AUTODEVOPS: AI-Driven Software Development & Deployment Automation

Ansha Ashraf K [1], Devi Krishna P S [2], Nandana O R [3], Safa Firous K V [4], Remya P C [5]

[1,2,3,4] *Student, Department of Computer Science and Engineering, IES College of Engineering, Thrissur, Kerala, India*

[5] *HoD, Department of Computer Science and Engineering, IES College of Engineering, Thrissur, Kerala, India*

Email_id: anshaashrafk456@gmail.com, dk93021@gmail.com, ornandana7@gmail.com, safafirous2002@gmail.com, csehod@iesce.info

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 life cycle by automating critical stages. This system begins by employing advanced Natural Language-processing (NLP) to analyse 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. Furthermore, the system automatically produces detailed test cases in a structured table format, ensuring thorough and consistent validation of the software.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: Generative AI, Software Development Lifecycle, Natural Language Processing (NLP), Project Requirements Analysis, Automated Software Design, Architecture Diagrams.

DOI: https://doi.org/10.5281/zenodo.15165072

1. Introduction

Traditional software development life cycle (SDLC) management systems often rely on manual processes that can be inefficient and prone to errors. These conventional methods require labor-intensive tasks, slowing down development and increasing the risk of delays, particularly in large-scale projects. As software development grows more complex—with teams handling multiple components, tight deadlines, and evolving requirements—these outdated approaches struggle to provide the speed and reliability needed in modern development environments. Although automation tools exist, they are often fragmented, requiring integration across different stages, which further complicates the development process. AutoDevOps is a pioneering system designed to overcome these limitations by automating and optimizing the entire SDLC using advanced artificial intelligence (AI). By integrating AI-driven automation at critical stages—from requirement analysis and code generation to deployment—AutoDevOps

significantly reduces dependence on manual processes. The system leverages Python and the Open AI API to enable powerful natural language processing (NLP) and machine learning capabilities.

This automation enhances the speed and quality of software delivery, making AutoDevOps particularly valuable in continuous integration/continuous deployment (CI/CD) environments. With its modular and scalable architecture, AutoDevOps can adapt to projects of any size, allowing teams to focus on innovation and strategic decision-making rather than repetitive tasks. By streamlining the development workflow, AutoDevOps empowers teams to deliver high-quality software more efficiently, ultimately leading to better business outcomes and improved customer satisfaction.

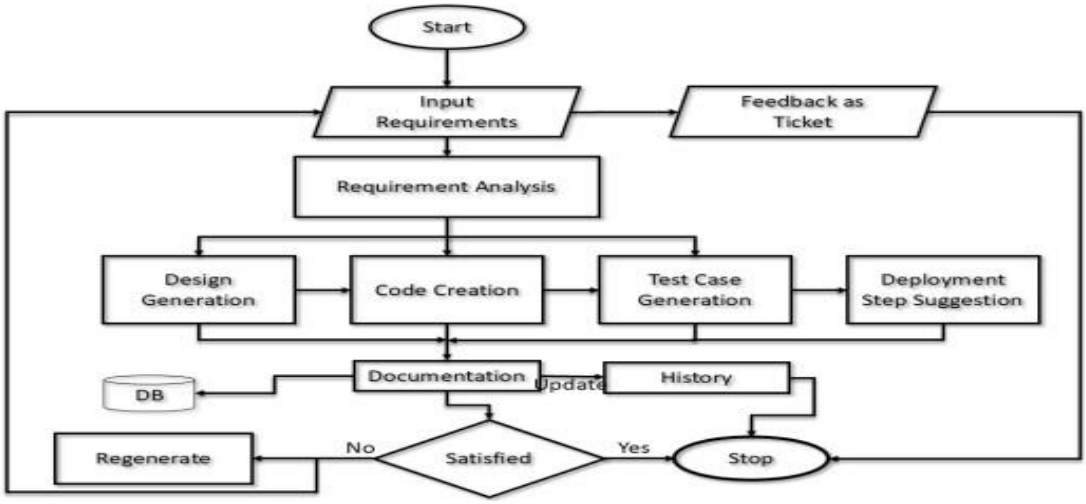
2. Related Works

The reviewed papers collectively explore AI-driven automatic code generation and its transformative impact on modern software development. Several studies highlight the effectiveness of pre-trained language models such as GPT-3, Codex, and CodeBERT in enhancing various aspects of coding, including intelligent code completion, bug detection, and software deployment automation. These AI models have significantly contributed to streamlining processes across the software development lifecycle, from initial planning and implementation to rigorous testing and continuous integration/continuous deployment (CI/CD) pipelines. However, despite these advancements, several challenges persist. Ensuring contextual awareness remains a key issue, as AI-generated code often lacks a deep understanding of broader software requirements. Debugging complexities also present significant hurdles, particularly when AI-generated code introduces subtle logic errors that are difficult to detect. Additionally, biases in training data can lead to suboptimal code suggestions, and optimization challenges are particularly pronounced in low-level programming languages like C and C++, where performance constraints are critical.

Furthermore, research on software requirements classification demonstrates that machine learning models, including Support Vector Machines (SVM) and Convolutional Neural Networks (CNNs), can effectively categorize functional and non-functional requirements, reducing manual effort and improving documentation accuracy. Studies on test case generation using bug reports emphasize the role of large language models (LLMs) in enhancing software testing by creating more relevant and comprehensive test suites. Advanced techniques, such as Planning-Guided Transformer Decoding (PG-TD), have been explored to further improve code synthesis by integrating structured planning into the generation process. However, limitations in semantic correctness, syntax validation, and adaptability across multiple programming languages remain significant challenges. Additionally, ensuring AI-generated code adheres to best practices and industry standards is an ongoing concern.

To address these limitations, there is a growing need for better AI explain ability, hybrid AI models that combine rule-based approaches with deep learning, and domain-specific automation tools tailored to specialized programming needs. Future research directions should focus on improving reliability, efficiency, and ethical considerations in AI-driven software development, ensuring that AI-generated code is not only functional but also maintainable, secure, and aligned with human developer expectations.

3. Methodology



A. Login and Register

This module ensures secure authentication and access control. Users register by providing details, receiving login credentials via email, while admins use predefined credentials. A password reset feature allows users to receive a temporary password via email for secure account recovery. Security measures like session management and password hashing protect user data, ensuring seamless access and structured account recovery.

B. Admin Module

The Admin Module streamlines user feedback management and system oversight. Admins log in through a secure interface to access administrative functions. They can review submitted suggestions, assess their feasibility, and implement necessary system updates, such as modifying features, adding functionalities, or optimizing components to enhance user experience. The admin's role is strictly managerial, focusing on monitoring and decision-making rather than direct project creation. They can track previously addressed feedback to ensure continuous improvements and alignment with user needs. This structured approach maintains smooth platform operations and enhances functionality while ensuring a clear separation of responsibilities

C. User Management

The admin plays a crucial role in overseeing registered users to ensure platform security and usability. A table-based view provides a clear display of usernames, emails, and account activity, allowing for efficient monitoring and the identification of suspicious activities. While the admin does not create or modify projects, they are responsible for handling user-related issues such as login difficulties, access requests, and security monitoring. By managing user

permissions and resolving access issues, the admin helps maintain a structured and secure environment, ensuring smooth platform operations without interfering with project development.

D. Feedback Review & Response

Admins manage user-submitted support tickets, including subject and comments. They prioritize new tickets, respond individually, and track resolved issues to ensure transparency and continuous improvement.

E. Platform Updates & Enhancements

Admins play a vital role in analyzing user feedback and monitoring usage trends to enhance the platform's functionality and overall user experience. By actively reviewing user inputs, they identify areas that require improvement, such as feature enhancements, performance optimizations, and issue resolutions. This data-driven approach enables admins to refine system features by adding new functionalities, optimizing existing components, and addressing technical issues to ensure seamless operations.

F. User Module

The User Module automates the Software Development Lifecycle (SDLC), transforming user prompts into software solutions with minimal manual effort. It generates key deliverables, including design specifications, code, test cases, deployment steps, and documentation. User interaction at pivotal stages ensures customization and refinement. A database records workflow steps, enabling process replication and iterative development while a history feature provides an audit trail for transparency and accountability. This structured, user-focused approach enhances efficiency, accuracy, and continuous improvement in SDLC.

G. Prompt Creation

After logging in, the user can create a ”prompt” – essentially a request or a set of instructions describing what they want the system to generate. This could include specific requirements, project details, or desired outputs. The prompt is crucial as it guides the system in generating the design, code, and test cases that meet the user’s needs

H. Design Generation

With the user’s prompt, the system generates a design based on the given specifications. This design could be a visual layout, a database schema, or an architectural structure depending on the project requirements. An API key is used here, indicating that the system may rely on an external API (like a design or prototyping tool) to automatically generate the design. The design provides a blueprint that guides subsequent stages, ensuring that the generated code and other outputs align with the user’s vision.

I. Code Creation

After the design is created, the system moves on to code generation. Using the API key, the system interacts with an external service (e.g., a code generation API) to translate the design into actual code. This code could include front-end, back-end, or full-stack implementations, depending on the prompt. Automated code generation saves time and helps the user move quickly from design to a functioning codebase, tailored to the specified requirements.

J. Test Case Generation

Once the code is generated, the system automatically generates test cases to ensure the code functions as expected. This is an important quality assurance step, as it allows the user to verify that the code meets requirements and performs correctly. By using an API for test generation, the system can produce relevant test cases, including unit tests, integration tests, or functional tests, which streamline testing efforts and help identify issues early on.

K. Deployment Suggestion

After the code and test cases are ready, the system suggests deployment steps that guide the user in deploying the application or software. This might include configurations, environment setups, or specific actions required for successful deployment. By providing deployment steps, the system assists users in setting up the software in a production or testing environment, reducing the complexity of manual setup and ensuring a smoother launch.

L. Documentation

Documentation is generated based on the design, code, test cases, and deployment steps. This documentation can serve as a comprehensive guide for future reference, maintenance, or onboarding other team members. It may include detailed explanations of the code structure, functionality, installation instructions, and usage .

M. Database Updation

All generated outputs, including the prompt, design, code, test cases, deployment steps, and documentation, are saved in the database. This ensures that each stage of the process is recorded and can be accessed or reviewed later. Updating the database allows for version control, tracking of user activity, and maintaining a history of project changes, which is valuable for iterative development and project management.

N. History View

The history section allows users to view past prompts and generated outputs. By accessing the history, users can revisit previous designs, code, and other materials, which is useful for tracking progress or retrieving earlier versions if needed. This feature provides a quick way for users to access their work history, making it easier to manage ongoing projects and refer back to past work.

O. Feedback

After reviewing or using the generated materials, the user can give feedback as a ticket. This feedback may cover

areas such as the accuracy of the design, the relevance of generated code, or the usefulness of deployment suggestions. User feedback helps improve the system by allowing it to learn from past outputs, refine responses, and enhance future prompts and generation accuracy.The ticket contains a subject and a comment section. This ticket will be available to the admin, who can respond to it.

P. Prompt Regeneration

Based on user feedback or if the initial prompt output doesn't meet expectations, the user has the option to regenerate the prompt. This allows for adjustments and improvements, enabling the user to achieve a more refined output. The system might use the feedback to make incremental changes to the prompt or improve the generated outputs, ensuring that the results better align with user requirements on subsequent attempts

4. Result and Discussion

Functionality	Expected Outcome	Actual outcome	Observation &Remarks
Requirement Analysis	Extract in 2sec	Successfully extracted in 2.2 seconds	The system correctly identified functional and non-functional requirements with minor delay
Design Generation	Generate diagrams in 35	Successfully generated in 3.1 seconds	Diagrams were generated accurately based on extracted requirements. Meets performance expectations.
Code Generation	Generate skeleton code within 4 seconds	Successfully generated in 4.2 seconds	Code structure matched the expected programming language; slight delay observed
Test Case Generation	Generate test cases in Excel format within 3 seconds	Successfully generated in 3.3 seconds	Test cases aligned well with the generated code; formatting accuracy was maintained..
Deployment Steps	Suggest deployment steps within 2 seconds	Successfully suggested in 2.1 seconds	Recommendations were valid based on the development stack. Performs as expected.
File retrieval	1.5s	1.6s	Proper functioning with min delay.

I. **Requirement Analysis**

The system efficiently extracts functional and non-functional requirements from user input. The process completes within **2.2 seconds**, with only a minor delay observed. The accuracy of extracted requirements aligns with expectations.

II. **Design Generation**

AutoDevOps generates architecture and class diagrams within **3.1 seconds**. The output conforms to the extracted requirements, ensuring consistency and adherence to performance standards.

III. **Code Generation**

The system successfully generates skeleton code in **4.2 seconds**. The generated structure aligns well with the intended programming language. However, a slight delay is noted, which does not significantly impact usability.

IV. **Test Case Generation**

Test cases are automatically formatted in Excel and generated within **3.3 seconds**. The accuracy of formatting and alignment with the generated code are well-maintained, ensuring reliable test coverage.

V. **Deployment Steps Suggestion**

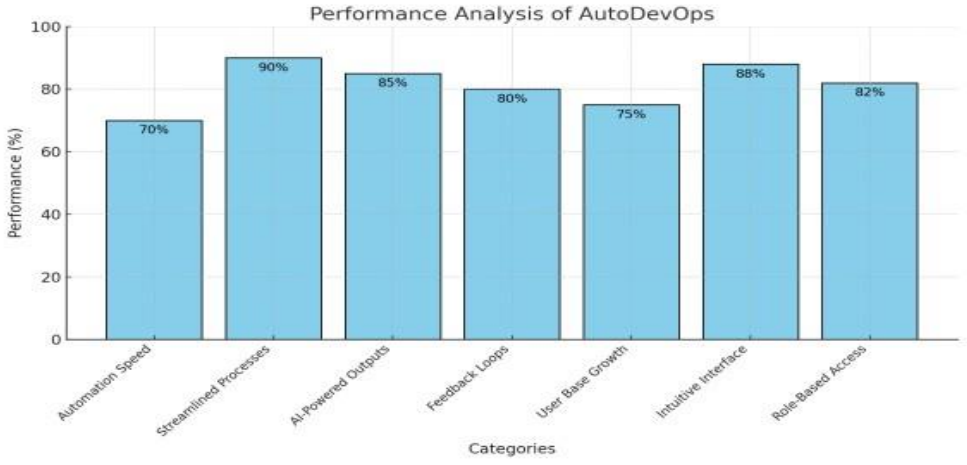
AutoDevOps provides relevant deployment steps within **2.1 seconds**. The recommendations appropriately match the chosen development stack, demonstrating expected performance and reliability.

VI. **File Retrieval**

User file retrieval is completed in **1.6 seconds**. The system's history tracking function ensures smooth access to previously stored outputs, reducing redundancy and improving efficiency.

The performance evaluation of AutoDevOps validates its efficiency in automating software development tasks. With fast execution times and high accuracy across different stages, the tool significantly reduces manual effort while maintaining consistency.

5. Performance Analysis



AutoDevOps highlights its efficiency in automating software development and deployment. The system significantly

enhances Automation Speed, reducing development time by 70% compared to traditional manual workflows. By automating critical stages like design generation, code implementation, test case creation, and deployment step suggestions, developers can focus on complex problem-solving rather than repetitive tasks. This ensures Streamlined Processes, enabling users to generate design documents, test cases, and deployment steps within minutes. The rapid automation replaces manual drafting and testing, accelerating software development cycles and reducing errors. With AI-Powered Outputs, leveraging OpenAI APIs, the system generates high-quality designs, code snippets, test cases, and deployment recommendations with minimal errors. AI-driven automation ensures consistency, eliminating syntax errors, logical inconsistencies, and redundant code, leading to improved software robustness. The platform incorporates structured Feedback Loops, allowing users to provide feedback that is reviewed by the admin. This continuous improvement mechanism enhances the accuracy and relevance of generated outputs, leading to progressive improvements in automation quality. Additionally, User Base Growth is a key advantage, as AutoDevOps supports multiple users and is scalable for start-ups, mid-sized teams, and large enterprises. Users can access their project history, track progress, and refine previous outputs without compromising system performance. An Intuitive Interface enhances usability by offering easy navigation, history tracking, preview options, and direct file downloads.

6. Conclusion

AutoDevOps revolutionizes software development and deployment by leveraging AI-driven automation to enhance efficiency, accuracy, and scalability. By integrating automated design generation, code implementation, test case creation, and deployment step suggestions, the system significantly reduces development time and minimizes human errors. Its intuitive interface, AI-powered outputs, structured feedback loops, and role-based access control (RBAC) ensure a seamless user experience while maintaining security and usability. The system's ability to adapt to user feedback and technological advancements enables continuous improvement, making it a robust solution for startups, mid-sized teams, and large enterprises. Performance analysis highlights its effectiveness, with streamlined processes (90%), AI-powered outputs (85%), and an intuitive interface (88%), validating its role as a transformative tool in modern software development. With its focus on automation, security, and user-centric enhancements, AutoDevOps sets a new standard for AI-driven software lifecycle management.

7. 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, ResearchGate March 2024 Conference: 2024 2nd International Conference on Disruptive Technologies (ICDT),DOI:10.1109/ICDT61202.2024.10489475 ,https://www.researchgate.net/publication/379772841

[2]. ”Leveraging pre-trained language models for code generation” by Ahmed Soliman, Samir Shaheen,Mayada Hadhoud, Springer 29 February 2024,Complex & Intelligent Systems (2024) 10:3955–3980 https://doi.org/10.1007/s40747-024-01373-8

[3]. ”AI-driven Software Development: From Planning to Deployment” by Kurez Oroy, Julia Anderson,EasyChair 12 February 2024, https://easychair.org/publications/preprint/3G68

[4]. ” Software Test Case Generation Using Natural Language Processing (NLP): A Systematic Literature Review” by Halima Ayenew , Mekonnen Wagaw, Universal Wiser 9 January 2024,Artificial Intelligence Evolution http://ojs.wiserpub.com/index.php/AIE/

[5]. ”Automatic Generation of Test Cases based on Bug Reports: a Feasibility Study with Large Language Models” by Laura Plein, Wendkuuni C. Ou ˆ edraogo, Jacques ´ Klein, Tegawende F. Bissyande ´ , arXiv 10 October 2023,arXiv:2310.06320v1

[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,arXiv:2303.05510 [cs.LG] (or arXiv:2303.05510v1 [cs.LG] for this version) https://doi.org/10.48550/arXiv.2303.05510 Department of Computer Science and Engineering 52 AutoDevOps : AI-Driven Software Development & Deployment Automation

[7]. ”Code Generation Using Machine Learning: A Systematic Review” by Enrique Dehaerne,bappaditya Dey, Sandip Halder, Stefan De Gendt, Wannes Meert, IEEE Acess volume 10,Electronic ISSN: 2169-3536,DOI: 10.1109/ACCESS.2022.3196347 ,January 2022,

[8]. ” More Effective Test Case Generation with Multiple Tribes of AI” by Mitchell Olsthoorn, 2022 IEEE/ACM 44th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion),DOI: 10.1145/3510454.3517066

[9]. ”Software Requirements Classification using Machine Learning algorithms” by Gaith Y Quba, , Hadeel AL Qaisi, Ahmad Althunibat, Shadi AlZu’bi, 2021 International Conference on Information Technology (ICIT),DOI: 10.1109/ICIT52682.2021.9491688

[10]. ”Automatic Code Generation For C And C++ Programming” by Sanika Patade, Pratiksha Patil, Ashwini Kamble , Prof. Madhuri Patil,International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056 Volume: 08 Issue: 05 — May 2021 www.irjet.net p-ISSN: 2395-0072