the unofficial

AI Code Diagnostic Checklist

⚒ AI Craftspeople Guild • The first rung • Do you have a problem worth solving — or are you already fine?

Signs you should book one

A nagging sense that the AI-assisted parts are healthier on the dashboard than in reality. If several ring true, a diagnostic is cheap insurance.

- [ ] Tokens spent where a simpler, cheaper tool would do
- [ ] Tests that always pass — and you're no longer sure they'd catch a real break
- [ ] Dependencies you didn't knowingly add
- [ ] Agents whose permissions no one can fully account for
- [ ] A security/governance dashboard that's green — but never independently checked
- [ ] A growing unease you can't articulate sharply enough to justify a full audit
- [ ] You've been told "you're fine" by someone with something to sell

TOP PRIORITY — CHECKED FIRST

Token Efficiency Assessment

Token spending is a primary concern in its own right — not a footnote to code review. Every unnecessary token is a future tax on every interaction: compounding cost, compounding latency, compounding context pollution. Wasteful token patterns are often the earliest, clearest signal a practice has drifted from craft to cargo cult.

- [ ] Prompt design — verbose, redundant, or repeated prompting
- [ ] Context window discipline — what's loaded vs. what's needed
- [ ] Model selection relative to task complexity
- [ ] AI invoked where a simpler, deterministic tool would suffice

What the diagnostic covers

Breadth over depth. We look in the drawers most likely to hold what a customer, regulator, or acquirer would find first. Token efficiency is assessed first; code quality follows.

- [ ] **Token efficiency** — the top-priority pass (above)
- [ ] **How AI is actually being used**
  - [ ] Which agents, models & workflows are in play
  - [ ] What proportion of the code is AI-authored
  - [ ] What review sits between generation and merge
  - [ ] The real practice — not the policy on paper
- [ ] **Obvious risk surfaces**
  - [ ] Hallucinated dependencies
  - [ ] Over-permissioned agents
  - [ ] Tests that confirm rather than challenge
  - [ ] Security defaults the agent never hardened
- [ ] **Agent & tool configuration**
  - [ ] A quick read of prompts, tools & permissions
  - [ ] Over-permissioning & runaway autonomy

What it covers (continued)

- [ ] **Structural smell test**
  - [ ] A whole-system glance for agent assistance accumulating with no architectural hand on the tiller
  - [ ] Contradictory patterns & redundant abstractions
  - [ ] Dead code & drifting conventions
- [ ] **Tooling & dashboard reality check**
  - [ ] Is your existing tooling configured to enforce anything meaningful?
  - [ ] Or just producing reassuring dashboards?
  - [ ] How much "we're covered" is unverified on inspection
- [ ] **Verdict & recommendation** — the deliverable
  - [ ] A short, candid written summary
  - [ ] What we saw, what concerned us, what didn't
  - [ ] Which rung we'd recommend — and which we wouldn't

What you walk away with

One working session + a surface scan. The deliverable is an honest verdict, not a sales pitch for the next rung.

- [ ] A read on whether tokens are being spent wastefully
- [ ] A candid written summary of what was seen
- [ ] A clear answer: is a full audit warranted? …an optimization pass? …continuous assurance?
- [ ] Or: "you're fine — keep doing what you're doing"
- [ ] One problem, or fifty — a sense of scale before you commit

Who it's for

Five situations where the cheapest rung earns its keep.

- [ ] You suspect something is off — but can't sharpen the concern enough to justify an audit
- [ ] You need to justify an audit — give leadership a defensible, independent "do we need this?"
- [ ] You're new to AI at scale — an outside eye before patterns harden into expensive habits
- [ ] You're buying/inheriting a codebase — is it in the shape its sellers claim?
- [ ] You want a second opinion — from the party with nothing to sell but its accuracy

What it is NOT

A diagnostic is not an audit. Knowing the difference keeps expectations honest.

- [ ] Not comprehensive — we don't open every drawer
- [ ] Not a documented finding for every surface
- [ ] Not remediation — it doesn't fix anything
- [ ] **Audit asks:** "what is wrong with this codebase?"
- [ ] **Diagnostic asks:** "is something wrong enough that we should ask the first question?"

The Guild position

- [ ] This is the only Guild service whose ideal outcome is "you don't need any more of our services." The cheapest rung exists so no one climbs the rest unnecessarily. Honest answers at the bottom are how the rest of the ladder earns its credibility.

The first rung = lowest cost, lowest commitment • **Unnecessary token** = a future tax on every interaction: compounding cost, latency & context pollution • **Hallucinated dependency** = a package the AI invented or wrongly added • **Over-permissioned agent** = an agent granted more access than its task needs

⚒ **AI Craftspeople Guild** | Vendor-neutral by charter — we assure against open standards, we sell no proprietary tooling | aicraftspeopleguild.github.io | v1.1

This checklist is a **discussion tool, not a compliance scorecard.** The first rung of the Guild service ladder. Inspired in form by Henrik Kniberg's "unofficial Scrum Checklist."

AI Code Diagnostic Checklist — How to use it

What is this? Who is it for?

Before you commit to an audit, you should know whether you need one. This is a self-assessment for that moment — a way to decide whether the unease about your AI-assisted codebase is real or imagined, and whether deeper review is warranted. **Token efficiency comes first:** wasteful token spending is the earliest signal that an AI-assisted practice has drifted, and every unnecessary token is a future tax on every interaction. Run it on yourself first; if the warning signs ring true, that's when an outside diagnostic pays off. These aren't rules. They are guidelines. A small, low-stakes project may rationally ignore most of this — as long as you decided that *on purpose*, having looked, rather than because no one looked. That judgment is the whole point.

How do I use it?

Use it as a **gut-check** before spending money on a full audit — alone, or with your team.

Maya: "Do we actually need a full audit, or are we being paranoid?"

Dev: "Looking at the signs… we're burning tokens on prompts a plain script could handle, our tests always pass and I don't trust that anymore, and nobody can tell me what our agent is allowed to touch."

Maya: "Wasteful token spend plus two more warning signs, all real. That's not paranoia — that's a reason to look closer, starting with how the tokens are going."

Dev: "But a full audit is a big spend to justify on a hunch."

Maya: "Right — which is exactly what the diagnostic is for. It's the cheap way to find out whether the expensive thing is warranted. One session, an honest verdict, then we decide."

That's the right use: turning a vague unease into a clear yes/no on whether deeper work is justified.

Why a diagnostic before an audit?

A full audit is an investment. The diagnostic is the cheapest, lowest-commitment way to find out whether that investment is warranted — and it gives leadership a defensible, independent answer to "do we actually need this?" before the budget is committed. It answers *"is something likely wrong enough to ask the first question?"* — not *"what is wrong?"*

Why is token efficiency the top priority?

Because token spending is a primary concern in its own right, distinct from code quality — and it is checked first. Every unnecessary token compounds: cost, latency, and context pollution multiply across every future interaction. Catching wasteful patterns early — over-verbose prompts, the wrong model for the task, AI invoked where a simpler tool would do — is the cheapest leverage on the ladder.

How do I *NOT* use it?

Don't mistake it for the audit itself.

Big Boss: "We ran the diagnostic and it was clean. Tell the board we're fully audited and secure."

Dev: "The diagnostic was a surface scan. It said deeper review wasn't urgent — it didn't examine everything."

Big Boss: "Close enough. A clean diagnostic is a clean bill of health. Done."

Maya: "It isn't. A diagnostic answers 'should we ask the first question.' An audit answers 'what's actually wrong.' Reporting one as the other is how you get surprised later."

The diagnostic is breadth, not depth — a small number of high-signal probes, not a comprehensive examination. Treating its verdict as audit-grade assurance is exactly the overconfidence it exists to prevent.

Is this an official standard?

No. It's a practitioner's gut-check, reflecting the Guild's view of the signals worth noticing early. Argue with it, adapt it to your context, experiment. If it turns a vague worry into a clear decision, it's done its job.

The one principle

The ideal outcome of a diagnostic is finding out you don't need one. Honest answers at the bottom of the ladder are how the rest of it earns its credibility.