
NeuroFlow IDE: An AI-Assisted Interactive Browser-Based Development Environment

Aniket Atram, Ansh Bagul, Yash Bhongale, Ruchika Birajdar, Shobha Patil

Department of Computer Engineering, Vishwakarma Institute of Technology, Pune, 411037, Maharashtra, India. aniket.atram24@vit.edu, yash.bhongale241@vit.edu, ruchika.birajdar24@vit.edu

Abstract.

Desktop and most cloud-based IDEs usually fail to integrate code writing, execution, debugging, assistance and visualization in unified and learner-friendly environment at single platform, creating a challenge for beginners and developer leading to affect learning and productivity timing. To tackle this problem, this project presents NeuroFlow IDE, a cloud-based integrated development environment that contains features like AI-assistance support for coding, learning support, and real time visualization of code flow of how code is working in real time. It is built on React and Monaco Editor with secured Dockerized backend and dynamically powered on D3.js for Visualization. NeuroFlow simplifies the setup process, promotes quicker debugging, and is interactive, allowing an environment if editing, executing, visualizing, and combining multiple platform features into one. Increases in learning and productivities but also contributes substantially to enhancement of practical skills and technical knowledge among the students and developers.

Keywords. Cloud-based IDE; AI-assisted Coding; Real-time Code Visualization; Assertive Debugging; Intelligent Code Analysis; Software Learning Platform.

1. INTRODUCTION

Integrated Development Environments (IDEs) provide essential tools for coding, compilation, debugging, and version control. While traditional IDEs such as IntelliJ IDEA, Eclipse, and Visual Studio are powerful, their setup can be complex for beginners. Cloud IDEs simplify development but often lack real-time execution visualization and intelligent debugging. To address this gap, NeuroFlow IDE combines AI-assisted debugging, real-time program visualization, and secure Docker-based code execution in a unified web environment, supporting effective learning and productivity.

2. RELATED WORK

Independent studies have focused on AI-assisted programming feedback, visualization-based learning tools, and intelligent debugging systems in the recent past. Large language models that generate hints in the style of an AI tutor are helpful in improving programming guidance, but do not support visualization [1]. Improved compiler error-messages research enhances, yet concentrates on syntax-level cognition, in the performance of learners [2]. Visualization systems based on algorithms and VR learning environments enhance the clarity of the concepts but fail to incorporate AI support [3][4]. Code improvement methods and language model-based build optimization techniques are used to improve efficiency of

compilation and debugging of a program [5][7]. Visual debugging tools enable execution tracing but are separate environments [8]. Nevertheless, there has not been a single platform which combines AI support, visualization, and real-time debugging; NeuroFlow IDE tries to fill this gap.

	AI Assistance	Visualization Support	Real-time Debugging	Unified Platform
[1]	Yes	No	No	No
[2]	Yes	No	No	No
[3]	No	Yes	No	No
[4]	No	Yes	No	No
[7]	Yes	No	No	No
[8]	No	Yes	Yes	No
This work	Yes	Yes	Yes	Yes

Table 1. Comparison between this work with the related work or previous research.

3. METHOD, EXPERIMENTS AND RESULTS

This section explains the design and implementation of proposed NeuroFlow IDE, experimental evaluation and performance measurement of proposed NeuroFlow IDE.

3.1. System Architecture

NeuroFlow IDE is based on three layers with the frontend, backend, and sandboxed execution layers as presented as Fig. 3.1.1. The frontend interface is a coding interface that gives interactive execution of displaying programs in real-time. The backend interface receives the execution requests, does a static analysis, and links to the AI services to assist in contextual debugging. The worker execution layer is safe as it operates the program with safety and creates execution traces which are visualized.

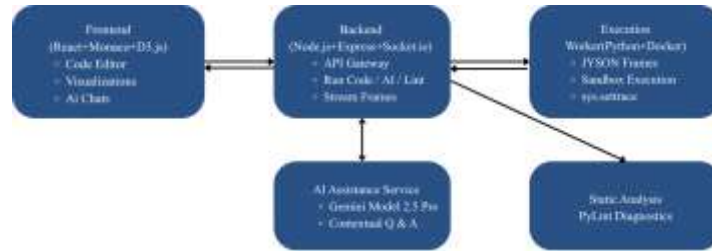


Figure 3.1.1. NeuroFlow System Architecture

3.2. Implementation

The frontend is built with React.js and Monaco Editor in order to create the minimal and fast coding environment resembling the contemporary desktop IDEs. D3.js visualization modules are dynamically used to execute with the help of algorithms like sorting tasks, recursion trees, stacks, queue, and linked lists execution flow. Backend is written in node.js, and it is based on the WebSocket communications between the execution environment and the editor for real-time interaction. Python has the `sys.settrace()` function which keeps line-

by-line track of execution states, variable assignments, and runtime exceptions, which is used to execute tracing. These traces have been converted into JSON frames and sent to the visualization frontend module. Fig. 3.2.1 represents the workflow of the tracing of the execution and its visualization.



Figure 3.2.1. NeuroFlow System Architecture

A contextual explanation, syntax correction proposal feature, and debugging guidance are available as an integral part of Gemini models with an AI assisting module through which an individual asks the computer questions. The Docker containers, which have restricted CPU, memory, and execution, are used to offer safety in sandboxing of code to avoid unsafe code execution.

3.3. *Experimental Setup*

This was also evaluated using standard programming tasks such as sorting algorithms, tracing of recursion, and simple data-structure operations. The tested system had 14 participants who were novice and intermediate programmers. Measures of evaluation were the accuracy of executions, synchronization of display, the reliability of AI-responses, and efficiency of system debugging, and stability through variability of browsers and devices

3.4. *Result and Performance Evaluation*

The results proved in improvement of about 65-75% in conceptual understanding of algorithm execution compared to the traditional IDE environments e.g. VS Code and Code Blocks. Execution flow, visualization minimized manual debugging methods, e.g. breakpoint tracing and console inspection. The integrated AI assistant also enhanced the efficiency of debugging by offering contextual explanations, and automatic syntax correction, which greatly minimized the amount of debugging effort required when using the system by beginner users. Visualization updates of the standard algorithms took 1-3 seconds, whereas recursive execution took about 6-8 seconds according to the size of the dataset and network conditions. Stability and compatibility were tested by using the Chrome, Edge, and Firefox browsers. The sandbox environment that was created using Docker ensuring that the execution is secure and did not influence responsiveness. Altogether, visualization, execution tracing, and AI-supported debugging increased the usability, efficiency of learning and understanding the program-flow in a browser-based programming environment.

4. **DISCUSSION**

The results of the experiments show that real-time visualization combined with AI-assisted debugging contributes to enhance the understanding of a program and lower the debugging effort of novice programmers. Docker-based sandboxing ensures controlled execution, and

trace streaming allows you to view the trace in the browser of your choice. The system, however, does only have support for a few programming languages and the use of third-party AI APIs can add latency. More languages, predictive code assistance, collaboration, integration with LMS, and version-control support are all areas that will be explored in the future.

5. CONCLUSION

NeuroFlow IDE brings AI-powered debugging, live code visualisation and safe container-based execution to one browser-based IDE. The concepts were better understood and less debugging was required when novice and intermediate programmers were evaluated experimentally. More language support, predictive help, collaboration and deployment on the cloud are planned for future work.

6. REFERENCES

- [1] T. Phung, V.-A. Pădurean, A. Singh, C. Brooks, J. Cambronero, S. Gulwani, A. Singla, and G. Soares, “Automating human tutor-style programming feedback: Leveraging GPT-4 tutor model for hint generation and GPT-3.5 student model for hint validation”, in Proc. 14th Int. Conf. Learning Analytics & Knowledge (LAK 24), Kyoto, Japan, Mar. 18–22, 2024, pp. 12–23. New York, NY: ACM.
- [2] Z. Zhou, S. Wang, and Y. Qian, “Learning From Errors: Exploring the Effectiveness of Enhanced Error Messages in Learning to Program”, *Frontiers in Psychology*, vol. 12, p. 768962, Nov. 30, 2021.
- [3] K. K. K. V. Kondabathini, and Dr. B. Bharathi, “Sorting Algorithm Visualizer”, Sathyabama Institute of Science and Technology, Chennai, India, Mar. 2021.
- [4] M. Mukasheva, Z. Kalkabayeva, and N. Pussymanov, “Visualization of sorting algorithms in the virtual reality environment”, *Frontiers in Education*, vol. 8, Article 1195200, Aug. 9, 2023.
- [5] H. Aïdasso, M. Sayagh, and F. Bordeleau, “Build Optimization: A Systematic Literature Review”, *ACM Computing Surveys*, vol. 58, no. 1, pp. 1–38, Jan. 2025.
- [6] S. Shahaba, R. Razeen, A. Zubia, T. Surwadkar, V. Shah, and N. Shaikh, “Sorting Algorithm Visualizer”, *International Journal of Creative Research Thoughts (IJCRT)*, vol. 12, no. 2, pp. 2568–2577, Feb. 2024.
- [7] J. Gong, V. Voskanyan, P. Brookes, F. Wu, W. Jie, J. Xu, R. Giavrimis, M. Basios, L. Kanthan, and Z. Wang, “Language Models for Code Optimization: Survey, Challenges and Future Directions”, *arXiv preprint arXiv:2501.01277*, Jan. 2, 2025.
- [8] T. Kräuter, H. König, A. Rutle, and Y. Lamo, “The Visual Debugger Tool,” *arXiv preprint arXiv:2404.12932*, Apr. 19, 2024.

Biographies



Aniket Atram is presently pursuing his Bachelor's degree in Computer Engineering at Vishwakarma Institute of Technology, Pune. His interests include software development and emerging technologies.



Ansh Bagul is presently pursuing his Bachelor's degree in Computer Engineering at Vishwakarma Institute of Technology, Pune. His areas of interest include computer engineering and emerging technologies.



Yash Bhongale is presently pursuing his Bachelor's degree in Computer Engineering at Vishwakarma Institute of Technology, Pune. His interests include web development and programming.



Ruchika Birajdar is presently pursuing his Bachelor's degree in Computer Engineering at Vishwakarma Institute of Technology, Pune. She is interested in AI and software development.



Mrs. Shobha Mahendrakumar Patil is an Assistant Professor at the Vishwakarma Institute of Technology (VIT), Pune in the Department of Computer Engineering. She is a PhD holder from Bharati Vidyapeeth University (2025), a ME holder from Savitribai Phule Pune University (2015) and BE holder from Shivaji University (2008).