Pre-launch checklist for AI agent systems

What quietly falls apart in production

AI agent systems fail differently from ordinary software. The failures are quiet: a test goes green, a demo works, a safety gate exists on paper, and everything looks fine until it doesn't.

Every item below was caught while I built and hardened my own self-hosted agent platform in May and June 2026. Each one has a date, what happened, and a check you can run on your system this week.

01 A passing end-to-end test can hide a dead feature.

A test can exercise the easy internal logic and skip the wiring that connects it to production. Every assertion is green while the headline feature never fires for a real user.

CAUGHT 4 June 2026. A 7-test end-to-end suite was green while the one-tap Approve/Reject step never reached the operator. A pre-merge code review found it.

CHECK Delete the production glue between two components. If no test goes red, the tests are decorative.

02 A safety gate can fail open, silently.

A gate meant to block risky actions can be built so that a missing dependency lets the action through instead of stopping it. From the outside this looks exactly like "working".

CAUGHT 4 June 2026. The production build had no policy engine, the guard returned early, and a REJECT decision still ran.

CHECK On the real running system, prove that a rejection actually stops the action.

03 "Automatic recovery" can mean recovery from graceful failures only.

Crash-recovery tests often simulate failure by raising an exception. A real process kill (crash, out of memory, power loss) can still lose work in flight.

CAUGHT 20 June 2026. Work resumed correctly after a raised exception, but a real SIGKILL lost the in-flight result and every side effect ran again.

CHECK Kill the process for real instead of throwing an error inside it.

04 A configured timeout may not time anything out.

Some libraries accept a timeout setting and ignore it under certain conditions. A call you believe is bounded can hang forever and wedge the whole system.

CAUGHT 27 May 2026. An LLM client library accepted timeout=4 and hung for more than 35 seconds against a stalled endpoint. About 40% of runs froze.

CHECK Point the call at something that accepts the connection and never answers.

05 One hung external call can freeze a whole multi-step run.

Without isolating each step, a single unresponsive dependency freezes the entire job instead of failing just that piece.

CAUGHT May 2026. One stalled model call froze 12-minute runs until a watchdog killed them. Under load, about a third of calls to a hosted model hit the 180-second cap.

CHECK Can one step fail, retry or be skipped without losing the work already done?

06 Small data mismatches can silently break what gets stored or retrieved.

A timezone mismatch, an unexpected wrapper around model output or the wrong similarity metric can make writes fail or retrieval return the wrong thing, with no error.

CAUGHT 25 May 2026. A timezone-aware vs naive column made memory writes skip silently. L2 distance instead of cosine broke retrieval. Code wrapped in markdown fences failed tests and looked like a quality problem.

CHECK Look at what was actually stored and retrieved, not just at “no exception was raised”.

07 A test can be built so it cannot fail.

A robustness test can be structured so the failure it should catch is unreachable. Then a green result means nothing.

CAUGHT 30 May 2026. A noise-robustness probe was green because its distractor set had saturated, so the failure it was meant to catch could not happen.

CHECK Before trusting a test, make it produce the bad outcome once.

08 Identical code can behave differently depending on where it runs.

A dependency can change its output based on environment variables it detects. A check that passes on a laptop can fail, or silently misbehave, in CI or production.

CAUGHT 29 June 2026. A test passed locally and failed in CI because pytest changes its output when it detects CI (CI=true).

CHECK Diff the real output across environments before calling a test flaky.

09 More retrieved context can make the system worse.

Extra examples or memory are not strictly helpful. They can distract the model from a task it would have solved on its own.

CAUGHT 29 May 2026. On one task family, retrieved memory dropped the success rate from 1.00 to 0.33. The prior distracted the model from an easy task.

CHECK Test whether added context hurts the tasks the system already handles well.

10 One model writing the task, the answer key and the grade invites confident wrong verdicts.

When the same model, or the same model family, writes the test, the expected answer and the evaluation, it disagrees with itself on edge cases.

CAUGHT 27 May 2026. A model-generated re-check used a different edge-case contract and failed a correct solution.

CHECK Anchor grading to ground truth that was verified independently.

11

When the same kind of fix keeps coming back, stop patching and measure.

Repeating similar fixes (raise the timeout, raise it again) burns hours while the real cause sits one layer deeper.

CAUGHT 27 May 2026. Three wrong guesses and repeated timeout changes before a stack dump showed socket reads with no timeout at all.

CHECK After the same class of fix fails twice, get real logs, traces and a minimal reproduction.

None of this is exotic. It is ordinary engineering discipline applied to systems that fail quietly instead of loudly, which is why it is easy to miss until production tells you.

Want a second set of eyes on your AI system?

An express audit of one system takes 3 working days, from $2,000. The first 30-minute call is free.

rostyslav.stekh@gmail.com · linkedin.com/in/rostyslav-stekh