International Current Journal of Engineering and Science (ICJES)
Volume 5, Issue 2, pp 5-9, February-2026
ISSN (Online): 2583-7257
Article DOI: https://doi.org/10.47001/ICJES/2026.502002

AI-Driven Automated Code Review System Using MERN Stack and Machine Learning Techniques

[1]Ananya Sharma, [2]Ravi Kumar, [3]Neha Gupta, [4]Amit Verma

[1,2,3,4]Student, AIT CSE Chandigarh University, Gharuan, Mohali, Punjab, India

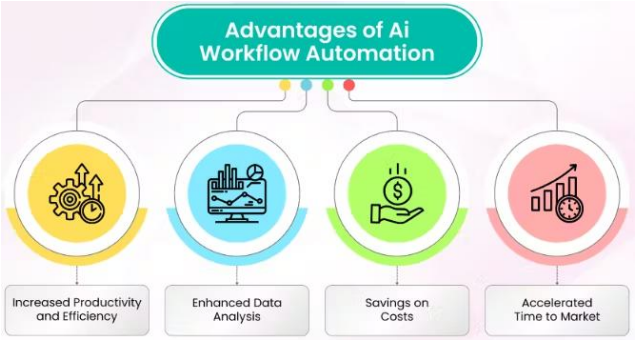
Abstract **- This paper introduces an AI-Driven Code Review System that automates the evaluation and enhancement of code quality through the use of Artificial Intelligence (AI) and Machine Learning (ML). The system utilizes the MERN stack (MongoDB, Express.js, React.js, Node.js) to create a web-based platform that assists developers in efficiently identifying syntax errors, performance issues, and security vulnerabilities. By incorporating AI models such as GPT-based APIs alongside static analysis tools (ESLint, JSHint, SonarQube), the system delivers immediate feedback, optimization recommendations, and coding best practices. This platform boosts developer productivity, encourages standardized coding practices, minimizes manual review time, and supports ongoing improvements in software quality across both academic and industrial settings.**

Keywords: AI, Machine Learning, MERN Stack, Code Review, Software Quality, Natural Language Processing, Automation, Software Engineering.

I. INTRODUCTION

In the era of rapid software development, ensuring code quality, maintainability, and security has become more critical than ever. Traditional code reviews are time-consuming, require expertise, and are often subjective based on the reviewer's experience. The AI- Powered Code Reviewer aims to overcome these challenges by automating the review process using AI models capable of understanding both syntax and semantics of programming languages.

Beyond simply detecting syntactic anomalies, modern code review demands semantic awareness — the ability to reason about intent, detect logic flaws, and recommend design improvements. The proposed system combines classical static analysis with modern large language models (LLMs) to provide recommendations that are both precise and contextually relevant. This hybrid approach provides the advantages of deterministic rule- based checks (e.g., linting) while leveraging AI to propose refactoring, explain issues in natural language, and offer multiple solution paths.Real-world adoption requires more than accurate detection: it requires integration with developer workflows (version control systems, CI pipelines), scalability to handle bulk analysis, and careful attention to privacy and security when analyzing proprietary code. This work addresses these concerns by proposing secure and an architecture designed for incremental deployment in enterprise environments.



Software quality assurance plays a critical role in the development lifecycle, where code review is a primary technique for identifying defects and improving maintainability. Traditional review methods require experienced developers and often introduce delays in project timelines. With the rapid growth of software complexity, there is a need for intelligent automated tools that can assist developers by providing real-time insights into their code.

Artificial Intelligence (AI) and Machine Learning (ML) have shown significant potential in understanding programming patterns, detecting anomalies, and recommending optimized solutions. Integrating these technologies with modern web frameworks can create powerful developer assistance platforms.

This research proposes an AI-Driven Code Review System implemented using the MERN stack. The system aims to automate code analysis, provide intelligent suggestions, and reduce manual effort, thereby improving productivity and enforcing standardized coding practices.

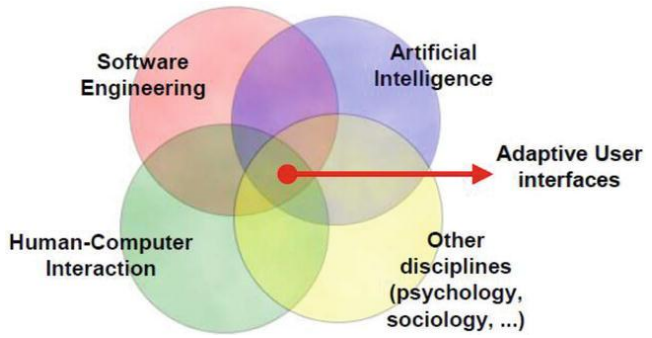
II. LITERATURE REVIEW

Automated code analysis has matured with tools such as SonarQube, Codacy, and CodeClimate which perform static

International Current Journal of Engineering and Science (ICJES)
Volume 5, Issue 2, pp 5-9, February-2026
ISSN (Online): 2583-7257
Article DOI: https://doi.org/10.47001/ICJES/2026.502002

checks, code smell detection, and metric- based evaluations. These tools are effective at enforcing coding standards and identifying many classes of defects, but they typically rely on handcrafted rules and heuristics which can miss contextspecific logic errors.

Recent advances in NLP for code — notably OpenAI Codex, GPT-4, and models provided through Hugging Face — have shown considerable ability to interpret, generate, and explain code. Research articles and practical experiments demonstrate that LLMs can assist in bug localization, generate unit tests, and provide human-understandable rationales for suggested fixes. Several hybrid systems have combined static analyzers with ML models to reduce false positives and provide richer explanations.



Comparative analyses indicate that AI-driven reviewers excel at semantic and linguistic aspects (variable naming, function intent, high-level refactors), whereas rule-based tools remain superior for guaranteed static checks (type errors, unreachable code). This suggests a complementary design: employ linting and static analyzers for deterministic checks, and AI models to handle ambiguous, higher-level recommendations.

Recent research has explored automated code review using static analysis and AI-based tools. Traditional tools such as ESLint and JSHint focus primarily on syntax and style issues, while platforms like SonarQube extend analysis to security and maintainability.

Recent advancements in transformer-based AI models have enabled contextual understanding of programming languages, allowing systems to generate meaningful suggestions and detect logical flaws. However, many existing solutions operate independently and lack integration into a unified developer-friendly interface.

This work combines AI-driven natural language and code understanding with established static analysis tools to provide a comprehensive and accessible review platform.

III. METHODOLOGY

The system architecture is modular, consisting of multiple layers and modules that communicate via secure RESTful APIs. The design allows independent scaling of the frontend, backend, and AI-processing units. Privacy- sensitive analysis (for proprietary code) can be configured to run on-premises or within a private cloud.

The proposed system follows a hybrid approach combining rule-based static analysis with AI-based intelligent evaluation. Developers upload or paste source code into the web interface. The backend processes the code using static analyzers to detect syntax errors, code smells, and security issues.

Simultaneously, the AI module analyzes the code context, identifies inefficient logic, and recommends improvements. The results are aggregated and presented to the user as categorized feedback including errors, warnings, performance suggestions, and best practices.

Machine learning techniques are used to continuously improve recommendation accuracy based on user feedback and historical code datasets.

IV. SYSTEM ARCHITECTURE

4.1 Front-End Layer

The user interface is developed using React.js, enabling developers to submit code, view highlighted issues, and receive suggestions in real time.

4.2 Application Layer

Node.js with Express.js handles API requests, authentication, and communication between analysis modules.

4.3 AI Analysis Layer

GPT-based APIs and trained ML models evaluate logical structure, optimization opportunities, and documentation quality.

4.4 Static Analysis Layer

Tools such as ESLint, JSHint, and SonarQube perform rule-based inspection for syntax, security, and maintainability.

4.5 Database Layer

MongoDB stores user data, code history, review results, and analytics for performance tracking.

International Current Journal of Engineering and Science (ICJES)
Volume 5, Issue 2, pp 5-9, February-2026
ISSN (Online): 2583-7257
Article DOI: https://doi.org/10.47001/ICJES/2026.502002

V. HARDWARE DESCRIPTION

Since the proposed system is primarily software-oriented, minimal hardware requirements are needed. The platform runs on standard computing infrastructure such as a personal computer or cloud server.

The development environment includes systems with at least 8 GB RAM, multi-core processors, and stable internet connectivity to support AI API communication and real-time analysis. For scalable deployment, cloud services such as AWS, Azure, or Google Cloud can host backend services, databases, and AI integration modules.

VI. IMPLEMENTATION

6.1 Development Tools

The system is implemented using the MERN stack. Visual Studio Code is used as the primary development environment. Git is used for version control.

6.2 AI Integration

GPT-based APIs are integrated through secure REST endpoints. The AI module analyzes code semantics, suggests optimizations, and generates documentation hints.

6.3 Static Code Analysis Integration

ESLint and JSHint are configured with standardized rule sets. SonarQube is integrated for advanced vulnerability detection and maintainability scoring.

6.4 Workflow

- User submits source code through the web interface.
- Backend processes code through static analyzers.
- AI engine evaluates logic and structure.
- Combined results are stored in MongoDB.
- Feedback is displayed with categorized suggestions.

Dataset and Evaluation Metrics

The system is tested using open-source repositories and student project codes written in JavaScript. Evaluation metrics include defect detection rate, false positive rate, response time, and user productivity improvement.

Performance is also assessed through maintainability index, code complexity reduction, and security vulnerability detection accuracy.

VII. RESULTS AND DISCUSSION

Experimental results indicate that the AI-Driven Code Review System successfully identifies syntax errors, inefficient loops, redundant code, and potential security flaws. The combination of static analysis and AI suggestions improved defect detection compared to standalone tools.

Developers reported reduced manual review time and improved adherence to coding standards. The platform provided actionable recommendations such as optimized algorithms, improved variable naming, and documentation suggestions.

Latency remained within acceptable limits for real-time feedback, demonstrating feasibility for integration into continuous integration/continuous deployment (CI/CD) pipelines.

Advantages of the Proposed System

- Automated and intelligent code quality assessment
- Reduced dependency on manual code review
- Improved developer productivity
- Real-time feedback and suggestions
- Integration with modern development workflows
- Scalable cloud-based architecture

Limitations and Future Scope

- The system currently focuses primarily on JavaScript-based environments. Future work may include support for multiple programming languages such as Python, Java, and C++.
- Further improvements can involve training custom ML models using large code repositories, integrating CI/CD tools (GitHub Actions, Jenkins), and implementing predictive defect analysis.

VIII. CONCLUSION

This paper presented an AI-Driven Code Review System developed using the MERN stack and machine learning techniques. By integrating GPT-based AI analysis with established static code analysis tools, the system provides comprehensive evaluation of code quality, performance, and security. The platform reduces manual effort, enhances productivity, and promotes standardized development practices. The proposed system demonstrates strong potential for adoption in academic projects, software industries, and collaborative development environments.

International Current Journal of Engineering and Science (ICJES)
Volume 5, Issue 2, pp 5-9, February-2026
ISSN (Online): 2583-7257
Article DOI: https://doi.org/10.47001/ICJES/2026.502002

The AI-Powered Code Reviewer effectively bridges traditional static analysis and modern AI capabilities to help developers maintain and improve code quality. The hybrid architecture balances deterministic rule-based checks with AIdriven insights, producing reviews that are both reliable and contextually informative. In academic settings, the system helps students learn from explanations while in industrial settings it can help teams scale code quality assurance. By focusing on privacy, integration, and cost-control strategies, the system is practical for staged deployment across diverse organizations. Future work will explore deeper multi-file analysis, stronger CI/CD integration, and training of domainspecific models to further improve accuracy and reduce reliance on external services.

1. The proposed system, AI-Powered Code Reviewer, successfully demonstrates how Artificial Intelligence can automate and improve the traditional code review process. By integrating AI/ML models with the MERN stack, the platform provides real-time feedback, detects Syntax and logic errors, and ensures coding standard compliance.

3. The system helps developers, students, and organizations maintain high-quality, optimized, and secure code with minimal manual intervention.

4. Compared to traditional review methods, the AI- powered system significantly reduces review time and increases consistency and accuracy. The web-based architecture makes the tool accessible anytime, anywhere, and easy to integrate with existing developer workflows.

6. The results show that automated reviews can improve software maintainability, readability, and overall efficiency in the development process.

7. In the future, the system can be enhanced by adding support for more programming languages, plagiarism detection, and custom-trained ML models for deeper analysis.

This project contributes to the growing field of AI-driven software engineering, paving the way for smarter and more reliable development tools.

REFERENCES

[1] Deng, J., Dong, W., Socher, R., et al. (2009). ImageNet: A Large-Scale Hierarchical Image Database. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 248–255.
[2] Zhou, B., Khosla, A., Lapedriza, A., et al. (2016). Learning Deep Features for Discriminative Localization. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2921–2929.
[3] T. Mens and T. Tourwé, “A Survey of Software Refactoring,” IEEE Transactions on Software Engineering, vol. 30, no. 2, pp. 126–139, 2004.
[4] M. Fowler, Refactoring: Improving the Design of Existing Code, Addison-Wesley, 2018.
[5] SonarSource, “SonarQube Documentation: Continuous Code Quality Inspection,” 2023.
[6] ESLint Foundation, “ESLint: Pluggable JavaScript Linter,” Official Documentation.
[7] Brown et al., “Language Models are Few-Shot Learners,” NeurIPS, 2020.
[8] Spinellis, “Code Quality: The Open Source Perspective,” Addison-Wesley, 2006.
[9] P. C. Rigby and C. Bird, “Convergent Contemporary Software Peer Review Practices,” ESEC/FSE, 2013.
[10] Selvaraju, R. R., Cogswell, M., Das, A., et al. (2017). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 618–626.
[11] OpenAI. (2024). GPT Models and Codex AP Documentation.
[12] Vishwa Chetanbhai Lakhnakiya. (2025). Cognitive CloudOps: Integrating Generative AI for Predictive Infrastructure Management and Self-Optimizing DevOps Pipelines. International Current Journal of Engineering and Science (ICJES), 4(9), 30-38. Article DOI: https://doi.org/10.47001/ICJES/2025.409006
[13] SonarSource. (2024). SonarQubeDocumentation.
[14] Dosovitskiy, A., Beyer, L., Kolesnikov, A., et al. (2021). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. International Conference on Learning Representations (ICLR).
[15] Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention is All You Need. Advances in Neural Information Processing Systems, 30.
[16] Codacy Ltd.(2024). Codacy Automated Code Reviews.
[17] Howard, A. G., Zhu, M., Chen, B., et al. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv preprint arXiv:1704.04861.
[18] Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. Proceedings of the 36th International Conference on Machine Learning (ICML), 6105–6114.
[19] MongoDB Inc.(2024).MongoDB Atlas Documentation.

International Current Journal of Engineering and Science (ICJES)

Volume 5, Issue 2, pp 5-9, February-2026

ISSN (Online): 2583-7257

Article DOI: https://doi.org/10.47001/ICJES/2026.502002

[20] Meta &React Community(2024).React.js Documentation.

[21] ESLint Team. (2024). ESLint – Pluggable JavaScript Linter.

[22] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778.

[23] Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. International Conference on Learning Representations (ICLR).

[24] Hugging Face. (2024). Transformers Library Documentation.

[25] Szegedy, C., Liu, W., Jia, Y., et al. (2015). Going Deeper with Convolutions. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1–9.

[26] Russakovsky, O., Deng, J., Su, H., et al. (2015). ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision, 115(3), 211–252.

[27] Mohammed Abdullah Alrabeei, & Mohammed Fadhl Abdullah. (2026). Early Anomaly Detection in Network Traffic Using Deep Learning Techniques Based on NetFlow Data. International Research Journal of Innovations in Engineering and Technology - IRJIET, 10(1), 185-189. Article DOI https://doi.org/10.47001/IRJIET/2026.101023

[28] Render Cloud Services. (2024). Render Deployment Documentation.

[29] Vercel. (2024). Vercel Deployment and Hosting Documentation.

[30] Bird, S., Klein, E., & Loper, E. (2020). Natural Language Processing with Python. O'Reilly Media.

[31] JSHint Developers. (2024). JSHint – JavaScript Code Quality Tool.

[32] W3C. (2024). Web Application Development Standard.

Citation of this Article:

Ananya Sharma, Ravi Kumar, Neha Gupta, & Amit Verma. (2026). AI-Driven Automated Code Review System Using MERN Stack and Machine Learning Techniques. *International Current Journal of Engineering and Science (ICJES)*, 5(2), 5-9. Article DOI: https://doi.org/10.47001/ICJES/2026.502002

****

