Journal of Science, Computing and Engineering Research (JSCER)
Volume-9, Issue-4, April 2026.
DOI: *https://doi.org/10.46379/jscer.2024.0709*

AI – POWERED CODE REVIEWER

[1] Mr. Arvind Sharma [2]Atul Pratap Singh [3]Chandrashekar Sharma [4]Mohit Yadav [5]Ajay Kumar Saini

[1]Assistant Professor, Department of CSE, Modern Institute of Technology and Research Centre, Rajasthan, India.

[2,3,4,5]UG Student, Department of CSE, Modern Institute of Technology and Research Centre, Rajasthan, India

Article Information

Received : 29 March 2026
Revised : 30 March 2026
Accepted : 31 March 2026
Published : 04 April 2026

Corresponding Author:

Atul Pratap Singh

Abstract— The rapid growth of software development has led to an overwhelming amount of code being produced, making manual code review a time-consuming and error-prone process. Traditional review methods may overlook critical bugs, security vulnerabilities, and inefficiencies, affecting overall software quality. To address this challenge, this project introduces a web-based, AI-powered Code Reviewer system. This application leverages advanced Artificial Intelligence techniques to automatically analyze and evaluate source code for quality, performance, and adherence to coding standards. By examining code structure and logic, the system provides quantitative assessments, highlights potential errors, and identifies areas for improvement. The primary objective is to streamline the development workflow by offering a fast, consistent, and efficient tool for developers to enhance code quality, reduce debugging time, and ensure better reliability, thereby improving productivity and overall software performance.

Keywords: **Artificial Intelligence, Automated Code Review, Static Analysis, Machine Learning, MERN Stack, Code Quality Metrics, Software Engineering, Bug Prediction, Code Optimization, Developer Productivity**

Citation: Mr. Arvind Sharma, Atul Pratap Singh, Chandrashekhar Sharma, Mohit Yadav, Ajay Kumar Saini. "AI – POWERED CODE REVIEWER", Journal of Science, Computing and Engineering Research, 9(4),April 2026.

I.INTRODUCTION

The AI-powered Code Reviewer project represents a sophisticated intersection of Artificial Intelligence (AI) and Software Engineering Technology. In the contemporary software development landscape, the sheer volume of code being produced—often across multiple projects and collaborative environments—has necessitated the transition from manual code review to automated intelligent analysis systems. However, this transition has created a "quality gap" where developers are often unaware of how their code is being evaluated in terms of efficiency, standards, and potential errors by automated tools.

Our project is an innovative, targeted solution explicitly designed to address the challenges of inefficient and inconsistent code review processes. The core capability of this system lies in leveraging advanced Artificial Intelligence techniques and machine learning approaches to simulate the behavior of modern automated code analysis tools. By acting deployment check" for developers, the tool provides a precise simulation of how contemporary systems analyze, evaluate, and optimize source code, ensuring improved quality, reliability, and adherence to best coding practices.

A. The Core Mechanism

The Code Reviewer functions by treating source code not merely as executable instructions, but as a structured dataset for intelligent analysis. It systematically replicates the evaluation logic used by modern code quality tools and development platforms. The user provides essential input in the form of source code written in supported programming languages. The system then executes a multi-stage processing pipeline:

• **Ingestion:** Extracting raw code input from various file formats and development environments.
• **Normalization:** Standardizing the code by removing unnecessary elements such as redundant spaces, comments (if required), and formatting inconsistencies.

• **Analysis:** Converting the code into structured representations to evaluate syntax, logic, and coding patterns for efficient comparison and validation.

II. PROPOSED METHOD

The proposed system is designed to provide accurate, intelligent, and context-aware code analysis by integrating automated code review techniques with Artificial Intelligence models. It utilizes user-submitted source code as the primary input and generates feedback, error detection, and optimization suggestions based on semantic and structural understanding of the code.

A.Code Ingestion and Preprocessing

The process begins with users uploading or entering source code in supported programming languages. The system extracts and processes the input code using backend services built on Node.js. The extracted code is cleaned to remove unnecessary formatting inconsistencies, redundant spaces, and irrelevant elements. The cleaned code is then segmented into logical components to ensure efficient analysis and improved accuracy during processing.

B. Code Representation and Analysis

Each segment of code is converted into structured representations using parsing techniques and intermediate models. These representations capture syntax, semantics, and logical flow rather than relying only on basic rules. This enables the system to understand relationships between different parts of the code and perform deeper analysis for error detection and optimization.

C. Intelligent Evaluation and Retrieval

The processed code is analyzed using trained machine learning models and rule-based systems. When a user submits code, the system evaluates it against predefined quality metrics and best practices. It identifies patterns, detects potential bugs, and retrieves relevant suggestions to improve performance and maintainability. This ensures that feedback is accurate and aligned with modern coding standards.

D. AI Model Integration

The analyzed code and extracted features are passed to an AI model that generates context-aware feedback through an intelligent processing pipeline. This approach ensures that suggestions are meaningful, reduces incorrect outputs, and improves reliability. The system supports integration with modern AI models to enhance adaptability and performance across different coding scenarios.

E. Automated Issue Detection and Optimization

A key feature of the system is its ability to automatically detect errors, inefficiencies, and non-optimized code structures. It evaluates logic, identifies redundant operations, and highlights areas that require improvement. Detected issues are presented with explanations and confidence levels, enabling developers to make informed decisions and refine their code effectively.

F. Dashboard Visualization and Output

All processed results, including detected issues, suggestions, and quality metrics, are presented through a web-based interface built using React. The dashboard provides an overview of code performance, identified errors, and recommended improvements. Visualization tools are used to represent code quality metrics, enabling users to understand analysis results in an intuitive and interactive manner.

G. System Architecture and Deployment

The frontend is developed using React.js for a responsive user interface, while Node.js and Express.js handle backend processing and API communication. MongoDB is used to store code data, analysis results, and user history. The system follows a modular architecture and can be deployed using containerization technologies for improved scalability, maintainability, and future integration in large-scale development environments.

Component Diagram

The Component Diagram illustrates the structural relationship among the software components.

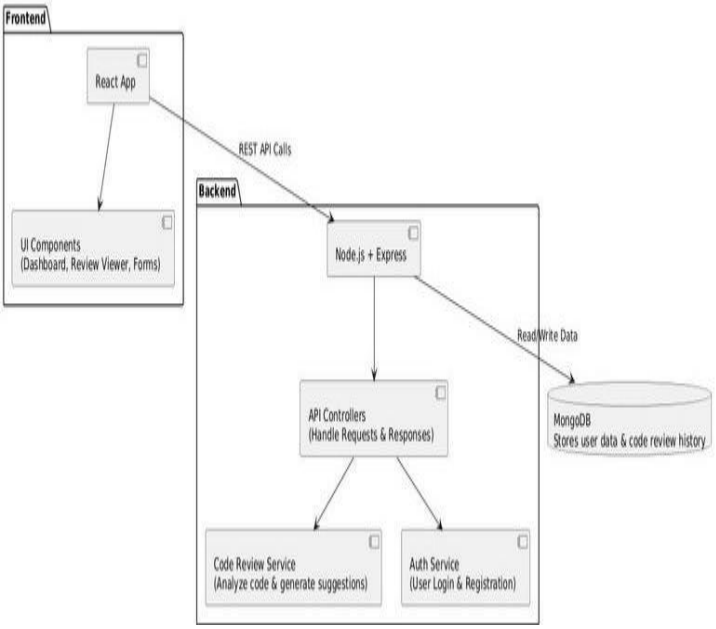

Fig. Component Diagram

III. TECH STACK

The selection of a technology stack is a critical architectural decision that determines the scalability, maintainability, and performance of the system. For the AI-powered Code Reviewer, we adopted a modern MERN-based stack that balances the flexibility of web technologies with the computational capabilities of Artificial Intelligence models.

A. Frontend Technologies

We utilized React.js to build a dynamic and responsive user interface. This layer efficiently manages asynchronous operations, ensuring that the interface remains interactive while intensive code analysis is processed in the background. The component-based architecture of React enhances reusability and improves overall performance.

B. Backend Technologies

Node.js with Express.js was chosen for the backend API layer. Its non-blocking, event-driven architecture is well-suited for handling multiple requests efficiently. It allows seamless integration with AI services and code analysis modules without the overhead of heavier backend frameworks, ensuring fast and scalable performance.

C. Database Technology

Given that source code and analysis results are semi-structured and can vary significantly, a NoSQL database like MongoDB is used. It provides a flexible schema to store code snippets, user data, and review outputs without requiring rigid table structures, enabling efficient data management and scalability.

D. Code Parsing and Processing

Reliable code extraction and parsing are essential for accurate analysis. The system uses parsing techniques and libraries capable of handling multiple programming languages, converting raw code into structured formats such as Abstract Syntax Trees (ASTs). This enables deeper understanding of syntax, structure, and logical flow within the code.

E. AI and Analysis Technologies

Artificial Intelligence and machine learning models form the core of the system. These models are used to analyze code patterns, detect bugs, and suggest optimizations. Rule-based static analysis is combined with intelligent models to ensure both accuracy and adaptability, enabling the system to deliver high-quality, context-aware feedback for improving code quality.

IV. RESULT SCREENSHOTS

The developed AI-powered Code Reviewer successfully demonstrates the practical application of Artificial Intelligence in automating software code analysis. The system is capable of providing intelligent, context-aware, and actionable feedback to developers by combining static code analysis techniques with machine learning models for bug detection and optimization suggestions. Overall, the results show that the proposed system can effectively function as an AI-driven code quality assurance platform, improving code reliability, maintainability, and development efficiency.

In addition, the system reduces the time required for manual code reviews, allowing developers to focus on higher-level design and feature implementation. It also enforces consistent coding standards across teams, minimizing discrepancies in code style and structure. The dashboard interface provides clear visualizations of detected issues and recommended improvements, enhancing usability and adoption. These outcomes collectively demonstrate the system's potential to accelerate development workflows and improve overall software quality.

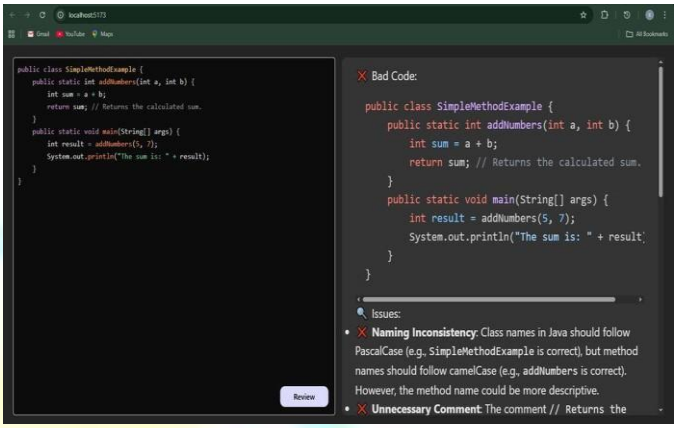

Fig: Code Error Screen

Fig: Code Review Screen

V. CONCLUSION

The proposed AI-powered Code Reviewer demonstrates how modern Artificial Intelligence technologies, including machine learning models, static code analysis, structured code representations, and context-aware evaluation, can be effectively combined to automate and enhance the software development workflow. By merging code ingestion,

preprocessing, parsing, intelligent evaluation, bug detection, optimization suggestions, and dashboard visualization into a unified pipeline, the system provides developers with a powerful tool that significantly reduces manual review effort. The use of AI-driven analysis ensures feedback is accurate, actionable, and aligned with coding standards, improving maintainability and reducing errors. The automated issue detection feature adds unique value by identifying logical inefficiencies, non-optimized structures, and potential bugs, supporting developers in producing higher-quality software. This closed-loop design makes the system a dynamic code quality tool rather than a simple static checker.

Overall, the model provides a scalable and practical framework for intelligent code review. It can be extended to integrate with version control platforms such as GitHub and GitLab, support multi-language codebases, incorporate advanced LLM-based analysis, and be deployed at enterprise scale. The structure lays a strong foundation for future improvements in AI-driven development tools and illustrates the transformative potential of intelligent systems in improving software reliability, productivity, and maintainability.

REFERENCES

1. Sharma, A., Kumar, R., Gupta, N., & Varma, A. (2023). AI-Driven Automated Code Review System Using MERN Stack and Machine Learning Techniques. Chandigarh University. – Introduces a MERN-based AI code review platform integrating GPT-based APIs with static analysis tools for real-time feedback.

2. Hisham, M. (2024). AI-Powered Code Reviewer: Complete MERN Stack Project. GitHub Repository. – Demonstrates a practical implementation of an AI-powered reviewer using MongoDB, Express.js, React.js, and Node.js for instant feedback.

3. Kumar, A. (2025). AI Code Reviewer: MERN Stack Project with Gemini-2.5-Pro. LinkedIn. – Explores integration of advanced AI models (Gemini-2.5-Pro) with MERN stack for collaborative, real-time code analysis.

4. Good fellow, I., Bingo, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. – Foundational text on deep learning, widely used for understanding AI models applied in code analysis.

5. Cholet, F. (2021). Deep Learning with Python (2nd Edition). Manning Publications. – Practical guide to implementing deep learning models, relevant for AI integration in automated reviewers.

6. Fowler, M. (2018*).* Refactoring: Improving the Design of Existing Code (2nd Edition). Addison-Wesley. – Essential reference for understanding code quality improvement and refactoring practices.

7. Martinez, A., & Campos, J. (2021). MongoDB Applied Design Patterns. Packt Publishing. – Provides insights into efficient database design, crucial for storing submissions and review histories.

8. Green, A. (2018*).* Learning React: A Hands-On Guide to Building Web Applications. Addison-Wesley. – Covers modern React.js practices for building interactive frontends in MERN-based systems.

9. Richardson, C., & Smith, F. (2019). Micro services Patterns. Manning Publications. – Useful for designing scalable backend architectures in Node.js and Express.js.