INTERNATIONAL JOURNAL OF ADVANCED RESEARCH
ISSN: 3106-7980: 3106-7972
Volume 3, Issue 3, 2026

HUMAN-CENTRIC SECURITY ASSURANCE IN AI-AUGMENTED SOFTWARE ENGINEERING ECOSYSTEMS: EXPLORING DEVELOPER STRATEGIES FOR VALIDATING LLM-GENERATED CODE ARTIFACTS

Muhammad Vahaj Ur Rehman[*1], Noor Nabi Dahar[2], Tabinda Arzoo[3], Jawaid Ahmed Siddiqui[4], Ashmira Majeed[5]

*[*1]MSc Research Scholar, GEM Alpine Business School, France*
[2]Lecturer, Computer Science Department, Sukkur IBA University, Pakistan
[3]PhD Scholar, Department of Computer Engineering, Istanbul Atlas University, Türkiye
[4]Assistant Professor, Computer Science Department, Sukkur IBA University, Pakistan
[5]Visiting Lecturer, Department of Computer Science, COMSATS University Islamabad, Vehari Campus, Pakistan

[*1]vahaj2000@outlook.com, [2]noornabi@iba-suk.edu.pk,[3]tabinda.nouman@gmail.com, [4]jawaid@iba-suk.edu.pk, [5]ashmiramajeed14@gmail.com

DOI: https://doi.org/10.5281/zenodo.21871602

Keywords
Human-Centric Security Assurance; Large Language Models; AI-Augmented Software Engineering; Human-Centered AI; Secure Software Development

Article History
Received: 02 May 2026
Accepted: 29 May 2026
Published: 31 July 2026

Corresponding Author: * Muhammad Vahaj Ur Rehman

Abstract

The rapid adoption of Large Language Models (LLMs) has transformed software engineering by enabling developers to automate code generation, debugging, testing, and documentation. While these technologies enhance productivity, they also introduce security vulnerabilities, inaccurate logic, dependency risks, and governance challenges that require effective human oversight. This study explores how software developers in Pakistan validate LLM-generated code artifacts and implement human-centric security assurance within AI-augmented software engineering ecosystems. Guided by an interpretivist philosophy, the study employed an exploratory qualitative design and conducted semi-structured interviews with 17 software professionals working across software houses, IT export companies, FinTech organizations, technology startups, multinational firms, and public-sector digital organizations. Data were analyzed using Reflexive Thematic Analysis. The findings identified six interconnected themes highlighting the importance of human judgment, multi-layered validation, security awareness, organizational governance, continuous human-AI learning, and Human-Centric Security Assurance as a socio-technical capability. The study extends Human-Centered AI Theory by demonstrating that secure AI-assisted software engineering depends on the interaction of developer expertise, structured validation practices, and organizational governance rather than AI capabilities alone. The findings provide theoretical insights and practical recommendations for strengthening secure AI adoption within Pakistan's evolving software industry and other emerging digital economies.

1. Introduction

1.1 Background of the Research

The rapid evolution of Artificial Intelligence (AI), particularly Large Language Models (LLMs), has fundamentally transformed the landscape of modern software engineering. Unlike traditional software development tools that primarily automate repetitive programming tasks, contemporary LLM-based coding assistants actively participate in software design, code generation, debugging, testing, documentation, and code optimization. This transformation has shifted software development from a purely human-driven activity toward collaborative human–AI ecosystems, where developers increasingly rely on intelligent systems to accelerate programming tasks while maintaining responsibility for the quality and security of software products. Consequently, software engineering is evolving into an AI-augmented discipline in which human expertise and machine intelligence jointly contribute to software creation and maintenance (Abrahão et al., 2025; Akhtar & Aftab, 2025).

Recent advances in AI-assisted software engineering have enabled developers to generate substantial portions of source code through natural language prompts, significantly reducing development time while improving coding efficiency. LLM-powered tools now support multiple phases of the software development lifecycle, including requirement interpretation, architecture design, automated code completion, bug detection, unit test generation, documentation, and code refactoring. These capabilities have altered conventional software engineering workflows by enabling developers to focus more on problem-solving, system architecture, and decision-making while delegating routine programming activities to AI systems. As organizations increasingly integrate AI into software development pipelines, AI-augmented software engineering ecosystems are becoming central to digital transformation strategies across industries (Madupati et al., 2025; Danyaro et al., 2025).

The growing adoption of AI-assisted programming has also expanded the concept of collaborative intelligence within software engineering. Rather than replacing developers, LLMs function as intelligent collaborators that augment human capabilities by generating alternative coding solutions, identifying programming errors, explaining unfamiliar code, and recommending software improvements. This collaborative relationship allows software engineers to leverage AI-generated knowledge while applying professional judgment to validate and refine machine-generated outputs. Recent studies suggest that effective software engineering increasingly depends on the complementary strengths of human creativity, contextual understanding, ethical reasoning, and AI-driven computational efficiency (Babulak et al., 2026; Mondal, 2026).

Despite these considerable advantages, the widespread integration of AI-generated code has introduced significant concerns regarding software quality, reliability, transparency, and cybersecurity. LLM-generated code may contain logical inconsistencies, hallucinated programming constructs, outdated libraries, insecure coding patterns, hidden software dependencies, or vulnerabilities that are difficult to detect through conventional development practices. Consequently, developers can no longer assume that AI-generated outputs are inherently trustworthy. Instead, rigorous human validation has become an essential component of responsible AI-assisted software engineering, requiring developers to critically evaluate generated code before deployment within production environments (Kashif et al., 2026; Danyaro et al., 2025).

These global developments are particularly relevant to Pakistan, whose software industry has experienced substantial growth during the past decade. The expansion of software houses, technology startups, multinational development centers, FinTech companies, and IT-enabled service providers has significantly increased the demand for efficient software development practices. Simultaneously, the country's growing information technology exports and increasing

participation in global outsourcing markets have encouraged organizations to adopt AI-powered software development tools to enhance productivity, reduce project delivery time, and improve competitiveness. As Pakistani software developers increasingly integrate LLM-assisted coding tools into their daily workflows, AI has become an important component of contemporary software engineering practices within the country.

However, the rapid adoption of AI-assisted software development in Pakistan has not been accompanied by equally mature governance mechanisms. Organizations vary considerably in their implementation of AI governance policies, secure coding standards, DevSecOps practices, and formal validation procedures for AI-generated code. Large multinational firms often possess structured security review processes and established software quality assurance frameworks, whereas many small and medium-sized software companies continue to rely heavily on individual developer expertise and informal review practices. Such variations create uncertainty regarding how developers evaluate AI-generated code and ensure its security before integration into software products. Understanding these human validation practices is therefore essential for promoting trustworthy AI-assisted software engineering within Pakistan's rapidly expanding digital economy (Fein et al., 2026; Akhtar & Aftab, 2025).

1.2 Problem Statement

The increasing adoption of Large Language Models has transformed software development by enabling developers to generate source code, automate debugging, produce documentation, and accelerate software delivery. Although these technologies substantially improve productivity and reduce development effort, they simultaneously introduce critical risks associated with software security and reliability. AI-generated code may contain insecure programming patterns, inaccurate logic, hidden dependencies, obsolete libraries, privacy vulnerabilities, licensing conflicts, and non-compliance with organizational coding standards. Since LLMs generate code probabilistically rather than through deterministic software engineering principles, developers cannot assume that generated outputs satisfy security and quality requirements without systematic verification (Shahzad & Iqbal, 2025; Zardari et al., 2026).

These challenges may be particularly pronounced within Pakistan's software industry, where organizations exhibit varying levels of technological maturity, cybersecurity capabilities, and AI governance readiness. While many Pakistani software companies have rapidly embraced AI-assisted development to improve productivity and international competitiveness, standardized validation procedures, formal AI governance frameworks, and consistent secure coding practices remain uneven across organizations. Consequently, developers frequently rely on personal experience, organizational norms, and informal review mechanisms when deciding whether AI-generated code is sufficiently secure for deployment (Mittal, 2025; Harrath et al., 2025).

Existing literature has primarily concentrated on evaluating the technical performance, accuracy, and productivity benefits of LLM-generated code, while comparatively little attention has been devoted to understanding how developers themselves validate AI-generated software artifacts before implementation. Moreover, empirical evidence from emerging software ecosystems such as Pakistan remains extremely limited. Addressing this gap is important for understanding the human-centric security assurance practices that underpin trustworthy AI-assisted software engineering and for informing organizational policies that support secure AI adoption within Pakistan's rapidly growing software industry (Torres, 2025; Zardari et al., 2026).

1.3 Research Questions

RQ1: How do software developers in Pakistan validate LLM-generated code artifacts before integrating them into software projects?

RQ2: What human-centric security assurance strategies are employed by Pakistani software

developers during AI-assisted software engineering?
RQ3: How do organizational policies, project characteristics, and developer experience influence secure validation practices within Pakistan's software industry?

1.4 Research Objectives

- To explore how software developers in Pakistan validate AI-generated code artifacts.
- To identify human-centric security assurance strategies used in AI-assisted software engineering.
- To examine organizational and contextual factors influencing secure AI-assisted coding practices in Pakistan.
- To develop a conceptual framework explaining human-centric security assurance within Pakistan's AI-augmented software engineering ecosystem.

1.5 Significance of the Research

This study contributes to the growing body of knowledge on AI-augmented software engineering by examining security assurance from a human-centered perspective. While existing research has predominantly focused on improving the technical capabilities of LLMs, considerably less attention has been directed toward understanding the human decision-making processes that determine whether AI-generated code is sufficiently secure for practical deployment. By exploring how Pakistani software developers evaluate, validate, and refine AI-generated code artifacts, this study extends software engineering literature beyond algorithmic performance toward the human practices that sustain trustworthy AI-assisted development (Patil, 2026; Hoffmann, 2025).
From a theoretical perspective, the study advances Human-Centered AI by emphasizing the indispensable role of developer expertise, critical judgment, accountability, and contextual reasoning in AI-assisted software engineering. Rather than viewing security solely as a technical property of software artifacts, the research conceptualizes security assurance as a socio-technical process emerging from continuous interaction among developers, AI systems, organizational governance structures, and secure software engineering practices. This perspective contributes to emerging discussions on responsible AI, trustworthy software engineering, and human–AI collaboration (Patil, 2026; Hoffmann, 2025).
The study also contributes to AI governance and DevSecOps literature by identifying organizational mechanisms that support secure AI adoption within software development environments. Understanding how developers perform validation, security reviews, and quality assurance can assist organizations in designing governance frameworks that integrate human oversight with AI-assisted development pipelines. Such insights are valuable for strengthening secure software engineering practices while promoting responsible organizational adoption of AI technologies (Oakes, 2025; Ghosh et al., 2026).
Practically, the findings are expected to benefit Pakistan's rapidly expanding software industry by providing evidence-based recommendations for software houses, technology startups, FinTech organizations, multinational development centers, and public-sector digital initiatives. The proposed human-centric security assurance framework may support the development of standardized validation procedures, secure coding guidelines, developer training initiatives, and organizational AI governance policies that enhance trust in AI-generated software artifacts. Furthermore, the findings can inform national discussions concerning AI policy development in emerging economies, where balancing technological innovation with cybersecurity, software quality, and responsible AI adoption has become increasingly important for sustaining long-term digital competitiveness (Kapadia & Islam, 2026; Oakes, 2025).

2. Literature Review and Theoretical Foundation

Recent scholarship further demonstrates that successful AI integration depends not only on

technological advancement but also on broader organizational, educational, and governance capabilities. Research highlights the importance of AI-enabled intelligent systems, digital twins, and autonomous technologies for enhancing decision support and operational efficiency across complex environments (Abouelenin & Jammal, 2026; Sharma & Sharma, 2026). Organizational transformation through AI is also influenced by digital innovation, workforce capabilities, and adaptive governance that support sustainable technological adoption (Rasheed & Ali, 2026; Yaseen et al., 2026). Studies conducted in education emphasize that effective AI implementation requires ethical awareness, responsible use, and continuous human oversight to ensure trustworthy outcomes (He, 2026; Rehmat et al., 2025). Similarly, research in sustainability, finance, and SME development underscores the importance of structured governance, stakeholder collaboration, and contextual decision making for successful AI-enabled systems (Pracucci & Giovanardi, 2026; Ali & Rasheed, 2026; Ashraf et al., 2025). Collectively, these studies reinforce the argument that AI creates greater value when technological innovation is complemented by human judgment, organizational readiness, and responsible governance.

2.1 AI-Augmented Software Engineering Ecosystems

Artificial Intelligence (AI), particularly Large Language Models (LLMs), has fundamentally transformed software engineering by shifting development from a purely human-centered activity to an AI-augmented collaborative process. Modern AI-powered programming assistants are capable of generating code, debugging applications, producing documentation, refactoring software, and recommending architectural improvements, enabling developers to complete complex programming tasks more efficiently than traditional development approaches. Rather than replacing software engineers, AI increasingly functions as an intelligent collaborator that augments developers' cognitive capabilities while allowing them to focus on higher-order activities such as software architecture, problem solving, and strategic decision making (Abrahão et al., 2025; Akhtar & Aftab, 2025).

The emergence of AI-augmented software engineering ecosystems represents a broader transition toward human-AI collaboration throughout the software development lifecycle. These ecosystems integrate intelligent programming assistants into coding, testing, deployment, maintenance, and knowledge management processes, creating continuous interactions between developers and AI systems. Such collaboration has accelerated software development while introducing new responsibilities related to code verification, security assurance, and ethical AI usage (Madupati et al., 2025; Babulak et al., 2026). Furthermore, advances in agentic AI suggest that future software engineering will increasingly involve autonomous agents capable of planning, generating, and refining software artifacts with minimal human intervention. Consequently, software developers are evolving from code creators to supervisors and validators responsible for ensuring the reliability and security of AI-generated outputs (Mondal, 2026; Fein et al., 2026).

2.2 LLM-Generated Code Artifacts and Software Security Challenges

LLM-generated code has become one of the most significant applications of generative AI within software engineering. Contemporary AI coding assistants support automated code generation, bug fixing, code completion, test case generation, documentation, and software optimization, resulting in substantial improvements in developer productivity and project efficiency. Systematic reviews consistently report that AI-assisted coding reduces programming time while increasing accessibility for novice developers and supporting experienced programmers in solving complex coding problems (Danyaro et al., 2025; Shahzad & Iqbal, 2025).

Despite these benefits, the reliability and security of AI-generated code remain major concerns. Since LLMs predict code based on probabilistic

language patterns rather than software engineering principles, generated outputs may contain logical errors, hallucinated functions, insecure APIs, outdated libraries, hidden dependencies, licensing conflicts, and exploitable vulnerabilities. Developers therefore cannot assume that AI-generated code satisfies security, compliance, or organizational quality standards without careful validation (Kashif et al., 2026). Moreover, increasing reliance on AI-generated artifacts raises privacy risks, particularly when proprietary source code or confidential organizational information is incorporated into AI prompts. Emerging applications involving blockchain, cloud computing, and Internet of Things (IoT) systems further increase the importance of secure validation due to the potentially severe consequences of deploying insecure AI-generated software (Zaidi et al., 2026). Consequently, recent research advocates integrating structured evaluation mechanisms, human review, and multimodal assessment techniques to improve the quality and trustworthiness of AI-generated software artifacts (Haseeb et al., 2026).

2.3 Human-Centric Security Assurance in AI-Assisted Development

The increasing integration of AI into software engineering has shifted attention from purely technical security mechanisms toward human-centric security assurance. Human oversight remains essential because AI systems cannot independently evaluate contextual requirements, organizational policies, ethical considerations, or project-specific security constraints. Developers therefore serve as the final decision-makers who determine whether AI-generated code is sufficiently secure, reliable, and appropriate for deployment (Patil, 2026; Hoffmann, 2025).

Human-centric security assurance encompasses explainability, accountability, trust calibration, and informed decision making throughout AI-assisted software development. Explainable AI enables developers to understand why particular coding recommendations are generated, thereby improving confidence in validation decisions. Accountability ensures that responsibility for software quality remains with developers rather than AI systems, reinforcing the importance of professional judgment during code review. Trust calibration further emphasizes that developers should neither blindly accept nor automatically reject AI-generated outputs but instead develop balanced trust based on experience, contextual knowledge, and systematic verification (Alami et al., 2025). This collaborative perspective aligns with emerging views of agentic information systems, in which humans and AI continuously exchange knowledge while maintaining clear human responsibility for security-critical decisions (Katsiuba, 2026). Integrating explainable AI with secure DevSecOps practices therefore represents an important step toward trustworthy AI-assisted software engineering (Mittal, 2025).

2.4 Organizational AI Governance and DevSecOps Practices

As AI-assisted software engineering becomes increasingly widespread, organizations must establish governance mechanisms that ensure responsible, secure, and transparent AI adoption. AI governance extends beyond technical implementation by incorporating organizational policies, ethical standards, risk management practices, and compliance procedures that regulate the use of AI throughout software development. Within DevSecOps environments, governance promotes continuous security validation by embedding automated testing, vulnerability assessment, policy enforcement, and human oversight throughout the software development lifecycle (Oakes, 2025).

Organizational learning also plays a critical role in developing secure AI capabilities. Developers require continuous training in prompt engineering, AI validation techniques, cybersecurity awareness, and responsible AI practices to effectively collaborate with intelligent programming assistants. Strategic AI governance therefore combines technical controls with organizational capability development and knowledge sharing (Kapadia & Islam, 2026; Lusi et al., 2026). These issues are particularly relevant

within Pakistan, where software organizations differ considerably in governance maturity. Large multinational corporations generally possess structured AI governance frameworks and secure DevSecOps practices, whereas many startups and small software firms continue to rely on informal review procedures and individual developer expertise. Such variation highlights the need to understand how governance influences developers' security assurance behaviors within Pakistan's rapidly expanding software industry (Khan et al., 2026).

2.5 Human-Centered AI Theory

This study adopts Human-Centered AI Theory as its theoretical foundation because it emphasizes that AI should augment rather than replace human intelligence. The theory argues that humans remain central to decision making, accountability, and ethical responsibility even when intelligent systems perform complex tasks autonomously. Within software engineering, Human-Centered AI explains why developers continue to play a critical role in evaluating AI-generated code despite advances in LLM capabilities (Patil, 2026).

The theory comprises several interrelated principles that align closely with secure AI-assisted software development. Human agency emphasizes developers' authority to make final security decisions. Human oversight ensures continuous monitoring and verification of AI-generated outputs. Developer accountability maintains responsibility for software quality and cybersecurity regardless of AI assistance. Human-AI collaboration promotes complementary interactions between intelligent programming assistants and software professionals, while security assurance behavior reflects developers' systematic evaluation of AI-generated code before deployment (Hoffmann, 2025; Alami et al., 2025). Collectively, these principles provide an appropriate theoretical lens for examining how Pakistani software developers balance productivity with security while working within AI-augmented software engineering ecosystems (Abrahão et al., 2025; Babulak et al., 2026).

2.6 Research Gaps

Although research on AI-assisted software engineering has expanded rapidly, several important gaps remain. First, existing studies primarily evaluate the technical performance and coding capabilities of LLMs rather than developers' validation behaviors during software development. Second, current security research focuses predominantly on identifying technical vulnerabilities while providing limited attention to human-centric security assurance practices. Third, qualitative evidence exploring software developers' lived experiences with AI-generated code remains scarce. Fourth, little empirical research examines how organizational governance and DevSecOps maturity influence secure AI-assisted coding practices. Fifth, studies conducted within emerging software ecosystems, particularly Pakistan, remain extremely limited despite the country's rapidly growing software industry and increasing adoption of AI-assisted development tools. Finally, no comprehensive qualitative framework currently explains how human oversight, organizational governance, security validation, and AI collaboration collectively shape Human-Centric Security Assurance within Pakistan's AI-augmented software engineering ecosystem. Addressing these gaps provides the foundation for the present study and supports the development of a context-specific conceptual framework explaining secure AI-assisted software engineering practices in Pakistan.

3. Methodology

3.1 Research Philosophy

This study adopts an interpretivist research philosophy to explore how software developers construct meaning from their experiences of validating Large Language Model (LLM)-generated code within AI-assisted software engineering environments. Interpretivism assumes that reality is socially constructed and that individuals interpret phenomena based on their professional experiences, organizational environments, and contextual understanding. Since developers' security validation practices are influenced by organizational policies, technical expertise, project

requirements, and personal judgment, an interpretivist perspective is appropriate for understanding the complexities of human-centric security assurance. Rather than objectively measuring security outcomes, this study seeks to understand how developers perceive, evaluate, and validate AI-generated code before integrating it into software systems.

3.2 Research Design

This study employs an exploratory qualitative research design to investigate developers' security assurance practices in AI-assisted software engineering. A qualitative approach was selected because limited empirical evidence exists regarding how software professionals validate AI-generated code within real-world software development environments, particularly in Pakistan. The exploratory nature of the study allows participants to describe their experiences, decision-making processes, organizational practices, and security concerns in depth without restricting their responses to predefined variables.

Semi-structured interviews were adopted as the primary data collection method because they provide flexibility to explore emerging issues while maintaining consistency across interviews. The collected data were analyzed using Reflexive Thematic Analysis (RTA), following the six-phase framework proposed by Braun and Clarke. The analytical process involved familiarization with the interview data, generation of initial codes, development of candidate themes, review and refinement of themes, definition and naming of themes, and preparation of the final report. Reflexive Thematic Analysis was selected because it facilitates rich interpretation of participants' experiences while acknowledging the researcher's active role in knowledge construction (Braun & Clarke, 2019; Braun et al., 2022; Terry et al., 2017).

3.3 Research Context

The study was conducted within Pakistan's rapidly expanding software industry, which has experienced substantial growth through increasing IT exports, digital transformation initiatives, technology entrepreneurship, and international software outsourcing. Pakistan hosts a diverse software ecosystem comprising software houses, IT export companies, FinTech organizations, technology startups, AI-enabled software development firms, multinational software companies, IT consultancies, and public-sector digital organizations. These organizations increasingly employ AI-powered programming assistants, including ChatGPT, GitHub Copilot, Gemini, and Claude, to improve software development efficiency, automate coding tasks, and enhance developer productivity.

Despite the widespread adoption of AI-assisted software engineering, organizational readiness for AI governance and cybersecurity differs considerably across Pakistani organizations. While multinational corporations and mature software firms often maintain structured DevSecOps practices, AI governance frameworks, and formal code review procedures, many startups and small software houses rely on informal validation processes and individual developer expertise. These differences make Pakistan an appropriate research context for exploring how developers validate AI-generated code under varying organizational conditions and security governance environments.

3.4 Sampling Strategy

Purposive sampling was employed to recruit information-rich participants capable of providing meaningful insights into AI-assisted software engineering practices. Participants were selected based on their professional experience with AI-powered programming assistants and their active involvement in software development projects. Eligible participants included Senior Software Developers, Software Engineers, DevSecOps Engineers, Software Architects, Technical Leads, and AI Software Engineers employed in Pakistani software organizations.

To ensure participants possessed adequate practical knowledge, the inclusion criteria required respondents to have at least two years of professional software development experience and regular use of LLM-based coding assistants such as

ChatGPT, GitHub Copilot, Gemini, or Claude. A total of 17 participants were interviewed, which was considered sufficient to achieve thematic saturation while ensuring representation from different organizational types and technology sectors across Pakistan.

3.5 Data Collection

Primary data were collected through semi-structured interviews conducted between software professionals working in different regions of Pakistan. Each interview lasted approximately 45 to 60 minutes and was conducted either online using video conferencing platforms or through face-to-face meetings, depending on participant availability. An interview guide comprising open-ended questions was used to encourage participants to discuss their experiences regarding AI-generated code validation, security assurance strategies, organizational governance, trust in AI-generated artifacts, and software quality practices. Prior to each interview, informed consent was obtained from participants. With participants' permission, all interviews were audio recorded to ensure accurate capture of responses. The recordings were subsequently transcribed verbatim using professional transcription procedures before being prepared for qualitative analysis.

3.6 Data Analysis

The interview data were analyzed using Reflexive Thematic Analysis, following Braun and Clarke's six-phase analytical framework. Initially, the researchers repeatedly reviewed the interview transcripts to become familiar with the data. Subsequently, meaningful segments of text were assigned descriptive codes representing developers' experiences and perspectives regarding AI-assisted software engineering. Similar codes were then grouped into broader categories that reflected recurring patterns across participants. These categories were refined into coherent themes through iterative comparison and interpretation. Finally, the themes were reviewed, defined, and interpreted to develop a comprehensive understanding of Human-Centric Security Assurance within Pakistan's AI-augmented software engineering ecosystem.

3.7 Ethical Considerations

The study adhered to established ethical principles throughout the research process. Participants received detailed information regarding the purpose of the study before providing informed consent. Participation was entirely voluntary, and respondents were informed of their right to withdraw at any stage without consequence. To protect participants' privacy, personal identifiers were removed from interview transcripts and replaced with participant identification codes. All interview recordings and transcripts were stored securely on password-protected devices accessible only to the research team. Furthermore, ethical approval was obtained from the relevant institutional review committee before the commencement of data collection.

3.8 Participant Profile

A total of 17 software professionals participated in the study. The participants represented diverse software organizations operating across Pakistan, including software houses, IT export companies, FinTech firms, technology startups, AI development companies, multinational software firms, IT consultancies, and public-sector digital organizations. To capture diverse perspectives, participants were recruited from the country's major technology hubs, including Islamabad, Lahore, Karachi, Faisalabad, and Peshawar.

Table 1. Participant Profile

ID	Position	Experience (Years)	Organization Type	Industry Domain	City	Primary AI Tool	Primary Responsibilities
P1	Senior Software Developer	9	Software House	Web Development	Islamabad	GitHub Copilot	Backend Development and Code Review
P2	Software Engineer	4	IT Export Company	Enterprise Solutions	Lahore	ChatGPT	Full Stack Software Development
P3	DevSecOps Engineer	10	FinTech Company	Financial Technology	Karachi	GitHub Copilot	Security Automation and CI/CD Management
P4	Software Architect	13	Multinational Software Company	Enterprise Systems	Islamabad	ChatGPT	Software Architecture and Technical Governance
P5	Technical Lead	11	Technology Startup	Software as a Service (SaaS)	Lahore	Gemini	Team Leadership and Solution Delivery
P6	AI Software Engineer	6	AI Startup	Artificial Intelligence	Islamabad	Claude	AI Application Development
P7	Software Engineer	5	Software House	Mobile Applications	Faisalabad	ChatGPT	Mobile Application Development
P8	Senior Software Developer	12	IT Export Company	Cloud Computing	Karachi	GitHub Copilot	Cloud Software Engineering
P9	DevSecOps Engineer	8	Public Sector Organization	Digital Government	Islamabad	ChatGPT	Secure Software Delivery
P10	Software Architect	15	FinTech Company	Digital Banking	Lahore	GitHub Copilot	Enterprise Software Architecture
P11	Technical Lead	9	Technology Startup	E-commerce	Peshawar	Gemini	Product Development and Technical Supervision
P12	AI Software Engineer	5	AI Development Company	Machine Learning	Islamabad	Claude	AI Model Integration
P13	Software Engineer	3	Software House	Enterprise Resource Planning	Karachi	ChatGPT	Backend Software Development
P14	Senior Software Developer	8	IT Consultancy	Software Services	Faisalabad	GitHub Copilot	Software Quality Assurance and Development
P15	DevSecOps Engineer	9	Multinational Software Company	Cloud Security	Lahore	GitHub Copilot	DevSecOps and Compliance Management

ID	Position	Experience (Years)	Organization Type	Industry Domain	City	Primary AI Tool	Primary Responsibilities
P16	Technical Lead	14	Public Sector Organization	Government Digital Services	Islamabad	ChatGPT	Digital Transformation and Project Management
P17	AI Software Engineer	7	Technology Startup	Generative AI Solutions	Peshawar	Gemini	Intelligent Software Development and LLM Integration

The participant profile demonstrates considerable diversity in terms of professional roles, organizational settings, years of experience, and AI tools used. This diversity strengthened the credibility of the findings by capturing perspectives from software professionals working in different sectors and organizational environments within Pakistan's software ecosystem.

4. Findings and Discussion

4.1 Theme 1: Human Judgment Remains the Primary Security Gatekeeper in AI-Assisted Software Development

The findings indicate that despite the growing reliance on Large Language Models (LLMs) for software development, participants consistently regarded human judgment as the most critical component of software security assurance. AI-generated code was generally perceived as a useful starting point that accelerates development rather than a finished product suitable for direct deployment. Regardless of organizational role or years of professional experience, developers emphasized that the responsibility for ensuring software quality, security, and compliance ultimately remains with human developers.

Participants described AI-generated code as technically impressive but contextually limited. Although coding assistants were considered highly effective for generating boilerplate code, suggesting algorithms, automating repetitive programming tasks, and assisting with debugging, developers believed that LLMs lacked sufficient understanding of project-specific requirements, organizational security policies, regulatory obligations, and business logic. Consequently, AI-generated outputs required careful human interpretation before integration into production systems.

The findings further suggest that developers do not blindly trust AI-generated recommendations. Instead, participants adopted a cautious approach in which AI served as an intelligent assistant whose outputs required critical evaluation. Human expertise was considered essential for identifying subtle logical flaws, inefficient implementation strategies, insecure programming practices, and inconsistencies that automated systems might overlook. Participants viewed software security as a contextual decision-making process requiring professional judgment, domain knowledge, and organizational awareness that cannot currently be replicated by AI systems.

Another important finding concerns accountability. Participants consistently indicated that responsibility for software failures remains with development teams rather than AI technologies. Regardless of how code was generated, developers believed they retained full professional responsibility for validating functionality, identifying vulnerabilities, and ensuring compliance with organizational standards before deployment. This perspective reinforces the principle that AI augments rather than replaces developer expertise within contemporary software engineering ecosystems.

These findings align with Human-Centered AI Theory, which argues that intelligent systems should enhance human capabilities while preserving human oversight and accountability. The findings also support previous studies emphasizing that AI-assisted software engineering depends on collaborative intelligence, where developers remain responsible for interpreting, validating, and refining machine-generated

outputs rather than delegating critical security decisions entirely to AI (Abrahão et al., 2025; Hoffmann, 2025; Patil, 2026). Within the Pakistani software industry, where organizations demonstrate varying levels of AI governance maturity, human judgment becomes even more important because developers frequently compensate for the absence of standardized organizational validation procedures through professional experience and contextual reasoning.

4.2 Theme 2: Multi-Layered Validation Practices Mitigate Risks of LLM-Generated Code

A second major finding demonstrates that developers employ multiple complementary validation mechanisms before accepting AI-generated code. Rather than relying on a single verification technique, participants described security assurance as a structured process involving successive layers of technical and human evaluation. This multi-layered approach reduces the likelihood of introducing vulnerabilities into production software while simultaneously increasing developers' confidence in AI-assisted programming.

Manual code review emerged as the initial and most frequently applied validation mechanism. Participants reported carefully examining AI-generated code to evaluate logic, coding structure, readability, maintainability, and alignment with project requirements. Following manual inspection, developers frequently employed automated testing techniques, including unit testing, integration testing, regression testing, and functional testing to confirm that generated code performed as intended. Static code analysis tools, vulnerability scanners, dependency management platforms, and peer code reviews provided additional validation layers before deployment.

The findings further indicate that dependency verification has become increasingly important within AI-assisted software engineering. Participants explained that although AI-generated code often functions correctly, it occasionally recommends outdated libraries, deprecated programming interfaces, or third-party dependencies containing known security vulnerabilities. Consequently, validating external packages and confirming compatibility with organizational security policies have become routine development practices, particularly within FinTech, cloud computing, and enterprise software projects.

Another notable finding concerns iterative prompt refinement. Participants rarely accepted the first code generated by an LLM. Instead, developers continuously refined prompts by adding technical details, security requirements, and contextual information until generated outputs better reflected project expectations. Validation therefore extended beyond evaluating generated code to improving AI interactions themselves. This iterative collaboration illustrates that effective AI-assisted software development requires developers to actively guide AI systems rather than passively consume generated outputs.

The findings also reveal that validation intensity varies according to software criticality. Applications involving financial transactions, healthcare systems, cloud infrastructure, government services, or sensitive customer information received significantly more extensive security reviews than internal prototypes or experimental software. Developers consistently adapted their validation procedures according to project risk, indicating that security assurance remains context dependent rather than universally standardized.

These findings reinforce previous research demonstrating that technical automation alone cannot guarantee secure software engineering. Instead, secure AI-assisted development emerges through continuous interaction between human verification, automated security tools, structured testing, and organizational review processes. The findings therefore extend existing literature by illustrating how developers integrate multiple validation mechanisms into coherent security assurance workflows within real-world software engineering environments (Danyaro et al., 2025; Haseeb et al., 2026; Mittal, 2025; Zardari et al., 2026).

4.3 Theme 3: Security Awareness and Professional Experience Shape Developers' Trust in AI-Generated Code

The findings indicate that developers' trust in AI-generated code is strongly influenced by their security awareness, professional experience, and technical expertise. Trust was not viewed as an inherent characteristic of AI systems but rather as a dynamic outcome developed through repeated interaction, practical experience, and continuous evaluation of AI-generated outputs. Participants consistently demonstrated that greater professional experience resulted in more cautious and evidence-based trust toward AI-generated code.

Senior developers and DevSecOps professionals generally adopted calibrated trust, recognizing both the strengths and limitations of LLM-generated software artifacts. While acknowledging substantial productivity gains, experienced participants remained cautious when evaluating authentication mechanisms, encryption routines, cloud configurations, payment systems, and security-sensitive modules. Their extensive exposure to software vulnerabilities enabled them to identify subtle implementation flaws that less experienced developers might overlook.

Conversely, participants observed that junior developers initially exhibited greater confidence in AI-generated recommendations because they often evaluated software primarily on functional correctness rather than long-term maintainability, security, or scalability. However, practical software development experience gradually transformed this perspective. As developers encountered inaccurate AI recommendations, hallucinated functions, obsolete programming libraries, and insecure implementation strategies, they developed increasingly sophisticated validation behaviors characterized by greater skepticism and systematic verification.

The findings also suggest that organizational learning contributes significantly to trust development. Participants working in organizations with mature cybersecurity practices, formal code review procedures, and DevSecOps cultures demonstrated more structured validation behaviors than those employed in smaller organizations with limited governance mechanisms. Continuous exposure to security training, peer learning, vulnerability assessments, and collaborative code reviews strengthened developers' ability to critically evaluate AI-generated software while maintaining appropriate levels of trust.

Another important observation concerns the relationship between trust and productivity. Participants did not perceive trust as unconditional acceptance of AI recommendations. Instead, trust was conceptualized as confidence that AI could accelerate software development when combined with appropriate human oversight and rigorous validation procedures. This balanced perspective enabled developers to benefit from AI-assisted programming without compromising software security or quality. Rather than replacing human expertise, AI functioned as a productivity-enhancing partner whose outputs required continuous professional evaluation.

These findings support Human-Centered AI Theory by demonstrating that trust emerges from effective collaboration between human expertise and intelligent systems rather than technological capability alone. They further extend previous studies by showing that security awareness, organizational learning, and accumulated professional experience collectively shape developers' willingness to adopt AI-generated code within software engineering practice. Within Pakistan's evolving software ecosystem, where organizations differ considerably in AI governance maturity and cybersecurity capabilities, experienced developers play an especially important role in fostering responsible AI adoption through critical evaluation, security awareness, and informed decision-making (Alami et al., 2025; Kashif et al., 2026; Patil, 2026; Katsiuba, 2026).

4.4 Theme 4: Organizational Governance and Project Criticality Influence Secure AI Adoption in Pakistan

The findings demonstrate that organizational governance significantly influences how developers validate and integrate AI-generated code within software engineering projects. Although participants recognized the productivity benefits of LLM-based coding assistants, their validation practices differed considerably according to organizational maturity, industry sector, project sensitivity, and client requirements. These variations suggest that secure AI adoption is shaped not only by individual developer expertise but also by the governance environment within which software development occurs.

Participants employed in multinational software companies and large IT export organizations described relatively structured AI governance practices. These organizations incorporated formal code review procedures, secure DevSecOps pipelines, automated vulnerability assessments, coding standards, and compliance requirements that governed the use of AI-generated software artifacts. AI-assisted development was therefore integrated into existing software engineering processes rather than replacing established security controls.

In contrast, developers working in technology startups and small software houses reported greater reliance on individual expertise and informal validation practices. Although AI coding assistants were extensively used to accelerate software development, formal organizational policies regarding AI-generated code were often limited or still evolving. Consequently, security assurance depended largely on developers' professional judgment and peer collaboration rather than standardized governance frameworks. This finding suggests that differences in organizational maturity contribute directly to variation in secure AI adoption across Pakistan's software industry.

Project criticality further influenced validation behavior. Participants consistently reported that software developed for financial institutions, healthcare organizations, government agencies, and cloud infrastructure required significantly more rigorous validation than internal business applications or prototype systems. High-risk projects generally involved multiple layers of code review, security testing, regulatory compliance verification, and management approval before deployment. Conversely, lower-risk applications often adopted more streamlined validation procedures while maintaining basic quality assurance requirements.

Client expectations also emerged as an important organizational driver. Participants working with international clients explained that contractual obligations, cybersecurity standards, and compliance requirements frequently restricted the direct use of AI-generated code unless extensive validation procedures were completed. This demonstrates that organizational governance extends beyond internal policies to include external stakeholder expectations and regulatory obligations.

These findings indicate that organizational governance functions as a critical mechanism linking AI adoption with secure software engineering. Rather than limiting innovation, governance frameworks provide structured processes through which developers can safely leverage AI-generated code while maintaining software quality, cybersecurity, and regulatory compliance. The findings therefore support previous research emphasizing the importance of AI governance, responsible AI implementation, and DevSecOps integration within modern software engineering environments (Oakes, 2025; Kapadia & Islam, 2026; Lusi et al., 2026).

4.5 Theme 5: Continuous Human-AI Learning Strengthens Secure Software Engineering Practices

The findings reveal that AI-assisted software engineering promotes a continuous learning relationship between developers and intelligent programming assistants. Rather than viewing AI as a static automation tool, participants described an evolving collaborative process in which developers continuously refined their technical knowledge, prompt engineering skills, and security awareness through repeated interaction with LLMs. This reciprocal learning process strengthened both software quality and security assurance practices.

Participants explained that effective use of AI-generated code required significantly more than technical programming knowledge. Developers increasingly learned how to formulate precise prompts, provide contextual information, evaluate multiple AI-generated alternatives, and iteratively refine outputs until they satisfied organizational requirements. Prompt engineering therefore emerged as an important professional competency within AI-assisted software engineering.

The findings further indicate that interaction with AI systems encouraged developers to expand their understanding of programming languages, software architectures, testing methodologies, and cybersecurity principles. Participants frequently reported reviewing unfamiliar programming techniques suggested by AI before implementation, thereby transforming AI-generated recommendations into opportunities for professional learning. Rather than accepting generated code without question, developers investigated alternative implementation approaches, compared different solutions, and enhanced their understanding of secure software engineering practices.

Continuous learning also occurred through organizational collaboration. Participants described knowledge sharing among development teams regarding effective prompting strategies, AI limitations, secure coding practices, and lessons learned from previous AI-assisted projects. Peer discussions, code reviews, technical workshops, and collaborative problem-solving strengthened collective organizational capability while reducing the likelihood of security errors associated with AI-generated software artifacts.

Importantly, participants emphasized that AI itself evolves rapidly through continuous updates and expanding capabilities. Consequently, developers believed that maintaining effective AI-assisted software engineering requires ongoing professional development rather than one-time training. Continuous learning therefore encompasses both technological adaptation and the development of stronger security awareness capable of supporting responsible AI adoption within changing software engineering environments.

These findings suggest that learning represents a foundational component of human-centric security assurance. Secure AI-assisted software development depends not only on technical tools but also on developers' willingness to continuously improve their knowledge, adapt to emerging AI capabilities, and refine validation practices over time. This interpretation reinforces previous studies emphasizing human-AI collaboration as an evolving partnership that enhances both organizational capability and software engineering performance (Babulak et al., 2026; Hoffmann, 2025; Patil, 2026).

4.6 Theme 6: Human-Centric Security Assurance Emerges as a Socio-Technical Capability within Pakistan's Software Ecosystem

The final and integrative theme demonstrates that Human-Centric Security Assurance extends beyond individual validation activities to become a broader socio-technical capability emerging through interactions among developers, AI systems, organizational governance, security technologies, and continuous learning. Participants consistently described secure AI-assisted software engineering as a collaborative process in which technological capabilities alone are insufficient to ensure trustworthy software development.

The findings indicate that security assurance results from the integration of multiple interconnected elements. AI coding assistants provide productivity gains through automated code generation, debugging, and documentation. Human developers contribute contextual understanding, professional judgment, ethical reasoning, and security expertise. Organizational governance establishes coding standards, review procedures, and compliance mechanisms that regulate AI usage. Security technologies, including automated testing, vulnerability scanners, and static analysis tools, provide additional technical safeguards throughout the development lifecycle. Together, these components create an integrated

ecosystem supporting secure AI-assisted software engineering.

Within the Pakistani software industry, participants emphasized that differences in organizational maturity further strengthen the importance of human-centric security assurance. Organizations possessing comprehensive governance frameworks generally demonstrated more consistent validation behaviors, whereas organizations with limited AI policies relied more heavily on experienced developers to maintain software security. Consequently, human expertise functioned as the central coordinating mechanism connecting technological capability with organizational governance.

The findings further reveal that Human-Centric Security Assurance should not be interpreted as resistance to AI adoption. Instead, participants viewed human oversight as an enabling mechanism that allows organizations to confidently integrate AI into software development while minimizing associated risks. This perspective reframes AI-assisted software engineering as a balanced partnership where automation improves productivity without diminishing developer responsibility.

Overall, the findings suggest that Human-Centric Security Assurance represents an organizational capability rather than a single technical practice. It combines developer expertise, organizational governance, continuous learning, structured validation, and AI collaboration into an integrated framework capable of supporting secure software engineering within Pakistan's evolving digital economy.

Table 2. Thematic Analysis of Findings

Theme	Core Categories	Representative Codes	Interpretation
Theme 1: Human Judgment Remains the Primary Security Gatekeeper	Human oversight, professional responsibility, contextual reasoning, accountability	Manual review, critical evaluation, business logic validation, contextual understanding, developer accountability	Human expertise remains the final authority for determining the security and suitability of AI-generated code before deployment.
Theme 2: Multi-Layered Validation Practices Mitigate Risks of LLM-Generated Code	Validation mechanisms, security testing, code quality assurance, dependency verification	Manual code review, peer review, unit testing, integration testing, static analysis, vulnerability scanning, dependency checking, prompt refinement	Developers reduce AI-related risks by combining multiple technical and human verification mechanisms throughout the software development lifecycle.
Theme 3: Security Awareness and Professional Experience Shape Developers' Trust	Security knowledge, professional experience, calibrated trust, organizational learning	Cybersecurity awareness, DevSecOps expertise, critical thinking, trust calibration, continuous evaluation, risk perception	Trust in AI-generated code develops through accumulated experience, security expertise, and systematic validation rather than blind acceptance of AI recommendations.
Theme 4: Organizational Governance and Project Criticality Influence Secure AI Adoption	AI governance, organizational policies, project risk, compliance, DevSecOps	Coding standards, governance frameworks, compliance requirements, project criticality, client expectations, organizational maturity	Secure AI adoption is strongly influenced by governance structures, project sensitivity, and organizational security practices within Pakistani software organizations.

Theme	Core Categories	Representative Codes	Interpretation
Theme 5: Continuous Human-AI Learning Strengthens Secure Software Engineering Practices	Learning, prompt engineering, knowledge sharing, professional development	Prompt refinement, continuous learning, peer collaboration, technical upskilling, security awareness, organizational learning	Effective AI-assisted software engineering requires continuous learning that enhances both developers' technical competencies and security capabilities.
Theme 6: Human-Centric Security Assurance as a Socio-Technical Capability	Human-AI collaboration, governance integration, security ecosystem, organizational capability	Human oversight, AI collaboration, governance, security technologies, continuous improvement, secure software delivery	Human-Centric Security Assurance emerges as an integrated socio-technical capability combining AI technologies, human expertise, governance mechanisms, and continuous organizational learning to achieve secure software engineering.

4.7 Synthesis of Findings

The six themes collectively demonstrate that secure AI-assisted software engineering depends on the interaction of technological, organizational, and human factors rather than the capabilities of AI alone. Developers consistently positioned human expertise as the foundation of security assurance, while AI coding assistants were viewed as productivity-enhancing technologies requiring systematic verification. Security awareness, professional experience, organizational governance, structured validation procedures, and continuous learning collectively shaped developers' confidence in AI-generated software artifacts.

The findings support a conceptual framework in which AI-generated code artifacts initiate the software development process by providing automated programming support. These artifacts are subsequently evaluated through developers' security awareness, which influences their ability to identify potential risks. Human expertise then performs verification through critical review and contextual interpretation. The verified code undergoes multi-layer security validation, including manual inspection, automated testing, dependency verification, and peer review. Organizational governance provides policies, coding standards, and DevSecOps practices that regulate AI usage, while continuous human-AI learning strengthens developers' technical capabilities and adaptive security behaviors. Collectively, these interconnected processes establish Human-Centric Security Assurance, ultimately contributing to secure software delivery within Pakistan's AI-augmented software engineering ecosystem.

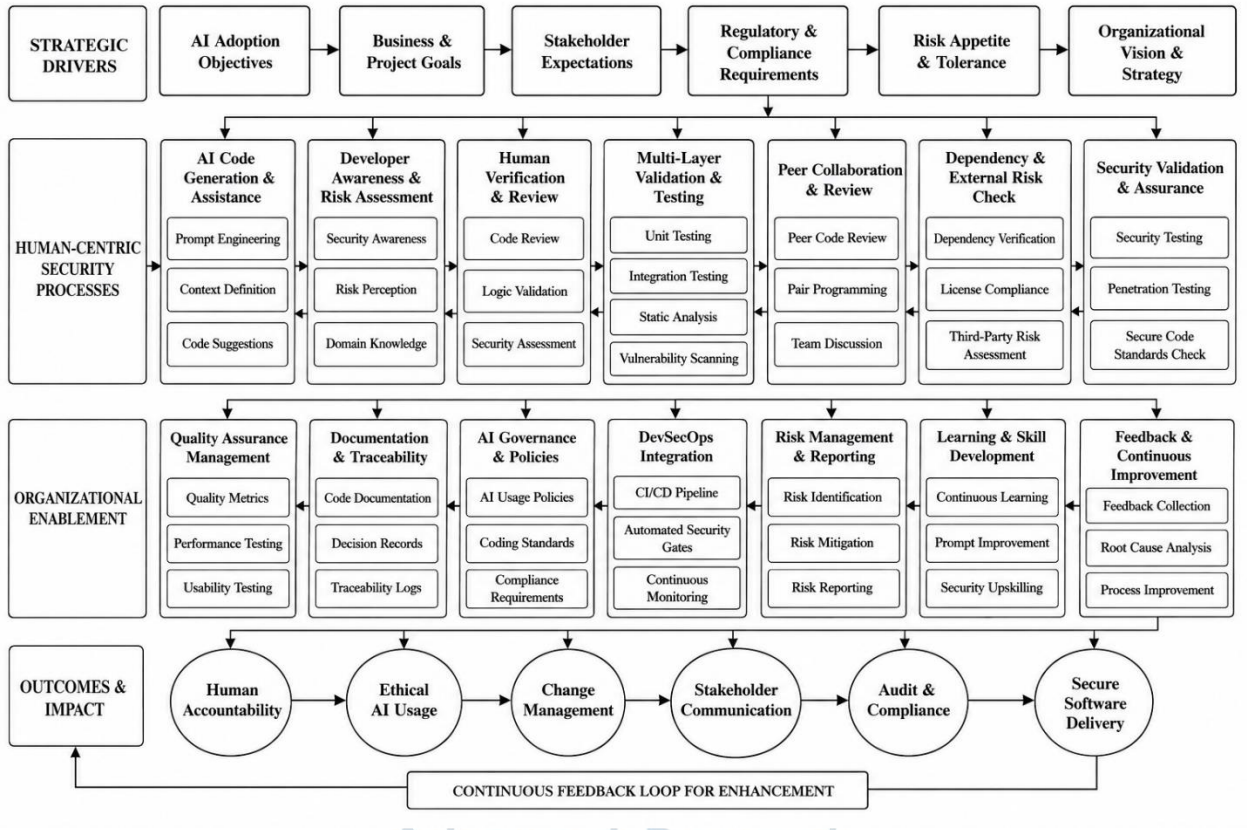

Figure 1. **Proposed Multi-Level Action Plan for Human-Centric Security Assurance in AI-Generated Code Development**

4.8 Discussion

The findings collectively demonstrate that Human-Centric Security Assurance is not merely a technical security practice but an integrated socio-technical capability emerging through the interaction of human expertise, AI technologies, organizational governance, and continuous learning. While Large Language Models (LLMs) substantially improve software development efficiency, participants consistently emphasized that secure software engineering remains fundamentally dependent on developers' contextual reasoning, professional accountability, and critical validation practices. This reinforces the argument that AI should augment rather than replace human decision-making, consistent with the principles of Human-Centered AI Theory (Patil, 2026; Hoffmann, 2025). The findings further extend previous research by illustrating that developers establish calibrated trust through continuous verification rather than unquestioningly accepting AI-generated recommendations, thereby supporting earlier studies on accountability, explainability, and responsible AI-assisted software development (Alami et al., 2025; Danyaro et al., 2025).

The study also demonstrates that effective security assurance requires an ecosystem perspective in which human judgment is complemented by structured validation mechanisms, DevSecOps practices, organizational governance, and continuous professional learning. This finding aligns with recent research advocating explainable AI, secure software engineering, and governance-driven AI implementation while highlighting the importance of embedding AI within established organizational security processes rather than treating it as an autonomous coding solution (Mittal, 2025; Oakes, 2025; Haseeb et al., 2026). Furthermore, the findings contribute context-

specific evidence from Pakistan by showing that differences in organizational maturity and governance frameworks significantly influence developers' validation behaviors and secure AI adoption. Consequently, Human-Centric Security Assurance should be viewed as an organizational capability that integrates technological innovation with human oversight, governance, and adaptive learning to achieve trustworthy AI-assisted software engineering. These findings therefore extend existing software engineering literature by providing an empirically grounded explanation of how secure AI adoption is achieved within an emerging digital economy (Abrahão et al., 2025; Babulak et al., 2026; Zardari et al., 2026).

5. Conclusion and Implications

5.1 Conclusion

This study explored how software developers in Pakistan validate Large Language Model (LLM)-generated code and implement human-centric security assurance practices within AI-augmented software engineering ecosystems. As AI-powered coding assistants become increasingly integrated into software development, understanding how developers balance productivity with security has become both a theoretical and practical priority. Using an interpretivist qualitative approach, the study examined the experiences of seventeen software professionals working across software houses, IT export companies, FinTech organizations, technology startups, AI development firms, multinational software companies, and public-sector digital organizations in Pakistan.

The findings demonstrate that AI-assisted software engineering is fundamentally a collaborative process in which human expertise remains central to secure software development. The first theme established that human judgment continues to serve as the primary security gatekeeper despite the increasing capabilities of LLMs. Developers consistently regarded AI-generated code as a useful starting point rather than a production-ready solution, emphasizing that professional judgment remains indispensable for evaluating business logic, software quality, security requirements, and organizational compliance.

The second theme revealed that developers employ multi-layered validation practices to reduce the risks associated with AI-generated code. Rather than depending on a single review mechanism, participants integrated manual code inspection, peer review, automated testing, dependency verification, vulnerability assessment, and security analysis into comprehensive validation workflows. These complementary practices enable organizations to benefit from AI-assisted productivity while maintaining software reliability and cybersecurity.

The third theme demonstrated that developers' trust in AI-generated code is shaped by security awareness and professional experience. Experienced software professionals exhibited calibrated trust, recognizing both the strengths and limitations of AI-generated artifacts. Security knowledge, practical software engineering experience, and continuous exposure to cybersecurity practices enabled developers to critically evaluate AI recommendations rather than accepting them unconditionally.

The fourth theme highlighted the importance of organizational governance and project criticality in determining secure AI adoption. Organizations with mature governance structures, formal DevSecOps practices, and established security policies demonstrated more systematic validation procedures than organizations relying primarily on individual developer expertise. Furthermore, project sensitivity, regulatory requirements, and client expectations significantly influenced the intensity of security validation throughout the software development lifecycle.

The fifth theme emphasized that continuous human-AI learning strengthens secure software engineering practices. Effective use of AI coding assistants requires developers to continuously improve prompt engineering skills, expand cybersecurity knowledge, refine validation techniques, and adapt to rapidly evolving AI technologies. Human learning therefore remains essential for sustaining secure AI-assisted software development over time.

Finally, the sixth theme integrated the preceding findings by demonstrating that Human-Centric Security Assurance represents a socio-technical capability emerging through interactions among developers, AI technologies, organizational governance, security tools, and continuous learning. Secure AI-assisted software engineering therefore depends not on AI alone but on the coordinated interaction of technological, organizational, and human capabilities.

Overall, this study concludes that although LLMs have become indispensable productivity tools within Pakistan's software industry, secure software engineering continues to depend primarily on developers' expertise, structured validation routines, organizational governance mechanisms, and continuous professional learning. AI enhances software development productivity, but human oversight remains indispensable for ensuring secure, reliable, and trustworthy software systems.

5.2 Theoretical Implications

This study contributes to the software engineering and Human-Centered AI literature in several important ways. First, it extends Human-Centered AI Theory by demonstrating that human oversight remains essential throughout AI-assisted software engineering rather than only during AI deployment or decision support. The findings illustrate that developers actively supervise, interpret, validate, and refine AI-generated software artifacts, reinforcing the theory's central premise that AI should augment rather than replace human expertise.

Second, the study introduces Human-Centric Security Assurance as a context-sensitive socio-technical capability within AI-augmented software engineering ecosystems. Rather than conceptualizing software security solely as a technical outcome, the proposed framework positions security assurance as an integrated capability emerging through interactions among developer expertise, AI technologies, organizational governance, validation practices, and continuous learning. This perspective expands existing software engineering research by emphasizing the inseparable relationship between technological and human factors.

Third, the findings explain how organizational governance, project characteristics, and developer expertise collectively influence secure AI adoption within an emerging economy. Existing research has largely concentrated on the technical performance of LLMs, whereas this study demonstrates that organizational context substantially shapes developers' security behaviors. By examining these interactions within Pakistan, the research extends Human-Centered AI Theory beyond predominantly Western software engineering environments.

Finally, the study broadens the literature by integrating AI governance with secure software engineering within an underrepresented national context. The proposed conceptual framework provides a foundation for future qualitative and quantitative investigations examining secure AI-assisted software development across emerging digital economies.

5.3 Practical Recommendations

The findings generate several practical recommendations for software organizations, policymakers, and educational institutions operating within Pakistan's evolving digital economy.

Software houses should establish comprehensive AI governance frameworks that define acceptable use policies, validation procedures, documentation standards, and accountability mechanisms for AI-generated code. Mandatory human review should remain an organizational requirement before AI-generated software artifacts are integrated into production environments.

FinTech organizations should strengthen AI-aware DevSecOps practices by incorporating automated security testing, vulnerability scanning, dependency management, regulatory compliance verification, and continuous monitoring throughout AI-assisted software development. Given the security sensitivity of financial systems, AI-generated code should undergo enhanced validation before deployment.

Technology startups should invest in structured AI governance despite limited organizational resources. Standardized coding guidelines, collaborative code reviews, secure prompt engineering practices, and peer-learning initiatives can substantially improve software quality while maintaining the productivity advantages of AI-assisted development.

Government digital agencies should develop national policies supporting responsible AI adoption within software engineering. Public-sector organizations should establish standardized AI governance frameworks, secure software development guidelines, cybersecurity compliance requirements, and procurement standards that regulate the use of AI-generated software within government information systems.

Higher education institutions should revise software engineering curricula to incorporate AI-assisted programming, secure prompt engineering, AI governance, cybersecurity, DevSecOps, and responsible AI development. Preparing future software engineers to critically evaluate AI-generated code will strengthen the national workforce as AI becomes increasingly embedded within software engineering practice.

At the national level, Pakistan should develop comprehensive AI coding and cybersecurity guidelines that provide standardized recommendations for validating AI-generated software. Such guidelines should integrate software engineering best practices, AI governance principles, cybersecurity standards, and ethical AI requirements while supporting innovation across the country's rapidly expanding software industry.

5.4 Limitations and Future Research

Despite its contributions, this study has several limitations. First, the qualitative research design prioritizes depth of understanding rather than statistical generalization. Consequently, the findings provide rich contextual insights into developers' experiences but should not be interpreted as representative of the entire Pakistani software industry.

Second, the study was conducted exclusively within Pakistan. Although this context provides valuable understanding of AI-assisted software engineering within an emerging digital economy, organizational practices may differ across countries with different regulatory environments, technological infrastructures, and governance systems.

Third, the research employed a cross-sectional design that captured developers' experiences at a single point in time. Given the rapid evolution of AI technologies, developers' validation behaviors and organizational governance practices are likely to continue evolving as LLM capabilities mature.

Fourth, purposive sampling was employed to recruit participants possessing relevant professional experience. While this approach ensured information-rich data, it may have excluded perspectives from developers with different organizational backgrounds or varying levels of AI adoption.

Future research should address these limitations by conducting comparative studies between Pakistan and other South Asian software ecosystems to examine how national context influences Human-Centric Security Assurance. Longitudinal investigations are also needed to explore how developers' validation practices evolve as AI-assisted software engineering becomes increasingly sophisticated. Researchers should further validate the proposed Human-Centric Security Assurance framework through mixed-method studies combining qualitative insights with quantitative hypothesis testing. Additional sector-specific investigations involving FinTech, healthcare, government, e-commerce, and critical infrastructure would provide a deeper understanding of industry-specific security requirements. Finally, future studies should examine the implications of autonomous coding agents and Agentic AI for software governance, developer accountability, and cybersecurity within Pakistan's rapidly evolving software engineering ecosystem.

References

Abouelenin, Y. A., & Jammal, M. (2026). Artificial intelligence-augmented digital twins: a comparative study of machine

learning and large language model integration in smart systems. In *Digital Twins* (pp. 53-70). Morgan Kaufmann.

Abrahão, S., Grundy, J., Pezzè, M., Storey, M. A., & Tamburri, D. A. (2025). Software Engineering by and for Humans in an AI Era. *ACM Transactions on Software Engineering and Methodology*, *34*(5), 1-46.

Akhtar, S., & Aftab, S. (2025). Towards AI-augmented software engineering: A theoretical framework. *ICCK Journal of Software Engineering*, *1*(2), 124-138.

Alami, A., Jensen, V., & Ernst, N. (2025). Accountability in code review: The role of intrinsic drivers and the impact of llms. *ACM Transactions on Software Engineering and Methodology*, *34*(8), 1-44.

Ali, G. A., & Rasheed, M. (2026). Investigating the drivers of escalating external debt in highly debt distressed nations using appropriate econometric techniques. *ACADEMIA International Journal for Social Sciences, 5*(3(s9), 463–485.

Ashraf, N., Arzu, F., Abrar, K., & Anwar, M. (2025). Narratives of SMEs on Access to Finance: Barriers and Opportunities in Pakistan's Banking Sector. *The Critical Review of Social Sciences Studies*, *3*(4), 262-277.

Babulak, E., Kumar, A., Guru, D., Kudithipudi, K., Maddini, S., Gonaygunta, H., & Nadella, S. (2026). Human-centric AI tools into agile methodologies for optimized software development. In *AI-driven approaches for fully automated smart engineering* (pp. 41-84). IGI Global Scientific Publishing.

Braun, V., & Clarke, V. (2019). Reflecting on reflexive thematic analysis. *Qualitative research in sport, exercise and health*, *11*(4), 589-597.

Braun, V., Clarke, V., & Hayfield, N. (2022). 'A starting point for your journey, not a map': Nikki Hayfield in conversation with Virginia Braun and Victoria Clarke about thematic analysis. *Qualitative research in psychology*, *19*(2), 424-445.

Danyaro, K. U., Nasser, M., Zakari, A., Abdullahi, S., Khanzada, A., Yakubu, M. M., & Shoaib, S. (2025). LLM-based code generation: a systematic literature review with technical and demographic insights. *IEEE Access*, *13*, 194915-194939.

Fein, B., Fraser, G., & Herbold, S. (2026, April). AI-Driven Software Development: A New Course Concept and Assessment Model for the Era of Large Language Models. In *Proceedings of the IEEE/ACM 48th International Conference on Software Engineering* (pp. 386-396).

Ghosh, S., Raj, R., & Goel, A. (2026). Adversarial AI-Driven Cyber Threat Intelligence: A Human-Centric Framework for Detecting and Mitigating AI-Augmented Social Engineering Attacks. In *Cyber Forensic Frameworks for User-Centric Human Threat Intelligence Analysis* (pp. 287-352). IGI Global Scientific Publishing.

Harrath, Y., Adohinzin, O., Kaabi, J., & Saathoff, M. (2025). Bridging domains: Advances in Explainable, Automated, and Privacy-Preserving AI for computer science and cybersecurity. *Computers*, *14*(9), 374.

Haseeb, M., Amjad, A. I., Amjad, M., & Sheng, V. S. (2026, June). From code to rubrics: A multimodal evaluation of LLM systems for automated feedback generation. In *Companion Proceedings of the ACM Web Conference 2026* (pp. 868-879).

He, W. (2026). Hybrid Translation Production: The Whole-Process AI-Empowerment. In *AI Through LLMs: Transforming Translation Practice and Teaching* (pp. 31-77). Singapore: Springer Nature Singapore.

Hoffmann, S. L. (2025). Human-Centered Cloud AI Collaboration Models for Sustainable and Secure Digital Enterprise Innovation. *International Journal of Engineering & Extended Technologies Research (IJEETR)*, *7*(5), 16062-16072.

Kapadia, H. P., & Islam, A. (2026). Strategic Foresight Leveraging AI and Predictive Analytics for Competitive Advantage. In *Harnessing AI and Predictive Analytics for Decision-Making and Competitive Advantage* (pp. 283-320). IGI Global Scientific Publishing.

Kashif, S. M., Liang, P., & Tahir, A. (2026). On developers' self-declaration of ai-generated code: An analysis of practices. *ACM Transactions on Software Engineering and Methodology*, *35*(7), 1-37.

Katsiuba, D. (2026). *Collaboration with agentic information systems* (Doctoral dissertation, University of Zurich).

Khan, H. A., Mahmood, A., Bibi, K., Abrar, K., & Arif, S. N. (2026). A Vision for Digital Education: A Comparative Exploration of Government Strategies for ICT and AI Integration in Pakistan and Malaysia. *Research Journal for Social Affairs*, *4*(3 (s5)), 205-214.

Khan, M., Akbar, M. A., Kasurinen, J., & Martín-Barroso, E. (2026). Prediction Model of Motivators and Demotivators of Integrating Large Language Models in Software Engineering Education: An Empirical Study. *arXiv preprint arXiv:2605.09393*.

Khan, M., Arzu, F., Amjad, S. I., Mahmood, A., & Abrar, K. (2026). Exploring how fintech entrepreneurs build consumer trust in emerging markets amid low financial literacy and high technological skepticism challenges adoption rates. *Bulletin of Management Review*, *3*(2), 524–552.

Khan, S. A., Dasgupta, L., Ghosh, S., & Ghosh, K. (Eds.). (2026). *Artificial Intelligence and Legal Evidence: The Indian Policy and Perspective*. CRC Press.

Lusi, J. A., Nikiforova, A., Lorenz, B., & Pappel, I. (2026). An LLM-augmented framework for interdisciplinary curriculum design and innovation: case of digital public infrastructure education. *Quality & Quantity*, 1-35.

Madupati, B., Vududala, S. K., & Temnikov, D. (2025). *From Code to Consciousness: Leveraging AI in Software Development*. Libertatem Media Private Limited.

Mittal, A. (2025, October). Explainable AI-Augmented DevSecOps for Secure and Reproducible Cloud-Native Research Software. In *International Conference on Computer Applications in Industry and Engineering* (pp. 69-86). Cham: Springer Nature Switzerland.

Mondal, S. (2026). *AI-Augmented Quality Engineering for Developer Knowledge Artifacts in Human-AI Ecosystems* (Doctoral dissertation).

Oakes, S. (2025, August). Holistic Risk Management for Next-Gen Technologies Using AI and Governance Standards. In *International Conference on Software Engineering of Emerging Technology* (pp. 670-679). Cham: Springer Nature Switzerland.

Patil, M. (2026). Enterprise Strategy for Human-Centered AI. In *Handbook of Human-Centered Artificial Intelligence* (pp. 2527-2588). Singapore: Springer Nature Singapore.

Pracucci, A., & Giovanardi, M. (2026). Ecodesign Prioritization for BIPV Manufacturers Under ESPR Compliance: An LLM-Assisted Multi-Criteria Framework with Use Cases Application. *Sustainability*, *18*(10), 4695.

Rasheed, M., & Ali, G. A. (2026). Digital technology and the economic growth: An empirical evidence from South Asia. *International Journal of Social Sciences Bulletin*, *4*(3), 2129–2151.

Rehmat, M. A., Hassan, H., Khan, H. A., & Abrar, K. (2025). Artificial intelligence in the classroom: teachers' lived experiences and ethical concerns in educational integration. *ASSAJ*, *4*(02), 629-645.

Shahzad, K., & Iqbal, S. (2025, May). Comparative analysis of ChatGPT, DeepSeek, and Gemini for automated code generation. In *2025 18th International Conference on Engineering of Modern Electric Systems (EMES)* (pp. 1-4). IEEE.

Sharma, S., & Sharma, S. (2026). Robotic Systems and LLM Algorithms for Transforming Manufacturing Into Smart Autonomous Ecosystem. In *Human-Robot Collaboration and Advanced Robotics in Modern Industry* (pp. 231-262). IGI Global Scientific Publishing.

Terry, G., Hayfield, N., Clarke, V., & Braun, V. (2017). Thematic analysis. *The SAGE handbook of qualitative research in psychology*, *2*(17-37), 25.

Torres, D. J. G. (2025). Quality-Assured Predictive Threat Intelligence in Cloud–Lakehouse Systems: Bayesian Statistical Learning and AI-Augmented Risk Monitoring for Data-Limited Environments. *International Journal of Advanced Research in Computer Science & Technology (IJARCST)*, *8*(6), 13200-13207.

Yaseen, M., Syed, M. I., Qamar, B., Mahmood, A., & Dalene, Y. (2026). Cultural and Regional Influences on AI Adoption in Hybrid Work Models: Perspectives from HR Managers in Pakistan and the UK. *Social Science Review Archives*, *4*(1), 5503-5520.

Zaidi, T., Aftab, M. U., & Usama, M. (2026, May). Leveraging Large Language Models for Scalable Smart Contracts in Blockchain IoT Systems. In *2026 5th International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)* (pp. 1-6). IEEE.

Zardari, S., Osama, M., Ammar, S. M., Reshamwala, R. I., & Siddiqui, A. (2026). A comprehensive analysis of security risks and productivity impacts of AI-generated code. *AI and Ethics*, *6*(4), 357.