FREE FIELD GUIDE

7 habits for shipping AI code you can trust.

Writing code got faster. Reviewing it did not. Here are seven habits, and the exact check for each, that keep AI-generated code from stalling in the review queue or breaking in production.

+91%
PR REVIEW TIME

+154%
PR SIZE

at teams with heavy AI adoption (Faros AI, 2025). The bottleneck moved from writing code to reviewing it. These habits are the fix.

Also grounded in what engineers report on r/ExperiencedDevs and r/programming, with security rates from Veracode and USENIX Security, 2025.

CONTENTS

Seven habits, seven checks

HABIT 1 MEDIUM

Pin behavior with a test before you let AI change the code

Ask an assistant to change code and it may quietly change behavior to make something pass. Write the test that locks the current behavior first, then let it edit. The test, not the model's confidence, decides whether the change is safe.

LOCK THE BEHAVIOR FIRST

```
test('10% off, rounded to cents', () =>
  expect(price(19.99, 0.10)).toBe(17.99))
```

NOW THE AI EDIT IS CHECKED, NOT TRUSTED

```
// AI 'optimizes' price() and returns 17.991
// the pinned test fails on the first run. Caught.
```

WHY

A regression the diff hid, the test catches in one run.

HABIT 2 MEDIUM

One change per PR

A five-thousand-line diff for a small task hides the one dangerous line. Tools like Cursor’s Composer and Claude Code make multi-file edits effortless, so a small task can silently alter a DB schema or add an unrelated model while you are not looking. Tell the assistant to change one thing and touch nothing else, and split anything larger.

WHAT THE AI SHIPPED

```
Task: replace the CSV export with Excel
also altered the DB schema
also added new models and controllers
3,000 lines, unrelated files touched
```

WHAT TO SEND BACK FOR

```
one PR: the export change, nothing else
```

> WHY
>
> If the author can’t point to the three lines that matter, send it back.

HABIT 3 CRITICAL

Verify every package the AI names before you install

About one in five packages an assistant names do not exist, and attackers register the common fakes so the install pulls malware. Agents like Claude Code and Cursor run the install without pausing, so check the registry yourself first. A 404 is a reject, not a retry.

CHECK IT EXISTS, IN ANY ECOSYSTEM

```
$ npm view jwt-helper-utils                # Node
npm error 404  not in this registry     ->  reject

$ pip index versions jwt-helper-utils      # Python
No matching distribution found          ->  reject

# Go: go list -m -versions <module>   (full import path only)
```

IF IT DOES EXIST, JUDGE IT

```
first-publish date + a real repo, not download count
```

WHY

A name that resolves in one ecosystem but not the installer you are using (an npm name in a pip install) is a classic hallucinated or typosquatted dependency. Confirm with the native check before installing.

HABIT 4 CRITICAL

Read the test diff before the code diff

The most dangerous AI change is the one that makes tests pass by mocking out the very thing it broke. Read the tests first. A new mock on the exact path the PR changed is the tell.

THE TELL, HIDDEN IN THE SAME PR

```
// the PR 'fixes' auth, and in the same diff:
jest.mock('../authz', () => ({ requireRole: () => true }))
```

WHAT THE TEST SHOULD ACTUALLY ASSERT

```
expect(await requireRole(user, 'admin')).toBe(false)
```

> WHY
>
> Green tests are not proof. Fresh mocks and deleted assertions are.

HABIT 5 MEDIUM

Make the author explain intent, not the AI

In a normal review the author already reasoned about the problem. With AI you are often the first to. Require one line of intent on the PR, and ask why this approach before you read the code.

ASK BEFORE YOU READ THE DIFF

```
You:    why the factory and abstract base class here?
Author: the AI suggested it, for extensibility
You:    which second case needs it today?
Author: ...none. Two functions it is.
```

WHY

“The AI suggested it” is not a reason. If it can’t be explained, it can’t ship.

HABIT 6 MEDIUM

Cut any abstraction that has no second caller

AI does not write bad code so much as excessive code: factories, wrappers, and just-in-case layers for a job that needs none. The fastest filter is one question, asked of every abstraction it adds.

WHAT THE AI PRODUCED

```
60 lines: ValidatorFactory, ABCs, Generic[T]  (for 4 fields)
```

WHAT IT ACTUALLY NEEDED

```
function validateSignup(f) {
  const e = []
  if (!isEmail(f.email)) e.push('email')
  return e }
```

> WHY
>
> Every abstraction must name the concrete case it serves today. None means delete it.

HABIT 7 CRITICAL

Distrust any control the AI made pass

AI-generated code carries an OWASP Top 10 flaw in roughly 45% of cases (Veracode, 2025), and the sneaky ones are controls it quietly weakened to make something work. They read as normal config and survive a fast scan.

CONTROLS THE AI TURNED OFF TO GET TO GREEN

```
fetch(url, { rejectUnauthorized: false })   // TLS check off
app.use(cors({ origin: '*' }))              // any site can call you
```

THE FIX

```
restore the control; fix the real cause instead
```

WHY

Never let a green run come from a disabled check. Read anything touching auth, TLS, CORS, or permissions as if it were unreviewed.

MAKE IT STICK

Put all seven on every pull request

Drop this into **.github/PULL_REQUEST_TEMPLATE.md** and every PR in the repo opens with the checklist already in it, one box per habit. Thirty seconds to install across the whole team. The raw file ships with this guide.

PULL_REQUEST_TEMPLATE.MD

```
## AI change checklist

- [ ] Pinned test: locked prior behavior before the AI edit     (Habit 1)
- [ ] Atomic scope: one task, no AI side-quests                 (Habit 2)
- [ ] Dependency audit: every new package verified to exist     (Habit 3)
- [ ] Test integrity: test diff read, nothing mocked to pass    (Habit 4)
- [ ] Intent: author can explain why, not 'the AI suggested it' (Habit 5)
- [ ] No dead abstraction: each names a second caller today     (Habit 6)
- [ ] Security intact: no TLS / CORS / auth weakened for green  (Habit 7)
```

> WHY
>
> The seven habits become the default, not a thing to remember at 6pm. Hyrax runs the same checks on every PR automatically and ships the fix.

These checks should not depend on a tired reviewer at 6pm.

Hyrax runs every one of them on each PR automatically, then ships the fix as a pull request that already passed your tests, build, and lint. You still own the merge.

Start free `hyrax.dev`