Ingénierie des Systèmes d'Information
Vol. 30, No. 5, May, 2025, pp. 1123-1134
Journal homepage: http://iieta.org/journals/isi

Software Requirement Specification Compatibility Design Method Through Use Case Diagram Artifacts Using Text Mining on Akuonline Learning Application

Yudi Priyadi

Software Engineering, Telkom University, Bandung 40257, Indonesia

Corresponding Author Email: whyphi@telkomuniversity.ac.id

https://doi.org/10.18280/isi.300502

Received: 3 November 2024
Revised: 11 May 2025
Accepted: 24 May 2025
Available online: 31 May 2025

Keywords:
SRS, requirement elicitation, steps performed, compatibility, similarity, use case diagram, text mining

ABSTRACT

Software Requirement Specification (SRS) is a document detailing the requirements for software development created from the Requirements Elicitation process between the Software Developer and the Client. Differences in interpreting requirements can occur between developers and clients. This study aims to determine and calculate the compatibility value of a user requirement with the software being built by measuring the similarity of the Requirement Elicitation results with the software modeling. Compatibility can be achieved through a Text Mining approach involving Text Preprocessing activities on the Requirement Specification results. Data extraction on Requirement Elicitation through interviews is achieved by performing Text Preprocessing and similarity, identifying and finding FR compatibility with the Steps Performed. Four conclusions are drawn from this research. First, based on the SRS named "Akuonline Learning Application," 11 FRs have been identified. Based on Cosine Similarity calculations comparing the FR matrix and the Step matrix performed, and validation results through Gwet's AC1 reliability measurement, the Python version = 0.299, while the questionnaire version = 0.379, showing a difference of 0.08. Referring to Gwet's AC1 Value Interpretation, the validity and reliability values fall into the "Fair agreement" category.

1. INTRODUCTION

In a Software Requirement Specification (SRS) documentation, some requirements statements must be carried out by software. This statement is a collection of specifications derived from the results of requirements elicitation activities between software developers and clients [1-3].

There are differences in interpreting a description of software development, and this occurs because each individual has a different level of system thinking in experience, learning process, insight, intuition, and assumptions in understanding interactions system [1]. Differences in the interpretation of requirements in software development can occur between software developers and clients. The main reason for this difference is that each individual has a different level of system understanding based on experience, learning processes, insights, intuitions, and assumptions in understanding system interactions. Software developers may have a deep technical perspective and focus on implementation, while clients are more oriented towards business needs and desired functionality. In addition, ineffective communication and a lack of clear documentation can exacerbate these differences in interpretation. For example, technical terms developers use may not be fully understood by clients, or client needs not described in detail can lead to different interpretations. The result of these differences is a decrease in the quality of SRS documentation, which can result in the need to repeat the Requirement Elicitation process from the beginning, either verbally or in writing, to ensure there are no errors or gaps in the translation of client needs into software specifications. Therefore, this study is important to identify and overcome these interpretation differences to improve the quality and reliability of SRS documentation. [1, 4, 5].

Use case diagram (UCD) is an artifact in the SRS documentation. This diagram must describe each use case that describes all events in full regarding the activities carried out by all actors on UCD. The description of UCD is in the form of a step performed Statement in the form of text so that data processing can be done through text analysis activities [1, 4].

In-text mining for a text analysis activity, preprocessing is carried out whose process can be adjusted depending on the type of text data and the required results. There is an extraction process in carrying out text preprocessing activities carried out [6-8], namely: Case Folding, Tokenization, Stopword Removal, and Stemming.

This study aims to determine and calculate the compatibility value of a user requirement with the software to be built by measuring the similarity of the results of the Requirement Elicitation activity with the software modeling contained in the SRS document. Compatibility can be done through a Text Mining approach in the form of text preprocessing activities on the results of the Requirement Specification, which is then compared with a description in the form of Step performed from each use case.

In this research, there are contributions and novelties as follows:

(1) Develop the concept of text mining in text pre-processing activities to extract SRS documentation.

(2) Application of similarity formulation for weighting the value of a text description.

(3) Generate methods for compatibility client requirements for software built by developers.

(4) Collaborating on the concepts of text mining, semantic similarity, and reliability.

2. RELATED WORKS

This section describes all the basic concepts related to processing text data for software documentation, weighting, similarity, and also how to validate research results.

2.1 SRS

SRS is a document that describes in detail the requirements in software development created from the results of the Requirements Elicitation process between Software Developers and Clients [1, 9]. From the Requirement Elicitation activities, it can be used as a reference in software development. This reference can be a set of Functional Requirement (FR) and Non-Functional Requirement (NFR) statements.

If observed in an FR group, then there is the use of words in expressing software development needs, ranging from essential to optional needs; usually, the characteristics of the words have to, can, and may [10, 11].

In this study, several related works have been cited to provide the theoretical basis and methodology used. The following is a further analysis of the similarities and differences between these works and current research:

A text-mining approach [1] was employed to evaluate the similarity between Requirement Elicitation and Requirement Specification results, using Cosine Similarity as the primary measurement tool. This work provides an initial contribution to the application of text analysis in software requirements. However, the approach did not address aspects of statistical validity and reliability. The current study builds upon this foundation by integrating Gwet's AC1 method, resulting in a more in-depth and academically rigorous analysis. This study highlights the need for methodological enhancement in prior research that focused solely on textual similarity.

Various elicitation techniques and challenges in mobile application development were identified, with an emphasis on the importance of SRS quality through a practical approach [9]. Although the study offers valuable insights into the communication process between developers and users, its approach remains descriptive and does not explore the quantitative analysis of requirement suitability. The current study addresses these limitations by systematically applying Text Mining techniques to measure the similarity and validity of requirements. This synthesis illustrates how a data-driven approach can enhance elicitation outcomes that were previously limited to qualitative insights.

While existing approaches have examined input/output changes in functional requirement test cases [10], they fail to assess semantic alignment between requirements and implementation. Our enhanced methodology builds upon this foundation by introducing textual similarity analysis (TF-IDF with Cosine Similarity) and statistical validation (Gwet's AC1) to provide more comprehensive verification. The critique of previous work lies in its limited attention to semantic dimensions and the absence of validity measures in assessing requirement compatibility, which this study addresses as a central focus.

The classification approach for system and software requirements [11] enhances their understanding and management. While this approach offers a solid conceptual framework, classification alone is not sufficient to ensure alignment between requirements and technical implementation. The current study integrates this classification approach with Text Mining-based analysis to quantitatively assess the similarity and validity of requirements. In doing so, it not only synthesizes the conceptual foundation but also critiques its limitations in delivering a more objective and measurable evaluation mechanism based on data.

2.2 UCD

Artifact UCD is made based on FR from the results of Requirement Elicitation. Each FR should be made into one Use Case in UCD [6, 12]. So the number of FR will be the same as the Use Case. In this UCD, all actors' involvement in a Use Case can be observed. After designing the UCD, a description is needed to explain the process that occurs for each Use Case. For this reason, a Use Case Description (scenario) must be made for each Use Case so that each step in a Use Case can be explained in detail through one Step Performed [6, 13, 14].

Prior work [6] developed an application to validate textual content and derive Functional and Non-Functional Requirements from documentation. While this aligns with the current research in its use of text analysis for requirement validation, the present work expands the scope by emphasizing semantic similarity and integrating Text Mining techniques to enhance the depth and accuracy of the validation process.

The development of an automated Use Case extractor using an XML Parser shares a common objective with this study—streamlining the modeling of requirements [12]. However, instead of relying solely on structural automation, the current approach leverages Text Mining to assess both the semantic alignment and the reliability of the extracted requirements, offering a more nuanced and data-driven evaluation.

The use of text processing software to map architectural requirements to quality attributes demonstrates the potential of textual analysis in requirement engineering [13]. Building upon this foundation, the present study incorporates similarity metrics to not only validate but also quantify the alignment between requirements and system attributes, thereby reinforcing the analytical rigor and practical relevance.

The automation of process performance analysis using textual descriptions and event logs [14] highlights the utility of text-based data in operational evaluation. Extending this concept, the current research applies Text Mining to examine requirement compatibility, placing greater emphasis on semantic similarity and statistical validation to support more informed design decisions.

2.3 Text preprocessing

The initial step that must be done in the Natural Language Processing method is Text Preprocessing. This stage must be carried out in order to prepare unstructured text data to be processed. Text Preprocessing activities implemented in this study include [15, 16]:

(1) Case Folding for converting to lowercase.
(2) Tokenization in breaking sentences into words.
(3) Stopwords removal in order to eliminate words that have no meaning in the Text Mining process.
(4) Stemming/Lemmatization to make the root word.

Text preprocessing techniques were used to assess the similarity between Requirement Elicitation and Requirement Specification in the CINEMALOKA application [15]. This text Preprocessing aligns with the current research in terms of methodology, particularly in leveraging preprocessing steps to analyze requirement texts. What sets the present study apart is its incorporation of statistical validation—specifically Gwet's AC1—to ensure the reliability and accuracy of similarity measurements, thereby extending the analytical depth beyond surface-level comparisons.

The role of text preprocessing in sentiment analysis has been explored using techniques including case folding, tokenization, stopwords removal, and stemming/lemmatization [16]. While employing similar text preprocessing methods, this study applies them to evaluate textual similarity within Software Requirement Specifications (SRS), contrasting with prior sentiment analysis applications focused on emotional tone and opinion mining [16]. Our work shifts the analytical purpose toward semantic alignment and requirements validation.

2.4 Method for weighting and similarity

Term Frequency–Inverse Document Frequency is a method used as a weighting factor for text mining. The product of the weighting of Term Frequency (TF) with Inverse Document Frequency (IDF) of a term/words can be done by using the following formula [17, 18]:

$$tf_i = \frac{freq_i(dj)}{\sum_{i=1}^{k} freq_i(dj)} \tag{1}$$

$$idf_i = \log \frac{|D|}{|\{d: t_i \in d\}|} \tag{2}$$

$$(tf - idf)_{ij} = tf_i(d_j) * idf_i \tag{3}$$

Term/words that often appear are topics that are sought in a text mining activity. Furthermore, to find the similarity of the results of TF-IDF can use calculations using Cosine Similarity through the use of the following formula [19-21]:

$$Sim = \cos(a,b) = \frac{a.b}{\|a\|.\|b\|} = \frac{a_1b_1 + \cdots + a_nb_n}{\sqrt{a_1^2 + \cdots + a_n^2} + \sqrt{b_1^2 + \cdots + b_n^2}} \tag{4}$$

Textual data from NFR has been processed to evaluate similarity between Requirement Citations and Deployment Diagrams, with proposed improvements for SRS documentation [17]. While the methodology shares common ground with the current study through the use of TF-IDF and Cosine Similarity, this research advances the approach by incorporating statistical validation techniques to ensure both reliability and analytical depth.

While employing computational methods similar to those used in patient support forum analysis (TF-IDF and Cosine Similarity) [18], this study applies them in a significantly different context. The current research adapts these techniques to the domain of Software Requirement Specifications, focusing on semantic alignment and validation of technical documentation rather than user-generated content.

The identification of causal loop variables for systems thinking has been explored using text mining approaches [19]. While sharing methodological similarities, the current study applies text mining to evaluate requirement similarity and validity in software engineering, demonstrating a more targeted application of these techniques.

Cosine Similarity measures have shown adaptability across domains, including their application to image fuzzy sets [20]. Building on this mathematical foundation, our work specifically applies the metric to textual data in SRS, focusing on semantic precision and compatibility assessment rather than visual pattern recognition.

Sentence similarity detection using Cosine Similarity has been investigated for the Malayalam language [21]. While employing the same technical approach, this study adapts the method to evaluate textual congruence between software requirements and their modeled representations, thereby advancing system specification validation.

2.5 Validity and reliability

Gwet's AC1 method is used to test validity and reliability based on the results of the agreement index between two experts/experts who tested a recommended situation [22-24]. Table 1 is the format of the questionnaire results between expert-1 and expert-2. Variations A, B, C, and D are the number of answers based on filling out the questionnaire.

Table 1. Elicitation statement results [1, 22-24]

Expert 1	Expert 2		
	Yes	**No**	**Total**
Yes	A	B	B1 = A+B
No	C	D	B2 = C+D
Total	A1 = A+C	A2 = B+D	N

Furthermore, the formula used to calculate it is as follows:

$$P = \frac{A + D}{N} \tag{5}$$

$$P_1 = \frac{(A_1 + B_1)/2}{N} \tag{6}$$

$$e(Y) = 2P_1(1 - P_1) \tag{7}$$

$$AC1 = \frac{P - e(Y)}{1 - e(Y)} \tag{8}$$

Table 2. Gwet's AC1 value index [1, 23, 24]

Kappa Index	Promotion Agreement
< 0.00	Less than chance-agreement.
0.01 – 0.20	Slight agreement.
0.21 – 0.40	Fair agreement.
0.41 – 0.60	Moderate agreement.
0.61 – 0.80	Substantial agreement.
0.81 – 1.00	Almost perfect.

For the measurement index of the value of the AC1 statistical calculation, we can refer to the Aggregation

Coefficient, which can be seen in Table 2.

Comparative analysis between Cohen's Kappa and Gwet's AC1 for inter-rater reliability assessment has been conducted in personality disorder evaluations [22]. While employing the same statistical methodology, this study adapts Gwet's AC1 for validating reliability in Software Requirement Specifications (SRS), demonstrating its cross-disciplinary applicability beyond clinical settings.

Inter-rater reliability variance under high-agreement conditions has been extensively analyzed [23]. Building on this statistical foundation, our research specifically applies Gwet's AC1 to assess requirement compatibility consistency in SRS documentation, providing a domain-specific implementation distinct from general agreement scenarios.

Comprehensive methodologies for inter-rater agreement measurement, including Gwet's AC1, have been well-documented across diverse contexts [24]. This study specializes these principles for textual compatibility evaluation in software engineering, particularly for SRS validation, showcasing how general reliability frameworks can be effectively tailored to technical documentation analysis.

By analyzing these similarities and differences, the current study highlights innovations in using various Text Mining techniques and similarity analysis to improve the quality and reliability of SRS documentation. This study also emphasizes the importance of validity and reliability of results through comprehensive statistical measurements.

3. DATASETS

In this study, the dataset used is a SRS document named "Akuonline Learning Application" (Table 3).

Table 3. Document labeling

Functional Requirement	Usecase Name	Step Performed	Document Labeling
FR01	Login		d1
FR01A	Student Login		d2
FR01B	Teacher Login		d3
FR02	Use Tutorial		d4
FR03	Attendance		d5
FR04	Access Main Features		d6
FR05	Access Entertainment Features		d7
FR06	Upload Material & Quiz		d8
FR07	Execute Quiz		d9
FR08	Upload Assignments		d10
FR09	Watch Video Material		d11
	Login	SP01	d12
	Student Login	SP01A	d13
	Teacher Login	SP01B	d14
	Use Tutorial	SP02	d15
	Attendance	SP03	d16
	Access Main Features	SP04	d17
	Access Entertainment Features	SP05	d18
	Upload Material & Quiz	SP06	d19
	Execute Quiz	SP07	d20
	Upload Assignments	SP08	d21
	Watch Video Material	SP09	d22
	Number of documents		22

Table 4. Summarizes the artifacts

Artifact	Description
Requirement Elicitation	Interview excerpt between the developer and the client
Requirement Specification	11 Functional Requirements (FR) and 9 Non-Functional Requirements (NFR)
Use Case Diagram	11 Use Cases that describe the interaction between actors and the system
Use Case Description	11 groups of Steps Performed as part of text data processing

Here is more complete background information about the dataset:

(1) Source: This dataset comes from the results of Requirement Elicitation conducted by software developers with clients. The elicitation process involves interviews and discussions to collect functional and non-functional requirements for the learning application to be developed.

(2) Scale: This dataset includes various artifacts used as reference data in software development. These artifacts include:

Requirement elicitation: The results of the elicitation process are in the form of interview quotes between developers and clients.

Requirement specification: Consists of 11 Functional Requirements (FR) and 9 Non-Functional Requirements (NFR).

Use case diagram: A diagram of 11 Use Cases describing the interaction between actors and systems.

Use case description: A description of each Use Case that includes 11 groups of steps performed as part of text data processing.

(3) Characteristics:

FR: These are functional requirements that the application must meet. Examples include user login, access to key features, and uploading materials.

NFR: These non-functional requirements include system availability, routine maintenance, and server technical specifications.

UCD: A diagram that describes the scenario of application usage by actors (students and teachers) in various situations.

Use case description: A detailed description of each Use Case that explains the steps taken by the actor in a particular scenario.

Based on the description of the dataset referred to in Table 3, there is a summary of the artifacts that are the most important part of the dataset used (Table 4).

4. METHODOLOGY

This section presents a process flow to describe all activities carried out in the research proposal. Based on the datasets that are used as references, the following are the steps and methods implemented to achieve the goals and results (Figure 1), namely:

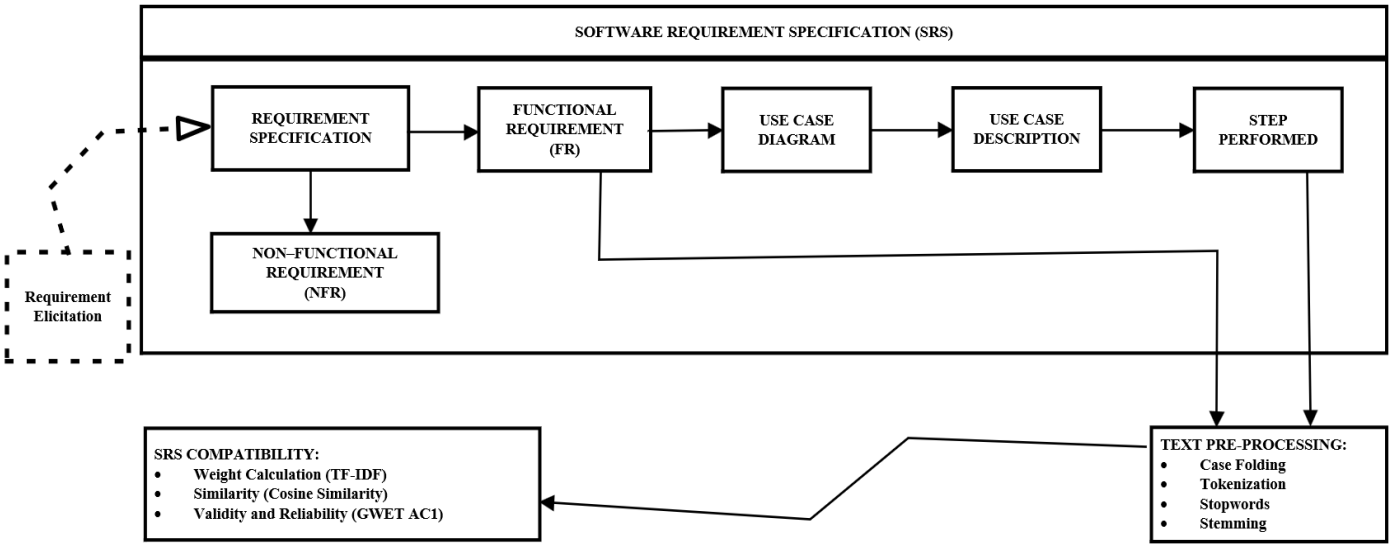


Figure 1. Methodology

(1) It begins with the Requirement Specification data identified previously, based on the results of the Requirement Elicitation.

(2) Perform extraction based on the results of the Requirement Specification on the FR section only.

(3) Based on the UCD artifact, then the Use Case Description extraction is carried out.

(4) Furthermore, based on all the Use Case Descriptions, the Step Performed can be identified, which will be used as datasets.

(5) In the Text Pre-Processing activity, all FR and Step Performed extraction results are carried out.

(6) The results of the Text Preprocessing will be processed to determine compatibility, which includes the Weight Calculation (TF-IDF) calculation stage, then look for the Cosine Similarity value, and the last stage is to carry out its validity/reliability (Gwet's AC1). Advantages of AC1 Gwet, namely:

Overcoming high agreement bias: AC1 Gwet is more accurate in conditions where the level of inter-rater agreement is very high, avoiding underestimation that often occurs in Cohen's Kappa.

Stability: AC1 Gwet provides more stable and consistent results in various data distribution conditions.

Ease of interpretation: AC1 Gwet values can be easily interpreted using clear agreement categories, such as "Fair agreement", "Moderate agreement", and "Almost perfect".

By using AC1 Gwet, this study can ensure that the validity and reliability of the SRS compatibility measurement results through the UCD artifact are more accurate and reliable.

5. RESULT AND DISCUSSION

This section describes the implementation of all the stages contained in the research methodology. There is an explanation of how several concepts are collaborated, starting from SRS, Text Mining, and the validation process.

5.1 Requirement elicitation as generating requirement specification / requirement statements

Based on the SRS named "Online Learning Features Application" and the stages of activity on the research methodology. Next is to analyze a snippet of an interview resume used as the beginning of dataset collection. The results of the analysis of this interview resume snippet are in the form of free conversation sentences between the developer and the client, which will be used in the data mining or text mining process, as shown in Table 5.

Based on Table 5, a Requirement Specification or Requirement Statements is made, consisting of Functional Requirements (FR) and Non-Functional Requirements (NFR). The results at this stage will be processed in Text-Preprocessing activities tailored to the needs of the next stage. The number of FRs generated based on interviews was eleven (Table 6), while for the NFRs, there were nine (Table 7).

5.2 Use case diagram as generating use case description/use case scenario

The next stage is to design a Use Case Diagram based on Table 6. For this diagram, the Use Case is made according to the number of FRs, eleven. At this stage, identification is carried out for use cases based on the numbering sequence in the Ascending Requirement Specification table. So it can be identified Use Case as in Table 8.

After knowing the naming of the use case and the order in which it is done, the next stage is to design a Use Case Diagram that can show the involvement of the actors in the use

case and the sequence of processes for each use case. The design of the diagram for the application called "Online Learning Features" can be seen in Figure 2. Based on the picture, there are two actors named Student and Teacher who must log in before accessing the application menu according to their access rights.

Table 5. Snippet of an interview resume

Developer	Client
Hello, good afternoon, sir. We continued our discussion last week. For this online learning application, how many users will there be? Because last week was still uncertain.	Yes, sir, after we discussed it with our friends, we made sure that there were two users of this application, namely: teachers and, of course, all students.
Okay, sir. Later we will divide into two access rights in the application menu. Next, we will give all users to log in using a user and password.	Ohhh, so there will be a different application menu.
Yes, sir. That's right. The difference in the menu application features is based on log-in access rights.	Please make it later for access based on NIS for students and NIP for teachers.
Okay. What are the main features needed?	Well, here it is, from the results of yesterday's meeting, the teachers suggested that the main feature should be for direct discussions and questions and answers between students and teachers.
Alright, sir, I'll take note of that. Besides that?	Maybe you can add additional features for entertainment about education, such as Fun Facts and Educational Memes.
Okay, sir, now the subject matter and activity evaluation assess the students.	Wow, if this, I'm asking for a suggestion, but what is certain is that teachers must be able to upload materials and evaluate through quizzes.
Okay, sir. Later there will be a menu for teachers; then, the students have to take a quiz to evaluate their assessment.	So I remember this, Students must also be able to upload assignments given by the teacher on the application features.
Oke, I note this.	Good sir. There was a request from the principal for a feature for teachers to upload videos of learning materials. Then, the students can watch the video.
Okay, sir. Thank you for today; that seems to be enough for now. We'll do it first. I'll beg for more time later. You'll contact me again about the progress of our work. Regards.	

Table 6. Functional requirement

ID.	Functional Requirement Specification
FR01	Students and teachers must log in to the Online Learning Features application.
FR01A	Teachers must get the application menu according to their access rights.
FR01B	Students must get the application menu according to their access rights.
FR02	Students and teachers can use the application tutorial as a feature introduction and the Online Learning Features application.
FR03	Students must fill in attendance in the Online Learning Features application.
FR04	Students and teachers can access key features such as live discussions and Q&A between teachers and students.
FR05	Students can access entertainment features such as fun facts, world facts, and memes about education.
FR06	Teachers can upload materials and quizzes.
FR07	Students can take quizzes.
FR08	Students can upload assignments.
FR09	Students can watch the material in the application without leaving the Online Learning Features application.

Table 7. Non-functional requirement

ID	Non-Functional Requirement Specification
NFR01	The Online Learning Features application can be active for 24 hours.
NFR02	Admin can perform routine maintenance on the Online Learning Features Application.
NFR03	Admin should be responsible for application recovery and repair.
NFR04	If application maintenance is carried out, the Online Learning Features Application will notify teachers and students 24 hours before application maintenance is carried out.
NFR05	The Online Learning Features application must be able to run on Android 5.0 and iOS 10 and above.
NFR06	The Online Learning Features server operating system uses Windows Server 2019.
NFR07	The Online Learning Features server CPU has a minimum of 16 cores.
NFR08	Database server using MySQL version 8.
NFR09	The Online Learning Features server has at least 64 GB of RAM and 1 TB of Storage.

Table 8. Identification of use case work sequence

ID.	Functional Requirement Specification	Use Case Name Identification
FR01	Students and teachers must log in to the Online Learning Features application.	Login
FR01A	Teachers must get the application menu according to their access rights.	Teacher Login
FR01B	Students must get the application menu according to their access rights.	Student Login
FR02	Students and teachers can use the application tutorial as a feature introduction and the Online Learning Features application.	Use Tutorial

FR03	Students must fill in attendance in the Online Learning Features application.	Attendance
FR04	Students and teachers can access key features such as live discussions and Q&A between teachers and students.	Access Main Features
FR05	Students can access entertainment features such as fun facts, world facts, and memes about education.	Access Entertainment Features
FR06	Teachers can upload materials and quizzes.	Upload Material & Quiz
FR07	Students can take quizzes.	Execute Quiz
FR08	Students can upload assignments.	Upload Assignments
FR09	Students can watch the material in the application without leaving the Online Learning Features application.	Watch Video Material

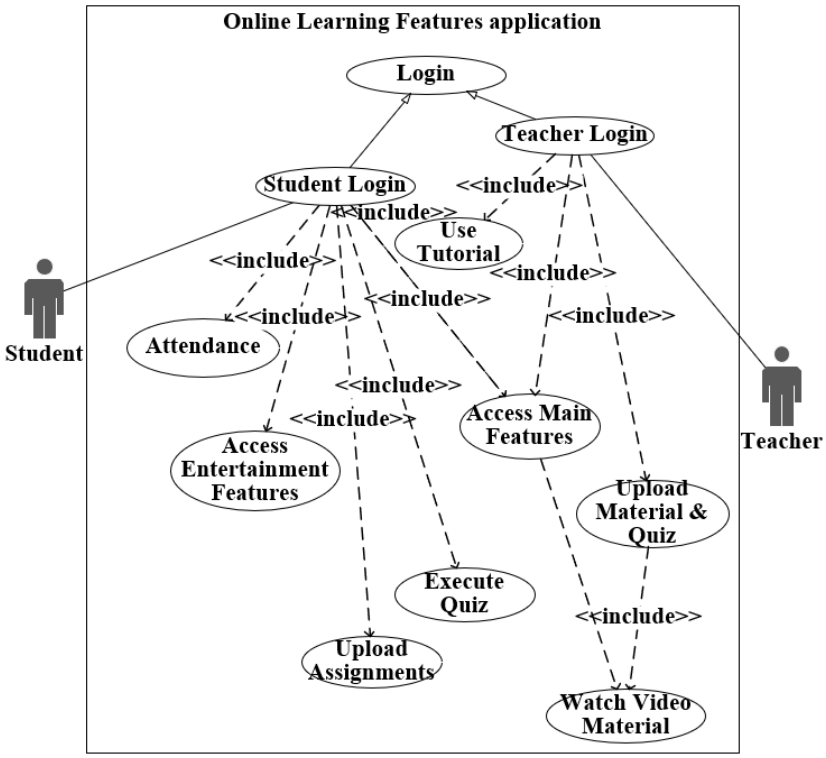
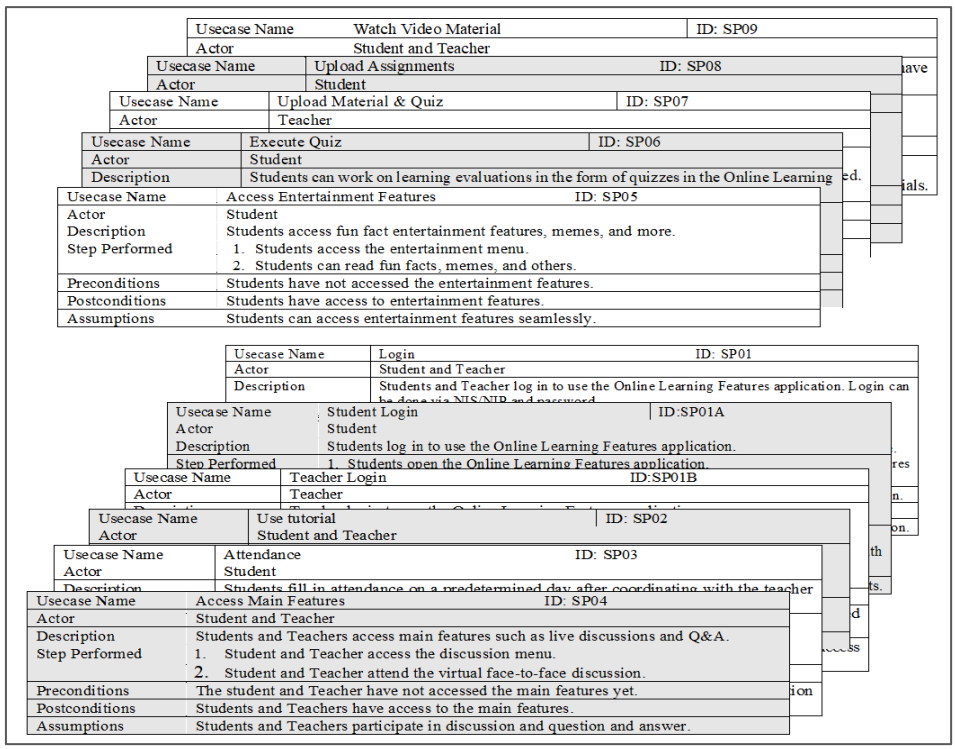


Figure 2. Use case diagram



Usecase Name	Watch Video Material	ID: SP09
Actor	Student and Teacher	
Usecase Name	Upload Assignments	ID: SP08
Actor	Student	
Usecase Name	Upload Material & Quiz	ID: SP07
Actor	Teacher	
Usecase Name	Execute Quiz	ID: SP06
Actor	Student	
Description	Students can work on learning evaluations in the form of quizzes in the Online Learning	

Usecase Name	Access Entertainment Features ID: SP05
Actor	Student
Description	Students access fun fact entertainment features, memes, and more.
Step Performed	1. Students access the entertainment menu. 2. Students can read fun facts, memes, and others.
Preconditions	Students have not accessed the entertainment features.
Postconditions	Students have access to entertainment features.
Assumptions	Students can access entertainment features seamlessly.

Usecase Name	Login ID: SP01
Actor	Student and Teacher
Description	Students and Teacher log in to use the Online Learning Features application. Login can
Usecase Name	Student Login ID:SP01A
Actor	Student
Description	Students log in to use the Online Learning Features application.
Step Performed	1. Students open the Online Learning Features application.
Usecase Name	Teacher Login ID:SP01B
Actor	Teacher
Usecase Name	Use tutorial ID: SP02
Actor	Student and Teacher
Usecase Name	Attendance ID: SP03
Actor	Student

Usecase Name	Access Main Features ID: SP04
Actor	Student and Teacher
Description	Students and Teachers access main features such as live discussions and Q&A.
Step Performed	1. Student and Teacher access the discussion menu. 2. Student and Teacher attend the virtual face-to-face discussion.
Preconditions	The student and Teacher have not accessed the main features yet.
Postconditions	Students and Teachers have access to the main features.
Assumptions	Students and Teachers participate in discussion and question and answer.

Figure 3. Step performed in use case description/use case scenario

After successfully designing the Use Case Diagram, the next step is to create a Use Case Description for each Use Case. Through this description, it will be possible to observe every step taken for each Use Case. There are eleven Use Case Descriptions created at this stage. Figure 3 is a Use Case Description for each Use Case. In the description, there is a "Step Performed" section, which explains the steps taken by a Use Case. This section is a description that will be processed through Text Preprocessing activities.

5.3 Datasets processing

As a dataset processing in this study, there are two groups of datasets derived from the Requirement Specification and Use Case Diagram, namely:

•Functional Requirements (FR)

•Use Case Description/Use Case Scenario

Based on the Functional Specification in this study, it can be seen that the Functional Requirements (FR) to be processed consist of eleven FRs. Therefore, in preparation for the use of the Dataset in this activity, it is divided into eleven documents, namely:

d1 = "Students and teachers must log in to the Online Learning Features application."

d2 = "Teachers must get the application menu according to their access rights."

d3 = "Students must get the application menu according to their access rights."

d4 = "Students and teachers can use the application tutorial as a feature introduction and the Online Learning Features application."

d5 = "Students must fill in attendance in the Online Learning Features application."

d6 = " Students and teachers can access key features such as live discussions and Q&A between teachers and students."

d7 = "Students can access entertainment features such as fun facts, world facts, and memes about education."

d8 = "Teachers can upload materials and quizzes."

d9 = " Students can take quizzes."

d10 = "Students can upload assignments."

d11 ="Students can watch the material in the application without leaving the Online Learning Features application."

Based on the Use Case Diagram in this study, it can be seen that the Use Case to be processed consists of eleven Use Case Descriptions. Therefore, in preparation for the use of the dataset in this activity, it is divided into eleven documents, namely:

d12 = "Students and Teachers open the Online Learning Features application. Students and Teachers log in by entering their NIS/NIP and password. If the NIS/NIP and password are correct, enter the Online Learning Features page. If the NIS/NIP and password are incorrect, return to the Online Learning Features login page."

d13= "Students open the Online Learning Features application. Students log in by entering their NIS and password. Students select menus and features contained in the application. Students interact with the teacher according to the selected features."

d14 = "The teacher opens the Online Learning Features application. The teacher logs in by entering the NIP and password. The teacher selects the menus and features contained in the application. The teacher interacts with the teacher according to the selected feature."

d15 = "Student and Teacher access the tutorial menu. Student and Teacher follow the step-by-step tutorial given."

d16= "Students open the attendance menu. Students fill in attendance in certain subjects by signing the attendance list."

d17 = "Student and Teacher access the discussion menu. Student and Teacher attend the virtual face-to-face discussion."

d18 = "Students access the entertainment menu. Students can read fun facts, memes, and others."

d19 = "Students access the quiz menu. Students choose subjects on the quiz menu provided. Students answer the quiz shown."

d20 = "The teacher accesses the learning material upload menu. The teacher uploads the learning material files and quizzes in the column provided. The teacher presses the button to upload learning materials to students."

d21 = "Students access the task menu. Students press the upload task button. Students choose the subjects whose results they want to upload. Students upload the assignment file."

d22 ="Student and Teacher access the menu of learning materials. Student and Teacher press the watch video material button."

5.4 Text processing

Referring to 22 documents that have been defined as datasets, then in this activity, text processing is carried out using the Natural Language Toolkit (NLTK). NLTK is a tool used in the field of Natural Language Processing through the use of the Python-3 programming language. Text Preprocessing activities implemented in this research include Case Folding, Tokenization, StopWords Removal, and Stemming/ Lemmatization.

At the beginning of the Text Preprocessing activity, in Figure 4, a snippet of the sample dataset process is processed in Text Preprocessing using Case Folding. All forms of writing in this dataset have changed into Lower Case form. This dataset consists of 22 documents. Case Folding is applied to documents contained in Functional Requirements and Step Performed documents.

```
#LowerCase FR
file = open('FR_beforecase.txt', 'r') #BeforeCase

lines = [line.lower() for line in file]
with open('FR_aftercase.txt', 'w') as out: #AfterCase
    out.writelines(sorted(lines))
print(lines)
```

```
['students and teachers must log in to the online learning f
ding to their access rights. students must get the applicati
can use the application tutorial as a feature introduction a
in attendance in the online learning features application. s
sions and q&a between teachers and students. students can ac
memes about education. teachers can upload materials and qui
s. students can watch the material in the application withou
```

```
#LowerCase Use Case Scenario (Description) STEP PERFORMED
file = open('STEP_beforecase.txt', 'r') #BeforeCase

lines = [line.lower() for line in file]
with open('STEP_aftercase.txt', 'w') as out: #AfterCase
    out.writelines(sorted(lines))
print(lines)
```

```
['students and teachers open the online learning features ap
p and password. if the nis/nip and password are correct, ent
d are incorrect, return to the online learning features logi
```

Figure 4. Case folding

After all, documents are prepared, the tokenization process is carried out to break all sentences into words. This process is combined with the stop words removal process to eliminate meaningless words in the Text Mining process. After that, Stemming/Lemmatization is done to turn the results into essential words. For the process of using Lemmatization, there is the use of NLTK "wordnet" as an English semantic dictionary. Normalized text from Lemmatization results on 22 previously tokenized documents, then indexed. Snippets of the process can be seen in the snippet of Figure 5 regarding tokenization and indexing of all word results.

```
#Tokenization, Stop Words, Steaming FR
LemDocuments = CountVectorizer(tokenizer=word_tokenize, stop_words='english')
LemDocuments.fit_transform(FR_dataset)
print (LemDocuments.vocabulary_)
```

```
{'students': 29, 'teachers': 30, 'log': 20, 'online': 25, 'learning': 17, 'fea
24, 'according': 4, 'access': 3, 'rights': 28, 'use': 33, 'tutorial': 31, 'fea
7, 'key': 16, 'live': 19, 'discussions': 8, 'q': 26, '&': 0, 'entertainment':
5, 'memes': 23, 'education': 9, 'upload': 32, 'materials': 22, 'quizzes': 27,
'leaving': 18}
```

```
#Tokenization, Stop Words, Steaming STEP PERFORMED
LemDocuments = CountVectorizer(tokenizer=word_tokenize, stop_words='english')
LemDocuments.fit_transform(STEP_dataset)
print (LemDocuments.vocabulary_)
```

```
{'students': 65, 'teachers': 69, 'open': 46, 'online': 45, 'learning': 32, 'fe
34, 'entering': 18, 'nis/nip': 44, 'password': 49, 'correct': 15, ',': 0, 'ent
n': 57, 'login': 35, 'nis': 43, 'select': 58, 'menus': 41, 'contained': 14, 'i
```

Figure 5. Combination of tokenization with stop words

The next step is to convert all the words in the document into their basic word form by stemming and cutting off the ends of a word, regardless of the context of the word's meaning. An example of the process can be seen in the snippet of Figure 6.

```
#Steaming FR
def stemSentence(sentence):
    token_words = word_tokenize(sentence)
    token_words
    stem_sentence=[]
    for word in token_words:
        stem_sentence.append(porter.stem(word))
        stem_sentence.append(" ")
    return "".join(stem_sentence)

for i in range (len(FR_dataset)):
    FR_dataset[i] = stemSentence(FR_dataset[i])
    print (FR_dataset[i])
```

```
student and teacher must log in to the onlin learn featur app
teacher must get the applic menu accord to their access right
student must get the applic menu accord to their access right
student and teacher can use the applic tutori as a featur int
student must fill in attend in the onlin learn featur applic
student and teacher can access key featur such as live discus
student can access entertain featur such as fun fact , world
teacher can upload materi and quizz .
student can take quizz .
student can upload assign .
student can watch the materi in the applic without leav the o
```

```
#Steaming STEP PERFORMED
def stemSentence(sentence):
    token_words = word_tokenize(sentence)
    token_words
    stem_sentence=[]
    for word in token_words:
        stem_sentence.append(porter.stem(word))
        stem_sentence.append(" ")
    return "".join(stem_sentence)
```

Figure 6. Stemming

5.5 Document value weight processing

Based on the results of indexing stemming from the tokenization process, the following process is to transform into a TF-IDF matrix on all documents. For snippets of the results of TF-IDF on Functional Requirements and TF-IDF on Step Performed, see Figure 7.

```
#TF-IDF Results FR
tfidf_matrixFR = tfidfTranFR.transform(tf_matrixFR)
print (tfidf_matrixFR.toarray())
```

```
[[0.         0.2767276  0.         0.         0.
  0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.7725
  0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.3271
```

```
#TF-IDF Results STEP PERFORMED
tfidf_matrixSTEP = tfidfTranSTEP.transform(tf_matrixSTEP)
print (tfidf_matrixSTEP.toarray())
```

```
[[0.3024033  0.25345012 0.         0.         0.         0
  0.         0.         0.         0.         0.         0
  0.         0.         0.         0.17689294 0.         0
  0.         0.         0.         0.         0.         0
  0.         0.         0.         0.         0.         0
  0.         0.         0.         0.         0.13297338 0
  0.         0.         0.         0.         0.         0
  0.         0.         0.53067883 0.         0.11883444 0
  0.35378589 0.39892015 0.         0.         0.         0
  0.         0.         0.         0.17689294 0.         0
  0.         0.         0.         0.         0.13775158 0
```

Figure 7. TF-IDF matrix results snippets

5.6 Similarity and validity values for SRS compatibility

In calculating the similarity of an object, it can be used with the Cosine Similarity formula. Through this process, it is possible to search for similarities between documents. Referring to the results of processing the document value weights in the previous TF-IDF process, then in this similarity process, an experiment has been carried out to calculate the similarity of the Step Performed matrix with the Functional Requirement matrix, as shown in Figure 8.

In calculating the similarity between documents, this study uses the cosine similarity metric. Cosine similarity is chosen as a metric for several important reasons that support the accuracy and effectiveness of text analysis in the context of SRS, namely: measuring Orientation, not Magnitude, effectiveness in text analysis, scalability, simplicity for easy understanding, and extensive use in research. By choosing cosine similarity as a metric, this study can ensure that the similarity analysis between SRS documents is carried out accurately and effectively, supporting the main objective of the study to measure the compatibility of software requirements.

For the next step, validation is carried out by measuring the reliability between the two matrices. The reliability result is 0.299. This value is based on a range from 0 to 1. If the value is closer to 1, then the SRS compatibility through the Use Case Diagram artifact will get better. Referring to the Interpretation

of Gwet's AC1 Value, the reliability value is included in the Kappa Index = "Fair agreement."

```
#Cosin Similarity
cos_similarity_matrix = (tfidf_matrixSTEP * tfidf_matrixFR.T).toarray()
print (pd.DataFrame(cos_similarity_matrix))
```

```
          0         1         2         3         4         5         6
0   0.301387  0.153470  0.122184  0.312873  0.130406  0.254534  0.314166
1   0.527053  0.271930  0.377427  0.550245  0.313782  0.449235  0.149522
2   0.550630  0.467644  0.176986  0.633683  0.114588  0.547284  0.054603
3   0.392960  0.504587  0.437778  0.618887  0.217899  0.632259  0.163715
4   0.141172  0.141791  0.234697  0.222337  0.858578  0.171595  0.076146
5   0.447585  0.574729  0.498633  0.704918  0.609957  0.720148  0.186473
6   0.178420  0.282024  0.405755  0.281000  0.201960  0.296924  0.876965
7   0.179207  0.282719  0.406962  0.282240  0.202851  0.271431  0.133826
8   0.309016  0.415398  0.272963  0.486681  0.131205  0.473149  0.100979
9   0.211688  0.213587  0.352960  0.333395  0.239617  0.304797  0.147107
10  0.388422  0.498761  0.432722  0.611741  0.215383  0.624958  0.161825

          7         8         9         10
0   0.179493  0.268850  0.146349  0.130406
1   0.226106  0.646904  0.352143  0.313782
2   0.470220  0.236239  0.128597  0.114588
3   0.332717  0.449228  0.244537  0.217899
4   0.072197  0.329445  0.179333  0.159798
5   0.378968  0.511675  0.278530  0.248189
6   0.091246  0.416368  0.226650  0.201960
7   0.091648  0.418204  0.227649  0.202851
8   0.860541  0.270496  0.734244  0.131205
9   0.570612  0.494003  0.771477  0.239617
10  0.328875  0.444041  0.241714  0.582677
```

```
from sklearn.metrics import cohen_kappa_score
cohen_kappa_score(tf_matrixSTEP[10], tf_matrixFR[10])
```

```
0.2992036405005687
```

Figure 8. Python Kappa score Gwet's AC1

Table 9. Questionnaire Kappa score Gwet's AC1

Expert 1	Expert 2		
	Yes	No	Total
Yes	26	9	35
No	9	6	15
Total	35	15	50

As a comparison of the validity and reliability results, a questionnaire was distributed regarding the compatibility of SRS to two groups of experts regarding UML artifacts. The results of filling out the expert questionnaires can be seen in Table 9.

Based on Table 9, the following steps are as follows:

Calculate the observed agreement (P) between the two experts.

$$P = \frac{26 + 6}{50} = 0.64 \tag{9}$$

Calculating Chance-Agreement or $e(Y)$.

$$P_1 = \frac{(35 + 35)/2}{50} = 0.7 \tag{10}$$

$$e(Y) = 2(0.7)(1 - 0.7) = 0.42 \tag{11}$$

Calculates AC1 statistics using Aggregation Coefficients.

$$AC1 = \frac{0.64 - 0.42}{1 - 0.42} = 0.379 \tag{12}$$

Gwet's AC1 formula results through a questionnaire completed by the expert, namely: 0.379. This value is in the position of the "Fair Agreement" category.

6. CONCLUSION AND FUTURE WORK

This research has succeeded in extracting data from SRS by doing Text Preprocessing and similarity to identify the compatibility of FR with Step Performed. The following are four things that can be concluded as the core of this research activity, namely:

(1) Based on the SRS named "Akuonline Learning Application," it has been identified that the Requirement Specification in the form of 11 FRs and 9 NFRs has been identified. The Requirement Specifications match FR (11) and the Use Case Diagram. In this diagram, there are 11 Use Case Descriptions whose number is the same as the number of Use Cases. Step Performed in the Use Case Description is used as a source for compatibility

(2) Through Python NLTK and based on the description text of an FR and Step Performed, this activity resulted in Case Folding, Tokenization, Stopwords Removal, and Stemming, which were applied to 22 documents (d1 to d22).

(3) Based on the results of the Cosine Similarity calculation by comparing the FR matrix and the Step performed matrix, the validation results through the Python version of Gwet's AC1 reliability measurement = 0.299, while for the questionnaire version = 0.379. There is a difference of 0.08.

(4) Referring to the Interpretation of Gwet's AC1 Value, the validity and reliability values are included in the Kappa Index = "Fair agreement" category.

Future work activities for further development are conducting semantic textual similarity activities applied to the SRS documentation case study. Gwet's AC1 Validity and Reliability will be used for validation and will be conducted on experts (people who understand UML) through questionnaire activities. To improve the effectiveness of the compatibility design method, semantic information can be incorporated into the similarity calculation with the following approaches: use of machine learning models, word embedding, use of sentence transformers, semantic similarity calculation, and integration with text preprocessing. By incorporating semantic information into the similarity calculation, future research can improve the effectiveness of the compatibility design method, allowing for deeper and more accurate identification of similarities between software requirements.

ACKNOWLEDGMENT

This work was supported by Department of Software Engineering (RPL), and the Directorate of Research and Community Service (PPM Tel-U) at Telkom University Bandung 40257.

REFERENCES

[1] Priyadi, Y., Putra, A.M., Lyanda, P.S. (2021). The similarity of elicitation software requirements specification in student learning applications of SMKN7 Baleendah based on use case diagrams using text mining. In 2021 IEEE 5th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE), Purwokerto, Indonesia, 2021, pp. 115-120. https://doi.org/10.1109/ICITISEE53823.2021.9655844

[2] Vaish, N., Sharma, A. (2018). Semi-automated system

based defect detection in software requirements specification document. In 2018 5th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON), Gorakhpur, India, pp. 1-5. https://doi.org/10.1109/UPCON.2018.8597101

[3] Khalid, S., Ayaz, S., Khalil, T., Akram, M.U., Sahar, S. (2017). Interview based iterative requirement elicitation for ARMD detection in OCT images. In 2017 Computing Conference, London, UK, pp. 610-614. https://doi.org/10.1109/SAI.2017.8252159

[4] Alfianto, M.A., Priyadi, Y., Laksitowening, K.A. (2023). Semantic textual similarity in requirement specification and use case description based on sentence transformer model. In 2023 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT), BALI, Indonesia, pp. 220-226. https://doi.org/10.1109/IAICT59002.2023.10205769

[5] Osman, M.H., Zaharin, M.F. (2018). Ambiguous software requirement specification detection: An automated approach. In Proceedings of the 5th International Workshop on Requirements Engineering and Testing, Gothenburg, Sweden, pp. 33-40. https://doi.org/10.1145/3195538.3195545

[6] Rizqi, A.M., Priyadi, Y. (2023). Text validity application forming functional and non-functional requirements based on documentation analysis on TESA applications. In 2023 International Conference on Electrical and Information Technology (IEIT), Malang, Indonesia, pp. 342-347. https://doi.org/10.1109/IEIT59852.2023.10335493

[7] Wu, C.S., Kuo, C.J., Su, C.H., Wang, S.H., Dai, H.J. (2020). Using text mining to extract depressive symptoms and to validate the diagnosis of major depressive disorder from electronic health records. Journal of affective disorders, 260: 617-623. https://doi.org/10.1016/j.jad.2019.09.044

[8] Salloum, S.A., Al-Emran, M., Monem, A.A., Shaalan, K. (2018). Using text mining techniques for extracting information from research articles. In Intelligent natural language processing: Trends and Applications, pp. 373-397. https://doi.org/10.1007/978-3-319-67056-0_18

[9] Dar, H., Lali, M.I., Ashraf, H., Ramzan, M., Amjad, T., Shahzad, B. (2018). A systematic study on software requirements elicitation techniques and its challenges in mobile application development. IEEE Access, 6, 63859-63867. https://doi.org/10.1109/ACCESS.2018.2874981

[10] Cherdsakulwong, N., Suwannasart, T. (2019). Impact analysis of test cases for changing inputs or outputs of functional requirements. In 2019 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Toyama, Japan, pp. 179-183. https://doi.org/10.1109/SNPD.2019.8935754

[11] Broy, M. (2018). Rethinking functional requirements: A novel approach categorizing system and software requirements. In Software Technology: 10 Years of Innovation in IEEE Computer: 10 Years of Innovation, pp. 155-187. https://doi.org/10.1002/9781119174240.ch9

[12] Vachharajani, V., Pareek, J. (2012). Use case extractor: XML parser for automated extraction and storage of use-Case diagram. In 2012 IEEE International Conference on Engineering Education: Innovative Practices and Future Trends (AICERA), Kottayam, India, pp. 1-5. https://doi.org/10.1109/AICERA.2012.6306714

[13] Seba, M.D., Priyadi, Y., Darwiyanto, E. (2024). Software development for text processing in mapping architecturally significant requirements towards quality attributes. In 2024 10th International Conference on Smart Computing and Communication (ICSCC), Bali, Indonesia, pp. 394-400. https://doi.org/10.1109/ICSCC62041.2024.10690818

[14] Resinas, M., del-Río-Ortega, A., van der Aa, H. (2023). From text to performance measurement: automatically computing process performance using textual descriptions and event logs. In 21st International Conference, BPM 2023, Utrecht, The Netherlands, pp. 266-283. https://doi.org/10.1007/978-3-031-41620-0_16

[15] Pamungkas, J.A., Priyadi, Y., Alibasa, M.J. (2022). Measurement of similarity between requirement elicitation and requirement specification using text pre-processing in the cinemaloka application. In 2022 IEEE World AI IoT Congress (AIIoT), Seattle, WA, USA, pp. 672-678. https://doi.org/10.1109/AIIoT54504.2022.9817193

[16] Haddi, E., Liu, X., Shi, Y. (2013). The role of text pre-processing in sentiment analysis. Procedia Computer Science, 17: 26-32. https://doi.org/10.1016/j.procs.2013.05.005

[17] Zahra, Z., Priyadi, Y. (2023). Text data processing on non-functional requirement for the similarity between requirement elicitation with deployment diagram and recommendation for SRS improvement. In 2023 IEEE World AI IoT Congress (AIIoT), Seattle, WA, USA, pp. 0830-0836. https://doi.org/10.1109/AIIoT58121.2023.10174437

[18] Alodadi, M., Janeja, V.P. (2015). Similarity in patient support forums using TF-IDF and cosine similarity metrics. In 2015 International Conference on Healthcare Informatics, Dallas, TX, USA, pp. 521-522. https://doi.org/10.1109/ICHI.2015.99

[19] Priyadi, Y., Kusumahadi, K., Lyanda, P.S. (2022). IdVar4CL: Causal loop variable identification method for systems thinking based on text mining approach. International Journal of Fuzzy Logic and Intelligent Systems, 22(4): 373-381. https://doi.org/10.5391/IJFIS.2022.22.4.373

[20] Wei, G. (2017). Some cosine similarity measures for picture fuzzy sets and their applications to strategic decision making. Informatica, 28(3): 547-564. https://doi.org/10.3233/INF-2017-1150

[21] Gokul, P.P., Akhil, B.K., Shiva, K.K. (2017). Sentence similarity detection in Malayalam language using cosine similarity. In 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, India, pp. 221-225. https://doi.org/10.1109/RTEICT.2017.8256590

[22] Wongpakaran, N., Wongpakaran, T., Wedding, D., Gwet, K.L. (2013). A comparison of Cohen's Kappa and Gwet's AC1 when calculating inter-rater reliability coefficients: A study conducted with personality disorder samples. BMC Medical Research Methodology, 13: 61. https://doi.org/10.1186/1471-2288-13-61

[23] Gwet, K.L. (2008). Computing inter-rater reliability and

its variance in the presence of high agreement. British Journal of Mathematical and Statistical Psychology, 61(1): 29-48. https://doi.org/10.1348/000711006X126600

[24] Gwet, K.L. (2014). Handbook of Inter-Rater Reliability: The Definitive Guide to Measuring the Extent of Agreement Among Raters. Advanced Analytics, LLC.