Research Article

HandSnap++: An Android Application for Debugging Handwritten C++ Code using Image Processing through Optical Character Recognition and Compiler

Arias, Clarq Anderson P, Miranda, Stephen Felipin S, Paguinto, Juliana Arla S, Pangkubit, Rhea Joy M, Gamboa, John Carlo L. MIT, Mallari, Marvin O. Ph.D, EnM, and Pinpin,Arzel P. MIT

1 Arias, Clarq Anderson P; clarqarias@gmail.com;

2 Miranda, Stephen Felipin S; sm8635299@gmail.com;

3 Paguinto, Juliana Arla S; julianapaguinto426@gmail.com;

4 Pangkubit, Rhea Joy M; rheajoypangkubit@gmail.com;

5. Gamboa, John Carlo L,MIT;jaycee.gamboa09@gmail.com;

6. Mallari, Marvin O, Ph.D, EnM;mallarimarvin022@gmail.com;

7. Pinpin, Arzel P, MIT; pinpinarzel012@gmail.com;

Abstract: *Paper coding, where students write C++ codes on paper during assessments, enhances logical and problem-solving skills but still has challenges, particularly in manual debugging. HandSnap++ is an innovative application that extracts handwritten C++ codes utilizing Optical Character Recognition (OCR), compiles the scanned codes, provides debugging feedback, and scores the students' codes. This study employed a descriptive and developmental method within the quantitative research framework, gathering interviews and evaluations from 20 instructors and 20 students in the department of computer studies to assess the effectiveness of the HandSnap++ application in achieving its objectives that greatly contribute to the educational field and institution. The application, upon its development, proved to be an effective alternative tool for debugging in resource-limited environments, such as in computer laboratories, especially in state universities, as the app is portable. It also reduces overlooking errors in checking through OCR, provides timely feedback on scores for students, stores their answers digitally, and helps students adapt to utilizing compilers in an application. The application was evaluated with ISO 25010, and among the categories, its functional suitability attained the high score average while its performance efficiency was scored last. Even with these results, it is recommended to explore OCR models focused on improving special character recognitions as well as to expand the supported language, not just C++, for compiling.*

Keywords: papercoding, C++, image-processing,ocr,compiler

DOI: 10.5281/zenodo.15074249

1. INTRODUCTION

Over the past years, inventors, researchers, and technologists have continuously advanced today's technology by automating services, optimizing human-computer interaction, and integrating several systems to function as a single system. In particular, incorporating technology into traditional methods is innovative and revolutionary, allowing researchers and developers to modernize them. This provides alternative ways to complete tasks using emerging technologies.

One instance is the implementation of technology in education. It is especially significant as it digitalizes traditional methods, instances are utilizing soft copies of learning materials and employing e-learning management systems. These advancements are being implemented and developed further, providing educational tools that enhance teaching efficacy in educational institutions. This transformation is evident in the Philippines, where developers are gradually integrating these innovative solutions into education, even making them accessible via mobile applications.

The utilization of mobile applications in the development of systems is now the standard for upgrading systems from other platforms due to their portability. In classroom settings, such technology can benefit students in various areas. Using the mobile app, students can ask instructors questions about school-related issues and attend courses via video call. This promotes anywhere, at any time, access to education (Drigas & Angelidakis, 2017)

In accordance to that, a study by Demir and Akpinar (2018) showed the effectiveness of utilizing mobile app in education of undergraduate college students. The results showed a factor that this may promote academic achievement for students.

Central to these advancements is the conversion of traditional handwritten texts into digital text in the educational field, particularly handwritten C++ code on students' paper. C++ has been described as a high-level programming language developed by Bjarne Stroustrup in 1979 and first released in 1985. It is widely used for software development due to its effectiveness, efficiency, and versatility. C++ supports both procedural and object-oriented programming paradigms, making it suitable for a variety of applications (C++ Introduction, n.d.).

According to a study by Ortiz (2023), learning C++ is an excellent starting point for programming because it provides a strong foundation as a compiled language that converts code into machine language. It also offers control over memory management, allowing for efficient code without wasting resources. It is ideal for large-scale projects, with a variety of libraries and tools that support robust software development.

Consequently, C++ is introduced and taught extensively in computer literacy courses during the first and second years of a student's academic program. During this period, students gain a foundational understanding of the language, learning essential concepts and programming techniques crucial for their further studies and future careers in computer related fields.

Despite these developments, a gap still remains in the availability of educational tools and resources. This gap presents a unique opportunity for innovation, particularly through the conversion of handwritten C++ codes to digital text. Proposing a study that aims to debug handwritten C++ codes on paper is an idea that could lead to various solutions addressing challenges within educational settings.

2. METHOD & MATERIAL

In this research study, the researchers used a quantitative approach in order to test the development and functionality of HandSnap++, which is an Android application designed to debug handwritten C++ codes through OCR and a compiler. This is because a quantitative approach allows testing and validation of hypotheses using numerical data. Descriptive and developmental research methods were applied within a quantitative framework. Descriptive research was used to gather feedback about the effectiveness of the application, while the developmental aspect was getting made and improved based on ongoing feedback and testing. Purposive sampling was employed in this

research to select college instructors and students who are engaged with C++ programming, and through this sampling, relevant data was collected against the objectives set in this study.

A structured approach to development based on Scrum, Agile methodology, guided the development of HandSnap++. This was important for Scrum, allowing for a very iterative process of development to divide the project into manageable sprints with focused and clear development, making the feedback applicable at every stage of development. In the "Plan" phase, the researchers interviewed and surveyed college instructors and students about their user needs, challenges, and expectations. This information gathered by the researchers was then summarized to develop a detailed backlog that also included features and user storyboards, wireframes, application flowcharts, as well as data base designs; hence the approach taken toward the development proved to be user and educator-aligned.

The Design stage included developing an interface amiable to users and a solid database structure. The researchers aimed at designing an intuitive and user-friendly layout for the system's user interface. Both students and instructors would appreciate it. Instructor feedback from the C++ was included in the system to guarantee it met educational requirements. Simultaneously, the ERD of data storage was created to hold data in the most efficient and organized manner possible. This stage was essential to make the application work and ensure data integrity.

The Develop phase focused on the development of HandSnap++. Here, researchers were busy creating the core functionalities. The front-end user interface was designed and coded. A database connection was also set up and integrated with Amazon Textract for OCR to extract C++ code from images. Furthermore, the system integrated a compiler for debugging and analyzing the extracted code. Throughout this development phase, tools such as Flutter, XAMPP, and Visual Studio Code were used to provide a smooth development environment and to ensure efficient cross-platform compatibility.

The Test phase is where the functionality and performance of the application are evaluated. Alpha and beta testing were both performed. The alpha testing was first done by the research team in a controlled environment to catch bugs and to stabilize the application. Following this, beta testing is conducted to permit the application to be tested by a limited number of instructors and students for usability and functionality purposes. This testing cycle is crucial for the improvement of the application toward the needs of its intended users. A pilot test was conducted within the Computer Studies Department of a private school in Region 3, Philippines, to evaluate its effectiveness and suitability before applying it to other institutions.

Finally, Implementation is about the roll out of the application to its users for real time testing. Through this, invaluable feedback was garnered by the researchers on how effective the application will be in actual practice. From this stage of feedback, it was guided how further adjustments to the application were made while ensuring that it was able to achieve the goal of the research study. Ethical standards on informed consent, confidentiality, and voluntary participation during all stages.

3. FINDINGS

The researchers conducted pre-survey questionnaires using Google Forms with instructors and students from private higher education institutions and a state university in Region 3, Philippines, allowing them to respond at their convenience. The goal was to analyze the gathered data and gain a better understanding of the proposed system, ensuring that it would be both efficient and effective.

3.1 Developing an application that addresses the insufficiency of student-to-computer ratio in the computer laboratories by providing an alternative solution for debugging C++ codes using mobile devices

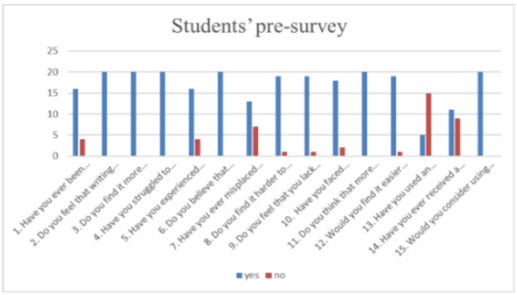


Figure 3.1. Students' Pre-survey

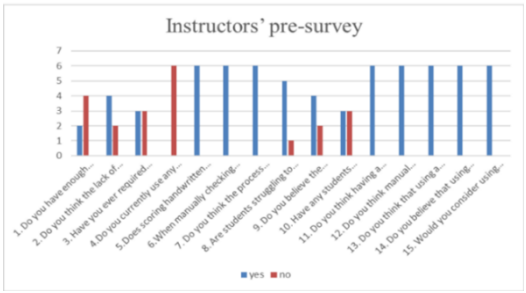


Figure 3.2. Instructors' Pre-survey

The figures 3.1 and 3.2 show the result of the responses of instructors and students from various colleges based on the pre-survey.

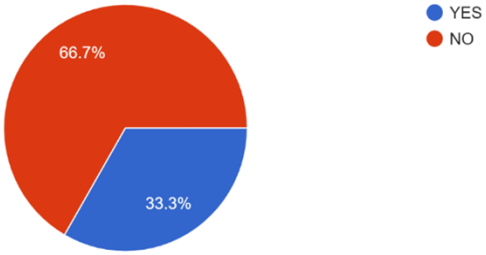


Figure 3.3. Q1 (instructor): Do you have enough computers for each student in computer laboratories during assessments (e.g., assignments, activities, exercises, or quizzes)?

From the pre-survey of instructors, Figure 3.3 shows a mixed perception regarding the availability of computers in computer laboratories during the pre-survey among instructors. Specifically, 66.7% of instructor respondents (4 out of 6) reported that the computers available are not enough for each student. However, a significant minority (33.3%, or 2 respondents) indicated that they do have sufficient computers, highlighting potential gaps in resource availability that could hinder effective assessment practices among instructors.

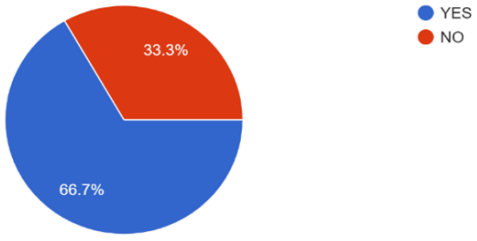


Figure 3.4. Q2 (instructor): Do you think the lack of computers in laboratories hinders students' ability to learn programming effectively?

Figure 3.4 reveals that most instructors, 66.7% (4 out of 6), agreed that not having enough computers makes it harder for students to learn programming, as hands-on practice is essential for understanding coding and problem-solving. In contrast, 33.3% (2 instructors) felt that the lack of computers does not necessarily prevent students from learning.

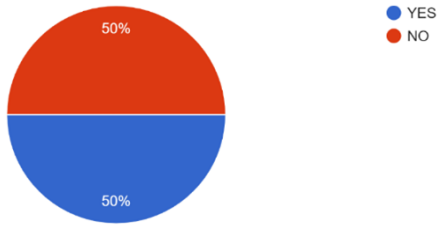


Figure 3.5. Q3 (instructor): Have you ever required students to write C++ or any programming language code due to lack of available computers during their assessments?

In addition, Figure 3.5 shows that 50% (3 instructors) reported asking students to handwrite code due to insufficient computers, while the other 50% (3 instructors) stated they have not done so. This indicates that some instructors may require students to manually write code not only because of a lack of computers but also as a part of their teaching methods to help students learn the structure and logic of coding without relying on compilers.

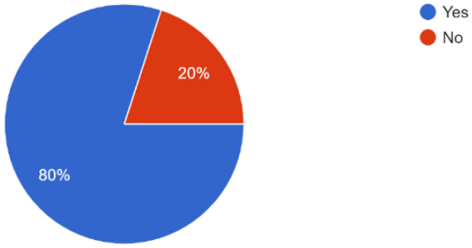


Figure 3.6. Q1 (student): Have you ever been required to write C++ code or any programming language by hand due to a lack of available computers in the lab?

Correspondingly, in Figure 3.6, 16 out of 20 students (80%) reported being required to write C++ code or code in other programming languages by hand because there were not enough computers available. This aligns with the instructors' experiences of resource limitations.

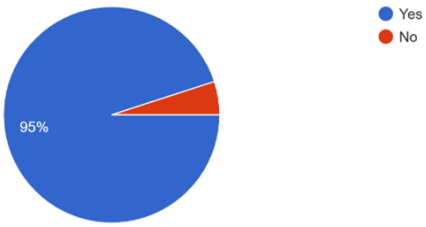


Figure 3.7 Q9 (student): Do you feel that you lack experience using compilers due to limited time on computers during coding lessons?

In Figure 3.7, it revealed that 19 out of 20 students (95%) feel they lack experience using compilers because they spend limited time on computers during coding lessons. This reflects the instructors' concerns that without sufficient time to practice on computers, students may struggle to understand how to compile and run their code, which is essential for becoming proficient programmers.

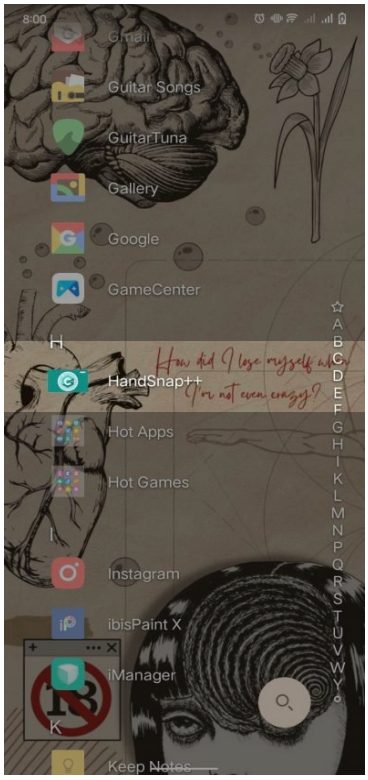


Figure 3.8 A Screenshot of HandSnap++ Installed in Android Device.

Due to this challenge, the researchers were driven to develop an app that offers an alternative solution for debugging programming codes, specifically in the C++ language, utilizing android mobile devices and not with just computers as shown in Figure 3.8.

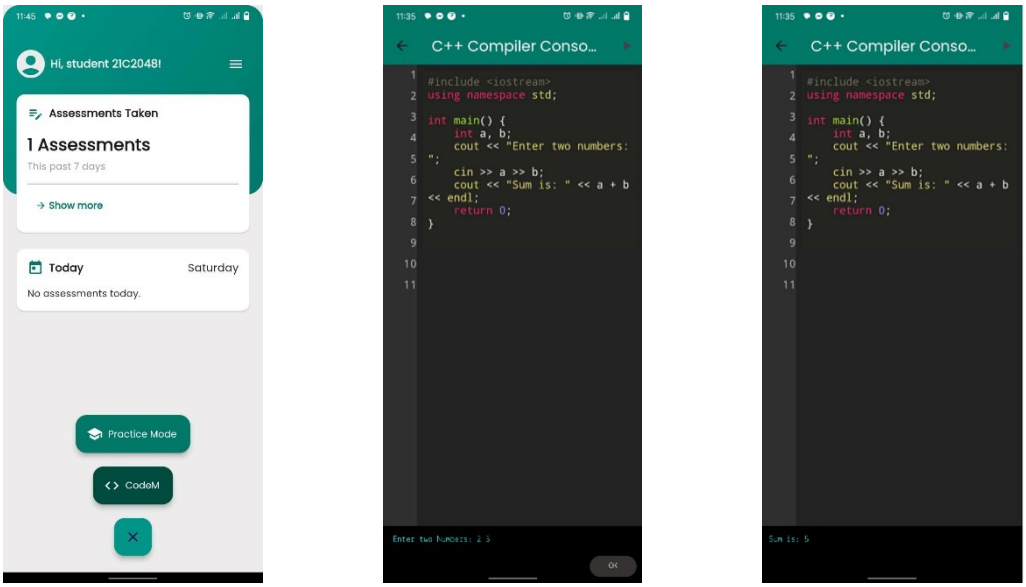


Figure 3.9. Screenshot of CodeM in student Account

The goal was to provide a more accessible and portable way for students to debug their code. This has been made possible through the use of Flutter as a development tool, allowing for the creation

of the HandSnap++ app. The app can effectively compile and debug beginner-level C++ code, bridging the gap in resources for traditional debugging tools. As shown in Figure 3.9, the app includes the CodeM feature that allow students to write their code directly in the editor.

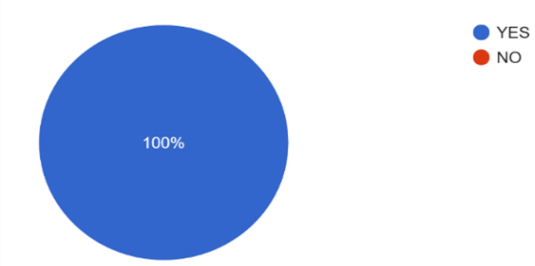


Figure 3.10 Do you feel that you lack experience using compilers due to limited time on computers during coding lessons?(Instructors)

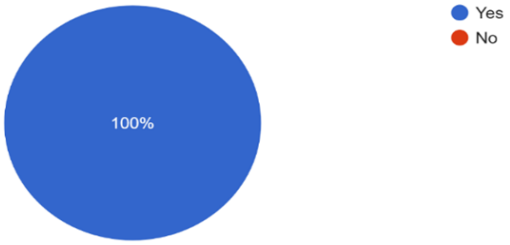


Figure 3.11 Q15 (student): Would you consider using an application that allows you to debug your code during practice lab activities?

Additionally, the respondents have agreed to utilize such a tool as this app if it is developed, as it is shown in Figure 3.10 and Figure 3.11. With 100% of students saying they would use the app, it shows that there is strong interest in this kind of tool. If developed, it could help many people by improving the way they check, debug, and evaluate code in the classroom.

3.2. Developing an application that reduces the likelihood of overlooked handwritten codes, ensuring more accurate evaluations of students' work.

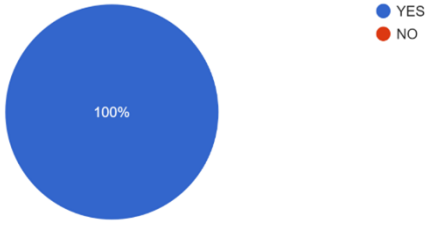


Figure 3.12 Q6 (instructor): When manually checking handwritten code, do you sometimes overlook syntax errors, logical mistakes, or inconsistencies?

In Figure 3.12, it showed that 100% (6 out of 6) instructors agreed that manual checking can lead to missed errors, which may affect the accuracy of assessments and hinder students' learning. The inability to catch these errors can result in students not fully understanding their mistakes, potentially leading to repeated issues in their coding practices.

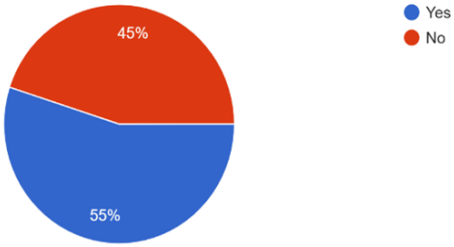
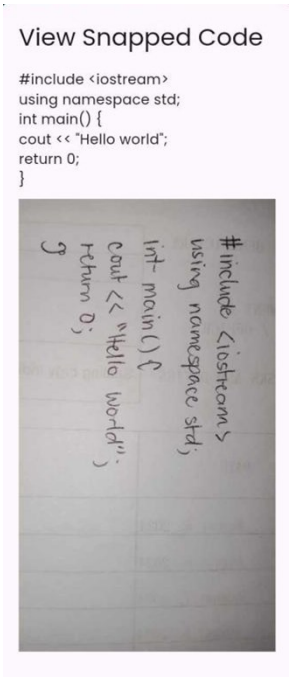


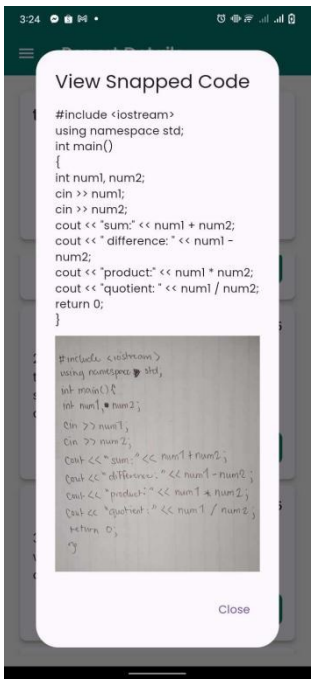
Figure 3.13 Q14 (student): Have you ever received a wrong score due to overlooked errors in your handwritten code?

In Figure 3.13, 11 out of 20 (55%) student respondents from table 2 experience receiving a wrong score due to overlooked errors in their handwritten code. This finding underscores the challenges instructors face when manually grading written assignments, as it can lead to mistakes that affect students' overall performance.

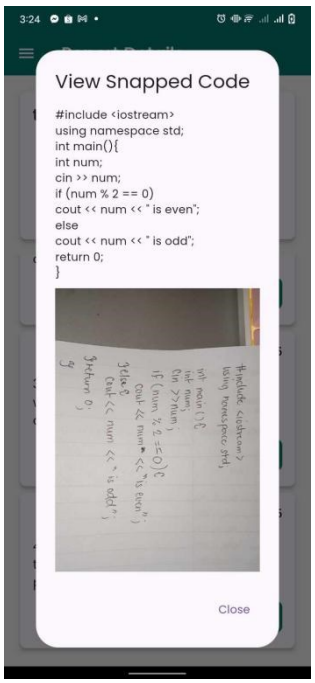
As a response to this challenge, the researchers have proposed an application that attempts to reduce the likelihood of overlooking handwritten code errors to ensure more accurate evaluations of student's assessments. Utilizing the OCR as a first phase before debugging a handwritten C++ code, this ensured the accuracy in the evaluation process of instructors. This technology allows the HandSnap++ to examine students' handwritten code for syntax and logical errors, not just instructors. This phase ensures the instructors have a tool that assists them in identifying subtle mistakes that are often overlooked during manual checking.



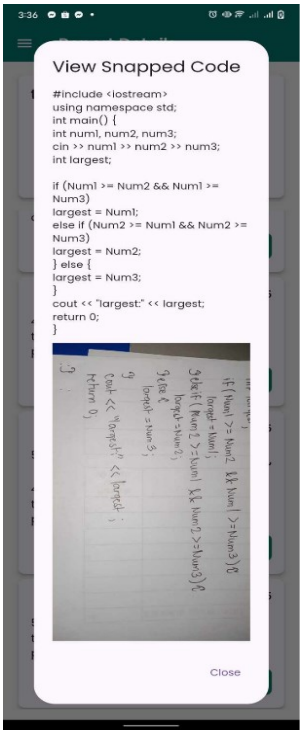
(a)



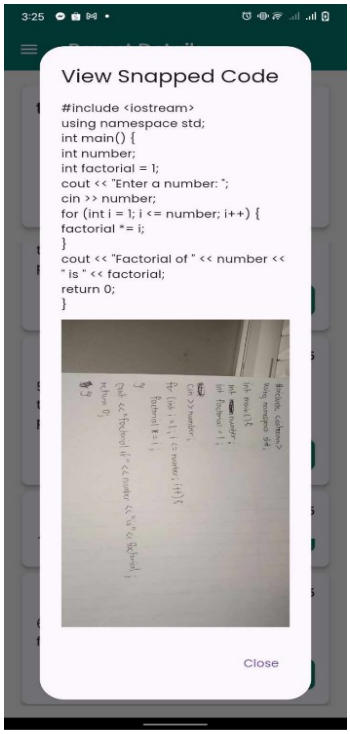
(b)



(c)



(d)



(e)

Figure 3.14. Extracting Handwritten Codes of a Student

During the development of the application, various handwritten C++ codes were tested to determine the application's accuracy in terms of extracting them. Some of it are the images in a, b, c, and e, under Figure 3.14, the app demonstrated high accuracy in extracting codes despite having crossed-out or erased words. However in image d, its performance in extracting special characters and symbols used in programming was lower by about 10%, resulting in its accuracy of 90%. To address this, the code field remains editable, allowing users to correct any missed scans. Additionally, the app supports multiple scans, enabling users to piece together code by scanning it in sections and combining them as needed. Symbols such as semicolons, colons, question marks, exclamation points, quotation marks, parentheses, brackets, braces, operators, underscores, vertical bars, and others were sometimes skipped or incorrectly extracted, indicating a need for further refinement in recognizing special characters critical to C++ syntax.

3.2. Developing an app that reduce the time instructors spend manual checking on handwritten C++ codes, allowing students to receive timely feedback.

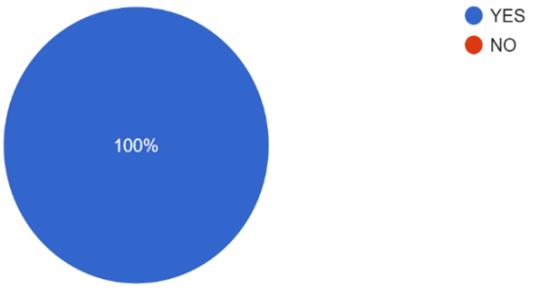


Figure 3.15. Q5 (instructors): Does scoring handwritten coding assessments take a significant amount of your time?

In Figure 3.15, it illustrated that 100% (6 out of 6) of instructors agree that manually checking handwritten coding assessments requires an excessive amount of time. This overwhelming consensus underscores a common frustration among instructors, who must navigate numerous handwritten submissions. The labor-intensive nature of this process can detract from the overall teaching experience, as instructors may feel burdened by the volume of work and the need for meticulous attention to detail.

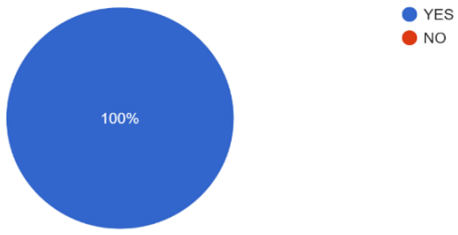


Figure 3.16. Q7 (instructor): Do you think the process of manual checking of handwritten code assessment of students prevents you from giving timely feedback to your students?

In Figure 3.16, it showed that 100% (6 out of 6) of instructors agree that the lengthy grading process directly impacts their ability to provide timely feedback. This delay can have serious implications for students, as timely feedback is essential for their learning and development. It allows students to understand their errors, learn from them, and apply this knowledge to future assignments. The excessive waiting time for feedback can lead to missed critical learning opportunities, resulting in confusion regarding their progress.

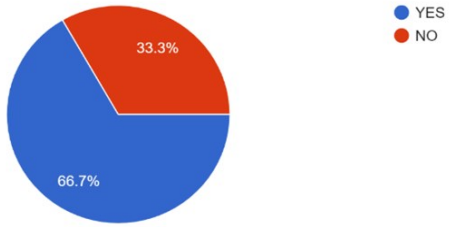


Figure 3.17. Q9 (instructor): Do you believe the delayed return of students' handwritten code assessments affects their learning?

Figure 3.17 revealed that 66.7% (4 out of 6) of instructors believe that delayed feedback significantly impacts students' learning. This majority opinion highlights the crucial role of timely assessment in the educational process.

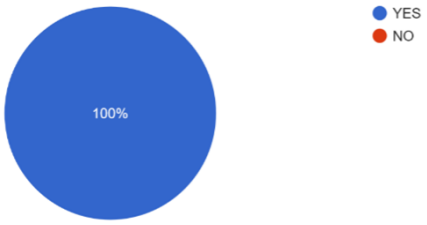


Figure 3.18. Q12 (instructor): Do you think manual checking of handwritten codes takes excessive time compared to checking digital submissions?

Figure 3.18 has supported these findings, showing that all instructors agree that grading handwritten coding assignments takes significantly more time than evaluating digital submissions. This agreement highlights a critical challenge in the current assessment process. Manual grading not only requires a detailed review of each handwritten submission but also demands greater cognitive effort as instructors strive to identify syntax errors, logical mistakes, and formatting issues. In contrast, digital submissions often facilitate a more streamlined grading process.

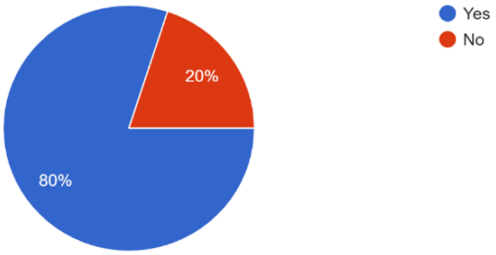


Figure 3.19. Q5 (student): Have you experienced delays in receiving feedback on handwritten coding activities?

Due to these challenges, Figure 3.19 indicates that 16 out of 20 student respondents (80%) have faced delays in receiving feedback on their handwritten assignments, while 4 students (20%) stated they have not. This significant number suggests that feedback on handwritten assignments often takes longer than desired, which can be disappointing for students eager to learn and improve their coding skills.

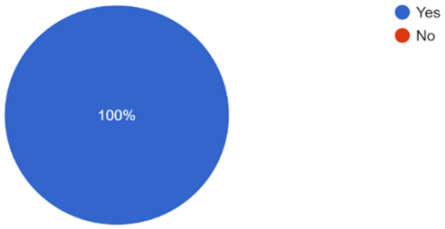
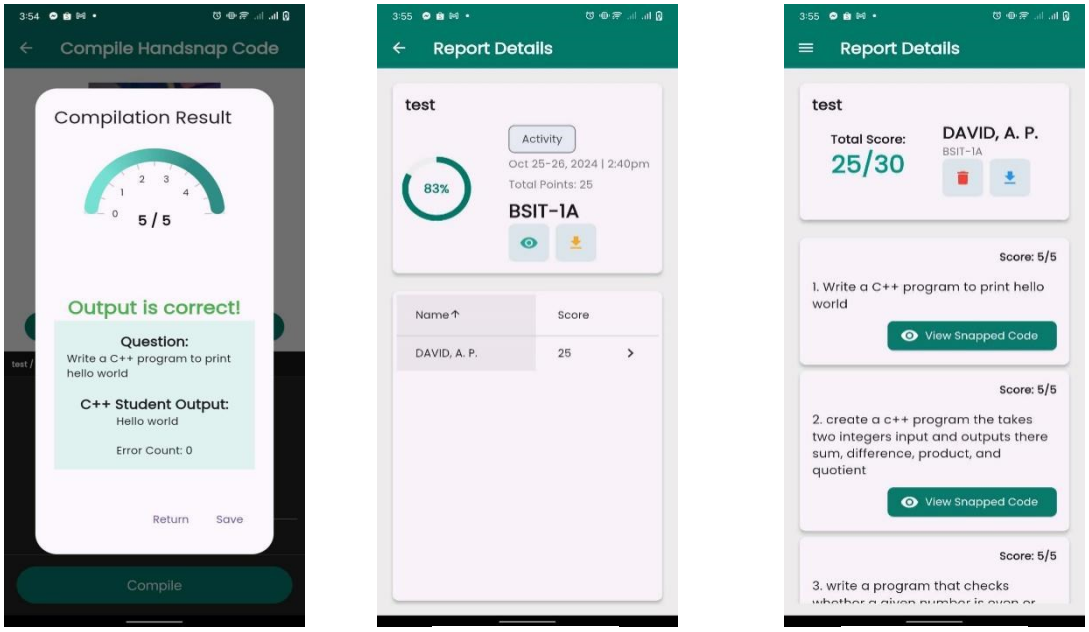


Figure 3.20 Q6 (student): Do you believe that receiving late feedback on your handwritten code affects your ability to learn and improve?

The Figure 3.20 further emphasizes this concern, as all 20 student respondents (100%) agreed that receiving late feedback on their handwritten code affects their ability to learn and improve. This highlights the crucial role of prompt feedback for students. When feedback is delayed, students miss opportunities to understand their mistakes and make corrections before moving on to new material.



(a) (b) (c)

Figure 3.21. User Interface of HandSnap++ that highlights the scoring system.

To address these challenges, the researchers integrated a digital scoring system in their proposed application in addition to debugging handwritten C++ codes, aiming to reduce time spent on manual verification using traditional methods. By utilizing the application, instructors can evaluate students' assessments both during their laboratory activity and after they have completed answers on paper. The system allows teachers to instantly view compiled output for each student's code, making it easier to assess correctness in real time. Under Figure 3.21, Image a illustrates the compiled result of a student's response to an assessment question, while Images b and c show class scores and detailed reports of each student's answers.

This approach solves the challenge of delayed feedback by enabling immediate score entry and record-keeping. After students submit their handwritten answers, they pass through the system, where scores are quickly recorded and instantly visible in the student's app account. This makes it easier for teachers to spend less time on paperwork and more time giving feedback to students, thus maximizing classroom productivity.

3.2. Developing an application that minimizes the loss of papers by recording activity sheets digitally

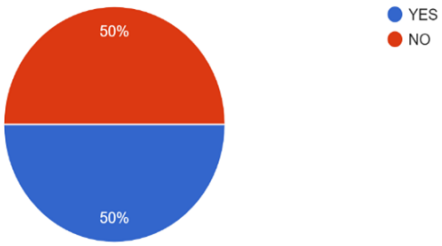


Figure 3.22. Q10 (instructor): Have any students frequently reported losing their handwritten code assessment papers?

The Figure 3.22 shows an even split with instructors, 50% (3 respondents) saying yes and 50% (3 respondents) saying no. This suggests that while some students struggle with keeping track of their activity sheets, others manage to maintain their submissions, reflecting variability in student organization skills.

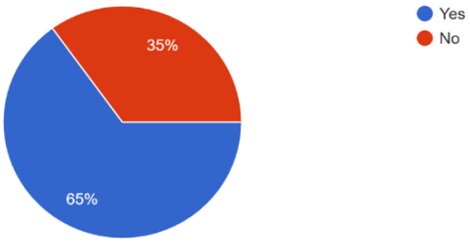


Figure 3.23. Q7 (student): Have you ever misplaced or lost your handwritten coding assignments or quizzes?

In response, Figure 3.23 has shown 13 out of 20 student respondents (65%) reported that they have misplaced or lost their handwritten coding assignments or quizzes, while 7 respondents (35%) stated that they have not. This finding shows that a lot of students faces challenges in keeping track of their written work, which can lead to unnecessary stress and hinder their ability to study effectively.

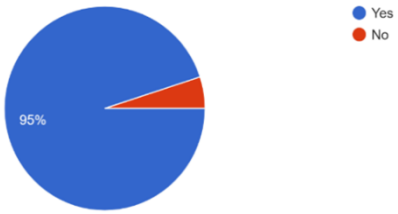
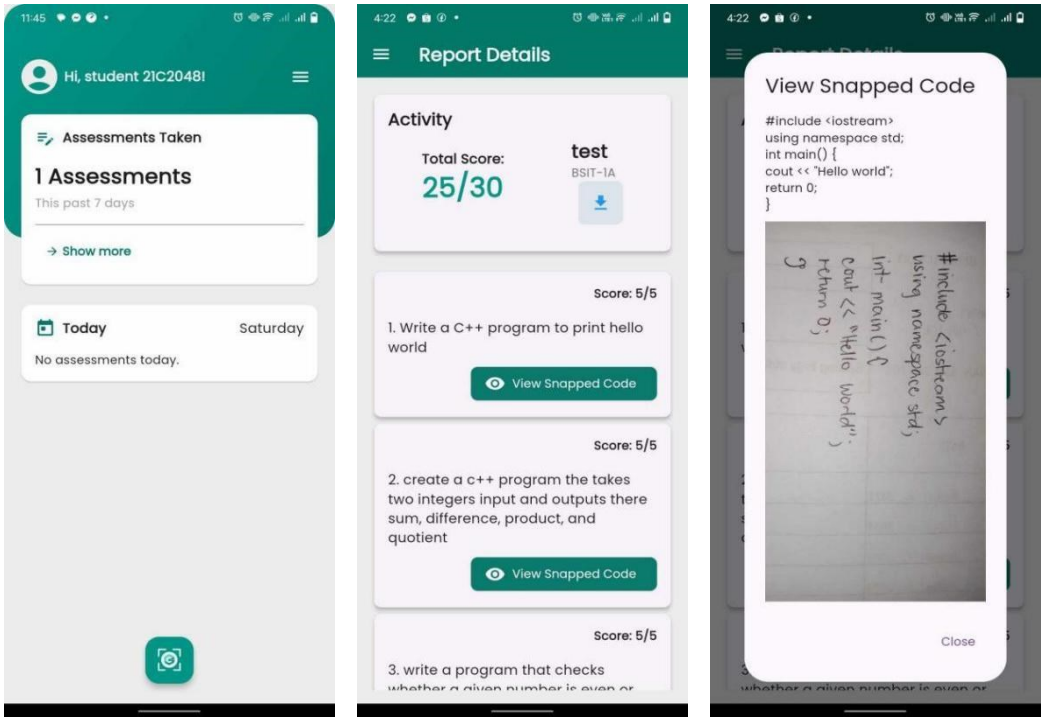


Figure 3.24. Q8 (student): Do you find it harder to review for exams or quizzes without your previous handwritten activity sheets?

In Figure 3.24, where 19 out of 20 student respondents (95%) indicated that they find it harder to review for exams or quizzes without their previous handwritten activity sheets. Only 1 respondent (5%) disagreed. This highlights the importance of having access to past assignments for effective exam preparation. Without these materials, students may struggle to recall important concepts or practices, making it more difficult to perform well in assessments.



(a) (b) (c)

Figure 3.25. User Interface of the HandSnap++ That Highlights the Assessment Report in Student Account

With having different role accounts in the application for instructors and students, as shown in Image a under the Figure 3.25, students will have digital access to their scores and codes (Images b and c). Each student has a digital copy of all their activities and other assessments, which serves as a reliable backup if they lose any paper documents.

3.2. Developing an application that enhances student's familiarity in using compilers, allowing them to adapt to real-world compiler environments and bridge the gap between handwritten code and practical debugging skills

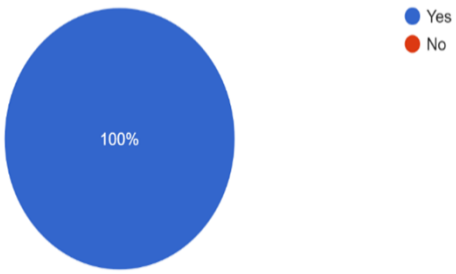


Figure 3.26. Q2 (student): Do you feel that writing code by hand is more difficult than typing it on a computer

Figure 3.26 shows that all 20 student respondents (100%) agreed that writing code by hand is more difficult than typing it on a computer. This highlights the challenges of manual coding, which lacks the immediate feedback and ease of editing that computer-based coding provides.

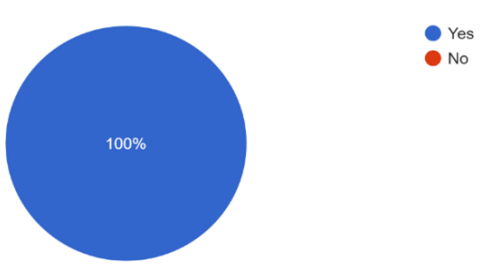


Figure 3.27 Q3 (student): Do you find it more difficult to spot syntax errors when handwriting code compared to using a compiler?

In Figure 3.27, all student respondents also reported (100%) that they find it more challenging to spot syntax errors when handwriting code compared to using a compiler. This indicates that the absence of real-time error detection while writing by hand increases the likelihood of mistakes going unnoticed.

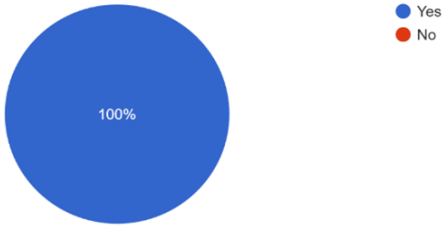


Figure 3.28. Q4 (student): Have you struggled to understand your mistakes in handwritten code without the ability to test it in a compiler?

The Figure 3.28 further supports this concern, as all student respondents (100%) stated they have struggled to understand their mistakes in handwritten code without the ability to test it in a compiler. This lack of testing capabilities makes it difficult for students to learn from their errors, as they do not receive clear guidance on what went wrong or how to fix it.

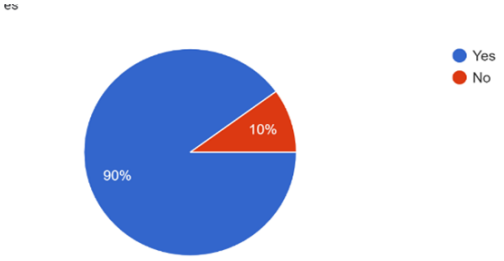


Figure 3.29. Q10 (student): Have you faced difficulties transitioning from handwritten C++ or any programming language code to coding in a compiler environment?

When transitioning from handwritten code to a compiler environment, Figure 3.29 shows that 18 out of 20 student respondents (90%) faced difficulties, while 2 of them (10%) reported no issues. This suggests that the shift from manual coding to using a compiler presents challenges for most students.

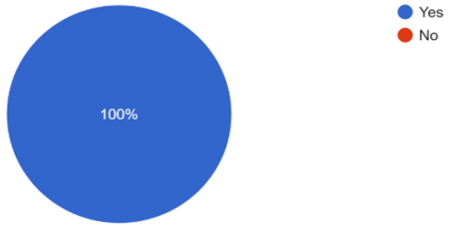


Figure 3.30. Q11 (student): Do you think that more hands-on experience with a compiler would help you debug your code more effectively?

In Figure 3.30, all student respondents (100%) agreed that more hands-on experience with a compiler would help them debug their code more effectively. This reinforces the importance of practical experience in learning programming, as hands-on practice with compilers is crucial for developing coding skills and building confidence in debugging.

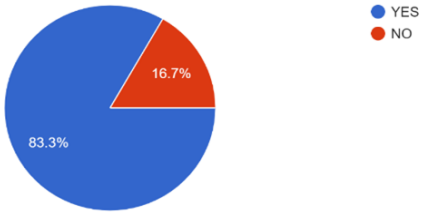
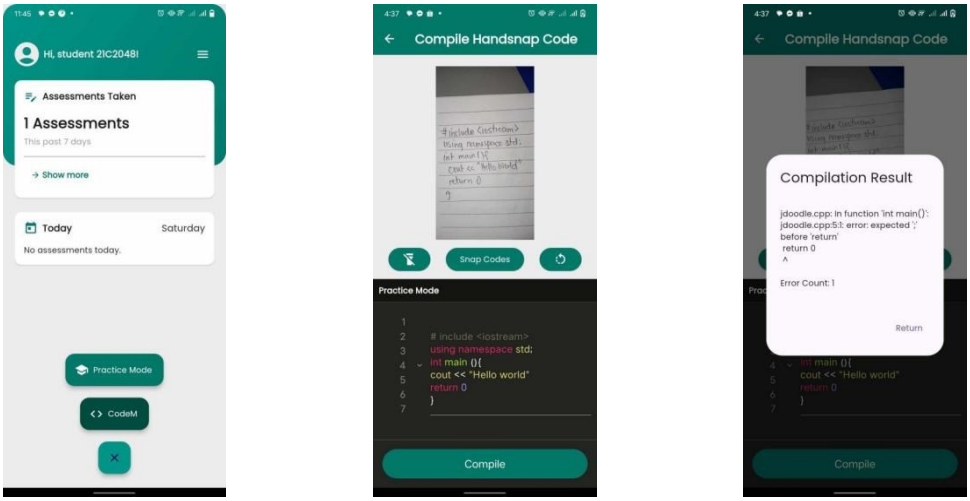


Figure 3.31. Q8 (instructor): Are students struggling to adapt to compilers due to a lack of hands-on experience in the lab?

Correspondingly, Figure 3.31 indicates that 83.3% (5 instructors) believe that insufficient practical experience with compilers limits students' ability to use them effectively, while 16.7% (1 instructor) disagreed. This highlights a critical gap in the learning process, emphasizing that familiarity with compilers is essential for writing and testing code.



(a) (b) (c)

Figure 3.32. User Interface of HandSnap++ Highlighting the Practice Mode in the Student Account

With these responses from respondents, the researchers developed the application which mainly features debugging programming code, particularly C++ language written in papers. This mobile feature assists students who lack experience with compilers by offering a portable solution while utilizing the paper coding still as their learning method. With the app's Practice Mode (as shown in Images a, b, and c, under Figure 3.32), students can gain hands-on experience with compilers as they try and practice their codes, helping them adapt more effectively and overcoming this challenge.

4. DISCUSSION

To assess the application's software quality, the researchers distributed an evaluation form to selected respondents from private higher education institutions and a state university in Region 3, Philippines. The respondents included instructors from the computer studies department and college students enrolled in computer-related programs. Two (2) sets of evaluation forms were used: one for instructors and another for students, ensuring that feedback was gathered from both perspectives of respondents.

Table 3.1. Overall mean for all the categories of ISO 25010 evaluation from All Respondents.

CATEGORY	Mean	Descriptive Interpretation
Functional Suitability	3.87	Strongly Agree
Performance Efficiency	3.45	Strongly Agree
Compatibility	3.48	Strongly Agree
Capability/Usability	3.81	Strongly Agree
Reliability	3.69	Strongly Agree
Security	3.72	Strongly Agree
Maintainability	3.69	Strongly Agree
Portability/Flexibility	3.80	Strongly Agree
Overall Mean for Category	**3.69**	**Strongly Agree**

The table 3.1 overall evaluation from all respondents showed a grand mean of 3.69 across all categories, indicating strong agreement from both instructors and students that the application meets the ISO 25010 standards. This suggests that the application has successfully passed the quality assessment criteria set by the ISO 25010 model.

5. CONCLUSION

Therefore, the researchers conclude that the HandSnap++ can serve as an alternative tool for compiling handwritten C++ programming code, especially in situations where there is a shortage of computer lab facilities. Using OCR in HandSnap++ improves the accuracy of evaluating handwritten C++ codes, helping instructors identify subtle errors that may be missed in manual reviews. The app provided a digital scoring system when checking handwritten codes, modernizing manual methods, and it enabled the instructors to have timely feedback for students. HandSnap++ allows instructors to store students' scores and scanned answers, offering convenience as students can access their assessments through their accounts. By using the app, students gain hands-on experience with compilers, helping them adapt to this while enhancing their practical debugging skills. The results and overall system evaluation rating, based on ISO 25010 standards, indicate that respondents strongly agree the system meets software quality standards.

Acknowledgments: We, Clarq Anderson P. Arias, Juliana Arla S. Paguinto, Rhea Joy M. Pangkubit, and Stephen Felipin S. Miranda, would like to express our sincere gratitude to those who have contributed to the successful completion of this capstone project. Their support and assistance have been invaluable, and we are truly thankful for their dedication.

First and foremost, we would like to give thanks to the Lord, our Almighty God, for giving us the opportunity to participate in the study and learn so much. He blessed us with knowledge, motivation, strength, and spiritual support to see the study through to its completion.

Next, we would like to express our heartfelt gratitude to our research adviser, Mr. John Carlo L. Gamboa, for his invaluable guidance and advice for us throughout this journey. Our appreciation also goes to Mr. Arzel P. Pinpin, our program chair in Bachelor of Science in Information Technology (BSIT) – Institute of Computer Studies (ICS), for his feedback throughout the study, and to Dr. Marvin O, Mallari, Dean of the School of Engineering, Computer and Library Studies (SECLS) in Holy Cross College (HCC) and our capstone professor, for his exceptional support for us. We sincerely appreciate these as they have contributed to the success of our research. This would not have been possible without their support and expertise.

Special thanks also to our grammarian, Arden C. Cabugos, for helping us improve our paper as well as to our statistician, Mr. Danilo M. Guzman Jr., who provided a reliable tally of the data we have obtained from our respondents. We would also like to extend our sincere thanks to our research questionnaire validators, Ms. Romalyn H. Gomez, research expert, as well as our IT experts, Mr. Rommel S. Malinao and Mr. Charles Reynald S. Tubil. Their willingness to collaborate significantly contributed to the success of this journey.

Finally, we are extremely grateful to our family, friends, and colleagues for their unwavering support and love for us. We are deeply grateful for our parents' sacrifices and prayers, which have enabled us to advance our education and prepare for our future. Words cannot describe how grateful we are to our friends and colleagues who helped us throughout this journey by encouraging us and giving us the foundation we needed for the paper.

References

About the Commision. (n.d.). Retrieved June 11, 2024, from Commission on Higher Education: https://ched.gov.ph/about-us/

Acot, M., Camacho, K., Guevara, M., & Lapid, N. (2022, August). BayBAIYIN: A Real-Time OCR Model for Handwritten Pre-Colonial Text - AIM Aboitiz School of Innovation, Technology, & Entrepreneurship. Retrieved from Aboiitiz School of Innovation, Technology, & Entrepreneurship.: https://asite.aim.edu/data_science/baybaiyin-a-real-time-ocr-model-for-handwritten-pre-colonial-text/

Anad, A., Rastogi, A. A., Chadichal, R. A., Surana, A., G, S., & R, L. N. (2023). Handwritten text recognition and conversion to speech. International Journal For Research In Applied Science And Engineering Technology, 11(6), 3904-3914. Retrieved from International Journal for Research in Applied Science & Engineering Technology (IJRASET): https://doi.org/10.22214/ijraset.2023.54317

Ang, G. D., Chong, D. G., Lin, J. S., & Gendrano, M. (2022). Extracting Medication Information from Typewritten Philippine Medical Prescriptions Using Optical Character Recognition (OCR) and Named Entity Recognition (NER). Research Congress Proceedings 2022, 10. Retrieved from Research Congress Proceedings 2022 (Vol. 10): https://www.dlsu.edu.ph/wp-content/uploads/pdf/conferences/research-congress-proceedings/2022/HCT-09.pdf

Awati, R. (2024, January). Retrieved from TechTarget: https://www.techtarget.com/whatis/definition/beta-test

Barphe, S. S., Lokare, V. T., Sanjay, S. R., & Arvind , K. W. (2021, December 17). Retrieved from Springer Link: https://link.springer.com/chapter/10.1007/978-981-16-6369-7_56

Britton, J. (2021, May 6). What Is ISO 25010? Retrieved from PERFORCE: https://www.perforce.com/blog/qac/what-is-iso-25010#:~:text=ISO%2025010%20is%20a%20software%20quality%20standard%20that%20provides

C++ Introduction. (n.d.). Retrieved June 2024, from W3Schools: https://www.w3schools.com/cpp/cpp_intro.asp

Casillano, N. B. (2019). Utilization of Optical Character Recognition (OCR) in the development of a Number System Converter Application. doi:10.17485/ijst/2019/v12i16/137794

Censoro, K. C. (2021). Document management system using optical character recognition, clustering, watermarking and QR coding algorithms. Retrieved from BAHÁNDÌAN, Institutional Repository of Central Philippine University: https://repository.cpu.edu.ph/handle/20.500.12852/1519

CHED. (2015). Retrieved from https://ched.gov.ph/2015-ched-memorandum-orders/

Cueva, D. (2023, March 7). List of IT Courses in the Philippines: 2023 Guide. Retrieved from TOPNOTCHER: https://topnotcher.ph/list-of-it-courses-in-the-philippines/

Demir, K., & Akpinar, E. (2018). The Effect of Mobile Learning Applications on Students' Academic Achievement and Attitudes toward Mobile Learning. Malaysian Online Journal of Educational Technology, 6(2), 48-59. Retrieved from Malaysian Online Journal of Educational Technology, 6(2), 48–59.: https://eric.ed.gov/?id=EJ1174817

Drigas, A., & Angelidakis, P. (2017, May 22). Mobile Applications within Education: An Overview of Application Paradigms in Specific Categories. International Journal of Interactive MObile Technologies, 11(4), 17-29. Retrieved from International Journal of Interactive Mobile Technologies, 11(4), 17.: https://doi.org/10.3991/ijim.v11i4.6589

Joshi, K., Patil, S., Patil, S., Patil, S., Patil, S., & Patil, T. (2023). Handwritten Text Recognition from Image. International Journal for Research in Applied Science & Engineering Technology, 11(6), 1528-1530. doi:10.22214/ijraset.2023.53364

Karthikeyan, G., Bharanidharan, G., Jeevanandha, D., & Balaji, B. (2022). Text Recognition Images using OCR. International Journal of Progressive Research in Science and Engineering, 3(5). Retrieved from International Journal of Progressive Research in Science and Engineering, 3(05).: https://journal.ijprse.com/index.php/ijprse/article/view/563/533

Kumar, A., Singh, P., & Lata, K. (2023, April 20). Retrieved from Ieee Xplore: https://ieeexplore.ieee.org/abstract/document/10100213

Linaac Jr., R. E., Tan, J. L., Rosal, B. S., & Arriola Jr., C. C. (2019). Course Crediting System Using Optical Character Recognition. SMCC Higher Education Research Journal (Computing Journal), 2(1). Retrieved from SMCC Higher Education Research Journal (Computing Journal), 2(1).: https://ejournals.ph/article.php?id=15532

Lutkevich, B. (2020). Retrieved from TechTarget: https://www.techtarget.com/searchsoftwarequality/definition/Scrum

Ogunbiyi, E. (2021). ONLINE COMPILER, IDE, INTERPRETER, AND REPL FOR MULTIPLE PROGRAMMING LANGUAGES. Retrieved from ResearchGate: https://www.researchgate.net/publication/370074438_ONLINE_COMPILER_IDE_INTERPRETER_AND_REPL_FOR_MULTIPLE_PROGRAMMING_LANGUAGES

Ortiz, G. (2023, Apirl 29). The Benefits of Learning C++ as Your First Programming Language. Retrieved from Medium: https://medium.com/@Georgeo90/the-benefits-of-learning-c-as-your-first-programming-language-b9adc9e168ef

Pino, R., Mendoza, R., & Sambayan, R. (2021). Block-Level Optical Character Recognition System For Automatic Transliterations Of Baybayin Texts Using Support Vector Machine. Philippine journal of science, 151(1). doi:10.56899/151.01.23

Preethi. (2022, August 17). Why (and how) I write code with pencil and paper | CSS-Tricks. Retrieved from CSS-Tricks: https://css-tricks.com/why-and-how-i-write-code-with-pencil-and-paper/

Semilof, M. (2021, October). Retrieved from TechTarget: https://www.techtarget.com/searchsoftwarequality/definition/alpha-testing

Sinanaj, G., Purva , K., Amiri, E., & Mahilaj, A. (2022, , June 15). Evaluation of Online Integrated Development Environments for Educational Use: A Comparative Study of Performance and Service Offerings. Journal of Computer Science and Technology, 38. Retrieved from https://doi.org/10.1007/s11390-022-9834-6

Sreekumar, D. (2023, March 23). Retrieved from Researcher Life: https://researcher.life/blog/article/what-is-quantitative-research-types-and-examples/

Technology and Livelihood Education (TLE) and Technical-Vocational-Livelihood (TVL) Track. (n.d.). Retrieved from Department of Education: https://www.deped.gov.ph/k-to-12/about/k-to-12-basic-education-curriculum/technology-and-livelihood-education-tle-and-technical-vocational-livelihood-tvl-track/

TESDA Computer Course - IT Training Centers, Fees, Duration and Where to Enroll. (n.d.). Retrieved from About Philippines: https://www.mypilipinas.com/tesda-computer-course.html

Vaithilingam, P., Markel, J. M., & Guo, P. J. (2020). Papercode: Generating Paper-Based User Interfaces for Code Review, Annotation, and Teaching. Retrieved from UIST '20 Adjunct.: https://doi.org/10.1145/3379350.3416191