**Cloud enabled AI framework for optimizing the Software Development Lifecycle**

**ABSTRACT**

Tasks, people, and resources must be carefully coordinated as part of the Software Development Lifecycle (SDLC). This frequently results in errors and inefficiencies that raise project costs. This study introduces a new cloud-enabled AI framework designed to improve SDLC management. It uses a transformer-based Large Language Model to automate phase analysis, improve decision-making, and optimize resource use. The model has been trained on a specific SDLC dataset through transfer learning, which helps with smart task scheduling and risk classification. It is connected to cloud infrastructure, providing flexibility and scalability for today’s software projects. The framework cuts down on manual work, helps avoid project delays, and encourages cost-effective operations. By using natural language processing capabilities, it gives project managers useful insights that support efficient and sustainable development practices. This work engages the use of AI in SDLC, providing a strong solution to boost efficiency and meet business goals in software engineering.

**KEYWORDS**

Artificial Intelligence, Software Development Lifecycle, Cloud Computing, Transformer Models, Resource Optimization, Natural Language Processing, Project Management, SLDC Optimization.

**AUTHORS**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Name of the Student** | **Register No** |
| I | Mutyala Hari Priyanka | 22K61A05A9 |
| II | Batthula Pavan Teja | 22K61A0512 |
| III | Thommandru Ratna Kumari | 22K61A05G2 |
| IV | Narim Subbalakshmi | 22K61A05B4 |

**I. INTRODUCTION**

Effective management of the Software Development Lifecycle (SDLC) necessitates intricate coordination among diverse tasks, stakeholders, and resources to ensure alignment with quality standards and organizational objectives [1]. This complexity often demands experienced project managers, whose expertise incurs significant costs [2]. Despite their proficiency, human errors frequently result in delayed deliveries, inefficient resource utilization, and escalated expenses, undermining project success [3][4]. The interdependence of dynamic project elements further amplifies these challenges, exposing projects to risks and inefficiencies [5]. Recent advancements in transformer-based Large Language Models (LLMs) offer promising capabilities for addressing these issues by processing vast, unstructured datasets and extracting meaningful insights [6][7]. These models excel in understanding complex contexts, making them suitable for enhancing SDLC management through intelligent automation [8]. This study presents a cloud-enabled AI framework designed to support project managers in optimizing SDLC processes. By leveraging a transformer model, the framework automates critical tasks, enhances decision-making, and promotes resource efficiency, thereby reducing operational costs [9][10].

The proposed framework employs a transformer model comprising multiple blocks, each incorporating multi-head attention, normalization layers, and feed-forward networks, with a vocabulary size exceeding tens of thousands and high-dimensional embeddings to process SDLC documentation [11]. Trained via transfer learning on a structured dataset derived from extensive SDLC-related documents, the model utilizes an adaptive optimization algorithm to refine weights, ensuring robust performance [12][13]. A cloud-based infrastructure underpins the framework, enabling scalability, real-time accessibility, and seamless integration with project management workflows [14]. Users interact through an intuitive interface, uploading SDLC documents and specifying objectives, with outputs delivered via API-driven communication with the transformer backend [15]. This approach minimizes manual errors, accelerates project timelines, and aligns with Environmental, Social, and Governance (ESG) principles by optimizing resource use and reducing computational overhead [16]. The framework draws on prior work in AI-driven software development [17], requirement analysis [18], and cloud-based automation [19], offering a novel solution to longstanding SDLC challenges.

The core contributions of this study encompass the development of a scalable AI framework for SDLC optimization, the application of transformer models to enhance project management efficiency, and the integration of cloud technology to ensure accessibility and adaptability. By automating phase analysis and risk assessment, the framework empowers project managers with actionable insights, mitigates delays, and supports sustainable practices, advancing the field of software engineering [20].

**II. BACKGROUND STUDY**

This chapter provides a comprehensive review of the foundational concepts underpinning this research. It begins by examining the traditional Software Development Lifecycle (SDLC), identifying its inherent challenges and limitations. Subsequently, it explores the transformative role of Artificial Intelligence (AI) and Large Language Models (LLMs) in addressing these issues. Finally, it discusses the pivotal function of cloud computing as the enabling infrastructure for modern, AI-driven software development frameworks.

**2.1 The Traditional Software Development Lifecycle and Its Challenges**

The Software Development Lifecycle (SDLC) provides a structured framework for engineering software, encompassing a sequence of phases from requirements elicitation to deployment and maintenance. Historically, models such as the Waterfall model established a linear, sequential approach, where each phase must be completed before the next begins [4]. While this methodology offers predictability, its rigidity makes it ill-suited for projects with evolving requirements. In response, Agile methodologies were introduced, promoting iterative development, stakeholder collaboration, and flexible adaptation to change [2]. However, even Agile processes, while superior in managing change, still rely heavily on manual effort for tasks like requirements analysis, prioritization, testing, and risk management, making them susceptible to human error, inconsistencies, and security oversights [5]. These manual dependencies often lead to process bottlenecks, increased costs, and the accumulation of technical and security debt over the project's lifecycle.

**2.2 The Emergence of Artificial Intelligence in Software Engineering**

The integration of Artificial Intelligence (AI) into software engineering has emerged as a powerful strategy to mitigate the limitations of traditional methodologies. Initially, AI applications focused on discrete tasks such as automated testing and defect prediction [19]. The advent of generative AI and sophisticated Large Language Models (LLMs) like those based on the transformer architecture has significantly expanded this role [7, 6]. These models can now automate complex, language-centric tasks across the SDLC. For instance, LLMs are being successfully employed to generate use case diagrams from textual user stories, bridging the gap between requirements and design [8]. Furthermore, specialized frameworks like the AI-Analyst have demonstrated the capacity of LLMs to perform detailed SDLC analysis, including phase classification, task prioritization, and risk assessment, leading to substantial optimizations in business costs [9]. This shift from task-specific automation to holistic, intelligent assistance marks a significant evolution in software development practices [1].

**TABLE II: Traditional SDLC Challenges and AI-Cloud solutions**

|  |  |  |
| --- | --- | --- |
| **Challenges in Traditional SDLC** | **AI and Cloud enabled solution** | **Key References** |
| Manual requirement analysis | Automated analysis and classification of requirements using NLP and LLMs. | [6], [8], [18] |
| Human error in planning | AI-driven risk classification, task prioritization, and resource scheduling. | [9], [7] |
| Rigid and slow processes | Scalable, on-demand cloud infrastructure supports iterative and parallel workflows. | [14] |
| High Cost and inefficiency | Reduction of manual effort and rework through intelligent automation, leading to significant cost savings. | [9], [3] |
| Security oversights | Proactive identification of security debts and vulnerabilities through continuous analysis. | [5] |

**III. SELECTION CRITERIA**

This chapter outlines the systematic methodology employed to identify, screen, and select the scholarly literature for this review. To ensure a comprehensive, transparent, and reproducible search process, the PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) framework was adopted. The objective was to curate a final corpus of high-quality, original research articles focused on the optimization of the SDLC through AI and cloud computing.

**3.1 Search Strategy**

The literature search was conducted across several prominent academic databases to ensure broad coverage of the relevant research. The primary sources included Google Scholar, IEEE Xplore, Arxiv, ResearchGate, ScienceDirect, and Semantic Scholar. The search was guided by a targeted set of keywords designed to capture the core themes of the project, including: "AI in software development lifecycle", "LLM for SDLC optimization", "Cloud computing in software engineering", "AI-driven project management", and "Automation in SDLC using AI".

**3.2 Inclusion and Exclusion Criteria**

All articles identified through the search strategy were subjected to a rigorous screening process based on the predefined inclusion and exclusion criteria outlined in Table III. This step was essential to filter for relevance, quality, and originality.

**TABLE III: Inclusion and Exclusion criteria for article selection**

|  |  |
| --- | --- |
| **Criteria Type** | **Description** |
| Inclusion | Focused on the application of AI, LLMs, and cloud computing for SDLC optimization. |
| Original research, not a review or survey paper. |
| The publication date is between 2021 and 2025. |
| The full text of the article is available for analysis. |

|  |  |
| --- | --- |
| Exclusion | Articles that were identical copies found in different databases were removed. |
| Studies where AI or the SDLC were mentioned only in passing without being the core topic. |
| Papers lacking a clear description of their methodology, results, or limitations. |

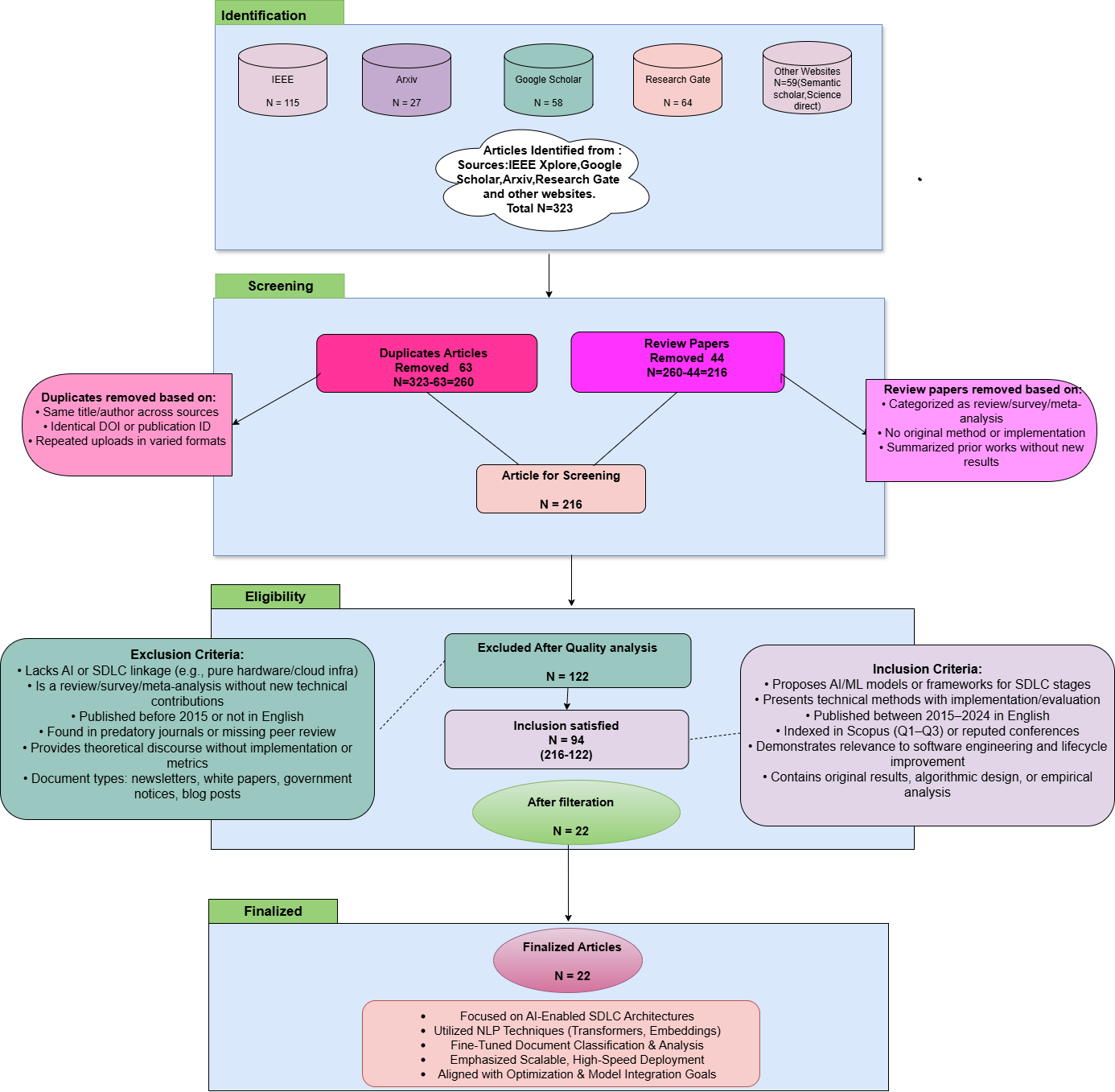
**3.3 Article Selection and Screening Process**

The PRISMA framework guided the multi-stage filtering process, which is visually summarized in the flowchart in Figure 1. An initial pool of 323 articles was identified through the database search. After removing 63 duplicates, 260 unique articles remained. A further 44 review articles were excluded to maintain a focus on primary research, leaving 216 articles for the initial screening phase.

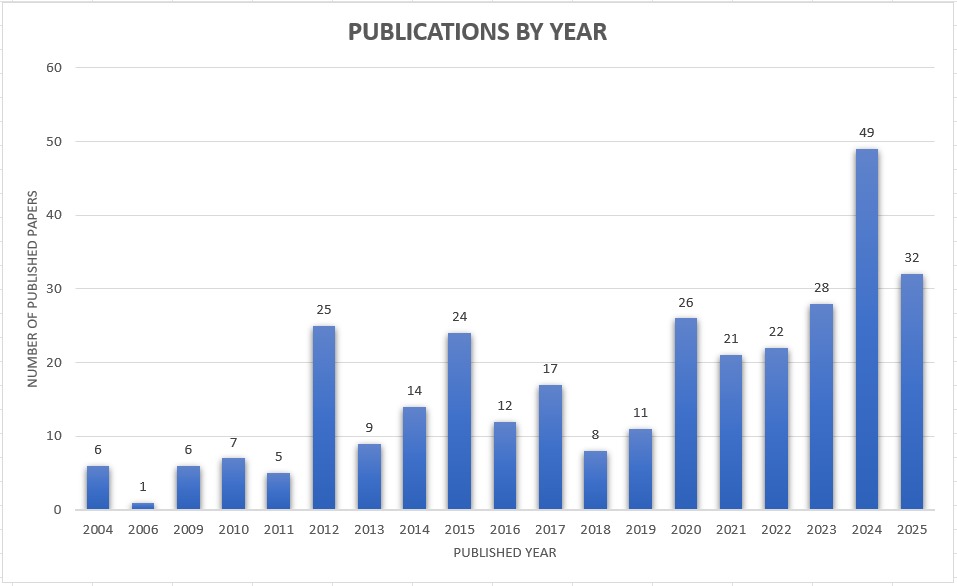
During this screening, the titles and abstracts of the 216 articles were evaluated against the inclusion criteria, resulting in a shortlist of 94 potentially relevant studies. These articles were then subjected to a full-text quality assessment, which examined methodological rigor, the clarity of results, and the discussion of limitations and future work. This final, intensive review yielded the 22 high-quality articles that form the basis of this literature review. The complete summary of this selection process is detailed in Table IV.

**TABLE IV: ARTICLES SELECTION SUMMARY AS PER PRISMA**

|  |  |  |
| --- | --- | --- |
| **Item** | **Description** | **count (N)** |
| Identification | Initial articles collected from database searches. | 323 |
| Screening | Duplicate articles removed from the initial pool. | 260 |
| Screening | Review papers and survey excluded | 216 |
| Eligibility | Articles selected after title and abstract screening. | 94 |
| Included | Final articles selected after Quality analysis | 22 |



**Figure 1: Flowchart representing the PRISMA standard applied for paper selection**



**Figure 2: Research papers on Optimization of SDLC published from 2004-2025**

**REFERENCES**

[1] J. Sauvola et al., “Future of software development with generative AI,” 2024.

[2] A. Q. Gill, “Agile system development lifecycle for AI systems: Decision architecture,” 2025.

[3] B. Madupati, “AI’s impact on traditional software development: A technical look,” 2024.

[4] P. Agarwal, A. Singhal, and A. Garg, “SDLC model selection tool and risk incorporation,” 2017.

[5] C. Boufaied, T. A. Ghaleb, and Z. Masood, “Security debt in practice: Nuanced insights from practitioners,” 2025.

[6] Y. Zhang and Y. Li, “The classification of software requirements using large language models: Emerging results and future directions,” 2024.

[7] K. R. Raghi et al., “Software development automation using generative AI,” 2024.

[8] H. R. P. Dias et al., “Automated use case diagram generator using NLP and ML,” 2023.

[9] N. Faruqui et al., “AI-Analyst: An AI-assisted SDLC analysis framework for business cost optimization,” 2024.

[10] A. Krishna Kaiser and V. Meda, “AI integration in software development and operations,” 2024.

[11] H. Cervantes et al., “An LLM-assisted approach to designing software architectures using ADD,” 2024.

[12] W. Uzair and S. Naz, “Six-tier architecture for AI-generated software development: A large language models approach,” 2023.

[13] B. S. Puranik et al., “Enhancement of model driven software development using AI,” 2024.

[14] N. Arshad, T. A. Butt, and M. Iqbal, “A comprehensive framework for intelligent, scalable, and performance-optimized software development,” 2025.

[15] A. Ferrari, S. Abualhaija, and C. Arora, “Model generation with LLMs: From requirements to UML diagrams,” 2024.

[16] M. T. Baldassarre et al., “POLARIS: A framework for trustworthy AI systems,” 2024.

[17] S. Jüngling, M. Peraic, and A. Martin, “Towards AI-based solutions in the SDLC,” 2020.

[18] V. K. Vineetha and P. Samuel, “A multinomial Naïve Bayes classifier for identifying actors and use cases from software requirement specification documents,” 2022.

[19] A. Fatima et al., “Software requirements prioritisation using machine learning,” 2023.

[20] G. J. Ramackers et al., “From prose to prototype: Synthesising executable UML models from natural language,” 2021.