

AI CODE REVIEWER AND BUGS PREDICTOR

Santhosh S, Sanjay Palani S, Vasanthakumar S, Vivin R Ms.K.Sugashini

Department of Information Technology, Batchelor of Technology, Sri Shakthi Institute of Engineering and Technology
(Autonomous) Coimbatore-641062

ABSTRACT

Software development requires continuous code review and bug detection to ensure quality and reliability, yet manual processes are often time-consuming and prone to human error. This project implements an automated system that streamlines the process of analyzing and reviewing source code. Code inputs are systematically processed through AI-driven models, static analysis techniques, and pattern recognition algorithms, with defect probabilities calculated according to established benchmarks in software engineering. Direct detection is derived from syntax-level and semantic analysis using weighted evaluation models, while indirect validation is obtained from historical bug datasets and developer feedback, ensuring a comprehensive assessment of code quality. The system also performs detailed mapping of potential error-prone sections to highlight vulnerable areas, using confidence scores to compute overall reliability. By digitizing the code review workflow, the project improves accuracy, consistency, and efficiency in software validation, enabling developers and organizations to continuously monitor and enhance code quality in line with modern development practices.

INTRODUCTION

In contemporary software development, the focus is shifting from traditional manual code review to automated systems that evaluate the quality and reliability of source code. Conventional inspection methods often fail to identify hidden bugs, logical errors, or inefficient patterns, leaving developers and organizations vulnerable to software failures and security risks. AI-based code review and bug prediction address this issue by systematically analyzing how well written code meets quality standards and how it contributes to broader goals of software reliability and maintainability. This project implements a web-based system that enables users to upload, process, and evaluate source code using a structured, stepwise approach. The system integrates AI-driven models, static code analysis, and pattern recognition techniques to assess code quality, highlight error-prone sections, and generate confidence scores. The platform strengthens accountability, enhances transparency, and ensures that every developer can align coding practices with organizational and industry standards for building robust and secure software systems.

Objective:

- To design and implement a system that detects bugs and potential errors in uploaded source code using artificial intelligence techniques
- To develop a web-based dashboard that allows users to easily upload code files and receive automated review results.

- To analyze code inconsistencies such as syntax errors, logical flaws, and inefficient patterns for accurate bug prediction.
- To provide a reliability score or confidence level indicating the probability of the code being correct or faulty.
- To integrate machine learning models (such as neural networks) for effective classification of error-prone and clean code.
- To ensure the system is scalable, user-friendly, and capable of handling multiple code inputs efficiently

LITERATURE SURVEY

- **Early Approaches in Code Review and Bug Detection**
 - Initial techniques for detecting bugs focused on traditional methods such as manual code review, rule-based systems, and static code analysis tools.
 - Methods like syntax checking, linting tools, and pattern matching were used to identify errors in source code.
 - These approaches were effective for simple issues but failed to detect complex logical bugs and hidden vulnerabilities.
- **Emergence of Machine Learning in Software Analysis**
 - With the introduction of machine learning techniques, automated bug detection systems began to evolve.
 - ML-based models can learn patterns from historical code and identify potential defects in new code.
 - This advancement increased the need for more intelligent and adaptive code review systems.
- **Modern AI-Based Code Analysis Techniques**
 - Recent research focuses on deep learning models such as neural networks for automated code review and bug prediction.
 - These models analyze code structure, syntax, and semantic patterns to detect potential issues.
 - For large codebases, sequence-based models are used to capture relationships between different parts of the code.
 - Pre-trained models and large datasets have significantly improved prediction accuracy.
- **Comparative Studies on Detection Methods**
 - Studies show that AI-based approaches outperform traditional static analysis tools in identifying complex bugs.
 - Hybrid models combining rule-based and learning-based techniques provide better results.
 - However, no system guarantees 100% accuracy due to the complexity and variability of software development.
- **Challenges in AI-Based Code Review Systems**
 - High computational cost for processing large codebases and training models.
 - Difficulty in detecting context-specific and logical errors.
 - Need for large labeled datasets for training accurate models.
 - Balancing prediction accuracy with real-time performance in web-based systems.
- **Conclusion and Future Trends**
 - AI-based code review is evolving with the integration of advanced models, including transformer-based architectures.

METHODOLOGY

The methodology for the AI Code Reviewer and Bug Predictor system starts with analyzing the growing

need for automated code review and early bug detection due to the increasing complexity of modern software development. A structured design is developed to process uploaded source code through preprocessing techniques and machine learning models for accurate analysis. The system includes a user-friendly web interface for uploading code files, automated analysis using models such as neural networks and static code analysis tools, and detection of inconsistencies such as syntax errors, logical flaws, and inefficient coding patterns. The prototype is implemented using web technologies to enable dynamic file upload, real-time processing, and clear display of results indicating potential bugs along with a reliability score. The system is tested for accuracy, performance, and usability, with developer feedback guiding further improvements. Finally, optimizations are made to enhance prediction accuracy, processing speed, and scalability, with future plans to integrate real-time IDE support and advanced AI models for improved reliability.

Existing system:

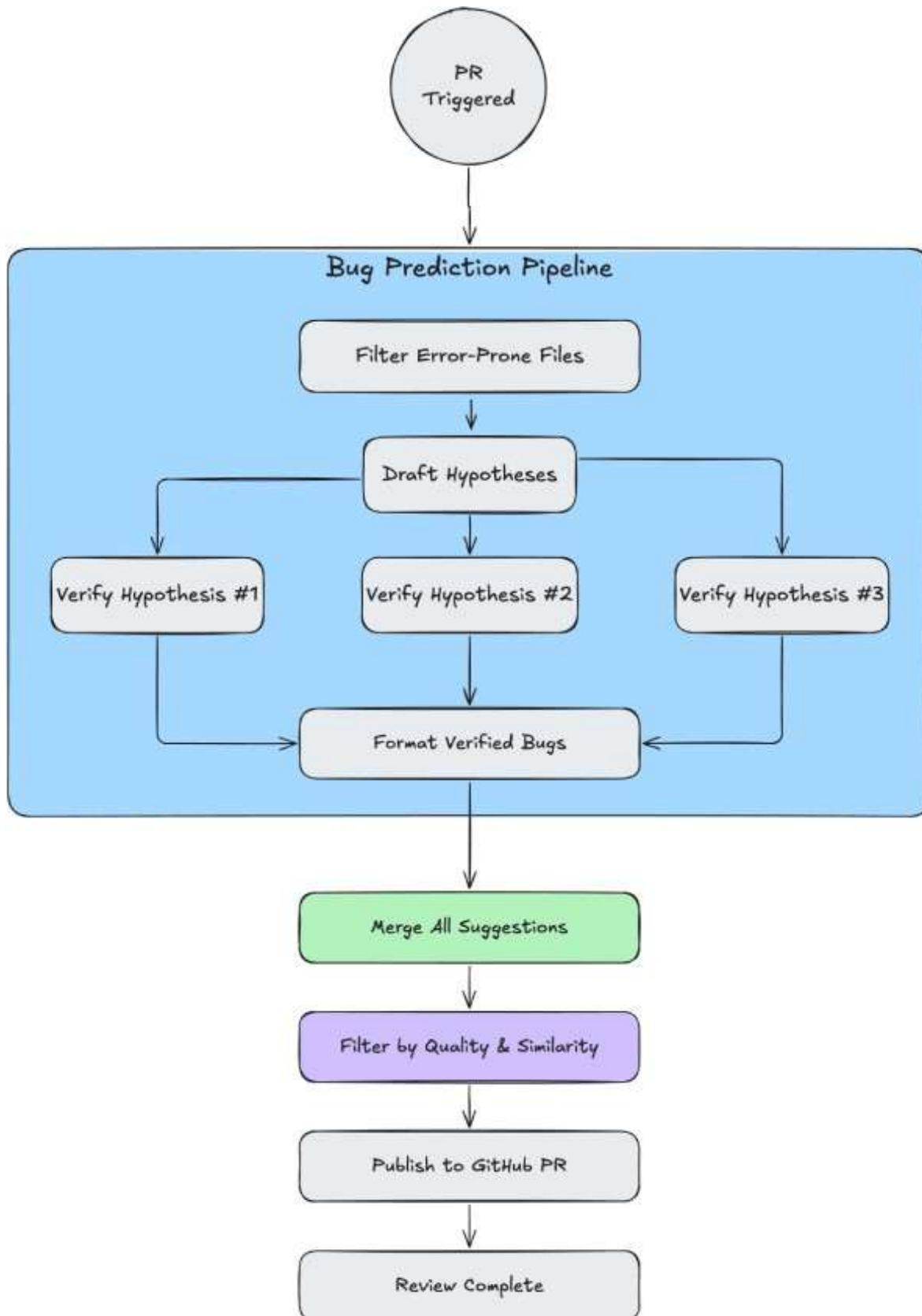
In existing code review and bug detection systems, analysis is often limited or partially automated, with developers relying on manual review or basic tools to identify errors. Many approaches depend on simple static analysis techniques or standalone tools, which can lead to inconsistent results and reduced accuracy. While some AI-based code analysis tools exist, they often lack full integration for handling different programming languages and complex code structures, limiting comprehensive evaluation of software quality. Additionally, most systems do not effectively combine syntax and semantic analysis or provide clear reliability scores, reducing their effectiveness for real-world applications and large-scale development environments.

Disadvantages

- Prone to errors and inconsistencies due to reliance on manual code review and basic analysis tools.
- Limited automation prevents efficient handling of large volumes of source code for analysis.
- Incomplete integration between different analysis techniques hampers comprehensive code evaluation.
- Lack of advanced pattern recognition reduces the ability to accurately detect complex and hidden bugs.
- User interfaces in existing systems may be complex, impacting developer experience and accessibility.

Proposed system:

The proposed system automates code review and bug prediction by integrating multiple analysis techniques into a unified platform that evaluates source code quality automatically. It features advanced AI-based models to analyze code inconsistencies and detect potential errors, aiding software reliability and development efficiency. The system prioritizes usability with intuitive code upload interfaces for developers and real-time result display through a web dashboard. It incorporates trained machine learning models and optimized analysis techniques to ensure consistent and accurate bug detection. Additionally, the system supports scalability for handling large volumes of code, includes error-handling mechanisms to improve reliability, and offers detailed confidence scores for better interpretation, aligning technological capabilities with practical software development needs. Furthermore, the system can provide suggestions for code improvement and best practices, helping developers enhance code quality during the development process. It also enables continuous monitoring by allowing repeated analysis of updated code, ensuring ongoing reliability and performance. The integration of modular components makes the system adaptable to different programming languages and development environments, increasing its flexibility and practical applicability in real-world scenarios. In addition, the system supports integration with version control platforms, enabling automated analysis during code commits and updates. It can generate detailed reports highlighting error-prone areas, code complexity, and maintainability metrics, which assist developers in making informed decisions.



SYSTEM REQUIREMENTS

1. Hardware Requirements

- **Processor:** Intel i3 or higher
- **RAM:** 4 GB minimum
- **Storage:** 100 MB free space
- **Display:** Monitor with minimum resolution 1024×768
- **Input Devices:** Mouse, Keyboard, or Touchscreen

2. Software Requirements

- **Operating System:** Windows 10/11, Linux, or macOS
- **Development Framework:** MERN stack (MongoDB, Express.js, React.js, Node.js)
- **Database:** MongoDB (preferred for MERN), or alternatives like MySQL
- **Browser:** Latest versions of Chrome, Firefox, Edge (for web deployment)

Module Description

1. Code Upload & Selection Module

- User uploads source code files through the web dashboard.
- Supports multiple formats (Python, Java, C++, etc.).
- Displays preview of the uploaded code for verification.
- Facilitates selection between different analysis modes (syntax check, bug prediction, etc.).

2. Preprocessing & Data Preparation Module

- Processes uploaded code using techniques such as tokenization, parsing, and normalization.
- Breaks code into smaller units like functions or statements for detailed analysis.
- Identifies and extracts important sections such as loops, conditions, and functions.
- Prepares structured input data suitable for machine learning models.

3. Feature Extraction Module

- Extracts important code features such as syntax patterns, control flow, and complexity metrics.
- Identifies inconsistencies, redundant code, and abnormal patterns.
- Converts code into feature representations suitable for model processing.
- Enhances relevant features to improve prediction accuracy.

4. AI-Based Bug Detection Module

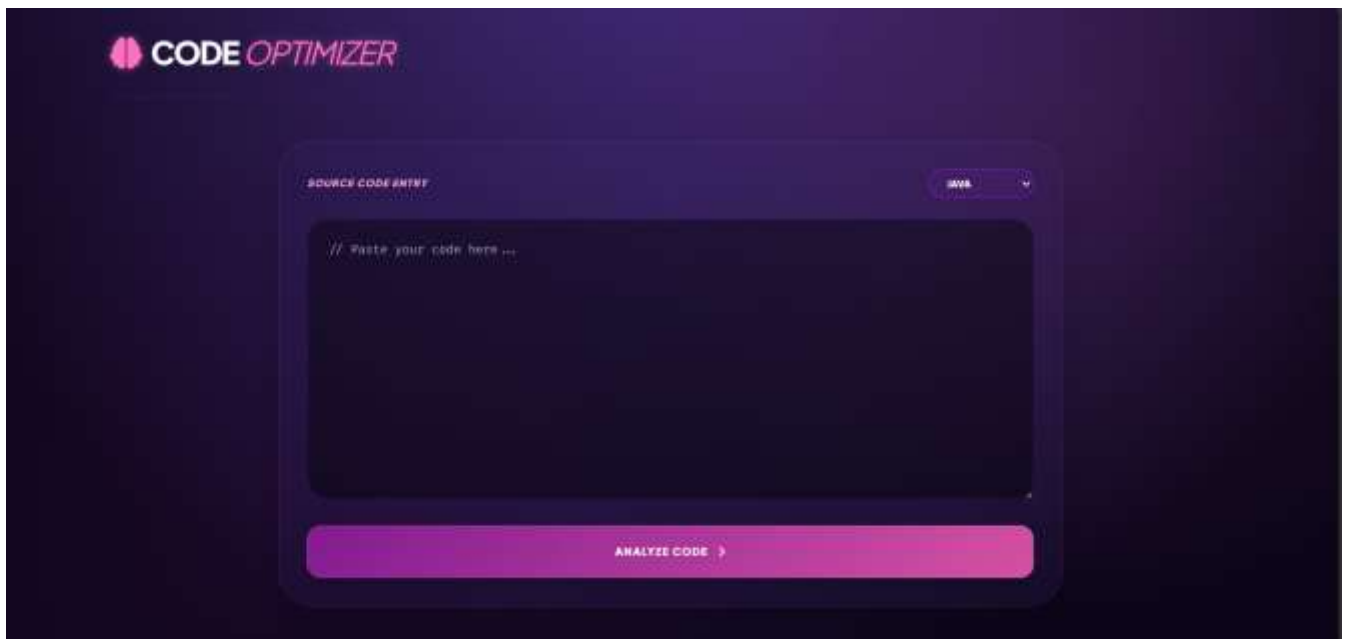
- Uses models such as machine learning algorithms or neural networks for classification.
- Analyzes syntax and semantic patterns in the code.
- Detects issues such as logical errors, inefficient coding practices, and potential vulnerabilities.
- Classifies code segments as error-prone or reliable based on learned patterns.

5. Reliability Scoring Module

- Generates a confidence score indicating the quality and correctness of the code.
- Applies threshold values to determine classification (Buggy/Clean).
- Ensures consistent evaluation across different types of code.
- Provides quantitative output for better interpretation.

6. Result Visualization & Dashboard Module

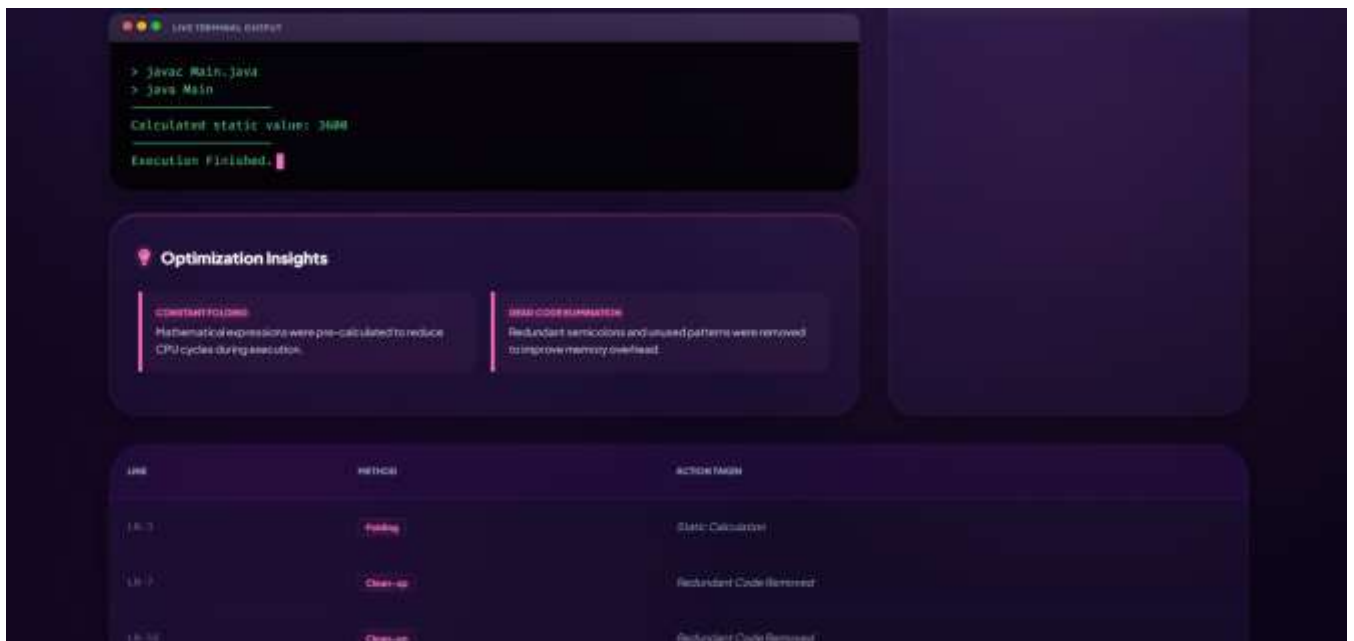
- Displays final results (Bug Detected or Clean Code) on the web interface.
- Shows confidence percentage and detailed analysis summary.
- Provides a user-friendly dashboard for easy understanding.
- Ensures fast response and smooth user interaction.

MAINPAGE:**Page 1:**

Page 2:



Page 3:



CONCLUSION

In conclusion, the development of the AI Code Reviewer and Bug Predictor system provides an effective framework to analyze and ensure the quality and reliability of source code. By systematically evaluating code using artificial intelligence techniques, the system ensures that review processes are accurate, consistent, and result-oriented. The project enhances software development efficiency by allowing developers to examine uploaded code through data-driven methods. The automatic classification of code as buggy or clean helps identify potential errors and supports better decision-making. Furthermore, the use of advanced models such as neural networks ensures that the system can detect complex patterns and hidden issues, improving prediction accuracy. The integration of multiple analysis techniques also provides a comprehensive evaluation of code quality. Overall, this project contributes to improving reliability, transparency, and efficiency in software development. It supports developers in implementing a structured, efficient, and technology-driven approach that helps reduce errors and promotes the development of robust and maintainable software systems.

REFERENCES

- [1] T. J. McCabe (1976). *A Complexity Measure*. IEEE Transactions on Software Engineering.
- [2] S. Panichella, A. Di Sorbo, E. Di Penta (2015). *How Developers' Collaborations Identify Code Defects*. IEEE International Conference on Software Maintenance and Evolution (ICSME).
- [3] Y. Zhou, H. Leung (2006). *Empirical Analysis of Object-Oriented Design Metrics for Predicting High and Low Severity Faults*. IEEE Transactions on Software Engineering.
- [4] T. Menzies, J. Greenwald, A. Frank (2007). *Data Mining Static Code Attributes to Learn Defect Predictors*. IEEE Transactions on Software Engineering.
- [5] V. J. Hellendoorn, P. Devanbu (2017). *Deep Learning Type Inference*. ACM Joint Meeting on Foundations of Software Engineering (FSE).
- [6] M. Allamanis, E. T. Barr, P. Devanbu, C. Sutton (2018). *A Survey of Machine Learning for Big Code and Naturalness*. ACM Computing Surveys.
- [7] Z. Li, D. Zou, S. Xu, H. Jin (2018). *VulDeePecker: A Deep Learning-Based System for Vulnerability Detection*. NDSS Symposium.