BUCTOL: VS Code Based Integrated Online Programming and Plagiarism Monitoring Platform

Yong Liu Zhe Li Yonghao Wu* Bin Du* Heng Jiang
College of Information Science and Technology
Beijing University of Chemical Technology
Beijing, China
appmlk@outlook.com, dubin@buct.edu.cn

***Abstract*—Detecting plagiarism in programming education is a non-trivial academic task. With the number of students being evaluated increasing and the level of difficulty in the test questions also increasing, the reliance on manual checks to verify if plagiarizing has occurred is becoming an inefficient strategy. To assist teachers in identifying cases where plagiarizing has occurred, we have developed a tool with built-in plagiarism monitoring capabilities called Beijing University of Chemical Technology Online Integrated Programming Platform (BUCTOL). BUCTOL was developed based on the open-source system HustOJ and the source code editor Visual Studio Code (VS Code), which provides a fast and accurate approach to aid in the detection of plagiarizing in computer programming-related assignments, exams, and competitions. The platform collects a variety of data while students complete assignments, exams, and competitions, which facilitates monitoring student activity to aid with the identification of plagiarizing. To evaluate BUCTOL's capacity to prevent and monitor plagiarizing, we compare the code repetition rate before and after the use of BUCTOL and find that the code repetition rate was significantly reduced after the adoption of BUCTOL. Additionally, we sent questionnaires to 92 students who utilized the platform. The results indicated that students were generally positive about BUCTOL's ability to prevent and monitor plagiarizing.**

Keywords—Programming education, Plagiarism monitoring, Online judge, Data collection

I. INTRODUCTION

The traditional method for teaching computer programming in a classroom is relatively simple. Students primarily learn by listening to theoretical lectures or watching a practical demonstration. However, these learning approaches are considered passive and often lack significant student participation. To boost students' practical competence, programming assignments have become integral to these assessments with submissions often sent via email [1].

However, this method introduces a slew of additional issues. For example, email can only be used to receive daily assignments, not to organize a multi-person live programming environment, such as a final exam. Additionally, this manner of code submission can easily breed plagiarizing [2], [3]. For instance, a survey of 287 students at Monash and Swinburne universities discovered that more than 70% of students at both universities admitted to plagiarizing[4]. Besides, teachers at Beijing University of Chemical Technology (BUCT) had reported instances when the exact same code was received in emails from many students.

Subsequently, as the number of students increases, email as a submission medium has ceased to meet the current demands of classroom education and assessment. This increase also impacts the workload associated with assessment leading to research into more effective uses of computing resources to evaluate students' programming abilities reliably and rapidly [5]–[8].

Then, Online Judge (OJ) system stands out due to its straightforward logic and rapid feedback capabilities and has been adopted by many universities [9]. OJ is now widely used to facilitate students in the development of programming skills, in training and selection of contestants, in the provision of automated program submissions, and in the judgment of programming courses [9]. The rapid feedback capability of the OJ platform enables live online exams, while relieving teachers of the tedium of manually marking submissions, and allowing assessments to scale to larger numbers of students.

Additionally, previous research has proposed several methods for determining whether students' submitted code has been plagiarired [10], [11]. In OJ, the most frequently used methods are code text matching and AST (Abstract Syntax Tree) matching. Specifically, when a student submits code, the OJ system compares the text and AST of the code to all previously accepted correct answers for the same question. As a result, a higher similarity result indicates that the corresponding submission is more likely to be plagiarized. To some extent, the OJ platform eliminates the naive plagiarizing practice of students duplicating whole code because the OJ platform can immediately notify the teacher when a complete duplicate is discovered.

However, students still manage to find ways to plagiarize within the OJ system. Specifically, after copying the correct answer from elsewhere, students can evade the OJ system's plagiarism detection feature by inserting noisy code such as nonsensical functions, nonsensical variables, or comments into the correct answer. Although numerous methods of

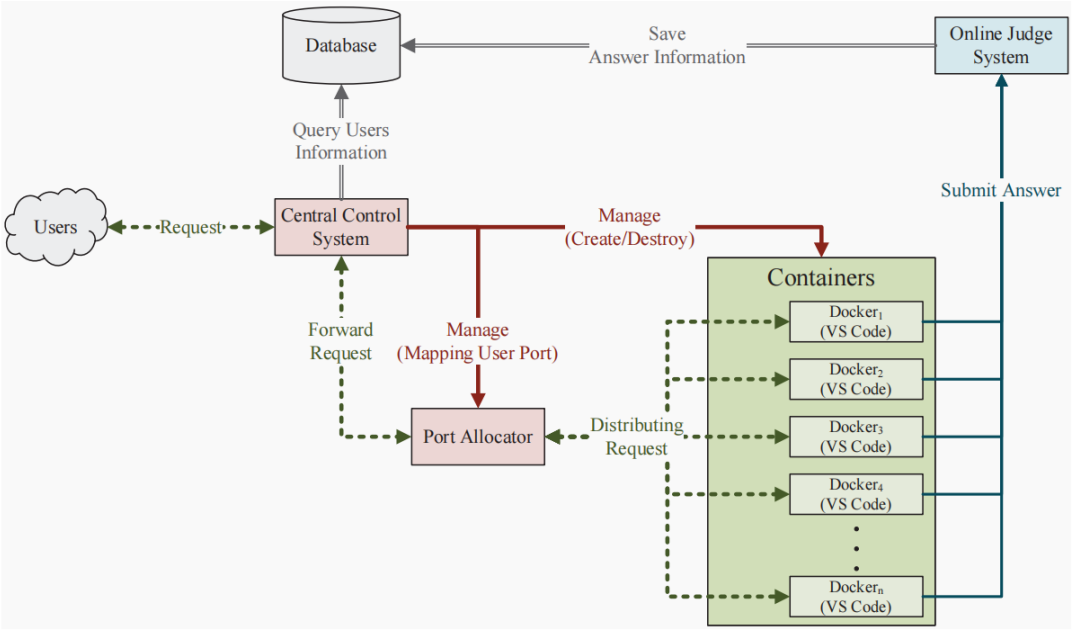

Fig. 1. BUCTOL Framework

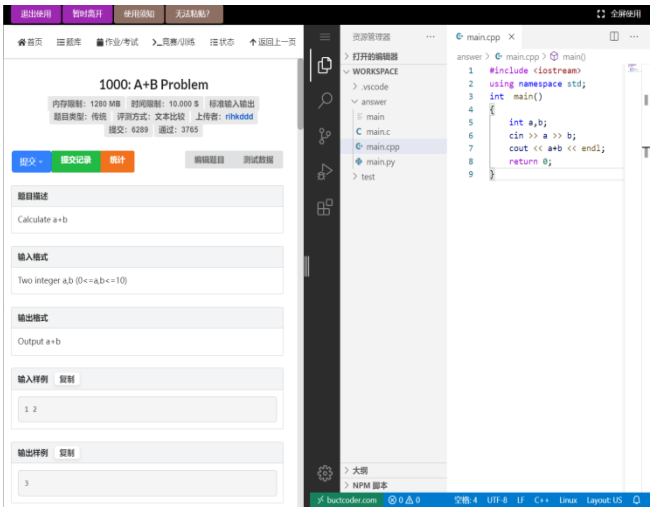

Fig. 2. Snapshot of BUCTOL

plagiarism detection have been proposed [10] [11]. Teachers are afraid that accidental faults may lead to wrongful accusations against non-plagiarized students, thus causing conflicts between teachers and students. Given the increasing volume of code submissions, it is impractical to require teachers to manually check for plagiarized code [12].

To address this issue, we developed the **B**eijing **U**niversity of **C**hemical **T**echnology **O**nline Integrated Programming **P**latform (BUCTOL) , which is based on the Browser/Server (B/S) framework. BUCTOL enables students to complete assignments, exams, and contests entirely within the same webpage without navigating to other webpages or programs. Additionally, BUCTOL includes a robust set of built-in plagiarism detection capabilities that enable teachers to observe or replay each student's code writing process in real-time. Once students have finished an assignment or exam, this real-time review helps identify, accurately and efficiently, if they are plagiarizing code.

Table 1 EXAMPLE OF CODE EDITING PROCES

Step	Code
1	#
2	#include (studio. h
3	#include (studio. h)
4	#include (studio.)
5	#include (studio.)
6	#include (studio.h)
7	#include studio.h int main()
8	#include studio.h int main() (}
9	#include studio.h int main() (}
10	#include <>studio.h int main() (}
11	#include <studio.h> int main() (int n}
12	#include <studio.h> int main() (int n scanf}
13	#include <studio.h> int main() (int n scanf:}

Next, we compared student behavior prior to using BUCTOL and following the use of BUCTOL. The study found that after students were required to use BUCTOL for classroom exams, the code repetition rate decreased significantly. Additionally, we evaluated BUCTOL's ability to influence student behavior with respect to plagiarism using a student-oriented questionnaire. These results indicate that BUCTOL can help ensure exam fairness and lower the probability of plagiarizing.

The contributions of our study can be summarized as follows:

- We developed BUCTOL on a B/S structure to enable online programming and support online code by students, and provide real-time feedback on results. BUCTOL allows real-time monitoring of student data for more effective online supervision and invigilation, thus easing the workload of teachers.
- The plagiarism detection function we built in BUCTOL supports real-time recording and playback of students' code editing behavior data, which improves the efficiency of teachers' judgment of plagiarism.
- The analysis of student questionnaires revealed that stu- dents generally believe that BUCTOL can help reduce plagiarism, thus promoting fairness in examinations and further urging students to improve their programming skills.

The rest of the paper is organized as follows: Section II introduces the organizational structure, technical framework and development mode of BUCTOL, and describes the complete use process of students. Section III introduces three types of student behavior data collected by BUCTOL, and describes how these data reflect whether students have plagiarism. Section IVillustrates the effectiveness of BUCTOL in maintaining the fairness of the examination process from two perspectives: code repetition rate comparison and a questionnaire. Section V introduces some existing limitations and threats of BUCTOL. Section VI introduces the work related to BUCTOL. Section VII summarizes the functions of BUCTOL and describes the future expansion and optimization of BUCTOL.

II. Framework and Components

BUCTOL was developed based on the open-source system HustOJ [13] and the widely used source code editor Visual Studio Code (VS Code). The overall framework of BUCTOL is shown in Fig. 1.

BUCTOL stores user-related information, test question information, and answer information in a MySQL database. Among them, the central control system built on the SpringBoot framework implements user login verification, application acceptance, VS Code instance creation and distribution, and instance destruction upon user exit. Nginx is being used by the BUCTOL system to classify users and distribute their requests to distinct VS Code instances based on the cookies bound during instance creation. Users may modify, build, and debug code in distinct, non-interfering instances of VS Code, as well as submit answers. The provided answers are stored in the MySQL database, where they await execution and feedback from the HustOJ system. After that, the results will be instantly relayed to users and displayed on their webpage. Additionally, VS Code instances run inside Docker containers, which prevents user code from interfering with the physical server, hence maintaining the system's security.

Fig.2 shows the main user interface of BUCTOL, where the left side of the webpage shows the topic detail interface provided by HustOJ, and the right side shows the VS Code interface. Students can browse the webpage on the left side and edit the code on the right side. After editing the code, students can click the "Submit" button in the webpage to submit their code for automated assessment.

III. Plagiarism Monitoring

BUCTOL includes various data collecting functions that enable the platform to record a series of student operations. By monitoring the collected data, teachers can accurately and efficiently determine whether students are plagiarizing while completing assignments or exams. It is worth noting that BUCTOL only collects data related to users' actions within the BUCTOL environment and does not collect data from other webpages or programs on the user's system, ensuring that the data collecting function does not breach users' privacy.

BUCTOL collects data in three parts. The following sections will be described in detail.

A. Accessing Test Question Data

The test question information collected from the OJ page, including:

- The time when a student starts to participate in a contest;
- The time when a student finishes participating in a contest;
- The time when a student starts to try to answer a question;
- The time when a student passes a question.

The data above can be used to determine the average time students spend on each question. If a student's answer time is significantly shorter in comparison to the average time, then that student is more likely to have plagiarized.

B. Editing behavior data

This is data collected from the VS Code environment relating to the student's actions during the completion of their solution, including:

- A complete history of the students' code editing. Table I shows an example of a student's code editing process recorded by the BUCTOL platform, which presents a process of writing C++ header files.
- Details of the students' copy and paste activity in BUCTOL;
- Details of the students' program debugging activity in BUCTOL.

The above data completely captures each step in the process of student's development of their code solutions. This data can then be used to identify potential plagiarism behaviors. For instance, if a student adds meaningless functions or variables to a correctly written code statement, the student may be concealing plagiarism. Moreover, if a student pastes a large amount of code from outside of BUCTOL platform during exams, the student is more likely to plagiarize.

C. Unrelated operation data

This relates to data captured on actions performed by students during the answer process that is not related to the completion of the assignment or exam, including:

- The number of times that a student minimized the BUCTOL webpage;
- The number of times that a student moved the mouse focus out of the BUCTOL webpage.

The above data is crucial for the invigilator during the exam because the teacher cannot closely monitor every computer during an exam. Some students may attempt to access software or websites outside of the BUCTOL environment to search for answers, when not being directly observed by the invigilator, in breach of the examination regulations. Therefore, the unre- lated operation data can help identify student activity which indicates an increased risk that a student has plagiarized.

In summary, the above data can help teachers monitor student behavior and which can ultimately reduce the level of students' plagiarizing on assignments and exams. During exams, teachers can also observe the data in real-time which can facilitate more effective on-site supervision and invigilation. For completed assignments, exams or competitions, teachers can review aggregated data to help them to more efficiently and accurately identify students who may have plagiarized.

Table 2 RECORDED DATA FOR ONE EXAM

Student ID	Minimize events	Length of time away	Debugging events	Copy/Paste events
A	1	1min	34	17/7
B	25	23mins	32	2/7

For instance, Table II summarizes two students' (A and B) examination performances. Taking student A as an example, the student minimized the BUCTOL webpage just once throughout the test, and the total time the mouse was off the BUCTOL webpage was only one minute. Additionally, Student A copied code 17 times in BUCTOL and pasted seven times, indicating that the pasted text was more likely copied inside the BUCTOL system. However, for student B, who exited the BUCTOL webpage 25 times throughout the exam and the mouse left the webpage for 23 minutes, this data shows that the student has a higher probability of plagiarizing. Additionally, student B copied twice in BUCTOL and pasted seven times, indicating that the pasted text was more likely to be copied from an external source. Therefore, Student B will be required to take an additional exam to ensure fairness after this exam.

IV. Evaluation

In this section, we will illustrate the effectiveness of BUCTOL in maintaining the fairness of the examination process from two perspectives: code repetition rate comparison and a questionnaire.

A. Code Repetition Rate Comparison

We select two exams from within the same class group. In one of the exams, students were required to use BUCTOL to answer questions, while the other exam did not require the use of BUCTOL. We then compared the rate of plagiarism between the two exams.

Specifically, we developed an AST-based code repetition rate calculation tool. When a student submits code, this tool can compare the code with all the correct answers submitted for the same question. It will then show a copy of an existing answer that best matches the submitted code and indicate the level of code duplication. The duplication rate between two answers is calculated by the similarity of two AST strings generated from the the corresponding code.

Note that because the code in OJ is generally short, even if a submission is not plagiarized at all, it may have a high degree of similarity to other correct answers. Therefore, only very high similarity (e.g., over 80% repetition rate) will be used as a basis for determining plagiarism.

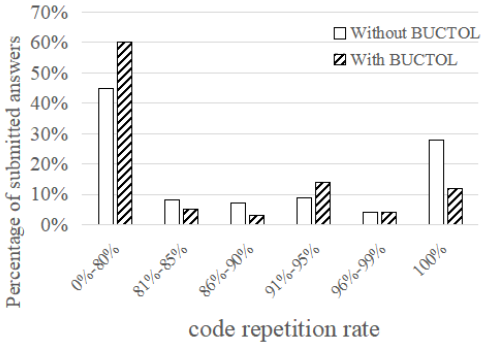

Fig. 3. Comparison Results

Fig.3 shows the comparison results of code repetition rate. Where the x-axis of Fig.3 indicates the different code repetition rates, and the y-axis indicates the proportion of codes with the corresponding repetition rates among all code submissions in a single examination.

As shown in Fig.3, at 0%-80% repetition rates, more answers come from the exam with BUCTOL. While at 100% repetition rates, more answers come from the exam without BUCTOL. This result indicates that students' submitted codes generally have lower repetition rates in the exam with BUCTOL, while submitted codes generally have higher repetition rates without BUCTOL.

In the exam that does not require the use of BUCTOL, students generally wrote code in their local programming environment and then copied the solution to the OJ system for submission. We counted all correct answer submissions in this exam and observed that 29% of the submissions

Table 3 Results of Student Questionnaires

No.	Question	Feedback				
		1 (Completely Disagree)	2	3	4	5 (Completely Agree)
1	Plagiarizing is a personal freedom, others have no right to interfere	44.60%	21.70%	27.20%	1.10%	5.40%
2	It is OK to allow others to plagiarise in regular assignments and exams	21.70%	27.20%	35.90%	10.90%	4.30%
3	Plagiarize monitoring should be built into the BUCTOL platform	7.60%	14.10%	39.10%	23.90%	15.20%
4	Grading of assignments and exam grades will be fairer with the plagiarism monitoring feature built into the platform	13.00%	7.60%	30.40%	21.70%	27.20%
5	BUCTOL's strict plagiarize detection feature violate users privacy	20.70%	27.20%	38.00%	8.70%	5.40%
6	It doesn't matter if you fail a course or even fail to graduate, you don't care about it	96.70%	2.20%	1.10%	0.00%	0.00%
7	You are satisfied with your current programming skills	19.60%	31.50%	33.70%	13.00%	2.20%
8	You are willing to spend a lot of time on programming exercises	2.20%	3.30%	18.50%	39.10%	37.00%

overlapped exactly when comparing students' correct answers. This demonstrated that when BUCTOL was not used, students exhibited significant levels of plagiarism and plagiarizing behavior.

However, after using BUCTOL, the percentage of submissions that overlapped was effectively reduced from 29% to 14%. Therefore, the plagiarism rate of code submission by students is greatly reduced by using BUCTOL. This also shows that BUCTOL helps discourage students from plagiarism and plagiarizing behaviors and reduces the overall level of plagiarizing in assessments.

B. *Questionnaire*

To investigate the reasons leading to the comparison results in Section IV-A, we designed a questionnaire to assess student attitudes toward the plagiarism detection system in BUCTOL. The questionnaire was sent to a total of 92 students in two programming-related courses at Beijing University of Chemical Technology all of whom had used BUCTOL for class exams. Students were explicitly informed of the details of the plagiarism detection functions within the system before using BUCTOL.

The results and content of the questionnaire are shown in Table III. The second column of Table III lists the eight statements against which students were asked to give feedback. Student attitudes towards the eight questions were determined by the agreement level on a scale of 1 to 5 where 1 indicating complete disagreement and 5 indicating complete agreement. The third to seventh columns of Table III show the results of the questionnaire. For example, 44.60% of students completely disagree that plagiarizing is personal freedom and others have no right to interfere. Based on the results in Table III, we can draw the following three conclusions:

(1) Based on the results of questions 1-2, most students desire examinations and assessments to be fair and free of plagiarism.

(2) Based on the results of questions 3-5, the students believe that the monitoring functions within BUCTOL promote fairness in exams and facilitate a reduction in plagiarizing. The majority of students also believe that the data collection and monitoring functions of BUCTOL do not violate student privacy.

(3) Based on the results of questions 6-8, it is clear that most students are not satisfied with their current programming skills, but are willing to spend time on programming practice to improve their programming ability.

V. Limitations and Threats to Validity

In this section, we discuss the potential limitations and threats to the validity of BUCTOL.

The first potential limitation relates to server performance which could potentially constrain the complexity and of user programs or the volume of users concurrently supported. However, because OJ code can be executed within a matter of seconds, no significant performance issues have been observed. Specifically, the platform has supported hundreds of live examinations conducted concurrently, with little additional latency encountered.

Secondly, the data monitoring function may be too sensitive, creating false positive indicators of plagiarizing behavior and thus not accurately reflecting the actual level of student plagiarizing. However, an evaluation of an individual student will be based on a combination of monitoring data across multiple assignments and exams. While the tool provides input on the risk of plagiarizing for an assignment, we will not rashly make a judgment of plagiarizing behavior. Instead, it can help focus on where the manual review can be most useful employed where a high level of risk is identified.

VI. Related Work

Currently, a number of studies have been conducted in the field of auxiliary programming education to aid teachers in the delivery and assessment of computer programming. For

example, Codecademy is a dedicated learning resource platform dedicated to teaching many different coding and programming skills. It offers a comprehensive set of text-based courses on web development and related programming languages. Students can pick which language they want to learn and advance through lessons which provide instant feedback on their code. However, while it focuses on entry level programming skills, it is not designed for university level coding labs, examinations, and competitions. To address this gap, we developed BUCTOL. BUCTOL enables students to complete assignments, exams, and competitions entirely within the same webpage without having to move to other webpages or programs. In addition, BUCTOL includes a powerful set of built-in plagiarism detection functions, enabling teachers to observe and replay the entire code writing process for each student. When students complete their assignments or exams, these functions can help identify where plagiarizing may have occurred. By reducing the student reliance on plagiarizing behaviors, BUCTOL can be a significant tool in allowing students to learn to code, allowing teachers to focus more of their time on teaching, rather than on plagiarism detection.

VII. Conclusion and Future Work

We have developed BUCTOL, an online programming education platform with built-in plagiarism monitoring functions. Through BUCTOL, students can complete all programming-related assignments, exams and competitions from within a single webpage. Different from the traditional offline programming tasks, it enhances students' initiative, meet multi person real-time programming, real-time criticism and real-time feedback.

To help teachers identify plagiarizing behaviors in programming education, The built-in plagiarism monitoring functions can calculate the code repetition rates based on AST. When students submit the code, compare the code with all the correct answers submitted for the same question, indicates the level of code duplication and supports real-time recording and playback of student code editing behavior data. Teachers can browse this data in real-time to ensure a more efficient and accurate examination supervision process which also aids in plagiarism identification.

We compared the code repetition before and after the use of BUCTOL in the examinations process. The comparison in results clearly demonstrated a reduction in the repetition rate of exams after adopting the BUCTOL platform. Therefore, the above results show that BUCTOL can prevent students from engaging in plagiarism and thus reduce the occurrence of plagiarizing. In addition, using a questionnaires, involving 92 students who have used BUCTOL, we found a highly positive attitude within students towards the inclusion of plagiarism monitoring functions in BUCTOL. Most of the students agreed that BUCTOL's plagiarism monitoring function could promote fairness in exams and reduce the occurrence of plagiarizing.

For future work, our goal is to develop a more efficient, accurate and automated plagiarizing monitoring function. We plan to extend the collection of students' actions in the programming process to more accurately record the students' answering process. We also plan to automatically aggregate all the data monitored by BUCTOL, and use algorithms to automatically generate the probability of each student's plagiarizing behavior, so as to further improve the efficiency and accuracy of the teacher's identification of plagiarizing behavior.

Acknowledgment

The work is supported by the National Natural Science Foundation of China (Grant nos. 61902015, 61872026 and 61672085), and Educational research projects of Beijing University of Chemical Technology (Grant nos, 2021BHDJGYB16 and G-JG-PTKC202107).

References

[1] S. Lonn and S. D. Teasley, "Saving time or innovating practice:Investigating perceptions and uses of learning management systems," Computers & education, vol. 53, no. 3, pp. 686–694, 2009.

[2] J. Baugh, P. Kovacs, and G. Davis, "Does the computer programming student understand what constitutes plagiarism," Issues in InformationSystems, vol. 13, no. 2, pp. 138–145, 2012.

[3] G. Cosma and M. Joy, "Towards a definition of source-code plagiarism," IEEE Transactions on Education, vol. 51, no. 2, pp. 195–200, 2008.

[4] J. Sheard, M. Dick, S. Markham, I. Macdonald, and M. Walsh, "Cheating and plagiarism: Perceptions and practices of first year it students," in Proceedings of the 7th annual conference on Innovation and technology in computer science education, 2002, pp. 183–187.

[5] E. L. Glassman, J. Scott, R. Singh, P. J. Guo, and R. C. Miller, "Overcode: Visualizing variation in student solutions to programming problems at scale," ACM Transactions on Computer-Human Interaction (TOCHI), vol. 22, no. 2, pp. 1–35, 2015.

[6] S. Kaleeswaran, A. Santhiar, A. Kanade, and S. Gulwani, "Semi supervised verified feedback generation," in Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Soft- ware Engineering, 2016, pp. 739–750.

[7] S. Gulwani, I. Radi˘cek, and F. Zuleger, "Feedback generation for performance problems in introductory programming assignments," in Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2014, pp. 41–51.

[8] 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, 2013, pp. 15–26.

[9] S. Wasik, M. Antczak, J. Badura, A. Laskowski, and T. Sternal, "A survey on online judge systems and their applications,"ACM Computing Surveys (CSUR), vol. 51, no. 1, pp. 1–34, 2018.

[10] S. Burrows, S. M. Tahaghoghi, and J. Zobel, "Efficient plagiarism detection for large code repositories," Software: Practice and Experience, vol. 37, no. 2, pp. 151–175, 2007.

[11] L. Luo, J. Ming, D. Wu, P. Liu, and S. Zhu, "Semantics-based obfuscation-resilient binary code similarity comparison with applications to software plagiarism detection," in Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2014, pp. 389–400.

[12] C. Kustanto and I. Liem, "Automatic source code plagiarism detection," in 2009 10th ACIS International conference on software engineering, artificial intelligences, networking and parallel /distributed computing. IEEE, 2009, pp. 481–486.

[13] H.-b. Zhang, "Design and implementation of the open cloud platform based open source online judge system," Comp. Sci., vol. 39, no. 11A, pp. 339–343, 2012.