Coding & Development / Code Reviews

Review code for readability, naming conventions, comment quality, and adherence to style guides (PEP 8, ESLint, Prettier).
Difficulty: Intermediate
Model: GPT-4 / Claude / Gemini
Use Case: Code Reviews, Quality Assurance, Team Standards
Updated: May 2026
Why This Prompt Exists
Most code reviews focus on functionality — ignoring readability, maintainability, and consistency.

You get:

- code that works but is hard to read
- inconsistent naming (camelCase mixed with snake_case)
- missing or outdated comments
- functions that are too long (100+ lines)
- no adherence to team style guides

But code quality is not subjective.

It is measurable by style guides and best practices.

- Naming: descriptive, consistent, follows language conventions
- Comments: explain why, not what (code should show what)
- Structure: functions under 20-30 lines, single responsibility
- Formatting: consistent indentation, line length, spacing
- Dead code: commented-out blocks, unused variables, unreachable code

Without quality reviews, technical debt compounds.

This framework forces AI to review code quality systematically.

The Prompt

```
Assume the role of a senior engineer who reviews code for quality and
maintainability.

Your task is to review code for quality issues.

Generate:

1. NAMING ISSUES
   - Unclear variable/function names
   - Inconsistent naming conventions

2. COMMENT ISSUES
   - Missing function/docstring comments
   - Outdated or misleading comments
   - Commented-out code blocks

3. STRUCTURE ISSUES
   - Functions that are too long (recommend max 20-30 lines)
   - Functions doing multiple things (violating single responsibility)
   - Deep nesting (if/for beyond 3-4 levels)

4. FORMATTING ISSUES
   - Inconsistent indentation
   - Line length violations
```

```
   - Missing whitespace

5. DEAD CODE
   - Unused variables
   - Unreachable code
   - Commented-out blocks that should be deleted

6. RECOMMENDATIONS (prioritized)

INPUTS:

Code (paste):
[PASTE CODE]

Language:
[PYTHON / JAVASCRIPT / TYPESCRIPT / JAVA / OTHER]

Style Guide:
[PEP 8 / ESLINT / GOOGLE / AIRBNB / CUSTOM]

Review Focus:
[READABILITY / MAINTAINABILITY / CONSISTENCY / ALL]

RULES:
- Flag unclear names (single letters, abbreviations)
- Flag missing docstrings for public functions
- Flag functions over 30 lines (or configurable limit)
- Flag nesting depth over 4 levels
- Flag commented-out code (should be deleted, not commented)
```

```
- Prioritize recommendations (fix these first)
- Don't nitpick formatting if a linter already handles it
```

How To Use It

- Run this on new code before committing — catch issues early.
- Prioritize structural issues over formatting (a linter can fix formatting).
- Use the same style guide as your team (PEP 8 for Python, ESLint for JS).
- Flag unclear names — if you can’t tell what a variable does, it’s a problem.
- Delete commented-out code — it’s in git history if you need it back.

Example Input

Code:
def calc(a,b):
# calculate stuff
x=a+b
y=a*b
# this is a comment
z=x+y
return z
print(“done”)

Language: PYTHON

Style Guide: PEP 8

Review Focus: ALL

Why It Works
Most code reviews ignore quality.

This framework improves outcomes by forcing:

- naming issue detection (readability)
- comment quality assessment (documentation)
- structure analysis (maintainability)
- formatting checks (consistency)
- dead code identification (cleanliness)

Great code quality reviews don’t just find bugs — they make code easier to read and maintain.

Build Better AI Systems

Subscribe for advanced prompt engineering, AI coding tools, code review frameworks, and practical strategies for developers and engineers.

Carefully engineered prompts for people doing real work.

Share this:

- Share on Facebook (Opens in new window) Facebook
- Share on X (Opens in new window) X

See also The Performance Reviewer