the unofficial

AI Code Audit Checklist

⚒ AI Craftspeople Guild • What to scrutinize when software is built with AI agents

The bottom line

If the codebase passes these, the speed didn't cost you what it usually costs. Start here.

- [ ] The system **fails safely** — errors are handled, not silently swallowed
- [ ] It actually does **what the sales page / spec claims**
- [ ] Every external call and dependency reference **points at something real**
- [ ] Sensitive data is **handled with care**, not over-fetched or over-logged
- [ ] The test suite **verifies behavior**, it doesn't just inflate coverage numbers

Core Audit

Central to auditing AI-assisted code. Skip these and you haven't really audited it.

- [ ] **Hallucination detection**
  - [ ] API methods & library functions verified to exist
  - [ ] Configuration keys checked against authoritative sources
  - [ ] No "phantom code" that fails silently in production
- [ ] **Test coverage integrity**
  - [ ] Tests **challenge** the output, not merely confirm it
  - [ ] No "test theatre" — coverage without genuine verification
  - [ ] Edge cases the agent never considered are covered
- [ ] **Security vulnerability review**
  - [ ] Input validation is adequate, not assumed
  - [ ] Permissions are scoped, not over-broad
  - [ ] No insecure defaults or outdated auth patterns
  - [ ] Audited against OWASP Top 10 + AI-specific threats
- [ ] **Architecture coherence**
  - [ ] Assessed from a **whole-system view**, not file by file
  - [ ] No contradictory patterns from local-only optimization
  - [ ] Redundant abstractions & structural debt identified

Core Audit (continued)

- [ ] **Data handling & privacy**
  - [ ] No over-fetching of sensitive fields
  - [ ] No unnecessary logging of sensitive data
  - [ ] Access control is sufficient
  - [ ] Data retention won't create regulatory exposure
- [ ] **Error handling & resilience**
  - [ ] Code is not happy-path only
  - [ ] No failures silently swallowed
  - [ ] System degrades gracefully under adverse conditions
- [ ] **Dependency & license audit**
  - [ ] No outdated or vulnerable package versions
  - [ ] No known-vulnerability dependencies
  - [ ] Licenses compatible with commercial use

Recommended

Usually needed, but depends on the system. Apply judgment — experiment.

- [ ] **Agent configuration review** *(if AI agents run inside the system)*
  - [ ] Agents are not over-permissioned
  - [ ] Prompt-injection exposure assessed
  - [ ] Runaway-autonomy risks bounded
  - [ ] Tools & permissions granted to agents reviewed
- [ ] Audit performed by someone **independent** of the build
- [ ] Reviewer works with AI agents in production daily
- [ ] Findings written down — a record, not just a conversation
- [ ] A clear **remediation priority** assigned to each finding
- [ ] Re-audit scheduled after the next major AI-assisted sprint

When to audit

The moments where independent scrutiny pays for itself many times over.

- [ ] **Before a production launch** — before real users, real consequences
- [ ] **Before a funding round or acquisition** — surface issues before due diligence does
- [ ] **After a major AI-assisted sprint** — catch drift before it compounds
- [ ] **When onboarding a new team** — an honest map, not inherited assumptions
- [ ] **After something has gone wrong** — separate human error, agent error & structural failure

Positive indicators

Signs of a healthy AI-assisted practice — not audited, but worth noticing.

- [ ] The team can **explain why** the code is the way it is — not just that it works
- [ ] Someone **owns accountability** for the output — no "the model said so"
- [ ] A green build is **trusted** as a real signal of safety
- [ ] Refactoring happens routinely, not as a rare and risky event
- [ ] The team treats AI as an instrument they **direct**, not an oracle they obey
- [ ] Speed is celebrated **and** verified — both, not either

The Guild position

- [ ] An audit tells you what your codebase **actually contains** — not what you hope it contains. Independence is the service: no stake in the outcome except the quality of the work.

AI agent = autonomous coding assistant • **OWASP** = Open Worldwide Application Security Project • **Test theatre** = tests that show coverage without verifying behavior • **Hallucination** = AI-invented API, function, or config that does not exist

⚒ AI Craftspeople Guild

AI Code Audit Checklist — How to use it

What is this? Who is it for?
A simple tool to help you scrutinize software built with AI agents — or assess an audit you're about to commission. Building with AI is fast. Dangerously fast. Speed hides problems, and this checklist is a map of where those problems tend to hide. **These aren't rules. They are guidelines.** A small internal tool with no sensitive data might reasonably skip the privacy and dependency-license sections. Fine — as long as you skipped them *on purpose*, having understood the risk, rather than because nobody looked. That judgment is the whole point.

How do I use it?

Use it as a **discussion tool**, ideally before a launch, a raise, or right after a heavy AI-assisted sprint.

> **Maya:** "Before we ship, is there anything on this list we haven't actually checked?"
>
> **Dev:** "We've got tests passing… but I'm not sure they *challenge* the code. They mostly assert what the agent already produced."
>
> **Maya:** "That's 'test theatre' — it's in Core Audit, so it matters. Green doesn't mean safe yet."
>
> **Dev:** "And I never verified that the third-party API methods the agent used actually exist. It invented one last month."
>
> **Maya:** "Hallucination detection. Also Core. Let's hold the launch one day and check both."

That's the right use: surfacing what nobody looked at, then deciding — deliberately — what to do about it.

Why AI-built code needs a different audit

Traditional review assumes a human wrote every line with intent. AI breaks that assumption. An agent can produce code that is syntactically perfect, passes linting, even passes its own tests — and still contains a kind of failure no experienced engineer would have written. That's a new category of risk, and it needs a new category of scrutiny.

How do I *NOT* use it?

Don't turn it into a compliance weapon.

> **Big Boss:** "Run the whole checklist on every repo and report a compliance percentage to me by Friday."
>
> **Dev:** "We're at 92%. We skipped the agent-configuration section because we don't run agents inside this product."
>
> **Big Boss:** "92% is a fail. It's on the list, so I want 100%. No exceptions!"
>
> **Maya:** "But that section doesn't apply to this system. Forcing it would just generate noise and waste a day."
>
> **Big Boss:** "The list says do it, so do it, or I'm calling in the AI Police."

A checklist used as a scorecard produces theatre — exactly the thing the audit exists to prevent. The number is never the point. **What the codebase actually contains** is the point.

Is this an official standard?

No. It reflects the Guild's craft-grounded, subjective view of what really matters when software is built with AI agents. It's meant to be argued with, adapted, and experimented on. If it sparks the right conversation before the wrong code ships, it's done its job.

The one rule

You cannot offload accountability to a model. Trust needs proof — and proof has to be checked.