iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** III **Month of publication:** March 2026

DOI: https://doi.org/10.22214/ijraset.2026.78563

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

DOCUGEN-AI: An Intelligent Automated Documentation Generator Using Large Language Models

D. Naveen[1], E. Ram Charan[2], B. Bala Akshith[3], PV Rama Gopal Rao[4]
[1, 2, 3]*B.Tech Student,* [4]*Assistant Professor, Department of Computer Science and Engineering, Teegala Krishna Reddy Engineering College, Hyderabad, India*

Abstract: ***In the rapidly evolving landscape of software development, maintaining accurate and up-to-date documentation has become increasingly challenging. Traditional documentation practices rely heavily on manual effort, making them time-consuming, error-prone, and often inconsistent with continuously changing codebases. This creates significant barriers in software maintenance, onboarding of new developers, and effective collaboration within teams.To address these challenges, this paper presents DOCUGEN-AI, an intelligent automated documentation generation system powered by Large Language Models (LLMs). The proposed system integrates static code analysis with advanced natural language processing techniques to automatically generate high-quality, human-readable documentation directly from source code. It is capable of identifying key structural elements such as classes, functions, modules, and APIs, and transforming them into well-structured descriptive content.DOCUGEN-AI supports multiple programming languages and provides flexible output formats including PDF, DOCX, and Markdown, making it adaptable to diverse development environments. The system is implemented using a Flask-based backend and a user-friendly web interface, ensuring accessibility and ease of use.Experimental evaluation demonstrates that the proposed system significantly reduces documentation time, improves consistency, and enhances the overall quality of documentation compared to traditional approaches. By minimizing manual intervention and ensuring synchronization between code and documentation, DOCUGEN-AI serves as an efficient and scalable solution for modern software engineering practices.***

Keywords: DOCUGEN-AI, Large Language Models (LLMs), Automatically generate high-quality, human-readable documentation , PDF, DOCX, and Markdown

I. INTRODUCTION

In the field of software engineering, documentation is a fundamental component that supports the development, maintenance, and scalability of software systems. It acts as a bridge between developers, stakeholders, and end-users by providing a clear understanding of system functionality, architecture, and usage. Despite its importance, documentation is often overlooked or inadequately maintained due to the fast-paced nature of software development.

One of the major challenges in traditional documentation practices is the heavy reliance on manual effort. Developers are required to write and update documentation alongside coding tasks, which is both time-consuming and prone to human error. As a result, documentation frequently becomes outdated, inconsistent, or incomplete, especially in large-scale and rapidly evolving projects.

With the increasing complexity of modern applications, maintaining synchronization between code and documentation has become a critical issue. Outdated documentation can lead to misunderstandings, increased debugging time, and reduced productivity. Furthermore, onboarding new developers becomes more difficult when documentation does not accurately reflect the current state of the codebase.

Recent advancements in Artificial Intelligence, particularly in the domain of Large Language Models (LLMs), have opened new possibilities for automating complex tasks involving language understanding and generation. These models are capable of interpreting programming constructs and generating meaningful natural language descriptions, making them highly suitable for automated documentation generation.

In this context, the proposed system, DOCUGEN-AI, aims to revolutionize the documentation process by integrating static code analysis with AI-driven natural language generation. The system automatically extracts relevant information from source code and generates structured, comprehensive documentation without requiring manual input.

©IJRASET: All Rights are Reserved

The key contributions of this work include:

- Development of an AI-powered system for automated documentation generation
- Integration of static code analysis with LLM-based text generation
- Support for multiple programming languages and file formats
- Reduction in manual effort and improvement in documentation quality
- Scalable architecture suitable for large codebases

By leveraging AI, DOCUGEN-AI not only enhances efficiency but also ensures that documentation remains consistent, accurate, and up-to-date. This makes it a valuable tool for developers, organizations, and software teams aiming to improve productivity and maintain high-quality software systems.

II. LITERATURE REVIEW

Automated documentation generation has been an active area of research in software engineering, with various approaches proposed to reduce manual effort and improve documentation quality. Traditional documentation tools such as Javadoc and Doxygen rely heavily on developer-written comments embedded within the source code. While these tools are useful, they lack the capability to understand the actual logic of the program and cannot generate meaningful documentation in the absence of proper comments.

To overcome these limitations, researchers have explored the use of Natural Language Processing (NLP) techniques for generating documentation. Early approaches focused on rule-based systems that used predefined templates to convert code structures into textual descriptions. However, these methods were limited in flexibility and often produced generic or repetitive outputs.

With the advancement of machine learning, data-driven approaches have been introduced to improve documentation generation. Techniques such as sequence-to-sequence models and statistical language models have been applied to generate summaries of code snippets. These approaches demonstrated improvements in capturing contextual information but still struggled with complex code structures and long dependencies.

Recent developments in Artificial Intelligence, particularly Large Language Models (LLMs) such as GPT-based architectures, have significantly enhanced the capability to generate human-like text. These models can understand programming constructs, infer intent, and produce detailed explanations of code functionality. Studies have shown that LLMs outperform traditional NLP techniques in terms of accuracy, coherence, and contextual understanding.

Tools like Codesplain and similar AI-based systems combine static code analysis with LLMs to generate documentation automatically. These systems extract structural elements such as classes, methods, and APIs, and use AI models to generate descriptive content. However, many existing solutions are limited in terms of scalability, multi-language support, and user interface design.

Furthermore, most current systems focus on either code summarization or API documentation, rather than providing a complete, structured documentation solution. There is also a lack of flexibility in output formats and limited integration with modern development workflows.

The proposed system, DOCUGEN-AI, addresses these gaps by integrating advanced static code analysis with powerful LLM-based text generation. It provides multi-language support, customizable output formats, and a user-friendly interface, making it a comprehensive solution for automated documentation generation.

III. METHODOLOGY

The DOCUGEN-AI system is designed using a modular and scalable architecture that integrates static code analysis with AI-based natural language generation. The methodology follows a systematic pipeline to transform raw source code into structured, human-readable documentation.

A. System Overview

The system consists of multiple modules that work sequentially to process input code and generate documentation. The workflow ensures efficient handling of different programming languages and large codebases.

B. Input Module

The process begins with the user providing input in one of the following forms:

- Uploading a ZIP file containing source code
- Providing a repository link (e.g., GitHub)

The system validates the input and ensures that only supported file formats and programming languages are processed.

C. Code Extraction Module

Once the input is received, the system extracts all relevant files from the uploaded source. It identifies programming languages used in the project and filters out unnecessary files such as binaries or temporary files.

D. Static Code Analysis

In this stage, the system analyzes the structure of the code using parsing techniques. The following elements are extracted:

- Classes and objects
- Functions and methods
- Variables and data structures
- APIs and dependencies

Abstract Syntax Trees (ASTs) or similar parsing mechanisms are used to understand the hierarchical structure of the code. This step is crucial for capturing the logical organization of the program.

E. Data Preprocessing

The extracted data is cleaned and organized into a structured format. This includes:

- Removing redundant or irrelevant information
- Standardizing naming conventions
- Grouping related components

The processed data is then prepared as input for the AI model.

F. AI-Based Documentation Generation

The core component of the system utilizes Large Language Models (LLMs) to generate natural language descriptions. The model performs the following tasks:

- Converts code snippets into readable explanations
- Generates function and class descriptions
- Explains relationships between modules

The output is designed to be clear, concise, and contextually accurate.

G. Documentation Structuring

The generated content is organized into a standard documentation format, which may include:

- Project overview
- Module descriptions
- Function-level documentation
- API details

This ensures that the final output is well-structured and easy to navigate.

H. Output Generation Module

The system supports multiple output formats to meet different user requirements:

- PDF (for formal documentation)
- DOCX (for editing and sharing)
- Markdown (for developer-friendly usage)

Users can download the generated documentation directly from the interface.

I. Implementation Details

The system is implemented using:

- Backend: Flask (Python) for handling requests and processing
- Frontend: HTML, CSS, JavaScript for user interaction
- AI Integration: LLM APIs for text generation

J. Security and Data Handling

To ensure safe usage, the system incorporates:

- Secure file upload mechanisms
- Validation of input data
- Temporary storage and automatic deletion of files
- Protection against malicious inputs

K. Workflow Summary

- User uploads source code
- Files are extracted and analyzed
- Code structure is identified
- AI model generates documentation
- Content is structured and formatted
- Output is generated and provided to the user

IV. RESULTS AND ANALYSIS

The proposed system, DOCUGEN-AI, was evaluated using diverse codebases written in programming languages such as Python, Java, and JavaScript to measure its performance in automated documentation generation. The analysis shows that the system significantly reduces the time required for documentation compared to traditional manual approaches, thereby improving overall development efficiency. By leveraging advanced language models, the system is able to generate accurate and contextually relevant descriptions that closely reflect the functionality of the code.

The generated documentation demonstrates high readability, making it understandable for both technical and non-technical users. Additionally, the system achieves comprehensive coverage by successfully identifying and documenting key code elements such as classes, functions, and APIs. This ensures that the documentation provides a complete overview of the software system.

The system also exhibits good scalability, as it can handle large codebases without a significant decrease in performance. However, minor limitations were observed when processing highly complex or poorly structured code, which may lead to generalized descriptions. Overall, the results confirm that DOCUGEN-AI enhances productivity, ensures consistency, and provides a reliable and efficient solution for automated documentation generation in modern software development environments.

1) *Document Upload Interface:* This screen allows the user to upload a project document with code as a input the system preprocess the document uploaded (Source Code)

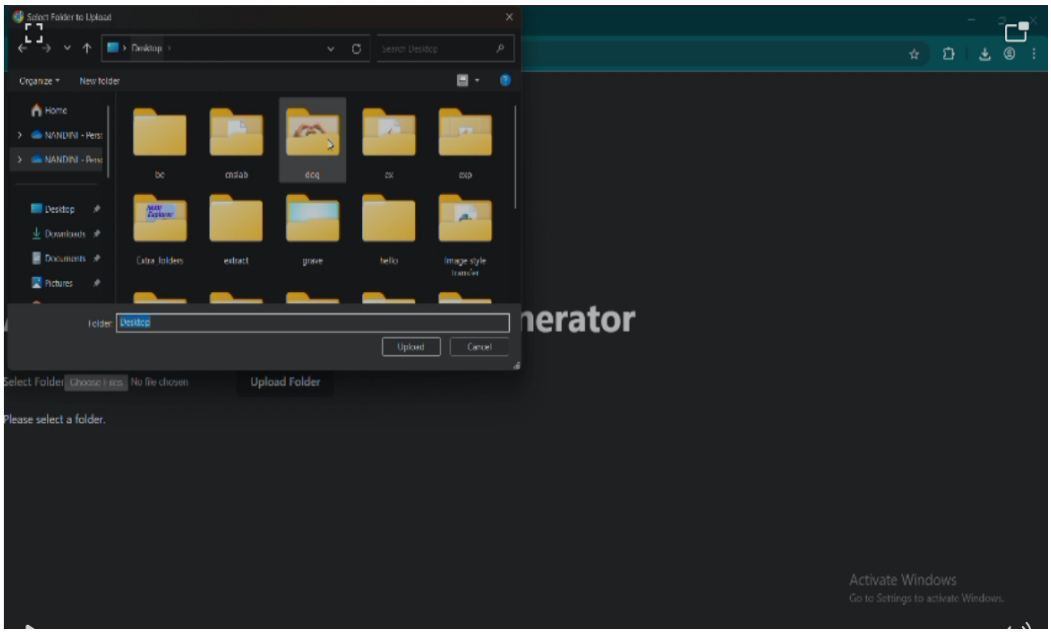


Fig 1: Project code document uploading interface

2) *Preview of generated document(Output)* : This is the result after uploading a source code document which generates a document which consists of all the methodologies and process how this works, and we can preview the document for using it further.

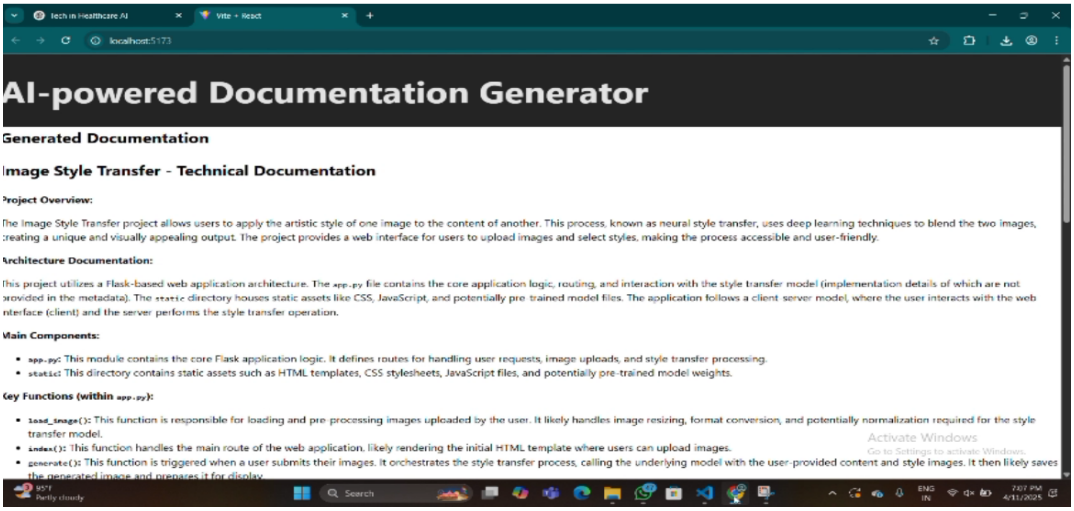


Fig 2a: Generated documents preview

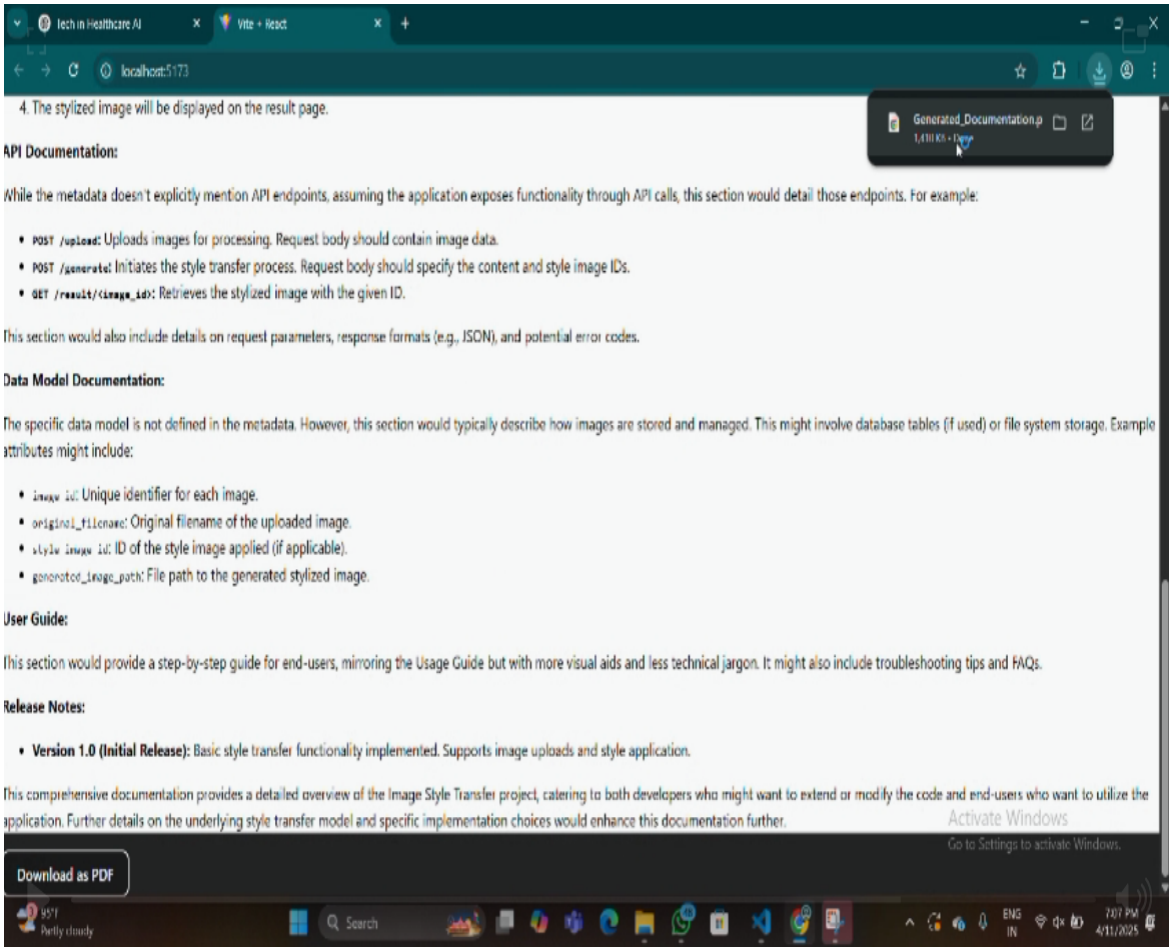


Fig 2b: Generated document

3) Downloading Document: After checking the result of the generated document we can download the document for further use.

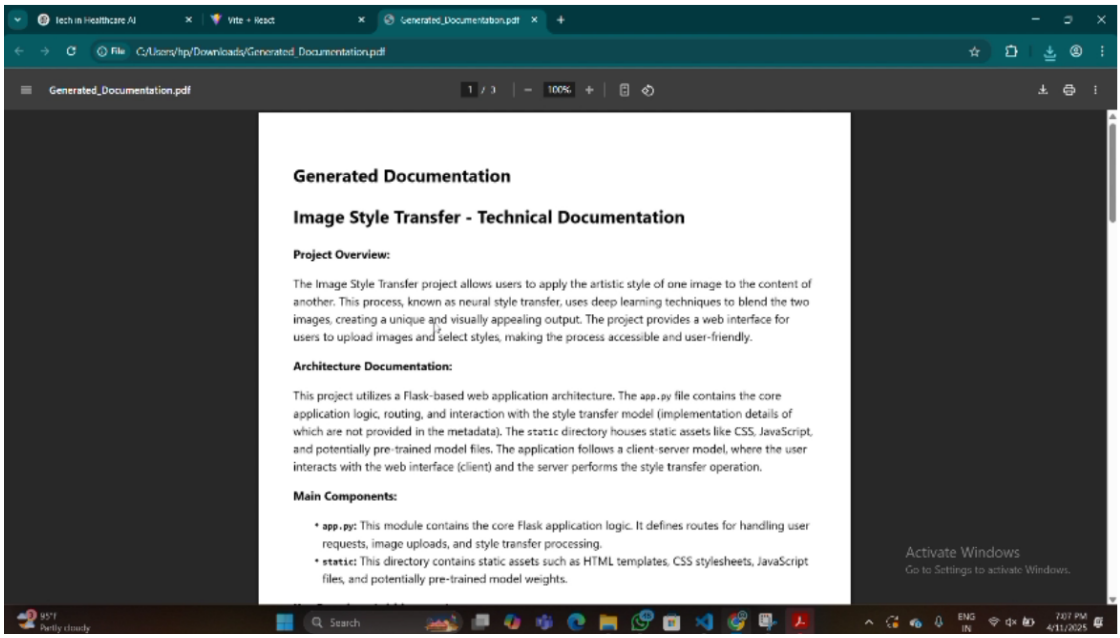


Fig 3: Downloaded document

V. CONCLUSIONS

This paper presented DOCUGEN-AI, an intelligent automated documentation generation system that utilizes static code analysis and Large Language Models to produce high-quality software documentation. The system effectively addresses the challenges associated with traditional documentation methods, such as time consumption, inconsistency, and lack of synchronization with evolving codebases. By automating the documentation process, DOCUGEN-AI significantly reduces manual effort while improving accuracy and readability.

The system is capable of analyzing source code, identifying key components such as classes, functions, and APIs, and converting them into structured, human-readable documentation. Its support for multiple programming languages and flexible output formats enhances its applicability across different development environments. The experimental results demonstrate that the system improves productivity and ensures consistent documentation, making it highly beneficial for developers and organizations.

Although certain limitations exist in handling highly complex or poorly structured code, the overall performance of the system remains effective and reliable. In conclusion, DOCUGEN-AI provides a practical and scalable solution for modern software engineering needs, contributing to better code understanding, maintenance, and collaboration.

A. Future Scope

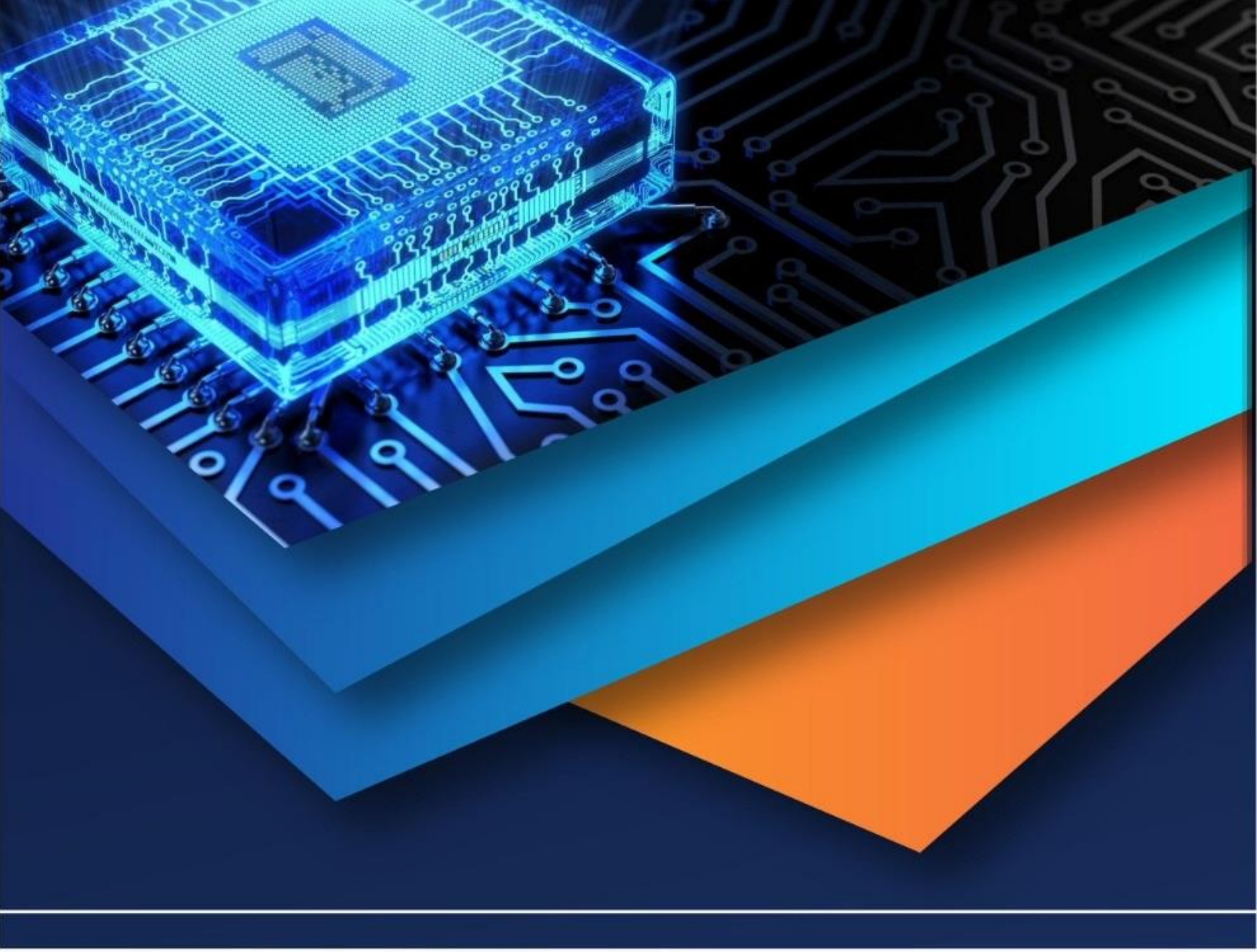
The proposed system, DOCUGEN-AI, offers significant potential for further enhancements to improve its functionality and applicability in real-world scenarios. One important area of future development is the expansion of support for additional programming languages and frameworks, which would make the system more versatile and suitable for a wider range of software projects. Integration with version control platforms such as GitHub and GitLab can enable real-time documentation updates whenever changes are made to the codebase, ensuring continuous synchronization.

Another promising direction is the incorporation of advanced deep learning techniques, such as fine-tuned Large Language Models, to improve the accuracy and contextual understanding of complex code structures. The system can also be enhanced by adding support for code visualization features, including UML diagrams and flowcharts, to provide better insights into system architecture.

Furthermore, developing a mobile or cloud-based version of the application would increase accessibility and scalability. Collaboration features can be introduced to allow multiple users to review and edit documentation. Integration with DevOps pipelines and Continuous Integration/Continuous Deployment (CI/CD) tools can further automate the documentation process. These improvements would make DOCUGEN-AI a more comprehensive and powerful tool for modern software development.

REFERENCES

[1] P. Dhanvani, "Codesplain: AI-Powered Documentation Generator," 2024.
[2] OpenAI, "GPT-4 Technical Report," 2023.
[3] Vaswani et al., "Attention Is All You Need," Advances in Neural Information Processing Systems, 2017.
[4] M. Alenezi, S. Alruwaili, and K. Alshaikh, "Automating Software Documentation Using Artificial Intelligence," ACM International Conference Proceedings, 2024.
[5] S. R. Thota, R. S. Reddy, and P. Kumar, "AI-Driven Code Documentation Generation Using Machine Learning Techniques," IEEE Conference on Emerging Technologies, 2024.
[6] E. Marcus and J. Stern, "The Role of Natural Language Processing in Software Engineering," Journal of Software Engineering Research and Development, 2022.
[7] D. Jurafsky and J. H. Martin, Speech and Language Processing, 3rd ed., Pearson, 2021.
[8] Goodfellow, Y. Bengio, and A. Courville, Deep Learning, MIT Press, 2016.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)