IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 10, October 2025

DOI: 10.17148/IJARCCE.2025.141030

# AI Code Analyzer Agent

**Mr.Vivek Dinesh Patil[1], Prof. Kaustubh Bhave[2], Prof. Manoj V Nikum[*3]**

Student of MCA, SJRIT Dondaicha, KBC NMU Jalgaon, Maharashtra[1]

Assistant Professor, MCA Department, SJRIT Dondaicha, Jalgaon,Maharashtra [2]

HOD, MCA Department, SJRIT Dondaicha, Jalgaon, Maharashtra[*3]

**Abstract**: The **AI Code Analyzer Agent** is a full-stack web application developed using the **MERN stack (MongoDB, Express.js, React.js, Node.js)** integrated with **Google Gemini**, an advanced generative AI model. This system acts as an intelligent assistant that analyzes user-submitted code, identifies potential syntax and logical errors, and provides improvement suggestions. The AI feedback is formatted and rendered using Markdown for readability, while syntax highlighting is achieved using **PrismJS** for enhanced visual clarity.

The system architecture includes a responsive frontend, secure backend communication, and a scalable database design. The application allows users to interact with AI in real time, receiving educational and practical insights to improve their coding skills.

The project showcases how artificial intelligence can be integrated into traditional web-based development workflows to create intelligent, user-friendly, and scalable solutions.

**Keywords:** AI Code Review, MERN Stack, Google Gemini, Syntax Highlighting, Code Optimization.

## I. INTRODUCTION

In today's fast-paced software development world, ensuring the quality, maintainability, and efficiency of code has become more important than ever. As applications grow more complex, even experienced developers often overlook errors, redundant code, or inefficient logic. Traditional static analysis tools like ESLint or SonarQube can detect syntactical and formatting errors but lack deep contextual understanding.

To address this limitation, artificial intelligence (AI) has emerged as a powerful solution capable of understanding, reasoning, and improving source code beyond conventional methods. The **AI Code Analyzer Agent** is an intelligent web-based system built using the **MERN (MongoDB, Express.js, React.js, Node.js)** stack, integrated with **Google Gemini AI**, a state-of-the-art generative model.

The system allows users to input code in an online editor, which is then analyzed by Google Gemini. The AI identifies potential errors, logic flaws, and inefficiencies, and provides meaningful suggestions for optimization. By combining modern full-stack web technologies with cutting-edge AI, the application offers a unique and user-friendly way to enhance code quality and learning.

## II. BACKGROUND AND MOTIVATION

In software development, writing clean, efficient, and bug-free code is essential to ensure maintainability and reliability. Over the years, numerous **code analysis tools** have been developed to assist programmers in identifying issues early in the development process. However, traditional tools often rely on static rule-based mechanisms that detect syntactic or structural errors without understanding the deeper **semantic meaning** of the code.

With the advent of **Artificial Intelligence (AI)** and **Natural Language Processing (NLP)**, the landscape of software engineering has changed significantly. AI-driven tools are now capable of **understanding code contextually**, suggesting improvements, and even generating code automatically.

This chapter explores the **background concepts**, **technologies used**, and **related systems** that have inspired and influenced the development of the **AI Code Analyzer Agent**.

The **MERN stack's scalability**, the system aspires to redefine the way code review and optimization are approached in modern software development.

### Background

Code review is a systematic examination of source code to identify errors, bugs, and areas for improvement. Traditionally, this process is carried out manually by developers or peers, which can be **time-consuming** and **error-prone**. Automated code analysis tools emerged to assist developers by scanning the code and detecting common issues, thereby improving productivity and software quality.

  

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 10, October 2025
DOI: 10.17148/IJARCCE.2025.141030

Code analysis is generally categorized into two types:

- **Static Analysis:** Examines code without executing it. Examples include ESLint, SonarQube, and PMD.
- **Dynamic Analysis:** Involves executing the code and monitoring behavior during runtime.

While these approaches improve quality assurance, they lack adaptability and fail to understand **developer intent** or **program logic** beyond predefined patterns.

**Related Work:**

**Traditional Tools**

1. **ESLint:**
   A popular open-source tool for identifying and fixing problems in JavaScript code. It enforces coding standards but lacks semantic understanding.
2. **SonarQube:**
   A robust platform for continuous inspection of code quality. It detects bugs, code smells, and vulnerabilities but cannot explain *why* an issue exists.
3. **PyLint / PMD / Checkstyle:**
   Static analyzers for Python and Java that help enforce coding standards. However, they rely on rigid rule sets and don't provide human-readable insights.

**AI-Based Tools**

1. **GitHub Copilot (OpenAI Codex):**
   Offers real-time code suggestions based on context. While useful, it focuses more on code generation than analysis.
2. **ChatGPT / Gemini / Bard:**
   Large Language Models capable of understanding and generating natural language and code. These tools can explain, debug, and optimize code, though they are not designed as full-stack integrated applications.
3. **Kite:**
   An AI coding assistant that helps with autocompletion but offers limited analytical capabilities

## III. CONCLUSION REMARKS AND FUTURE WORK

The rapid advancement of **artificial intelligence (AI)** has revolutionized nearly every domain, including software development and code analysis. The **AI Code Analyzer Agent** was developed to integrate the power of **AI-based reasoning** into a **web-based full-stack platform**.

This chapter concludes the report by summarizing the major findings, outcomes, and lessons learned during the development and implementation of the system. It also outlines the potential areas for improvement and future expansion.

**Advantages of the System**

The **AI Code Analyzer Agent** offers several benefits over existing static or AI-assisted tools:

| Advantage | Description |
|---|---|
| **AI-Driven Feedback** | Provides context-based, intelligent suggestions instead of rule-based error lists. |
| **Real-Time Processing** | Delivers instant AI feedback without requiring local setup. |
| **User-Friendly Interface** | Offers a clean, minimalistic design built for ease of use. |
| **Cross-Platform Access** | Can be used from any modern browser, on desktop or mobile. |
| **Educational Value** | Acts as an AI tutor, explaining why errors occur and how to fix them. |
| **Scalable Infrastructure** | Built using modular MERN stack, enabling future feature extensions. |

**CONCLUSION**

The **AI Code Analyzer Agent** successfully achieves its objective of providing an **intelligent, full-stack AI-driven code review system**. By leveraging the MERN stack and Google Gemini's generative capabilities, it delivers a seamless, context-aware analysis experience within a browser interface.

The system demonstrates how **AI and modern web technologies can merge** to build tools that not only detect coding errors but also teach better coding practices. It bridges the gap between automation and human reasoning, serving as both a **development aid** and a **learning platform**.

The success of this project confirms that AI-driven tools will continue to play a transformative role in the **future of software engineering**, enabling faster development, cleaner code, and smarter debugging.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 10, October 2025
DOI: 10.17148/IJARCCE.2025.141030

**Future Enhancements**

The **AI Code Analyzer Agent** is designed with scalability in mind. The following enhancements are planned for future versions:

**Multi-Language Support:**
Extend support to programming languages such as Python, Java, C++, and TypeScript.

**User Authentication:**
Add login and registration functionality using JWT (JSON Web Tokens) for personalized experiences.

**Code Auto-Correction:**
Implement an auto-fix feature that allows users to directly apply AI-recommended changes.

**Integrated IDE Features:**
Include live debugging tools, file uploads, and execution options.

**Analytics Dashboard:**
Display insights such as error frequency, coding style trends, and improvement tracking.

**Offline Mode:**
Explore possibilities of caching Gemini responses or local AI inference for offline use.

**Mobile Application:**
Develop an Android/iOS app using React Native for on-the-go accessibility.

**Collaborative Review System:**
Allow multiple users to share and review code collaboratively in real time.

## IV. VALIDATION AND RESULTS

Validation and testing are essential stages in the software development lifecycle that ensure the system meets its functional and non-functional requirements. The **AI Code Analyzer Agent** was validated through a series of **functional, integration, and performance tests** to verify its accuracy, stability, usability, and efficiency.
This chapter presents the testing methodologies, test cases, results obtained, and performance evaluation metrics for the developed system. The goal of this validation process was to ensure that the system performs reliably across various programming inputs and real-world usage scenarios.

**Objectives of Validation**

The main objectives of system validation are:

1. To verify that the system functions according to the specified requirements.
2. To ensure the AI-generated feedback is contextually accurate and meaningful.
3. To test the integration between frontend, backend, database, and AI API.
4. To evaluate system performance in terms of response time and scalability.
5. To ensure the user interface is intuitive and user-friendly.

**Validation Methodology**

The following validation approaches were used:

**1. Functional Testing**

Ensures that each feature performs as intended.

- Code input and submission functionality
- Communication with backend server
- AI response retrieval and display
- Syntax highlighting and Markdown rendering

**2. Integration Testing**

Verifies the seamless communication between system components — frontend, backend, database, and AI API.

**3. Performance Testing**

Analyzes the response time and stability under varying loads.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 10, October 2025

DOI: 10.17148/IJARCCE.2025.141030

**4. Usability Testing**
Measures how easily users can interact with the platform, focusing on readability, design, and accessibility.
**5. Security Testing**
Ensures safe API communication and data handling using **CORS** and secure POST requests.

## V. PROPOSED SOLUTION

The **AI Code Analyzer Agent** has been developed to overcome the limitations of traditional rule-based code analyzers by integrating **artificial intelligence (AI)** with a **modern web-based full-stack architecture (MERN Stack)**.
The system provides a **smart, interactive, and real-time platform** for analyzing and improving source code using **Google Gemini**, a powerful generative AI model.
The proposed solution delivers intelligent code analysis, syntax highlighting, and improvement suggestions directly within a web interface, making it highly accessible to both students and professional developers. This chapter details the system's design, architecture, data flow, and module-level implementation.

**System Overview**
The **AI Code Analyzer Agent** is a **full-stack web application** where users can:

1. Enter or paste their code into an online editor.
2. Submit it to the backend for processing.
3. Receive **AI-generated suggestions**, including syntax corrections, performance optimizations, and readability improvements.

The architecture integrates:

- A **frontend** built with React and Vite,
- A **backend** using Node.js and Express,
- A **database** using MongoDB, and
- An **AI engine** powered by Google Gemini.

**User Interface Module (Frontend)**

- Developed using **React.js** with **Vite** for fast build performance.
- Provides a code editor with syntax highlighting using **PrismJS**.
- Uses **Axios** to send HTTP POST requests to the backend.
- Displays AI-generated suggestions using **React-Markdown** for readable formatting.
- Features responsive design for desktop and mobile users.

**Key Features:**

- Code Editor (multi-line input area)
- "Analyze" button to trigger AI review
- AI Response section with Markdown formatting
- Real-time syntax highlighting

**Server-Side Module (Backend)**

- Built using **Node.js and Express.js**.
- Receives code input via POST requests from the frontend.
- Sends code data to **Google Gemini AI API** for processing.
- Returns the AI-generated feedback to the frontend.
- Implements **CORS middleware** for secure communication between client and server.

**Workflow of the System**
The workflow of the **AI Code Analyzer Agent** can be summarized as follows:

**User Input:** The user writes or pastes code into the editor.
**Request Generation:** The frontend sends a POST request containing the code to the backend API.
**Processing:** The backend forwards the code to Google Gemini API.
**AI Analysis:** Gemini analyzes the code and generates a feedback response.
**Response Handling:** The backend receives, formats, and returns the feedback to the frontend.
**Display:** The frontend renders the formatted AI response with syntax highlighting and Markdown formatting.
**Storage (Optional):** The response and code are stored in MongoDB for later review.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 10, October 2025
DOI: 10.17148/IJARCCE.2025.141030

## REFERENCES

[1] Google AI, "Gemini API Overview – Google AI Developer Documentation," 2024. [Online]. Available: https://ai.google.dev
[2] Meta Platforms Inc., "React.js Official Documentation," 2024. [Online]. Available: https://react.dev
[3] OpenJS Foundation, "Node.js Official Documentation," 2024. [Online]. Available: https://nodejs.org/en/docs
[4] Express.js Contributors, "Express.js Guide – Web Framework for Node.js," 2024. [Online]. Available: https://expressjs.com
[5] MongoDB Inc., "MongoDB Documentation – NoSQL Database for Modern Applications," 2024. [Online]. Available: https://www.mongodb.com/docs
[6] PrismJS Community, "PrismJS Syntax Highlighting Library – Open Source Project," 2024. [Online]. Available: https://prismjs.com
[7] Axios Developers, "Axios Library Documentation – Promise-Based HTTP Client for Node.js & Browser," 2024. [Online]. Available: https://axios-http.com
[8] Unified Community, "React-Markdown Package Documentation – Markdown Renderer for React," 2024. [Online]. Available: https://www.npmjs.com/package/react-markdown
[9] Evan You et al., "Vite Documentation – Next Generation Frontend Tooling," 2024. [Online]. Available: https://vitejs.dev
[10] OpenAI and Google DeepMind, "ChatGPT and Gemini Model Insights – AI-Based Language Models for Code Understanding," White Paper, 2024.
[11] SonarSource, "SonarQube Documentation – Static Code Analysis Tool," 2024. [Online]. Available: https://docs.sonarqube.org
[12] ESLint Team, "ESLint User Guide – JavaScript Linting Utility," 2024. [Online]. Available: https://eslint.org/docs/latest/
[13] Postman Inc., "Postman API Platform – API Testing and Integration Tool," 2024. [Online]. Available: https://www.postman.com
[14] GitHub Inc., "GitHub Documentation – Version Control and Collaboration Platform," 2024. [Online]. Available: https://docs.github.com
[15] Express.js Community, "CORS Middleware for Express.js – Cross-Origin Resource Sharing," 2024. [Online]. Available: https://www.npmjs.com/package/cors

  