© Universiti Tun Hussein Onn Malaysia Publisher's Office

IJIE

http://penerbit.uthm.edu.my/ojs/index.php/ijie
ISSN : 2229-838X e-ISSN : 2600-7916

The International Journal of Integrated Engineering

Exam Marks Summation App Using Tesseract OCR in Python

Kalaimathi Saravanan[1], Chang Choon Chew[1], Kim Gaik Tay[1*], Sei Long Kek[2], Audrey Huong[1]

[1]Department of Electronic Engineering, Faculty of Electrical and Electronic, Universiti Tun Hussein Onn Malaysia, Parit Raja, 86400, MALAYSIA

[2]Department of Mathematics and Statistics, Faculty of Applied Science and Technology, Universiti Tun Hussein Onn Malaysia, Parit Raja, 86400, MALAYSIA

*Corressponding Author

DOI: https://doi.org/10.30880/ijie.2022.14.03.011
Received 26 December 2021; Accepted 04 June 2022; Available online 20 June 2022

Abstract: There are no tools to auto grade subjective questions and no auto summations app to date. As a result, examiners need to mark subjective answer scripts manually and they might overlook some marks while summing up subjective answer scripts scores obtained by students. Therefore, this study aims to develop an exam mark summation app to ease miscalculation problems. Tesseract OCR was implemented in this Marks Summation App as a platform to scan the multi-page handwritten marks using pen tip above a 0.5mm size in red, blue, and black to recognise the marks wrote by the examiners. The captured marks images will be uploaded to server for image processing by promptly executing the python script in the server. The extraction of numbers from the processed image will sum up the scores to determine each candidate's final score. It is recommendable to scan the original materials to produce more accurate accuracy. Out of 118 numbers from 21 testing samples, 88 numbers are correctly recognised, whereas 30 are not recognised correctly. Thus, the accuracy of the developed exam mark summation app is 74.58%. The app's performance is more superior than car plate recognition result in the field. In the future, we hope to increase the app's recognition accuracy by exploring deep learning in recognizing handwritten numbers.

Keywords: Summation, tesseract, OCR, python, app

1. Introduction

Assessment is a vital evaluation process in student learning [1]. According to researchers [2], assessment sets the target more influentially than any syllabus or course and is deeply experienced by students. It is a criterion-referenced of outcome-based teaching and learning to check if the outcomes have been attained [3]. It can ensure educational provider preserves academic standards [4]. Assessment is designed to measure a person's knowledge, skill, ability, or physical fitness. Assessment can be divided into formative and summative. Formative assessment is an ongoing evaluation at each stage to ensure immediate feedback can be obtained after the assessment. Thus, students' learning performance can be monitored at each stage and the teaching process can be improved. Summative assessment is conducted at the end of the learning.

Assessment in higher education consists of both formative and summative assessments. Formative assessment can be conducted through quizzes, tests, assignments and projects, whereas summative is the final examination during each semester. The quiz, test and final examination assessment can be evaluated using multiple-choice, true and false, matching, and subjective questions. Subjective questions may include short answer questions, structure questions, long

**Corresponding author: tay@uthm.edu.my* answers or essays. For assignments and projects, peer assessment [5], self-assessment and group assessment may be included.

An examination or test can be carried out verbally, handwritten or on a computer system [6]. There are various format styles and difficulty levels in an examination, and it may be conducted in a formal and informal mode. An examiner will examine an examination or test script written by a candidate. Evaluators or teachers must evaluate the assessment or exam papers based on marking schemes to give the students marks. However, marking an answering script in the current days is becoming unfortunate a heavy load for the lecturers, especially with poor handwriting or pencil writing [7]. On top of that, evaluation of subjective test or exam scripts is tedious and time-consuming [8] [9].

To avoid tedious marking, objective questions with an auto-grading system done by optical mark recognition (OMR) or most e-learning platforms such as google classroom can be a good option. However, making an excellent objective question with their options is not as easy as setting up subjective questions.

As a solution for marking subjective answer scripts, Rao [7] developed a fair and innovative idea of evaluating the script in a computerized format. They used computer vision and convolution neural networks to convert text from handwritten scripts. Then they applied natural language processing (NLP) to tokenize the words required and then checked with the words already uploaded by the lecturers. Hence, the evaluation correctly could have been done, and the burden will become lesser.

NLP-based automatic answer script evaluation projects were done by [8], [9]. They used Pytesseract to extract text from the answer script and then denoised it to improve accuracy. Later, a keyword-based technique from NLP was used to generate a short and precise summary. Various similarities between summarized extracted text and the marking schemes were measured using Cosine, Jaccard, Bigram, and Synonym to generate the last marks.

A study in [10] introduced an automatic quick short answer grading system using MaLSTM deep network and sense vectors containing students' and model answers using Mohler and CU-NLP datasets. They obtained 0.949 Pearson's correlation and 0.084 root mean square error.

A simple short question grading system was developed by [11] using short text and augmented similarities. The system takes the short scheme answer and short student answer as input and computes semantic similarities to give a score.

SergVoloshyn [12] proposed the Android OCR application based on Tesseract using the Tesseract OCR engine of Tesseract 3 to recognize character patterns. The Tesseract has Unicode (UTF-8) can recognize over 100 languages.

Bautista and Comendador [13] used Image Madisk to enhance the quality of the scanned grade sheets and optical character recognition (OCR) engine in Tesseract to automatically recognise and storing students' scores into respective database.

Researchers [14] modified LeNet-5 network based on deep learning to identify street view house number (SVHN) from SVHN public dataset. They obtained 92.32% identification rate after a training of 7 hour and 24 minutes. While in another study by [15], they obtained 60% accuracies for car plate number recognition respectively using open CV and Tesseract OCR.

As far as we know, there are no available tools to mark subjective questions and sum the total marks automatically to date. In addition, there are also no developed tools used to detect the handwritten marks and do the summation. Therefore, the examiners need to mark manually and sum up the marks. They might overlook the marks of the exam scripts while summing up individual graded marks, leading to a miscalculation in grading the overall exam mark. To recheck the exam script, the students have to pay a specific fee to crosscheck the answers, and it causes a double load task and time consumption for an examiner.

Therefore, to overcome this issue, an android-based Marks Summation App that can recognise pen tip above a 0.5mm size of handwritten marks in red, blue, and black was developed to perform the auto summation of the handwritten marks. The handwritten marks can be integer or floating-point numbers. A set of exam scripts with a maximum number of 10 pages can be captured for each submission. An unlimited set of exam scripts can be submitted to the Marks Summation App. The App requires the Tesseract OCR library to recognise the handwritten marks after preprocessing the captured images. The recognised handwritten marks are then sum up using Python. A CSV file that contains the recognised and total mark is generated

2. Methodology

The exam summation App was built using Python, Visual Studio Code, React Native, JavaScript Object Notation (JSON), PHP and MySQL.

Figure 1 shows the flowchart of the application process of the Marks Summation App. As shown in Figure.1, the process begins by capturing the marks of an exam script document as an image. The captured image is then uploaded to the server via a web service. The server runs a python script to do the image preprocessing process. The first image processing step is to invert the scanned image to a black background image. The second step is converting the RGB image to a grayscale image as a grayscale image is one layer from 0-255 values which forms 256 levels or 8 bits, while RGB image has three layers of 256 levels to represent red, green and blue colours. Thus, a grayscale image simplifies the image processing algorithm and saves computation times. Most of the image processing is sufficient to be processed with a grayscale image.

Once the grayscale image conversion is done, the image undergoes the third image processing, thresholding the image using the OTSU algorithm to convert the grayscale image to a binary image (black and white colour only) for morphological transformation later.

Morphological operation of dilation is carried out to fill the hole and broken areas. Dilation will thicken the numbers and enhance the readability of the numbers in the image of the exam script document. Next, the dilated image undergoes another morphological process which is morphological closing. Morphological closing is dilation followed by erosion. It is beneficial if there are any dots present on the numbers of the image, it will close the dots. This will increase the efficiency of the Tesseract OCR library to recognise the numbers on the image.

The last step of the image processing is inverting the morphological closed image into a white background and the numbers on the image are black in colour. Furthermore, the inverted image is cropped to the right-hand side of the image. It will ensure the App reads only the numbers that are written as marks instead of the answers on the exam script document.

Finally, the Tesseract OCR library tries to recognise the numbers on the image. Suppose the Tesseract OCR cannot recognise the numbers. It will redirect us to the beginning stage of the process, scanning the exam script document or proceeds to the summation of the recognised numbers by using Python. The summed marks are displayed to the end-users of the mobile application. A CSV file is generated and stored in the server and database. The users may choose the option to download the generated CSV file containing the marks anytime.

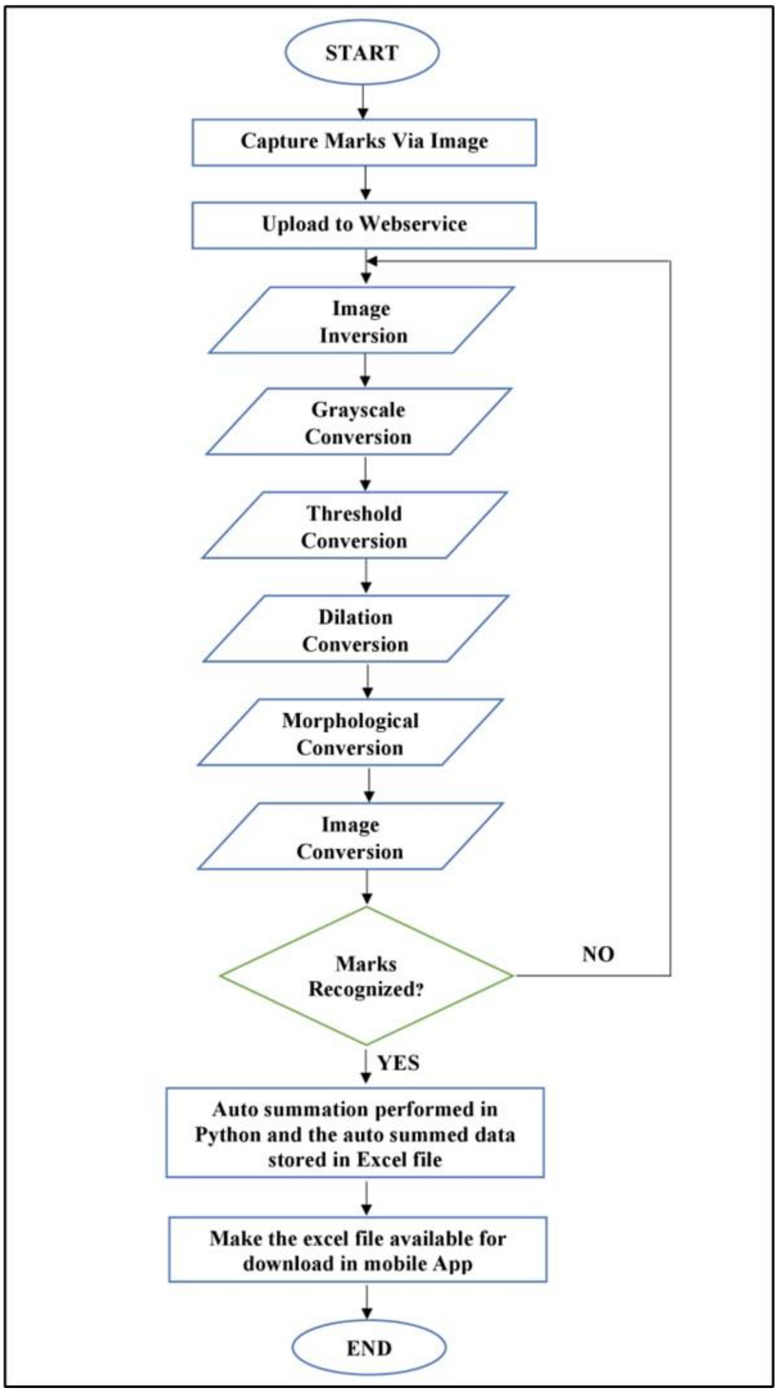


Fig. 1 – Flowchart of application process

Figure 2 shows the flowchart of the Mark Summation Application Process. When the users launch the App, the splash page (introduction page) executes. The splash page has a timeout of 3 seconds. After the timeout, the login page is loaded. Users can then login to their account or sign up for an account by proceeding to the sign-up page. If the login is unsuccessful, users will be redirected to the login page. If the login is successful, users will be redirected to the dashboard page.

Users can then capture a maximum of 10 images at a time of the exam mark sheet on the scanner page and press the submit button for submission. Upon pressing the submit button, the captured images are uploaded to the server for image processing, as described in Figure 1. After the image processing, Tesseract OCR will try to recognise the numbers in the image. If the recognition of numbers is not successful, then users will be redirected to the scanner page.

If the recognition of numbers is successful, the summation process takes place. After the summation is completed, the total is displayed on the App later. Later, a CSV file is generated and stored on the server. As a reference, the CSV file name is stored on the database to allow users to download their files. Finally, the summation result is displayed on the download page and a download link to the CSV file from the App. The process is summarised in the form of a flowchart in Figure 2.

The UI design pages such as splash, login, sign up, forget the password, dashboard, scanner, download file and document pages were designed with programming codes using Visual Studio Code as the text editor. The programming codes are compiled and converted into UI designs when the App is executed.

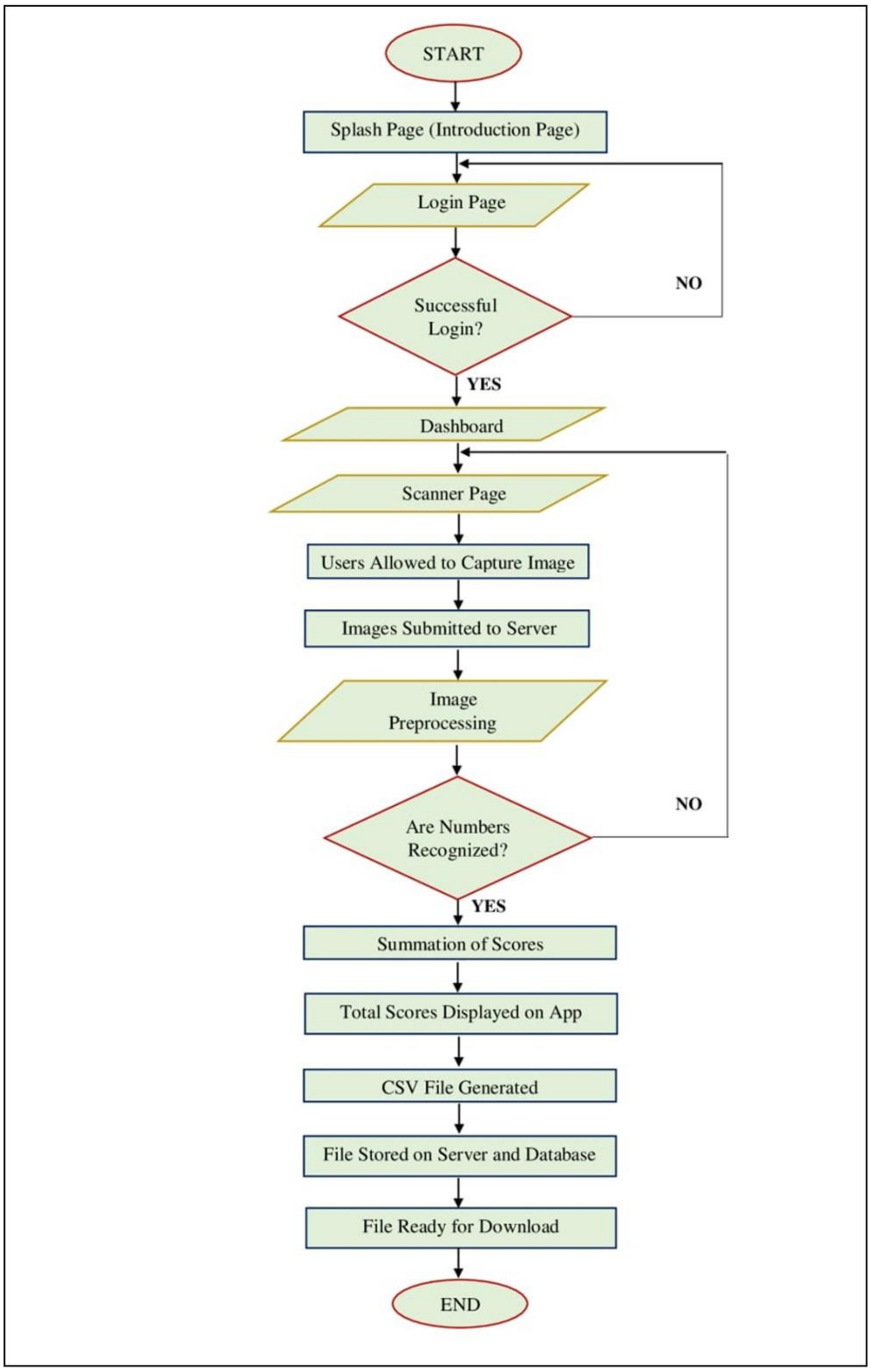


Fig. 2 – Flowchart of Exam Mark Summation Application process flow

The Marks Summation App was built using React Native, PHP and My SQL as shown in Figure 3. React Native acts as frontend framework. Whenever the App needs to perform create, read, update or delete (CRUD) operations to the database, the App sends the data in JSON text format to the server via an API call to PHP which works as backend.

PHP then passes data over MySQL queries. After performing the necessary CRUD operations in the database, the result is returned to the PHP backend. Multiple CRUD operations to the database or execution of the Tesseract OCR library python script for image processing and marks summation can be performed. Finally, the data is converted into JSON text format and will be returned to the API as a callback. The data is then displayed in the Marks Summation App according to the requested action.

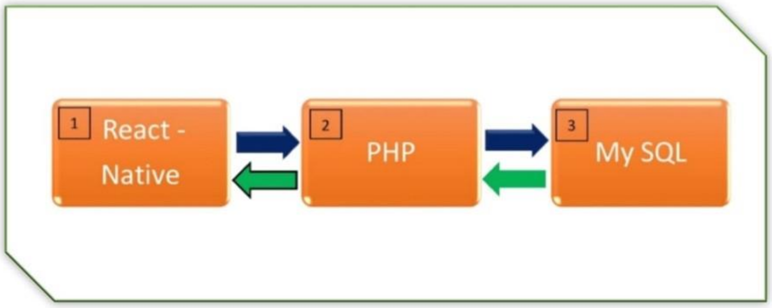


Fig. 3 – Block diagram of CRUD operation flow in fetching data of user

3. Results and Discussion

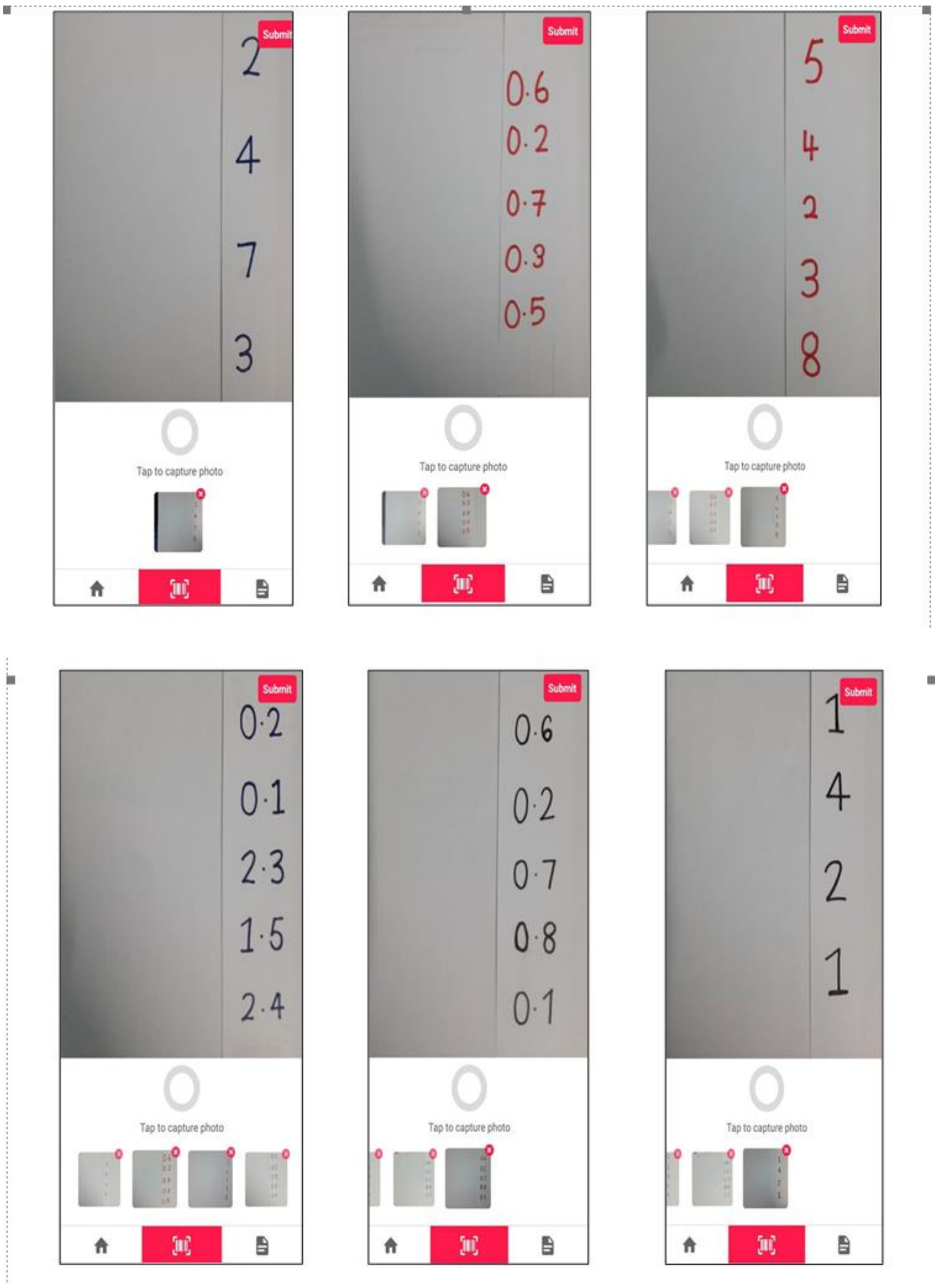
20 single-page testing and one multi-page testing were conducted. Figure 3 shows the multiple images of integer and floating-point numbers in red, blue and black colour.

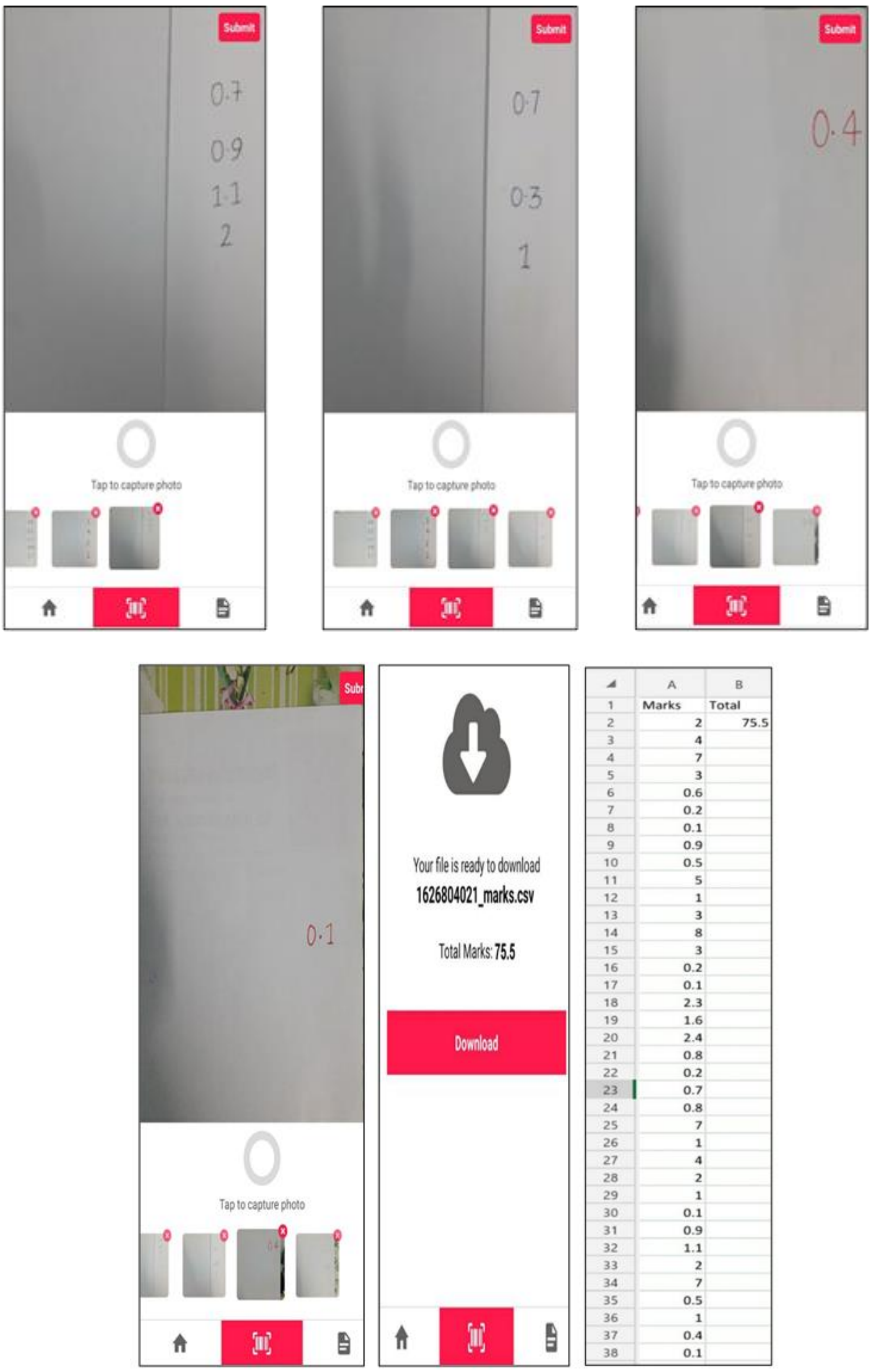
Figure 4 shows that the integer and floating-point numbers in red, blue, and black have a thin, normal, and bold font style. In the first captured image, the integer numbers written in a blue colour pen are 2, 4, 7 and 3 have been recognised correctly as shown in the CSV file from cell A2 to A5. The second captured image displays the floating-point numbers written in red colour pen are 0.6, 0.2, 0.7, 0.3 and 0.5 are not recognised accurately as per the recognized values 0.6, 0.2, 0.1, 0.9 and 0.5 of cell A6 to A10. In the third image, the integer numbers written in red colour are 5, 4, 2, 3 and 8 are not recognised accurately as per the recognised values 5, 1, 3, 8 and 3 of cell A11 to A15.

The fourth image shows that the floating-point numbers 0.2, 0.1, 2.3, 1.5 and 2.4 were written in blue colour pen. According to the cell A16 to A20, the values are recognised as 0.2, 0.1, 2.3, 1.6 and 2.4. Meanwhile, the floating-point numbers in the fifth image shows the recognised numbers in cell A21 to A25 as 0.8, 0.2, 0.7, 0.8 and 7 instead of 0.6, 0.2, 0.7, 0.8 and 0.1. The sixth image displays that the integer numbers 1, 4, 2 and 1 were written in black color pen and recognised correctly as per cell A26 to A29. The seventh image shows the values of cell A30 to A33 as 0.1, 0.9, 1.1 and 2 instead of 0.7, 0.9, 1.1 and 2.

As for the eighth image, the written numbers are 0.7, 0.3 and 1, but the Tesseract OCR recognised it as 7, 0.5 and 1. The ninth and tenth image numbers are written as 0.4 and 0.1, respectively and the values are correctly recognised. The summation of the recognised values is given as 75.5. Based on the overall numbers of 37, the correctly recognised numbers are 27. The accuracy of this testing is 72.97%.

Out of 118 numbers from 21 testing samples, 88 numbers are correctly recognised, whereas 30 are not recognised correctly. Thus, the accuracy is obtained as 74.58%. Our results outperformed reported work in [14-15].


Submit
2
4
7
3
Tap to capture photo
Submit
0.6
0.2
0.7
0.3
0.5
Tap to capture photo
Submit
5
4
2
3
8
Tap to capture photo
Submit
0.2
0.1
2.3
1.5
2.4
Tap to capture photo
Submit
0.6
0.2
0.7
0.8
0.1
Tap to capture photo
Submit
1
4
2
1
Tap to capture photo



Your file is ready to download

1626804021_marks.csv

Total Marks: 75.5

	A	B
1	Marks	Total
2	2	75.5
3	4	
4	7	
5	3	
6	0.6	
7	0.2	
8	0.1	
9	0.9	
10	0.5	
11	5	
12	1	
13	3	
14	8	
15	3	
16	0.2	
17	0.1	
18	2.3	
19	1.6	
20	2.4	
21	0.8	
22	0.2	
23	0.7	
24	0.8	
25	7	
26	1	
27	4	
28	2	
29	1	
30	0.1	
31	0.9	
32	1.1	
33	2	
34	7	
35	0.5	
36	1	
37	0.4	
38	0.1	

Fig. 4 – Multi page testing sample - integer and floating-point numbers in red, blue and black colour

4. Conclusion

This study managed to create an android based App for the examiners to avoid miscalculating marks during the summation of scores achieved by students using Tesseract OCR in Python. The testing accuracy obtained is 74.58%, and it is recommendable to scan the original materials to perform more accurate results. This result is better than existing works in the field. However, there are still some rooms to improve the accuracy of the App. Thus, future work could implement the Tensor Flow or deep learning to increase the efficiency in recognising handwritten numbers.

Acknowledgement

The authors would like to thank the Ministry of Education Malaysia for supporting this research under the Fundamental Research Grant Scheme FRGS/1/2018/TK04/UTHM/02/13 and partially sponsored by Universiti Tun Hussein Onn Malaysia.

References

[1] Boud, D. and Falchikov, N. (2007). Introduction: Assessment for the longer term. In Boud, D. & Falchikov, N. (Eds.), Rethinking assessment in higher education: learning for the longer term. (pp.3-13). London: Routledge.

[2] Thomas, G., Martin, D., & Pleasants, K. (2011). Using self- and peer-assessment to enhance students' future-learning in higher education. J. Univ. Teach. Learn. Pract., 8, 1-17.

[3] Biggs, J., Tang, C. (2011). Teaching for quality learning at university. UK: Open University Press

[4] Dunn, L., Morgan, C. O'Reilly, M., & Parry, S. (2004). The student assessment handbook. New York: Routledge Falmer.

[5] Kvale, S. (2007). Contradictions of assessment for learning in institutions of higher learning. In Boud, D. & Falchikov, N. (Eds.), Rethinking assessment in higher education: learning for the longer term (pp. 57–71). London: Routledge.

[6] Mohler, M., & Mihalcea, R. (2009). Text-to-text semantic similarity for automatic short answer grading. EACL 2009 - 12th Conf. Eur. Chapter Assoc. Comput. Linguist. Proc., 567–575.

[7] Venkateshwara Rao, M., Sri Harshitha, I., Sukruthi, Y., & Sudharshan, T. (2020). Automatic answer script evaluator. Int. J. Sci. Technol. Res., 9, 4023–4027.

[8] Rahman, Md. M., & Siddiqui, F. H. (2018). NLP-based automatic answer script evaluation. DUET J., 4. 35-42.

[9] Rahman, M., & Akter, F. (2019). An automated approach for answer script evaluation using Natural Language Processing, 9, 39–47.

[10] Tulu, C. N., Ozkaya, O, & Orhan, U. (2021). Automatic short answer grading system with SemSpace vense vectors and MaLSTM. IEEE Access, 9, 19270-19280.

[11] Sultan, M. A., Salazar, C., & Sumner, T. (2016). Fast and easy short answer grading with high accuracy. 2016 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. NAACL HLT 2016, 1070–1075.

[12] Voloshyn, S. Android OCR application based on Tesseract - CodeProject. Retrieved from web https://www.codeproject.com/Articles/1275580/Android-OCR-Application-Based-on-Tesseract.

[13] Bautista, M. M., & Comendador, B. E. V. (2016). Adoption of an open source optical character recognition (OCR) for Database Buildup of the Students' Scholastic Records. Int. J. Inf. Electron. Eng. 6, 206–209.

[14] Yang, H. & Yao H. (2019). Street house number identification based on deep learning. International Journal of Advanced Network, Monitoring and Controls, 4, 47-52.

[15] Agbemenu, A. S., Yankey, J. & Addo, E. O. (2018). An automatic number plate recognition system using OpenCV and Tesseract OCR Engine. International Journal of Computer Applications. 180, 1-5.