Exploring the Effectiveness of Web-based Programming Environments for MOOCs: A Comparative Study of CodeOcean and OpenJupyter

Mohamed Elhayany
Hasso Plattner Institute
University of Potsdam
Potsdam, Germany
Mohamed.Elhayany@hpi.de
https://orcid.org/0000-0002-7689-7622

Sebastian Serth
Hasso Plattner Institute
University of Potsdam
Potsdam, Germany
Sebastian.Serth@hpi.de
https://orcid.org/0000-0003-1236-6600

Christoph Meinel
Hasso Plattner Institute
University of Potsdam
Potsdam, Germany
Christoph.Meinel@hpi.de
https://orcid.org/0000-0002-3410-3193

***Abstract*—Programming courses offered by openHPI, the European MOOC platform of the Hasso Plattner Institute, feature hands-on programming exercises to support learners in practicing the newly acquired skills. These exercises are facilitated by two tools: *CodeOcean* and *OpenJupyter*. *CodeOcean* is user-friendly and suitable for beginners, while *OpenJupyter* is more advanced and used in data science courses. In this paper, we compare and discuss the advantages and limitations of both tools, providing recommendations for instructors and researchers in programming courses. We also address technical details, such as scalability and execution environments. Furthermore, we explore future research possibilities, particularly in learner collaboration and automated feedback. Our work supports learners in acquiring knowledge and testing it at their own pace, with individualized feedback and minimal technical requirements, contributing to an open education landscape in programming education.**

Keywords—MOOC, Programming, Web-based Environment, Auto-Grader, Data Science, CodeOcean, OpenJupyter

I. Introduction

The widespread adoption of Massive Open Online Courses (MOOCs) has transformed how individuals acquire new skills and knowledge. Learners are no longer limited to traditional classroom settings but can access a vast array of courses online, at their own pace and convenience [1]. One of the European providers offering these courses is openHPI[1], the MOOC platform of the Hasso Plattner Institute (HPI). The platform provides learners with digital literacy skills, covering both programming fundamentals and advanced topics.

Within the openHPI platform, introductory programming courses are particularly popular. These courses enable learners to apply and test their newly acquired knowledge through embedded programming exercises. To help learners develop practical skills, these courses feature hands-on programming exercises, in addition to video-based knowledge acquisition. Depending on the course context, two tools are used to provide exercises: *CodeOcean* and *OpenJupyter*. *CodeOcean* is a user-friendly environment for executing code and receiving peer support, geared towards beginners. In contrast, *OpenJupyter* is designed for more advanced learners in data science courses. Together, these tools complement each other and offer a comprehensive learning experience for learners of different levels.

In this paper, we present a comparison of these two tools and discuss their respective advantages and limitations. Our analysis provides a recommendation for course instructors and researchers in the field of programming courses. Furthermore, we provide technical details on the scalability of the coding environments, including the number of parallel users supported and the requirements for isolated execution environments. Finally, we provide an outlook on current and future research, focusing on learner collaboration and automated feedback. By providing a comprehensive overview of these two programming exercise tools, we aim to contribute to an open education landscape in programming education. Our work supports learners in acquiring and testing knowledge at their own pace with individualized feedback and a minimal, technical setup.

II. Background and Related Work

Our work is surrounded by the existing research around openHPI as a MOOC platform and other auto-graders.

A. Background: openHPI

openHPI is an online learning platform that offers Massive Open Online Courses (MOOCs). Our MOOCs provide learners with access to high-quality educational resources, such as videos, quizzes, and interactive coding exercises. Each course is structured around a specific topic, consisting of several modules that build upon another (see Fig. 1). Usually, these modules are made available on a weekly basis. In a programming course, learners acquire points through weekly homework and programming exercises. Learners achieving at least 50% of all points are rewarded with a graded Record of Achievement. All courses on openHPI are designed to be accessible to a broad audience, regardless of prior experience or expertise in the field. These courses are taught by experienced instructors from Hasso Plattner Institute and are available to individuals worldwide, regardless of their location. To further lower the entry barrier for learners interested in learning programming, those courses feature one of our setup-free programming environments.

Fig. 1. Structure of a module consisting of multiple topics and an exam.

B. Related Work

Past studies have shown that acquiring programming skills is easier for learners when hands-on exercises are available to

[1] https://open.hpi.de

2023 IEEE Learning with MOOCS (LWMOOCS) | 979-8-3503-1559-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/LWMOOCS58322.2023.10306228

them [2], especially in contrast to multiple-choice quizzes [3]. Hence, learners need access to a programming toolchain, including a compiler or interpreter for the respective programming language. From a practical perspective, however, teaching teams cannot support learners to configure the respective tools on their local machine, as this would consume too many resources [2]. Consequently, learners benefit from a ready-to-use environment provided by MOOC instructors.

In learning contexts, two main approaches exist to provide programming environments: Client-side environments solely executing code in the learners' browser (e.g., JavaScript [4], transpiling Java [5], or even providing a full Linux VMs for C code [6]) or server-based environments, executing learners' code remotely in a sandboxed environment [7, 8, 9]. Both programming environments compared in this paper use the server-side execution mode since this allows adding support for different programming languages more easily. However, this execution mode also raises various questions on the isolation of code executions, which has been investigated by researchers in the past [7, 10, 11, 12].

III. CodeOcean

To encourage structured thinking, problem-solving, and digital literacy, many see learning programming fundamentals as an excellent opportunity. For beginner-level programming courses, we use *CodeOcean*, an educational programming environment [13]. It has been utilized in more than 50 courses and by more than 100,000 learners over the past eight years [14]. The educational programming environment allows learners to work on assignments, execute code, and observe program behavior directly through their web browser without requiring any local setup. Supported programming languages include Java, Python, Ruby, R, and Julia.

Learners' programs are executed in a secure environment on our infrastructure, and the workload is distributed across multiple servers [7]. *CodeOcean* provides tailored feedback to learners on their code, utilizing exercise-specific tests and static code analysis [15]. Based on their performance within a programming assignment, learners can earn points, reflected in their score on the HPI MOOC platform, that count towards graded certification. In addition to functional feedback, our experience suggests that learners benefit significantly from feedback on their code style, which is well-suited from the first exercise onwards.

While automated evaluation of learners' code enables them to learn at their own pace and receive instant feedback, some learners may require additional support. Previous work found that contextual hints are highly valued by most learners (and especially beginners) [16], while human support from peers can be requested by learners struggling with a given task, leading to a significant improvement in their performance [17]. Data demonstrate that offering both automated and human support reduces the barrier for learners to request help and shows that different socio-demographic groups use these help offerings to varying degrees [18].

With a clear focus on introductory programming courses, *CodeOcean* is designed with novices in mind. As a result, simplicity is the main objective, allowing learners to concentrate on applying programming concepts, rather than discovering all features usually available in a professional development environment.

IV. OpenJupyter

JupyterLab, an open-source web-based platform, has gained popularity in small classrooms as a teaching tool [19]. It offers interactivity, allowing students to experiment with code, equations, and visualizations in real time, improving the learning experience and comprehension of complex concepts [20]. With support for various programming languages, JupyterLab is versatile for teaching multiple subjects [20]. Collaboration is enabled through built-in version control and notebook sharing, facilitating teamwork on assignments and projects. Its web-based nature ensures accessibility from anywhere with an internet connection, making it convenient for MOOC students [20].

Recognizing the potential of JupyterLab as a powerful tool for data science education, a survey was conducted to identify the specific needs of instructors in various educational institutions [21]. The survey findings revealed a strong interest in utilizing JupyterLab as an infrastructure for hands-on programming exercises, but instructors expressed concerns about the complexity of server and environment setup, assignment submissions, grading, and providing timely feedback to students.

OpenJupyter [22] was developed as a solution to address these challenges. The tool was created in 2022 and successfully integrated into the HPI MOOC platform for a course on applied edge AI, attracting a large enrollment of over 2,000 learners. Leveraging the power of Docker containers, *OpenJupyter* ensures that each learner had a functionally isolated environment to work in, eliminating the need for any prior setup or configuration. This approach significantly reduced the stress on learners and allowed them to focus on solving exercises without worrying about complex technical requirements.

The integration of *OpenJupyter*, leveraging the features of JupyterLab and Docker containers, offers an advanced educational platform for teaching data science, particularly in projects involving large datasets. This integration enables instructors to customize courses, simplify assignment submissions, automate grading processes, and deliver timely feedback, resulting in an enhanced and hands-on learning experience for data science students. The seamless integration of *OpenJupyter* into the educational platform optimizes the utilization of JupyterLab's capabilities for working with big datasets, empowering instructors to effectively tackle the challenges of teaching data science and enabling students to explore and analyze large datasets with confidence.

V. Comparative Analysis of Key Features

To assess the effectiveness of *CodeOcean* and *OpenJupyter* as educational platforms, a comparative analysis of key features is presented in this section. The comparison will consider the target groups, interactive user experience, auto-grading, system architecture, and scalability of both tools. These factors are essential to evaluate the suitability of these platforms for programming education and to identify the current strengths and limitations of each tool. While we focus on identifying the differences between both tools in this paper, other publications highlight the learner-facing impact [2, 7, 14, 16, 17, 18, 21, 22].

A. Target Groups

First, the target group of each tool will be explored, highlighting the intended audience and user base for which

they are designed. This analysis will consider the specific needs and requirements of different user groups, such as novice learners or experienced advanced learners.

1) CodeOcean is primarily designed to target beginners, offering a simple and intuitive user interface that enables them to get started with a programming assignment easily [14]. It provides an easy-to-use platform for novices to learn programming and software engineering concepts. Next to being an educational development environment, *CodeOcean* also features integrated assistance features (as introduced in the next section), to specifically support the target group of beginners. CodeOcean is also suitable for instructors who want to embed their existing programming assignments and projects in a MOOC context since no specific changes need to be implemented.

2) OpenJupyter: In contrast, *OpenJupyter* is aimed at more advanced learners in the field of data science who are already familiar with programming languages and concepts. It is designed to offer a more flexible and powerful platform that enables them to customize their learning environment and explore more complex concepts. By providing a range of tools and resources, OpenJupyter enables learners to go deeper into the field of data science, exploring advanced topics in more detail. OpenJupyter is also suitable for instructors who want to design challenging programming exercises and projects that require a high level of skill and expertise.

B. Interactive User Experience

In this section, we will examine the user experience of *CodeOcean* and *OpenJupyter* as perceived by learners and describe their potential usage scenarios. This includes evaluating the ease of use, intuitiveness of the user interface, and the availability of interactive features that enable learners to actively engage with the programming environment.

1) CodeOcean: Regarding the target group of beginners, *CodeOcean* focuses on providing a clean and streamlined programming experience without overwhelming learners. For each programming exercise embedded in a MOOC, *CodeOcean* offers a dedicated, preconfigured workspace with an exercise description, an optional file tree, a main editor window, and an output area. Each exercise may consist of multiple files, for which changes performed by learners are automatically persisted and restored on subsequent page loads. At any time, learners can execute their code, receive feedback through an automated evaluation, or request help through one of the various assistance features included [18]. Among them are step-by-step feedback messages (based on unit tests and a linter), contextual tips, or an embedded peer support mechanism, allowing learners to request comments in case they got stuck.

Regardless of the assistance features, which can be configured by the teacher to meet different requirements (i.e., some might be disabled in an examination), *CodeOcean* supports many diverse use cases and teaching scenarios. Besides small programs just featuring in- and output through the command line, learners can interact with Turtle graphics (and, for example, create small 2D games) [14] or generate graphs, which are directly shown and downloadable.

2) OpenJupyter is a powerful and flexible tool for data analysis and programming education. One of its key features is its built-in debugging extension, which allows learners to identify and resolve errors in their code. This feature is particularly useful for learners who are new to programming and may not yet have a strong grasp of debugging techniques. By providing a user-friendly and accessible way to debug code, OpenJupyter helps learners to build confidence in their programming skills and develop a more intuitive understanding of the underlying logic of their code.

In addition to its debugging extension, *OpenJupyter* also offers powerful visualization tools for the generation of graphs and figures. These tools make it easy for learners to explore and analyze complex data sets, and to present their findings in a clear and visually compelling way. Whether working with numerical data, text data, or multimedia content, *OpenJupyter* provides a wide range of visualization options to suit a variety of use cases.

Finally, *OpenJupyter*'s interactive Notebook style is another key feature that sets it apart from other programming environments. This style allows learners to work through programming exercises and assignments in a highly interactive and engaging way, with immediate feedback on their progress and the ability to explore and experiment with different code snippets and data sets. This makes *OpenJupyter* an ideal tool for both individual learners and collaborative learning environments, where learners can work together to explore and analyze complex problems and datasets in real time.

C. Auto-Grading

Both programming environments feature auto-grading capabilities and can transmit the scoring result to the MOOC platform. This section covers the tools' ability to automatically evaluate and grade programming assignments, providing timely feedback to learners, and facilitating the assessment process for instructors.

1) CodeOcean: For each exercise, instructors can specify multiple, weighted test cases, which are reflected in the final score. Besides structural and functional tests, *CodeOcean* has dedicated support for style tests, supported by a linter in the respective language. Feedback from all tests is directly shown to learners through the web interface, and the most relevant output of the test is further extracted to pinpoint learners to the actual error [14]. This processing is designed to remove or rephrase aspects of the raw output confusing to novices, such as unrelated stack traces.

While the final score a learner achieves in each exercise is sent back to the MOOC platform complementing the learner dashboard, instructors also have access to more fine granular learning analytics. For example, teachers may use exercise-specific statistics to identify rather difficult assignments or uncover misunderstandings. They can also read optional feedback provided by the learners so that unclear parts of the exercise can be improved. Additionally, for a full course with dozens of exercises *CodeOcean*

automatically performs an anomaly detection, identifying those exercises where learners spent the most time or have the most problems.

2) OpenJupyter: We have streamlined the grading process in our MOOC platform, openHPI, by integrating a JupyterLab-managed service in OpenJupyter that updates learners' grades efficiently through the Learning Tools Interoperability (LTI) protocol. To enable this feature, we have included a unit test file inside the learners' Docker container, hidden from view. When a learner runs the grading cell in the notebook, the hidden unit test file runs and calculates a grade based on the learners' answers. Currently, the grading process provides a binary outcome indicating whether the task was correct or not.

The Jupyter Notebook includes code that interacts with the JupyterLab-managed service to send the grade back to the Learning Management System (LMS). This is done by calling an Application Programming Interface (API) provided by the service and passing in the necessary information, such as the learner's grade and assignment details. When a learner submits an assignment, the JupyterLab-managed service collects the necessary information and sends it to the LMS through the LTI protocol. The LMS then uses this information to update the learner's grade in the course. The service also ensures secure communication by handling the authentication of the learner and the JupyterLab instance with the LMS. In summary, this approach provides a simple, efficient, and secure process for updating learners' grades in the MOOC platform and allows instructors to use *OpenJupyter* with a variety of different LMS.

D. System Architecture

From a technical perspective, *CodeOcean* and *OpenJupyter* use similar technologies to provide learners with an execution environment. The specific details including the underlying infrastructure, and deployment options are introduced in this section.

1) CodeOcean is realized as a micro-service architecture [7], allowing easier scalability for high-demand periods. Overall, the system consists of a web application written with Ruby on Rails, a dedicated PostgreSQL database, and a Nomad cluster [2] orchestrating the actual executions. Within the Nomad cluster, several so-called agents act as a host for Docker containers in conjunction with the gVisor runtime [3]. This setup allows the agents to execute learners' code in a sandboxed environment, isolating potentially malicious code from the host and other learners.

For the communication between the Nomad cluster and the Ruby on Rails web application, we use a custom executor middleware called *Poseidon*. It simplifies the management of code executions, for example by maintaining a pool of already running but idling containers to be ready for a learner or by configuring network access of the containers. Further, the executor middleware enforces resource limitations, such as limiting the allowed execution time to reduce the impact of infinite loops. Finally, a monitoring instance visualizes technical metrics, allowing system administrators to inspect the overall system's health. With all those components, we currently allocated 240 GB RAM and 104 vCPU cores to *CodeOcean* and all its components.

2) OpenJupyter's system architecture is designed to provide learners with a comprehensive and practical learning environment. One of its key features is the ability for learners to access multiple files, such as coding scripts, within a single notebook. This feature allows learners to organize their work and build on previously written code, enabling them to take a more structured and iterative approach to their learning. By having access to multiple files, learners can work on larger projects, as well as experiment with different programming approaches and techniques.

To ensure functional isolation and reproducibility, *OpenJupyter* employs Docker as part of its system architecture [22]. Each learner's environment is isolated in a separate Docker container, which provides a consistent and secure environment for learners to work in. This allows learners to experiment with different software configurations and programming languages without worrying about affecting other learners or the system's stability. Additionally, the use of Docker enables *OpenJupyter* to scale easily and support a large number of learners simultaneously.

To fetch practical exercise notebooks, *OpenJupyter* uses Nbgitpuller[4], a JupyterLab extension that simplifies pulling Jupyter Notebooks from a Git repository into a JupyterLab environment. Nbgitpuller offers version control, enabling easy tracking of changes and rollbacks. The extension also ensures automatic updates of Jupyter Notebooks, providing learners with the most recent version of the material. This approach guarantees learners are working with up-to-date materials, making the learning experience more efficient and effective.

E. Scalability

Lastly, we evaluate the scalability of both tools. For this aspect, we consider their ability to handle large-scale courses and accommodate a growing number of users without compromising performance.

1) CodeOcean: Thanks to the micro-service architecture, *CodeOcean* can scale easily across multiple instances. Especially for the code executions, which already require the most resources, adding additional Nomad agents to the existing cluster can be performed at any time. With the current setup, we successfully served a MOOC with more than 40,000 enrollments and about 17,000 active learners. During the respective course lasting six weeks, the platform executed learners' code more than 4.7 million times. While *CodeOcean* copes with an increasing number of learners, we noticed up to 17% slower execution times for those high-demand periods. Still, with these real-world performance metrics obtained from public courses, we consider *CodeOcean* to be stable and scalable for most scenarios.

[2] https://www.nomadproject.io
[3] https://gvisor.dev
[4] https://jupyterhub.github.io/nbgitpuller/

2) OpenJupyter: The ability of *OpenJupyter* to handle a growing number of learners is a crucial consideration when implementing it as a supportive tool for programming education. Currently, *OpenJupyter*'s scalability is determined by the server's size that hosts it, which is estimated based on the average RAM usage per learner. Our analysis reveals that each learner typically requires around 160 MB of RAM, and our current virtual machine has a capacity of 32 GB of RAM. Consequently, the system is presently capable of supporting approximately 250 learners concurrently working on a specific exercise. However, it is important to note that this estimation may vary depending on factors such as exercise complexity and the resource demands of each learner's programming environment. Therefore, it is imperative to regularly monitor and assess *OpenJupyter*'s performance to ensure its capacity aligns with the needs of an expanding learner base.

The scalability of *OpenJupyter* has been effectively demonstrated in a real-world scenario, as it has been tested in a MOOC with over 2,000 enrolled learners. During this testing phase, *OpenJupyter* seamlessly accommodated all learners, allowing them to work on their exercises without any issues. This successful implementation further affirms *OpenJupyter*'s capability to handle a large learner base and indicates its robustness in scaling up to meet the demands of a substantial number of concurrent users.

TABLE I. FEATURE COMPARISON: CODEOCEAN VS OPENJUPYTER

	CodeOcean	**OpenJupyter**
Target Group	Beginners	Intermediate and advanced learners
Interactive User Experience	- Exercise-focus - Integrated assistance features (Request for comments, tips, step-by-step feedback) - Interactive turtle graphics - Figure visualizations	- In-depth exercise structure - Code debugging - Interactive notebook style - Resource usage information - Figure visualization
Auto-Grading	- Unit Tests - Linter integrated	- Unit tests
System Architecture	- Micro-services - Nomad Cluster - Docker Containers	- Git integration - Docker container
Scalability	- Used in a MOOC with more than 40,000 learners, about 17,000 were actively using *CodeOcean* - "Scale Out" among multiple hosts	- Used in a MOOC with around 2,000 learners - Currently implements a "Scale Up" strategy to host more users

VI. DISCUSSION

Both *CodeOcean* and *OpenJupyter* are web-based platforms that offer access to programming exercises and execution environments for students and educators alike. The comparative analysis of *CodeOcean* and *OpenJupyter* revealed several distinguishing features and characteristics of each tool, as highlighted in TABLE I. These aspects play a crucial role in determining their suitability for different target groups, the user experience they offer, their auto-grading capabilities, system architecture, and scalability.

In terms of the target group, *CodeOcean* primarily caters to beginners, providing a user-friendly environment for introductory programming courses. On the other hand, *OpenJupyter* is designed to accommodate intermediate and advanced learners, offering an in-depth exercise structure and interactive notebook style that aligns with the needs of more experienced users.

The interactive user experience sets both tools apart. *CodeOcean* emphasizes an exercise-focused approach, offering integrated assistance features such as requests for comments, tips, and step-by-step feedback. Additionally, *CodeOcean* provides interactive turtle graphics and figure visualizations, enabling learners to engage with the programming concepts visually. In contrast, *OpenJupyter* offers an interactive notebook style potentially listing multiple (sub-)exercises, facilitating a more flexible and exploratory coding experience. It also provides code debugging capabilities and resource usage information, allowing learners to analyze and optimize their code effectively.

Both *CodeOcean* and *OpenJupyter* employ unit tests for auto-grading programming assignments, and *CodeOcean* enhances this with a linter for coding best practices. *CodeOcean*'s architecture is based on a Nomad cluster and Docker containers, offering scalability by scaling out among multiple hosts. In contrast, *OpenJupyter* integrates Git and utilizes Docker containers, scaling up to host more users and expanding the capacity of a single host.

Scalability has been demonstrated through the successful adoption of both tools in MOOC environments. *CodeOcean* has been utilized in a MOOC with over 40,000 learners, with approximately 17,000 actively using the platform. *OpenJupyter*, on the other hand, has been employed in a MOOC with around 2,000 learners, highlighting its ability to handle a substantial user base effectively.

VII. FUTURE WORK

While *CodeOcean* and *OpenJupyter* offer a range of features to enhance the learning experience, there is still room for improvement. In particular, both platforms can be expanded to introduce collaborative work features. This would enable learners to work together on group projects, share code and insights, and collaborate in real time. We believe that this feature would further enhance the learning experience for our users.

Specifically, with *CodeOcean*, we further want to improve our learner support by leveraging artificial intelligence for the request for comments, currently handled by peers. By reducing the response time and providing individualized responses, we aim to reduce the course drop-out of learners struggling with the exercises. Also, we want to improve the scalability and therefore suggest evaluating serverless functions, as an alternative to Docker containers.

For *OpenJupyter*, another area of future work is the development of a better feedback mechanism. The current system provides binary feedback indicating whether a task was completed correctly or not. However, a more detailed feedback mechanism can be developed to provide learners with feedback on their coding style, efficiency, and design choices. This will encourage learners to write cleaner and more efficient code, which will help them develop better coding practices and improve their overall coding skills. A better feedback mechanism can also improve learner engagement and motivation, as learners will receive more personalized and constructive feedback.

Finally, since both programming environments require dedicated unit tests written by course instructors for their feedback mechanism, we also propose to establish an exchange platform for auto-gradable programming exercises.

VIII. Conclusion

In summary, we contributed to programming education through Massive Open Online Courses (MOOCs) and open education with the development of *CodeOcean* and *OpenJupyter*. These auto-grading tools offer web-based programming environments, eliminating the need for local setup. *CodeOcean* excels in introductory programming courses, providing a user-friendly interface and emphasizing simplicity, fostering effective application of programming concepts, and promoting learner engagement. *OpenJupyter* addresses challenges related to big datasets, integrating with JupyterLab and utilizing Docker containers to create an interactive, streamlined, and hands-on learning experience for advanced learners. Learners can explore and analyze large datasets without the hassle of setup complexity.

Our contributions to MOOCs and open education have paved the way for instructors to customize their courses, simplify assignment submissions and grading, and provide timely feedback to students. With the detailed analysis of *CodeOcean* and *OpenJupyter* in this paper, we allow instructors to choose the auto-grading tool that is the most appropriate for their teaching needs by introducing several different showcases.

Moving forward, it is crucial to continue refining and expanding the capabilities of *CodeOcean* and *OpenJupyter*, addressing emerging challenges in the realm of online education. This includes keeping pace with technological advancements, incorporating cutting-edge tools and techniques, and embracing the evolving needs of educators and learners. By doing so, we can continue to make remarkable contributions to MOOCs and open education, fostering a dynamic and enriching learning experience for individuals across the globe.

References

[1] J. Daniel, "Making Sense of MOOCs: Musings in a Maze of Myth, Paradox and Possibility," in *Journal of Interactive Media in Education*, no. 18, 2012, DOI: 10.5334/2012-18.

[2] T. Staubitz, H. Klement, J. Renz, R. Teusner, and C. Meinel, "Towards practical programming exercises and automated assessment in massive open online courses," in *2015 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, Zhuhai, China, 2015, DOI: 10.1109/TALE.2015.7386010.

[3] P. Blayney and M. Freeman, "Automated formative feedback and summative assessment using individualised spreadsheet assignments," in *Australasian Journal of Educational Technology*, no. 2, 2004, DOI: 10.14742/ajet.1360.

[4] V. Karavirta and P. Ihantola, "Serverless automatic assessment of javascript exercises," in *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education*, Bilkent Ankara Turkey, 2010, DOI: 10.1145/1822090.1822179.

[5] M. Pabst, "[online-IDE] LernJ vs. Java: Unterschiede." (2023): https://www.learnj.de/doku.php?id=unterschiede_zu_java:start.

[6] R. Sharrock, L. Angrave, and E. Hamonic, "WebLinux: A scalable in-browser and client-side Linux and IDE," in *Proceedings of the Fifth Annual ACM Conference on Learning at Scale*, London United Kingdom, 2018, DOI: 10.1145/3231644.3231703.

[7] S. Serth, D. Köhler, L. Marschke, F. Auringer, K. Hanff, J.-E. Hellenberg, T. Kantusch, M. Paß & C. Meinel, "Improving the Scalability and Security of Execution Environments for Auto-Graders in the Context of MOOCs," in *Proceedings of the Fifth Workshop "Automatische Bewertung von Programmieraufgaben" (ABP 2021)*, virtual event, October 28–29, 2021. DOI: 10.18420/abp2021-1.

[8] J. Breitner, M. Hecker, and G. Snelting, "Der Grader Praktomat," in *Automatisierte Bewertung in der Programmierausbildung*, Münster, Germany, 2016, https://elan-ev.de/dateien/band6-openaccess.pdf.

[9] H. T. Tran, H. H. Dang, K. N. Do, T. D. Tran, and Vu Nguyen, "An interactive web-based IDE towards teaching and learning in programming courses," in *Proceedings of 2013 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*, Bali, Indonesia, 2013, DOI: 10.1109/TALE.2013.6654478.

[10] S. Strickroth, "Security considerations for java graders," presented at the Workshop "Automatische Bewertung von Programmieraufgaben" (ABP 2019), Essen, Germany, 2019, DOI: 10.18420/ABP2019-5.

[11] R. Garmann, "Sicherheitsimplikationen beim Einsatz von Test Doubles zur automatisierten Bewertung studentischer Java-Programme mit Graja und mockito," in *Workshop "Automatische Bewertung von Programmieraufgaben" (ABP 2013)*. volume 1067 of CEUR workshop proceedings, CEUR-WS.org, Hannover, Germany, p. 6, October 2013, http://ceur-ws.org/Vol-1067/abp2013_submission_1.pdf.

[12] O. Flauzac, F. Mauhourat, and F. Nolot, "A review of native container security for running applications," in *The 17th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2020)*, Leuven, Belgium, 2020, DOI: 10.1016/j.procs.2020.07.025.

[13] T. Staubitz, H. Klement, R. Teusner, J. Renz and C. Meinel, "CodeOcean – A versatile platform for practical programming excercises in online environments," in *2016 IEEE Global Engineering Education Conference (EDUCON)*, pp. 314–323, Abu Dhabi, United Arab Emirates, 2016, DOI: 10.1109/EDUCON.2016.7474573.

[14] S. Serth, T. Staubitz, R. Teusner, and C. Meinel, "CodeOcean and CodeHarbor: Auto-Grader and Code Repository," in *Seventh SPLICE Workshop at SIGCSE 2021 "CS Education Infrastructure for All III: From Ideas to Practice (SPLICE'21)*, March, 2021, Virtual Event, https://cssplice.github.io/SIGCSE21/proc/SPLICE2021_SIGCSE_paper_13.pdf.

[15] S. Serth, "Integrating Professional Tools in Programming Education with MOOCs," in *2019 IEEE Frontiers in Education Conference (FIE)*, Covington, KY, USA, 2019, DOI: 10.1109/FIE43999.2019.9028643.

[16] S. Serth, R. Teusner, and C. Meinel, "Impact of Contextual Tips for Auto-Gradable Programming Exercises in MOOCs," in *Proceedings of the Eighth ACM Conference on Learning @ Scale (L@S '21)*, Association for Computing Machinery, pp. 307–310, New York, NY, USA, 2021, DOI: 10.1145/3430895.3460166.

[17] R. Teusner, T. Hille, and T. Staubitz, "Effects of automated interventions in programming assignments: evidence from a field experiment," in *Proceedings of the Fifth Annual ACM Conference on Learning at Scale (L@S '18)*, Association for Computing Machinery, pp. 1–10, New York, NY, USA, 2018, DOI: 10.1145/3231644.3231650.

[18] S. Serth, R. Teusner, C. Hagedorn, and C. Meinel, "Demographic Influences on Help-Seeking Behavior in Programming Courses," Manuscript in preparation, 2023.

[19] T. Kluyver, B. Ragan-Kelley, F. Perez, B. E. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. B. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla and C. Willing, "Jupyter Notebooks – a publishing format for reproducible computational workflows", in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pp. 87–90, 2016, DOI: 10.3233/978-1-61499-649-1-87.

[20] E. V. Dusen, "Jupyter for Teaching Data Science," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20)*, Association for Computing Machinery, p. 1399, New York, NY, USA, 2020, DOI: 10.1145/3328778.3372538.

[21] M. Elhayany, R-R. Nair, T. Staubitz, and C. Meinel, "A Study about Future Prospects of JupyterHub in MOOCs," in *Proceedings of the Ninth ACM Conference on Learning @ Scale (L@S '22)*, Association for Computing Machinery, pp. 275–279, New York, NY, USA, 2022, DOI: 10.1145/3491140.3529537.

[22] M. Elhayany and C. Meinel, "Towards Automated Code Assessment with OpenJupyter in MOOCs," in *Tenth ACM Conference on Learning @ Scale*, Copenhagen, Denmark, 2023, DOI: 10.1145/3573051.3596180.