Assessing security implications of using generative AI in software development in Tanzania: A quantitative study

Kelvin Vedasto Katto

mubezikatto@gmail.com

https://orcid.org/0009-0001-8781-6197

College of Business Education, Tanzania

https://doi.org/10.51867/ajernet.7.4.126

ABSTRACT

Generative artificial intelligence (AI) is increasingly used in software development, raising concerns about the security of AI-generated code. This study assessed the security implications of generative AI use in software development in Tanzania. A quantitative, cross-sectional survey design, guided by the Technology Acceptance Model (TAM) and Protection Motivation Theory (PMT), was employed to collect structured questionnaire responses from 77 software developers. Descriptive statistics and non-parametric tests, including Spearman's rank correlation and the Kruskal–Wallis test, were used to examine four hypotheses. The findings showed that 78% of respondents used generative AI tools daily or several times a week, with ChatGPT being the most commonly used tool (68.8%). Respondents generally perceived these tools as easy to use (mean = 3.94), while trust in the security of AI-generated code was relatively low (mean = 2.93). Programming experience was significantly associated with trust in AI-generated code security ($H(4)=17.431$, $p=.002$). Respondents with less than one year of programming experience reported higher trust than several more-experienced groups. However, no statistically significant relationships were found between AI usage frequency and perceived security risk, AI tool choice and reported security issues, or perceived ease of use and perceived security risk. The findings highlight the importance of reviewing AI-generated code, applying secure coding practices, and conducting security testing. Larger studies incorporating objective code-security assessments are recommended to strengthen the evidence.

Keywords: Code Security, Generative AI, Software Development, Trust and Programming Experience

I. INTRODUCTION

Generative Artificial Intelligence systems have revolutionized how the world operates in different fields, now they are emerging as a fast-growing technology used in different fields (Annapureddy et al., 2025; R. He et al., 2025). Generative Artificial Intelligence, commonly known as Generative AI or Gen AI, uses artificial intelligence technology to create text, images, music, and computer code through algorithmically learned pattern recognition from big databases (R. He et al., 2025). The evolution of Generative AI is contributed to and linked to the development and advancement of Machine Learning Technology and Deep Learning (DL), particularly with the introduction of neural network architectures that enable AI models to recognize and replicate complex patterns (He et al., 2025). The application of Generative AI has now spanned different industries, from small to large industries (Annapureddy et al., 2025). They now provide innovative solutions that enhance efficiency, automation, and decision-making (Annapureddy et al., 2025). Some key areas benefiting from Generative AI tools are software development, cybersecurity, the health sector, the education sector, the finance sector, and creative industries (Annapureddy et al., 2025).

Generative Artificial Intelligence (AI) has achieved rapid technological progress in Tanzania, like any other developing country, which drives fundamental software development changes, as developers can now use automatic code generation and optimization solutions with enhanced deployment capabilities (Banyasz et al., 2024). Generative AI tools like GitHub Copilot and ChatGPT offer developers complex assistance for generating code and detecting bugs, as well as enhancing system coding in real time (Bull & Kharrufa, 2023). AI programming tools boost developer performance by saving time in coding tasks and simplify programming work by delivering suggestions for developers of varying experience levels (Dakhel et al., 2023). The use of Generative AI in software development offers several benefits; however, security challenges have also emerged in its application (Garcia, 2024). Critical vulnerabilities can appear in AI-generated code, spreading insecure programming patterns and introducing code flaws caused by faulty logic (Perry et al., 2023a; Sandoval et al., 2025).

At the same time, Tanzania faces challenges related to the availability of software-development security knowledge and skills(Binamungu et al., 2024). For example, a study of software practitioners in Tanzania involving 70 industry professionals and 25 follow-up interviews identified significant gaps in software-safety knowledge and skills and emphasized the need to strengthen software-safety competencies among software professionals and ICT graduates

Licensed Under Creative Commons Attribution (CC BY-NC)

(Binamungu et al., 2024). Although this study did not specifically investigate Generative AI, its findings provide an important contextual basis for examining whether the increasing adoption of AI-assisted software development may introduce additional security concerns within the Tanzanian environment. However, existing studies on the security implications of Generative AI in software development have predominantly focused on international datasets, controlled experiments, or software repositories outside Tanzania(Fu et al., 2025; J. He & Vechev, 2023; Klemmer et al., 2024). There is comparatively limited empirical evidence concerning how developers in Tanzania use Generative AI tools, how frequently they use them, how much they trust AI-generated code, what security risks they perceive or encounter, and whether developer characteristics influence these security-related outcomes.

Therefore, this study seeks to assess the security implications of using Generative AI in software development in Tanzania using a quantitative research approach. Specifically, the study will examine developers' use of Generative AI tools, their perceptions of security risks, trust in AI-generated code, and security issues encountered during AI-assisted software development. The study will further investigate whether factors such as AI usage frequency, programming experience, tool choice, and perceived ease of use are associated with security-related outcomes. By generating empirical evidence from the Tanzanian context, the study is expected to contribute to the growing body of research on secure AI-assisted software development and provide evidence that may inform developers, software organizations, ICT educators, cybersecurity practitioners, and policymakers on the responsible and secure adoption of Generative AI in software engineering.

1.1 Statement of the Problem

AI is rapidly integrated into software development, which continues to reshape coding practices by automating tasks such as code generation, debugging, and refactoring (Banyasz et al., 2024). The increased adoption of these tools (such as GitHub Copilot, ChatGPT, etc.) has offered developers the promise of enhanced productivity and efficiency (Liang et al., 2024; Davila et al., 2024). However, previous studies have highlighted serious concerns about the security of AI-generated code (Fu et al., 2025; Perry et al., 2023b; Asare et al., 2023). Studies indicate that, although these tools are effective at assisting with coding tasks, they also often reproduce vulnerabilities embedded in their training data, which lead to the propagation of insecure programming patterns (Fu et al., 2025; Perry et al., 2023c; Asare et al., 2023). Furthermore, conflicting findings from recent studies have deepened the uncertainty (Perry et al., 2023; Sandoval et al., 2025). While Perry et al. (2023) concluded that AI assistance increases security bugs, Sandoval et al. (2023) observed that Generative AI tools do not significantly raise security-critical errors, despite the presence of vulnerabilities. This inconsistency highlights a gap in understanding, particularly concerning how developers themselves perceive and respond to these security risks in practice.

In Tanzania, the growing adoption of digital technologies and AI makes this issue increasingly relevant. However, empirical evidence on the security implications of generative AI use among software developers in Tanzania remains limited. Previous research has identified gaps in software-safety knowledge and skills among Tanzanian software practitioners (Binamungu et al., 2024), but limited attention has been given to the emerging security challenges associated with AI-assisted software development. Therefore, this study seeks to quantitatively assess the security implications of using generative AI in software development in Tanzania, focusing on developers' usage patterns, perceived security risks, trust in AI-generated code, and security issues encountered.

1.2 Research Objectives

i. To explore software developers' awareness and understanding of Generative AI tools commonly used in software development.
ii. To explore the factors influencing developers' perceptions of security risks related to Generative AI.

II. LITERATURE REVIEW

2.1 Theoretical Review

This study draws on the Technology Acceptance Model (TAM) and Protection Motivation Theory (PMT) to examine developers' interactions with Generative AI coding tools and their perceptions of AI-generated code security (Davis, 1989; Rogers, 1975). TAM provides a foundation for understanding technology usability, while PMT explains how individuals appraise potential threats and protective responses. However, neither theory directly explains all the relationships examined in this study. Therefore, empirical research on AI-generated code security and developer trust is also used to develop the hypotheses.

Davis (1989) developed TAM to explain technology acceptance through constructs including perceived usefulness and perceived ease of use. Perceived ease of use refers to the degree to which an individual believes that using a system requires little effort (Davis, 1989). In AI-assisted software development, this construct helps explain how developers perceive the effort required to generate, understand, and integrate code. Nevertheless, TAM primarily

Licensed Under Creative Commons Attribution (CC BY-NC)

explains technology acceptance rather than the security of technological outputs. It therefore provides a theoretical basis for examining perceived ease of use but does not directly predict its relationship with perceived security risk.

Protection Motivation Theory, proposed by Rogers (1975) and developed further by Maddux and Rogers (1983), explains how individuals appraise potential threats and evaluate their ability to respond to them. In software development, PMT provides a useful perspective for understanding developers' perceptions of security threats and the importance of protective practices, such as reviewing and testing AI-generated code. However, it does not directly establish that usage frequency, programming experience, or AI solution choice determines perceived security risk or reported security issues. Accordingly, TAM and PMT provide complementary perspectives, while the remaining relationships are developed through construct-specific reasoning and empirical literature.

2.2 Empirical Review

Generative AI and Software Security in Developing Countries

Research on Generative AI and software security in developing countries is emerging. In India, Sankhe et al. (2025) conducted an experiment involving 120 professional developers and reported increased security vulnerabilities in AI-assisted code. In Nigeria, evaluated AI-generated web application code and identified weaknesses involving session management, file handling, and access control. These findings suggest that AI-generated code requires careful security verification in different development contexts. In Tanzania, Binamungu et al. (2024) identified gaps in software safety knowledge and skills among practitioners, highlighting a relevant local challenge. However, evidence specifically examining Tanzanian developers' trust in AI-generated code and their perceived security risks remains limited, supporting further context-specific empirical investigation.

Generative AI Usage Frequency and Perceived Security Risk

Frequent use of Generative AI tools may expose developers to more AI-generated code and provide opportunities to identify potential security weaknesses. (Perry et al., 2023) found that participants using an AI code assistant produced less secure code in their experimental tasks and were more likely to believe their code was secure. Asare et al., (2023) also investigated vulnerabilities in code generated by GitHub Copilot. These studies demonstrate the relevance of security concerns in AI-assisted development, but they do not directly establish that frequent use increases perceived security risk. Two competing mechanisms may explain this relationship. Repeated use may increase exposure to insecure outputs and awareness of security risks (Kudriavtseva et al., 2025) . Conversely, familiarity with AI tools may increase confidence and reduce perceived uncertainty, potentially creating a mismatch between perceived and actual code security (Perry et al., 2023). Since these mechanisms suggest different possible outcomes, the direction of the relationship cannot be assumed. PMT informs the interpretation of threat appraisal but does not directly predict an effect of usage frequency. Therefore, this study examines the association empirically.

H1: Generative AI usage frequency is significantly associated with developers' perceived security risk of AI-generated code.

Programming Experience and Trust in AI-Generated Code Security

Programming experience may influence developers' ability to evaluate AI-generated code, recognize vulnerabilities, and assess whether outputs satisfy security requirements (Pinto et al., 2024). Developers with greater experience may be better equipped to verify generated code before integrating it into software (Ryan et al., 2019). However, experience may also lead to more critical evaluations and lower trust in code that has not been adequately verified (Ryan et al., 2019). Research on trust in AI-powered code-generation tools highlights challenges associated with evaluating and validating AI-generated suggestions (Wang et al., 2024). Similarly, Brown et al. (2024) found that developers' expertise in a programming language influenced their acceptance of AI-generated code suggestions. These findings support the importance of developer judgement but do not establish that greater programming experience necessarily increases trust in code security. From an expertise-based perspective, experience may improve developers' ability to assess generated outputs, while also making them more aware of potential vulnerabilities. Consequently, this study examines the relationship without assuming a particular direction.

H2: Programming experience is significantly associated with developers' trust in the security of AI-generated code.

Generative AI Solution Used and Reported Security Issues

Generative AI coding solutions may differ in their code-generation behaviour and ability to satisfy security requirements. Asare et al. (2023) investigated vulnerabilities in Copilot-generated code, while Fu et al. (2025) examined security weaknesses associated with AI-generated code in GitHub projects. Such research demonstrates the importance of evaluating generated outputs rather than assuming that functional code is secure. However, reported security issues may depend not only on the AI solution used but also on the programming tasks undertaken, frequency of use, and developers' review and testing practices. Developers with stronger security knowledge may also detect issues that others overlook. Neither TAM nor PMT directly predicts that choosing a particular AI solution determines security issues.

Licensed Under Creative Commons Attribution (CC BY-NC)

Therefore, this hypothesis is grounded primarily in empirical research on AI-generated code security. The study examines reported experiences without assuming that any observed difference proves one tool is inherently less secure.
H3: The Generative AI solution used is significantly associated with developers' reported experience of security issues in AI-generated code.

Perceived Ease of Use and Perceived Security Risk

Perceived ease of use is a central TAM construct describing the degree to which a person believes that using a system requires little effort (Davis, 1989). In AI-assisted development, developers may perceive a tool as easy to use when it simplifies code generation, debugging, or code modification. However, ease of use does not necessarily indicate that generated code is secure.

Perry et al. (2023) found that participants using an AI code assistant were more likely to believe their code was secure, despite producing less secure code in their experimental tasks. Wang et al. (2024) also highlighted the importance of validating AI-generated suggestions. These findings demonstrate that confidence in AI-assisted development does not necessarily correspond to actual code security, although they do not directly establish that perceived ease of use affects perceived security risk. Ease of use may increase confidence and reduce perceived uncertainty, but it could also encourage developers to accept generated outputs with less scrutiny. These competing possibilities mean that TAM supports the inclusion of perceived ease of use but does not directly predict its relationship with security risk. This study therefore examines the relationship empirically.
H4: Developers' perceived ease of use of Generative AI coding tools is significantly associated with their perceived security risk of AI-generated code.

III. METHODOLOGY

3.1 Research Design

This study is philosophically grounded in a positivist epistemology and objectivist ontology, providing a clear foundation for our research design. The study used a quantitative research design and a structured survey method, which allowed us to gather numerical data objectively and test hypotheses about how generative AI affects software security, exemplifying the positivist aim of explaining phenomena through unbiased analysis (Tuli, 2011).

3.2 Participants and Sampling

The target population for this study was software developers who have experience using generative AI coding tools (such as GitHub Copilot or OpenAI's ChatGPT) in their development workflow. A purposive sampling strategy was used to intentionally recruit participants meeting this criterion, as our research questions center on the perceptions of developers who are familiar with generative AI assistants. Purposive sampling (a form of non-probability sampling) was appropriate because a random sample of all software developers might include many who have never used AI coding tools, whereas we specifically needed insights from active users of tools like GitHub Copilot or ChatGPT. The channel used to recruit the participants was WhatsApp groups for software developers from different institutions. The final sample consisted of N = 77 software developers.

3.3 Data Collection

Data were collected through an online survey using the Google Form instrument. The survey was sent directly to software developers of the selected institutions in Tanzania. The survey was sent to 18 different institutions from different sectors, including financial, Tech industries, Healthcare, telecommunications, education, Energy and Agriculture, transport and logistics, and e-commerce. The survey introduction made it clear that it sought participants who had used generative AI in coding, and asked those who did not meet this criterion to refrain from participating. Interested participants could click the anonymous link to access the questionnaire. Upon opening the survey, respondents first saw an informed consent statement (discussed more under the

3.7 **Ethical Considerations** section below) and had to agree before proceeding. The survey then presented the questions in a fixed order: demographics and background first, followed by AI tool usage patterns, then the sets of Likert-scale opinion statements (grouped by topic), and finally an optional free-text question at the end for any additional comments (optional). Participants typically took about 10 minutes on average to complete the survey, based on form timestamps.

3.4 Data Analysis

Once the survey was closed and the data compiled, a structured data analysis plan using IBM SPSS (Version 30) was followed. The analysis consisted of both descriptive and inferential statistics to address the research questions. The study began with descriptive statistics to summarize the dataset and characterize the sample. This included calculating frequencies and percentages for categorical variables. The study also computed measures of central tendency

Licensed Under Creative Commons Attribution (CC BY-NC)

(mean and median) and dispersion (standard deviation) for key Likert-scale derived indices (Falcão & Canedo, 2024; May et al., 2024). For hypothesis testing, the study employed appropriate non-parametric inferential statistical tests in line with the nature of our data. First, to examine relationships between key ordinal variables, Spearman's rank-order correlation (Spearman's ρ) was used. In addition, for hypotheses involving comparisons across groups, the Kruskal-Wallis H test, a non-parametric alternative to one-way ANOVA, was used. This test evaluates whether the distribution (median ranks) of the outcome variable differs significantly across the groups (Rachmatullah et al., 2023).

3.5 Reliability and Validity

Reliability refers to the consistency or stability of a measurement over time and items (Ottenbacher et al., 1996; Pesesse et al., 2024). In our survey, we assessed reliability via internal consistency using Cronbach's alpha. Ottenbacher et al. (1996) recommend that a Cronbach's alpha of 0.70 or higher indicates acceptable reliability. In our data, all multi-item scales easily exceed this threshold. Validity concerns whether the instrument measures what it claims to measure (Mellinger & Hanson, 2021). The study addressed validity primarily through careful survey design. Most of the questionnaire items were drawn from existing, published instruments in technology acceptance and security literature, which enhances content validity, and some were generated based on the nature of the study.

3.5.1 Instrumentation

Data were collected using a structured questionnaire administered through Google Forms. The questionnaire covered demographic characteristics, AI tool usage patterns, and attitudinal measures adapted from established literature on trust in AI, perceived security risk, and technology acceptance (Davis, 1989; Artino et al., 2014). Attitudinal items used five-point Likert scales ranging from "Strongly disagree" to "Strongly agree" and from "Never" to "Very often." The instrument assessed trust in AI-generated code security, perceived security risks, perceived ease of use, AI tools used, usage frequency, and development tasks. Items were adapted to the generative AI software-development context, drawing on previous studies, including Falcão and Canedo (2024) and Klemmer et al. (2024). Table 2 presents some of the questionnaire items, measured variables, and their sources.

Table 1

Survey Questionnaire with variable

Variable	Type/Scale	Item Used	Adapted/modified
Developer experience	Multiple choice	What is your level of professional software development experience?	(Ryan et al., 2019)
Frequency of AI usage	Multiple choice	How often do you use Generative AI tools	(Klemmer et al., 2024)
Type of AI	Multiple choice	Which Generative AI tool do you mostly use in your development activities?	(Wang et al., 2024)
Perceived Ease of Use	5-Point Likert Scale	PEU1 - I find Generative AI coding tools easy to use in my daily development tasks. PEU2 - Learning to use AI tools like GitHub Copilot or ChatGPT is straightforward. PEU3 - I can use AI-generated code with minimal effort. PEU4 - It is easy to integrate AI-generated code into my existing development workflow. PEU5 - I feel comfortable navigating and interacting with AI coding tools.	(Davila et al., 2024)
Perceived Security Risk	5-Point Likert Scale	PSR1 - I believe AI-generated code is more prone to security risks. PSR2 - AI-generated code increases the likelihood of security breaches. PSR3 - I feel that AI-assisted coding introduces new security vulnerabilities. PSR4 - Using AI tools for software development poses significant security threats. PSR5 - AI-generated code requires more security scrutiny than human-written code.	(Falcão & Canedo, 2024)
Trust in AI Security	5-Point Likert Scale	TRUST1 - I trust that AI-generated code is generally secure. TRUST2 - AI coding tools follow best security practices by default. TRUST3 - I feel safe using AI-generated code without additional security checks. TRUST4 - AI-generated code is as secure as manually written code. TRUST5 - I have confidence that AI tools minimize security risks.	(Klemmer et al., 2024)
Encountered Security Issues	5-Point Likert Scale	ESI1 - I identify security flaws in AI-generated code. ESI2 - AI-generated code introduces vulnerabilities into my projects. ESI3 - I rewrite AI-generated code due to security concerns.	

Licensed Under Creative Commons Attribution (CC BY-NC)

		ESI4 - AI-generated code causes security-related incidents in my projects. ESI5 - I encounter more security flaws in AI-generated code than in human-written code.	

3.6 Data Screening and Quality Control

Before statistical analysis, the dataset was screened for missing values and duplicate records. Two consecutive records containing identical responses were identified. However, the available information was insufficient to determine whether they represented a duplicate submission or an accidental repetition. To assess the potential influence of these records, a sensitivity analysis was conducted by removing one of the identical records and repeating the statistical analyses. The results remained unchanged, indicating that the inclusion or exclusion of the record did not affect the final findings. The dataset was subsequently analysed using SPSS version 30. Descriptive statistics were used to summarise respondents' characteristics and study variables, while Spearman's rank correlation and Kruskal–Wallis tests were used to examine the relationships and differences relevant to the study hypotheses. Statistical significance was assessed at the 5% level.

3.7 Ethical Considerations

This study was conducted with careful attention to ethical research standards, particularly because it involves human participants (survey respondents) and potentially sensitive perceptions about technology use at work. The study followed Diener and Crandall's (1978) guidelines to ensure research integrity and participant protection. Informed consent was obtained from all participants at the start of the survey.

IV. FINDINGS & DISCUSSION

4.1 Descriptive Statistics

The study first presents the descriptive statistical results of the key variables measured in this study. The descriptive part provides an overview of participants' responses across various constructs, including Participant Programming Experience, AI Tool Usage, Perceived Ease of Use (PEU), Perceived Security Risk (PSR), Trust in AI Security (TRUST), and Encountered Security Issues (ESI). Descriptive statistics such as means and standard deviations are used to summarize the central tendencies and variability of responses from the 77 participants. These results help to identify general patterns, highlight areas of high or low perception, and offer a foundational understanding of how software developers view and interact with generative AI tools in practice. This descriptive overview serves as a preliminary step before deeper analysis through hypothesis testing in the following section.

4.1.1 Participant Programming Experience

The distribution of respondents by software development experience is presented in

. Among the 77 valid responses, participants demonstrated a range of experience levels, with a majority falling within the early-to-mid-career categories (see table 2). Cumulatively, over 70% of participants had six years of experience or less, indicating that the sample primarily reflects the views of relatively early-career developers.

Table 2

Participant Programming Experience

EXPERIENCE					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Less than 1 year	16	20.8	20.8	20.8
	1–3 years	20	26.0	26.0	46.8
	4–6 years	18	23.4	23.4	70.1
	7–10 years	16	20.8	20.8	90.9
	More than 10 years	7	9.1	9.1	100.0
	Total	77	100.0	100.0	

Participants were asked how frequently they use Generative AI tools, such as GitHub Copilot and ChatGPT, for software development tasks. The results, shown in

Licensed Under Creative Commons Attribution (CC BY-NC)

Table , indicate that a significant majority of respondents use AI tools regularly. Specifically, 40.3% of participants (n = 31) reported using these tools daily, while another 37.7% (n = 29) use them several times a week. Combined, this means that nearly 78% of the sample are frequent users of Generative AI tools. In contrast, 19.5% (n = 15) reported using such tools only occasionally, and just 2.6% (n = 2) reported using them rarely. These findings suggest that Generative AI tools have already been widely adopted among developers, with most incorporating them into their regular workflow.

Licensed Under Creative Commons Attribution (CC BY-NC)

Table 3
GAI tool Usage Frequency

frequency					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Rarely	2	2.6	2.6	2.6
	Occasionally (few times a month)	15	19.5	19.5	22.1
	Several times a week	29	37.7	37.7	59.7
	Daily	31	40.3	40.3	100.0
	Total	77	100.0	100.0	

Participants were asked to identify which Generative AI tool solution they primarily use in their software development work. As presented in Table 5 and **Error! Reference source not found.**, the majority of respondents (68.8%, n = 53) reported using ChatGPT for programming help. This was followed by GitHub Copilot, used by 15.6% (n = 12), while Claude AI, DeepSeek, and Grok were selected by 7.8% (n = 6), 5.2% (n = 4), and 2.6% (n = 2) of participants, respectively. These figures suggest that while a range of tools are being adopted, some are more widely used than others.

Table 4
Most Used GAI tool

AI_TOOL					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	ChatGPT	53	68.8	68.8	68.8
	GitHub Copilot	12	15.6	15.6	84.4
	Grok	2	2.6	2.6	87.0
	Claude AI	6	7.8	7.8	94.8
	DeepSeek	4	5.2	5.2	100.0
	Total	77	100.0	100.0	

The participants were also asked to indicate which coding tasks they commonly apply these tools to. Since this was a multiple-response question, participants could select more than one task. The most frequently reported use was writing new code, selected by 58 participants (75.3%). This was followed by debugging existing code, reported by 49 respondents (63.6%), and writing documentation, mentioned by 35 participants (45.5%). Code refactoring (32 respondents, 41.6%) and generating test cases (25 respondents, 34.6%) were common uses, and detecting vulnerable code was reported by 17 participants (32.7%) (see Figure 5).

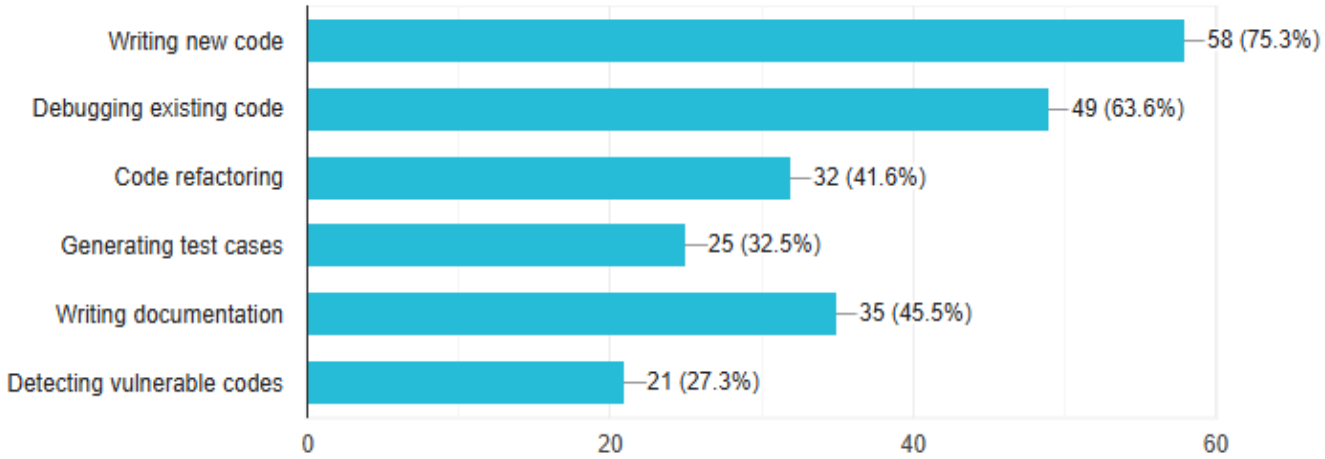


Figure 1
Tasks done in GAI by Developers

4.1.2 Distribution of Respondents by Industry Sector

The results showed that most respondents worked in the Tech / Software sector (33.8%, n = 26), followed by Education (23.4%, n = 18) and Finance (10.4%, n = 8). Healthcare accounted for 6.5% (n = 5), while E-commerce & Retail and Transportation & Logistics each represented 5.2% (n = 4). Other sectors accounted for 7.8% (n = 6). The remaining respondents worked in Energy & Utilities (3.9%), Agriculture (2.6%), and Automotive & Mobility (1.3%).

Licensed Under Creative Commons Attribution (CC BY-NC)

No respondents reported working in Telecommunications or Real Estate & PropTech. Overall, the sample was predominantly composed of respondents from the Tech / Software and Education sectors.

4.2 Descriptive Statistics for the Likert-Scale Constructs

Table 5 summarizes the item-level means and standard deviations for the four constructs. The overall construct means are calculated as the averages of the respective item means.

Table 5

Summary of Descriptive Statistics for the Study Constructs

Construct	Number of items	Item mean range	Item SD range	Overall mean
Perceived Ease of Use (PEU)	5	3.64–4.25	0.914–1.050	3.94
Perceived Security Risk (PSR)	5	3.00–3.16	1.207–1.347	3.07
Trust in AI-Generated Code Security (TRUST)	5	2.70–3.22	1.124–1.288	2.93
Encountered Security Issues (ESI)	5	2.48–3.05	1.102–1.197	2.71

The PEU construct had an overall mean of 3.94, with item means ranging from 3.64 to 4.25. These results indicate generally favorable perceptions of the ease of using generative AI tools for software development. The item standard deviations ranged from 0.914 to 1.050, suggesting some variation in respondents' views. The PSR construct had an overall mean of 3.07, with item means ranging from 3.00 to 3.16. This indicates a moderate level of perceived security risk among the respondents. The item standard deviations ranged from 1.207 to 1.347, showing variation in perceptions of the security risks associated with AI-generated code. For TRUST, the overall mean was 2.93, with item means ranging from 2.70 to 3.22. These findings suggest that respondents did not express consistently high trust in the security of AI-generated code. The item standard deviations ranged from 1.124 to 1.288, indicating differences in respondents' levels of trust. Finally, ESI had an overall mean of 2.71, with item means ranging from 2.48 to 3.05. This suggests variation in respondents' reported experiences with security issues involving AI-generated code. The item standard deviations ranged from 1.102 to 1.197.

Overall, the descriptive results indicate that respondents generally perceived generative AI tools as easy to use, while their perceptions of security risk, trust, and encountered security issues were more moderate. However, these descriptive findings alone cannot establish whether the relationships between the study variables are statistically significant. The following section presents the hypothesis-testing results.

4.3 Inferential Statistics and Hypothesis Testing

Four hypotheses were tested to examine the relationships between generative AI usage frequency, programming experience, AI tool choice, perceived ease of use, perceived security risk, trust in AI-generated code security, and encountered security issues. Spearman's rank-order correlation was used to examine associations between variables, while the Kruskal–Wallis test was used to compare distributions across groups. Statistical significance was assessed at the 5% level ($\alpha = .05$). For the Kruskal–Wallis test in H2, an effect-size estimate was calculated using epsilon-squared. The results are presented separately for each hypothesis.

4.3.1 AI Usage Frequency and Perceived Security Risk

Ho_1: Generative AI usage frequency is significantly associated with developers' perceived security risk of AI-generated code. Spearman's rank-order correlation was conducted to examine the association between generative AI usage frequency and perceived security risk. The analysis revealed a weak negative correlation between the two variables ($\rho = -.126$, $p = .274$, $N = 77$). The association was not statistically significant at the 5% level. The approximate 95% confidence interval for the correlation was [−.342, .099] (see Table 6). Because the interval includes zero, the data do not provide sufficient evidence of a statistically significant association between generative AI usage frequency and perceived security risk. Therefore, H1 was not supported. This finding should not be interpreted as proof that no association exists; rather, the study did not detect a statistically significant relationship in the sample.

Table 6

Spearman's Correlation between AI Usage Frequency and Perceived Security Risk

Variables	Spearman's ρ	p-value	N	Approximate 95% CI	Decision
AI usage frequency and perceived security risk	−.126	.274	77	[−.342, .099]	Not supported

4.3.2 Programming Experience and Trust in AI-Generated Code Security

Ho_2: proposed that programming experience is significantly associated with developers' trust in the security of AI-generated code. As the hypothesis did not specify a directional relationship a priori, a Kruskal–Wallis H test was

Licensed Under Creative Commons Attribution (CC BY-NC)

conducted to examine whether trust scores differed across the five programming-experience groups. The results revealed a statistically significant difference in trust scores across the groups, (H(4)=17.431), (p=.002), (N=77) (see Table 7), with an estimated effect size of 0.186. Therefore, H2 was supported, indicating that developers' trust in the security of AI-generated code differed significantly according to their programming experience. To identify the groups contributing to this difference, Bonferroni-adjusted post hoc pairwise comparisons were conducted. The results showed that developers with less than one year of programming experience reported significantly higher trust than those with 4–6 years ((p=.008)), 7–10 years ((p=.014)), and more than 10 years of experience ((p=.009)). However, the difference between developers with less than one year and those with 1–3 years of experience was not statistically significant ((p=.084)). No other pairwise comparisons were statistically significant after adjustment. These post hoc findings indicate that the observed differences were concentrated between the least-experienced group and several more-experienced groups. Because the direction of the relationship was not specified a priori, this pattern is interpreted as an observed finding rather than confirmation of a pre-established directional prediction.

Table 7

Kruskal–Wallis Test for Programming Experience and Trust in AI-Generated Code Security

Test	H	df	p-value	N	Epsilon-squared (ε^2)	Decision
Kruskal–Wallis	17.431	4	.002	77	.186	Supported

4.3.3 AI Tool Choice and Encountered Security Issues

Ho_3: The generative AI tool used is significantly associated with developers' reported experience of security issues in AI-generated code. A Kruskal–Wallis test was conducted to examine whether reported security issues differed according to the generative AI tool used. The analysis did not reveal a statistically significant difference in encountered security-issue scores across the tool categories, H(4) = 1.555, p = .817, N = 77 (see Table 8). Since the p-value exceeded the .05 significance threshold, the null hypothesis of equal distributions across the tool categories was not rejected. Thus, the analysis did not provide sufficient evidence that reported security issues differed across the AI tools represented in this sample. Therefore, H3 was not supported**.** This result should be interpreted cautiously because the numbers of respondents using the different tools were uneven, including very small groups for some tools. Consequently, the test may have had limited ability to detect differences between tools.

Table 8

Kruskal–Wallis Test for AI Tool Choice and Encountered Security Issues

Test	H	df	p-value	N	Decision
Kruskal–Wallis	1.555	4	.817	77	Not supported

4.3.4 Perceived Ease of Use and Perceived Security Risk

H4**:** Developers' perceived ease of use of generative AI coding tools is significantly associated with their perceived security risk of AI-generated code. Spearman's rank-order correlation was used to examine the association between perceived ease of use and perceived security risk. The results indicated a weak negative correlation ($\rho = -.134$, p = .244, N = 77) (see Table 9). However, the relationship was not statistically significant at the 5% level. The approximate 95% confidence interval for the correlation was [−.349, .091]. This interval includes zero, indicating that the data do not provide sufficient evidence of a statistically significant association between perceived ease of use and perceived security risk. Therefore, H4 was not supported. Although the estimated correlation was negative, the result does not establish that perceived ease of use is associated with lower perceived security risk. A larger sample may help clarify the nature and strength of this relationship.

Table 9

Spearman's Correlation between Perceived Ease of Use and Perceived Security Risk

Variables	Spearman's ρ	p-value	N	Approximate 95% CI	Decision
Perceived ease of use and perceived security risk	−.134	.244	77	[−.349, .091]	Not supported

4.4 Discussion

4.4.1 Adoption, Ease of Use, and Security Awareness

The findings indicate substantial adoption of generative AI among respondents: approximately 78% reported using these tools daily or several times a week. ChatGPT was the most frequently selected tool (68.8%), followed by GitHub Copilot (15.6%). Writing new code and debugging existing code were the most commonly reported tasks. In contrast, only 17 respondents selected detecting vulnerable code as a use case. This pattern suggests that respondents primarily used generative AI to support routine programming activities rather than specifically to identify security

Licensed Under Creative Commons Attribution (CC BY-NC)

weaknesses. The overall perceived ease-of-use mean of 3.94 out of 5 indicates that respondents generally found these tools accessible. This aligns with the Technology Acceptance Model, which identifies perceived ease of use as an important factor in technology acceptance (Davis, 1989). However, usability should not be interpreted as evidence of security. Although generative AI can assist with programming, empirical studies have identified vulnerabilities in AI-generated code (Hamer et al., 2024; Pearce et al., 2023). Consequently, the relatively high ease-of-use rating in this study should be understood as a positive usability assessment, not as confirmation that generated code is secure. The perceived security risk mean of 3.07 indicates a moderate assessment of security concerns, while the trust mean of 2.93 suggests that respondents were not strongly confident in the security of AI-generated code. These results reveal an important distinction between accepting a tool for its practical convenience and trusting its technical outputs. Asare et al. (2024) found that the security effects of GitHub Copilot varied with task difficulty, illustrating that AI-assisted development does not produce uniformly positive or negative security outcomes. Developers therefore need to evaluate generated code in relation to its context and security requirements.

4.4.2 AI Usage Frequency and Perceived Security Risk (H1)

H1 proposed an association between AI usage frequency and perceived security risk. The Spearman correlation showed a weak negative relationship that was not statistically significant ((\rho=-.126, p=.274, N=77)). Therefore, H1 was not supported. The result does not provide sufficient evidence that more frequent AI use is associated with either higher or lower perceived security risk.One possible interpretation is that frequency of use alone does not capture how developers assess security. Developers may use AI tools frequently for routine tasks while varying considerably in their security knowledge, review practices, and understanding of generated code. Research has demonstrated that AI-generated code can contain security weaknesses, but the presence of such weaknesses does not necessarily correspond to users' perceived risk (Hamer et al., 2024; Pearce et al., 2022). The non-significant finding must nevertheless be interpreted cautiously. The sample of 77 respondents may have limited the ability to detect a weak association. Moreover, the study measured perceived risk rather than objectively verified vulnerabilities. Future research could combine larger samples with secure-code analysis to investigate whether usage frequency is related to actual security outcomes.

4.4.3 Programming Experience and Trust in AI-Generated Code Security (H2)

H2 examined whether trust in AI-generated code security differed across programming-experience groups. The Kruskal–Wallis test identified a statistically significant difference (H (4)=17.431, p=.002), with an effect size of (0.186). Thus, H2 was supported at the overall group-comparison level. Descriptively, respondents with less than one year of programming experience reported the highest mean trust score (3.78), whereas those with more than ten years of experience reported the lowest (2.26). Bonferroni-adjusted post hoc comparisons showed that the less-than-one-year group differed significantly from the groups with 4–6 years ((p=.008)), 7–10 years ((p=.014)), and more than 10 years of experience ((p=.009)). Its comparison with the 1–3-year group was not statistically significant ((p=.084)); the remaining pairwise comparisons were also non-significant. This pattern suggests that trust in AI-generated code security varies by programming experience, although the analysis does not establish why these differences occur. One possible explanation is that more experienced developers may have greater exposure to software vulnerabilities and code-review practices, encouraging more critical evaluation of AI-generated outputs. However, this explanation remains tentative because the study did not directly measure security expertise or review behaviour. The findings are also cross-sectional and cannot establish that experience causes lower trust. The result is relevant to developer training. Beginners may benefit from explicit instruction on validating generated code, recognizing common vulnerabilities, and using security-testing tools. More experienced developers may also require guidance on integrating AI into established secure-development workflows. These implications are particularly relevant in Tanzania, where Binamungu et al. (2024) identified gaps in software-safety knowledge and skills among practitioners. The present findings concern trust in AI-generated code security rather than software-safety competence directly; therefore, the two studies provide complementary, but distinct, insights.

4.4.4 AI Tool Choice and Encountered Security Issues (H3)

H3 proposed that encountered security issues differed according to the AI tool used. The Kruskal–Wallis test found no statistically significant difference (H(4)=1.555, p=.817). Accordingly, H3 was not supported. This result should not be interpreted as evidence that all tools have equivalent security performance. The tool groups were uneven, ranging from two Grok users to 53 ChatGPT users, which limits the strength of comparisons, particularly for tools with very few respondents. Furthermore, the outcome measured respondents' reported security issues, not vulnerabilities identified through systematic code testing. Previous research has found that AI coding tools can generate insecure code, although outcomes depend on the tool, prompt, and task (Cotroneo et al., 2024; Siddiq et al., 2024). Future studies should compare tools using consistent programming tasks, objective vulnerability-detection methods, and sufficiently large user groups.

Licensed Under Creative Commons Attribution (CC BY-NC)

4.4.5 Ease of Use and Perceived Security Risk (H4)

H4 examined the relationship between perceived ease of use and perceived security risk. The Spearman correlation was weakly negative and not statistically significant (rho=-.134, p=.244, N=77). Therefore, H4 was not supported. The findings do not establish that easier-to-use AI tools reduce developers' security concerns. This result reinforces the distinction between usability and security assurance. Davis (1989) explains the relevance of ease of use to technology acceptance, but ease of use alone does not establish the reliability or security of a system's outputs. Organizations should therefore complement AI adoption with secure coding standards, human review, automated testing, and vulnerability scanning.

V. CONCLUSION & RECOMMENDATIONS

This study assessed the security implications of generative AI use in software development in Tanzania. The findings indicate that respondents widely use these tools and generally perceive them as easy to use, although their trust in the security of AI-generated code remains relatively low. Programming experience was significantly associated with trust, whereas AI usage frequency, tool choice, and perceived ease of use showed no statistically significant relationships with their respective security-related outcomes. These findings highlight the need to balance the productivity benefits of generative AI with appropriate security precautions.

Developers are encouraged to review and validate AI-generated code, apply secure coding practices, and use automated testing and vulnerability-detection tools before deployment. Training programmes can strengthen developers' ability to identify security weaknesses, particularly among less-experienced programmers who reported higher trust in AI-generated code. Organisations are advised to establish guidelines for responsible AI-assisted software development and maintain human oversight of security-critical code. Future studies would benefit from larger and more diverse samples and objective security testing to assess actual vulnerabilities in AI-generated code. These measures can support safer adoption of generative AI in Tanzania's software-development sector.

Declaration of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding Declaration

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

REFERENCES

Annapureddy, R., Fornaroli, A., & Gatica-Perez, D. (2025). Generative AI Literacy: Twelve Defining Competencies. Digital Government: Research and Practice, 6(1), 1–21. https://doi.org/10.1145/3685680

Asare, O., Nagappan, M., & Asokan, N. (2023). Is GitHub's Copilot as bad as humans at introducing vulnerabilities in code? Empirical Software Engineering, 28(6), 129. https://doi.org/10.1007/s10664-023-10380-1

Banyasz, P., Szadeczky, T., & Vaczi, K. B. (2024). The relationship between generative artificial intelligence and cybersecurity. Proceedings of the Central and Eastern European eDem and eGov Days 2024, 209–215. https://doi.org/10.1145/3670243.3670781

Binamungu, L. P., Maro, S., Justo, G., & Ndanguzi, J. (2024). Assessing Software Safety Knowledge and Skill Gaps in Tanzania. 2024 IST-Africa Conference (IST-Africa), 1–8. https://doi.org/10.23919/IST-Africa63983.2024.10569763

Brown, A., D'Angelo, S., Murillo, A., Jaspan, C., & Green, C. (2024). Identifying the Factors That Influence Trust in AI Code Completion. Proceedings of the 1st ACM International Conference on AI-Powered Software, 1–9. https://doi.org/10.1145/3664646.3664757

Bull, C., & Kharrufa, A. (2023). Generative AI Assistants in Software Development Education: A vision for integrating Generative AI into educational practice, not instinctively defending against it. https://doi.org/10.48550/ARXIV.2303.13936

Cotroneo, D., Improta, C., Liguori, P., & Natella, R. (2024). Vulnerabilities in AI Code Generators: Exploring Targeted Data Poisoning Attacks. Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension, 280–292. https://doi.org/10.1145/3643916.3644416

Dakhel, A. M., Majdinasab, V., Nikanjam, A., Khomh, F., Desmarais, M. C., Ming, Z., & Jiang. (2023). GitHub Copilot AI pair programmer: Asset or Liability? (arXiv:2206.15331). arXiv. https://doi.org/10.48550/arXiv.2206.15331

Licensed Under Creative Commons Attribution (CC BY-NC)

Davila, N., Wiese, I., Steinmacher, I., Lucio Da Silva, L., Kawamoto, A., Favaro, G. J. P., & Nunes, I. (2024). An Industry Case Study on Adoption of AI-based Programming Assistants. Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice, 92–102. https://doi.org/10.1145/3639477.3643648

Davis, F. D. (1989). Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. MIS Quarterly, 13(3), 319. https://doi.org/10.2307/249008

Falcão, F. D. S., & Canedo, E. D. (2024). Investigating Software Development Teams Members' Perceptions of Data Privacy in the Use of Large Language Models (LLMs). Proceedings of the XXIII Brazilian Symposium on Software Quality, 373–382. https://doi.org/10.1145/3701625.3701675

Fu, Y., Liang, P., Tahir, A., Li, Z., Shahin, M., Yu, J., & Chen, J. (2025). Security Weaknesses of Copilot-Generated Code in GitHub Projects: An Empirical Study. ACM Transactions on Software Engineering and Methodology, 3716848. https://doi.org/10.1145/3716848

Hamer, S., d'Amorim, M., & Williams, L. (2024). Just another copy and paste? Comparing the security vulnerabilities of ChatGPT generated code and StackOverflow answers. 2024 IEEE Security and Privacy Workshops (SPW), 87–94. https://doi.org/10.1109/SPW63631.2024.00014

He, J., & Vechev, M. (2023). Large Language Models for Code: Security Hardening and Adversarial Testing. Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, 1865–1879. https://doi.org/10.1145/3576915.3623175

He, R., Cao, J., & Tan, T. (2025). Generative Artificial Intelligence: A Historical Perspective. National Science Review, nwaf050. https://doi.org/10.1093/nsr/nwaf050

Klemmer, J. H., Horstmann, S. A., Patnaik, N., Ludden, C., Burton, C., Powers, C., Massacci, F., Rahman, A., Votipka, D., Lipford, H. R., Rashid, A., Naiakshina, A., & Fahl, S. (2024). Using AI Assistants in Software Development: A Qualitative Study on Security Practices and Concerns. Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, 2726–2740. https://doi.org/10.1145/3658644.3690283

Kudriavtseva, A., Hotak, N. A., & Gadyatskaya, O. (2025). My Code Is Less Secure with Gen AI: Surveying Developers' Perceptions of the Impact of Code Generation Tools on Security. Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing, 1637–1646. https://doi.org/10.1145/3672608.3707778

May, R., Krüger, J., & Leich, T. (2024). SoK: How Artificial-Intelligence Incidents Can Jeopardize Safety and Security. Proceedings of the 19th International Conference on Availability, Reliability and Security, 1–12. https://doi.org/10.1145/3664476.3664510

Mellinger, C. D., & Hanson, T. A. (2021). Methodological considerations for survey research: Validity, reliability, and quantitative analysis. Linguistica Antverpiensia, New Series – Themes in Translation Studies, 19. https://doi.org/10.52034/lanstts.v19i0.549

Ottenbacher, K. J., Hsu, Y., Granger, C. V., & Fiedler, R. C. (1996). The reliability of the functional independence measure: A quantitative review. Archives of Physical Medicine and Rehabilitation, 77(12), 1226–1232. https://doi.org/10.1016/S0003-9993(96)90184-7

Pearce, H., Tan, B., Ahmad, B., Karri, R., & Dolan-Gavitt, B. (2023). Examining Zero-Shot Vulnerability Repair with Large Language Models. 2023 IEEE Symposium on Security and Privacy (SP), 2339–2356. https://doi.org/10.1109/SP46215.2023.10179324

Perry, N., Srivastava, M., Kumar, D., & Boneh, D. (2023a). Do Users Write More Insecure Code with AI Assistants? Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, 2785–2799. https://doi.org/10.1145/3576915.3623157

Perry, N., Srivastava, M., Kumar, D., & Boneh, D. (2023b). Do Users Write More Insecure Code with AI Assistants? Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, 2785–2799. https://doi.org/10.1145/3576915.3623157

Perry, N., Srivastava, M., Kumar, D., & Boneh, D. (2023c). Do Users Write More Insecure Code with AI Assistants? Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, 2785–2799. https://doi.org/10.1145/3576915.3623157

Pesesse, P., Vanderthommen, M., Durieux, N., Zubkov, M., & Demoulin, C. (2024). Clinical Value and Reliability of Quantitative Assessments of Lumbosacral Nerve Root Using Diffusion Tensor and Diffusion Weighted MR Imaging: A Systematic Review. Journal of Magnetic Resonance Imaging, jmri.29213. https://doi.org/10.1002/jmri.29213

Pinto, G., De Souza, C., Rocha, T., Steinmacher, I., Souza, A., & Monteiro, E. (2024). Developer Experiences with a Contextualized AI Coding Assistant: Usability, Expectations, and Outcomes. Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering - Software Engineering for AI, 81–91. https://doi.org/10.1145/3644815.3644949

Licensed Under Creative Commons Attribution (CC BY-NC)

Rachmatullah, A., Vandenberg, J., Shin, S., & Wiebe, E. (2023). Cross-Country Variation in (Binary) Gender Differences in Secondary School Students' CS Attitudes: Re-Validating and Generalizing a CS Attitudes Scale. ACM Transactions on Computing Education, 23(4), 1–22. https://doi.org/10.1145/3630014

Rogers, R. W. (1975). A Protection Motivation Theory of Fear Appeals and Attitude Change1. The Journal of Psychology, 91(1), 93–114. https://doi.org/10.1080/00223980.1975.9915803

Ryan, T. J., Alarcon, G. M., Walter, C., Gamble, R., Jessup, S. A., Capiola, A., & Pfahler, M. D. (2019). Trust in Automated Software Repair: The Effects of Repair Source, Transparency, and Programmer Experience on Perceived Trustworthiness and Trust. In A. Moallem (Ed.), HCI for Cybersecurity, Privacy and Trust (Vol. 11594, pp. 452–470). Springer International Publishing. https://doi.org/10.1007/978-3-030-22351-9_31

Sandoval, G., Pearce, H., Nys, T., Karri, R., Garg, S., & Dolan-Gavitt, B. (2025). Lost at C: A User Study on the Security Implications of Large Language Model Code Assistants.

Sankhe, P., Patil, N., Ghorpade, M., Prasad, P., & Linkesh, M. (2025). Empirical Analysis of AI-Assisted Code Generation Tools Impact on Code Quality, Security and Developer Productivity. International Journal For Multidisciplinary Research, 7(6), 61350. https://doi.org/10.36948/ijfmr.2025.v07i06.61350

Siddiq, M. L., Roney, L., Zhang, J., & Santos, J. C. D. S. (2024). Quality Assessment of ChatGPT Generated Code and their Use by Developers. Proceedings of the 21st International Conference on Mining Software Repositories, 152–156. https://doi.org/10.1145/3643991.3645071

Tuli, F. (2011). The Basis of Distinction Between Qualitative and Quantitative Research in Social Science: Reflection on Ontological, Epistemological and Methodological Perspectives. Ethiopian Journal of Education and Sciences, 6(1). https://doi.org/10.4314/ejesc.v6i1.65384

Wang, R., Cheng, R., Ford, D., & Zimmermann, T. (2024). Investigating and Designing for Trust in AI-powered Code Generation Tools. The 2024 ACM Conference on Fairness, Accountability, and Transparency, 1475–1493. https://doi.org/10.1145/3630106.3658984

Licensed Under Creative Commons Attribution (CC BY-NC)