CS 102 *Spring 2020/21*

Project Group **G2C**

Instructor: **Aynur Dayanık**
Assistant: **Haya Shamim Khan Khattak**

Criteria	TA/Grader	Instructor
Presentation		
Overall		

~ LabConnect ~

Group Name

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

Requirements Report

(Draft Version (pre-review))

February 25, 2021

1 Introduction

LabConnect facilitates communication between students, TA's, tutors, and instructors. In the background, it is mainly a web application (can be ported to Android possibly) that aims to assist CS introductory courses in terms of organization and communication. Proposed ideas for features include priority queuing for TA zoom rooms. For example, those who have completed their labs can be tested using pre-defined (by TA or instructor) unit tests, if studens 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 Literature Review

As a part of their curriculum most of the big engineering instutions and universities have a laboratory session, that is mandatory for student to attend them. Each lab session, every student has to submit approximately 9 to 12 assignments. Additionally, every student takes lab tests (up to three lab test). Cumulatively the amounts of submissions are approximately 10000 per semester. Even there is 20 evaluators, each evaluator should almost take 500 assignments. Without the help of automation, evaluators spent most of their time to grading and testing work rather than creating more useful assignments for students [1].

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

At the same time, the quantity of the feedback given by teachers is directly proportional to the effectiveness of the student work that is done after the class. The students find easier time studying for the areas of least performance if they are informed about their performace properly. The autonomous behaviour of the instructor feedback frees everyone from the manual assessment, which is a time consuming task [3].

If there are much more students than there are teachers, this can hinder the proccess of manual assessment of student code. Although this problem can be solved with increased teacher number, this solution can damage the university economically. The idea of automated assesment comes to play with its ability to solve this problem without increased costs. There is many evidence that shows that student code is evaluated more properly using automatic assesment rather than using paper pencil testing. Evidence also shows that automated assesment reduces the rates of dropout in programming majors [4].

The effectiveness of different testing procedures is 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 proccess easier. As the system has automatic easy tests, it also comes with automatic grading. It also manages students by aiding the grouping and scheduling issues, by helping with the students checking in, the deadlines and overall project grades. As the system is avaliable all the times, the students will be able to get feedbacks instantly on the correctness of the solution. This way the students can do their work on any wanted pace [6].

And finally, it was seen that the approach used for the automation of assignments in education is used in the 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.

2.1 LabConnect - Instructor Side

2.1.1 Task Definition Stage

- The instructor decides upon the name and the language of the assignment.
- The instructor uploads the instructions either as a document, or as a Markdown or a plaintext file, in which case it will be rendered and displayed on the website.
- The instructor writes the unit tests as input-output pairs and groups 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 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 determines 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. Thrcide 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 put a certain threshold that needs to be exceeeded for a submission to be classified as complete.

2.1.2 Task Update Stage

- The instructor will have better control over how code of the students' are tested. The instructor will be 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 itself can be corrected this way.

2.1.3 After the Lab

- The common errors that are made by the student such as missing documentation of the written code, conventions that aren't followed (naming conventions, styling guidelines), 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.
- With the unit tests, instructors will be able to see which end cases of the program that the students mostly failed at. These unit tests can again show the common weaknesses of the programmers.
- The instructor will be able to assess their students properly by considering their performance in the lab. 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 to have more idea about their students since the data is properly organised and accessable.

2.2 LabConnect - Student Side

2.2.1 Code Submission Process

- The students will upload the individual files specified by the instructor during the task specification stage one by one. That way, they won't have to reupload the whole assignment if only one specific file is problematic.
- Once the student submits their assignment, the automated tests and checks are run in the server side.
- If the submission fails any test, it is 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

- When the student's submission passes all unit tests, they are placed into the code review queue.
- When the student is put to the queue, they can see how many students are in front of them. Their place in the queue is stored in the server side, so in case of a momentary disconeection, they don't lose their place.
- Once the student reaches to the very front of the queue, they have 30 seconds to click an "Iḿ ready" button to receive the meeting link for their TA. Otherwise, it is assumed that the student is not ready and they are placed at the end of the queue.

2.2.3 Code Revision Process

- If the student's submission is marked as incomplete either by their TA or automatically by the system, they are directed to the revision stage.
- In this stage, the student either receives written feedback by their TA or they receive a message generated server-side explaining what went wrong during the tests. In case of a runtime exception/compiler error, the backtrace 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.
- Once the student is confident that they fixed the problem, they can re-submit their code as described in the Code Submission Process section of the Student interface.

2.3 LabConnect - Grader/Assistant Side

2.3.1 Code Review Process

- At the start of the lab, the assistants will enter their meeting links to the system. This will allow for the links to be distributed to the students when it is their turn.
- After the assistant enters their meeting link, they will be directed to the code review interface This interface will contain 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 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.

- Once the assistant finishes evaluating a submission, they need to manually confirm that they are ready for the link to be revealed to the next student in queue. This allows the assistants to take short breaks if necessary.

2.3.2 Task Revision Process

- As stated, the program will be able to detect common errors made in student code. Also, the unit tests can be arranged in a way they that they test specific skills that are required. During the lab, these properties of the program will increase the efficiency of the graders.

- The Grader can create pre-messages 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 this way they will find easier time to correct other unique mistakes that the students have made. The graders will find greater motivation on teaching the students proper techniques if they are freed from the repetitiveness of same mistakes. Of course the student can demand further help for their "repeated" mistake.

2.4 LabConnect - Server Side

2.4.1 Task Definiton Stage

- The distrubition of students to the TAs, that should be uploaded by instructor (probably once at the beginning of the semester), will be stored in the database.

- For each TA a unique identification number will be 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 by system) for the storage of the code.

- The test cases (inputs & outputs), that is uploaded by instructor, will be stored in the database.

2.4.2 Post-Submission Stage

- Each submission will be stored in a version control system. This allows LabConnect to provide easy access to submission history for TAs and instructors.

- If the current submisson doesn't differ from the last submission, it will be automatically rejected.

- When a student makes a submission, the server-side program will compile the source code in the submission files if the language picked by the instructor is a compiled language.

- Then the server-side program runs the resulting binary if the language is a complied language or runs 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 part of the unit tests are fed to the standard input of the program and the output is captured. This allows for easy multilanguage 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 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 will be added to the queue for their TA.
- The students that didn't receive any manual code review are prioritized over those that received one. This allows for 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. will be used to create statistics about the students' performace. The server will be used to 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 find easier time collobrationg with each other properly. Especially in online education, instructors can find hard time monitoring student performance on labs. With LabConnect, instructors can shape their classes according to the reponse taken from the labs.

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.