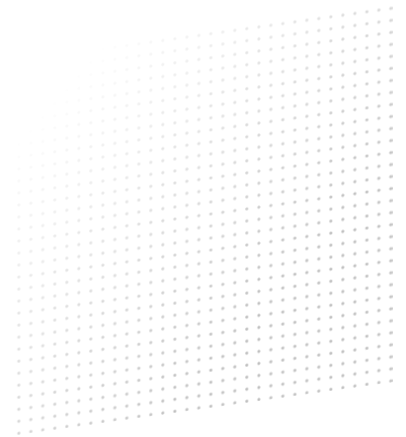

A RESOURCE FOR DESIGN SYSTEM AND ENGINEERING TEAMS

AI code review checklist.



For pull requests where a model wrote the first draft.

Scripts first. Then people. Nobody grades their own work.

38

checks, each with its evidence

11

errors the script finds in the
fixture PR

00

errors in the clean version

Generated code fails in its own ways.

A person who guesses an API usually knows they guessed. A model does not flag its guesses: the invented prop, the package that was never published and the test that checks nothing all arrive with the same confidence as the correct lines. In a controlled study, developers with an AI assistant wrote less secure code and were more likely to believe it was secure [S11].

Some of these failures are cheap to catch mechanically, and one of them is a supply-chain risk. Across 576,000 generated code samples, 19.7% of the packages the models suggested did not exist, and 43% of those hallucinated names came back in all ten repeated runs of the same prompt [S04]. A name that repeats is a name somebody can register.

This guide orders the review so that scripts go first and human attention goes where only humans help. The kit has a pull request template with the same item numbers and a checker that finds hallucinated imports, invented props and empty tests. On the fixture pull request it reports 11 errors; on the clean version of the same feature, none.

Version 1.1 / Sources checked 24 September 2026

Field Guide 20 of the Design × AI series. Checks props against the Field Guide 02 Button contract and pairs with Field Guide 03 (token lint), 19 (doc generation) and 21 (automation decisions). OWASP Top 10 for LLM Applications 2025, npm CLI 12 docs and GitHub Docs checked 24 September 2026.

Practical guidance, not a standard. The checker is a scanner, not a compiler: it complements your typecheck, lint, tests and security tooling and replaces none of them. WCAG references are to WCAG 2.2. Prepared with AI assistance and edited by hand.

START HERE

Pick your route, then read the labels.

Run the checker on both fixtures first. The two versions of the same dialog show every class of problem this guide covers.

Reviewing one pull request

Run `check-ai-diff.mjs` on the changed files, then work Sections 01 to 07 in order. Stop at the first section that fails and send it back.

Setting up the gate

Add the PR template, run the checker next to typecheck, lint and tests, and make its errors blocking. Warnings go to the reviewer.

Agents opening pull requests

The same checks apply, with two additions: C06 (a reviewer who did not write it) and C32 (the PR must not widen its own permissions).

Deciding what to automate

Field Guide 21 scores the import and prop checks as automate and the review itself as assist. This guide is where that line sits.

Read the labels before the checks.

Label	Meaning
SCRIPT	Found by <code>check-ai-diff.mjs</code> or another tool named in the evidence line.
REVIEW	Needs a person reading the code or using the feature.
SECURITY	Traces to a risk in the OWASP Top 10 for LLM Applications 2025.
WCAG A / AA	Traces to a WCAG 2.2 success criterion at that level.
PRACTICE	A working method with a review signal rather than a hard gate.

The author owns every line, whoever typed it.

A pull request is a claim that the code is ready. Generated code does not lower the bar; it moves the effort from writing to verifying.

SECTION 01

Check the pull request before the code.

Six checks decide whether this pull request is ready for anyone's time.

Suggested owners: Author + reviewer

☐ **The PR says what a tool wrote**

C01 **PRACTICE**

Tool, model, task and the generated parts. Reviewers read generated code for different mistakes.

Evidence: The AI section of the PR template is filled in.

Recommended practice.

☐ **The author ran it and can explain it**

C02 **REVIEW**

The author used the feature and can explain any line. "The model wrote that" is not an answer.

Evidence: Template checkboxes; answers in review without regenerating.

Claude Code docs: if you cannot verify it, do not ship it. S09.

☐ **The diff does what the ticket asks, and no more**

C03 **REVIEW**

Agents over-deliver: extra files, abstractions, refactors. Out-of-scope changes get their own PR.

Evidence: Every changed file maps to the ticket.

GitHub Docs: verify context and intent. S07.

☐ **One concern per pull request**

C04 **PRACTICE**

Generated refactors apart from behaviour changes, dependency changes apart from both.

Evidence: The PR title names one change.

Recommended practice.

☐ **Machine checks run before a person reads**

C05 **SCRIPT**

Typecheck, lint with token rules, tests and `check-ai-diff.mjs` pass, with output in the PR.

Evidence: The Machine checks block in the template.

GitHub Docs: start with functional checks. S07.

☐ **The reviewer did not write it**

C06 **PRACTICE**

Another person, or a reviewer agent in a fresh context. Never the session that wrote the code.

Evidence: Reviewer field in the template.

Claude Code docs: review in a fresh context. S09.

Review time is the scarcest input in the pipeline. Spend it on ready code.

SECTION 02

Verify every package.

A hallucinated package name fails the build if nobody owns it, and runs someone else's code if somebody does. Check the manifest first, the registry second.

Suggested owners: Reviewer + platform owner

☐ Every bare import is declared in package.json

C07 **SCRIPT** **SECURITY**

An import that is not a dependency is either hallucinated or works only by accident of hoisting. Both fail somewhere else later.

Evidence: `import/undeclared-package`: 2 errors on the fixture (`@acme/ui-hooks`, `clsx`).

OWASP LLM09: models suggest non-existent libraries. S03.

☐ Every new dependency is real and intended

C08 **REVIEW** **SECURITY**

Run `npm view <name>`: first publish date, maintainers, repository. A week-old package with a model-suggested name is a warning sign. `--registry` does the lookup.

Evidence: `npm view` output in the PR; `registry/not-found` on the fixture for `@acme/ui-hooks`.

19.7% hallucinated packages, 43% repeated in all ten runs. S04. Slopsquatting. S05.

☐ Nobody installs a package to find out whether it exists

C09 **SECURITY**

Install runs the package's preinstall and postinstall scripts; `npm view` installs nothing. To evaluate, use `--ignore-scripts` in a throwaway environment.

Evidence: Template checkbox.

npm scripts and config docs. S12, S13. A registered hallucinated name drew over 30,000 real downloads. S06.

☐ Lockfile changes match package.json changes

C10 **REVIEW** **SECURITY**

No new transitive packages without a new direct one, no registry URL changes, no integrity hashes rewritten for unchanged versions.

Evidence: Lockfile diff reviewed; a lockfile lint in CI if you have one.

OWASP LLM03 supply chain. S01.

Check the name against the manifest, then the registry. Never against the installer.

SECTION 03

Hold imports and props to the contract.

The contract lists what exists (Field Guide 02). Generated code is where things get invented.

Suggested owners: Reviewer + design-system lead

☐ Every relative import resolves

C11 **SCRIPT**

Generated code imports helpers it assumed exist, such as `./lib/analytics`.

Evidence: `import/missing-file`: 1 error on the fixture.

GitHub Docs: hallucinated APIs. S07.

☐ Design-system imports use the package entry point

C12 **SCRIPT**

`@acme/ui/src/components/Dialog` couples product code to internals that change without notice.

Evidence: `import/deep-import`: 1 error on the fixture.

Field Guide 02 01.02: the exact import is written down.

☐ Every named import is exported

C13 **SCRIPT**

`PrimaryButton` sounds right and does not exist. Named imports are checked against the exports.

Evidence: `import/unknown-export`: 1 error on the fixture.

Field Guide 04 W16: name the hallucinations.

☐ Every prop is in the component contract

C14 **SCRIPT**

`variant`, `shadow`, `className`: if the contract does not list it, it does not exist.

Evidence: `props/unknown-prop`: 2 errors on the fixture.

Field Guide 02 02.02: invented props fail validation.

☐ Every literal value is in the prop's enum

C15 **SCRIPT**

`size="small"` and `tone="ghost-primary"` are plausible and wrong.

Evidence: `props/invalid-value`: 2 errors on the fixture.

Field Guide 02 01.07: closed sets are enums.

☐ Forbidden combinations fail the typecheck

C16 **SCRIPT**

The Field Guide 02 union types reject forbidden pairs. No `as` casts or `@ts-ignore` around them.

Evidence: `tsc --noEmit` passes; grep the diff for new casts and ignores.

Field Guide 02 02.03.

If the contract does not list it, the code cannot use it.

SECTION 04

Keep the code on the system.

Code can pass every contract check and still route around the design system. These five checks catch the detours.

Suggested owners: Reviewer + design-system lead

☐ No spread props onto design-system components

C17 **SCRIPT**

`<Button {...rest}>` forwards whatever the caller passed, and no reviewer or script can see it.

Evidence: `props/spread`: 1 warning on the fixture.

Recommended practice.

☐ No re-created components

C18 **REVIEW**

A local `PrimaryButton`, a styled `div` with a click handler, a hand-rolled modal. Look for new components that overlap the system's exports.

Evidence: Reviewer note; a lint rule for raw `<button>` in product code if you want one.

Field Guide 02 06.07: verify, never invent.

☐ No deprecated APIs

C19 **SCRIPT**

Models learned from older versions. Deprecated props and components come back unless a lint rule rejects them.

Evidence: A `no-deprecated-apis` style lint rule.

Field Guide 02 07.03.

☐ No raw values or core tokens

C20 **SCRIPT**

Colours, spacing and radii come from semantic tokens. Inline `style={{ color: "#d4351c" }}` is the common generated shortcut.

Evidence: The Field Guide 03 ESLint and stylelint gates pass.

Field Guide 03.

☐ Patterns match the codebase, not the internet

C21 **REVIEW**

State, data fetching, file layout and naming follow this repository, not what was common in training data.

Evidence: Reviewer note.

GitHub Docs: assess code quality and consistency. S07.

A detour around the system is a fork of it.

SECTION 05

Check the experience, not only the rules.

No script in this kit sees the fixture's unnamed icon button. axe and a keyboard do.

Suggested owners: Reviewer + accessibility lead

☐ Every interactive element has an accessible name

C22 **REVIEW** **WCAG A**

Icon-only controls above all: with Button, `iconOnly` plus `aria-label`, enforced by the types.

Evidence: axe `button-name`; the typecheck for `iconOnly`.

WCAG 2.2 SC 4.1.2 Name, Role, Value. S16.

☐ Native elements before ARIA

C23 **REVIEW** **WCAG A**

No `div` with `onClick`, no `role="button"` on a button, no ARIA repeating native semantics.

Evidence: Review of new JSX; axe role and nesting rules.

APG: no ARIA is better than bad ARIA. S17.

☐ The flow works with a keyboard alone

C24 **REVIEW** **WCAG A**

Every action reachable, focus visible and returned after a dialog, no `outline: none`.

Evidence: Template checkbox; an interaction test per keyboard path.

WCAG 2.2 SC 2.1.1, 2.4.3 and 2.4.7. S16.

☐ Inputs have labels and errors are linked

C25 **REVIEW** **WCAG A**

Generated forms lean on placeholders. Every input needs a visible label and linked error text.

Evidence: axe `label`; a test for `aria-describedby`.

WCAG 2.2 SC 3.3.1 and 3.3.2. S16.

☐ Results are announced

C26 **REVIEW** **WCAG AA**

Saving, deleting and failing produce announced text. The fixture skipped exactly this test.

Evidence: A status message test that is not skipped.

WCAG 2.2 SC 4.1.3 Status Messages. S16.

☐ axe passes and a person checked

C27 **SCRIPT** **REVIEW**

Tools find part of the issues: all of them together found 71 percent of 143 planted barriers in a GOV.UK test [S19]. That is a floor.

Evidence: axe failing on violations in changed stories; keyboard pass recorded.

S19.

A passing axe run means no rule fired. It does not mean the dialog works.

SECTION 06

Treat generated code as untrusted input.

OWASP's 2025 LLM list applies to the code a model wrote and to features that call a model.

Suggested owners: Reviewer + security owner

☐ **No raw HTML without a sanitizer**

C28 **SCRIPT** **SECURITY**

`dangerouslySetInnerHTML` renders whatever the data holds. Sanitize it, or render text.

Evidence: `security/raw-html`: 1 warning on the fixture.

OWASP LLM05. S02. Veracode 2026: XSS handled securely in 15% of relevant tasks. S15.

☐ **No dynamic code execution**

C29 **SCRIPT** **SECURITY**

`eval`, `new Function` and string timers turn data into code.

Evidence: `security/dynamic-code`.

S02.

☐ **No secrets, internal URLs or personal data**

C30 **REVIEW** **SECURITY**

In code, fixtures, snapshots, prompts and comments. Generated fixtures copy real-looking data.

Evidence: Secret scanning in CI; reviewer note.

OWASP LLM02 sensitive information disclosure. S01.

☐ **Model output in the feature is untrusted**

C31 **REVIEW** **SECURITY**

If the code calls a model, its output is user input: encode, validate, least privilege.

Evidence: Threat notes in the PR for any model-calling feature.

OWASP LLM01 prompt injection and LLM06 excessive agency. S01.

☐ **The PR does not widen its own permissions**

C32 **REVIEW** **SECURITY**

CI, lint, compiler, coverage or CODEOWNERS changes need a reason. Agents sometimes edit the check.

Evidence: Template checkbox; CODEOWNERS on those paths.

GitHub Docs: watch for tests deleted or skipped instead of fixed. S07.

☐ **Automated security review runs on trusted PRs only**

C33 **PRACTICE** **SECURITY**

A diff can carry instructions. Anthropic's security review action is not hardened against that.

Evidence: Workflow requires approval for external contributors.

S10.

The code a model writes is input to your system. Review it like input.

SECTION 07

Make the tests prove something.

The essay behind this guide describes 200 generated tests that all passed and tested nothing useful [S18]. Green is a claim; a failing test is evidence.

Suggested owners: Reviewer + engineering lead

☐ **Every test asserts behaviour**

C34 **SCRIPT**

A test without `expect` or `assert` passes by running. A test that compares a literal with a literal passes by definition.

Evidence: `test/no-assertion` and `test/tautology`: 1 error each on the fixture.

Kit script.

☐ **Assertions check values, not existence**

C35 **SCRIPT** **REVIEW**

`toBeDefined()`, `toBeTruthy()` and a lone snapshot prove that something rendered. Assert the role, the name, the call and its arguments.

Evidence: `test/existence-only`: 1 warning on the fixture.

S18: tests that checked functions returned values, not correct values.

☐ **No test was skipped, deleted or loosened**

C36 **SCRIPT** **REVIEW**

`.skip`, a removed case, a widened snapshot or a lowered threshold to get to green.

Evidence: `test/skipped`: 1 warning on the fixture; the test diff read line by line.

GitHub Docs: tests deleted or skipped instead of fixed. S07.

☐ **Each test fails when its behaviour breaks**

C37 **REVIEW**

Break the code once and watch the test fail, or run mutation testing: a surviving mutant marks a test that asserts too little.

Evidence: Mutation score for the changed files, or the author's note of the deliberate break.

Stryker: surviving mutants and mutation score. S14.

☐ **Empty, loading, error and long content are covered**

C38 **REVIEW**

Generated tests favour the happy path. Name the edge cases in the PR, tested or listed as follow-ups.

Evidence: Test names or a follow-up list.

The essay's review list: null, undefined, empty, loading. S18.

A test you have never seen fail has not told you anything yet.

APPENDIX A

Two versions of one dialog.

`fixtures/clean` and `fixtures/ai-pr` implement the same delete-project dialog. Output below is from the kit, run on 24 September 2026.

terminal

```
$ node scripts/check-ai-diff.mjs --root fixtures/clean
3 file(s), 0 error(s), 0 warning(s)

$ node scripts/check-ai-diff.mjs --root fixtures/ai-pr --registry
warn  ...test.tsx:6    test/existence-only  "renders" only checks that something exists
error  ...test.tsx:11  test/no-assertion    "calls onDelete" asserts nothing
error  ...test.tsx:17  test/tautology       "works" compares a literal with a literal
warn  ...test.tsx:21  test/skipped         "announces the result" is skipped
error  ...Dialog.tsx:6  import/unknown-export @acme/ui does not export PrimaryButton
error  ...Dialog.tsx:7  import/deep-import   @acme/ui/src/components/Dialog: past the entry
point
error  ...Dialog.tsx:8  import/undeclared-package @acme/ui-hooks is not in package.json
error  ...Dialog.tsx:9  import/undeclared-package clsx is not in package.json
error  ...Dialog.tsx:12 import/missing-file   ./lib/analytics does not resolve to a file
error  ...Dialog.tsx:23 props/unknown-prop   <Button> has no prop variant in its contract
error  ...Dialog.tsx:23 props/invalid-value   size="small" is not one of sm, md, lg
error  ...Dialog.tsx:29 props/invalid-value   tone="ghost-primary" is not one of primary, ...
warn  ...Dialog.tsx:32 props/spread         <Button {...rest}> cannot be checked against the contract
error  ...Dialog.tsx:32 props/unknown-prop   <Button> has no prop shadow in its contract
warn  ...Dialog.tsx:28 security/raw-html    dangerouslySetInnerHTML: confirm it is sanitized
warn  (registry)      registry/not-found   @acme/ui-hooks is not on the npm registry
3 file(s), 11 error(s), 5 warning(s)
```

Shortened for print. Without `--registry` the run makes no network requests and reports 4 warnings. What it does not see in the same file: the trigger Button has no accessible name (C22), `Delete` does not say what is destroyed and the dialog has no title. That is the reviewer's half.

DO

- Run it on changed files: `git diff --name-only origin/main...`
- Keep contracts next to components and point `--contracts` at them
- Generate `design-system.json` from your package's real exports
- Treat warnings as reviewer prompts, not noise

DON'T

- Read a clean run as a review
- Add a hallucinated name to package.json to silence it
- Allow-list props to pass a PR; change the contract instead
- Expect it to see accessibility, logic or forbidden combinations

APPENDIX B

Put the checklist where the pull request is.

GitHub reads pull request templates from the default branch. Several templates go in a `PULL_REQUEST_TEMPLATE` folder and are chosen with a `template=` query parameter [S08].

`.github/PULL_REQUEST_TEMPLATE/ai-assisted.md` (excerpt)

```
## What an AI tool wrote (C01)

- Tool and model:
- Task or prompt given (link or paste):
- Files or parts written by the tool:

## Machine checks (C05)

```text
typecheck:
lint (including token rules, Field Guide 03):
tests:
check-ai-diff.mjs:
```

## Dependencies and imports (C07 to C13)

- [ ] No new dependency, or each new one is listed here with its `npm view` result
- [ ] I did not install anything to find out whether it exists.

## Tests (C34 to C38)

- [ ] Every new test fails when the behaviour it names breaks. I broke it once to see.
- [ ] No test was skipped, deleted or loosened to make this pass.
```

Open it with `?quick_pull=1&template=ai-assisted.md` on the compare URL, or copy it to `.github/pull_request_template.md` to make it the default.

| Where it runs | What it covers | Items |
|------------------------|--|--|
| Author, before opening | Disclosure, ran it, scope | C01 to C04 |
| CI | Typecheck, lint, tests, <code>check-ai-diff.mjs</code> | C05, C07, C11 to C17, C19, C20, C28, C29, C34 to C36 |
| Reviewer, reading | Packages, lockfile, patterns, security, test quality | C08 to C10, C18, C21, C30 to C33, C37, C38 |
| Reviewer, using it | Keyboard, names, announcements, edge cases | C22 to C27 |

KEEP WITH THE PULL REQUEST

Leave a review record.

For pull requests where generated code is a large share of the diff. It shows what was verified by whom, so the next incident review does not start from nothing.

| PR / ticket | Tool and model / share generated |
|---------------------------------------|---|
| Machine checks (links) | check-ai-diff.mjs errors and warnings |
| New dependencies and npm view results | Keyboard and screen-reader pass (who, AT) |
| Security items waived and why | Tests broken on purpose to confirm them |
| Reviewer (not the author) | Follow-ups filed |

Waive nothing silently.
A waived item with a reason is a decision. A skipped item without one is how the 200 useless tests shipped.

SOURCES / MAINTENANCE

Keep the guide current.

Sources checked 24 September 2026. Research and vendor figures are cited as their authors report them. The fixture package name @acme/ui-hooks returned 404 from the npm registry on that date; do not register or install it.

S01 / OWASP Top 10 for LLM Applications 2025

The ten risks, including LLM01 prompt injection, LLM02 sensitive information, LLM03 supply chain and LLM06 excessive agency.

<https://genai.owasp.org/llm-top-10/>

S02 / OWASP LLM05:2025 Improper Output Handling

Treat model output with zero trust; encoding, validation and CSP.

<https://genai.owasp.org/llmrisk/llm052025-improper-output-handling/>

S03 / OWASP LLM09:2025 Misinformation

Unsafe code generation, including non-existent libraries; attackers registering hallucinated names.

<https://genai.owasp.org/llmrisk/llm092025-misinformation/>

S04 / Spracklen et al., We Have a Package for You! (USENIX Security 2025)

576,000 samples from 16 models; 19.7% of packages hallucinated; 43% of hallucinations repeated in all ten runs.

<https://www.usenix.org/conference/usenixsecurity25/presentation/spracklen>

S05 / Socket, the rise of slopsquatting

Registering hallucinated package names; term credited to Seth Larson (April 2025).

<https://socket.dev/blog/slopsquatting-how-ai-hallucinations-are-fueling-a-new-class-of-supply-chain-attacks>

S06 / Lasso Security, diving deeper into AI package hallucinations

An empty package under a hallucinated name drew over 30,000 downloads in three months (2024).

<https://www.lasso.security/blog/ai-package-hallucinations>

S07 / GitHub Docs, review AI-generated code

Functional checks first, context and intent, dependencies, hallucinated APIs, skipped tests.

<https://docs.github.com/en/copilot/tutorials/review-ai-generated-code>

S08 / GitHub Docs, creating a pull request template

Template locations, multiple templates, default branch.

<https://docs.github.com/en/communities/using-templates-to-encourage-useful-issues-and-pull-requests/creating-a-pull-request-template-for-your-repository>

S09 / Claude Code, best practices

Give the model a way to verify; review in a fresh context; if you cannot verify it, do not ship it.

<https://code.claude.com/docs/en/best-practices>

S10 / Anthropic, claude-code-security-review

Not hardened against prompt injection; review trusted PRs only.

<https://github.com/anthropics/claude-code-security-review>

S11 / Perry et al., Do Users Write More Insecure Code with AI Assistants? (CCS 2023)

Assistant users wrote less secure code and were more confident it was secure.

<https://arxiv.org/abs/2211.03622>

S12 / npm CLI 12, npm view

Prints registry metadata; installs nothing.

<https://docs.npmjs.com/cli/v12/commands/npm-view>

S13 / npm CLI 12, config: ignore-scripts

Install runs lifecycle scripts unless ignore-scripts is true.

<https://docs.npmjs.com/cli/v12/using-npm/config>

S14 / Stryker Mutator documentation

Killed and surviving mutants; mutation score.

<https://stryker-mutator.io/docs/>

S15 / Veracode, Spring 2026 GenAI code security update

55% of generation tasks produced secure code across 150+ models; XSS secure in 15%.

<https://www.veracode.com/blog/spring-2026-genai-code-security/>

S16 / W3C, WCAG 2.2

Success criteria cited in Section 04.

<https://www.w3.org/TR/WCAG22/>

S17 / W3C, APG: Read Me First

No ARIA is better than bad ARIA.

<https://www.w3.org/WAI/ARIA/apg/practices/read-me-first/>

S18 / Petri Lahdelma, AI in design systems: what actually works

The essay's review list and the 200-test failure story.

<https://petrilahdelma.com/writing/ai-design-systems-practical>

S19 / GOV.UK accessibility blog, What we found when we tested tools on the world's least-accessible webpage

24 February 2017: 143 planted barriers; the best single tool found 37 percent (41 with manual-check prompts), all tools together 71 percent.

<https://accessibility.blog.gov.uk/2017/02/24/what-we-found-when-we-tested-tools-on-the-worlds-least-accessible-webpage/>

Maintenance: recheck the OWASP list when a new edition appears, npm's install-script defaults at each npm major (npm 12 added allow-scripts controls), and rerun both fixtures after any change to the checker. Update the PDF, HTML, Markdown and JSON together.