# Smart Git Repo Debugger with AI Powered Reviews

**Dr. P. Sumathi**
*Head of the Department,*
*Department of Information Technology*
*SNS College of Technology*
Coimbatore, India
psumathi.it@gmail.com

**Kolaa Pravin**
*Student,*
*Department of Information Technology*
*SNS College of Technology*
Coimbatore, India
kolaapravin@gmail.com
ORCID: 0009-0006-6165-3154

**Subhashith V**
*Student,*
*Department of Information Technology*
*SNS College of Technology*
Coimbatore, India
subhashithv@gmail.com

**Priyadharshini K**
*Student,*
*Department of Information Technology*
*SNS College of Technology*
Coimbatore, India
priyadharshini7324@gmail.com

**Sri Varun N.S**
*Student, Department of Information Technology*
*SNS College of Technology*
Coimbatore, India
srivarunss04@gmail.com

***ABSTRACT:*** ***In today's fast-moving software world, writing clean and error-free code is very important, especially for developers and students working on large projects. This paper introduces an AI-powered system that reviews and debugs code from a GitHub repository link. Instead of uploading zip files, users can simply provide a repository URL. The system will automatically clone the repository, scan all code files, and generate a report with issues like syntax errors, logical bugs, code quality problems, and security concerns. It also suggests possible fixes. Additional features like GitHub login, email authentication, and password recovery with OTP are included. An AI-based formatter reads the coding style of the repo and formats the code accordingly. The tool uses technologies like ReactJS, NodeJS, MongoDB, Python, and Gemini API. The main goal is to help both freshers and experienced developers by saving time, reducing manual errors, and improving code quality in a simple and efficient way.***

***Keywords: Code Review, GitHub Integration, Bug Detection, AI Debugging, Code Quality, Syntax Errors, Gemini API, AI Code Formatter.***

## I. INTRODUCTION

In recent times, software development has become an essential part of almost every industry. With more people building applications, the demand for clean, error-free, and well-formatted code has grown. Many developers, especially beginners, face challenges in identifying bugs, syntax errors, and code quality issues in their projects. Manually reviewing code line by line takes time and often leads to missed errors. To solve this issue, we have developed an AI-powered code review and debugging system that works directly with GitHub repositories. Instead of uploading files manually, users can simply enter the GitHub link of their project. The system will automatically clone the repository, read through the files, and identify problems in the code. It also provides suggestions to fix those issues in a simple and understandable way. The tool also includes user-friendly features like GitHub login, email-based sign-up and login, and OTP-based password reset. It supports an AI-based formatter that learns the existing code style and formats other files accordingly. The full system is built using modern technologies such as ReactJS, NodeJS, MongoDB, and Python, with the Gemini API used for AI-based suggestions. This project mainly focuses on saving time, reducing errors, and improving code quality, especially for students, freshers, and developers who want to maintain clean and secure code without spending hours on debugging manually.

## II. RELATED SYSTEM

In the past few years, many tools have been introduced to help with code review and debugging. Platforms like SonarQube and ESLint are widely used in the industry to find code quality issues and enforce coding standards. SonarQube supports multiple languages and shows bugs, code smells, and security vulnerabilities, but it usually requires proper setup and is more suitable for large teams. ESLint is mostly used for JavaScript and helps in identifying syntax and style problems, but it does not support deep-level bug fixing or logic-based suggestions. GitHub itself provides code review features where team members can comment on pull requests. However, this is a manual process and depends on the reviewer's experience.

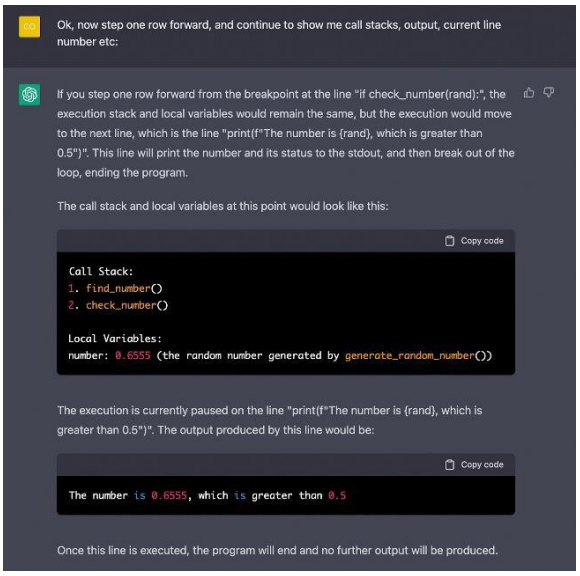


*Fig 1. ChatGPT Interface – The most used AI in recent days*

Some AI-based tools like DeepCode and Codacy have also entered the market, offering suggestions using machine learning, but most of them are paid or limited in features for free users. Unlike the above systems, our project offers a free and user-friendly solution for students and small teams. It accepts a GitHub repository link and performs automatic scanning of the full codebase, detects various errors, and gives suggested fixes. It also supports AI formatting by learning the structure of the code. Our system brings all these features together in one place with a simple interface and works well even for beginners with limited technical knowledge.

## III. PROPOSED SYSTEM

The proposed system is designed to make the code review and debugging process faster, simpler, and smarter with the help of artificial intelligence. In traditional methods, users often upload zip files or copy-paste large blocks of code to get them reviewed. This is time-consuming, has upload limits, and often causes confusion. In this system, that entire process is replaced with just one step – the user simply needs to paste the GitHub repository URL. Once the link is submitted, the system will automatically clone the entire project using Git commands and start working on it instantly. After the repository is cloned, the system goes through each file in the project. It scans the code line by line and checks for different types of issues. These include syntax errors (like missing brackets, wrong function calls), logical bugs (like incorrect loops or unreachable code), and even security issues such as hardcoded credentials, SQL injection risks, or unsafe imports. The tool uses static analysis techniques and also integrates with the Gemini AI API to detect deeper problems which may not be visible through normal scanning. After scanning, a detailed report is prepared with the file name, exact line number of the issue, type of error, and a simple suggestion on how to fix it. This report is shown in a clean UI where users can easily navigate and understand each issue, even if they are not highly experienced. In addition to error detection, the system has another useful feature – an AI-based code formatter. It reads the existing formatting style used across the project (for example, spacing, naming conventions, comment styles) and applies the same format to all the other files. This ensures that the entire codebase looks clean, consistent, and professional. This feature is especially helpful in group projects where different developers may follow different writing styles.

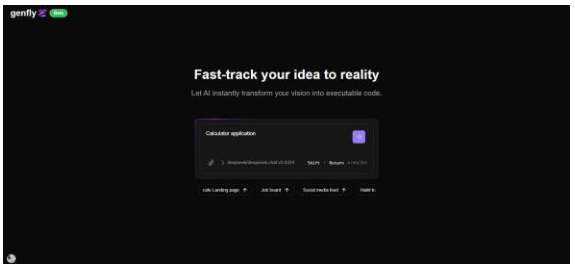


*Fig 2. The homepage of the application, where users create applications in their preferred format*

With this formatter, the final output looks uniform and easy to maintain. The system also provides secure user management. All data is saved securely in MongoDB. This includes user login details, project history, and scan results. Each user's data is kept private and safe using backend access control and secure API routing. The frontend is developed in ReactJS, which gives a fast and responsive experience. Gemini API plays a major role in understanding the code and suggesting human-like responses for errors and fixes. This tool is mainly developed for students, freshers, and small developer teams who want a quick and intelligent way to review their code. With this system, they can instantly find their mistakes and learn from the suggestions. It also helps in reducing human error and saves a lot of time when compared to manual reviews. Overall, the system brings a complete package of automation, intelligence, and usability that improves code quality and makes the development process smoother and more efficient.

## IV. METHODOLOGY

The system is developed with a full-stack architecture using modern web technologies and AI integration. The flow starts when the user enters a GitHub repository URL. The backend, built using Node.js and Express.js, uses a Git library to clone the repository into the server. Once the files are extracted, a file scanner goes through each file based on its extension, such as .js, .py, .java, etc.

The scanner checks for syntax errors using language-specific parsers. For logical and security issues, the code is passed to a Python module which uses custom rules and the Gemini AI API. The AI model analyses the code and returns suggestions for fixing detected problems. These suggestions are displayed in the frontend with proper line numbers and explanations.

For formatting, the system first observes the structure and style of the existing code in the repo. Based on indentation, naming conventions, and spacing, the AI formatter adjusts the remaining code to follow the same pattern. This keeps the code clean and consistent.

User authentication is handled with email/password or GitHub login using OAuth. Password recovery works through OTP sent to the user's email. The frontend is made using React.js and styled with Tailwind CSS and Framer Motion for smooth transitions. All the project data, user login info, and reports are stored in MongoDB.

The overall system is designed to be scalable, secure, and easy to use. Each part of the flow is modular, so future improvements like multi-language support or deeper code analysis can be added easily.

The system is designed with five main modules. Each module handles a specific task and works in coordination with the others to make the system function smoothly. Below is a detailed explanation of each module.

***Module 1: GitHub Repository Integration Module***

This is the first and most important module. Instead of the traditional method of uploading zip files, the user is asked to paste the GitHub repository URL. Once the link is submitted, the backend uses Git commands to clone the repository into the server environment. After cloning, the files are separated and stored in temporary folders for processing. This module checks for valid links, handles errors like private repo access, and ensures that only supported file types are considered for scanning. This approach saves time, removes upload limitations, and allows users to work directly with live codebases.

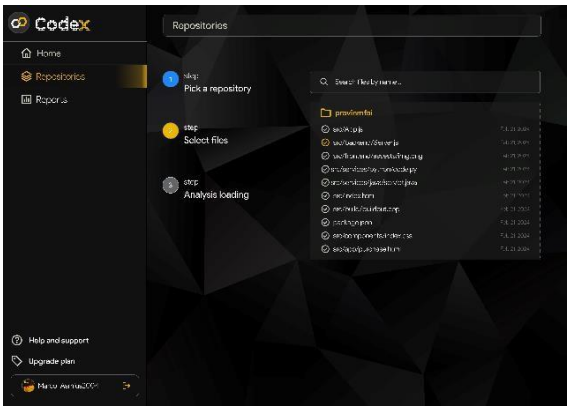


*Fig 3. The repositories adding screen in the application, where users are prompted to sync the repositories in their account*

***Module 2: Code Scanner and Error Detection Module***

Once the files are cloned, this module begins scanning each code file. It supports multiple programming languages like JavaScript, Python, and Java. The scanner first performs static analysis to detect basic syntax errors such as missing brackets, undefined variables, or improper loops. After that, the files are passed through a Python-based checker that looks for deeper issues like logical bugs and possible security threats such as code injection or bad practice usage. For even better analysis, the content is also sent to the Gemini AI API, which returns suggestions in natural language. The system matches these with the file and line number, storing them for reporting.

***Module 3: AI-Powered Code Formatter Module***

This module improves the readability and consistency of the code. When the system clones the repository, it observes the formatting style followed in the project. It checks for indentation (tabs or spaces), line spacing, function naming styles (camelCase or snake_case), and comment alignment. Using this pattern, it formats other parts of the code that may not follow the same standard. The formatter uses Gemini API to understand the structure and apply formatting in a smart way. This is especially useful in group projects where multiple developers may follow different styles. The output is a clean and consistent codebase that looks professional.

***Module 4: User Authentication and Security Module***

Security is important when working with user data and code. This module handles user login, signup, and password recovery. Users can register using email and password or choose GitHub OAuth login for faster access. Passwords are encrypted and stored securely in MongoDB. In case a user forgets the password, the system sends a one-time password (OTP) to the registered email for verification. The module also protects all routes using access tokens and session checks to avoid unauthorized use. User-specific project data and reports are stored securely and only accessible by the respective user.

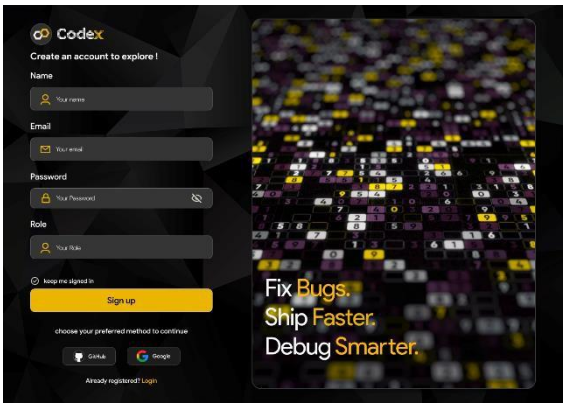


*Fig 4. The signup page of the application, where users signup using their Google/Git/Email IDs*

***Module 5: Error Report and UI Display Module***

After scanning and formatting are completed, this module presents the final output to the user. The report shows each error with file name, line number, error type, and suggested fix. The user can navigate through each file and see the problem highlighted directly. The frontend is developed using ReactJS and styled with Tailwind CSS to make it modern and responsive. Framer Motion is used to add smooth transitions and animations, giving the tool a polished look. This module ensures that even a beginner can understand the error and correction without needing deep technical knowledge.

## V. System Architecture

The architecture of this project follows a modular and layered design, combining both frontend and backend technologies with AI integration. The system mainly has four layers: the user interface layer, the server-side logic layer, the AI and processing layer, and the database layer. Each component communicates with others through secure APIs and structured data flow.

**1. User Interface Layer**

This is the frontend of the system where users interact. It is built using ReactJS for a fast and responsive experience. Users can paste their GitHub link, view scanned results, and navigate through files and error reports. The UI also handles login, registration, and password recovery. Tailwind CSS is used for styling, and Framer Motion is used to add smooth animations, giving a modern look and feel to the tool.

**2. Server-Side Logic Layer**

This layer is built using Node.js and Express.js. It handles incoming API requests from the frontend, such as cloning repositories, scanning files, and sending them for processing. This layer also validates GitHub links, manages authentication logic, and handles communication between the database and AI services. It acts as the main controller of the entire system.

**3. AI and Code Processing Layer**

Once the repository is cloned, this layer takes over. It includes a Python-based script that performs file-level scanning using static analysis techniques. It checks for syntax, logic, and security issues. Then, the system sends code snippets to the Gemini AI API for advanced review and suggestion generation. It also includes the AI-based code formatter, which reads the project's existing coding style and formats the rest of the code accordingly. This layer is responsible for making the tool intelligent and more helpful than manual reviewers.

**5. Integration Flow**

The flow starts when a user pastes a GitHub link on the frontend. The request goes to the backend, where the repository is cloned. Files are scanned and processed, results are saved in MongoDB, and then displayed back on the frontend in an organized report format. All communications are done through secure REST APIs, ensuring performance and security.

**DATAFLOW DIAGRAM:**

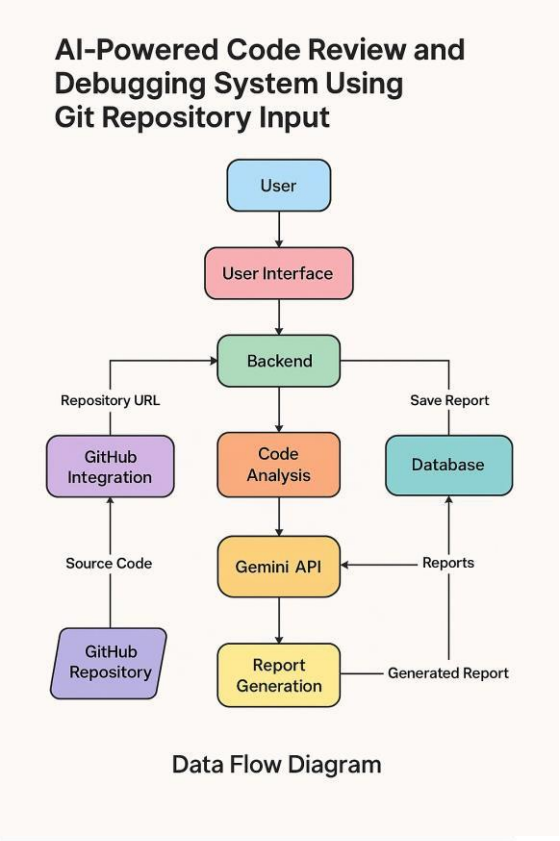


*Fig 5. A flow diagram showing the working of the project*

## VI. CONCLUSION

The main aim of this project was to make code review and debugging simple, fast, and beginner-friendly using artificial intelligence. Many students and fresh developers struggle to find mistakes in their code, especially when working in teams or large files. This system solves that problem by allowing users to just paste a GitHub repository link. From there, the tool handles everything automatically – cloning the repo, scanning all files, finding syntax and logic errors, and even formatting the code in a clean and consistent way.

The use of the Gemini API brings in smart suggestions that are often as good as what a human reviewer would say. Along with this, the OTP-based login, GitHub authentication, and a clean frontend make the tool easy to use for anyone. It saves time, avoids manual checking, and improves the overall quality of software development. This system is especially useful for final-year students, new developers, and small teams who cannot afford costly tools or do not have senior developers for reviews. It is also built in a way that more features can be added in the future without disturbing the existing flow.

In short, the project brings together automation, AI, and clean design to offer a complete code review platform that is both powerful and user-friendly.

## VII. REFERENCES

[1] A. Kumar, B. Singh, and S. R. R. K. Varma, "AI-Based Code Review System: A Survey," International Journal of Computer Applications, vol. 184, no. 6, pp. 50–55, Jul. 2024.
[2] S. Sharma, R. Gupta, and P. Shah, "Improving Code Quality Using AI and Static Analysis Tools," Journal of Software Engineering, vol. 22, no. 4, pp. 180–192, Dec. 2024.
[3] M. Patel and D. K. Verma, "Automation of Code Reviews Using Deep Learning Techniques," IEEE Access, vol. 11, pp. 42312–42325, Apr. 2023. doi: 10.1109/ACCESS.2023.3267812.
[4] J. K. Lee and H. S. Kim, "Smart IDE Plugin for Real-Time Bug Detection Using Machine Learning," in Proc. Int. Conf. Artificial Intelligence and Software Engineering (ICAISE), Singapore, 2023, pp. 89–94.
[5] SonarSource, "SonarQube: Continuous Inspection of Code Quality," [Online]. Available: https://www.sonarqube.org
[6] GitHub, "Cloning a Repository," [Online]. Available: https://docs.github.com/en/repositories/creating-and-managing-repositories/cloning-a-repository.
[7] Gemini AI, "Gemini API Documentation," [Online]. Available: https://gemini.google.com/api
[8] DeepSource, "DeepSource: Automated Code Review with Static Analysis," [Online]. Available: https://deepsource.io