e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:11/November-2024 Impact Factor- 8.187 www.irjmets.com

AUTODEVOPS: AI-DRIVEN SOFTWARE DEVELOPMENT & DEPLOYMENT AUTOMATION

Ansha Ashraf[*1], Devi Krishna PS[*2], Nandana OR[*3], Safa Firous[*4], Remya PC[*5]

[*1,2,3,4]Department Of Computer Science And Engineering, IES College Of Engineering, Chittilappilly, Thrissur, India.

[*5]Head Of Computer Science Department, IES College Of Engineering, Chittilappilly, Thrissur, India.

ABSTRACT

In traditional software development, the +++process of translating project requirements into a functional design, writing code, and creating test cases is often labour-intensive, time-consuming, and prone to human error. These challenges can lead to delays, inconsistencies, and increased costs, particularly in complex projects. Addressing these issues, we propose a Generative AI system that revolutionizes the software development lifecycle by automating critical stages. This system begins by employing advanced Natural Language Processing (NLP) to analyses project requirements, extracting essential information to generate a comprehensive project design, including architecture diagrams and class structures. Following the design phase, the system autonomously generates code tailored to the identified design patterns and suggests deployment steps, leveraging containerization tools like Docker and Kubernetes for efficient implementation. Furthermore, the system automatically produces detailed test cases in a structured table format, ensuring thorough and consistent validation of the software. By integrating state-of-the-art NLP models, code generation techniques, and deployment technologies, this Generative AI system enhances the efficiency, accuracy, and reliability of software development, addressing common challenges and paving the way for more streamlined and effective software engineering practices.

Keywords: Software Development Life Cycle (SDLC), Natural Language Processing (NLP), Kubernetes, Docker, Open AI.

I. INTRODUCTION

Existing systems for managing the software development lifecycle (SDLC) often rely on traditional, manual processes that can be inefficient and prone to errors. These systems typically involve labor-intensive tasks, which slow down development and increase the likelihood of delays, especially in large-scale projects. As software development grows more complex, with teams managing multiple components, tight deadlines, and evolving requirements, these conventional methods fall short in delivering the speed and reliability necessary for modern development environments. While automation tools exist, they are often fragmented; requiring integration across different stages, which further complicates the development process.

AutoDevOps is a groundbreaking system designed to address these limitations by automating and optimizing the entire SDLC using advanced artificial intelligence (AI) and modern software engineering tools. By integrating AI-driven automation at critical stages—from requirement analysis and code generation to deployment—AutoDevOps significantly reduces reliance on manual processes. The system leverages Python, Django, Docker, and Kubernetes for seamless deployment, while libraries like SpaCy and the OpenAI API enable powerful natural language processing (NLP) and machine learning capabilities. AutoDevOps enhances the speed and quality of software delivery by automating routine tasks, making it particularly valuable for continuous integration/continuous deployment (CI/CD) environments. Its modular, scalable architecture allows it to adapt to projects of any size, enabling teams to focus on innovation and strategy. Ultimately, AutoDevOps empowers development teams to deliver high-quality software more efficiently, ensuring better business outcomes and increased customer satisfaction.

The main keywords that we used are explained in the following section:

SDLC: It is a step-by-step process for creating software. It begins with planning and defining requirements, followed by designing how the system will work. The development phase involves coding the software, and then it's thoroughly checked for issues during testing. This ensures high-quality, efficient software creation.

NLP: It is a branch of artificial intelligence that helps computers understand, interpret, and respond to human

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:11/November-2024 Impact Factor- 8.187 www.irjmets.com

language. It enables machines to process and analyze large amounts of natural language data, such as text or speech, and perform tasks like language translation, sentiment analysis, text classification, and chatbots. NLP combines linguistics with machine learning to bridge the gap between human communication and computer understanding.

Kubernetes: is an open-source platform that automates the deployment, scaling, and management of containerized applications. It helps ensure apps run smoothly by distributing workloads, scaling based on demand, and restarting failed containers.

Docker: it is a platform that allows developers to package applications and their dependencies into containers. These containers are lightweight, portable, and can run consistently across different environments. Docker simplifies the process of developing, testing, and deploying applications by ensuring that they work the same regardless of where they arerun, whether on a developer's machine, in a test environment,or in production.

Open AI: Open AI is an artificial intelligence research organization that focuses on developing advanced AI technologies. It aims to create and promote friendly AI that benefits humanity as a whole. OpenAI is known for its cutting-edge work in AI, particularly with language models like GPT-3 and GPT-4, which can understand and generate human-like text. OpenAI also works on a wide range of AI applications, including robotics, image generation, and reinforcement learning, with a mission to ensure that AIserves the best interests of society.

II. LITERATURE SURVEY

AI-driven Continuous Integration and Continuous Deployment (CI/CD) automates the entire software delivery process, using AI to streamline code submission, integration, testing, and deployment. It enhances agility by minimizing manual errors and enabling parallel testing of multiple versions, accelerating feature deployment. Key technologies include AI-powered automated testing (unit, integration, regression, and end-to-end), which analyzes results and suggests fixes, and automated source code versioning to track changes and detect issues. AI-driven CI/CD also automates deployment across environments like staging and production, while managing dependencies and build processes by identifying compatibility issues. This approach reduces costs, improves efficiency, and accelerates the SDLC, making it essential for modern software development[1].Code assistance tools, powered by hybrid models that integrate pre- trained language models like BERT, RoBERTa, and LUKE with the Marian Causal Language Model, are transforming developer productivity and code generation accuracy. These models, tested on datasets like CoNaLa and DJANGO, show significant gains in precision, thanks to their advanced natural language understanding and code synthesis capabilities. However, the complexity of these models and the requirement for fine-tuning can present challenges during deployment and integration into existing development workflows. Despite this, their potential to streamline coding tasks and improve accuracy makes them valuable tools in modern software development[2]. AI-driven software development transforms the entire lifecycle, from planning to deployment, by leveraging technologies such as automated code generation with models like OpenAI Codex, intelligent testing with AI- based frameworks like Test.ai, and continuous integration tools like Jenkins and CircleCI. These technologies enhance efficiency, quality, and innovation, enabling faster and more reliable development. However, the integration of AI-driven tools may introduce complexity, requiring teams to adapt to new processes and workflows for effective implementation. Despite these challenges, the impact on software delivery speed and accuracy makes AI-driven development a powerful approach in modern software engineering.[3]. An analysis of 13 research articles from 2015 to 2023 on NLP technologies for automating software testing tasks, especially test case generation, highlights several advancements. Key findings include seven NLP techniques, such as Named Entity Recognition (NER) and Dependency Parsing, and tools like spaCy and NLTK. Additionally, it identifies the use of the Testing Automation Framework (TAF) and algorithms like Sequence-to-Sequence Models and Transformer Networks. These technologies enhance test case automation by improving efficiency and accuracy, providing valuable insights for researchers and engineers aiming to advancetesting practices. [4].

Large language models like ChatGPT to generate test cases from bug reports, improving upon traditional methods that often miss key scenarios. Testing on the Defects4J dataset showed that LLMs can create executable test cases for up to 50% of bugs, even with new reports, enhancing fault localization and automated repair[5].Planning-Guided Transformer Decoding (PG-TD), a new approach that improves code generation by using a planning algorithm for lookahead searches and testing against public test cases. PG- TD consistently

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:11/November-2024 Impact Factor- 8.187 www.irjmets.com

outperforms traditional decoding methods, producing higher-quality, more controllable code[6].37 studies on machine learning models for automatic code generation, covering areas like description-to-code, code-to-description, and code-to-code, and highlights advancements made possible by models such as RNNs, transformers, and CNNs. It emphasizes the potential of ML to automate complex tasks beyond traditional tools, contributing to the development of commercial solutions like Tabnine and GitHub Copilot[7]. By integrating AI techniques like grammar-based fuzzing, machine learning, behavioral modeling, and control dependency analysis, I aim to enhance. Automated Test Case Generation (TCG), improving test effectiveness and efficiency and potentially revolutionizing software testing[8]. Automating the classification of software requirements into Functional and Non-Functional categories was achieved using the Bag of Words model combined with a Support Vector Machine (SVM), which resulted in an average F-measure of 0.74. Future work aims to improve accuracy by incorporating additional algorithms, such as Logistic Regression and advanced models like BERT or RoBERTa, to refine classification performance and enhance the overall effectiveness of the system.[9]. A tool that automatically generates source code from structured flowcharts, which helps in designing algorithms. Evaluated by 5 experts and 93 general users, the tool achieved high satisfaction scores, indicating its effectiveness in meeting its objectives[10].

System of Systems Lifecycle Management (SoSLM), a new approach extending Product Lifecycle Management (PLM) to improve the management of large-scale automation systems, with practical implementation shown through the Arrowhead framework for IIoT applications[11]. The Use Case Modeling for System-level Acceptance Tests Generation (UMTG) approach, which uses NLP to convert natural language use case specifications into structured test models and applies constraint-solving techniques to generate precise test cases. Validation through industrial case studies demonstrates that UMTG enhances the efficiency and accuracy of acceptance test case generation[12]. Machine learning algorithms like SVM and Random Forest to predict future code changes by analyzing a repository's development history, aiming to improve resource allocation and reduce costs. The approach highlights how AI can enhance software development and maintenance by forecasting changes and automating tasks[13]. An end-to-end machine learning model, fine-tuned from pre-trained language models, is proposed for generating Python code, achieving a BLEU score of 0.22 and a 46% improvement over baseline models. This approach automates function generation from code signatures and documentation, highlighting advancements inefficiency and automation in software development[14].

A novel automated compliance checking (ACC) system is presented, integrating semantic NLP for regulatory text extraction and EXPRESS data techniques for transforming Building Information Models (BIMs). With a recall rate of 98.7% and precision of 87.6%, the system enhances accuracy and efficiency in compliance checking[15]. A method for automating source code generation is introduced, using component-oriented programming and metadata to improve software quality and accelerate development. This approach enhances efficiency, reduces manual effort, and ensures robust, well-structured code, especially for large-scale systems.[16].GADGET, an advanced test generation system that uses combinatorial optimization to achieve full conditionand decision coverage for complex C/C++ programs, addressing limitations of previous methods[17].

III. BACKGROUND & MOTIVATION

a) The Evolution of Software Development

The software development landscape has undergone significant changes over the years, with increasing complexity and demands for faster delivery cycles. Traditional software development practices, which often involve a high degree of manual effort, are struggling to keep pace with these evolving demands. The shift towards more agile methodologies and the integration of continuous integration/continuous deployment (CI/CD) pipelines have improved efficiency to some extent, but they still rely heavily on human intervention. This reliance can lead to errors, inconsistencies, and delays, particularly in large-scale projects where multiple teams must collaborate across different phases of the software development lifecycle (SDLC).

The emergence of DevOps practices has further pushed the boundaries of what can be achieved in software development, fostering a culture of collaboration between development and operations teams. However, even with DevOps, many processes remain manual and prone to human error. This is where AI-driven automation can play a transformative role, automating repetitive tasks, enhancing decision-making processes, and ensuring that the SDLC is more streamlined, efficient, and reliable.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:11/November-2024 Impact Factor- 8.187 www.irjmets.com

b) The Need for Automation

The motivation behind developing AutoDevOps is rooted in the challenges faced by modern software development teams. As projects grow in complexity and the pressure to deliver high-quality software quickly increases, the need for automation becomes more apparent. Manual processes, such as requirement analysis, code review, and testing, are not only time-consuming but also introduce the risk of human error. These errors can lead to costly delays, rework, and in some cases, failures in production environments.

AutoDevOps aims to address these challenges by integrating AI-driven automation into every stage of the SDLC. By automating the analysis of project requirements using natural language processing (NLP) and generating code through AI models, AutoDevOps reduces the burden on developers and ensures that software is developed according to best practices. Additionally, automated testing and deployment processes further enhance the reliability and efficiency of software delivery.

c) Advancements in AI and NLP

The development of AutoDevOps is also motivated by recent advancements in AI and NLP technologies. Tools like SpaCy and the OpenAI API have made it possible to process and understand unstructured text with a high degree of accuracy, enabling the automated analysis of project requirements. Furthermore, AI models capable of generating code based on structured inputs have matured, making it feasible to automate large portions of the development process.

These advancements have created an opportunity to rethink how software is developed and deployed. By leveraging AI and NLP, AutoDevOps not only automates routine tasks but also ensures that the software is developed in a way that is aligned with the latest industry standards and best practices. This automation leads to significant improvements in productivity, quality, and time-to-market, making it an essential tool for modern development teams.

d) Addressing Industry Challenges

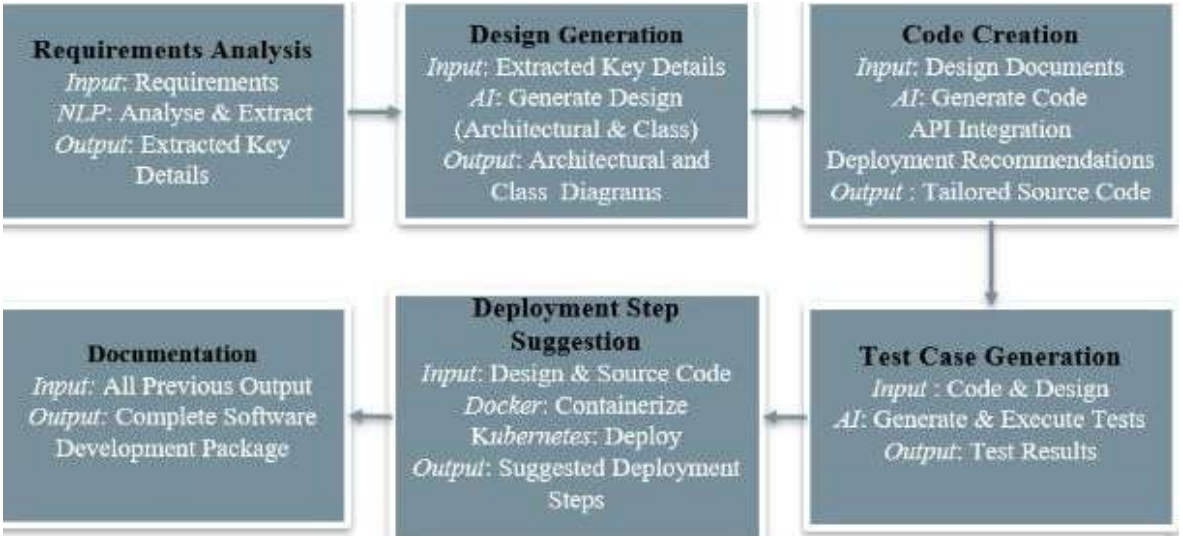
The software industry is facing several key challenges, including the need to accelerate development cycles, reduce costs, and maintain high standards of quality and security. AutoDevOps is designed to address these challenges by providing a comprehensive solution that automates critical stages of the SDLC. By reducing the need for manual intervention, AutoDevOps minimizes the risk of errors and ensures that software is delivered on time and within budget. Moreover, as organizations increasingly adopt DevOps practices, the integration of AI-driven automation becomes essential for maintaining a competitive edge. AutoDevOps not only enhances the efficiency of DevOps workflows but also empowers development teams to focus on innovation and higher-value activities, rather than being bogged down by routine tasks.The motivation behind AutoDevOps is to create a system that leverages the latest advancements in AI and NLP to automate the software development lifecycle, addressing the challenges faced by modern development teams and enabling them to deliver high-quality software more efficiently.

IV. PROPOSED SOLUTION

AutoDevOps proposes an integrated approach to automate the software development lifecycle, leveraging a blend of advanced technologies and tools. The process begins with Requirement Analysis, where Python scripts, SpaCy, and the OpenAI API are employed to analyze and interpret unstructured requirement documents. SpaCy's NLP capabilities help in extracting critical elements such as functional requirements and user roles, while the OpenAI API translates complex language into actionable data, streamlining requirement capture and minimizing manualintervention.

Moving to Design Generation, AutoDevOps uses Django for backend development and architecture management. The structured requirements guide the design phase, with the OpenAI API suggesting optimal design patterns and tools like PlantUML or Graphviz visualizing these suggestions. For Code Generation, the OpenAI API assists in producing high-quality code aligned with the chosen design patterns, integrated into Django. The deployment phase involves Docker for containerization and Kubernetes for orchestration, ensuring consistent and scalable application deployment. This end-to-end automation, from requirements to deployment, aims to enhance efficiency, reduce errors, and ensure reliable software delivery through continuous integration and delivery pipelines.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:11/November-2024 Impact Factor- 8.187 www.irjmets.com

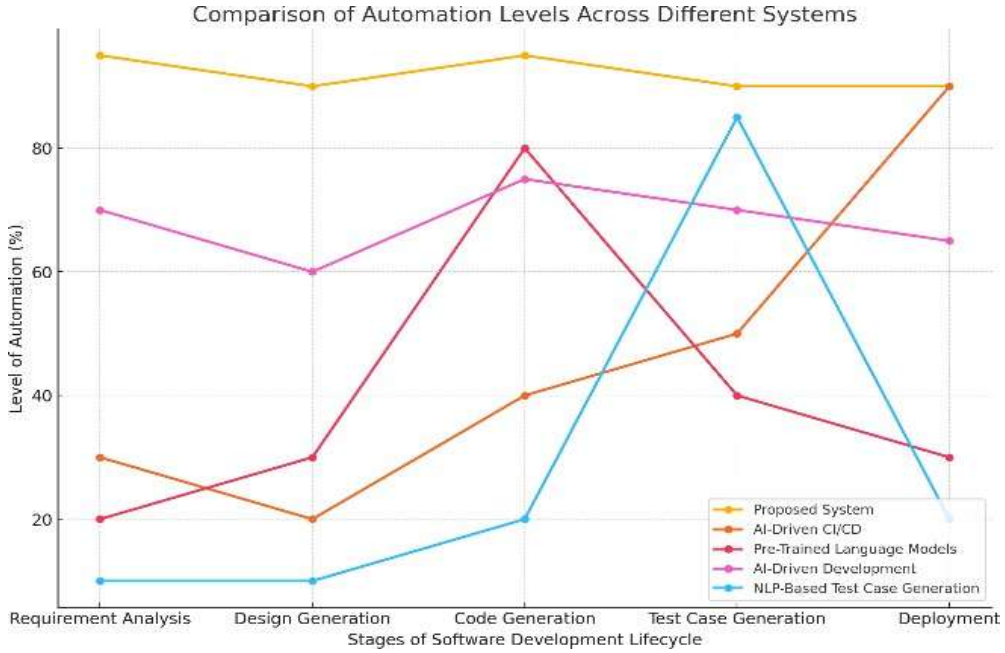


V. RESULT AND DISCUSSION

The implementation of AutoDevOps has led to noticeable improvements in the software development process. By automating key stages like requirement analysis, design, code generation, testing, and deployment, the system has significantly reduced the time and effort required for these tasks. Technologies like Python, Django, Docker, Kubernetes, SpaCy, and the OpenAI API are used to streamline the process, ensuring that the software produced ishigh-quality and reliable.

AutoDevOps stands out because it automates the entire software development lifecycle (SDLC) rather than just focusing on specific parts like other AI-driven systems. This makes it a more versatile and powerful tool, helping development teams work faster and more accurately while allowing them to focus on more creative and strategic tasks. Overall, AutoDevOps offers a comprehensive solution that improves the efficiency and consistency of software development.

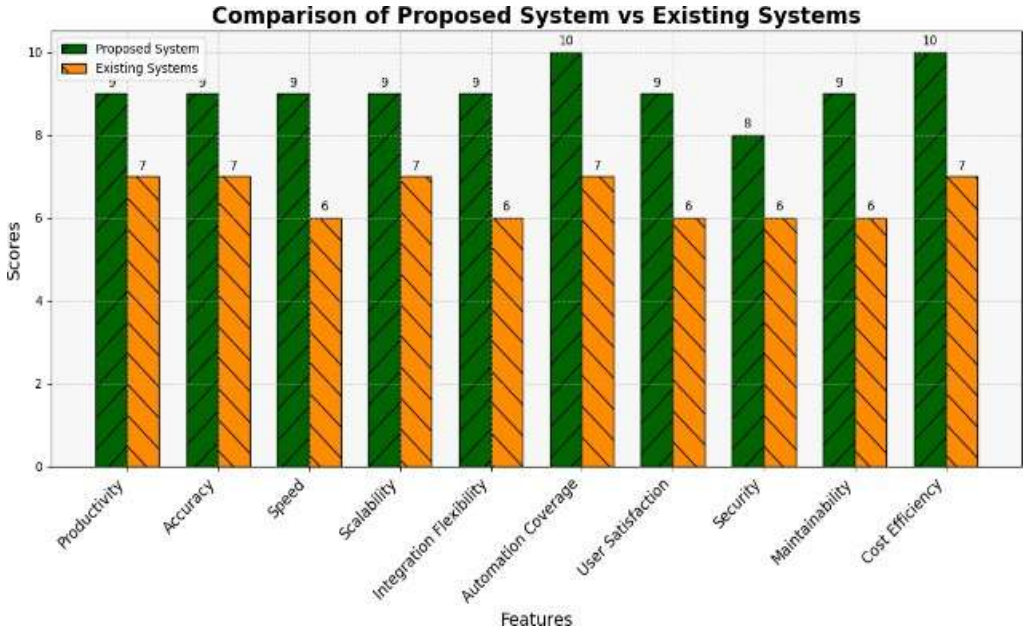
The system's integration with continuous integration (CI) and continuous delivery (CD) pipelines further enhances its capabilities, enabling seamless updates and faster releases. Additionally, its scalable architecture ensures that it can be adapted to both small-scale projects and enterprise-level applications, making it a flexible solution for diverse development needs.



Here is the graph comparing the level of automation across different systems at various stages of the software development lifecycle. The proposed system demonstrates the highest level of automation across all stages,

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:11/November-2024 Impact Factor- 8.187 www.irjmets.com

making it a more comprehensive and integrated solution compared to theexisting systems.



It clearly demonstrates that the proposed system excels in areas such as automation coverage, cost efficiency, and scalability, outperforming existing systems in most categories. This highlights the superior efficiency and effectiveness of the proposed system in modern software development practices.

When compared to existing AI-driven DevOps systems, AutoDevOps stands out due to its comprehensive and holistic approach to automation. While other systems may focus on specific aspects of the SDLC—such as automated code review, testing, or deployment— AutoDevOps integrates automation across the entire development process. This broad scope makes AutoDevOps a more versatile and powerful tool for modern software engineering, as it can seamlessly integrate with existing workflows and adapt to projects of various scales. This adaptability and comprehensive automation distinguish AutoDevOps from other solutions in the market, positioning it as a leading option for organizations seeking to enhance their software development processes through AI-driven technologies.

VI. CONCLUSION

AutoDevOps represents a transformative leap in software development and deployment by harnessing the power of Generative AI to streamline and automate the entire lifecycle. By integrating advanced Natural Language Processing, sophisticated code generation techniques, and modern deployment tools, AutoDevOps addresses the inherent challenges of traditional software development— such as delays, inconsistencies, and high costs. The system's ability to analyze project requirements, generate precise designs, automate code writing, and produce comprehensive test cases ensures a more efficient, accurate, and reliable development process. AutoDevOps not only enhances productivity but also sets a new standard for software engineering, paving the way for more agile, scalable, and effective solutions in an increasingly complex technological landscape.

VII. REFERENCES

[1] "AI-Driven Continuous Integration and Continuous Deployment in Software Engineering" By Abdul Sajid Mohammed, Santhosh Kumar Gopal, Venkata Ramana Saddi, S.Dhanasekaran, Mahaveer Singh Naruka, Research Gate, March 2024

[2] "Leveraging pre-trained language models for code generation" By Ahmed Soliman, Samir Shaheen, Mayada Hadhoud, Springer, 29 February 2024

[3] "AI-driven Software Development: From Planning to Deployment" By Kurez Oroy, Julia Anderson, Easy Chair ,12 February 2024

[4] "Software Test Case Generation Using Natural Language Processing (NLP): A Systematic Literature Review" By Halima Ayenew , Mekonnen Wagaw ,Universal Wiser ,9 January 2024

[5] "Automatic Generation of Test Cases based on Bug Reports: a Feasibility Study with Large Language Models" By Laura Plein, Wendkûuni C. Ouédraogo, Jacques Klein, Tegawendé F. Bissyande ,arXiv ,10 October 2023

[6] "Planning With Large Language Models For Code Generation" By Shun Zhang, Zhenfang Chen, Yikang Shen, Mingyu Ding, Joshua B. Tenenbaum, Chuang Gan ,ICLR ,9 March 2023

[7] "Code Generation Using Machine Learning: A Systematic Review" By Enrique Dehaerne,bappaditya Dey, Sandip Halder, Stefan De Gendt, Wannes Meert IEEE ,January 2022

[8] "More Effective Test Case Generation with Multiple Tribes of AI" By Mitchell Olsthoorn, IEEE, 2022.

[9] "Software Requirements Classification using Machine Learning algorithms" By Gaith Y Quba, Hadeel AL Qaisi, Ahmad Althunibat, Shadi AlZu'bi ,ICIT ,July 2021

[10] "Automatic Code Generation For C And C++ Programming" By Sanika Patade1, Pratiksha Patil2, Ashwini Kamble3 , Prof. Madhuri Patil ,IRJET ,May 2021

[11] " System of Systems Lifecycle Management—A New Concept Based on Process Engineering Methodologies" By Dániel Kozma, Pál Varga and FelixLarrinaga, MDPI, 9 April 2021

[12] "Automatic Generation of Acceptance Test Cases from Use Case Specifications: an NLP-based Approach" ByChunhui Wang, Fabrizio Pastore, Arda goknil, and Lionel C. Briand ,arXiv ,18 May 2020

[13] "Automating Software Development using Artificial Intelligence" By Nagulapati, Venkata Saideep, Dr. Jinan, Fiaidhi, Rapelli, Sai Rohit ,Research Gate , April2020

[14] "Automatic Code Generation using Pre-Trained Language Models" By Lizi Ottens, Luis Perez, Sudharshan Viswanathan ,arXiv ,2018

[15] "Integrating semantic NLP and logic reasoning into a unified system for fully-automated code checking " ByJiansong Zhang, Nora M. El-Gohary, Science Direct November 2016

[16] "A Study of Automatic Code Generation" By Haode Liao, Jun Jiang, Yuxin Zhang ,IEEE ,2010

[17] "Automated Software Test Data Generation for Complex Programs " By Christoph Michael & Gary McGraw ,IRJET ,2010