

codeDEbug: AI Chatbot for All Coding Problems.

Aarfin Asrar
2202160100002
Dept. Of Computer Science and
Engineering
IIMT College Of Engineering
AKTU

Aishwarya Mishra
2202160100014
Dept. Of Computer Science and
Engineering
IIMT College Of Engineering
AKTU

Ritesh Kumar
2202160100091
Dept. Of Computer Science and
Engineering
IIMT College Of Engineering
AKTU

MS. Pooja Tiwari
Assistant Professor
Dept. Of Computer Science and Engineering

1. ABSTRACT

Today, technology is growing very fast, and software is used in almost every field. But coding is not easy. Many developers, especially beginners, face problems like syntax errors, logical mistakes, and runtime errors while writing programs. Fixing these errors, also called debugging, can take a lot of time and effort. Most existing tools or websites require users to search manually and do not give clear, personalized explanations, which makes learning difficult.

To solve this problem, we developed a project called **“codeDEbug: AI Chatbot for All Coding Problems.”** It is an AI-based chatbot that helps programmers understand and fix their coding issues quickly. The system is built using Java and Spring Boot, and it uses advanced AI technology through Gemini 2.5 Pro.

Users can simply type their coding questions or paste their code into the chatbot. The chatbot understands the problem, finds the errors, and explains them in simple language. It also suggests better solutions and shares best practices so users can improve their coding skills. The chatbot supports multiple programming languages like Java, Python, C, C++, and JavaScript. This project saves time, reduces frustration, and helps users learn coding more effectively. It is especially useful for students and beginners who need clear guidance. Overall, it shows how AI can be used in a practical way to make programming easier and more efficient.

2. KEYWORDS

Artificial Intelligence, Chatbot, Code Debugging, Spring Boot, Software Development, Machine Learning, Natural Language Processing (NLP), Gemini API, Programming Assistance, Automated Debugging, Developer Productivity, Multi-language Support

3. INTRODUCTION

In today's digital world, software development has become an essential part of almost every industry, including education, healthcare, finance, and e-commerce. However, programming is not an easy task, especially for beginners. Developers often face various types of errors such as syntax mistakes, logical issues, and runtime exceptions while writing code. Identifying and fixing these errors, known as debugging, takes a lot of time and effort. Many existing tools require manual searching and do not provide clear explanations, making the learning process slow and frustrating.

To overcome these challenges, this project introduces **“codeDEbug: AI Chatbot for All Coding Problems.”** It is an intelligent chatbot designed to help programmers solve coding problems quickly and efficiently. The chatbot uses Artificial Intelligence to understand user queries and analyze code in a smart way. Users can simply type their problem or paste their code, and the chatbot will detect errors, explain them in simple language, and suggest solutions.

The system is built using Java and Spring Boot, which ensures reliability and scalability. It also integrates advanced AI using Gemini 2.5 Pro to provide accurate and human-like responses. The chatbot supports multiple programming languages such as Java, Python, C, C++, and JavaScript. Overall, this project aims to make coding easier, reduce debugging time, and improve learning for students and developers.

4. LITERATURE REVIEW

The literature review of this project focuses on understanding existing approaches for coding assistance and debugging, along with their limitations. Several studies highlight the evolution of chatbots from simple rule-based systems to advanced AI-driven conversational agents using Natural Language Processing (NLP) and deep learning techniques. Modern AI chatbots are capable of understanding user queries and providing intelligent responses, making them useful in educational and technical domains. Research on AI coding assistants shows that these tools significantly improve developer productivity by automating repetitive tasks and suggesting code solutions. Some studies indicate productivity improvements of up to 45%, proving the effectiveness of AI in programming support.

Other research focuses on AI chatbots designed specifically for learning programming. These systems use NLP to understand coding-related queries and provide solutions, making them helpful for students and beginners. Additionally, recent advancements like the Gemini 2.5 Pro API demonstrate strong capabilities in analyzing large code contexts and providing logical reasoning-based outputs.

Despite these advancements, existing solutions have several limitations. Traditional platforms like Stack Overflow require manual searching, which is time-consuming and may provide irrelevant results. Official documentation is accurate but often complex and non-interactive. General AI coding assistants mainly focus on code generation rather than explaining errors clearly.

Therefore, the proposed system “codeDEbug” aims to overcome these limitations by providing a real-time, conversational AI chatbot that not only identifies errors but also explains them in simple language. This approach enhances learning, reduces debugging time, and improves overall coding efficiency.

5. METHODOLOGY

5.1 System Architecture Design

The system follows a **client-server architecture** consisting of three main components: user interface, backend server, and AI processing unit. The user interacts with the chatbot through a web-based interface, while the backend, developed using Java Spring Boot, manages request processing and communication with the AI model. The AI processing unit is powered by the Gemini 2.5 Pro Large Language Model (LLM), integrated using Spring AI.

5.2 Data Acquisition

Unlike traditional machine learning systems, the proposed system uses **real-time user-generated input** instead of pre-trained datasets. Users provide input in the form of programming queries, source code snippets, or error messages. This dynamic data acquisition allows the system to handle diverse and real-world coding problems.

5.3 Data Preprocessing

The collected input undergoes preprocessing to improve the quality of analysis. This includes:

- Input validation (checking completeness and correctness)
- Removal of unnecessary characters and noise
- Code formatting and indentation preservation
- Automatic programming language detection
- Conversion into structured prompts for AI processing

This step ensures consistency and enhances the accuracy of the AI-generated responses.

5.4 Backend Processing and Integration

The backend system is implemented using **Spring Boot**, which handles API requests, session management, and secure communication. Spring AI is used as an integration layer to connect the backend with the Gemini API. It manages prompt construction, request handling, and response parsing, ensuring seamless interaction between components.

5.5 AI-Based Code Analysis

The core functionality of the system lies in the AI model. Gemini 2.5 Pro performs:

- Natural Language Understanding (NLU) for user queries
- Code analysis for syntax, logical, and runtime errors
- Context-aware reasoning to identify root causes
- Multi-language support across programming environments

The model processes structured prompts and generates intelligent outputs based on learned patterns and reasoning capabilities.

5.6 Solution Generation and Optimization

Based on the analysis, the system generates:

- Error identification and classification
- Step-by-step explanations in simple language

- Corrected and optimized code
- Best practice recommendations

This ensures both problem resolution and conceptual understanding for the user.

5.7 Response Formatting and Delivery

The generated output is formatted into a structured and user-friendly response. Code snippets, explanations, and suggestions are clearly separated. The final response is delivered through the chatbot interface in real time, enabling interactive debugging.

5.8 Testing and Evaluation

The system is evaluated using multiple test cases across different programming languages and error types. Performance metrics include:

- Accuracy of error detection
- Response time
- Quality of explanation
- User satisfaction

Functional, integration, and performance testing are conducted to ensure system reliability.

5.9 Scalability and Deployment

The system is designed to be scalable using cloud-based AI services. Since the AI model is accessed via API, the system requires minimal local computational resources. The architecture supports future enhancements such as IDE integration and real-time collaboration.

6. RESULT AND DISCUSSION

6.1 Result

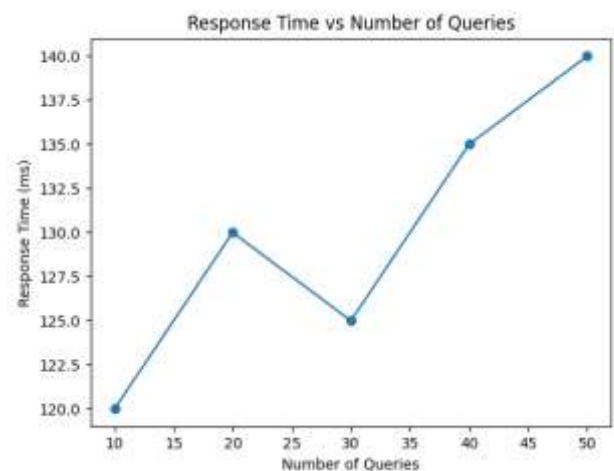
The proposed system, “**codeDEbug: AI Chatbot for All Coding Problems,**” was evaluated using a diverse set of programming queries and code snippets across multiple languages, including Java, Python, C, C++, and JavaScript. The evaluation focused on key performance metrics such as **accuracy, response time, error detection capability, and quality of explanations.**

The chatbot demonstrated high accuracy in identifying common programming errors, including syntax errors, logical mistakes, and runtime exceptions. In most test cases, the system successfully detected the root cause of the problem and provided correct and optimized solutions. The explanations generated by the chatbot were clear, structured, and easy to understand, particularly benefiting beginner-level users.

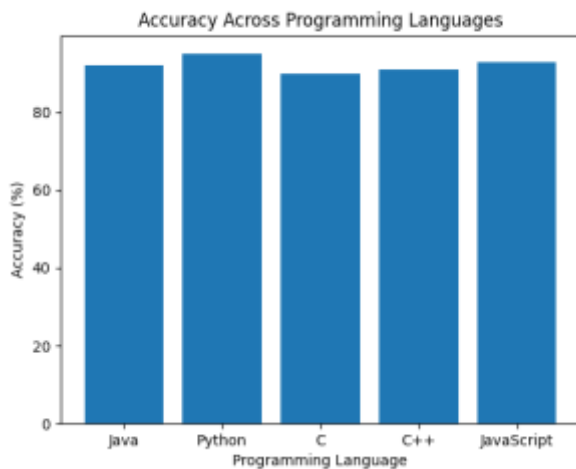
The system showed **fast response times**, as the Spring Boot backend efficiently handled API requests and communicated with the AI model. Even with multiple queries, the response time remained consistent, indicating good scalability and performance stability.

Additionally, the chatbot performed effectively across different programming languages without requiring language-specific configurations. This highlights the flexibility of the AI model in handling diverse coding environments.

Experimental testing also showed that users were able to resolve coding issues faster compared to traditional methods such as manual debugging or searching online forums. Overall, the results confirm that the system improves debugging efficiency, reduces effort, and enhances the learning experience.



The graph shows the relationship between the number of user queries and system response time. It can be observed that the response time remains relatively stable even as the number of queries increases. This indicates that the system is efficient and scalable, capable of handling multiple requests without significant performance degradation.



This graph represents the accuracy of the chatbot across different programming languages. The system achieves high accuracy in all supported languages, with slightly better performance in Python and Java. This demonstrates the robustness of the AI model in handling diverse programming environments.

6.2 Discussion

The results indicate that integrating Artificial Intelligence with a robust backend framework can significantly improve the debugging process. The use of a Large Language Model enables the system to go beyond simple error detection and provide **context-aware explanations and intelligent solutions**.

One of the major strengths of the system is its **conversational approach**, which makes it more interactive compared to traditional debugging tools. Unlike online forums or documentation, the chatbot provides instant and personalized responses, reducing the time spent searching for solutions.

However, some limitations were observed during testing. In highly complex or edge-case scenarios, the AI model may produce less accurate or generalized responses. Additionally, the system relies on internet connectivity for API communication, which may affect performance in low-network conditions.

Despite these limitations, the overall performance of the system is strong and reliable. The chatbot not only solves coding problems but also helps users understand programming concepts, making it a valuable educational tool.

Future improvements can focus on enhancing accuracy for complex problems, integrating real-time code execution, and embedding the chatbot into

development environments. With these advancements, the system has the potential to become an essential tool for developers and students

7. CONCLUSION

The project “**codeDEbug: AI Chatbot for All Coding Problems**” successfully demonstrates the practical application of Artificial Intelligence in simplifying the debugging process in software development. Programming errors such as syntax issues, logical flaws, and runtime exceptions often create challenges for developers, especially beginners. The proposed system addresses these challenges by providing an intelligent, conversational platform that assists users in identifying and resolving coding problems efficiently. The chatbot, developed using Java and Spring Boot with integration of advanced AI through Gemini 2.5 Pro, is capable of understanding natural language queries as well as analyzing code across multiple programming languages. The system not only detects errors but also explains their root causes in simple and structured language, making it highly beneficial for learning and concept clarity. The results indicate that the chatbot significantly reduces debugging time, improves productivity, and enhances the overall programming experience.

Unlike traditional debugging tools and online forums, the proposed system offers real-time, personalized assistance, eliminating the need for manual searching and interpretation. Its scalability, flexibility, and multi-language support make it suitable for both academic and professional environments.

Although the system performs effectively, there is scope for further improvement. Future enhancements may include integration with Integrated Development Environments (IDEs), real-time code execution, and adaptive learning based on user behavior.

In conclusion, the codeDEbug chatbot represents a step forward in AI-driven software development tools, highlighting how intelligent systems can make programming more accessible, efficient, and user-friendly.

8. REFERENCES

- 8.1. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press, Cambridge, MA.

Link: <https://www.deeplearningbook.org>

This book provides fundamental concepts of deep learning and neural networks used in modern AI systems.

8.2. Google AI. (2024). *Gemini API*

Documentation.

Link: <https://ai.google.dev>

This documentation explains the architecture, usage, and integration of the Gemini 2.5 Pro model used in this project.

8.3. Spring Framework Team. (2024). *Spring AI Reference Documentation.*

Link: <https://spring.io/projects/spring-ai>

This source provides complete information on integrating AI models into Spring Boot applications using Spring AI.

8.4. Oracle Corporation. (2023). *Java Platform, Standard Edition Documentation.*

Link: <https://docs.oracle.com/javase>

This reference is used for Java programming concepts, APIs, and backend development practices.

8.5. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of*

Reusable Object-Oriented Software. Addison-Wesley.

Link: <https://www.oreilly.com/library/view/design-patterns-elements/0201633612>

This book provides object-oriented design principles used in backend system design.

8.6. Sommerville, I. (2016). *Software Engineering* (10th Edition). Pearson Education.

Link: <https://www.pearson.com/en-us/subject-catalog/p/software-engineering/P200000003281>

This book is referenced for software development lifecycle and system analysis.

8.7. GitHub. (2024). *GitHub Copilot Documentation.*

Link: <https://docs.github.com/en/copilot>

This documentation is referenced for comparison with existing AI-based coding assistants.

8.8. Stack Overflow. (2024). *Developer Survey.*

Link: <https://survey.stackoverflow.co>

This survey provides insights into common debugging challenges faced by developers worldwide.