International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2019 IJSRCSEIT | Volume 5 | Issue 2 | ISSN : 2456-3307
DOI : https://doi.org/10.32628/CSEIT11951144

Online Adaptive Assessment Platform

Dr. S. Lokesh, Suvetha. S, Swathi. M
Department of Information Technology, Hindusthan Institute of Technology, Coimbatore Tamil Nadu, India

ABSTRACT

In this paper the use of online learning, assessment and self-evaluation platform to aid in teaching and assessment of computer programming and Aptitudes in classrooms are discussed. Based on the skills of the users, the programming and aptitude concepts are taught. This paper describes the technology and implementation of the learning and assessment platform and new methods for automated assessment of programming assignments and for competitive exams. Finally, the application of the system is to help the users to learn the concept and to crack the exams easily.

Keywords : Assessment, Automated Assessment, Learning Platform, Online Learning System

I. INTRODUCTION

Teaching for many learners at a specific time is a challenging exercise. The teaching, learning and self-evaluation of computer programming and aptitude on a single platform is a challenging task. Learning the computer programming is a toxic task, it needs hands-on practicing more and more than the other subjects. Practicing programming concepts regularly helps the learners to solve computational programming concepts easily. This hands-on approach to solving problems through tinkering and actively engaging in the making of solution for solving a problem [1].

In large programming classes, there is a problem of a scarcity of computer and network facilities for students, which can place a hugely problematic constraint on assessment, self-evaluation and learning methods. Authentic and constructively aligned assessment of programming should follow the setting in which students might write their programs in the real world: on a computer with the instantaneous feedback from the programming environment, iterative submissions, and a debugging or trial-and-error approach to produce functional programs. However, many programming classes are assessed using traditional written exams despite many lecturers' discomfort with that approach programming. Written assessments also have a significant administrative disadvantage in that the process of grading these assessments is a laborious and extremely time-consuming process. The authors have noted that grading a typical written programming exam for a 500-student class can require up to 50 person-hours. This paper describes the design and implementation of an online learning and assessment platform for teaching, assessing and self- evaluating programming [2-5]. This platform allows for online completion of programming and aptitude assignments using automated assessment tools, allowing both the standard compiler/interpreter for programming feedback as well as customized contextual instructor guidance. This platform was utilized for both formative and summative assessments in an introductory

programming class of many engineering students. This application has a future scope that helps the students how to learn program and aptitude[6-8].

II. EXISTING SYSTEM

The process of learning the skills of computer programming involves understanding the basic concepts. However, learning these concepts form only a small part of the true complexity of solving programming problems. Even when solving a simple programming problems will cause emergent issues [9-12]. It is difficult to teach all the logics to solve the problems. However, it is impossible to teach all the permutations of the instructions and the algorithmic structure that helps to solve the complexity problems. Hence, the users of programming each construct their own body of knowledge and understanding that expands when the user encounters and overcomes additional problems.

In modern, internet-connected learning environment, the process of learning programming can be neatly described by constructionism automated assessment and self-evaluation, where learning occurs through creatively solving problems rather than the transmission and reception of knowledge of solving the problems. This building of knowledge is most reliably achieved when the learner is experiencing the process of constructing a meaningful product rather than reproducing rote learned series of concepts automated assessment[13-15]. To foster these processes, a teaching and learning strategy should allow a student to tackle problems in an environment where they can explore and tinker with their solutions while receiving real-time guidance. The method used to assess a user level of understanding of a topic is also an important factor to consider when designing a teaching and learning strategy. We consider two key characteristics of programming assessment: authenticity and constructive alignment. In order to be authentic, assessment of programming ability should allow the user to tinker with a solution with the benefit of feedback from the compiler or interpreter. This is a far more authentic situation than a written test, as a programmer is rarely asked to produce a working program on paper[16-20]. Further, if the natural method of learning programming is through a constructionist process of experimentation and problem solving, then the assessment should also take place in a context which allows the same type of experimentation during the assessment.

The online learning and evaluating system presented in this paper addresses the approach concerns of both the learning environment and assessment of programming. As described in the coming sections, the system emulates a development environment, allowing the users to engage with the programming tasks in a setting comparable to real-world programming environment. In addition to the built-in compiler and interpreter feedback, the system also allows the instructor to devise automated guidance to help users overcome problems. The exact same system can be used for formal assessment, ensuring that the users are assessed in a realistic manner that is aligned with their learning process.

III. STRUCTURE OF THE ONLINE LEARNING ADAPTIVE SYSTEM

We have proposed system and implementing a web application for descriptive type answers checking and its automatic assessment .Till now the systems are developed for the Automatic Evaluation of Single Sentence Descriptive Answer but by this application we are trying to provide automatic evaluation of multiple sentence descriptive answers. To increase the tolerance of the system we are going to use the Pattern Matching Technique Algorithm. We are building new system in which the descriptive examinations are also online [21-23].

The system described in this work is based on the popular Moodle open-source learning platform. This platform was developed by Dougiamas, who decided to build a free and open-source learning and assessment platform focused on constructivist pedagogical principles. Moodle is licensed under the GPL written for the GNU project this protects the rights of end users to run, study, share and modify the software. This open source principle has fostered a large and active development community surrounding the Moodle project with constant active contribution to the core code-base and many community developed plug-ins that extend the capabilities of the platform in a variety of ways. Moodle itself is a PHP-based Course Management System (CMS) that can be deployed on a variety of operating systems, web servers and database systems. The full source code is available for free and is well documented which allows for end users to tinker with the system. However, easy one-click automated installation packages have also been created by companies like Bitnami to make deployment exceedingly simple. Moodle provides all the core features one would expect from a CMS, which includes robust user management, diverse content management tools, scheduling tools, a variety of assessment tools, messaging, a grade management system, integration of plug-ins and support for learning module standards like SCORM .This open community has led to the development of many useful plug-ins for Moodle, and the system described in this paper is based on a Moodle plug-in called the Virtual Programming Lab (VPL) [24-26].

A. Virtual Programming Lab (VPL) Plug-In

The VPL plug-in is a system designed to present and assess programming assignments through the Moodle platform. The plug-in consists of three main elements. The first is the main plug-in module that runs on the Moodle server; the second is an editor component that allows for the editing of source code in the browser. Finally, there is the jail server that acts as a sandbox environment that executes the student's code. The main plug-in module of VPL, which runs on the Moodle server, manages the descriptions of the assignments, the marking scripts and protocols, the grading process, scheduling settings, access restrictions, similarity checks and controls how a student's code will be executed on the jail server. The editor is an integral part of the VPL plugin and provides a capable in-browser editor environment that supports syntax highlighting for the various supported languages and multiple file support through tabs. The editor allows students to edit the assignment source code, provides the interface to the development environment to receive feedback from compilers or interpreters and allows the student to submit assignments for automatic assessment, feedback and grading. The final element of the VPL system are the jail servers These are the servers that the VPL plug-in transmits a student's code to for execution. These servers are where the tool chains for the various supported languages are installed. As of the writing of this report, VPL can currently execute 27 languages with varying levels of support for syntax highlighting, debugging and graphical user interfaces. Executing student's code is a risky endeavour for a server as students are prone to producing bad code that can inadvertently compromise an operating system through memory leaks, infinite loops or system calls. Some students are also bound to test the limits of an execution environment and attempt to execute malicious code on the server. In this way, the VPL system can control the maximum allowed system resources that a given student program can use and protect the jail system from erroneous or malicious code during execution.

The VPL system supports using multiple jail servers for a single Moodle environment and manages the load balancing between these servers when many students are using the system simultaneously. When a student submits a program for execution it is

transmitted to the jail server along with the execution scripts created by the instructor. These scripts are then used to execute the code using the appropriate compiler or interpreter. The output of the program, compiler or interpreter is then sent back to the student with a standard command-line interface[27-29]. VPL has recently added support for graphical output from programs in addition to the command line interface. This is achieved by using the VNC remote access software that is built into most modern Linux distributions. This interface then streams a basic windows environment back to the student's browser allowing them to interact with the graphical elements of the environment they are currently engaged with. The VPL system also includes a source code similarity measurement system which is used to analyses the submissions for a given assignment and report back on their relative similarities using an easy to navigate interface. This tool makes it possible to determine which students submitted plagiarized code with a minimal time investment. For use of VPL in strict testing environments, the plug-in includes the standard Moodle activity security features that provide the ability to control access to an activity using a password and to limit access by network addresses. This allows an instructor to limit access to an activity to a specified set of computers, such as those in the lab where the assessment is being conducted. VPL can also disable the ability to copy and paste text in the editor, which ensures that a student was the author of an activity[30-32].

B. Automated Assessment using VPL

The real power of VPL as a tool for teaching and assessing programming assignments is in its automated execution and assessment capabilities. VPL runs a student's code using a BASH script to prepare the source files, compile the code (or send it to the relevant interpreter) and then execute the code. The standard output stream of this process is then provided to the student via a console window in their browser where a student can interact with the running program. When a program is being assessed, a different BASH script is used to compile This assessment script will then provide automated input to the program and evaluate its behaviour for grading purposes. Out of the box VPL includes default run, evaluation and debugging scripts for all the supported languages. A standard format for defining basic evaluation test cases is also provided.

Using the built-in evaluation scripts involves defining a series of test cases where the input provided to the program is defined along with the expected output that the program should print to the standard output stream. Grading conditions are provided for each test case. VPL includes a C++ grading program which takes in the specified test cases and uses them to evaluate a student's submission and automatically grade the submission. While this capability provides an easy way to quickly produce automatically assessed programming assignments it is very limited. Input can only be provided via the standard input stream and the output is naively assessed as a single output numeric or direct string comparison. This means that for a student to be graded as correct their program's output must conform exactly to the specified solution output with no extra spaces or new-line characters. This can often lead to confusion: defining the exact form of the required output can be challenging, and students will often not understand the formatting issues created by spaces and new-line characters which are difficult to spot in a text console. Unfortunately, the execution script side of VPL is not very well documented and the only real source of information provided comes in the form of the default run and evaluation BASH scripts provided with the VPL source code. The first thing the custom evaluation does is load the file which contains the flags produced by the validation process. If any of the three validation conditions have been detected the appropriate message is given to the student and they receive a zero grade. Next, the script will run the

student's submission. In order to confirm that the random data set is indeed random, another phase of validation is performed. The data set is recorded to file. The student's submission is run a second time and the new data set is compared to first data set. If the data set is truly random these arrays' will be different. If they are not, then it indicates that the student has manipulated the contents of the dataset. They receive an appropriate error message and grade. The script then calculates the model sorting signature for the random data set. If the student's submission matches this sorting signature, then they receive full marks. If they sorted the algorithm using a different algorithm but the array is sorted correctly, they will receive half credit. Otherwise they have not met the question specifications and will receive a zero grade.

IV. APPLICATION OF LEARNING SYSTEM TO GOALS

This section will discuss how the learning system supports the pedagogical philosophy described in Section II. The first major benefit of the system is that it lives in the cloud and is accessible from any decently sized device that can run a modern web browser. This means that students have access to a code editor and the relevant tool chains from any internet connected computer without needing to install any specialized software. This has greatly increased the access to these technologies, especially for students who have limited resources at their disposal. The system was built to support a constructionist pedagogical philosophy and does so far better than traditional assessment of programming. The automated assessment tools can provide real-time feedback from the compiler/interpreter and built-in clues and guidance from the instructor, meaning that students can work at their own pace to solve programming problems but are given appropriate feedback every step of the way. An instructor can write scripts that detect common mistakes and provide clues on how to address these common errors to the student. In this way, the student can work at their own pace and work towards a solution on their own terms and when they have solved a problem the system will inform them of the fact. This is important because often students are not able to identify when a problem is adequately solved and by having the system automatically inform them they get the satisfaction of having solved a problem on their own without having to be told days later that they were successful through a manually graded assignment. The system also provides an appealing alternative to traditional partial credit grading. Paper-based assessments are inherently submitted once, and the submitted work must be evaluated as is. With the system presented here, students can be expected to correct small mistakes based on the compiler or instructor feedback and produce (and test) a working program. This expectation of a working solution fosters learning to develop an understanding rather than incentivizing memorization to earn partial credit. Note that the system does not preclude partial credit: an instructor can build assessments that assess levels of completion of a programming task, or allow for common mistakes (as in the sort example). This flexibility caters for all levels of student performance but still ensures that each student is producing at least basic complete programs. The automation also means that it becomes possible to very quickly and consistently assess the level of capability that a student has demonstrated through their performance. If deemed appropriate, grading schemes can also be adapted to include number of incorrect submissions or compilation attempts, code efficiency, submission time, and more; the combination of the course management system and scripting environment is remarkably flexible. The system also provides a trove of data about student participation and progress, keeping track of almost every action a student takes. This data can definitively answer questions about student participation and progress which might traditionally rely on self-reporting from students.

V. CONCLUSION

This article describes the implementation of an online learning assessment system that enables the automated teaching and assessment of computer programming tasks for large classes. The pedagogical philosophy employed by the authors in teaching programming is described and how the implemented system addresses the goals of this philosophy is discussed. The learning system is implemented using the open source learning platform Moodle and an open source plug-in, VPL, which supports the execution and assessment of a large variety of programming languages. The author describes the VPL system and how the scripting engine used by VPL can be used to build flexible and robust automatically assessed programming activities. A number of new techniques for building these evaluation scripts are presented in detail, allowing future efforts to reproduce and expand upon this work. The benefits of the system for use in a large class are also discussed. Firstly, the system increases the amount of real time feedback the students receive which is significantly higher than in the traditional model where a single lecturer and a handful of tutors can only provide limited attention to the large number of students in a class. The students can learn and experiment with the work at their own pace and in their own way. The amount of administration work is significantly reduced for the lecturer and content produced can be reused in many ways. Finally, the system provides a trove of data that can be used to ask interesting questions about the class to aid in self-reflection and course execution. The learner‘s descriptive answer and standard answer is converted into its graphical form and then, to apply some of the similarity measures such as string match, word Net and spreading process for the calculation of similarity score are the major steps in the proposed algorithm. The algorithm provides a solution for the automation of descriptive answer evaluation process. Automatic evaluation of single sentence descriptive answer would be beneficial for the universities, schools and colleges for academic purpose by providing ease to faculties and the examination evaluation cell.

VI. FUTURE SCOPE

In the future, we are going to introduce advanced code analysis mechanism that can inspect students' code according to software quality metrics. We encourage that students can write code with good qualities in addition to writing code that meets assignment requirements. Moreover, we also want to develop a team project feature that allows instructors to form teams and assign team projects.

VII. REFERENCES

[1]. B. S. Bloom, M.D. Engelhart, E.J. Furst, W.H. Hill, D.R. Krathwohl, Taxonomy of education objectives: The classification of educational goals. Handbook I: Cognitive domain. New York: David McKay Company, 1956.

[2]. S. Papert, “Constructionism: A New Opportunity for Elementary Science Education,” Massachusetts Institute of Technology, Media Laboratory, Epistemology and Learning Group: National Science Foundation, Award 8751190, 1986.

[3]. J. Sheard, Simon, A. Carbone, D. D'Souza, and M. Hamilton. “Assessment of programming: pedagogical foundations of exams,” in Proc. of the 18th ACM conference on Innovation and Technology in Computer Science Education (ITiCSE '13), 2013, pp. 141-146.

[4]. P. Ihantola, T. Ahoniemi, V. Karavirta, and O. Seppälä. “Review of recent systems for automatic assessment of programming assignments,” in Proc. of the 10th Koli Calling International Conference on Computing

Education Research (Koli Calling '10), 2010, pp. 86-93.

[5]. E. Costello, "Opening up to open source: looking at how Moodle was adopted in higher education," in Open Learning: The Journal of Open, Distance and e-Learning, Vol. 28, Issue 3, 2013.

[6]. (2016) The GNU Licenses Website. Available: http://www.gnu.org/licenses/licenses.en.html

[7]. (2016) Bitnami Website. OnlineAvailable: https://bitnami.com/

[8]. (2016) SCORM Website. OnlineAvailable: http://www.adlnet.gov/adl-research/scorm/

[9]. J.C. Rodríguez-del-Pino, R-R. Enrique, and H-F. Zenón, "A Virtual Programming Lab for Moodle with automatic assessment and antiplagiarism features," in Proceedings of the 2012 International Conference on e-Learning, e-Business, Enterprise Information Systems, & e-Government, 2012.

[10]. (2016) VPL Plug-In website. OnlineAvailable: http://vpl.dis.ulpgc.es/

[11]. D. Thiébaut, "Automatic evaluation of computer programs using Moodle's virtual programming lab (VPL) plug-in," Journal ofComputing Sciences in Colleges, Vol. 20, Issue 6, 2015.

[12]. J.B. Biggs, C. Tang, Teaching for quality learning at university: What the student does. McGraw-Hill Education (UK), 2011.

[13]. T. Staubitz, H. Klement, J. Renz, R. Teusner and C. Meinel, "Towards practical programming exercises and automated assessment in Massive Open Online Courses," Teaching, Assessment, and Learning for Engineering (TALE), 2015 IEEE International Conference on, Zhuhai, 2015, pp. 23-30.

[14]. Kumar R, Lokesh S & Ramya Devi, M. (2018), Identifying Camouflaging Adversary in MANET Using Cognitive Agents, Wireless Personal Communication, https://doi.org/10.1007/s11277-018-5378-1.

[15]. S. Lokesh, S. Malathy, K. Murugan and, G. Sudhasadasivam (2010), Adaptive Slot Allocation and Bandwidth Sharing for Prioritized Handoff Calls in Mobile Networks, International Journal of Computer Science and Information Security, Vol.8 , 52-57.

[16]. S.Lokesh and G.Balakrishnan, "Robust Speech Feature Prediction Using Mel-LPC to Improve Recognition Accuracy", Information Technology Journal, vol. 11, no.11, pp. 1644-1699, 2012.

[17]. Lokesh, S., Malarvizhi Kumar, P., Ramya Devi, M. et al. An Automatic Tamil Speech Recognition system by using Bidirectional Recurrent Neural Network with Self-Organizing Map Neural Comput & Applic (2018). https://doi.org/10.1007/s00521-018-3466-5

[18]. Lokesh, S. & Devi, M.R. Speech recognition system using enhanced mel frequency cepstral coefficient with windowing and framing method Cluster Comput (2017). https://doi.org/10.1007/s10586-017-1447-6

[19]. Kanisha, B., Lokesh, S., Kumar, P.M. et al. Speech recognition with improved support vector machine using dual classifiers and cross fitness validation Pers Ubiquit Comput (2018). https://doi.org/10.1007/s00779-018-1139-0

[20]. S.Lokesh and G.Balakrishnan, "Speech Enhancement using Mel-LPC Cepstrum and Vector Quantization for ASR", European Journal of Scientific Research, vol.73,No.2, pp. 202-209, 2012.

[21]. Selvaraj, L., and Ganesan, B. (2014) Enhancing speech recognition using improved particle swarm optimization based Hidden Markov Model. Scientific World J. DOI: 10.1155/2014/270576.

[22]. S. Lokesh, G. Balakrishnan, S. Malathy, and K. Murugan, "Computer Interaction to human through photorealistic facial model for inter-process communication", in International

Conference on Computing Communication and Networking Technologies (ICCCNT), 2010, pp. 1-7.

[23]. Priyan Malarvizhi Kumar, S. Lokesh, R. Varatharajan, Gokulnath Chandra Babu, P. Parthasarathy, Cloud and IoT based disease prediction and diagnosis system for healthcare using Fuzzy neural classifier, Future Generation Computer Systems,2018, https://doi.org/10.1016/j.future.2018.04.036.

[24]. Lokesh, S., Kanisha, B., Nalini, S. et al. "Speech to speech interaction system using Multimedia Tools and Partially Observable Markov Decision Process for visually impaired students", Multimed Tools Appl (2018). PP.1-20, https://doi.org/10.1007/s11042-018-6264-2

[25]. Kumar, R., & Lokesh, S. (2015). "Fast and secure transmission of information among groups using a key management scheme". International Journal of Computer Science and Mobile Computing, 4(11), 40–47.

[26]. B. Manikandan, T. Senthilkumar, Dr. S. Lokesh, M. Ramya Devi, " Opportunities and Challenges in Airborne Internet with Fly-In, Fly-Out Infrastructure", International Journal of Scientific Research in Science, Engineering and Technology(IJSRSET), Print ISSN : 2395-1990, Online ISSN : 2394-4099, Volume 4 Issue 4, pp.1464-1469, March-April 2018.

[27]. T. Senthilkumar, B. Manikandan, M. Ramya Devi, S. Lokesh, "Technologies Enduring in Internet of Medical Things (IoMT) for Smart Healthcare System", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 3 Issue 5, pp. 566-572, May-June 2018.

[28]. M. Ramya Devi, S. Lokesh, B. Manikandan, T.Senthilkumar, "Vehicular Cloud Computing Based Intelligent Transportation System for Traffic Management and Road Safety", International Journal of Computer Sciences and Engineering, Vol.6, Issue.7, pp.970-975, 2018.

[29]. Krishnan, S., Lokesh, S. & Ramya Devi, M. "Internet of things for knowledge administrations by wearable gadgets", Journal Medical System (2018) 42: 230. https://doi.org/10.1007/s10916-018-1081-8

[30]. Vijayarangam, S., Megalai, J., Krishnan, S. et al. "Vehicular Cloud for Smart Driving Using Internet of Things " Journal Medical System (2018) 42: 240. https://doi.org/10.1007/s10916-018-1105-4

[31]. Ramya Devi M, Krishnan S, Lokesh S. An optimal Internet of Things–based smart cities using vehicular cloud for smart driving. Concurrency Computat Pract Exper. 2018; e5037. https://doi.org/10.1002/cpe.5037

[32]. Sivakumar Krishnan, S. Lokesh, M. Ramya Devi, "An efficient Elman neural network classifier with cloud supported internet of things structure for health monitoring system", Computer Networks, Volume 151, 2019, Pages 201-210.
https://doi.org/10.1016/j.comnet.2019.01.034

Cite this article as :

Dr. S. Lokesh, Suvetha. S, Swathi. M, "Online Adaptive Assessment Platform", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 5 Issue 2, pp. 21-28, March-April 2019. Available at doi : https://doi.org/10.32628/CSEIT11951144
Journal URL : http://ijsrcseit.com/CSEIT11951144