International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

# GENERATIVE AI-BASED TECHNICAL DOCUMENTATION GENERATOR

[1] K Sunanda, [2] M Nikhil, [3] B Sai, [4] R Mukesh
[1]AssistantProfessor, [234]Students
Department of IOT
Siddhartha Institute of Technology & Sciences, Narapally
kondapallisunanda.cse@siddhartha.co.in, 23tq1a6926@siddhartha.co.in, 23tq1a6943@siddhartha.co.in, 23tq1a6955@siddhartha.co.in

## Abstract

Technical documentation is an essential part of software development because it explains the functionality, architecture, configuration, APIs, installation procedures, and usage of software systems. Developers and technical teams often need to create and maintain large amounts of documentation throughout the software development lifecycle. However, preparing documentation manually can be time-consuming, repetitive, and difficult to keep synchronized with frequently changing source code. The Generative AI-Based Technical Documentation Generator is proposed as an intelligent system that automatically generates technical documentation using Generative AI and Natural Language Processing.

The proposed system accepts different software resources such as source-code files, project repositories, API specifications, configuration files, database schemas, and existing documentation. The system analyzes these resources to understand the structure and functionality of the software project. It identifies important components such as classes, functions, modules, APIs, dependencies, parameters, return values, configuration requirements, and relationships between different components.

A Generative AI model is then used to transform the extracted technical information into structured and human-readable documentation. The system can generate documents such as API documentation, module descriptions, installation guides, configuration instructions, code explanations, system overviews, developer guides, and troubleshooting information. Retrieval techniques can be used to provide the AI model with relevant project-specific information and reduce unsupported or inaccurate content.

The generated documentation can be presented through a centralized web interface where developers can review, edit, regenerate, and export the content. The system can also compare documentation with updated source code and identify sections that may require modification. Version-controlled documentation can help teams maintain a consistent history of changes and make it easier to track documentation updates.

## I. Introduction

Technical documentation plays an important role in the development, maintenance, and operation of software systems. It provides developers, testers, administrators, and users with information about how a system works and how it should be configured or used. Documentation may include architecture descriptions, API references, installation instructions, configuration details, code explanations, database

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

information, and troubleshooting procedures. Good documentation can make software easier to understand, maintain, and extend.

Traditionally, technical documentation is created manually by developers, technical writers, and project teams. Developers need to inspect source code, understand system functionality, and convert technical information into structured documents. This process requires considerable time, particularly for large projects containing thousands of files and numerous APIs. Documentation can also become outdated when software changes are made but corresponding documents are not updated.

Natural Language Processing and Machine Learning have created opportunities to automate the extraction and organization of information from software projects. Source-code analysis techniques can identify functions, classes, variables, dependencies, APIs, and other structural elements. These extracted details can be used as input to automated documentation systems and can significantly reduce the amount of manual information gathering required from developers.

Recent advances in Generative AI and Large Language Models have further improved automated content generation. Generative AI can summarize source code, explain complex functions, generate descriptions, answer questions about software repositories, and convert structured information into readable documentation. When combined with retrieval mechanisms, the system can use project-specific source material to generate documentation that is more closely connected to the actual software.

The proposed **Generative AI-Based Technical Documentation Generator** combines source-code analysis, knowledge retrieval, and Generative AI to automatically create technical documents. Users can provide source-code files or connect an authorized project repository, after which the system analyzes the available project information and identifies important technical components. The AI engine then generates documentation according to selected document types and formatting requirements.

The main objective of the system is to reduce documentation effort while improving consistency and maintainability. The generated content can be reviewed and edited by developers or technical writers before publication. This human-review process is important because automatically generated documentation may contain incorrect interpretations or incomplete information. The system therefore acts as an intelligent documentation assistant rather than replacing technical expertise.

## II. Literature Survey

Traditional technical documentation systems depend heavily on manual writing and maintenance. Developers and technical writers inspect source code and project specifications before preparing documents such as API references, user manuals, architecture documents, and installation guides. Although manual documentation can provide detailed and accurate information, the process becomes expensive and time-consuming for large software projects.

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

Documentation-generation tools were introduced to automate portions of the process by extracting information directly from source code. Tools based on structured comments, annotations, and source-code parsing can automatically generate API references and descriptions of program components. These approaches reduce repetitive work but often depend on developers maintaining accurate comments and annotations within the codebase.

Natural Language Processing has been used to summarize source code and extract meaningful information from software repositories. NLP-based techniques can identify functions, entities, relationships, and important technical concepts. Automated summarization can help developers understand large codebases more quickly, although traditional approaches may have difficulty producing detailed documentation that is consistent with project-specific requirements.

Machine-learning approaches have also been investigated for code summarization, documentation generation, and software repository analysis. Learning-based systems can generate descriptions of functions and code snippets by learning relationships between programming constructs and natural language. Their performance depends on the quality and relevance of training data, and generated descriptions may sometimes lack important contextual information.

Large Language Models have significantly expanded the possibilities for automated technical documentation. These models can analyze code, generate explanations, summarize modules, create API descriptions, and answer questions about software projects. Retrieval-Augmented Generation can provide relevant source-code and project information to the model during generation, helping ground generated content in available project data.

Recent intelligent documentation systems increasingly combine code parsing, repository analysis, semantic retrieval, Generative AI, document templates, and automated quality checks. These systems can potentially generate documentation continuously as software evolves. However, accuracy, privacy, outdated information, hallucinated content, and consistency remain important challenges. The proposed system addresses these issues by combining project-specific information retrieval, structured source-code analysis, controlled generation, validation, version tracking, and human review.

## III. System Analysis

The proposed system provides an intelligent platform for automatically generating technical documentation from software project resources. The process begins when a developer uploads source-code files or connects an authorized project repository. A document and source-code processing module analyzes the available files and extracts relevant textual and structural information. The code-analysis module identifies classes, functions, modules, APIs, parameters, return values, dependencies, configuration files, and other software components. A knowledge-indexing module converts the extracted project information into searchable representations for efficient retrieval. The user can select the required documentation type such as API

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

documentation, developer guide, installation guide, module documentation, or system overview. The retrieval module identifies relevant project information required for the selected documentation section. A Generative AI engine uses the retrieved context to generate structured and human-readable documentation. A validation module checks the generated content against available project information and identifies possible inconsistencies. Users can review, edit, regenerate, and approve individual documentation sections. The system can automatically organize the content using predefined templates and headings. Generated documents can be exported into formats such as Markdown, HTML, PDF, or other supported formats. Version tracking can maintain previous documentation versions and highlight changes between generated documents. The system can also identify source-code changes that may require corresponding documentation updates. This approach provides a continuous workflow for creating, reviewing, and maintaining software documentation.

**Existing System**

Existing technical documentation processes generally rely on manually written documents, static documentation tools, source-code comments, and API-generation utilities. Developers and technical writers manually examine software projects to understand functionality and prepare documentation. API-generation tools can automatically extract information from structured comments and annotations but may produce limited documentation when source code lacks sufficient descriptions. Documentation is often maintained separately from source code, creating a risk of outdated information after software modifications. Developers may need to manually update API references, architecture descriptions, configuration guides, and troubleshooting instructions. Large repositories can contain extensive amounts of code that are difficult to analyze manually. Different teams may follow different documentation formats and writing styles, resulting in inconsistent documents. Existing systems generally have limited contextual understanding of complex software behavior. Traditional documentation tools may also require developers to learn specialized syntax or templates. Detecting outdated documentation often requires manual comparison between documents and source code. Documentation generation may therefore become a repetitive activity that receives less attention during rapid development cycles. Existing tools generally automate extraction rather than complete documentation generation and validation. These limitations create a need for an AI-based system that can understand software context and generate structured technical documentation automatically.

**Disadvantages of Existing System**

- Heavy dependence on manual documentation writing.
- Documentation creation can be time-consuming.
- Difficult to document large codebases manually.
- Documentation can become outdated after code changes.
- Requires developers to maintain code comments.
- Limited understanding of source-code context.
- Traditional tools may generate only basic API references.

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

- Inconsistent documentation styles across teams.
- Manual comparison between code and documentation is required.
- Limited automatic documentation quality checking.
- Repetitive documentation tasks consume developer time.
- Difficult to generate multiple documentation formats quickly.
- Limited intelligent summarization of complex modules.
- Manual version management can be difficult.
- Limited automatic detection of documentation gaps.

**Proposed System**

The proposed system introduces a Generative AI-based platform for automatically creating and maintaining technical documentation. Developers can upload source-code files, project repositories, API specifications, configuration files, or other approved project resources. The system analyzes the input using source-code parsing and Natural Language Processing techniques to identify important software components. Extracted information includes modules, classes, functions, APIs, parameters, dependencies, configuration requirements, and relationships between components. The system builds a searchable project knowledge base so that relevant information can be retrieved during documentation generation. Users can select the type of documentation they require, such as API references, developer guides, installation manuals, module descriptions, or system overviews. The Generative AI engine uses retrieved project context to create structured and readable documentation. A validation module checks generated content against available project information and flags potential unsupported or inconsistent statements. Developers can review and modify generated sections before publishing them. The system can regenerate selected sections when the user provides additional instructions or corrections. Documentation can be exported into multiple formats according to project requirements. Version management allows teams to compare documentation revisions and track changes over time. The system can also monitor source-code changes and identify documentation sections that may need updating. Human review remains an important part of the workflow to ensure that generated technical content is accurate and appropriate.

**Advantages of Proposed System**

- Automates technical documentation generation.
- Reduces manual documentation effort.
- Analyzes source code automatically.
- Extracts functions, classes, APIs, and dependencies.
- Uses Generative AI to create readable explanations.
- Supports project-specific documentation.
- Uses retrieval to provide relevant project context.
- Generates multiple types of technical documents.
- Supports customizable documentation templates.
- Helps identify outdated documentation.
- Provides documentation validation and review.
- Supports regeneration and editing of content.

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

- Helps developers understand large codebases faster.

## IV. Methodology

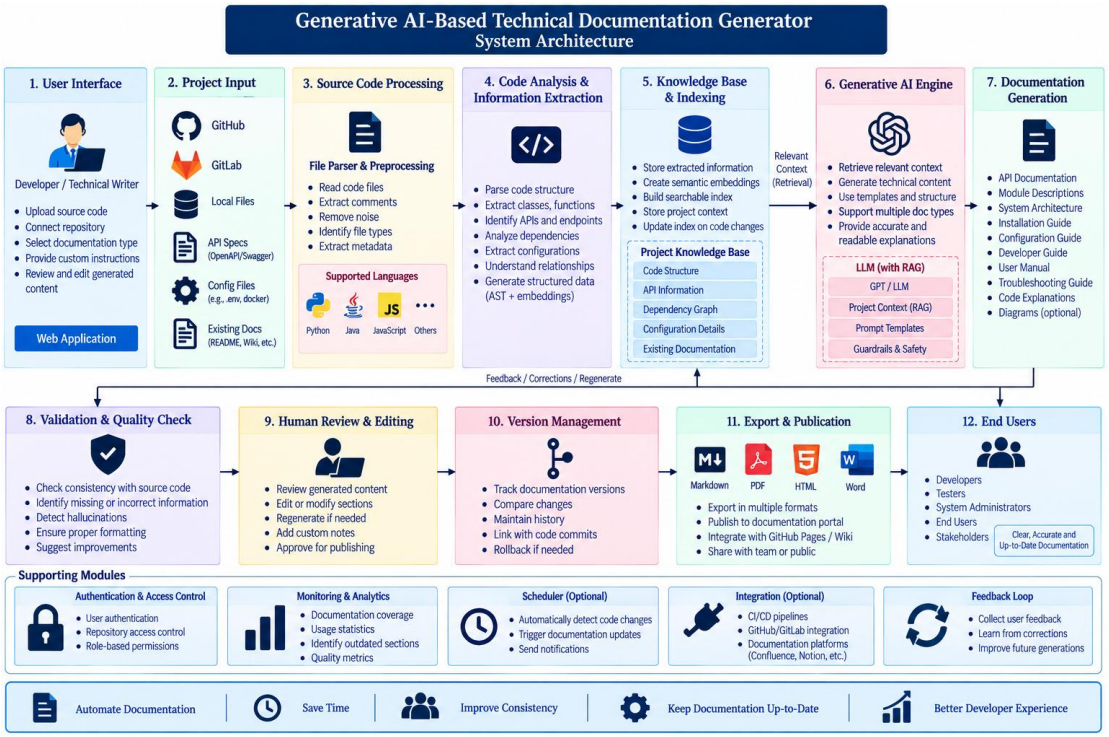
The methodology of the proposed system consists of several stages beginning with project input and ending with validated technical documentation. The first stage is project resource collection, where developers upload source-code files or connect an authorized repository. A preprocessing module identifies supported file types and extracts relevant textual and structural information. The source-code analysis module identifies components such as classes, functions, APIs, dependencies, configuration files, and comments. Natural Language Processing techniques are used to process existing descriptions and project documentation when available. The extracted information is stored in a structured project knowledge repository. A semantic indexing module creates searchable representations that allow relevant project information to be retrieved efficiently. The user selects the required documentation type, sections, language, and output format. The retrieval module gathers relevant code and project context for each requested documentation section. The Generative AI engine generates descriptions, explanations, API references, procedures, and other technical content using the retrieved context. A validation module compares generated information with available project data and identifies possible unsupported statements or inconsistencies. Developers review the generated documentation and can edit or regenerate individual sections. Approved documentation is stored with version information and can be exported in the required format. When source code changes, the system can compare the new project state with previous information and identify documentation sections that may require updates.

### System Architecture

The system architecture consists of several interconnected components including the Developer Interface, Project Upload/Repository Connector, Source-Code Processing Module, Code Analysis Engine, NLP Processing Layer, Knowledge Base, Semantic Retrieval Module, Generative AI Engine, Documentation Template Manager, Validation Module, Version Management System, Export Module, and Documentation Dashboard. The process begins when developers upload project resources or connect an authorized source-code repository through the user interface. The source-processing module reads supported files and extracts source code, comments, configuration information, and existing documentation. The code-analysis engine identifies important program structures such as classes, functions, APIs, dependencies, and modules. The NLP module processes textual information and converts relevant content into structured representations. The knowledge-base layer stores project information and creates searchable indexes for efficient retrieval. When a user requests documentation, the retrieval module finds the most relevant project information for the selected documentation type. The Generative AI engine uses this retrieved context to produce structured technical content according to predefined templates and user instructions. The validation module checks generated documentation for consistency with available project information and flags potentially unsupported content. Developers can review, edit, regenerate, and approve the generated documentation through the dashboard. The version-management module

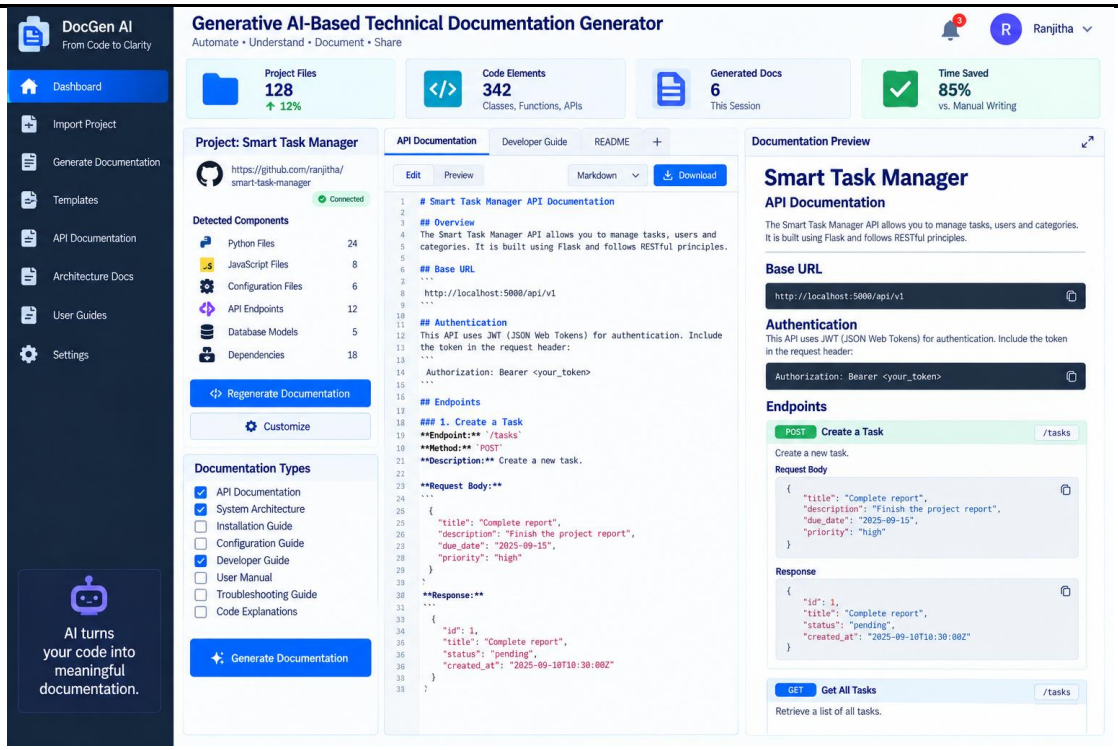
International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

stores previous versions and tracks documentation changes. The export module converts approved content into formats such as Markdown, HTML, PDF, or other supported formats. The complete architecture follows the flow **Developer → Project Upload/Repository → Source-Code Processing → Code Analysis & NLP → Project Knowledge Base → Semantic Retrieval → Generative AI → Documentation Generation → Validation → Human Review → Version Management → Export & Publication**.




## V. Result and Output

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper




## VI. Conclusion

The Generative AI-Based Technical Documentation Generator provides an intelligent and efficient solution for automating the creation of software documentation. By combining Generative AI, Natural Language Processing, source-code analysis, semantic search, and knowledge retrieval, the system can understand software projects and convert technical information into clear and structured documentation.

The proposed system can automatically analyze source code, identify classes, functions, APIs, dependencies, configurations, and modules, and generate documentation such as API references, developer guides, installation guides, architecture descriptions, and troubleshooting guides. This significantly reduces the repetitive effort required from developers and technical writers.

The system also includes validation and human-review mechanisms to improve the reliability of generated documentation. Developers can review, edit, regenerate, and approve content before publication. Version management helps track documentation changes and identify sections that may need updates when the underlying source code changes.

Overall, the project demonstrates how Generative AI can make technical documentation faster, more consistent, and easier to maintain. The system can improve developer productivity and help teams keep documentation aligned with evolving software projects. Future enhancements can include automatic architecture-diagram generation, multilingual documentation, real-time repository monitoring, documentation quality scoring, AI-based documentation testing, and integration with CI/CD pipelines and documentation platforms.

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

## References

[1] Kumar, R. D., Prudhviraj, G., Vijay, K., Kumar, P. S., & Plugmann, P. (2024). Exploring COVID-19 through intensive investigation with supervised machine learning algorithm. In Handbook of Artificial Intelligence and Wearables (pp. 145-158). CRC Press.

[2] Swathi, B., Vijay, K., Sushanth Babu, M., & Dinesh Kumar, R. (2024, November). Machine Learning Techniques in Cloud Based Intrusion Detection. In The International Conference on Artificial Intelligence and Smart Environment (pp. 557-564). Cham: Springer Nature Switzerland.

[3] Sv satyakrishna, shirisha rangu ,bhargavi nalacheruve.(2024) Prospective investigation on colorectal cancer with SMOTE on machine learning Algorithm

[4] Dr.G.Vishnu Murthy, BhargaviNalacheruve 1Professor, Department of computer Science & engineering, Anurag University, TS, India. 2Student, Department of computer Science & engineering, Anurag University, TS, India.

[5] V. N. S. Manaswini, K. K, C. Nigam, S. S. Ali, R. Niranjana, and Suman, "Real-Time Object Detection in Drone Surveillance Using YOLOv5," in Proc. 2025 3rd Int. Conf. IoT, Communication and Automation Technology (ICICAT), Gorakhpur, India, 2025, pp. 1–6, doi: 10.1109/ICICAT68430.2025.11414670.

[6] B. Soundarya, V. N. S. Manaswini, M. Ayyakrishnan, R. D. Kumar, "Contextual Analysis of Big Data Analytics in Intelligent Transportation Frameworks," in Intersection of Artificial Intelligence, Data Science, and Cutting-Edge Technologies: From Concepts to Applications in Smart Environment, Lecture Notes in Networks and Systems, vol. 1353, Cham: Springer, 2025, doi: 10.1007/978-3-031-88304-0_79.

[7] R. D. Kumar, V. N. S.Manaswini, "Applications of blockchain in smart cities: detecting fake documents from land records using blockchain technology," in Blockchain for Smart Cities, Elsevier, 2021, pp. 105–117, doi: 10.1016/B978-0-12-824446-3.00017-X.

[8] Tejavath Veeramma, Badarla Anil, Guguloth Ravinder, "An advanced movie recommender using collaborative filtering and sentiment analysis," International Research Journal of Modernization in Engineering Technology and Science, vol. 7, no. 7, July 2025, doi: 10.56726/IRJMETS81618.

[9] Ravi Kumar Banoth, Ramana Murthy B V, "Automatic crop recommendation system using LightGBM and decision tree machine learning models," Journal of Machine and Computing, vol. 5, no. 1, pp. 343, Jan. 2025, doi: 10.53759/7669/jmc202505026.

[10] Ravi Kumar Banoth, Dr. B.V. Ramana Murthy, "Smart agriculture through IoT and machine learning for analyzing carbon footprints," in Proc. Int. Conf. Computer Science and Communication Engineering (ICCSCE), Apr. 2025.

[11] Ravi Kumar Banoth, B. V. Ramana Murthy, "Soil image classification using transfer learning approach: MobileNetV2 with CNN," SN Computer Science, vol. 5, art. no. 199, 2024, doi: 10.1007/s42979-023-02500-x.