**Interview Transcript-** Participant: [Anonymized]

**Date of Interview:** 26 February 2025

**Mode of Interview:** Physical

**Interviewer:** Anonymized

**Participant Role:** Senior Software Engineer

**Duration:** 50:45 mins

**Consent Obtained:** Yes

**Interviewer:** Thanks for joining me today. I appreciate you taking the time.

**Participant:** Of course, happy to be here.

**Interviewer:** Let’s begin with a quick background. How long have you been involved in software development?

**Participant:** I've been coding for more than eight years now, and I’ve spent the last four years working professionally as a software developer.

**Interviewer:** That’s solid experience. Are you familiar with code generation tools like ChatGPT or Gemini, or other LLMs?

**Participant:** Yes, I use code LLM. They've become a helpful part of my workflow.

**Interviewer:** Interesting. Do you tend to use them while actively writing code?

**Participant:** Definitely. I often turn to them for help when I’m trying to structure something or when I hit a roadblock. I don’t rely on them fully, but they’re great for generating an outline or providing inspiration I can then modify to fit my needs at work.

**Interviewer:** How familiar are you with the concept of code clones?

**Participant:** I’d say I have a decent understanding. To me, it’s mostly about duplicated logic, like when the same blocks of code show up in different places. At work, we actually have a clone-check step built into our GitHub Actions pipeline to catch copy-paste slip-ups during review.

**Interviewer:** Have you noticed any changes in how your clone detection behaves since you started using AI coding assistants?

**Participant:** Yes, actually, there have been a couple of unexpected shifts. First, AI tends to generate lots of small helper functions, usually between 4 to 10 lines, which are too short to be flagged by our current detector. So those duplicates slip through. Second, I've seen the same helper function pop up across different repositories. Since our detection runs on a per-repo basis, it doesn’t catch those cross-repo clones. The problem is, if there’s a bug in one version, it’s likely in all of them. So even though we used to be pretty confident in our clone-checking, we’re now getting false negatives in places we didn’t expect.

**Interviewer:** What do you think is causing the tool to miss those duplicates?

**Participant:** I personally think there are a few key reasons. First,the detector is configured to skim small code blocks by default, which just happens to be the size range most AI-generated helpers fall into. Second, even when the logic is the same, AI tends to use different placeholder names. For example, maxRetries in one place and timeoutMs in another. So the code looks different enough that the hashes don't match. And lastly, our scanner only checks within a single repository, but AI doesn’t follow those boundaries. It reuses the same templates across repos, so the detector misses clones that span projects.

**Interviewer:** Given these challenges, what’s your team planning to do?

**Participant:** Lower the size threshold to catch sub‑15‑line snippets and recalibrate the detectors with the new AI code clones.

**Interviewer:** Once you update the tool, do you see any benefits?

**Participant:** Yes. AI’s consistency could actually help us auto‑refactor common helpers across services. But we need the detector to see them first.

**Demonstration Phase – Summary**

During this phase, the interviewer presented participants with several code snippets: one set written by humans (sourced from Stack Overflow) and another set generated by AI tools, both designed to perform the same functionality. Participants were asked to review the snippets and identify which ones were AI-generated and which were human-written.

**Key Insights:**

* The participant couldn’t determine all the AI-generated code. (after removal of comments)
* The participant did agree that sometimes LLMs provide some unique solutions which is not a common implementation at all.
* The participant pointed out that AI code tends to appear "templated," especially in helper functions, making it distinguishable from more varied human-written styles.
* The participant observed that certain AI-generated snippets were functionally identical but written with slight lexical variations.
* The participant specifically noted that many short AI-generated helpers (under 10–15 lines) were repeated in function but ignored by detection tools due to preset length thresholds.
* The participant saw AI’s repetitiveness as an advantage, suggesting that if detection tools were improved, they could automatically consolidate repeated helpers into common utilities, enhancing maintainability across services.