e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:07/Issue:04/April-2025 Impact Factor- 8.187 www.irjmets.com

CODE AND DOCUMENT ANALYSIS USING AI

Asst. Prof. S.S. Lad*[1], Shreya Gholap*[2], Arifa Jahagirdar*[3], Rutuja Awale*[4], Sanika Mohite*[5]

*[1,2,3,4,5]Department Of Computer Science And Engineering, Dr. J. J. Magdum College Of Engineering Jaysingpur, India.

DOI : https://www.doi.org/10.56726/IRJMETS71887

ABSTRACT

Coding AI is an innovative tool designed to automate code documentation and enhance code comprehension. By allowing users to upload their code folders, the tool leverages OpenAI's capabilities to generate detailed documentation, including a project overview and file-wise analysis. This significantly reduces the time spent on knowledge transfer (KT) and improves productivity for developers working on complex projects.

The platform provides an interactive dashboard where users can easily upload their code, view generated documentation, and ask custom prompts for better understanding. Additionally, the tool enables users to download documentation in PDF format for offline use. With a seamless user authentication system and an intuitive interface built using React, Coding AI ensures a smooth experience for developers at all levels.

Powered by Node.js, Express, and MongoDB, the backend efficiently manages user data, code uploads, and documentation storage. The integration of OpenAI allows for accurate analysis, making debugging and learning easier. By streamlining documentation and providing interactive assistance, Coding AI serves as a valuable resource for software teams and individual developers aiming to enhance productivity and efficiency.

I. INTRODUCTION

In software development, understanding and documenting code is a crucial yet time-consuming task. Developers often struggle with maintaining proper documentation, leading to inefficiencies in knowledge transfer (KT) and onboarding. Coding AI addresses this challenge by providing an automated solution that generates detailed documentation for uploaded code, streamlining the process and enhancing productivity.

Coding AI allows users to upload an entire folder containing their code, which is then analyzed using OpenAI's advanced capabilities. The tool generates structured documentation, including a project overview and a detailed breakdown of each file. This ensures that developers can quickly grasp the purpose, functionality, and structure of the codebase without manually reviewing each component.

Beyond documentation, Coding AI offers interactive assistance through custom prompts. Users can ask specific questions about their code, helping them debug issues, understand logic, or optimize performance. This interactive approach reduces dependency on manual code reviews and fosters a more efficient development workflow.

The platform is built with a modern technology stack, including React for the frontend, Node.js and Express for backend operations, and MongoDB for storing user data and generated documentation. OpenAI is integrated to perform code analysis and generate high-quality documentation, making the tool both powerful and user-friendly.

By automating documentation and providing AI-driven assistance, Coding AI revolutionizes how developers interact with their code. Whether for individual developers or software teams, this tool enhances efficiency, minimizes documentation efforts, and ensures seamless knowledge transfer, making it an essential addition to any development environment.

II. METHODOLOGY

1. Requirement Gathering: The first phase involves understanding user needs and ensuring alignment with safety measures. User research is conducted through interviews and surveys with female students to identify common safety concerns. Additionally, campus officials are consulted to ensure the app's functionalities align with institutional security protocols. Based on these findings, essential features such as real-time GPS tracking,

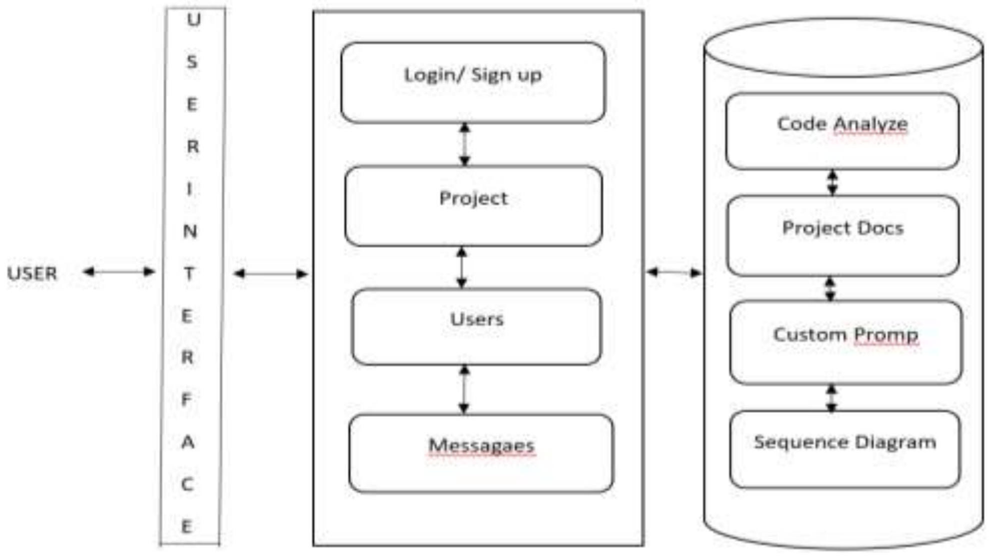
a panic button for emergency alerts, and emergency contact notifications are prioritized to enhance user safety and quick response during emergencies.

2. Design: The application is developed using the MERN (MongoDB, Express, React, and Node.js) technology stack to ensure scalability and efficiency. A well-structured user interface is designed to provide an intuitive experience, allowing users to register, access emergency contacts, and send alerts with minimal effort. The system architecture includes backend APIs for real-time location sharing, emergency notifications, and secure user authentication, ensuring seamless and secure operations.

3. Development: The frontend is built using React, ensuring a responsive and easily navigable interface for quick access to safety features. The backend is implemented with Node.js and Express, utilizing RESTful APIs to handle user authentication, geolocation services, and emergency SMS notifications. Security measures such as data encryption are integrated to safeguard user information during transmission, preventing unauthorized access and ensuring compliance with privacy standards.

4. Testing: A comprehensive testing strategy is employed to ensure the app's reliability and security. Unit and integration testing validate that individual components function correctly and interact smoothly within the system. User testing is conducted with students to gather feedback and improve usability. Additionally, security testing is performed to identify potential vulnerabilities, ensuring that user data remains protected and that the app complies with privacy regulations like GDPR.

III. SYSTEM ARCHITECTURE



1. USER INTERFACE

Represents the interaction layer between the user and the system. Users perform actions like logging in, uploading projects, and analyzing code.

2. MAIN SYSTEM MODULES (Middle Section)

Login / Sign up: Allows users to create an account or log in to access the system. Ensures authentication before accessing project-related features.

Project: Users upload their project folders containing code files. The system processes the uploaded code for further analysis and documentation.

Users: Manages user profiles, session handling, and authentication. Ensures each user has personalized access to their projects.

Messages Stores user interactions, AI-generated responses, or notifications. Helps users track previous queries and receive updates.

3. DATABASE & ANALYSIS MODULES (Right Section - Data Storage & Processing)

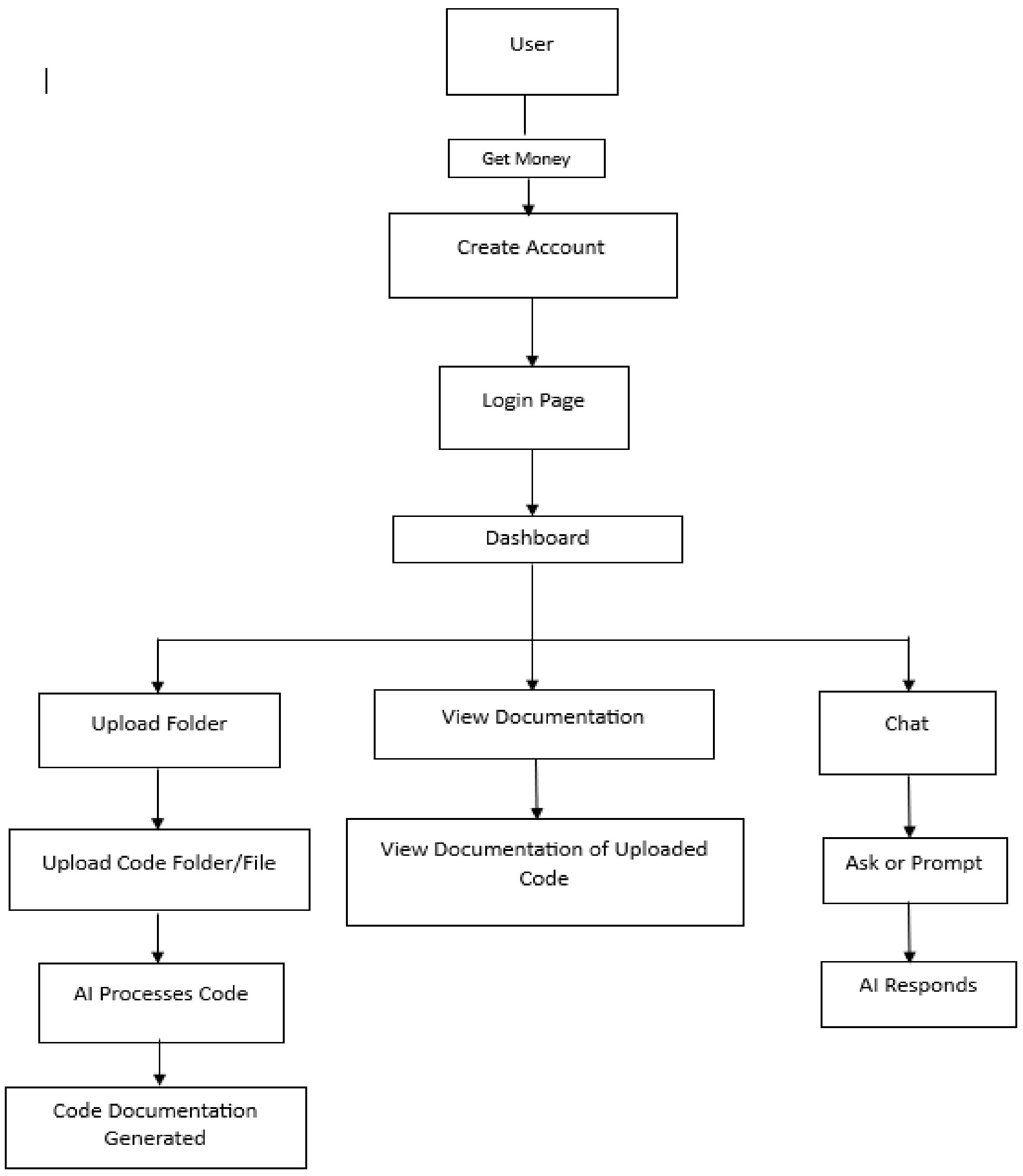
Code Analyze: Processes uploaded code using OpenAI for automated documentation. Provides insights, structure, and explanations for better understanding.

Project Docs: Stores and displays generated documentation. Users can access file-wise project details, explanations, and system overviews.

Custom Prompt: Users can input specific queries about their code. AI responds with relevant answers, helping with debugging and learning.

Sequence Diagram: Represents the interaction between system components. Visualizes the step-by-step process of user actions and system responses

Data flow dig:



1. Login Page – Authenticates users before accessing features; redirects to dashboard.

2. Create Account – New users sign up to create a personal account for secure access.

3. User – Interacts with the system to upload code, view docs, and use AI help.

4. Dashboard – Main user interface to manage uploads, view docs, and use prompts.

5. Upload Folder / File – Users upload code folders or files for AI analysis.

6. AI Processes Code – OpenAI analyzes the uploaded code to understand structure and logic.

7. Code Documentation Generated – System auto-generates readable, structured documentation.

8. View Documentation – Users can read project summaries and detailed file-wise docs.

9. Ask or Prompt – Users ask specific questions about their code for AI assistance.

10. AI Responds – AI replies with explanations, suggestions, or code improvements.

11. Chat – Logs the conversation history between the user and AI for reference.

12. Get Money – Possibly a future feature for monetization or reward (needs clarification).

Techniques to be used:

Frontend (React): The user interface is developed using React to provide a dynamic and responsive experience. React's state management ensures smooth interactions, while React Router enables seamless navigation within the dashboard. This enhances user accessibility for uploading code, viewing documentation, and interacting with custom prompts.

Backend (Node.js & Express): The server-side logic is handled using Node.js and Express to efficiently manage API requests, user authentication, and file processing. This setup ensures secure and fast handling of uploaded code, documentation generation, and communication with the AI model for analysis.

Database (MongoDB): A NoSQL database, MongoDB, is used to store user accounts, uploaded code files, and generated documentation. Its flexible schema allows efficient data retrieval and management, ensuring scalability and smooth performance for handling multiple user requests.

AI Integration (OpenAI): The system integrates OpenAI's capabilities to analyze uploaded code and generate structured documentation. This enables detailed project overviews, file-wise breakdowns, and real-time assistance through custom prompts, helping users understand their code effectively.

File Handling & Processing: The application incorporates file-handling mechanisms to support folder uploads, ensuring proper extraction and processing of code files for analysis. This allows seamless interaction between the user interface and backend services.

Security & Authentication: User authentication is implemented using JWT (JSON Web Token) for secure login and session management. Data encryption techniques are applied to protect sensitive user information and uploaded code files from unauthorized access.

Documentation & PDF Generation: A PDF generation module is incorporated to allow users to download their generated documentation for offline access and sharing. This ensures that users can easily retain structured information about their projects.

IV. FUTURE WORK

Enhanced AI Analysis: Implement AI-driven insights for performance optimization, security vulnerability detection, and best practice recommendations. This will help developers improve code quality alongside documentation.

Multi-Language Support: Expand AI capabilities to analyze and document a wider range of programming languages. This will cater to developers working with diverse coding environments.

Collaborative Features: Enable real-time collaboration where multiple users can view, edit, and interact with documentation simultaneously. Version control integration will help track changes efficiently.

Integration with Developer Tools: Seamlessly integrate with platforms like GitHub, GitLab, and VS Code. This will allow automatic documentation generation within developers' workflows.

Chat-Based Interaction: Introduce AI-powered voice and chatbot-based assistance for easier code debugging. Users can ask natural language queries to understand complex code structures.

Automated Code Refactoring: Implement AI-driven code refactoring to enhance readability and maintainability. It will suggest improvements without altering core functionality.

Mobile App Development: Develop a mobile-friendly version to enable code documentation access on the go. This will improve accessibility for developers outside their workstations.

Security Enhancements: Strengthen data protection with advanced encryption, role-based access control, and AI-powered threat detection. This will ensure secure handling of user data and code.

V. CONCLUSION

Coding AI is a powerful tool designed to automate and streamline the code documentation process while enhancing developers' understanding of t heir projects. By leveraging AI-driven analysis, it simplifies knowledge transfer, reduces manual documentation efforts, and provides detailed insights into code structure. With features like real-time assistance through custom prompts, file-wise analysis, and downloadable documentation, Coding AI significantly improves productivity and efficiency for developers working on complex projects.

Future enhancements will further elevate its capabilities, including multi-language support, AI-powered refactoring, real-time collaboration, and integration with popular development platforms. With ongoing advancements in AI and security, Coding AI aims to become an indispensable tool for developers, ensuring better code management, improved documentation quality, and a seamless development experience.

VI. REFERENCES

[1] Li, S., Jiang, J., & Chen, C. (2021). "Code-to-doc: A Tool for Automatically Generating Code Documentation." This paper discusses an NLP-based approach to generating code documentation, aligning with Coding AI's automation process. Source: ACM Digital Library.

[2] Allamanis, M., Brockschmidt, M., & Khademi, M. (2020). "Machine Learning-Based Code Summarization: A Survey." Covers AI techniques for code summarization, offering insights into Coding AI's documentation capabilities. Source: arXiv.

[3] Russo, A., Harman, M., & Carvalho, A. B. (2019). "Natural Language Processing Meets Software Engineering: A Survey." Explores NLP for code analysis and documentation, aligning with Coding AI's custom prompts feature. Source: Springer.

[4] Zhang, J., Xie, X., & Su, T. (2022). "A Comprehensive Survey on Code Summarization Using Deep Learning." Examines deep learning methods for code understanding, supporting Coding AI's automated documentation. Source: IEEE.

[5] Zafar, H., & Rehman, S. U. (2021). "Automatic Code Documentation Generation Using Machine Learning: A Review." Reviews ML models for automating code documentation, providing a foundation for Coding AI's AI-driven features.