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

LARGE LANGUAGE MODELS IN AI-AUGMENTED DEVSECOPS PIPELINES: EXPLORING SECURE SOFTWARE ENGINEERING PRACTICES AND ARCHITECTURAL CHALLENGES

Abdul Mannan[*1], Muhammad Ali[2], Yasir Siddiq[3], Sadaf Anwar[4], Muhammad Huzaifa Noor[5]

*[*1]Research Scholar, Department of Engineering for Innovation, University of Salento, Lecce, Italy*
[2]Assistant Professor, Department of Computing and Innovation, Riphah International University, Sahiwal Campus
[3]Independent Researcher, Barcelona, Spain
[4]Researcher, Department of Service and Information System Engineering, UPC, Barcelona, Spain
[5]Research Scholar, Department of Software Engineering, NUML, Islamabad, Pakistan

[*1]abdul.mannan@studenti.unisalento.it, [2]m.ali@riphahsahiwal.edu.pk, [3]yasir.siddiq@gmail.com, [4]sadaf.anwar@upc.edu, [5]huzaifanoor.web@gmail.com

DOI: http://doi.org/10.5281/zenodo.21771477

Keywords
Large Language Models, AI-Augmented DevSecOps, Secure Software Engineering, Thematic Analysis, Software Architecture

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

Corresponding Author: *
Abdul Mannan

Abstract

The increasing adoption of Large Language Models (LLMs) has transformed AI-augmented DevSecOps by enhancing secure software engineering practices throughout the Software Development Life Cycle. Despite their growing use, limited research has explored how practitioners experience the integration of LLMs within secure software development environments and the architectural challenges that accompany their implementation. This study employed a qualitative research design using semi-structured interviews with 20 software engineers, DevSecOps practitioners, security architects, cloud specialists, AI engineers, and technology consultants. The collected data was analyzed using Braun and Clarke's Reflexive Thematic Analysis. Five major themes emerged: LLMs as intelligent enablers of secure software engineering, the continuing importance of human oversight, architectural integration challenges, governance and explainability for responsible AI adoption, and the future of collaborative AI-enabled DevSecOps. The findings reveal that LLMs significantly improve secure coding, vulnerability analysis, documentation, and operational efficiency while introducing concerns related to model reliability, data privacy, governance, software supply chain security, and regulatory compliance. The study proposes a practitioner-informed action framework that emphasizes human-centered governance, secure architectural integration, and continuous validation, thereby contributing practical and theoretical insights for the responsible adoption of LLMs within enterprise DevSecOps ecosystems.

Introduction

Background of the Research

The rapid advancement of Artificial Intelligence (AI) has transformed the landscape of software engineering by introducing intelligent systems capable of automating complex development and security tasks. Among these innovations, Large Language Models (LLMs) have emerged as one of the most influential technologies, fundamentally changing how software is designed, developed,

tested, and maintained. Unlike conventional automation tools that rely on predefined rules and scripts, LLMs employ deep learning and natural language understanding to generate programming code, explain vulnerabilities, review software architecture, create documentation, and assist developers in solving complex engineering problems. These capabilities have accelerated the integration of LLMs into modern software development environments, particularly within AI-augmented DevSecOps pipelines that emphasize continuous security throughout the Software Development Life Cycle (SDLC) (Robert et al., 2024; Joshi, 2025).

DevSecOps represents the evolution of traditional DevOps by embedding security practices into every stage of software development rather than treating security as a separate phase. Modern organizations increasingly adopt DevSecOps to enable continuous integration, continuous security testing, continuous monitoring, and continuous deployment while maintaining rapid software delivery. The emergence of cloud computing, containerization, microservices, and distributed software architectures has significantly increased the complexity of managing secure software pipelines. Consequently, organizations are adopting AI-driven solutions capable of automating vulnerability detection, security testing, compliance verification, risk assessment, and incident response across Continuous Integration and Continuous Delivery (CI/CD) environments (Wei, 2019; Isabella et al., 2023; Allam, 2025).

Recent developments demonstrate that LLMs have become intelligent assistants within DevSecOps environments by supporting secure code generation, automated code reviews, vulnerability remediation, threat modeling, security documentation, policy generation, and intelligent debugging. These capabilities enable software engineers to reduce repetitive tasks while improving development efficiency and strengthening software security. AI-powered DevSecOps also supports scalable cloud-native architectures by facilitating continuous security integration across distributed software systems (Beeram, 2026; Mahimalur et al., 2026; Mittal, 2026).

Despite these benefits, the integration of LLMs into secure software engineering introduces numerous technical, architectural, and governance challenges. AI-generated code may contain vulnerabilities, hallucinated recommendations, insecure configurations, or inaccurate security advice that can compromise enterprise software systems if not properly validated. Similarly, integrating LLMs within existing CI/CD infrastructures introduces concerns related to software supply chain security, data privacy, API orchestration, model governance, scalability, latency, and regulatory compliance (Ahi & Valizadeh, 2025; Bajlur et al., 2026; Rusum, 2023).

Furthermore, successful adoption of AI-assisted DevSecOps depends on human trust, explainability, and organizational readiness. Security engineers continue to rely on human expertise to validate AI-generated outputs, making explainable AI and human oversight essential components of responsible software engineering practices (Rehmat et al., 2025; Mittal, 2026).

Although considerable research has examined AI-assisted software engineering from technical and quantitative perspectives, relatively little is known about how software engineers experience the integration of LLMs within real-world DevSecOps environments. Existing studies largely emphasize algorithms, architectures, and system performance while overlooking practitioner perspectives on trust, governance, architectural decision-making, and secure software engineering practices (Kumar, 2025; Kumari et al., 2026; Lin et al., 2025). Therefore, this study adopts a qualitative approach to explore practitioners' lived experiences of integrating Large Language Models into AI-augmented DevSecOps pipelines.

The rapid digital transformation of organizations has significantly increased the demand for secure, scalable, and intelligent software engineering practices. Traditional software development models often treated security as a final verification activity, resulting in delayed vulnerability detection and increased remediation costs. The

emergence of DevSecOps addressed these limitations by integrating security activities throughout the software development lifecycle. Simultaneously, advances in Artificial Intelligence have introduced intelligent automation capabilities that enhance software engineering through code generation, vulnerability analysis, automated testing, architectural recommendations, and continuous security monitoring (Wei, 2019; Allam, 2025).

Large Language Models represent the latest evolution of AI-assisted software engineering. Their ability to understand source code, programming languages, security documentation, and developer intent enables them to perform sophisticated engineering tasks that previously required significant human expertise. Modern DevSecOps platforms increasingly employ LLMs to generate secure code, explain vulnerabilities, automate compliance documentation, prioritize security risks, and recommend mitigation strategies (Beeram, 2026; Mahimalur et al., 2026; Coston et al., 2025).

However, integrating LLMs into secure software pipelines introduces challenges beyond technical implementation. Organizations must address concerns related to explainability, governance, model reliability, software supply chain security, cloud deployment, and ethical AI adoption. These emerging issues necessitate an in-depth qualitative understanding of practitioner experiences to complement existing technical research.

Problem Statement

Despite the increasing adoption of Large Language Models in AI-augmented DevSecOps pipelines, there remains limited understanding of how software engineers and security practitioners experience their integration within real-world software development environments. Existing studies predominantly focus on algorithmic performance, automation capabilities, and architectural frameworks while paying comparatively little attention to practitioner perspectives regarding trust, explainability, governance, architectural decision-making, and organizational challenges (Ahi & Valizadeh, 2025; Bajlur et al., 2026; Lin et al., 2025).

Furthermore, concerns surrounding AI hallucinations, insecure code generation, software supply chain vulnerabilities, privacy risks, and human oversight continue to hinder enterprise adoption of LLM-assisted DevSecOps. Consequently, qualitative evidence is required to understand how practitioners navigate these challenges while maintaining secure software engineering practices.

Research Questions

The study seeks to answer the following research questions:

RQ1: How do software engineers and DevSecOps practitioners integrate Large Language Models into AI-augmented DevSecOps pipelines?

RQ2: What secure software engineering practices are enhanced through the adoption of LLM-assisted DevSecOps?

RQ3: What architectural challenges emerge during the integration of LLMs within secure CI/CD environments?

RQ4: How do practitioners perceive trust, explainability, and human oversight when using LLM-generated security recommendations?

RQ5: What governance mechanisms are necessary for the responsible adoption of LLMs in secure software engineering?

Research Objectives

The objectives of this study are:

1. To explore how software engineers integrate Large Language Models into AI-augmented DevSecOps pipelines.
2. To identify secure software engineering practices enabled by LLM-assisted automation.
3. To examine architectural challenges associated with integrating LLMs into DevSecOps ecosystems.
4. To investigate practitioner perceptions regarding trust, explainability, and validation of AI-generated security recommendations.
5. To develop practical recommendations for responsible and secure implementation of

LLMs within enterprise DevSecOps environments.

Significance of the Research

This study contributes to both software engineering theory and DevSecOps practice. From a theoretical perspective, it extends existing knowledge by integrating perspectives from AI-assisted software engineering, secure software development, explainable AI, and socio-technical systems. It also enriches the emerging literature on practitioner experiences with LLM-enabled software development, complementing existing technical research that primarily focuses on system performance and algorithmic capabilities (Robert et al., 2024; Kumar, 2025).

Practically, the findings will assist software architects, DevSecOps engineers, cloud security specialists, and technology managers in understanding how LLMs can be responsibly integrated into secure development pipelines. The study also provides insights for organizations developing AI governance frameworks, secure software engineering policies, and cloud-native DevSecOps strategies. Furthermore, by emphasizing trust, explainability, and human oversight, the research supports responsible AI adoption within software engineering and contributes to broader digital transformation initiatives (Rehmat et al., 2025; Khan et al., 2026; Yaseen et al., 2026). The findings may also inform organizations operating in emerging digital economies where AI adoption, technological capability, and institutional readiness continue to evolve (Rasheed & Ali, 2026).

Literature Review

The integration of Artificial Intelligence (AI) into software engineering has transformed the development of secure, intelligent, and scalable software systems. Among recent AI innovations, Large Language Models (LLMs) have emerged as powerful tools capable of generating source code, reviewing software quality, identifying vulnerabilities, automating testing, producing technical documentation, and supporting decision-making throughout the Software Development Life Cycle (SDLC). Simultaneously, DevSecOps has evolved from conventional DevOps by embedding security into every stage of software development, thereby enabling continuous integration, continuous security testing, and continuous deployment. The convergence of LLMs and AI-augmented DevSecOps has introduced new opportunities for improving software quality and security while simultaneously presenting architectural, governance, and operational challenges. This chapter reviews the existing literature by examining secure software engineering, AI-augmented DevSecOps, LLM-enabled software development, architectural challenges, and explainable AI before identifying theoretical foundations and research gaps that justify the present study.

Theoretical Framework

This study is primarily underpinned by the Socio-Technical Systems (STS) Theory and the Technology–Organization–Environment (TOE) Framework. These complementary theories provide a comprehensive lens for understanding how organizations integrate Large Language Models into AI-augmented DevSecOps pipelines while balancing technological innovation, human expertise, organizational processes, and environmental influences.

Socio-Technical Systems Theory argues that organizational performance depends upon the joint optimization of technical systems and social systems rather than focusing exclusively on technology. Within DevSecOps environments, technical components include CI/CD pipelines, cloud infrastructures, security automation platforms, vulnerability scanners, and LLM-powered coding assistants. The social subsystem consists of software engineers, DevSecOps practitioners, security architects, organizational governance, collaborative workflows, and managerial decision-making. The successful implementation of LLMs therefore depends not only on technical accuracy but also on human trust, explainability, collaboration, governance, and continuous learning. This theoretical

perspective is particularly relevant because software engineers remain responsible for validating AI-generated recommendations, interpreting security alerts, and making final architectural decisions despite increasing automation (Rehmat et al., 2025; Robert et al., 2024).

The Technology–Organization–Environment (TOE) Framework further explains organizational adoption of emerging technologies by considering technological readiness, organizational capabilities, and external environmental pressures. Technological readiness includes LLM capabilities, AI infrastructure, cloud-native architectures, and security automation tools. Organizational factors encompass leadership support, AI governance, employee expertise, security culture, and resource availability. Environmental factors include regulatory compliance, cybersecurity threats, competitive pressures, and evolving software engineering standards. Together, STS Theory and the TOE Framework provide an appropriate theoretical foundation for examining both the technological and organizational dimensions of LLM-enabled DevSecOps adoption.

Large Language Models in Software Engineering

Software engineering has undergone substantial transformation following the emergence of generative AI and LLMs. Earlier software automation tools relied primarily on predefined rules, syntax checking, and deterministic programming logic. Modern LLMs, however, utilize transformer-based architectures capable of understanding programming languages, contextual information, software documentation, and developer intent. Consequently, LLMs have become intelligent assistants capable of generating source code, debugging software, producing technical documentation, recommending software architectures, generating unit tests, and supporting collaborative software development (Robert et al., 2024).

Joshi (2025) argues that generative AI has become an integral component of modern DevOps by facilitating CI/CD automation, MLOps integration, intelligent code generation, and agentic automation. Similarly, Kumar (2025) emphasizes that next-generation software engineering increasingly depends on AI-assisted development environments that improve developer productivity while reducing repetitive programming tasks. Kumari et al. (2026) further highlight that AI technologies accelerate software engineering by automating testing, software maintenance, project management, defect prediction, and quality assurance. Collectively, these studies demonstrate that LLMs have evolved from experimental technologies into practical engineering tools that support multiple stages of the SDLC.

Despite these advantages, researchers caution that LLM-generated code should not be accepted without verification. Hallucinated outputs, insecure coding practices, outdated libraries, and inaccurate recommendations continue to present challenges that require human validation and secure engineering practices (Ahi & Valizadeh, 2025).

AI-Augmented DevSecOps Pipelines

DevSecOps extends DevOps by embedding security throughout software development rather than introducing security only during final testing stages. This paradigm enables continuous vulnerability assessment, automated security testing, compliance verification, secure deployment, and continuous monitoring. AI augmentation has significantly enhanced these capabilities by automating repetitive security activities and enabling intelligent decision support.

Wei (2019) introduced one of the earliest conceptual discussions regarding AI-augmented DevSecOps, arguing that intelligent automation enables continuous security integration throughout software development. Isabella et al. (2023) further demonstrated that AI-assisted DevSecOps automates vulnerability scanning, code review, penetration testing, and threat identification within CI/CD environments. Allam (2025) similarly reported that AI-driven CI/CD systems improve deployment efficiency

while reducing security risks through automated pipeline intelligence.

Beeram (2026) specifically examined AI-augmented DevSecOps within Azure Pipelines and concluded that integrating AI into cloud deployment environments strengthens security monitoring, compliance enforcement, and deployment automation. Likewise, Mahimalur et al. (2026) proposed AI-integrated DevSecOps architectures capable of supporting scalable cloud-native software systems through intelligent orchestration and automated security controls. These studies collectively indicate that AI has become an indispensable component of secure software delivery by enhancing operational efficiency while reducing manual security workloads.

Secure Software Engineering Practices Enabled by LLMs

The integration of LLMs into software engineering has significantly transformed secure development practices. Rather than functioning solely as coding assistants, modern LLMs actively support secure code generation, vulnerability remediation, threat modeling, security documentation, compliance verification, and software quality assurance.

Coston et al. (2025) introduced the AI-integrated AZTRM-D framework, demonstrating how DevSecOps, Zero Trust Architecture, and AI-driven risk management collectively improve software security throughout development pipelines. Similarly, Lin et al. (2025) proposed an AI-augmented risk-based framework for assessing software security maturity by combining intelligent risk prioritization with continuous security evaluation.

Singh (2026) reviewed machine learning approaches for software testing and concluded that AI-assisted testing substantially improves vulnerability detection, software reliability, and testing efficiency. Mittal (2026) further argued that AI-enabled DevSecOps enhances secure cloud deployments by automating threat detection, policy enforcement, and continuous compliance monitoring. These findings suggest that LLMs increasingly function as collaborative security assistants capable of supporting software engineers throughout the entire SDLC while reducing development complexity and strengthening software resilience.

Architectural Challenges of Integrating LLMs into DevSecOps

Although LLMs offer considerable advantages, their integration within enterprise DevSecOps architectures introduces several technical and organizational challenges. Cloud-native systems frequently consist of distributed microservices, containerized applications, multiple APIs, hybrid cloud environments, and complex dependency structures. Integrating LLM services into such environments requires careful orchestration, scalability management, latency optimization, and governance.

Rusum (2023) emphasized that software supply chain security has become increasingly important due to growing dependencies on external packages, APIs, and third-party AI services. The introduction of LLMs further expands the software attack surface by introducing risks associated with prompt injection, model poisoning, malicious dependencies, and unauthorized access to proprietary code repositories.

Vas (2025) similarly argued that AI-augmented DevSecOps environments must address architectural issues involving scalability, deployment latency, cloud integration, and security interoperability. Mahimalur et al. (2026) further noted that enterprise cloud architectures require adaptive orchestration mechanisms capable of integrating AI services without compromising software reliability or operational efficiency.

Additionally, Ahi and Valizadeh (2025) identified dual-use risks associated with LLMs, including AI-generated malware, adversarial prompt engineering, privacy violations, explainability limitations, and cybersecurity misuse. Bajlur et al. (2026) likewise concluded that while LLMs create substantial opportunities for software security, they also introduce emerging threats requiring robust governance and continuous monitoring.

Explainability, Human Oversight, and Trust in AI-Augmented DevSecOps

Successful integration of LLMs within secure software engineering depends not only on technical performance but also on practitioner trust and explainability. AI-generated recommendations significantly influence software architecture, vulnerability remediation, and deployment decisions; therefore, software engineers must understand the reasoning behind AI-generated outputs before implementation.

Rehmat et al. (2025) demonstrated that explainable AI substantially improves trust among non-technical users by increasing transparency and confidence in AI-assisted decision-making. Although their research focused on emerging markets, its implications extend to software engineering because explainability similarly affects software engineers' willingness to adopt AI-generated recommendations.

Mittal (2026) further proposed Explainable AI-Augmented DevSecOps as a mechanism for improving transparency, reproducibility, and accountability within secure cloud-native environments. Explainability enables developers to verify security recommendations, interpret vulnerability assessments, and justify deployment decisions, thereby reducing overreliance on opaque AI systems.

Moreover, organizational trust extends beyond individual developers. Responsible AI adoption requires governance policies, auditing mechanisms, human validation, ethical guidelines, and continuous monitoring to ensure that LLM-generated outputs remain secure, reliable, and aligned with organizational objectives (Lin et al., 2025).

Organizational Adoption and Digital Transformation

The integration of AI into software engineering reflects broader organizational digital transformation initiatives. Khan et al. (2026) demonstrated that national strategies supporting AI and ICT integration significantly strengthen digital capability and technological innovation. Similarly, Yaseen et al. (2026) found that cultural and regional contexts influence organizational AI adoption, suggesting that technological implementation depends upon organizational readiness in addition to technical capability.

Research investigating trust within emerging technologies further reinforces these observations. Khan et al. (2026) reported that consumer trust remains a critical determinant of fintech adoption under conditions of technological uncertainty. Although situated within financial technology, these findings similarly highlight the importance of trust when organizations adopt LLM-enabled software engineering practices. Likewise, Rasheed and Ali (2026) argued that digital technologies contribute significantly to organizational competitiveness and economic development, reinforcing the strategic importance of AI-enabled software engineering within contemporary digital economies.

Gaps in Existing Research

The existing literature demonstrates substantial progress in AI-assisted software engineering, DevSecOps automation, cloud security, and secure software development. However, several important research gaps remain.

First, most existing studies adopt technical, quantitative, experimental, or framework-development approaches that emphasize algorithmic performance, software architectures, and automation efficiency while largely overlooking practitioner experiences (Allam, 2025; Beeram, 2026; Mahimalur et al., 2026). Consequently, limited qualitative evidence exists regarding how software engineers actually integrate LLMs into everyday DevSecOps workflows.

Second, current literature extensively discusses AI capabilities but provides comparatively little understanding of human trust, explainability, governance, and collaborative decision-making within AI-augmented software engineering environments (Rehmat et al., 2025; Mittal, 2026).

Third, although researchers have examined software supply chain security, AI-generated vulnerabilities, and cloud-native architectures, limited research explores how these architectural

challenges influence practitioners' implementation strategies within enterprise DevSecOps ecosystems (Rusum, 2023; Bajlur et al., 2026).

Fourth, existing studies frequently investigate isolated technical components such as vulnerability detection, software testing, secure deployment, or AI-generated code rather than examining the complete integration of LLMs across secure software engineering pipelines (Coston et al., 2025; Lin et al., 2025; Singh, 2026).

Finally, there is a notable absence of qualitative research exploring the experiences of DevSecOps engineers, software architects, AI specialists, and cybersecurity professionals regarding responsible LLM adoption. Addressing these gaps will provide richer insights into organizational practices, architectural decision-making, trust formation, governance mechanisms, and secure software engineering strategies. Accordingly, the present study adopts a qualitative exploratory approach to investigate practitioner experiences and develop a comprehensive understanding of how Large Language Models are transforming AI-augmented DevSecOps pipelines.

Methodology

Research Design

This study adopted a qualitative exploratory research design to investigate how software engineers, DevSecOps practitioners, AI engineers, and security architects experience the integration of Large Language Models (LLMs) within AI-augmented DevSecOps pipelines. A qualitative approach was considered appropriate because the study sought to understand participants' perceptions, experiences, interpretations, and decision-making processes rather than measuring predefined variables quantitatively (Abrar & Amjad, 2026; Ashraf et al., 2025; Imtiaz et al., 2026). Given the rapidly evolving nature of LLM adoption in secure software engineering, qualitative inquiry provides the flexibility to explore emerging practices, architectural challenges, governance concerns, and human-centered experiences in depth. Semi-structured interviews were employed to encourage participants to share detailed insights while allowing the researcher to probe issues related to secure software development, CI/CD automation, explainability, trust, and AI governance. The study followed the principles of Reflexive Thematic Analysis proposed by Braun and Clarke, which emphasizes researcher reflexivity and iterative theme development throughout the analytical process (Braun et al., 2022; Byrne, 2022; Hole, 2024).

Sampling Strategy

Purposive sampling was employed to recruit information-rich participants possessing direct experience with DevSecOps, software engineering, cloud security, AI-assisted software development, or Large Language Model implementation. To obtain diverse perspectives, participants were selected from software houses, multinational technology companies, cloud service providers, fintech organizations, cybersecurity firms, and AI startups. Snowball sampling was subsequently used to identify additional professionals through referrals from initial participants. A total of twenty participants were interviewed, representing a broad range of technical expertise and organizational backgrounds. This sample size was considered adequate because thematic saturation was achieved, with no substantially new insights emerging during the final interviews.

Data Collection

Primary data were collected through semi-structured interviews conducted virtually using Microsoft Teams and Google Meet. Each interview lasted between 45 and 60 minutes and was audio-recorded with participants' informed consent. The interview protocol included open-ended questions addressing the adoption of LLMs in DevSecOps pipelines, secure coding practices, vulnerability management, AI explainability, software architecture, governance mechanisms, software supply chain security, and organizational readiness. All interviews were transcribed verbatim before analysis.

Data Analysis

The collected data were analyzed using Braun and Clarke's six-phase Reflexive Thematic Analysis. The analytical process included familiarization with interview transcripts, generation of initial codes, searching for candidate themes, reviewing themes, defining and naming themes, and producing the final report. Coding was conducted iteratively to capture recurring patterns related to AI-enabled software engineering practices, architectural integration, trust, governance, and security challenges. Reflexive memo writing and continuous comparison across transcripts enhanced analytical rigor and credibility throughout the study.

Ethical Considerations

Ethical approval was obtained before data collection commenced. All participants voluntarily participated after receiving detailed information regarding the study's objectives, confidentiality procedures, and their right to withdraw at any stage without consequence. Written informed consent was obtained prior to each interview. Participant identities were anonymized using pseudonyms (P1–P20), and no identifiable organizational information was disclosed. Interview recordings, transcripts, and coded data were securely stored in encrypted digital storage accessible only to the researcher

Table 1: Participant Profiles

Participant	Current Position	Organization Type	Country	Primary Technical Expertise	Experience with LLMs	DevSecOps Experience
P1	Senior DevSecOps Engineer	Software House	Pakistan	CI/CD Security Automation	2 years	5 years
P2	Principal Software Architect	Multinational IT Company	Pakistan	Cloud Architecture	3 years	8 years
P3	Application Security Engineer	FinTech	Pakistan	Secure Coding	2 years	6 years
P4	AI Engineer	AI Startup	Pakistan	Generative AI & LLMs	3 years	2 years
P5	Senior DevOps Engineer	SaaS Company	Pakistan	Kubernetes & CI/CD	2 years	7 years
P6	Backend Software Engineer	Software House	Pakistan	Secure API Development	2 years	3 years
P7	Cloud Security Engineer	Cloud Service Provider	UAE	Zero Trust Security	2 years	7 years
P8	Engineering Manager	Enterprise Organization	Pakistan	Software Delivery Management	2 years	8 years
P9	Security Architect	Banking Sector	Pakistan	Threat Modeling	2 years	10 years
P10	Platform Engineer	SaaS Company	Pakistan	Kubernetes Security	2 years	5 years
P11	AI Research Engineer	Research Laboratory	Pakistan	Foundation Models	4 years	2 years

Participant	Current Position	Organization Type	Country	Primary Technical Expertise	Experience with LLMs	DevSecOps Experience
P12	DevSecOps Consultant	Technology Consultancy	Pakistan	Enterprise DevSecOps	3 years	10 years
P13	Senior Backend Engineer	E-commerce Company	Pakistan	API Security	2 years	5 years
P14	Cloud Solutions Architect	Multinational IT Company	Malaysia	Cloud Infrastructure	2 years	7 years
P15	Software Security Analyst	FinTech	Pakistan	Vulnerability Assessment	2 years	5 years
P16	DevOps Lead	Software House	Pakistan	DevSecOps Governance	3 years	8 years
P17	AI Product Engineer	AI Startup	Pakistan	LLM Applications	3 years	3 years
P18	Security Operations Engineer	Cybersecurity Firm	Pakistan	Threat Detection	2 years	6 years
P19	Principal Software Engineer	Enterprise Software Company	USA	Software Supply Chain Security	3 years	11 years
P20	Technology Strategy Consultant	Digital Transformation Consultancy	Pakistan	AI Governance & Digital Strategy	3 years	9 years

Findings and Discussion

The interview data revealed that the integration of Large Language Models (LLMs) within AI-augmented DevSecOps pipelines is reshaping secure software engineering practices while simultaneously introducing new architectural, governance, and organizational challenges. Through reflexive thematic analysis, five overarching themes emerged from the participants' narratives. These themes collectively demonstrate that although LLMs substantially improve software engineering productivity, practitioners consistently perceive them as intelligent decision-support tools rather than autonomous security solutions. Across all interviews, participants emphasized the continued importance of human expertise, governance mechanisms, and secure architectural design for responsible AI adoption.

Theme 1: LLMs as Intelligent Enablers of Secure Software Engineering

Participants consistently described LLMs as valuable assistants that enhance software development by improving code quality, security analysis, documentation, vulnerability interpretation, and deployment efficiency. Rather than replacing developers, AI supported engineers in performing repetitive and knowledge-intensive activities more efficiently.

Several participants explained that LLMs accelerated secure coding, automated documentation, infrastructure scripting, API development, and vulnerability assessment. Participants also highlighted improvements in developer productivity because engineers could spend less time searching documentation and more time solving complex engineering problems. Junior developers particularly benefited from AI-assisted learning and technical explanations.

These findings suggest that LLMs strengthen multiple phases of the Software Development Life Cycle by augmenting developer capabilities while improving collaboration between development, operations, and security teams. The findings support previous studies arguing that AI-powered software engineering enhances productivity, code quality, and continuous security integration (Robert et al., 2024; Joshi, 2025; Kumari et al., 2026). Similarly, participants' experiences reinforce evidence that AI-augmented DevSecOps facilitates continuous security throughout CI/CD environments (Allam, 2025; Beeram, 2026).

Theme 2: Human Oversight Remains Central to Secure AI Adoption

A dominant finding across all interviews was that practitioners did not trust LLMs to make autonomous security decisions. Participants repeatedly emphasized that AI-generated code, architectural recommendations, and vulnerability assessments require manual verification before implementation.

Software engineers explained that LLMs occasionally generated inaccurate, outdated, or overly generalized recommendations that failed to reflect organizational requirements or application-specific contexts. Consequently, organizations continued relying on peer reviews, penetration testing, static analysis tools, automated security testing, and senior engineering approval before deploying AI-generated outputs.

Participants viewed AI as complementary rather than substitutive, arguing that professional expertise remained indispensable for architectural decisions, security validation, and regulatory compliance. These findings strongly align with Explainable AI literature emphasizing transparency, accountability, and human-centered AI adoption (Rehmat et al., 2025; Mittal, 2026). The findings also support Socio-Technical Systems Theory, which proposes that organizational performance depends upon balancing technological innovation with human expertise and collaborative decision-making.

Theme 3: Architectural Integration Creates Technical and Governance Challenges

Although participants acknowledged significant productivity benefits, they consistently reported that integrating LLMs into enterprise DevSecOps pipelines introduces complex architectural challenges. These challenges extended beyond software implementation to include governance, scalability, interoperability, data privacy, and cloud infrastructure management.

Participants working in multinational organizations, banking, cloud computing, and enterprise software emphasized concerns regarding API dependency, deployment latency, software supply chain security, prompt engineering, and integration with existing CI/CD automation. Organizations operating within highly regulated environments also highlighted the need to ensure compliance with internal security standards before introducing AI services into production systems.

Another recurring concern involved protecting proprietary source code and sensitive organizational knowledge when interacting with external LLM services. Many participants therefore preferred private deployments, locally hosted models, or enterprise AI platforms that minimized data exposure.

These findings reinforce earlier research emphasizing software supply chain security, cloud-native DevSecOps, and AI governance as critical architectural priorities (Rusum, 2023; Vas, 2025; Coston et al., 2025; Mahimalur et al., 2026). They further demonstrate that successful LLM integration requires comprehensive architectural planning rather than simple tool adoption.

Theme 4: Trust, Explainability, and Governance Drive Responsible AI Implementation

Another major theme concerned organizational trust and governance. Participants consistently argued that successful LLM adoption depends upon explainability, transparent decision-making, organizational policies, and clearly defined governance frameworks.

Managers, consultants, architects, and security professionals explained that organizations must

establish formal policies governing AI usage, data privacy, model evaluation, accountability, and continuous monitoring. Participants warned that excessive dependence on AI could reduce developers' critical thinking and increase organizational security risks if appropriate governance mechanisms were absent.

Several participants emphasized employee training as an equally important component of AI adoption. Organizations introducing LLMs without educating developers regarding model limitations, hallucinations, privacy risks, and secure prompt engineering may inadvertently increase rather than reduce cybersecurity risks.

These observations correspond closely with the Technology–Organization–Environment Framework, which argues that successful technology adoption depends not only upon technological capability but also organizational readiness and governance structures. The findings also complement previous research highlighting explainability, transparency, and responsible AI governance as prerequisites for enterprise AI implementation (Lin et al., 2025; Mittal, 2026; Rehmat et al., 2025).

Theme 5: Future DevSecOps Will Combine Intelligent Automation with Human Expertise

Participants viewed the future of software engineering as one characterized by collaboration between AI systems and experienced professionals rather than full automation. Engineers anticipated increasing use of LLMs for secure coding, automated documentation, threat modeling, compliance support, vulnerability explanation, software architecture reviews, and incident response.

However, respondents consistently rejected the notion that AI would replace experienced software engineers or security specialists. Instead, they envisioned hybrid DevSecOps environments where intelligent automation performs routine technical activities while humans retain responsibility for architectural design, governance, ethical decision-making, and final security validation.

Participants further suggested that future enterprise adoption will require domain-specific models, stronger explainability, improved benchmarking, integration with specialized security knowledge bases, and continuous model evaluation to improve reliability and reduce hallucinations.

These findings support recent literature suggesting that future AI-enabled software engineering will increasingly emphasize human–AI collaboration, explainable AI, secure automation, and responsible governance rather than fully autonomous software development (Ahi & Valizadeh, 2025; Bajlur et al., 2026; Singh, 2026).

Synthesis of Findings

Collectively, the five themes demonstrate that LLMs are significantly transforming AI-augmented DevSecOps pipelines by improving productivity, secure coding, vulnerability management, collaboration, and software quality. Nevertheless, practitioners consistently perceived AI as an intelligent assistant rather than an autonomous decision-maker. Human oversight, organizational governance, explainability, secure software architecture, and continuous validation emerged as essential conditions for successful implementation. The findings indicate that responsible adoption requires balancing technological innovation with organizational capability, regulatory compliance, and professional engineering expertise. Overall, the study illustrates that sustainable AI-enabled DevSecOps depends upon integrating intelligent automation with robust governance and human-centered secure software engineering practices.

Table 2: Thematic Analysis of Findings

Theme	Representative Codes	Subthemes	Interpretation
Theme 1: LLMs as Intelligent Enablers of Secure Software Engineering	Secure code generation; vulnerability explanation; documentation automation; CI/CD support; infrastructure scripting; productivity	AI-assisted coding; developer augmentation; secure SDLC; operational efficiency	LLMs improve productivity and software quality by augmenting developers throughout the secure software lifecycle while reducing repetitive engineering tasks.
Theme 2: Human Oversight Remains Central to Secure AI Adoption	Human validation; peer review; penetration testing; explainability; hallucinations; manual approval	Trust; professional judgment; AI limitations; secure verification	Practitioners consistently validate AI-generated outputs before deployment, demonstrating that human expertise remains indispensable despite increasing automation.
Theme 3: Architectural Integration Creates Technical and Governance Challenges	Cloud integration; API dependency; software supply chain; scalability; interoperability; private deployment; data privacy	Enterprise architecture; cloud-native DevSecOps; governance; secure infrastructure	Successful LLM integration requires comprehensive architectural planning, governance, and secure infrastructure rather than simple AI implementation.
Theme 4: Trust, Explainability, and Governance Drive Responsible AI Implementation	AI governance; transparency; organizational policy; compliance; employee training; accountability	Explainable AI; organizational readiness; policy development; risk management	Responsible AI adoption depends upon explainable outputs, governance frameworks, workforce capability, and organizational security policies.
Theme 5: Future DevSecOps Will Combine Intelligent Automation with Human Expertise	Human–AI collaboration; intelligent automation; model evaluation; security knowledge bases; continuous improvement	Future software engineering; responsible AI; collaborative intelligence	Future DevSecOps ecosystems will combine AI automation with human expertise to achieve secure, scalable, and trustworthy software engineering.

Summary of Findings

The thematic analysis revealed that practitioners overwhelmingly recognize the strategic value of LLMs in strengthening secure software engineering and AI-augmented DevSecOps pipelines. While AI substantially improves development efficiency, code quality, and security automation, organizations continue to rely on human expertise for validation, governance, and architectural decision-making. Five interconnected themes emerged: intelligent software engineering support, human oversight, architectural integration challenges, governance and explainability, and the future of collaborative AI-enabled DevSecOps. Together, these findings suggest that the long-term success of LLM adoption depends not on replacing software engineers but on creating secure, explainable, and well-governed human–AI partnerships within enterprise software engineering environments.

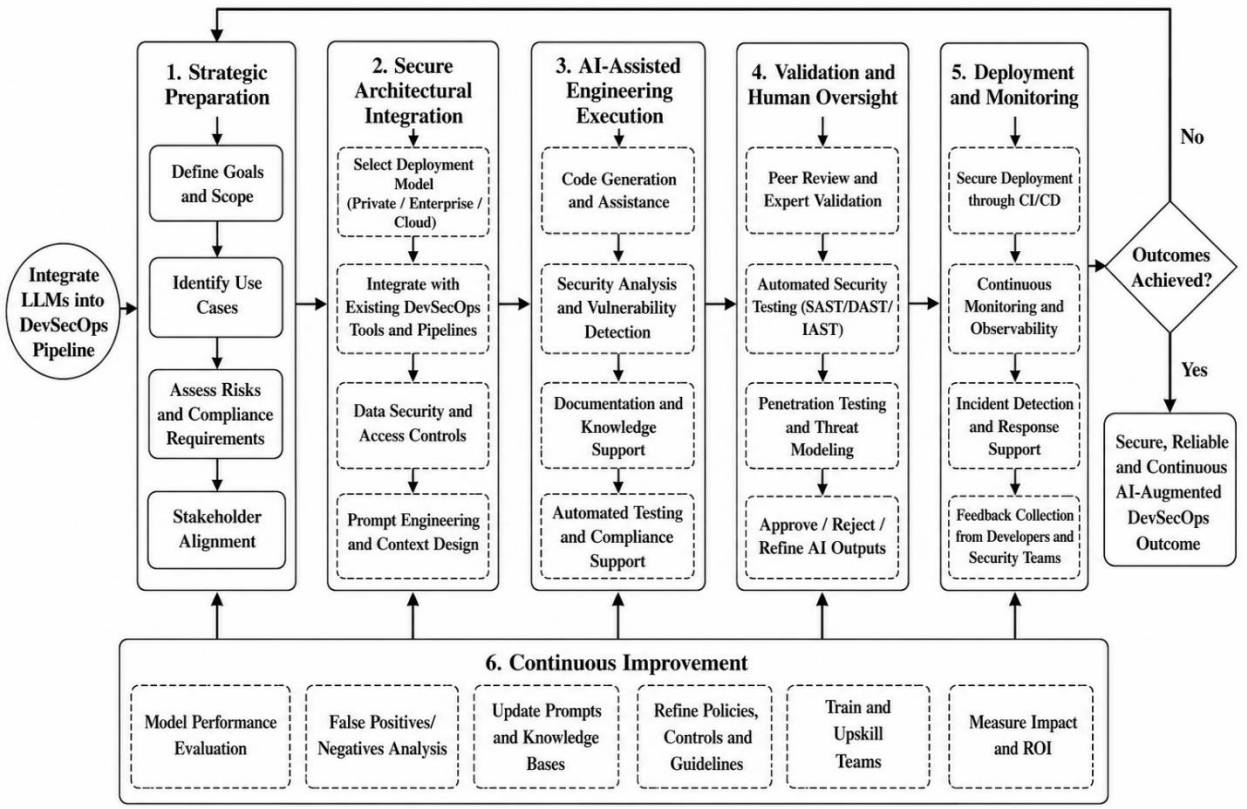

Figure 1. **Practitioner-Informed Action Model for Secure Integration of Large Language Models in DevSecOps Environments**

Conclusion

This study explored how Large Language Models (LLMs) are being integrated into AI-augmented DevSecOps pipelines and examined their influence on secure software engineering practices and architectural decision-making. Drawing upon the perspectives of software engineers, DevSecOps practitioners, security architects, cloud specialists, AI engineers, and technology consultants, the study identified five major themes that collectively explain the opportunities and challenges associated with enterprise adoption of LLMs. The findings indicate that LLMs have evolved beyond simple code-generation tools and are increasingly supporting secure coding, vulnerability analysis, security documentation, architectural reviews, compliance assistance, incident investigation, and software deployment activities. These capabilities significantly enhance productivity, improve collaboration among development, operations, and security teams, and accelerate secure software delivery.

Despite these advantages, the findings also demonstrate that organizations remain cautious about relying exclusively on AI-generated recommendations. Participants consistently emphasized that human expertise, governance mechanisms, explainability, and rigorous validation remain fundamental to secure software engineering. Architectural challenges such as software supply chain security, cloud integration, data privacy, model reliability, scalability, and regulatory compliance continue to influence implementation decisions. Consequently, successful AI-augmented DevSecOps depends upon achieving an appropriate balance between intelligent automation and professional engineering judgment. Overall, the study concludes that LLMs should be viewed as collaborative partners that strengthen secure

software engineering while preserving human accountability for critical security and architectural decisions.

Theoretical Implications

This study contributes to the growing body of knowledge on AI-assisted software engineering by extending the application of Socio-Technical Systems Theory and the Technology–Organization–Environment (TOE) Framework within the context of AI-augmented DevSecOps. The findings reinforce the socio-technical perspective by demonstrating that successful LLM adoption depends not only on technological capability but also on human expertise, organizational collaboration, governance, and continuous learning. The study further supports the TOE Framework by illustrating that technological readiness, organizational maturity, security culture, and external regulatory pressures collectively shape enterprise adoption of LLM-enabled DevSecOps practices.

Additionally, the research extends existing literature by providing qualitative insights into practitioner experiences, an area that remains comparatively underexplored. While previous studies have predominantly focused on technical architectures, algorithms, and automation performance, this research highlights the importance of trust, explainability, governance, and human-centered decision-making in determining the effectiveness of AI-assisted secure software engineering.

Practical Recommendations

The findings provide several practical recommendations for organizations seeking to integrate LLMs into secure software development environments.

First, organizations should position LLMs as decision-support systems rather than autonomous security solutions. AI-generated outputs should always be validated through peer reviews, automated security testing, and experienced software engineers before deployment.

Second, enterprises should establish comprehensive AI governance frameworks covering data privacy, model evaluation, prompt engineering, explainability, accountability, and continuous monitoring. Clear organizational policies will reduce risks associated with hallucinations, insecure recommendations, and inappropriate handling of proprietary software assets.

Third, organizations should prioritize private or enterprise-grade LLM deployments when processing sensitive source code, security documentation, or confidential architectural information. This approach strengthens software supply chain security while ensuring compliance with regulatory and organizational requirements.

Fourth, continuous workforce development should accompany AI adoption. Training programs focusing on secure prompt engineering, AI limitations, responsible AI usage, and secure software engineering practices will improve developer capability and reduce excessive dependence on automated recommendations.

Finally, organizations should integrate LLMs with existing DevSecOps tools such as static application security testing, dynamic security testing, vulnerability scanners, compliance monitoring platforms, and threat intelligence systems. Such layered integration will maximize the benefits of AI while maintaining robust security assurance throughout the software development lifecycle.

Limitations and Future Research Directions

Although this study provides valuable insights into AI-augmented DevSecOps, several limitations should be acknowledged. The study employed a qualitative research design that focused on obtaining rich practitioner perspectives rather than producing statistically generalizable findings. Participants represented selected software engineering domains and organizational settings; therefore, the findings may not reflect all industries, geographical regions, or organizational contexts. Furthermore, the rapidly evolving nature of Large Language Models means that technological capabilities, governance practices, and architectural approaches are likely to continue changing over time.

Future research may adopt mixed-methods or quantitative approaches to validate the themes identified in this study across larger and more diverse populations. Comparative studies examining differences between industries, organizational maturity levels, cloud environments, and national regulatory frameworks would further enrich understanding of enterprise AI adoption. Researchers may also investigate domain-specific LLMs for cybersecurity, explainable AI techniques for secure software engineering, human–AI collaboration models, software supply chain protection, AI governance frameworks, and autonomous DevSecOps agents. Longitudinal studies exploring how organizations adapt to continuous advances in generative AI would also provide valuable insights into the future evolution of secure software engineering. Collectively, these research directions can contribute to developing more trustworthy, resilient, and human-centered AI-augmented DevSecOps ecosystems.

REFERENCES

Abrar, K. (2018). Impact of augmented reality on consumer purchase intention with the mediating role of customer brand engagement: moderating role of interactivity in online shopping. *Bahria University Journal of Management & Technology*, *1*(2), 64-80.

Abrar, K., & Amjad, S. I. (2026). Gamified Advertising in the Metaverse: How Pakistani Brands Can Leverage Virtual Experiences for Customer Engagement. *Social Science Review Archives*, *4*(1), 6022-6041.

Ahi, K., & Valizadeh, S. (2025, June). Large Language Models (LLMs) and Generative AI in Cybersecurity and Privacy: A Survey of Dual-Use Risks, AI-Generated Malware, Explainability, and Defensive Strategies. In *2025 Silicon Valley Cybersecurity Conference (SVCC)* (pp. 1-8). IEEE.

Allam, H. (2025). Code Meets Intelligence: AI-Augmented CI/CD Systems for DevOps at Scale. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *6*(1), 137-146.

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.

Bajlur, R. M., Jamil, H. M. S., Khan, M. I., Sharaban, T., Aiasha, S., Islam, P. M., ... & Hossain, S. (2026). A Survey on Large Language Models in Software Security: Opportunities and Threats. *Computers*, *15*(4), 226.

Beeram, S. (2026). AI-Augmented DevSecOps in Azure Pipelines. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *7*(1), 46-48.

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.

Byrne, D. (2022). A worked example of Braun and Clarke's approach to reflexive thematic analysis. *Quality & quantity*, *56*(3), 1391-1412.

Coston, I., Hezel, K. D., Plotnizky, E., & Nojoumian, M. (2025). Enhancing secure software development with AZTRM-D: An AI-integrated approach combining DevSecOps, risk management, and zero trust. *Applied Sciences*, *15*(15), 8163.

Hole, L. (2024). Handle with care; considerations of Braun and Clarke's approach to thematic analysis. *Qualitative research journal*, *24*(4), 371-383.

Imtiaz, A., Khan, S. S., & Ullah, I. (2026). Peacebuilding Narratives in Conflict Zones: Insights from NGOs Working in Afghanistan and Pakistan. *Social Science Review Archives*, *4*(1), 5462-5482.

Isabella, R., Kenji, W., & Alejandro, S. (2023). AI-Augmented DevSecOps: Automating Security Testing in CI/CD Pipelines. *Best Journal of Innovation in Science, Research and Development*, *2*(10), 688-704.

Joshi, S. (2025). A review of generative AI and DevOps pipelines: CI/CD, agentic automation, MLOps integration, and large language models. *CD, Agentic Automation, MLOps Integration, and Large Language Models (June 01, 2025)*.

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., 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, R. A. S., Abrar, K., Parveen, S., Bhurgri, S. S., & Panhwar, A. J. (2026). *Integrating generative AI into software development lifecycles: A comparative qualitative study of engineering practices in Pakistan and the USA*. *The Asian Bulletin of Big Data Management*, *6*(1), 329–345.

Kumar, V. (2025). Next-Generation Software Engineering: A Study on AI-Augmented Development, DevSecOps and Low-Code Frameworks. *Next-Generation Software Engineering: A Study on AI-Augmented Development, DevSecOps and Low-Code Frameworks (April 24, 2025)*.

Kumari, K. S., Sundaravadivazhagan, B., Indumathy, V., & Maladhy, D. (2026). Detailing AI techniques and tools for software engineering acceleration and automation. In *Advances in Computers* (Vol. 141, pp. 183-209). Elsevier.

Lin, H. W., Stout, W. M., & Morris, T. J. (2025, October). Assessing Software Security Maturity via an AI-Augmented Risk-Based Framework. In *2025 IEEE International Carnahan Conference on Security Technology (ICCST)* (pp. 1-6). IEEE.

Mahimalur, R. K., Kolla, V. G., Tank, C. D., & Meesala, S. T. (2026, February). AI-Integrated DevSecOps for Scalable, Secure, and Modern Cloud Architectures. In *2026 International Conference on Electronics and Renewable Systems (ICEARS)* (pp. 1742-1749). IEEE.

Mittal, A. (2026). AI-Enabled DevSecOps for Secure Cloud Deployments. In *Revolutionizing the Cloud: Generative AI, Security, and Sustainability* (pp. 323-347). Cham: Springer Nature Switzerland.

Mittal, A. (2026, May). Explainable AI-Augmented DevSecOps for Secure and Reproducible Cloud-Native. In *Computer Applications in Industry and Engineering: 38th International Conference, CAINE 2025, New Orleans, LA, USA, October 20–21, 2025, Proceedings* (p. 69). Springer Nature.

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., Rumaan, M., Baig, J., & Abrar, K. (2025). Human-Centred Explainable AI in Emerging Markets: Trust and Confidence Among Non-Technical Users in Pakistan. *Annual Methodological Archive Research Review*, *3*(10), 145-163.

Robert, J. E., Ozkaya, I., & Schmidt, D. C. (2024). Transforming Software Engineering and Software Acquisition with Large Language Models. In *Artificial Intelligence and Large Language Models* (pp. 180-200). CRC Press.

Rusum, G. P. (2023). Secure Software Supply Chains: Managing Dependencies in an AI-Augmented Dev World. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *4*(3), 85-97.

Rusum, G. P. (2025). Adaptive DevSecOps: Integrating AI-Driven Threat Detection in Continuous Delivery Pipelines. *American International Journal of Computer Science and Technology*, *7*(5), 13-27.

Singh, R. P. (2026). A Review of Machine Learning Approaches for Software Testing in Secure Modern Systems. *Journal of Artificial Intelligence and Semiconductor Integrated Hardware*, *1*(2), 14-20.

Vas, M. R. (2025). AI-Augmented DevSecOps for Cloud Environments: Bridging Security Gaps in Modern Continuous Delivery Pipelines. *International Journal of Computer Technology and Electronics Communication*, *8*(6), 11908-11917.

Wei, L. (2019). AI-Augmented DevSecOps Pipelines: Enabling Continuous Security Integration in Large-Scale Software System. *American International Journal of Computer Science and Technology*, *1*(5), 1-9.

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.