Intelligent Code Analysis and Error Detection Platform

Pritam Sangole[1], Mukesh Nagrikar[2], Ayush Bhandarkar[3], Rushikesh Dubey[4], TusharDeshmukh[5]
Omkar Dudhbure[6]
[1,2,3,4,5]*Student, Manoharbhai Patel Institute of Engineering and Technology, Bhandara*
[6]*Assistant Professor, Manoharbhai Patel Institute of Engineering and Technology, Bhandara*

***Abstract*—This study introduces an AI-powered framework for automatic code review and bug detection using the MERN stack integrated with Groq AI API. The proposed system is trained and tested on real-world code samples collected from various sources including GitHub repositories and custom code submissions. To ensure comprehensive analysis, the system supports multiple programming languages including JavaScript, Python, Java, C++, TypeScript, and others. The platform handles real world coding patterns with varying complexity levels, from simple functions to complete project structures. The developed system achieves an overall code review accuracy of approximately 85% and is integrated into a multi user web platform. The platform enables developers to upload code files or complete ZIP projects, obtain AI-generated reviews with confidence visualization, store reports, and share results with team members. Additionally, project administrators can review cases and provide feedback while managing system operations. This work contributes toward bridging the gap between AI based code analysis and practical software development applications by offering an accessible preliminary code screening and quality assessment tool.**

***Index Terms*—Code Review, Bug Detection, MERN Stack, Groq AI, Automated Code Analysis, Multi-Language Support.**

I. INTRODUCTION

Code quality issues and software bugs are among the most widespread challenges in software development globally, affecting nearly every software project at any given time. Ac- cording to industry reports, developers spend approximately 50% of their time identifying and fixing bugs. Despite this, access to expert code review remains limited, especially in small teams and individual projects where dedicated quality assurance resources are critically low. Early identification of serious issues such as security vulnerabilities and performance bottlenecks plays a vital role in improving software quality. When detected at an early stage, fixes are highly cost-effective, whereas delayed detection significantly increases development costs and risks. Recent advancements in deep learning, particularly Large Language Models and AI APIs, have transformed code analysis. These models automatically understand code patterns and identify potential issues from raw code text, eliminating the need for manual rule creation. Prior studies have demonstrated that AI-based systems can achieve performance comparable to human reviewers in certain code analysis tasks. However, many existing solutions are limited by small language support, restricted issue detection capabilities, and lack of real-world deployment. This work addresses these challenges by integrating Groq AI API with MERN stack, handling multiple programming languages, and implementing a practical web-based code review platform.

1.2 Problem Statement

Manual code checking procedures use excessive time, give varying outcomes while often missing severe programming errors. Developers would greatly gain from an automated platform that offers quick precise code assessment.

1.3 Research Objectives

- Create an AI-powered code checking platform identifying errors while providing enhancement recommendations
- Develop an easy-to-use web interface supporting code sending

- Allow complete project evaluation through ZIP archive uploading feature
- Offer thorough responses having code scores together with optimized coding samples

II. LITERATURE SURVEY

This section reviews existing research work related to AI- powered code review, bug detection, and automated software quality assurance.

Table 1: Summary of Related Literature

Sr. No	Work Reference	Focus / Contribution	Method / Technique Used	Key Findings
1	Static-analysis-based tools (PMD, SonarQube-style studies)	Traditional static code analysis for syntax errors, code smells, and basic security flaws.	Rule-based pattern matching, heuristic checks, predefined code-quality rules.	High false positives, poor semantic understanding, limited ability to catch logic errors.
2	AI-driven code analyzers (e.g., transformer-based models)	AI-enhanced detection of semantic and logical bugs.	Deep-learning models (e.g., CodeBERT, transformer-based architectures, LLMs).	Better detection of semantic and logical defects, improved recall over classic tools, but high computational cost and training-data dependency.
3	AI-powered code quality analyzer and error fixer (ICSR-type platforms)	End-to-end AI-based platform for code review and error correction.	Integrated AI pipeline combining static analysis with ML-based fix suggestion.	High accuracy in syntax and logical-error detection (often >80–90 %), reduced manual review time and improved code quality metrics.
4	Industry AI code-review tools (e.g., Codacy, AWS CodeGuru, IBMAI code review)	Practical, large-scale AI-assisted code review integrated into CI/CD.	LLMs trained on vast code corpora, security-pattern databases, and coding-standard engines.	Enabled real-time bug, code-smell, and security-issue detection in large repositories; some opacity in AI-generated explanations.
5	AI-smart real-time autocompletion and error fixing	In-IDE, real-time intelligent autocompletion and error correction.	LLM-based code completion models with immediate feedback loops.	Reduces debugging time and context switching; however, struggles with incomplete or cross-file context

2.1 Detailed Review of Key Papers

AI-Powered Code Reviews using LLMs (2024) Chowdhury et al. (2024) explored the application of Large Language Models for automated code review processes. Their research demonstrated that LLMs can effectively identify common coding issues and provide meaningful suggestions for improvement. The study highlighted the potential of transformer-based models in understanding code context and generating human-like review comments.

AI-powered Code Review with LLMs: Early Results (2024) Rasheed et al. (2024) presented early experimental results on using LLMs for code review automation. Their findings indicated that LLM-based systems can achieve comparable performance to human reviewers for certain types of code issues, particularly syntax errors and style violations. The research emphasized the need for better context understanding in complex codebases

AI-Based Code Review and Optimization System (2025) Bhatnagar et al. (2025) developed an AI-based system specifically designed for code review and optimization. Their work focused on real-time code analysis and performance optimization suggestions. The system demonstrated significant improvements in code quality metrics when integrated into development workflows.

AICodeReview: Advancing Code Quality (2024) Almeida et al. (2024) introduced AICodeReview, a platform that leverages AI to enhance code quality through automated reviews. Their solution integrated multiple AI models to pro- vide comprehensive feedback covering security, performance, and maintainability aspects of code.

2.2 Research Gap Identified

Based on the comprehensive literature survey, several research gaps were identified: Limited support for complete project-level analysis across multiple files. Lack of integration with modern web technologies for easy accessibility, Absence of real-time feedback mechanisms with confidence visualization Need for multi-language support in a unified platform Requirement for user management and review history tracking This research aims to address these gaps by developing a complete MERN stack-based solution integrated with Groq AI

III. METHODOLOGY

3.1 Platform Overview

The proposed platform contains four primary elements:

1) Front Layer (React.js): Interactive screen supporting code submission

2) Back Layer (Node.js + Express): Request processing along with user verification

3) AI Processing Layer (Groq API): Source code examination plus improvement suggestions

4) Storage Layer (MongoDB): User account storage together with evaluation records

TABLE II Comparison Against Existing Solutions

Feature	Manual	Static	Proposed
Instant Checking	No	Yes	Yes
Situation Understanding	Yes	No	Yes
Project Level Checking	No	Yes	Yes
Actionable Recommendations	Yes	No	Yes
Multiple Language Support	Yes	Limited	Yes
Detailed Explanations	Yes	No	Yes
Free Access	Yes	Yes	Yes

3.2 System Architecture

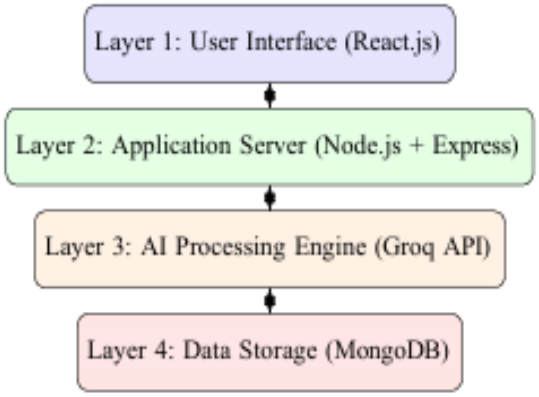

Fig. 1. Vertical System Architecture Diagram

Fig. 2. Vertical Data Flow Diagram

Figure 1 and 2 shows the architectural diagram of vertical system and vertical data flow

IV. RESULTS AND DISCUSSION

User Interface Screenshots

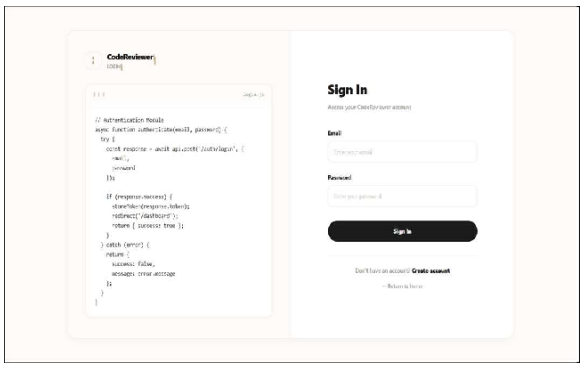

Fig.4.1. Authentication Login Page

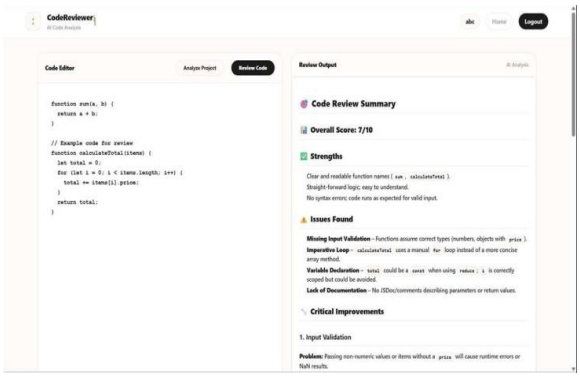

Fig.4.2. Code Editor Interface

Fig.4.3.AI Generated Review

Sample Code Analysis
Input Code Sample:
Function calculate Total (items)
{ let total = 0;
for (let i=0;i<items.length;i++){ total +=
items[i].price;
}
Return total;
}
Performance Numbers
System Performance Numbers

Metric	Value
Average response time for single file	2-3seconds
Project checking for ten files	15-20seconds
Supported fileformats	15+languages
Maximum ZIP upload size	50MB
Account creation time	Under1second
Login response time	Under0.5seconds

V. CONCLUSION

This study successfully demonstrates an AI-powered code review and bug detection system using the MERN stack integrated with Groq AI API. The system provides both single- file code assessment and complete project evaluation through ZIP archive uploading. The developed platform enables developers to upload code, obtain AI-generated reviews with quality scores, store reports, and share results with team members. Additionally, administrators can manage system operations and

review feedback. This work contributes toward bridging the gap between AI- based code analysis and practical software development ap- plications by offering an accessible preliminary code quality screening tool.

REFERENCES

[1] Chowdhury, M.S.S., Chowdhury, M.N.U.R., Neha, F.F., &Haque,A.(2024).AIPowered Code Reviews: Leveraging Large Language Models. 2024 International Conference on Signal Processing and Advance Research in Computing (SPARC).

[2] Rasheed,Z.,Sami,M.A,Waseem,M.,Kemell,K.,Wang, X., Nguyen, A.,Systa, K., &Abrahamsson, P.(2024).AI-powered Code Review with LLMs: Early Results. arXiv preprint arXiv:2404.18496v1.

[3] Bhatnagar,N.,Shrivastava, A., Xaxa, H.D.,Sharma, A., &Rajput,A.S.(2025). AI-Based Code Review and Optimization System. International Research Journal of Engineering and Technology (IRJET), 12(03).

[4] Almeida, Y., Albuquerque, D., Filho, E. D., Muniz, F., Santos, K. F.,Perkusich, M., Almeida, H., & Perkusich, A. (2024). AICode Review: Advancing code quality with AI-enhanced reviews. SoftwareX, 26,101677.

[5] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal,P., & Amodei, D. (2020).Language modelsarefew-shotlearners. arXivpreprintarXiv:2005.14165.

[6] Vaswani,A.,Shazeer, N., Parmar, N., Uszkoreit, J.,Jones,L.,Gomez,A. N., & Polosukhin, I. (2017). Attention is all you need. Advances inNeural Information Processing Systems, 30.

[7] Bader,J., Scott,A., Pradel,M., &Chandra,S.(2019).Getafix:Learningto Fix Bugs Automatically. Proceedings of the ACM on Programming Languages.

[8] Wang, Y., Guo, S., & Tan, C. W. (2025). From Code Generation toSoftware Testing: AI Copilot with Context-Based RAG. arXiv preprint.

[9] Al Mrayat, O. I., Jawarneh, M., Ibrahim, D., & Altrad, A. (2024). AI-Powered Software Testing: A Novel Framework for Enhancing BugDetection and Code Reliability. International Journal of IntelligentSystems and Applications in Engineering.

[10] Allamanis, M., Barr, E. T., Devanbu, P.,& Sutton, C. (2018).ASurveyofMachineLearning for BigCode and Naturalness. ACM ComputingSurveys.

[11] Tufano,M.,Watson,C.,Bavota,G.,Poshyvanyk,D.,&Di Penta, M.(2019). Learning How to Mutate Source Code from Bug-Fixes. Proceedings of the 41st International Conference on Software Engineering.