NEAR DUPLICATE DOCUMENT DETECTION FOR MALAYALAM

Presented By,
Balasankar C
Krishnanunni R
Sameena T A

PROBLEM STATEMENT

The world is becoming a single global e-village and everything is going to the web. This, in turn, make sure that all the new developments and creations happen in the web. This induces a need for a system to check data theft and fraud. For both these scenarios – **reducing search engines' burden and checking for data theft** – similarity checking systems are required. Since, duplicate detection systems cannot work on Indic languages due to their grammatical and linguistic features like inflection and agglutination, near duplicate detection is only possible, which implements approximate search.

SCOPE

A similar duplicate detection system already exists for English and other Latin based languages. But, no such system exists for Indic languages, specifically for Dravidian Languages. This projects focuses mainly on Malayalam, the native language of the state Kerala and Lakshadweep. The project compares two plain text documents written (typed) in Malayalam and analyze them for similarity percentages. Though, the project is for Malayalam, similar workflow can be used to design such system for any Indic languages as all the Indic languages have somewhat similar grammatical structure.

EXISTING SYSTEM & LIMITATIONS

- There are currently no existing systems for Malayalam Language. However, there exists some similar works which are intended for other Indic languages like Hindi, Marathi and Gujarathi.

- In Malayalam the only advancement is done by an organization named Swathanthra Malayalam Computing, through their Indic language framework called SILPA (Swathanthra Indic Language Processing Applications).

- The is that, it does not incorporate stemmer in document comparison. That **disadvantage of SILPA project** is, inflections are not handled in the SILPA project. It performs only character to character comparison which is not at all useful for Indic languages.

PROPOSED SYSTEM & ARCHITECTURE

- The proposed document comparison system is a modification to the SILPA project done by Swathanthra Malayalam Computing; This project, in short, enhances the Prathyaya rules, incorporates stemmer with document comparison module and perform comparison.

- The step involved in document comparison are:

    1. Stemming
    2. Tokenizing document to n-grams
    3. Comparing
    4. Calculating Similarity
    5. Displaying result

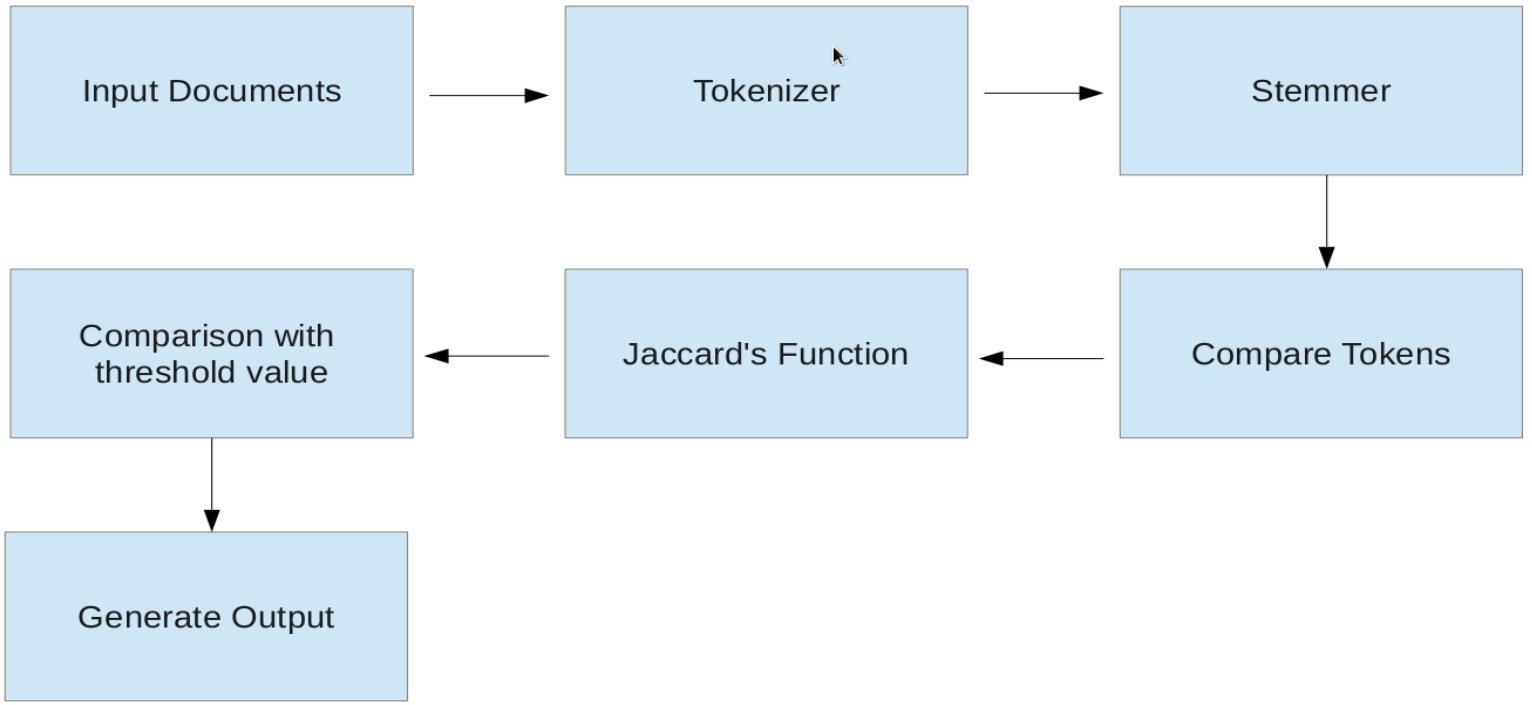


Fig : Block diagram of duplicate document detection

FUNCTIONAL REQUIREMENTS

1. TEXTUAL RULES

- Provide the rules for generation of root words according to the Malayalam grammar criteria.
- An external file is defined with the syntax <existing suffix> = <root word suffix>. A sample of the file is described below:

#അനുസ്വാരത്തിലവസാനിക്കുന്ന ക്രിയ/നാമം

ത്തിൽ = ◌ം

ത്ത് =◌ം

ത്തു് =◌ം

ത്തു =◌ം

ത്തെ = ◌ം

വുമായി = ◌ം

ത്തിനെ=◌ം

ത്തിലെ=◌ം

This should be implemented for all existing Prathyaya rules and inflections.

2. INFLECTION REMOVAL

Inflection ,the modulation of intonation or pitch in the voice, is the easiest way to create duplicate documents. The main intention of this project is to extract root word from the inflected word and use them for comparison of documents.

3. ORDER OF PARTS OF SPEECH

The order of parts of speech may be used for duplication. But order of parts of speech is not affected in our comparison because we are ordering the sentence as a sorted sentence.

4. THRESHOLD VALUE BY USER

The user may be considering a certain percentage as the maximum possible limit for flagging a document as error. So user may provide the maximum possible duplication for the documents.

5. PARAGRAPH WISE COMPARISON

The original document and the document to be compared are analyzed on a paragraph basis.

6. TOKENIZER

Tokenizer is used to obtain the shingles from a particular sentence of the documents that are comparing .

7. STEMMER

Stemmer ,as the name suggest, is used to stem an inflected word for obtaining the root word.

8. COMPARATOR

After obtaining stemmed shingles, the comparator compares the shingles of the two documents and generate the percentage similarity. If user has set a threshold value, the output also shows if the duplication amount exceeds the threshold value.

9. GUI

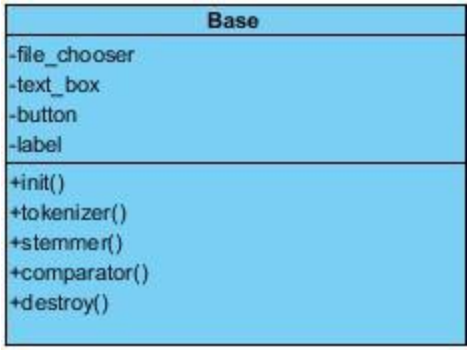
The user interface will be having two text-areas , two file-choosers , a text field to enter threshold, a button to start comparison.

NON FUNCTIONAL REQUIREMENTS

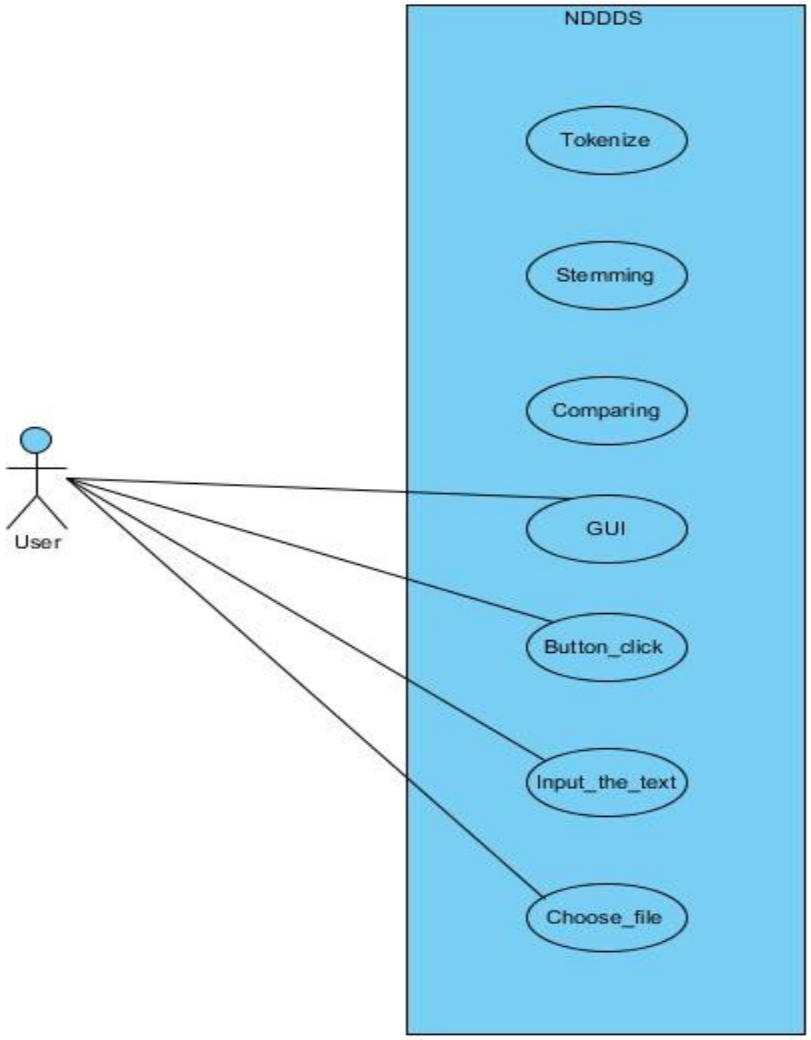
- **User should have execute permissions on the file.**
- **Python enabled OS**
- **1GB RAM**
- **2GHz Processor**
- **Python**
- **PyGTK**
- **High Tokenizer Efficiency**
- **High Stemmer Efficiency**
- **High Comparator Efficiency**
- **Fast Response**
- **Data Dictionary – High Precedence**
- **Stemmer Rules – High Precedence**
- **Order of POS – High Precedence**

DESIGN DIAGRAMS

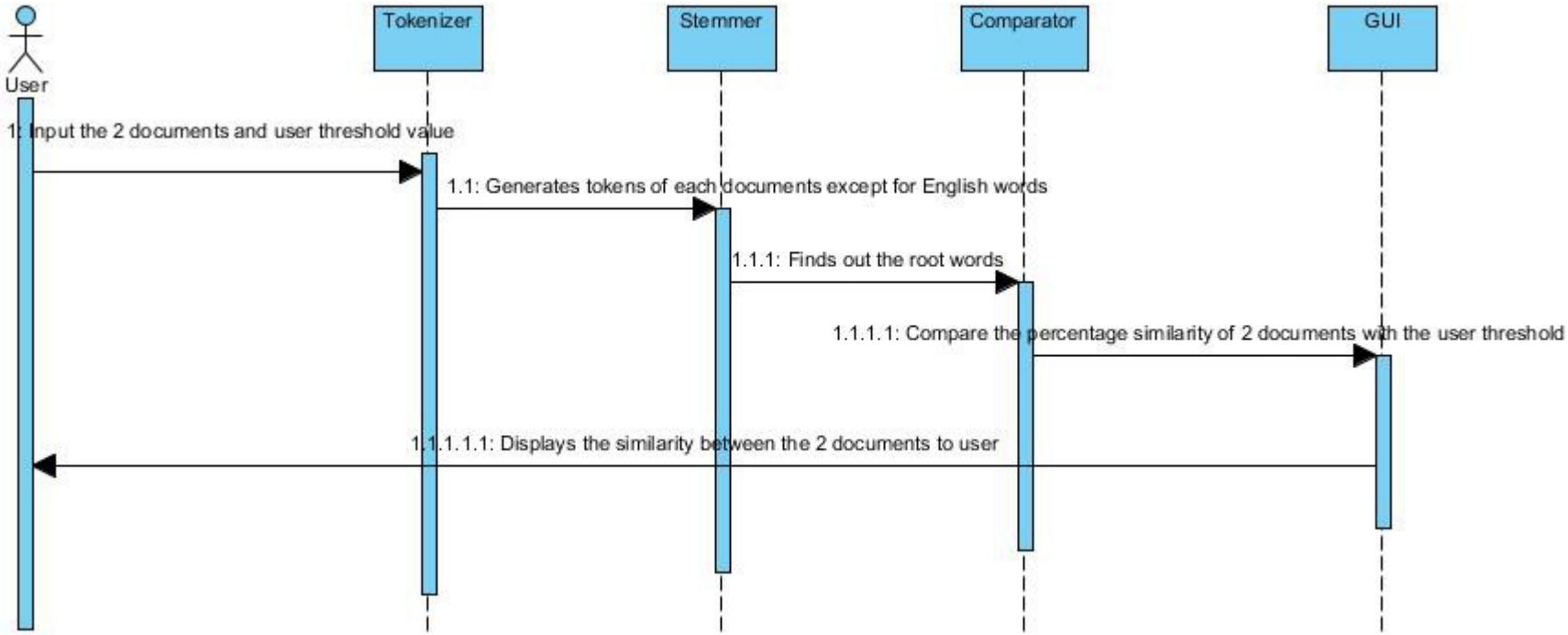
1. CLASS DIAGRAM



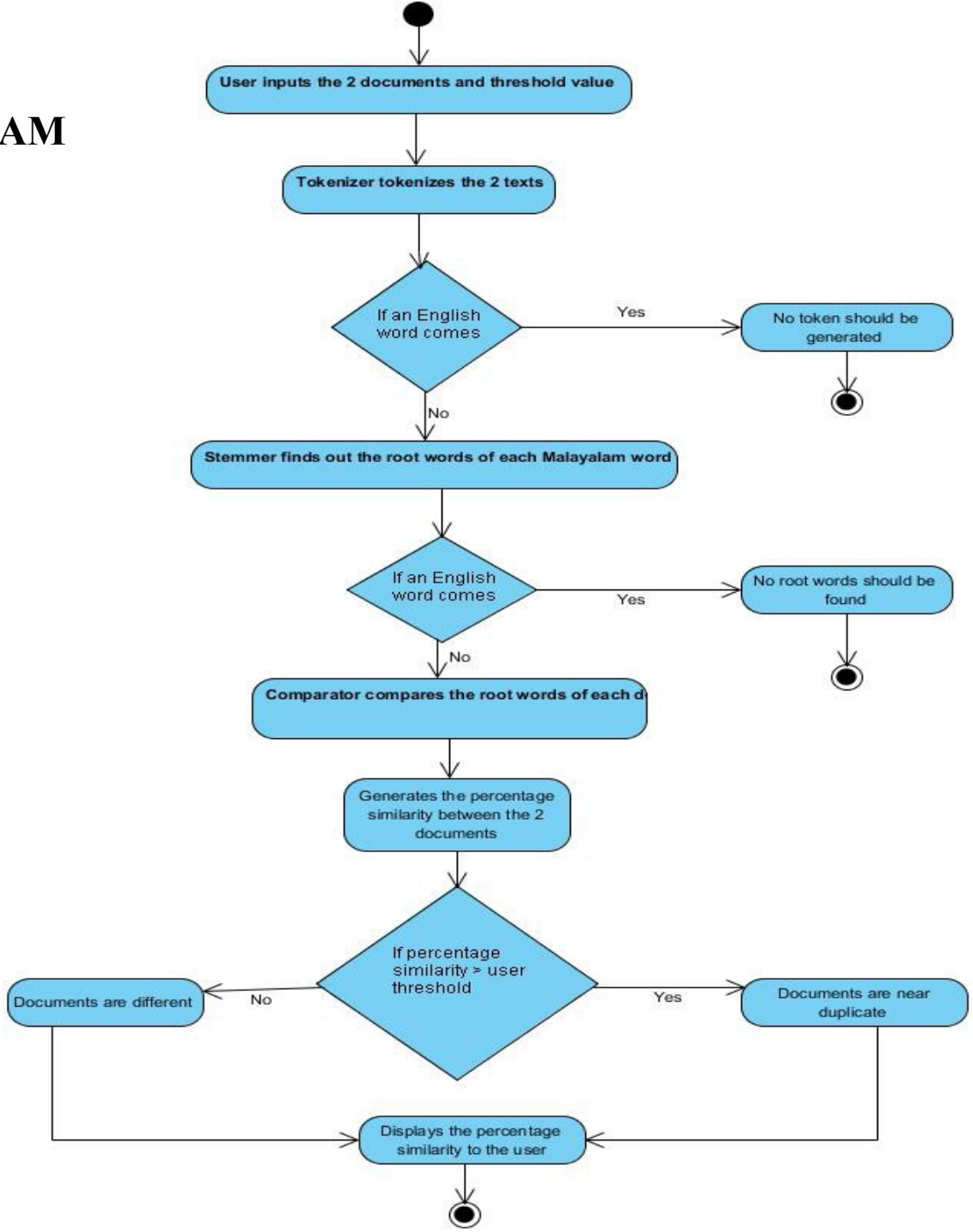
2. USE CASE DIAGRAM



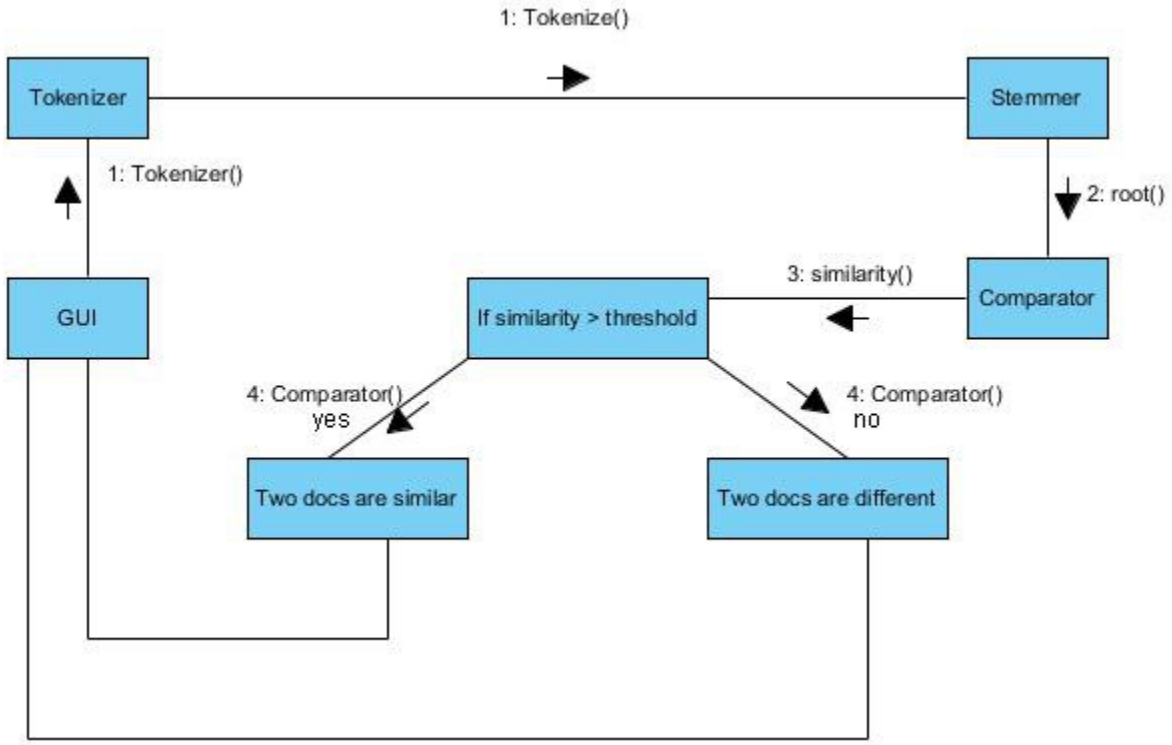
3. SEQUENCE DIAGRAM



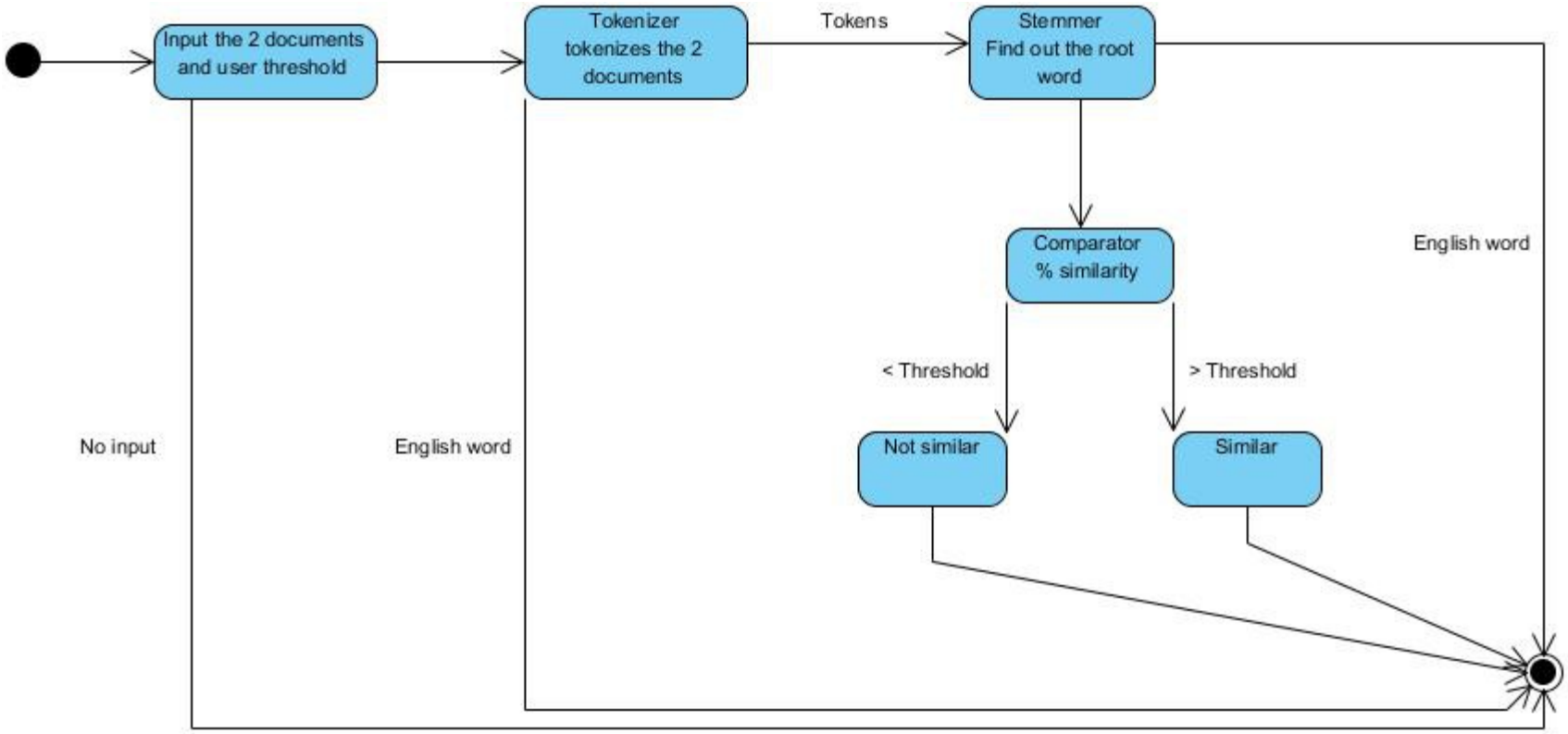
4. ACTIVITY DIAGRAM



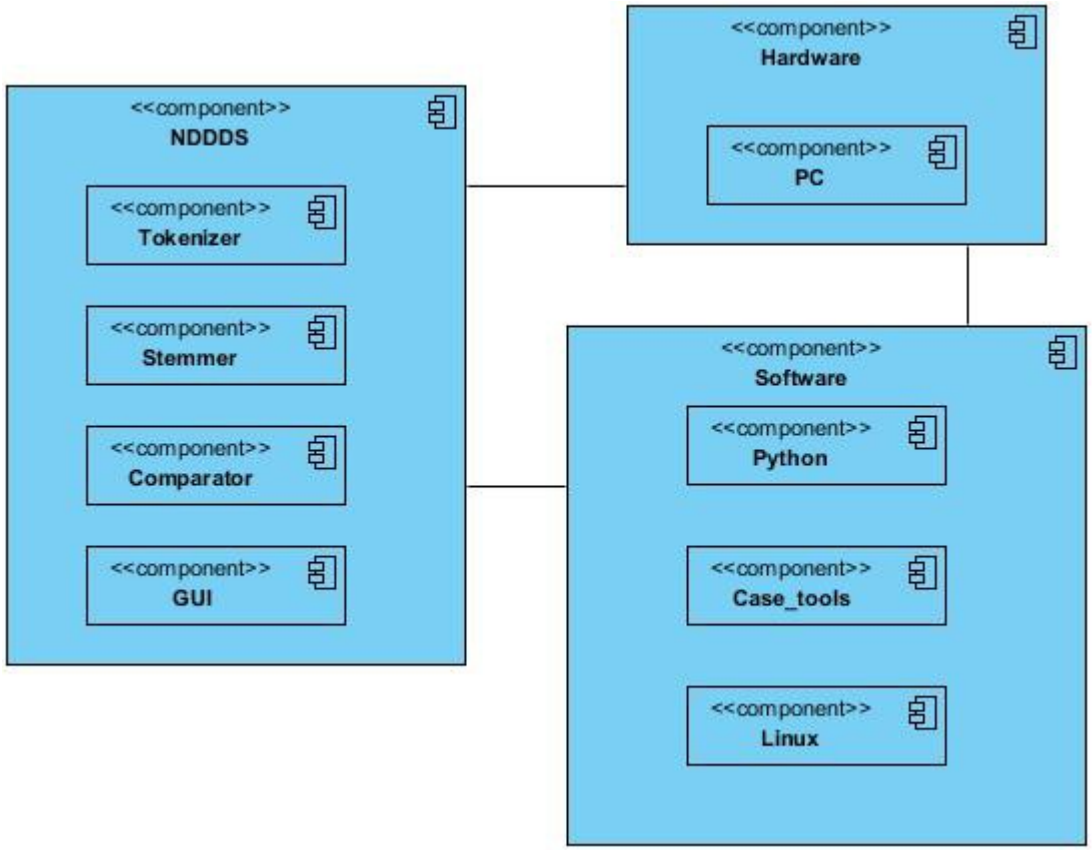
5. COMMUNICATION DIAGRAM



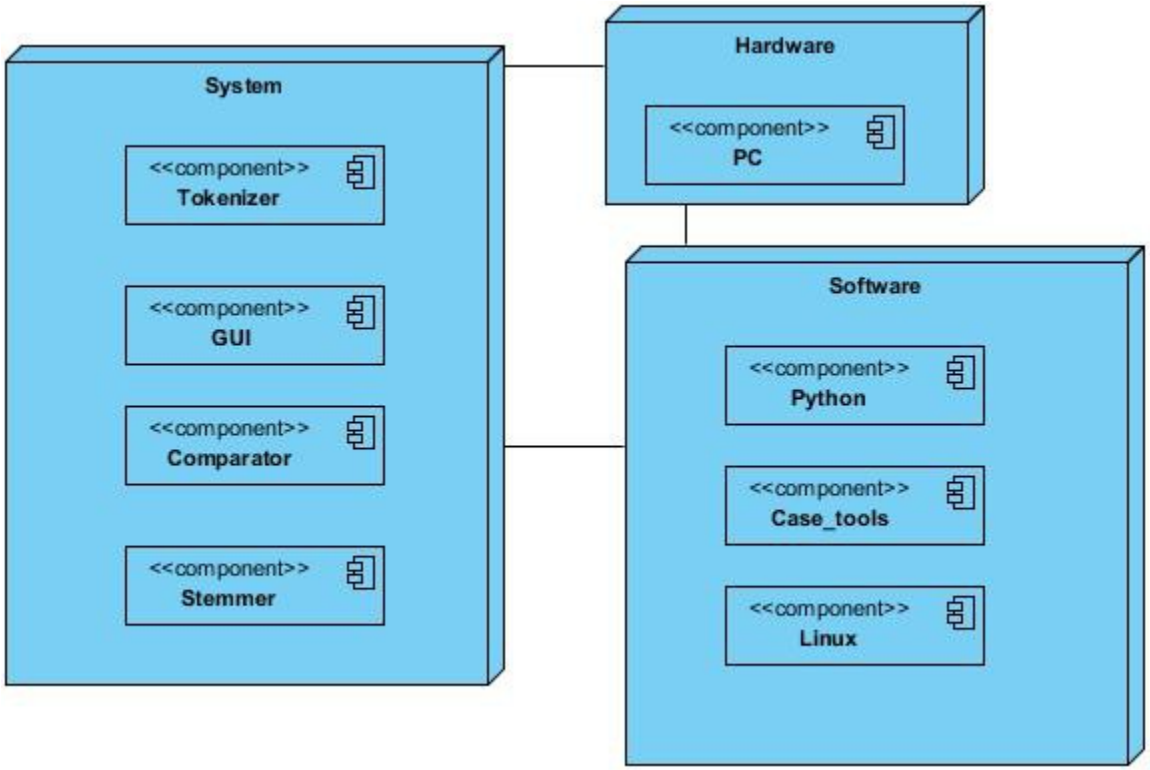
6. STATE MACHINE DIAGRAM



7. COMPONENT DIAGRAM



8. DEPLOYMENT DIAGRAM



TEST DATA

INPUT	EXPECTED OUTPUT
Exact duplicate documents. Eg: Document 1 : അവൻ തൃശ്ശൂരിൽ പോയി Document 2 : അവൻ തൃശ്ശൂരിൽ പോയി	Document 1 Before Stemming – ['അവൻ', തൃശ്ശൂരിൽ', 'പോയി'] Tokens – ['അവൻ', 'തൃശ്ശൂർ' ,'പോയി'] Document 2 Before Stemming – ['അവൻ', തൃശ്ശൂരിൽ', 'പോയി'] Tokens – ['അവൻ', 'തൃശ്ശൂർ' ,'പോയി'] Output Similarity : 100%

Near Duplicate Documents Eg : Document 1 : അവൻ തൃശ്ശൂർ പോയി Document 2 : അവൻ തൃശ്ശൂരിൽ പോയി	Document 1 Before Stemming – ['അവൻ', 'തൃശ്ശൂർ' ,'പോയി'] Tokens – ['അവൻ', 'തൃശ്ശൂർ' ,'പോയി'] Document 2 Before Stemming – ['അവൻ', തൃശ്ശൂരിൽ', 'പോയി'] Tokens – ['അവൻ', 'തൃശ്ശൂർ' ,'പോയി'] Output Similarity : 100%
Distinct Documents Eg: Document 1 : അവൻ കാലടിയിൽ നിന്നും വന്ന Document 2 : അവൻ തൃശ്ശൂരിൽ പോയി	Document 1 Before Stemming – ['അവൻ', 'കാലടി' ,'നിന്നും' 'വന്ന'] Tokens – ['അവൻ', 'തൃശ്ശൂർ' ,'പോയി'] Document 2 Before Stemming – ['അവൻ', തൃശ്ശൂരിൽ', 'പോയി'] Tokens – ['അവൻ', 'തൃശ്ശൂർ' ,'പോയി'] Output Similarity : 20%

DEMONSTRATION DETAILS

- The program takes two Malayalam texts as input and tokenize them, find the root words, and compare those root words.

- The user has to select two files containing the Malayalam texts using the File Chooser buttons in the program.

- The text boxes will display the contents of both the files.

- The user may also set the threshold value to be compared for data fraud.

- On clicking the button, the result, i.e whether the documents are near duplicated will be displayed.