A Powerful Labs Environment for Computer Science Courses

Dr. Chi Yan Daniel Leung, zyBooks, A Wiley Brand

Chi Yan (Daniel) Leung is the Content Software Engineer (Labs lead) at zyBooks. He oversees the content creation and maintenance of labs across different titles at zyBooks. Before joining zyBooks, he was a lecturer at the School of Engineering at the University of California at Merced. He received his Ph.D. in Computer Vision from the University of California at Merced.

Joseph Mazzone, zyBooks, A Wiley Brand
Ms. Efthymia Kazakou, zyBooks, A Wiley Brand

Efthymia Kazakou is Sr. Assessments manager at zyBooks, a startup spun-off from UC Riverside and acquired by Wiley. zyBooks develops interactive, web-native learning materials for STEM courses. Efthymia oversees the development and maintenance of all zyBo

Chelsea Gordon, zyBooks, A Wiley Brand

Chelsea Gordon received her PhD in Cognitive Science at University of California, Merced in 2019. Chelsea works as a research scientist for zyBooks, a Wiley company that creates and publishes interactive, web-native textbooks in STEM.

Dr. Alex Daniel Edgcomb, zyBooks, A Wiley Brand

Alex Edgcomb is Sr. Software Engineer at zyBooks, a startup spun-off from UC Riverside and acquired by Wiley. zyBooks develops interactive, web-native learning materials for STEM courses. Alex actively studies and publishes the efficacy of web-native lear

Dr. Yamuna Rajasekhar, zyBooks, A Wiley Brand

Yamuna Rajasekhar is Director of Content Development at zyBooks, a Wiley Brand. She is an author and contributor to various zyBooks titles, and leads authoring and research across all disciplines. She was formerly an assistant professor of Electrical and Computer Engineering at Miami University. She received her M.S. and Ph.D. in Electrical and Computer Engineering from UNC Charlotte.

A Powerful Labs Environment for Computer Science Courses

Abstract

Programming assignments are key to any computer science course. In today's digital landscape of education, summative assessments, often called labs, are assigned on a weekly basis to students. The goals of these assessments are often to reinforce and to evaluate mastery of the concepts taught in the course. Upon graduation, students are tasked with programming complex projects. A key aspect of a CS student's success in the real world is their ability to develop complex software in professional IDEs (integrated development environments). In this paper we describe a new and powerful labs environment that enables students to master their skills in software development through a cloud-based IDE with support for over 50 programming languages. This labs environment supports an auto-grader and professional unit testing frameworks. Additionally, the labs environment also provides instructors the opportunity to collaborate with students in real-time. We present student usage and behavior data from use of these labs in 300 introductory programming courses across 219 universities.

Introduction

Most introductory computer science (CS) courses assign one or more programming tasks each week. In decades past, such programming assignments were graded by hand. A student was provided a prompt and perhaps initial program files. The student may also be provided some example usages of the program. Then, the student would go to a computer lab to develop the program. During lab hours, a student that was stuck could ask the instructor. However, often, a student would not complete the assignment during lab hours, so would have to wait for office hours to get an instructor's help. To submit, a student would upload the developed program files, then wait a week or more for grading to be completed and feedback to be provided.

In the last decade, many auto-graded programming assignment systems have been developed, both in academia and commercially [1–4]. Such systems are often web-based, save instructor's time with grading, and provide students more rapid feedback. Such systems have enabled instructors to switch from assigning one-large-program to many-small-programming assignments each week, wherein each assignment was more focused on a particular programming concept.

One such system, zyLabs, was released by zyBooks in late 2015. zyLabs focused on ease of use for introductory CS courses. Since then, 100s of thousands of students have submitted programs against more than 200 programming exercises in each of the popular programming languages (C, C++, Java, and Python). The simplicity of the system and the many-small-programs approach has allowed students in such programming courses to master one to two concepts at a time, without being distracted by the sometimes complex settings of an integrated development environment (IDE). Prior research and feedback from instructors and students have also confirmed that such programming labs are effective learning materials for early learners [5]. However, zyLabs was less effective in providing large-scale programming experiences in subsequent programming courses due to the lack of support for a full IDE, popular frameworks, command-line tools, starting web servers, rendering graphic user interfaces, and more.

This paper describes a new system, Advanced zyLabs, that provides powerful features, developed to be effective for large-scale assignments and advanced courses. Then, the paper seeks to answer whether such powerful features hurt student outcomes.

Advanced zyLabs

At the beginning of 2023, zyBooks released Advanced zyLabs, including over 50 programming language configurations, a professional-grade Linux development environment, powerful instructor tools, and advanced auto-grading capabilities. The goal of Advanced zyLabs was to provide students and instructors with an easy to use yet professional-grade programming environment directly in their core learning material, called a zyBook. Advanced zyLabs automatically installs and opens additional environments needed for an assignment, reducing student frustration and saving student's time, as well as instructor's time from needing to help debug each unique student device. Instead, students can focus on learning and instructors can focus on teaching. No less, each student (and instructor) has a highly reproducible programming environment accessible on the web from any device with internet access.

Advanced zyLabs offers students an accessible programming environment from their CS1 course, while also introducing a professional-grade tool without the complexities of system setup. Advanced zyLabs provides a consistent environment in the subsequent courses as well, minimizing distractions and challenges associated with switching tools. Additionally, the professional-grade tool enables students to develop practical skills, such as code testing, maintenance, and collaboration, that supplement the individual concepts taught in various computer science courses [6].

Industry-standard capabilities

Figure 1: Advanced zyLabs IDE running a Python lab with IDE on top and console on bottom.

Advanced zyLabs is well-suited to large, real-world projects and lab assignments. Advanced zyLabs supports industry-standard IDEs, custom package installs, use of command-line tools, having large data files in the lab, and much more. Tools such as git/GitHub, vim, and programming language compilers/interpreters are all available in the lab's interactive Bash console session. Advanced zyLabs allows users to start web servers and create full-stack web applications with React, Angular, Node.js, and Django. Advanced zyLabs even allows users to access the Linux machine's desktop so they can build desktop GUI applications.

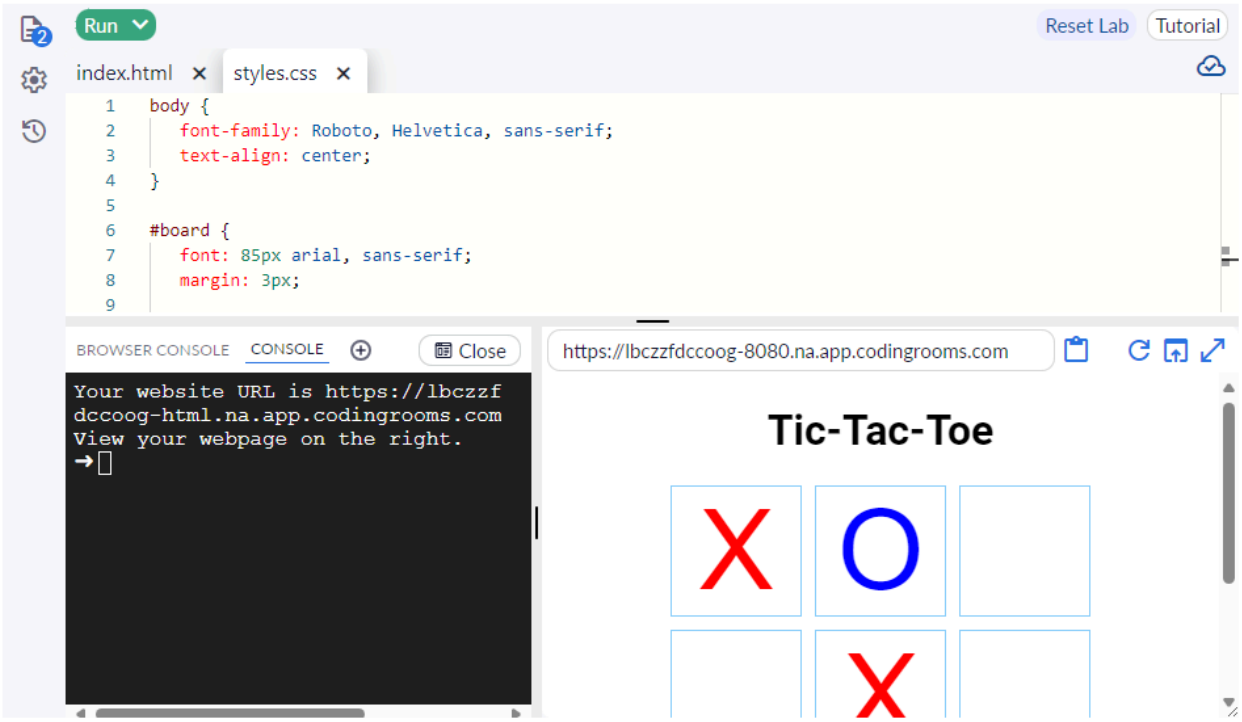


Figure 2: Advanced zyLabs web programming lab with IDE on top, browser console in bottom-left, and rendered webpage on bottom-right.

Fully-integrated, auto-graded student projects and lab assignments

Instead of expecting students to create HTML, CSS, and Javascript files in an external editor then upload such files for auto-grading, Advanced zyLabs enables a student to write such code directly in the environment and a student to test incremental developments with live-rendered previews of their webpage. Advanced zyLabs works on a real web server, so creating auto-graded assignments on full-stack web development with popular client or server-side frameworks like React, Angular, and Node.js is supported. Likewise, Advanced zyLabs allows students to access large datasets from the live MySQL database on Linux. With the integration of MySQL Workbench, one of the most popular query tools, students can now learn database development, administration, and architecture in this visual tool directly in the zyBook. Furthermore, the support of Python and Java with MySQL in Advanced zyLabs means that instructors can create auto-graded assignments of real-life database applications.

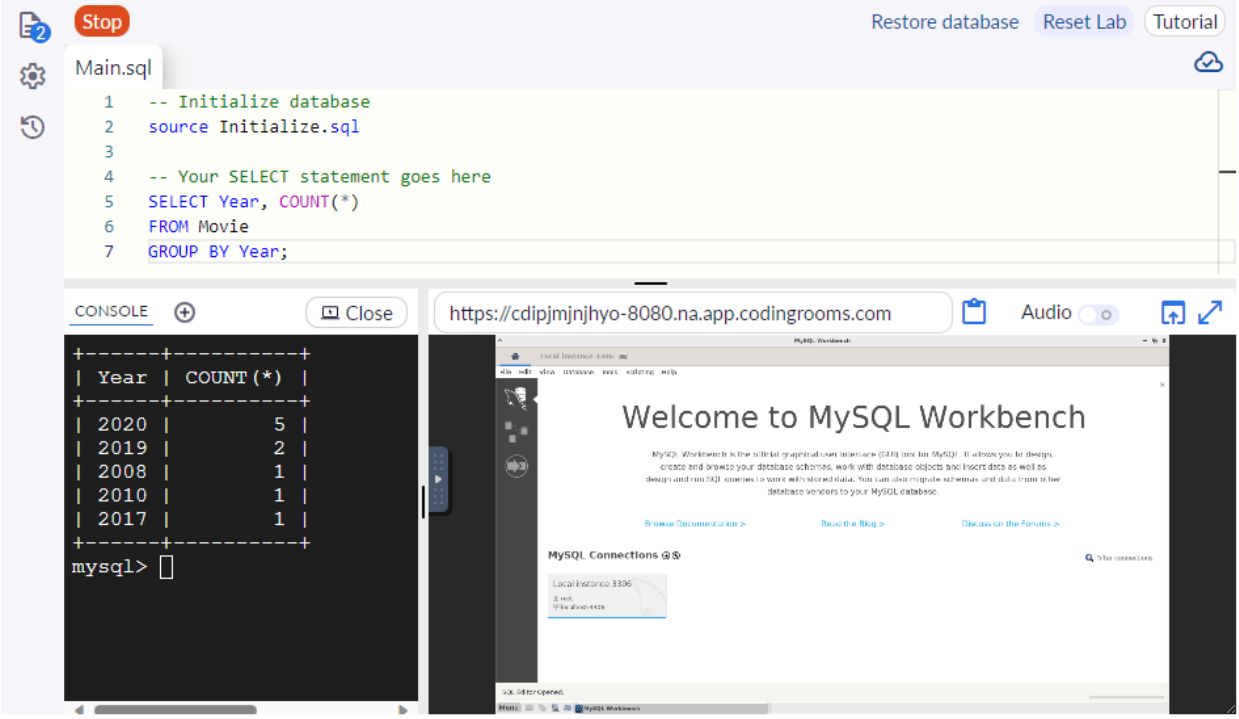
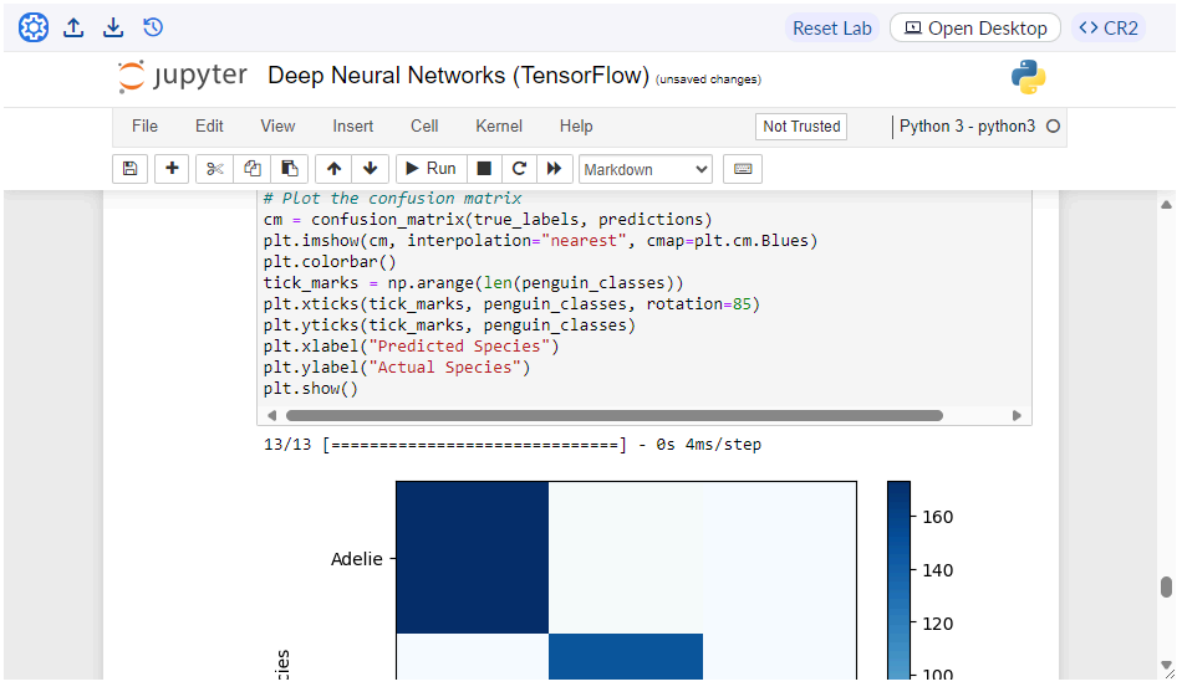


Figure 3: Advanced zyLabs with MySQL console on left and MySQL Workbench on right.

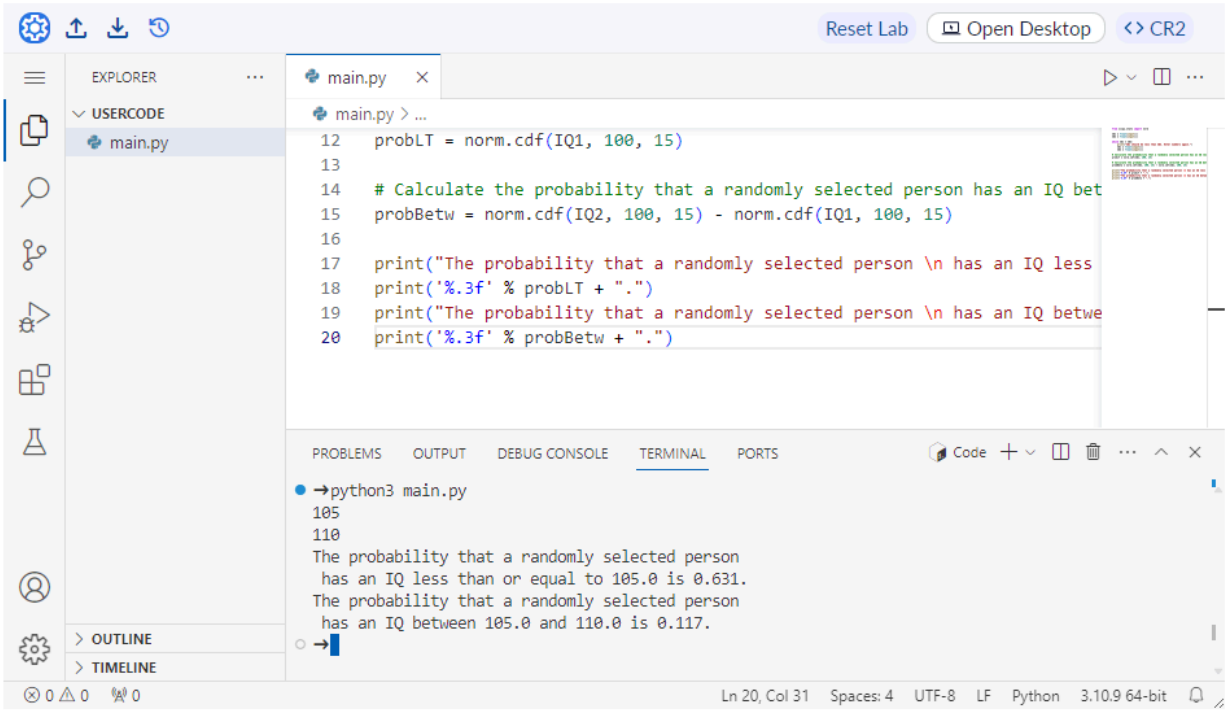
Instructor creatable and customizable

zyBooks provides pre-made programming assignments; however, an instructor can create and edit a programming assignment. An instructor can choose the IDE that students are provided: One of the custom-developed IDEs or an industry-standard IDE, like Visual Studio Code, Jupyter Notebook, or RStudio. An instructor can write the assignment's prompt and define the files to give the student. An instructor can add, edit, or remove various types of test cases: Simple input-output test; more advanced autograded test cases, such as using JUnit and GoogleTest; a bash script with custom logic to run in the Linux auto-grading environment; and even a manually-graded test case.



Submit for grading

Figure 4: Advanced zyLabs using Jupyter Notebook IDE with Python code.



Submit for grading

Figure 5: Advanced zyLabs using Visual Studio Code IDE with Python code

Collaboratory features

Even with autograding and rapid feedback, a student may still get stuck and need instructor help. Advanced zyLabs enables an instructor to enter a student's coding environment, edit code, and run commands, along with the student. Such a collaborative feature reduces the need for a student to send the instructor screenshots, and then the instructor to try to recreate the issue.

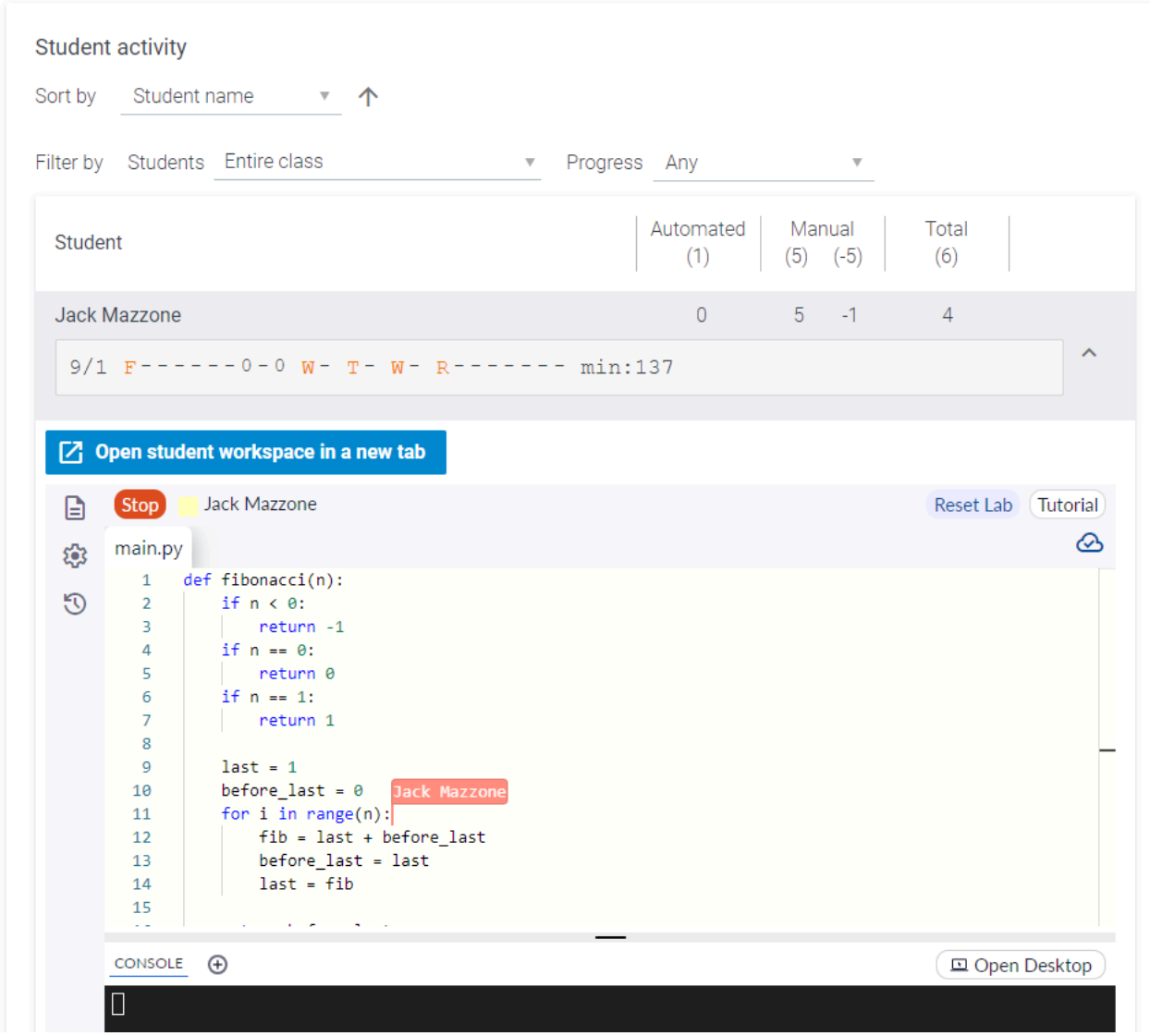


Figure 6: Instructor viewing the live edits of student Jack Mazzone, who is working in an IDE.

Advanced zyLabs saves each edit of each text file by a student. An instructor can playback a student's edit history, keystroke-for-keystroke. This includes the code a student ran on each development run, the code a student had on each graded submission, and all of the steps in between.

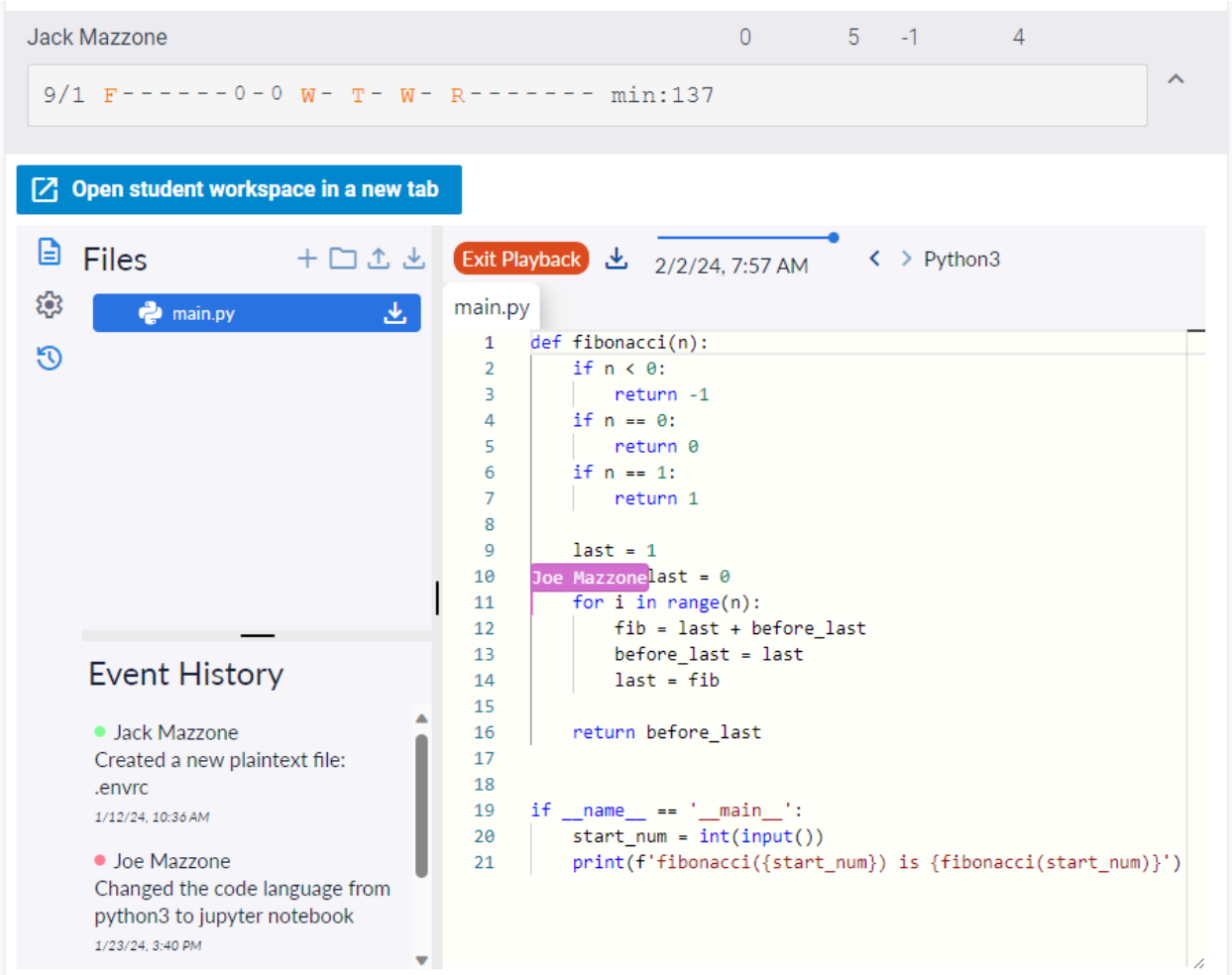


Figure 7: Playback history open for student Jack Mazzone.

Advanced zyLabs offers many powerful student-facing features. The remainder of this paper focuses on how these features affect student outcomes.

Early usage data

Methods

A total of 25,196 students used Advanced zyLabs, from 506 courses at 219 institutions. We identified the lab activities that were most commonly assigned in Python, C++, and Java zyBooks. Instructors opted in to using AZLs during such and such time frame. Any course using these labs was included in this dataset.

We defined the following metrics to help analyze student usage of the lab activities between the original zyLabs (classic labs) and the new Advanced zyLabs (advanced labs).

Student-level: For each student, the following metrics were applied to each lab activity:

● Completion: Whether the student completed the activity. Value of 1 was assigned if the student completed the lab activity by submitting a correct answer at some point for that activity. Otherwise, a value of 0 was assigned.
● Number of tries: The total number of submissions for a particular lab activity. Each time a student submitted code for a lab activity, we interpreted that submission as one more try of that lab activity by that student. We stopped counting once a correct submission was made for that lab activity by that student.
● Time spent: The estimated number of hours between a student's first and final submissions on a lab.

Lab-level: The following metrics were applied to each lab activity:
● Completion rate: The percentage of students who completed the lab activity.
● Average number of tries: Of students who completed the activity, the sum of each student's number of tries divided by the number of students.
● Average time spent: Of students who completed the activity, the sum of each student's time spent divided by the number of students.

Results

Figures 8–9 show the lab-level metrics across for ten selected Python lab activities, in the order in which they are typically assigned. The metrics are shown for advanced lab activities completed during Fall 2023, and for classic lab activities from Fall 2022, for comparison. The average completion rate was 93% for advanced labs and 94% for classic labs. The average time spent was 7.2 hours for both advanced and classic labs. The average number of tries was 4.0 for advanced labs and 3.8 for classic labs.

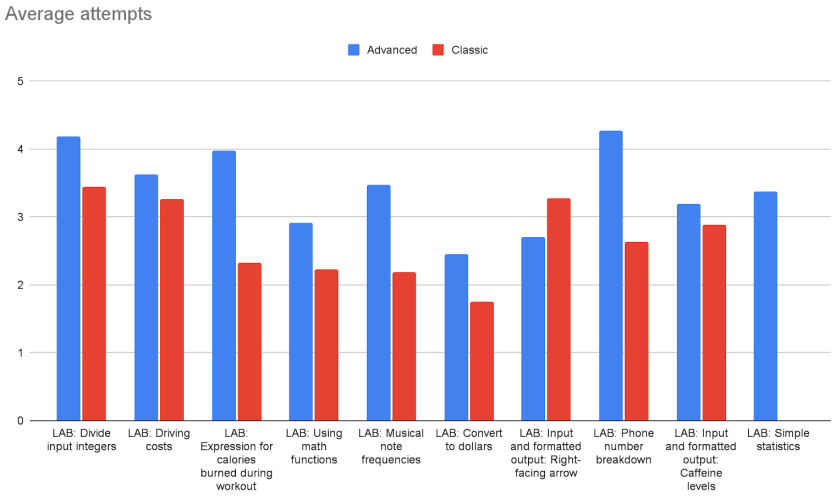


Figure 8: Average attempts on selected Python labs for advanced and classic lab platforms.

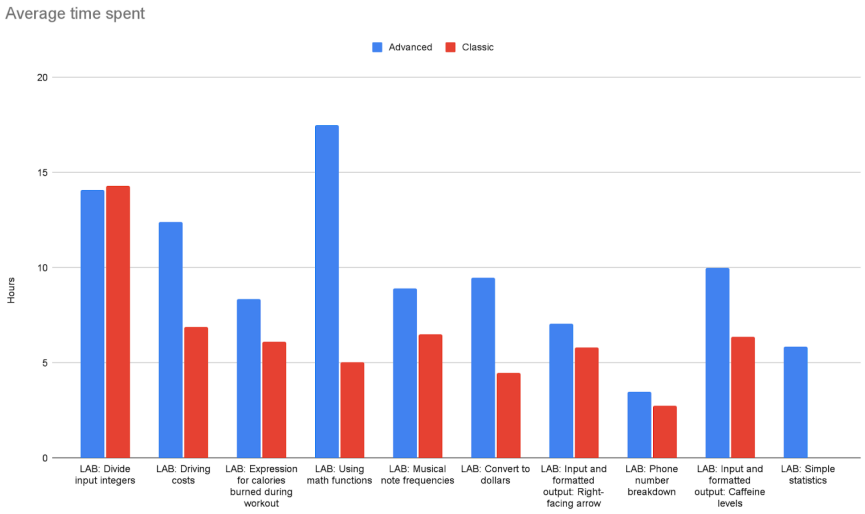


Figure 9: Average hours spent on selected Python labs for advanced and classic lab platforms.

Figures 10–11 shows the lab-level metrics across ten selected C++ lab activities, in the order in which they are typically assigned. The metrics are shown for advanced lab activities completed during Fall 2023, and for classic lab activities from Fall 2022, for comparison. The average completion rate was 96% for advanced labs and 93% for classic labs. The average time spent was 3.5 hours for advanced labs and 5.3 hours for classic labs. The average number of tries was 3.9 for both advanced and classic labs.

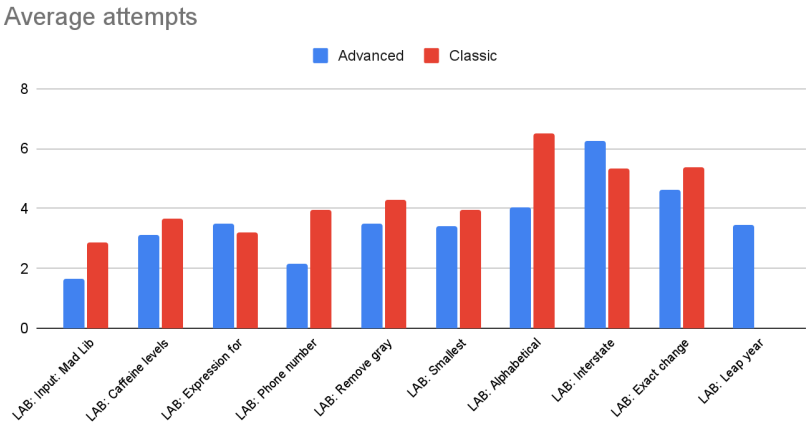


Figure 10: Average attempts on selected C++ labs for advanced and classic lab platforms.

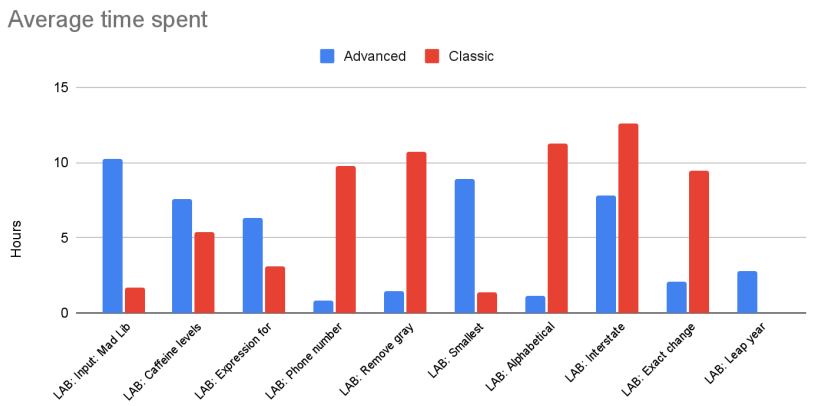


Figure 11: Average hours spent on selected C++ labs for advanced and classic lab platforms.

Figures 12–13 shows the lab-level metrics across ten selected Java lab activities, in the order in which they are typically assigned. The metrics are shown for advanced lab activities completed during Fall 2023, and for classic lab activities from Fall 2022, for comparison. The average completion rate was 96% for advanced labs and 94% for classic labs. The average time spent was 8.3 hours for advanced labs and 7.4 hours for classic labs. The average number of tries was 3.6 for advanced labs and 3.8 for classic labs.

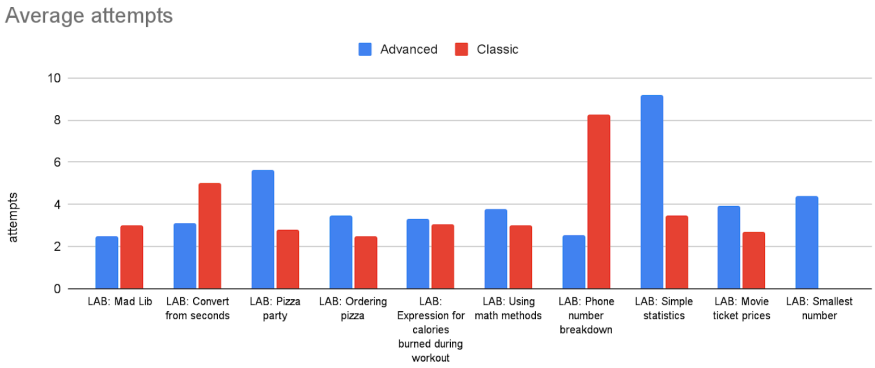


Figure 12: Average attempts on selected Java labs for advanced and classic lab platforms.

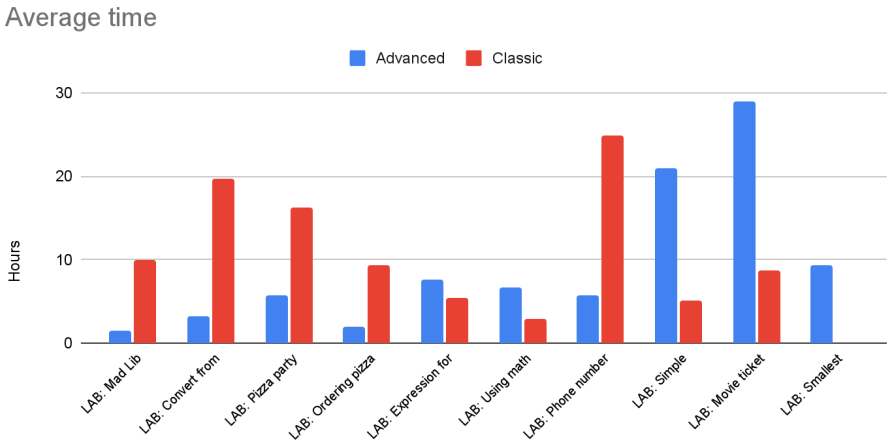


Figure 13: Average hours spent on selected Java labs for advanced and classic lab platforms.

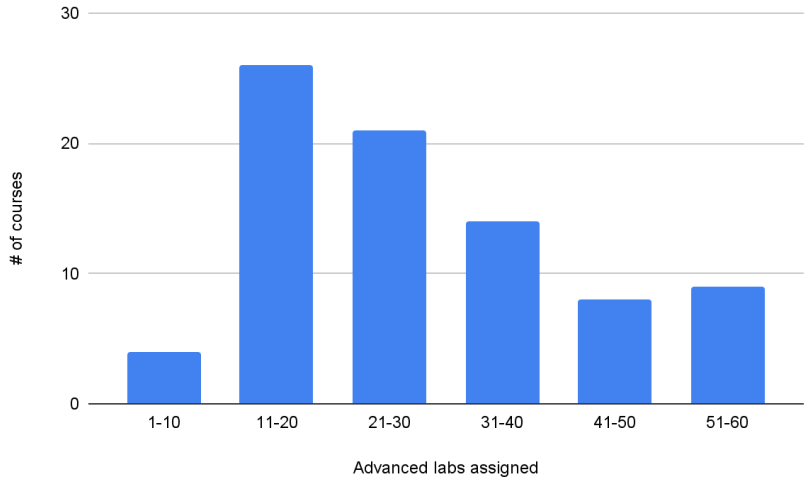


Figure 14: Histogram of number of advanced labs assigned per course.

Lab system	Language	Completion rate	Time spent (hours)	Number of tries
Advanced	Python	93%	7.2	4.0
Classic	Python	94%	7.2	3.8
Advanced	C++	96%	3.5	3.9
Classic	C++	93%	5.3	3.9
Advanced	Java	96%	8.3	3.6
Classic	Java	94%	7.4	3.8

Table 1: Summary of early usage data. Advanced and classic labs had similar measures for each metric.

Discussion

Early data usage of the advanced labs platform looks promising. The struggle, measured by time spent and number of submissions on assignments, appears to have stayed consistent after introducing advanced labs. For C++ labs, we actually observed a slight decrease in time spent on most of the advanced labs. Further investigation is needed to understand this decrease in time spent, and more precise measurement of actual time spent *working actively* on the lab is needed for this investigation.

Limitations

This research shows the average time spent and number of attempts on labs. The data was not limited to particular courses with specific characteristics, but all courses using the labs. This means that the largest courses have high influence on the data, and looking more deeply at targeted courses may provide deeper insight. As instructors could choose whether or not to assign advanced labs instead of classic labs, there is also a self-selection bias in the data, where all of the advanced labs users are in courses with instructors who wanted to use advanced labs. Finally, the time spent metric used in the analysis is coarse. A more accurate measure of how much time a student spends *actively working* on their programming assignment will improve this work in the future.

Future Directions

The powerful advanced labs platform opens doors to many new features and analytics. Future research may look at whether more time is saved on grading for teaching assistants and instructors with advanced labs. Further, future work may also look at usage of the collaborative programming feature to evaluate how the feature gets used and measure how collaborative programming impacts student outcomes.

The playback feature of advanced labs allows an instructor to recreate a student's code at any point in time. Future work may develop insights into student behavior using such data. One such insight may be determining when a student begins to struggle excessively, such as if a student submits many times in a short period without making many changes and without achieving a higher score. Another insight may also be to determine when suspicious behaviors take place, such as pasting in large sections of text, which in some cases might indicate cheating.

Hints: Preliminary data

This subsection provides a step into a particular future direction: Providing students with hints. Many students in introductory computer science courses struggle excessively with their coding homeworks. Such excessive struggle is often caused by a specific logic or syntax error. A student may be unable to resolve such a struggle from such an issue, and the student's learning may be negatively impacted.

To provide assistance in real time and reduce such struggle, zyBooks recently released an experimental hints system in Advanced zyLabs as a limited beta. A hint was generated with an AI language model and

integrated with Advanced zyLabs, allowing students to generate and view hints on each failing test case. Every time a student failed a test case, a button showed underneath the test case with the label "**Help me with this test**" as shown in Figure 15.

Figure 15: Failing test on advanced lab with hints enabled.

Each hint provided a targeted message to help the student identify and understand the issue. A student could select between two options to get another hint for the same test case. As shown in Figure 16, a student can choose "**Explain it another way**" to get the same hint explained differently, or **"Provide me with a more detailed explanation**" to get a more explanation on the hint provided.

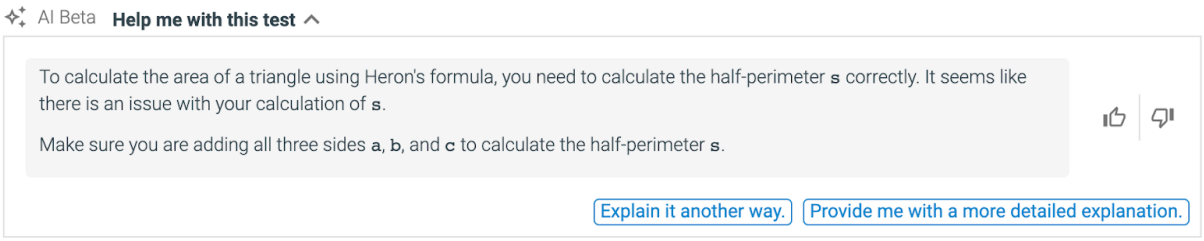


Figure 16: AI generated hint for failing test case.

So far the hint system has been used by 9 courses, 55 students and has resulted in 344 total conversations and 923 total messages with an average conversation including 2.68 messages. Figure 17 shows the conversations distribution by zyBook, and Figure 18 shows the hint conversations distribution by student. Finally, Figure 19 shows the distribution of messages in conversations.

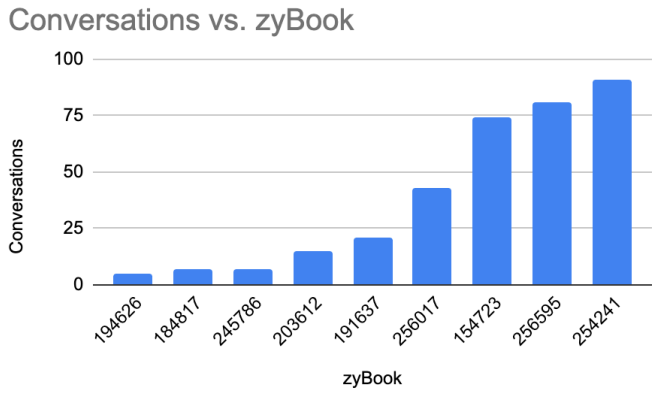


Figure 17: Hint conversations across all 9 zyBooks with AI hints enabled.

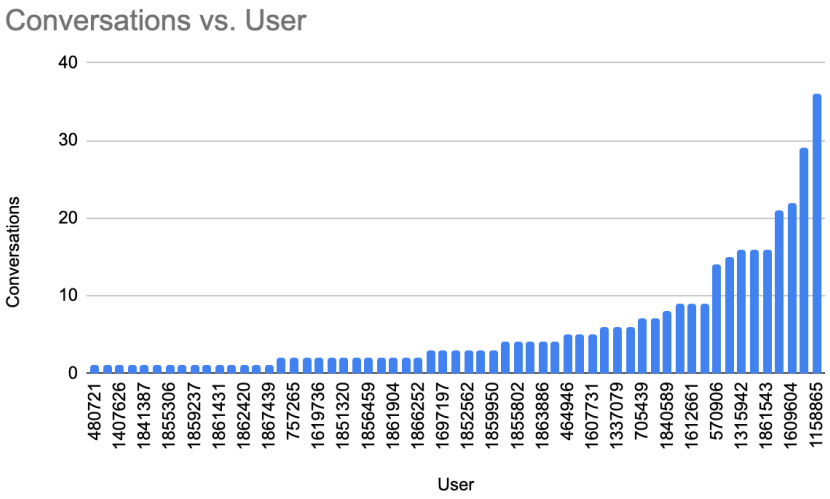


Figure 18: Hint conversations across all 55 students that generated at least one hint.

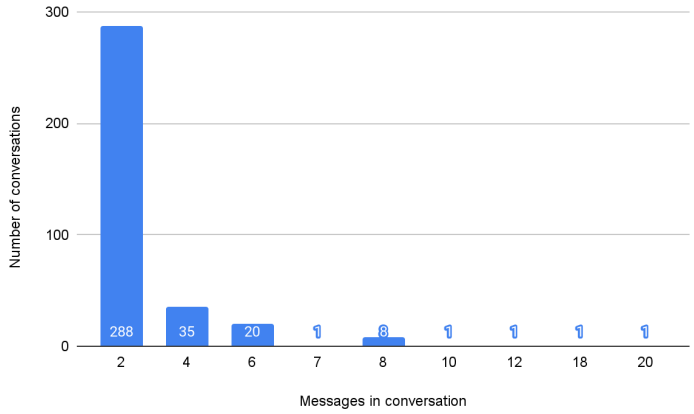


Figure 19: Distribution of hints messages in conversations

Note that these 55 students were included in the "Early usage data", which had 25,196 students, so were 0.2% of the total students therein. Future work may measure the impact of such hints on student struggle. Future work may also include analyzing the efficacy of such AI generated hints and the impact such hints had on programming labs' completion. Such analysis may compare courses where hints were provided and courses where hints were not provided for the same problems, including controls for other confounds, such as different instructors, course offerings, student demographics, and more. Future work may also evaluate student self-efficacy, including a student's belief that the hint system impacted that student's self-efficacy.

Conclusion

Advanced zyLabs includes many powerful features, for students and instructors, including industry-standard IDEs, highly-customizable development environment and tools, Linux machine's desktop, collaborative environments, and more. Nonetheless, each metric of student usage was about the same: 93-96% average completion rate, 3.5-8.3 average time spent (hours), and 3.6-4.0 average number of tries. Such measures indicate that Advanced zyLabs do not impede student outcomes. Future work may analyze novel features of Advanced zyLabs, such as the hints system, and may measure the impact on student outcomes specifically in advanced computer science courses.

References

[1] M. Sherman, S. Bassil, D. Lipman, N. Tuck, and F. Martin, "Impact of autograding on an introductory computing course," Journal of Computing Sciences in Colleges, vol. 28, no. 6, pp. 69-75, Jun 2013.
[2] R. Pettit, J. Homer, R. Gee, S. Mengel, and A. Starbuck. "An Empirical Study of Iterative Improvement in Programming Assignments." in Proceedings of the 46th ACM Technical Symposium on Computer Science Education, SIGCSE, pp.410-415, Feb 24 2015.
[3] G. Haldeman, A. Tjang, M. Babeş-Vroman, S. Bartos, J. Shah, D. Yucht, and T.D. Nguyen, "Providing meaningful feedback for autograding of programming assignments," in Proceedings of the 49th ACM Technical Symposium on Computer Science Education, SIGCSE, pp. 278-283, Feb 21 2018.
[4] H. Keuning, J. Jeuring, and B. Heeren. "Towards a Systematic Review of Automated Feedback Generation for Programming Exercises," in Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '16, pp. 41-46, Jul 2016.
[5] Gordon, C. L., Lysecky, R., & Vahid, F. (2021, July). "The rise of program auto-grading in introductory cs courses: A case study of zylabs," in *2021 ASEE Virtual Annual Conference Content Access*.
[6] Essi Lahtinen, Kirsti Ala-Mutka, and Hannu-Matti Järvinen. "A study of the difficulties of novice programmers," in SIGCSE Bull. 37, 3 (September 2005), 14–18.