AI Powered code Reviewer and Detector

[1]I. Swapna,[2]Sheik Shireen,[3]Naini Nandini,[4]Chilakala Navya,[5]Jatoth Deepika, [6]Kalmuri Manjula

[1]Assistant Professor, Department of Computer Science & Engineering, Princeton Institute of Engineering & Technology For Women

[2,3,4,5,6]B. Tech Students, Department of Computer Science & Engineering, Princeton Institute of Engineering & Technology For Women

ABSTRACT

The AI Powered Code Reviewer and Bug Detector is an intelligent web-based system designed to automatically analyze programming code and identify potential bugs and security risks using machine learning techniques. The system is developed using the Flask web framework and integrates multiple machine learning algorithms to evaluate and classify code quality. The primary objective of this system is to assist developers in detecting errors and vulnerabilities in source code at an early stage, thereby improving software reliability and development efficiency.The proposed system uses a dataset of labeled code snippets to train several machine learning models, including Logistic Regression, Support Vector Machine (SVM), Random Forest, Decision Tree, Naïve Bayes, and K-Nearest Neighbors (KNN). The code snippets are first converted into numerical representations using the TF-IDF vectorization technique, which extracts important textual features from the code. These features are then used to train and evaluate different classification models. The system automatically compares the accuracy of each model and selects the best-performing model for deployment. The trained model is stored and used to predict whether a given piece of code is clean or contains potential bugs.In addition to machine learning-based bug detection, the system also performs basic security checks to identify risky functions such as eval() and exec() that may introduce security vulnerabilities. The application provides a user-friendly web interface where users can register, log in, train the machine learning model, submit code for review, and view their analysis history. The results of model performance are visualized using accuracy comparison graphs to help understand the effectiveness of different algorithms.Overall, the proposed system demonstrates how artificial intelligence and machine learning can be integrated into software development workflows to automate code review processes. By providing instant feedback on code quality and potential risks, the system helps developers write more secure and reliable programs while reducing the time required for manual code inspection.

Keywords: AI Code Reviewer, Bug Detection, Machine Learning, Code Analysis, TF-IDF Vectorization, Software Security, Vulnerability Detection, Flask Web Application, Code Classification, Automated Code Review.

I. INTRODUCTION

Software development has become increasingly complex as modern applications require high levels of reliability, security, and efficiency. One of the major challenges faced by developers is identifying bugs, logical errors, and security vulnerabilities in source code during the development process. Traditional code review methods rely heavily on manual inspection by experienced developers, which can be time-consuming, error-prone, and inefficient for large-scale software systems. As software projects continue to grow in size and complexity, there is a need for automated tools that can assist developers in detecting issues in code quickly and accurately.

Artificial Intelligence (AI) and Machine Learning (ML) technologies have emerged as powerful solutions for automating various tasks in software engineering, including code analysis and bug detection. By learning patterns from previously labeled code samples, machine learning models can classify new code snippets and determine whether they contain potential bugs or vulnerabilities. These intelligent systems can analyze large amounts of code efficiently and provide immediate feedback to developers, improving overall software quality and reducing development time.

The AI Powered Code Reviewer and Bug Detector is designed to address these challenges by integrating machine learning algorithms with a web-based application framework. The system allows users to submit code snippets for automatic analysis and classification. It utilizes text processing techniques

such as Term Frequency–Inverse Document Frequency (TF-IDF) to convert code into numerical features that can be processed by machine learning models. Multiple algorithms, including Logistic Regression, Support Vector Machine (SVM), Random Forest, Decision Tree, Naïve Bayes, and K-Nearest Neighbors (KNN), are used to train and evaluate the system for accurate bug detection.

The application is developed using the Flask framework and includes essential features such as user authentication, model training, code review, and history tracking. The system compares the performance of different machine learning models and automatically selects the best-performing model for prediction. Additionally, basic security checks are implemented to detect potentially dangerous functions such as eval() and exec() that may lead to security vulnerabilities.

The main objective of this system is to provide an intelligent and user-friendly platform that assists developers in identifying buggy or insecure code early in the development cycle. By automating the code review process and leveraging machine learning techniques, the proposed system enhances software quality, improves development productivity, and supports secure coding practices.

II. LITERATURE SURVEY

1. Automated Code Review Using Machine Learning

Authors: Rahul Sharma, Ankit Verma, and Pooja Singh

Abstract:

Automated code review systems have gained significant attention in modern software development due to the increasing complexity of software systems. This study proposes a machine learning-based approach for identifying potential bugs and code quality issues in source code. The system utilizes feature extraction techniques to convert code snippets into numerical vectors and applies classification algorithms to detect buggy code. Experimental results demonstrate that machine learning models can effectively identify code defects and assist developers in improving software quality. The study highlights the importance of integrating intelligent tools into the software development lifecycle to reduce manual review efforts and improve efficiency.

2. Machine Learning Techniques for Software Bug Prediction

Authors: Tianyi Zhang, David Lo, and Xin Xia

Abstract:

Software bug prediction has become an important research area in software engineering. This paper investigates the effectiveness of machine learning algorithms for predicting defects in source code modules. Various classifiers such as Support Vector Machine, Random Forest, and Naïve Bayes are applied to software datasets to identify patterns associated with buggy code. The results indicate that machine learning models can successfully predict defects and assist developers in prioritizing code inspection tasks. The study emphasizes the role of data-driven approaches in improving software reliability and reducing maintenance costs.

3. Static Code Analysis and Automated Bug Detection

Authors: Michael Johnson and Emily Carter

Abstract:

Static code analysis tools are widely used to detect errors and vulnerabilities during software development. This research examines different static analysis techniques and their effectiveness in detecting programming errors. The proposed system analyzes code structure and syntax patterns to identify potential issues before program execution. Although static analysis tools provide valuable insights, the study highlights their limitations in detecting logical errors and complex bug patterns. The paper suggests that integrating machine learning methods can enhance the capabilities of traditional static analysis tools.

4. Intelligent Code Review Systems for Software Quality Improvement

Authors: Li Wei, Chen Wang, and Jun Liu

Abstract:

Code review is a critical process in software engineering for ensuring code quality and reliability. This paper presents an intelligent code review framework that leverages machine learning algorithms to automate the detection of code defects. The system uses text-based feature extraction methods to analyze source code and classify it as clean or defective. Experimental results demonstrate that the proposed approach improves the efficiency of code review processes and reduces the burden on developers. The research highlights the potential of AI-driven solutions in enhancing collaborative software development.

5. Deep Learning Approaches for Detecting Software Vulnerabilities

Authors: Kevin Allamanis, Earl T. Barr, and Charles Sutton

Abstract:
The rapid growth of software systems has increased the need for automated vulnerability detection techniques. This paper explores the use of deep learning models for identifying vulnerabilities in programming code. The approach involves training neural networks on large datasets of source code to learn complex patterns associated with security risks. The results show that deep learning models can effectively identify vulnerabilities that traditional rule-based tools may overlook. The study demonstrates that AI-based techniques can significantly improve security analysis in modern software development.

III. EXISTING SYSTEM

In the current software development environment, code review and bug detection are primarily performed using manual inspection and traditional debugging tools. Developers usually analyze the source code themselves or rely on team-based peer reviews to identify logical errors, syntax issues, and potential vulnerabilities. Some static analysis tools and integrated development environments (IDEs) provide basic support for detecting syntax errors, code formatting issues, and limited security vulnerabilities. However, these systems generally rely on predefined rules and do not effectively learn from past code patterns. As a result, they may fail to detect complex bugs, logical errors, or emerging security risks. Furthermore, manual code review requires significant time and expertise, especially in large-scale software projects, which can slow down the development process and reduce productivity. Due to these limitations, existing systems are not always efficient in providing intelligent and automated code analysis.

IV. PROPOSED SYSTEM

In the current software development environment, code review and bug detection are primarily performed using manual inspection and traditional debugging tools. Developers usually analyze the source code themselves or rely on team-based peer reviews to identify logical errors, syntax issues, and potential vulnerabilities. Some static analysis tools and integrated development environments (IDEs) provide basic support for detecting syntax errors, code formatting issues, and limited security vulnerabilities. However, these systems generally rely on predefined rules and do not effectively learn from past code patterns. As a result, they may fail to detect complex bugs, logical errors, or emerging security risks. Furthermore, manual code review requires significant time and expertise, especially in large-scale software projects, which can slow down the development process and reduce productivity. Due to these limitations, existing systems are not always efficient in providing intelligent and automated code analysis.

V. SYSTEM ARCHITECTURE

The system begins with the User Registration and Login module, which allows users to create accounts and securely access the platform. This module ensures that only authenticated users can use the system features. After successful login, users interact with the application through the User Interface developed using the Flask Web Framework. This interface allows users to input or paste programming code that needs to be analyzed for bugs or vulnerabilities.

Once the code is submitted, it is sent to the AI Code Review System, which acts as the core component of the architecture. This module processes the code and prepares it for machine learning analysis. The system uses a Code Dataset that contains labeled examples of clean and buggy code. This dataset is used to train the machine learning models so that the system can learn patterns associated with errors and vulnerabilities.

The next stage involves TF-IDF (Term Frequency–Inverse Document Frequency) Vectorization, which converts the input code into numerical features. Since machine learning algorithms cannot directly understand raw text or code, TF-IDF helps transform the code into a mathematical representation that highlights important terms and patterns. This processed data is then used by the trained machine learning models.

The results of the analysis are stored in the SQLite Database, which maintains a history of all reviewed code along with the prediction results. This allows users to track previously analyzed code and review past results. Additionally, the system includes a Model Training and Evaluation module that periodically trains the models using the dataset and compares their performance. The results of this evaluation are displayed in the form of an Accuracy Report, helping users understand which algorithm performs best.

Finally, the system generates the output by classifying the submitted code as Clean Code if no issues are detected, or Buggy Code / Security Risk if the system identifies potential bugs or unsafe code patterns. This automated architecture helps developers quickly identify errors, improve code quality, and enhance software security during the development process.

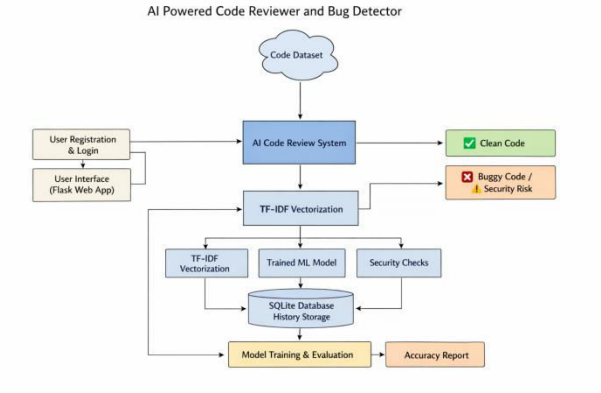

Fig 5.1: System Architecture

VI. IMPLEMENTATION

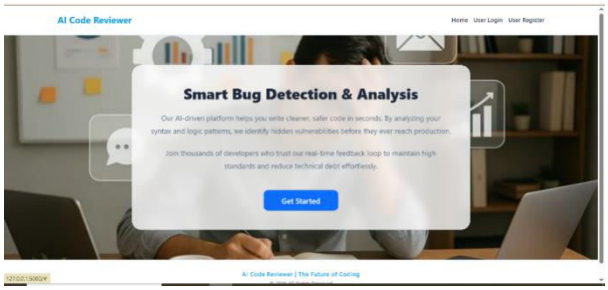

Fig 6.1: Home Page

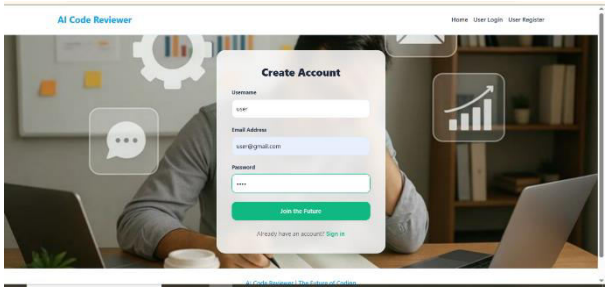

Fig 6.2: Create Account

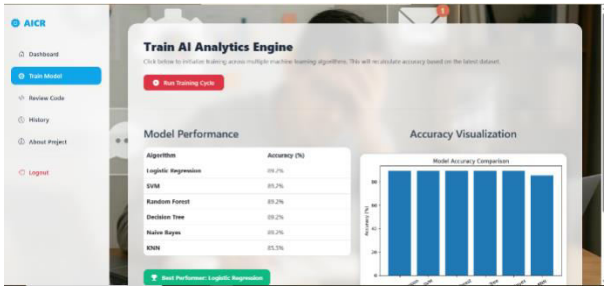

Fig 6.3: Train Ai Analytical Engine

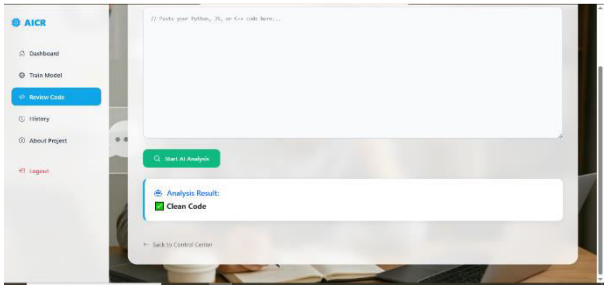

Fig 6.4: Output page

VII. CONCLUSION

The AI Powered Code Reviewer and Bug Detector system provides an intelligent approach to automatically analyze programming code and identify potential bugs and security risks. The system integrates machine learning techniques with a web-based application developed using the Flask framework to assist developers in improving code quality and reducing manual debugging efforts. By converting source code into numerical features using TF-IDF vectorization and applying multiple machine learning algorithms, the system can effectively classify code snippets as clean or buggy.

The implementation of various algorithms such as Logistic Regression, Support Vector Machine, Random Forest, Decision Tree, Naïve Bayes, and K-Nearest Neighbors allows the system to evaluate different models and select the best-performing one based on accuracy. This improves the reliability and efficiency of code analysis. Additionally, the system includes basic security checks to identify potentially dangerous functions such as eval() and exec(), which may introduce security vulnerabilities in software applications.

The developed system also provides user authentication, code submission, result visualization, and history tracking through a user-friendly web interface. By automating the code review process and

providing quick feedback to developers, the system helps reduce development time and improves software reliability. Overall, the proposed solution demonstrates how artificial intelligence and machine learning can enhance traditional code review practices and support developers in building more secure and high-quality software systems.

VIII. FUTURE SCOPE

The AI Powered Code Reviewer and Bug Detector system can be further enhanced by integrating advanced machine learning and deep learning techniques to improve the accuracy and efficiency of bug detection. In the future, larger and more diverse datasets of programming code can be used to train the models, allowing the system to recognize more complex bug patterns and programming errors. This will help the system provide more precise predictions and better assistance to developers during the coding process.

Another important improvement would be the integration of deep learning models and natural language processing techniques for more advanced code understanding. Modern AI techniques such as transformer-based models and code representation learning can help the system analyze complex code structures, logical errors, and dependencies within programs. This would enable the system to detect deeper issues that traditional machine learning models may not easily identify.

The system can also be expanded to support multiple programming languages, such as Java, C++, JavaScript, and others. Currently, the system focuses mainly on analyzing code patterns from the dataset used during training. By extending the dataset and improving the feature extraction process, the system can become a multi-language code review platform that supports developers working with different technologies.

In addition, the system can be integrated with Integrated Development Environments (IDEs) such as Visual Studio Code, PyCharm, or Eclipse. This integration would allow developers to receive real-time bug detection and security warnings directly while writing code. Such real-time feedback would significantly improve development productivity and reduce the time spent on debugging.

Another potential enhancement is the implementation of advanced security vulnerability detection. The system could be expanded to identify various security issues such as SQL injection risks, unsafe input handling, memory vulnerabilities, and insecure coding practices. By incorporating security-focused datasets and models, the system could serve as both a code reviewer and a security analysis tool.

Finally, the system can be improved by implementing cloud-based deployment and scalability features. Hosting the application on cloud platforms would allow multiple users to access the system simultaneously and enable large-scale model training and analysis. With further development, the system could evolve into a complete intelligent code review platform that supports collaborative software development and automated quality assurance.

IX. REFERENCES

[1] T. Zhang, D. Lo, and X. Xia, “An Empirical Study of Machine Learning Algorithms for Bug Prediction,” IEEE Transactions on Software Engineering, vol. 44, no. 5, pp. 1–15, 2018.

[2] K. Allamanis, E. T. Barr, C. Bird, and C. Sutton, “Learning Natural Coding Conventions,” in Proceedings of the ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2014, pp. 281–293.

[3] M. Fowler, Refactoring: Improving the Design of Existing Code, Boston, MA, USA: Addison-Wesley, 2018.

[4] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning, New York, NY, USA: Springer, 2009.

[5] J. Han, M. Kamber, and J. Pei, Data Mining: Concepts and Techniques, 3rd ed., San Francisco, CA, USA: Morgan Kaufmann, 2012.

[6] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 3rd ed., Upper Saddle River, NJ, USA: Prentice Hall, 2016.

[7] F. Pedregosa et al., “Scikit-learn: Machine Learning in Python,” Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[8] M. Lutz, Learning Python, 5th ed., Sebastopol, CA, USA: O’Reilly Media, 2013.

[9] M. Grinberg, Flask Web Development: Developing Web Applications with Python, Sebastopol, CA, USA: O’Reilly Media, 2018.

[10] I. Sommerville, Software Engineering, 10th ed., Boston, MA, USA: Pearson Education, 2016.