CODECHUM: AN ONLINE LEARNING AND MONITORING PLATFORM FOR C PROGRAMMING

Jemar Jude A. Maranga, Leilla Keith J. Matugas, Jorge Frederick W. Lim
and Cherry Lyn C. Sta. Romana
Cebu Institute of Technology- University, Philippines

ABSTRACT

Teaching an introductory programming course to an average of 40 students while monitoring their performance can be a challenge for instructors. Preparing coding exercises with test cases and checking students' programs can prove to be time consuming at times. Moreover, programming has been known to be quite difficult for students to learn. To address these concerns, a web-based Integrated Development Environment (IDE) for C Programming called CodeChum was developed. CodeChum provides instructors with an easier way of creating programming exercises, an automated way of checking student codes and a dashboard for monitoring students' performance. CodeChum allows students to apply their skills using problem stories attached to learning outcomes, test their solutions, get instant feedback and monitor their current status through a leaderboard system. CodeChum was tested within a span of two months on 120 undergraduate students taking up introductory programming courses and 6 programming instructors from Cebu Institute of Technology- University. A usability and feedback survey for students and instructors was conducted getting inputs of their experience while using the IDE. Results of the survey show that the system was well received both by students and instructors. Moreover, with the continued use of CodeChum, data can be gathered that can be used towards learning analytics.

KEYWORDS

Online Monitoring and Learning Software, Web-based Tool for Teacher Support, Student Assessment

1. INTRODUCTION

In this fast paced world of emerging technologies, it has become an undeniable fact that the Information Technology industry is already rapidly growing and does not intend to stop. Because of this, a lot of jobs in relation to the field have started to emerge and the demand for people with the necessary skill set in both hardware and software engineering is becoming significantly high. In fact, startups and big companies alike are looking for creative, innovative developers and software engineers to design and refine computer programs. According to the U.S. Bureau of Labor Statistics, there is an expected growth of more than 17% with regards to the demand for software engineers by 2022 – an increase far faster than average, giving a median salary of $102,280.00 (Lopaze, 2018). In the Philippines alone, IT related programs have emerged as one of the top 10 degree programs students took with 398,765 students enrolled, making it the fourth most popular career path for incoming college students in the Philippines (Enriquez, 2018). However, although the demand both by recruiters and aspiring professionals seem to be high, the problem with incompetent graduates still arise. A study done last 2017 showed that over 50% of college students under Information Technology courses in the Philippines deemed incompetent and lacked programming skills that could pass for international standards (Suarez, 2017). Another survey showed that an extensive amount of data on perceived difficulties were related to programming concepts and program construction (Ala-Mutka, et.al, 2005). With this in mind, a solution that could help programming instructors monitor a student's performance, an easier way to create programming exercises, automate checking for outcome based problems and monitor the student's performance relative to his/her peers through a web-based Integrated Development Environment (IDE) called CodeChum was developed. CodeChum would also allow students to better understand the applications of each problem relative to its learning outcome through producing problem stories that would stress the practical applications of the said outcome. In general, CodeChum aims to provide an accessible IDE for C programming that aims to become a fun, interactive and challenging platform for novice student programmers and an easy monitoring and assessment tool to complement a programming instructor's teaching strategies. CodeChum was deployed within a span of two months in Cebu Institute of Technology- University and was used by 120 undergraduate students taking up introductory courses in programming and 6 programming instructors. A usability survey was conducted in order to assess the usefulness and value of the proposed IDE to its users.

2. LITERATURE REVIEW

Programming has been a relevant skill to learn as it can be applied in various fields. However, the learning curve for learning basic programming concepts seem to be high for most novice programmers. Drop out, unmotivated and high level retention of student's in the first year are problems that is imperative to solve (Costa & Piteira, 2012). A previous study last 2007 was conducted, with participating universities all over the world, in order to assess whether there is an alarmingly high failure rate for CS1 students (Bennedsen & Caspersen, 2007). Unfortunately, due to lack of participants it was hard to conclude with 33% that the failure rate was alarming enough. However, a follow up study was conducted revisiting the said matter in 2014 (Watson, & Li, 2014), and similarly it was proved to show a worldwide pass rate of 67.7% which did not necessarily increase over the years.

With that said, studies relating to the difficulties of novice programmers have been conducted in order to know why and in what aspect they find it difficult. A study conducted (Muraina & Rahman, 2014) revealed that practical classes and laboratory usage have significant contribution in improving a student's achievement in computer programming (Costa & Piteira, 2013, Muriana & Rahman, 2014). Another similar study (Lahtinen, et.al) suggests that the biggest problem of novice programmers does not seem to be the understanding of basic concepts but rather learning to apply those (Boudouvis, et.al, 2011, Mhashi & Alakeel, 2013). Also, another factor that may result to the difficulty in learning programming concepts is the gap between the understanding of the instructor and the students. The same study also suggested to construct different learning materials that could assess the student's program generation, modification and debugging skills.

Currently, programming instructors use traditional methods in evaluating a student's performance in class namely paper-and-pencil testing (PPT) and computer-based assessment (CBA) among others, with studies leaning more towards a successful skill assessment for novice programmers using the CBA method (Boudouvis, et.al, 2011, Nouri & Öqvist, 2018). A study conducted to assess a novice programmer's progress while using virtual appliances (Pardo, et.al, 2012) resulted in a significant correlation with student academic achievement while using online methods. Furthermore, it also validated the use of a prediction mechanism for further improvements (Pardo, et.al. 2012).

With the acceptance of computer-based assessment for novice programmers and the recommendation of laboratory activities and more advanced learning materials, CodeChum implements the following features that can both aide the students to better their performance and instructors in better assessing and monitoring a student's performance:

F1- automated checking for students' coding exercises

Common advantages of automation are the speed, availability, consistency and objectivity of assessment (Ala-Mutka, 2007). However, a disadvantage would be not being able to monitor the structure and design of a student's code. CodeChum addresses this problem through the ability to be able to revisit a student's code after every activity, which can be accessed by the programming instructor.

F2- creating problem stories to better understand programming concepts and their real-life applications

CodeChum would present different coding exercises as problem stories in order to support the study mentioned above and how it could possibly provide a real-life application for different programming concepts for the student's to better understand why they are taught certain concepts, which will potentially lead to an improved learning in introductory programming.

F3- leaderboard system to encourage additional personal motivation

CodeChum implements a leaderboard system in order for the students to access their performance and to know their current standing in class. The leaderboard system is accompanied by time-bound activities which can provide a game-like effect to students answering the activities (Tan, et.al, 2009). This is also beneficial for teachers in monitoring the performance of their students which can encourage further consultations and a more personal teaching style depending on the student.

F4- online IDE

CodeChum is presented as an online IDE for C programming which is accessible to students and teachers anytime, anywhere. This enables teachers to monitor their students right away, instead of using traditional or conventional IDEs like Dev-C++. This also places a student's learning data in one place which can be used for further learning analytics to improve the instructor's teaching style depending on the student.

3. METHODS

3.1 Development of the System

The system has been developed on the web platform. It is specifically made with a solid and well-known Python web framework, Django. For its database, it uses MySQL. The tool has two major parts:

I. Integrated Development Environment
II. Teacher's Panel

3.1.1 Integrated Development Environment

This is the part of the whole system where the student solves programming problems made by their teacher. Currently, it only supports the C programming language. Its appearance is somehow similar to existing competitive coding platforms online such as Hackerrank and Codefights. It basically has 5 major parts:

1. Problem Description - This shows the description of the problem (which was made by the teacher) currently solved.

2. Code Editor - This is where the student types his/her code. The programming language to be used is the programming language set by the teacher for this specific task. However, for this research, only the C programming language is supported.

3. Playground Input/Output – This is where the student can supply a sample input for his code and test run his code. When the student clicks the Run button, the value in the sample input field will be supplied to the student's code and the output of his/her code will show in the output section.

4. Test Cases – This is where the shown test cases for the problem are listed. When the code is submitted, these test cases will be executed to score the answer of the student.

5. Leaderboards – This shows the ranking of each student taking the same exercise after the allotted time is finished.

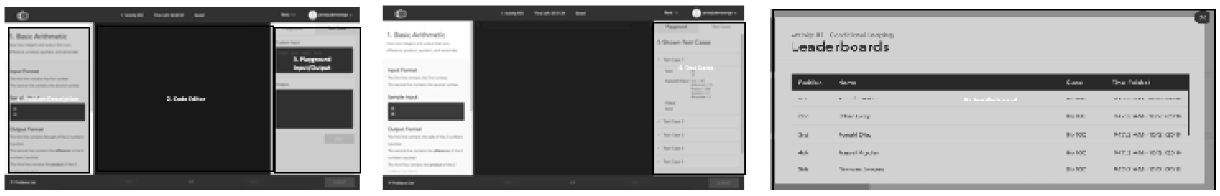


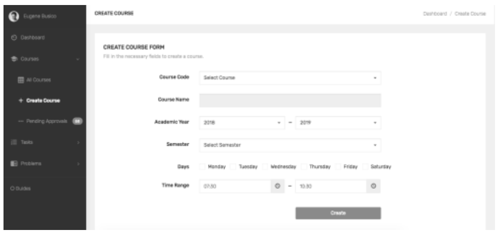
Figure 1. System's Integrated Development Environment

ISBN: 978-989-8533-83-8 © 2019

3.1.2 Teacher's Panel

This is where the programming instructors monitor the status and performance of each of their students for each of their respective classes. The teacher's panel contains the features that are needed for teachers to interact with their students. It has 3 major sections:

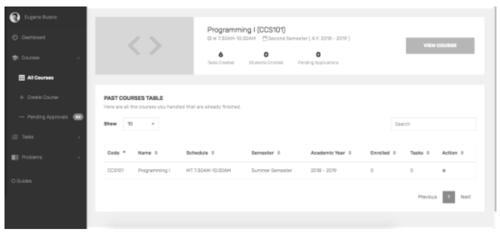
1. Courses Section – In this section, the teacher can create a course at the start of the semester (see Figure 2). Furthermore, the teacher can view his/her courses and their corresponding details such as the number of tasks created, students enrolled, schedule, and etc. (see Figure 3).



CREATE COURSE FORM
Fill in the necessary fields to create a course.

Course Code	Select Course
Course Name	
Academic Year	2018 – 2019
Semester	Select Semester
Days	Monday Tuesday Wednesday Thursday Friday Saturday
Time Range	07:30 – 10:30

Figure 2. Create Course



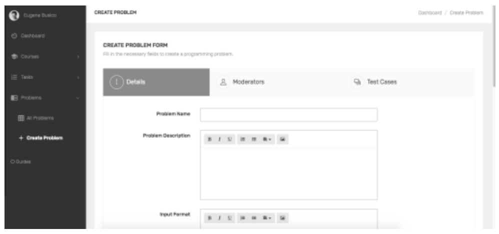
Programming I (CCS101)
W 7:30AM-10:30AM Second Semester (A.Y. 2018 - 2019)
6 Tasks Created 0 Students Enrolled 0 Pending Applications

PAST COURSES TABLE
Here are all the courses you handled that are already finished.

Code	Name	Schedule	Semester	Academic Year	Enrolled	Tasks	Action
CCS101	Programming I	MT 7:30AM-10:30AM	Summer Semester	2018 - 2019	0	0	✕

Figure 3. All Courses

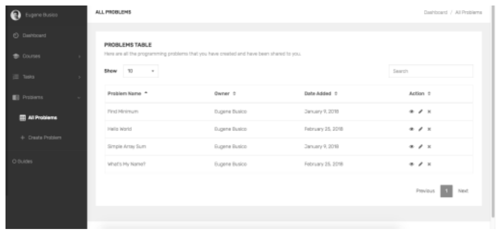
2. Problems Section – In this section, the teacher can create programming problems that can be used in the tasks he/she creates (see Figure 4). In the creation of the problem, the teacher adds the details of the problem, the moderators of the problem, and the test cases of the problem, which will be used for the automatic checking feature. Also the teacher can view all the problems he/she has created (see Figure 5).



CREATE PROBLEM FORM
Fill in the necessary fields to create a programming problem.

Problem Name
Problem Description
Input Format

Figure 4. Create Problem

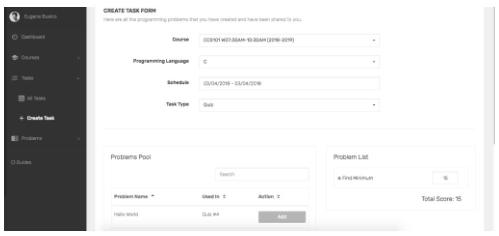


PROBLEMS TABLE
Here are all the programming problems that you have created and have been shared to you.

Problem Name	Owner	Date Added	Action
Find Minimum	Eugene Busico	January 9, 2018	👁 ✎ ✕
Hello World	Eugene Busico	February 25, 2018	👁 ✎ ✕
Simple Array Sum	Eugene Busico	January 9, 2018	👁 ✎ ✕
What's My Name?	Eugene Busico	February 25, 2018	👁 ✎ ✕

Figure 5. All Problems

3. Tasks Section – In this section, the teacher can create a task for a specific course he/she has. In doing so, the teacher needs to input the basic details of the task and then assign problems to it from the problem he/she has created or has been shared with (see Figure 6). The teacher can then view all the tasks he/she has created for all of his/her courses (see Figure 7).



CREATE TASK FORM
Here are all the programming problems that you have created and have been shared to you.

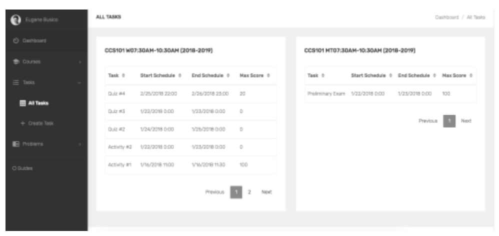
Course	CCS101 W07:30AM-10:30AM (2018-2019)
Programming Language	C
Schedule	03/04/2018 - 03/04/2018
Task Type	Quiz

Problems Pool

Problem Name	Used In	Action
Hello World	Quiz #4	Add

Problem List
✕ Find Minimum 15
Total Score: 15

Figure 6. Create Task



CCS101 W07:30AM-10:30AM (2018-2019)

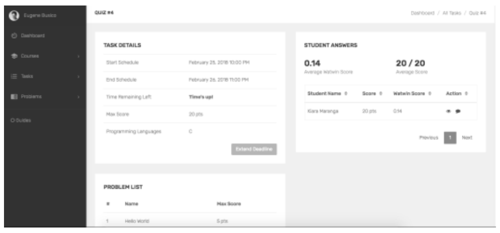
Task	Start Schedule	End Schedule	Max Score
Quiz #4	2/25/2018 22:00	2/26/2018 23:00	20
Quiz #3	1/22/2018 0:00	1/23/2018 0:00	0
Quiz #2	1/24/2018 0:00	1/25/2018 0:00	0
Activity #2	1/22/2018 0:00	1/23/2018 0:00	0
Activity #1	1/16/2018 11:00	1/16/2018 11:30	100

CCS101 MT07:30AM-10:30AM (2018-2019)

Task	Start Schedule	End Schedule	Max Score
Preliminary Exam	1/22/2018 0:00	1/23/2018 0:00	100

Figure 7. All Tasks

Then, the teacher can view all the details of a task. Aside from the main details of the task, the teacher can view the average actual scores of the students who answered the task along with their individual actual scores (see Figure 8).



QUIZ #4

Dashboard / All Tasks / Quiz #4

TASK DETAILS

Start Schedule	February 25, 2018 10:00 PM
End Schedule	February 26, 2018 11:00 PM
Time Remaining Left	Time's up!
Max Score	20 pts
Programming Languages	C

STUDENT ANSWERS

0.14 Average Watwin Score

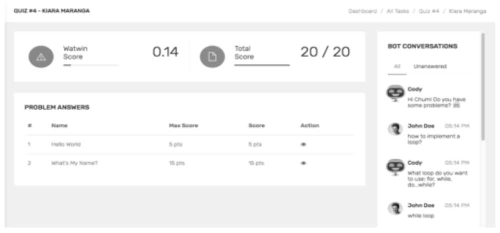
20 / 20 Average Score

Student Name	Score	Watwin Score	Action
Kiara Maranga	20 pts	0.14	

PROBLEM LIST

#	Name	Max Score
1	Hello World	5 pts

Figure 8. Single Task Details



QUIZ #4 - KIARA MARANGA

Dashboard / All Tasks / Quiz #4 / Kiara Maranga

Watwin Score 0.14

Total Score 20 / 20

PROBLEM ANSWERS

#	Name	Max Score	Score	Action
1	Hello World	5 pts	5 pts	
2	What's My Name?	15 pts	15 pts	

BOT CONVERSATIONS

Cody: Hi Chum! Do you have some problems?

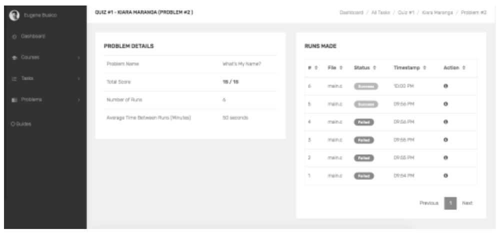
John Doe 05:14 PM: how to implement a loop?

Cody 05:14 PM: What loop do you want to use: for, while, do...while?

John Doe 05:14 PM: while loop

Figure 9. Monitor Student

The teacher can also monitor further his/her student (see Figure 9). In this page, the teacher can view the score of the student in each of the problems of the task. From here, the teacher can monitor further down his/her student by viewing how the student really answered the programming problem. This solves the problem that programming teachers are facing: all they know is if the student was able to answer or not, but do not know the details in between (see Figure 10).



QUIZ #1 - KIARA MARANGA (PROBLEM #2)

Dashboard / All Tasks / Quiz #1 / Kiara Maranga / Problem #2

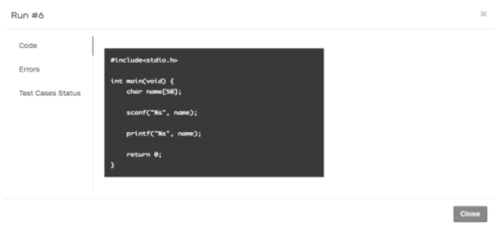
PROBLEM DETAILS

Problem Name	What's My Name?
Total Score	15 / 15
Number of Runs	6
Average Time Between Runs (Minutes)	50 seconds

RUNS MADE

#	File	Status	Timestamp	Action
6	main.c	Success	10:00 PM	
5	main.c	Success	09:56 PM	
4	main.c	Failed	09:56 PM	
3	main.c	Failed	09:55 PM	
2	main.c	Failed	09:55 PM	
1	main.c	Failed	09:54 PM	

Figure 10. List of runs made by student



```
#include<stdio.h>

int main(void) {
    char name[50];

    scanf("%s", name);

    printf("%s", name);

    return 0;
}
```

Figure 11. Specific details per run made by student

For each run the student made on that problem, the teacher can view the actual code of that run, the errors in that run, and the test cases status (see Figure 11).

3.2 Testing

The participants are 120 undergraduate students taking up introductory programming courses and 6 programming instructors from Cebu Institute of Technology- University. All of them were able to use CodeChum in the span of two months for their laboratory exercises. A total of 120 surveys for the students were returned with a response rate of 100%, and a total of 6 surveys for the programming instructors were returned. Two separate surveys were prepared in order to assess the usability of CodeChum for the students and the instructors. Tables 1 and 2 show the questions prepared for the students and the programming instructors respectively. The questions below were formulated in order to assess specific features and the scale of how the users would accept the system. Data will be presented by displaying the frequency of answers in percentage and average depending on the question type.

Table 1. Student Usability Survey Questions

No.	***Question***	***Question Type***
1	Overall, how satisfied were you with CodeChum?	4-point Likert Scale
2	How easy was CodeChum to use?	4-point Likert Scale
3	Rate the problem stories based on its ability to explain the problem?	4-point Likert Scale
4	Are the problem stories fun and interesting?	Yes / No
5	Would you recommend CodeChum as a way of testing and learning?	Yes / No
6	When answering activities in the lab, which of the following do you prefer to use	Multiple Choice (CodeChum / Dev-C++)
7	List down all the things you like about CodeChum.	Open-ended
8	List down all the things you dislike about CodeChum.	Open-ended
9	If you were the owner and developer of CodeChum, what would you do to improve it?	Open-ended

Table 2. Instructor Usability Survey Questions

No.	***Question***	***Question Type***
1	The software and its features are easily understandable. I did not find it hard to navigate through each page even though I am a beginner.	4-point Likert Scale
2	I was able to create coding exercises for students easily.	4-point Likert Scale
3	The existing database of questions available in CodeChum is aligned to the intended learning outcomes of the course.	4-point Likert Scale
4	CodeChum made it faster for me to check and evaluate my student's coding exercises.	4-point Likert Scale
5	I would prefer to use CodeChum over other C Programming IDEs because of its usability and accessibility.	4-point Likert Scale
6	I will recommend my fellow instructors to use this online environment for monitoring their students.	4-point Likert Scale
7	Before CodeChum, what did you use in assessing your student's performance? Please input all possible answers.	Open-ended
8	List down all the things you like about CodeChum.	Open-ended
9	List down all the things you dislike about CodeChum.	Open-ended
10	If you were the owner and developer of CodeChum, what would you do to improve it?	Open-ended

For questions involving the 4-point Likert Scale question type, a response average will be computed for each using the formula show in Figure 12 where the sum of the frequency of the results *(f)* multiplied to their corresponding points *(p)* will be divided by the total number of responses *(n)* and will result to a response point ranging from 1-4. The proponents have chosen a threshold of 3.0 as an indicator for good performance.

$$AVERAGE = \frac{\sum f * p}{n}$$

Figure 12. Response Point Average Formula

4. RESULTS AND DISCUSSION

Tables 3-6 show the responses from 120 undergraduate students taking up introductory programming courses. As shown in Table 3, the data are presented as frequency percentages for the 4-point Likert scale. Responses for the Yes/No and multiple choice question types are shown in Table 4 and another separate table (see Table 5 and 6) shows the results for the open-ended question types from the same student usability and feedback survey.

Table 3. Responses: Student Usability and Feedback Q1-Q3

No.	***Frequency of Responses***				***Response Average***
	1	**2**	**3**	**4**	
1	2	7	36	75	3.53
2	2	14	40	64	3.38
3	2	11	56	51	3.30

For Q1, majority of the participants (62.5%) responded that they were *very satisfied* with CodeChum, with an average point of 3.53 out of 4 points. For Q2, 53.3% agreed that CodeChum was *very easy* to use with an average point of 3.38. For Q3, majority of the participants (46.7%) said that the problem stories were *fairly easy to understand* with an average point of 3.30.

Table 4. Responses: Student Usability and Feedback Q4-Q6

No.	***Frequency of Responses***		***Favorable Average***
	YES	**NO**	
4	116	4	96.67%
5	114	6	95.0%
	Frequency of Responses		***Favorable Average***
	CODECHUM	**DEV-C++**	
6	83	37	69.2%

For Q4, majority of the participants (96.67%) found the problem stories fun and interesting to answer, leaving 3.3% thinking otherwise. For Q5, majority of the participants (95.0%) would highly recommend CodeChum as a way of assessing and learning introductory programming concepts, leaving 5% thinking otherwise. For Q6, 69.2% would prefer to use CodeChum for their laboratory activities, 30.8% would prefer to use Dev-C++.

Table 5. Top 5 Responses: Student Usability and Feedback Q7-Q8

No.	***Q7: Good Points***		***Q8: Bad Points***	
	Description	**Frequency**	**Description**	**Frequency**
1	User Interface	43	Slow Compilation Time	18
2	User Experience	33	None	16
3	Test Cases	22	Hidden Test Cases	14
4	Problem Stories	22	Codes in C++ Don't Work in CodeChum	10
5	Improves Critical Thinking	14	Internet Lag	9
6	Leaderboard Feature	9	Code Can't Be Revisited After Activity	6
7	Time-bounded Activities	8	Questions Are Too Hard	5

For Q7 and Q8, the participants were asked to list down the good and bad points of CodeChum. Listed in Table 5 above shows items with a significant number 5 frequent responses and above. Other items suggested were not included as they didn't meet the said minimum number of frequent responses. Based on the aforementioned table, most of the participants felt the user interface and user experience, test cases and problem stories were good points of CodeChum. They also mentioned that the time-bounded activities which improve critical thinking and the leaderboard feature also make it more fun and challenging. However, majority of the participants also felt the need to improve on the slow compilation time among others, but a significant number of participants also felt that there were no bad points for CodeChum.

Table 6. Top 5 Responses: Student Usability and Feedback Q9

Q9: Improvement Points		
No.	**Description**	**Frequency**
1	None	36
2	Practice Exercises Outside the Course	18
3	Faster Compilation Time	8
4	Improved User Interface	7
5	Better Server Connection	5

Similarly, for Q9, listed above (see Table 6) are the significant items which the participants felt were improvement points for CodeChum and although majority responded that they had nothing to suggest as improvements, a significant number of participants pointed out that more practice exercises outside recorded activities, and a faster compilation time among others could be seen as points for improvement for CodeChum.

Tables 7-10 shows the responses from 6 programming instructors from Cebu Institute of Technology- University. The participating programming instructors are the ones who facilitate the laboratory activities of the 120 undergraduate students participating in the testing as well. A ratio of 1:20 corresponds to the number of programming instructors to students under introductory programming courses. Data presented in Table 7 shows responses for questions 1-6.

Table 7. Responses: Instructor Usability and Feedback Q1-Q6

No.	***Frequency of Responses***				***Response Average***
	1	**2**	**3**	**4**	
1	0	0	1	5	3.83
2	0	2	2	1	2.8
3	0	0	1	5	3.83
4	0	0	1	5	3.83
5	0	1	1	4	3.5
6	0	0	1	5	3.83

For Q1, 83.3% of the participants strongly agreed that the system was easily navigable for beginners indicating its user-friendliness with an average point of 3.83. For Q2, majority of the 5 respondents (one did not choose to answer) found it *hard* to create coding exercises using of CodeChum, with an average point of 2.8. For Q3, majority of the participants (83.3%) *strongly agreed* that questions present in CodeChum's database is aligned to their current intended learning outcomes, with an average point of 3.83. For Q4, majority of the participants (83.3%) *strongly agreed* that CodeChum made it faster for them to check and evaluate their student's performance, with an average point of 3.83. For Q5, 66.7% of the participants would *highly prefer* to use CodeChum over other programming IDEs mainly because of its usability and accessibility. For Q6, 83.3% of the participants would *highly recommend* CodeChum to their fellow instructors as an online environment for learning and monitoring their students, with an average point of 3.83.

Table 8. Top 5 Responses: Instructor Usability and Feedback Q7

Q7: Assessment Methods Before CodeChum		
No.	**Description**	**Frequency**
1	Traditional	1
2	Offline Practical Exam	2
3	Dev-C++	3
4	Manual Checking of Activities	1

Based on the answers of the participants, their main method for assessing student performance with regards to their programming skills would be the traditional or conventional way with specifics on hands-on or practical exams, exams using Dev-C++ as the IDE and these exercises are then assessed using manual checking of activities, quizzes, seatworks and the like.

Table 9. Top 5 Responses: Instructor Usability and Feedback Q8-Q9

No.	*Q8: Good Points*		*Q9: Bad Points*	
	Description	**Frequency**	**Description**	**Frequency**
1	Automated Checking	5	None	2
2	Database of Exercises	2	Does not check algorithm logic	2
3	Student Performance Monitoring	2	Site Bugs and Crashes	2
4	UI/UX	2		
5	Leaderboard System	1		

For Q8 and Q9, listed in Table 9 shows the good and bad points of CodeChum based on responses from the programming instructors. Based on the aforementioned table, most of the participants felt that the automated checking, database of exercises, student performance monitoring, the user interface of CodeChum, and the leaderboard system were its good points. However, majority of the participants also felt the need to improve on fixing the site bugs and crashes, as well as improving the checking system for exercises.

Table 10. Top 5 Responses: Instructor Usability and Feedback Q10

Q10: Improvement Points		
No.	**Description**	**Frequency**
1	Include Item Analysis	1
2	Implement Student Dashboard	1
3	Check Algorithm Logic for Exercises	1
4	Improve Scoring System	2
5	Fix site bugs and issues	1
6	Check for Cheating Anomalies	1

Similarly, for Q9, listed above (see Table 10) are the significant items the participants felt were improvement points for CodeChum and although majority responded that they had nothing to suggest as improvements, a significant number of participants pointed out that more practice exercises outside recorded activities, and a faster compilation time among others could be seen as points for improvement for CodeChum.

5. CONCLUSION AND FUTURE RESEARCH

Based on the usability and feedback survey conducted, results show that the system was well received both by the students and the instructors. The user interface and user experience were major factors for the choice, but the results also pointed out the effectiveness of the problem stories, time-bounded activities, and leaderboards in creating a better leaning experience during coding exercises. It can also be significantly inferred that students would prefer to use CodeChum over other conventional or traditional programming IDEs. Although certain factors like slow compile time and hidden test cases among others, are some of CodeChum's weak points, majority of the participants were still very satisfied with CodeChum and would highly recommend it for learning and assessing one's programming skills.

As for the instructors, although they found it fairly hard to create coding exercises, overall positive feedback was shown in features like automated checking, CodeChum's current problem database, student monitoring, leaderboard system, and the overall user interface. Although certain factors like site bugs and crashes are some of CodeChum's weak points, majority of the participants were still very satisfied with CodeChum and would highly recommend it for monitoring and assessing their student's performance. Participants also suggest to improve on the scoring system, implementing a Student Dashboard, and fixing bugs and crashes, among others.

For future works, the proponents would like to incorporate learning analytics through using the data collected from each of the student's activities while using the IDE in order to provide more data to the programming instructors on how they could incorporate different teaching styles and strategies depending on what the students need the most. In addition, another usability survey is likely to be conducted again for both instructors and students, now using the SUS scoring method to properly calibrate and quantify the effectiveness and usability of CodeChum.

REFERENCES

Ala-Mutka, K., et.al. (2005). *A study of the difficulties of novice programmers.* ACM SIGCSE Bulletin, Volume 37, Issue 3, pp. 14-18.

Ala-Mutka, K. M. (2007). *A Survey of Automated Assessment Approaches for Programming Assignments.* Computer Science Education, Volume 15, Issue 2, pp. 83-102.

Bennedsen, J. & Caspersen, M. E. (2007). *Failure rates in introductory programming.* ACM SIGCSE Bulletin, Volume 39, Issue 2, pp. 32-36. Boudouvis, A. G., et.al. (2011). *Computer-based assessment of student performance in programing courses.* Retrieved December 18, 2018 from https://onlinelibrary.wiley.com/doi/pdf/10.1002/cae.20512

Costa, C. & Piteira, M. (2012). *Computer programming and novice programmers.* ISDOC '12 Proceedings of the Workshop on Information Systems and Design of Communication, pp. 51-53. Retrieved from https://dl.acm.org/citation.cfm?id=2311927

Costa, C. & Piteira, M. (2013). *Learning computer programming: study of difficulties in learning programming.* ISDOC '13 Proceedings of the 2013 International Conference on Information Systems and Design of Communication, pp. 75-80

Enriquez, F. (2018). *What Are The Top Courses in the Philippines? Courses by the Numbers!*. Retrieved December 18, 2018 from https://blog.edukasyon.ph/college-life/what-are-the-top-college-courses-in-the-philippines-courses-by the-numbers/

Lopaze, K. (2018). *10 of the most in-demand jobs for 2018*. Retrieved December 18, 2018 from https://www.thejobnetwork.com/10-of-the-most-in-demand-jobs-for-2018/

Mhashi, M. M. & Alakeel, A. M. (2013). *Difficulties Facing Students in Learning Computer Programming Skills at Tabuk University.* Proceedings of the 12th International Conference on Education and Educational Technology, pp. 15-24. Retrieved December 18, 2018 from https://pdfs.semanticscholar.org/bf2e/811dc31edcb7c5b85348c131634497756b64.pdf

Muriana, I. O. & Rahman, M. A. (2014). *Assessment of Students' Achievement in Computer Programming: An Inquiry into some Influencing Factors.* International Journal of Scientific & Engineering Research, Volume 5, Issue 3.

Nouri, J. & Öqvist, M. (2018). Coding by hand or on the computer? Evaluating the effect of assessment mode on performance of students learning programming. ," Journal of Computers in Education, Volume 5, Issue 2, pp. 199-219.

Pardo, A., et.al. (2012). *Monitoring student progress using virtual appliances: A case study.* Computers & Education, Volume 58, Issue 4, pp. 1058-1067.

Suarez, I. (2017). *Study: Over 50% of Pinoy IT students lack programming skills.* Retrieved December 18, 2018 from http://newsbytes.ph/2017/02/01/study-over-50-of-pinoy-it-students-lack-programming-skills/

Tan, P. H., et.al. (2009). *Learning Difficulties in Programming Courses: Undergraduates' Perspective and Perception.* 2009 International Conference on Computer Technology and Development. Retrieved December 20, 2018 from https://ieeexplore.ieee.org/document/5359977

Watson, C. & Li, F. W.B. (2014). *Failure rates in introductory programming revisited.* ITiCSE '14 Proceedings of the 2014 conference on Innovation & technology in computer science education, pp. 39-44.