Journal of Applied Informatics and Computing (JAIC)
Vol.9, No.4, August 2025, pp. 1383~1393
e-ISSN: 2548-6861

Esscore: An OCR-Based Android App for Scoring Short Handwritten Answer Using Levenshtein Distance

I Kadek Krisna Apriana Dwi Guna [1]*, I Made Agus Dwi Suarjaya [2]*, Ni Kadek Dwi Rusjayanthi [3]*
* Teknologi Informasi, Universitas Udayana
krisnaapriana@student.unud.ac.id [1], agussuarjaya@it.unud.ac.id [2], dwi.rusjayanthi@unud.ac.id [3]

Article Info

Article history:

Received 2025-06-02
Revised 2025-07-01
Accepted 2025-07-03

Keyword:

Black Box,
EasyOCR,
Levenshtein Distance,
Short Handwritten Answer,
System Usability Scale.

ABSTRACT

Manual evaluation of short answer tests is time-consuming and prone to subjectivity. This study presents Esscore, an Android-based application that automates the scoring of handwritten short answers using EasyOCR and the Levenshtein Distance algorithm. EasyOCR extracts text from student answers image, while Levenshtein Distance measures similarity against predefined answer keys, allowing tolerance for varied correct responses. The system was tested on 350 student's handwritten answers, achieving 95.7% accuracy. Functional testing using 14 black box scenarios showed all features operated correctly without failure. A usability test conducted with the SUS method produced a score of 76.5, rated "Good" with a grade "B" and an "Acceptable" acceptance level. The Net Promoter Score (NPS) placed the application in the "Passive" category. These results confirm Esscore as a functional, accurate, and user-friendly solution for automated answer scoring in educational environments.

I. INTRODUCTION

The digital era has made the use of technology in education more crucial, especially for automating teaching processes [1]. One critical area is the assessment of student learning outcomes, especially in short answers, which are traditionally graded manually. Manual scoring takes considerable time and is vulnerable to personal bias, potentially causing variations in results [2]. To overcome these issues, automated grading systems have been introduced as an effective way to enhance efficiency, accuracy, and fairness in educational evaluation.

Several approaches have been developed to automate the scoring process. The use of Cosine Similarity, Jaccard Similarity, and Transformer-based approaches has been widely studied for comparing student responses with the expected answers. In parallel, the OCR technology has also been employed to convert handwritten or printed images into machine-readable text, though most existing studies focus on numerical or typed text recognition [3].

Research involving Cosine Similarity demonstrated encouraging results, achieving an average accuracy of 81%, precision reaching 93%, and recall at 86% in evaluating online test responses [4]. Another study utilized the Jaccard Similarity algorithm combined with keyword extraction for grading Indonesian student answers, resulting in a correlation of 0.78 with manual scoring and a mean absolute error (MAE) of 0.49 [5]. Transformer-based models such as IndoBERT have also shown significant potential, with a fine-tuned version achieving a low MAE of 0.1285, outperforming traditional methods based on TF-IDF and Linear Regression [6]. In terms of OCR-based scoring, a study applied TesseractOCR to read handwritten scores, correctly recognizing 74.58% of test image data [7].

While these studies show the strengths of various similarity and OCR approaches, most rely on exact or near-exact word matching. Literal character-based matching methods like Levenshtein Distance, although not semantically aware, have a unique advantage when paired with OCR results, especially in handwritten answer evaluation. Handwritten texts introduce variability in character shape, size, and alignment, often resulting in imperfect recognition. Levenshtein Distance is highly effective in handling these small recognition errors, as it tolerates minor misspellings and character substitutions [8].

This becomes particularly important in the context of Indonesian handwritten text. As of today, there is no publicly available OCR model specifically trained to recognize Indonesian handwritten. Most general-purpose OCR engines such as EasyOCR or TesseractOCR are trained predominantly on printed text or non-Indonesian handwriting samples, resulting in limited accuracy when applied to native handwritten responses [9]. In this scenario, a forgiving matching algorithm like Levenshtein becomes not just beneficial, but necessary, to achieve usable grading performance.

Despite its effectiveness in error tolerance, Levenshtein Distance also has clear limitations. It does not account for semantic equivalence or synonyms, which is critical in educational assessments where a single question may have multiple correct answer variants. Therefore, while literal matching is useful for handling OCR noise, it is less suitable for capturing meaning-level correctness [10]. This shortcoming highlights the need for future integration of semantic-aware methods such as synonym matching, rule-based logic, or transformer-based NLP models.

To bridge this gap, this study introduces Esscore, a mobile application designed to automatically assess Indonesian short-answer responses written by hand. The application utilizes EasyOCR to extract text from handwritten Indonesian answer images [11]. To evaluate the accuracy of the recognized text, the system applies the Levenshtein Distance algorithm, which calculates the character-level similarity between student responses and predefined answer keys, allowing tolerance for minor spelling or recognition errors [12]. To accommodate synonymous or semantically equivalent expressions, Esscore supports multiple valid answer keys per question, thus increasing the flexibility and fairness of the scoring process. Ultimately, Esscore aims to reduce teachers grading workload and offer a scalable, practical solution for improving formative assessment in the classroom.

II. METHOD

This study's methodological framework is visualized in Figure 1. The application's back-end was created using Python and FastAPI, and the Android-based front-end was developed in Java using Android Studio. This research begins with a data collection stage, where primary data in the form of handwritten images of student answers are gathered. The image data is collected directly from students in Junior High School (SMP Negeri 9 Denpasar) to ensure the dataset reflects real classroom conditions. Next, is building the model, which involves two main modules: an answer key detection model and a student answer evaluation model. Text recognition is handled by EasyOCR, with the Levenshtein Distance algorithm employed to assess how closely student answers match the key.

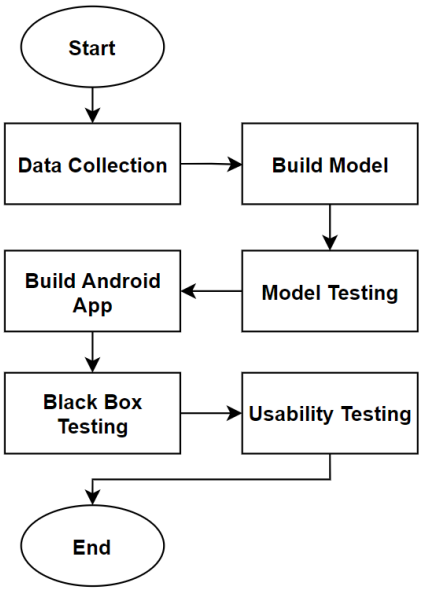

Figure 1. Research Flowchart

Following model development, a model testing phase is conducted to evaluate the accuracy and reliability of the OCR and scoring algorithms. This includes validating the model outputs against manually labeled data to measure performance. Once the models are validated, the next phase involves the development of the Android mobile application, which integrates the OCR and evaluation model into a user-friendly interface for teachers. After the mobile application is built, black box testing is conducted to ensure that all system functionalities work as expected from the user's perspective and usability testing aims to examine how effectively and efficiently users engage with the application, along with their overall satisfaction.

A. Data Collection

This study collected handwritten short-answer responses from 35 eighth-grade students in one class at SMP Negeri 9 Denpasar. Each student was assigned to complete 10 short-answer questions designed to assess conceptual understanding with a total of 350 handwritten answers. To enhance the diversity of student responses and increase the variety of vocabulary used, the questions were divided into two different sets, referred to as Package A and Package B. Both packages contained equivalent difficulty levels but varied in question phrasing. The process of collecting data occurred during classroom activities, where students wrote their answers on paper. Each completed answer sheet was then captured using smartphone cameras and stored in image format (.jpg). This method ensured the inclusion of natural handwriting variations, such as differences in writing style, spacing, and letter formation.

B. Application Workflow

The workflow that guides the operation of the Esscore application is shown in Figure 2. This workflow details all

stages of the application usage process, from the creation of an answer key to the evaluation of student's handwritten answers. The process begins with the creation of the answer key, which can be done in two ways depending on the user's preference. If the user chooses to use a camera, the answer key is written on paper and then photographed using the mobile device. The captured image is cropped if necessary, and then processed by the EasyOCR engine to extract text data, which will serve as the reference answers for scoring. Alternatively, if the user does not use a camera, the answer key can be entered manually through a form provided in the application. This method allows direct input of each correct answer for every question number. Once the answer key is either extracted or input manually, it is then saved into the local database for later use.

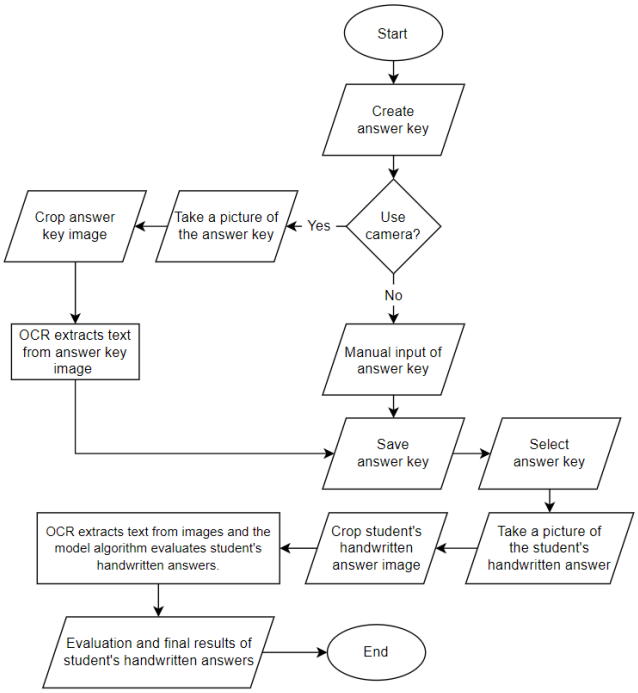

Figure 2. Application Workflow

Following the creation and storage of the answer key, users proceed to the student's handwritten answer evaluation phase. First, the previously saved answer key must be selected, and the user is prompted to capture student's handwritten answer images using the camera. Similar to the answer key process, the captured student's handwritten answer images can be cropped to focus on the relevant text area. Once the student's handwritten answer image is prepared, the system utilizes OCR to extract handwritten text. This extracted student answer is then compared to the selected answer key using Levenshtein Distance algorithm to calculate the degree of textual similarity. The model accommodates variations in handwriting and minor spelling differences to ensure fair and accurate evaluation. The final stage of the workflow involves generating evaluation results, where the application displays the score for each student answer, along with the overall performance. The results are presented to the user in an interpretable format.

C. *EasyOCR*

EasyOCR is an open-source tool that leverages the Convolutional Recurrent Neural Network (CRNN) algorithm [13]. EasyOCR is capable of recognizing text from various types of writing, including handwriting and unusual fonts. EasyOCR also provides a feature that allows users to convert text from various languages [14]. As shown in Figure 3, EasyOCR was built using PyTorch as a backend to support deep learning implementation. EasyOCR goes through several stages to detect text in an image. The first stage is preprocessing, such as noise reduction, binarization, and skew correction. Next, the CRAFT algorithm is used to detect the presence of text. The CRAFT algorithm predicts two main values for each character: Region Score represents the area where the character is located and Affinity Score measures the relationship between one letter and another that forms a word [15].

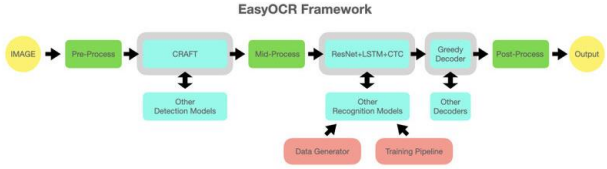

Figure 3. EasyOCR Framework [16]

EasyOCR utilizes a Convolutional Recurrent Neural Network (CRNN) framework, which integrates three stages: feature extraction (ResNet), sequence modeling (LSTM), and decoding (CTC). Initially, convolutional layers extract spatial features from input images, preserving visual structure while reducing dimensions. These features are then passed to LSTM layers, which capture the temporal context of character sequences and help infer unclear text by learning common patterns. Finally, the CTC module decodes the sequence of predictions into readable text by selecting the most probable characters per time step, using a greedy decoding approach that filters out blanks and repeated symbols to form coherent word outputs [17].

D. *Levenshtein Distance*

The Levenshtein Distance algorithm was introduced by Vladimir Levenshtein in 1965 [18]. Levenshtein Distance is a method used to measure the distance between two strings. The distance between two strings is calculated based on the minimum number of operations required to convert one string into another [19]. The equation for the Levenshtein Distance is shown in equation 1 and 2.

$$D(i,j) = \begin{cases} i & if\ j = 0 \\ j & if\ i = 0 \\ min \begin{cases} D(i-1,j)+1 \\ D(i,j-1)+1 \\ D(i-1,j-1)+c \end{cases} & \text{otherwise} \end{cases} \quad (1)$$

where

$$c = \begin{cases} 0 \; if \; a_i = b_j \\ 1 \; if \; a_i \neq bj \end{cases} \quad (2)$$

Description:
$D(i,j)$: Minimum edit distance
$D(i,j-1)+1$: Insertion cost
$D(i-1,j)+1$: Deletion cost
$D(i-1,j-1)+c$: Substitution cost

Equation (1) defines the recursive formulation of the Levenshtein Distance, which calculates the minimum cost to transform a prefix of one string into another. The distance is computed by evaluating three possible operations: deletion, insertion, and substitution. The cost of a deletion refers to the removal of a character from the first string. Similarly, the cost of an insertion corresponds to adding a character to match the second string. The substitution cost depends on whether the characters at the current positions are equal or not, as specified in Equation (2). The base cases handle situations where one of the strings is empty. This recurrence is applied iteratively to construct a matrix of computed subproblems, where the final distance is stored in the matrix's bottom-right position [20]. This method enables efficient calculation of the minimum edits needed to transform one string into another.

E. Evaluation Method

Evaluation methods describe the performance of an short handwritten answer scoring system is measured based on the implemented techniques. Several evaluation approaches are employed, including accuracy measurement of the model performance, functional validation using black box testing, and user experience analysis through usability testing.

1) Accuracy

Accuracy is a performance metric that calculates the number of correct predictions (True Positive and True Negative) divided by the total amount of data evaluated by the model. A higher accuracy value indicates better classification performance [21]. The accuracy formula is illustrated in equation 3.

$$acc = \frac{tp + tn}{tp + fp + tn + fp} \quad (3)$$

Description:
tp : True positive
fp : False positive
tn : True negative
fn : False negative

2) Black box Testing

Black box testing evaluates the system by examining the relationship between inputs and outputs, without any knowledge of its internal implementation, as well as unexpected system behavior from the end user perspective [22]. In this study, a total of 14 features of the application are tested, as presented in Table 1.

TABLE I
BLACK BOX SCENARIO

Test Case	Feature Testing Scenario	Input	Expected Output
T1	Create answer key	Press the “Create” button	The answer key input page is opens successfully
T2	Capture the answer key using a camera	Press the camera button	The answer key is captured successfully
T3	Open gallery and select the answer key image	Press the gallery button	The gallery opens and the answer key image is selected successfully
T4	Crop the answer key image	Press the “Crop” button	The image is cropped and processed successfully
T5	Edit text results from answer key detection	Edit incorrect text	The text is edited successfully
T6	Save the answer key	Press the “Save” button	The answer key is saved successfully
T7	Manual answer key input	Type the answer key	The manual inputted answer key is saved successfully
T8	View answer key details	Select an answer key from history	The answer key details are displayed successfully
T9	Delete the answer key	Press the ”Delete” button	The answer key is deleted successfully
T10	Access student answer scoring page	Press the “Scoring” button	The student answer input page is opens successfully
T11	Capture student’s handwritten answer using a camera	Press the camera button	The student answer is captured successfully
T12	Open gallery and select the student’s handwritten answer image	Press the gallery button	The gallery opens and student answer image is selected successfully
T13	Crop the student’s handwritten answer image	Press the “Crop” button	The image is cropped and processed successfully
T14	Display final student scoring results	Answer keys and evaluated student answer	The final scoring results are displayed successfully

3) Usability Testing

A high level of usability encourages user interest in utilizing the application. Conversely, if the usability value is

low, the lower the user interest in using the application [23]. This research makes use of the System Usability Scale (SUS), originally created by John Brooke in 1986 [24]. In this study, usability testing involved respondents with teaching experience to ensure relevant and practical feedback. The respondent details are shown in Table 2.

TABLE II
RESPONDENT PROFILE

Respondent	Total
Junior High School Teacher	4
Senior High School Teacher	1
Total Respondent	5

The Likert scale serves as a method for evaluating how individuals or groups perceive, interpret, or feel about a given issue or statement. The Likert scale in questionnaires aims to obtain data that is more objective and can be analyzed statistically [25]. In this study, the Likert scale was used to assess user responses in the questionnaire. Table 3 presents each response option is assigned a numeric score to facilitate quantitative analysis.

TABLE III
LIKERT SCALE

Category	Score
Strongly Agree	5
Agree	4
Neutral	3
Disagree	2
Strongly Disagree	1

The System Usability Scale (SUS) is employed in this study to assess the application's usability. The SUS consists of ten statements designed to capture user impressions regarding various aspects of system usability, as shown in Table 4. Responses to each item are provided on a Likert scale, ranging from strong disagreement, neutral, until strong agreement. The items are phrased alternately in positive and negative forms to minimize bias and balancing assessment [26].

TABLE IV
QUESTIONNAIRE

No.	Question
1	I am likely to use the Esscore app frequently
2	I found the Esscore app unnecessary complex
3	I found the Esscore app to be user-friendly
4	I felt I needed help from someone with technical expertise to use the Esscore app
5	I find that the features provided in the Esscore app are seamlessly integrated
6	I feel that there are too many inconsistencies in the Esscore app
7	I believe that most users will quickly become familiar with how to use the Esscore app
8	I found using the Esscore app quite challenging and confusing
9	I feel confident when using the Esscore app
10	I had to learn a lot first before I could operate the Esscore app

The score is derived by aggregating the scores assigned to each question. In the case of odd-numbered (positive) questions, the score contribution is determined by the scale value minus 1. For even-numbered (negative) items, the score is calculated by subtracting the selected scale value from 5. The total usability score is then obtained by multiplying the combined item scores by 2.5 [27]. The equation for calculating the average SUS score across all respondents is shown in equation 4.

$$\bar{x} = \frac{\sum x}{n} \tag{4}$$

Description:
$\bar{x}$: Average SUS score
$\sum x$: Total respodent score
n : Total number of respondents

The interpretation guidelines are used to categorize the results into qualitative descriptors. These categories help translate numeric scores into usability grades, making them easier to understand. The SUS score can be associated with corresponding letter grades, adjective ratings, acceptability, and Net Promoter Score classification. The interpretation of SUS scores is shown in Table 5.

TABLE V
SUS SCORE INTERPRETATION

Grade	SUS	Adjective	Acceptable	NPS
A+	84.1-100	Best Imaginable	Acceptable	Promoter
A	80.8-84.0	Excellent	Acceptable	Promoter
A-	78.9-80.7	Good	Acceptable	Promoter
B+	77.2-78.8		Acceptable	Passive
B	74.1-77.1		Acceptable	Passive
B-	72.6-74.0		Acceptable	Passive
C+	71.1-72.5		Acceptable	Passive
C	65.0-71.0	OK	Marginal	Passive
C-	62.7-64.9		Marginal	Passive
D	51.7-62.6		Marginal	Detractor

III. RESULT AND DISCUSSION

A. Application Interface

This section describes the design and implementation of the application within the user interface. The interface consists of several key components, as illustrated in the figure below.

Figure 4. Main Menu

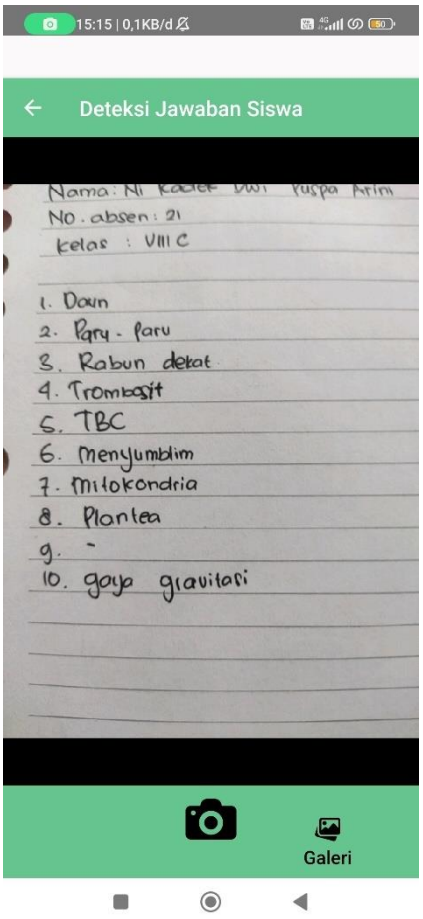

Figure 5. Student's Handwritten Answer Scoring

The main menu functions as the application's primary navigation center, providing users with access to its key features.From this page, users can initiate the creation of answer keys manually or through image-based detection which will later be used to evaluate student responses. Additionally, the main menu provides a dropdown list displaying all previously created and saved answer keys as shown in Figure 4. This feature enables users to easily select the appropriate key when conducting the student answer scoring process.

The answer scoring page allows users to upload images of student's handwritten answer by capturing them directly using a camera as shown in Figure 5. To ensure optimal recognition performance and accurate scoring, users are encouraged to upload images with sufficient clarity and resolution. Through this process, the system extracts text via OCR and evaluates student responses using the established answer key.

Figure 6. Final Student Scoring Result

After the answer evaluation process is completed, users are directed to the final scoring results page, as illustrated in Figure 6. On this page, the system presents a summary of the student's total score along with detailed results for each individual answer. These results include a comparison between the recognized student's handwritten answer and the corresponding correct answers from the selected key using Levenshtein Distance. The displayed outcomes allow users to assess the accuracy of the evaluation and ensure that the automated scoring reflects student performance appropriately.

B. Evaluation Result

This section presents the result of performance, reliability, and usability of the application through structured testing approaches: model performance testing, black box testing, and usability testing.

1) *Model Performance Result*

Model performance testing was conducted to measure the accuracy of the EasyOCR engine in recognizing handwritten text from student responses. The evaluation involved 35 students, each of whom completed 10 short-answer questions, yielding a total of 350 handwritten responses. All answers were written manually on lined paper and no typed responses were used in this test. The extracted text from each image was compared to a predefined answer key using the Levenshtein distance algorithm. A response was considered correct if the similarity score met or exceeded a defined threshold (≥40%). The distribution of model testing results for each student is presented in Table 6.

TABLE VI
MODEL TESTING RESULT

Student Number	Student's Handwritten Answer (A1-A10)			
	True Positive (TP)	False Positive (FP)	True Negative (TN)	False Negative (FN)
S-1	7	0	3	0
S-2	7	0	3	0
S-3	9	0	0	1
S-4	9	0	1	0
S-5	6	0	4	0
S-6	3	0	7	0
S-7	8	0	1	1
S-8	5	0	4	1
S-9	8	0	1	1
S-10	2	0	4	4
S-11	9	0	0	1
S-12	4	1	5	0
S-13	10	0	0	0
S-14	7	0	3	0
S-15	9	0	1	0
S-16	8	0	2	0
S-17	8	0	1	1
S-18	6	0	4	0
S-19	9	0	1	0
S-20	9	0	1	0
S-21	9	0	1	0
S-22	9	0	1	0
S-23	9	0	1	0
S-24	8	0	2	0
S-25	8	0	1	1
S-26	8	0	2	0
S-27	7	0	2	1
S-28	9	0	1	0
S-29	10	0	0	0
S-30	8	0	2	0
S-31	7	0	2	1
S-32	5	0	4	1
S-33	6	0	4	0
S-34	9	0	1	0
S-35	6	0	4	0

Table 7 shows that 260 correct answers were identified as true (TP), 75 incorrect answers were identified as false (TN), 14 correct answers were identified as false (FN), and 1 incorrect answer was identified as true (FP). The model achieved an accuracy rate of 0.957 or 95.7%, based on the evaluation a total of 350 student's handwritten answers. This outcome reflects the model high accuracy in handwritten recognition.

TABLE VII
MODEL ACCURACY SUMMARY

TP	FP	TN	FN	Accuracy
260	1	75	14	95,7%

The key challenges in OCR-based evaluation systems is the occurrence of misrecognition during the text extraction process. Misrecognition refers to instances where the OCR engine incorrectly interprets handwritten characters, resulting in inaccurate or distorted outputs. These errors are often caused by factors such as variations in handwriting styles, poor image quality, overlapping answer, or characters that closely resemble one another, as illustrated in the figure below.

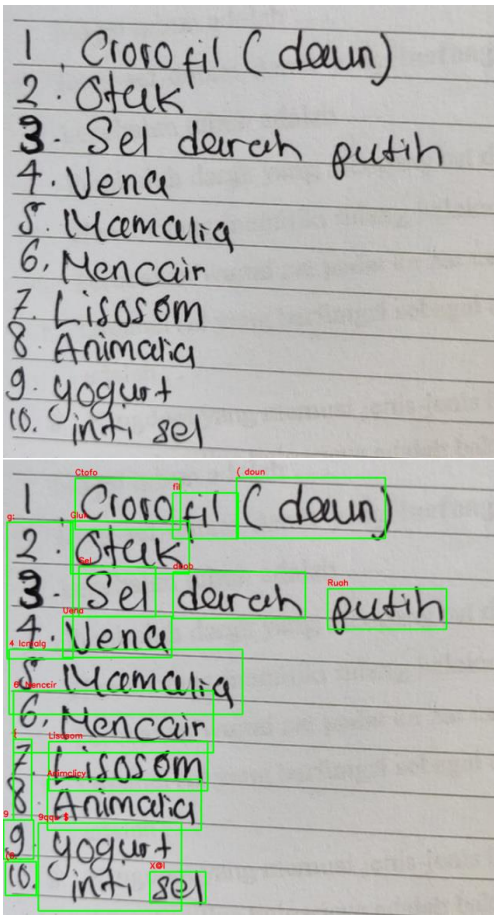

Figure 7. Input Image and OCR Output with Recognition Errors

Figure 7 illustrates an example of a student's handwritten input alongside the text output generated by the OCR system. These recognition errors occur due to overlapping answer and the variability in handwriting styles, which often diverge significantly from the printed or machine-typed fonts that the EasyOCR is predominantly trained on. Unlike standardized digital text, handwritten input tends to vary in terms of stroke thickness, character spacing, alignment, and individual writing habits. These inconsistencies can confuse the OCR engine, leading to misinterpretations of similar characters. For instance, the string "otak" misreaded as a "gluk" because the student's handwriting of the letter "O" looks like the letter "G", the letter "t" looks like the letter "l", and the letter "a" looks like the letter "u". In addition, the answers number 9 and number 10 are overlap, and the inconsistent handwriting of the students makes it difficult for OCR to recognize the text properly, resulting in the answers being marked as incorrect (False Negatives) because the similarity score is lower than 40%. However, only a few students received that errors, most of the other student answers were recognized correctly, as shown in the figure below.

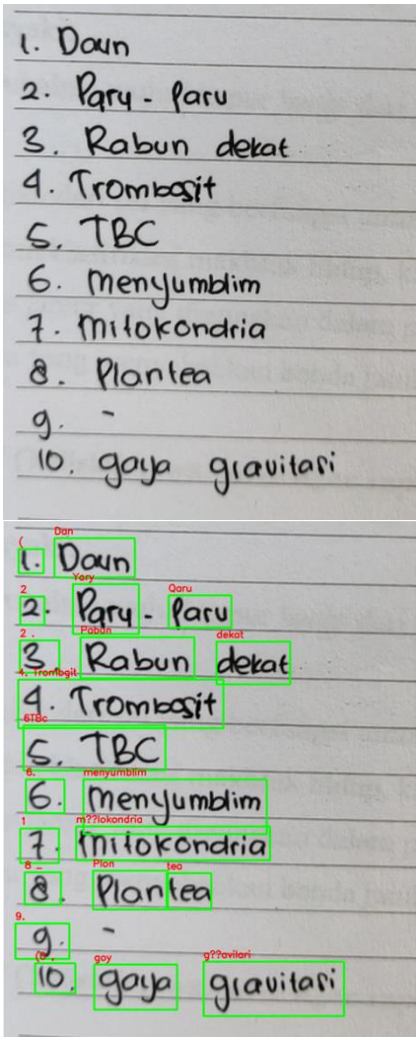

Figure 8. Input Image and OCR Output with Blank Answer

Figure 8 illustrates the examples of OCR processing results with a neatly written student answer and with a blank response. The student's handwriting is clear and well-structured, which allows the OCR engine to accurately detect most of the characters. Although minor recognition errors still occur, they do not significantly affect the outcome. As a result, the similarity score between the OCR output and the answer key remains above 40%, and the answer is classified as a True Positive.

The OCR system also handles the blank response case effectively by detecting only the question number (number

"9") without extracting any string. When this result is compared to the predefined answer key, the similarity score falls below 40%, and the system correctly marks the response as false, categorizing it as a true negative. These examples demonstrate the system capability to differentiate between correct, incorrect, and blank responses based on similarity thresholds and text detection outcomes.

2) *Black Box Testing Result*

The evaluation using black box testing covered a total of 14 scenarios, which represent the core functionalities of the application, including answer key creation, image input, student answer scoring, and result display. All test cases were executed successfully, and the system behaved as expected across all tested functionalities. No bugs, system crashes, or abnormal behaviors were observed during the testing phase, indicating that the application is stable under standard usage conditions.

In addition, the system was also tested using unexpected inputs, such as blurry images, photos without text, or non-answer content. In these cases, the system responded appropriately by displaying a "Error: No text detected. Please try again!" message. This message effectively guided users to reupload clearer images, thereby preventing the application from processing invalid data or generating incorrect scoring results. This demonstrates the robustness of the system in handling edge cases and minimizing user errors. Table 8 shows the details of successful black box testing results without any failures.

TABLE VIII
BLACK BOX TESTING RESULT

Test Case	Feature Testing Scenario	Input	Expected Output	Result
T1	Create Answer Key	Press the "Create" button	The answer key input page is opens successfully	Success
T2	Capture the answer key using a camera	Press the camera button	The answer key is captured successfully	Success
T3	Open gallery and select the answer key image	Press the gallery button	The gallery opens and the answer key image is selected successfully	Success
T4	Crop the answer key image	Press the "Crop" button	The image is cropped and processed successfully	Success
T5	Edit text results from answer key detection	Edit incorrect text	The text is edited successfully	Success
T6	Save the answer key	Press the "Save" button	The answer key is saved successfully	Success
T7	Manual answer key input	Type the answer key	The manual inputted answer key is saved successfully	Success
T8	View answer key details	Select an answer key from history	The answer key details are displayed successfully	Success
T9	Delete the answer key	Press the "Delete" button	The answer key is deleted successfully	Success
T10	Access student answer scoring page	Press the "Scoring" button	The student answer input page is opens successfully	Success
T11	Capture student's handwritten answer using a camera	Press the camera button	The student answer is captured successfully	Success
T12	Open gallery and select the student's handwritten answer image	Press the gallery button	The gallery opens and student answer image is selected successfully	Success
T13	Crop the student's handwritten answer image	Press the "Crop" button	The image is cropped and processed successfully	Success
T14	Display final student scoring results	Answer keys and evaluated student answer	The final scoring results are displayed successfully	Success

3) *Usability Testing Result*

The System Usability Scale (SUS) was used to measure the overall usability of the application with five teacher respondents participated in the SUS assessment after interacting with the application and performing key tasks such as creating answer key and evaluating student's handwritten answer. Each respondent answered ten SUS items based on a Likert scale. Table 9 shows the distribution of questionnaire responses from each respondent.

TABLE IX
QUESTIONNAIRE RESULT USING THE LIKERT SCALE

Score per Question	Respondent				
	R1	R2	R3	R4	R5
Q1	4	4	4	4	4
Q2	2	2	1	2	2
Q3	4	5	4	4	5
Q4	2	2	3	2	2
Q5	4	4	5	4	3
Q6	2	2	2	2	1
Q7	4	5	4	4	4
Q8	2	2	2	2	2
Q9	4	4	4	4	4
Q10	2	2	2	3	2

The scoring method follows the standard SUS calculation process. Scores for the positive (odd-numbered) items are calculated by subtracting 1 from each response, while scores

for the negative (even-numbered) items are derived by subtracting each response from 5. The resulting values are then summed and multiplied by 2.5 to generate a final score out of 100 for each respondent. As shown in Table 10, the SUS scores yielded result of 76.5, indicating that the application achieved a positive level of usability.

TABLE X
SUS TESTING RESULT

Respondent	Total Question		Score	Total Score (Score × 2.5)
	Odd (Scale-1)	Even (5–Scale)		
R1	15	15	30	75
R2	17	15	32	80
R3	16	15	31	77.5
R4	15	14	29	72.5
R5	15	16	31	77.5
SUS Score				**76.5**

Table 11 presents the interpreation of the SUS score of the Esscore application. The application received a SUS score of 76.5, placing it in the "Good" range according to adjective ratings, corresponding to a grade of B, and is considered acceptable in terms of usability and the NPS "Passive" indicating that while users are generally satisfied with the application, they may not yet actively promote it to others. From the perspective of the teacher respondents who participated in the testing, this result suggests that the application is well-designed and usable in its current state.

TABLE XI
SUS INTERPRETATION RESULT

Grade	SUS Score	Adjective	Acceptable	NPS
B	76,5	Good	Acceptable	Passive

IV. CONCLUSION

Based on the research conducted on Esscore, an Android-based application developed to scoring short handwritten answers using EasyOCR and the Levenshtein Distance algorithm, it can be concluded that the system has been successfully implemented as a functional and accurate automatic scoring solution in the educational context. The application is capable of recognizing and evaluating short answers written by hand, while also tolerating minor writing variations or recognition errors through the use of character-level similarity matching.

The black box testing with total of 14 functional features are passed successfully without failure, indicating a high degree of application stability and reliability. Furthermore, the application was evaluated using a total of 350 student's handwritten answers, resulting in an accuracy rate of 95.7%, suggesting a strong correlation between the final score and the expected correct answer. In terms of usability, the application underwent System Usability Scale (SUS) testing involving relevant respondents, particularly school teachers. The application received a SUS score of 76.5, placing it in the "Good" range, corresponds to a grade scale of "B", and is considered "Acceptable" in terms of user acceptance. The Net Promoter Score (NPS) places the app in the "Passive" category, meaning users find the application usable but may not actively recommend it to others. The strongest usability aspect perceived by users is the application's user-friendly interface. However, the weakest aspect identified is the initial learning curve, as users may need to familiarize themselves with the application's workflow and features during first-time use before operating it smoothly.

This study also recognizes critical aspect in the implementation of automated scoring systems in educational settings. Firstly, the risk of scoring inaccuracies must be minimized, as incorrect evaluations may lead to unfair academic outcomes. The influence of handwriting style on OCR accuracy remains a challenge, as highly cursive or unclear handwriting can reduce recognition precision. Although Esscore has demonstrated high accuracy and user acceptance, there are still limitations that can be improved in future development. For example, integrating a more robust correction mechanism to adapt to various handwriting styles. Future research may also explore the integration of Natural Language Processing (NLP) models to handle more complex answer structures beyond keyword similarity.

REFERENCES

[1] A. Sadriani, M. Ridwan, S. Ahmad, and I. Arifin, "PROSIDING SEMINAR NASIONAL Peran Guru Dalam Perkembangan Teknologi Pendidikan di Era Digital." [Online]. Available: https://journal.unm.ac.id/index.php/Semnasdies62/index

[2] Amir Salim Khairul Rijal and Billy Hendrik, "Studi Literatur Sistem Penilaian Esai Otomatis Pada E-Learning Dengan Algoritma Winnowing," *Jurnal Sistem Informasi dan Ilmu Komputer*, vol. 1, no. 3, pp. 163–172, Aug. 2023, doi: 10.59581/jusiik-widyakarya.v1i3.1227.

[3] K. Hamad and M. Kaya, "A Detailed Analysis of Optical Character Recognition Technology," *International Journal of Applied Mathematics, Electronics and Computers*, vol. 4, no. Special Issue-1, pp. 244–244, Dec. 2016, doi: 10.18100/ijamec.270374.

[4] E. L. Amalia[1], A. J. Jumadi, I. A. Mashudi[3], W. Wibowo[4], P. N. Malang, and P. Korespondensi, "Analisis Metode Cosine Similarity Pada Aplikasi Ujian Online Esai Otomatis (Studi Kasus Jti Polinema) Cosine Similarity Method Analysis On Automatic Esai Online Test Application", doi: 10.25126/jtiik.202184356.

[5] D. Riyanto and A. Azis, "Implementation of the Jaccard Similarity Algorithm on Answer Type," *International Journal of Informatics and Information Systems*, vol. 5, no. 2, pp. 76–83, 2022.

[6] K. Ayu Pradani, L. Hulliyyatus Suadaa, and P. Korespondensi, "Automated Essay Scoring Menggunakan Semantic Textual Similarity Berbasis Transformer Untuk Penilaian Ujian Esai Automated Essay Scoring Using Transformer-Based Semantic Textual Similarity For Essay Assessment", doi: 10.25126/jtiik.2023107338.

[7] K. Saravanan, C. C. Chew, K. G. Tay, S. L. Kek, and A. Huong, "Exam Marks Summation App Using Tesseract OCR in Python," *International Journal of Integrated Engineering*, vol. 14, no. 3, pp. 102–110, 2022, doi: 10.30880/ijie.2022.14.03.011.

[8] C. Garrido-Munoz, A. Rios-Vila, and J. Calvo-Zaragoza, "Handwritten Text Recognition: A Survey," Feb. 2025, [Online]. Available: http://arxiv.org/abs/2502.08417

[9] E. Zhang, V. A. Putra, and G. P. Kusuma, "Improving Optical Character Recognition Accuracy For Indonesia Identification Card Using Generative Adversarial Network," *J Theor Appl Inf Technol*, vol. 100, no. 8, 2022, [Online]. Available: www.jatit.org

[10] A. Pal and A. Mustafi, "Vartani Spellcheck-Automatic Context-Sensitive Spelling Correction of OCR-generated Hindi Text Using BERT and Levenshtein Distance."

[11] M. Flores, D. Valiente, M. Alfaro, M. Fabregat-Jaén, and L. Payá, "Evaluation of Open-Source OCR Libraries for Scene Text Recognition in the Presence of Fisheye Distortion," in *Proceedings of the 21st International Conference on Informatics in Control, Automation and Robotics*, SCITEPRESS - Science and Technology Publications, 2024, pp. 133–140. doi: 10.5220/0012927600003822.

[12] M. R. Batisya and K. Jevanda BS, "Implementasi Algoritma Levenshtein Distance Untuk Misspelled Word Pada Pencarian Lagu Melayu," *Jurnal Informatika*, vol. 10, no. 1, pp. 72–78, Mar. 2023, doi: 10.31294/inf.v10i1.15208.

[13] S. A. Nugroho *et al.*, "Rancang Bangun Sistem Deteksi Label Kardus Berbasis Model Kecerdasan Buatan YOLO dan EasyOCR serta ESP32-CAM 190 Rancang Bangun Sistem Deteksi Label Kardus Berbasis Model Kecerdasan Buatan YOLO dan EasyOCR serta ESP32-CAM."

[14] M. Fadhel Haidar and F. Utaminingrum, "Deteksi Plat Nama Ruangan untuk Kendali Kursi Roda Pintar menggunakan YOLOv5 dan EasyOCR berbasis TX2," 2023. [Online]. Available: http://j-ptiik.ub.ac.id

[15] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee, "Character Region Awareness for Text Detection," Apr. 2019, [Online]. Available: http://arxiv.org/abs/1904.01941

[16] N. Haque, S. Islam, R. A. Tithy, and M. S. Uddin, "Automatic Bangla License Plate Recognition System for Low-Resolution Images," in *2022 4th International Conference on Sustainable Technologies for Industry 4.0, STI 2022*, Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/STI56238.2022.10103289.

[17] B. Shi, X. Bai, and C. Yao, "An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition," Jul. 2015, [Online]. Available: http://arxiv.org/abs/1507.05717

[18] R. Adawiyah and N. E. Saragih, "Implementasi Algoritma Levenshtein Distance Dalam Mendeteksi Plagiarisme."

[19] M. F. Azhri, D. Swanjaya, and R. K. Niswatin, "Penerapan Algoritma Levenshtein Distance pada Aplikasi Asisten Guru Bahasa Inggris."

[20] M. Khalid, M. M. Yousaf, and M. U. Sadiq, "Toward Efficient Similarity Search under Edit Distance on Hybrid Architectures," *Information (Switzerland)*, vol. 13, no. 10, Oct. 2022, doi: 10.3390/info13100452.

[21] H. M and S. M.N, "A Review on Evaluation Metrics for Data Classification Evaluations," *International Journal of Data Mining & Knowledge Management Process*, vol. 5, no. 2, pp. 01–11, Mar. 2015, doi: 10.5121/ijdkp.2015.5201.

[22] I. S. Handayanto and I. Nuryasin, "Pengujian Blackbox Decision Table pada Sistem Aplikasi Mobile Sharing Story App," *Smart Comp: Jurnalnya Orang Pintar Komputer*, vol. 13, no. 2, Apr. 2024, doi: 10.30591/smartcomp.v13i2.6572.

[23] M. Shania and T. Tranggono, "Analisis Usability Pada Aplikasi Shopee Menggunakan Metode System Usability Scale (SUS)," *Briliant: Jurnal Riset dan Konseptual*, vol. 9, no. 2, pp. 452–465, May 2024, doi: 10.28926/briliant.v9i2.1884.

[24] A. P. Sukma, R. Yusuf, and R. H. Dai, "Analisis Pengukuran Usability Sistem Informasi Manajemen Baznas (Simba) Menggunakan Metode System Usability Scale (SUS)," vol. 3, no. 2, 2023.

[25] A. Aditya Santika, T. Hamonangan Saragih, D. Kartini, and R. Ramadhani, "Penerapan Skala Likert Pada Klasifikasi Tingkat Kepuasan Pelanggan Agen BRILink Menggunakan Random Forest," vol. 11, no. 3, doi: 10.26418/justin.v11i3.

[26] D. Irawan, D. Syamsuar, T. B. Kurniawan, and M. Akbar, "Analisis Usability Sistem Informasi Akademik (Studi Kasus SISFO Universitas PGRI Palembang)," *JSI : Jurnal Sistem Informasi (E-Journal)*, vol. 13, no. 2, p. 2021, [Online]. Available: http://ejournal.unsri.ac.id/index.php/jsi/index

[27] J. Brooke, "SUS: A quick and dirty usability scale." [Online]. Available: https://www.researchgate.net/publication/228593520