **Domain works** — Root domain and subdomain routing configured cleanly with zero DNS issues
- ☐ **404 page works** — Custom branded 404 page appears with home link on invalid routes
- ☐ **Analytics works** — Live visitor counts and conversion goal events register in analytics dashboard
- ☐ **Legal pages are accessible** — Privacy Policy and Terms of Service linked, populated, and fully active

4. 10 VALUABLE TIPS FOR AI-BUILT WEBSITES

1. **Don't Trust AI-Generated Code Blindly.**

AI models generate code that looks convincing and syntactically correct, but frequently hallucinate outdated library methods, insecure configurations, or non-existent API options. Always conduct a line-by-line review of generated utility functions, regular expressions, and authentication logic.

2. **Test Real User Flows End-to-End.**

Do not assume that because an AI agent created a beautiful signup modal, the actual registration, session token storage, and redirection flow works. Perform complete end-to-end user journeys: sign up, log out, reset password, complete checkout, and request a support ticket.

3. **Always Test on a Real Physical Phone.**

Desktop browser DevTools device emulation cannot simulate touch latency, soft-keyboard view adjustments, iOS Safari address bar resizing, or hardware performance constraints. Pick up a physical iPhone and Android device and use your website with one thumb before launch.

4. **Never Expose Secret API Keys.**

AI tools routinely hardcode private environment variables (`STRIPE_SECRET_KEY`, `OPENAI_API_KEY`, database credentials) into client-side components. Ensure private keys are confined to backend endpoints or secure serverless functions, exposing only public publishable keys to the browser.

5. **Don't Rely Solely on Client-Side Validation.**

AI helpers love generating clean HTML5 or React client-side validation rules, but client checks can be bypassed with a simple curl request. Enforce schema validation (e.g., Zod, Joi) and rate limiting on every backend API endpoint to safeguard your database.

6. **Rigorously Check Loading and Empty States.**

AI demos look spectacular when filled with dummy data, but real users encounter zero-data dashboards and slow network latencies. Explicitly build and review skeleton loading screens, button spinners, and helpful empty states for new user profiles.

7. **Compress and Dimension Every Image.**

AI visual generators export heavy PNGs that choke bandwidth and degrade Google Core Web Vitals. Run every image through WebP/AVIF compression tools and specify explicit width and height attributes to prevent Cumulative Layout Shift.

8. **Keep Dependencies Lean and Updated.**

Vibe coding prompts often import massive, redundant npm packages for simple tasks that native JavaScript can handle in three lines. Regularly audit `package.json`, eliminate bloatware, and run dependency audits to patch security vulnerabilities.

9. **Actively Audit Accessibility (A11y).**

AI coding tools commonly replace native HTML button and anchor elements with styled div tags lacking ARIA attributes or keyboard listeners. Use Lighthouse or browser audit

tools to confirm proper heading hierarchy, color contrast ratios, and complete keyboard navigability.

10. **Test Again After Every Major AI-Generated Change.**

When you prompt an AI agent to adjust a single layout component or fix a styling bug, it can silently refactor adjacent code and break existing logic. Run regression checks, automated tests, and build commands after every major AI iteration before committing to production.

5. **WEBSITE QUALITY SCORECARD**

Use this self-assessment scorecard to evaluate production readiness. Rate each category from 1 to 5:

- **1 = Poor** (Critical bugs, missing essentials, launch blocker)
- **3 = Needs Improvement** (Functional but rough around the edges)
- **5 = Ready** (Production grade, tested, robust)

Category	Score	Priority	Notes
UX	/5	High ▾	Visual hierarchy, clear CTAs, intuitive navigation, working empty states
SEO	/5	High ▾	Unique titles, meta descriptions, sitemap.xml, robots.txt, schema
Performance	/5	High ▾	Fast Core Web Vitals, compressed images, lightweight bundles
Accessibility	/5	Medium ▾	Keyboard tab flow, color contrast, semantic HTML, descriptive alt text
Security	/5	High ▾	Server-side validation, secure headers, environment variable isolation
Mobile	/5	High ▾	Tested on physical iOS/Android devices, sticky CTAs, no horizontal scroll

Category	Score	Priority	Notes
Conversion	/5	Medium	Live analytics, event tracking, thank-you pages, clear value proposition

Note: This scorecard is an internal personal self-assessment framework designed to enforce engineering discipline, not an official certification.

6. MASTER CLAUDE / CURSOR / CODEX AUDIT PROMPT

Copy and paste the exact prompt block below into Claude Code, Cursor Composer, OpenAI Codex, or your preferred AI coding terminal: Audit my entire website before changing anything.

Act as a Principal Full-Stack Engineer, Technical SEO Architect, and Production QA Specialist. Your objective is to perform a comprehensive, rigorous pre-launch audit of this codebase across every route, component, and configuration file.

Inspect the following categories systematically:

1. Structure & Architecture: Code modularity, clean folder structure, separation of concerns.
2. Pages & Routes: All public and protected routes, dynamic routing parameters, redirect handling.
3. UX & Interactivity: Visual hierarchy, clear call-to-action (CTA) paths, intuitive layout.
4. SEO & Metadata: Unique <title> and meta descriptions per page, Open Graph cards, sitemap.xml, robots.txt.
5. Mobile Responsiveness: Breakpoint handling (360px to 4K), no horizontal overflow, touch target sizing (min 44×44px).
6. Accessibility (A11y): Semantic HTML tags, aria labels, keyboard navigation, focus rings, image alt attributes.
7. Performance & Core Web Vitals: Bundle size, lazy loading, font optimization, image formatting (WebP/AVIF) and dimensions.
8. Forms & Inputs: Client-side AND server-side input validation, rate limiting, anti-spam safeguards.

9. Loading & Error States: Disabled button states during fetch, skeleton loaders, clear error feedback for failures.

10. Security & Hygiene: Zero exposed private API keys/secrets, strict environment variable separation, secure HTTP headers.

11. Analytics & Tracking: Correct placement of tracking tags, conversion event triggers, cookie consent compatibility.

12. Legal & Trust Pages: Functional Privacy Policy, Terms of Service, real contact details, custom 404 page.

13. Broken Links & Dead Ends: Internal links, external anchor targets, social icon links.

14. Build & Console Errors: Zero linter errors, zero TypeScript compilation errors, zero unhandled browser console warnings.

Follow this strict methodology:

AUDIT → PRIORITIZE → FIX → TEST → RETEST

Execution Rules:

1. First, inspect the existing project thoroughly. Do NOT modify any files yet.

2. Generate a concise audit document named "WEBSITE_AUDIT.md" in the project root detailing your findings.

3. Categorize all issues by severity: Critical (blocks launch), High (affects conversions/security), and Medium (quality improvement).

4. When instructed to fix issues, do NOT unnecessarily rewrite working code or replace established architectures.

5. Preserve all existing functionality and design intentions unless an explicit fix is required.

6. Never invent mock credentials, fake API endpoints, or hallucinated database schemas.

7. Never expose secrets or hardcode sensitive values into client-side bundles.

8. Verify every code modification by running available linters, TypeScript checks, and test suites.

9. Run local build commands (e.g., npm run build, yarn build) to confirm zero build failures.

10. Conduct a final production-readiness review against the pre-launch standards.

At the conclusion of your audit and remediation, output a summary in this exact format:

FINAL STATUS:

🔴 Not Ready | 🟡 Needs Work | 🟠 Almost Ready | 🟢 Production-Ready

(Do NOT mark as Production-Ready unless every critical and high issue has been verified and passed.)

- Fixed Issues: [List of specific issues resolved]
- Remaining Issues: [List of non-blocking or manual issues pending]
- Files Changed: [List of modified file paths]
- Tests Performed: [Linters, build steps, or automated tests executed]
- Manual Checks Still Required: [Actions that require human testing, e.g., real payment processing, email inbox verification]

7. THE PRODUCTION LAUNCH WORKFLOW

Follow this continuous operational cycle for every web release:

BUILD → AUDIT → FIX → TEST → SECURITY → SEO → MOBILE → PERFORMANCE → LAUNCH → MONITOR

- **BUILD:** Generate UI and features using your preferred AI coding tools (Claude, Cursor, Codex, Lovable, Bolt, Replit).
- **AUDIT:** Run automated and manual audits across architecture, UX, accessibility, and SEO.
- **FIX:** Address critical blockers and errors methodically without breaking adjacent features.
- **TEST:** Execute build commands, component tests, and end-to-end user journeys.
- **SECURITY:** Audit environment variables, server-side validation, and security headers.
- **SEO:** Verify unique meta tags, sitemap.xml, robots.txt, and Open Graph previews.
- **MOBILE:** Validate on real mobile devices to ensure touch targets and viewport responsiveness.
- **PERFORMANCE:** Optimize asset weight, compress images, and inspect Core Web Vitals.
- **LAUNCH:** Point DNS, verify SSL certificates, and flip production traffic live.

- **MONITOR:** Track real-time analytics, user feedback, and error logs for proactive fixes.

Final Takeaway:

"AI can build your website. You still need to make sure it actually works."

Code generation is instantaneous. Production reliability is deliberate.