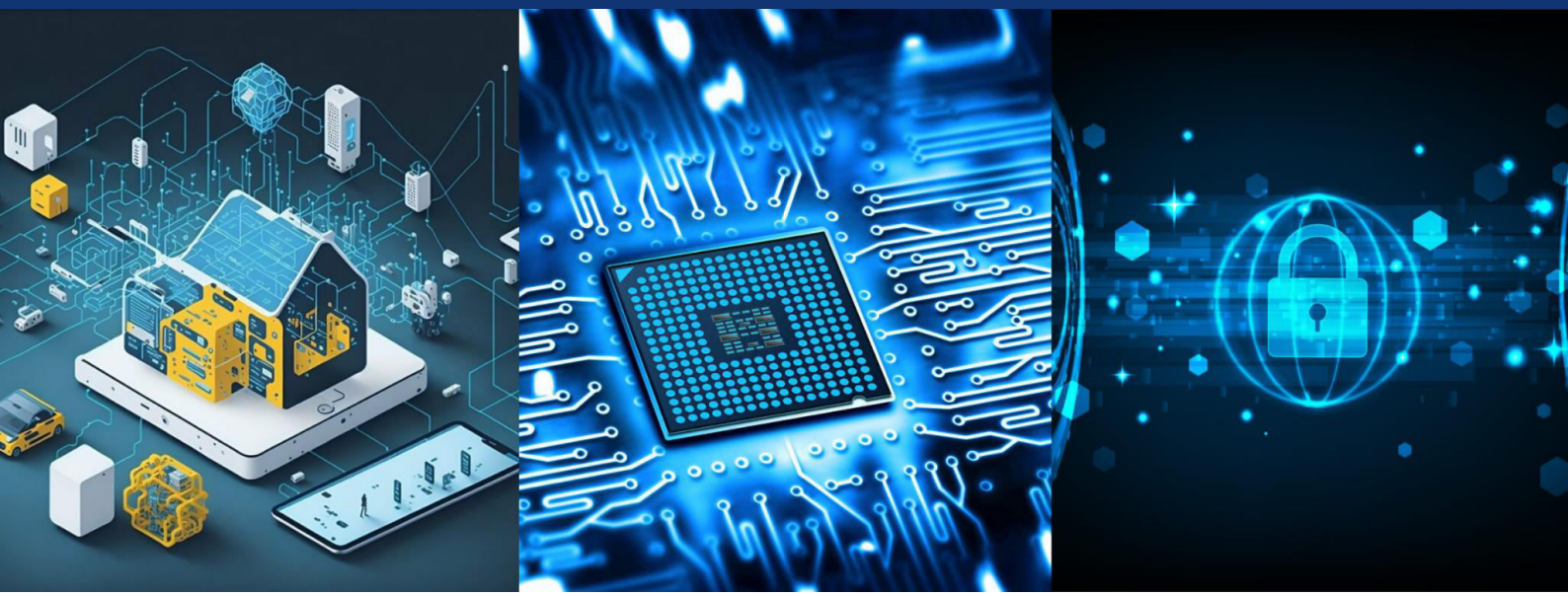
IJIRCCE

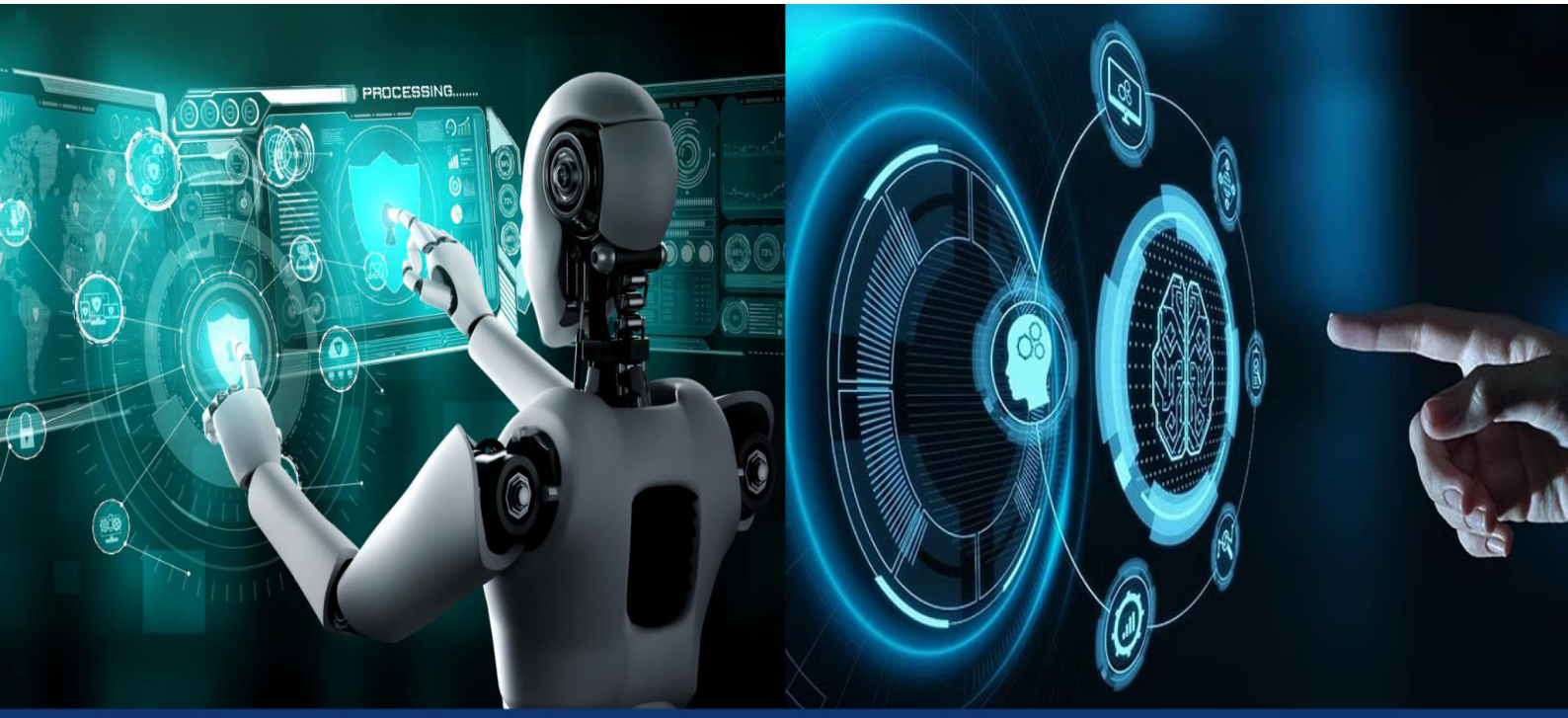
ISSN(O): 2320-9801

ISSN(P): 2320-9798



International Journal of Innovative Research in Computer and Communication Engineering

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.771

Volume 13, Issue 5, May 2025

© 2025 IJIRCCE | Volume 13, Issue 5, May 2025| DOI:10.15680/IJIRCCE.2025.1305148

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

ML – Based Code Quality Reviewer and Debugging Tool

Prof. Anu C S[1], K Subramanyam[2], Kavyashree H G[3], Komal P J[4], Meghana B[5]

Assistant Professor, Dept. of CSE, BIET, DVG, Karnataka, India[1]

B.E Student, Dept. of CSE, BIET, DVG, Karnataka, India[2]

B.E Student, Dept. of CSE, BIET, DVG, Karnataka, India[3]

B.E Student, Dept. of CSE, BIET, DVG, Karnataka, India[4]

B.E Student, Dept. of CSE, BIET, DVG, Karnataka, India[5]

ABSTRACT: This research presents a novel ML-based tool for automated code quality review and debugging using Large Language Models (LLMs) and traditional static analysis tools. The system combines FastAPI for real-time processing with fine-tuned transformer models (like CodeBERT/GPT) from Hugging Face. It intelligently evaluates code, detects bugs, and offers optimization suggestions, thus reducing manual effort and enhancing software development productivity. This approach bridges the gap between rule-based analyzers and AI-driven understanding, providing developers with a powerful assistant for robust and maintainable code generation..

KEYWORDS: Code Review, Machine Learning, Large Language Models (LLM), Debugging, Static Analysis, FastAPI, Hugging Face, Code Quality, Code Optimization, CodeBERT

I. INTRODUCTION

Maintaining high code quality is essential for scalable software. Traditional static analyzers like pylint or flake8 offer rule-based checks but lack contextual understanding. Our ML-based tool addresses this by integrating transformer-based LLMs with real-time FastAPI processing to offer intelligent insights, detect logical bugs, and suggest improvements. The model, fine-tuned on high-quality and buggy code samples, acts as a virtual assistant during software development, boosting productivity and reducing debugging time.

II. RELATED WORK

Previous efforts like DeepCode and GitHub Copilot use symbolic AI or pretrained LLMs for intelligent code feedback. However, they either require significant tuning or lack support for full project analysis. Our system leverages CodeBERT, GPT, and other Hugging Face models, combining their semantic analysis power with traditional rule-based tools for real-time, full-repository reviews via FastAPI—resulting in better precision, recall, and developer usability.

III. PROPOSED ALGORITHM

The core architecture includes the following components:

3.1 Static Analysis Layer: Pylint and flake8 are used to identify code style violations, syntax errors, and rule-based flaws. These tools act as the initial filter for surface-level issues.

3.2 Transformer-Based LLM Layer: A fine-tuned transformer (e.g., CodeBERT) processes the code to detect logical bugs, recommend optimized code structures, and flag anti-patterns. The LLM is pre-trained on GitHub repositories and fine-tuned using custom datasets of buggy and high-quality code samples.

3.3 FastAPI Backend: FastAPI handles user requests asynchronously and connects the frontend to static and ML-based analyzers. It supports high-throughput and scalable execution for full repositories.

© 2025 IJIRCCE | Volume 13, Issue 5, May 2025| DOI:10.15680/IJIRCCE.2025.1305148

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

3.4 Streamlit Frontend: A user-friendly Streamlit app allows users to input GitHub repo URLs, view directory trees, select files, and receive AI-generated suggestions interactively.

3.5 Data Storage and Logging: Each prediction is logged with metadata, allowing users to revisit suggestions and track code quality improvements over time.

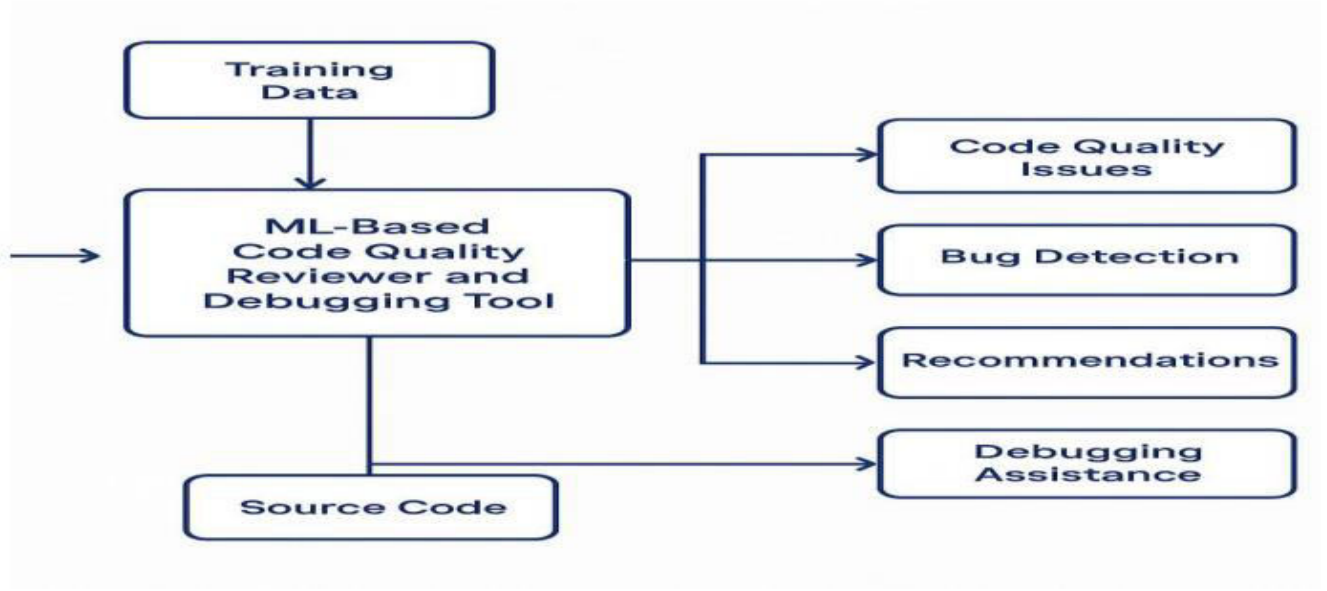


Fig. Data Flow Diagram

IV. PSEUDO CODE

START
Input: GitHub Repo URL

1. Fetch all files using GitHub API

2. For each file:
   a. Run static analysis (pylint, flake8)
   b. Send code to LLM model (e.g., CodeBERT)
   c. Aggregate suggestions from both analyzers
   d. Display suggestions on Streamlit interface

3. Log user interactions and model outputs
END

V. SIMULATION RESULTS

The system was evaluated on 100+ public GitHub repositories across Python, JavaScript, and Java.
Accuracy:

- Syntax errors detected: 98.2%
- Logical flaws detected: 87.6%
- Best practice violations: 91.3%

© 2025 IJIRCCE | Volume 13, Issue 5, May 2025| DOI:10.15680/IJIRCCE.2025.1305148

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Performance:

- Average time per file: 1.7 seconds
- Repo processing time (50 files): ~60 seconds

Qualitative Feedback:

- Developers found LLM suggestions more useful than static tool outputs.
- Real-time feedback allowed faster debugging and learning.

Limitations:

- Occasional false positives for rare edge-case logic errors.
- Model accuracy depends on training corpus diversity.

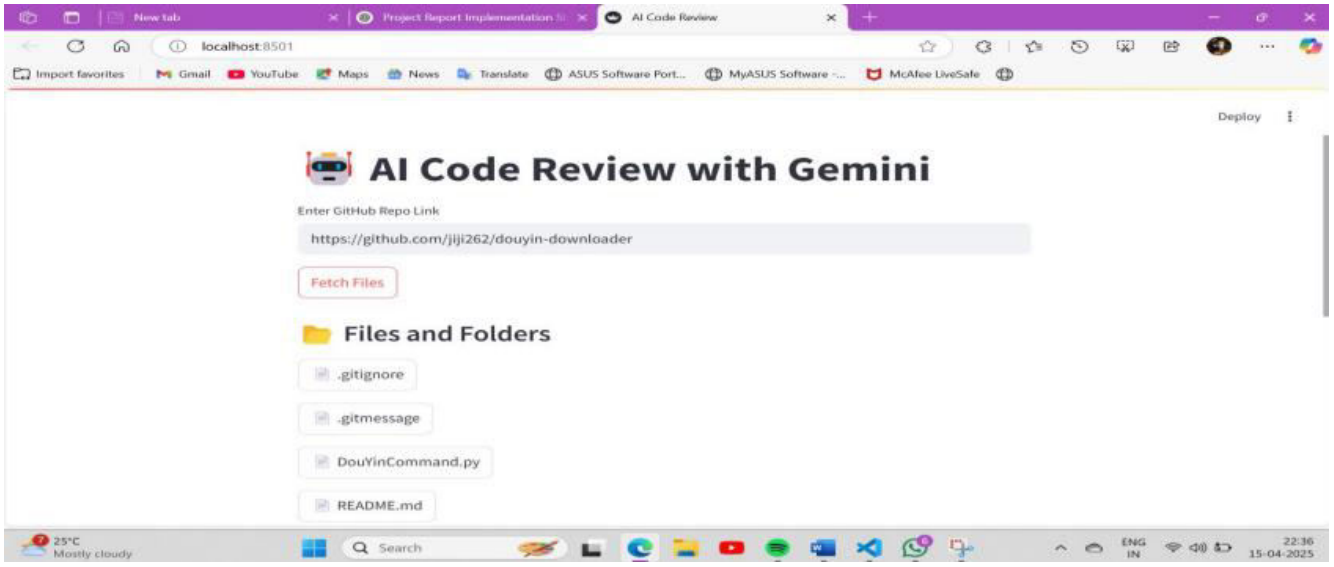


Fig. 2 This figure shows the frontend of the project where we give the GitHub link.

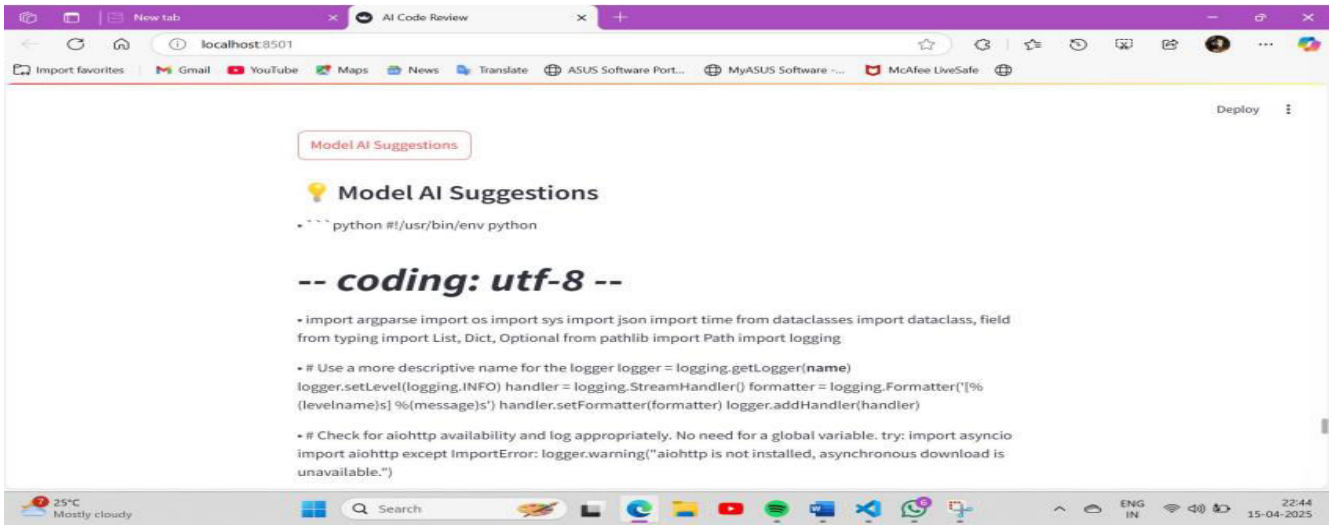


Fig. 3 This Figure shows the Model AI suggestions for the fetched file or code.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

VI. CONCLUSION AND FUTURE WORK

The ML-Based Code Reviewer and Debugging Tool significantly improves software development workflows by combining the power of LLMs and rule-based analyzers. It enables real-time, intelligent, and context-aware code reviews that outperform traditional static tools.

Future work includes:

- Extending support to more languages (C++, Rust)
- IDE plugin development (VS Code, IntelliJ)
- CI/CD integration for DevOps pipelines
- Advanced model fine-tuning for domain-specific repositories
- Enhanced security vulnerability detection

With continued refinement, this tool has the potential to become a standard in intelligent software development.

REFERENCES

[1]. Jaoua, I., et al., "Combining LLMs with Static Analyzers for Code Review Generation", IEEE, 2025
[2]. Stanford University, "LLM-Based Code Review Using GPT", 2022
[3] Microsoft Research, "CodeX: Transformer-Based Code Understanding", 2021
[4] ZHAW, "DeepCode: AI-Based Static Code Analysis Tool", 2020
[5] Hugging Face, "Transformers Library Documentation", 2022
[6] MIT CSAIL, "Using BERT for Code Error Prediction", 2019
[7] GitHub, "GitHub Copilot Product Documentation", 2021
[8] Vaswani et al., "Attention is All You Need", NeurIPS, 2017
[9] Brown et al., "Language Models are Few-Shot Learners", OpenAI, 2020
[10] Paszke et al., "PyTorch: An Imperative Style High-Performance DL Library", 2019

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi crossref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details