COLLEGE OF COMPUTING AND INFORMATION SCIENCES

A programming assignment autograder for moodle

By

CS18-02

DEPARTMENT OF COMPUTER SCIENCE

SCHOOL OF COMPUTING AND INFORMATICS TECHNOLOGY

A Project Proposal Submitted to the School of Computing and Informatics Technology For the Study Leading to a Project Report in Partial Fulfillment of the Requirements for the Award of the Degree of Bachelor of Science in Computer Science Of Makerere University

Supervisor

Ernest Mwebaze

School of Computing and Informatics Technology, Makerere University

emwebaze@gmail.com, +256-772-121-272

October, 2017

GROUP MEMBERSHIP

SNo	Names	Registration Number	Signature
1	ABULE AUGUSTINE ARUMADRI	15/U/2633/PS	
2	AYIKO JEREMIAH SARA	15/U/186	
3	OWOMUGISHA ISAAC	15/U/12351/PS	
3	SENDIKADDIWA MARVIN	15/U/1154	

Contents

1 Introduction

1.1 Background

E-learning is a style of teaching and learning done over a network. It is independent of location and time constraints in regard to attendance. As soon as the materials (lectures notes, assignments, etc.) are uploaded, a student can access them at any time. It offers a way to design courses in a more interactive way e.g. through the use multimedia and is cost effective. Today, multiple universities all over the world use e-learning platforms to enhance understanding of course content such as Harvard, Stanford, and MIT [1]. With time more efficiently spent, multiple assignments can be given and automatically graded to identify and improve areas that the students did not understand fully.

Moodle is an e-learning platform designed for educators, administrators and learners with a single robust, secure and integrated system to create personalized learning environments [4]. It is easy to use and open source. Moodle will be used as the core system on which the e-learning platform will be designed. Makerere University already uses Moodle to run it's e-learning platform called MUELE [3]. We are building the system on top of moodle because of the overwhelming challenges involved in building a system from scratch (such as security considerations and user management issues).

1.2 Problem Statement

Monitoring each student's performance in Makerere university's classes is a tiresome task for lecturers. When students are given a programming course work, they expect fair and continuous feedback from their lecturers in the least time possible. However, at times students are given complex tasks to do as course work. For a lecturer to give reasonable feedback to every student's course work, he would be required to run the students code into the IDE to correctly analyze the running time, correctness and generate relevant feedback. Besides analyzing code efficiency and correctness of the code, the lecturer is also subjected to the task of checking for plagiarism among the submitted code. The current prevailing solution to improve the lecturer's productivity is grouping students. The problem with this is that only a few students contribute to the course work and the rest do not benefit. This is also not a permanent solution since only group effort is monitored at the expense of individual effort.

This project's intention is to solve the above problem by creating an on-line system that will keep track of a student's individual progress in the programming field and monitoring the student's general performance in the least time possible. With this implemented, student's submitted course work will be attended to quickly, plagiarism acts will be detected, teachers' productivity will be improved and student's passion for programming and computer science will be boosted.

1.3 Objectives

1.3.1 Main Objective

The general intention of the project is to develop a Moodle plugin that will automatically grade programming assignments uploaded to the system by students, keep track of each students performance based on scores attained from previously uploaded assignments and provide relevant feedback both to the students and the lecturer. The system should also enable lecturers to give in-class assignments for real-time assessment.

1.3.2 Specific Objectives

- To design a Moodle plugin that will automatically grade the submitted programming assignments
- To implement a system that will keep records of submissions and scores attained, categorized according to student registration numbers
- To design a feedback system to both the lecturer and student, on basis of records stored.
- To design a system that will detect plagiarized source code among submitted assignments.

1.4 Scope

Computer programs and source code exist in many programming languages and size (in terms of actual lines of code) and they require different computer resource to execute properly. In order to support a broad range of courses, it is important to build a general platform that can support as many languages as possible. However, the aim of this project is to build a proof of concept platform. Thus, we mainly be concentrating on the C and Java programming languages. However, in the future we want the system to be platform independent.

When designing a platform used for automatic grading purposes, accurate assessment is of utmost importance. The programming assignments must therefore be tailored with the intention of keep the auto-grading process simple and and stable while at the same time challenging and relevant to students in order to hone their programming skills at the same time pass that particular course unit. Another important aspect is user feedback. The system is designed to be a substitute of the lecturer when it comes to assessment. Therefore it must provide relevant feedback to the students regarding use of better or correct algorithms, and also to the lecturer on topics where students are finding challenges.

For purposes of testing the effectiveness of the working system, we will focus on programming course units taught in the Computer Science course only within Makerere University. Programming languages covered in this course are C and Java.

1.5 Significance

A delivered working system will be of great help to both students and lecturers who use it as part of their teaching tools. Many at times students do not get the personal attention from lecturers as they so do desire. This may be attributed to the large number of students offering a particular course unit, each most likely with their unique challenges they face. There also exists the general perception that university students are not supposed to be spoon-fed, but rather engage in rigorous research and extensive personal reading. Much as this encourages academic independence, many students need one on one academic guidance. Our auto grading system provides this guidance.

By having the ability to suggest better algorithms and language structures/functions to use, each student can be guided on better coding techniques and how to optimize their programs. In the current setup at CIT, this is very difficult to achieve through manual means by the lecturer.

By enabling the lecturer to give in-class assignments that can be automatically graded, students are encouraged to constantly practice programming skills in order to be prepared for these assignments. This has a direct positive impact on their academic performance in that particular course unit, at the same time improving their general programming skills even though not examinable.

2 Literature Review

In this section, we provide a summary the previous work that has been done on E-learning platforms, auto-graders and online judges.

Many of the leading universities of computer science education around the world incorporate auto grading in their programming and algorithms courses. For example, in 2013, MIT researchers Singh et al [1] introduced a new method for automatically providing feedback for introductory programming problems. According to [2], due to fair and timely feedback results from online judge websites, online practice outperforms traditional programming practice.

2.1 Related work

2.1.1 Muele and Moodle

MUELE (Makerere University E-learning Environment) [3] is the dominant e-learning platform used at Makerere University. It is used for publishing notes and assignments for students. It also has functionality for giving on-line quizzes (mainly objective type and single-response questions). MUELE is based on the moodle online learning platform.

One of the most dominant e-learning platform used worldwide is Moodle. The moodle website [4] states that "Moodle is a free and open-source software learning management system written in PHP and distributed under the GNU General Public License. Developed on pedagogical principles, Moodle is used for blended learning, distance education, flipped classroom and other e-learning projects in schools, universities, workplaces and other sectors." The Moodle platform is composed of several components that enable its dynamism and thus can be adopted for the project.

"Plugins are a flexible tool set, allowing Moodle users to extend the features of the site" [4]. Plugins for moodle can be thought of as modules or widgets with specific functionality. Currently moodle is consists of thousands of plugins developed to suite different teaching curriculum. Some of the existing plugins relevant to our project are: URKUND plagiarism plugin [5] used for checking plagiarism in submitted files and Reengagement plugin [6] for sending timely feedbacks to remind students of uncompleted course activities. Since some of these plugins are open source, they can be used as sub modules in our project.

2.1.2 Online judges and autograders

The most popular application of online judges is in programming contests such as TopCoder, ACM/ICPC and Google Code Jam. The most prominent of these is the ACM/ICPC [7]. The ACM/ICPC consists of algorithmic and programming problems for contestants to solve. The contestants' submissions are evaluated based on the output of their programs. A typical problem in this contest consists of the following parts:

- Problem description: This is a description of the problem to be solved.
- Input description: This gives the specification of the input file from which the users solution programs read data

- Output description: This specifies the format in which the program should produce results.

- Sample input and sample output: These provide examples to clarify the input and output specifications.

For each problem in ACM/ICPC, there are multiple sets of data to be processed. This is an important feature that serves two major purposes [2]: first, they are used to test that the program can work in all possible situations (including corner cases). Second, they are used to calculate the running time of a program (a good solution should be able to work on both small datasets and larger datasets). The use of multiple datasets therefore ensures that a students submission is evaluated comprehensively.

Test datasets of problems in the ACM/ICPC are either generated by hand for small-sized datasets or by programs written by jury members that generate datasets according to some predefined patterns or at random [8].

In [9], Dr. Antti Laaksonen describes how a competitive programming approach using an online judge to teach algorithm concepts has helped improve the problem solving skills of the students at the University of Helsinki. In the course, the TMC system [10] is used. TMC allows students to create their Java solutions in the NetBeans IDE and evaluates the solutions using JUnit tests. The tests are written by the course staff. In this paper, he describes how this model of teaching helps students understand the importance of developing efficient algorithms. He also points out some limitations of using online judges. He gives an example of the following problem from [11]:

Describe an $O(n)$-time algorithm that, given a set S of distinct numbers and a positive integer $k \leq n$, determines the k numbers in S that are closest to the median of S.

He points out the difficulty of automatically determining whether a students submission of the above problem is really $O(n)$, since the same problem can be solved in $O(nlogn)$ time. This therefore means that auto-graders are not perfect, however they seem to be better than traditional means of evaluation as the following section shows.

2.2 Benefits of auto-grading and practice oriented teaching of programming

Researchers Wang GP et al. conducted two sets of experiments [2] in order to evaluate the effectiveness and validity of using their online judge system called OJPOT as compared to traditional teaching methods. They set out to answer the following questions [2]:

- Does OJPOT work effectively to enhance students' practical abilities compared to the traditional teaching idea?

- In which aspects can OJPOT improve the student's practical abilities?

- Output description: This specifies the format in which the program should produce results.

In the first set of experiments, two classes were chosen: one class defined as the control class (CC) and another defined as the experimental class (EC). In the CC, the traditional

teaching idea was applied; while in the EC, the OJPOT online judge was used [2]. Before the experiment, both classes were given a pre-test to evaluate their elementary knowledge in C programming. Then during the semester, both classes were taught by the same teacher with the same timetable. The following data were collected during the experiment: pre-test scores, online practice and statistical data, post-test scores and course project scores. Below is a summary of the results of their experiment ([2]):

- Both classes had similar programming foundation before the experiment
- At the end of the semester, there were significant differences between the CC class and the EC class: the EC class has overall better results in the final tests and project.

They conclude their results by remarking that the EC class performed better because the online judge system made the students more enthusiastic about the material they were studying. It was also observed that the general programming abilities of the EC class were improved much better than those of the control class

3 Methodology

In this section, we explain the methods and tools that we're going to use to complete this project. The proposed system is going to be built as a plugin for the Moodle online learning platform. It will consist of a website and a mobile application. Users of the system will be lecturers and students.

3.1 Auto-grader/on-line judge

3.1.1 Method of grading

Grading shall be done by comparing the result of a student's submissions to the expected result. The students are going to be submitting their programs through a web interface, these programs shall be ran in the back-end and tested on various test data to determine the correctness of the student's solution.

Besides correct output, the lecturer might want the students' submissions to meet other requirements such as API specification, coding style or use of a particular language feature. In order to check that these requirements are met, the system shall use tools such as checkstyle [12] and PMD [13]. These tools provide mechanisms for extensive source code analysis.

3.1.2 Limiting resources

Since computer resources are finite, we have to incorporate a mechanism that limits how much resources the students' submission uses. Some of the resources to be limited include time, memory usage and access to restricted features of a language (for example features that could lead to security breaches). This can be done by running the code inside a virtual machine such that malicious code cannot affect the servers and the overall system. To limit amount of time used, a script that keeps track of how long the code has been running will be used to terminate the execution and send a Time Limit Exceeded verdict to the student if their code takes too long to run. A similar approach will be used to enforce memory usage constraints.

3.1.3 Tools available for the lecturer

The proposed system shall include a simple tool to allow lecturers to prepare assignments. This tool will enable lecturers to create model solutions to the assignments whose output will be compared to the students output. The tool will also provide an API to enable the lecturers to generate test data.

3.2 Analytics and other evaluation metrics

Various kinds of data will be collected from users' interaction with the system. This data includes (but is not limited to): number of tries a student uses to get the correct solution, number of students that have successfully completed a particular assignment, average score of students and difference in scores for assignments in particular topics.

We intend to use this data that is collected to provide useful feedback to lecturers in order for them to gauge how well they are teaching a particular course unit and where they can make improvements. Students shall also get feedback that will enable them to see where their weaknesses are and help them improve. We are going to explore various machine-learning algorithms in order to see how best to make use of this data. An example use of the data is determining the difficulty of a problem (or assignment) and the skills of a student. Given that we have collected data about how long it takes various students solve a problem, we can formulate this as a machine learning/statistics problem of the following form:

- Data is of the form: Student s solved problem p in time t_{sp}.
- We need to estimate student skills and problem parameters used for determining how difficult a problem is.

With a proper formulation of the problem, we can then apply algorithms such as gradient-descent [14] or a collaborative filtering algorithm like SVD [15] to make recommendations and provide insights into individual and general performance of the class.

3.3 Moodle

Moodle[4] is an open source platform for educators to develop and manage courses online. It's a modular system based on plugins. We plan on building this system as a plugin for Moodle. Moodle uses PHP as its underlying programming language and thus the components of this project will be primarily written in PHP. Building our proposed system on top of Moodle will ensure that we do not have to worry about issues like user management and security thus enabling us to focus on the core functionality of the application.

References

[1] R. Singh, S. Gulwani, and A. Solar-Lezama, "Automated feedback generation for introductory programming assignments," in *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '13, (New York, NY, USA), pp. 15–26, ACM, 2013.

[2] G. P. Wang, S. Y. Chen, X. Yang, and R. Feng, "Ojpot: online judge & practice oriented teaching idea in programming courses," *European Journal of Engineering Education*, vol. 41, no. 3, pp. 304–319, 2016.

[3] "MUELE." `https://muele.mak.ac.ug/`.

[4] "Moodle." `https://moodle.org/`.

[5] "URKUND plagiarism plugin." `https://moodle.org/plugins/plagiarism_urkund`.

[6] "Reengagement." `https://moodle.org/plugins/mod_reengagement`.

[7] "ACM ICPC." `http://icpc.baylor.edu`.

[8] M. Buzdalov, A. Buzdalova, and I. Petrova, "Generation of tests for programming challenge tasks using multi-objective optimization," in *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '13 Companion, (New York, NY, USA), pp. 1655–1658, ACM, 2013.

[9] A. LAAKSONEN, "A competitive programming approach to a university introductory algorithms course," *International Olympiad in Informatics*, vol. 11, pp. 87–92, 2017.

[10] M. Prtel, M. Luukkainen, A. Vihavainen, and T. Vikberg, "Test my code," *International Journal of Technology Enhanced Learning*, vol. 5, pp. 271–283, 2013.

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd ed., 2009.

[12] "Checkstyle." `http://checkstyle.sourceforge.net/`.

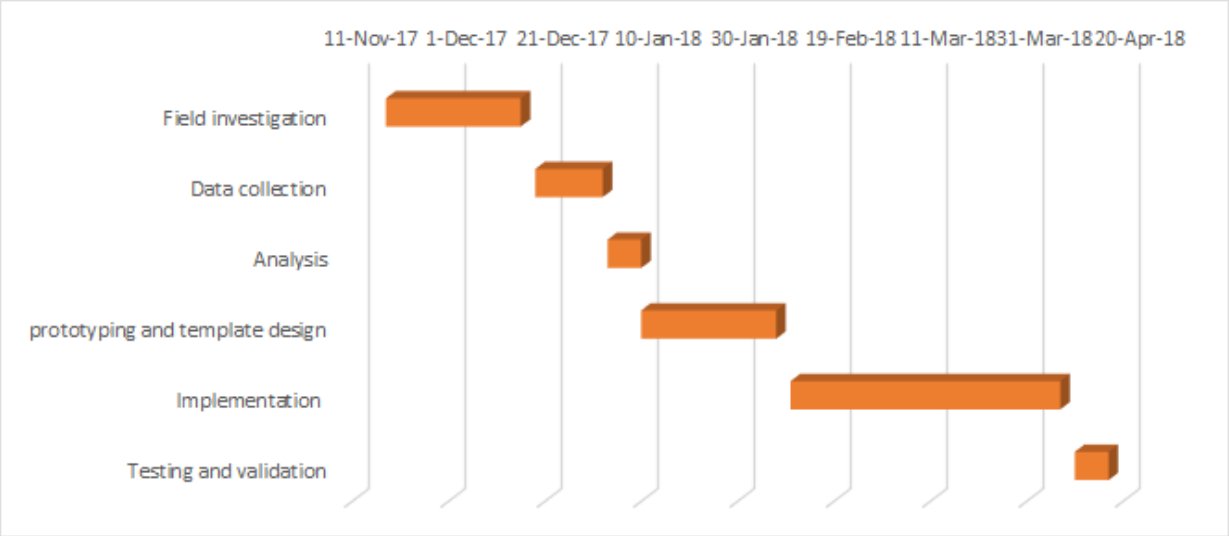
[13] "PMD." `https://pmd.github.io/`.

[14] M. Andrychowicz, M. Denil, S. G. Colmenarejo, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas, "Learning to learn by gradient descent by gradient descent," *CoRR*, vol. abs/1606.04474, 2016.

[15] D. Kalman, "A singularly valuable decomposition: The svd of a matrix," *College Math Journal*, vol. 27, pp. 2–23, 1996.

4 Appendices

4.1 Appendix A: Activities Gantt chart



4.2 Appendix B: Financial Requirements

Requirement	Quantity	Unit price (UShs)	Total price
Generating questionnaires	50	1,000	50,000
Data(Internet)	20gb	12,000	240,000
Private GitHub repository	1	32,400(per month)	64, 800
Miscellaneous	-	100,000	100,000
Total			454,800