EcomLabs

STEP-BY-STEP GUIDE

AI code review: a two-level checklist

What the AI checks, what the author does with each finding, and what a person reviews next: business rules, decisions and data. With a change description template, CODEOWNERS for sensitive paths and the signs to hold the approval.

FOR	Teams or solo developers who use AI to review code before the merge
TESTED WITH	Node 22.22.3 · CODEOWNERS patterns checked with git check-ignore · our own practice, no external figures
UPDATED	November 19, 2026
DOCS	google.github.io/eng-practices/review
ARTICLE	ecomlabs.dev/g/review-en
AUTHOR	Luis Morales · @luismoralesdev

Contents

Before you start	How to use this checklist
Level 1	What the AI checks
Between levels	What the author does
Level 2	What the person reviews
Before the merge	When not to approve
Setup	A person on sensitive paths

BEFORE YOU START

How to use this checklist

Two levels, in order. Level 1 is the AI on the diff, after lint, types and tests pass. The author answers every finding. Level 2 is a person, and only that approval allows the merge.

LEVEL	WHO	REVIEWS	DECIDES
0	CI	Formatting, types, tests	Pass or fail
1	AI	What is in the diff	Nothing: it reports
2	Person	What is not in the diff	Approve or not

LEVEL 1

What the AI checks

Give it the diff and a versioned instructions file with the repo's short business rules. Ask it to report without editing. Every item must be provable with an input or a line:

- [ ] Edge inputs: `null`, empty list, 0, negatives, the exact threshold value.
- [ ] What happens when a call fails, times out or returns nothing.
- [ ] A missing `await`, an empty `catch`, an unhandled promise.
- [ ] Magic numbers or strings with no name.
- [ ] Names and comments that say something other than what the code does.
- [ ] New branches with no test, and tests that would pass even if the code were wrong.
- [ ] Tests deleted, skipped or with a changed expected value.
- [ ] Secrets, tokens or personal data in code, logs or fixtures.
- [ ] The diff against each business rule written in the instructions file.
- [ ] At the end: the list of what it did not check.

Format for each finding: `path:line · blocks | question | minor · what fails · input that shows it`.

BETWEEN LEVELS

What the author does

No finding goes unanswered before a person is asked to review:

FINDING	WHAT YOU DO
Blocks, with an input	Reproduce it, fix it and add a test with that input.
Blocks, no input	Look for one yourself. If you cannot find it, answer why it does not apply.
Question	Answer in the PR. If the answer is a rule, move it into the instructions file.
Minor	Fix it if it is cheap; otherwise note it.
"No findings"	Copy into the PR what the AI said it did not check.

Then write the description for the person, not for the AI. This template gives them what is not in the diff:

```
## What changes
One or two sentences.

## Business rule that applies
E.g.: free shipping from 999 MXN of subtotal after discounts.

## Sensitive paths touched
Payments, migrations, middleware, personal data: which ones and why.

## What was not tested
What the tests do not cover and how you checked it by hand.

## AI findings
Blocks: fixed (list) · Answered without a change: (list with the reason)
Not checked, per the AI: (list)
```

LEVEL 2

What the person reviews

Do not redo the mechanical pass. Read the findings and the answers, read the whole diff, and spend the time here:

- [ ] The change follows the business rule the description names, and that rule still holds.
- [ ] The instructions file does not contradict what was agreed; if the rule changed, it is updated in the same PR.
- [ ] The change should exist: it does not add something you do not want in the system.
- [ ] It respects decisions already made (libraries, what you agreed not to solve).
- [ ] Who calls this, what becomes public and which field is personal data.
- [ ] Where the logs go and who reads them.
- [ ] Money, stock, paid orders: the case that costs real money if it fails.
- [ ] The author's answers to the findings hold up.
- [ ] You can explain in your own words what the change does and why.

BEFORE THE MERGE

When not to approve

- [ ] A "blocks" finding with no fix and no answer.
- [ ] It touches a protected path and its owner has not reviewed it.
- [ ] A test changed, was skipped or was deleted with no written reason.
- [ ] It touches money, stock or personal data and the description does not name the rule.
- [ ] "No findings" with no list of what was not checked.
- [ ] It mixes a refactor with a behavior change.
- [ ] It makes code health worse, even if it works.

If none of that applies and the change improves the code even if it is not perfect, approve it.

SETUP

A person on sensitive paths

```
# .github/CODEOWNERS
/lib/payments/          @your-username
/db/migrations/         @your-username
/middleware.ts          @your-username
/.github/ai-review.md   @your-username
/.github/CODEOWNERS     @your-username
```

On GitHub, turn on "Require review from Code Owners" in the main branch's protection rule. The owner needs write access. A leading `/` anchors to the root and a trailing `/` covers everything inside; the last matching line wins.

> **CAUTION**
>
> An AI comment is never an approval. If you work alone, do level 2 at a different time, with this list and not the AI's.