Requirements for an Online Integrated Development Environment for Automated Programming Assessment Systems

Eduard Frankford[a], Daniel Crazzolara[b], Clemens Sauerwein[c], Michael Vierhauser[d] and Ruth Breu[e]

University of Innsbruck, Institute of Computer Science, Technikerstraße 21a, 6020 Innsbruck, Austria

Keywords: Programming Education, Automated Programming Assessment Systems, Integrated Development Environments, Requirements, Online IDE.

Abstract: More and more introductory programming courses are being held online, using Automated Programming Assessment Systems (APASs). Some APASs provide online editors where students can solve and submit their exercises, because some course teachers want to avoid that students have to set up an Integrated Development Environment (IDE) on their PCs, fearing that this could increase the entry barrier to the course. However, most of the available online editors for APASs only provide rudimentary programming support, making it harder to write code and, therefore, have the undesired side effect of increasing the entry barrier to learning programming. To gain a better understanding of the requirements of an online IDE for APASs, we conducted a survey asking 48 APASs users to give their opinions on the importance of different online IDE features. Based on their responses we derived a set of main requirements for an online IDE within APASs. The survey was complemented by a tool review of state-of-the-art online IDEs, to see whether existing online IDEs meet the requirements. Finally, we evaluated whether the online IDEs can be integrated within APASs. This study establishes a framework for online IDEs tailored to APASs, creating the foundation for subsequent improvements.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

Automated assessment of exercises in programming courses is becoming more and more important (Barra et al., 2020). Particularly, in large introductory programming courses, the number of participants can exceed several hundred or even thousands of students, making it infeasible to grade all exercises by hand. Furthermore, recent developments related to the COVID-19 pandemic, or the popularity of massive open online courses (MOOCs) have resulted in a shift towards virtual teaching and learning (Mekterović et al., 2023). As a result, APASs have emerged to support course instructors in handling these large number of students (Mekterović et al., 2023). APASs allow students to submit their exercises online, which are then automatically evaluated, using a build pipeline to execute predefined test cases on the students' code (Krusche and Seitz, 2018). Afterwards, the students receive the test results as feedback, which they can use to further improve their submissions. Most APASs use version control systems combined with continuous integration servers to handle submissions (Chen et al., 2020). This introduces an entry barrier to novice programmers, as most beginners are not yet familiar with version control systems. As a result, some APASs already provide the ability for students to solve the exercises in a built-in online editor (Krusche and Seitz, 2018) (Mekterović et al., 2023). However, currently, most of these editors lack important IDE features, such as Syntax Highlighting, Auto-Completion, or Key Shortcuts, to name but a few. This makes programming in APASs unnecessarily difficult and puts additional burden on students yet again raising the entry barrier. To establish the theoretical foundations to evaluate and improve online IDEs, we seek to address the following research questions:

- **RQ1.** What are the most important requirements for online IDEs as perceived by students?

[a] https://orcid.org/0009-0005-5959-4936
[b] https://orcid.org/0009-0007-3548-4665
[c] https://orcid.org/0009-0009-9464-5080
[d] https://orcid.org/0000-0003-2672-9230
[e] https://orcid.org/0000-0001-7093-4341

Frankford, E., Crazzolara, D., Sauerwein, C., Vierhauser, M. and Breu, R.
Requirements for an Online Integrated Development Environment for Automated Programming Assessment Systems.
DOI: 10.5220/0012556400003693
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 16th International Conference on Computer Supported Education (CSEDU 2024) - Volume 1*, pages 305-313
ISBN: 978-989-758-697-2; ISSN: 2184-5026
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

- **RQ2.** To what extent do existing online IDEs meet these requirements?
- **RQ3.** To what extend, and with what effort can existing online IDEs be integrated within APASs?

Based on these research questions, our initial step involved conducting a comprehensive analysis of features provided by various well known desktop IDEs. Using this foundational analysis, we surveyed APASs users to gain insights about their preferences regarding different IDE features. Based on these derived preferences, we evaluated existing online IDEs to create an extensive list showcasing which requirements are fulfilled and which programming languages are supported by these IDEs. In a third step, we evaluated whether these tools are available as open-source or if they are of a proprietary nature. This helps to get a complete picture of the online IDE landscape.

The remainder of this paper is structured as follows: Section 2 provides an overview of related work. Section 3 elaborates on the research methodology. Section 4 presents the main findings of this study, which are further discussed in Section 5. Section 6 outlines potential constraints of this study, and we conclude in Section 7, summarizing the main insights and reflecting on the broader implications of this research.

2 RELATED WORK

(Kusumaningtyas et al., 2020) compared several online IDEs, specifically evaluating their internet data usage and the availability of supporting libraries. Eleven online IDEs that support Python programming, offer free access, and do not require to login were examined. The analysis revealed that Replit, Codechef, and Ideone have the most comprehensive library support. Notably, Replit was found to require the least data transfer, offering a user-friendly interface that effectively organizes code and output, and is free from distracting advertisements. Thus, Replit is recommended for users with limited internet resources, especially for learning computer programming during the pandemic.

(Sinanaj et al., 2022) compared ten online compilers, assessing their service offerings and execution performance. The following service offerings were analyzed: (1) Open Source, (2), Personal Account, (3), Cloud Storage, (4) Size Cloud Storage, (5) File Size, (6) Programming Language Supported, (7) Choice Compiler Version, (8) Intellisense, (9) Collaborative Real-Time, (10) Permalink Share Code and (11) Debugging Code. Findings showed that OnlineGDB, JDoodle, Replit, and Paiza.IO offered the most services, with JDoodle, CodeInterview, Ideone, WandBox, OnlineGDB, Judge0, and Replit leading in performance. Based on a combined assessment of services, performance, and user-friendliness, JDoodle, OnlineGDB, and Replit were recommended as the top choices for beginners, students, and teachers.

(Satav et al., 2011) presents a comparative study of various desktop IDEs for C, C++, and Java, with the goal of identifying key specifications for developing a comprehensive and optimal IDE. The findings indicate that more popular IDEs tend to offer better user support and provide many features to assist the creation of source code.

(Hausladen et al., 2014) developed an online IDE for embedded systems and defined the following functional requirements: (1) Project and version management capabilities, (2) Control over the build process including configuration of compilers, (3) Ability to download or flash the binary to the target hardware connected to the client's computer, (4) Debugging capabilities, such as controlling the flow of execution with breakpoints, and inspecting variables and memory locations, and (5) Standard source code editing features like syntax highlighting and code indentation. Additionally, they defined the following non functional requirements: (1) Stability and responsiveness, (2) Intuitive user interface and (3) Extensibility regarding plugins.

(Tran et al., 2013) developed Ideol, an IDE that significantly focuses on collaborative programming and learning features. Key features include a real-time interactive editor for simultaneous multi-user editing and instant change viewing, a real-time discussion board with enhanced tagging and commenting capabilities. Furthermore, it allows debugging, along with an interactive execution tool to provide support for C/C++ programming. Designed for a wide range of users, from beginners to experts, Ideol offers a user-friendly interface and is compatible with HTML5-supported browsers on various platforms, including partial tablet support.

The related work shows that few scientific research has been conducted regarding the definition of requirements of online IDEs. We found that some developers of online IDEs defined specific functional and non functional requirements for their developed IDEs. Additionally, online IDEs have only been compared regarding their internet data usage, availability of libraries, performance and specific offered services. However, a scientific requirements definition and a detailed comparison of IDE features focusing on supporting novice programmers is missing. We seek to address this gap by first creating a requirements checklist for online IDEs including common

IDE features and then conducting a tool review to analyze what existing tools fulfill which requirements.

3 METHODOLOGY

To address the aforementioned research questions (cf. Section 1), we employed a two-stage approach: first, we collected requirements from APASs users, and second, we conducted a systematic tool review. In the following, we provide a detailed description of each step.

3.1 Requirements Specification

In order to gain an overview of the main requirements for an online IDE in context of APASs, we first conducted a survey asking APASs users to rate the importance of common (non-online) IDE features and to mention whether they are missing some features. The survey was distributed to APAS users from the following universities: University of Innsbruck, Johannes Kepler University Linz and Paris Lodron University Salzburg. To reach as many participants as possible, the survey was carried out online, and was promoted directly on the APAS Artemis hosted at the University of Innsbruck [1]. The survey was designed to be completely anonymous using the LimeSurvey tool[2].

3.1.1 Survey

The survey was divided into three parts. First, we asked participants about their familiarity and experiences in using APASs (P). Second, we asked them to rate IDE features with regards to their perceived importance (F), and finally, two open-ended questions concluded the questionnaire (O).

For the first part, we asked participants the following questions, with the main intention to collect initial information about the participants of the survey

- *P1: In which semester are you currently in?*
- *P2: In how many university courses (including current ones) did you use Artemis?*
- *P3: What did you use to solve the programming exercises in Artemis?*
  (Single choice regarding five options ranging from "online editor only" to "local IDE only")

In the second part, the survey included a list of IDE features and participants were able to rate these regarding their perceived importance. The participants were asked to rate each of the listed features using a 5-point Likert scale ranging from "Very Important" (1) to "Very Unimportant" (5). Each feature included a brief description and screenshot to assist the participants in understanding its functionality. Such a description is depicted in Figure 1.

The following list shows all the features we evaluated and the corresponding descriptions displayed in the survey:

- *F0: Syntax Highlighting - distinguishes various parts of source code using colors.*
- *F1: Debugger / Breakpoints - provides the ability to pause code execution at a given line of code and inspect the initialized variables.*
- *F2: Auto-completion / Code Suggestions - displaying potential code completion / suggestions based on the currently typed code.*
- *F3: Syntax Error Highlighting - underlining potentially incorrect code leading to syntax errors.*
- *F4: Brace Matching Highlighting - highlights the matching brace in code to improve the visibility of its scope.*
- *F5: Key Shortcuts - integrated basic shortcuts (e.g. copy, paste, cut, find etc.*
- *F6: Compiler / Interpreter (Run Button) - possibility to compile/run the code without executing related tests.*
- *F7: Dedicated Shell Console - access to a console where code can be run manually (e.g. using custom arguments or properties).*

Finally, in the last part of the survey, we asked two open-ended questions to collect additional features and requirements we might have missed.

- *O1: Do you have any further suggestions on how the editor could be improved?*
- *O2: Are there any other highly important features a code editor should have, in order to support you in writing code?*

3.2 Systematic Tool Review

Complementing the survey, to better understand the state-of-the-art online IDEs, we conducted a systematic tool review. For this purpose, we searched three digital libraries: IEEE Explore[3], ACM Digital Library[4] and Springer[5].

[1] https://github.com/ls1intum/Artemis
[2] https://www.limesurvey.org
[3] http://ieeexplore.ieee.org
[4] http://dl.acm.org
[5] https://link.springer.com/

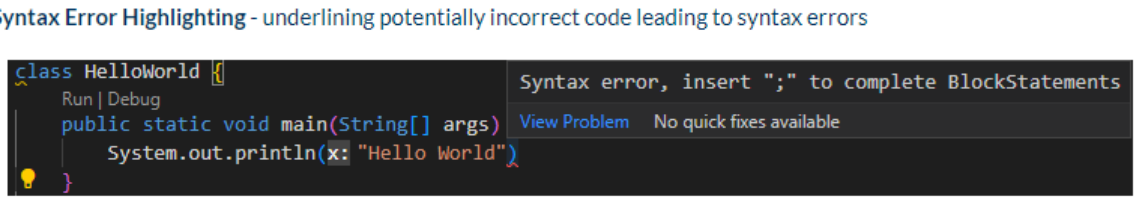

Figure 1: Example of a feature description and screenshot provided to the survey participants.

The scope covered all publications from 2013 until 2023 using the following search term: *"online code editor" OR "online IDE" OR "web-based IDE" OR "browser-based IDE" OR "cloud IDE" OR "online compiler"*.

The search term was carefully chosen to include a wide variety of IDE synonyms used in research. Before finalizing the search term, we conducted a pilot search. This initial search helped to test the adequacy and appropriateness of the chosen search string. Like this we were able to verify that the used terms were resulting in relevant literature. Additionally, this pilot search resulted in insights into the volume and nature of the available literature, which helped in defining the scope of the review, including defining which databases to search and what time frame to consider.

In total we found 39 results in IEEE Explore, 243 results in the ACM Digital Library and 122 results in the Springer database. On these results, we used the following inclusion and exclusion criteria:

Table 1: Inclusion and Exclusion criteria.

Inclusion Criteria	Exclusion Criteria
Papers Accessible in Full text	Non-english articles
Presenting a new online IDE	Not presenting an online IDE
Focusing on supporting programming	Not focusing on code
Available to test online or open source	Not available to test.

Based on the results we created a list of tools, and prepared a checklist that includes the feature set as proposed by the survey respondents. Following this, each tool was tested, and a comprehensive list of the provided features was extracted. In addition to the provided features, the supported programming languages, as well as the integrated editor on which the tool relied on, were noted down in the same document. This review was conducted to find commonly offered and practically implementable features. It also enabled a comparative evaluation of integrated editors across various platforms, providing a broad view of their strengths and weaknesses.

4 RESULTS

In the following, we present the results of the conducted survey and tool analysis analysis.

4.1 Requirements for an Online IDE

For the survey that has been displayed in the APAS Artemis we received 48 valid responses of mainly first year computer science students. In the following the results will be presented.

4.1.1 Demographics

The preliminary questions revealed that the APAS is primarily utilized by students in the initial stages of their university journey, with a significant concentration of users in their first semester. The results further show that around 60% of participants mostly use GIT combined with a local IDE when solving programming exercises. Only, approximately 20% of the respondents use mainly the online editor and the remaining 40% use both equally. The exact distribution can be seen in Figure 2. The substantial reliance on external tools, like local IDEs and GIT, suggests the need for improvement of the online code editor used in the APAS, as this would help beginner programmers get started without having to learn GIT and install an IDE first. This validates the necessity of this study, aiming to understand the requirements for online IDEs for APASs in general.

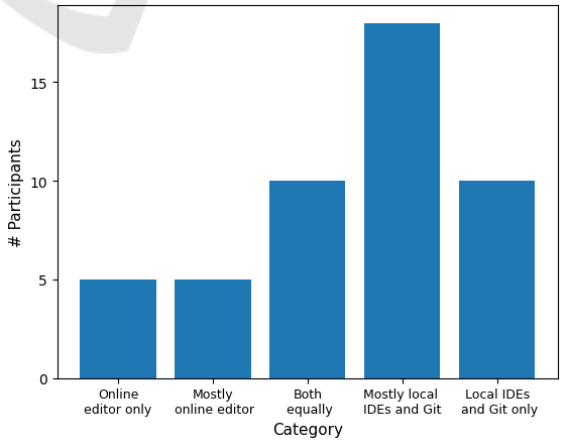

Figure 2: Preferred tools for solving programming exercises.

4.1.2 Features' Importance Questions

Following the methodology outlined in Section 3, participants were asked about the significance they place on various features offered by online IDEs.

The prioritization of IDE features according to the participants' responses is as follows:

- *Syntax Highlighting:* This feature has received the highest number of "Very Important" ratings, making it the most valued feature among the surveyed participants. Syntax highlighting is crucial because it allows developers to quickly distinguish elements of the code, such as variables and functions, which can enhance readability and reduce the likelihood of errors.
- *Syntax Error Highlighting:* Similarly to syntax highlighting, syntax error highlighting is considered very important. It provides immediate feedback on errors, without the need to compile or execute the code, which is essential for learning and improving code.
- *Compiler / Interpreter (Run Button)*: The feature is considered important, reflecting its role in the actual running and testing of the code. This indicates that a seamless transition from coding to execution is valued by users.
- *Auto-completion / Code Suggestions:* Rated as very important, this feature suggests that users value assistance in coding for speed and efficiency. It helps in reducing typos and understanding available functions and methods.
- *Debugger / Breakpoints:* This feature is highly rated, but already more on the lower end of regarding the importance compared to other features. Indicating that beginner programmers might not directly realize the importance of debugging to understanding code flow and identifying bugs.
- *Brace Matching Highlighting:* This feature is important but has a slightly lower rating compared to other features. It helps users in navigating complex code blocks, which is useful in understanding and maintaining code structure.
- *Key Shortcuts:* This feature has the most neutral ratings. It indicates that while some users find shortcuts essential for faster coding, others may not rely on them as much, which could depend on the user's familiarity with the IDE or personal coding habits.
- *Dedicated Shell Console:* Although it has received a significant number of negative ratings, it still has enough positive responses to suggest that for certain tasks or users, a dedicated shell console within the IDE is an essential feature. This might be more important for those who work with command-line interfaces or scripting.

The aggregate data suggests that while some features are universally acknowledged as critical (like syntax highlighting and error detection), others are subject to more personal preference (like key shortcuts and shell consoles). This emphasizes the need for customizable online IDEs that allow users to tailor their environment to their personal workflow and the tasks at hand. In Figure 3 we ordered the results with the most important feature being Syntax Highlighting and the least important being Dedicated Shell Console using a weighted sum.

For the open questions, students frequently highlighted the need for enhanced compilation performance in the Artemis platform. Currently, Artemis does not support direct code compilation. Instead, upon submission, the Artemis Platform runs all predefined test cases in the CI environment to validate the solutions, leading to longer wait periods for students to see whether their solution compiles or not. This concern was raised four times, with students specifically suggesting the implementation of a "Run-only" button to allow code compilation without executing tests. Additionally, there was a strong desire, mentioned five times, for increased editor pane size or the introduction of a presentation mode to improve the user interface. The improvement of the editor's intellisense, particularly for code completion and file explorer functionality, was highlighted twice. Furthermore, there were individual requests for the integration of a debugger and enhancements to the feedback mechanism provided by the APAS.

Main Findings for RQ 1

Based on the survey responses we could derive the following ranking for requirements of an online IDE: Syntax Highlighting, Syntax Error Highlighting, Compiler and Interpreter, Auto-completion and Code Suggestions, Debugger and Breakpoints, Brace Matching Highlighting, Key Shortcuts and a Dedicated Shell Console .
The answers to the open ended questions indicated a strong demand for improved compilation speed in Artemis, with students proposing a "Run-only" button to bypass test runs during compilation. Additionally, improving the user interface via larger editor panes or a presentation mode was a notable request.

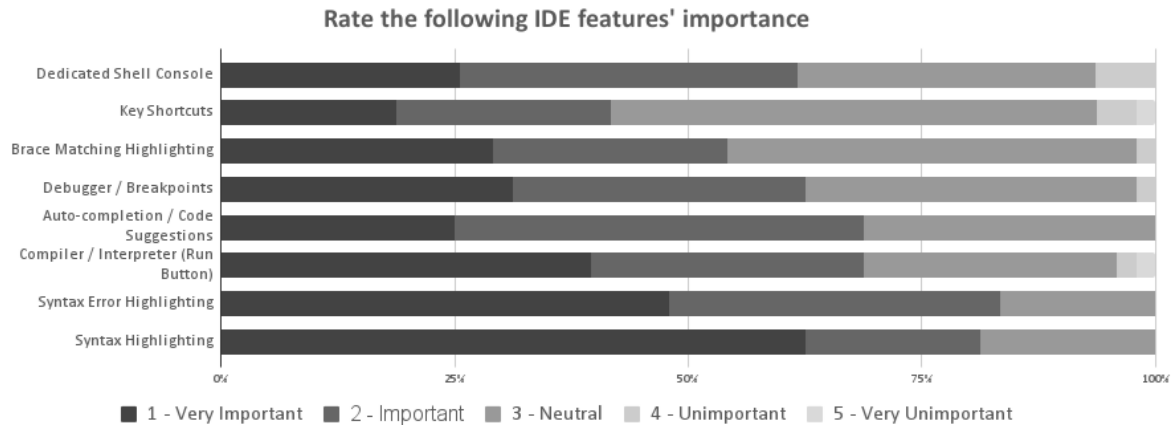

Figure 3: The detailed feelings of students about different IDE features.

4.2 State of the Art Solutions

The results of the systematic tool review are shown in Table 2, showcasing a total of 15 analyzed online IDEs, along with their features, supported programming languages, used editors and whether they are available open source or not.

From this analysis, it is evident that three prominent editors dominate as a basis for online IDEs: (1) Monaco (2) ACE and (3) CodeMirror. The Monaco editor stands out as being the basis for more than 50% of the tools. Furthermore, we found that the support for various programming languages is extensive across the board, with most tools catering to a vast majority of the selected languages. Tools utilizing the Theia framework, which builds on the 'Monaco' editor, tend to offer the most comprehensive IDE feature set compared to other editors. Notably, the features, syntax error highlighting and auto-completion are only found in tools using the Monaco editor, with the exception of 'Online-ide' that uses the 'ACE' editor and provides auto-completion for a subset of the languages.

Last but not least, Table 3 also shows that Codespaces and Gitpod, fulfill all the requirements we have defined in this study. Proving that it is possible to create a complete online IDE.

Main Findings for RQ 2

Based on the systematic tool review we found that the online IDEs Gitpod and Codespaces already fulfill all of the defined requirements. Proving that it is possible to create a functional online IDE.

4.3 Online Ides' Usability Within APASs

For an online IDE to be usable within APASs it has to be either open source or has to offer an API to which to connect to. However, connecting an APAS via an API to an external online IDE is not recommended, because it introduces complexities in maintaining consistent and secure data exchange, potential latency issues and can significantly increase dependency on the external service's uptime and reliability, which usually does not align with the APAS's operational requirements and standards. Therefore, we mainly concentrated on whether the source code of an online IDE is available as open source during this review. Based on this we found that only three tools were available as open source: (1) Artemis, (2) Coderunner and (3) Exercism. Inspecting these, we found that they offer important features like: (1) Syntax highlighting, (2) Compiler / Interpreter, (3) Brace matching highlighting, (4) Line numbers, (5) Key shortcuts, (6) File navigation tools and (8) Line numbers. However, important features are missing. Among others, the following: (1) Debugger, (2) Auto-completion / suggestions, (3) Syntax error highlighting and (4) Dedicated shell console.

Main Findings for RQ 3

There are open source online IDEs like the built-in editor of the APAS Artemis or Coderunner's and Exercism's online IDEs that can be easily integrated within APASs. However, all only provide partial feature support. In order to create an online IDE that fulfills all the requirements defined in this study it seems that improving Artemis' editor should be the recommended path to follow because the editor is already integrated into an open source automated assessment system.

Table 2: Systematic Tools Review.

Tool name	Provided features (table 3)	Supported languages (table 3)	Base Editor used	Open Source
Artemis	1,2,6,8	1,3,4,5	Monaco	Yes
Codeboard	1,2,3,7,8,(9)	1,2,3,4,5	ACE	No
Coderunner	1,3,7,8,(9)	1,2,3,4,6	ACE	Yes
Codebyte-editor	1,2,3,7,8,9	1,2,3,4,6	Monaco	No
Exercism	1,2,3,7,8,9	1,2,3,4,5,6	CodeMirror	Yes
Sololearn	1,3,7,8,9	1,2,3,4,6	Monaco	No
Replit	1,2,3,4,(5),6,7,8,9,10	1,2,3,4,5,6	Monaco	No
Codesandbox	1,2,5,7,8,9,(10)	6	Monaco	No
W3schools	1,3,(9)	1,2,3,4,6	CodeMirror	No
Codewars	1,3,7,(9)	1,2,3,4,5,6	CodeMirror	No
Online-ide	1,2,3,(5),7,8,9	1,2,3,4	ACE	No
Onlinegdb	1,(2),3,4,7,8,(9)	1,2,3,4,6	ACE	No
Codespaces	1,2,3,4,5,6,7,8,9,10	1,2,3,4,5,6	Theia (Monaco)	No
Gitpod	1,2,3,4,5,6,7,8,9,10	1,2,3,4,5,6	Theia (Monaco)	No
Codeanywhere	1,2,3,4,5,6,7,8,9,10	1,2,3,4,6	Theia (Monaco)	No
(#) means partially supported				

Table 3: Reviewed features and languages.

#	Features	#	Programming languages
1	Syntax highlighting	1	C
2	File navigation tools	2	C++
3	Compiler / Interpreter	3	Java
4	Debugger	4	Python
5	Auto-completion / suggestions	5	Haskell
6	Syntax error highlighting	6	JavaScript
7	Brace matching highlighting		
8	Line numbers		
9	Key shortcuts		
10	Dedicated shell console		

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

5 DISCUSSION

In this study we found that first-semester computer science students prioritize online IDE features as follows: (1) Syntax Highlighting, (2) Syntax Error Highlighting, (3) Compiler / Interpreter (Run Button), (4) Auto-completion / Code Suggestions, (5) Debugger / Breakpoints, (6) Brace Matching Highlighting, (7) Key Shortcuts, and (8) Dedicated Shell Console. The preference for syntax highlighting and error highlighting aligns with the expectations for beginner programmers, who often have problems with understanding the syntax and structure of code. These features provide immediate visual cues that can help beginners identify and correct errors more efficiently, thereby enhancing their learning experience and reducing frustration. The preference for a compiler or interpreter with a simple 'Run' button reflects the desire for straightforward, immediate feedback on code execution, which is particularly appealing for beginners not yet comfortable with command-line interfaces or complex build processes. The appreciation for auto-completion and code suggestions further indicates the students' need towards features that provide direct assistance and reduce the cognitive load of remembering syntax and function names. By reducing the cognitive load associated with understanding syntax and executing programs, these features can help mitigate early frustrations and enhance the overall learning experience. This is particularly relevant in the context of APASs, where the immediate feedback loop and the opportunity for iterative improvement are key. The ability of an online IDE to provide clear, instant feedback aligns with the key requirements of

APASs, facilitating a learning environment where students can rapidly adapt and refine their code based on automated assessments.

The mid-tier ranking of debugging tools might initially seem surprising, given their importance in the software development process. However, this can be understood in the context of novice programmers' limited exposure to and understanding of systematic debugging strategies. As students' programming skills evolve, they are likely to recognize the value of these tools in diagnosing and resolving issues more effectively.

Beside this, feedback from open-ended questions in the survey brought to light additional desired features, such as a presentation mode and a large editor pane, which are particularly relevant in an educational context where students often present their work. The need for enhanced compilation performance, as specifically mentioned in relation to the Artemis platform, points to a broader requirement for more efficient code execution within online IDEs. Students' suggestions for a run button, to bypass full test execution for quicker compilation feedback, illustrates a critical area for improvement in APASs.

The systematic review of online IDEs, emphasizing the robustness of editors like Monaco and the comprehensive capabilities of Theia, provides a roadmap for the development of APASs. However, the challenge of balancing comprehensive feature sets with resource efficiency remains vital, especially in educational settings where scalability is key. Additionally, the systematic tool review allows educators and alike to quickly decide on which tool to use depending on the programming language and feature support needed.

In summary, this study not only identifies the online IDE features most valued by beginner programmers but also highlights the critical role these features play in supporting the objectives of APAS by helping to create an environment that reduces initial barriers to learning programming.

6 LIMITATIONS

This study largely leaned on a survey and a systematic tool analysis. Regarding the survey, it might be the case that we overlooked important online IDE features. However, this threat to validity should be minimized by allowing the students to add missing features via open-ended feedback.

In addition, the over representation of first-year students to judge the most important features for an IDE presents a potential limitation. These students, due to their limited experience, might prioritize features that offer immediate convenience and are easy to understand over those that could provide significant long-term benefits in learning and coding proficiency. For instance, syntax highlighting, being an intuitive feature, is easily appreciated by beginners for its ability to improve code readability and reduce errors. This direct and immediate benefit might lead students to rate such features as highly important. On the other hand, the utility of a debugging tool, which is more complex and requires a deeper understanding of programming concepts to use effectively, might not be as readily apparent to them. Even-though, over time, as students' programming skills improve, the ability to systematically diagnose and fix errors using a debugger significantly enhances their productivity and learning outcomes. Future work could involve conducting a similar study with programming educators and more experienced programmers to gain a broader perspective on the importance of various IDE features. However, starting with students as the primary respondents was a deliberate choice, considering they are the main users of online IDEs in APASs and as a results understanding their preferences and immediate needs can provide valuable insights into designing more accessible and user-friendly programming environments.

Last but not least, in the systematic tool analysis, we might have missed important online IDEs. However, the search string was developed with input from two subject matter experts, aiming to encompass most significant online IDEs. Despite this, the review process could still be affected by subjective interpretations, even-though we used a systematic data extraction methodology to mitigate such biases.

7 CONCLUSION

This research aimed to identify and evaluate the essential features required for an effective online IDE in the context of APASs. Through a comprehensive survey conducted among APASs users, primarily composed of first-semester computer science students, we gained valuable insights into the features that are most valued in an online IDE. The ranking of these features, led by Syntax Highlighting and Syntax Error Highlighting, reflects the specific needs and preferences of beginners in programming. This ranking, complemented by the additional requirements identified through open-ended questions, provides a clear direction for future development and enhancement of online IDEs in educational contexts.

The systematic review of existing online IDEs underscored the prominence of the Monaco editor and revealed the comprehensive feature set offered by the Theia IDE. However, the resource-intensive nature of Theia points to a need for further investigation into its feasibility for widespread use. This study's findings also highlight a gap in the availability of open-source online IDEs with only Artemis, Coderunner and Exercism, offering partial feature support.

Future research should on the one hand expand the survey's population to also include experienced programmers and educators and on the other hand explore the resource demands of comprehensive online IDEs, like Theia, and investigate the feasibility of developing open-source solutions that fully meet the identified requirements.

In conclusion, this study contributes to the understanding of what constitutes an effective online IDE for APASs. By prioritizing features according to user preferences and exploring the current state of online IDEs, we lay the basis for future advancements in this area, ultimately aiming to improve the learning experience of programming students and the efficiency of APASs.

ACKNOWLEDGMENTS

The CodeAbility Austria project has been funded by the Austrian Federal Ministry of Education, Science and Research (BMBWF).

REFERENCES

Barra, E., Lopez-Pernas, S., Alonso, A., Sanchez-Rada, J. F., Gordillo, A., and Quemada, J. (2020). Automated assessment in programming courses: A case study during the COVID-19 era. *Sustainability*, 12(18):7451.

Chen, H.-M., Nguyen, B.-A., Yan, Y.-X., and Dow, C.-R. (2020). Analysis of learning behavior in an automated programming assessment environment: A code quality perspective. *IEEE Access*, 8:167341–167354.

Hausladen, J., Pohn, B., and Horauer, M. (2014). A cloud-based integrated development environment for embedded systems. In *Proc. of the 2014 IEEE/ASME 10th Int'l Conference on Mechatronic and Embedded Systems and Applications*, pages 1–5. IEEE.

Krusche, S. and Seitz, A. (2018). Artemis: An automatic assessment management system for interactive learning. In *Proc. of the 49th ACM Technical Symposium on Computer Science Education*, pages 284–289.

Kusumaningtyas, K., Nugroho, E. D., and Priadana, A. (2020). Online integrated development environment (ide) in supporting computer programming learning process during COVID-19 pandemic: A comparative analysis. *International Journal on Informatics for Development*, 9(2):66–71.

Mekterović, I., Brkić, L., and Horvat, M. (2023). Scaling automated programming assessment systems. *Electronics*, 12(4):942.

Satav, S. K., Satpathy, S., and Satao, K. (2011). A comparative study and critical analysis of various integrated development environments of C, C++, and Java languages for optimum development. *Universal Journal of Applied Computer Science and Technology*, 1:9–15.

Sinanaj, L., Ajdari, J., Hamiti, M., and Zenuni, X. (2022). A comparison between online compilers: A case study. In *Proc. of the 11th Mediterranean Conference on Embedded Computing*, pages 1–6.

Tran, H. T., Dang, H. H., Do, K. N., Tran, T. D., and Nguyen, V. (2013). An interactive web-based IDE towards teaching and learning in programming courses. In *Proc. of 2013 IEEE Int'l Conference on Teaching, Assessment and Learning for Engineering*, pages 439–444. IEEE.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS