International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 22 No. 3 (2026)

ijerst.editor@gmail.com

editor@ijerst.com

ISSN 2319-5991 www.ijerst.com

Peer Reviewed, Referred & Indexed Journal

Research Paper

CODE REVIEW ASSISTANT (SPOTS BUGS, SUGGESTS FIXES)

[1] M Mohan rao,[2] M Viswa Simha, [3] B Naveen, [4] A Ashwith kumar, [5] P Pranay
[1]Assistant Professor, [2345]Students
Department of CSE(AIML)
Siddhartha Institute of Technology & Sciences, Narapally
dr.m.mohanrao.co.in, 23TQ1A6602@siddhartha.co.in, 23TQ1A6614@siddhartha.co.in, 23TQ1A6661@siddhartha.co.in, 23TQ1A6662@siddhartha.co.in

ABSTRACT

The Code Review Assistant is an AI-powered software tool designed to analyze source code, identify potential bugs, and suggest suitable fixes. Code review is an important part of software development because it helps developers identify errors, security issues, performance problems, and coding-quality concerns before software is deployed. However, manually reviewing large amounts of code can be time-consuming and may cause developers to overlook certain issues.

The proposed system uses Artificial Intelligence, Natural Language Processing, Static Code Analysis, and Large Language Models to examine source code. Users can upload or paste code into the system and select the programming language being used. The system analyzes the code and identifies syntax errors, logical problems, inefficient code patterns, possible security vulnerabilities, and maintainability issues.

After detecting potential problems, the system provides explanations for the identified issues. An AI-based recommendation module suggests possible corrections and can generate improved code snippets. The assistant explains why an issue may occur and provides guidance on how the developer can resolve it. This makes the system useful not only for debugging but also for learning better programming practices.

The system can also classify detected issues based on categories such as Bug, Security, Performance, Code Quality, and Best Practice. Developers can review the detected issues through an interactive interface and compare the original code with the suggested corrected version. The user remains responsible for reviewing and testing any AI-generated changes before applying them.

Overall, the Code Review Assistant aims to improve software quality while reducing the manual effort involved in code review. It can support individual developers, students, software teams, and organizations by providing faster and more consistent code analysis. Future enhancements can include repository integration, pull-request review, multi-language support, automated testing, security scanning, and continuous code-quality monitoring.

I. INTRODUCTION

Software applications are becoming increasingly complex, and developers write large amounts of code to implement different features and services. As the size of a software

project increases, the possibility of programming errors, security vulnerabilities, and inefficient code also increases. Code review is therefore an essential software development activity that helps identify problems before they affect users or production systems.

Traditional code review is generally performed by developers or reviewers who manually inspect source code. Reviewers examine code structure, logic, security, performance, and adherence to coding standards. Although manual review can provide valuable insights, it requires significant time and effort, especially when the project contains thousands of lines of code or frequent code changes.

Static analysis tools provide automated methods for detecting certain types of programming issues. These tools can identify syntax errors, unused variables, code smells, security patterns, and violations of predefined coding standards. However, traditional static analysis tools may produce large numbers of warnings and may not always provide detailed explanations or context-aware solutions for developers.

Recent advances in Artificial Intelligence and Large Language Models have created new possibilities for intelligent code analysis. AI models can understand programming languages, analyze relationships between code components, explain potential problems, and generate possible solutions. This makes AI useful as an additional layer on top of traditional code-analysis techniques.

The proposed Code Review Assistant combines automated code analysis with AI-based reasoning and fix suggestions. It accepts source code from the user, analyzes the code for different categories of issues, explains detected problems, and provides possible improvements. The system is intended to assist developers rather than replace human review, allowing users to validate and test suggested changes before incorporating them into a project.

II. LITERATURE SURVEY

Traditional software code review has been widely used to improve software quality and identify defects during development. Manual peer review allows experienced developers to examine source code and provide feedback about programming logic, design, readability, and maintainability. However, manual review can become difficult when development teams handle large codebases and frequent changes.

Static Code Analysis has been developed to automate the detection of common programming problems. Static analyzers inspect source code without executing the complete application and can identify syntax problems, unused variables, code smells, complexity issues, and certain security vulnerabilities. These tools are valuable for continuous integration and development workflows, but their findings may sometimes require additional interpretation by developers.

Machine Learning techniques have also been investigated for software defect prediction and code-quality analysis. Machine learning models can learn patterns from historical software repositories and identify code sections that may have a higher probability of containing defects. Such approaches can assist development teams in prioritizing code-

review efforts, although their effectiveness depends on the quality and relevance of training data.

Deep learning and Transformer-based models have significantly improved automated programming-language understanding. Models trained on source code can learn relationships between programming tokens, functions, classes, and common coding patterns. These capabilities have enabled applications such as code completion, code summarization, bug detection, and automated code generation.

Large Language Models have introduced more flexible approaches to code review because they can analyze code and provide natural-language explanations. They can identify potential logical issues, suggest alternative implementations, generate corrected code, and explain programming concepts. However, AI-generated suggestions may occasionally be incorrect or incomplete, making human validation and automated testing important parts of an AI-assisted code-review workflow.

Modern intelligent development tools therefore increasingly combine static analysis, security scanning, testing, and AI-based assistance. A combined approach can provide both deterministic checks and contextual explanations. The proposed Code Review Assistant follows this approach by integrating code parsing, issue detection, AI analysis, fix suggestions, severity classification, and developer feedback into a unified platform.

III. SYSTEM ANALYSIS

The Code Review Assistant analyzes source code to identify potential bugs, vulnerabilities, performance problems, and code-quality issues. The user provides source code by uploading a file, pasting code, or connecting a supported repository. The system first identifies or receives the programming language and performs basic input validation. A preprocessing module analyzes the source code structure and prepares it for further inspection. Static analysis techniques detect syntax errors, code smells, unused elements, complexity issues, and predefined security patterns. An AI analysis module then examines the code using contextual understanding to identify possible logical problems and development concerns. Detected issues are classified into categories such as Bug, Security, Performance, and Code Quality. Each issue is assigned a suitable severity level based on its potential impact. The system generates a clear explanation describing the detected problem and the affected portion of the code. An AI recommendation module then provides a possible fix or improved code snippet. Users can review the original code, detected issues, explanations, and suggested fixes through the application interface. The final recommendations should be tested and reviewed by developers before being applied to production code.

Existing System

Existing code-review processes mainly depend on manual developer review and traditional static-analysis tools. In manual review, developers inspect source code line by line to identify programming errors and quality problems. This process can require significant time when reviewing large projects or frequent code changes. Static-analysis tools automate certain checks and can detect syntax errors, code smells, unused variables, complexity issues, and known vulnerability patterns. However, these tools may generate many warnings that developers need to interpret manually. Traditional

tools generally operate according to predefined rules and may have limited understanding of the broader intent of the code. They may identify a problematic line without providing a detailed explanation of why the problem occurs. Some tools provide documentation links but do not always generate context-specific fixes. Developers may therefore need to investigate the issue separately and write the correction manually. Different programming languages and frameworks may also require different tools and configurations. Existing systems may not provide a single interface combining bug detection, explanations, fix generation, and code-quality recommendations. Therefore, developers can benefit from an intelligent assistant that combines automated analysis with contextual AI assistance.

Disadvantages of Existing System

- Manual code review requires considerable time and effort.
- Large codebases are difficult to inspect manually.
- Reviewers may overlook certain issues.
- Traditional static analyzers depend heavily on predefined rules.
- Large numbers of warnings can create alert fatigue.
- Some tools provide limited explanations for detected problems.
- Developers may need to manually research solutions.
- Fix suggestions may not be context-specific.
- Different programming languages may require separate tools.
- Logical bugs can be difficult to detect using simple rule-based analysis.
- Code-quality improvement recommendations may be limited.
- Continuous manual review can reduce development productivity.

Proposed System

The proposed Code Review Assistant is an AI-powered platform that combines static code analysis with Large Language Model capabilities to identify potential problems and suggest possible fixes. Users can paste source code, upload files, or optionally provide code through a repository integration. The system detects the programming language and validates the submitted code before beginning the review process. A preprocessing and parsing module examines the structure of the source code and prepares it for analysis. Static analysis detects deterministic issues such as syntax errors, code smells, unused variables, complexity problems, and known security patterns. The AI analysis module examines the code context to identify potential logical bugs and additional quality concerns. Detected issues are categorized and assigned severity levels such as Low, Medium, High, or Critical based on defined criteria. The system provides explanations that describe the issue, its possible impact, and the relevant code section. The LLM-based recommendation module generates possible fixes and improved code snippets when appropriate. Users can compare the original code with the suggested version and decide whether to apply the recommendation. The assistant can also provide general best-practice suggestions for improving readability, maintainability, performance, and security. This approach provides developers with an integrated environment for understanding and improving source code.

Advantages of Proposed System

- Automatically analyzes source code for potential problems.

- Detects bugs and common code-quality issues.
- Identifies certain security vulnerabilities and risky patterns.
- Provides explanations for detected issues.
- Generates possible fixes and improved code snippets.
- Classifies issues according to severity and category.
- Reduces repetitive manual review effort.
- Supports faster debugging and development.
- Provides context-aware recommendations.
- Helps students understand programming mistakes.
- Can support multiple programming languages.
- Can be integrated with repositories and development workflows.

IV. METHODOLOGY

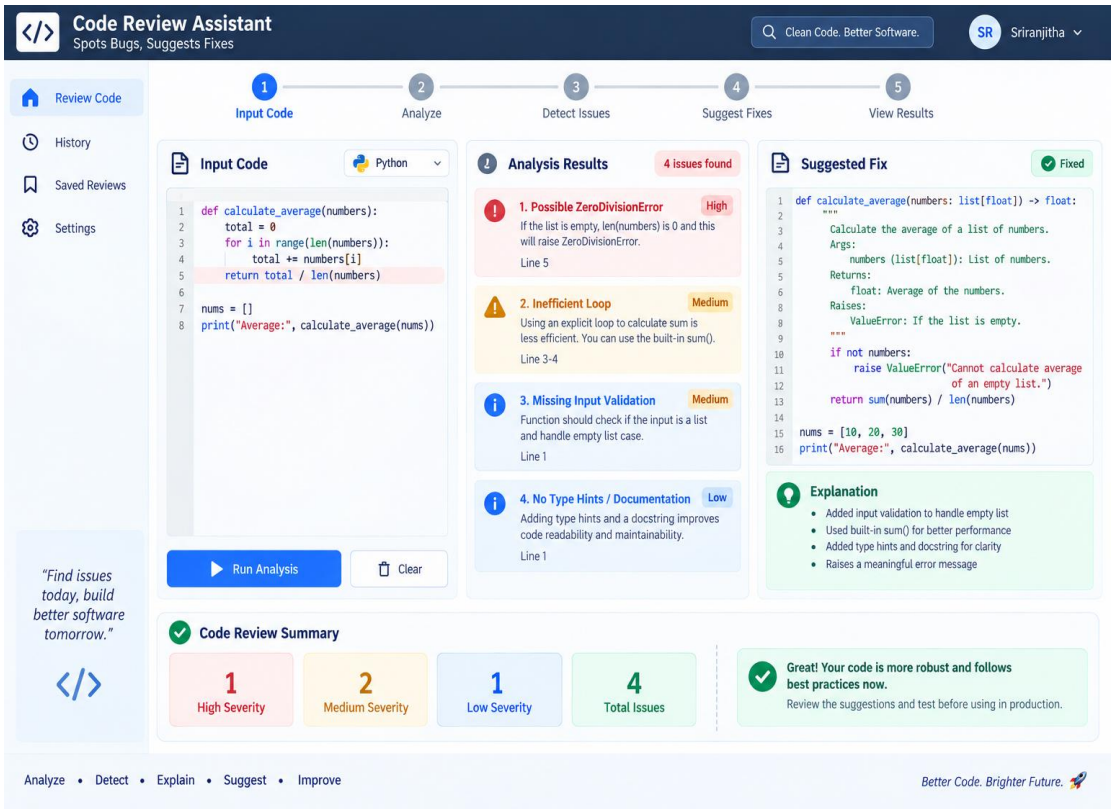
The methodology begins when the user submits source code through the application by pasting code, uploading a file, or connecting a supported repository. The system validates the input and identifies the programming language using file information or code characteristics. The source code is then preprocessed to remove unnecessary information and prepare it for analysis. A parser and static-analysis engine examine the code for syntax errors, code smells, unused variables, complexity problems, and predefined security patterns. The extracted findings are combined with relevant code context and passed to the AI analysis module. The Large Language Model analyzes the code to identify potential logical issues and explain problems that may not be easily detected through predefined rules. Each detected issue is categorized according to its type, such as Bug, Security, Performance, or Code Quality. A severity assessment module assigns an appropriate priority level to help developers focus on important issues. The fix-generation module creates possible corrected code or recommendations based on the detected problem. The system presents the original code, issue explanation, severity, and suggested fix through an interactive review interface. Users can accept, reject, modify, or regenerate suggestions and then test the resulting code independently. Feedback and review results can be stored to improve future analysis and provide useful code-quality insights.

SYSTEM ARCHITECTURE

The system architecture consists of multiple layers that work together to analyze source code and provide intelligent review suggestions. The User Interface Layer allows developers to paste code, upload files, or connect supported repositories for analysis. The Input Processing Layer validates the source code, detects the programming language, and prepares the submitted content. The Code Parsing Layer analyzes the structure of the program and identifies functions, classes, variables, dependencies, and other code elements. The Static Analysis Layer performs rule-based checks for syntax errors, code smells, complexity issues, and known security patterns. The AI Code Analysis Layer uses a Large Language Model to understand code context and identify potential logical problems and quality concerns. The Issue Classification Layer categorizes findings into Bug, Security, Performance, Code Quality, and Best Practice categories. The Severity Assessment Layer determines the priority of each identified issue. The Fix Recommendation Layer generates explanations, possible solutions, and improved code snippets. The Validation Layer checks the generated recommendations for consistency and allows developers to review the proposed changes. The Storage

Layer manages code-review history, findings, user preferences, and feedback securely. The Integration Layer can connect the assistant with Git repositories, CI/CD pipelines, and development environments. Finally, the Output Layer displays the reviewed code, detected issues, explanations, severity levels, and suggested fixes to the developer.

V. RESULT AND OUTPUT



VI. CONCLUSION

The Code Review Assistant provides an intelligent solution for analyzing source code, identifying potential problems, and suggesting possible improvements. The system combines Static Code Analysis, Artificial Intelligence, and Large Language Models to assist developers during the code-review process. It reduces the effort required for manually inspecting large amounts of source code.

The system can detect different types of issues, including possible bugs, syntax errors, security vulnerabilities, performance problems, code smells, and maintainability concerns. Each detected issue can be classified according to its type and severity. The assistant also provides explanations that help developers understand why a particular section of code may cause a problem.

One of the important features of the proposed system is its AI-based fix recommendation capability. After identifying an issue, the system can suggest an improved code snippet and explain the changes made. Developers can compare the original code with the suggested solution and modify the recommendation according to their requirements. AI-generated fixes should be reviewed and tested before they are applied to production systems.

The proposed system can be useful for software developers, students, development teams, and organizations that want to improve code quality and development efficiency. It can support faster debugging, improve understanding of programming errors, and encourage better coding practices. By combining automated static checks with contextual AI analysis, the system provides a more comprehensive code-review experience.

In the future, the Code Review Assistant can be enhanced with support for additional programming languages, GitHub and GitLab integration, automated pull-request reviews, CI/CD pipeline integration, unit-test generation, advanced security analysis, and personalized coding recommendations. These enhancements can make the system a powerful AI-assisted development tool for improving software quality, security, and maintainability.

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.

[12] 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.

[13] 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.

[14] 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.

[15] M. M. Rao, M. B. Mohiuddin, P. Srinu, A. Satyanarayana, J. Madhavan and R. D. Kumar, "Carbon-Neutral-Agriculture: Renewable-Energy Powered IoT Systems and Resource Optimization," *2026 3rd International Conference on Research Methodologies in Knowledge Management, Artificial Intelligence and Telecommunication Engineering (RMKMATE)*, Chennai, India, 2026, pp. 1-6, doi: 10.1109/RMKMATE69073.2026.11518903.

[16] R. Uma, M. Maggidi, A. R. Ekkati, S. Bandlamudi, C. Madugundi and S. K. Taduri, "Zero-Trust Architecture for Hybrid Cloud Security in Critical Infrastructures," 2026 IEEE International Conference for Convergence in Computing

Technology (I3CTCON), Lonavala, India, 2026, pp. 1-6, doi: 10.1109/I3CTCON68242.2026.11507458.

[17] VNS Manaswini, Chithanoor Arjun, Yamini Chouhan, & V. Sumalatha. (2026). MB-EFF: A Multi-Branch Environmental Feature Fusion Deep Learning Model for Crop Classification and Recommendation. *International Journal of Computer Information Systems and Industrial Management Applications*, *18*(22s), 460–468. https://doi.org/10.70917/ijcisim-2026-5453

[18] Yamini Chouhan, SIDDOJU DIVYA, VNS Manaswini, & J. Priyanka. (2026). Vision-Threatening Diabetic Retinopathy Detection Using EfficientNetB4: A Two-Stage Transfer Learning Approach. *International Journal of Computer Information Systems and Industrial Management Applications*, *18*(22s), 469–476. https://doi.org/10.70917/ijcisim-2026-5454

[19] Real Time Vehicle Counting and Classification system, **Dr. A.Satyanarayana,et.al** International Research Journal of Moderrnization in Engineering Technology and Science, Vol-06, Issue-05, 2024 **[ISSN: 2582-5208] doi:** 10.56726/IRJMETS57804

[20] Facial Depth Maps: Detecting Sleep Apnea with Deep Learning, **Dr. A.Satyanarayana,et.al** , International Journal of Scientific Development and Research, Vol 9, Issue-5, 2024 **[ISSN: 2455-2631]**

[21] Moodchef: Mood Based Smart Recipe Recommender with cooking Assistance, **Dr. A.Satyanarayana,et.al ,** International Journal of Progressive Research in Engineering Management and Sciences, Vol-05,Issue-7, July 2025 **[ISSN: 2583-5162]** doi: **10.58257/IJPREMS42696**

[22] News Article Text Summarization, **Dr. A.Satyanarayana,et.al,** Journal of Emerging Technologies and Innovative Research, Vol-11, Issue -5, 2024, **[ISSN: 2349-5162]**

[23] Voice Assistant and gesture controlled virtual mouse using Deep learning Techniques, **Dr. A.Satyanarayana,et.al ,** International Research Journal of Moderrnization in Engineering Technology and Science Vol-6, Issue-5, 2024 **[ISSN: 2582-5208] 10.56726/IRJMETS57214**