Accredited SINTA 2 Ranking
Decree of the Director General of Higher Education, Research, and Technology, No. 158/E/KPT/2021
Validity period from Volume 5 Number 2 of 2021 to Volume 10 Number 1 of 2026

Published online at: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 8 No. 1 (2024) 42 - 53 ISSN Media Elektronik: 2580-0760

The Design of a C1 Document Data Extraction Application Using a Tesseract-Optical Character Recognition Engine

Ircham Aji Nugroho[1], Bety Hayat Susanti[2*], Mareta Wahyu Ardyani[3], Nadia Paramita R.A.[4]
[1,2,3,4]Department of Cryptographic Engineering, Politeknik Siber dan Sandi Negara
[1]ircham.aji@bssn.go.id, [2]bety.hayat@poltekssn.ac.id, [3]mareta.wahyu@poltekssn.ac.id, [4]nadia@poltekssn.ac.id

Abstract

The 2019 election process employed the Vote Counting Information System, also known as Sistem Informasi Penghitungan Suara (Situng), to provide transparency in the recapitulation process. The data displayed in Situng is from the C1 document for 813,336 voting stations in Indonesia. The data collected from the C1 document is entered and uploaded into Situng by officers at the municipal General Election Commission (GEC). Since this process is performed by humans, it is not immune to errors. In the recapitulation process of the 2019 election results, there were 269 data entry errors, and the data entry process also did not run according to the specified target, resulting in delays. Furthermore, there were cases of C1 document modification, raising concerns about the data's authenticity. To avoid human errors and increase data entry speed, automatic data entry is a plausible option. The data entered is text data in image documents with the same template format, so that optical character recognition (OCR) can be used to read the text while improving image quality and alignment, resulting in a more accurate OCR reading area. In this study, we developed a C1 document data extraction application using the waterfall SDLC method, which has undergone a systematic and thorough process. The application was developed using Tesseract optical character recognition. Tesseract is an open-source OCR engine and command-line program that allows for the recognition of text characters within a digital image. The accuracy obtained by using this method is still not optimal as a substitute for Situng's data entry officer. To guarantee the integrity of the C1 document, we used the RSA-2048 digital signature scheme. Using the Tesseract-OCR Engine for character recognition, combined with digital signature capabilities, provides a comprehensive solution to reduce the human error factor that might result in miscalculations and inaccurate processes.

Keywords: affine transformation; digital signature; automatic data entry; optical character recognition; RSA-2048; SHA-256; tesseract-OCR

How to Cite: Ircham Aji Nugroho, B. H. Susanti, Mareta Wahyu Ardyani, and Nadia Paramita R.A., "The Design of a C1 Document Data Extraction Application Using a Tesseract-Optical Character Recognition Engine", *J. RESTI (Rekayasa Sist. Teknol. Inf.)* , vol. 8, no. 1, pp. 42 - 53, Feb. 2024.
DOI: *https://doi.org/10.29207/resti.v8i1.5151*

1. Introduction

The 2019 elections were held simultaneously to elect the President and Vice President for the 2019-2024 period, along with the election of other legislature members. The election was held on 17 April 2019, and the vote counting and recapitulation process was carried out using an information technology-based tool known as the Vote Counting Information System hereinafter referred to as Situng. Situng is a tool developed by the General Election Commission (GEC) to guarantee the accountability of all voting stages, including the counting, the recapitulation, and the establishment of vote count results for the 2019 election [1]. The use of Situng aims to provide transparency in the vote-counting process to the public so that the community has a control function over the process, which is carried out in several phases. Following the process in Situng, after counting the votes at each voting station, the recapitulation of the vote count based on the C1 document is collected at the municipal GEC for further scanning of the C1 document and formatted into a Microsoft Excel file. Data in the form of Excel files and scanned images of C1 documents are then uploaded to Situng. The GEC allows five days after the general election to complete the data input process [1].

Based on GEC Decree Number 536 of 2019 concerning Instructions for the Use of the 2019 General Election Vote Counting Information System (Situng), there are

Received: 09-06-2023 | Accepted: 07-01-2024 | Published Online: 04-02-2024

three roles of the municipal GEC regarding the use of Situng, including an entry operator, a scanning operator, and a verifier. The entry operator is in charge of entering data from the C1 document into Situng. The scanning operator is in charge of scanning C1 documents and transmitting data to the server. The oversees the verifying of the data entered with the scan results that have been uploaded to the server. Due to human errors, the process may result in miscalculations and inaccuracies. Several organizations documented 708 cases of recapitulation [2]. GEC chairman Arief Budiman revealed that there were 269 errors in the data input process, with human error being one of the causes [3]. This factor is mainly due to fatigue and inability to focus, other factors are also present [4]. In addition to the human error that caused data input errors, GEC Commissioner Evi Novida Ginting Manik stated that the process of inputting election results data into Situng did not run according to the specified time target. The process experienced delays of up to eight days after the election took place [5]. Moreover, the National Winning Body or Badan Pemenangan Nasional (BPN) claimed that the C1 document was edited during the trial of the dispute over the election of the president and vice president.

In general, optical character recognition (OCR) is a method of converting optically processed characters from image capture form to computer-readable form (see [6] - [9] for more details on OCR). OCR is typically divided into three stages. First, the layout is examined to determine the locations of text lines in the document to identify the reading order. Second, the character recognition engine recognizes individual characters and processes lines of text to create text strings. Finally, the modeling language uses a dictionary to correct text strings [10]. Images containing this text may have concatenated characters and images with other text captions [11]. Shafait and Smith [10] implemented OCR with Tesseract to detect table columns in various documents (company reports, articles, news newspapers, and magazines). Tesseract is an open-source OCR engine that can be used to train models or use existing models for character reading. Tesseract has the advantage of being able to segment text on a per-character mode and using a Long Short-Term Memory (LSTM) engine based on Recurrent Neural Network, with a focus on line recognition.

According to [12], data entry by humans in large numbers of documents has limitations and is at risk of error. To avoid these errors, automatic data entry can be done by doing template matching or aligning the input image with the template image. The image alignment method is done by using Affine Transformation. According to preliminary research, the scan results of C1 documents in Situng are not always perpendicular and have variations in image tilt so that the parallel input image can be read quickly and accurately at the desired image location. The data read from the C1 document is in the form of numeric text, which can be converted to digital form using OCR. According to [13], before performing the OCR process, an image quality improvement process by the Tesseract library documentation is required so that the OCR engine can read better. Based on the background previously explained, in this research, a C1 document data extraction application will be built by implementing optical character recognition. In addition, the authentication process is needed to ensure that the C1 document displayed and processed in Situng is complete and is by the initial state of the scanning process at the municipal GEC. Authentication is accomplished through the use of a digital signature scheme that employs the RSA-2048 algorithm and the Secure Hash Algorithm (SHA) 256 for compression. The use of OCR as a solution presented in this study is to eliminate human error in the data extraction process for Situng data input and to ensure that the documents utilized are complete.

2. Research Methods

2.1. Automated Data Entry

As Sanguansat [12] explained, automated data entry in physical documents is called Optical Mark recognition. The system being built is used to collect data in the form of a large number of questionnaires. Furthermore, based on the Quick Response Code (QR-Code) detection algorithm, the system can be applied to various conversions, sizes, and rotations.

The system provided in this study aims to obtain data from the questionnaire by blackening the circle mark on the options provided. The input device used is a scanner machine in general, a computer as a processing unit, and a display as a resulting report. Because the scanner machine is used as an input device, the size and rotation of the scan results will vary. The QR code also includes a detection method that is based on a position detection pattern, as shown in Figure 1.

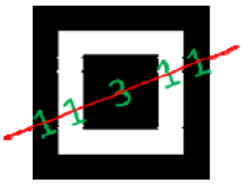

Figure 1. Position detection pattern QR-Code

2.2. Tesseract-OCR Engine

Tesseract is an open source Optical Character Recognition (OCR) engine developed by HP from 1984 to 1994. Tesseract does not require analysis because HP has independently developed technology for analyzing page layouts that are used in its products (but are not open-source). Therefore, Tesseract uses input in the form of binary images [14]. Currently, Tesseract is available in version 5.0, which includes an OCR engine based on Long Short-Term Memory (LSTM) Neural Networks, which focuses on line recognition. In addition, the ability to recognize character patterns in

Tesseract version 3.0 is still used with OCR Engine Mode. Tesseract supports Unicode (UTF-8) and can recognize more than 100 languages. Tesseract also supports several output formats, such as plain text, hOCR (HTML), PDF, TSV, and XML.

Python provides a Tesseract library named pytesseract that can be used to implement the Tesseract-OCR engine in Python. Python-Tesseract can read all image formats supported by the Pillow and Leptonica libraries, including jpeg, png, gif, bmf, tiff, and others, making it useful as a stand-alone library for Tesseract. When used as a script, Python-Tesseract will also print the recognized text rather than writing it to a file. In addition to being based on LSTM, this library has the advantage that the algorithm in Tesseract can directly recognize text or characters in images and separate parts of characters from other parts for further direct classification of text into digital form. Tesseract also provides a choice of several text segmentation modes, one of which is text segmentation, which treats the input image as a single character. This is by the OCR target in the C1 document, which is one character per column. Tesseract can also do training to create models for different cases with its dataset.

2.3. Optical Character Recognition

In [13], it is explained about how character recognition is implemented. It has been observed that the template matching method is an application of Artificial Neural Network (ANN), namely backpropagation and pattern dependency, which provide several advantages in the ability to recognize patterns. The following are the steps used by Singh et al. [13].

Images are collected and preprocessed to adjust resolution, zoom, crop, rotate, and reduce noise. The parameters used to perform the preprocessing stage are not explained in this method. The image is grayscale, resulting in only two colors. During segmentation scanning, the image is filtered between foreground and background. Characters are separated from other image components using feature extraction. After the characters in the image are separated, the classification process is carried out to read what characters are in the image with the backpropagation algorithm. The results of the characters read earlier are then grouped into one word and processed in the post-processing stage to correct character reading errors that are adapted to the existing word dictionary.

2.4. Performance Measure

According to [15], retro-conversion is the activity of extracting and recognizing content from document images. To carry out this process, some OCR generally runs it in two steps, namely, segmentation of the text area and character recognition in a predetermined area. To measure or evaluate the performance of OCR, the read text must be the same as and parallel to the original text. Once aligned, the OCR performance can be calculated. ZoneMapAltCnt is a new method provided by [15] to evaluate OCR performance. The character metric is based on the Levenshtein distance, which represents the lowest number of edit operations (character insertions, deletions, and substitutions) required to correct the hypothetical or predictive text (HZ) to match the reference or original text (RZ). CH is the number of characters in HZ, and CR is the number of characters in RZ. The total error characters are shown in Formula 1.

$$C_{error} = C_{ins} + C_{del} + C_{sub} \quad (1)$$

The correct number of characters in the equation is shown in Formula 2.

$$C_{correct} = C_{aln} - C_{error} \quad (2)$$

with C_{aln} is the number of readable characters that are parallel to the original characters. The precision and recall of characters can be calculated by Formula 3 Both precision and recall are relative measures of the OCR accuracy because they are computed as ratios of the correct output to the total output (precision) or input (recall).

$$C_{Recall} = \frac{C_{correct}}{C_R} \quad (3)$$

2.5. Digital Signature

Menezes [16] and Regenscheid [17] are cited in all discussions of the concept of a digital signature. A digital signature is a string of data that associates a message (in digital form) with its original entity. A digital signature scheme or mechanism consists of a signature generation algorithm and a correlated verification algorithm. There are two types of digital signature schemes, i.e., digital signature schemes with appendix and digital signature schemes with message recovery. A digital signature scheme with an appendix is a scheme that requires input of messages in the verification algorithm. The digital signature scheme with message recovery is a verification algorithm scheme that does not require a message. In this study, the RSA algorithm uses a digital signature scheme with message recovery. RSA is a signing scheme that relies on the difficulty of factoring integers The RSA algorithm used has a bit length of 2048 because it is based on National Institute of Standards and Technology (NIST) standards that require a minimum specification of or equal to 2048 bits. The SHA-256 algorithm is used as the underlying hash function in the RSA digital signature scheme.

3. Research Methods

The methodology used in this research is Design Research Methodology (DRM) [18], which is a set of methods used as a framework to determine the research design. DRM was chosen because it can determine the points of each strategy and create the proper approach from a research design, allowing the study to be successful and efficient. DRM is divided into four stages: research clarification, descriptive study I (DS-I),

descriptive study II (DS-II), and descriptive study III (DS-III). Figure 2 depicts these stages.

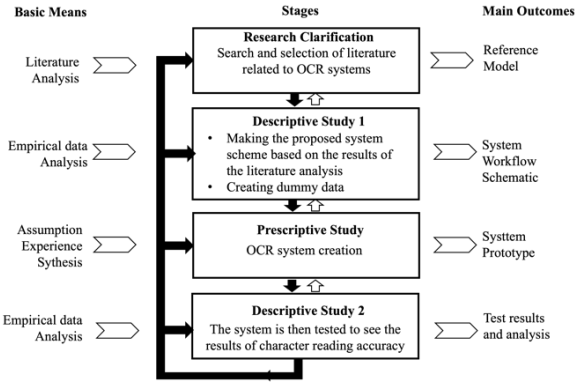

Figure 2. Stages in DRM [18]

3.1. Research Clarification

The RC stage is where you look for evidence to support your assumptions to develop useful and realistic research objectives [18]. The evidence is documentation from the literature related to the problems to be resolved in research. The schematic of the literature study is shown in Figure 3.

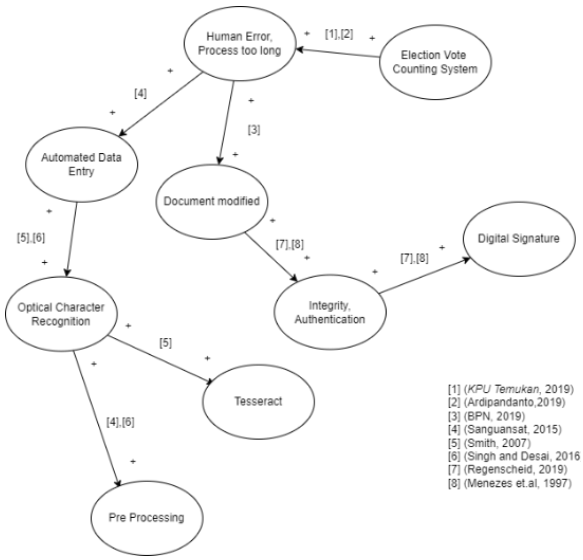

Figure 3. Literature Study Scheme

Based on CNN Indonesia [3], [4], and Ardipandanto [2] news, it is indicated in Figure 3 that Situng has various problems, including the presence of human error and the duration of the data entry procedure. Aside from that, there are additional issues, such as claims that the C1 document was changed by editing, as communicated in the election dispute trial based on the National Winning Body [19].

According to Sanguansat [12], the problem of human error and the duration of the data entry process may be solved by performing automatic data entry by reading the precise position of the variables required in the document. The variables read in Document C1 are handwritten numeric characters that may be transformed to digital form using the Tesseract Optical Character Recognition (OCR) library based on Smith [14] and OCR steps based on Singh *et al.* [13]. Meanwhile, the problem of modifying documents may be avoided by preserving the integrity of the C1 document through the use of a digital signature based on Menezes *et al.* [16] and Regenscheid [17].

3.2. Descriptive Study

These success criteria and key factors are based on the solutions proposed in this study to answer the problem formulation. Figure 4 explains the success criteria and critical components and shows their position in the literature study method. The criteria obtained are Success criteria, conditions that must be met, and determine the objectives of a study. In this study, the success criteria are the generation of Automated Data Entry and the integrity of the document itself; Key factors are factors that influence other factors and are the main factors for achieving the goals of the success criteria. In this study, the key factors are the accurate optical character recognition process and the document integrity that is maintained with digital signatures.

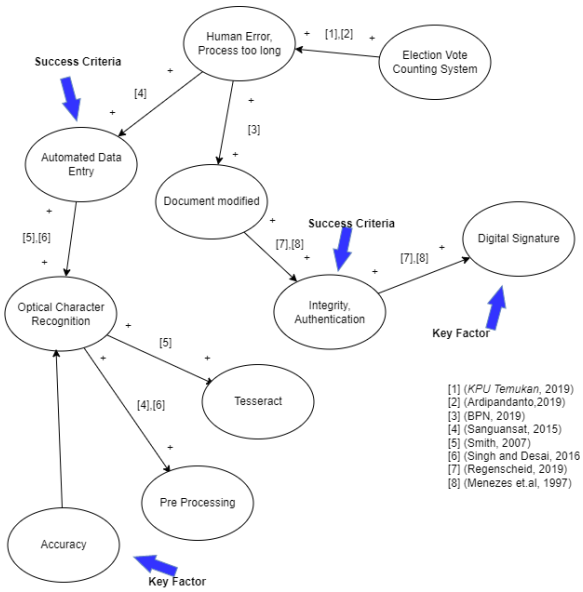

Figure 4. Impact Model for Describing Goals

3.3. Prescriptive Study

The development stage of the C1 document data extraction application with the Tesseract OCR Engine uses the waterfall approach to the SDLC. Waterfall is a method that assumes the phases of development in stages [20]. The following are the stages in the waterfall approach:

Planning is the first stage in determining the background of creating the application by determining how it will work. At this stage, it is explained how the application is built by the current circumstances.

At the Analysis stage, requirement analysis and general description of the C1 document data extraction application using the Tesseract OCR Engine, as well as the use of digital signatures, are performed. The purpose

of requirement analysis is to explain the functions required in the application. It includes both functional and non-functional requirements.

At the Design stage, the Tesseract OCR Engine and digital signatures are used to design the C1 document data extraction application. Use Case Diagrams and Data Flow Diagrams are used for modeling. Use Case Diagrams are used to determine the application's business processes. The Data Flow Diagram is then used to visually determine the data flow from the system.

At the Implementation stage, the application is being developed based on the design that has been created. After the application is built, it is then tested. Unit testing, integration testing, and system testing are all performed. This study will employ black box testing for unit testing, scenario testing for integration testing, and requirements testing for system testing.

3.4. Descriptive Study 2

The test was performed on the implementation by measuring performance using the ZoneMapAltCnt method to measure character errors. This method compares the correctly read characters to the OCR targeted characters, which are parallel to the actual target characters. The character error value, or Cerror is a character that is not readable according to the actual character. OCR area targeting is also included in app performance measurement because the app's goal is to correctly read the characters in the correct area.

The dummy C1 document used for testing was filled out by ten different people based on the original data available on Situng GEC's official website. The dummy C1 document is then scanned or photographed by each of the ten people using a scanner machine or mobile phone. Each of the ten people created dummy data of 40 C1 documents, which consisted of 40 pairs of C1 documents, the first 40 pages, and 40-second pages. On a national scale, a data sample of 400 pairs of C1 documents was collected from a population of 813,336 voting stations, with one C1 document for each station. Because text classification using OCR is done at the district/city level, the population is divided by the number of districts and cities. According to the Ministry of Home Affairs [22], there are 416 districts and 98 cities. So, the average population at the district and city levels is 1,583, with Bogor district having the most voting stations (15,000). As a result, the test data used represents the average voting stations at the municipal level by up to 25.27%.

The data collected is prepared by determining whether or not it is damaged. There were 40 damaged files on page 2 of the C1 document, so they were not included in the test data. There were 400 C1 documents on page 1 and 360 C1 documents on page 2. To provide a clear picture of OCR performance, groups were divided into three scales based on image quality, taking into account the clarity of the scanning results, file resolution, and scan devices used. The grouping based on quality are: Image quality scale of 3 consists of images of good quality; Image quality on a scale 2 consists of images of medium quality; Image quality on a scale of 1 consists of images of poor quality.

The image quality scale is regrouped into three categories of test data: Category A test data for image quality at scale 3; Category B test data for scale 3 and 2 image quality; Category C test data for image quality at scale 3, 2, and 1.

3.5. Planning

In general, this system is divided into two sections: the signing section for the municipal GEC and data extraction with OCR, hereinafter referred to as the Signer Section, and the document verification section for the Central GEC, hereinafter referred to as the Verifier Section. This is by the recapitulation process of vote counting using Situng, in which data entry, scanning, and uploading of C1 documents is carried out at the municipal GEC and the final recapitulation nationally is carried out at the Central GEC. Because it is a component of Situng's main system, the application is built on the backend.

3.6. Analysis

The research was conducted based on the assumption that the Signer section is installed on the municipal GEC Situng application and the Verifier section is installed on the Central GEC Situng. The entire process is performed on a local computer, and data transactions between the municipal GEC and the Central GEC are handled by Situng, which is not part of the scope of the study. The overall scheme of the system in this study is shown in Figure 5.

Based on GEC Regulation Number 3 of 2019 concerning Voting and Counting of Votes, C1 documents from all TPS in the Regency or City are collected at the Regency or City GEC, and then, if adjusted to add a signing feature, the documents are signed by an authorized officer at the Regency or City GEC. The collection of C1 documents in each regency or city is carried out by the data entry process in Situng, which was previously carried out by the entry operator and was replaced by automatic data extraction using OCR. Then, the private key and public key that has been previously generated and distributed to each section of the Regency/City GEC and the Central GEC will be used for signing and verifying documents. The signed document is then sent to the Central GEC for verification. The results of the OCR then become input for tabulation in Situng.

The RSA 2048 and SHA 256 algorithms are used in the digital signature process. Subsection 2.5 contains a broad framework for the signature and verification procedure based on Menezes *et al.* [12]. The adjustment in this study is that the signed document is a JPG file with the signature technique indicated in Figure 6 and the verification scheme provided in Figure 7.

The JPG file is converted to byte array format, which is then used to generate the hash value with SHA 256. The hash value is encrypted using the private key of the Regency or Municipal GEC, which generates the signature value and is placed as a signature in the original JPG image.

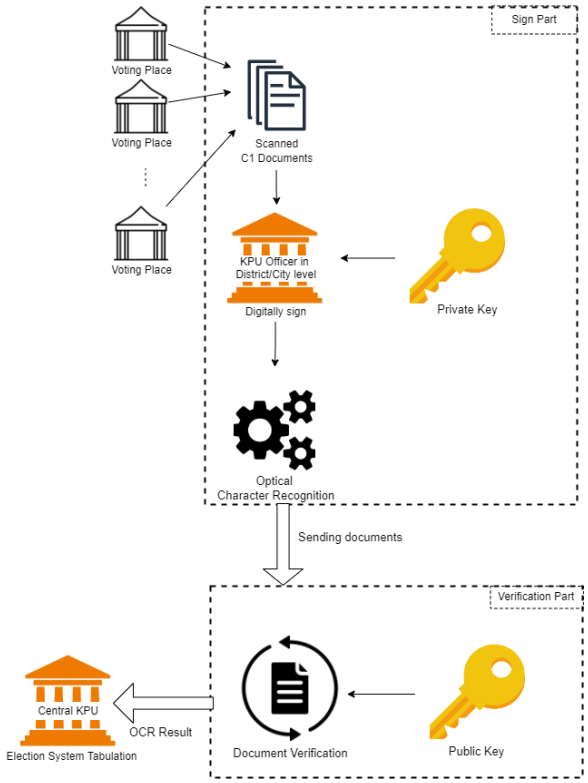

Figure 5. Overview of the proposed system

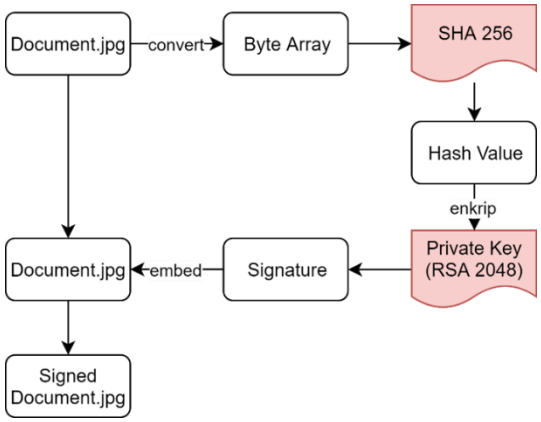

Figure 6. Image Document Signing Scheme

The goal of verification is to determine if the documents received are still intact or not. The signed JPG file is isolated from the original file by the signature value. The signature value is decrypted using a public key that corresponds to the private key used to sign the document, and a hash value is produced. The hash value is determined after comparing it to the hash value acquired during the process of transforming the original file into a byte array.

Document C1 is submitted to an OCR data extraction procedure in the Signer section. Before character reading, Singh *et al.* [13] perform an image enhancement and correction procedure that includes multiple phases such as image acquisition, preprocessing, grey scaling, and segmentation scanning (crop image on the OCR target). Following this step, character reading is performed with a trained model from Tesseract OCR, and the results are collected from OCR via the post-processing stage. This overall process is performed for each document and all C1 papers. Figure 8 depicts the OCR steps.

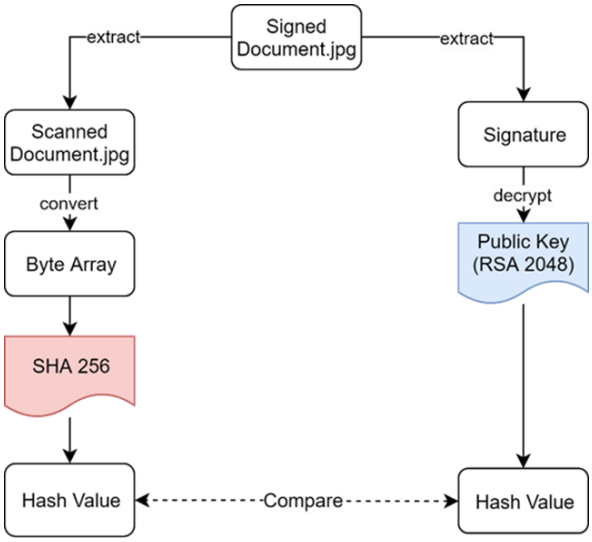

Figure 7. Image Document Verification Scheme

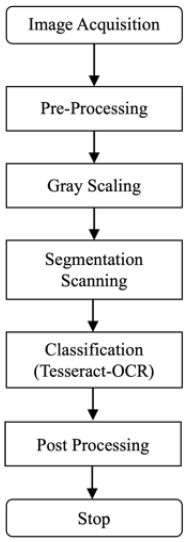

Figure 8. Image Document OCR Scheme

To increase the performance of OCR reading results, the preprocessing stage tries to improve picture quality and image alignment. Getting rid of noise improves image quality. Alignment is meant to ensure that the coordinates read match a specified template. An affine transformation is used for alignment, which requires three points of origin and three points of destination as parameters. The three dots in document C1 take advantage of the position detection pattern (PDP) present in the document's corners. The position of the

PDP point may be determined using a function in the CV2 library called findContours. This function can identify the contours in a picture. The acquired contours are then found to have a pattern comparable to the PDP. he steps of the Sanguansat method are described in Figure 9.

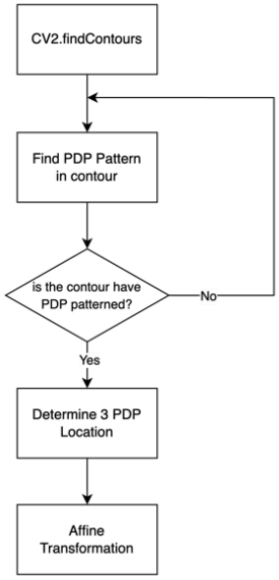

Figure 9. Image Document Alignment Method

The grey scaling stage clarifies the text on the picture. Following that, the picture is clipped at the segmentation scanning step at the same position as the template, which is a variable for the Tesseract OCR process. The Tesseract Python library is used in the OCR process. This library accepts picture files as input and creates text as output. Tesseract employs the LSTM algorithm with a setup option for reading single characters and classification with Tesseract's default model. The accuracy of this library will increase if quality enhancement, also known as preprocessing in this study, is carried out. The next phase is post-processing, which involves combining individual text segments into an Excel file for additional performance analysis.

Based on the results of the research stages of the Descriptive Study 1, the functional requirements for the application design process in this study were grouped based on the application section as follows:

The Signer section has several functions: Preprocessing for image quality improvement and alignment; Grayscaling to clarify writing; Segmentation scanning is used to remove unneeded image components and crop parts of the image at the specified variable; Classification to perform OCR; Postprocessing to collect the text results into an Excel file; Signing to sign the document with the private key; Embedding signature to embed the signature value into the JPG file.

The Verifier section has two functions: Reading the signature to read the signature value in the JPG file and verification to authenticate the integrity of a JPG file using the public key.

3.7. Design

Use Case Diagram and Data Flow Diagram are used to model the system. The use case diagram can be seen in Figure 10. A context diagram is built before producing a Data Flow Diagram (DFD) that illustrates all external entities and managed data flows from or into the system represented in Figure 11. The DFD fragment is then constructed based on the context diagram and represents each use case that has been created. This DFD fragment is subsequently referred to as the level 0 DFD, as illustrated in Figure 12.

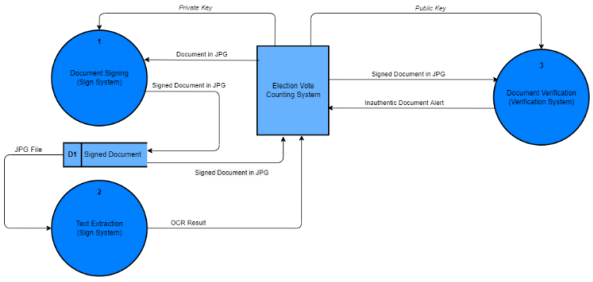

Figure 10. Use case diagram

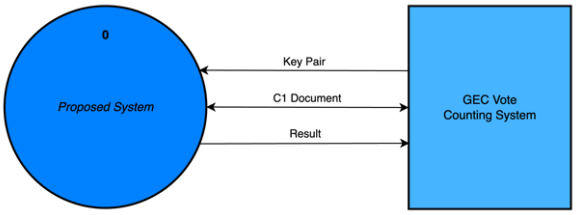

Figure 11. Context diagram

The diagram depicts how the built system interacts with external entities. The system communicates with the Situng GEC entity, either the Regency/City section or the Central section, to acquire input data in the form of key pairs and C1 documents and to deliver output data in the form of OCR resulting in data and document authenticity.

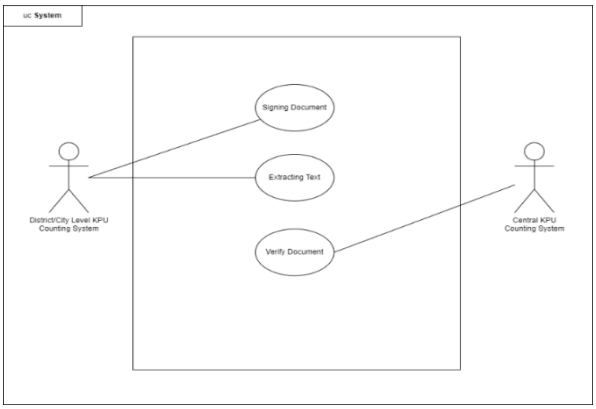

Figure. 12 Data flow diagram level 0

Figure 12 is a level 0 DFD that describes the interaction of each process in the system using the use cases from Figure 10. Previously, this research had two products: a Signer Section and a Verifier Section. The Signer

Section consists of two processes: document signing and text extraction, while the Verifier Section consists of one process: document verification. The document signing mechanism communicates directly with Situng GEC external entities, especially Situng GEC in Regencies/Cities, by accepting private keys and C1 document files in JPG format. The D1 Signed Document data storage receives output from this operation. The signed C1 document is extracted from the D1 data storage and then subjected to a text extraction procedure, which produces OCR results in the form of text.

In this study, text was gathered in the form of an Excel file for analysis. The OCR findings, along with the signed C1 document from data store D1, are delivered to the Situng GEC external entity, namely the Situng GEC in the regency/city. The document is forwarded from the Situng GEC system to the Central Situng GEC for document verification. The document verification process interacts with Situng GEC entities, notably Situng GEC at the center, to receive input in the form of C1 papers signed in JPG format and offer output in the form of alerts if the validated documents are not legitimate. The verified documents will be returned to the Situng GEC entity at the center.

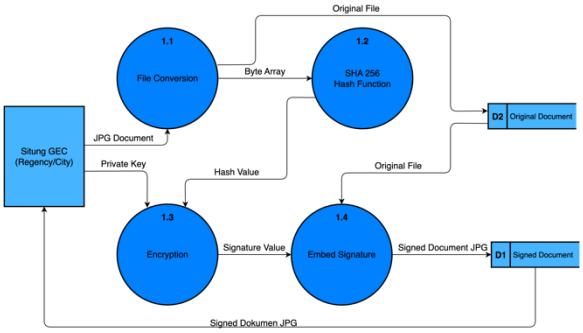

Figure 13. Data Flow Diagram Level 1 Document Signing

Based on DFD level 0, each use case is converted into more comprehensive phases on DFD level 1. One use scenario is detailed in one DFD level 1, which is depicted in Figures 13, 14, and 16. Figure 13 depicts DFD level 1 for document signing use cases, which is divided into four processes: file conversion, hash function, encryption, and embed signature. The file conversion process receives input from the Situng GEC external entity in the form of a C1 document file in JPG format and converts it to a byte array to compute its hash value in the hash function process. The hash function method employs the SHA 256 algorithm to generate a hash value, which is subsequently encrypted. The encryption procedure employs the RSA 2048 technique, getting input in the form of a private key from the Situng GEC external entity and creating a signature value. The signature value is then added to the original file, which was previously retrieved during the file conversion procedure and is temporarily held in the D2 Original Document data storage. The signature embed process attempts to inject the signature value into the JPG file's exif data. This method produces a signed C1 document JPG file, which is subsequently saved in the D1 Signed Document data storage. Document C1 from data store D1 is subsequently supplied to the external entity Situng GEC district/city section.

Figure 14 depicts a level 1 DFD for the text extraction use case based on the procedures from Singh et al., which are separated into five primary processes: preprocessing, grey scaling, segmentation, classification, and post-processing. The D1 Signed Document data store provides input to the preprocessing procedure in the form of a C1 document file in JPG format. The goal of the technique is to increase image quality and match photographs with templates.

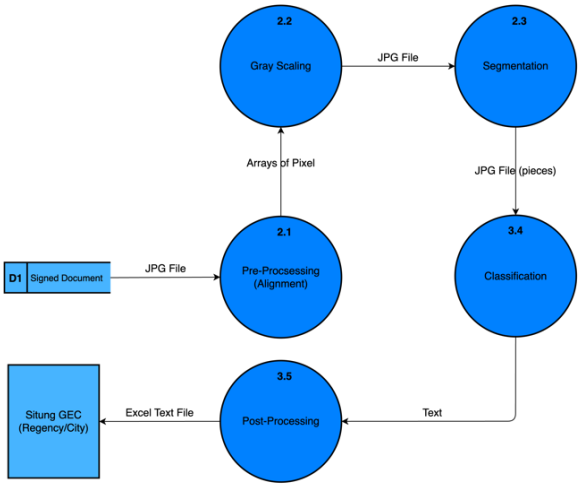

Figure 14. Data Flow Diagram Level 1 Document Extraction

Figure 15 shows a more detailed explanation of the preprocessing procedure in DFD level 2. Gray Scaling is a technique for making images binary colored and enhancing text structure to make them more legible. The segmentation method involves cutting parts of an image into the target column where the character reading procedure will take place, resulting in image fragments. The classification process employs a trained OCR model from the Pytesseract library, which is based on Tesseract-OCR. This process produces separate text and post-processing is carried out to collect the text in the form of an Excel file for analysis. The text output is subsequently delivered to the external organization Situng GEC, district/city division.

Figure 15 depicts the alignment preprocessing procedure in four phases. The first step takes an image file as input and looks for contours in the picture using the CV2.findContours function. This technique generates data in the form of a collection of contours and searches for those in the collection that have a PDP pattern. The position of the discovered PDP contour is then identified and used as a parameter in carrying out the affine transformation. The affine transformation yields an image that is aligned with the template,

allowing the following operation to better target the OCR region.

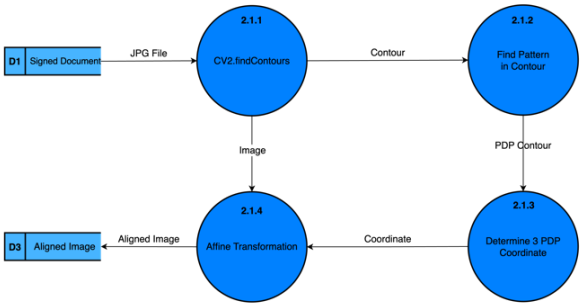

Figure 15. Data Flow Diagram Level 2 Preprocessing

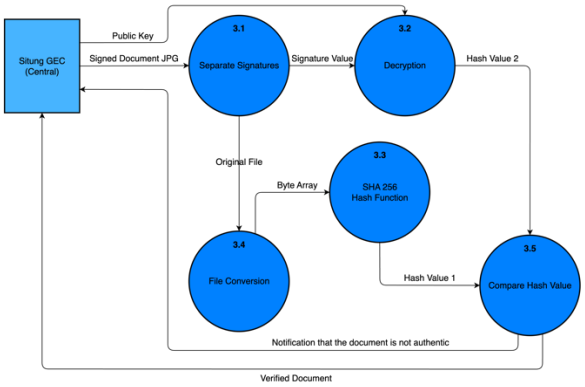

Figure 16. Data Flow Diagram Level 1 Document Verification

Figure 16 depicts a level 1 DFD for the document verification use case, which is separated into five processes: signature separation, decryption, hash functions, file conversion, and hash value comparison. The process of separating signatures is fed by external Situng GEC entities, particularly the core section of Situng GEC, in the form of a C1 document file signed in JPG format. The signature value and the actual file are separated during the procedure. The original file will be converted into a byte array, and the hash value calculated will be compared with the hash value of the signature, which is decrypted using the RSA 2048 algorithm with a public key obtained through interaction with external Situng GEC entities, particularly the Central part of the Situng GEC. Along with the papers, the Situng GEC external entity receives the results of document authentication.

3.8. Implementation

The C1 document used in this study has been adapted to the needs of the system. There are three position detection patterns, also known as PDP, that distinguish the documents used in the original process, while the C1 document's content remains unchanged. According to the C1 document, the items that are required for the tabulation process in Situng based on the implementation of Situng in the 2019 election: The total number of registered voters; The number of people who have the right to vote; The number of votes for each candidate; The total number of valid votes; The total number of invalid votes; The total number of valid votes and invalid votes

Figures 17 and 18 show the C1 document template for the system in this study, as well as the areas that are clipped or cropped, which are indicated by red lines. The six requirements listed above are shown by a dotted red line. The intended number of OCR reading places is six on page one and fifteen on page two.

In the signing process, the signer section in this program receives input in the form of the path of the file to be signed, the path of the directory that will contain the signed file, and the path of the private key during the signing process. Each file in the directory is signed according to the stages described in the design sub-section. The file is converted to bytes and duplicated into the signed files directory. The byte value is a calculated hash value that is encrypted in .pem format with a private key. The encryption results in a signature value that is inserted into the *EXIF* data of the JPG file in the previously signed file directory. Each directory containing page 1 and page 2 files is divided into two looping processes.

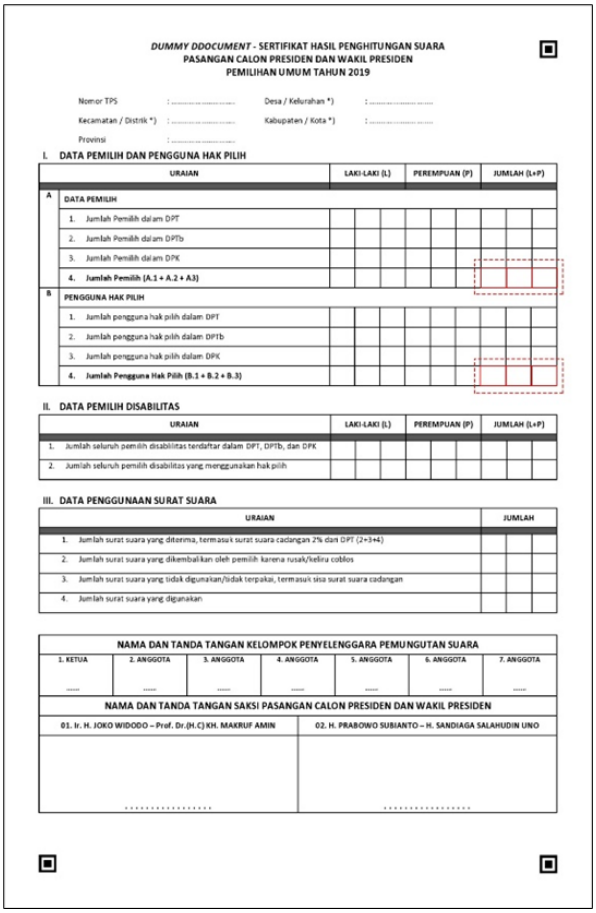

Figure 17. C1 Document Template Page 1 and Crop Location

The signer section of this program receives input in the form of file paths to be extracted during the data extraction process. To improve image quality and alignment, files are preprocessed. The image is then grey-scaled to clarify the writing and remove unnecessary parts of the image. Following that, the text classification stage is completed, followed by the segmentation stage to cut the target coordinates.

Because the target text location differs for each page, the looping process is divided into two for each directory containing page 1 files and page 2 files. The data extraction results are saved in an Excel file.

In the verification process, the verifier section receives input in the form of the file path to be verified and the public key path. Each file is verified by the stages outlined in the design sub-section. The file signature value is taken, which is then decrypted with the public key into a hash value. The same file is converted to bytes and the hash value is calculated. The two hash values are compared, and if the results are the same, then the file is authentic.

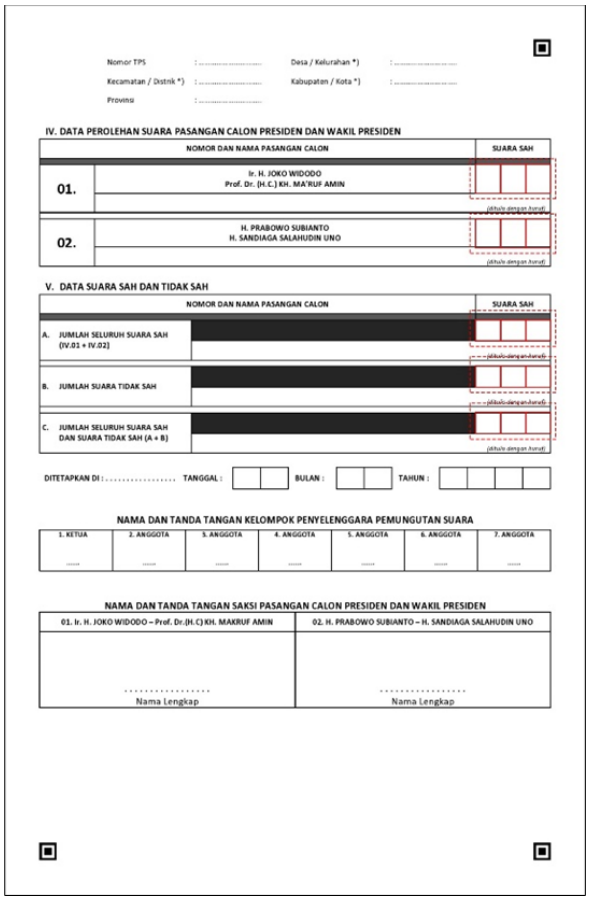

Figure 18. C1 Document Template Page 2 and Crop Location

Figure 19 shows a collection of sample data used in application testing. It can be seen that documents have different scanning qualities and different handwriting characteristics.

3.9. Application Testing

In this study, the black box testing method was used to unit test the application. The results of unit testing show that the functions in the application have worked well according to the parameters and the resulting output. Table 1 summarizes the results of unit testing. The summary table results demonstrate that the functions in the program operated successfully based on the parameters and the resultant output.

The comprehensive unit testing detailed in Table 1 showcases the individual reliability and robustness of each module within our application. The successful testing of crucial functions like Preprocessing, gray scaling, and Segmentation is particularly significant, as these steps play a vital role in enhancing OCR accuracy. This rigorous testing framework ensures that each component not only meets its designated role but also contributes effectively to the overall functionality and reliability of the OCR process in handling C1 documents

Table 1. Unit Testing Result

No	Section	Function	Description
1.	Signer	Conversion()	Successful
		Hash_func()	Successful
		Encryption()	Successful
		Embed_signature()	Successful
		Preprocessing()	Successful
		Gray_scaling()	Successful
		Segmentation()	Successful
		Classification()	Successful
2.	Verifier	Read_signature()	Successful
		Conversion()	Successful
		Hash_func()	Successful
		Verify()	Successful

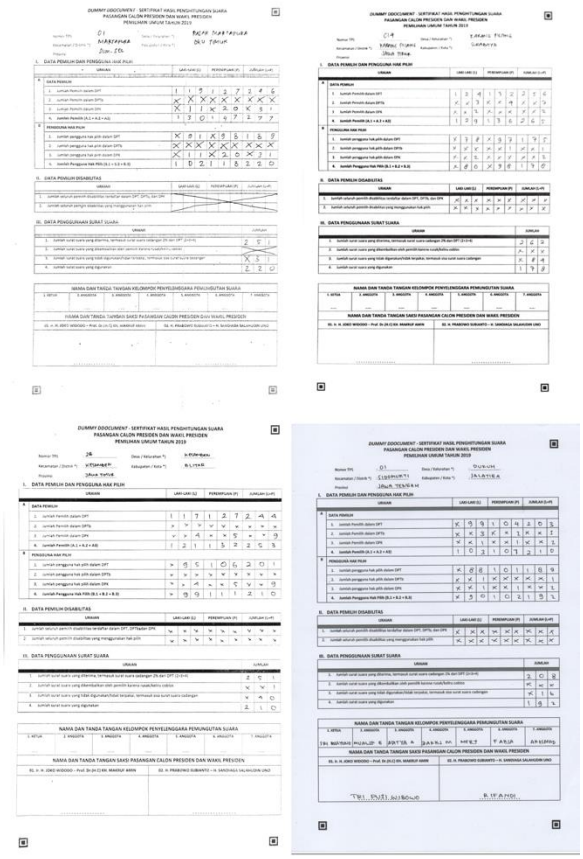

Figure 19. Data Samples of C1 Document Template

In this study, scenario testing is used to perform integration testing on the application. The results of

integration testing show that the results obtained from all scenarios in the three use cases can be met.

Requirements testing is used to perform system testing on the application. These results indicate that the application has met all the requirements that have been determined. Table 2 shows the results of request testing for functional requirements while Table 3 shows the results of requirements testing for non-functional requirements. Both results tables reveal that the application satisfied all of the requirements.

Table 2. Functional Requirements Testing Results

No	Parts of Program	Functional Requirements	Description
1.	Signer part	Preprocessing function for image quality improvement and alignment with templates.	Fulfilled
		Grey scaling function to clarify writing	Fulfilled
		Segmentation scanning function to remove unnecessary image components and cut parts of the image on the specified variable.	Fulfilled
		Classification function to perform OCR.	Fulfilled
		Post-processing function to collect text results into an Excel file.	Fulfilled
		The signing function signs the document with the private key.	Fulfilled
		Embed signature function to embed signature values into JPG files.	Fulfilled
2.	Verifier part	Read signature function to read signature values in JPG files	Fulfilled
		Verification function to authenticate the integrity of JPG files using a public key.	Fulfilled

The exhaustive testing of functional requirements, as encapsulated in Table 2, demonstrates the application's robust capability in fulfilling its designed purpose. Each function, rigorously evaluated against its predefined criteria, exhibits the application's proficiency in managing and executing complex tasks. This not only attests to the application's operational effectiveness but also its alignment with the projected functional objectives, ensuring a comprehensive and reliable solution for C1 document data extraction.

The fulfillment of the non-functional requirements, as meticulously detailed in Table 3, reassures the application's comprehensive quality attributes. This encompasses not only its performance and security but also its operational stability and efficiency. The rigorous validation of these parameters is indicative of the application's resilience and reliability, especially in the context of handling sensitive and high-stake electoral data. Such an extensive evaluation of non-functional aspects ensures that the application is not only functionally adept but also robust and secure in its practical deployment.

Table 3. Non-Functional Requirements Testing Results

No	Non-Functional Requirements	Description
1.	The application consists of a signer application for document signing and data extraction as well as a verifier application for document verification.	Fulfilled
2.	Digital Signature uses the RSA 2048 algorithm and hash function uses the SHA 256 algorithm.	Fulfilled
3.	Backend-based application with the Python programming language.	Fulfilled

3.10. Performance Analysis

After being run, the application generates predictive data, which is then calculated and classified into three categories using the previously described method in subsection 3.4. The performance of OCR is shown in Figure 20 and Table 4.

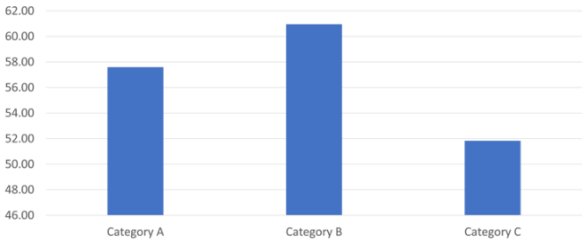

Figure 20. OCR performance graph

Table 4. OCR Performance by Data Category

No.	Test Data Category	Amount of data	Recall
1.	Category A	1920	57.60 %
2.	Category B	6120	60.96%
3.	Category C	7800	51.83%

Using the formula outlined in section 2.4, the recall result represents a measure of the percentage of correctly identified characters about the total number of characters available. According to Table 4, the application's recall in reading handwriting from document C1 is 57.60% for category A test data, 60.96% for category B test data, and 51.83% for category C test data. The results obtained indicate that even when using the model from the Tesseract-OCR Engine, the application's ability to read characters correctly remains limited.

The detailed analysis of OCR performance by data category, as exhibited in Table 4, sheds light on the intricate relationship between image quality and OCR accuracy. The varying recall rates across different categories underscore the pivotal influence of image quality on the OCR process. This insight is instrumental in guiding future improvements in image preprocessing methodologies, aiming to optimize OCR accuracy irrespective of the input image quality. Such a detailed performance breakdown is crucial for understanding the

limitations and potential areas of enhancement in OCR technologies, particularly in the specialized context of C1 document processing.

4. Conclusions

In this study, we developed the C1 document data extraction application using the waterfall SDLC method, which has undergone a systematic and thorough process. It encompasses planning, analysis, design, implementation, and testing phases. The application is divided into two distinct sections, signer and verifier, with specific functional and non-functional requirements. Using the Tesseract-OCR Engine for character recognition, combined with digital signature capabilities, provides a comprehensive solution to reduce the human error factor that might result in miscalculations and inaccurate processes.

However, it is worth mentioning that the application's performance might be improved. The percentages of characters reading recognized from C1 documents show that recall with image quality in category A is 57.60% of 1,920 characters, 60.96% of 6,120 characters, and 51.83% of 7,800 characters, indicating that even when using the Tesseract-OCR Engine model, the application's ability to read characters correctly remains limited.

This study may contribute to the field of automatic data extraction by improving the performance of the Tesseract-OCR engine in reading and processing C1 documents. The approach integrates advanced image preprocessing techniques and RSA-2048 digital signature for data integrity, setting a new benchmark in the accuracy and reliability of OCR applications in election data processing. Unlike previous research that primarily focused on OCR accuracy in general contexts, our study specifically tailors the Tesseract-OCR engine to the unique challenges of C1 document data extraction. We employ specialized image preprocessing methods and digital signature verification, which are not extensively explored in existing literature, to ensure high accuracy and data integrity specific to election document processing.

Acknowledgement

This research was supported by Politeknik Siber dan Sandi Negara, Bogor, West Java, Indonesia. The authors are very grateful to the referees, whose valuable comments and suggestions resulted in a better-organized and improved paper.

References

[1] The General Election Commission of the Republic of Indonesia, Decree Number 536 of 2019 GEC concerning Instructions for Use of the 2019 General Election Vote Counting Information System.

[2] A. Ardipandanto, "Problems of Implementing the Connective Elections in 2019," Research Center of the Indonesian House of Representatives Expertise Board, p. 6, 2019.

[3] “GEC Finds 269 Situng Data Input Errors,” CNN Indonesia, 2019. https://www.cnnindonesia.com (accessed Nov. 21, 2019).

[4] "About Data Input Errors, GEC Admits There Was a Human Error," CNN Indonesia, 2019. https://www.cnnindonesia.com (accessed Nov. 21, 2019).

[5] FC Farisa, “GEC: The Situng Data Calculation Process Missed the Target,” Kompas, 2019. https://nasional.kompas.com (accessed Nov. 21, 2019).

[6] M. A. Awel and A. I. Abidi, “Review on Optical Character Recognition” in International Research Journal of Engineering and Technology Vol. 6 Issue 6, 2019

[7] V. Geetha, Ch. V. V. Sudheer, A. V. Saikumar, and C. K. Gomathy, “Optical Character Recognition” in Journal Of Engineering, Computing & Architecture, 2022

[8] V. Sellam, A. Aruna, A. Joseph, S. Rahul, A. Rahul, “Optical character recognition using localization techniques” in AIP Conference Proceedings Volume 2463 Issue 1, 2022

[9] K. M. Sai, H. Chandrika, K. Bebe, G. S. R. Pramila, G. S. Rao, “Optical Character Recognition using CRNN” in International Journal of Innovative Technology and Exploring Engineering (IJITEE) Volume 9 Issue 8, 2020

[10] F. Shafait and R. Smith, “Table detection in heterogeneous documents,” in Proceedings of the 8th IAPR International Workshop on Document Analysis Systems - DAS '10, Boston, Massachusetts, 2010, pp. 65–72.

[11] Min Cai, Jiqiang Song, and MR Lyu, “A new approach for video text detection,” in Proceedings. International Conference on Image Processing, Rochester, NY, USA, 2002, vol. 1, pp. I-117-I–120.

[12] P. Sanguansat, “Robust and low-cost Optical Mark Recognition for automated data entry,” in 2015 12th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), Hua Hin, Cha-am, Thailand , Jun. 2015, pp. 1–5.

[13] A. Singh and S. Desai, “Optical character recognition using template matching and backpropagation algorithm,” in 2016 International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, Aug. 2016, pp. 1–6.

[14] R. Smith, “An Overview of the Tesseract OCR Engine,” in Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2, Curitiba, Parana, Brazil, Sep. 2007, pp. 629–633.

[15] R. Karpinski, D. Lohani, and A. Belaid, “Metrics for Complete Evaluation of OCR Performance,” The 22nd International Conference on Image Processing, Computer Vision, & Pattern Recognition, p. 8, Jul. 2018.

[16] AJ Menezes, PC Oorschot, and SA Vanstone, Handbook of Applied Cryptography, 1st ed. USA: CRC Press Inc., 1996.

[17] A. Regenscheid, “Digital Signature Standard (DSS): Elliptic Curve Domain Parameters,” National Institute of Standards and Technology, preprint, Oct. 2019.

[18] LTM Blessing and A. Chakrabarti, DRM: A Design Research Methodology. Springer, London, 2009.

[19] BPN, “Uncovering the Pattern of GEC Fraud Using IT Forensics,” p. 32, 2019.

[20] A. Dennis, BH Wixom, and D. Tegarden, “System Analysis & Design With UML Version 2.0.,” United States of America: John Wiley & Sons, 2009.

[21] A. Dennis, BH Wixom, and RM Roth, Systems Analysis and Design, 5th Edition. WIley, 2012.

[22] R. Minister of Home Affairs, Regulation of the Minister of Home Affairs of the Republic of Indonesia Number 72 of 2019.

