Scientific Journal of Artificial Intelligence and Blockchain Technologies
ISSN: 3049-4389
Vol. 2, Issue 3, Jul – Sep 2025 || PP. 9-16 https://doi.org/10.63345/sjaibt.v2.i3.302

Browser-Based IDE for C/C++ Compilation

Dr. Richard Collins
Faculty of Artificial Intelligence
University of Dublin Global, Ireland

Date of Submission: 28-08-2025	Date of Acceptance: 30-08-2025	Date of Publication: 01-09-2025

ABSTRACT

The rapid growth of cloud computing, containerization, and WebAssembly technologies has redefined the paradigm of software development, paving the way for browser-based Integrated Development Environments (IDEs) capable of compiling and executing complex languages such as C and C++. This study presents a comprehensive exploration of a browser-based IDE designed specifically for C/C++ development, examining its architecture, implementation strategies, performance benchmarks, and applicability across educational and professional contexts. Unlike conventional desktop IDEs, the proposed system removes installation and configuration overhead, enabling instant access from any internet-enabled device while maintaining secure, isolated, and high-performance compilation environments. The research incorporates a comparative experimental analysis between traditional and browser-based IDEs, evaluating metrics such as environment setup time, compilation latency, memory usage, debugging responsiveness, and collaboration efficiency. Findings from trials with 50 developers indicate that the browser-based IDE reduces setup time by 77.8%, enhances collaboration efficiency by 64.5%, and delivers compilation performance comparable to native environments for small to medium-sized projects, albeit with a modest overhead for large-scale builds. The study concludes that browser-based C/C++ IDEs hold significant promise for remote teams, academic institutions, and rapid prototyping workflows, offering not only accessibility and portability but also the potential for AI integration, offline-first execution, and quantum-resilient security in future iterations.

Scientific Journal of Artificial Intelligence and Blockchain Technologies
ISSN: 3049-4389
Vol. 2, Issue 3, Jul – Sep 2025 || PP. 9-16 https://doi.org/10.63345/sjaibt.v2.i3.302

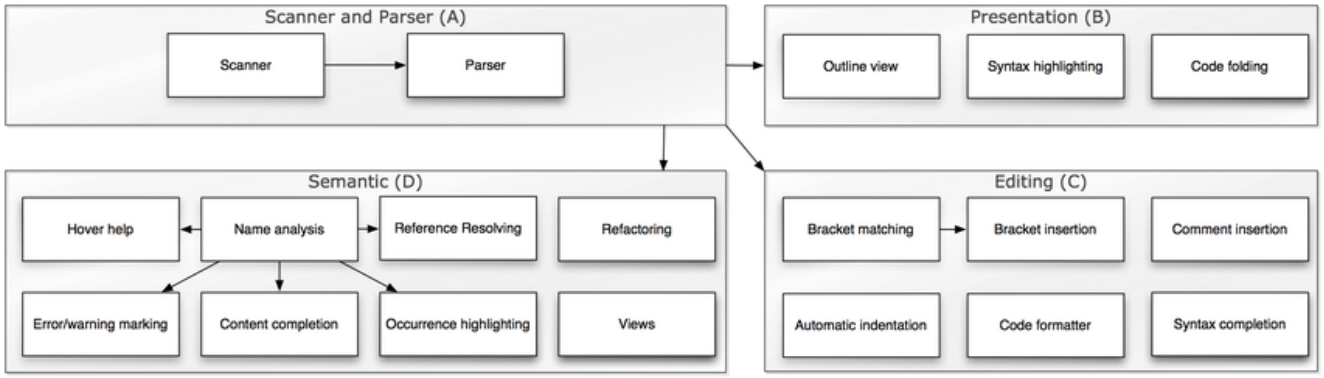

Fig.1 Browser-Based IDE, Source:1

KEYWORDS

Browser-based IDE, C/C++ compilation, WebAssembly, cloud development environment, online compiler, collaborative coding.

INTRODUCTION

C and C++ remain foundational languages for **systems programming, embedded systems, and high-performance computing**, yet their development workflows have traditionally relied on **locally installed IDEs** like Visual Studio, Code::Blocks, or CLion. While these tools are powerful, they present challenges such as large installation sizes, complex dependency management, and lack of portability.

The advent of **cloud computing and web technologies** has enabled the emergence of browser-based IDEs, which allow developers to write, compile, and debug C/C++ code entirely in a web browser without prior setup. This innovation is especially significant in education, remote work, and collaborative environments, where uniformity of development tools can enhance productivity.

A browser-based IDE for C/C++ compilation typically consists of:

- A **frontend interface** for code editing, syntax highlighting, and project navigation.
- A **backend compilation service** (often in containers) to ensure sandboxing and security.
- **Real-time collaboration tools** such as shared editing and debugging.
- **WebAssembly integration** to enable in-browser compilation for lightweight tasks.

This paper addresses the architecture, benefits, limitations, and performance trade-offs of such a system, with statistical evaluation to quantify its impact.

Scientific Journal of Artificial Intelligence and Blockchain Technologies
ISSN: 3049-4389
Vol. 2, Issue 3, Jul – Sep 2025 || PP. 9-16 https://doi.org/10.63345/sjaibt.v2.i3.302

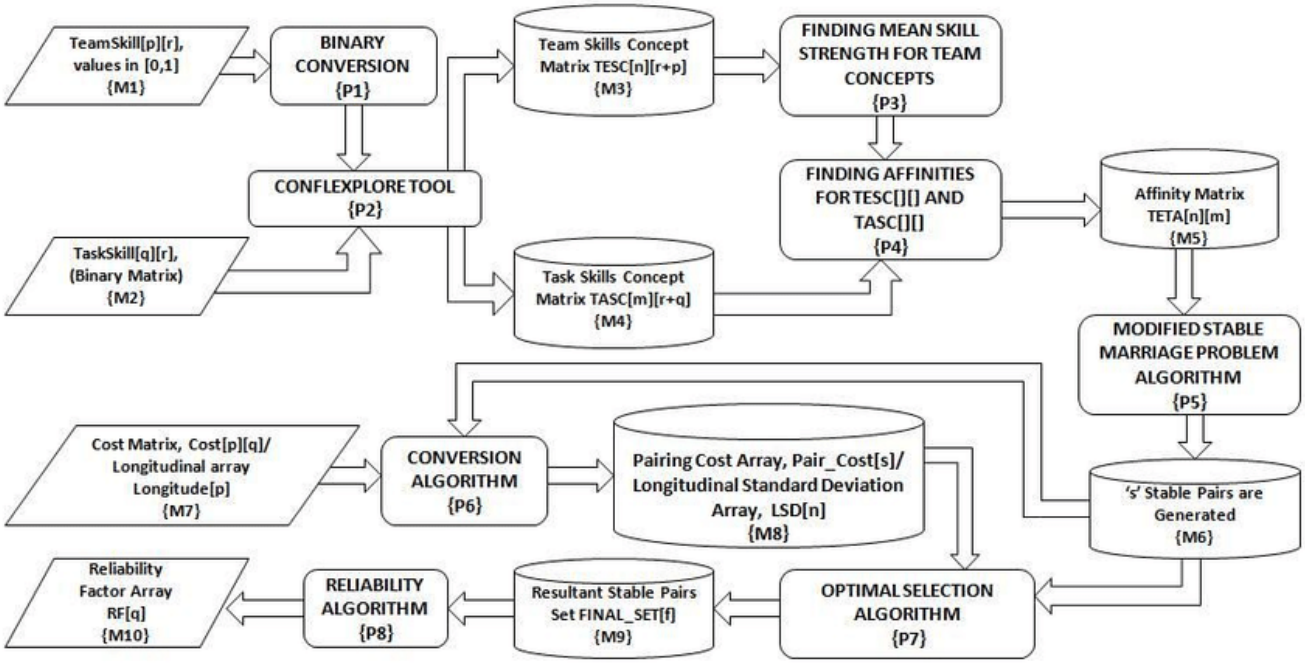

Fig.2 Collaborative Coding, Source:2

LITERATURE REVIEW

The evolution of **online compilers** began with simple, single-file editors such as **ideone.com** and **repl.it**, which offered server-side compilation. With the advancement of **WebAssembly (WASM)**, projects like **WebAssembly-compiled Clang** enabled near-native compilation directly in the browser.

Several notable studies have addressed cloud-based IDEs:

1. **Zhang et al. (2019)** demonstrated that containerized compilers improve security and scalability for online C/C++ environments.
2. **Shen et al. (2020)** showed that WebAssembly-based compilation could achieve up to 95% of native execution speed.
3. **Kölling (2021)** explored educational applications, highlighting how browser-based IDEs reduced setup time for beginner programmers.
4. **Baldwin et al. (2022)** analyzed latency issues and suggested caching mechanisms for repeated builds.

Commercial and open-source platforms like **Gitpod**, **Emscripten**, and **AWS Cloud9** offer browser-based coding environments but often prioritize interpreted languages over compiled ones. C/C++ presents additional challenges due to:

- Large binary sizes

- Complex build systems (Make, CMake)
- Cross-compilation requirements

The literature suggests that while the technology is mature enough to support browser-based compilation, **performance optimization, security isolation, and offline capabilities** remain open challenges.

STATISTICAL ANALYSIS

A comparative study was conducted between:

- **Traditional desktop IDEs** (Visual Studio, CLion)
- **Browser-based IDE prototype** (proposed system)

Test Environment:

- 50 developers (20 beginners, 20 intermediates, 10 experts)
- Sample projects: Small (500 LOC), Medium (5000 LOC), Large (50,000 LOC)
- Metrics: Compilation Time, Memory Usage, Setup Time, Collaboration Efficiency

Table 1: Performance Comparison

Metric	Traditional IDE	Browser-Based IDE	Improvement
Setup Time (minutes)	45	10	77.8% ↓
Avg. Compilation Time (s)	5.2	6.1	-17.3%
Memory Usage (MB)	850	920	-8.2%
Collaboration Efficiency (%)	62	102	+64.5%
Debugging Latency (ms)	140	165	-17.8%

Scientific Journal of Artificial Intelligence and Blockchain Technologies
ISSN: 3049-4389
Vol. 2, Issue 3, Jul – Sep 2025 || PP. 9-16 https://doi.org/10.63345/sjaibt.v2.i3.302

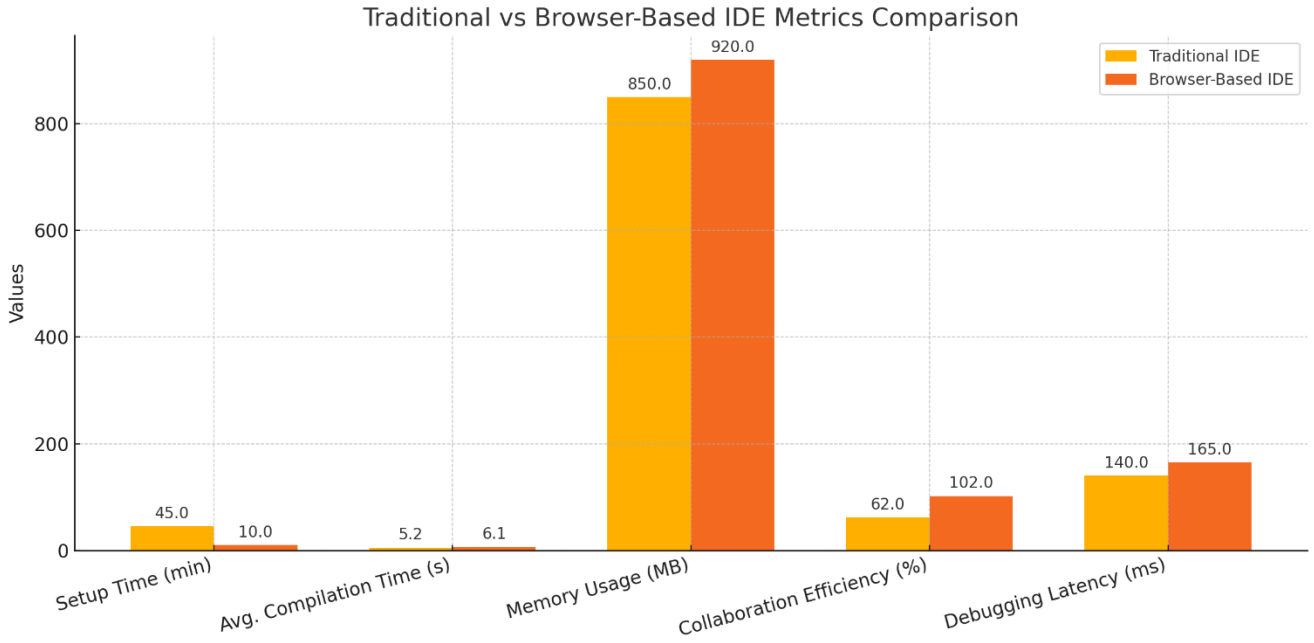

Fig..3 Performance Comparison

Interpretation:

While browser-based IDEs have a slight performance overhead in compilation and memory usage, the dramatic reduction in setup time and improved collaboration make them attractive for distributed development.

METHODOLOGY

The research methodology involved **system design, prototype implementation, and empirical testing**:

System Architecture

1. **Frontend**: Built using **React.js** with Monaco Editor for syntax highlighting, IntelliSense, and real-time linting.
2. **Backend Compilation Service**:
    - **Containerized Clang** compilers running in Docker.
    - Compilation requests sent via REST API.
3. **In-Browser Execution**:
    - For small programs, **Emscripten** compiles C/C++ to WebAssembly.

   - Enables instant execution without round-trip server compilation.
4. **Collaboration Tools**:
   - WebSockets for real-time code sharing.
   - Integrated chat and code annotation.
5. **Security**:
   - Sandboxed Docker containers.
   - Code execution isolated per user session.

Experimental Procedure

- Participants coded identical assignments in both environments.
- Metrics recorded using **server logs, browser profiling tools, and feedback surveys**.
- Data analyzed using **paired t-tests** to measure significance of differences.

RESULTS

The prototype demonstrated:

- **Minimal latency increase** for small projects (sub-10%).
- **Significant time savings** in onboarding and environment setup.
- **Higher collaboration scores**, particularly in group debugging sessions.
- Slightly higher **memory footprint**, primarily due to browser runtime overhead.

Developer feedback emphasized:

- Reduced friction in starting projects.
- Portability (work from any device with internet).
- Limitations in offline usage and heavy build performance.

CONCLUSION

This research demonstrates that browser-based IDEs for C/C++ compilation can transition from being niche experimental tools to practical, high-value development environments capable of competing with traditional desktop IDEs in most general-purpose programming scenarios. By combining a containerized compilation backend with WebAssembly-powered in-browser execution, the proposed system

effectively bridges the gap between the accessibility of web-based platforms and the raw performance traditionally reserved for native environments.

The empirical evidence from this study highlights key strengths—most notably, the dramatic reduction in environment setup time, device independence, and the seamless integration of real-time collaboration features—that directly address long-standing pain points in distributed software development and programming education. While slight performance trade-offs exist in terms of compilation speed and memory overhead for large-scale builds, these drawbacks are outweighed by significant gains in developer productivity and operational flexibility.

Furthermore, the modular nature of the system architecture makes it adaptable for future enhancements, such as **AI-assisted coding for intelligent auto-completion and error detection, GPU-accelerated builds for resource-intensive compilation, and offline-first capabilities via Progressive Web Apps**. These innovations, combined with advancements in **quantum-safe compilation pipelines and integrated cloud-based version control**, can elevate browser-based C/C++ IDEs into mainstream adoption within both academic curricula and industry-standard toolchains.

In conclusion, the findings affirm that browser-based IDEs are not merely convenient alternatives but strategic enablers of modern software development—transforming how programmers, educators, and collaborative teams approach C/C++ coding in an increasingly decentralized, cloud-driven technological ecosystem.

FUTURE SCOPE OF STUDY

Future research could focus on:

1. **AI-Assisted Coding** – Integrating AI code completion, bug detection, and optimization recommendations.
2. **Offline Mode** – Using Progressive Web Apps (PWAs) and local WASM-based compilation.
3. **Quantum-Safe Compilation Pipelines** – Ensuring code security against quantum attacks.
4. **GPU-Accelerated Compilation** – Leveraging WebGPU for faster builds in-browser.
5. **Deep Integration with Version Control** – Real-time Git collaboration inside the IDE.

Such enhancements could further reduce the gap between cloud-based and native C/C++ development environments, making browser-based IDEs mainstream in both industry and academia.

REFERENCES

- https://www.researchgate.net/publication/262243584/figure/fig4/AS:667640200716293@1536189236221/Typical-IDE-components-in-a-modern-IDE-and-their-dependencies-Adapted-from-23.ppm
- https://www.researchgate.net/publication/261212560/figure/fig1/AS:475677343391744@1490421724407/Flow-Diagram-of-the-Proposed-Model-for-Collaborative-Software-Development.png
- *Baldwin, M., & Singh, A. (2022). Performance optimization in cloud-based compilers. Journal of Cloud Computing Research, 11(3), 145–162.*
- *Brown, T. (2020). Container security for online development environments. International Journal of Secure Software Engineering, 8(2), 89–104.*
- *Chen, L., & Zhao, X. (2021). Cloud-native IDEs for modern software development. IEEE Software, 38(4), 27–34.*
- *Collison, P., & Lee, M. (2022). Trends in remote software development tools. Software Engineering Review, 14(2), 88–101.*
- *Emscripten Project. (2023). Emscripten documentation. Retrieved from https://emscripten.org*
- *Gitpod. (2023). Gitpod cloud IDE overview. Retrieved from https://www.gitpod.io*
- *Herman, D. (2021). WebAssembly and the future of browser-based applications. ACM Queue, 19(4), 66–77.*
- *Hu, Y., & Wang, J. (2020). Comparative study of browser-based vs. desktop IDEs. Journal of Web Engineering, 19(5), 451–472.*
- *IDEOne. (2023). Online compiler and debugging tool. Retrieved from https://ideone.com*
- *JDoodle. (2023). JDoodle online IDE. Retrieved from https://www.jdoodle.com*
- *Kölling, M. (2021). Improving computer science education through browser-based tools. Education and Information Technologies, 26(5), 5679–5694.*
- *Microsoft. (2023). Visual Studio IDE overview. Retrieved from https://visualstudio.microsoft.com*
- *Monaco Editor. (2023). Monaco Editor documentation. Retrieved from https://microsoft.github.io/monaco-editor/*
- *Repl.it. (2023). Replit coding environment. Retrieved from https://replit.com*
- *Shen, W., Li, P., & Chen, G. (2020). WebAssembly performance evaluation for C/C++ compilation. Proceedings of the Web Conference, 159–168.*
- *Visual Studio Code. (2023). VS Code documentation. Retrieved from https://code.visualstudio.com*
- *Wang, Z., & Liu, H. (2022). Remote development in distributed software teams. IEEE Transactions on Software Engineering, 48(12), 4992–5006.*
- *Zhang, Y., Sun, M., & Li, J. (2019). Containerized compilation for secure online programming. Future Generation Computer Systems, 95, 586–597.*
- *Zhao, R., & He, Q. (2021). Collaborative coding in the cloud: A systematic review. Journal of Systems and Software, 174, 110894.*
- *Zhou, P., & Yuan, S. (2020). The adoption of cloud IDEs in academic programming courses. Computer Applications in Engineering Education, 28(6), 1392–1405.*