International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

AI CODE STYLE ANALYZER

Shonal Vaz[1], Ishaan Salgaonkar[2], Amey Nadkar[3],Vraaj Painter[4] , Sarthak Aute[5], Yogita Khandagale[6]

[a]Department of Information Technology, Vidylankar Polytechnic, Mumbai, India

ABSTRACT :

In the field of artificial intelligence and natural language processing (NLP), AI-powered chatbots play a crucial role in enhancing user experience. This project introduces a web-based AI chatbot that learns and adapts to a user's unique coding style. By leveraging APIs, NLP, and XAMPP for backend processing and Angular for the frontend, the system analysis user-input code and provides real-time suggestions, debugging assistance, and optimized solutions. The chatbot processes historical code snippets using NLP techniques and generates responses that align with the user's style. Unlike conventional AI chatbots, this system offers a personalized experience, aiding developers in writing efficient and stylistically consistent code. The project has applications in AI-assisted code review, automated coding suggestions, and programming education. Future improvements include expanding support for multiple languages and refining adaptability through reinforcement learning techniques.

Keywords: AI Chatbot, Natural Language Processing (NLP), Code Style Adaptation, Machine Learning

1. Introduction :

With the increasing accessibility of AI-driven coding assistants, developers now have advanced tools to enhance their coding efficiency. AI has become an integral part of various applications, with software development undergoing significant transformations. Many programmers prefer AI-driven assistance over traditional documentation, as it offers faster solutions, clearer explanations, and optimized code generation.
However, while instant access to solutions accelerates development, it may also impact problem-solving skills. Traditionally, developers learn best by writing, executing, and debugging code manually, which reinforces a deeper understanding of programming concepts. Yet, AI assistance provides distinct advantages, enhancing learning efficiency when used appropriately.
Beyond simply using AI for development, how AI adapts to individual coding styles is a critical aspect that needs exploration. This project presents an AI-powered chatbot that analyzes and adapts to a user's unique coding style, ensuring personalized code suggestions and debugging assistance. Various technologies have been integrated for different tasks:

- APIs for real-time communication and data processing
- NLP for analyzing and replicating the user's coding style.
- XAMPP for backend infrastructure and database management.
- Angular for building an interactive frontend interface.

AI CODE STYLE ANALYZER

1.1. Initial planning and research

Before starting development, we defined the *core objectives* of the project—creating an AI-powered chatbot that adapts to a user's coding style. We researched existing AI-driven code assistants, explored technologies, and structured our approach.

- Identified the need for customized AI code suggestions
- Researched Natural Language Processing (NLP) techniques
- Decided on frontend and backend technologies for implementation

1.2. Frontend Development (User Interface)

We built an interactive and responsive UI using HTML, CSS, and Angular to ensure a smooth user experience.

- Designed a clean and structured UI for developers

- Implemented real-time input fields for user code
- Created interactive elements for better usability

1.3. Backend Development (Core AI Processing)

For the backend, we used pure JavaScript instead of frameworks like Flask or Django. This allowed us to have more control over the server logic and AI processing.

- Implemented code analysis and pattern learning using NLP
- Used Google API for real-time AI processing
- Integrated a database for storing user-specific coding patterns

1.4. AI Code Adaptation & Processing

The chatbot was designed to analyze user-written code and generate new code in a similar style.

- Preprocessed input using tokenization and lemmatization
- Trained the system to match the user's style and syntax
- Used Google API for AI-based code suggestions

1.5. Testing & Debugging

Once the core functionalities were implemented, we conducted multiple testing phases to ensure smooth operation.

- Tested code output accuracy using sample inputs
- Verified user interaction flow in the UI
- Debugged backend processing errors for optimal efficiency

1.6. Final Integration & Deployment

After refining the chatbot, we ensured seamless communication between the frontend and backend, integrating APIs and optimizing performance.

- Connected Angular UI with JavaScript backend
- Ensured secure and efficient API calls
- Optimized response time and system performance

These tools were particularly beneficial because they analyzed the entire source code directly, unlike GPT and Claude, which required manual context input.

1.7. Future Enhancements

We identified areas for future improvements, such as:

- Expanding support for more programming languages
- Enhancing NLP models for better style adaptation
- Integrating with popular IDEs for real-time coding assistance

IMPACT ON CODE STYLE ANALYSIS AND DEVELOPER LEARNING :

Using AI-driven tools for code style analysis has significantly impacted how developers approach coding standards and logical structuring. While AI helps automate code formatting and style adherence, understanding the rationale behind coding conventions remains essential. AI-generated suggestions follow best practices, but they require prior knowledge of syntax rules, coding patterns, and project-specific constraints to be effectively implemented.

One of the biggest challenges is in understanding context-specific code adjustments. AI can recommend style improvements, but developers must critically analyze whether these align with team guidelines and project requirements. While AI enhances code readability and consistency, manually refining the output fosters a stronger grasp of structured programming.

Additionally, debugging AI-suggested improvements is crucial. While AI helps detect inconsistencies in style and logic, developers must evaluate whether these changes affect performance, maintainability, or security. Over-reliance on AI may lead to blindly accepting modifications without fully understanding their impact.

Our experience with AI in the AI Code Style Analyzer project highlights these aspects. While AI boosts efficiency in style enforcement, it does not replace the need for logical reasoning and personal judgment in coding. The effectiveness of AI depends on a developer's prior experience, engagement with AI-generated recommendations, and willingness to refine results independently.

4. BENEFITS AND DRAWBACKS :

4.1 Benefits of AI Code Style Analysis

AI has provided several advantages in the development of the AI Code Style Analyzer, making code review more efficient and structured. Some key benefits include:

- Time Efficiency & Productivity: AI streamlines code analysis by automatically formatting, structuring, and enforcing style guidelines, reducing manual effort. This allows developers to focus on logic and functionality rather than minor syntax inconsistencies.
- Error Reduction & Code Consistency: AI helps detect inconsistencies in code style and offers corrections based on predefined standards. This reduces manual code reviews and ensures uniformity across projects.
- Exposure to Best Practices: AI introduces developers to industry-standard coding styles and helps them understand how to write cleaner, more maintainable code.
- Guided Code Refinement: AI provides step-by-step suggestions to improve readability and maintainability, helping developers follow structured programming principles.
- Improved Problem-Solving in Code Optimization: AI suggests optimizations for performance improvements, allowing developers to explore more efficient ways to write and structure their code.

4.2 Drawbacks & Challenges of Using AI

While AI has been highly beneficial, it also comes with limitations and challenges that must be considered:

- Contextual Errors & Repetitive Outputs: AI sometimes provided the same output despite changes in context, requiring repeated prompt modifications to get the desired result.
- Lack of Context Awareness: AI sometimes misinterprets coding style preferences and enforces changes that may not align with project-specific guidelines.
- Overgeneralized Suggestions: AI-generated solutions are not always project-specific, requiring developers to manually review and modify suggestions before implementation.
- Difficulty Handling Unique Code Structures: While AI is excellent at enforcing common styles, it may struggle with custom coding patterns, leading to incorrect or inconsistent suggestions.
- Over-Reliance on AI for Code Review: Excessive reliance on AI for style enforcement can reduce the development of independent code refinement skills, making developers overly dependent on AI-generated formatting.
- Occasional Misleading Recommendations: AI does not always provide accurate or project-appropriate optimizations, requiring manual verification before accepting suggestions.

5. Optimal Use of AI for Code Style Improvement :

AI should be used strategically in coding, whether by students, employees, or developers. Based on our experience, we recommend:

- Attempt Manual Code Refinement First: Before relying on AI, individuals should review and refine their code at least 3-4 times to build critical problem-solving skills.
- Use AI with a Strong Coding Foundation: AI-generated style suggestions are most effective when the user understands core programming concepts and can evaluate AI feedback critically.
- Leverage AI for Learning Best Practices: AI should be used to enhance coding efficiency but not as a substitute for manual practice and logical thinking.
- Balance AI Assistance with Hands-on Debugging: While AI speeds up error detection, manually analyzing and fixing code develops deeper understanding and adaptability.

These recommendations apply to students learning programming, employees working on professional projects, and developers optimizing their codebase. A balanced approach ensures AI enhances productivity without reducing logical problem-solving skills.

6. Conclusion.

The integration of AI in code style analysis and software development has significantly transformed how students, employees, and developers approach coding and debugging. In our AI Code Style Analyzer project, AI tools played a crucial role in enhancing productivity, reducing inconsistencies, and providing insights into efficient coding practices. Our experience highlights that AI can be a powerful learning companion when used strategically.

AI met our expectations by streamlining development workflows and improving efficiency. However, we observed that excessive reliance on AI can reduce hands-on problem-solving skills, creating a dependency on AI-generated suggestions. This raises an important consideration: AI should complement, not replace, manual coding and logical thinking.

If we were to start another project, we would adopt a balanced approach—using AI as a reference tool rather than a direct code generator. Independently solving problems first before consulting AI would help reinforce coding skills and develop self-sufficiency.

Ultimately, we conclude that AI can enhance learning and development efficiency if used with discipline and strategy. It should be treated as a guide rather than an automatic solution provider. Developers, students, and employees alike must establish clear boundaries to ensure that AI aids in skill-building rather than diminishing critical thinking and problem-solving abilities.

Moving forward, future research could focus on how AI can be further optimized for personalized learning and adaptive development environments, striking a balance between automation and manual expertise.

7. Acknowledgements

We would like to express our sincere thanks to our guide Er. Shonal Vaz and our Head of the Information Technology Department Er. Yogita Khandagle and all the staff in the faculty of Information Technology Department for their valuable assistance.

REFERENCES :

Reference Page:

1. [1] https://youtu.be/CKhV7-NF2OI?si=F1iOoMyoGD0fL9ND
2. [2] https://youtu.be/adMDcnd7GyU?si=DolF90qPnqWcbM4u
3. [3] https://youtu.be/co-xyHRdHRg?si=mBwmYVNIgocLuPcB
4. [4] https://youtu.be/7P5Oj8heApU?si=U-x_stGnFNf1SvsO
5. [5] https://openai.com

Research Papers and Articles:

1. [1] https://www.researchgate.net/publication/379005838_AI_Based_Smart_Robot_Chatbot_using_Python?utm_source=chatgpt.com
2. [2] https://pmc.ncbi.nlm.nih.gov/articles/PMC11546207/?utm_source=chatgpt.com
3. [3] https://www.irjmets.com/uploadedfiles/paper//issue_5_may_2024/55847/final/fin_irjmets1715768590.pdf?utm_source=chatgpt.com

Books Referred:

1. Natural Language Processing with Python
   Author: Steven Bird, Ewan Klein, Edward Loper
2. Generative AI for Beginners
   Author: Sridhar Alla, Suman Kalyan Adari
3. The Conversational AI Handbook
   Author: Robert E. King