JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

# BROWSER BASED IDE

**[1]Hitesh Ramnani, [2]Shubham Bakhal, [3]Tanishq Pawar, [4]Manish Basargekar, [5]Dr. (Mrs.) R. A. Khan**
Computer Engineering Department,
M. E. S. College of Engineering Pune – 411001, India

*Abstract :* The potential advantages of a browser-based Integrated Development Environment (IDE) for programming assignments and auto-grading answers are examined in this research paper. There is a demand for scalable and cost-effective alternatives due to the rising demand for programming exercises and the prohibitive cost of standard IDEs. This study intends to solve these issues and investigate the implications of using browser-based IDEs for code creation and collaboration by utilizing the features of online IDEs.

The high expense of conventional IDEs and the barriers to collaboration and accessibility they impose are the research problems found in this study. This study aims to examine the viability and efficacy of browser-based IDEs for tasks and auto-grading solutions, as well as their potential cost-savings and influence on scalability and code collaboration.

The methodology used in this study includes conducting a thorough review of the body of knowledge on browser-based IDEs, examining case studies of businesses that have adopted them, and gathering information through surveys and interviews with programmers and instructors.

The main conclusions of this study are that browser-based IDEs provide a workable and affordable alternative to conventional IDEs. Greater accessibility is made possible by these online IDEs, which don't require installation and rely less on local hardware resources. They also make it possible for numerous users to work on the same project at once by facilitating fluid code collaboration. The report also emphasizes the advantages of open-source online IDEs that support a neighborhood-driven environment for code development and sharing.

This study has important repercussions for academic institutions, teachers of programming, and programmers. By removing the need to buy and maintain pricey software licenses, the adoption of browser-based IDEs can result in significant cost savings for universities. Online IDEs' improved accessibility and collaboration features also improve student learning and encourage more productive and efficient code development practices in business environments.

In summary, this study offers insightful information about the advantages of using browser-based IDEs for assignments and auto-grading answers. The results demonstrate the possibility for cost reductions, scalability, and enhanced code collaboration. The results of this study may encourage the use of browser-based IDEs and promote practices in software development and programming education.

***Keywords:*** Assignments, Auto-grading solutions, excessive expense, programming-exercises, cost saving, scalable, Integrated Development Environment (IDE), internet connection, open-source, online-IDE, code collaboration.

---

## I. INTRODUCTION

The need for effective and accessible Integrated Development Environments (IDEs) is critical in the current digital era, as software development and programming have become essential elements of many disciplines. Programmers may write, edit, build, and debug code on a complete platform thanks to IDEs, which speed up the software development process. Traditional IDEs, however, are sometimes quite expensive, only partially accessible, and difficult to collaborate with.

This research study investigates the idea of a browser-based integrated development environment (IDE), a web-based platform that enables programming assignments and auto-grading solutions while providing cost-saving advantages, scalability, and improved code collaboration. The main objective is to examine the viability and effects of using browser-based IDEs in settings for professional software development and programming education.

The current research issue is on the prohibitive cost of traditional IDEs, which limits their scalability and makes them less accessible to educational institutions. Additionally, the effectiveness and productivity of developers working on collaborative projects are limited by the absence of functional code collaboration tools. These difficulties investigated into browser-based IDEs as a potential substitute that can get around them and provide other benefits.

This research paper has three main goals: first, it evaluates the viability and efficacy of browser-based IDEs for programming assignments and auto-grading solutions; second, it assesses the potential cost savings of switching from traditional IDEs to browser-based IDEs; and third, it examines the influence of browser-based IDEs on code collaboration and scalability in educational and professional settings.

A detailed technique has been used to accomplish these goals. To gain knowledge of the capabilities, benefits, and limits of online IDEs, a thorough analysis of the current literature on the subject was carried out. In order to comprehend the practical uses and advantages of browser-based IDEs, case studies of organizations that have used them have been examined. To acquire factual data and direct experiences with utilizing browser-based IDEs, surveys and interviews with developers and programming teachers have been done.

In conclusion, by investigating the possibilities of browser-based IDEs, this research article seeks to make a contribution to the fields of programming education and software development practices. The results will clarify their viability, the advantages of cost savings, and how they will affect code cooperation. Browser-based IDEs have the ability to completely change the way programming is taught and practiced by offering an accessible and collaborative platform for programming assignments and auto-grading solutions.

## II. METHODS

Several sub-processes made up the comparative analysis process between the online IDEs, as shown in Figure 1. We started by searching for online IDEs that support the various programming language, are accessible for free, and don't need checking in on the Google search engine using pertinent keywords. To guarantee the inclusion of broadly accessible solutions, this step was essential. The page ranking mechanism of the Google search engine, which prioritizes search results based on relevance and makes use of numerous approaches to increase search efficiency, was the main reason we picked it.

The second phase was a thorough literature research to compile data on each cited online IDE. For this evaluation, extensive study of pertinent scientific literature was required. The main search engine used to obtain scholarly material was Google Scholar. The results and discussion portion of this research report analyzed and presented the findings from the literature review. Through this method, we were able to learn more about the characteristics, capabilities, and user interfaces of each online IDE.

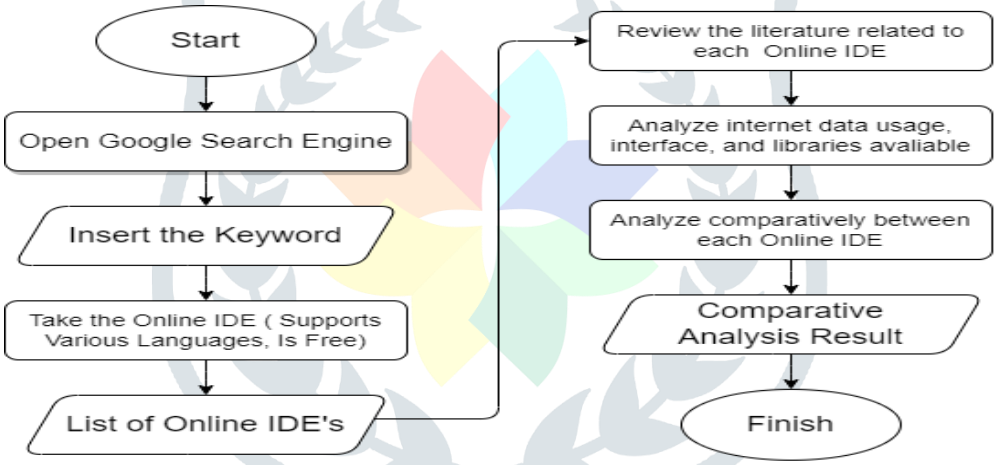


*Figure 1. Process of comparative analysis*

We experimented by creating Programming code from data structure material to gauge each online IDE's utilization of the internet. Then, in order to gauge and contrast each online IDE's data use, we used a Google Chrome plugin called Data use. We were able to learn how much data each website used to execute code thanks to this plugin. We might assess the effectiveness and possible effects on consumer internet connectivity by analyzing internet data use.

We directly collected the essential information from the primary websites of the relevant platforms in order to gather information on the capabilities that are offered by each online IDE. Using this method, we were able to compile reliable and recent information on the features, functions, and tools provided by each online IDE.

The final phase was performing a comparison of the various online IDEs based on criteria such as internet data utilization and feature availability. This comparative analysis approach has been used in numerous research across numerous fields.

We were able to perform a thorough comparison study of the selected online IDEs by adhering to these methodological approaches. This study is the basis for the results and discussion section, in which we present and debate the results to offer insightful information about the advantages and disadvantages of each online IDE in terms of internet data use and feature availability.

## III. RESULTS

In this study, we compared several programming languages and environments that provide alternatives to traditional development settings in basic programming courses. These platforms were divided into three categories by us: online learning resources, platforms for online code editors, and online editors with an educational purpose.

| No. | Online IDE Names | Ads | Support For Multiple Languages | Educational Resources | Code Collaboration | User Interface | Free or paid |
|---|---|---|---|---|---|---|---|
| 1. | Rextester | No | Yes | No | No | Complex | Free |
| 2. | jDoodle | No | Yes | No | No | Complex | Free |
| 3. | Ideone | Yes | Yes | No | No | Complex | Free |
| 4. | Repl.it | No | Yes | No | No | Easy | Both |
| 5. | Tutorials Point | Yes | Yes | Yes | No | Complex | Both |
| 6. | CodeChef | No | Yes | Yes | No | Easy | Both |
| 7. | Programiz | Yes | Yes | Yes | No | Complex | Both |
| 8. | Code Blocks | Yes | Yes | Yes | No | Easy | Both |

*Table 1. Experiment Result*

Online learning tools like Khan Academy and Code Academy make up the first category. These online resources include practice activities and step-by-step learning aids, such as written explanations and video lessons. However, they don't cover all phases in the problem-solving process and aren't integrated into curriculum. They also lack capabilities like planning, manual testing, and debugging.

Platforms for online code editors like CodingGrond, Rextester, jDoodle, and Ideone are included in the second group. These platforms provide web-based IDEs that can accept one or more files and occasionally come with an integrated console. Although they provide common input and output, they do not offer a separate area for task descriptions or phases that have been organized methodically. Furthermore, testing possibilities are constrained. These platforms provide as examples of how web-based platforms may be used to develop command-line browser applications.

Platforms like CodeChef or Codewars, which mix features of online programming contests with learning how to code, are included in the third category. Task descriptions, automated tests, and occasionally manual tests are available. These platforms don't, however, allow for unique work assignments from teachers or provide thorough activity tracking. They work as a cross between a judging system and an online IDE.
Repl.it distinguishes out among the alternative ecosystems as an online editor with educational functions. It features task assignment, automated tests, tracking of student engagement, and classroom administration. Even while these capabilities have their uses, they do not address manual testing and fall short of meeting all methodology-based requirements.

## IV. Discussion

In this study, we compared several programming languages and environments that provide alternatives to traditional development settings in basic programming courses. These platforms were divided into three categories by us: online learning resources, platforms for online code editors, and online educational editors.

Tools and development environments are used to study the problem-solving process. Understanding the job, which can be delivered orally, on a board, or via digital tools like websites or task libraries, is the first stage. The job description can be in a variety of formats, such as HTML, PDF, or docx. Planning is usually done on paper or a board and includes specification and algorithm design. However, a growing number of students are taking notes on digital devices. This change can be linked to the ease of digital editing as well as the lack of exercise books or pencils. It's vital to remember that planning often uses a different environment or tool than coding.

The problem-solving process gets more interesting and creative when the plan is put into practice. This stage often occurs in a testing environment that provides thorough features for programming language development. Tools for editing, compiling, and executing programs include Integrated Development Environments (IDEs), such as Code Blocks, Visual Studio, and Net Beans. The fundamental disadvantage of IDEs is that they are made for more sophisticated programs than are normally seen in basic courses, while being user-friendly and having default settings. IDEs usually include projects with many files and a variety of features, which might take attention away from the primary duties.

As editors must be set up in classrooms and students must install them on their own computers, installing desktop environments can be difficult. Compatibility with various operating systems and potential dependencies complicate the procedure and increase the risk of mistakes

Coding is followed by testing, which includes checking for syntax and semantic issues before defects are found and fixed. Typically, syntax issues are found during compilation, and mistakes are often noted in the source code or shown in a separate panel. Creating test cases and running the tests are both steps in semantic testing. Test cases can be spoken, printed on the board, or recorded in exercise books. Students often do manual testing by inputting input data into a command-line window and then watching the program's reaction. Longer inputs can be written to a file and then routed to a normal input for the program. Even though development is still predominantly done in separate IDEs, there are occasions when online judging platforms are used to impartially check the code.

The process study exposes a lot of problems with how introductory courses, new students, and programming environments interact with one another. Even if they are extensive, IDEs are not designed for the systematically based, tightly focused problem-solving technique that is best for beginning students. They lack the direction that professors or students themselves must supply and instead concentrate mostly on the execution stage. Students also need to learn how to use the command line, redirect files, upload files, and download files. It takes a lot of effort to teach these tools and the complete toolchain, which might take time away from teaching the principles of problem-solving. Furthermore, for students who are used to gamified settings, adopting professional IDEs might be a big barrier.

The procedure becomes more complicated when you consider all the ancillary platforms and programs that are used. The assignment descriptions, design notes, coding in an IDE, compilation, running, authoring test files, testing with files, and automated testing in a different web application are just a few of the eight different knowledge sources that students must travel to succeed across seven platforms.

Certain prerequisites for an efficient programming environment may be determined when considering the variety of basic courses and the complexity of conventional programming environments. It should support newcomers by making things simpler.

## V. CONCLUSION

The restrictions and difficulties encountered in basic programming classes might be addressed by alternate programming platforms. Online learning options provide individual learning chances. However, some phases of problem solving are not integrated or supported. Online code editor platforms show that web-based command-line programs are possible, but they don't meet all needs of a learning programming environment. Platforms that combine programming contests and learning are more functional, but they still have restrictions on task customization and flexibility. Online editors with an emphasis on education provide beneficial instructional elements but may fall short in meeting specific methodology-based demands.

Our research presents a novel remedy, the Browser Based IDE, to get over these restrictions. To address cost reductions, usability, and code collaboration, this open-source, scalable, online IDE and auto-grader for computer science education was created. The Browser Based IDE strives to provide an effective environment for beginner programming classes by providing a user-friendly interface, thorough functionality, and support for a broad range of programming languages.

To improve the functionality, functionality, and methodology-based aspects of alternative programming platforms, further research and development is required. We can enhance the learning process for novice programmers and support their advancement in the field of programming by solving these difficulties.

## REFERENCES

**[1]** M. Mesihovi´c, V. Ljubovi´c and I. Muharemovi´c, ”Using WebIDE as a distance learning tool for high school programming,” 2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO), 2020, pp. 893- 898, doi: 10.23919/MIPRO48935.2020.9245263.

**[2]** Jonathan Cazalas, Max Barlow, Ibraheem Cazalas, and Chase Robinson. 2022. MOCSIDE: An Open-source and Scalable Online IDE and Auto-Grader for Computer Science Education. In Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2022). Association for Computing Machinery, New York, NY, USA, 1114. https://doi.org/10.1145/3478432.3499125

**[3]** Jeong Yang, Young Lee, Kai H. Chang, Evaluations of JaguarCode: A web-based object-oriented programming environment with static and dynamic visualization, Journal of Systems and Software, Volume 145, 2018, Pages 147-163, ISSN 0164-1212, https://doi.org/10.1016/j.jss.2018.07.037

**[4]** Kusumaningtyas, K., Nugroho, E. D., Priadana, A. (2020). Online Integrated Development Environment (IDE) in Supporting Computer Programming Learning Process during COVID-19 Pandemic: A Comparative Analysis. IJID (International Journal on Informatics for Development), 9(2), 66–71.https://doi.org/10.14421/ijid.2020.09202

**[5]** Vinhthuy Phan and Eric Hicks. 2018. Code4Brownies: an active learning solution for teaching programming and problem solving in the classroom. In Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE 2018). Association for Computing Machinery, New York, NY, USA, 153–158. https://doi.org/10.1145/3197091.3197128

**[6]** Atef Chorfi, Djalal Hedjazi, Sofiane Aouag Djalleleddine Boubiche (2022) Problembased collaborative learning groupware to improve computer programming skills, Behaviour Information Technology, 41:1, 139-158, DOI: 10.1080/0144929X.2020.1795263

**[7]** Korolija, N., Zamuda, A. (2019). On Cloud-Supported Web-Based Integrated Development Environment for Programming DataFlow Architectures. In: Milutinovic, V., Kotlar, M. (eds) Exploring the DataFlow Supercomputing Paradigm. Computer Communications and Networks. Springer, Cham. https://doi.org/10.1007/978-3-030-13803-52

**[8]** Manjunath R, Arunjith C, 2015, Cloud based Integrated Development Environment for Android Devices, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH TECHNOLOGY (IJERT) NCRTS – 2015 (Volume 3 – Issue 27)

**[9]** Renate Andersen, Anders I. Morch Kristina Torine Litherland (2022) Collaborative learning with block-based programming: investigating human-centered artificial intelligence in education, Behaviour Information Technology, 41:9, 1830-1847, DOI: 10.1080/0144929X.2022.2083981