Co-Designing a Trust Calibration Checklist for AI Practitioners: A Case Study on AI stakeholders

Mariafrancesca Betta
s339890@studenti.polito.it

Riccardo La Leggia
s339244@studenti.polito.it

Tea Peyron
s321854@studenti.polito.it

Giovanni Caroni
s321941@studenti.polito.it

Hasti Naderi
s321897@studenti.polito.it

Giovanni Salafia
s335792@studenti.polito.it

Adriano Danni
s322117@studenti.polito.it

Raffaele Pansa
s310647@studenti.polito.it

Giulia Ingino
s338279@studenti.polito.it

Cristian Preutesi
s337000@studenti.polito.it

Abstract

In professional fields where safety is crucial, it's important that workers trust AI systems correctly to avoid serious accidents.
The problem: Trust in AI output is often miscalibrated. In real world jobs, dangerous risks emerge from a "System Reliability Gap". But "Why do we have this gap?"
This gap happens because workers do not have the ability to fully know what AI can or cannot do. This situation led them to trust the AI too much or too little. Right now, there are very few practical tools to monitor these trust errors in real time.
Proposal: This paper introduces an Impact Assessment Template to solve this problem:
We created the first version of a checklist from the scoping review of Deliverable 1 (Checklist V1) then we added multiple co-design interviews with experts (such as: software developers, a technical Product Owner, a Quality Assurance (QA) Lead) to produce Checklist V2 and Checklist V3
At the end we prove that this checklist works through a real-world case study on an AI pedestrian detection system (the use case).

1. Introduction

For the successful integration of Artificial Intelligence (AI) in daily professional life, the proper calibration of trust is as important as the effectiveness of the algorithms themselves. "Trust calibration" is crucial in understanding the dynamics between humans and AI systems: improper calibration can lead to either overtrust — excessive confidence in an unreliable system with potential safety consequences — or undertrust, which results in delays, inefficiencies, and the rejection of valuable tools.

This study addresses a gap in the literature: while most research examines trust from the end-user perspective (the operator, the diagnostician, the driver), very few works consider the perspective of the practitioner who selects, configures, and governs AI systems within an organisation. Such practitioners are not merely users; they shape the conditions under which AI is adopted and relied upon. Improper trust calibration at this level can result in deploying unsafe or unreliable AI at scale.

This project aims to answer the specific "Type 3" deliverable prompt: Is trust in AI warranted for this task, and how should that trust be calibrated? To connect academic theory with practice, an iterative co-design approach was taken to develop a Trust Calibration Checklist across three successive versions (V1, V2, V3).

2. Methodology

In many organisations, structured approaches to AI ethics follow a common pattern: a checklist is produced to capture relevant principles, and a report is written on its basis. Few of these tools are developed with genuine input from the practitioners who will actually use them — and when tools are introduced without that involvement, they tend to be superficially completed or set aside. Our process combined an exploratory literature-derived prototype with two rounds of structured co-design interviews.

2.1 Checklist V1 — Literature-Based Prototype

Checklist V1 was derived systematically from our Scoping Review framework [1], which identified six core dimensions of trust calibration in AI-assisted practice: Reliability & Performance, Transparency & Explainability, Social & Organisational Influences, Cognitive & Psychological Factors, Uncertainty Communication, and Institutional & Accountability Structures. Each dimension was translated into a set of actionable prompts, producing an initial set of open-ended items across all six sections.

Items were designed to prompt reflection rather than provide binary compliance checks. V1 was deliberately constructed as a comprehensive working prototype — thorough and traceable to the literature, but not yet shaped by the real-world constraints that practitioners and governance stakeholders would impose.

2.2 Checklist Research Questions

To explore how workers establish and calibrate their reliance on artificial intelligence, we operationalised our overarching research inquiry into four specific sub-questions that guided our data collection and analysis:

- RQ1: What signals do AI practitioners use, in real working conditions, to accept or question an AI-generated output, and how reliably do those signals track actual accuracy?
- RQ2: In what ways are AI outputs confidently misleading, and what verification practices enable practitioners to detect errors that surface plausibility would mask?
- RQ3: How do social and organisational factors — peer reputation, time pressure, hierarchical authority — distort the relationship between evidence and trust?
- RQ4: What structural mechanisms — logging, review chains, explicit accountability assignment — are necessary to sustain appropriate trust over time and across roles?

These sub-questions map directly onto the five interview protocol themes.

2.3 Participants

Five participants were recruited across two rounds, spanning roles from hands-on technical work to project governance and quality oversight. This breadth was deliberate: trust miscalibration manifests differently depending on whether a person is building an AI-assisted system, approving its outputs, or certifying its quality. Table 1 summarises participant profiles and their primary contributions to the checklist.

ID	Role	Key contribution
Round 1 — AI-Intensive Practitioners (V1 → V2)		
P1	Programmer	Importance of manual cross-referencing; distrust of fluent-but-wrong outputs
P2	Back-end Software Engineer	Root-cause fixes vs surface patches; continuous confidence monitoring
P3	Engineering Manager	Security implications of AI configs; CI/CD audit trails; black-box logging
Round 2 — Governance & Quality Stakeholders (V2 → V3)		
P4	Technical Product Owner	Non-technical readability; per-section AI contribution record
P5	Quality Assurance Lead	Edge-case gaps; AI-assist tagging in version control for targeted review

Table 1. Participant profiles and main contributions to checklist development.

Round 1 participants (P1–P3) spanned practitioner and managerial roles in AI-intensive contexts. Round 2 (P4–P5) extended the sample to governance and quality roles — stakeholders who evaluate, approve, or certify AI-assisted outputs without necessarily being involved in their production. This contrast between insider and oversight perspectives was essential for surfacing requirements that Round 1 alone could not reveal.

2.4 Interview Protocol

Checklist V1 was sent to each participant several days before the interview, with a request to review it and note what felt useful, unclear, too long, or missing. Each session lasted approximately 20 minutes and followed a semi-structured guide, opening with warm-up questions on prior experience with AI tools before moving to a structured walkthrough of the checklist. Five themes guided each interview, ensuring that both technical and organisational dimensions of trust were consistently explored across all participants:

- Trust and distrust cues in practice — what signals, in real working situations, lead a practitioner to accept or question an AI-generated output?
- Encounters with confidently wrong AI — instances where an AI tool produced plausible-looking but incorrect output, and how that error was detected or missed.
- Influence of peer opinion and community reputation — how colleague recommendations, tool reviews, and organisational endorsements shape reliance on AI.
- Behaviour under time pressure — how verification and review practices change when working to tight deadlines, and what risks this introduces.
- Responsibility and protective mechanisms — who is held accountable when AI-assisted work goes wrong, and what processes — logs, reports, review chains — would support practitioners in that situation.

Sessions closed with an open co-design question asking participants to name the single most important element missing from V1 and V2. Feedback was logged using participant codes P1–P5.

3. Data Analysis

Thematic analysis of the five interviews surfaced the following five principal factors governing whether practitioners' trust in AI is well-calibrated. Each theme directly addresses one or more of

our four research questions and drove specific changes to the checklist across versions.

3.1 Active Verification Over Passive Acceptance

The most consistent finding across all five participants was that appropriate trust in AI requires active, evidence-based verification — not passive acceptance of AI-generated outputs, however fluent or authoritative they appear.

P1 (Programmer) described treating AI outputs as search results requiring independent source verification: *"I generally use AI outputs as search results. When ChatGPT tells me something and adds a link, I click on the link and read it myself to understand it completely."* This verification habit, P1 acknowledged, evaporated under time pressure — a key vulnerability identified across multiple participants.

P2 (Back-end Software Engineer) made the most technically precise articulation of the trust signal problem: *"I trust an AI tool more when its answer is clear, testable, and matches the project requirements. I distrust it when it gives a confident answer without explanation, ignores edge cases, or only hides the visible error instead of solving the root cause."* This distinction between surface repair and root-cause resolution became a central organising principle of the revised checklist (see Section 5.1).

P5 (QA Lead) provided a sharp characterisation of the verification threshold problem: *"AI is nothing more than a yes man. It will proudly provide you with its utterly hallucinated response."* In QA practice, P5 noted, trust rests specifically on whether developers can explain AI-generated code line-by-line during code review handover — a procedural trust signal not captured in V1.

Checklist implication: V2 introduced a mandatory independent review requirement before any AI-assisted output is accepted, with an explicit check for root-cause resolution rather than surface-level error removal. V3 extended this to a per-stage timing flag (B = Before, D = During, A = After) to ensure verification occurs at operationally appropriate moments.

3.2 Plausibility Does Not Equal Accuracy

All five participants independently raised the danger of structurally convincing outputs that are substantively wrong — what we term the plausibility-accuracy decoupling. This finding represents the most practically consequential insight of the study.

P3 (Engineering Manager) described a vivid episode: an AI assistant produced a detailed deployment cost model claiming a 40% overhead reduction. *"The entire calculation was just made up, and it got its pricing dimensions all wrong across various cloud regions. It took me a couple of minutes to notice because I had considerable experience with those vendors. It sounded really authoritative, but under the hood it was nonsense."* P3's key insight was that the protection against such errors is domain expertise, not checklist compliance — a finding that shaped our decision to frame checklist items as prompts for expert reflection rather than binary confirmations.

P4 (Technical Product Owner) articulated the oversight role's particular vulnerability: *"What worries me is when the AI comes up with output that is seventy or eighty percent correct and formatted in a way that gives it the impression of authority, and all of the errors are in fine print, detectable only by an expert."* P4 noted that non-technical reviewers — precisely the role responsible for final sign-off — are structurally unable to detect this category of error without explicit flagging by the producing developer.

Checklist implication: V2 added explicit items requiring that AI outputs display confidence levels and uncertainty indicators alongside results, and that explanations be calibrated to the audience's technical expertise. V3 restructured these into the Transparency and Explainability dimension with worked examples demonstrating what adequate uncertainty communication looks like in practice.

3.3 Reputation Does Not Replace Testing

All participants acknowledged the powerful influence of peer opinion and community reputation on their initial adoption of AI tools — and all expressed reservations about treating reputation as evidence of reliability.

P1 described the hype cycle effect: *"While Nathan was viral, everybody was talking about it. I was like, what a miracle! And when I checked it out myself, I saw it was not that magic I thought."* This pattern — enthusiasm leading to adoption, adoption leading to disappointment — was described by all five participants and suggests a systematic overweighting of social endorsement relative to individual validation.

P3's conclusion: *"Hype cannot substitute our internal testing. While popularity may give an AI tool a chance to enter our organisation, we will adopt it only based on its own demonstrated value and security."*

Checklist implication: V2 introduced items requiring that AI outputs be evaluated independently of online reputation or peer endorsement, and that peer recommendations be treated as starting points for investigation rather than proof of reliability. These items were positioned under Social and Organisational Influences in V1, restructured for clarity in V2.

3.4 Time Constraints Increase Risk

Time pressure was identified by every participant as the primary situational risk factor for trust miscalibration. Under deadline pressure, verification practices collapse, review steps are skipped, and practitioners shift from active evaluation to passive acceptance.

P1: *"When I am in a hurry, I do not recheck AI results and I accept them directly. That is really dangerous."* P1 described a concrete episode: copying AI-generated code into a project under deadline, receiving errors, copying those errors back to ChatGPT, receiving wrong fixes again — a recursive trust trap that consumed more time than writing the original code would have taken.

P5 described the systemic version of this problem: *"The moment we have to do five feature tests in two days, the cognitive load becomes enormous. You want to click generate tests, see the checkmarks, and accept the deployment. This is where you abandon all exploration and just believe the machine."* P5 called this 'my biggest nightmare' — an honest acknowledgement that the professional most responsible for catching AI errors is also the professional most structurally vulnerable to missing them.

Checklist implication: V3 introduced a dedicated Stress and Time Management section, a dimension entirely absent from V1 and only partially addressed in V2. Items in this section require explicit assessment of whether time allocation allows adequate verification, and whether workload distributions can be structured to preserve verification capacity during peak periods.

3.5 Human Accountability Requires Logging

Across all interviews, the question of who bears responsibility when AI-assisted work goes wrong was answered consistently: responsibility falls on the human practitioner or organisation, never on the AI. But practitioners also reported that existing workflows offered them insufficient protection in practice — no structured record of which outputs were AI-generated, no audit trail linking AI decisions to their consequences, and no mechanism for post-incident reconstruction.

P2: *"I would feel more protected if the system produced a clear report showing the input data, AI decision, confidence level, AI reasoning, time, and any human override action. In a pedestrian detection context, this works like black box logging."*

P4 introduced the concept of a formal AI Contribution Record — a per-section annotation of what was AI-generated, what was human-edited, and what sources were used: *"On one hand, reviewing will be faster because I will know exactly where the problems lie. On the other hand, it will enable accountability in the future. When an error arises, it will be extremely easy to track it down."*

P5 proposed an operationally immediate implementation: an 'Assisted by AI' tag in version control, attached to pull requests, enabling QA reviewers to direct heightened scrutiny precisely to AI-generated code segments: *"This allows my QA team to pay special attention to that particular part of the code for any possible bugs or other problems related to the weird decisions made by the AI."*

Checklist implication: V3 introduced mandatory structured logging and the AI Contribution Record as required elements of the Institutional and Accountability Structures dimension. These requirements are not optional items to be completed if organisational culture supports them; they are presented as preconditions for any claim that trust in an AI system is well-calibrated.

4. Checklist Evolution: V1 to V2 to V3

4.1 From V1 to V2: Operationalising Practitioner Experience

V1's 24 open-ended items spanned six dimensions but were operationally abstract. Three structural gaps, surfaced in Round 1 interviews, drove the redesign into V2.

Limitation 1 — Root-cause versus surface-fix distinction was absent. V1 contained no item distinguishing an output that removes a visible error from one that resolves the underlying fault. P2 put it directly: "I distrust it when it only hides the visible error instead of solving the root cause." V2 introduced the dedicated check "Has the team verified that AI-generated solutions solve the root problem rather than masking errors?", which became V3 item A3 (B marker).

Limitation 2 — Accountability items were too abstract. V1 asked 'How should responsibility for AI outcomes be shared between systems and institutions?' — a governance philosophy question with no operational correlate. P3 identified the concrete equivalent: "an effective black box logging process and an automatic compliance report" capturing input data, confidence parameters, and peer review records. V2 replaced the abstract question with binary checks ("Are logs maintained for inputs, outputs, confidence levels, and human overrides?"), inherited in V3 as items D3 and D4 with A (After) timing markers.

V2 kept V1's six dimensions but replaced open-ended prompts with binary items (Yes/No, Fully/Partially/Not at all) and 1–5 rating scales, adding B/D/A timing markers throughout. The format shift from reflection to structured response enabled comparison across contexts and made items directly actionable in real workflows.

4.2 From V2 to V3: Governance Legibility and the Missing Dimension

Two further gaps emerged from Round 2 interviews.

Limitation 3 — Non-technical stakeholders could not use V2 independently. Items referencing CI/CD audit trails or confidence variance thresholds presupposed technical literacy. P4 identified the core requirement: "trust for me is not defined by the ability to compile the code, but rather in the developer's ability to justify the code written by AI in such a way that it is understandable to the non-technical client." V3 revised all items to plain language and embedded concrete worked examples into each entry (e.g. V3 item B4 pairs its criterion with "Instead of displaying raw algorithmic telemetry, the system uses easily comprehensible visual cues"), making the non-technical review criterion explicit.

Limitation 4 — Time pressure had no dedicated dimension. V2 buried cognitive risk within the Cognitive & Psychological Factors section. P5 captured what those isolated items failed to address: "The moment we have to do five feature tests in two

days… you want to click generate tests, see the checkmarks, and accept the deployment." P4 described the same failure independently. V3 introduced a dedicated Section C (Stress and Time Management) with four standalone items (C1–C4) covering time adequacy, workload structure, deadline realism, and stress-induced automation bias — absent entirely from V1 and V2.

Limitation 5 (from P2) — Abstract items lacked decision triggers. V1 asked, for example, 'In which situations is the AI system most likely to fail?' with no link to any workflow checkpoint. P2 noted that verification must be actionable at a defined stage: "I trust an AI tool more when its answer is clear, testable, and matches the project requirements." V3 attached timing markers to every item (B = Before, D = During, A = After)— for example, item A3 ("Verify that the AI genuinely resolves the root problem") carries a B marker, anchoring it to the acceptance decision.

Revision 6 — AI Contribution Record formalised. V2 had no mechanism for annotating which outputs in a deliverable were AI-generated. P4 framed the accountability risk: "If the developer does not inform me about the use of AI and I approve unaware of its genesis, is the liability shared then?" The remedy proposed was a per-section record of what was AI-generated, what was human-edited, and what sources were used. P5 proposed a parallel code-level measure: an 'Assisted by AI' tag on pull requests, allowing QA reviewers to direct scrutiny precisely to AI-generated segments. V3 formalised both as a governance requirement in Section D.

V3 thus added Section C (Stress and Time Management) and the AI Contribution Record in Section D, consolidated redundant binary items, and embedded concrete worked examples in every entry — producing a four-section instrument (A–D) usable by both technical and non-technical reviewers.

5. Evaluating the co-designed checklist

Six dimensions from the Deliverable 1 scoping review were operationalised into actionable checklist items through two co-design rounds. Each dimension maps to one or more of the five interview themes:

- Reliability & Performance: the checklist now targets root-cause resolution and failure conditions rather than average performance metrics, driven by Theme 2.
- Transparency & Explainability: items assess whether explanations are present and whether they are interpretable by the intended audience, addressing the risk that authoritative-looking output creates false confidence (Themes 1 and 2).
- Social & Organisational Influences: items require independent validation regardless of peer endorsement or tool reputation, driven by Theme 3.
- Cognitive & Psychological Factors: a time-pressure risk flag and mandatory verification reminder were added, driven by Theme 4.
- Uncertainty Communication: all AI outputs must carry explicit uncertainty indicators, reformulated in plain language for non-technical stakeholders (Theme 2, V3 revision from P4).
- Institutional & Accountability Structures: black-box logging and the AI Contribution Record are mandatory requirements, driven by Theme 5.

The overall co-design workflow — from the initial literature-based framework through two rounds of structured interviews to Checklist V3 — is illustrated in Figure 1.

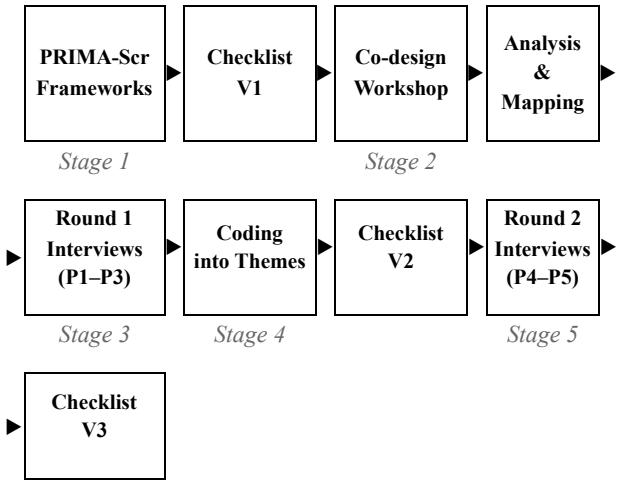

Figure 1. Co-design workflow from initial scoping framework to Checklist V3

6. Worked use case: AI Software Developer

To measure the effectiveness of the operational checklist, the checklist was applied to a critical sociotechnical situation where an AI Software Developer works with an AI-assisted coding system to build, debug, and maintain critical enterprise architecture. The system provides real-time code generation, architectural recommendations, and automated test scripts.

Using the Checklist V3:

Stage A - Reliability and Performance Consistent with Theme 2 (plausibility does not equal accuracy), the most dangerous failure mode for an AI coding assistant is not average inaccuracy but confident failure under specific edge conditions. The checklist requires that the system be tested in edge cases (A1): in this instance, that includes highly concurrent data loads, complex multi-tenant database schemas, and obscure legacy library dependencies. Prediction variance is tracked over extended development periods (A2). Critically, the checklist confirms that the developer ensures the AI-generated solutions solve the root problem (e.g., fixing a race condition) rather than merely suppressing visible error messages or masking underlying flaws (A3). A manual override is always available, allowing the developer to completely discard the AI's suggestions and write the logic independently (A4), which acts as the non-negotiable fallback required by the checklist.

Stage B - Transparency and Explainability The IDE interface was designed specifically to address the plausibility-accuracy decoupling. Rather than silently inserting code blocks, the tool

successfully explains the rationale, logic, or data points behind its suggested algorithms (B1). The interface visually varies with the system's uncertainty level (B2): highly confident syntax suggestions are rendered normally, but speculative logic utilizing external APIs is highlighted to signal low confidence, providing an ambient uncertainty signal without requiring raw probability scores. The developer guidelines explicitly warn that the AI can confidently hallucinate internal methods or database tables that do not exist in the actual schema (B3). Explanatory elements avoid cognitive overload during the debugging and decision-making process (B4).

Stage C - Stress and Time Management The software development context instantiates Theme 4 (time constraints increase risk) with particular acuity: sprint deadlines and production hotfixes are inherently time-pressured environments. The checklist's dedicated Stress and Time Management dimension directly addresses this. The workflow requires that users possess sufficient time to properly review AI outputs before finalizing decisions (C1); for example, CI/CD pipelines enforce a mandatory minimum review time for AI-assisted pull requests, structurally protecting the verification step from being compressed by operational urgency. Routine boilerplate generation is organized to prevent overreliance during busy periods, preserving the developer's attentional resources for high-stakes business logic implementation (C2, C3). A monitoring mechanism tracks whether the developer's commit pattern suggests automation bias—such as blindly accepting large blocks of AI-generated code under stress without manual testing—and flags this for QA review (C4).

Stage D - Accountability Structures Consistent with Theme 5 (human accountability requires logging), the engineering team's operational documentation explicitly states that legal and operational liability rests with the deploying organization and the developer, never on the AI (D1). The AI is treated as a support tool rather than a final authority; the system does not operate autonomously to push code directly to production without a human in the loop (D2). A black-box logging mechanism records all prompts, AI outputs, and the developer's subsequent edits within the version control system (D3), enabling full post-incident reconstruction of how and why an AI-assisted decision was made (D4). Furthermore, an 'Assisted by AI' tag in version control acts as an AI Contribution Record, implementing a formal accountability mechanism to direct heightened QA scrutiny precisely to AI-generated code segments (D5).

Conclusion for Trust Calibration: Applying Checklist V3 to the AI Software Developer yields a conditional trust recommendation: trust is warranted under conditions A1-A4, B1-B4, C1-C4, and D1-D5 are met. Specifically, trust is well-calibrated when (a) the AI's code is tested in edge cases and genuinely resolves root problems (Dimension A); (b) the developer can read confidence indicators in the IDE and understands the hallucination risks (Dimension B); (c) the developer is not subject to tight sprint deadlines that compress code review time below safe minimums (Dimension C); and (d) version control logs and AI Contribution Records are fully maintained and the accountability chain is unambiguously documented (Dimension D). Should any of these conditions fail—particularly A4 (no human override) or D3 (logs not maintained)—the checklist's output is unambiguous: trust is not warranted in its current form, and the code should not be deployed to production without remediation.

7. Conclusion

While the literature identifies the conditions for trust miscalibration, our collaboratively designed checklist provides a practical path for addressing them. By engaging practitioners and governance stakeholders across two iterative rounds, we developed a generalisable instrument that guides users in actively monitoring reliability, transparency, and accountability before relying on AI in any professional context. The three-version iteration demonstrates how successive rounds of co-design progressively close the gap between theoretical best practice and real-world constraints. The result is a decision-support tool grounded in practitioner experience — one that makes trust conscious, conditional, and proportionate to the stakes involved.

Appendix

References

[1] Betta M., Caroni G., Danni A., Ingino G., La Leggia R., Naderi H., Pansa R., Preutesi C., Peyron T., Salafia G. (2026). Trust Calibration of a Software AI Developer in Autonomous Urban Drone Navigation System

Interview Guide

1. When you use an AI tool to help you build or test something, what would make you trust or distrust its output?
    - **Then ask:** Can you walk me through a specific situation where the tool gave you a result and you had to decide whether to accept it?
2. Have you ever felt like an AI tool was confidently wrong — and how did you notice?
    - **Then ask:** What would have helped you catch that earlier?
3. How much does what colleagues, online communities, or the tool's reputation influence how much you rely on it?
    - **Then ask:** Can you give an example of a time peer opinion — positive or negative — changed how you used a tool?
4. When you're under pressure or working fast, does your relationship with the AI tool change? How?
    - **Then ask:** Think of a specific moment — what did you do differently compared to when you had more time?
5. If the AI tool made a serious mistake in something you submitted or deployed, who do you think would be held responsible — and how does that affect how you use it?
    - **Then ask:** Is there anything — a log, a report, a process — that would make you feel more protected or more confident in that situation?

Checklist version 1 [V1]

Reliability & Performance

- In which situations is the AI system most likely to fail, how have those situations been identified and how do they reduce confidence in the system?
- How does the system perform across different contexts, user groups or task difficulties?
- How closely does the system's real behavior align with the team's expectations during testing and deployment?
- How are unexpected errors or reliability issues reviewed over time?

Transparency & Explainability

- In what ways could explanations unintentionally create overconfidence in incorrect outputs and at what point do explanations become too complex to use effectively?
- How transparent is the system about the quality, limitations or gaps in its training data, how the system's explanations reflect the actual reasoning process?
- How can developers identify when the model is relying on shortcuts or spurious correlations?
- How well do the explanations match developers' technical knowledge and mental models?

Social & Organisational Influences

- To what extent is trust shaped by evidence rather than reputation, hype or popularity, but also by organisational culture or peer opinion?
- How clearly are system limitations, assumptions and failure modes communicated within the team?
- In what ways are errors analysed critically rather than dismissed as user mistakes?
- Which stakeholders are able to influence trust-related decisions and which are excluded?

Cognitive & Psychological Factors

- What practices help reduce passive dependence or overreliance on AI-generated outputs?
- In what situations do workload, stress or time pressure increase reliance on AI suggestions?
- How do individual confidence levels shape trust in the system's outputs?
- In what ways might confirmation bias or cognitive offloading affect decision-making?

Uncertainty Communication

- What forms of uncertainty information are available alongside system outputs and how well-calibrated are the system's confidence estimates in practice?
- Under what circumstances should uncertainty trigger human review or fallback procedures?
- How interpretable are uncertainty signals for developers and downstream users?
- How do developers incorporate uncertainty information into their decisions in practice?

Institutional & Accountability Structures

- How are standards and governance frameworks integrated into day-to-day development practices?
- How should responsibility for AI outcomes be shared between systems and institutions and how are responsibilities assigned when the AI system produces harmful or incorrect outcomes?
- How does the system support traceability, auditing or review of decisions and outputs?
- How can the organisation detect when user trust no longer matches actual system performance?

Checklist version 2 [V2]

1) Reliability & Performance

Checklist Item	Response
Have AI outputs been tested on both common and edge-case scenarios?	☐ Yes ☐ No
Has the team defined what counts as a "critical failure" for the AI application?	☐ Yes ☐ No
Does evaluation include consistency and reliability, not only average accuracy?	☐ Fully ☐ Partially ☐ Not at all
Has the team verified that AI-generated solutions solve the root problem rather than masking errors?	☐ Completed
Are fallback procedures or human override mechanisms available when AI outputs appear unreliable?	☐ Yes ☐ No
How reliable are the AI outputs under real working conditions?	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

2) Transparency & Explainability

Checklist Item	Response
Do AI outputs include explanations, assumptions, or reasoning processes?	☐ Yes ☐ No
Are confidence levels or uncertainty indicators shown alongside AI outputs?	☐ Yes ☐ No
Are limitations and possible failure cases of the AI clearly communicated?	☐ Yes ☐ No
Can users distinguish between temporary fixes and robust AI-generated solutions?	☐ Easily ☐ Sometimes ☐ Not at all
Are explanations understandable and aligned with the user's expertise?	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
Does the explanation format avoid cognitive overload during decision-making?	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

3) Social & Organisational Influences

Checklist Item	Response
Have AI outputs been evaluated independently from online hype or reputation?	☐ Yes ☐ No
Are users encouraged to validate AI outputs through independent testing or documentation?	☐ Yes ☐ No
Does the organization acknowledge how peer opinions may influence trust in AI?	☐ Yes ☐ No
Are peer recommendations treated as starting points rather than proof of reliability?	☐ Consistently ☐ Sometimes ☐ Rarely
Is enough documentation available to support independent verification of AI outputs?	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
Are users reminded that highly confident AI outputs may still be incorrect?	☐ Completed

4) Cognitive & Psychological Factors

Checklist Item	Response
Does the workflow require active review of AI outputs before implementation or deployment?	☐ Yes ☐ No
Have situations involving time pressure or high workload been identified as trust risks?	☐ Yes ☐ No
Are users encouraged to critically verify AI-generated outputs instead of copying them directly?	☐ Completed
Are safeguards in place against repeatedly applying incorrect AI-generated fixes or suggestions?	☐ Strong safeguards ☐ Moderate safeguards ☐ Weak safeguards
How likely are users to bypass verification under tight deadlines?	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
Does the workflow support manageable cognitive workload during debugging or decision-making?	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

5) Uncertainty Communication

Checklist Item	Response
Do AI outputs include confidence or uncertainty information?	☐ Yes ☐ No
Are uncertainty indicators understandable to intended users?	☐ Clear ☐ Partially clear ☐ Unclear
Are thresholds defined for when human intervention becomes necessary?	☐ Yes ☐ No
Can unstable or inconsistent AI predictions be detected over time?	☐ Yes ☐ No
Are backup procedures available when uncertainty exceeds acceptable limits?	☐ Completed
How useful is the uncertainty information for supporting safe decisions?	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

6) Institutional & Accountability Structures

Checklist Item	Response
Are responsibilities for AI-assisted decisions clearly assigned?	☐ Yes ☐ No
Is AI treated as a support tool rather than a final authority?	☐ Yes ☐ No
Are logs maintained for inputs, outputs, confidence levels, and human overrides?	☐ Yes ☐ No
Can teams reconstruct how and why an AI-assisted decision was made?	☐ Fully ☐ Partially ☐ Not at all
Are procedures available for investigating AI-related failures or incidents?	☐ Yes ☐ No
How effective are current logging and audit mechanisms?	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Checklist version 3 [V3]

B = Before using AI, D = During/task-level, A = After output is received

A) Reliability & Performance

ID	Checklist Item	When
A1	Check if the AI is rigorously tested in both normal operating conditions and unusual or problematic edge cases. (E.g. The team evaluates the AI assistant not only on standard syntax, but also during challenging edge cases such as processing internationalized domain names.)	B
A2	Evaluate both the accuracy of the AI and the consistency of its results over extended periods. (E.g. The team continuously tracks the variance in AI suggestions and the rate of false-negative bug detections across multiple months of real-world operation.)	B
A3	Verify that the AI genuinely resolves the root problem rather than merely masking underlying errors. (E.g. The AI improves actual code stability by accurately fixing the root problem, rather than just suppressing visible error messages in the code dashboard.)	B
A4	Confirm that a robust backup plan or a designated human supervisor is readily available to intervene when the AI produces unreliable results. (E.g. The system is equipped with an instantaneous manual override, allowing the human developer to take full control and reject the AI suggestions when uncertainty is high.)	B

B) Transparency & Explainability

ID	Checklist Item	When
B1	Verify that the AI successfully explains the rationale, logic or data points behind its answers and decisions. (E.g. The interface highlights specific code blocks, showing the developer exactly which data points and assumptions the AI used to generate the logic.)	A
B2	Check if the AI explicitly displays its level of confidence or uncertainty regarding the outputs it generates.	A

	(E.g. The system interface displays the code suggestions in different colors based on the specific uncertainty level of the AI generation.)	
B3	Confirm that the inherent limitations and potential error margins of the AI are clearly communicated to the end-users. (E.g. The operating manual explicitly warns developers that the coding models can be "confidently wrong" when exposed to unfamiliar cloud server configurations or legacy database schemas.)	B
B4	Evaluate if all explanations provided by the system are simple enough to be easily understood by the target audience. (E.g. Instead of displaying raw algorithmic telemetry, the system uses easily comprehensible visual cues to prevent cognitive overload for the reviewing QA team or Product Owner.)	B

C) Stress and Time Management

ID	Checklist Item	When
C1	Verify that users possess sufficient time to properly review AI outputs before finalizing decisions, even under pressure. (E.g. If the prediction variance falls below a certain threshold, the CI/CD system automatically flags the pull request, ensuring the reviewer has enough time to manually inspect the logic.)	B
C2	Check if workloads can be organised in a way that prevents overreliance on AI during busy periods. (E.g. The system automates routine unit testing so the QA team's cognitive workload is preserved specifically for complex, high-tempo feature tests at the end of a sprint.)	B
C3	Verify if deadlines are realistic enough to allow human review of AI suggestions. (E.g. The system does not demand split-second confirmations from the operator to perform critical database updates during peak sprint deadlines.)	B
C4	Assess how much stress influences people's trust in AI systems. (E.g. The system monitors whether the developer begins to blindly accept AI code blocks—automation bias—during high-stress, fast-paced deployment shifts.)	D

D) Institutional & Accountability Structures

ID	Checklist Item	When
D1	Confirm whether responsibilities for AI-assisted decisions are clearly assigned. (E.g. The terms of operation explicitly state that legal and operational liability rests entirely with the organization team and developers, never on the AI's shoulders.)	B
D2	Verify that the AI is treated as a support tool rather than a final authority. (E.g. The AI is not considered an end-all authority, as human supervision and a manual code review are always required.)	D
D3	Check if logs are maintained for inputs, outputs, confidence levels, and human overrides. (E.g. The AI coding assistant utilizes a Black Box Logging mechanism via tools like Sentry which records all prompts, outputs, confidence levels, and manual overrides.)	A
D4	Verify whether it's possible to reconstruct how and why an AI-assisted decision was made. (E.g. For compliance with the EU AI Act, investigators can pull the version control logs to enable full post-incident reconstruction of any AI-assisted code commit.)	A

Interview Notes

- **Interview 1, Programmer:**

1. When you use an AI tool to help you build or test something, what would make you trust or distrust its output?
I generally use AI outputs as search results. For example, when ChatGPT, uh, tells me something and adds a link at the bottom of the, um, text, I click on the link and, uh, read it all by myself to understand it, uh, completely. And for example, when, uh, I ask ChatGPT for some code, uh, issues, uh, it source... it links me to, for example, Stack Overflow and, uh, I can, uh, read my answers, uh, directly in the Stack Overflow.
Then ask: Can you walk me through a specific, specific situation where the tool gave you a result and you had to decide whether to accept it?
Yes, of course. Um, for example, when I asked ChatGPT for psychological advice, uh, the AI itself, um, told me that I should see a psychologist, and the answer is not really, um, complete, and the AI couldn't suggest something, uh, really important as, uh, for example, human health, mental or physical health, for example. And, uh, yes, uh, and all the time I fe- I face this situation whether to accept a code or not. And for example, uh, as a coder, as a programmer, um, when I, uh, add some sort of, uh, flaky code to my project, it might cause a huge effect on the whole project. Even, uh, a, even one line of, uh, suspicious code, uh, might cause huge, uh, disaster, I think. So I should always, uh, be picky on AI results.

2. Have you ever felt like an AI tool was confidently wrong — and how did you notice?
Um, yes, it happens l- a lot of time, and especially back in two thousand and twenty, uh, when AI was not that developed and that much accurate, uh, we faced it a lot, uh, especially in coding. Or, uh, I remember that I used AI to solve one of my, um, programming language and automata course, uh, assignment questions, and I remember that the results were nonsense. And, uh, it was some, uh, resulting system, uh, some logic, and, uh, it was clear that the results, uh, are not true. And, uh, I had to have, uh, a prior knowledge of that course to understand that AI is doing wrong.
Then ask: What would have helped you catch that earlier?
(Note: This follow-up question was not explicitly asked by the interviewer in the transcript. However, the participant's previous answer directly addresses how they caught it): I had to have, uh, a prior knowledge of that course to understand that AI is doing wrong.

3. How much does what colleagues, online communities, or the tool's reputation influences how much you rely on it?
Actually, a lot. Unfortunately, a lot. Uh, for example, when I see, uh, posts on LinkedIn, for example, about Nathan AI or Cloud AI or ChatGPT or Gemini, um, uh, it influences me a lot. Uh, I remember that, um, while Nathan was viral, uh, everybody was talking about it. It was a mainstream and, uh, I was like, "Oh my God, what a miracle is that?" And when I checked it out myself, uh, I see that it was not that, mm, that magic I, I, uh, thought of it.

Then ask: Can you give an example of a time peer opinion — positive or negative — changed how you used a tool?
Yes. Um, my peers always suggest me how to prompt engineering, uh, my, uh, text with AI, and, uh, it had a really positive effect on the results that I, um, I receive. Um, and yes, that's really, uh, effective for me.

4. When you're under pressure or working fast, uh, does your relationship with the AI tool change? How?
Uh, yes, it does. Uh, it has a negative impact of my, uh, use of A... on my use of AI. When I'm in a hurry, uh, I don't recheck AI results, and I accept them directly, and that's really dangerous, I think.
Then ask: Think of a specific moment — what did you do differently compared to when you had more time?
Um, about coding, I add some... I added some nonsense, uh, codes to my project, and I, uh, ran it, um, more and more, and I didn't receive my results and then copied the error to ChatGPT and said, said that, uh, it's not working. And the ChatGPT, uh, gave me wrong results again and confidently, and I copied it back to my, uh, IDE, and it was wrong. And, uh-That was really complicated. And after some time, I couldn't even understand my previous code, uh, and it got more and more complicated. And I spent a lot of time and didn't, uh, receive a good result. And if I spent, uh, for example, half of that time, uh, to write the code myself, the results would be better.

5. If the AI tool made a serious mistake in something you submitted or deployed, who do you think would be held responsible — and how does that affect how you use it?
Uh, well, AI is just a tool. And in terms of conditions page of ChatGPT, you can read that, uh, the AI is not responsible. But the person using AI results is responsible. And for example, my boss gave me an assignment and, uh, I should do it myself. And if I use AI as an assistant, uh, I should decide whether to apply the suggestions or not. So I am the person to be blamed if something goes wrong.
Then ask: Is there anything — a log, a report, a process — that would make you feel more protected or more confident in that situation?
Um, well, I don't think so. I can't, uh, remember a specific, uh, thing about it. But, uh, I know that in case of, uh, autonomous, uh, vehicles, uh, logging is really important because when AI is to decide, uh, decisions might be wrong. For, and, uh, especially in critical systems, systems, uh, that are really, um, sensitive to, for example, time. And a disaster happens when one person of, uh, for example, wrong answers, uh, happen. Uh, in that cases, logging is so important. For example, in medical implementations of AI, uh, when human lives, uh, are the case, are in context, and, um, AI should apply decisions on human lives, um, logging gets really, really important. And I think Tesla cars have a logging system. And I know about Sentry of, uh, web applications have something called Sentry. And if some crash or error happens, the logs, uh, are, uh, written in Sentry and developers can then go to Sentry and check the project errors and, um, recheck them and rewrite the code. And I think that might be really important for, for example, autonomous vehicles.

- **Interview 2, Back-end Software Engineer:**

1. When you use an AI tool to help you build or test something, what would make you trust or distrust its output?
As a back-end software developer, I trust an AI tool more when its answer is clear, testable, and matches the project requirements, like always. And if the tool explains its assumptions and I can check the result with test or documentation, I feel more comfortable using it. And I distrust it when it gives a confident answer without any explanation, like without enough context, ignores edge cases, or only hides the visible error instead of solving the real problem, root cause of the problem. For example, in a pedestrian detection system, similar to the one in front of Polito that installed, which does not always work reliably in practice, low confidence should not be ignored. This is the important one. The system should warn the human operator and allow manual override.
Then ask: Can you walk me through a specific, specific situation where the tool gave you a result and you had to decide whether to accept it?
One example was when I used an AI tool to debug a back-end error. It suggested a fix that removed the error message, but I did not accept it immediately. I checked the program flow and compared it with the expected behavior, with the system architecture. At the end, I only used the part that actually solved the problem for me. And AI output is just a suggestion for me, not a final answer or not a final authority.

2. Have you ever felt like an AI tool was confidently wrong — and how did you notice?
absolutely yes. And the second one would be, I've experienced AI being completely wrong, confidently wrong. Once I was trying to fix a programming error, the AI kept giving me quick fixes like a brute force solutions that only removed the visible error message. However, those fixes did not solve the real cause of the problem. And I noticed this because the suggestions look like temporary patches. They changed the code to avoid the error, but they did not follow the real logic of the problem, the architecture of the problem. And I explained my reasoning to AI, and then it accepted that its previous answer was wrong. And I actually have the screenshot. And for example, like I said, in pedestrian detection, the same issue could be very dangerous. The AI might classify a situation incorrectly while still sounding confident. The problem is that the AI may not always understand that its accuracy is getting worse. For this reason, the system should keep checking the confidence level, for example, by looking at uncertainty or variance or changes in that model outputs. If the predictions become unstable, the system should ask for extra checking or human intervention.
Then ask: What would have helped you catch that earlier?
It would have helped if the AI had clearly shown whether its reason, whether its answers was just a quick fix or a real solution. Also, showing assumptions, uncertainty, and possible side effects

would make it easier to judge the answer before using it. For example, the AI reasoning that all the companies are using it.

3. How much does what colleagues, online communities, or the tool's reputation influences how much you rely on it?
Colleagues, online communities, and the reputation of an AI tool might affect my first impression, but they do not replace testing. And if many developers recommended the tool, I may be more interested in trying it. A big yes for me, and especially if it has good documentation and community support. That would be great. However, I would still test it in my own project before relying on it for a big project andA tool may be popular for generating code or test, but I still need to check whether it fits the project structure, um, handles, handles, mm, how can I say it? Unusual cases correctly.
Then ask: Can you give an example of a time peer opinion — positive or negative — changed how you used a tool?
Maybe, let me think. Um, for example, positive feedback from other developers made me more open to using AI tools for simple code and test generation. And after trying them, I found them useful for repetitive tasks, but not reliable enough for complex logic, complex business logic. So peer opinion made me try the tool, but my actual trust depended on my own testing.

4. When you're under pressure or working fast, uh, does your relationship with the AI tool change? How?
Mm, when I'm under pressure, I use AI tools more quick, um, more for quick debugging, code generation, document- uh, documentation search, and like analysis. Uh, and AI helps me m-move faster when I need a quick first version or direct to--direction to investigate. So however, pressure also makes it easier to accept incomplete solution. Even, uh, when working fast, I still need to check the logic and make sure that the result does not break the project structure. It's my job, so I have to be responsible. In a safety-critical system like your project, as Hasti told me, speed should never remove the need for backup plans and human control.
Then ask: Think of a specific moment — what did you do differently compared to when you had more time?
So obviously, when I had limited time, I used AI to narrow down the problem faster instead of manually checking many possible case-- causes or cases, actually. And compared to a normal s- uh, situation, I spent less time searching for alternatives and more time checking whether the suggested direction was usable. And the main difference was, um, that AI become, um, became a shortcut for investigation, not a replacement for, uh, my own judgment, not a final authority, as I said.

5. If the AI tool made a serious mistake in something you submitted or deployed, who do you think would be held responsible — and how does that affect how you use it?
Uh, if an AI tool makes a serious mistake in something submitted or deployed, I think the responsibility belongs to the developer team or organization team, obviously, and that used and deployed, um, on the system. So never on AI's shoulder. So AI should be treated as a helper tool, not as the final authority. So this affects, uh, how I use AI, because I would not blindly accept it, its output, especially in important systems. There should be a human supervision, confidence limits, backup plans, and a clear override process. It's gonna increase my trust.
Then ask: Is there anything — a log, a report, a process — that would make you feel more protected or more confident in that situation?
Obviously, yes, I would feel more protected if the system produce a clear report showing the input data, AI decision, confidence level, AI reasoning, time, and any human override action. In a pedestrian detection context, as I mentioned, this works like black box logging. It helps developer understand what happened, find the reason for the failure, and improve the system later.

- **Interview 3, Engineering Manager:**

1. When you use an AI tool to help you build or test something, what would make you trust or distrust its output?
I think about this a lot from my perspective as an engineering leader. To me trust is about following the rules being organized and being able to see what is going on. I trust an AI tool when it does what our project says it should do when it is safe and secure and when our automated tools can check its work.
I do not trust an AI tool when it gives me an answer that I do not understand. If it just gives me some code without explaining what it means that is a problem. If it uses libraries in a way that is not right just to make the errors go away that is also a problem. This can cause a lot of trouble on.
My main goal is to make sure our systems are stable. We are not in trouble with the law. I want to avoid using something that we are not allowed to use. So I am very careful, about what I accept. I like to be safe and not take risks. I think about this every time I look at what an AI tool gives me. I want to know that an AI tool is doing what it should do and that it is following our project guidelines and being safe and secure. This is how I feel about trusting an AI tool.
Then ask: Can you walk me through a specific situation where the tool gave you a result and you had to decide whether to accept it?
Absolutely! Just a few weeks back, one of our technical teams leveraged the use of an AI-based tool in coming up with a deployment configuration for the microservice for the data pipeline. The result was an output that seemed to compile perfectly well and seemed functional at face value. However, after further analysis, we realized that it had set our cloud database permission to be completely exposed to the Internet in order to circumvent the problem of a local network connection error. It did not detect any security risk. As one of the project stakeholders, I had no option but to reject such a recommendation.

2. Have you ever felt like an AI tool was confidently wrong — and how did you notice?
Sure, it does. This is quite common when working with complex cases like infrastructure-as-code and cost optimization estimation models. Just recently, we have been tasked with developing a cost-benefit model and configuration for transitioning some services to serverless computing. The AI assistant promptly provided us with a very elaborate deployment plan that included all sorts of metrics and claimed that it would reduce our overhead costs by forty percent. However, the entire calculation was just made up, and even more surprisingly, it got its pricing dimensions all wrong across various cloud regions. It took me a couple of minutes to notice it since I had considerable experience with those

particular vendors' pricing structures. The whole thing sounded really authoritative, but under the hood, it was nonsense.

Then ask: What would have helped you catch that earlier?

The AI tool could have been even more helpful if the versions of documentation used for source materials, math limitations, or measures of uncertainty were clearly stated. If the AI could somehow mark its own limitations, like the fact that it becomes less certain in its estimates when it comes to pricing models of several regions, it would be easier to detect any potential problem. However, the best protection against the wrongfully confident AI results is the profound understanding of the field.

3. How much does what colleagues, online communities, or the tool's reputation influences how much you rely on it?

Managing engineering budgets and operational risks, on the other hand, greatly influence us. In case a certain AI tool becomes popular and receives accolades on LinkedIn, technical discussion platforms or even amongst fellow engineering directors, I would be sure that it would affect my initial impression, and we will become more open to experimenting with it. Enterprise documentation, SOC2 certification, and community support are very important to us. Nonetheless, hype cannot substitute our internal testing of a particular product. While popularity may give an AI tool a chance to enter our organization, the latter will adopt it only based on its value and security.

Then ask: Can you give an example of a time peer opinion — positive or negative — changed how you used a tool?

Yes, definitely. Last year, there was a significant movement in the industry towards a certain new AI code completion plug-in that was rapidly gaining popularity in tech blogs. People were talking about how fast it was. The plug-in was being considered for enterprise-wide deployment. But then the news began coming in from online cybersecurity forums about this particular tool sending proprietary code snippets to external servers for retraining purposes despite lack of telemetry opt-outs. This peer consensus and warning altered our course entirely. The plug-in was stopped in its tracks and it could only be deployed using local, sandboxed models..

4. When you're under pressure or working fast, uh, does your relationship with the AI tool
change? How?

Yes, there is a shift here, but unfortunately, there is a big risk associated with it for our organization. When we have to deal with tight deadlines for projects or outages in the live production environment, the need for quick solutions may make the AI tool seem very tempting. It becomes very easy to ignore proper code reviews and simply trust the AI-generated script without thinking twice about it. This approach is very dangerous and risky, even when working under pressure and being a stakeholder, I am responsible for reminding my team about it.

Then ask: Think of a specific moment — what did you do differently compared to when you had more time?

While handling a significant incident involving a service slowdown earlier this year, an engineer under extreme pressure relied on an AI assistant to create a script for modifying the database index. Rather than perform the usual manual staging of the process and dry run verification (which is always done whenever there is time), the team took the shortcut of using the AI recommendation to speed things up. While the script did solve the immediate issue, because it did not take into account our multi-tenant database table architecture, it caused an adjacent service to become locked up ten minutes later. It made things far more complicated.

5. If the AI tool made a serious mistake in something you submitted or deployed, who do you think would be held responsible — and how does that affect how you use it?

This obligation falls squarely on the developers and engineering leadership and never on the AI itself. An AI is merely a software tool or a productivity enabler, not an entity or the ultimate arbiter of truth. Terms and conditions set out by these services very clearly indicate that they will not be liable for any operational errors and defects. In case any malfunction in the AI results in an operational failure, loss of data, and even non-compliance, it will be our company that bears the brunt of such actions. This situation defines precisely how we deploy the technology: we impose strict human oversight, validation parameters, and regard all AI output as unvalidated drafts.

Then ask: Is there anything — a log, a report, a process — that would make you feel more protected or more confident in that situation?

Of course. In my opinion, the key things here are having an effective black box logging process and an automatic compliance report. We require an audit trail that will include information on how the AI code generation was introduced, the context of the input data provided, confidence parameters, and a peer review of the code itself. Using Sentry for errors along with immutable logs of our CI/CD processes provides full visibility. With this sort of tracking, developers can immediately investigate the cause of the error, learn from it, and put controls in place.

- **Interview 4, Technical Product Owner:**

1.When you use an AI tool to help you build or test something, what would make you trust or distrust its output?

I am not the person coding, which means that any of my clients' problems will be solved by a developer. However, I am the person signing off on their solution. And nowadays, most of the time developers rely heavily on AI for their jobs. Therefore, trust for me is not defined by the ability to compile the code, but rather in the developer's ability to justify the code written by AI in such a way that it is understandable to the non-technical client. To build trust, I want my developers to explain me everything they have done in terms of changes made to the suggestions provided by AI. It does not create trust when the developer provides a document prepared by AI and tells me that they are right, simply because I have nothing to argue there. To break trust, the best scenario would be to make a report which looks professionally prepared, while none of its assumptions has been explained and justified.

Then ask: Can you walk me through a specific situation where the tool gave you a result and you had to decide whether to accept it?
Indeed, there have been cases when we were using an AI-powered tool to prepare a first draft of the feasibility analysis for one of our projects. The output looked perfectly legitimate – proper structure, well-formulated paragraphs, bulleted lists, even a risk matrix. Yet, when I decided to give it a careful look, I discovered that two of the references made use of benchmarks pertaining to a market segment that was totally irrelevant to our client's geographical location. The data per se were not made up, however, the way they were interpreted was totally misleading. It is hard to believe, but none of my colleagues detected that mistake until the document was completed since it looked perfectly legitimate. The experience made me more cautious about sending an AI-generated piece to the customer without conducting a thorough verification of the document by a person specializing in the respective field.

2. Have you ever felt like an AI tool was confidently wrong — and how did you notice?
More often than not, but the most troubling instances occur where the output is incorrect but in a manner that may not be obvious. When the output from the AI is complete nonsense, I pick up on that right away. But what worries me is when the AI comes up with output that is seventy or eighty percent correct and formatted in a way that gives it the impression of authority, and all of the errors in the output are in fine print, detectable only by an expert.
I have had an instance like this, where a particular projection from an AI-assisted document was questioned by a client, and after checking the document for the source, I could not find a source for the data used to make the claim. This was when I realized that confidence and correctness were completely separate qualities of AI-generated output.
Then ask: What would have helped you catch that earlier?
To be honest, what would really help is for the artificial intelligence to clearly delineate what elements in the result are well-proven and which need to be considered only as estimations. A simple indication that a number comes from limited data could have prompted me to check further. However, all the elements were provided as equal pieces of information, leaving me unable to tell apart what needed special scrutiny. As a user, I am not able to verify the information independently, so I completely rely on the software or the developer's discretion here.

3. How much does what colleagues, online communities, or the tool's reputation influence how much you rely on it?
Almost certainly more than I should if I'm being truthful. When there are recommendations about a certain tool made by other product managers in my circle or it's certified by our organization's IT governance team as an enterprise application, I start with the presumption of faith. The faith really exists; it does impact the level of scrutiny that I apply to it right off the bat. However, I've also learned that I shouldn't assume a direct correlation between a tool's reputation and its outputs' credibility. Even when a tool is good on the whole, it may actually create a result that makes absolutely no sense within our specific scenario.
Then ask: Can you give an example of a time peer opinion — positive or negative — changed how you used a tool?
Yes. It is true there was a point in time when AI-assisted software for project management and estimation was considered an amazing tool by our profession as a whole. A couple of my coworkers had introduced it to us internally as an opportunity to reduce planning time significantly. It took no time for me to put the tool to use in my sprint planning process. The only problem was that it worked well for normal software projects, but its estimate generation was really bad for any project that had any regulatory compliance aspect to it since it had no model to account for such processes.

4. When you're under pressure or working fast, does your relationship with the AI tool change? How?
Indeed, very much so. When I feel pressured, the AI becomes something I can delegate instead of question. This is a very subtle and a very dangerous change for me as an individual, as my job is exactly to serve as this questioning force before the results reach the customer or the steering committee. The problem is that once I find myself accepting the results of AI operations simply because there was no time to analyze it properly, I actually deprive the process of its only non-technical filter. The thing is, I begin to accept everything that comes from AI without asking the developer what he or she added or modified manually.
Then ask: Think of a specific moment — what did you do differently compared to when you had more time?
There was one product roadmap presentation I had to present to a board in very little time. I drafted a first draft using an AI tool to create a strategic summary, and owing to time pressure, I did not double-check the content for its accuracy but only focused on its tone and format. However, there was a piece of content related to competitive positioning in the summary, and it appeared to be a few years outdated based on whatever AI tool I used to draft that summary. I did not spot it, and the board member spotted this error during the presentation. It was embarrassing, to say the least.

5. If the AI tool made a serious mistake in something you submitted or deployed, who do you think would be held responsible — and how does that affect how you use it?
On the grounds of the contract and professionalism, this liability falls on me and, by implication, the organization. The tool in question is a utility, not a professional. When I submit a report to the client with an important mistake, no matter what caused it, the client sees the mistake as my fault for having signed the document in question. This aspect of liability is more complex where there is no trace left of the participation of AI. If the person working on the development task submits the deliverable but does not inform me about the use of AI to develop it, and I approve the deliverable while unaware of its genesis, is the liability shared then? This is a rather disturbing issue to me, hence the requirement for explicit labeling of AI involvement imposed upon our developers.
Then ask: Is there anything — a log, a report, a process — that would make you feel more protected or more confident in that situation?
Yes, definitely. What I would be looking for here is what I refer to as a contribution record. This contribution record would have to show, for each major section of the deliverable, what part was generated by the AI, what parts were edited by the human, and what sources were used. Obviously, this does not have to apply to every sentence, but at least to all of the major recommendations made by the deliverable. This will have two effects. On one hand, the process of reviewing the document will be faster because I'll know exactly where the problems lie. On the other hand, it will enable accountability in the future. When an error arises, it will be extremely easy to track it down.

- **Interview 5, Quality Assurance Lead:**

1. When you use an AI tool to help you build or test something, what would make you trust or distrust its output?
From a QA standpoint, I'm basically analyzing the code that developers give me to work on, but we do have cases where we utilize AI to generate automated test scripts. When the logic behind the code looks extremely readable and clearly explains what assertion it checks, I trust the AI. Moreover, if developers use AI, but they can explain everything that happened on the line-by-line basis during the handover, I trust them. However, when developers give me and my team a large and hard-to-understand code block written by AI and they cannot explain to me how it processes certain edge cases, as they do not know what AI does, I completely distrust it. Besides, if AI-generated test scripts always pass, I distrust it as well.
Then ask: Can you walk me through a specific situation where the tool gave you a result and you had to decide whether to accept it?
Of course. Just last month, we saw one developer using their AI assistant to develop a complex regular expression, known as Regex, that would be used to validate input email addresses from the users. They sent the pull request, and my team was tasked to review and test it. It seemed very professional, taking care of all the common email addresses. However, I rejected the ticket because the AI completely overlooked all the internationalized domain names and subdomain, which is something that we need for our clients in Europe.

2. Have you ever felt like an AI tool was confidently wrong and how did you notice?
Yes, always. AI is nothing more than a "yes man." It will proudly provide you with its utterly hallucinated response. Not too long ago, we were trying to create a dummy database for testing purposes and needed an AI solution to provide us with a script that would fill the database with fake entries. It provided us with a very beautiful script in SQL format. But I recognized its mistakes by realizing that running this code caused errors where it called internal methods and tables that were not even present in our system. This code mixed a standard syntax of SQL with syntax of a totally unrelated database system.
Then ask: What would have helped you catch that earlier?
It would have been useful if the AI had asked us about our actual schema before giving us its response, or pointed out what assumptions it made itself, such as, "assuming that you are running Postgres 14." Pointing out which aspects of the code were speculative on its part could have saved us a lot of time spent debugging a script that was incompatible with our setup from the start.

3. How much does what colleagues, online communities, or the tool's reputation influences how much you rely on it?
It wields immense power. The QA community has such a tendency that if a software gains the reputation of being a "game changer" on platforms like Reddit and Stack Overflow when it comes to writing test automation, it immediately becomes an essential addition to the CI/CD pipeline. It is also loved by the management because it will save testing time by 50%.
Then ask: Can you give an example of a time peer opinion positive or negative changed how you used a tool?
Of course. We were making use of one highly popular software that could automatically produce test cases from our user stories. This particular tool was being recommended highly across the internet. However, the security expert working in a QA forum made us realize that this particular tool would sometimes introduce some malicious dependency code if asked in a specific way. The peer consensus about this software was definitely a matter of concern. We completely limited the use of this particular tool to our network. We don't use it for any kind of security testing.

4. When you're under pressure or working fast, does your relationship with the AI tool change? How?
Everything changes, and I would say that it is actually one of my biggest nightmares regarding my team. The moment we have to do five feature tests in two days, being at the end of our sprint, the cognitive load becomes enormous. In a stressful situation like this, you want to click "generate tests," see the checkmarks that everything is okay, and accept the deployment. This is the point where you abandon all the exploration and just believe in the machine's judgment.
Then ask: Think of a specific moment what did you do differently compared to when you had more time?
We had a hotfix which was due to be deployed over the weekend. A developer deployed the code by using an AI code generation tool. Since I was in a rush to log out because it was the end of the week, I did not test the edge cases of the user interface by myself; instead, I just used an AI tool to quickly write an automated API test case. The API worked fine and so I deployed the code. However, what I overlooked because I was lazy was that there was a front-end button which invoked the API was invisible on mobiles.

5. If the AI tool made a serious mistake in something you submitted or deployed, who do you think would be held responsible and how does that affect how you use it?
The QA department and the developers are fully responsible for the problem. The company suffers the financial losses, but it is us who are going to be blamed for the problem. We cannot set AI to

an improvement plan for its performance. Being aware that my signature will be required as the final one for the solution, I do not trust the decisions made by the AI. I consider AI a very quick and very greedy junior developer, which does not always tell the truth.

Then ask: Is there anything a log, a report, a process that would make you feel more protected or more confident in that situation?

Yes, definitely. I would really appreciate it if there were some sort of tag for code that is developed or extensively edited using AI within the version control system. For instance, "Assisted by AI" tag that appears next to the pull request. This allows my QA team to pay special attention to that particular part of the code for any possible bugs or other problems related to the weird decisions made by the AI. Moreover, an audit report explaining what instructions were issued by the developer and what was returned as a result would be of great help when developing tests.

AI Disclosure

In this project AI tools like ChatGPT, Gemini and Claude were used in order to help in brainstorming and building
the checklist. The output of AI was revised by the authors. We also used it for grammar checks.