IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

CodeSage AI: Intelligent Code Review and Optimization Platform

Vedika Patil, Riya Badwal, Sakshi Bane, Jasvendersingh Badbujara Researchers, Computer Engineering

K.C. College of Engineering, Management Studies and Research. Kopri, Thane (E), Maharashtra, India

Abstract:

The increasing complexity of modern software development demands ef7icient and reliable methods for ensuring code quality. Traditional manual code reviews, while valuable, are often time-consuming, error- prone, and unable to keep pace with rapid development cycles. Existing automated tools provide static rule-based analysis but lack adaptability, context-awareness, and real-time feedback. To address these challenges, this project proposes **CODE SAGE AI**, an intelligent code review platform that leverages **arti2icial intelligence and machine learning (ML)** to provide automated, context-sensitive, and real-time feedback on source code.

INTRODUCTION:

Software development today operates in fast-paced agile and continuous delivery environments, which demand quick, consistent, and reliable code review processes. While traditional manual code reviews are valuable, they are often time-consuming, subjective, and prone to missing defects, especially in large and frequently changing codebases. The rapid growth of Arti7icial Intelligence and Machine Learning provides an effective solution by enabling automated, intelligent, and context-aware code analysis. NLP-based models such as CodeBERT and GPT treat source code as a form of language, allowing systems to understand syntax, semantics, and developer intent more accurately. Built on this approach, CodeSage AI is an AI-powered code review platform that analyzes code in real time to identify bugs, inef7iciencies, and potential security risks. It supports multiple programming languages and integrates seamlessly with developer tools and CI/CD pipelines. Additionally, CodeSage AI continuously learns from developer feedback, improving the quality of its suggestions over time and signi7icantly reducing manual review effort.

MOTIVATION:

Developers spend a signi7icant amount of time reviewing and debugging code, reducing overall productivity.Manual reviews often miss subtle issues related to maintainability, scalability, or security.AI- based platforms can provide consistent and unbiased reviews, minimizing human error.The growing need for faster software delivery cycles (Agile/DevOps) demands automated, intelligent tools to assist in code review. Thus, CODE SAGE AI is motivated by the need to bridge the gap between traditional manual reviews and modern AI-powered solutions.

Problem Statement:

“How can AI be leveraged to build an intelligent, adaptive, and real-time code review platform that improves code quality, ensures compliance with best practices, and assists developers effectively?”

Objectives:

- To analyze existing code review practices and tools.
- To identify gaps and limitations in current automated review platforms.
- To design a machine learning/NLP-based system capable of understanding and analyzing code semantics.
- To implement real-time suggestions for improving code quality, ef7iciency, and security.
- To evaluate the platform against existing tools through performance metrics (accuracy, precision, recall, developer usability feedback).

.

RESEARCH METHODOLOGY:

1. **Project Setup:**

The problem of inefficient manual code reviews is identified, existing tools are studied, and system requirements, datasets, and technologies are selected.

2. **Design and Development:**

The system architecture is designed and rule-based and AI/NLP models are developed to analyze code and generate review suggestions.

3. **Implementation and Monitoring:**

The system is implemented and integrated with developer tools and CI/CD pipelines, while performance and user interactions are continuously monitored.

4. **Evaluation and Enhancement:**

The system is evaluated using accuracy and performance metrics, compared with manual reviews, and improved through retraining and feedback-based refinement.

LITERATURE SURVEY:

Modern Code Review (MCR) is widely adopted because it improves defect detection, knowledge sharing, and code ownership—but it still depends heavily on reviewer time, expertise, and consistency. Recent surveys consolidate evidence that review effectiven**ess varies with context (change size, reviewer experience, time pressure), motiv**ating automation to reduce reviewer load.Pretrained transformer models treat source code and developer text (comments, descriptions) as a joint language, enabling tasks like code representation learning, defect prediction transfer, and suggestion generation. CodeBERT-style approaches are frequently used as a backbone for code understanding in research, including automated review tasks that combine sequence information with program structure signals (e.g., graphs).

Survey of Limitation in the Existing System or Research Gap:

- Most traditional static analysis and code review tools rely heavily on predefined rules, making them rigid and unable to adapt to project-specific coding styles or evolving development practices.
- Existing tools often generate a high number of false positives, which leads to alert fatigue and causes developers to ignore important warnings.
- Many AI-based solutions focus mainly on defect detection but provide limited or unclear explanations, making it hard for developers to trust or understand the suggestions.
- Current systems have weak learning mechanisms and do not effectively improve from developer feedback such as accepted or rejected suggestions.
- Integration with real-time development workflows and CI/CD pipelines is sometimes incomplete or slow, reducing practical usability.

Security-focused review is still shallow in many tools, with limited ability to detect complex, context-dependent vulnerabilities.

- There is a lack of hybrid systems that effectively combine rule-based precision with AI-based adaptability

in a single, intelligent code review platform.

PROPOSED SYSTEM:

This project was developed to help out students in need of the hour by providing them with different learning materials like Syllabus, E-Books and PYQs.
We have also provided quizzes for the students to test their knowledge and skills.
Discussions section helps students to access information registered by their seniors with a scope to learn how to tackle common academic issues.

For the above, the webapp was developed by using technology as:

- Minimum: Dual-core processor, 8 GB RAM, 20 GB storage
- Recommended: Quad-core processor, 16 GB RAM, 100 GB storage (for training ML models)
- Programming Languages: Python, JavaScript
- Frameworks: TensorFlow/PyTorch (ML), Flask/Django (backend), React/Angular (frontend)
- Databases: PostgreSQL / MongoDB
- Development Tools: Git, Docker, VS Code, Postman
- Deployment: AWS / Azure / Google Cloud

Architecture / Framework:

We have defined the architecture of the website as follows:

1. **User Interface Layer**

Provides dashboards, IDE plugins, and web interfaces where developers submit code, view suggestions, and give feedback on AI recommendations.

2. **Integration Layer**

Connects with GitHub/GitLab, IDEs, and CI/CD tools to trigger automatic code analysis on commits, pull requests, or in real time.

3. **Preprocessing Layer**

Parses source code, removes noise, builds syntax trees, and converts code into formats suitable for rule-based and AI models.

4. **Rule-Based Analysis Engine**

Uses static analysis rules to detect syntax errors, style violations, and known vulnerability patterns.

5. **AI/ML & NLP Engine**

Applies trained ML models and NLP/LLM models to detect code smells, logical issues, performance risks, and generate review comments.

6. **Decision Fusion Layer**

Combines outputs from rule-based and AI engines, removes duplicates, prioritizes findings, and assigns confidence scores.

7. **Feedback & Learning Module**

Collects developer responses (accept/reject/modify) and uses them to retrain and fine-tune models over time.

8. **Data Storage Layer**

Stores code samples, analysis results, logs, model data, and feedback history securely.

9. **Reporting & Visualization Layer**

Generates detailed review reports, trend analysis, and quality metrics for teams and managers.

10. **Security & Access Control Layer**

Handles authentication, authorization, data encryption, and secure handling of source code.

Flowchart

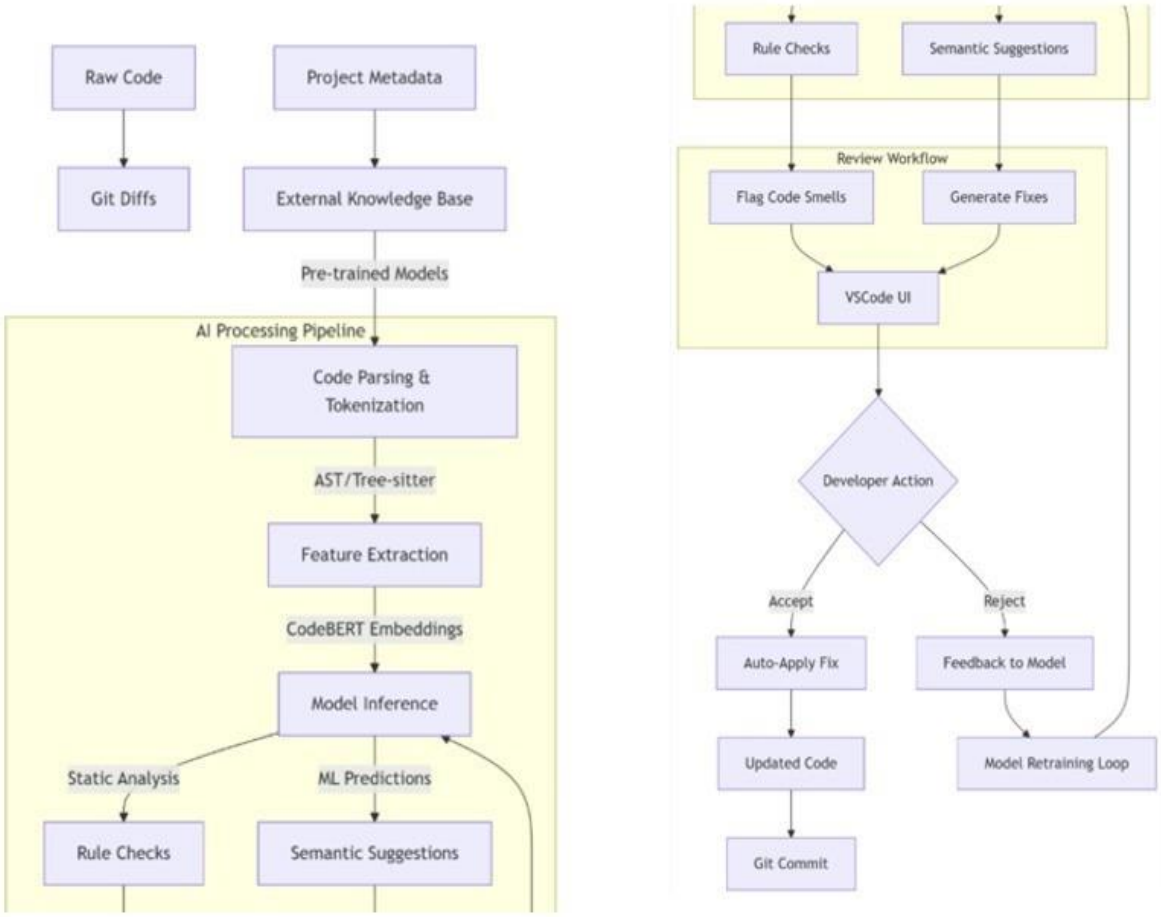

Fig. 1. Flow Chart On Home Energy Monitor Webapp

Main Home Page:

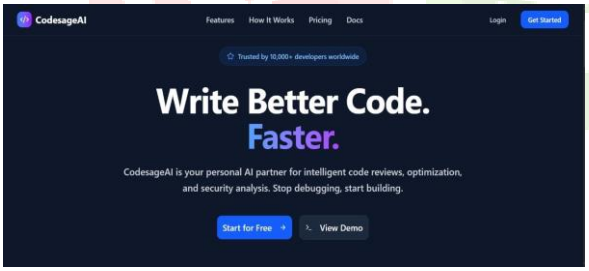

Figure 2

Figure 2: The 7irst page presented to the user upon opening the WebApp.

Login Page:

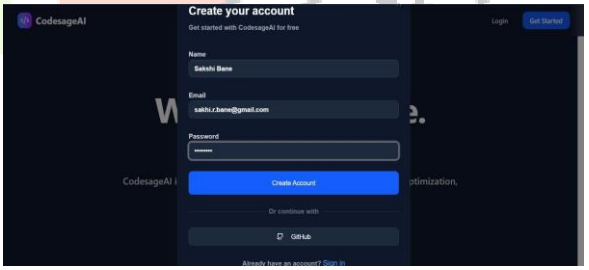

Figure 3

Figure 3- The Register dialog box allows the user to fill credentials with name, email and password for access purposes. It also provides an option to jump on the login page if the user already has an account.

.

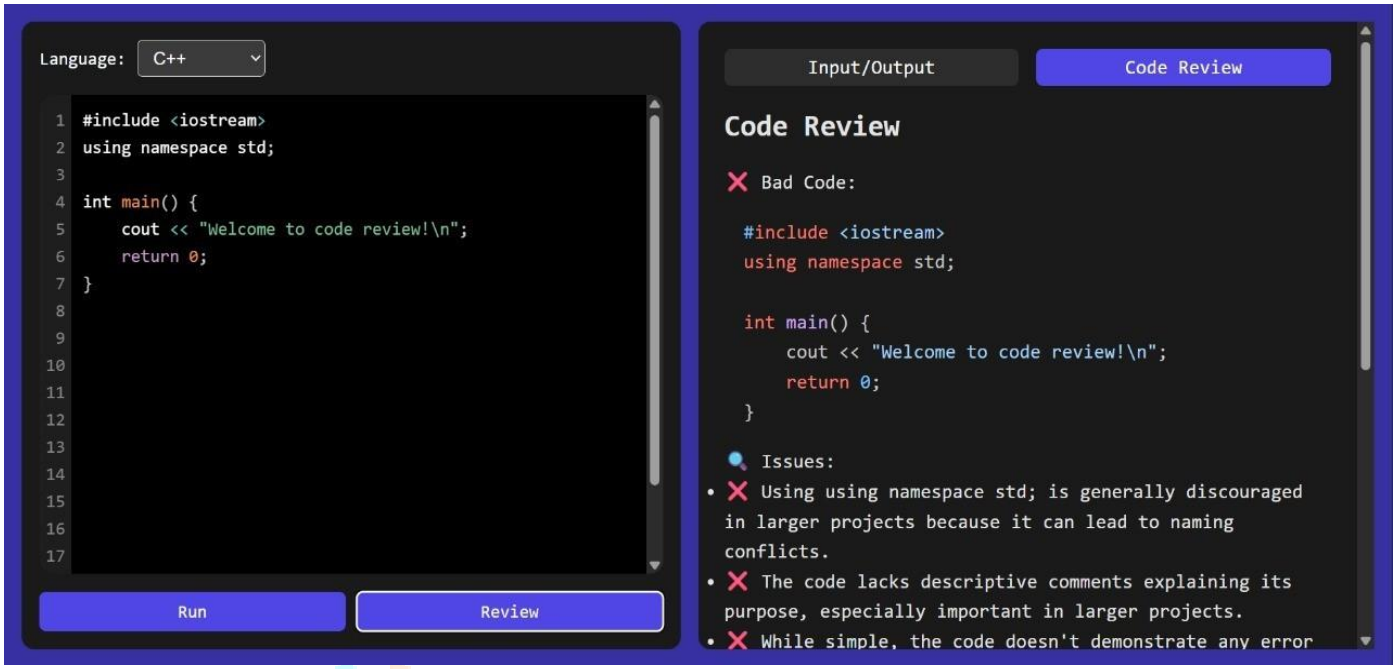

Figure 4

Figure 4 - This page provides the interface to users for practising their codes

FUTURE SCOPE:

The future scope of the system includes expanding support for a wider range of programming languages, frameworks, and domain-speci7ic coding standards to make it more versatile across projects. It also aims to integrate automatic code 7ixing and patch generation to quickly resolve common bugs and security vulnerabilities. Advanced security analysis will be enhanced through threat modeling and deep-learning- based vulnerability prediction, enabling proactive risk detection. Additionally, personalized review models will adapt to individual developers' or teams' coding styles, improving the relevance and accuracy of feedback. Finally, deeper integration with IDEs and DevOps pipelines will enable fully automated, real- time quality assurance throughout the development lifecycle.

REFERENCES:

1. Y. Almeida, "AICodeReview: Advancing code quality with AI-enhanced reviews," *Procedia Computer Science*, vol. 202, pp. 487–494, 2024. ScienceDirect

2. U. Cihan et al., "Automated Code Review In Practice," *arXiv preprint arXiv:2412.18531*, Dec. 2024. arXiv

3. Z. Rasheed et al., "AI-Powered Code Review with LLMs: Early Results," *arXiv preprint arXiv:2404.18496*, Apr. 2024. arXiv

4. X. Tang et al., "CodeAgent: Autonomous Communicative Agents for Code Review," arXiv preprint arXiv:2402.02172, Feb. 2024. arXiv arXiv

5. J. Kumar and S. Chimalakonda, "Code Review Automation Via Multi-task Federated LLM — An Empirical Study," arXiv preprint arXiv:2412.15676, Dec. 2024. arXiv arXiv

6. Y. Peng, K. Kim, L. Meng, K. Liu, "iCodeReviewer: Improving Secure Code Review with Mixture of Prompts," arXiv preprint arXiv:2510.12186, Oct. 2025. arXiv arXiv