Why do I feel responsible enough to be blamed, but not in control enough to feel safe?

Diffusion of Responsibility
AI-assisted coding can split responsibility across the developer, the AI tool, the team, management pressure, delivery targets, and fear of replacement. The developer is pushed to trust outputs they did not fully create, meet speed expectations they did not fully choose, and defend results they did not fully control.

PTSD/PTBS Risk
In military settings, this kind of responsibility split can contribute to **PTSD/PTBS-related stress patterns** when action is distributed across command chains, teams, systems, tools, weapons, and orders, but the individual still feels morally exposed to the outcome. The closest clinical frame is **moral injury**: guilt, shame, betrayal, loss of agency, and loss of trust after someone feels involved in harm without clear control. Source: VA National Center for PTSD. (ptsd.va.gov)

Core Insight
When responsibility is distributed but consequence feels personal, the nervous system does not experience “support.” It experiences exposure.

Practical Prevention: Protect the Developer’s Brain, Not the AI Workflow

- **Bounded responsibility:** define what the developer owns, what the tool owns, what management owns.
- **Permission to distrust:** make “I do not trust this AI output” an acceptable engineering statement.
- **Blame boundary:** no individual blame when the system pressured AI use.
- **Control map:** before AI-heavy work, name what is under developer control and what is not.
- **AI after-action review:** after failures, separate tool error, human review error, process pressure, deadline pressure.
- **Authorship anchor:** developer must make one explicit human design decision before merge.
- **Manager burden ownership:** if management pushes AI adoption, management owns part of the psychological and quality risk.
- **Team trust review:** talk with the team about situations where AI output is not trusted: generated architecture, tests, edge cases, security-sensitive code, refactors, unfamiliar libraries, vague prompts.
- **AI distrust map:** collect recurring “I don’t trust this” moments and turn them into visible categories: hallucinated APIs, fake confidence, hidden assumptions, brittle tests, wrong abstractions, silent security risk.

- **Wall of AI Shame:** public team board for AI failures, not developer failures. What did AI produce? Why did it look trustworthy? What was actually wrong? How did we catch it? What rule prevents this next time?
- **Purpose of the wall:** remove private shame from the developer and put the pattern on the system.
- **Core line:** We do not shame the person who trusted AI. We expose the situations where AI becomes easy to trust and dangerous to trust.