CS 102 *Spring 2020/21*		Project Group **G2C**
Instructor:	**Aynur Dayanık**	
Assistant:	**Haya Shamim Khan Khattak**	

Criteria	**TA/Grader**	**Instructor**
Presentation		
Overall		

~ LabConnect ~

Borga Haktan Bilen	22002733
Vedat Eren Arıcan	22002643
Berkan Şahin	22003211
Berk Çakar	22003021
Alp Ertan	22003912

Requirements Report

(version 3.0)

March 12, 2021

1 Introduction

LabConnect facilitates communication between students, TA's, tutors, and instructors. In the background, it is mainly a web application (If sensible/necessary, it may possibly be ported to Android) that aims to assist CS introductory courses in terms of organization and communication. Proposed ideas for features include priority queuing for TA zoom rooms among many other enhancements to TA/instructor productivity. For example, those who have completed their labs can be tested using pre-defined (by TA or instructor) unit tests, if students pass the tests successfully then they will be ordered by the number of visits to TA in the same session, in order to decrease waiting times for the students who are waiting from the beginning, and to optimize the process in general. TA's can also use the system to see previous versions of each student's code in a more practical way, similar to real version control managers in spirit. The style guidelines put forth by the instructors can be enforced automatically by parsing the student's sent code files. Much of the repetitive work that course staff need to do can be reduced substantially by automated actions, allowing TA's to allocate time for more hands-on help towards students. The student experience can be improved further by adding helpful features such as personal notes for students and so on.

1.1 Related Work

As a part of their curriculum, most of the big engineering institutions and universities have hands-on laboratory sessions, that are mandatory for students to attend them. At some particularly crowded institutions; for each lab session, every student has to submit approximately 9 to 12 assignments. Additionally, students and the teaching staff alike may also need to deal with lab tests, perhaps even multiple of them. Cumulatively, the amount of submissions can reach approximately 10000 per semester. Even if there were 20 evaluators, each evaluator would need to take care of almost 500 assignments. Without the help of automation, evaluators spend most of their time grading and testing work, rather than creating more useful assignments for students or simply spending more time assisting them [1].

Some of the positive sides of the automated grading are syntactic correctness, maintainability and efficiency. Moreover, automated grading systems weigh down the lack of objectivity in conventional grading and these systems can track all of the building process. In the contemporary context, these kinds of automated grading systems are commonly used by some commercial competitive programming and recruitment sites (for instance, HackerRank, TopCoder and HackerEarth) [2].

At the same time, the quantity of the feedback given by the teaching staff is directly impactful to the effectiveness of the student's work that is done after the class. The students have an easier time studying weak areas if they are informed about their performance properly. The autonomous nature of instructor feedback frees everyone from manual assessment, which is a time consuming and unproductive task [3].

If there are many more students than there are teachers, this can hinder the process of manual assessment of the student's code. Although this problem can be solved by increasing the amount of teachers, this solution can harm the university economically. The idea of automated assessment comes into play with its ability to solve this problem without increased costs. There is much evidence to assert that student code is evaluated more properly using automatic assessment rather than using pen&paper testing. Research also interestingly shows that automated assessment reduces the rate of dropout in programming majors [4].

According to one source, the effectiveness of different testing procedures are as follows: Unit testing is able to catch 50% of the errors. Integration testing is able to catch 40% of the errors. Regression testing is able to catch 30% of the errrors [5].

The evaluation system decreases the work load of the evaluation process since it makes the evalution process easier. Just as the system has automatic testing, it also comes with automatic grading. Additionally, it manages students by aiding grouping and scheduling issues, helping with the students checking in to lab sessions, the deadlines and overall project grades. As the system is ideally available at all times, the students will be able to get feedback instantly on the correctness of their solutions. This way, students are able to keep working on their work at their own pace [6].

And finally, it was seen that the approach used for automation of assignments in education is used in DevOps fields in a similar way [7].

2 Details

LabConnect is designed to contain three user interfaces for instructors, students, and assistants/graders. It will also contain a server-side program where the submissions are stored and tested. This section will provide details on proposed features of the software at the time of this writing.

2.1 LabConnect - Instructor Side

2.1.1 Task Definition Stage

- The instructor is able to input the name and the programming language of the assignment.
- The instructor can upload the instructions either as a document, or as a Markdown or a plaintext file, after which, it will be rendered and displayed on the website.
- The instructor can configure the unit tests as input-output pairs and group them if they wish. Some groups of unit tests can be hidden, in which case they won't be shown to the students prior to submission.
- The instructor can provide a tester class that tests the code provided by the student by calling the methods. This class is the one that will be run by the server. This allows students to be more flexible in their own main class.
- The instructor can determine a time constraint for unit tests. If the execution of the code takes longer than the determined time, it will fail said test.
- The instructor can determine a time frame for submissions. They can determine a seperate deadline for re-submissions, if they wish.
- The instructor can assign students to assistants either at random or by hand. They can also choose to not assign assistants at all, in which case the students will be assigned to the assistants during the lab session, based on the length of the queue.
- The instructor can either define PAIN-style (Proficient, Acceptable, Incomplete, Nothing) tiers for grading or can define certain criteria for which assistants can enter a numeric value. If they decide to use tiers, they can define certain tiers (such as Proficient and Acceptable) as "complete", in which case the submission of a student receiving said grades will automatically be classified as complete. If they decide to use numeric grades, they can define a certain threshold that needs to be exceeeded for a submission to be classified as complete.

2.1.2 Task Update Stage

- The instructor should ideally have better control over how students' codes are tested. The instructor is also able to add more unit tests as the lab progresses. The students' unit tests can be updated within the lab period, so any mistakes made on the tests themselves can be corrected this way.

2.1.3 After the Lab

- The common errors that are made by students such as missing documentation (i.e. JavaDoc) of the written code, and predefined conventions that aren't followed (naming conventions, styling guidelines, etc.), will be detected by LabConnect. This data will be shared with the student and their instructor. The instructor can later on determine to act up on the most important mistakes that are made by the students.

- Instructors are able to see which unit test cases of the program the students mostly fail at. These unit tests can again reveal the common weaknesses of programmers.

- The instructor is able to assess their students properly by considering their performance in the lab, for which, LabConnect will supply additional analytics. As in, LabConnect will provide detailed performance of the student, based on the mistakes they made, and overall unit test scores. This information can help the instructor achieve a better grasp of their students since the data is properly organised and accessible.

2.2 LabConnect - Student Side

2.2.1 Code Submission Process

- The students' files can be individually uploaded as specified by the instructor during the task specification stage. As such, they won't have to reupload their whole assignment each time a certain file needs to be modified.

- Automated tests and checks are run on the server side each time the student submits their assignment.

- Submissions that fail any test are marked as incomplete and the student is redirected to the revision stage. Otherwise, the student is placed into the code review queue.

2.2.2 Code Review Queue

- The prerequisite of the code review queue is to pass all (or possibly most) unit tests.

- Students waiting in line are shown how many students are waiting in front of them. Student places in the queue are stored on the server side, so in the event of a momentary disconnection, places in the queue are preserved.

- Students at the top of the queue are sent a prompt to confirm that they are ready to be graded by the TA, after which they will be sent a meeting link. In the event that they fail to confirm in a certain amount of time, they will lose their place in the queue and will be sent to the end of the current queue.

2.2.3 Code Revision Process

- Students whose submission is marked as incomplete either by the TA or the system will be redirected to the revision stage.

- The student can either receive written feedback from their TA or a message generated server-side explaining roughly what went wrong during the tests. In case of a runtime exception/compiler error, the stack trace is also provided and the lines causing the problem are highlighted. Code can also be highlighted manually by the TA, as described in the Code Review Process section of the TA interface.

- Students are able to resubmit their files once they are confident that they have solved the issues in the given feedback.

2.3 LabConnect - Grader/Assistant Side

2.3.1 Code Review Process

- At the start of the lab session, the assistants enter their meeting links to the system. This allows for the links to be distributed to the students when it is their turn.
- The assistant will be directed to the code review interface which contains information about the student, as well as the source code submitted by the student and a field to write feedback about the submission.
- The assistant/grader can choose to browse the code submitted by the student from their computer without wasting bandwidth and time with screen sharing. They can also write feedback referencing specific lines in the code to make their feedback clearer.
- After evaluating the code, the assistants can either give the assignment a score or pick from the tiers determined by the instructor. If the determined grade or tier is within the satisfactory threshold, the student's assignment is marked as complete. Otherwise, the assignment is returned to the student with the feedback written by the TA.
- The assistant can manually confirm that they are ready for the link to be revealed to the next student in queue. Hence, allowing the assistants to take short breaks if necessary.

2.3.2 Task Revision Process

- The program is able to detect common errors made in the student's code. Furthermore, the unit tests can be arranged in a way such that they test specific skills that are required. These enhancements to the revision process are aimed to increase the efficiency of the graders during live lab sessions.
- The Grader can create announcements for the students if there is a highly repeated mistake. This way, they won't need to repeat the same small correction for each student, and they will find it easier to correct other unique mistakes that the students have made. The graders may find greater motivation on teaching the students proper techniques if they are freed from the repetitiveness of correcting the same mistakes. Of course, the student can demand further help for their "repeated" mistakes.

2.4 LabConnect - Server Side

2.4.1 Task Definition Stage

- The distribution of students to the TAs, whose registration should be done by instructor (probably once at the beginning of the semester), are stored in the database.
- For each TA, a unique identification number is generated and the students who are allocated to a TA will get the identification number of that TA.

- For each lab, every student will have their own git repository (created automatically by the system) for the storage of the code on the server-side.
- The test cases (inputs & outputs), that are uploaded by instructor, will be stored in the database.

2.4.2 Post-Submission Stage

- Each submission is stored in a version control system. This allows LabConnect to provide easy access to submission history for TAs and instructors.
- If the current submission doesn't differ from the last submission, it is automatically rejected.
- When a student makes a submission, the server-side program then compiles the source code in the submission files, if the language picked by the instructor is a compiled language.
- Then the server-side program runs either the resulting binary if the language is a compiled language, or the interpreter for the language if it is an interpreted one.
- If there's an error during the compilation or interpretation/execution stages, the submission is automatically failed and the error message is shown to the student.
- Using I/O redirection, the input parts of the unit tests are fed to the standard input of the tester program and the output is captured. This allows for easy multi-language support and doesn't require students to deal with file I/O while writing the program.
- If the output of the program doesn't match the expected results, the submission is automatically failed and the input, expected output and actual output for the specific unit test is shown to the student.
- The submission is run separately for each unit test. These runs are timed and the results for each unit test is stored in a database. If the instructor has specified a time limit for tests, the test in question will automatically fail once the time limit is reached.

2.4.3 Queue Management

- Once a submission passes all the automated tests, the student who made the submission is added to the queue for their TA.
- The students that haven't yet received any manual code review are prioritized over those that have received one. This allows each student to get their code reviewed at least once during the labs.
- If the TA's were not assigned to students by the instructor, the server will pick a TA on the spot based on the queue length. This behavior will help make the workloads of the TA's more balanced.

2.4.4 Statistics Generation

- After the lab is finished, the data acquired from the unit tests, the error detection algorithm, etc. is used to create statistics about the students' performance. The server can maintain this data for further use.

3 Summary & Conclusions

The main purpose of LabConnect is to provide students and instructors a place where everyone will have an easier time collaborating with each other properly. Especially in online education, instructors can have a hard time monitoring students' performance on labs. With LabConnect, instructors can shape and take control of their classes according to the response given to the lab sessions.

There are some types of software (for instance, Mooshak, Ejudge) that are similar to what LabConnect aims to achieve, however, they are not directly developed to supply instructors/TAs/students with quality of life enhancements in terms of CS courses and their lab sessions/assignments. These kinds of software usually exist for the sake of judging programming contests automatically, which is a vital task for contests with high participation, similar to the vitality of automating CS course evaluations with a high participant count. Other related systems include: Code Wars, Hackerrank, Leetcode, and so on. These systems also are not intended to be used for interactive education, but rather for self-education. However, the popularity and success of these self-education systems nonetheless suggests a viability for the type of automation that LabConnect is attempting to undertake.

A CS course that successfully integrates LabConnect to their education process may enjoy the benefit of automation over numerous aspects of the course that are often inefficiently evaluated by the teaching staff manually. Students may also have an easier time as features such as the queue system are designed to improve their communication process with the teaching staff. The downside of this integration may be that the flexibility of an entirely manual evaluation process will inevitably be higher compared to the flexibility offered by LabConnect. This downside, however, is not as problematic as it may seem. The teaching staff may choose to include only certain parts of the LabConnect system, continuing to perform the remaining parts manually as per usual. The development of LabConnect aims to provide such modularity to balance the compromise of flexibility as much as possible.

References

[1] Amit Kumar Mandal, Chittaranjan Mandal, and Chris Reade. “A System for Automatic Evaluation of Programs for Correctness and Performance”. In: *Web Information Systems and Technologies*. Ed. by Joaquim Filipe, José Cordeiro, and Vitor Pedrosa. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 367–380. ISBN: 978-3-540-74063-6.

[2] Felipe Restrepo-Calle, Jhon J. Ramírez Echeverry, and Fabio A. González. “Continuous assessment in a computer programming course supported by a software tool”. In: *Computer Applications in Engineering Education* 27.1 (Sept. 2018), pp. 80–89. DOI: 10.1002/cae.22058. URL: https://doi.org/10.1002/cae.22058.

[3] Felipe Restrepo-Calle, Jhon Jairo Ramírez-Echeverry, and Fabio A. Gonzalez. “UNCODE: INTERACTIVE SYSTEM FOR LEARNING AND AUTOMATIC EVALUATION OF COMPUTER PROGRAMMING SKILLS”. In: *EDULEARN18 Proceedings*. IATED, July 2018. DOI: 10.21125/edulearn.2018.1632. URL: https://doi.org/10.21125/edulearn.2018.1632.

[4] Aldo Gordillo. “Effect of an Instructor-Centered Tool for Automatic Assessment of Programming Assignments on Students’ Perceptions and Performance”. In: *Sustainability* 11.20 (2019). ISSN: 2071-1050. DOI: 10.3390/su11205568. URL: https://www.mdpi.com/2071-1050/11/20/5568.

[5] Steve Fenton. “Automated Testing”. In: *Pro TypeScript: Application-Scale JavaScript Development*. Berkeley, CA: Apress, 2018, pp. 245–256. ISBN: 978-1-4842-3249-1. DOI: 10.1007/978-1-4842-3249-1_10. URL: https://doi.org/10.1007/978-1-4842-3249-1_10.

[6] Aguiar Nogueira et al. “Automatic Evaluation of Programming Projects”. In: 2011.

[7] Frank Faber. “Testing in DevOps”. In: Jan. 2020, pp. 27–38. ISBN: 978-3-030-29508-0. DOI: 10.1007/978-3-030-29509-7_3.