The Role of Web-Based IDEs in Modern Computer Science Education

Emir Taalaibekov
Ala-Too International University, Bishkek, Kyrgyzstan
Email: emir.taalaibekov@aiu.edu.kg

***Abstract*—Web-based integrated development environments (IDEs) have emerged as a significant tool in computer science education. By providing accessible, browser-based programming platforms, they eliminate the need for local software installation and configuration, thus lowering entry barriers for novice programmers. This article examines the role of web-based IDEs in modern CS education, comparing them to traditional desktop IDEs and reviewing their applications in pedagogy. We discuss the advantages of web-based IDEs in facilitating hands-on learning, collaborative coding, and scalable course management, as well as the challenges and limitations such as reduced feature sets and reliance on internet connectivity. Drawing on recent academic studies and practical deployments, we highlight how web-based IDEs can enhance learning outcomes and student engagement while addressing common pain points in teaching programming. The paper concludes that web-based IDEs are a valuable complement to traditional development environments in education, particularly for introductory programming and remote learning scenarios, although they are not without limitations.**

***Index Terms*—Web-based IDE, online IDE, programming education, educational technology, cloud IDE**

I. Introduction

In recent years, web-based integrated development environments (IDEs) have gained popularity as platforms for teaching and learning programming. A web-based IDE is an IDE that runs entirely in the web browser, requiring no local installation of compilers, interpreters, or development tools. This approach offers a uniform and readily accessible programming environment for students, which can significantly reduce the setup time and configuration effort typically associated with traditional desktop IDEs [1]. Prior work has noted that installing and configuring a full programming environment can be a frustrating and time-consuming task for novices, often distracting them from fundamental learning objectives [1]. By contrast, a cloud-hosted or browser-based IDE allows students to begin coding immediately, which can help them focus on learning programming concepts rather than environment configuration.

The appeal of web-based IDEs in education is multifaceted. First, the "zero-installation" nature of these IDEs means that any student with a web browser can access the same standardized environment. This uniformity helps ensure all students work under identical conditions, thereby improving the reproducibility of programming assignments and easing the support burden on instructors [1]. Second, web-based IDEs offer accessibility across various devices and operating systems; students can code from school computers, personal laptops, or even tablets without compatibility issues. Third, many web-based IDE platforms support real-time collaboration and easy sharing of code, features which enable pair programming exercises and remote tutoring in ways that traditional IDE setups may not easily support [4]. The rise of online learning, especially during the COVID-19 pandemic, further underscored the importance of having robust online coding platforms that allow classes to continue without requiring physical labs or complex student setup. Educational platforms and MOOCs (Massive Open Online Courses) increasingly integrate web-based coding environments directly into their instructional materials, reflecting a broader trend towards cloud-based educational tools.

This paper explores the role and impact of web-based IDEs in modern computer science (CS) education. We provide an overview of related work and existing systems, compare web-based IDEs with traditional IDEs in terms of functionality and user experience, discuss pedagogical applications and benefits observed in classroom and online settings, and outline the challenges and limitations faced when adopting web-based IDEs for teaching. Through this analysis, we aim to elucidate how web-based IDEs can be effectively leveraged to improve CS education and what considerations educators should keep in mind when using these tools.

II. Related Work

Numerous systems and studies have contributed to the development of web-based IDEs for education. Early explorations of browser-based programming environments date back over a decade. For instance, van Deursen *et al.* introduced *Adinda*, a "knowledgeable" browser-based IDE, demonstrating the feasibility of moving development tools to the web and integrating tutorial knowledge into the coding environment [5]. Around the same time, Jenkins *et al.* developed *JavaWIDE*, an online IDE designed to support Java programming courses with a simplified interface accessible through the web [6]. These pioneering efforts underscored the potential of web IDEs to lower barriers for students by providing ready-to-use programming workspaces.

In the years since, a variety of web-based IDE platforms have emerged both in academic research and as commercial or open-source offerings. Cloud-based IDEs such as AWS Cloud9, Eclipse Che, CodeAnywhere, and Repl.it gained traction by offering full development environments in the cloud accessible via browser. In educational contexts, researchers

have implemented custom web IDEs tailored to specific needs. For example, Kodethon is a web IDE developed at UC Davis and used across several courses; it supports multiple languages and includes features like syntax highlighting, an integrated shell, and even real-time collaboration to facilitate pair programming and live instructor assistance [1]. Over three years of deployment, Kodethon was adopted by thousands of students as an optional programming environment, and studies of its use collected student feedback on its usefulness and usability [1].

Other studies have examined how web-based coding tools affect student learning outcomes. Benotti *et al.* evaluated a web-based coding tool that provided automatic feedback on programming exercises, finding positive effects on students' performance and perceptions [3]. This suggests that web IDEs can be effectively combined with automated assessment and feedback systems to enhance learning. In addition, the proliferation of interactive online learning platforms (such as Khan Academy and free online judges) has accustomed students to practicing programming in a browser environment. Tools like *Online Python Tutor*, while not full IDEs, further illustrate the educational value of in-browser tools by enabling students to visualize code execution step by step on the web [7].

More recently, the integration of web-based IDEs into software engineering education has expanded beyond introductory programming. For example, Neelakanta Iyer *et al.* present an online IDE specifically designed for DevOps education, which packages complex toolchains (like Docker and Kubernetes) into a cloud environment for student use [8]. Their findings indicate that such specialized web IDEs can help overcome accessibility barriers (e.g., limited hardware or software setup) and are well-received by students preparing for modern software engineering practices. Similarly, Frankford *et al.* investigated requirements for integrating an online IDE into an automated programming assessment system (APAS) and developed a prototype addressing key missing features like syntax highlighting and autocompletion [2]. These and other efforts in the literature highlight a growing interest in harnessing web-based development environments to improve and broaden computer science education.

III. Comparison with Traditional IDEs

Web-based IDEs differ from traditional desktop IDEs in several important ways that affect their use in educational settings. In terms of setup and accessibility, the advantages of web IDEs are evident: traditional IDEs require installing software development kits (SDKs), configuring environment variables, and ensuring compatibility with the operating system, whereas a web IDE can be launched instantly in a browser with minimal effort [1]. This not only saves time at the beginning of a course or assignment but also avoids issues of students having misconfigured systems. All students using a web IDE access an identical environment hosted in the cloud, which standardizes the development experience and can prevent the notorious "it works on my machine" problem during assignments.

Another key difference lies in resource requirements and computing power. Traditional IDEs run locally and leverage the student's machine resources; this can be a limitation for students with older or less powerful computers, especially when working with resource-intensive tools. Cloud-based IDEs offload much of the computation to a server. While this means even a lightweight Chromebook can handle coding tasks that might be slow on its own hardware, it also introduces dependency on internet connectivity and server reliability. Network latency or downtime can directly impact the user experience in a web IDE, whereas a desktop IDE remains usable offline and is only limited by local performance. Students have reported that network slowdowns can cause lag in web IDEs, an issue nonexistent in offline setups [1]. In critical learning moments, such delays might frustrate learners, highlighting that reliability and internet access are essential considerations when adopting web-based IDEs.

Feature richness is another point of comparison. Modern desktop IDEs (such as IntelliJ IDEA, Visual Studio, or Eclipse) are often loaded with advanced features: powerful debuggers, extensive plugin ecosystems, customizable interfaces, and offline support for large codebases. Web-based IDEs historically offered a more streamlined feature set to prioritize simplicity and performance in the browser environment. For instance, early or basic web IDEs might lack a full debugger or support for advanced language refactoring tools. Studies have found that while beginners appreciate the simplicity of web IDEs, more advanced students sometimes find web IDEs insufficient for complex projects, requesting features like better code navigation, debugging capabilities, or editor customizations (such as Vim keybindings) that they are accustomed to in traditional environments [2]. However, the gap is closing: many web IDE platforms now include debugging support, integration with version control, and even containerized environments to simulate real-world development stacks. The trade-off often comes down to convenience versus completeness. In educational contexts, where the goal is to teach programming concepts rather than tool intricacies, a slightly limited feature set is usually acceptable if it means smoother onboarding and fewer distractions.

Collaboration and sharing features tend to be more naturally integrated in web-based IDEs. Because the development session runs on a central server, it is easy to enable multiple users to view or edit the same code simultaneously, or to share a project link with an instructor for assistance. While it is possible to collaborate using traditional IDEs (using plugins or external version control like Git), web IDEs streamline this process. Goldman *et al.* demonstrated real-time collaborative coding in a web IDE as early as 2011, showing how multiple programmers could effectively work together through a browser interface [4]. Such capabilities align well with modern pedagogical practices like pair programming, remote office hours, and peer review of code. Traditional IDEs are catching up with live collaboration features (for example, Microsoft

Visual Studio Code's Live Share extension), but these often require additional setup. With web IDEs, collaboration is typically a built-in feature, requiring only that participants have an internet connection and a browser.

In summary, compared to traditional IDEs, web-based IDEs excel in ease of access, uniformity of environment, and collaborative affordances, while they may lag in certain advanced functionalities and are dependent on internet connectivity. For many educational scenarios, the strengths of web IDEs align with the needs of instructors and novice students, though it is important to recognize scenarios where a desktop IDE might be more appropriate (for example, in an upper-level systems programming course that requires fine-grained control over the local environment, or when students need to learn the process of setting up a development toolchain as a learning outcome in itself).

IV. Pedagogical Applications

Web-based IDEs have been leveraged in various pedagogical contexts within computer science education. One common use is in introductory programming courses, where students often struggle with the mechanics of setting up a programming environment. By using a web IDE in such courses, instructors can ensure that all students start coding from day one without technical hurdles. The Kodethon system is an illustrative example: it was deployed as an optional tool across multiple programming classes, and roughly one-third of students chose to use the web IDE for writing their programs [1]. The primary reasons cited for using Kodethon were its web-based nature and the lack of installation required—factors directly tied to reducing overhead for students [1]. Even students who opted not to use the web IDE acknowledged that it provided a useful service, indicating that it fills a niche as a convenient learning tool.

Another pedagogical application is in large-scale courses and online courses, including MOOCs. Web-based IDEs embedded in online learning platforms allow tens of thousands of learners to write and execute code in their browser as they go through lessons. This integration has been shown to improve engagement, as immediate feedback and the ability to experiment with code examples make the learning experience more interactive. Benotti *et al.*'s study on an automatic feedback-generating web coding tool demonstrated that such integration can positively influence student performance by catching mistakes early and guiding learners to correct them [3]. From a teaching standpoint, having every student use an identical cloud environment also simplifies the task of providing support and grading. In automated assessment systems, a web IDE can be tied directly into the submission and evaluation pipeline. Frankford *et al.* describe how an online IDE integrated with an APAS allows students to write and test code in the same environment where automated tests will run, reducing discrepancies between student testing and official grading [2]. This kind of seamless workflow can enhance fairness and clarity in assessment.

Web IDEs also facilitate new forms of collaborative learning. In lab sessions or live coding classes, an instructor can ask students to share their code via a web IDE link, making it easy to demo solutions or to debug a student's code in real-time. Some platforms support a teacher view where the instructor can monitor all student coding sessions concurrently, which is valuable for identifying who is stuck on a problem during class. Collaborative features also enable pair or group programming assignments even when students are not physically co-located. This was particularly significant during periods of remote instruction; students could continue pair programming using the collaborative editing features of web IDEs, and instructors could join a student's session to provide help as needed, all through the browser.

Furthermore, web-based IDEs are being used to introduce students to professional development practices in a controlled environment. For example, some courses use cloud IDEs to teach web development or DevOps, where the environment can simulate deploying applications to the cloud without requiring students to install complex software locally [8]. These environments can include pre-configured containers or sandboxes that replicate real-world stacks (for instance, a web IDE might provide a Linux terminal with Docker installed, so students can practice containerization within their browser). This approach gives students hands-on experience with industry-relevant tools while avoiding the setup pitfalls that might otherwise consume a lot of course time.

Finally, the data tracking capabilities of web IDEs offer pedagogical insights. Because student interactions with the IDE occur on a server, it is possible to log compile errors, runtime outputs, and even keystrokes or time spent on tasks (within privacy and ethical constraints). Such data can be analyzed to identify common misconceptions (e.g., frequent syntax errors) or to provide analytics to instructors about how students are progressing through coding assignments. For instance, Kim *et al.* describe a system called Watcher that uses a web IDE to automatically collect coding activity data, helping instructors detect plagiarism and better understand student coding behaviors in an online class [9]. This kind of tool shows that beyond direct student learning, web IDEs can also empower educators with feedback and research data to refine their teaching methods.

Overall, web-based IDEs are versatile in pedagogical application: they act as enablers for beginner-friendly programming instruction, as platforms for scaling CS education to large audiences, and as conduits for innovative teaching techniques such as live collaboration and data-driven instruction.

V. Challenges and Limitations

Despite their advantages, web-based IDEs come with a set of challenges and limitations that educators and institutions must consider:

1) **Limited Feature Set for Advanced Usage:** While web IDEs cover the essentials of editing, compiling, and running code, they may lack some of the advanced features

found in mature desktop IDEs. For example, sophisticated debugging tools, extensive language-specific refactoring support, or certain plugins might not be available or as powerful in a browser environment. As students progress to more complex projects, these limitations can become apparent. A study of student perceptions found that some users of an educational web IDE felt it was not useful for more advanced assignments and desired features like improved autocompletion, debugging, and editor customization options [2]. Thus, a web-based IDE might need to be supplemented or eventually replaced by a desktop IDE as students move to higher-level courses or larger software projects.

2) **Performance and Reliability Issues:** The performance of a web-based IDE is contingent on internet bandwidth and server-side resources. Network latency can introduce delays in typing or in seeing program output, which can disrupt the coding process and frustrate users. Likewise, if the server hosting the IDE experiences high load or downtime, students might be unable to work on their assignments. Such outages can be highly disruptive, especially if they occur close to assignment deadlines. In contrast, a local IDE does not depend on network connectivity (once all tools are installed) and can generally offer a more responsive experience. Reports from deployments of web IDEs note that occasional network glitches or slowdowns negatively impacted the user experience for some students, an issue nonexistent in offline IDE use [1]. Ensuring adequate infrastructure and providing offline fallbacks or extensions (when possible) is important to mitigate this limitation.
3) **Learning Curve and User Interface Differences:** Introducing a web-based IDE means students must adapt to its interface, which might differ from traditional IDEs that they encounter in other contexts. While many web IDEs strive for simplicity, the transition can still pose a learning curve. Some students may initially find a web IDE unfamiliar or less efficient if they have prior experience with a desktop environment. Additionally, if a course later expects students to know how to use a standard IDE or certain developer tools locally, exclusive reliance on a web IDE might leave a gap in their tool knowledge. Instructors should clarify whether the goal is to teach tool usage or to abstract it away. Studies have noted that some students and instructors worry that using a web-based environment exclusively could lead students to conflate programming proficiency with the ability to set up systems (so-called "system building" skills) [1]. This concern can be addressed by making learning objectives explicit: for beginner courses focusing on programming concepts, using a web IDE is appropriate, whereas courses aiming to teach software tooling might intentionally require a traditional IDE.
4) **Dependence on External Services and Cost:** Adopting a web-based IDE often means relying on third-party services or maintaining a custom server. If an instructor uses a free online IDE service, they must consider data privacy (student code residing on external servers) and longevity (services can change terms or shut down, as was the case with some earlier cloud IDE providers). On the other hand, running an in-house web IDE (like an open-source solution on university servers) involves costs for infrastructure and technical support. The team behind Kodethon, for example, spent years developing and scaling their own web IDE to meet the needs of thousands of students [1]. They caution that building a reliable custom web IDE requires significant effort and technical resources, and suggest that educators evaluate existing platforms before deciding to create their own. In any case, the sustainability and support for the chosen solution should be planned – whether that means budgeting for a commercial cloud service or ensuring university IT support for a self-hosted system.

It is clear that web-based IDEs are not a one-size-fits-all solution. They shine in many scenarios, but educators should remain aware of when these tools might fall short. Blending approaches (e.g., offering a web IDE for early assignments and allowing or encouraging a switch to a local IDE for more complex tasks) can sometimes provide the best of both worlds. Ultimately, understanding the limitations of web IDEs helps in designing curriculum and support structures that maximize their benefits while providing alternatives or workarounds when necessary.

VI. Conclusion

Web-based IDEs represent a transformative development in computer science education, aligning with the needs of modern learners for convenience, accessibility, and collaboration. As discussed, these platforms lower the entry barrier to programming by removing the overhead of environment setup and ensuring a consistent workspace for all students. Empirical studies and experience reports show that web IDEs can improve students' focus on programming tasks by minimizing technical distractions [1], and that they can be effectively integrated with automated assessment and feedback systems to enhance learning outcomes [3][2]. Features intrinsic to web IDEs—such as instant sharing of code and real-time collaborative editing—open up new pedagogical possibilities, from pair programming in remote classes to live coding assistance, which have been well-received in educational contexts [4].

At the same time, it is evident that web-based IDEs are complementary to, rather than outright replacements for, traditional IDEs. For advanced coursework or professional preparation, students may still need exposure to the richer functionalities of desktop development environments and to the skills of configuring and managing those tools. The limitations of web IDEs, including their dependence on internet connectivity and sometimes reduced feature set, mean that educators should adopt them thoughtfully. In many cases, the optimal strategy is a hybrid one: leveraging web-based IDEs for their strengths in early and intermediate learning stages or for specific activities

(like auto-graded exercises and collaborative projects), while gradually introducing traditional tools when appropriate.

Looking forward, the gap between web IDEs and traditional IDEs continues to narrow. With improvements in browser technologies and cloud infrastructure, new web-based IDE offerings are becoming more powerful and more akin to full-fledged development studios. Additionally, the lessons learned from educational deployments provide guidance for future systems to address current shortcomings (for example, by adding offline modes, better performance optimization, and more extensibility for advanced users). The ongoing evolution of these tools is likely to further cement the role of web-based IDEs in CS education.

In conclusion, web-based IDEs have proven to be a valuable asset in modern computer science education, promoting greater accessibility and innovation in teaching practices. They enable students to engage with programming in a flexible, user-friendly environment, and help educators streamline course logistics and focus on teaching concepts. By acknowledging their limitations and using them in conjunction with traditional methods, instructors can create a robust learning ecosystem that prepares students both in foundational programming skills and in practical software development competencies. The continued refinement and adoption of web-based IDEs will undoubtedly play a significant role in shaping the future landscape of programming education.

REFERENCES

[1] M. Velez, M. Yen, and A. Alipour, "Student Adoption and Perceptions of a Web Integrated Development Environment: An Experience Report," in *Proc. 51st ACM Technical Symposium on Computer Science Education (SIGCSE)*, Portland, OR, USA, 2020, pp. 604–610. DOI: 10.1145/3328778.3366949.

[2] E. Frankford, D. Crazzolara, M. Vierhauser, N. Meißner, S. Krusche, and R. Breu, "An Online Integrated Development Environment for Automated Programming Assessment Systems," in *Proc. 17th Int. Conf. on Computer Supported Education (CSEDU)*, 2025, pp. 48–59. DOI: 10.5220/0013203000003932.

[3] L. Benotti, F. Aloi, F. Bulgarelli, and M. J. Gomez, "The Effect of a Web-based Coding Tool with Automatic Feedback on Students' Performance and Perceptions," in *Proc. 49th ACM Technical Symposium on Computer Science Education (SIGCSE)*, 2018, pp. 2–7. DOI: 10.1145/3159450.3159579.

[4] M. Goldman, G. Little, and R. C. Miller, "Real-time collaborative coding in a web IDE," in *Proc. 24th ACM Symposium on User Interface Software and Technology (UIST)*, 2011, pp. 155–164. DOI: 10.1145/2047196.2047215.

[5] A. van Deursen, A. Mesbah, B. Cornelissen, A. Zaidman, M. Pinzger, and A. Guzzi, "Adinda: A knowledgeable, browser-based IDE," in *Proc. 32nd ACM/IEEE Int. Conf. on Software Engineering (ICSE)*, vol. 2, 2010, pp. 203–206. DOI: 10.1145/1810295.1810330.

[6] J. Jenkins, E. Brannock, and S. Dekhane, "JavaWIDE: Innovation in an online IDE," *J. Comput. Sci. Coll.*, vol. 25, no. 5, pp. 102–104, May 2010.

[7] P. J. Guo, "Online Python Tutor: Embeddable web-based program visualization for CS education," in *Proc. 44th ACM Technical Symposium on Computer Science Education (SIGCSE)*, 2013, pp. 579–584. DOI: 10.1145/2445196.2445368.

[8] G. N. Iyer, A. G. Yisheng, M. C. H. Er, W. X. Choong, and S. W. Koh, "A Web-Based IDE for DevOps Learning in Software Engineering Higher Education," arXiv preprint arXiv:2501.10363, 2024. DOI: 10.48550/arXiv.2501.10363.

[9] Y. Kim, K. Lee, and H. Park, "Watcher: Cloud-based coding activity tracker for fair evaluation of programming assignments," *Sensors*, vol. 22, no. 19, Article 7284, 2022. DOI: 10.3390/s22197284.