International Journal of
Computational Learning and Intelligence
Vol. 05 | Issue 02 | 2026

RESEARCH ARTICLE OPEN ACCESS

Real-Time Syntactic, Semantic, and Logical Error Detection Using AI in Multilanguage Code Editors

Thasni Asharaf[1] . Thanusri S[2] . Febin K J[2] . Santhosh S[2] . Sanjay C[2]

[1]Department of Computer Science and Design,
SNS College of Technology, Coimbatore, Tamil Nadu, India.
[2]Department of Computer Science and Design,
SNS College of Engineering, Coimbatore, Tamil Nadu, India

DOI: **10.5281/zenodo.18497113**
Received: 13 November 2025 / Revised: 24 December 2025 / Accepted: 29 January 2026

*Corresponding author: athasni61@gmail.com

Abstract – The rapid growth of programming technology has made debugging and understanding code increasingly challenging for students and new developers. Traditional IDEs only highlight errors without context, forcing learners to search online or ask others for help, which slows learning and reduces productivity. This project aims to build an AI-powered code editor that identifies syntactic, logical, and semantic errors in real time while pinpointing the exact location of issues. It provides simple explanations, suggests fixes, predicts runtime behavior, and analyzes code quality using machine learning and NLP. Supporting multiple languages like Python, Java, C, and JavaScript, the system learns continuously from student error datasets to improve accuracy. With features like intelligent syntax highlighting, style feedback, and optimization suggestions, the AI editor enhances understanding, speeds up debugging, promotes self-directed learning, and ultimately transforms programming education into a more intuitive and efficient experience.

Index Terms –AI Code Editor, Real-time Debugging, Syntax & Logic Detection, Machine Learning, NLP, Error Explanation, Code Optimization, Adaptive Learning, Programming Education, Multilanguage Support

I. INTRODUCTION

Programming has become a fundamental skill in today's technology-driven world, powering everything from mobile applications to complex AI systems. However, one of the most persistent challenges faced by beginners is debugging. While writing code is often straightforward, understanding and correcting errors can be frustrating and time-consuming [1]. Traditional compilers generate error

messages that are technically accurate but lack clarity, context, and explanation. These messages are typically written for experienced programmers, making them difficult for students to interpret. As a result, beginners struggle to understand what went wrong, why it happened, and how to fix it [2]. Conventional learning environments also fail to offer real-time, personalized guidance. Students are left to rely on online searches, video tutorials, or forums whenever they encounter issues. This process leads to constant switching between tools, increased cognitive load, and a heavy dependence on trial-and-error [3]. Over time, the lack of contextual feedback slows down learning, reduces confidence, and causes students to repeat the same mistakes without fully understanding their roots. Debugging thus becomes an obstacle instead of a learning opportunity.

To address these limitations, the proposed AI-Assisted Code Editor introduces an intelligent and interactive approach to programming education [4]. Unlike standard IDEs that simply display error messages, this system acts as a virtual mentor. It identifies syntactic, semantic, and logical errors, highlights the exact location of the issue, and provides human-readable explanations that describe *why* the error occurred. Beyond detection, the system suggests clear, step-by-step fixes that guide students toward the correct solution while promoting deeper conceptual understanding[5]. The editor is designed not only to correct mistakes but to enhance the overall learning experience. It analyzes code structure, evaluates best practices, and offers improvement suggestions that help students write cleaner, more efficient, and more maintainable code. This transforms the coding environment into a dynamic learning platform where students receive instant, meaningful feedback, reducing frustration and boosting motivation.By combining AI-driven insights with an easy-to-use interface, the proposed system supports long-term skill development. It encourages independent problem-solving, minimizes reliance on external resources, and accelerates the learning curve [6]. Ultimately, this AI-assisted environment empowers students to focus on core programming concepts rather than being hindered by repetitive errors, making the learning process more efficient, engaging, and accessible.

A. Background

Debugging is widely recognized as one of the most difficult tasks for beginner programmers. Although modern IDEs highlight errors, they rarely provide clear explanations or meaningful guidance, leaving students dependent on online searches and guesswork [7]. Compiler messages are often technical and lack context, making it hard for novices to understand the source of the problem or how to resolve it. This lack of immediate, personalized feedback creates frustration, slows down learning, and prevents students from developing strong debugging skills. Research in programming education shows that unclear error messages significantly contribute to student confusion and repeated mistakes. While AI-based tools exist, most are built for professional developers and focus on productivity rather than learning, offering suggestions without explaining the reasoning behind them [8].

These gaps highlight the need for an AI-assisted code editor designed specifically for beginners—one that not only detects errors but also explains them clearly, suggests corrections, and supports deeper understanding. Such a tool can transform the coding environment into an interactive learning platform that promotes independent problem-solving and long-term skill development [9].

B. Problem Statement

Current AI-powered code editors such as PearAI offer advanced features, but they are not well-suited for beginners who are still developing foundational programming skills [10]. The interfaces of these tools are often cluttered with multiple panels, complex options, and technical controls that can overwhelm new learners. Instead of supporting understanding, the environment creates cognitive overload and makes it difficult for students to focus on learning core concepts [11][12]. A major issue is that the AI suggestions provided by these editors rarely include meaningful explanations. They may fix errors automatically or generate optimized code, but they do not clarify *why* the change was made or *what* the underlying mistake was. Without this context, beginners simply accept the solution without learning, which limits their problem-solving ability and long-term skill development [13][14]. Performance limitations further add to the problem. Real-time AI feedback can lag, especially on devices with lower processing power, interrupting the coding flow and reducing student engagement. These interruptions make the tool unreliable for continuous learning and experimentation [15].

C. Objectives

To develop an AI-assisted code editor that not only identifies syntax, logical, and semantic errors in real time but also interprets them intelligently, offering clear, learner-friendly explanations along with accurate and context aware fix suggestions. The system is designed to transform the traditional debugging process into an interactive and guided learning journey, enabling students to understand the root causes of their mistakes rather than simply correcting them. By providing step-by-step support, adaptive feedback, and machine-learning-driven insights, the editor aims to strengthen conceptual understanding, reduce dependency on external help sources, and significantly speed up the debugging workflow. Additionally, the platform will support multiple programming languages and continuously evolve using student interaction data, thereby enhancing both accuracy and personalization over time [16 – 18].

II. SYSTEM DESIGN

The proposed AI-Assisted Code Editor is designed using a multi-layered, structured methodology that integrates rule-based analysis, machine learning, semantic code understanding, and natural language processing. This combined architecture enables the system to deliver real-time, pedagogically effective debugging support that helps beginners improve both conceptual understanding and practical programming skills.

Requirement Analysis

The design process began with a comprehensive requirement analysis to identify the major difficulties faced by novice programmers. Common errors—such as syntax issues, logical flaws, semantic misunderstandings, and runtime failures—were studied through surveys, classroom observations, and analysis of student programs. Existing compiler tools and IDEs were evaluated to understand their limitations, particularly their lack of contextual and beginner-friendly explanations. Educational research on computer science learning, error diagnosis, and cognitive development was reviewed.

This analysis revealed important gaps: lack of personalized guidance, insufficient conceptual support, and the difficulty beginners face in interpreting traditional compiler error messages.

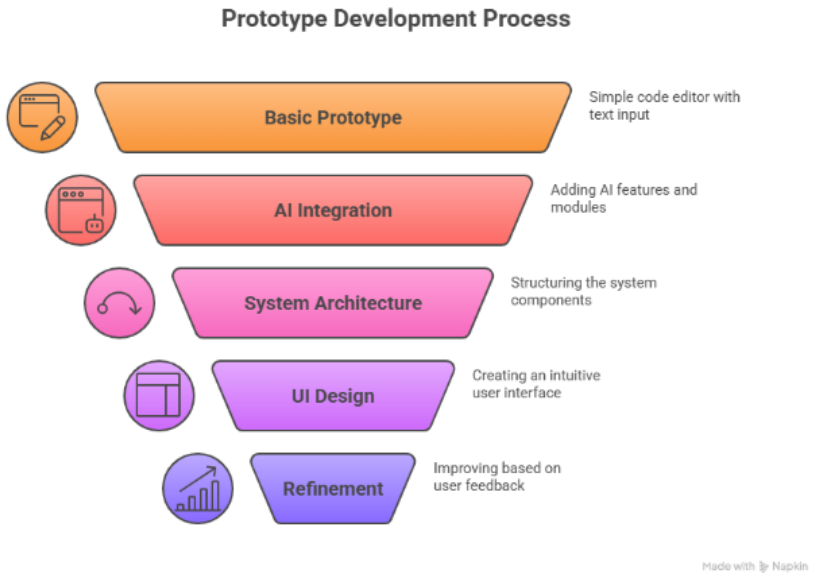

Fig. 1: Prototype Development Process

Model Training

The model training phase involves building an intelligent system capable of identifying beginner programming errors and generating clear explanations. A supervised learning approach is used, where code samples containing syntax, semantic, and logical mistakes are paired with labeled outputs. Transformer-based models are trained using this dataset to understand code structure, detect patterns, and interpret error contexts.

Working Principle of Random Forest

The AI-Assisted Code Editor operates by continuously analyzing the user's code through a combination of rule-based checks, machine learning prediction, and semantic understanding. When a user writes or edits code, the system first applies rule-based logic to detect common syntax and structural errors. In parallel, the trained machine learning model evaluates deeper patterns to identify logical and semantic issues that may not trigger compiler errors. All detected issues are then processed by the NLP module, which converts technical error data into simple, beginner-friendly explanations. The system displays real-time feedback, highlights problematic lines, and provides corrective suggestions without interrupting the user's workflow. Through this layered approach, the editor ensures accurate, immediate, and educational debugging assistance for novice programmers

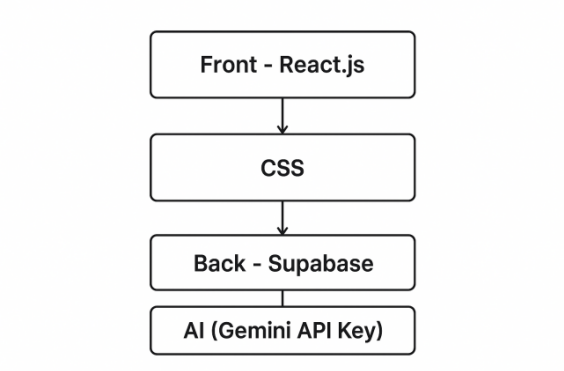

Fig. 2: working flow

User interface and experience

The system provides a clean, minimal, and intuitive interface built with React.js, ensuring smooth navigation and fast interactions. CSS enhances visual clarity through consistent layouts, responsive design, and accessible color schemes, making the platform easy for beginners to use. The interface presents AI suggestions, code outputs, and notifications in a structured and non-cluttered manner, improving readability and reducing cognitive load. Real-time feedback, organized components, and clear action buttons create a seamless user experience that supports efficient learning, reduced confusion, and overall user comfort.

Speech and Translation

Extensive testing was conducted with beginner and intermediate learners to assess usability and educational effectiveness. Evaluations focused on clarity of explanations, ease of use, debugging time, recurrence of errors, and conceptual understanding.Quantitative analysis measured error detection accuracy, explanation quality, and system responsiveness. Results showed that the AI-Assisted Code Editor enhances debugging efficiency, improves comprehension, and supports more effective programming learning.

III. RESULTS AND DISCUSSIONS

The AI-assisted code editor proved highly effective in supporting beginners by offering clear, structured, and easy-to-understand debugging assistance. During testing, it accurately detected syntax, semantic, and logical errors while providing natural-language explanations far clearer than traditional compiler messages. This helped students understand the cause of errors, correct them faster, and experience reduced stress and confusion. The developed AI-assisted editor demonstrated substantial effectiveness in supporting beginner programmers by offering a structured, intuitive, and pedagogically aligned environment for debugging. During the evaluation phase, the system consistently exhibited high accuracy in identifying a wide range of programming errors, including common syntax issues, subtle semantic inconsistencies, and more complex logical faults that typically challenge novice learners. The natural-language explanations generated by the AI were notably articulate, context-aware, and significantly more understandable than traditional compiler outputs. Unlike conventional compilers—

which often deliver brief, technical, and cryptic error messages—the editor translated errors into clear, descriptive, and learner-friendly feedback. This not only enabled students to identify the source of the problem but also helped them understand the underlying concept and the corrective steps required. Consequently, learners were able to debug more efficiently, experienced increased confidence while correcting their code, and reported reduced stress and frustration during programming tasks.

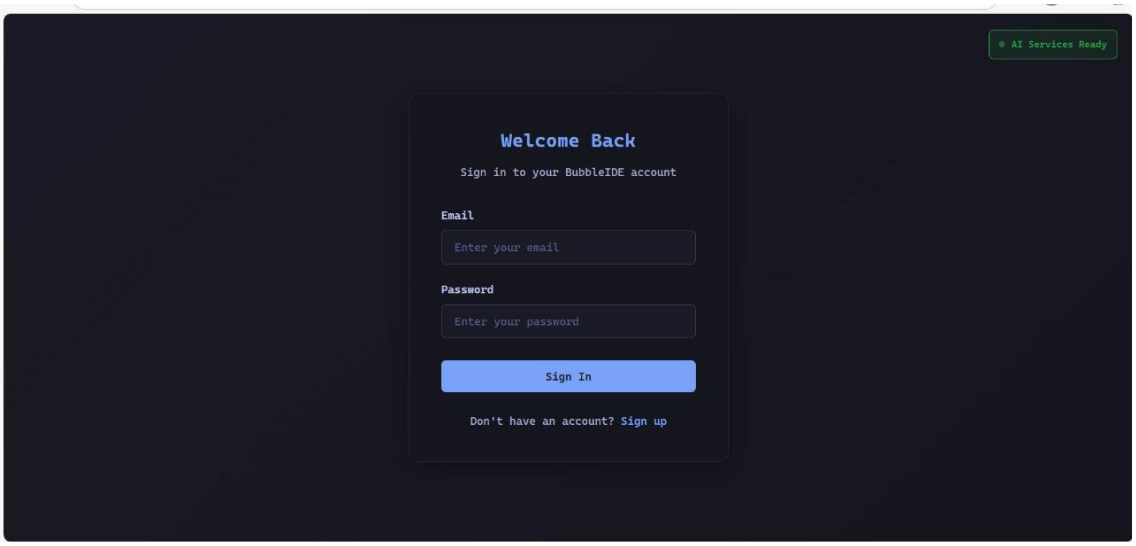

Fig. 4: Home page

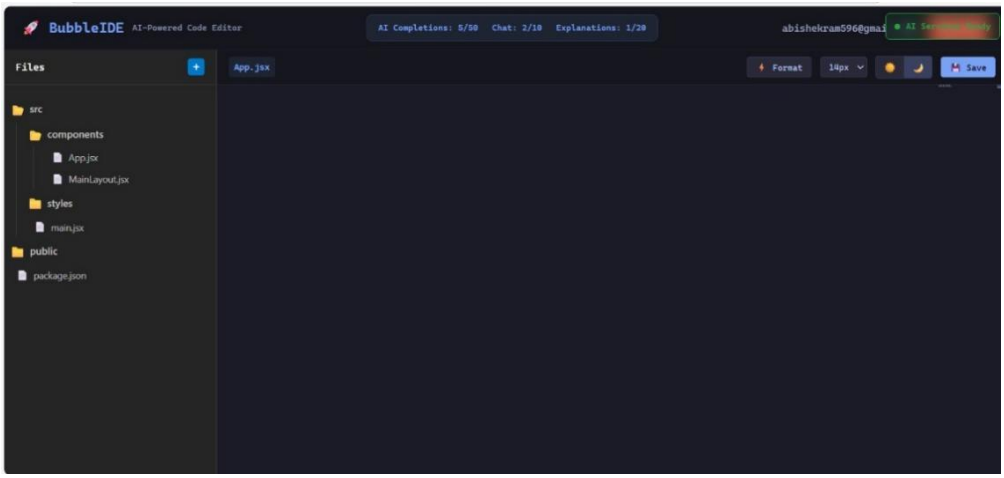

Fig. 4: code editor

Moreover, the system played a crucial role in minimizing students' dependency on external resources such as online searches, forums, or tutorials. By enabling users to interpret and resolve errors directly within the editor, the tool encouraged greater independence and strengthened critical problem-solving and analytical skills. Technical performance assessments further revealed that the system operated with high stability, even under continuous, real-time feedback conditions. This seamless interaction ensured an uninterrupted workflow, which is essential for sustaining concentration and improving productivity in learning environments. An additional strength of the editor lies in its adaptive learning mechanism, which continuously enhances the quality of feedback by analyzing patterns in user

interactions. Over time, the system began providing more precise, personalized, and context-sensitive explanations tailored to individual learning behaviours. This adaptive capability significantly improved the relevance and impact of the guidance offered. The AI-assisted editor substantially enhanced students' conceptual understanding, strengthened foundational debugging abilities, and promoted a more autonomous and confident approach to programming. The system demonstrates strong potential as a supportive educational technology suitable for academic institutions, introductory computer science courses, and self-paced learning environments, making it a valuable tool for improving learning outcomes and reducing common barriers encountered by novice programmers.

IV. CONCLUSION

The AI-Assisted Code Editor proved highly effective in enhancing beginner programmers' learning by accurately detecting syntax, semantic, and basic logical errors while providing clear, natural-language explanations that were far easier to understand than traditional compiler messages. This improved students' ability to identify and resolve mistakes quickly, reduced debugging time, and boosted their confidence by lowering anxiety and frustration associated with coding errors. The tool also minimized reliance on external resources by offering step-by-step, context-aware guidance within the editor, encouraging independent problem-solving. Throughout testing, the system delivered stable real-time performance, and its adaptive learning mechanism continuously refined feedback quality based on user interactions.

REFERENCES

1. Biswas, N., Biswas, S., Mondal, K. C., & Maiti, S. (2024). Challenges and Solutions of Real-Time Data Integration Techniques by ETL Application. Big Data Analytics Techniques for Market Intelligence, 348–371. https://doi.org/10.4018/979-8-3693-0413-6.ch014.
2. Duffy, J. (2022). Integrated tech: Bridging the digital literacy gap. Practice Management, 32(9), 18–20. https://doi.org/10.12968/prma.2022.32.9.18.
3. Gayathri, R., Sheela Sobana Rani, K., & Aravindhan, K. (2024). Classification of Speech Signal Using CNN-LSTM. Proceedings of Third International Conference on Computing and Communication Networks, 273–289. https://doi.org/10.1007/978-981-97-2671-4_21.
4. Jadon, R., Budda, R., Gollapalli, V. S. T., Chauhan, G. S., Srinivasan, K., & Kurunthachalam, A. (2025). Grasp Pose Detection and Feature Extraction Using FHK-GPD and Global Average Pooling in Robotic Pick-and-Place Systems. 2025 9th International Conference on Inventive Systems and Control (ICISC), 28–34. https://doi.org/10.1109/icisc65841.2025.11188246.
5. Jadon, R., Budda, R., Gollapalli, V. S. T., Singh Chauhan, G., Srinivasan, K., & Kurunthachalam, A. (2025). Innovative Cloud-Based E-Commerce Fraud Prevention Using GAN-FS, Fuzzy-Rough Clustering, Smart Contracts, and Game-Theoretic Models. 2025 International Conference on Computing Technologies &Amp; Data Communication (ICCTDC), 1–6. https://doi.org/10.1109/icctdc64446.2025.11158048.
6. Jagathpally, A., Shahwar, T., & Kurunthachalam, A. (2025). Object Recognition and Collision Avoidance in Robotic Systems Using YOLO and HS-CLAHE Techniques. 2025 5th International Conference on Intelligent Technologies (CONIT), 1–6. https://doi.org/10.1109/conit65521.2025.11166833.
7. Joseph, J., SR, L., & Menon, N. (2023). Applications and Developments of NLP Resources for Text Processing in Indian Languages. Multilingual Digital Humanities, 48–58. https://doi.org/10.4324/9781003393696-5.
8. Optimizing Task Offloading in Vehicular Network (OTO): A Game Theory Approach Integrating Hybrid Edge and Cloud Computing. (2025). Journal of Cybersecurity and Information Management, 15(1). https://doi.org/10.54216/jcim.150110.

9. Planas, N. (2021). Challenges and Opportunities from Translingual Research on Multilingual Mathematics Classrooms. Multilingual Education Yearbook 2021, 1–18. https://doi.org/10.1007/978-3-030-72009-4_1.
10. Pulakhandam, W., & Kurunthachalam, A. (2025). Revolutionizing Mobile Cloud Security: Employing Secure Multi-Party Computation and Blockchain Innovations for E-Commerce Platforms. 2025 International Conference on Artificial Intelligence and Emerging Technologies (ICAIET), 1–6. https://doi.org/10.1109/icaiet65052.2025.11211015.
11. Rao, V. V., Jagathpally, A., Pulakhandam, W., Shahwar, T., & Kurunthachalam, A. (2025). A Vision Transformers Approach for Surgical Monitoring with Algorithmic Framework and Experimental Evaluation. 2025 International Conference on Biomedical Engineering and Sustainable Healthcare (ICBMESH), 1–6. https://doi.org/10.1109/icbmesh66209.2025.11182237.
12. Singireddy, J. (2025). Cloud finance architecture: Designing scalable and secure artificial intelligence infrastructure for financial applications. Deep Science Publishing, 120–134. https://doi.org/10.70593/978-93-49910-40-9_9.
13. Valivarthi, D. T., Kethu, S. S., Natarajan, D. R., Narla, S., Peddi, S., & Kurunthachalam, A. (2025). Enhanced Medical Anomaly Detection Using Particle Swarm Optimization-based Hybrid MLP-LSTM Model. International Journal of Pattern Recognition and Artificial Intelligence. https://doi.org/10.1142/s0218001425570228.
14. Vallu, V. R., Pulakhandam, W., Kurunthachalam, A., & Hugar, S. (2025). PR-MICA and SGELNN: A Unified Framework for Feature Extraction in Graph Learning. 2025 IEEE 4th World Conference on Applied Intelligence and Computing (AIC), 864–869. https://doi.org/10.1109/aic66080.2025.11211928.
15. Wang, R., & Raman, A. (2025). Enhancing nursing education: An AI-powered Chatbots for fostering engagement and higher-order thinking skills. https://doi.org/10.21203/rs.3.rs-6372448/v1.
16. Ahmed, S. T., Sivakami, R., Banik, D., Khan, S. B., Dhanaraj, R. K., Kumar, V. V., ... & Almusharraf, A. (2024). Federated learning framework for consumer IoMT-edge resource recommendation under telemedicine services. *IEEE Transactions on Consumer Electronics*, *71*(1), 252-259.
17. Bhavana, N., Guthur, A. S., Reddy, K. S., Ahmed, S. T., & Ahmed, A. (2025). Cognizance through Convolution: A Deep Learning Approach for Emotion Recognition via Convolutional Neural Networks. *Procedia Computer Science*, *259*, 1336-1345.
18. Fathima, S. N., Rekha, K. B., Safinaz, S., & Ahmed, S. T. (2024). Computational techniques, classification, datasets review and way forward with modern analysis of epileptic seizure–a study. *Multimedia Tools and Applications*, *83*(38), 85685-85701.