A Study about Future Prospects of JupyterHub in MOOCs

Mohamed Elhayany
Hasso Plattner Institute
Potsdam, Germany
Mohamed.Elhayany@hpi.de

Ranjiraj-Rajendran Nair
Otto von Guericke University
Magdeburg, Germany
Ranjiraj-Rajendran.Nair@guest.hpi.de

Thomas Staubitz
Hasso Plattner Institute
Potsdam, Germany
Thomas.Staubitz@hpi.de

Christoph Meinel
Hasso Plattner Institute
Potsdam, Germany
Christoph.Meinel@hpi.de

ABSTRACT

The Hasso Plattner Institute (HPI) has been successfully delivering courses on several MOOC (Massive Open Online Course) platforms for the last 10 years, offering courses on various topics in the context of Artificial Intelligence (AI), Machine Learning (ML), and Data Science. In recent years, Jupyter Notebooks have become one of the most widely used tools for data science applications, a platform for learning and practicing various programming languages. We want to integrate JupyterHub into our learning platform in order to provide students with hands-on experience in AI. We have conducted a survey with a series of research questions in order to understand the needs of instructors in their courses at different institutions. In this paper, we present a detailed analysis of our survey results and we discuss our future approach to using JupyterHub as an infrastructure to solve hands-on programming exercises on our platform. We propose the idea of creating a tool to automate server and environment creation for students to work on. This tool would give instructors a platform to operate from and allow them to customize their courses. Moreover, it would help them automate assignment submissions, grading, and provide feedback to their students.

CCS CONCEPTS

• **General and reference** → *Surveys and overviews*; • **Applied computing** → *E-learning*; *Collaborative learning*; *Distance learning*.

KEYWORDS

MOOC; JupyterHub; Programming; Automated grading, Scheduling

ACM Reference Format:
Mohamed Elhayany, Ranjiraj-Rajendran Nair, Thomas Staubitz, and Christoph Meinel. 2022. A Study about Future Prospects of JupyterHub in MOOCs. In *Proceedings of the Ninth ACM Conference on Learning @ Scale (L@S ’22), June 1–3, 2022, New York City, NY, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3491140.3529537

1 INTRODUCTION

The past few years have witnessed significant growth in the number of users attending online courses [8]. Moreover, MOOCs have become a phenomenon, offering the opportunity of free high class education to everyone [10]. They bear a tremendous potential for teaching programming to a large and diverse audience [10]. Currently, there is an enormous amount of interest in ML and AI and what these new technologies can create for the present and future [2]. To truly learn how to explore the world of AI, it is essential to use the right set of learning tools and, most importantly, resources. Platforms like Jupyterhub and Jupyter notebooks have exploded in popularity since late 2014 [12], becoming very useful learning tools to program and solve assignments in a well structured, supportive environment.

Having an auto-grading tool can support instructors in providing a formative and quick feedback to their students. We have already gained experience in the field of auto-graded programming exercises with *CodeOcean* [11]. However, we now want to explore the usage of an auto-grading tool integrated with JupyterHub to offer a more interactive, user-friendly learning experience. As a starting point, this survey was aimed at better understanding the needs of the teachers on our platform. The respondents included teachers from KI-Campus, working professionals, and subject experts in AI and ML, sharing their requirements and expectations about Jupyter Notebooks.

Our goal is to set up and design a tool that allows students to work on hands-on programming exercises. Therefore, all questions that we created only targeted this particular task, and wherever we mention exercises, it refers to programming exercises only. Some of our questions also focus on the actions the students have to take on the server-side. We want to understand the usage of computing resources like GPU and CPU to create appropriate time slots for students to work on various exercises.

2 RELATED WORK

Jupyter Notebook is the most widely-used system for interactive literate programming [9]. With its user-friendly design, it makes data analysis easier to document and reproduce. Despite the fact that Jupyter notebooks were intended as a tool to be used in scientific workflows for data analysis, they are quickly becoming a common choice for university courses. In his survey of 2016, Hamrick [4] reported that across multiple countries, over 100 courses use Jupyter. Many university classes often use Jupyter notebooks as the preferred medium for homework, labs, projects, and lectures [5]. Furthermore, Jupyter Notebooks have been famous for programming exercises and running small student tests for various Data Science courses [1, 7]. For example, UC Berkeley's flagship data science courses serve thousands of students every year and use Jupyter for all of their course components.

When the number of users and memory requirements are low, it is easy to setup JupyterHub on a single server. However, setup becomes more complicated when we need to serve Jupyter Notebooks at scale to tens or hundreds of users [13]. In MOOCs, where the number of learners in a given course can reach hundred of thousands [3], setting up JupyterHub on a single server is basically impossible. Zonca and Sinkovits [13] introduced three scheduling strategies to deploy JupyterHub on a large scale for science gateways and workshops. This highlights the importance of having the adequate scheduling strategy to successfully deploy JupyterHub on a scale as large as MOOCs.

Without a tool for automated assessment of programming assignments, the teaching teams would be restricted to offer optional ungraded exercises only. [10]. Manzoor et. al. [6], performed a survey to see the effectiveness of using an auto-grading tool for grading student submissions in Jupyter Notebooks. The main goal of their work was to provide instantaneous feedback to students on their assignments as they progress through their course, which is a similar requirement for our work. They further mentioned sending scores back to a Learning Management System (LMS) using Learning Tools Interoperability (LTI) protocol standard.

HPI already developed a tool, called *CodeOcean*, that allows automatic grading of assignments in our MOOCs. Programming assignments in *CodeOcean* may contain unit tests and style checkers to provide instant feedback to learners [11]. However, *CodeOcean* does not support grading assignments in the form of notebooks. We aim to integrate a similar auto-grading tool with our JupyterHub platform. This would hugely support the teachers during their course and moreover, provide students with summative as well as formative feedback for their submitted assignments.

3 RESEARCH QUESTIONS AND EXPECTED BENEFITS

***RQ1**: What will be the expected number of learners to participate in the courses using JupyterHub provided on our platform?*

Expected Benefits: By answering this question, we would have the insight needed to deploy our JupyterHub platform and support multi-user access on a large scale.

***RQ2**: What are the desired JupyterHub technical settings and requirements from the instructors on our platform?*

Expected Benefits: We specifically elaborated questions to understand the preferred programming language and most common type of programming exercises. In addition, we focused on questions related to auto-grading criteria, feedback after exercises, exercises deadlines and exercise re-attempts. These questions help us understand how to set up JupyterHub and tailor it to the needs of our instructors.

***RQ3**: How will a possible scheduling system support programming exercises on Jupyter notebooks?*

Expected Benefits: From the answers to technical requirements in our survey, we get an estimate of the computational resources needed to build a scheduler. Furthermore, we can create time slots tailored to each programming exercise.

4 STUDY DESIGN

4.1 The Survey: Questions and Purpose

The survey was designed in a way that would give us the insight we need to answer our research questions. Moreover, it provided us with a way to communicate with the instructors on our HPI platform to understand their expectations regarding JupyterHub. The analysis was conducted in 2021 among the teaching teams who are offering courses with programming exercises on our platform. It consisted of 29 questions and all the information was collected using an online survey tool. The survey was composed of multiple choice questions with the option to leave an explanatory comment. It involved 28 individual respondents, however not all of them answered every question. Exact counts will be provided in the upcoming evaluation section.

The survey targeted 3 main topics. *Course characteristics*, which gave us information regarding the course structure, number of expected students to work simultaneously and the type of exercises that will be offered in the course. *Technical requirements*, which focused on desired programming languages, hardware and server requirements and processing speed. And finally, *Assignment submission process*, which gave us an insight into the preferences of our instructors concerning assignment submissions, deadlines, reattempts, grading, and feedback criteria.

4.2 The Survey: Evaluation of Results

The first set of questions we asked focused on understanding the general course characteristics, such as number of participants, type of exercises and environment. When asked where would the students work on the programming exercises within the course, most instructors prefer their students to work and run their code on a dedicated server, rather than on their local machines. Concerning the average number of participants, the majority (8) of responders expect a total of 100-10000 students during the time span of their course. As shown in Figure 1, this implicates that between 11 and 100 students are expected to work on an exercise simultaneously.

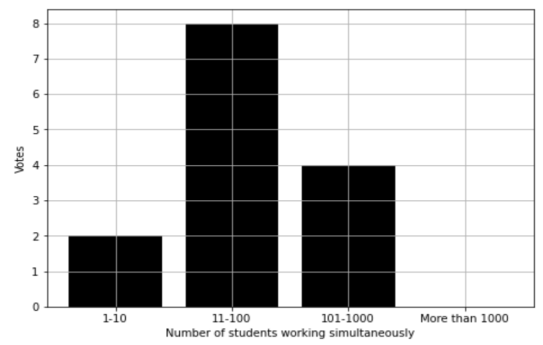

Figure 1: Votes of our Instructors on the number of expected learners working simultaneously on an exercise

The number of students working on an assignment simultaneously also depends on the type and complexity of the exercise. Therefore, we asked the instructors which type of practical assignments they intend to use in their class. Varying in complexity and time, we considered four different options:

- *Finger Exercises*: are usually very quick exercises to practice a concept that can be solved with a few lines of code.
- *Module-focused exercises*: are more complex and time-consuming exercises that cover a wider range of content. Usually they are divided across the modules of the course.
- *Mini projects*: are end-to-end projects that usually cover more than one teaching module and require more effort and time.
- *Capstone projects*: are culminating assignments, on which students usually work on at the end of the course. They require further research from the learner for a successful completion.

As shown in Figure 2, most of the instructors favored finger and module-focused exercises. Whereas, mini-projects and capstone projects were respectively occasionally and rarely chosen. Based on these findings, we expect instructors to generally design a higher number of short assignments and only a few more time-consuming projects.

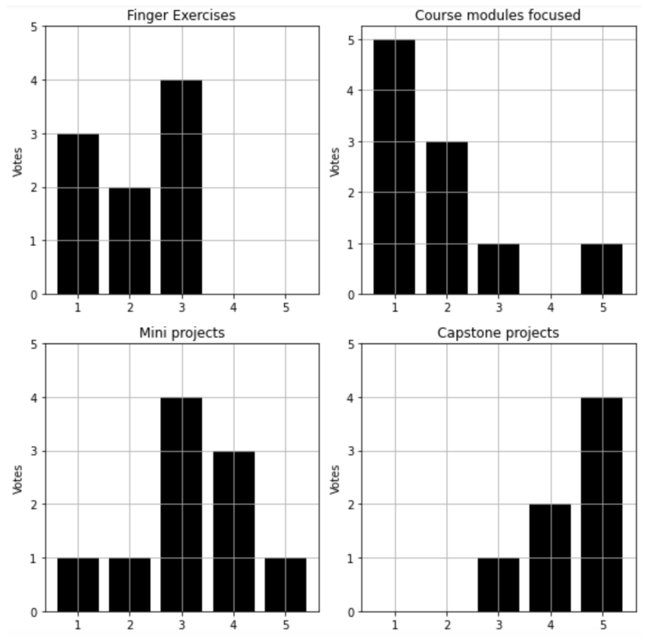

Figure 2: Different types of programming exercises that can be offered in a given course. 1 - Very Frequently, 2 - Frequently, 3 - Occasionally, 4 - Rarely, 5 - Very Rarely

Another section of the survey was related to technical requirements such as programming languages to be installed, storage space on the server and processing speed. Concerning the programming language, Python was chosen by most (9) to be frequently used, whereas R was the second most selected. The size of datasets utilized for solving exercises was indicated as generally smaller than 8 GB, as the students will mainly work with toy datasets such as MNIST and ImageNet. Especially for AI courses, instructors mentioned the need for GPUs to allow fast processing of the students' solutions.

An additional aspect that we targeted is the possibility to configure a time limit for the students in each exercise. In this case we received contrasting answers, as 4 respondents agreed on limiting the time students will spend on an exercise, whereas 5 others would rather avoid time pressure. Another suggestion we received was to close the session after an inactive period of time. Given the heterogeneous responses, we have to consider implementing a time-limiting feature in a configurable way depending on the instructor.

The next section of the survey was focused on the process of assignment submission, specifically on grading criteria and feedback. Concerning the grading, as shown in Figure 3, most of the respondents identified "correctness of the outcome", i.e. dynamic code analysis, as main criteria. Other instructors would however prefer to use static code analysis for their exercises. Having a comprehensive type of code analysis should be considered in order to satisfy both needs.

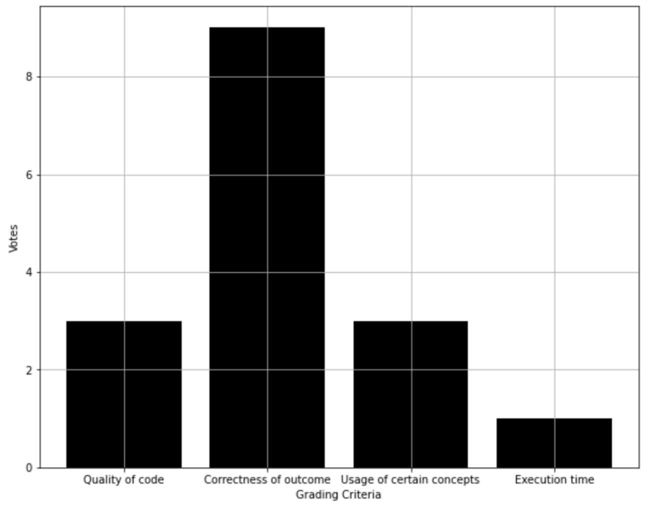

Figure 3: Possible grading criteria for assignment submissions

As shown in Figure 4, it was relevant for most of the instructors to give almost immediate feedback to their students, once they submit their assignments. This implicates the possibility of developing a scheduling strategy to optimize feedback delivery.

Finally we asked instructors if they would need to set deadlines for submitting specific exercises. Most of the respondents (9) would not like to have deadlines for exercises, allowing the students to access the assignments and earn credits as long as the course is available. Only two respondents wanted to have weekly deadlines for each exercise. As a supplementary open-ended-question we further asked what kind of action should be taken after a specific deadline has passed. Two instructors answered that students can

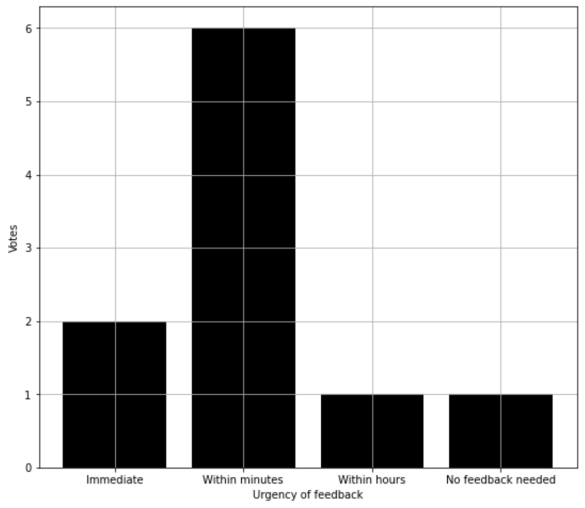

Figure 4: Time taken to provide feedback as desired by our instructors

continue to work on their exercises and submit, however they cannot earn any further credits. Two respondents further clarified that they would allow late submissions but students would be penalized with a deduction in credits. The question was then asked about the allowed number of re-attempts in exercises and its frequency. Majority of the respondents (4) required learners to have unlimited number of attempts, which can be visualized in Figure 5. It can be observed that most instructors would allow at least two attempts. This further highlights the importance of having a scheduling strategy to optimize resource allocation and give a fair chance for all students to work on their assignments.

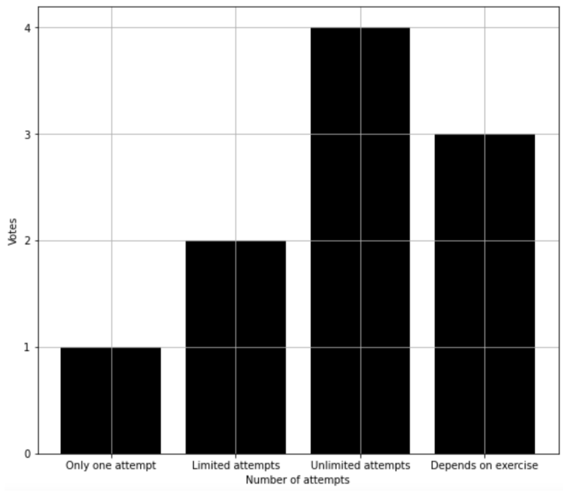

Figure 5: Number of attempts allowed for students to submit their assignments

4.3 The Survey: Discussion

The answers received from the survey participants gave us an idea about how we want to shape the JupyterHub infrastructure. They showed that having a dedicated server would be an advantage, as the majority of respondents highlighted that they would require a server for students to work on and instructors to operate from. As the resources of that server would be limited, the need for a scheduler to organize access to the server and allow proper distribution of resources to students is highly important. In fact, many students are expected to be accessing the server and working on the exercises simultaneously. The technical requirement section gave us additional insight about the preferred programming language, Python, and the type of exercises used in the courses. This will help us estimate an average time for students to work on a given assignment. Notably, the last few questions related to submission, grading criteria and feedback, received heterogeneous answers, as every instructor would want to shape the structure of the course differently. Hence, creating a tool by which instructors can customize submission, deadlines and feedback methods according to their needs would be very helpful. This tool would also automate the creation of the environment that students need in a given course and provide formative feedback after assignment submissions.

5 FUTURE WORK

The motivation of the survey described in this paper was to understand the requirements for the use of Jupyter Notebook in our online learning platform according to different experts in the field of AI and ML. Despite the number of responses that we received, we were able to gather valuable insight and have our research questions answered. The most insightful part of our survey was the expressed interest in having almost instant feedback. Since a course platform will undoubtedly contain many online participants, a mechanism to schedule these tasks individually is needed.

The next steps on our agenda would be to create a user-friendly platform hosted on JupyterHub. This platform would allow students to access their assignments, solve them and submit them for auto-grading and feedback. It would also give instructors the ability to customize their courses and grading criteria. Furthermore, integrating an auto-grading tool to JupyterHub would undoubtedly benefit both instructors and students. This tool would assist instructors in providing both formative and summative feedback to their students, aiding them in improving their programming skills and course progress.

6 CONCLUSION

At HPI, we are always trying to create the best environment for students to learn and grow their skill set. We want to exploit the benefits of JupyterHub by integrating it into our platform. JupyterHub offers a supportive and well-structured interface that would benefit our learners. The survey, that is presented in this study, was the first step in our road to creating a user-friendly infrastructure, which will provide an interactive platform to solve assignments and receive formative feedback. Moreover, it will give the instructors the ability to tailor the courses and automate the assignment solving process. Evaluating the observations of the instructors on our platform was a key phase in our development process.

REFERENCES

[1] Robert J. Brunner and Edward J. Kim. 2016. Teaching data science, In ICCS 2016. The International Conference on Computational Science. Teaching Data Science. *Procedia Computer Science* 80, 1947–1956. https://doi.org/10.1016/j.procs.2016.05.513

[2] Raffaele Cioffi, Marta Travaglioni, Giuseppina Piscitelli, Antonella Petrillo, and Fabio De Felice. 2020. Artificial intelligence and machine learning applications in smart production: Progress, trends, and directions. *Sustainability (Switzerland)* 12 (1 2020), 121–195. Issue 2. https://doi.org/10.3390/su12020492

[3] Sir John Daniel. 2012. Making Sense of MOOCs: Musings in a maze of myth, paradox and possibility. *Journal of Interactive Media in Education* 515 (2012), 1139–1144.

[4] Jessica B. Hamrick and Jupyter Development Team. 2016. *2016 Jupyter Education Survey*. Jupyter Development Team. https://doi.org/10.5281/zenodo.51701

[5] Samuel Lau and Joshua Hug. 2018. nbinteract: generate interactive web pages from Jupyter notebooks. *Master's Thesis, Master's thesis* Part F128771 (2018), 1139–1144.

[6] Hamza Manzoor, Amit Naik, Clifford A. Shaffer, Chris North, and Stephen H. Edwards. 2020. Auto-grading jupyter notebooks, In WebCAT - An autograder for student solvable Jupyter Notebooks. *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE* Part F128771, 1139–1144. https://doi.org/10.1145/3328778.3366947

[7] Jonathan Reades. 2020. Teaching on jupyter – using notebooks to accelerate learning and curriculum development. *Region* 7 (2020), 21–34. Issue 1. https://doi.org/10.18335/region.v7i1.282

[8] Ambika Selvaraj, Vishnu Radhin, Nithin KA, Noel Benson, and Arun Jo Mathew. 2021. Effect of pandemic based online education on teaching and learning system. *International Journal of Educational Development* 85 (9 2021), 1139–1144. https://doi.org/10.1016/j.ijedudev.2021.102444

[9] Helen Shen. 2014. Interactive notebooks: Sharing the code. *Nature* 515 (11 2014), 151–2. https://doi.org/10.1038/515151a

[10] Thomas Staubitz, Hauke Klement, Jan Renz, Ralf Teusner, and Christoph Meinel. 2015. 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)*. Association for Computing Machinery, Potsdam, Germany, 23–30. https://doi.org/10.1109/TALE.2015.7386010

[11] Thomas Staubitz, Hauke Klement, Ralf Teusner, Jan Renz, and Christoph Meinel. 2016. CodeOcean - A Versatile Platform for Practical Programming Excercises in Online Environments. In *CodeOcean - A Versatile Platform for Practical Programming Excercises in Online Environments*. Association for Computing Machinery, Potsdam, Germany. https://doi.org/10.1109/EDUCON.2016.7474573

[12] Eric Van Dusen. 2020. *Jupyter for Teaching Data Science*. Association for Computing Machinery, New York, NY, USA, 1399. https://doi.org/10.1145/3328778.3372538

[13] Andrea Zonca and Robert S. Sinkovits. 2018. Deploying jupyter notebooks at scale on XSEDE resources for science gateways and workshops, In Deploying jupyter notebooks at scale on XSEDE resources for science gateways and workshops. *ACM International Conference Proceeding Series* Part F128771, 1139–1144. https://doi.org/10.1145/3219104.3219122