Volume-2, Issue-7, November-2018: 79-87
International Journal of Recent Innovations in Academic Research ISSN: 2635-3040

E-Checker: A Secure Assessment and Interactive Feedback Generation System of Object Oriented-Based Programming Exercises with a Reliable Connascence Recognition and Encapsulation Tool

[1]Bucad, Maria Graciela Ramos and [2]De Castro, Erwin F.

Batangas State University JPLPC-Malvar, Malvar, Batangas, Philippines
Instructor, College of Engineering and Computing Sciences
[1]E-mail: mgracerbucad@gmail.com; [2]E-mail: decastroerwinf@gmail.com

Abstract: Programming is an essential portion of every technology course. For almost every technology student, they may say that programming is not an easy subject to learn; and that also goes for the instructors–teaching programming subjects has never been an easy task. Consider the usual scenario in one programming class inside a computer laboratory: a 1:45 teacher-student ratio in a 3-hour period held once a week, around 2-3 machine problems to be solved in a span of 2 hours by students using a programming language and finally, 1 hour left for the instructor to check and assess students' output. Barely, less than 2 minutes is allotted for the instructor to check, assess and record one student's work. The said scenario depicts one situation that lessens the efficiency of learning and teaching a programming language. Students hardly hear and get immediate feedback from instructors as to what areas needs to be improved. As for the students, feedback might be very important because it improves their learning experience and could help them become more motivated. The proponents consider the importance of having an automatic assessment tool that will help students improve their programming experience. A tool that can help lessen the effort of tedious analysis and grading huge amounts of similar student program solutions to teacher-provided machine problems; a tool that will incorporate software metrics and criteria like functionality, design, and style in program assessments; a computerized tool that will objectively grade and give immediate feedback to students developed programs.

Keywords: Automatic Feedback, Programming, Program Checker, Automatic Assessment Tool, Software Metrics.

Citation: Bucad, Maria Graciela Ramos and De Castro, Erwin F. 2018. E-Checker: A Secure Assessment and Interactive Feedback Generation System of Object Oriented-Based Programming Exercises with a Reliable Connascence Recognition and Encapsulation Tool. International Journal of Recent Innovations in Academic Research, 2(7): 79-87.

Introduction

Grading programming assignments and projects are similar to grading traditional assignments such as written essays. The primary distinctions between them are the unique keywords or constructs across different programming languages and the diverse possible solutions associated with a particular problem-solving technique. Traditional assessment for computer programming assignments and projects usually depends on an answer scheme that includes

the source code as a model answer with marks allocated to specific lines of code. This model answer is then used by the instructors to allocate marks to the students' programs based on the provided source code in the answer scheme. There are a lot of factors that contribute to learning (for students) and teaching (for instructors) programming subjects efficiently. A working computer unit, technology usage, well-trained instructors, focus and determined attitude for both students and instructors, good instructor-teacher relationship, well-defined feedback channel, to name a few. Many tools and approaches have also been devised to improve teaching and learning programming.

Consider the usual scenario in one programming class inside a computer laboratory: a 1:45 teacher-student ratio in a 3-hour period held once a week, around 2-3 machine problems to be solved in a span of 2 hours by students using a programming language and finally, 1 hour left for the instructor to check and assess students' output. Barely, less than 2 minutes is allotted for the instructor (in one laboratory session) to check, assess and record one student's work. The said scenario depicts one situation that lessens the efficiency of learning and teaching a programming language. Students hardly hear and get immediate feedback from instructors as to what areas needs to be improved. As for the students, feedback might be very important because it may improve their learning experience and could help them become more motivated.

As for the above scenario, the proponents consider the importance of having an automatic assessment tool in teaching programming subjects and consequently, will help students improve their programming experience. The proponents intend to develop a tool that can help lessen the effort of tedious analysis and grading huge amounts of similar student program solutions to teacher-provided machine problem; a tool that will incorporate software metrics and criteria like functionality, design, and style in program assessments; a tool that will objectively grade and give immediate feedback to students developed programs.

Review of Literature

Automatic grading of programs has existed in various fields for many years ago. A proposed method for evaluating C programs was developed by Arifi *et al., (*2016). In the project, two approaches are distinguished such as: static and dynamic analysis methods. The proposed method is based on static analysis of programs where the evaluated program is compared with the evaluator-provided program through a Control Flow Graphs. Unlike the dynamic analysis that requires an executable program to be evaluated, static analysis can evaluate a program even if it is not totally correct. A great challenge is to deal with multiple of solutions that exists for the same machine / programming problem.

As a solution to this, the authors proposed an innovative similarity measure that compares two programs according to their semantic executions (Arifi *et al.,* 2016). Xu and Zhang (2006) developed a prototype tool known as SimC. This tool automatically generates test data for unit testing of programs developed in C. and symbolically simulates the execution of the given program. The pointer operations are also simulated precisely making it capable of generating test data for programs involving pointer and structure operations (Xu and Zhang, 2006).Similarly, an open-source computer program called Spotter allows students to check their answers to symbolic homework problems was conceptualized and developed by Crowell (2006). The software can be used, copied, and modified freely, and full documentation is available online. The instructor installs the program and an answer file on a server, and students check their answers through a web browser. Common incorrect answers can be added to the answer file, along with an appropriate hint for the student (Crowell, 2003). A

generic assessment rubric for computer programming courses was devised by Mustapha *et al.,* (2016) to basically come up with a standardized grading system for different logics and constructs (but same output) given by students as answer/solution to programming problems. In the authors' scenario, it is not always the instructors who check and give marks to students; there is an involvement of either laboratory assistants or demonstrators.

This scenario led to grading inconsistencies in terms of the marks awarded when the same solution is being graded by different persons, hence, the generic assessment rubric was conceptualized. To further address this issue, a set of assessment rubric is necessary in order to provide flexibility for critical and creative solutions among students as well as to improve grading consistencies among instructors and teaching assistants or demonstrators. A rubric for each domain in computer programming courses such as: cognitive, psychomotor, and affective was incorporated to the developed rubric. (Mustapha *et al.*, 2016)

Objectives of the Study

This project aims to:

1. Devise a special tool that performs static code analysis of students' object-oriented-based exercises;
2. Perform automatic assessment and grading of dynamic object-oriented-based exercises;
3. Employ a flexible computerized object-oriented software metrics in assessing object-oriented-based (VB.Net, C++, Java) exercises;
4. Return immediate results and feedback to students and instructors;
5. Provide a progress-monitoring facility to students and instructors

Materials and Methods

Provided in figure 1 is the framework of the project, key features are presented to show (1) how laboratory assignments are submitted and gathered (repository), (2) running student programs and scoring the results vis-à-vis written instructor's solutions and (3) providing feedbacks and grading reports.The grading requirements include homegrown grading rubric. The back-end of the project's framework is a set of shell scripts that automate the batch grading of the students' submitted programming assignments. A web interface was provided to allow instructors and students send, view, mark, receive feedback on a real-time basis. Combination of script-based and web-based technologies were utilized to make the project functional.

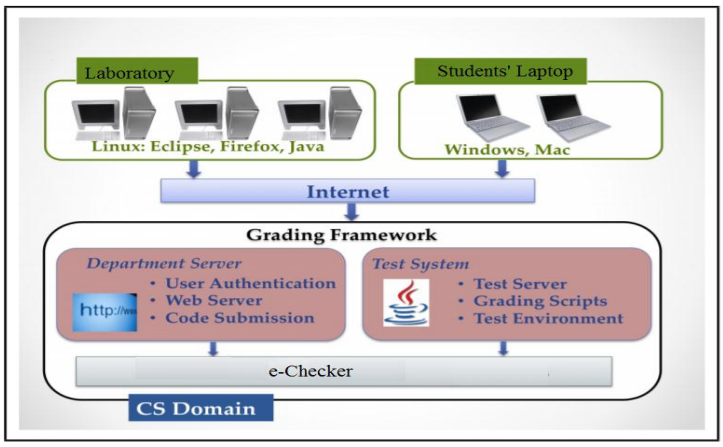

Figure 1. Framework of the project

Results and Discussion

Human grading of programming assignments is a tedious and error-prone task, a problem compounded by the large enrolments of many programming courses. The nature of programming assignments makes it perhaps the most difficult type of assignment to grade. As a result, students in such courses tend to be given fewer programming assignments and lesser feedbacks than should be ideally given. One solution to this problem is to automate the grading process such that students can electronically submit their programming assignments and receive instant feedback. Three main components were looked at in grading programming tasks/assignments: correctness, efficiency and maintainability. Manual grading can lead to great difficulty in judging the correctness of and efficiency of computer programs. Providing a web-based interface for both instructors and students is a great tool to support improved programming experience, improved grading consistencies by incorporating a generalized programming rubric to be used across all object-oriented programming languages.

It is obvious that great potential benefit can be reaped if the grading of programming assignments can be automated or at least computerized. There have been several studies into the feasibility of using computer networks and other technologies in programming courses. However, considering the potential benefits of automated grading, few academic institutions have implemented such a system because of the several issues that need to be addressed before such a system can be used. These include, but not limited to the danger of malicious programs, plagiarism, various psychological aspects of automated grading and technology-related requirements. As an action to such, students and instructors were provided with individual accounts to ensure security. A server was allotted to become repository of all submitted programming assignments, solutions made by instructors and marks automatically graded by the software.

Looking at the instructors' scenario, many tasks, such as grading and providing customized feedback on programming assignments, require instructors to go through and understand students' code. The workload of these tasks is prohibitively huge. However, skipping or delaying such tasks prevents instructors from keeping track of students' performance. On students' part, it is difficult for them to get prompt, customized feedback and help from the instructors. Although students may seek help from peers, peers are often not capable of helping or providing valuable feedback in many cases. Instructors or peers cannot always sit with students while the students are coding or provide prompt hints when the students encounter problem. Automation Tools such as the developed project is indeed needed to maintain the quality of education and provide solutions to mentioned pitfalls. The capability of quantifying behavioral similarity between programs is helpful for both teaching and learning programming.

Instructors' and Students' Portal

The project includes portals or modules for both instructors and students. In the instructors' module: facility for posting exercises, repository of submitted programming assignments and exercises and grading or marking options were added. For the students' module, they will have an interface: for viewing instructors' posted exercises/ programming assignments, uploading their accomplished files and a progress monitoring facility for feedbacks and marks given by instructors. When a student has completed an assignment/exercise, he or she will hit the submit program; this will take a copy of the student's source code and save it somewhere accessible only to the instructor. It also allows students to resubmit all or part of their assignment, at least up to the due date and perhaps thereafter, keeping track of the new

submission time. The system also keeps a log of each transaction, to support or refute student claims of system failure or unavailability (due, most often, to file systems filling up just before the due date). Figures 2 and 3 present the modules discussed.

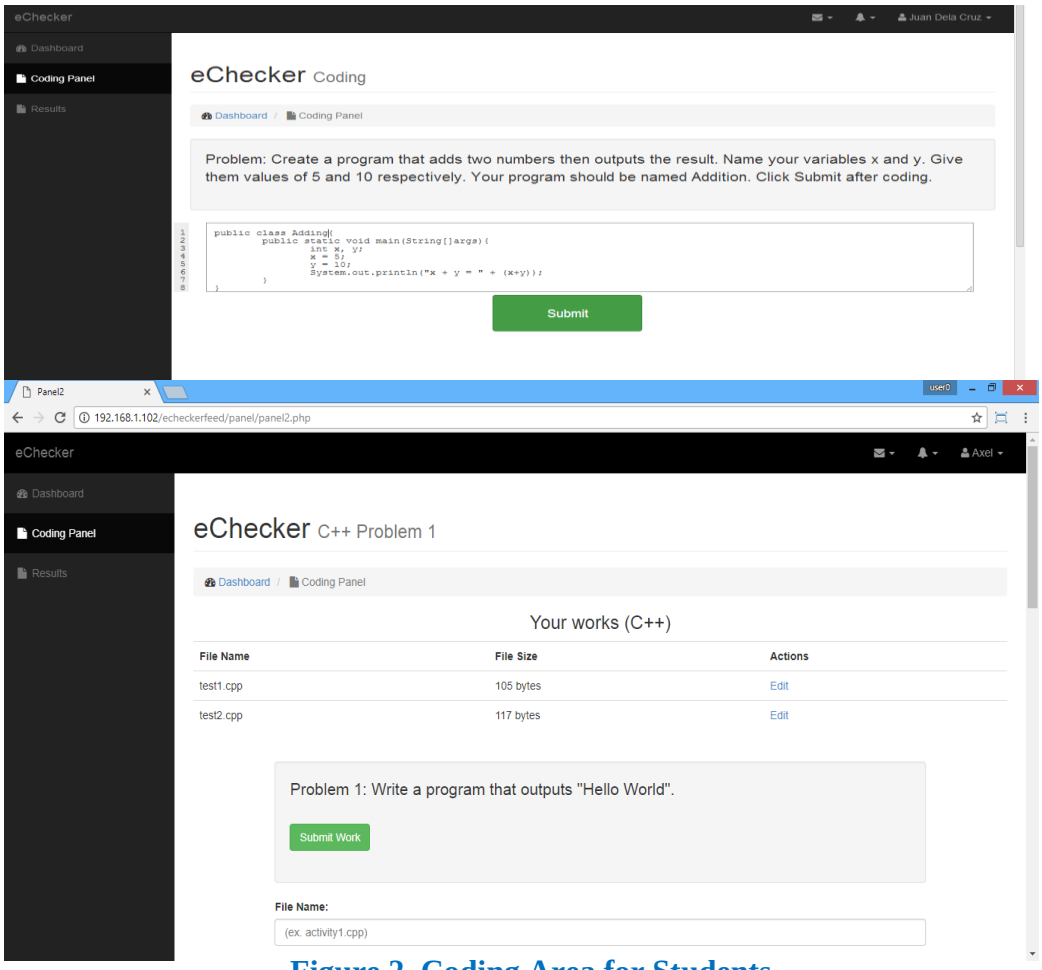

Figure 2. Coding Area for Students

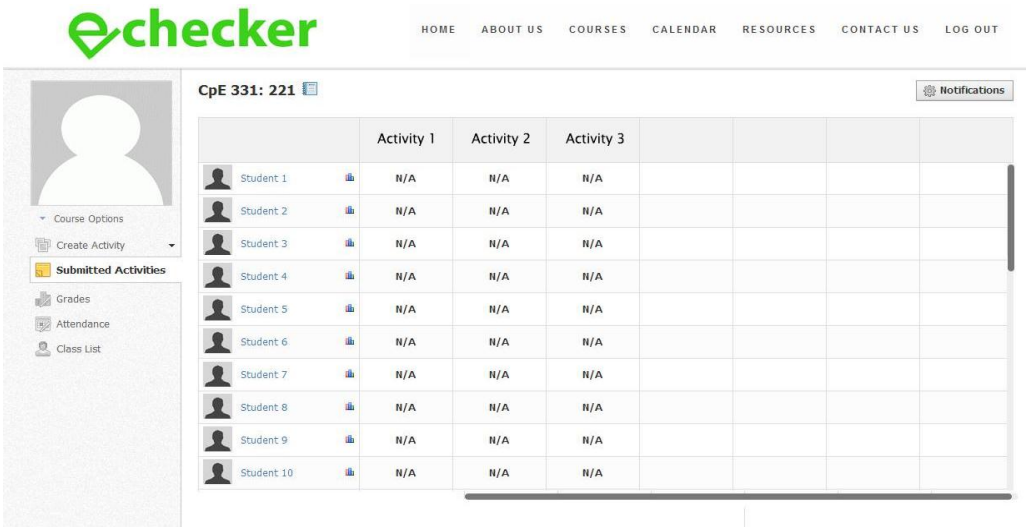

Figure 3. Repository Portal for Instructors

Automatic Assessment and Grading Module

For the criteria in grading computer programs (as gathered during the data gathering phase), the proponents incorporated the rubric in the Assessment Module. Such tool recognizes students misinterpreted instructions, missed deadlines, errors, both due to misconceptions on their part and inevitable ambiguities in the problem specification. When the student submits a program, the system compiles it and even run it against published cases from instructors. This alerts the student to any gross failures (such as having added a last-minute comment without an ending delimiter) and any unexpected discrepancies with the automated running process. Criteria and corresponding details are provided in the table 1. Area where instructors can configure the rubric for grading students' works is provided in figure 4.

Table 1. Computer Programming Grading Rubric

Criteria	**Assessment**			
	Exceptional (4 points)	Acceptable (3 points)	Amateur (2 points)	Unsatisfactory (1 point)
Correct Output (30%)	The program works and meets all of the specifications	The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	The program produces correct results but do not display them correctly.	The program is producing incorrect results.
Readability (10%)	The code is exceptionally well organized and very easy to follow.	The code is fairly easy to read.	The code is readable only by someone who knows what it is supposed to be doing.	The code is poorly organized and very difficult to read.
Application (30%)	Student shows a high-level ability to use the most efficient and logical programming techniques and processes in creating the program.	Student shows considerable ability to use the most efficient and logical programming techniques and processes in creating the program.	Student shows some ability to use the most efficient and logical programming techniques and processes in creating the program.	Student shows a limited ability to use the most efficient and logical programming techniques and processes in creating the program.
Program Execution (30%)	Program executes correctly with no syntax or runtime errors.	Program executes correctly with little or tolerable syntax or runtime errors.	Program executes correctly with many syntax or runtime errors.	Program does not execute at all.

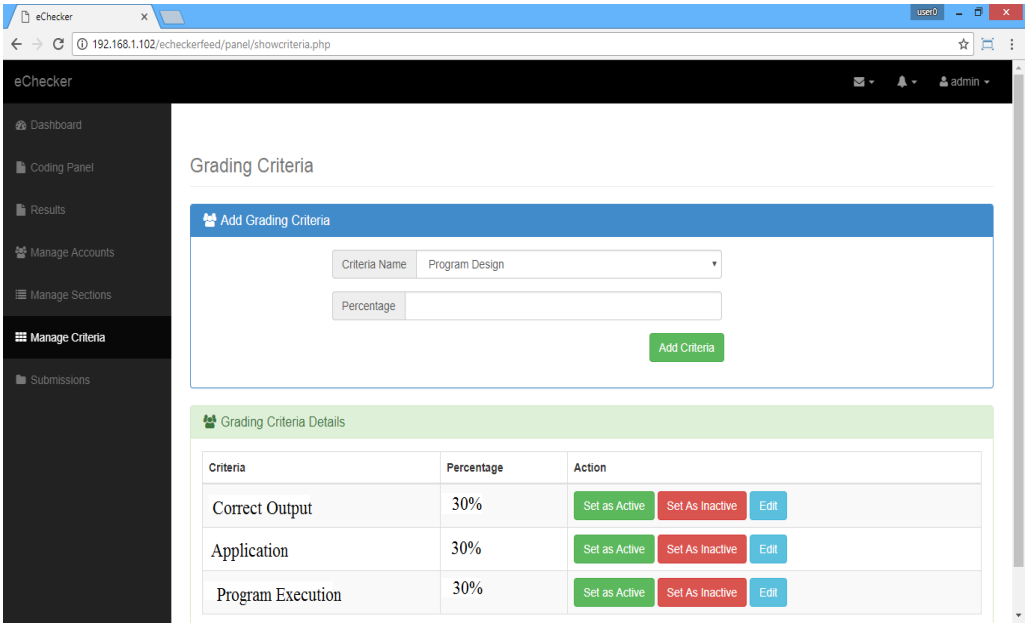

Figure 4. Module for Grading Criteria Configuration

Instructor's Dashboard

A dashboard for instructors was also included in the project. This area enables instructors to organize content (exercises, assignments) and help control the flow of the course, view and print reports for entire classes, giving an overview of the class's strengths and weaknesses, view the standards used by default for a course, with the option of editing standards pertaining to that class only. Figures 5, 6 and 7 present the different modules included in the Instructor's Dashboard.

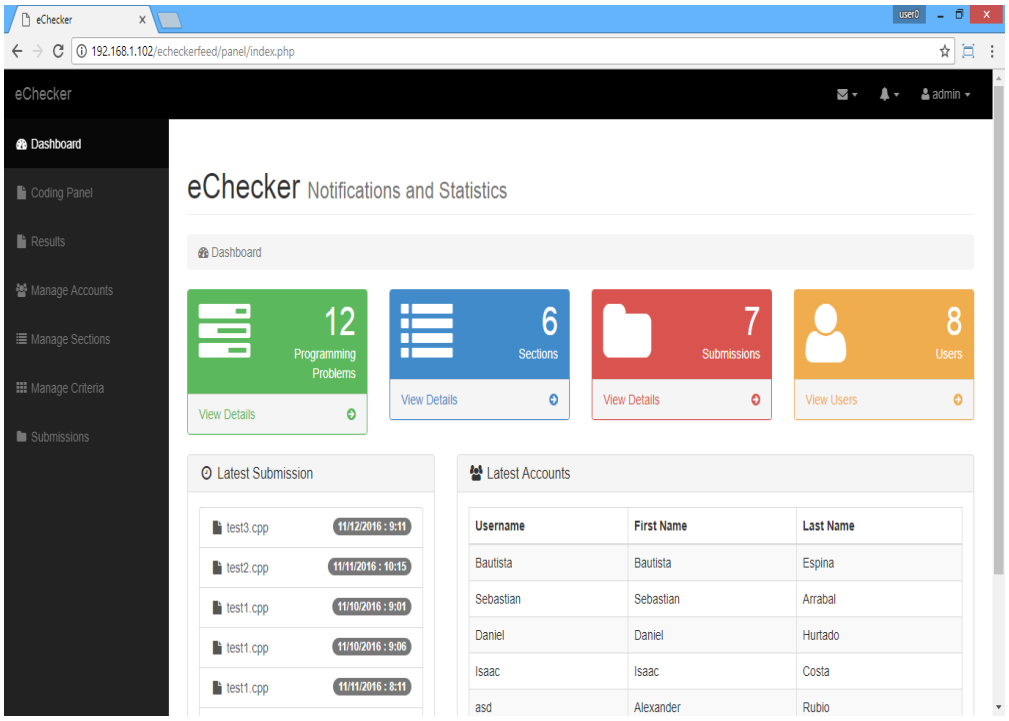

Figure 5. Notifications and Statistics (Instructor's Dashboard)

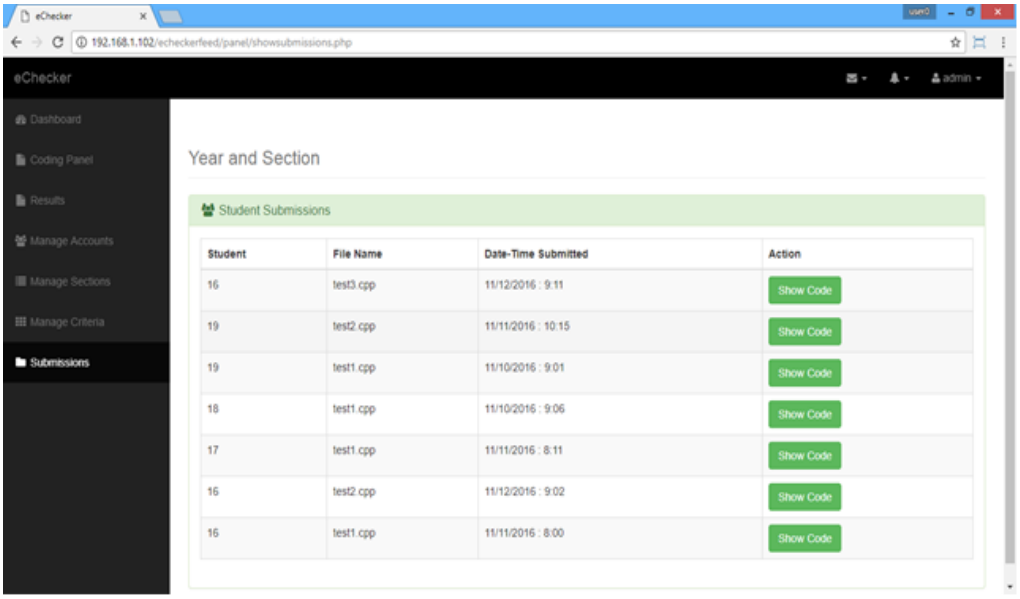

Figure 6. Manage Section Module (Instructor's Dashboard)

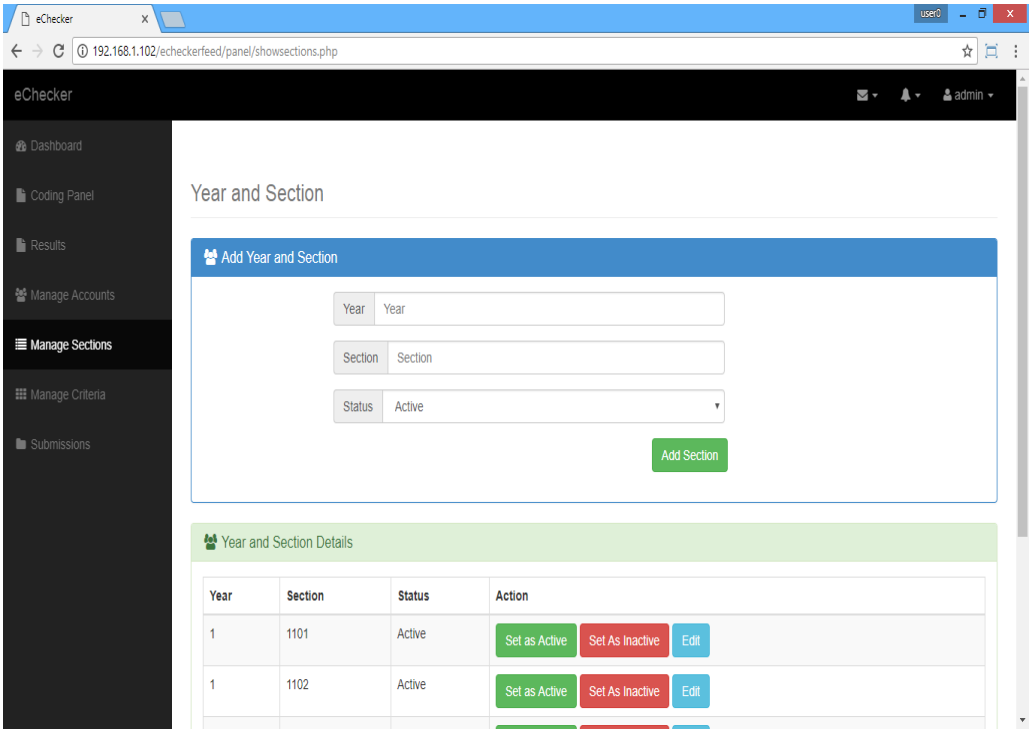

Figure 7. Student Submission Module (Instructor's Dashboard)

Conclusions

Based on the findings of the study, the proponents have drawn the conclusions about the developed system:

1. The proponents found out that static analysis method is more appropriate to use than dynamic analysis method because dynamic analysis requires the executable program to be evaluated while static analysis can evaluate the program even if it is not finished.
2. By comparing the students' output program to the instructor-provided program, the application successfully performed an automatic assessment and grading of the Object-oriented-based programming exercises.

3. Using the Connascence Flexible Software Metrics, the application was able to assess Object-Oriented based exercises.
4. The application was able to send immediate feedback and results to the student's portal since it uses static code analysis.
5. Students and faculty where able to track and monitor their progress in the course and activity using the interface provided for the users.

References

1. Arifi, S.M., Zahi, A. and Benabbou, R. 2016. Semantic similarity based evaluation for C programs through the use of symbolic execution. In Global Engineering Education Conference (EDUCON), 2016 IEEE (pp. 826-833). IEEE.

2. Crowell, B. 2003. Checking Students' Symbolic Math on a Computer. The Physics Teacher, 41(8): 478-480.

3. Xu, Z. and Zhang, J. 2006. A test data generation tool for unit testing of C programs. In: Quality Software, 2006. QSIC 2006. Sixth International Conference on IEEE, 107-116 pp.