A SPELLING CHECKER FOR KREOL MORISIEN

By

Tushaar Mungul (ID: 2013149)

A Final Year Project

Submitted to the Department of Information and communication Technologies

University of Mauritius

Faculty of Information, Communication and Digital Technologies

In Partial Fulfilment for the Degree of

BSc (Hons) Computer Science

July 2023

Supervised by Dr. Sameerchand Pudaruth

UNIVERSITY OF MAURITIUS

PROJECT/DISSERTATION DECLARATION FORM

Name:	**Mungul Tushaar**
Student ID:	**2013149**
Programme of Studies:	**BSc (Hons) Computer Science**
Module Code/Name:	**ICT 3000Y**
Title of Project/Dissertation:	**A Spelling Checker for Kreol Morisien**
Name of Supervisor(s):	**Dr. Smeerchand Pudaruth**

Declaration:

In accordance with the appropriate regulations, I hereby submit the above dissertation for examination and I declare that:

(i) I have read and understood the sections on **Plagiarism and Fabrication and Falsification of Results** found in the University's "General Information to Students" Handbook (20…./20….) and certify that the dissertation embodies the results of my own work.

(ii) I have no objection to submit a soft copy of my dissertation through the Turnitin Platform. I confirm that the hard copies and soft copy, submitted to the Faculty/Centre Registry**,** and the soft copy (main body, i.e, introduction up to the last Chapter) uploaded through Turnitin Platform are identical in content.

(iii) I have adhered to the 'Harvard system of referencing' or a system acceptable as per "The University of Mauritius Referencing Guide" for referencing, quotations and citations in my dissertation. Each contribution to, and quotation in my dissertation from the work of other people has been attributed, and has been cited and referenced.

(iv) I have not allowed and will not allow anyone to copy my work with the intention of passing it off as his or her own work.

(v) I am aware that I may have to forfeit the certificate/diploma/degree in the event that plagiarism has been detected after the award.

(vi) Notwithstanding the supervision provided to me by the University of Mauritius, I warrant that any alleged act(s) of plagiarism during my stay as registered student of the University of Mauritius is entirely my own responsibility and the University of Mauritius and/or its employees shall under no circumstances whatsoever be under any liability of any kind in respect of the aforesaid act(s) of plagiarism.

Signature: 	Date: **25/07/2023**

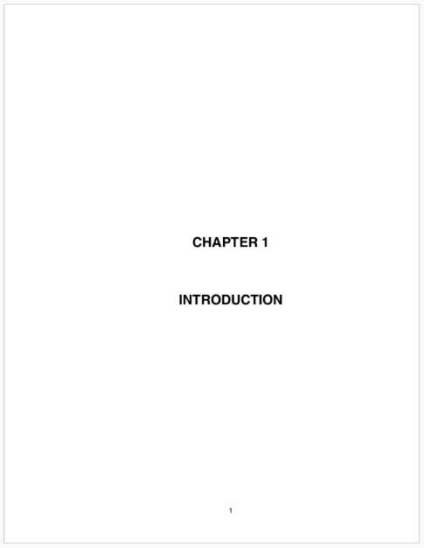
turnitin

Digital Receipt

This receipt acknowledges that Turnitin received your paper. Below you will find the receipt information regarding your submission.

The first page of your submissions is displayed below.

Submission author:	Tushaar Mungul
Assignment title:	BSc Final Year Project 2022/2023
Submission title:	A Spelling Checker for Kreol Morisien
File name:	CS_2013149_SP3_BD.pdf
File size:	8.36M
Page count:	146
Word count:	12,043
Character count:	64,730
Submission date:	26-Jul-2023 01:40PM (UTC+0400)
Submission ID:	2137044114



CHAPTER 1

INTRODUCTION

1

Dissertation Supervisor Statement

25 July 2023

To whom it may concern:

Dear Sir/Madam,

This is to confirm that I have supervised the project of **Mr Tushaar Mungul** during the academic year 2022-2023. His student ID is **2013149**. The title of his project is: ***Kreol Morisien Spell Checker***.

Thanks and regards.

Associate Professor (Dr) Sameerchand Pudaruth
ICT Department
Faculty of Information, Communication and Digital Technologies (FoICDT)
University of Mauritius
Tel: 403 7754
Email: s.pudaruth@uom.ac.mu

Contents

List of Tables

List of Figures

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. Sameerchand Pudaruth, for his invaluable guidance, support, and mentorship throughout the entire dissertation process. His expertise, constructive feedback, and unwavering encouragement have been instrumental in shaping the outcome of this research.

I am also immensely grateful to my family for their constant support, understanding, and patience. Their belief in me and their unwavering encouragement have been the driving force behind my perseverance. I am particularly indebted to my mother, whose unwavering love, sacrifice, and belief in my abilities have been a constant source of inspiration.

Furthermore, I would like to extend my heartfelt appreciation to my friends for their encouragement, understanding, and motivation during this demanding journey. Their presence, both as a source of camaraderie and as a sounding board for ideas, has been truly invaluable.

Finally, I would like to acknowledge all the individuals who have contributed to this research in various ways, directly or indirectly, but whose names may not be mentioned here. Your support and contributions are greatly appreciated.

Thank you all for your unwavering support, guidance, and belief in my abilities. This dissertation would not have been possible without each and every one of you.

Abstract

This paper presents a Kreol Morisien web-based spellchecking application designed to enhance written communication by ensuring accuracy and precision. The system uses state-of-the-art techniques to achieve unparalleled performance. By employing a dictionary lookup approach with a 100% recall rate, the system effectively detects misspelled words, providing comprehensive coverage. To cater to the unique linguistic characteristics of Kreol Morisien, the application integrates a mapping system that converts French words to Kreol leveraging their phonetic similarities. To enhance the effectiveness of correction suggestions, the manipulated (mapped) word undergoes processing using the Damerau-Levenshtein and Jaro-Winkler algorithms to identify the closest words in terms of writing, while Soundex is utilized to identify the closest words in terms of pronunciations. Additionally, the metaphone algorithm is used to enhance the ranking of same-sounding words in the suggestion list as it is more accurate than Soundex, resulting in a remarkable mean reciprocal rank of 93.75%. Accessible through a web-based interface, the system offers users an intuitive experience and delivers powerful spellchecking capabilities. As Kreol Morisien gains recognition in Mauritius with its inclusion in primary, secondary, and tertiary education, and plans to introduce it in the parliament, the availability of digital content in Kreol becomes essential. Developing tools like this application encourages more people to learn the language while preserving its cultural significance.

List of Abbreviations

API	Application Programming Interface
FN	False Negative
FP	False Positive
KM	Kreol Morisien
MRR	Mean Reciprocal Rank
NLP	Natural Language Processing
RR	Reciprocal Rank
SC	Spell Checker
TN	True Negative
TP	True Positive

CHAPTER 1

INTRODUCTION

1. Introduction

Spell checking is a well-known problem in Natural Language Processing (NLP) dating back to the 1960s (Damerau, 1964). Spell checkers are now an essential component of a wide range of computer applications, including web browsers, document processing, email clients and so on. There are several spell checkers that are commercial and non-commercial that are available today in the market for various popular languages.

Methods of spell checking serve two purposes. The first is to identify potential typing errors that a user may make. Misspellings, according to Mitton (1996) can be connected to the writer's bad writing and spelling skills, learning difficulties like dyslexia, and basic performance mistakes known as typos. Non-native speakers' written output is especially significant in spell checking considering that they are more prone to mistakes than native speakers. These occurrences provide a broad variety of spelling options, which a spellcheck should really be trained to identify. The second role of spell checkers is to recommend the appropriate spelling of a typographical error or, at the very least, a list of options that include the intended word appears. This is frequently accomplished by computing the distance between the misspelled word and a group of probable candidates.

1.1 Problem Statement

Kreol Morisien (KM), a creole language spoken in Mauritius, lacks a dedicated spell checker despite its increasing usage and importance as a major aspect of Mauritian cultural identity. With influences from French and Haitian Creole, KM serves as a lingua franca and is taught in schools at the School Certificate level, with plans to include it in the Higher School Certificate (HSC) curriculum by 2025. The Prime Minister of Mauritius, Pravind Kumar Jugnauth, has also expressed intentions to establish KM in legislature, further emphasizing its significance (News, 2022). However, individuals, including students, struggle with writing accurate KM words due to the limited availability of digital content and resources for the language. To address this challenge, the development of a KM spellchecker and corrector is crucial to support accurate writing and transcription of KM terms, benefiting students, as well as legislative and court stenographers.

The current state of KM highlights several issues: a lack of knowledge among individuals regarding precise KM word writing, limited availability of digital content, and a shortage of resources for the language. This situation discourages people from learning KM, despite its increasing prominence in various sectors in Mauritius. To address these challenges and promote the learning and use of KM, the development of tools, such as a spellchecker and increased digital content, is necessary. By providing support for accurate KM writing and facilitating accessibility to learning resources, these tools can encourage individuals from around the world to embrace the language and contribute to its preservation and growth, aligning with the vision set forth by Prime Minister Pravind Kumar Jugnauth.

1.2 Objectives of the Study

The aim of this dissertation is to develop a Kreol Morisien spell checker to enhance the accuracy and readability of written communication in the language. This will be achieved through the following objectives:

1. Building a comprehensive database of Kreol Morisien words.
2. Developing algorithms to detect spelling errors effectively.
3. Creating a mapping system to predict appropriate corrections based on French influence.
4. Generating accurate suggestions for correcting misspelled Kreol Morisien words.
5. Designing a user-friendly interface for easy usage of the spell checker.

By attaining these goals, this research hopes to contribute to the development of language technology for under-resourced languages, to encourage the use of Kreol Morisien in a variety of sectors, to improve language diversity and to include it in digital communication.

1.3 Gantt Chart

Table 1 illustrates the project schedule.

TABLE 1: GANTT CHART

	Dec	Jan	Feb	Mar	Apr	May	Jun	Jul
Introduction								
Background Study								
Analysis								
Design								
Implementation								
Testing & Evaluation								
Documentation								

The remaining chapters of this report are structured as follows:

- ***Chapter 2 (Background Study)*** provides a definition of relevant terms and explores previous works in the field.
- ***Chapter 3 (Analysis)*** presents the functional and non-functional requirements and conducts an analysis of various tools and technologies applicable to this project.
- ***Chapter 4 (Design)*** showcases the user interface of our spellchecker, along with architectural and sequence diagrams that illustrate the interactions among different modules and users.
- ***Chapter 5 (Implementation)*** details the actual implementation of the application, including code snippets for reference.
- ***Chapter 6 (Testing and Evaluation)*** both black-box and white-box testing are carried out, as well, the spellchecker performance is evaluated.
- ***Chapter 7 (Conclusions)*** offers a summary of the work accomplished, highlights the obtained results, discusses encountered challenges, and outlines potential future endeavors.

CHAPTER 2

BACKGROUND STUDY

2. Background Study

This section highlights the project's research activities and current systems. This will allow crucial elements of present systems to be identified and used to construct a Kreol Morisien spell checker and corrector system. Several crucial terms are also defined.

2.1 Key Terminology and Concepts for Spell Checker

When we talk about spell checkers, there are many important terms that fall under the spell-checking area. Table 2 contains keywords that one needs to know when investigating this field.

TABLE 2: WORDS RELATED TO SPELL CHECKING

Spell Checker	A piece of software that identifies any spelling errors and provides corrections for those mistakes. (Bhaire *et al.*, 2015)
Auto correct	A software function that automatically identifies and corrects incorrect words. (Basri, Alfred and On, 2012).
Language Model	A statistical model that assigns probabilities to sequences of words in a language (Jurafsky and Martin, 2019).
Corpus	A big, organized collection of texts used to provide linguistic data for language analysis (Sinclair, 1991).
Orthography	The set of norms and rules for writing a language, including spelling, punctuation, and capitalization (Crystal, 1997).
Machine Learning	It is an artificial intelligence discipline that focuses on the creation of models and algorithms that enable computer systems to boost their effectiveness on a given job by learning from data (Alpaydin, 2010).
Natural Language Processing (NLP)	Natural language processing (NLP) is a branch of artificial intelligence and computer science that studies how machines interact with human language (Jurafsky and Martin, 2019).
Text Processing	Text processing is the automated process of extracting relevant information from text or transforming it into a more organized format (Mani, 2001).
Tokenization	The process of splitting a sentence into individual words/tokens.(Pirinen, Pirinen and Lindén, 2010)
Stemming	The process that separates a word into its fundamental root component and affixes, without conducting a thorough morphological analysis. (Islam and Khan, no date)
Stop Words	They are frequently used words that are omitted from text analysis (Manning, Prabhakar and Schütze, 2009).
Part-of-speech tagging	The technique of recognizing and marking the part of speech of each word in a document is known as tagging (Jurafsky and Martin, 2019).

2.2 Spell Checking and Kreol Morisien.

Spell checker is a program that assists users in identifying and correcting spelling problems in text. Spell checker is perhaps one of the most essential technologies for personal computers since it truly helped people to communicate effectively by providing error-free writing. It is very professional to use grammar checking when communicating (Milliman, 2022) and an online spell checker allows you to confidently express your thoughts (Microsoft, 2023).

Kreol Morisien is a Mauritian language that is dependent on French and has influences from other languages such as Haitian Creole and Bhojpuri. Implementing a spell checker for Kreol Morisien presents various obstacles due to its unique properties. However, there is a considerable need for a Kreol Morisien spell checker because the language is widely spoken in Mauritius. The language spoken at home by 86.5% of Mauritians is Kreol, whereas just 4.1% speak French and 0.5% speak English (Ministry of Finance and Economic Development, 2012). There is a rising need for digital material in the language since the number of people who speak this language is outrageously high.

The auto recommendation of words is an essential aspect of spell checkers. Based on the context of the phrase, this function recommends alternate spellings for misspelled words. Because of the complexities of the KM language, auto-suggestion is very effective. Suggestion of words can greatly assist users in quickly and properly correcting misspelled words.

The creation of a Kreol Morisien spell checker is critical to the language's promotion and preservation. With the increased usage of digital technology, there is a greater need for digital material in Kreol Morisien, such as spell checkers.

2.3 Types of spelling errors.

Several strategies based on spelling errors and trends, often known as error patterns, have been researched in this section. The prominent among these studies were performed by Damerau (1964). According to these investigations, there are two categories of spelling mistakes, typographic errors, and Cognitive errors.

2.3.1 Typographic errors

This error occurs when the spelling of words is known to the one typing but due to his mistyping the error occurred. These errors have a direct relationship with the keyboard and thus they do not adhere to any linguistic requirements. According to the study of Damerau (1964), it shows that 80% of these errors fall into one of the categories listed below.

Taking the Kreol Morisien word “laboutik” as example to explain the below cases:

a) Insertion of single letter

For example, typing “laboutick”.

b) Deletion of single letter

For example, typing “labutik”.

c) Substitution of single letter

For example, typing “labootik”.

d) Transposition of adjacent letters

For example, typing “labuotik”.

2.3.2 Cognitive errors

These errors occur when the correct spelling of words is not known to the one typing, and these may happen because the pronunciation of correct words and misspelled words are the same. For example, typing “grimace” for grimas and “lafrick” for lafrik.

2.4 Error detection and correction in spell checkers.

2.4.1 Error Detection Techniques

To propose a set of potential corrections, a spellchecker must initially detect misspelled words. This necessitates the creation of an error model, an area of research that has been explored by numerous authors referenced in Hládek et al.'s survey (2020).The primary techniques employed to identify erroneous words within a text are either rooted in dictionary research or rely on n-gram analysis.

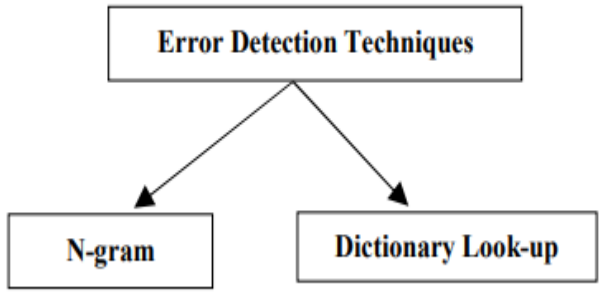


FIGURE 1: ERROR DETECTION TECHNIQUES

2.4.1.1 N-gram Analysis

Shannon (1951), introduced the concept of the n-gram model in language processing, which has since been widely used for various purposes, including spelling correction. One significant benefit of the n-gram model is its language independence, making it applicable across different languages. N-gram analysis is a technique used for error detection, specifically for identifying incorrect words within a text. Instead of comparing individual words to a dictionary, this approach involves creating a square matrix of size n, where the most frequently occurring n-grams are stored (Pollock and Zamora, 1984). The text string is then divided into sets of adjacent n-grams for comparison.

2.4.1.2 Dictionary lookup

In spelling correction, the most common method is to check a word against a dictionary. If the word is not found, it is considered incorrect. To speed up the process, dictionaries store basic word forms, and morphological analyzers help generate derived forms. These analyzers determine word structure and can guess unknown words. They handle inflection (verb conjugation, noun/adjective declension) and derivation (word formation from roots and affixes or other words). Tools like lemmatizers, conjugators, and decliners aid in this analysis.

Researchers have worked on improving dictionary search speed for faster results among which we mention:

- **Hash table** is a popular data structure for efficient dictionary searches (Kukich, 1992). It was initially introduced in 1953 (Salifou and Naroua, 2014). The key benefit of using a hash table is its ability to provide selective access to the desired word using a hash code, leading to a significant reduction in response time. However, a major drawback is that the hash function should ideally produce minimal collisions and evenly distribute clues within the given range, which can be a limitation.
- **Binary search trees** are valuable for determining if a word belongs to a larger collection of dictionary words (Bentf et al., 1985). Various variations of binary search trees have been employed to accelerate dictionary searches in spell checking processes.
- **Finite automata** have found applications in dictionary and text search algorithms. One notable algorithm in this domain is the (Aho and Corasick, 1975) algorithm. This algorithm operates by traversing an abstract data structure called a dictionary, which contains the desired words, by reading the text letter by letter. The data structure is designed for efficiency, ensuring that each letter of the text is read only once. Typically, the dictionary is implemented using a digital sort or tree with suffix links added. The sort can be viewed as a representation of the transition function of a deterministic finite automaton. Once the dictionary is constructed, the algorithm exhibits linear complexity relative to the size of the text and the searched strings.

2.4.2 Error Correction Techniques

After detecting errors in a spelling correction process, the subsequent stage involves generating potentially correct forms of the misspelled word.

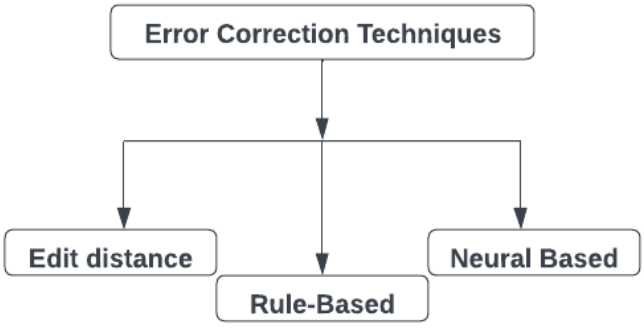


FIGURE 2: ERROR CORRECTION TECHNIQUES

a) Edit-Distance

The edit distance is a measure of the minimum number of operations required to transform one word into another, including inserting, deleting, transposing, and substituting letters (Vienney and Bioud, 2004). The higher the number of dissimilarities between the strings, the greater the edit distance.

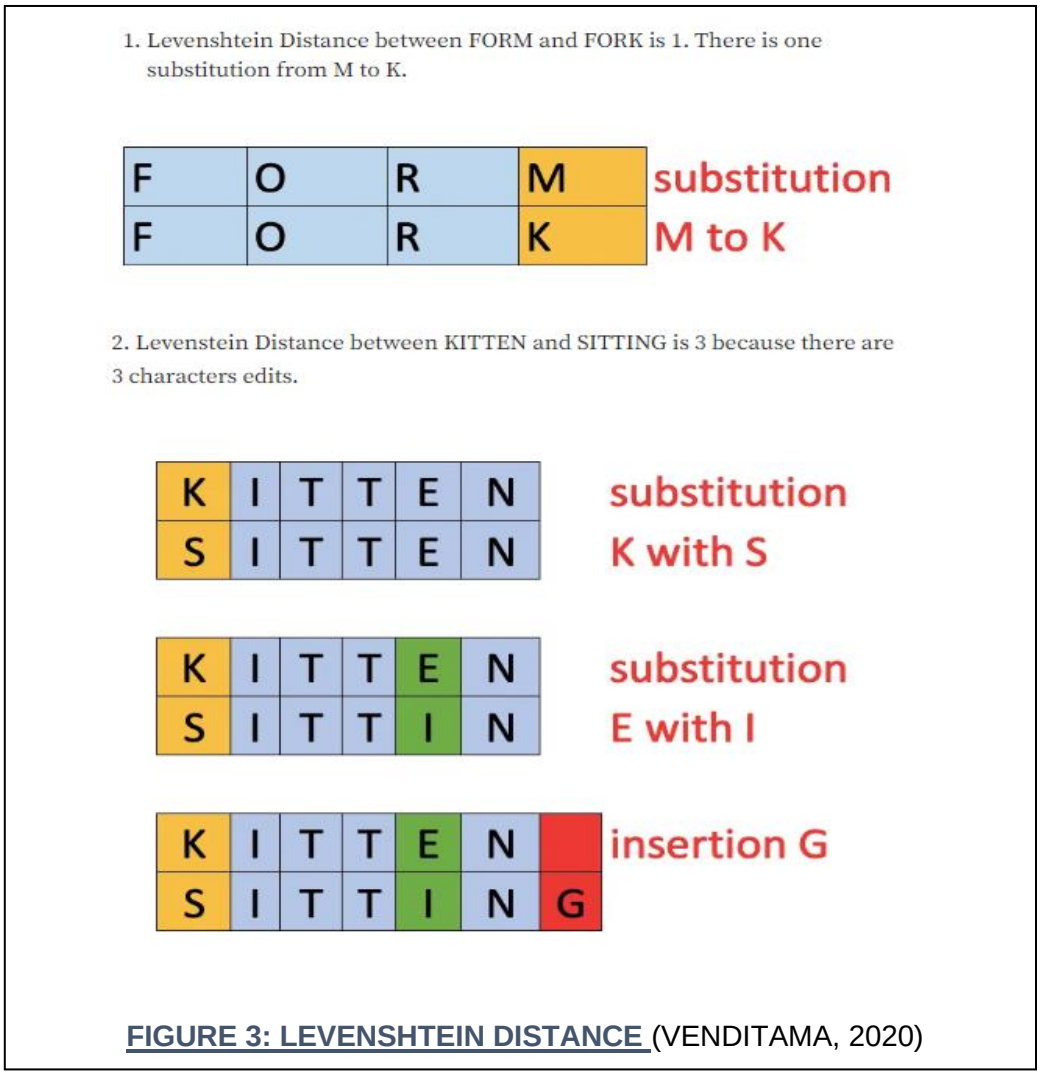


FIGURE 3: LEVENSHTEIN DISTANCE (VENDITAMA, 2020)

b) Rule-Based

In spell checkers, rule-based procedures are critical for fixing misspelled words. Phonetic techniques, which examine the sounds of words to propose improvements, are one prominent rule-based method. The Soundex algorithm, for example, employs a set of criteria to translate words into a four-character code depending on their pronunciation. If a misspelled word is input, the algorithm can propose a correction by comparing its Soundex code to the codes of correctly spelt words from a dictionary. Another rule-based strategy is to include an array of common misspellings and their fixes in the spell checker's rules, a rule can advise changing “evidamen” to “ evidaman” or “acsan” to “aksan”. usage and spelling norms. Finally, rule-based procedures are critical in guaranteeing that spell checkers find and fix misspelled words.

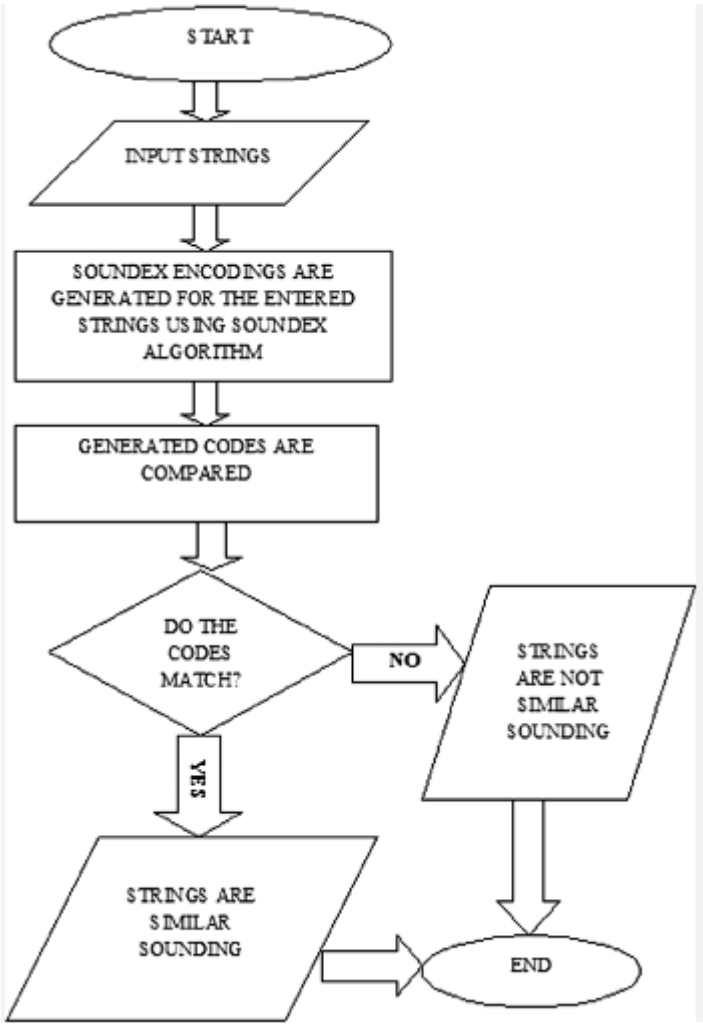


FIGURE 4: FLOWCHART SHOWING THE FLOW OF THE SOUNDEX (BHATTACHARYYA, 2018)

c) Neural Based

This approach employs deep learning algorithms to propose replacements for misspelled words, are a more contemporary approach to spell checking. The usage of language models such as GPT (Generative Pretrained Transformer) and BERT (Bidirectional Encoder Representations from Transformers) which have been pre-trained on massive volumes of text data and can offer contextually appropriate word recommendations depending on the user's input.

For example, if a user has entered “Mo anvi manz enn pom” but accidentally types “pomme” instead of “pom”, the neural-based spelling checker may utilize the context of the phrase to propose the right spelling of “pom”. This is because the language model was trained on a vast corpus of literature and recognizes the circumstances in which terms are used.

2.5 Related Works

In this section, different techniques that were employed by existing spell checkers for any language will be explored.

2.5.1 Use of morphology technique to provide suggestions for misspellings.

Shaalan et al. (2012), proposed this approach for the Arabic language and Dhanabalan et al. (2003), as well suggested this to correct misspellings. The approach taken by them is morphological based, which means misspelled words would be deconstructed into their constituent morphemes. Then, to identify the inaccuracy, each morpheme is compared to a lexicon of known morphemes and words. The morphemes may include a suffix, a prefix, or none, as well as a root word.

The proposed approach of the authors in steps are as follows here “dysconnection” as a misspelled word to explain the steps:

- Detect the misspelled word: The spell checker first determines that the input word "dysconnection" is spelled incorrectly and must be corrected.

- Morphological analysis: It is used by the spell checker to break down the incorrect word into its constituent morphemes. The term may be split down into "dys-" and "connection" in this example.

- Error detection: A typo in the prefix "dys-" is detected by the spell checker. It compares the recognized morpheme to a database of known morphemes and determines that the proper prefix is "dis-" rather than "dys-."

- Affix files: The spell checker uses affix files, which contain lists of common prefixes, suffixes, and other word elements, to suggest the correct spelling of the word. In this case, the affix file contains the prefix "dis-", which the spell checker uses to suggest the correct spelling of "disconnection".

- Suggestion: Based on the results of the morphological analysis and the affix file lookup, the spell checker suggests the correct spelling of "disconnection" as the replacement for the misspelled word "dysconnection".

- Correction: If the user accepts the suggestion, the spell checker replaces the misspelled word with the correct spelling, "disconnection".

To summarize, the morphological method and the usage of affix files aid in spell checking by breaking words into smaller constituent pieces, finding problems, and recommending repairs based on recognized word elements.

Figure 5 shows how error is detected, and suggestions are provided using the morphology technique.

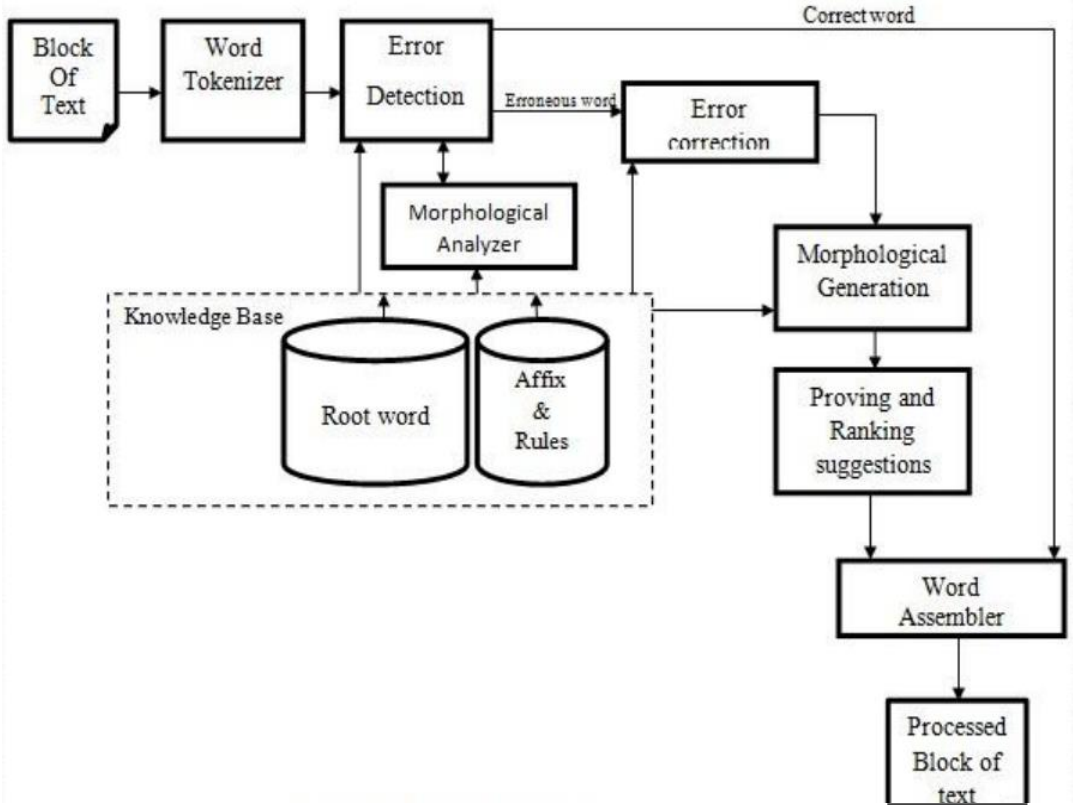


FIGURE 5: SYSTEM ARCHITECTURE(GANFURE AND MIDEKSO, 2014)

2.5.2 Use of Soundex to suggest candidates based on matching pronunciations and shaping to give suggestions of similar looking characters.

Uzzaman and Khan (2004), presented a phonetic encoding for Bangla that spelling checkers may take advantage to generate more accurate recommendations for misspelled words. The process of encoding is based on the Soundex technique, which has been tailored to correspond to Bangla phonetics. Bhatti et al. (2014), built a system with a combination of the phonetic-based Soundex algorithm and the Shapex algorithm for pattern or glyph matching, resulting in an accurate and efficient recommendation list for erroneous or misspelled Sindhi words.

The proposed approach of the author of using Soundex in steps:

1. Built a table for Soundex coding rule.
    - In this step, a unique code is assigned to each alphabet of the language.
    - Table 3 contains Soundex coding rules:

TABLE 3: PHONIX CODING RULES (UZZAMAN AND KHAN, NO DATE)

Code	*Letters*
0 (not coded)	A, E, I, O, U, H, W, Y
1	B, F, P, V
2	C, G, J, K, Q, S, X, Z
3	D, T
4	L
5	M, N
6	R

2. Encode the misspelled word to a four-character reduced form.

1. Replace all but the first letter of *s* by its phonetic code.
2. Eliminate any consecutive repetition of codes.
3. Eliminate all occurrences of code 0 (i.e., eliminate vowels, and the letters *H*, *W* and *Y*).
4. Return the first four characters of the resulting string.

FIGURE 6: THE SOUNDEX ALGORITHM(BHATTI ET AL., 2014)

For example, to calculate the encoded value of the Kreol Morisien misspelled the word "ledicasyon".

- Rule 1 from figure 6, substitute all the letters by their respective phonetic code except for the first letter. (Use table 4 as reference for the phonetic codes)

TABLE 4: ENCODING FOR THE WORD MISSPELLED WORD "LEDICASYON"

Letter	Phonetic Code
L	L {remains as it is because it is the first letter}
E	0
D	3
I	0
C	2
A	0
S	2
Y	0
O	0
N	5

- As you can see the encoding for "ledicasyon" becomes "L030202005". From rule 2 in figure 6, "L030202005" becomes "L0302025" when remove consecutive occurrences of same code.
- From rule 3 in figure 6, all 0's should be removed so, "L0302025" becomes "L3225".

- “L3225” should be reduced to a four-character code so it becomes “L322”.

For example, to calculate the encoding of the correct spelling of “ledicasyon” which is “ledikasion”.

- Rule 1 from figure 6, substitute all the letters by their respective phonetic code except for the first letter. (Use table 4 as reference for the phonetic codes)

TABLE 6: ENCODING FOR THE CORRECT WORD “LEDIKASION”

Letter	Phonetic Code
L	L {remains as it is because it is the first letter}
E	0
D	3
I	0
K	2
A	0
S	2
I	0
O	0
N	5

- As you can see the encoding for “ledikasion” becomes “L030202005”. From rule 2 in figure 6, “L030202005” becomes “L0302025” when remove consecutive occurrences of same code.
- From rule 3 in figure 6, all 0’s should be removed so, “L0302025” becomes “L3225”.
- “L3225” should be reduced to a four-character code so it becomes “L322”.

Both correct word “ledikasion” and incorrect word “ledicasyon” have same encoding so one of the probable candidates for the misspelled word is “ledikasion” when use the Soundex algorithm.

The proposed approach of the author of using Shapex after applying Soundex in steps:

1. Built a table for Shapex coding rule.
    - In this step, a unique code is assigned to each resembling alphabet of the language.
    - Table 5 assigned shape number to resembling character.

TABLE 5: SHAPEEX BASED CHARACTER GROUPS (BHATTI ET AL., 2014)

Shape Code	Similar Shaped Character Groups						S.No
0		إ	أ	ل	ا	آ	
1			ﭘ	ﭙ	ﺑ	ب	
2		ﭧ	ﭨ	ﭩ	ﺗ	ت	
3		ﭽ	ﭼ	ﭽ	ﭽ	ج	
4					ﺧ	ح	
5			ذ	ڈ	ﺬ	د	
6					ﺪ	ﺩ	
7		و	م	ز	ﺰ	ر	
8					ﺵ	س	
9					ﺿ	ص	
A					ﻆ	ط	
B					ﻎ	ع	
C				ق	ﻗ	ف	
D						ک	
E			گ	ﮒ	ﮓ	ک	
F					ﻦ	ن	
G					ﻫ	ھ	
H				ء	ﯨ	ي	

TABLE 6: LIST OF SIMILAR WORDS GENERATED USING SHAPEX(BHATTI ET AL., 2014)

S.No	Sindhi Similar Words List using	ShapeEx Code
1.	اسامي , اسامه , اساري , آسامي , آساري , اشاري , اشاره , لشاري	x0807ZZ
2.	وگر , مگر , مگر , زگر , رڳڙ , رڳو , رڳر , رڳو , رگو , رگڙ , رگر , وڳڙ , وڳو , وڳر	x7E7Z
3.	ارهو , ارهر , ازهو , لوهڙ , لوهو , لوهر , لڙهو , آزهر , آرهڙ , اوهڙ , اوهو , اوهر , ازهر	x07G7Z
4.	زماني , مزاني , رزاني , رمائي , روائي , مولئ , موليه , مراني , وراني , وزاني , ومائي ,	x770ZZ
5.	مرون , مرمن , وروڻ , ورون , وڙون , ورمن , ورزن , روڙڻ , رومڻ , رومن , روزن , موڙڻ , موڙن , رمون , رمزن , مومن , زمون , مورڻ , مورن , ممون , ووڙڻ , وورڻ ,	x777FZ
6.	ومري , رزمي , مروّ , مروه , مرمي , ورڙي , وزمي , وروئ , ورمي , ورزي , ورري , زومي , زوري , زمره , زرڙي , روڙي , رومي , روزي , روزه , روره , موڙي , رمزي , مومي , موزي , موزه , موري , ووڙي , ومڙي ,	x777ZZ
7.	ڳول , ڳوا , ڳرا , ڳڙا , ڳرا , ڳول , ڳوا , ڳزل , ڳرا , ڳڙا , ڳڙا ,	xE70Z
8.	ڳوه , ڳوئ , ڳمي , ڳري , ڳڙي , ڳري , ڳوي , ڳوه , ڳوئ , ڳزي , ڳري , ڳره , ڳڙي , ڳڙه , ڳڙي	xE7ZZ
9.	كرايل , كوليا , كرايل , كرايا , كمايل , كوليل ,	xD70Z0Z
10.	كنجڙ , كنجو , كنجر , كڇو , كنچر , كنجر , كنجو , كنجر , كڇجو	xDF37Z

Both the Soundex and Shapex techniques have been widely used to handle the issue of misspellings and variations in words and names in a variety of applications. While Soundex is a simpler and easier to develop technique, it has limits in properly portraying specific names and words. Shapex, on the other hand, is a more sophisticated algorithm that can handle more variances and deliver better outcomes.

2.5.3 Use of n-gram technique to correct real-word errors (context-sensitive errors).

Verberne (2002), proposed a technique to identify real-word errors in a sentence. He has used the n-gram approach to recommend better suggestions of real-word errors.

a) Real-word errors

Real-word errors, also known as semantic errors, arise when a word is spelled incorrectly in a way that creates a different, real word that is not the intended word. For example, if someone types “form” instead of “from”, that would be a real-word error.

Because the misspelled word is still a correct word and may not be identified by spellcheck or other automated programs, real-word mistakes can be more difficult to detect than other forms of spelling errors, such as phonetic errors.

Proofreading your work carefully and slowly, evaluating each word for correctness and meaning, is one technique to discover real-word mistakes. It might also be beneficial to have someone else go through your work to detect any mistakes you may have overlooked. Using a range of spelling and grammatical tools, including both automatic and human editors, can also assist in reducing real-word errors and other sorts of errors in your work.

Kukich (1992), categorizes real-word mistakes based on their potential outcomes. She classifies the outcomes of real-world mistakes into four categories:

1. Syntactic errors (for example, the students are doing *their* assignments).
2. Semantic errors (for example, he spent his life travelling around the *word*).
3. Structural errors (for example, I need *three* colors: red, blue, yellow, and green).
4. Pragmatic errors (for example, he studies at the University of Mauritius in *Moka*, and she studies at the University of Toronto).

The approach for context-sensitive spell checking that Verberne (2002), used for his study seeks to detect and repair all forms of errors independent of the (resulting) word class. Only Mays et al. (1991), considered all forms of mistakes in all word classes. Both devised a strategy based on the probability of word trigrams.

Figure 7 is the graphical representation of the context-sensitive spell-checking algorithm.

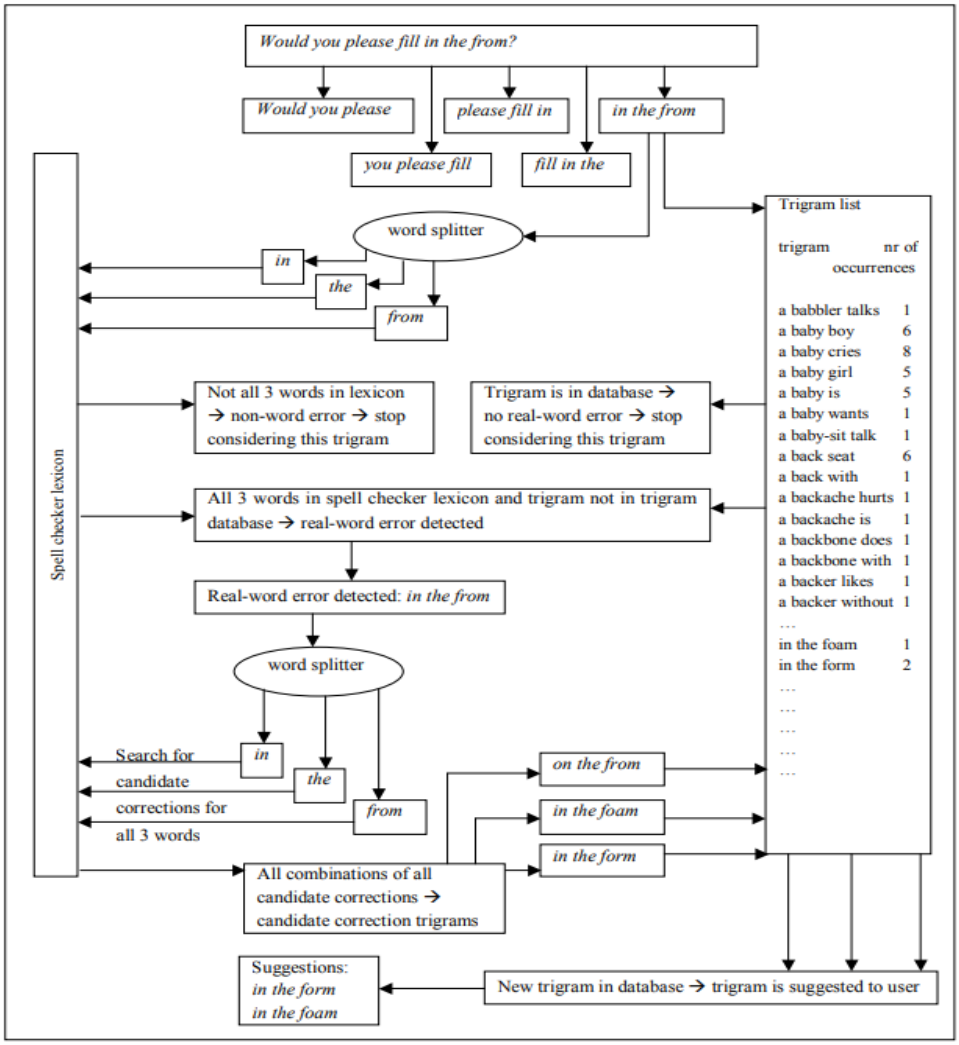


FIGURE 7: A GRAPHICAL REPRESENTATION OF THE CONTEXT-SENSITIVE SPELL-CHECKING ALGORITHM (VERBERNE, 2002)

b) How does context-sensitive spell-checking work as show in figure 8?

Steps:

1. When the sentence "*Would you please fill in the from?"* is typed, the word "*from*" is a correct word but it is not used in the correct context.

2. The trigram of the sentence "*Would you please fill in the from?*" are generated which are "*Would you please*", "*you please fill*", "*please fill in*", "*fill in the*" and "*in the from*".

3. All the trigrams are looked in the list of trigrams, if it is found in the trigram that means there are no real-word errors.

4. The trigram "*in the from*" is least likely to be found in the corpus of trigrams because it is not a correct trigram so, this trigram is then passed to a word splitter to extract each word from the trigram.

5. All the words generated by the splitter are checked in the lexicon and if the word is not found in the lexicon, then it is a non-word error.

6. If all the words are found in the lexicon but the trigram is not found in the list of trigrams then, it is a real-word error.

7. The real-word error is "*in the from*" and this trigram is again split, and their candidate corrections are suggested and are replaced and are again looked in the trigram. For example, for the trigram "*in the from*" their suggestions are "*on the form*", "*in the foam*" and "*in the form*" and these generated trigrams are again looked in the corpus of trigrams and if one of them is not found it is discarded so here "*on the form*" is least likely to be located in the context so this trigram is discarded and we are left with "*in the foam*" and "*on the form*" and these are the final suggestions.

2.5.4 Deep Learning-based Spell Correction.

Ahmadzade et al. (2021), presented a spelling correction solution based on deep learning. The method is divided into two parts: an autoencoder-based spelling mistake detector and an attention-based spell correction model.

The proposed approach of Ahmadzade et al. (2021) in five stages comprising of the two main components, autoencoder-based spelling mistake detector and attention-based spell correction model:

A. Data Collection and Pre-Processing:

1. Collect a large corpus of text in the target language, containing both correctly spelled and misspelled words.
2. Pre-process the text data by removing unnecessary punctuation, converting all text to lowercase, and applying any necessary cleaning techniques, such as stemming or lemmatization.

B. Tokenization and Padding:

1. Tokenize the pre-processed text into individual words.
2. Convert the tokens into numerical representations using one-hot encoding or other vectorization techniques.
3. Pad the tokenized sequences to ensure that they are of equal length.

C. Autoencoder and Decoder:

1. Train an autoencoder model on the correctly spelled text from the corpus.
2. The autoencoder is trained to take in a sequence of words as input and produce a reconstructed sequence of words with the same length as output.
3. The decoder is trained to take in the encoded representation produced by the autoencoder and generate a sequence of words.
4. During inference, use the trained autoencoder and decoder to reconstruct the input text without errors.

5. If the autoencoder and decoder are unable to reconstruct the input text without errors, flag the input text as containing spelling errors.

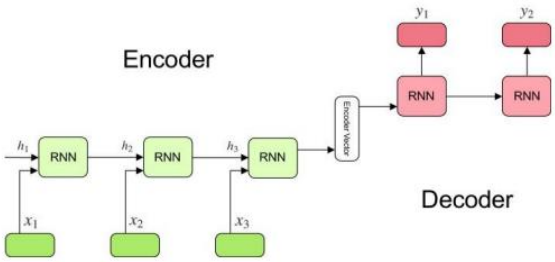


FIGURE 8: ENCODER-DECODER ARCHITECTURE.(AHMADZADE ET AL., 2021)

D. Attention Mechanism:

1. For each misspelled word detected by the autoencoder, generate a list of candidate corrections using a set of spelling correction rules or a language model trained on a large corpus of text.
2. Use an attention mechanism-based neural network model to select the correct spelling correction from the list of candidates.
3. The attention mechanism takes as input the misspelled word, its context, and the list of candidate corrections and produces a corrected version of the word.
4. The attention mechanism is trained on a dataset of misspelled words and their correct corrections.

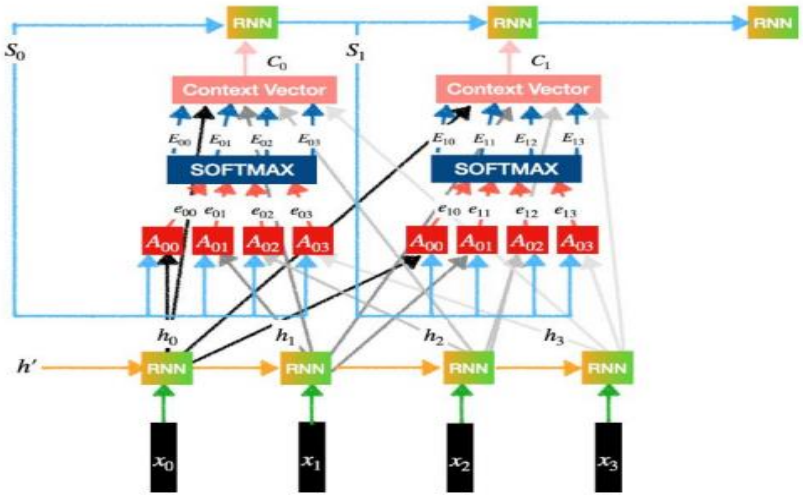


FIGURE 9: ATTENTION MECHANISM(AHMADZADE ET AL., 2021)

E. The Proposed Model for Spelling Correction:

1. The input text is tokenized and padded as in step B.
2. The autoencoder and decoder are used to detect spelling errors in the input text as in step C.
3. For each misspelled word detected by the autoencoder, a list of candidate corrections is generated as in step D.
4. The attention mechanism is used to select the correct spelling correction from the list of candidates as in step D.
5. The corrected text is produced by replacing the misspelled words with their corrected versions.
6. The model's performance is evaluated on a separate test dataset of misspelled words and their correct corrections, using metrics such as accuracy, precision, recall, and F1-score.

CHAPTER 3

ANALYSIS

3. Analysis

This section will assess the functional and non-functional requirements of the project, evaluate available tools and technologies, and present functional modeling diagrams. The approach will be formal, adhering to academic standards, and crucial for the project's successful implementation. The examination will include a comparison of the benefits and drawbacks of each tool and technology, determining their suitability for the project. The presentation of the functional modeling diagrams will provide an overview and set expectations for the project.

3.1 Proposed Solution

The proposed solution encompasses a comprehensive five-step approach to address the challenges in language processing and error correction for Kreol Morisien (KM). The steps are as follows:

1. **Data Collection:** The initial step involves the collection of relevant data to create a comprehensive Kreol Morisien dictionary. This dictionary will serve as a valuable resource for validating KM words and ensuring their accuracy.

2. **Text Preprocessing:** A rigorous text preprocessing stage will be employed to clean and standardize the input KM text. This includes tasks such as tokenization, removing punctuations, converting text to a consistent case, and handling special characters or symbols.

3. **Rule-Based and Dictionary-Lookup Approach:** The solution will leverage a rule-based approach in combination with dictionary lookup techniques. Linguistic rules will be defined to identify and correct errors, inconsistencies, and anomalies in the KM text. These rules will encompass grammar, syntax, and spelling patterns.

4. **Mapping System for French-to-KM Stemming:** To enhance language processing capabilities, a mapping system will be developed specifically for stemming French words to their corresponding Kreol Morisien equivalents as French closer to KM. This system will facilitate accurate translation and transformation of French terms into KM, thereby expanding the scope of language coverage.

5. **User-Friendly Error Correction Suggestions:** The final step involves providing user-friendly suggestions to correct misspelled words. When encountering misspelled or erroneous words, the system will employ algorithms to generate intelligent suggestions based on similarity measures or phonetic algorithms. These suggestions will assist users in rectifying errors and improving the overall accuracy of the KM text.

By integrating these five steps, the proposed solution aims to enhance language processing, error correction, and text quality in the context of Kreol Morisien. It offers a systematic and rule-based framework to ensure the accuracy, consistency, and comprehensiveness of KM text, thereby benefiting various applications and users relying on KM language resources.

3.2 Functional and Non-Functional Requirements

3.2.1 Functional Requirements

Table 7 shows the functional requirements of the Kreol Morisien spell checker.

TABLE 7: FUNCTIONAL REQUIREMENTS OF THE SYSTEM

FR1	**User login**	The system shall allow admin to log in with their credentials to do administrative tasks.
FR2	**Real-time spell-checking**	Real-time spell-checking: The system shall enable users to type KM text and check for spelling mistakes in real-time.
FR3	**Underline incorrect words**	The system shall underline incorrect KM words in red colour to draw the user's attention.
FR4	**Suggest correct words**	When a user clicks on an KM incorrect word, the system shall display a list of suggestions for the correct spelling.
FR5	**Replace incorrect word**	When a user selects a word from the suggestion list, the system shall replace the incorrect KM word with its correct spelling.
FR6	**Copy/paste/clear functionality**	The system shall allow users to copy, paste and clear text from the spell-checking interface.
FR7	**Display character, word, and error count**	The system shall display the number of characters, words, and errors made by the user on the spell-checking interface.
FR8	**File spell-checking**	The system shall allow users to select a file to check for spelling mistakes and correct errors.
FR9	**Corpus management**	The system shall allow administrators to add and remove words to the corpus of correct Kreol Morisien words.
FR10	**Image to text spellcheck**	The system shall allow users to choose an image file for spellchecking and correct errors.
FR11	**Last word completion suggestion**	The system shall allow users to see a set of complete text suggestions for the last word being typed when its length is equal to three.
FR12	**API for spellchecking**	Develop an API that provides spellchecking functionality specifically designed for the KM language. The API should accept input word in KM and provide suggestions for the correction.

3.2.1 Non-Functional Requirements

Table 8 shows the functional requirements of the Kreol Morisien spell checker.

TABLE 8: NON-FUNCTIONAL REQUIREMENTS

NFR1	**User-friendly interface**	The system shall be designed with a user-friendly interface to ensure ease of use and minimal training required for users.
NFR2	**Password protection**	The system shall be password-protected to prevent unauthorized access by non-admin users.
NFR3	**Web-based application**	The system shall be implemented as a web-based application to ensure accessibility from any device with internet connectivity.
NFR4	**Scalability**	The system shall be scalable to handle multiple users connecting simultaneously with the application.
NFR5	**Speed of suggestions**	The system shall provide suggestions for misspelled words within a reasonable amount of time to ensure that users do not experience delays in their work.
NFR6	**Timely updates**	The system shall add new words to the dictionary to ensure that the system is up to date.
NFR7	**Authentic dataset**	The system shall provide an authentic dataset of KM words to ensure accuracy and reliability of the spell-checking process.

3.3 Evaluation of Tools and Approaches

We gather and analyze data on available tools and technologies to assist with project planning and implementation. This information will guide decision-making regarding the most suitable tools and approaches for meeting the project's goals.

3.3.1 Programming Languages

TABLE 9: PROGRAMMING LANGUAGES

Feature	Python	JavaScript	C++
Easy to use	√ √ √	√ √	√
Have a range of powerful libraries	√ √ √	√ √	√
Community Support	√ √ √	√ √ √	√
Web-Based Real-Time Communication	√ √ √	√ √	√
Web Development Frameworks	√ √ √	√ √ √	√
Cross-Platform Capability	√ √ √	√ √ √	√ √

3.3.2 Data Collection

Data needs to be collected from various websites and articles written in Kreol Morisien. This is done to create a comprehensive database of Kreol Morisien words, which would be used to train the spell checker. The data collection process is essential to ensure the accuracy and completeness of the database, as it allowed for the inclusion of a wide range of words and their variations. The data collected from these sources would be used to develop and refine the algorithms used in the spell checker.

3.3.2.1 Beautiful Soup

According to Richardson (2019), Beautiful Soup is a Python library for extracting data from HTML and XML files. It is particularly effective when working with nested tags or unstructured data due to its ability to navigate the document tree and support a range of search mechanisms. With robust error handling and encoding support, Beautiful Soup is a popular choice for data collection in the Python community, simplifying the process of data analysis from websites and other sources.

3.3.2.2 Scrapy

Scrapy, as defined by Mcguffee (2015), is an open-source web crawling framework written in Python. It serves the main purpose of supporting web scraping. It provides tools for crawling websites, scraping data, and storing it in various formats such as CSV, JSON, and SQL databases. Scrapy is widely used for web scraping and data mining tasks.

Each of the three tools presented has its advantages and disadvantages, table 10 summarizes this (Faruque, 2020):

TABLE 10: COMPARISON BETWEEN SCRAPY AND BEAUTIFULSOUP

	Scrapy	BeautifulSoup
Easy to learn	***	***
Readout dynamic content	**	*
Realize complex applications	***	**
Robustness against HTML errors	**	****
Optimized for scraping performance	***	*
Pronounced ecosystem	***	**

3.3.3 The text-preprocessing Process

Text Preprocessing is a critical step in natural language processing that involves cleaning and preparing raw text for further analysis. This process involves transforming the input text into a set of tokens, which are individual words or groups of words that are used to identify patterns and derive insights. Tokenization helps to standardize the text, making it easier to analyze and draw meaningful conclusions (Anandarajan et al., 2019). This step will help identify misspelled words.

For example, we have the string “Mo* pe al !labutik, e apre $a mo pou al Lamer.”.

After passing through steps, we will see its output.

1. Case Conversion

   Converting Kreol Morisien text to a consistent case in spell checkers, specifically lowercase, helps reduce the number of misspelled words to be flagged by ignoring capitalization errors. Studies, such as the work of Uysal and Gunal (2014), have shown that converting text to lowercase improves the accuracy of machine learning classifiers used in spell checkers.

   Output: “mo pe al !labutik e apre $a mo pou al lamer.”*

2. Tokenization

   Tokenization, as defined by Michelbacher, Brenz and Schütze (2013), involves splitting input text into individual words using white space characters as delimiters.

   Output: [“mo”, “pe”, “al”, “!labutik,”, “e”, “apre”, “$a”, “mo”, “pou”, “al”, “lamer.”]*

3. Punctuation and special characters removal

   The removal of punctuation and special characters is a critical step in spell checking that enhances accuracy when checking similarity between incorrect and words in dictionary. Rastogi (2022), emphasizes that eliminating punctuation marks ensures fair and equitable treatment of all text.

   Output after punctuation removal:

   [“mo”, “pe”, “al”, “labutik”, “e”, “apre”, “$a”, “mo”, “pou”, “al”, “lamer”]*

Output after special characters removal:

[“mo”, “pe”, “al”, “labutik”, “e”, “apre”, “a”, “mo”, “pou”, “al”, “lamer”]

4. Stemming and lemmatization

Stemming and lemmatization are preprocessing techniques employed in spell checkers to enhance accuracy by reducing words to their base or root form. By converting variations of a word into a common base form, stemming and lemmatization help streamline the analysis process, resulting in more efficient and effective spell checkers (Manning et al., 2009).

Output remains same as previous stage

3.3.4 Data Structure for Dictionary lookup

To efficiently perform spell-checking, we must first divide the text into separate words. Then, we can compare each word to a dictionary to identify any mistakes. As the number of words increases to spellcheck, it becomes crucial to select an appropriate data structure for fast searches.

3.3.4.1 Brute Force Linear Search

When we use brute force method, we must go through each item in the initial array (words we want to investigate in the dictionary) and verify its presence in the second array (the dictionary). When going through each item in the initial array, it is possible that you may need to conduct a linear search in the second array, which will take O(m) time (with m representing the size of the second array).

The time complexity of the brute force method would be O (n * m) if the first array contains n elements. As the sizes of both arrays increase, the time complexity grows proportionally to the product of their sizes.

3.3.4.2 Sets

When a set is used, we would start by converting the second array into a set, which takes O(m) time when using sets. Then, for each element in the first array, we would test for membership in the set. The average time complexity for membership testing in a set is O (1). Therefore, the overall time complexity for this step would be O(n).

The time complexity of using sets for two arrays, one with n elements and the other with m elements, would be O (n + m) overall. The rate at which the time complexity increases is directly related to the combined sizes of the arrays as they grow.

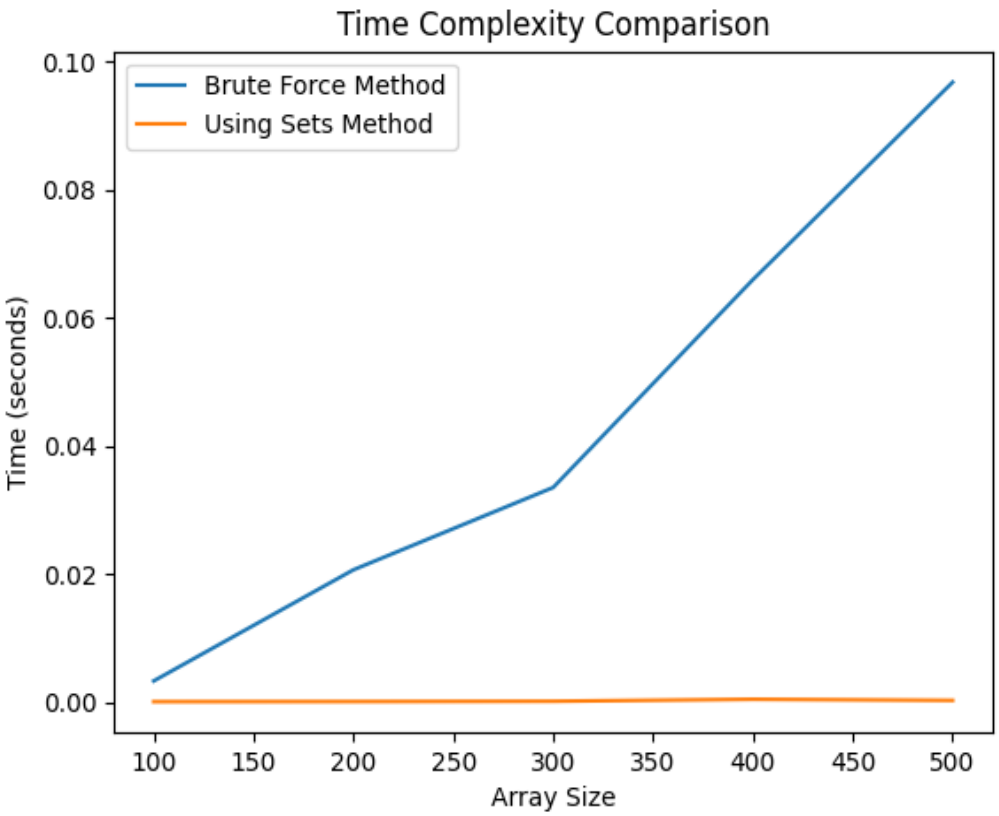


FIGURE 10: TIME COMPLEXITY COMPARISON

3.3.5 Approaches for error detection in spell checker

3.3.5.1 Dictionary-Based Approach

In the field of spell checking, Mishra and Kaur (2013), offer valuable insights into the dictionary lookup technique. As per their reference, this technique involves meticulously examining each word within the input text to ascertain its presence in a specific dictionary. The spell checker checks if a given word is included in the designated dictionary. If the word is found in the dictionary, it is deemed correct and passes the spell check. Conversely, if the word is absent from the dictionary, it is classified as an error and promptly flagged for further attention.

TABLE 11: ADVANTAGES AND DISADVANTAGES OF DICTIONARY LOOKUP

Advantages	Disadvantages
1. **Accuracy:** Ensures high accuracy in detecting misspelled	1. **Limited Vocabulary:** may incorrectly flag newer terms not included in the dictionary as misspelled.
2. **Efficiency:** Quick and efficient identification of misspelled words.	2. **Lack of Contextual Understanding:** Focusing on individual words.
3. **Language Support:** Wide range of users and linguistic variations.	3. **Proper Noun and Personal Names:** Struggle with proper nouns and personal names.
4. **User Customization:** Users can customize and expand dictionaries	4. **Maintenance and Updates:** Need to regularly update the dictionary to add new word

3.3.4.2 Rule-Based Approach

Rule-based approaches use predefined rules or patterns to identify spelling errors. These rules can be based on spelling patterns, phonetics, context, or specific linguistic rules. For example, rules can be created to detect repeated letters, missing vowels, or common typing errors. According to Sydenham and Thorn (2005), a rule-based system comprises if-then rules, a collection of facts, and an interpreter that governs the application of the rules based on the provided facts.

Below are some examples of rule-based approach:

1. **Pattern Matching:** Rule-based spell checkers use patterns to identify common spelling errors. For example, a rule could be defined to flag words that end in “ment” but should end in "man" (e.g., “kareman” instead of “karement”).

2. **Phonetics:** Rule-based systems can incorporate phonetic rules to detect errors based on sound-alike words. For instance, a rule might identify “plastic” as a potential error when “plastik” is more contextually appropriate.

3. **Contextual Rules:** Spell checkers can exploit contextual rules to identify errors based on nearby words or grammatical structures. For instance, a rule may flag “p al laboutik” as an error and suggest “pe al laboutik” instead.

4. **Capitalization and Punctuation:** Rule-based spell checkers often include rules to check for capitalization and punctuation errors, such as flagging a lowercase letter at the beginning of a sentence or missing punctuation at the end.

5. **Doubled or Repeated Letters:** Rules can be defined to detect repeated or doubled letters within words, like “bann” being misspelled as “ban”.

TABLE 12: ADVANTAGES AND DISADVANTAGES OF RULE-BASED

Advantages	Disadvantages
1. **Accuracy:** High accuracy by relying on predefined rules for error detection.	1. **Limited Coverage:** May miss errors or struggle with words and patterns not covered by the predefined rules, leading to false negatives.
2. **Contextual Understanding:** Incorporate contextual rules, enabling consideration of neighboring words	2. **Maintenance and Updates:** Regular updates and maintenance are necessary to keep rule-based systems up to date with evolving language, new words, and changing patterns.
3. **Customizability:** Customized with specific rules to accommodate domain-specific terms and vocabulary.	3. **Complexity:** Complex, requiring expertise and effort to create and fine-tune the rules.
	4. **Handling Exceptions**: Challenges when handling exceptions that do not fit the predefined rules.

3.3.5.3 Statistical Language Models

In the field of Statistical Language Modeling, also referred to as Language Modeling or LM, researchers focus on developing probabilistic models capable of predicting the next word in a sequence by analyzing the preceding words (Engati Simply Intelligent, 2021). These models learn the probability of word occurrence using text examples, with some models considering a short context of words while more advanced models operate at the level of sentences or paragraphs. Generally, language models primarily operate at the word level (Engati Simply Intelligent, 2021).

Below are some examples of Statistical Language Models:

1. **N-gram Models:** N-gram models analyse the frequency and likelihood of word sequences in each language. Spell checkers can use these models to suggest corrections by considering the probabilities of word combinations. For example, if “Mo pe al bwar delwil” is entered, the model can suggest “Mo pe al bwar delo” based on the higher probability of the word sequence.

2. **Language Modelling with Context:** Statistical language models consider the surrounding context to provide more accurate suggestions. By analysing the probabilities of word combinations within a given context, spell checkers can offer corrections that fit the context better. For example, if “Li pe al zwer” is entered, the model can suggest “Li pe al zwe” based on the higher likelihood of “zwe” following “Li pe al”.

3. **Word Frequency Analysis:** Statistical language models consider the frequency of words in a language corpus. Spell checkers can exercise this information to prioritize suggestions based on the likelihood of certain words. For instance, if “garson” is misspelled as “garcon” the model can suggest “garson” as a more probable correction due to its higher frequency in the language corpus. According to Engati Simply Intelligent (2021), Statistical Language Models (SLMs) have both advantages and disadvantages. Table

13, provides a summary of these pros and cons, highlighting the benefits and challenges associated with SLMs.

TABLE 13: ADVANTAGES AND DISADVANTAGES STATISTICAL LANGUAGE MODELS

Advantages	Disadvantages
1. Easy to train on a large corpus.	1. **Zero probabilities:** In a tri-gram language model with a large vocabulary, unobserved events can lead to zero probabilities and infinite perplexity. Smoothing techniques are needed to address this issue.
2. Work well in most tasks.	2. **Exponential growth:** The number of n-grams grows exponentially with the vocabulary size, leading to many n-grams to handle.
	3. **Lack of generalization:** MLE techniques may assign zero probabilities to unseen events, leading to limited generalization capabilities. For example, if “black pawn” is unseen in training, "MLE" assigns zero probability to “black pawn”.

3.3.5.4 Machine Learning Algorithms

Machine learning algorithms, such as supervised learning classifiers or neural networks, can be trained on labeled data to detect spelling errors. These algorithms learn from patterns in the data and can classify words as correct or incorrect based on the learned knowledge. Ray and Rachna (2019), describe Machine Learning as the process where a computer program is assigned tasks and considered to have learned if its performance in these tasks improves with accumulated experience. By making decisions and predictions based on data, the machine enhances its abilities in executing these tasks over time.

Below are some examples of ML Algorithms:

1. **Linear Regression:** A supervised learning model used for predicting a continuous output variable based on one or more input features, assuming a linear relationship between them.
2. **Logistic Regression:** A binary classification model used to predict the probability of an event occurring, based on input variables.
3. **Decision Tree:** A tree-like model that makes decisions by splitting data based on feature values, allowing for complex decision-making.
4. **Random Forest:** An ensemble model consisting of multiple decision trees, combining their predictions to make more accurate predictions.
5. **Support Vector Machines (SVM):** A model that separates data points into different classes by finding an optimal hyperplane in a high-dimensional space.
6. **Naive Bayes:** A probabilistic classification model based on Bayes' theorem, assuming independence between features.
7. **K-Nearest Neighbors (KNN):** A model that classifies data points based on the classes of their nearest neighbors in the feature space.
8. **Neural Networks:** Complex models inspired by the structure and function of the human brain, capable of learning and making predictions on various tasks.

9. **Gradient Boosting Machines (GBM):** An ensemble model that combines weak predictive models, such as decision trees, in a sequential manner to improve overall performance.
10. **Convolutional Neural Networks (CNN**): Neural networks specifically designed for processing grid-like data, such as images, by leveraging convolutional layers.

Highlighted by Data Flair (2023), some notable advantages and disadvantages of Machine Learning are as follows:

TABLE 14: ADVANTAGES & DISADVANTAGES OF MACHINE LEARNING ALGORITHMS

Advantages of Machine Learning	Disadvantages of Machine Learning
1. Easily identifies trends and patterns in large datasets	1. Requires massive and inclusive/unbiased high-quality training data
2. No human intervention needed, allowing for automation	2. Time and resource-intensive for learning and processing
3. Continuous improvement in accuracy and efficiency	3. Interpretation of results can be challenging
4. Capable of handling multi-dimensional and multi-variety data	4. Highly susceptible to errors, especially with biased training data
5. Wide range of applications across industries.	

3.3.6 Lexicographical Distance to get correction of closest similar strings.

Distance algorithms are computational techniques used to measure the similarity or dissimilarity between two strings or sequences. They calculate the distance based on specific rules or metrics, such as the number of insertions, deletions, or substitutions required to transform one string into another. These algorithms play a crucial role in tasks such as spell checking, approximate matching, and clustering similar data, enabling effective analysis and decision-making processes in various fields.

a) Levenshtein Distance

Levenshtein distance, as stated by Haldar and Mukhopadhyay (2011), is a straightforward metric that can serve as a powerful tool for approximating strings. It calculates the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform one string into the other. The algorithm is named after Vladimir Levenshtein, who introduced it in 1965.

TABLE 15: ADVANTAGES & DISADVANTAGES OF LEVENSHTEIN

Advantages	Disadvantages
1. Simple and widely used metric for string similarity	1. Does not consider transpositions
2. Captures insertions, deletions, and substitutions	2. Not suitable for measuring similarity between unequal-length strings

The formula to calculate the Levenshtein distance (excluding character transposition) is as follows:

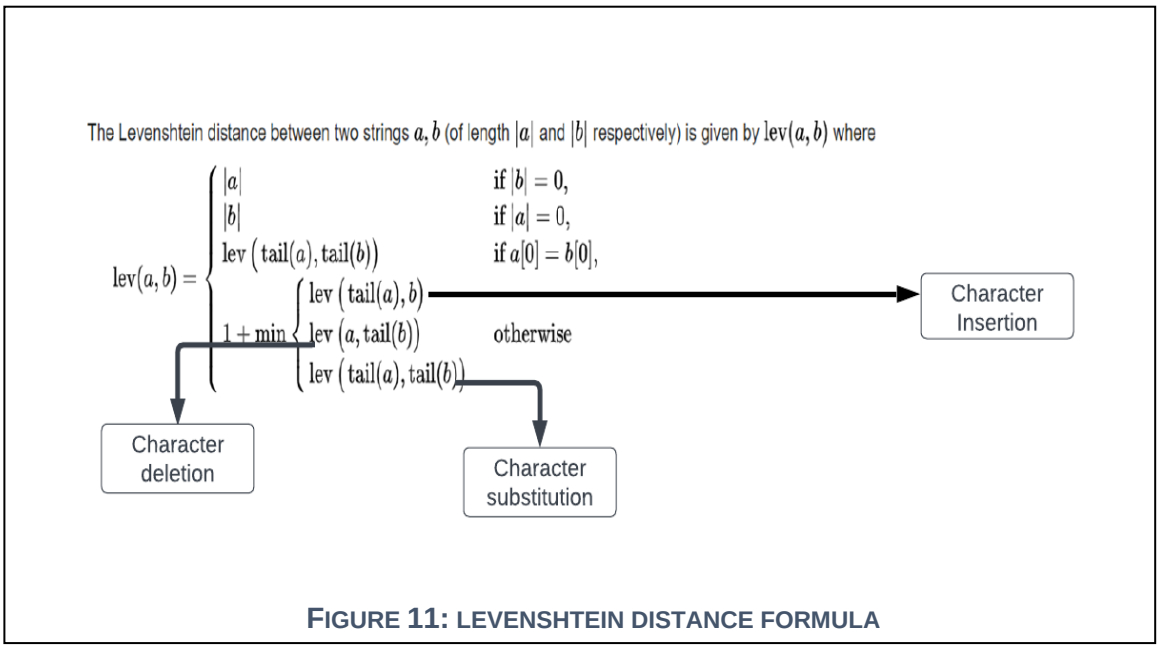


FIGURE 11: LEVENSHTEIN DISTANCE FORMULA

Example of how Levenshtein distance works:

We must calculate distance between string "*zornaz*" and "*zoranz*" so here is the calculation using Levenshtein distance:

		z	o	r	a	n	z	
	0	1	2	3	4	5	6	
z	1	0	1	2	3	4	5	"z" to "zoranz" requires 5 edit distance
o	2	1	0	1	2	3	4	"zo" to "zoranz" requires 4 edit distance
r	3	2	1	0	1	2	3	"zor" to "zoranz" requires 3 edit distance
n	4	3	2	1	1	1	2	"zorn" to "zoranz" requires 2 edit distance
a	5	4	3	2	1	2	3	"zorna" to "zoranz" requires 3 edit distance
z	6	5	4	3	2	3	2	"zornaz" to "zoranz" requires 2 edit distance

TABLE 16: LEVENSHTEIN EDIT DISTANCE CALCULATION

To convert “zornaz” to “zoranz”, 2 edit distance is required:

1. Start substring “zor” remain as it is.
   - z o r n a z
   - z o r a n z
2. End substring “z” remains as it is.
   - z o r n a z
   - z o r a n z
3. We are left with “na” in “zornaz” and “an” in “zoranz” which are causing the difference.
   - z o r n a z
   - z o r a n z
4. We must transform “na” to “an” so we can do this as follows:
   - “n” replaces “a” => “zoraaz”
   - “a” replaces “n” => “zoranz”

b) Damerau-Levenshtein

The Damerau-Levenshtein (DL) distance, as described by Zhao and Sahni (2019), refers to the length of the optimal edit sequence when all four edit operations are allowed. It calculates the minimum number of single-character edits (insertions, deletions, substitutions, or transpositions) required to transform one string into the other. The algorithm is named after Fred Damerau, who introduced it in 1964.

TABLE 17: ADVANTAGES & DISADVANTAGES OF DAMERAU-LEVENSHTEIN

Advantages	Disadvantages
1. Includes transpositions as a valid edit operation	1. More computationally expensive than Levenshtein distance
2. Suitable for correcting common typing errors	2. Similarity may be affected by the number of transpositions

The formula to calculate the Damerau-Levenshtein distance (including character transposition) is as follows:

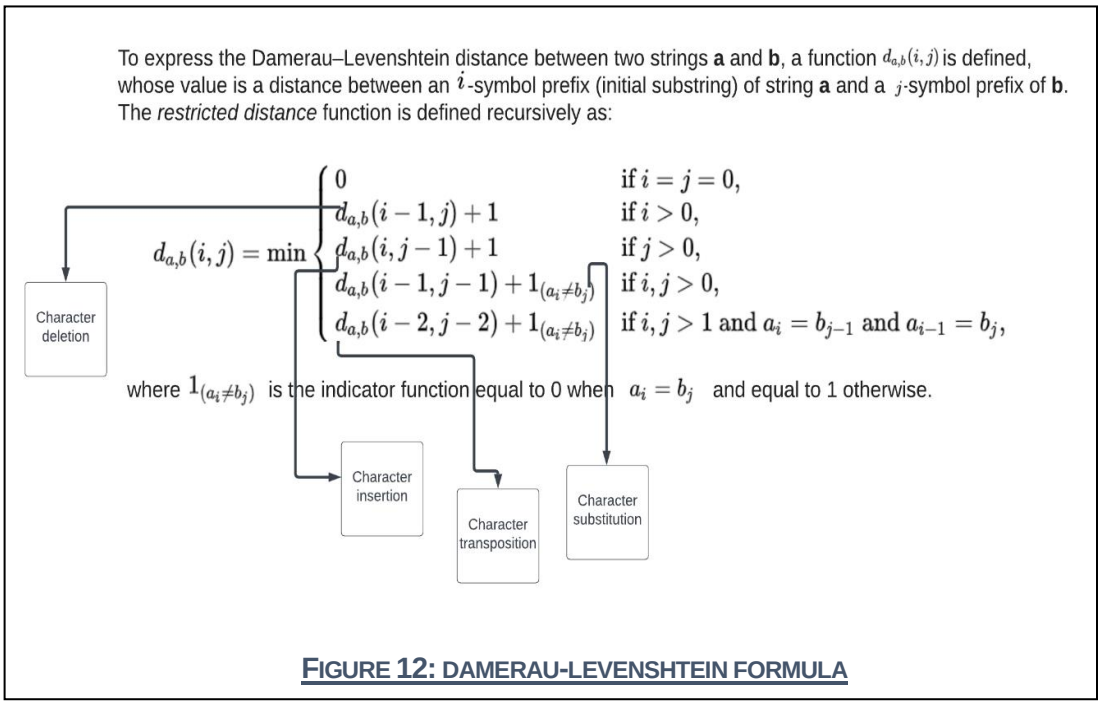


FIGURE 12: DAMERAU-LEVENSHTEIN FORMULA

Example of how Damerau-Levenshtein distance works:

We must calculate distance between string "*zornaz*" and "*zoranz*" so here is the calculation using Damerau-Levenshtein distance:

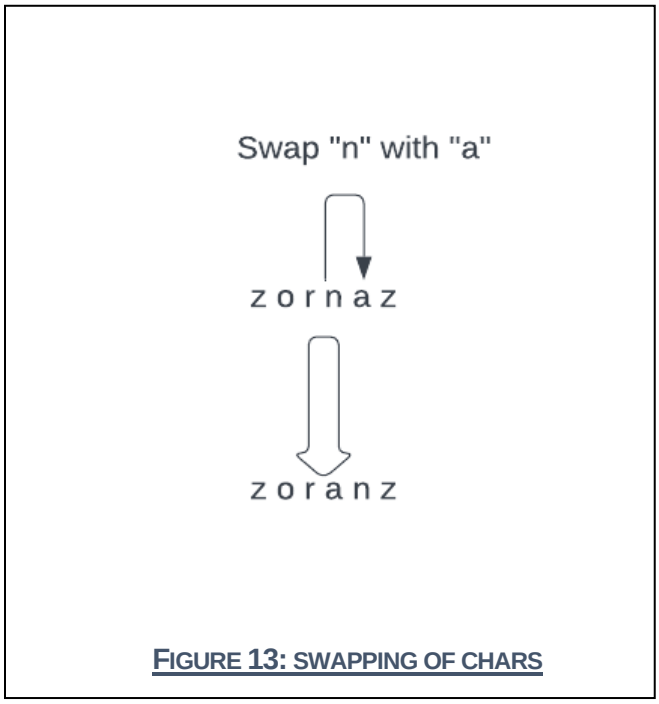


FIGURE 13: SWAPPING OF CHARS

1. We can see to convert "zornaz" to "zoranz" we just needed to swap "n" with "a" so this leads to one edit distance instead of two. Instead of substituting "n" with "a" and "a" with "n".
2. So, the Damerau-Levenshtein distance is same as Levenshstein distance, but it also caters for transposition of chars.

c) Jaro Distance

The Jaro distance, as outlined by Laux (2015), was specifically developed for comparing strings of shorter length, such as names. Jaro distance is a measure of the similarity between two strings. It calculates a value between 0 and 1 that indicates how closely the strings match. The algorithm was introduced by William E. Winkler in 1990.

TABLE 18: ADVANTAGES & DISADVANTAGES OF JARO DISTANCE

Advantages	Disadvantages
1. Captures similarity in terms of character matches and order	1. Ignores transpositions
2. Handles variations in string lengths	2. Fails to account for common typing errors

The formula to calculate the Jaro similarity is as follows:

$$Jaro\ similarity = \begin{cases} 0, \text{ if } m=0 \\ \frac{1}{3}\left(\frac{m}{|s1|} + \frac{m}{|s2|} + \frac{m-t}{m}\right), \text{ for } m!=0 \end{cases}$$

where:

- **m** is the number of matching characters
- **t** is half the number of transpositions
- where **|s1|** and **|s2|** are the lengths of strings s1 and s2 respectively.

FIGURE 14: JARO SIMILARITY FORMULA

Example of how Jaro similarity works:

We must calculate the similarity score between string “*zornaz*” and “*zoranz*” so here is the calculation using Jaro similarity:

Calculation:

- Let s1=”zornaz”, s2=”zoranz”, so the maximum distance to which each character is matched is 4.
- It is evident that both the strings have 6 matching characters but the number of characters that are not in order is 2, so the number of transposition is 1.
- Therefore, Jaro similarity can be calculated as follows: **Jaro Similarity** = (1/3) * {(6/6) + (6/6) + (6-1)/6 } = **0.94444**

d) Jaro-Winkler Similarity

Jaro-Winkler similarity is a variation of Jaro distance that gives more weight to prefix matches. It calculates a value between 0 and 1 that indicates how closely the strings match, with higher values indicating a greater degree of similarity. The algorithm was introduced by William E. Winkler in 1990.

TABLE 19: ADVANTAGES & DISADVANTAGES JARO-WINKLER SIMILARITY

Advantages	Disadvantages
1. Builds upon Jaro distance and adds a prefix scale factor	1. Ignores transpositions
2. Emphasizes similarity in the beginning of strings	2. Fails to account for common typing errors

The formula to calculate the Jaro-Winkler similarity is as follows:

- Jaro Winkler similarity is defined as follows

  $Sw = Sj + P * L * (1 - Sj)$

  where,

  - Sj, is jaro similarity
  - Sw, is jaro- winkler similarity
  - P is the scaling factor (0.1 by default)
  - L is the length of the matching prefix up to a maximum of 4 characters.

FIGURE 15: JARO-WINKLER FORMULA

Example of how Jaro-Winkler similarity works:

We must calculate the similarity score between string "*zornaz*" and "*zoranz*" so here is the calculation using Jaro similarity:

Calculation:

- Let s1='zornaz', s2='zoranz'. The Jaro similarity of the two strings is 0.94444 (From the above calculation in *section 3.3.6 (c)*)
- The length of the matching prefix is 3 and we take the scaling factor as 0.1.
- Substituting in the formula;
  Jaro-Winkler Similarity= 0.94444 + 0.1 * 3 * (1-0.94444) = 0.961108
- The score 0.961108 for Jaro-Winkler similarity is greater than the score 0.94444 for Jaro similarity because it caters for prefix as well.

e) Smith-Waterman

The Smith-Waterman algorithm is a precise method used to characterise the alignment quality between two sequences. It enables the determination of the optimal alignment for these two sequences (Khajeh-Saeed et al., 2010).

Table 20, highlights the advantages and disadvantages of the Smith-Waterman algorithm mentioned by (Wieds, 2007):

TABLE 20: ADVANTAGES & DISADVANTAGES OF SMITH-WATERMAN

Advantages of Smith-Waterman	Disadvantages of Smith-Waterman
1. Guarantees the local optimal alignment	1. Time consuming
2. Best performance on accuracy	2. Computer power intensive
3. Precise results	

Figure 16 shows how the similarity score is computed between the two words '*rinse*' and '*rise*' using a matrix. The alignment of *r, i, s, e* gives a score of (2*4=8) and one misalignment of n with – gives a negative score of 1 so the final similarity score (8-1=7):

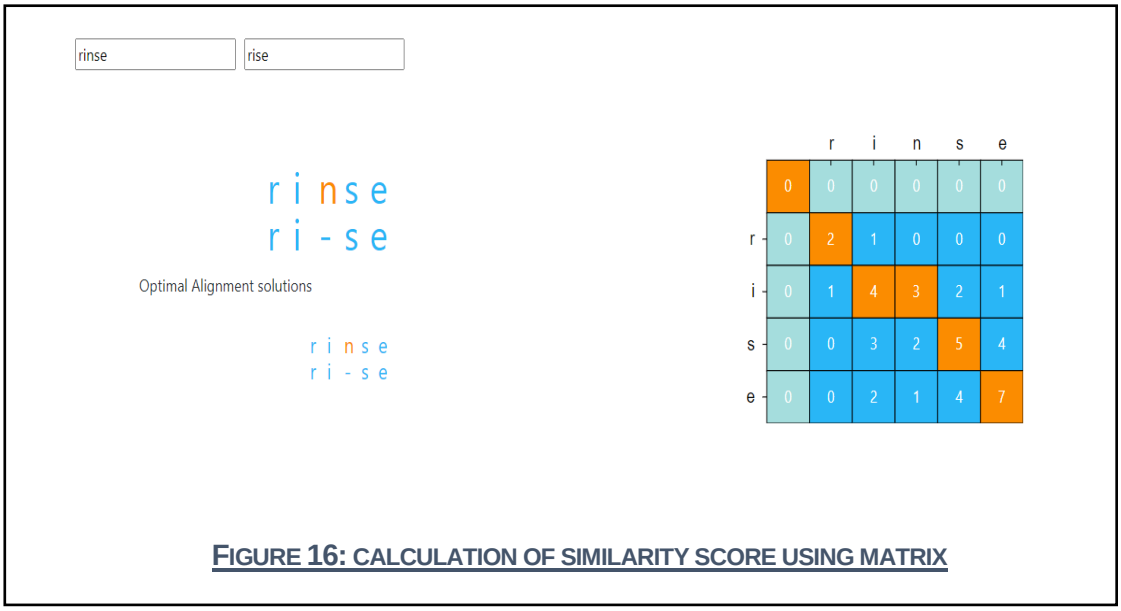


FIGURE 16: CALCULATION OF SIMILARITY SCORE USING MATRIX

3.3.7 Phonetic Algorithms to return words with similar pronunciation.

A phonetic algorithm is a specialized algorithm that detects and groups words with similar pronunciations. It enables indexing and categorization of words based on their phonetic properties (Parmar and Kumbharana, 2014).

3.3.7.1 Soundex

As cited by Parmar and Kumbharana (2014), the Soundex algorithm, initially devised by Robert C. Russell and Margaret K. Odell in 1918, creates a four-character code for each word. The first character signifies the word's initial letter, while the remaining three characters are numeric codes that capture its phonetic features. The primary objective of the Soundex algorithm is to offer a compact phonetic representation of words, facilitating efficient indexing and comparison based on pronunciation.

3.3.7.2 Metaphone

Parmar and Kumbharana (2014), highlights that the Metaphone algorithm was first created by Lawrence Phillips in 1990. Phillips further enhanced the algorithm by introducing variations such as Double Metaphone, which expanded its applicability to multiple languages. Notably, the third version of Metaphone, developed by Phillips in 2009, achieved an impressive 99% accuracy rate for English words. As a result, the Metaphone algorithms have become widely used in English spell checkers and dictionaries, demonstrating their effectiveness in handling diverse English vocabulary.

Table 21 lists the pros and cons of Soundex and Metaphone:

TABLE 21: PROS AND CONS OF SOUNDEX AND METAPHONE

Algorithm	Pros	Cons
Soundex	1. Compact codes for short names	1. Limited accuracy for longer strings
	2. Effective for family grouping	2. Difficulty handling certain variations
	3. Well-established algorithm	3. Pronunciation discrepancies
Metaphone	1. Improved accuracy for longer strings	1. Longer code lengths
	2. Handles variations more effectively	2. Relatively more complex algorithm
	3. Supports multiple languages	3. Potential for false positives

3.3.8 Database

In this section, we delve into a variety of databases that can be applied for our project. We examine and analyze the suitability of these databases in relation to our specific project requirements.

3.3.8.1 MySQL

MySQL is a popular relational database management system known for its stability, reliability, and robust feature set, making it suitable for storing and managing structured data in a wide range of applications.

Table 22 shows the advantages and disadvantages of MySQL mentioned by Denton and Peace (2003):

TABLE 22: ADVANTAGES AND DISADVANTAGES OF MYSQL

Advantages	Disadvantages
1. Low cost	1. Does not enforce foreign key relationships
2. High connectivity	2. Does not have as many features as its competitors
3. Easy to install and use	3. Lacks commit and rollback operations for transactions
4. Less demanding on resources	4. Does not support subqueries

3.3.8.2 MongoDB

MongoDB is a popular and powerful document-oriented NoSQL database that provides flexible and scalable data storage solutions for modern applications.

Boicea, Radulescu and Agapin (2012), provided insights into both the positive and negative aspects of MongoDB as listed in table 23.

TABLE 23: ADVANTAGES AND DISADVANTAGES OF MONGODB

Advantages	Disadvantages
1. Accepts larger data	1. Do not offer integrity features
2. Speed	2. Complex
3. Cross-platform	3. Lack of SQL support
4. Offers drivers for multiple programming languages	4. Indexing challenges

3.3.9 Choice of Tools

Based on the analysis of various tools, the following tools were selected.

TABLE 24: CHOICE OF TOOLS

Procedure	Chosen Tool(s)	Justification
Programming Language	Python and JavaScript	Python has support for scraping, web development and NLP libraries. JavaScript were used for dynamic content updates without requiring a server round-trip
Data Collection	BeautifulSoup	Beautiful Soup excels at parsing HTML and XML documents. It provides a convenient and intuitive interface for extracting data from complex HTML structures
Text Pre-Processing	Case conversion, tokenization, punctuations/special characters removal and stemming/lemmatization	In spell-checking, converting text to lowercase helps ignore uppercase differences. Tokenization splits sentences into smaller units for checking in the dictionary. Removing punctuation and special characters focuses on alphanumeric values for dictionary matching. Stemming and lemmatization simplify words to their base form or modify them.
Data Structure for Dictionary Lookup	Sets	Sets offer a linear time complexity (O (n + m)) for searching elements, making them more efficient than brute force (O (n * m)) as array sizes increase. Thus, sets are the preferred approach for larger arrays due to their scalability and efficiency.

Approach	Dictionary-lookup and rule-based	Dictionary-lookup enables efficient validation and processing of words based on existing lexical resources while rule-based can be beneficial for under-resourced languages by allowing the creation of language-specific rules to handle common misspellings and linguistic patterns.
Distance Algorithm	A combination of all distance algorithms	The hybrid approach helps to get more accurate suggestions of misspellings.
Phonetic Algorithm	Both Soundex and Metaphone	It produces more accurate phonetic encodings.
Database	MongoDB	MongoDB provides fast query performance, retrieval of a large volume of data and better for loosely structured data.

3.4 Use Case Model for the Spell-Checking Application

The use case diagram represents the interaction between normal users and the spell-checking application being developed. It visually illustrates a set of actions that the system is expected to perform when interacting with the user.

Figure 17 shows the features available in our spellchecker:

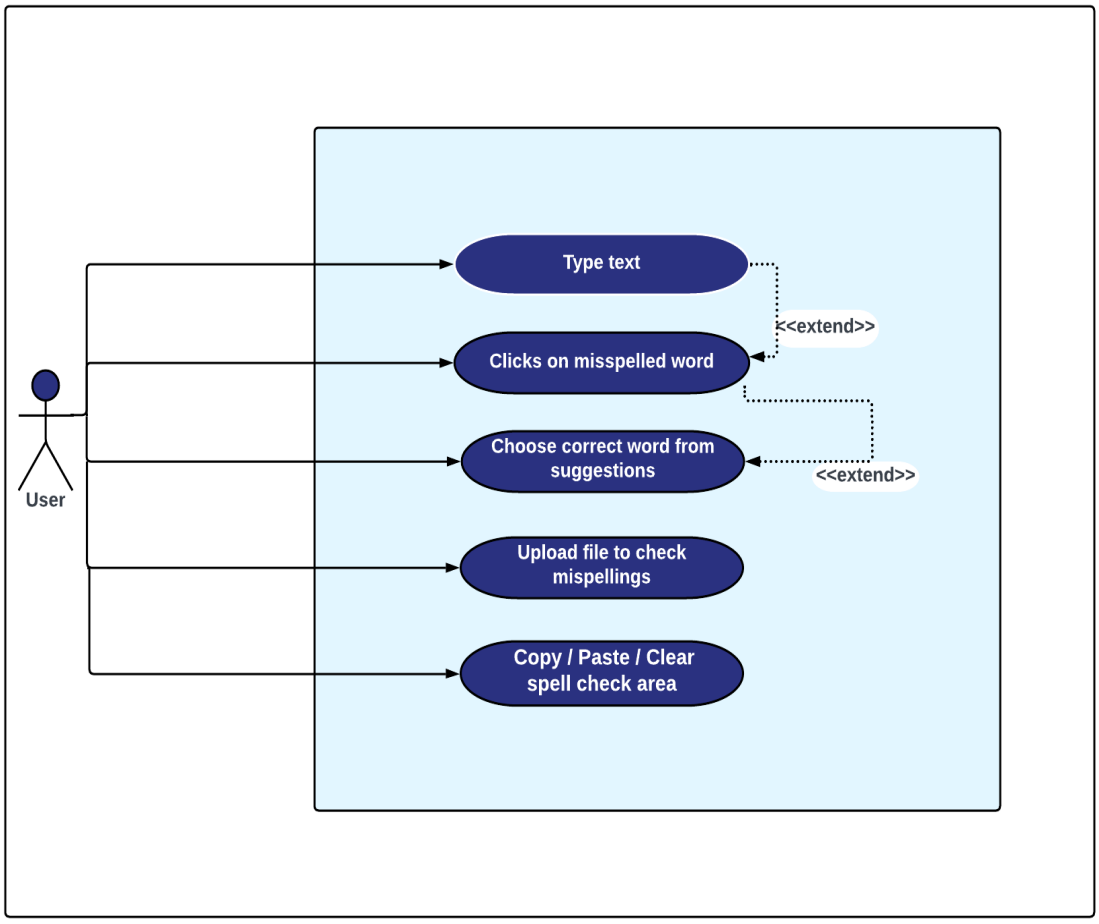


FIGURE 17: USE CASE DIAGRAM FOR USER

Figure 17 demonstrates that users have the capability to input text into the spellcheck area, select misspelled words by clicking on we choose correct alternatives from the suggestions, upload files for spellchecking, as well as perform operations such as copying, pasting, and clearing text.

Figure 18 showcases admin-system interaction and associated actions.

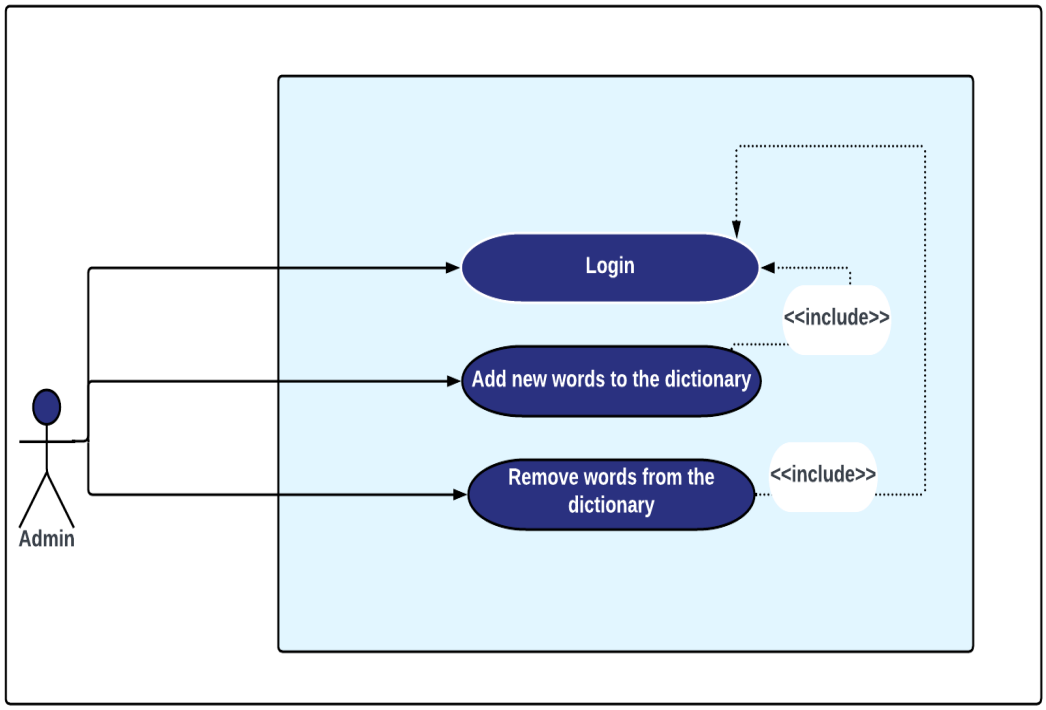


FIGURE 18: USE CASE DIAGRAM FOR ADMIN

Figure 18 illustrates that the admin can access the system by logging in and could manage the dictionary by adding or removing words.

CHAPTER 4

DESIGN

4. Design

This chapter provides an overview of the system's design and architecture. It encompasses architectural design, interface design, and database design, while also demonstrating the interaction between components and users through sequence diagrams.

4.1 Design Methodology

The design methodology chosen for this project is the agile methodology. It follows an iterative and incremental approach, allowing for the incremental development of the software. After each iteration, new modules are added, and each increment undergoes thorough testing before proceeding to the next iteration.

Figure 19 shows the design methodology diagram:

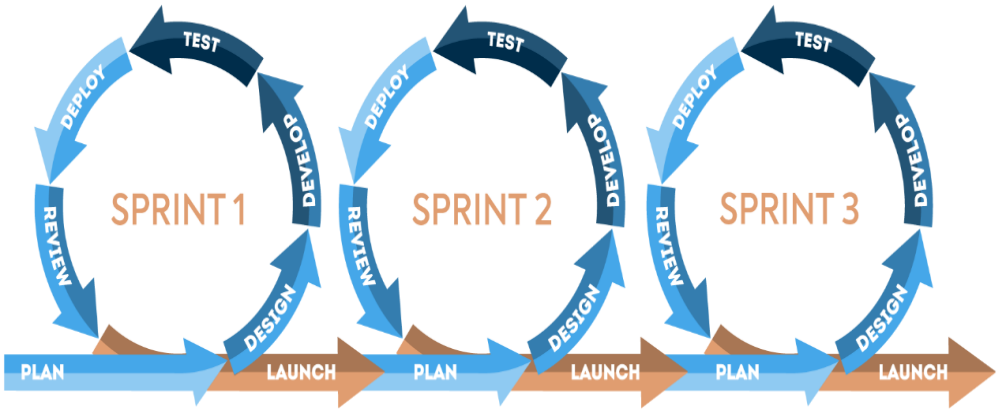


FIGURE 19: DESIGN METHODOLOGY

4.2 Architectural Diagram

The architectural design diagram depicts the interaction between different components in the system. The scraper component extracts data from web, HTML, and XML documents, creating a comprehensive dictionary. The spellchecker employs this dictionary for word validation, flagging any words not found as misspelled. Misspelled words are then passed to a hybrid distance model for accurate suggestions. This design ensures a seamless flow of data and an effective spellchecking process, enhancing the overall quality of text analysis and refinement.

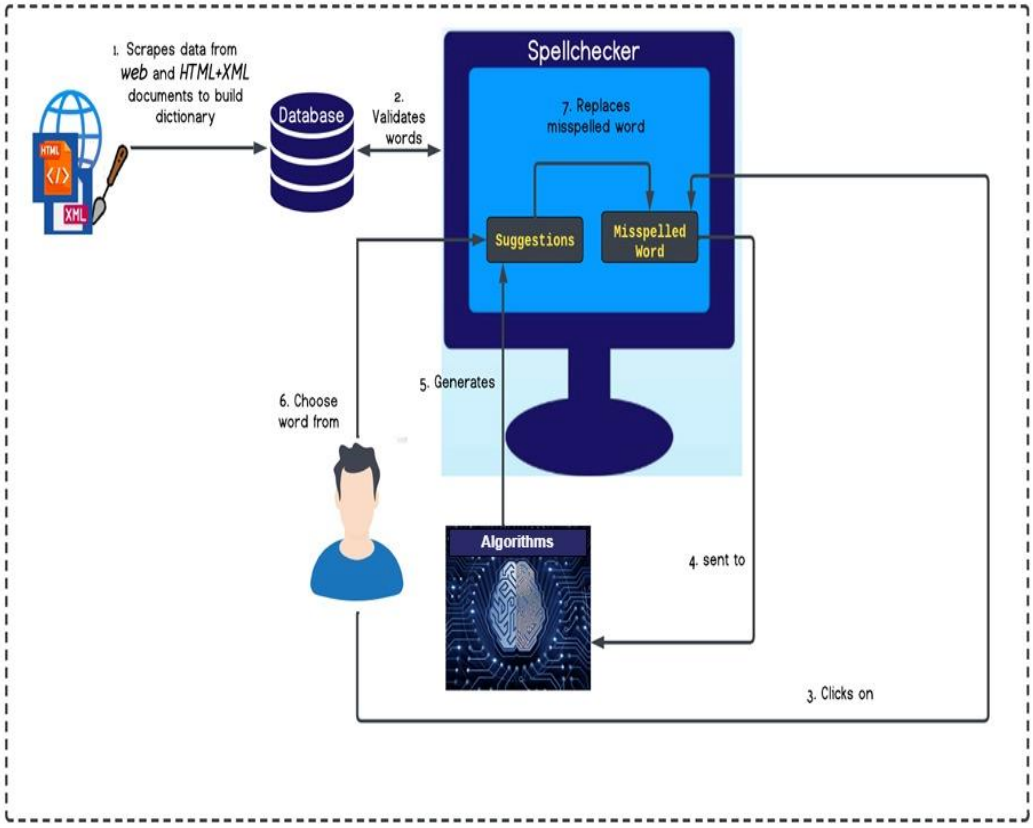


FIGURE 20: ARCHITECTURAL DESIGN

4.2.1 Dictionary Lookup Diagram

When a user inputs text into the spellchecker, the system undergoes preprocessing to analyze and cross-check the words. The spellchecker verifies whether each word exists in its dictionary. If a word is not found, it is flagged as incorrect or potentially misspelled.

Figure 21 illustrates the dictionary lookup technique:

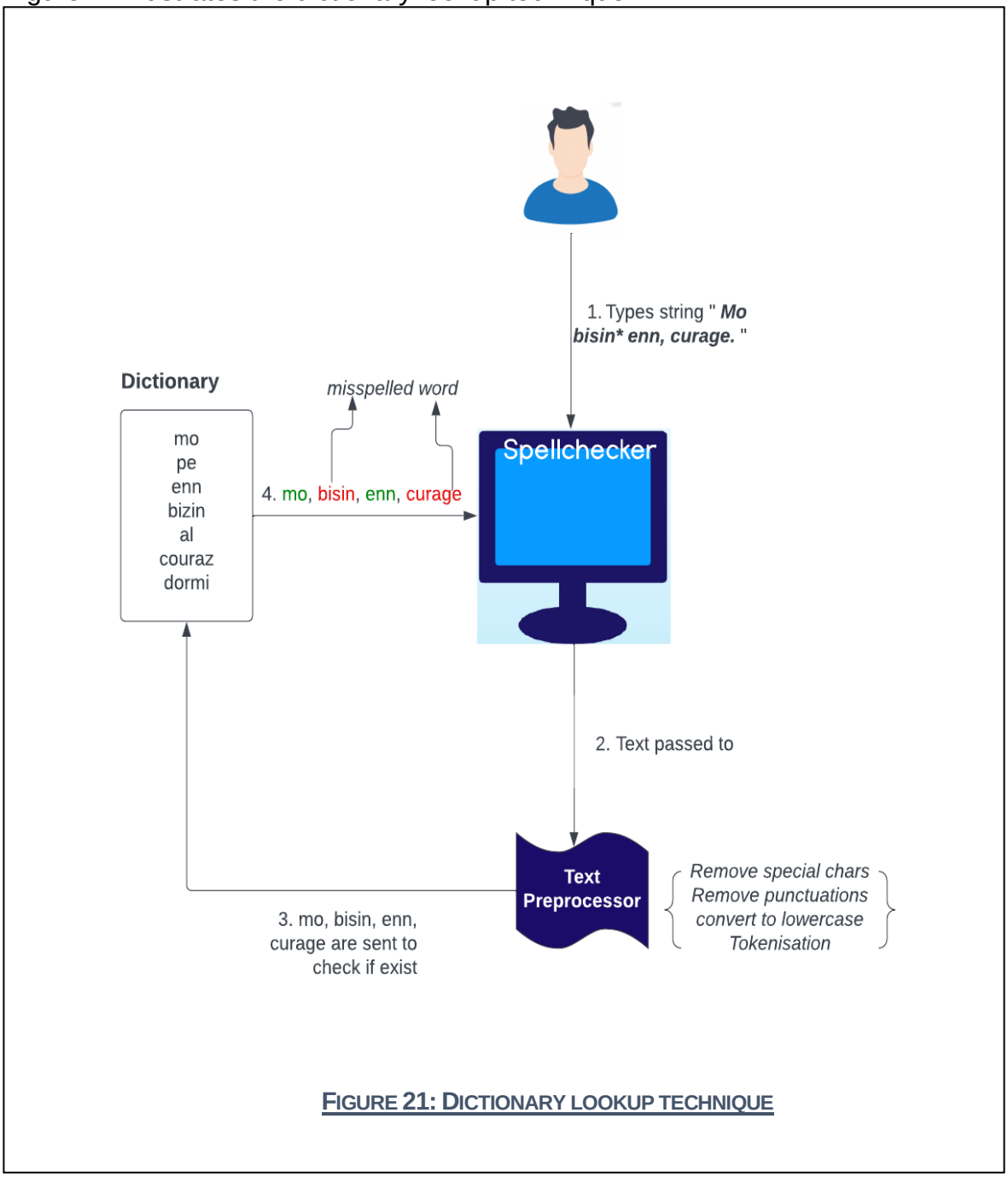


FIGURE 21: DICTIONARY LOOKUP TECHNIQUE

4.2.3 Suggestion of correction Diagram

When the system detects a misspelled word, it undergoes a series of steps to provide suggestions for correction. First, consecutive similar characters are removed then the word is passed to a mapping system that maps French to Kreol if an appropriate mapping exists. Then, the transformed word is fed into a hybrid distance function (Damerau-Levenshtein and Jaro-Winkler Similarity). The top eight candidates are returned, and they are passed to the Phonetic Algorithm, Metaphone to rank candidates with similar sounding on top of the suggestion list. After that Jaro-Winkler Similarity is again performed on the Metaphone encoding and encoding having a Jaro-Similarity score of greater than 85% are on top of the suggestion list and Bi-grams and Jaro-Winkler Similarity are applied to the rest.

Figure 22 illustrates how the correction is suggested:

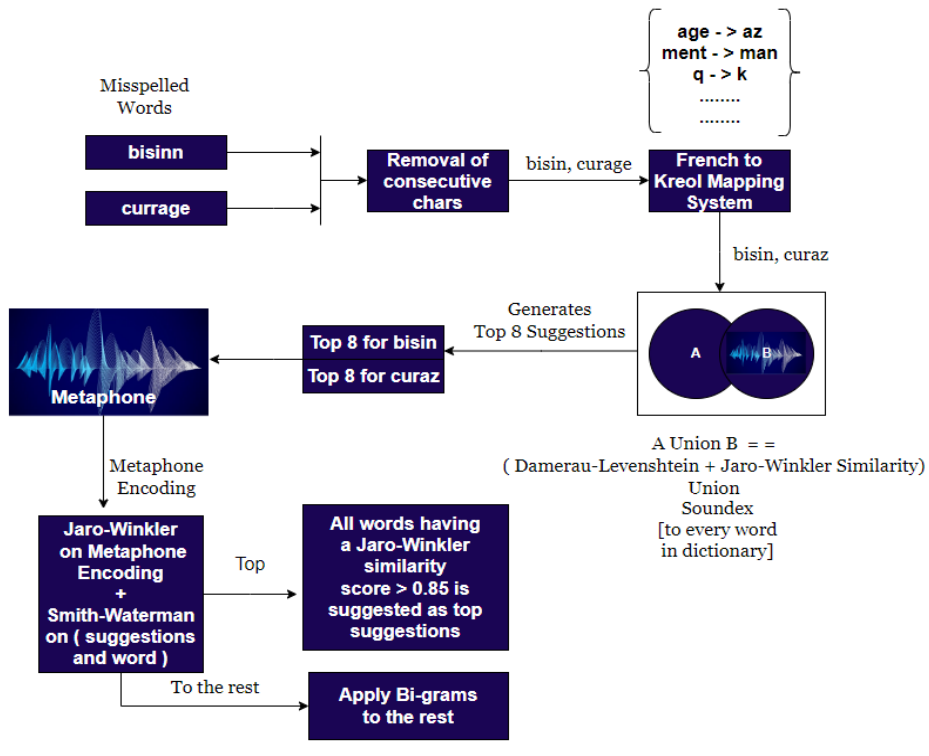


FIGURE 22: CORRECTION SUGGESTION

4.3 Class Diagram

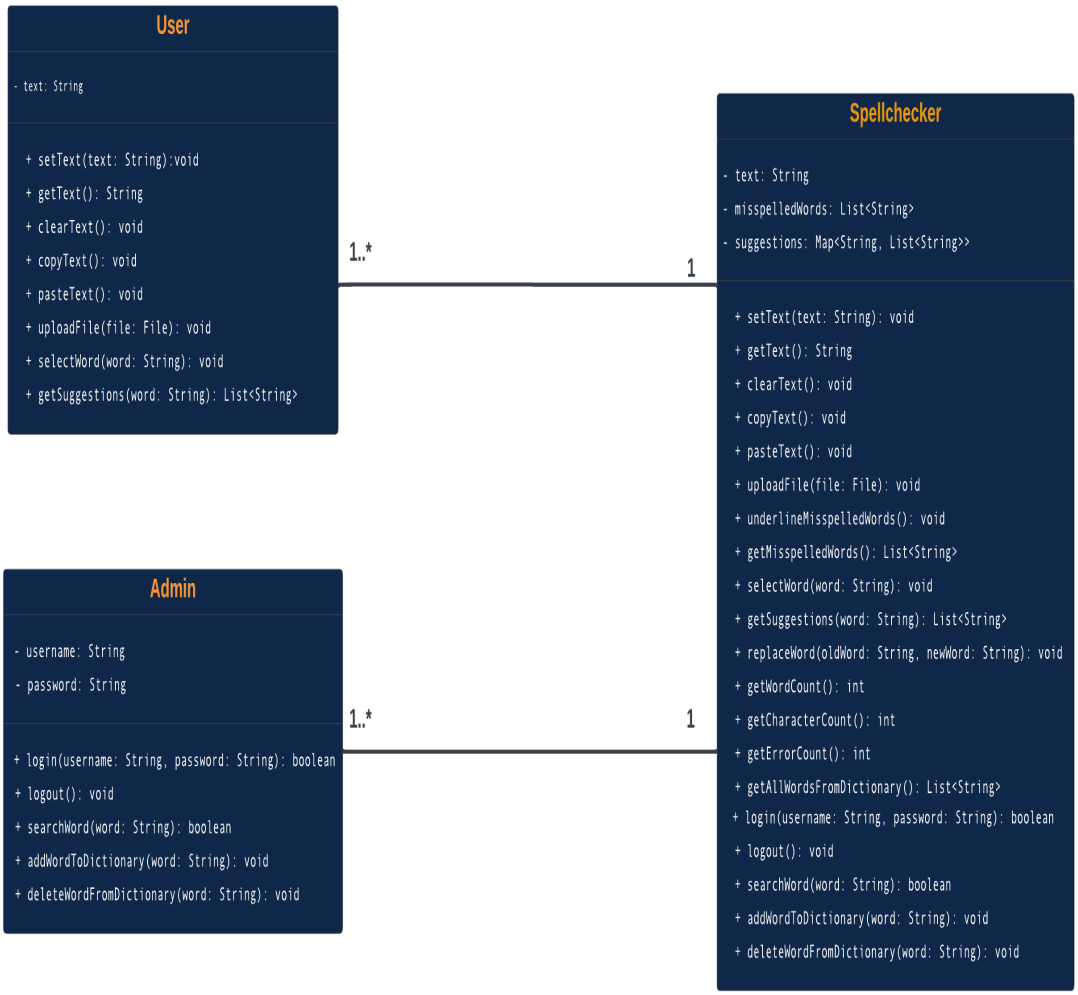


FIGURE 23: CLASS DIAGRAM

4.4 Sequence Diagram

4.4.1 Login to system to add/remove/search for word

The admin is required to log in to the system to perform actions like adding, removing, and searching for words. If the login attempt fails, the system will redirect the admin back to the login page. Once successfully logged in, the admin can access the admin panel and proceed with the desired actions. These actions may include adding words to the dictionary, removing existing words, or searching for specific words. The system will handle these operations, accordingly, ensuring that the admin is authenticated and authorized before making any changes to the dictionary. This scenario is illustrated in figure 24:

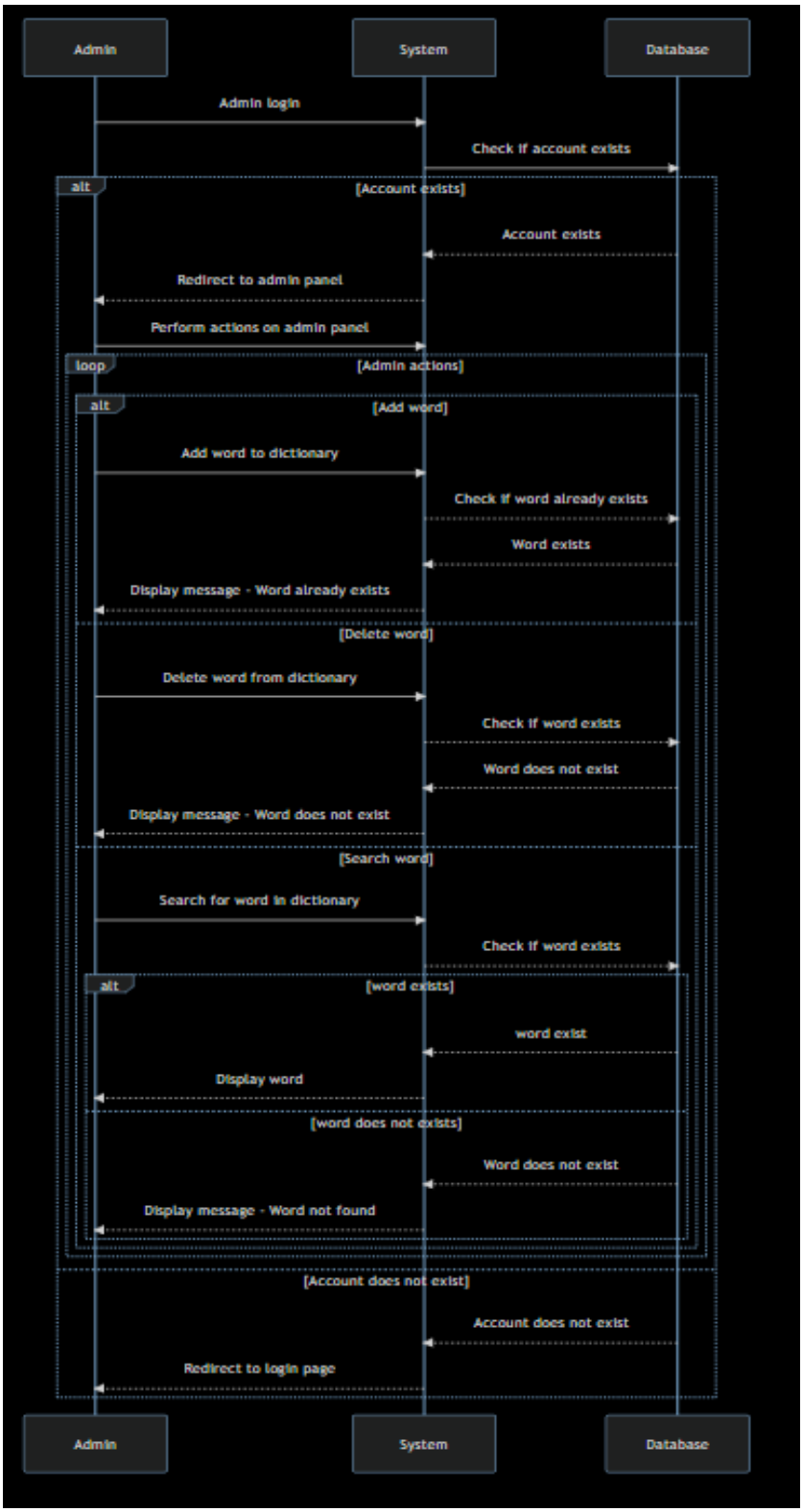


FIGURE 24: LOGIN TO ADD/REMOVE/SEARCH WORD SEQUENCE DIAGRAM

4.4.2 Spellchecking and correction

When the user types in the spell-checking area, the text is tokenized into words. Each word is then checked to see if it exists in the dictionary. If a word is found in the dictionary, no action is taken. However, if a word is not found, it is considered misspelled, and the system underlines it in red to indicate the error. When the user clicks on a misspelled word, a suggestion window pops up, providing possible suggestions to correct the incorrect word. The user can choose a word from the suggestions, and if selected, the system replaces the incorrect word with the chosen suggestion. The process is shown in figure 25:

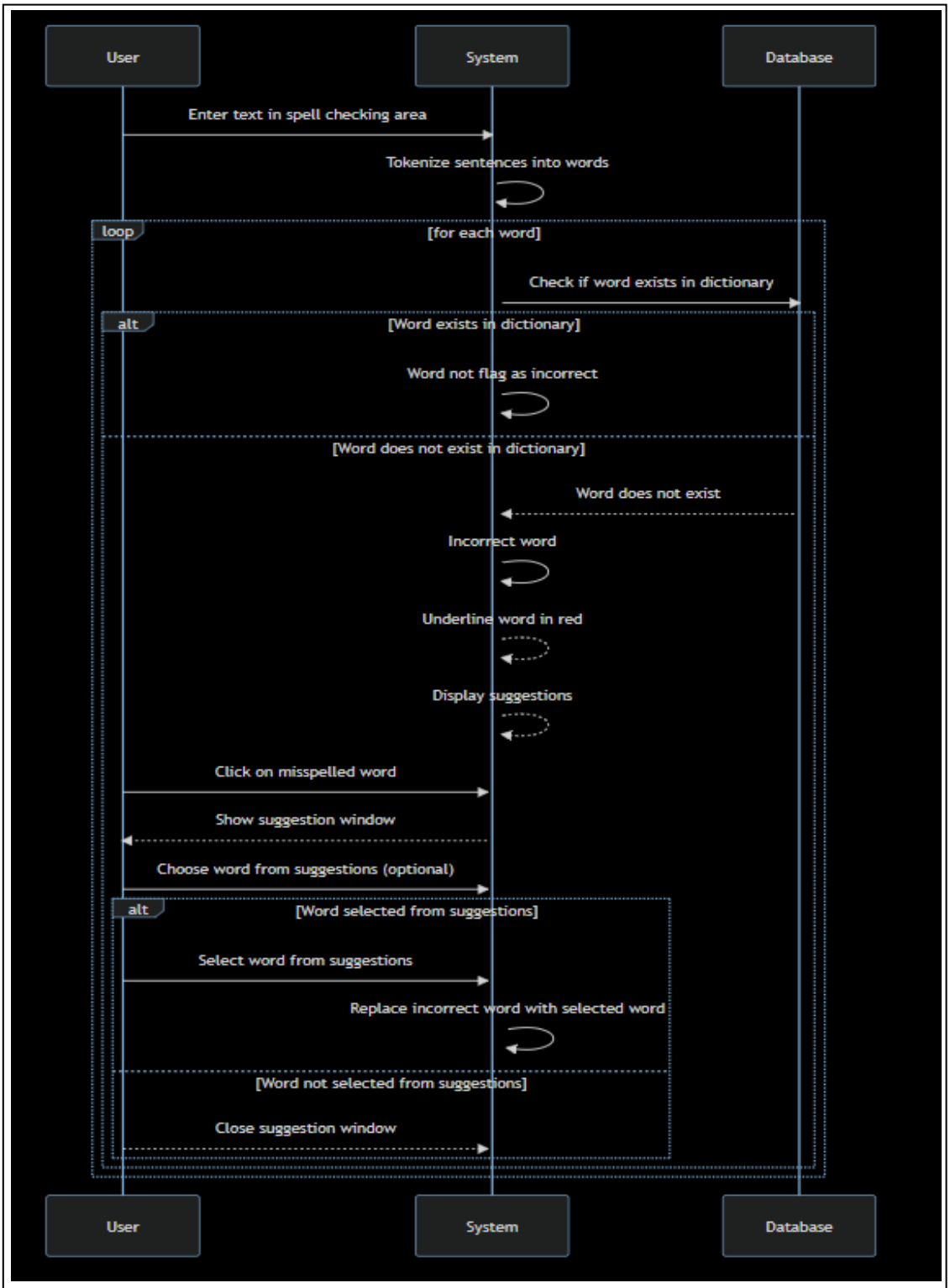


FIGURE 25: SPELL-CHECKING AND CORRECTION SEQUENCE DIAGRAM

4.5 Database Design

In our project, we have opted for MongoDB as the database of choice. This section provides an overview of the schema used in our system's collections, ensuring structured and organized data storage. The defined schema enables efficient data management and retrieval for our system.

4.5.1 Spellchecker Database Design

In this section, we show the schema of the database used for the Spellchecker.

4.5.2 ACCOUNT COLLECTION

This collection stores a group of documents that store admin credentials.

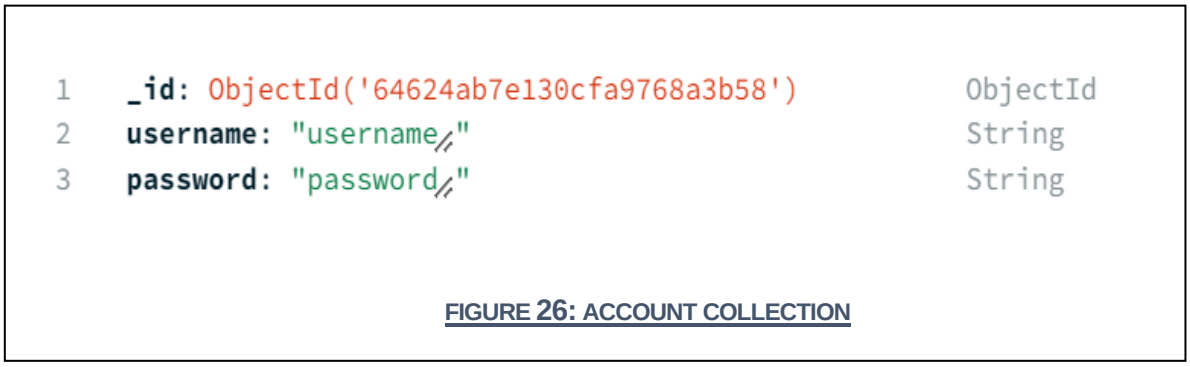


FIGURE 26: ACCOUNT COLLECTION

4.5.3 DICTIONARY COLLECTION

This collection stores a group of documents that store the valid words.

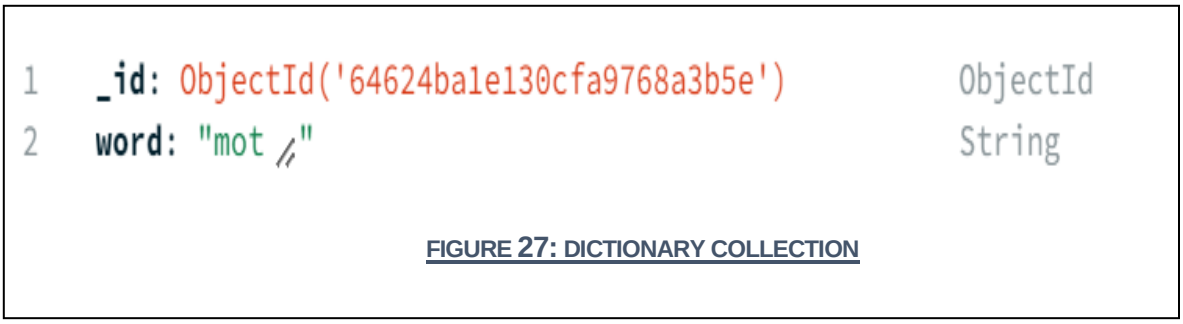


FIGURE 27: DICTIONARY COLLECTION

4.6 Interface Design

This section shows the different user interface designs for our spellchecker.

4.6.1 Spellchecker Interface Design

4.6.1.1 Login Interface

Figure 28 shows the login interface design:

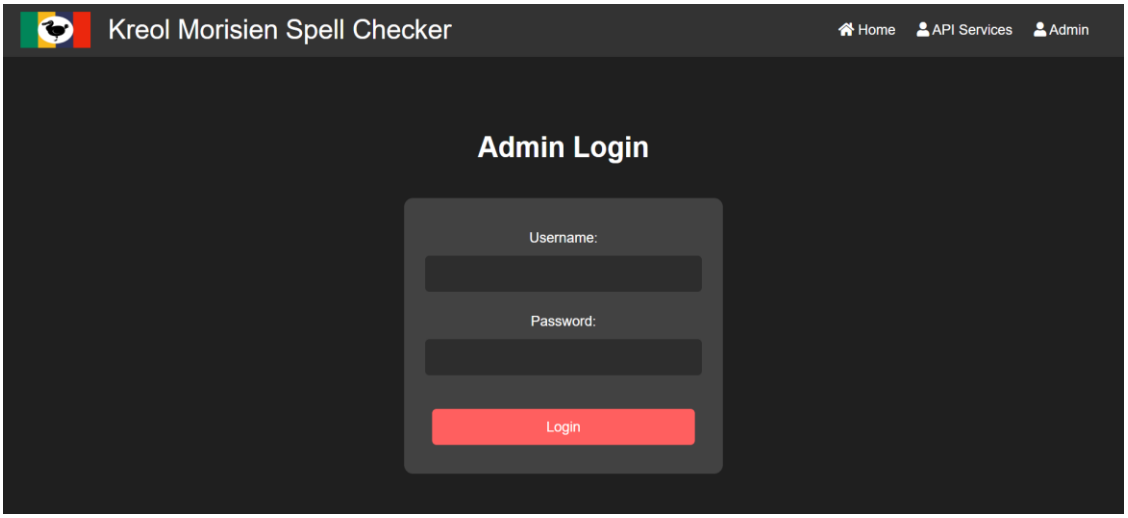


FIGURE 28: LOGIN INTERFACE

4.6.1.2 Spell checker and corrector Interface

Figure 29 shows the spellchecker and corrector interface design:

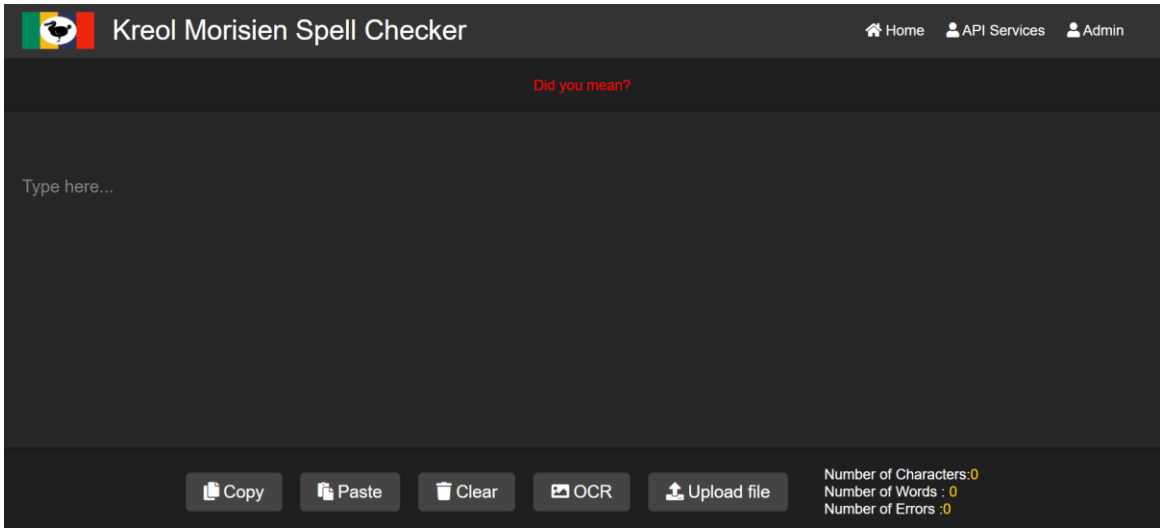


FIGURE 29: SPELLCHECKER AND CORRECTOR INTERFACE

4.6.1.3 Admin Interface

Figure 30 shows the admin panel interface design:

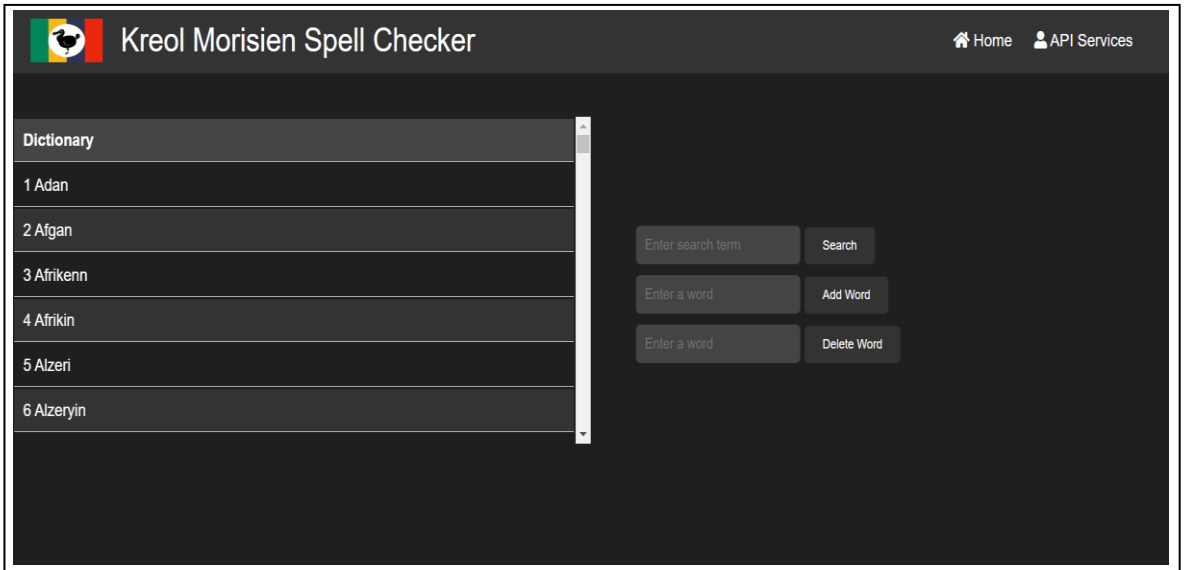


FIGURE 30: ADMIN INTERFACE

4.6.1.4 Suggestion Pop Up Screen Design

Figure 31 shows the suggestion pop up screen design:

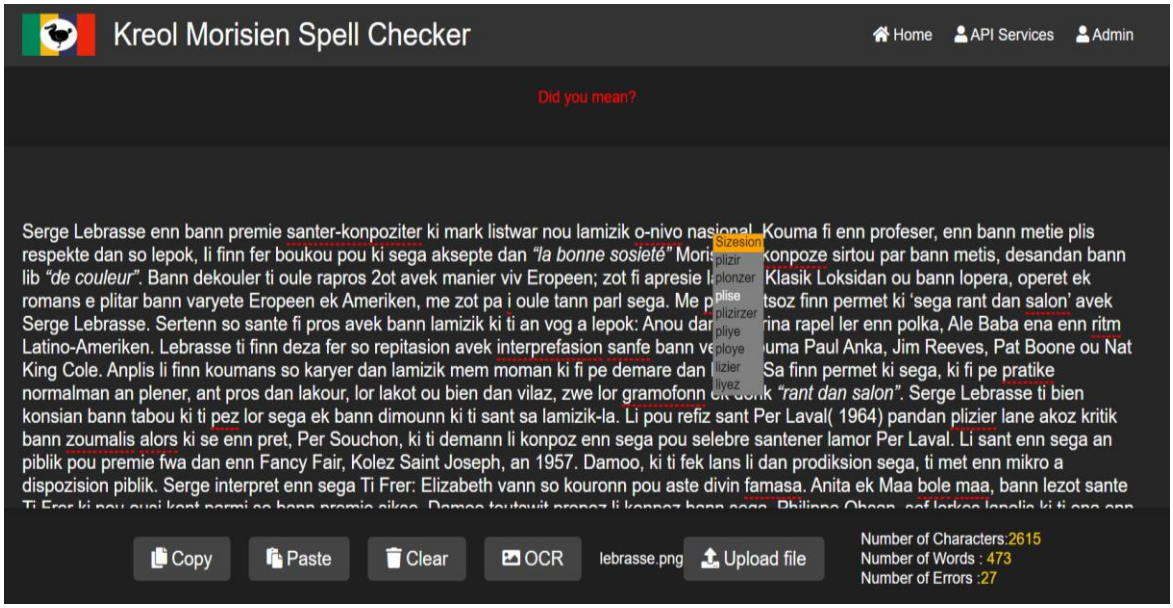


FIGURE 31: SUGGESTION POP UP SCREEN

4.6.1.5 API Services Interface

Figure 32 shows the API Services interface design:

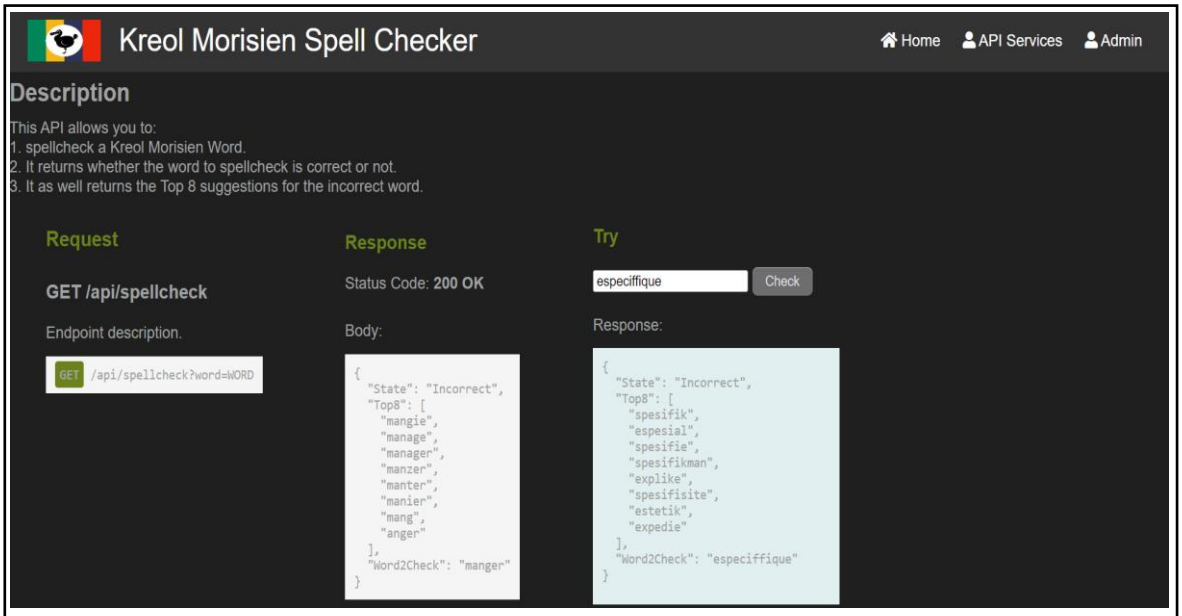


FIGURE 32: API SERVICES INTERFACE

CHAPTER 5

IMPLEMENTATION

5. Implementation

During this stage, the chosen components are used to develop the actual system. This section describes the specific steps taken to implement the software. It includes an explanation of the tools used and presents code snippets showcasing important features of our software.

5.1 Data Collection for Dictionary Building

To build the dictionary, web pages were scraped. HTML and XML documents are parsed to obtain data from specific tags. The data extraction is done using python libraries like requests and BeautifulSoup.

5.1.1 Requests library

This library in python simplifies making http requests to interact with web resources. it allows you to easily send requests using various methods like get, post, put, delete. We can retrieve web pages, download files, and work with APIs.

5.1.2 BeautifulSoup library

BeautifulSoup provides a convenient way to extract data from structured markup. With it, we can navigate, search, and access specific elements or content in the document. It is commonly used for web scraping tasks.

The code snippet for data collection is shown in figure 33:

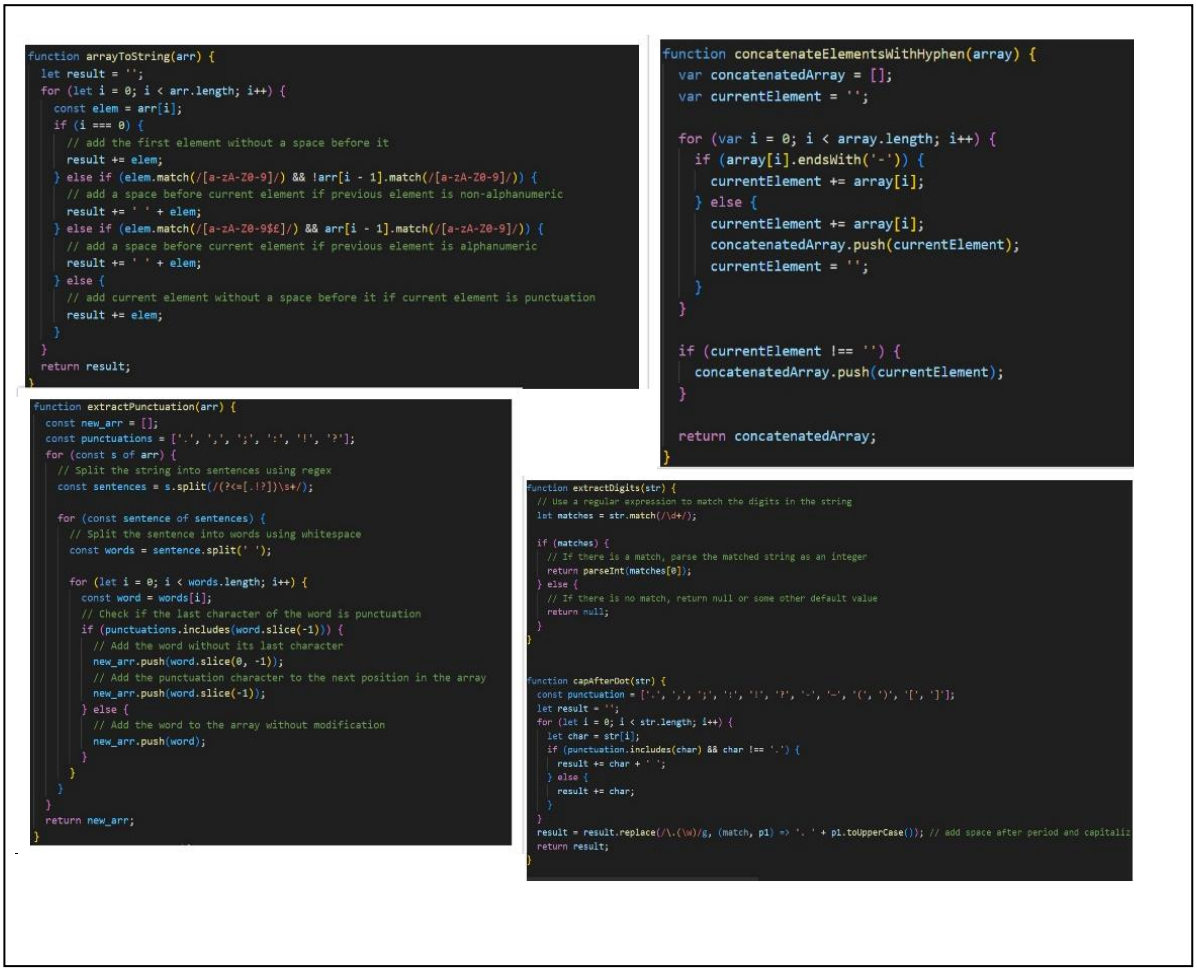
```
import requests
from bs4 import BeautifulSoup

# Send a GET request to the web page
response = requests.get("https://www.lalitmauritius.org/en/dictionary.html?letter=n")

# Create a BeautifulSoup object by passing the response text and specifying the parser
soup = BeautifulSoup(response.text, "html.parser")

# Find all the span tags with the class "main"
span_tags = soup.find_all("span", class_="main")

# Extract the values from the span tags
values = [tag.text for tag in span_tags]

print(values)
```

FIGURE 33: CODE SNIPPET FOR DATA COLLECTION

5.2 Text-Preprocessing for Dictionary Lookup

Text-preprocessing is used to make the dictionary lookup technique easier. When users type text for spell checking, it may contain punctuation, special characters, and different casing. Preprocessing helps by removing punctuation and special characters, breaking long sentences into smaller parts, and converting all words to lowercase. This ensures that uppercase and lowercase words are treated the same way during the spell-checking process.

The code snippet for text-preprocessing and dictionary lookup is shown in figure 34:



```
function arrayToString(arr) {
  let result = '';
  for (let i = 0; i < arr.length; i++) {
    const elem = arr[i];
    if (i === 0) {
      // add the first element without a space before it
      result += elem;
    } else if (elem.match(/[a-zA-Z0-9]/) && !arr[i - 1].match(/[a-zA-Z0-9]/)) {
      // add a space before current element if previous element is non-alphanumeric
      result += ' ' + elem;
    } else if (elem.match(/[a-zA-Z0-9$£]/) && arr[i - 1].match(/[a-zA-Z0-9]/)) {
      // add a space before current element if previous element is alphanumeric
      result += ' ' + elem;
    } else {
      // add current element without a space before it if current element is punctuation
      result += elem;
    }
  }
  return result;
}
```

```
function concatenateElementsWithHyphen(array) {
  var concatenatedArray = [];
  var currentElement = '';

  for (var i = 0; i < array.length; i++) {
    if (array[i].endsWith('-')) {
      currentElement += array[i];
    } else {
      currentElement += array[i];
      concatenatedArray.push(currentElement);
      currentElement = '';
    }
  }

  if (currentElement !== '') {
    concatenatedArray.push(currentElement);
  }

  return concatenatedArray;
}
```

```
function extractPunctuation(arr) {
  const new_arr = [];
  const punctuations = ['.', ',', ';', ':', '!', '?'];
  for (const s of arr) {
    // Split the string into sentences using regex
    const sentences = s.split(/(?<=[.!?])\s+/);

    for (const sentence of sentences) {
      // Split the sentence into words using whitespace
      const words = sentence.split(' ');

      for (let i = 0; i < words.length; i++) {
        const word = words[i];
        // Check if the last character of the word is punctuation
        if (punctuations.includes(word.slice(-1))) {
          // Add the word without its last character
          new_arr.push(word.slice(0, -1));
          // Add the punctuation character to the next position in the array
          new_arr.push(word.slice(-1));
        } else {
          // Add the word to the array without modification
          new_arr.push(word);
        }
      }
    }
  }
  return new_arr;
}
```

```
function extractDigits(str) {
  // Use a regular expression to match the digits in the string
  let matches = str.match(/\d+/);

  if (matches) {
    // If there is a match, parse the matched string as an integer
    return parseInt(matches[0]);
  } else {
    // If there is no match, return null or some other default value
    return null;
  }
}


function capAfterDot(str) {
  const punctuation = ['.', ',', ';', ':', '!', '?', '-', '–', '(', ')', '[', ']'];
  let result = '';
  for (let i = 0; i < str.length; i++) {
    let char = str[i];
    if (punctuation.includes(char) && char !== '.') {
      result += char + ' ';
    } else {
      result += char;
    }
  }
  result = result.replace(/\.(\w)/g, (match, p1) => '. ' + p1.toUpperCase()); // add space after period and capitaliz
  return result;
}
```

```
function splitWords(arr) {
  const words = [];
  for (const string of arr) {
    let current_word = "";
    for (const char of string) {
      if (/\s/.test(char) || /[.,?!]/.test(char)) {
        if (current_word) {
          words.push(current_word);
          current_word = "";
        }
        if (/[.,?!]/.test(char)) {
          words.push(char);
        }
      } else {
        current_word += char;
      }
    }
    if (current_word) {
      words.push(current_word);
    }
  }
  return words;
}
```

```
function removeAdjacentPunctuations(arr) {
  const cleanedArr = [];
  let prevPunct = false;  // flag to keep track of whether the previous element was a punctuation mark
  for (let i = 0; i < arr.length; i++) {
    if (['.', ',', '?', '!', ':', ';'].includes(arr[i])) {
      // if the previous element was not a punctuation mark, add the current punctuation mark
      if (!prevPunct) {
        cleanedArr.push(arr[i]);
      }
      prevPunct = true;
    } else {
      cleanedArr.push(arr[i]);
      prevPunct = false;
    }
  }
  return cleanedArr;
}
```

```
function removeStartingPunctuations(s) {
  let i = 0;
  while (i < s.length && [',', '.', '?', '!', ':', ';'].includes(s[i])) {
    i++;
  }
  return s.slice(i);
}
```

```
function removeSpaceAfterHyphen(sentence) {
  return sentence.replace(/-\s/g, '-');
}


function findPositions(array1, array2) {
  const positions = [];
  for (let i = 0; i < array2.length; i++) {
    const element = array2[i];
    const elementwithoutla = element.replace(/-la/g, "");
    if (array1.includes(element.toLowerCase()) === false && array1.includes(elementwithoutla.toLowerCase()) === fals
      if (element.charAt(0) !== element.charAt(0).toUpperCase()) {
        positions.push(i);
      }

    }
  }
  return positions;
}
function underlineElements(array1, array2) {
  for (let i = 0; i < array2.length; i++) {
    if (array2[i] < array1.length) {
      if (isNaN(array1[array2[i]])) {
        array1[array2[i]] = "<span class='suggestclass'  id='error" + i + "' style='text-decoration-line: underline;
      }
    }
  }
  return array1;
}
```

```
function splitSentence(sentence) {
  // Split the sentence into words by space
  const words = sentence.split(' ');

  // Return the list of words
  return words;
}

function removeStartingPunctuations(s) {
  let i = 0;
  while (i < s.length && [',', '.', '?', '!', ':', ';'].includes(s[i])) {
    i++;
  }
  return s.slice(i);
}
```

FIGURE 34: CODE SNIPPETS FOR PREPROCESSING IN JAVASCRIPT

5.3 Solution for Inaccurate Word

5.3.1 Distance Algorithms

Distance algorithms are used to retrieve closest word from dictionary to incorrect word. When single distance algorithm is used, the accuracy of suggestions is not that accurate.

A hybrid approach is adopted that uses a combination of distance algorithms and it is found that it is returning mostly accurate result.

Figure 35 shows the code snippet for the hybrid approach:

```
def custom_distance(word1, word2):
    # Calculate the distances using different algorithms
    dlev_distance = distance.edit_distance(word1, word2, transpositions=True)
    jw_similarity = distance.jaro_winkler_similarity(word1, word2)

    # Combine the distances using weights
    distance_sum = dlev_distance + (1 - jw_similarity)
    return distance_sum
```

FIGURE 35: CODE SNIPPET FOR HYBRID APPROACH FOR DISTANCE ALGORITHMS

5.3.2 Prioritizing Distance Algorithms on Metaphone Encoding for Improved Results

When we applied only Jaro-Winkler Similarity on the Metaphone encoding to prioritize more accurate word in the suggestion list, but it happens that there may me more than word having same score, so Smith-Waterman algorithm as well is applied to resolve this issue.

Figure 36 shows the code snippet for the implementation of Jaro-Winkler together with Smith-Waterman on the Metaphone encodings:

```
def metaPhone(word, array):
    metacode_start, metacode_end = doublemetaphone(word)
    print(doublemetaphone(word))
    arr = array
    arrmeta = [(element, doublemetaphone(element)) for element in arr]

    matchingMeta = [element for element, metaphone in arrmeta if jellyfish.jaro_winkler_similarity(
        metaphone[0], metacode_start) > 0]

    sortedMeta = sorted(matchingMeta, key=lambda x: (jellyfish.jaro_winkler_similarity(
        doublemetaphone(x)[0], metacode_start), smith_waterman_similarity(x, word)), reverse=True)

    for word in sortedMeta:
        metaphone = doublemetaphone(word)
        jaro_start_similarity = jellyfish.jaro_winkler_similarity(
            metaphone[0], metacode_start)
        smith_waterman_score = smith_waterman_similarity(word, word)
        print(f"Word: {word}, Metaphone: {metaphone}, \
              Jaro-Winkler Similarity (Start): {jaro_start_similarity},\
                Smith-Waterman Score: {smith_waterman_score}")

    return sortedMeta
```

FIGURE 36: CODE SNIPPET FOR JARO-WINKLER & SMITH-WATERMAN

5.3.3 Phonetic Algorithms

Using the edit distance algorithms on their own is covering only similarity in writing but not in pronunciation. To solve this issue, we have implemented a technique that covers both, so we have done a *union* like in sets between Damerau-Levenshtein and Soundex to get best results of incorrect word.

Below is a code snippet which illustrates the implementation of Soundex:

```python
def soundex(word):
    # Map each letter to its Soundex digit
    soundex_mapping = {
        "b": "1", "f": "1", "p": "1", "v": "1",
        "c": "2", "g": "2", "j": "2", "k": "2", "q": "2", "s": "2", "x": "2", "z": "2",
        "d": "3", "t": "3",
        "l": "4",
        "m": "5", "n": "5",
        "r": "6"
    }

    # Convert the word to lowercase
    word = word.lower()

    # Remove non-alphabetic characters
    word = ''.join(c for c in word if c.isalpha())

    if not word:
        return ""

    # Convert the first letter to uppercase
    soundex_code = word[0].upper()

    # Encode the remaining letters using Soundex mapping
    for i in range(1, len(word)):
        if word[i] in soundex_mapping:
            digit = soundex_mapping[word[i]]
            if digit != soundex_code[-1]:
                soundex_code += digit

    # Pad the Soundex code to a length of 4
    soundex_code += "000"
    soundex_code = soundex_code[:4]

    return soundex_code


def find_matching_soundex(word, array):
    word_soundex = soundex(word)

    matching_words = []
    for item in array:
        if soundex(item)  == word_soundex :
            matching_words.append(item)

    return matching_words
```

FIGURE 37: CODE SNIPPET FOR SOUNDEX

5.4 Mapping System to stem French to Kreol Morisien

During the study of this project, it is found that majority of French and Kreol Morisien words are closely related. French and KM words usually have same sounding, so a mapping system is used to manipulate French words to KM words and after this mapping system distance and phonetic is applied and this has improved the accuracy of suggestions if users have typed a French word instead of a KM word because of same sounding.

For example:

Incorrect word: '*communique*'

Correct word: '*kominike*'

So, without mapping system 'communique' and 'kominike' differ a lot when writing despite they have closely same sounding, so system will not be able to suggest correct word 'kominike'.

Table 25 shows all the mappings (to note that the mapping may not always be accurate in some cases):

TABLE 25: MAPPING SYSTEM

Word	Replace With	List of words
é	e	
è	e	
ê	e	
à	a	
â	a	
ô	o	
û	u	
ç	c	
qui	ki	ki, kokin, rekin, lekip
que	k	piknik, astronomik, siklonik, atmosferik, akademik
q	k	ekipe, kesion, kat, alfabetik
j	z	labouzi, bizou, zournal, zour
aign	eny	zarenie
tion/tyon	sion	adision, kalkilasion, abolision,absorpsion,adiksion
ends with 'yon'	ion	avion, pavion, espion, lion, bilion
ends with 'eux'	e	de, ere, kouraze, seve, kirye
ends with 'end'	an	pran, apran, atan, depan, desan, sispan
pu	pou	dalpouri, epouvantab, gropoumon, karipoule,lanpoul
ends with 'eur'	er	anplwayer, freyer, gaspiyer, inferyer, travayer
iel	yel	aktiel, arkansiel, artifisiel, diferansiel, dimiel
troi	trwa	detrwa, trwakar, trwazer, trwaziem, etrwa
toi	twa	depotwar, zistwar, viktwar, twalet, teritwar

moi	mwa	kerdemwazel, lamwatie, larmwar, lemwayin, mademwazel
cy	si	bisiklet, siklik,siklis, siklonn, resiklaz
clet	klet	motosiklet, raklet, bisiklet
huil	wil	delwil
ail	ay	batay, ray, fiansay, mitrayez
age	az	reglaz, resiklaz, anbouteyaz, garaz, feyaz, amenazman
cul	kil	agrikiltir, kiltir, agrikilter, akwakiltir, artikilasion
cal	kal	kalkil, kalkilatris, amikal, bokal, kalamite
mau	mo	moris, zanimo, move, mov
phar	far	far, faraon, farmasi
ch	s	seve, seval, akouse, aprose, simik
onc	onk	fonksion, konklision, konkour
cir	sir	sirkonskripsion, sirkilasion, sirkonferans, sirkonstans, sirkwi
ance	ans	kanser, avanse, sans, fianse,balans
plu	pli	plim, plis, lapli, plise, parapli
phy	fi	fisik, fizisien, klorofil, kaligrafi,fiziolozi
psy	si	sikolozi, sikiat
ment	man	aktivman, dousman, egalman, ekipman, exakteman
ace	as	fas, glase, danse, agase, brasle
oir	war	epilatwar, fitwar, koulwar, laboratwar, lafwar

ence	ans	absans, odians, aparans, komans, diferans
euse	ez	dansez, gazez, nazez, amourez, kwafez
eau	o	lapo, kado, sato, bato, samo
aire	er	fer, kler, boner, orer, banker
iste	is	lis, pis, existe, asiste, artis
istre	is	minis, rezis, anrezistre
ends with 'ise'	iz	aktializ, anglez, otorize, santralize
ends with 'ice'	is	vis, zepis, aktris, zistis, lapolis
ends with 'eil'	ey	boutey, akey, anboutey, konsey
ends with 'aitre'	et	met, tret, paret, rekonet, aparet
coin	kwin	kwin, kwinsidans
oix	wa	pwa, swa, lakrwaze, swasant
able	ab	fiab, diab, zetab, potab,tablo
ends with 'gile'	zile	zile, frazil
ends with 'igue'	ige	rodrige, lalige, fatige
then	tan	otantik
aut	ot	otantik, astronot, kominote, sote
aid	ed	ed, led, red
huit	wit	wit, zwit
emp	anp	tanp,anpann, anplasman, anplwaye
oue	we	zwe, fwete
ier	yer	yer, anplwayer, baryer, deryer
auce	os	lasos, sosblans, sosializ, sosiolozi, sospikant
cui	kwi	lakwis, lakwizinn, kwiyer, kwiv
aune	onn	zonn, fonn, dezabonn, exagonn
aud	o	aplodi, odasite, odio, so, odians

aine	enn	lasenn, grenn, afrikenn, abdomenn
auvre	ov	pov, sov
auve	ov	sovsouri, sovtaz, sovgard, mov
sul	zil	conziltasion, mizilman, rezilta
riene	ryenn	sezaryenn, veneryenn
uge	iz	ziz, refiz, santrifiz
syer	sir	blesir, brosir, koursirkwi, krosir
soi	swa	swazir, aswafe, aswar
starts with 'con'	kon	koneksion, konferans,konfizion, konform, konkiran
starts with 'frais'	fre	fre, freser, frez, freyer
oo	ou	trou, fou, mou, mouswar, bizou
ract	rak	trakter, fraktir, abstrak, fraksion, atraksion
ends with 'nyin'	nien	tanzanien, armenien, ikranien, iranien
ends with 'ais'	e	fre, franse, abese, angle, rezon
ends with 'ait'	et	tret, parfe, swete, reprezante, disparet
starts with 'super/ souper'	siper	Sipermarse, siperfisi, siperfisiel, siperpwisans, siperyer
ends with 'rium'	ryom	akwaryom, oditoryom, antiryom, deleryom
ends with 'ture'	tir	faktir, fraktir, avantir, fournitir, fitir
ends with 'dure'	dir	prosedir, prodir, ordir, dire
ends with 'sible'	zib	plozib, vizib
starts with 'du'	di	disik, disel, dimiel, divin

Figure 38 shows the code snippet that illustrates the mappings and some rules (only some mappings are listed here):

```python
def replace_word(word):
    # Define all the replacements as a dictionary
    replacements = {
        'aign': 'en',
        'é': 'e',
        'è': 'e',
        'ê': 'e',
        'à': 'a',
        'â': 'a',
        'ô': 'o',
        'û': 'u'
    }

    # Iterate over all the keys in the replacements dictionary and replace the occurrences of each key with its value
    for key in replacements:
        if key in word:
            word = word.replace(key, replacements[key])

    # Return the modified word
    return word
```

```python
    value = value.replace("oo", "ou")
    value = value.replace("com", "kom")
    if value.startswith("con"):
        value = "kon" + value[3:]
    if value.startswith("frais"):
        value = "fre" + value[3:]
    if value.startswith("super"):
        value = "siper" + value[5:]
    if value.startswith("souper"):
        value = "siper" + value[5:]
    if value.endswith("ait"):
        value = value[:-3] + "et"
    if value.endswith("ais"):
        value = value[:-3] + "e"
    if value.endswith("nyin"):
        value = value[:-4] + "nien"
    if value.endswith("ture"):
        value = value[:-4] + "tir"
    if value.endswith("rium"):
        value = value[:-4] + "ryom"
```

FIGURE 38: CODE SNIPPETS FOR MAPPING SYSTEM

5.5 Maintaining Caret Position

The spellchecker is a real-time web-based application that replaces text as the user types, ensuring a seamless user experience. To achieve this, the application captures the cursor position each time the user types a character, replacing the text in real-time without causing any disruption or interference to the user's typing. This allows the spellchecker to update the text accurately and dynamically without the user noticing or being affected by the ongoing corrections.

Figure 39 shows the code snippet of how to keep track of caret position:

```
! function(e, t) {
    if ("function" == typeof define && define.amd) define("VanillaCaret", ["module"], t);
    else if ("undefined" != typeof exports) t(module);
    else {
        var n = {
            exports: {}
        };
        t(n), e.VanillaCaret = n.exports
    }
}(this, function(e) {
    "use strict";

    function t(e, t) {
        if (!(e instanceof t)) throw new TypeError("Cannot call a class as a function")
    }
    var n = function() {
            function e(e, t) {
                for (var n = 0; n < t.length; n++) {
                    var o = t[n];
                    o.enumerable = o.enumerable || !1, o.configurable = !0, "value" in o && (o.writable = !0), Object.defineProperty(e, o.key, o)
                }
            }
            return function(t, n, o) {
                return n && e(t.prototype, n), o && e(t, o), t
            }
        }(),
        o = function() {
            function e(n) {
                t(this, e), this.target = n, this.isContentEditable = n && n.contentEditable
            }
            return n(e, [{
                key: "getPos",
                value: function() {
                    if (document.activeElement !== this.target) return -1;
                    if ("true" === this.isContentEditable) {
                        this.target.focus();
                        var e = document.getSelection().getRangeAt(0),
                            t = e.cloneRange();
                        return t.selectNodeContents(this.target), t.setEnd(e.endContainer, e.endOffset), t.toString().length
                    }
                    return this.target.selectionStart
                }
            }, {
                key: "setPos",
                value: function(e) {
                    if ("true" === this.isContentEditable) {
                        if (e >= 0) {
                            var t = window.getSelection(),
                                n = this.createRange(this.target, {
                                    count: e
                                });
                            n && (n.collapse(!1), t.removeAllRanges(), t.addRange(n))
                        }
                    } else this.target.setSelectionRange(e, e)
                }
            }, {
                key: "createRange",
                value: function(e, t, n) {
                    if (n || ((n = document.createRange()).selectNode(e), n.setStart(e, 0)), 0 === t.count) n.setEnd(e, t.count);
                    else if (e && t.count > 0)
                        if (e.nodeType === Node.TEXT_NODE) e.textContent.length < t.count ? t.count -= e.textContent.length : (n.setEnd(e, t.count), t.count = 0);
                        else
                            for (var o = 0; o < e.childNodes.length && (n = this.createRange(e.childNodes[o], t, n), 0 !== t.count); o++);
                    return n
                }
            }]), e
        }();
    e.exports = o
});
```

FIGURE 39: CODE SNIPPET FOR MAINTAINING CARET POSITION

5.6 Spellchecking and correction suggestions API

An API is exposed that facilitates the detection of word accuracy and generation of the top eight suggestions for a given word.

```python
@app.route('/api/spellcheck', methods=['GET'])
def apispellcheck():
    user_query = str(request.args.get('word'))  # /api/spellcheck?word=WORD
    user_query = user_query.lower()

    # Assuming `processVal()` returns the response object
    response = processVal(user_query)

    json_data = response.data  # Retrieve the response content as bytes
    json_data_str = json_data.decode('utf-8')  # Decode the bytes to a string

    data = json.loads(json_data_str)  # Parse the JSON string

    words = data["words"]  # Access the "words" field
    state = ""
    if user_query.lower() in words:
        state = 'Correct'
    else:
        state = 'Incorrect'

    data_set = {'Word2Check': user_query, 'State': state, 'Top8': words}
    return jsonify(data_set)
```

FIGURE 40: CODE SNIPPET FOR SPELLCHECKING & CORRECTION API

CHAPTER 6

TESTING AND EVALUATION

6. Testing and Evaluation

In order to ensure thorough testing, we conducted both black box and white box testing on the spellchecker. These testing methods allowed us to assess the functionality and correctness of the spellchecker from different perspectives. The combination of these evaluation metrics and testing approaches provided a comprehensive analysis of the spellchecker's effectiveness and efficiency in detecting and correcting errors in Kreol Morisien writing. Next, to evaluate the performance of our spellchecker, we employed several evaluation metrics, including true positives, false positives, true negatives, and true positives, which were captured in a confusion matrix. In addition, we used accuracy, recall, precision, and F1 score as evaluation measures. Furthermore, we implemented Mean Reciprocal Rank (MRR) as an additional metric.

6.1 Black Box Testing

Black box testing assesses the application's behavior solely by examining its external characteristics, without delving into the internal structure or implementation details of the system.

6.1.1 Test Case 1: Authentication to access Admin Panel

TABLE 26: TEST CASE 1

Objective	Correct login details entered
Test Data	Username and Password
Expected Result	The system should verify the user's credentials against the stored database records. If the credentials match, the user should be redirected to the admin panel. In case of incorrect credentials, the system should display an appropriate error message.

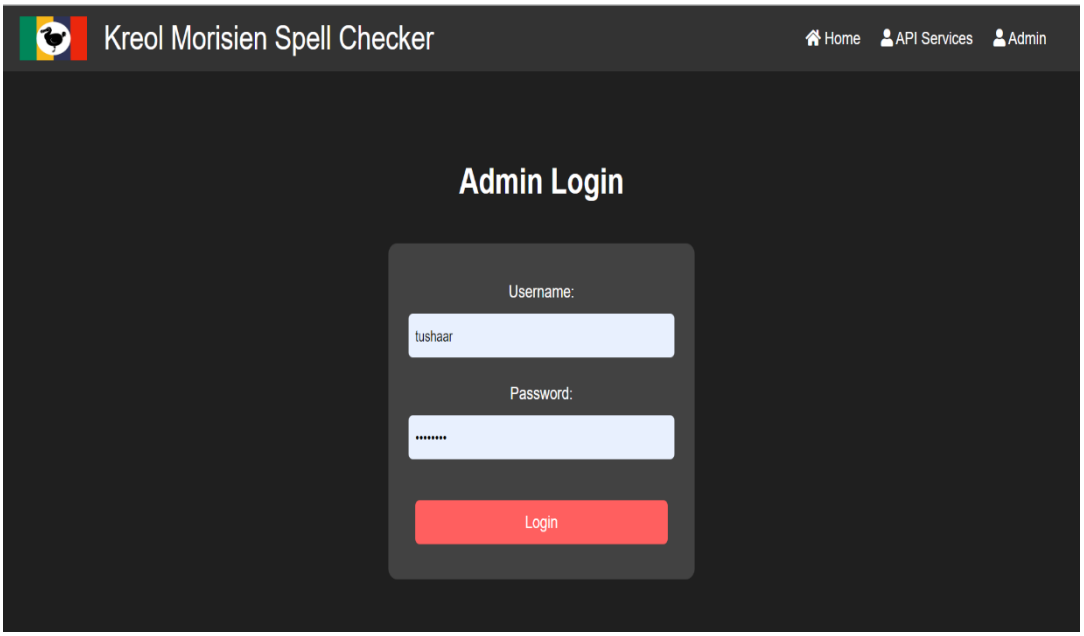


FIGURE 41: INCORRECT LOGIN CREDENTIALS

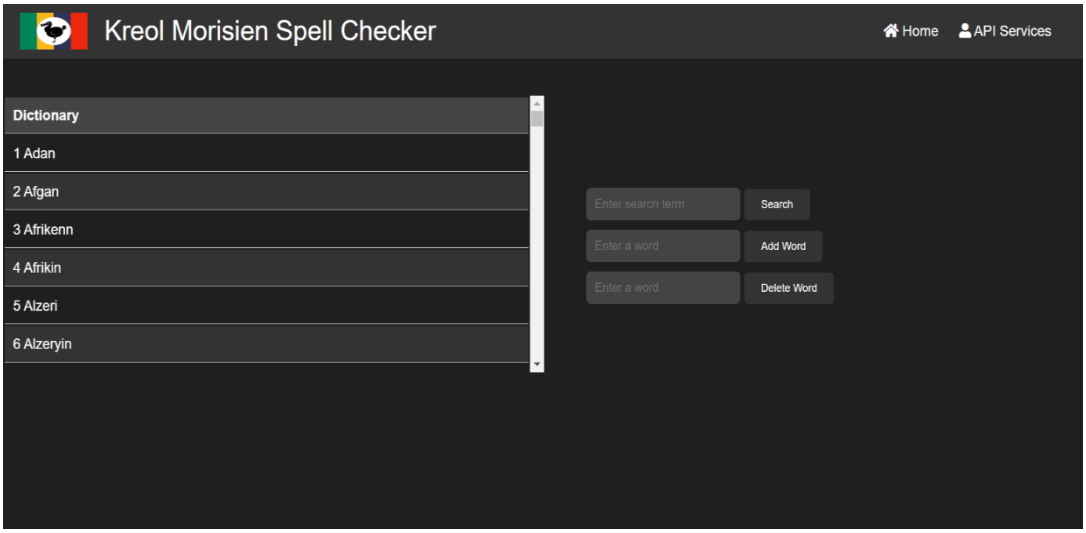


FIGURE 42: SUCCESSFUL LOGIN LEADS TO ADMIN PANEL

6.1.2 Test Case 2: Detection of misspelled Words

TABLE 27: TEST CASE 2

Objective	Detect incorrect words.
Test Data	Text entered in the spellcheck area.
Expected Result	The system should underline any misspelled word detected in red.

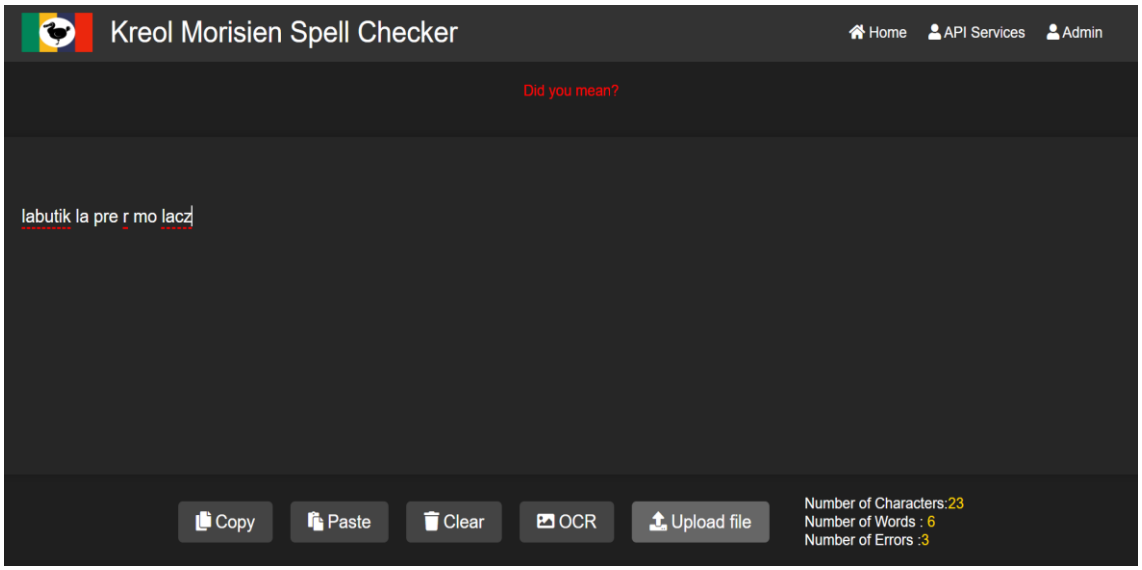


FIGURE 43: MISSPELLED WORD DETECTED SUCCESSFULLY IN RED

6.1.3 Test Case 3: Suggestion Pop up on Clicking Incorrect Word

TABLE 28: TEST CASE 3

Objective	Provide suggestions of misspellings.
Test Data	Misspelled word.
Expected Result	A suggestion window should appear to allow user to see correct suggestions of incorrect word.

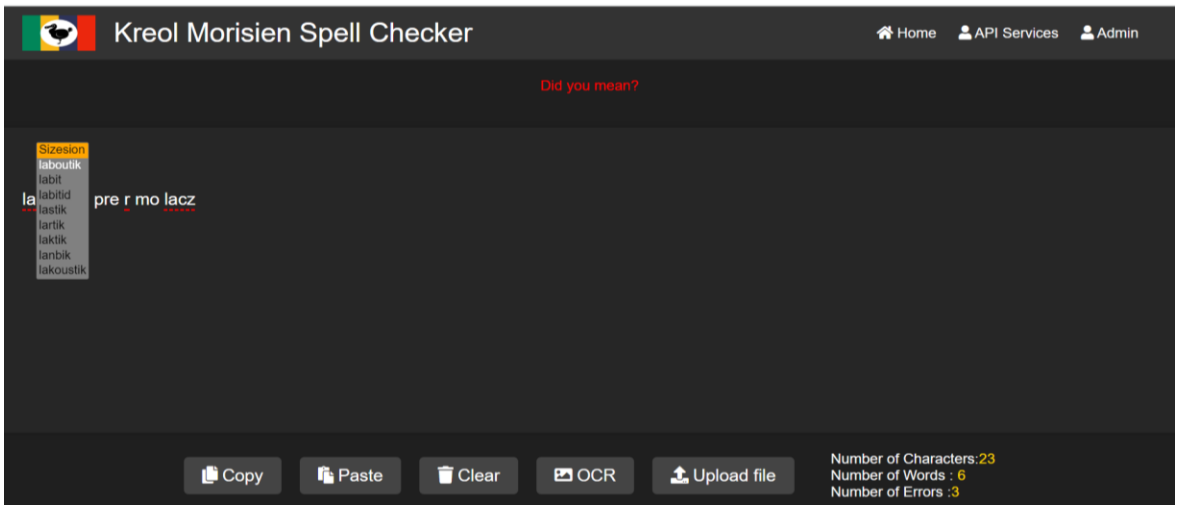


FIGURE 44: SUGGESTION POP UP

6.1.4 Test Case 4: Replace Incorrect Word on Selecting a Word from Suggestion List

TABLE 29: TEST CASE 4

Objective	Replace misspelled word on choosing a correct value from suggestions.
Test Data	Correct word from suggestions.
Expected Result	Respective incorrect words should be replaced upon selecting a correct value from suggestion list.

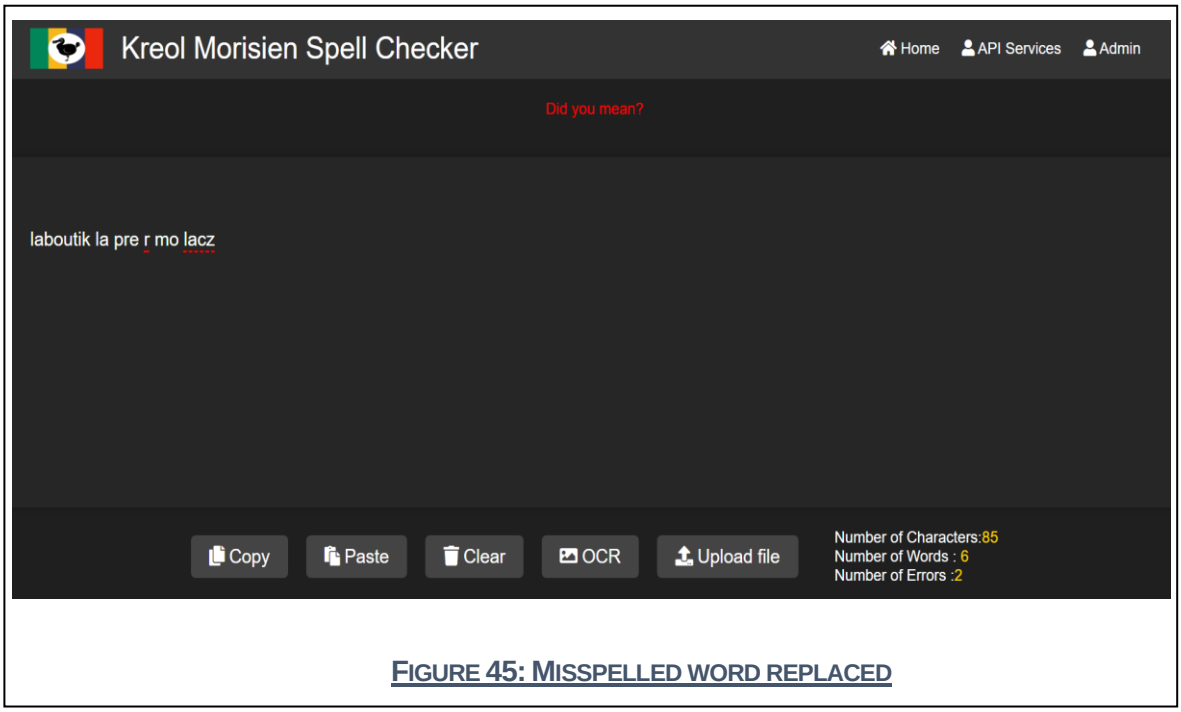


FIGURE 45: MISSPELLED WORD REPLACED

6.1.5 Test Case 5: Upload File to Check for Misspellings

TABLE 30: TEST CASE 5

Objective	Upload File to check misspellings and do correction.
Test Data	File content.
Expected Result	When file is uploaded the content of the file is displayed on the spellcheck area to allow user to see the misspellings in the file and correct them.

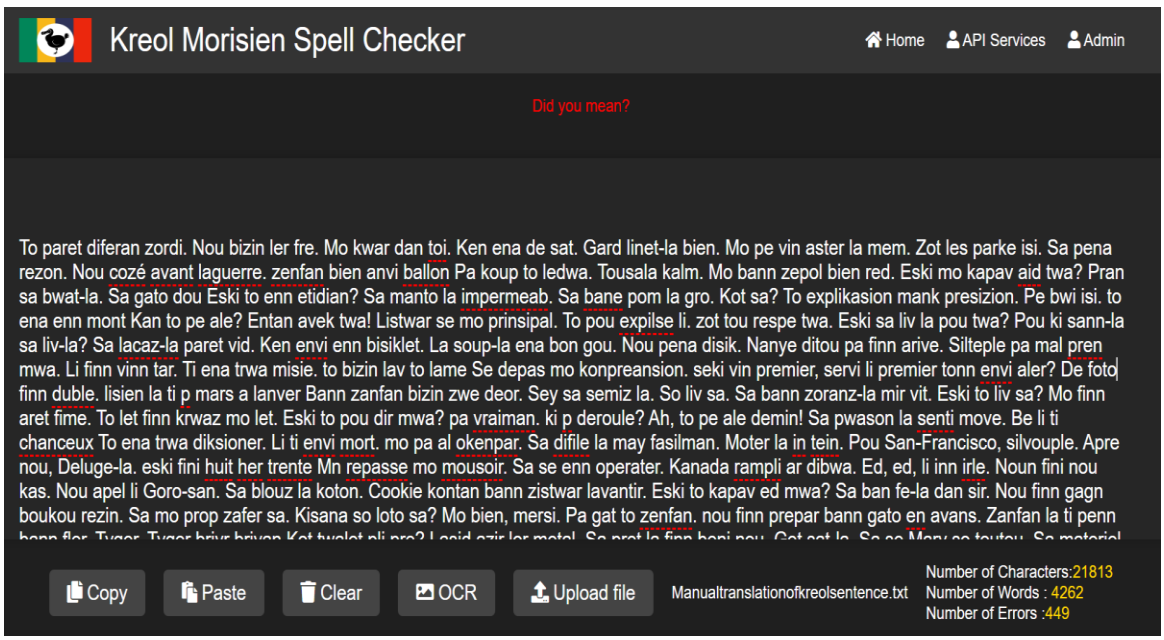


FIGURE 46: UPLOAD FILE TO CHECK MISSPELLINGS

6.1.6 Test Case 6: Upload Image File to Check for Misspellings

TABLE 31: TEST CASE 6

Objective	Upload File to check misspellings and do correction.
Test Data	Text in image
Expected Result	When image file is uploaded the text detected in the file is displayed on the spellcheck area to allow user to see the misspellings in the image file and correct them.

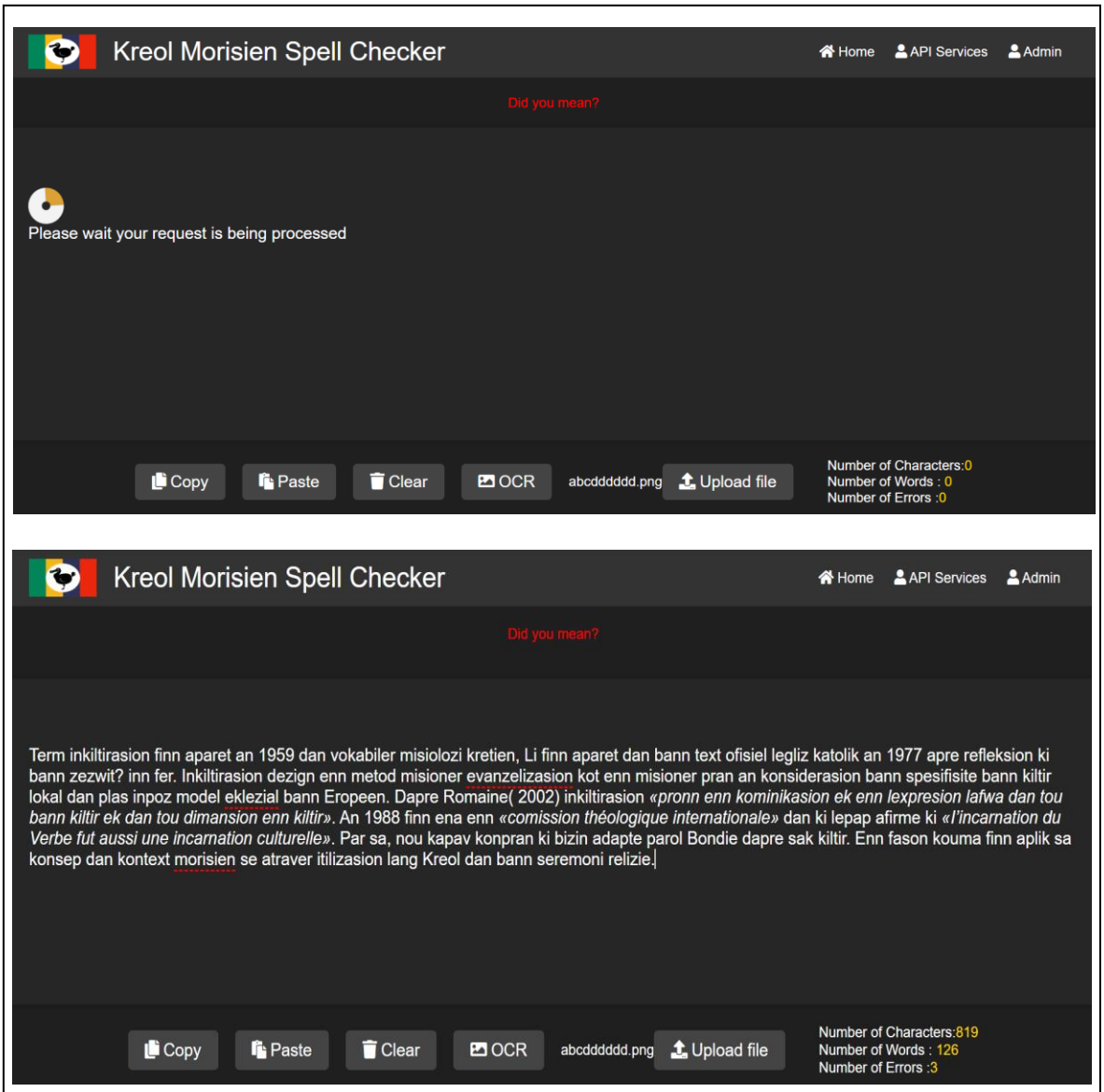


FIGURE 47: IMAGE2TEXT TO CHECK MISSPELLINGS

6.1.7 Test Case 7: Search for Word in the Dictionary

TABLE 32: TEST CASE 7

Objective	Search for words in the dictionary.
Test Data	Word to be search
Expected Result	When search word is entered in the field and search button is clicked, appropriate message should be displayed to indicate whether word is found successfully or not.

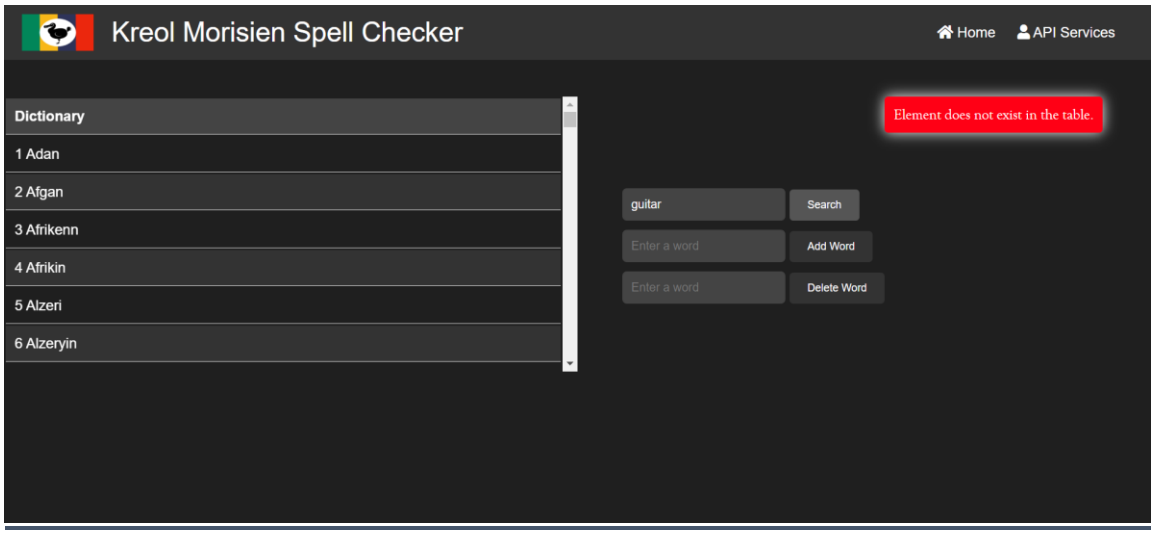


FIGURE 48: SEARCH WORD NOT FOUND IN DICTIONARY

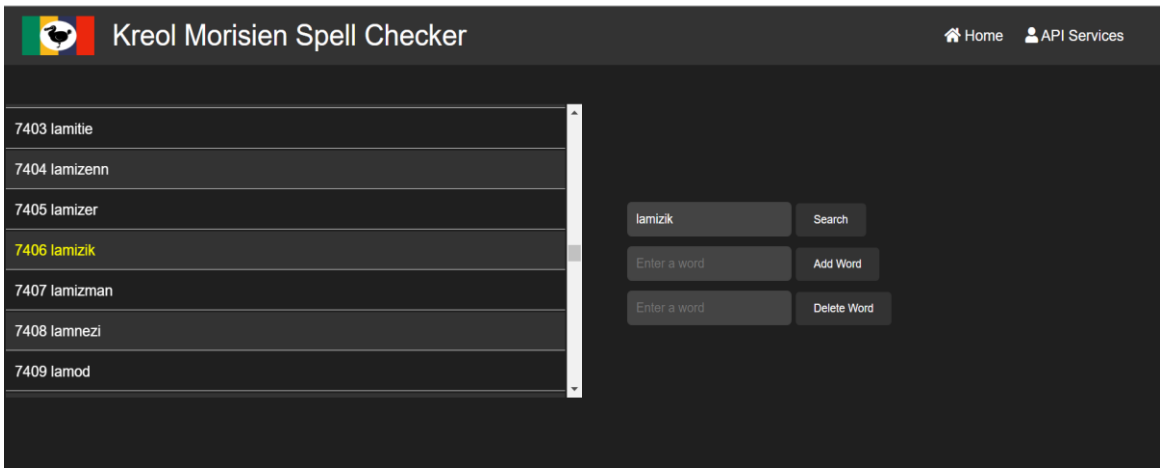


FIGURE 49: SEARCH WORD FOUND IN DICTIONARY

6.1.8 Test Case 8: Add Word to Dictionary

TABLE 33: TEST CASE 8

Objective	Add words to the dictionary.
Test Data	Word to be added.
Expected Result	When word is added successfully, or not appropriate message should appear to indicate admin of the status.

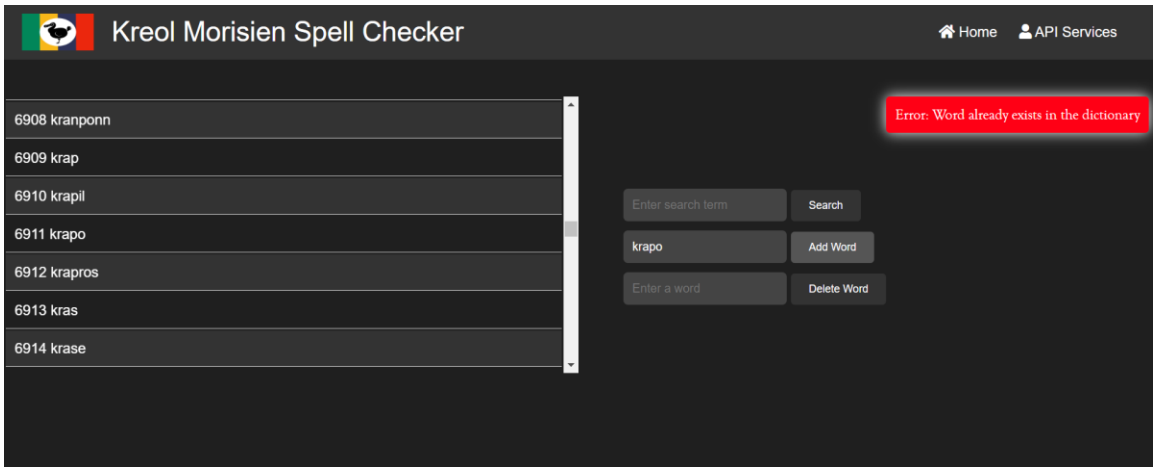


FIGURE 50: ADDING EXISTING WORD

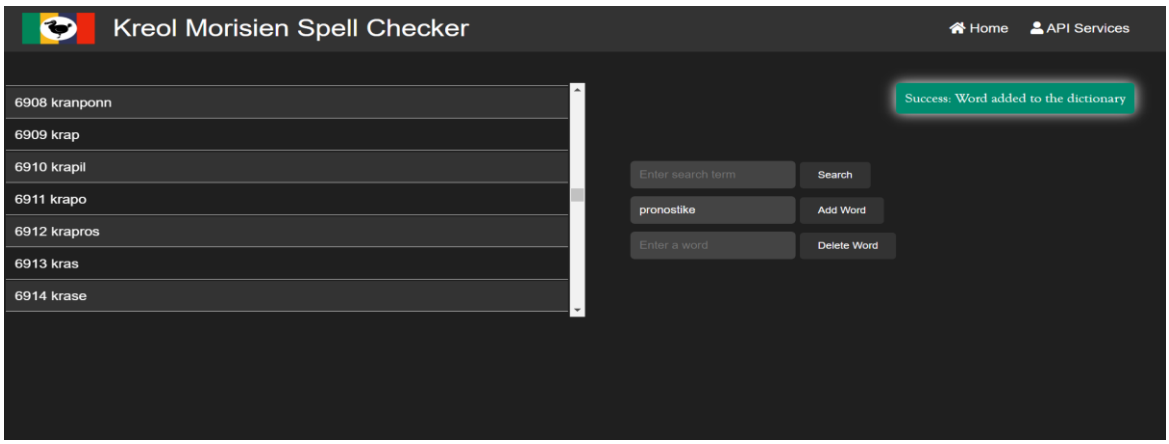


FIGURE 51: WORD SUCCESSFULLY ADDED TO DICTIONARY

6.1.9 Test Case 9: Delete Word from Dictionary

TABLE 34: TEST CASE 9

Objective	Delete words from the dictionary.
Test Data	Word to be deleted.
Expected Result	When word is deleted successfully, or not appropriate message should appear to indicate admin of the status.

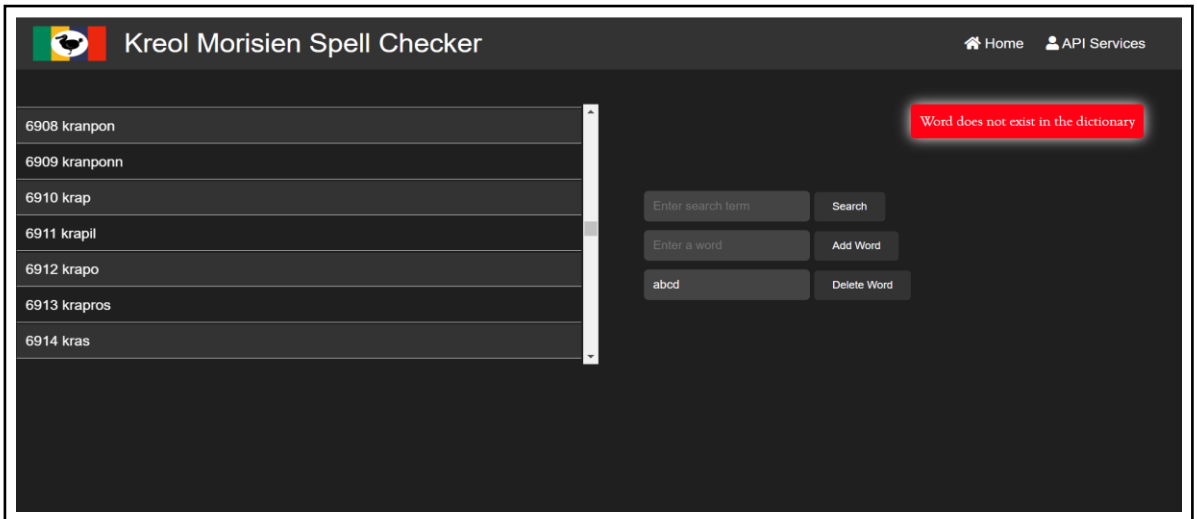


FIGURE 52: DELETING A WORD NOT EXIST IN DICTIONARY

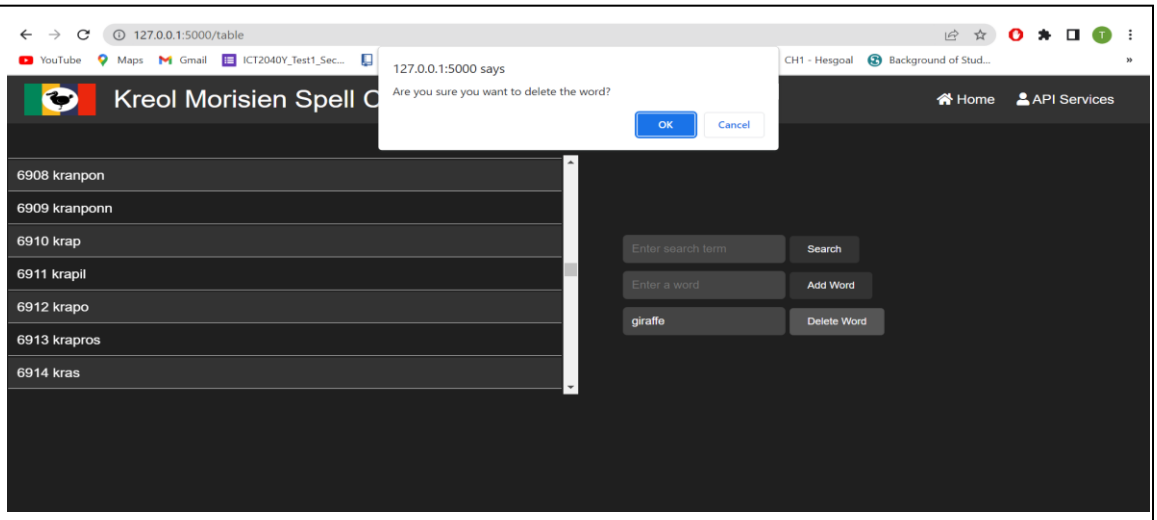


FIGURE 53: PROMPT USER TO CONFIRM HIS DELETE

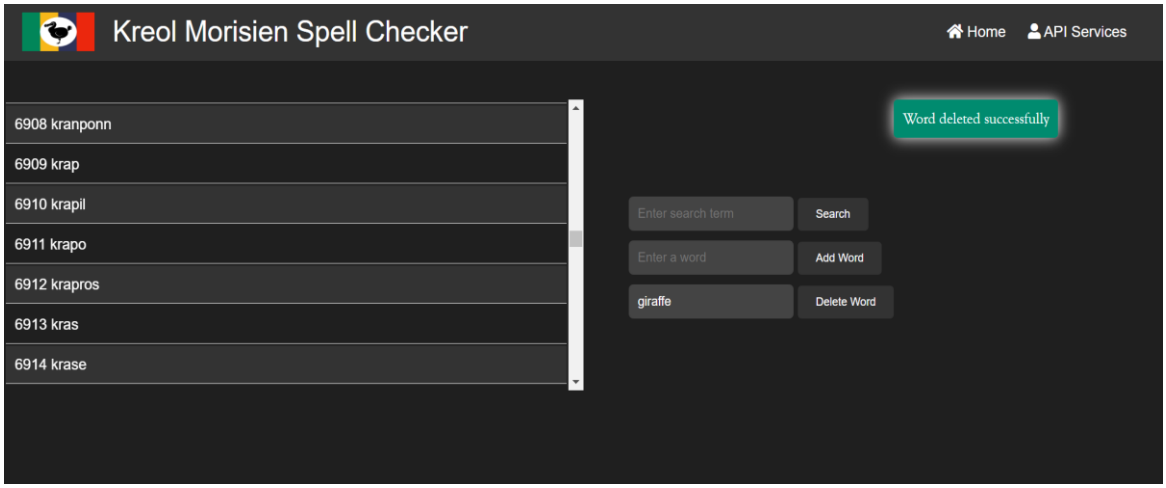


FIGURE 54: WORD DELETED SUCCESSFULLY

6.2 White Box Testing

White box testing encompasses evaluating both the internal structure and behavior of the system.

6.2.1 Test Case 10: Incorrect Word

TABLE 35: TEST CASE 10

Objective	Suggest correct word for misspelled word.
Misspelled Word	lacuisinne
Expected Result	lakwisinn should be listed as suggestion for the misspelled word.
Actual Result	lakwisinn is listed on the suggestion list.
Test Result	Pass

FIGURE 55: CORRECT SUGGESTION FOR THE INCORRECT WORD

6.2.2 Test Case 11: Correct Word

TABLE 36: TEST CASE 11

Objective	Correct word should not be flagged as incorrect.
Correct Word	plastik
Expected Result	plastik is a correct word and should not be underline in red.
Actual Result	The word plastik is not underlined
Test Result	Pass

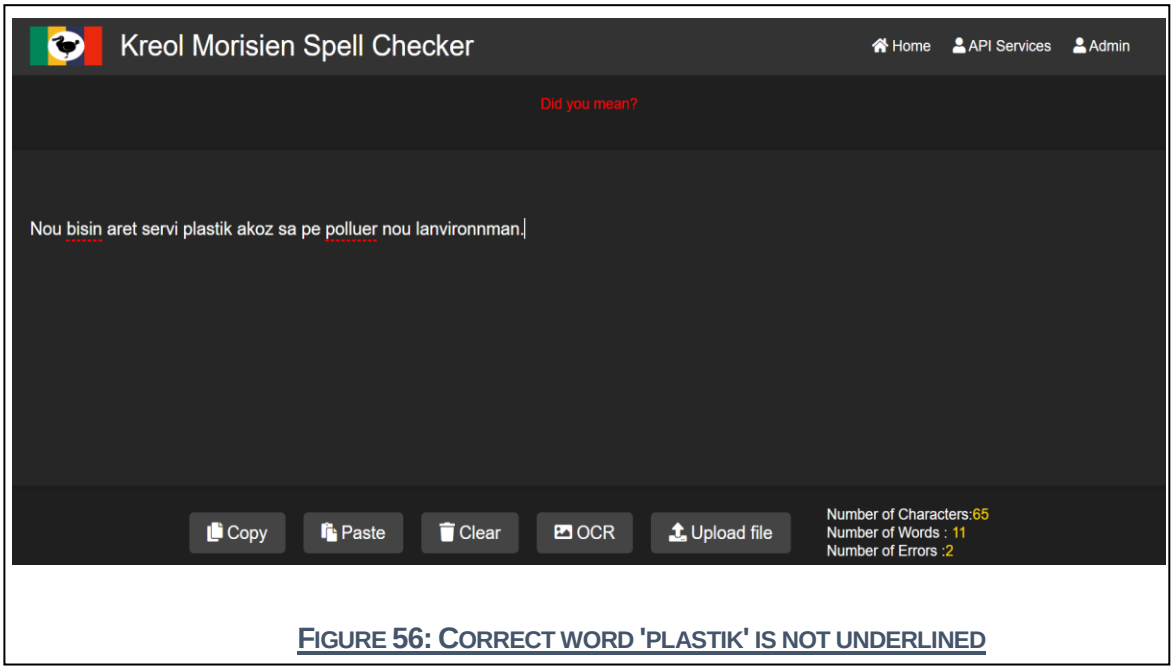


FIGURE 56: CORRECT WORD 'PLASTIK' IS NOT UNDERLINED

6.2.3 Test Case 12: Incorrect word with Uppercase letter(s)

TABLE 37: TEST CASE 12

Objective	Suggestion for incorrect word with uppercase letter(s) should be listed.
Misspelled Word with Uppercase	fAciliTatyon
Expected Result	felicitasion should be listed in the suggestion list.
Actual Result	The word is provided in the suggestion list.
Test Result	Pass

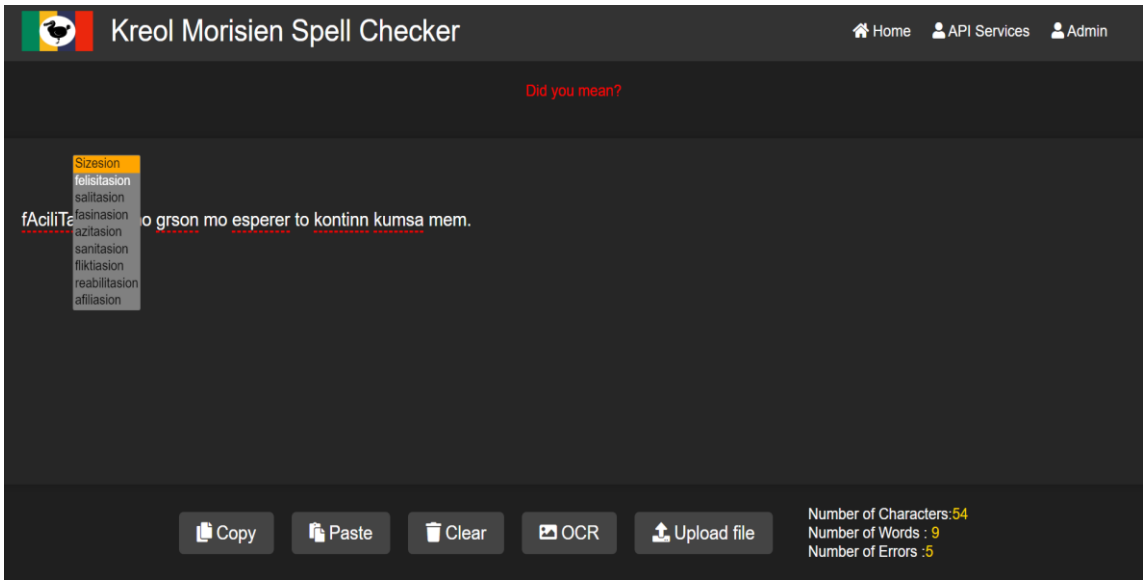


FIGURE 57: CORRECT WORD 'FELISITASION' PRESENT IN SUGGESTIONS

6.2.4 Test Case 13: Incorrect word with punctuations

TABLE 38: TEST CASE 13

Objective	Sentence should be manipulated in a way that after punctuation, first letter of next word is turned to uppercase automatically, and consecutive punctuations are removed.
Sentence with incorrect punctuations and consecutive punctuations.	mo pa envi manz gateau!,paul finn dir so mama.
Expected Result	mo pa envi manz gateau! Paul finn dir so mama.
Actual Result	The sentence is manipulated to the correct format.
Test Result	Pass

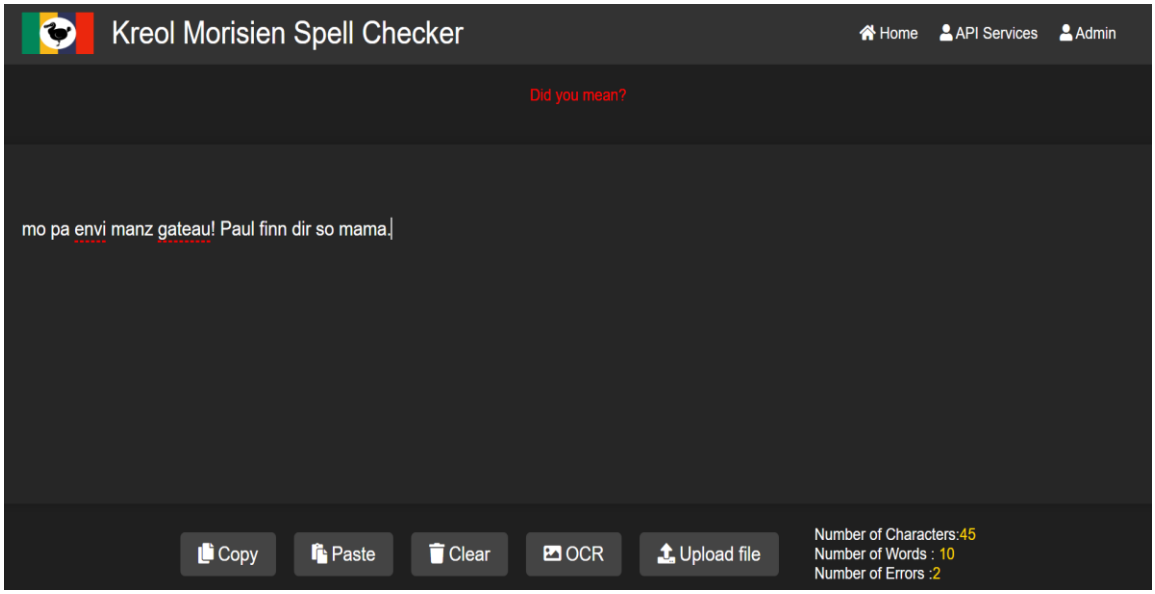


FIGURE 58: CORRECT SENTENCE FORMAT

6.2.5 Test Case 14: Digits not Recognized as Incorrect

TABLE 39: TEST CASE 14

Objective	Digits should not be considered as misspelled.
Sentence With Digits	moris ti ggn so lindepandans le 12 mars 1968.
Expected Result	The digits 12 and 1964 should not be flagged as incorrect.
Actual Result	The digit in the sentence is not underlined.
Test Result	Pass

FIGURE 59: DIGITS NOT CONSIDERED AS INCORRECT

6.2.6 Test Case 15: Word with same Sounding as Top Suggestion

TABLE 40: TEST CASE 15

Objective	Word with matching pronunciation should be ranked among the top suggestions.
Incorrect Word that Sounds correct.	agriculture
Expected Result	agrikiltir is expected to be ranked at the top of the suggestion list.
Actual Result	agrikiltir provided as 1st suggestion in the suggestion list.
Test Result	Pass

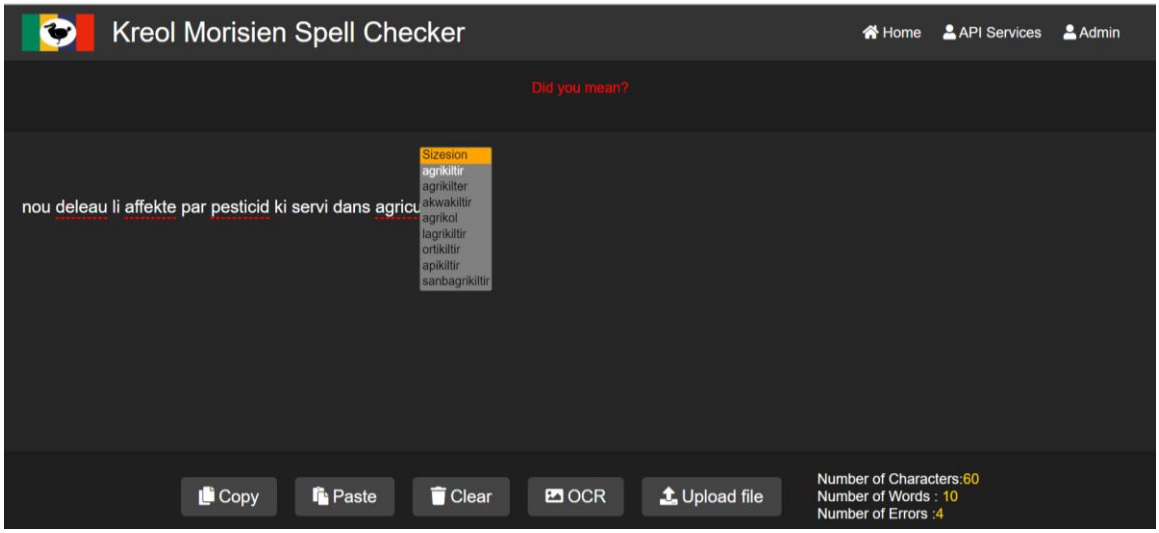


FIGURE 60: CORRECT WORD BASED ON PHONETIC

6.2.7 Test Case 16: Incorrect Word with Consecutive Repeating Characters

TABLE 41: TEST CASE 16

Objective	Correct suggestion for word that has consecutive repeating characters should be suggested.
Incorrect Word with consecutive Repeating similar chars	rennnovvaatyyyyooonnn
Expected Result	The correct word renovasion should be provided in the suggestion list.
Actual Result	The word is provided in the suggestion list.
Test Result	Pass

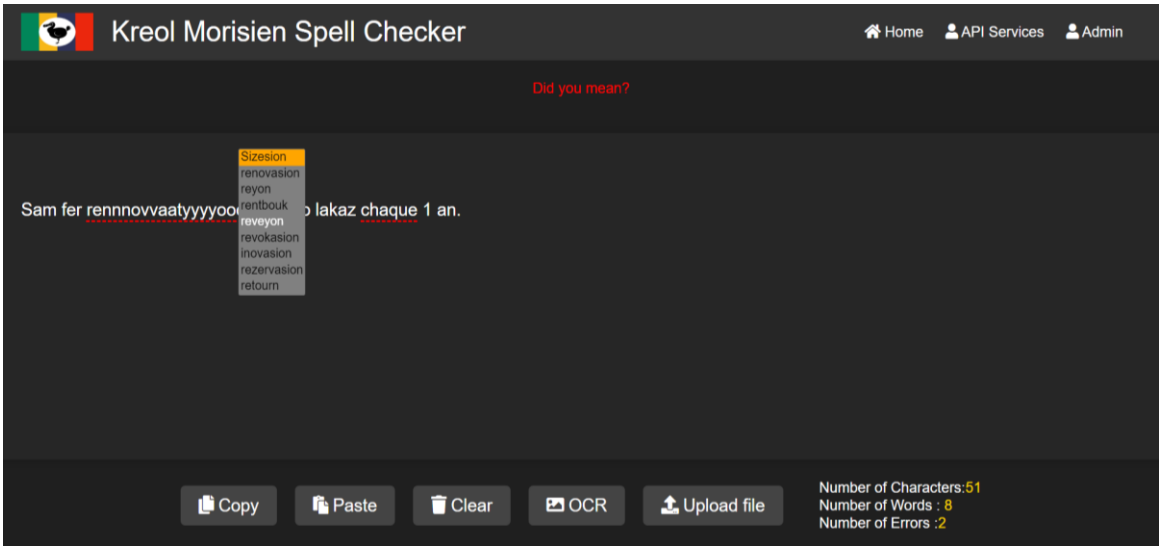


FIGURE 61: WORD WITH REPEATING SIMILAR CONSECUTIVE CHARS

6.2.8 Test Case 17: Word with Special Characters

TABLE 42: TEST CASE 17

Objective	Correct suggestion for words that contain special characters should be mentioned in the suggestion list.
Sentence With Special Characters	Mo content mange s@mu$$a.
Expected Result	The word samousa should be provided in the suggestion window.
Actual Result	The word is provided in the suggestion list.
Test Result	Pass

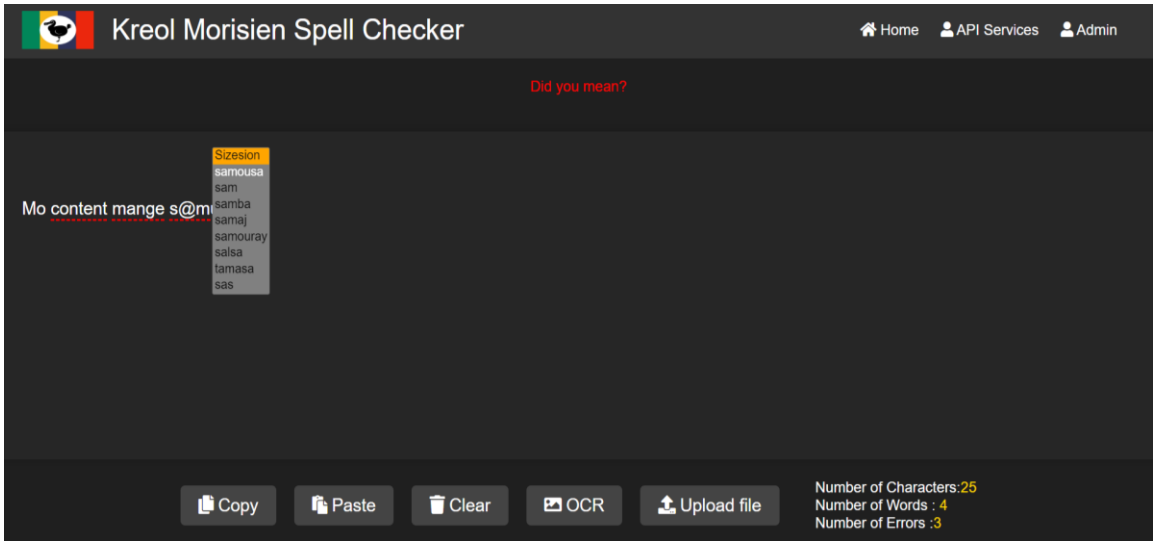


FIGURE 62: CORRECT WORD FOR INCORRECT WORD THAT CONTAIN SPECIAL CHARS

6.2.9 Test Case 18: Words that Ends with ‘la’

TABLE 43: TEST CASE 18

Objective	Correct word for incorrect word that ends with “la” should be provided.
Incorrect word that Ends with ‘la’	chocolatela
Expected Result	sokola-la should appear in the suggestion list.
Actual Result	The word is provided in the suggestion list.
Test Result	Pass

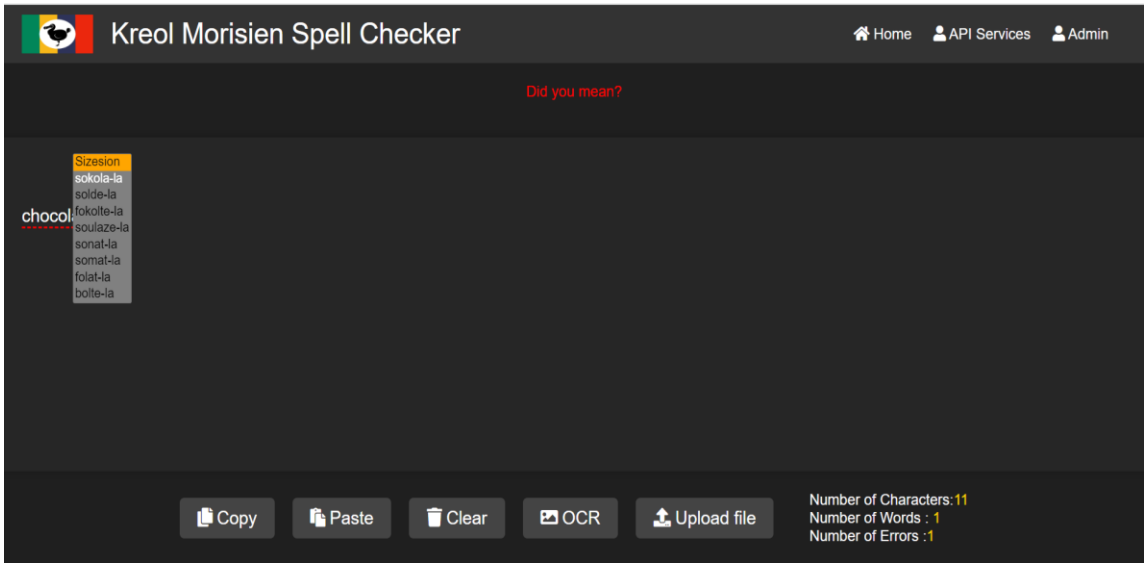


FIGURE 63: CORRECT WORD THAT ENDS WITH ‘LA’

6.2.10 Test Case 19: Last Word Completion Suggestion

TABLE 44: TEST CASE 19

Objective	Last word text completion suggestions should be listed on the bottom of '*Did you mean?*' label when the word being typed length reaches 3.
Last word being typed	baz
Expected Result	Atmost 10 words that starts with 'baz' should be suggested on the bottom of '*Did you mean?*' label.
Actual Result	The last word completion suggestions appear.
Test Result	Pass

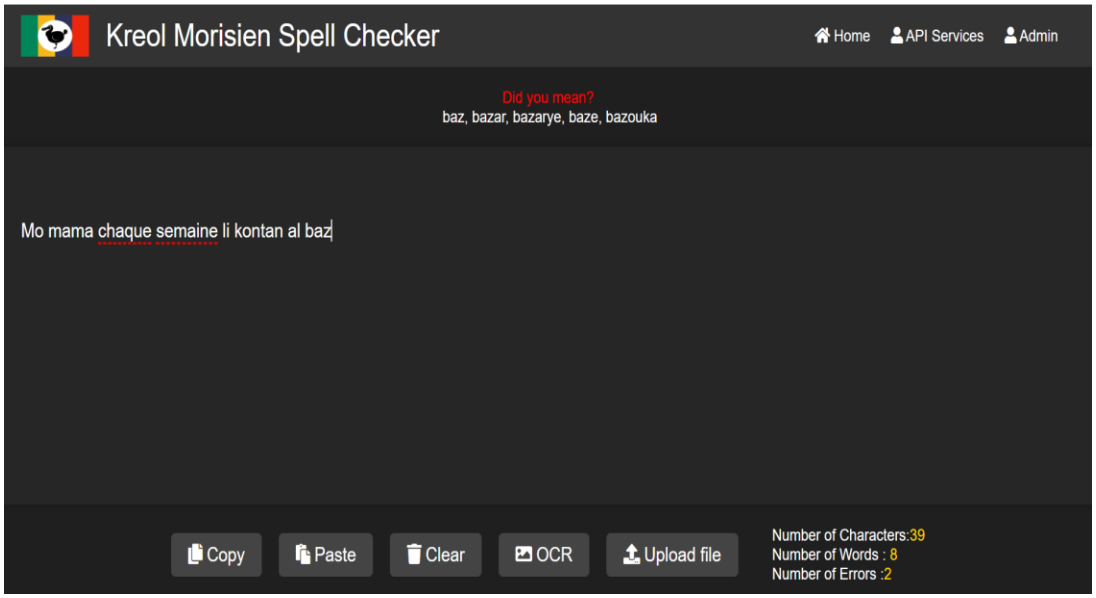


FIGURE 64: LAST WORD COMPLETION SUGGESTIONS

6.2.11 Test Case 20: Adding Word to Dictionary

TABLE 45: TEST CASE 20

Objective	Test whether admin can add word to dictionary.
Test Data	Word to be added.
Expected Result	A record should be created for the new word added in the collection dictionary in the database.

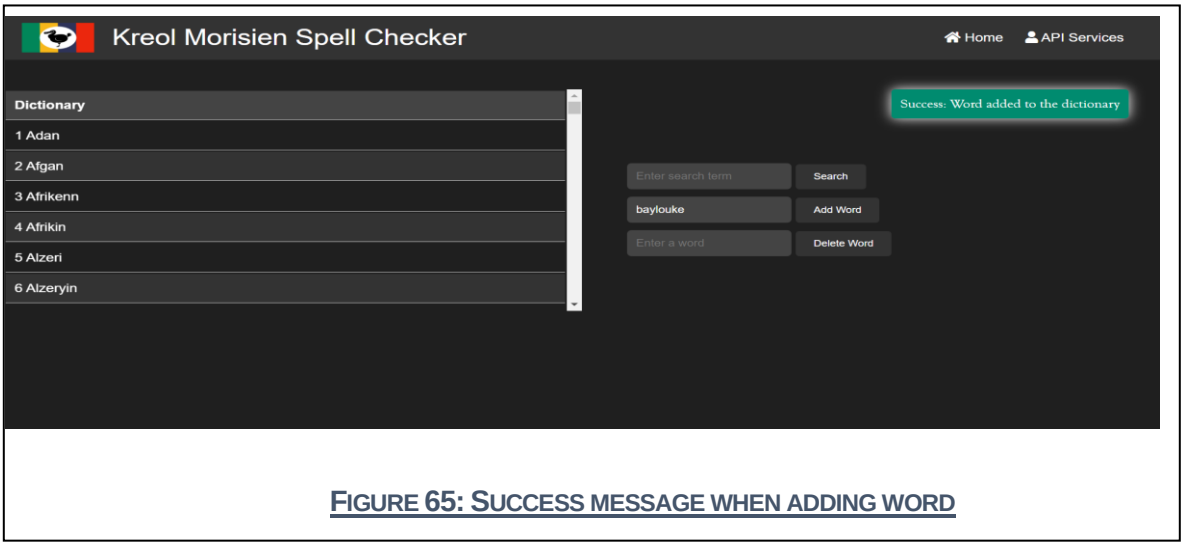


FIGURE 65: SUCCESS MESSAGE WHEN ADDING WORD

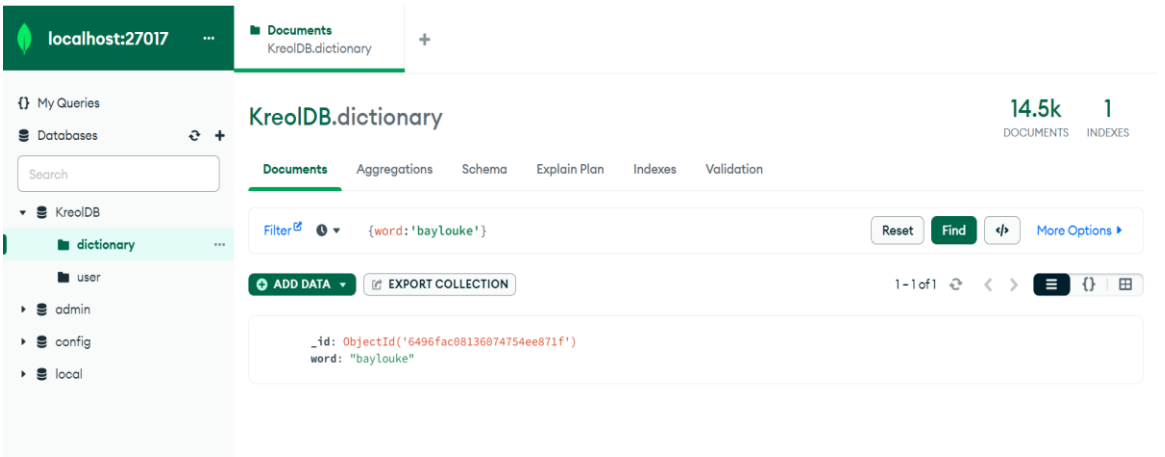


FIGURE 66: WORD RECORD ADDED IN DICTIONARY

6.2.12 Test Case 21: Deleting Word from Dictionary

TABLE 46: TEST CASE 21

Objective	Test whether admin can delete word from dictionary.
Test Data	Word to be deleted.
Expected Result	Record should be deleted for the new word from the collection dictionary in the database.

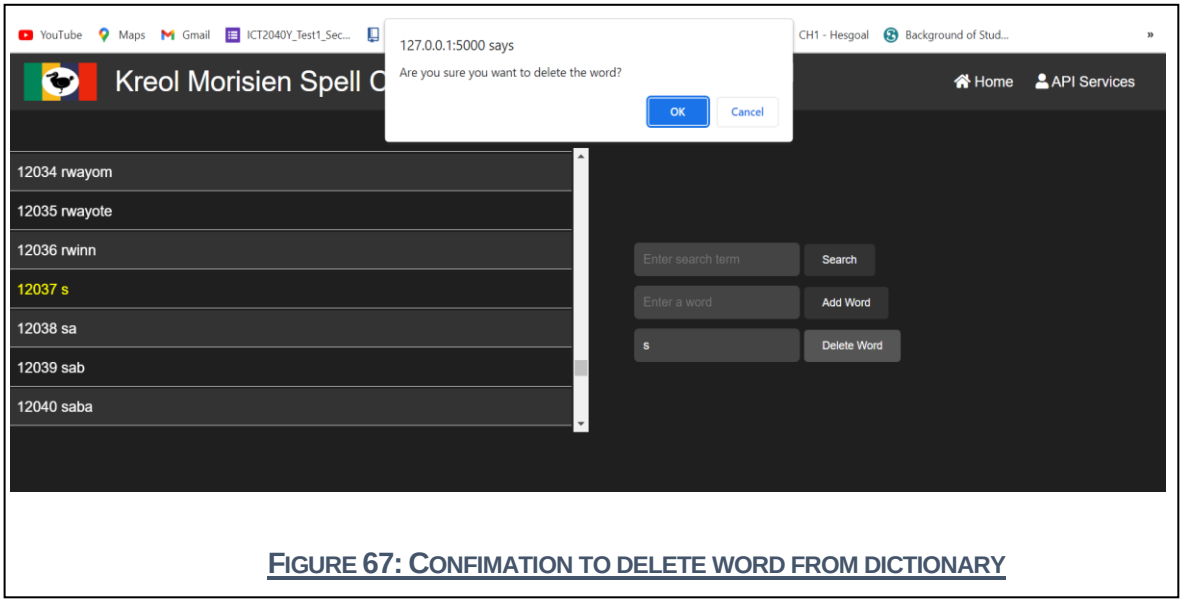


FIGURE 67: CONFIMATION TO DELETE WORD FROM DICTIONARY

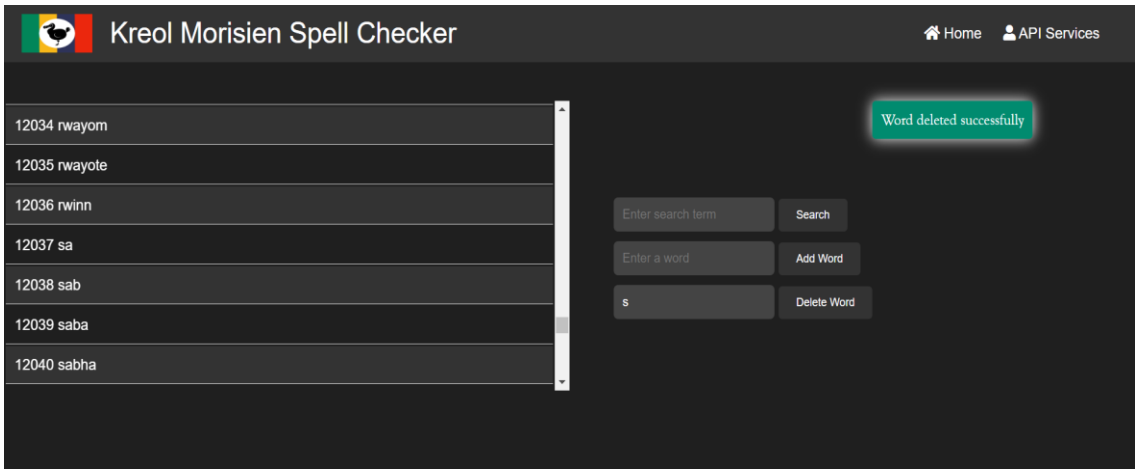


FIGURE 68: SUCCESS MESSAGE IF WORD SUCCESSFULLY DELETED FROM DATABASE

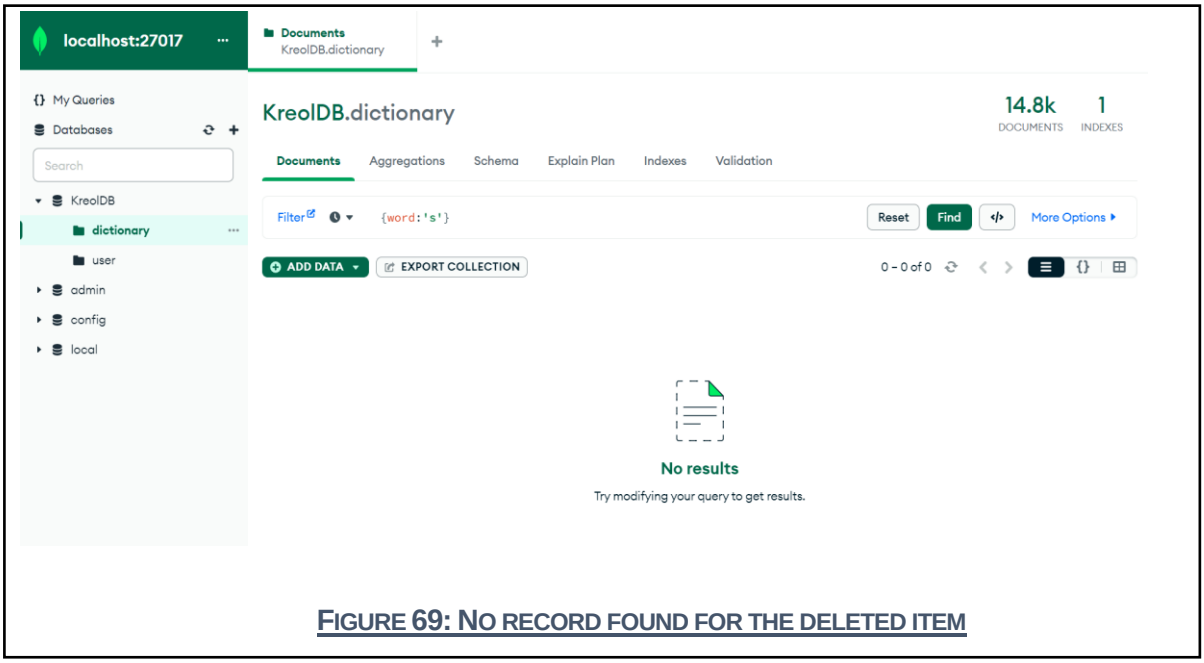


FIGURE 69: NO RECORD FOUND FOR THE DELETED ITEM

6.2.13 Test Case 22: Excluding Direct During Spell Checking

TABLE 47: TEST CASE 22

Objective	Direct speech should be excluded during spellchecking. Direct speech should be enclosed between or “ ” to escape spellchecking feature.
Sentence which contains error in direct speech	Dapre Romaine (2002) inkiltirasion *“pronn enn cominication ek enn lexpresion lafwa dan tou bann kiltir ek dan tou dimansion enn kiltir”*.
Expected Result	The incorrect word ‘cominication’ should not be flagged as an error in direct speech.
Actual Result	‘cominication’ in direct speech not flagged as incorrect.
Test Result	Pass

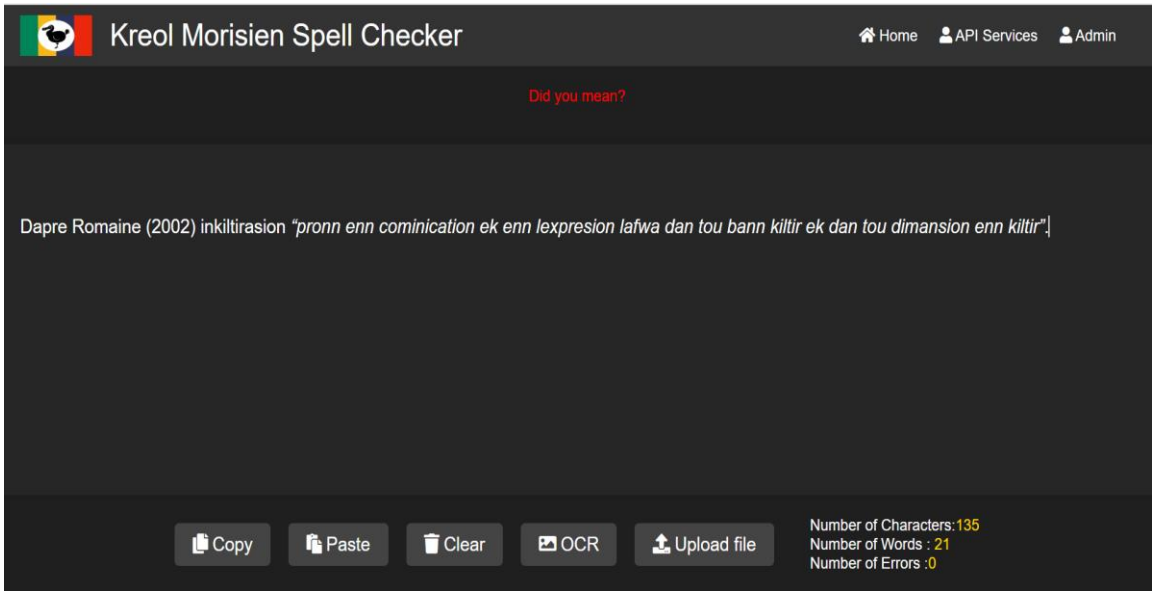


FIGURE 70: THE WORD INCORRECT WORD 'COMINICATION' NOT FLAGGED AS INCORRECT

6.2.14 Test Case 23: Testing spellchecking and correction suggestions API

TABLE 48: TEST CASE 23

Objective	Test API which checks whether a word is correct or not and provides top 8 suggestions.
Test Data	Word to be spellchecked.
Expected Result	A JSON response of whether the word is correct or not along with its top 8 suggestions.

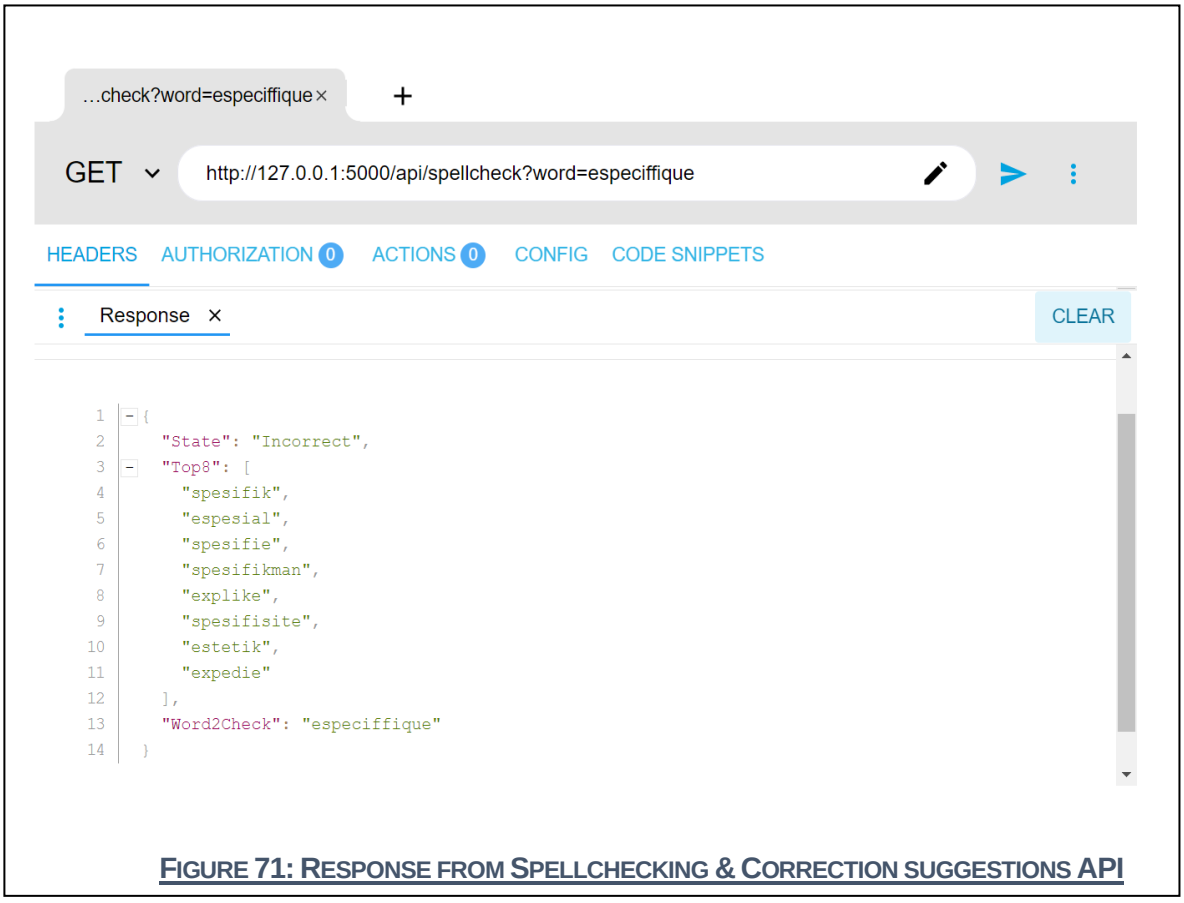


FIGURE 71: RESPONSE FROM SPELLCHECKING & CORRECTION SUGGESTIONS API

6.3 Spell Checker Test Data: Correctly Spelled and Misspelled Words

- The data for this study is collected from five distinct sources encompassing a range of topics, including a book, a dictionary, an article, a news website, and a dissertation. It is important to note that all the gathered data is confirmed to be accurate and reliable.

Table 26 shows the number of correct words that is flagged as misspelled by the spellchecker:

TABLE 49: CORRECT WORD FLAGGED AS MISSPELLED

Document #	Document Type	# of Correct Words	# of Correct Words identified as Incorrect	Reference
1	Book	445	10	(Natchoo, 2021)
2	Dictionary	360	1	(Carpooran, 2019)
3	Article	487	14	(Florigny, Police-Michel Daniella and Carpooran Arnaud, 2011)
4	News Website	771	5	(La Chronique de...Cronos, 2016)
5	Dissertation	335	2	(Rungiah, 2022)
	Total:	*2398*	*32*	
	# of words identified as correct when actually correct	(2398-32)	**2366**	
	# of words identified as incorrect when actually correct		**32**	

- A set of 200 accurately spelled words is selected for our study, wherein we intentionally introduced errors with varying levels of string similarity. These errors encompassed both high string similarity errors and low string similarity errors. All the 200 incorrect words are correctly marked as incorrect by the SC.

6.4 True Positives (TP)

A true positive occurs when the spell checker correctly identifies a misspelled word, which are actually misspelled.

6.5 True Negatives (TN)

When the spell checker does not flag a correctly spelled word as misspelled, it is considered a true negative.

6.6 False Positives (FP)

When the spell checker marks a word as misspelled, but it is, in fact, spelled correctly, it is a false positive.

6.7 False Negatives (FN)

When the spell checker does not flag a word as misspelled, but it is indeed misspelled, it is a false negative.

6.8 Performance Measures

The performance of our Kreol Morisien SC is evaluated using four metrics: accuracy, precision, recall and F1-score. A confusion matrix helps in calculating those metrics. Table 50 shows the confusion matrix.

TABLE 50: CONFUSION MATRIX

	Predicted Values	
Actual Values	Positive	Negative
Positive	Number of True Positives	Number of False Negatives
Negative	Number of False Positives	Number of True Negatives

Based on 6.1 we can deduce:

- Total number of correctly spelled words = 2398
- Total number of misspelled words = 200
- Number of flagged words = 32 (correct words flagged as misspelled)

Calculation of TP, TN, FP, FN:

- TP: 200
- TN: 2366 (2398-32)
- FP: 32
- FN: 0 (all misspelled words are detected)

TABLE 51: CONFUSION MATRIX FOR OUR SPELLCHECKER

	Predicted Values	
Actual Values	Positive	Negative
Positive	200	0
Negative	32	2366

6.8.1 Accuracy

Accuracy is the number of words that are correctly recognized as correct and incorrect.

Formula to calculate Accuracy:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

FIGURE 72: FORMULA TO CALCULATE ACCURACY

6.8.2 Precision

Precision is out of all predictive correct words, how much our spellchecker predicted correctly. It is defined as follows:

Formula to calculate Precision:

$$Precision = \frac{TP}{TP + FP} \text{ or } \frac{\text{True Positive}}{\text{Predictive Results}}$$

FIGURE 73: FORMULA TO CALCULATE PRECISION

6.8.3 Recall

Recall indicates the proportion of actual positive instances that were correctly flagged as positive by our spellchecker. It is defined as follows:

Formula to calculate Recall:

$$Recall = \frac{TP}{TP + FN} \quad or \quad \frac{True\ Positive}{Actual\ Results}$$

FIGURE 74: FORMULA TO CALCULATE RECALL

6.8.4 F1 Score Measure

It provides a balanced evaluation of our spellchecker's performance by considering both false positives and false negatives.

Formula to calculate F1 Score:

$$F\text{-}score = \frac{2 * Recall * Precision}{Recall + Precision}$$

FIGURE 75: FORMULA TO CALCULATE F1 SCORE

6.8.5 Mean Reciprocal Rank (MRR)

This is used to evaluate the effectiveness of ranking the correct suggestions of our spellchecker. MRR measures the average rank of the first relevant item in a list of ranked results.

Formula to calculate MRR:

$$\mathrm{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\mathrm{rank}_i}$$

FIGURE 76: FORMULA TO CALCULATE MRR

Refer to ***table 53*** to get an idea of how the MRR is calculated:

1 * 178	**=**	**178**	*178 words appear 1st in suggestion list*
$\frac{1}{2}$ *** 14**	**=**	**7**	*14 words appear 2nd in suggestion list*
$\frac{1}{3}$ *** 7**	**=**	$\frac{7}{3}$	*7 words appear 3rd in suggestion list*
$\frac{1}{6}$ *** 1**	**=**	$\frac{1}{6}$	*1 word appears 6th in suggestion list*

0.9375

6.8.6 Calculation of Accuracy, Precision, Recall, F1 Score, MRR

TABLE 52: PERFORMANCE OF OUR SPELLCHECKER

Accuracy	$\frac{200 + 2366}{200 + 2366 + 32 + 0}$	98.77%
Precision	$\frac{200}{200 + 32}$	86.21%
Recall	$\frac{200}{200 + 0}$	100%
F1 Score	$\frac{2 * 1 * 0.8621}{1 + 0.8621}$	92.59%
Mean Reciprocal Rank	Refer to 6.8.5	93.75%

As we can see in table 52, our spellchecker boasts a high accuracy of 98.77%, ensuring it effectively detects misspelled words in texts. With a precision of 86.21%, it precisely identifies incorrect spellings, while achieving a perfect recall of 100%, capturing all misspelled words without omissions. The F1 score of 92.59% reflects its balanced approach between precision and recall. Furthermore, the Mean Reciprocal Rank of 93.75% demonstrates its proficiency in providing relevant and well-ranked suggestions for correcting misspellings. Overall, these evaluation metrics validate the spellchecker's reliability in enhancing text accuracy and aiding users in producing error-free content.

CHAPTER 7

CONCLUSIONS

7. Conclusions

In this paper, we have achieved successful development of a spell checker for Kreol Morisien (KM). We use advanced algorithms and techniques to automatically detect errors and provide accurate correction suggestions. The spell checker fulfills all functional and non-functional requirements, demonstrating its robustness, reliability and ease of use.

The system incorporates a mapping feature that translates some French words to KM, and it also promotes a technique to suggest corrections for misspelled words, considering both spelling and pronunciation variations. This ensures comprehensive coverage for different types of errors.

To enhance accessibility, we have created a user-friendly web-based application. This real-time KM spell checker actively underlines incorrect words as they are typed and provides prompt suggestions for incorrect words in a reasonable timeframe. The combination of real-time feedback and quick suggestions improves the overall user experience and facilitates effective error correction.

This tool can be utilized by people who are learning KM and especially this will benefit students who are studying Kreol Morisien, as it saves them time and effort in manually identifying spelling errors in their writing. Additionally, this will prove invaluable to stenographers, aiding them in accurately transcribing Kreol Morisien speech to text for example, in courts. By contributing to the recognition and preservation of Kreol Morisien, this tool paves the way for the availability of more digital materials and tools for the language. Furthermore, it has the potential to encourage more individuals to learn and embrace Kreol Morisien, fostering its growth and widespread adoption.

Overall, the development of this Kreol Morisien Spell Checker represents a significant contribution to the field, offering a reliable and efficient solution for error detection and correction.

7.1 Challenges

During the development of the Kreol Morisien Spell Checker, we encountered several challenges that required careful consideration and innovative solutions. These challenges included building a Kreol Morisien dataset, performing data cleansing and developing a mapping system for language analysis. Through extensive research, problem-solving, and diligent testing, we are able to address these challenges and create a functional and reliable Kreol Morisien Spell Checker tool.

7.2 Future Works

Despite successfully developing a spell checker that checks the spelling of individual words, there are areas for improvement that can enhance its performance. One key aspect to address is reducing the false positive rate, where correct words are mistakenly flagged as errors due to being absent in the dictionary. To mitigate this issue, the spell checker can be enhanced by incorporating a broader range of words, including different variations and forms. This expansion of the dictionary will help improve the accuracy and effectiveness of the spell checker. Additionally, incorporating a large corpus of correctly spelled Kreol Morisien text can enable contextualization within the spell checker. This contextualization can leverage techniques such as part-of-speech tagging, allowing for a more sophisticated and context-aware spell-checking mechanism. Also, a spellchecker plugin for software like mail will be developed, ensuring accurate information exchange by identifying and suggesting corrections for misspelled words which will enhance communication. By undertaking these future works, the spell checker can be further refined and optimized, leading to enhanced performance and accuracy in detecting and correcting spelling errors in Kreol Morisien text.

Appendix A: Reciprocal Rank to Calculate MRR

Table 53 shows how the Reciprocal Rank (RR) is calculated:

TABLE 53: RECIPROCAL RANK

Correct Response	Query	Proposed Results	Reciprocal rank
agrikol	ackrickoll	agrikol, akrilik, ankreol, arkaik, akimil, apriko, abrigo, brikol	1
akt-de-nesans	actedenaissance	akt-de-nesans, atenant, tenis, aterisaz, alienasion, amenazman, antiderapan, lenpasians	1
agrikiltir	aggreeickkculture	agrikiltir, agrikilter, agrikol, lagrikiltir, akwakiltir, ortikiltir, sanbagrikiltir, apikiltir	1
atlantik	alantique	antik, alantour, atlantik, antikite, lantikite, anantie, antite, avantirye	1/3
alkolik	alcoholique	alkolik, alterlikou, alou, apostolik, lotelie, lozik, kalchoul, choli	1
adilter	aldulteur	alter, adilter, adiltere, altern, alterne, adapter, atouler, aler	1/2
alfabet	alphabeth	alfabet, alfabetiz, alfabetik, alfabetism, aplat, labet, aparet, laplanet	1

akwaryom	aquarium	akwaryom, akaro, arom, karom, akros, antiryom, akazou, karyol	1
arkansiel	arc-en-ciel	arsenal, esansiel, arkansiel, apresie, armenien, arteriel, zarenie, larkansiel	1/3
artifisiel	arttiphyciel	artifisiel, artis, artiso, arteriel, artikil, artistic, ratifie, astisie	1
astronot	ashtraunauth	antrenou, astronot, astronom, astronomi, aspiran, aspirasion, astrolog, astrolozi	1/2
atasman	attachemn	atasman, atann, atase, atake, atan, atenn, atarde, atable	1
otoekol	autoecole	otoekol, otokolan, otoexame, otozere, otonom, otorite, otonomi, tole	1
bakteri	bactteiri	bakteri, batir, bater, batri, bati, balti, latiri, ateri	1
barmenn	barman	parman, barmenn, bayman, batman, basman,rarman, aman, aran	1/2
barsketborl	basketball	barsketborl, baskil, baset, barket, bankal, bakanal, kasket, baprebap	1

bavardaz	bavardge	bavar, bavard, bavardaz, bazarye, avarye, batarde, azarde, atarde	1/3
belser	belsoeur	belser, blese, leser, belmer, belo, belie, bese, belot	1
bonnaniverser	bonanifversair	bonnaniverser, boner, laniverser, bonatoufer, banier, bonker, konvers, konferans	1
bidzeter	budgetaire	bweter, bidzeter, baget, deter, deger, dester, blager, gete	1/2
birokrasi	bureaucracy	birokrasi, birokrat, birokratik, brodri, groras, otokrasi, demokrasi, biografi	1
boutey	buteille	boutey, bote, bate, be, foteye, meteye, teyer, te	1
kafetarya	cafetaria	kafetarya, vedetarya, chawtari, safari, tafta, afermi, tantarik, lafet-die	1
kalkilater	calcculattreur	kalkilater, kalkilatris, kalkil, kapsilater, lakizater, karbirater, katrer, vantilater	1
kalbas	calebasses	kalbas, kaleptis, kales, koales, kalasi, kalis, kale, katles	1
servolan	cerfevolant	servolan, servant, revoltan, fervan, ferblan,	1

		efervesan, benevolman, efreyan	
chalenj	challenge	chalenj, chalni, chal, chali, chalao, chana, chalta, chake	1
sapito	chapiteau	chapati, chabi, sapito, charter, anpite, chali, chake, chatni	1/3
semiz	chemise	semiz, chom, chirz, ches, chek, cheke, chekof, chi	1
sirirzi	chirurgie	sirirzi, sirpriz, sire, siryin, sirkile, sirfile, sirviv, sirkil	1
sokola-la	chocolatela	sokola-la, solde-la, fekolte-la, soulaze-la, sonat-la, somat-la, folat-la, bolte-la	1
chawtari	chouwthary	chawtari, chamar, chadar, chatkar, charter, chaw-chaw, chacha, chalta	1
sinematograf	cinematograph	sinematograf, sinematografik, sinema, oseanografi, demografi, animator, onomatope, inatandi	1
sinkantenn	cinquantianne	sinkantenn, sinkant, inkonsian, inkonsians, inkonstan, intension, instantane, finansie	1
klarifikasion	clarificlaatyon	klarifikasion, kalifikasion, klasifikasion, gratifikasion, planifikasion, ratifikasion, ramifikasion, falsifikasion	1

kwinsidans	coincidance	kwinsidans, kwinsid, konfidans, kondans, kwins, insidan, inpridans, linpridans	1
kominote	communaute	kominote, komant, kominoter, komanter, komite, komete, konte, koute	1
konsey	conseil	konsey, konseye, konseyab, konser, konsep, kone, konsern, kose	1
konservater	conservater	konservater, konserte, konserv, konspirateur, konstrikter, konsomater, konzelater, kontestater	1
konsistans	consistane	konsistans, konsistan, konsiste, konsist, konstan, konsian, konsians, konstate	1
koutchia	couchia	koutcha, koucha, chacha, chachi, choulha, chocho, chi, cha	1
definitivman	deatfinnitetivvman	definitivman, definitive, destitie, definision, efektivman, positivman, gratwitman, dernierman	1
diabetik	diablebettetictique	diabetik, diabitid, diabet, diabolik, dezabitie, dialektik, sibstitie, benitie	1
diksioner	dicksionnyer	diksioner, demisioner, adiksion, aksioner,	1

		reaksioner, diskision, misioner, fonksioner	
diskision	dicssussyon	diskision, disansion, adiksion, odision, edision, adision, dimension, sision	1
diferans	diffwrnce	diferans, diferansie, diferan, difere, diform, dife, diare, diformite	1
dinozor	dinausaur	dinozor, dinasti, dinamis, nazo, dinamiter, dynamo, dinamit, dinamik	1
direkteman	directtement	direckteman, diskretman, direkter, divertisman, dernieman, korekteman, dezisteman, dezieman	1
dictatorial	ditctatorial	dictatorial, diktatir, editorial, difteri, distrik, ditor, dizital, diagonal	1
dokimanter	documentaire	dokimanter, dominer, dominater, komanter, dousman, dormer, douser, douler	1
dousman	douchemwt	dousman, dous, douser, doue, kouset, fouset, souset, bousert	1
drwat	droite	drwat, drwatie, drafte, adrwat, drwa, dwatet, debwate, rate	1
lodkologn	eaudecologn	lodkologn, adeton, dedlagn, deboulonn, omolog, lapologn, zeolog, detonn	1

ekonomik	eggonommikc	ekonomik, ekonomiks, ekonomiz, economize, egois, ekolozik, lekonomi, eroik	1
rezional	egioanl	rezional, egal, ezitan, evidan, evidans, eminans, exizans, exizan	1
egalman	eglman	egalman, alman, eleman, regleman, telman, selman, folman, glian	1
exekitif	egzekitif	exekitif, exekit, exekite, azektif, lexekitif, obzektif, efektif, dezekilib	1
exotic	ekzotique	exotic, ekoute, ezot, erotic, kotite, kotize, ekolie, kotie	1
anlevman	enlevement	anlevman, anpleman, inkleman, nevyeman, enervan, eleman, regleman, elegan	1
epanwir	epanouir	epanwir, epanwi, epanaz, anou, antour, ankour, pano, pani	1
eparpiye	eparpiller	eparpiye, oparler, popiler, papie, exanpler, serkiler, epise, kapiler	1
epidermik	epidremic	epidermik, ipodermik, epiderm, premie, pyramid, extremis, lepidermi, epise	1
espagnol	espanyoll	espagnol, espion, espadon, espesial, esanz, etanol, esantiyon, pano	1

exazere	exagerer	exazere, exazer, exize, egrer, extre, expre, egare, efare	1
exkirsion	excrusyon	exkirsion, expresion, expirasion, extorksion, exsepsion, extansion, expansion, exanpsion	1
experyanse	experiense	experyanse, experians, expres, experimante, lexperyans, expedie, extenie, expekte	1
extansion	extnesion	extansion, extinksion, expansion, exanpsion, extorksion, exzanpsion, exekision, expression	1
felisitasion	facilitatyon	felisitasion, salitasion, fascination, azitasion, sanitasion, fliktiasion, reabilitasion, afiliasion	1
fakter	facquetere	fakter, faktir, feter, federe, fakilte, faver, faner, akselere	1
flanbwayan	falmboiyant	flaman, lamyant, flanbwayan, alimant, falokrat, pliyant, fabrikan, buoyant	1/3
feedback	feetbak	feedback, fatak, fet, feb, lefbak, fetaz, ferblan, eta	1
fezabilite	fesablite	fezabilite, fasilite, febli, fatalite, fertilite, egalite, legalite, realite	1

feyte	feyyet	feyte, fet, fey, fe, keyet, fele, freyer, eye	1
frikase	ficasse	frikase, fikse, fianse, finansie, fis, fisle, fas, faset	1
figir	figuire	figir, fitir, fisir, fier, figiran, filtre, fimie, filier	1
fiksasion	fixation	fiksasion, fiksion, finision, fiasko, fizion, fason, mitasion, bitasion	1
fourset	fourchette	fourset, fore, foursetaz, forse, fouset, fournez, foule, fouke	1
frekantasion	fraisquandtation	fragmantasion, frekantasion, frustrasion, fermantasion, fekonfasion, rekomandasion, argimantasion, reinkarnasion	1/2
fromaz	framz	fromaz, fraz, foraz, franz, fam, ram, frap, fran	1
fransman	francehement	fransman, fonsierman, franse, fragman, fransez, tranbleman, ansegnman, ransegnman	1
friktifie	fructifye	friktifie, fortifie, rektifie, frontier, fevriye, fouti, ratifie, fotif	1
fristrasion	furustraityon	fristrasion, restriksion, orkestrasion, prestasion, larestasion, fraksion, prostitision, abstraksion	1

fouter-dezord	futerredesord	fouter-dezord, freser, federe, fetdemor, literer, interes, terer, stereo	1
gardien	gardian	gardien, gradian, kardan, gradin, gardinaz, garan, zardin, gandia	1
gradielman	gradelment	gradielman, gratman, gradien, gardin, gravman, oralman, alman, galan	1
gramer	grammemaire	gramer, gram, gramo, graver, greser, grener, gramatin, grander	1
gran-madam	grandmadam	granmama, gran-madam, grandi, grander, gratman, gran-mal, gradian, gravman	1/2
grenouy	grenouille	grenouy, granile, grenie, grenadie, grofil, renouvle, grefie, redouble	1
griyad	grillade	griyad, gril, gerila, grad, grenad, griyaz, gilas, grimas	1
garanti	guarantee	garanti, karanter, garan, gante, parante, aparante, arzante, galate	1
elikopter	helicapteure	elikopter, selibater, delibere, realizater, headmaster, reantere, mediater, desinater	1
eritaz	herritage	eritaz, leritaz, etit, bitaz, veritab, meritan, debitaz, berbiaz	1

yerarsi	hierarchy	yerarsi, yerarsik, yer, ayer, vyeyar, lerouy, berkay, headboy	1
linpridans	imprudence	inpridans, inpridan, amerdan, linpotans, mordan, irzans, linpridans, prezans	1
iniorans	ignorance	iniorans, inzerans, inioran, inportans, intans, liniorans, inpotans, inosan	1
inklinasion	inclnasion	inklinasion, inkarnasion, inplantasion, inondasion, intension, inplikasion, invokasion, linalasion	1
inkoeran	incoherent	inkoeran, inkonpetans, intern, inport, inporte, interpret, interes, chermenn	1
inkonpreansib	incomphrehensible	inkonpreansib, inkorizib, inkonparab, inkoriptib, inkonpatib, inkonpetans, konpreansib, inkorpore	1
inzenier	ingenieur	inzenier, ine, anger, inzer, interyer, inferyer, interfere, linteryer	1
insansib	insensible	insansib, insanzab, ansenie, inzis, insandi, insestie, insertin, inzir	1
invizib	invisible	invizib, inkorizib, invit, invite, ini, ilizib, tinizi, vizib	1

iregilie	irregulier	regle, regale, iregilie, treler, reseler, relier, freyer, reouver	1/3
zistifikasion	justifjtion	zistifikasion, sizession, estimasion, institision, destinasion, distilasion, distinksion, destriksion	1
karanbol	kauaranbwol	karanbol, kranaval, karabi, kanbryol, kafenol, parabol, kalanbour, kamanber	1
labalenn	labaleine	labalenn, labalans, lalin, lalinn, lalenn, lale, lasalinn, lasemine	1
laboutik	labuteek	laboutik, labet, labit, labek, labonte, labe, labre, label	1
lakonpagnman	laccompagnement	lakanpagn, lakonpagnman, lakoutpagn, lagreman, lakoutreman, akonpagn, lankourazman, lankadreman	1/2
lakoklis	lacoqueluche	lakoklis, laklos, lakel, lalkolis, lapolis, laboks, lanzelis, labous	1
lakwisinn	lacuisine	lakwisinn, lakwis, lakize, lakinn, lasimine, lamine, lafwinn, lapwint	1
ladwann	ladouane	ladwann, ladan, latourne, ladou, ladefans, ladous, lamone, lazourne	1

lafore	lafoeretet	leveret, lafore, lafot, lapovrete, lafnet, lafont, loperet, lalert	1/2
lalimier	lallumiere	lalimer, lamer, lalert, lale, lamerd, lame, laliberte, labyer	1
lalimier	lallumyere	lalimier, lamer, lalert, lale, lamerd, lame, laliberte,labyer	1
lamizik	lamiusique	lamizik, lamous, lamizer, lamikal, lamitie, lamine, lasimine, laik	1
lankourazman	lancoutragemnt	lankourazman, lakoutreman, lantouraz, ankourazan, ankouraz, lakoutpagn, latant, lanouyman	1
larozwar	laoreausoir	larozwar, larmwar, lamaswar, yeroswar, laboratwar, lakrosaz, lantonwar, oswar	1
laranzman	larrangement	laranzman, largiman, lanterman, lansman, langazman, lavansman, aranzman, lagreman	1
lastrolozi	lastrology	lastrolozi, lastron, lastronomi, lastok, astrolog, astrolozi, ladrog, lapologn	1
lekarir	lecarrure	lakarir, lare, letre, lezar, lamare, egare, lesarp, efare	1

lekleraz	leclairage	lekleraz, lakolaz, legliz, lesklavaz, leparaz, letalaz, lelvaz, laraz	1
lanfans	lenfance	lanfans, linflians, lofans, linpotans, lenn, lezans, lesans, defans	1
linstriman	linstrueman	linstriman, lanterman, linstan, linstin, linstans, listre, zisteman, sinserman	1
likid	liquide	likid, likidite, liki, lisid, likisek, lide, lipie, lider	1
makatia	macaatchchia	makatia, machak, mach, macho, kacha, facha, bacha, katha	1
maltretans	maltraitance	maltretans, maltret, latirans, maltande, malsans, malantandi, matapan, maravann	1
matematik	mathemhatthique	matematik, matematisien, machak, matine, matlasie, maternite, atletik, materialism	1
menopoz	menopause	menopoz, menase, menas, fenous, lepase, depase, debouse, enonse	1
menwizri	menuiserie	menwizri, menazer, menizie, menwizie, mezire, mennie, meni, metrize	1
mirakile	miraculeux	mirakile, mirak, rakle, metikile, miskle, sirkile, miskiler, mitile	1

mikroekonomik	mircoecnomique	mikroekonomik, makroekonomik, mikroskopik, ekonomik, ironic, fizionomi, minorite, indenmite	1
mitiel	mutuel	mitiel, metal, mortel, metie, meteo, mamel, miel, moul	1
nofraz	naufrage	nofraz, nofraze, namaz, paraz, maraz, laraz, garaz, baraz	1
nimerikman	numeicqumean	nimerikman, mankman, medikaman, anmemtan, inkleman, nevyeman, amerikenn, ekipman	1
obzeksion	objection	obzeksion, obsesion, obzervasion, obzektif, otozesion, sizesion, konzesion, zesion	1
okipasion	occupation	okipasion, opsion, operasion, konpasion, ovasion, osilasion, ovilasion, pasion	1
obsenn	oobscene	obsenn, obsenite, oubien, obsede, obsek, obsed, bouse, oubliye	1
payason	paillasion	payason, pasion, pasione, pasian, pasasion, passion, papion, varyasion	1
pasians	pattience	pasians, patin, patani, patine, pasian, patant, patang, latirans	1

povrete	pauvrete	pivote, povrete, pov, poste, porte, ponte, porter, pot	1/2
permanan	permannon	permanan, permanans, pirman, parman, permanganot, peyman, perdan, serman	1
farmasi	pharmachy	farmasi, farmasien, faraon, marmay, faranayt, facha, facho, maray	1
fertilizan	pheretillisan	fertilizan, retisans, partizan, partisipan, petision, apetisan, pertinan, etidian	1
fosforesan	phosphorescent	fosforesan, posesion, proporsion, prosenn, porsion, porslenn, popcorn	1
fotokopiyez	photocopieuse	fotokopiez, fotokopie, fotokopi, fotokopiy, otorize, prototip, ponpie,	1
pistas	pistache	paste, pistas, pistole, piston, piston, pitay, pitasie, pisar	1/2
plozib	plausible	plozib, plizir, plezir, plastik, labib, plati, lamib, lazil	1
plitar	plutard	plitar, plito, plakarde, oplitar, aplitar, palisad, pintad, pliyab	1
poligonn	polygone	poligonn, polone, plegne, polyo, ploye, golgote, portmone, pone	1

ponktiasion	ponctuation	ponktiasion, penetrasion, konpasion, onksion, fonksion, donasion, kotasion, lokasion	1
proporsionel	poporsionelle	personel, proporsionel, promosionel, porsion, pasione, proporsion, pozisionel, opsionel	1/2
posede	possédais	posede, posed, poset, poste, prosed, obsede, pouse, pede	1
kategori	qategory	kategori, kategoriz, kategorik, katori, katora, kateri, katorz, alegori	1
kalifikasion	qualiffiquastion	kalifikasion, klarisikasion, klasifikasion, gratifikasion, planifikasion, beatifikasion, ramifikasion, falsifikasion	1
katrovin	quatretropvin	katrovin, katrer, katrepeng, katrovindis, katrepis, kartron, katastrofik, troplin	1
konfidansialite	confeedanecsyalite	konfidansialite, konfidansiel, konfidans, kondansater, konfederasion, konferansie, konferans, kondanasion	1
kotidien	quotidien	kotidien, kotie, kretien, komedien, kontinie, kostin, kontiniel, soutien	1
realokasion	reallocaton	realokasion, realoke, realite, revoltan,	1

		reaplikasion, realizasion, realizater, realign	
rekapitilasion	recaputilasuon	rekapitilasion, kapitilasion, repitasion, reaplikasion, realokasion, reaparision, lapopilasion, realizasion	1
rekomandab	recommandable	rekomandab, rekomann, rekomandasion, rekomans, rekoumanse, rekondane, komandan, remarkab	1
rekonsiliasion	reconciliation	rekonsiliasion, konsiliasion, reabilitasion, kontiniasion, rezolision, revolision, enonsiasion, denonsiasion	1
rekeyir	recueilleir	rekeyir, rekeyi, re-elir, reouver, retenir, reseler, reveye, reysi	1
reanbarke	reembarquer	remorker, remarke,remark, remarkab, remarye, reanbarke, remagnder, reakter	1/6
reabilitasion	rehabilitation	reabilitasion, repitasion, rekapitilasion, reaplikasion, resitasion, realizasion, demobilizasion, afiliasion	1
relasionel	relatoinelle	relasionel, relwe, relanse, relans, relasman, resanble, latwal, reantere	1

renouvlab	renuvelable	renouvlab, renouvle, renouvab, resevab, revokab, reouver, renege, reveran	1
reperkision	repercussion	reperkision, represion, reparasion, repetision, repitasion, resepsion, rezervasion, resesion	1
repiblik	republique	repiblik, repiblikin, repibliye, replike, replik, redouble, repous, renouvle	1
residans	residence	rezidans, prezidans, retisans, residivis, reasirans, resan, resipian, evidans	1
rezireksion	ressurrecsion	rezireksion, restriksion, respirasion, resesion, rekreasion, resitasion, resepsion, repression	1
reversib	reversible	reversib, revers, rever, revri, reverifie, reveran, reveni, resezi	1
romantic	rhommemyanchique	roman, romantik, romans, mekanik, kominike, revandik, reanbarke, oseanik	1/2
grosierman	rossierement	rasiran, grosierman, revirman, forseman, sirman, fonsierman, ronfleman, posibleman	1/2

skandal	sacndal	skandal, sandal, sandalet, sandia, santral, vandal, sakal, sann-la	1
sengnman	saignement	sanzman, sengnman, serman, selman, sema, semans, fenean, eleman	1/2
sarkastik	sarcastheique	sarkastik, sase, saririk, saraspati, sartie, sarite, satanic, sarpantie	1
sarkastik	sarcastigre	sarkastik, sartie, sarpantie, santigrad, satire, satir, sansire, satisfe	1
skizofrenn	schizophrene	skizofrenn, skizofreni, sizron, choper, sipre, sikore, siprem, sipoze	1
skorpion	scorepyons	scorpion, sekresion, repons, respons, soping, koersion, morpion, koreksion	1
sedwizan	seduisant	sedwizan, sedision, sersan, simant, sezisman, servant, sermant, senkant	1
spesialite	sespecialliter	spesialite, sexialite, spesializ, spesialis, spektater, spektakiler, egaliter, sterilite	1
sketbord	skateboard	sketboard, debord, kestard, katar, katora, saybord, kategori, kanbar	1
skelet	skeleton	skelet, kolaton, selet, seleksion, selon, segon, keson, sezon	1

sipermarse	soupermarse	sipermarse, siprem, siperyer, siper, sipre, siperstisie, experyanse, serse	1
sinonim	synonyme	sinonim, sinon, segonnme, anonim, renome, denome, enonse, anonse	1
tabazis	tabbagejus	tabazis, tabazi, tapaz, tapazer, abaz, labaz, abazour, labazour	1
tablo	tabbleaux	tablo, tabla, tablet, tab, tabou, tal, gablou, talon	1
tanperatir	temperature	tanperatir, tanporer, tanpet, tanperaman, taper, tander, anpratik, anper	1
tilapia	titlapya	tilapia, titalber, titway, tilak, tirayta, tap, takya, tiray	1
trazektwar	trajectoir	trazektwar, trakter, trwazer, trazedi, teritwar, traver, rezervwar, razwar	1
tribunal	tridunal	tribunal, trin, trisoul, triang, trinke, orizinal, ringal, tina	1
iniversite	unnivairecithe	iniversite, iniversiter, anverite, liniversite, diversite, diversifie, nimerote, sinserite	1
iniversite	unniverrecithe	iniversite, iniversiter, anverite, liniversite, diversite, severite, nesesite, serenite	1
investisman	unvesstissment	investisman, investiser, linvestisman, anpesman,	1

		ouvertman, avertisman, flesisman, divertisman	
ourdou	urdou	ourdou, indou, egdou, endou, trou, sadou, ladou, dou	1
istansil	ustensyl	istansil, zetinsel, itansil, stenn, senay, steril, senser, stepoul	1
itilizasion	utilsatyon	itilizasion, osilasion, emilsion, ovilasion, imiliasion, afiliasion, mitilasion, fiksasion	1
vantilater	ventiylaliter	vantilater, vitalite, vantilasion, fertilite, itiliter, netralite, evantialite, vyabililite	1
voltize	voltiegr	voltize, volier, voler, vastier, voter, routier, goutier, mortier	1
zepinar	zepinard	zepinar, zepinn, zeping, zepi, seminar, pintad, zepis, zefir	1
dispanser	zyspensoeur	dispanser, senser, sinser, serser, senater, aseser, sper, lepeser	1

Appendix B: Progress Log

ANNEX 2

UNIVERSITY OF MAURITIUS

FACULTY ……FoICDT……..

PROGRESS LOG

Student Name	:	**Mungul Tushaar**
Student ID	:	**2013149**
Department	:	**Information and Communication Technologies**
Programme	:	**BSc (Hons) Computer Science**
Title of Dissertation	:	**A Spelling Checker for Kreol Morisien**
Supervisor	:	**Dr. Sameerchand Pudaruth**
Project Coordinator	:	**Dr. Jeetendranath Seetohul**

- Your Progress Log serves as a record of your transferable skills and participation and attainment as a student for dissertation purposes.
- Its purpose is to help you to plan your own dissertation and to record the outcomes.
- As well as gaining valuable skills, you will find that the information accumulated in this Log will prove helpful during the write up of the dissertation.
- The document belongs to you and it is your responsibility to keep it up to date.
- It is your responsibility to ensure your supervisor is aware of the dissertation activities you have undertaken.

You should sign the appropriate statement below when you submit your Progress Log:

I confirm that the information I have given in this Log is a true and accurate record:

Signed: …… Date: **25/07/2023**

PROGRESS LOG

RECORD OF STRATEGIC MEETINGS WITH SUPERVISOR(S)

Meetings	Date	Topics/ Themes Discussed	Comments (If any)	Supervisor's Initials	Student's Initials
1	01/12/22	Introduction to Project's topic		SP	T.M
2	19/01/23	Dataset - Data collection discussion	L d	SP	T.M
3	02/02/23	Brief description on literature review		SP	T.M
4	09/02/23	Analysis and looking for tools for development	I	SP	T.M
5	23/02/23	Feedback on literature review		SP	T.M
6	16/03/23	Discuss about implementation		SP	T.M
7	13/04/23	Feedback on Analysis		SP	T.M
8	20/04/23	Feedback on Implementation		SP	T.M
9	11/05/23	Discuss about evaluation	L N	SP	T.M
10	16/07/23	Feedback on Abstract and conclusion		SP	T.M

Supervisor(s)	Signature(s)	Date
Someerchand Pudaruth		24.07.23

N.B: Both the supervisor(s) and the student should retain a copy of this Project Progress Log.
A copy of the duly filled and signed Progress Log should be **included and submitted in the section 'Appendices'** of the Dissertation.

References

Ahmadzade, A. and Malekzadeh, S., 2021. Spell correction for azerbaijani language using deep neural networks. *arXiv preprint arXiv:2102.03218*.

Aho, A.V. and Corasick, M.J., 1975. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, *18*(6), pp.333-340.

Alpaydin, E. (2010) *The Knowledge Engineering Review*. 2nd ed. Edited by P. McBurney, S. Parsons, and P. Mannion. Cambridge University Press.

Anandarajan, M., Hill, C., Nolan, T., 2019. Text preprocessing. *Practical text analytics: Maximizing the value of text data*, pp.45-59.

Basri, S.B., Alfred, R. and On, C.K. (2012) 'Automatic spell checker for Malay blog', in *Proceedings - 2012 IEEE International Conference on Control System, Computing and Engineering, ICCSCE 2012*. IEEE Computer Society, pp. 506–510. Available at: https://doi.org/10.1109/ICCSCE.2012.6487198.

Bentf, S.W., Sleator, D.D. and Tarjant, R.E. (1985) *Biased Search Trees*. Available at: http://www.siam.org/journals/ojsa.php.

Bhaire, V.V., Jadhav, A.A., Pashte, P.A. and Magdum, P.G., 2015. Spell checker. *International Journal of Scientific and Research Publications*, *5*(4), pp.5-7.

Bhattacharyya, S., Hassanien, A.E., Gupta, D., Khanna, A. and Pan, I., 2018. International Conference on Innovative Computing and Communications. *Proceedings of ICICC*, *1*.

Bhatti, Z., Waqas, A., Ismaili, I.A., Hakro, D.N. and Soomro, W.J., 2014. Phonetic based soundex & shapeex algorithm for sindhi spell checker system. *arXiv preprint arXiv:1405.3033*.

Boicea, A., Radulescu, F. and Agapin, L.I., 2012, September. MongoDB vs Oracle--database comparison. In *2012 third international conference on emerging intelligent data and web technologies* (pp. 330-335). IEEE.

Carpooran, A. (2019) Diksioner Morisien. 3rd edn. Edition Le Printemps Ltée (ELP).

Crystal, D. (1997) *The Cambridge Encyclopedia of the English Language*. 2nd ed. Cambridge University Press.

Damerau, F.J., 1964. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, *7*(3), pp.171-176.

Data Flair, 2023. Advantages and Disadvantages of Machine Learning Language [WWW Document]. Data Flair. URL https://data-flair.training/blogs/advantages-and-disadvantages-of-machine-learning/ (accessed 5.10.23).

Denton, J.W. and Peace, A.G., 2003. Selection and use of MySQL in a database management course. *Journal of Information Systems Education*, *14*(4), p.401.

Parthasarathi, T., Geetha, R. and Dhanabalan, T.V., 2003. Tamil spell checker. In *Sixth Tamil Internet 2003 Conference*.

Engati Simply Intelligent, 2021. Statistical Language Modeling [WWW Document]. Engati Simply Intelligent. URL https://www.engati.com/glossary/statistical-language-modeling (accessed 5.9.23).

Faruque, A., 2020. Python Web Scraping Tools Compared [WWW Document]. Web Scraping. URL https://wscraper.com/python-web-scraping-tools-advantages-and-disadvantages-simply-explained/ (accessed 4.26.23).

Florigny, G. lf, Police-Michel Daniella and Carpooran Arnaud (2011) *Akademi Kreol Morisien*. Available at: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://education.govmu.org/Documents/educationsector/Documents/GRAMER%20KREOL%20MORISIEN%202211.pdf (accessed 4.7.23).

Ganfure, G.O. and Midekso, D., 2014. Design and implementation of morphology based spell checker. *Int J Sci Technol Res*, *3*(12), pp.118-125.

Haldar, R., Mukhopadhyay, D., 2011. Levenshtein distance technique in dictionary lookup methods: An improved approach [WWW Document]. URL https://arxiv.org/abs/1101.1232 (accessed 5.7.23).

Hládek, D., 2020. Staš J Pleva M. *Survey of automatic spelling correction Electronics*, *9*(10), p.1670.

Islam, M.Z., Uddin, M.N. and Khan, M., 2007. A light weight stemmer for Bengali and its Use in spelling Checker.

Jurafsky, D. and Martin, J.H. (2019) *Speech and Language Processing An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition Third Edition draft Summary of Contents*.

Khajeh-Saeed, A., Poole, S., Blair Perot, J., 2010. Acceleration of the Smith-Waterman algorithm using single and multiple graphics processors. J Comput Phys 229, 4247–4258. https://doi.org/10.1016/j.jcp.2010.02.009

Kukich, K., 1992. Techniques for automatically correcting words in text. *ACM computing surveys (CSUR)*, *24*(4), pp.377-439.

La Chronique de...Cronos (2016) Enn ti reflexion lor langaz ek kiltir kreol, lexpress.mu. Available at: https://lexpress.mu/blog/292638/enn-ti-reflexion-lor-langaz-ek-kiltirkreol (accessed 4.7.23).

Laux, Friedrich., 2015. DBKDA 2015 the Seventh International Conference on Advances in Databases, Knowledge, and Data Applications, Rome, Italy, May 24 - 29 2015. s. n.].

Mani, I., 2001. Automatic summarization. *Automatic Summarization*, pp.1-298.

Manning, C.D., Prabhakar, R., Schütze, H., 2009. Introduction to Information Retrieval.

Mays, E., Damerau, F.J. and Mercer, R.L., 1991. Context based spelling correction. *Information Processing & Management*, *27*(5), pp.517-522.

McGuffee, J.W. and Myers, D., 2015. Choosing scrapy. *Journal of Computing Sciences in Colleges*, *31*(1), pp.83-89.

Michelbacher, L., 2013. Multi-word tokenization for natural language processing.

Microsoft, 2023. Online spell checker - improve writing: Microsoft editor [WWW Document]. Microsoft . URL https://www.microsoft.com/en-us/microsoft-365/microsoft-editor/spell-checker (accessed 2.19.23).

Milliman, H., 2022. The importance of grammar checking in business communications [WWW Document]. BusinessWritingBlog. URL

https://www.businesswritingblog.com/business_writing/2022/05/the-importance-of-grammar-checking-in-business-communications.html (accessed 2.19.23).

Ministry of Finance and Economic Development, 2012. HOUSING AND POPULATION CENSUS.

Mishra, R. and Kaur, N., 2013. A survey of spelling error detection and correction techniques. *International Journal of Computer Trends and Technology*, *4*(3), pp.372-374.

Natchoo, N. (2021) *Kreol Morisien Grad 10*. Available at: https://fliphtml5.com/eisr/uigg/Kreol_Morisien-Grade_10/ (accessed: 7.4.23).

News, 2022. PM stresses on importance of Mauritian creole as binding force for unity [WWW Document]. URL https://pmo.govmu.org/News/SitePages/PM-stresses-on-importance-of-Mauritian-Creole-as-binding-force-for-unity.aspx (accessed 5.22.23).

Parmar, V.P. and Kumbharana, C.K., 2014. Study existing various phonetic algorithms and designing and development of a working model for the new developed algorithm and comparison by implementing it with existing algorithm (s). *International Journal of Computer Applications*, *98*(19), pp.45-49.

Pirinen, T., Pirinen, T.A. and Lindén, K. (2010) *Finite-State Spell-Checking with Weighted Language and Error Models : Building and Evaluating Spell-Checkers with Wikipedia as Corpus Finite-State Spell-Checking with Weighted Language and Error Models-Building and Evaluating Spell-Checkers with Wikipedia as Corpus*. Available at: http://hdl.handle.net/10138/29358.

Pollock, J.J. and Zamora, A., 1984. Automatic spelling correction in scientific and scholarly text. *Communications of the ACM*, *27*(4), pp.358-368.

Rastogi, K., 2022. Removing Punctuations [WWW Document]. Analytics Vidhya. URL https://www.analyticsvidhya.com/blog/2022/01/text-cleaning-methods-in-nlp/#:~:text=Removing%20Punctuations&text=The%20punctuation%20removal%20process%20will,process%20of%20removal%20of%20punctuations. (accessed 4.27.23).

Ray, S., 2019, February. A quick review of machine learning algorithms. In *2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon)* (pp. 35-39). IEEE.

Richardson, L., 2019. Beautiful Soup Documentation Release 4.4. 0. *Media. Readthedocs. Org*, pp.1-72.

Rungiah, M.C. (2022) *Kreolizasion Lingwistik Dan Kristianism: Propozision Enn Gloser An Kreol Morisien*. University of Mauritius.

Salifou, L. and Naroua, H., 2014. Design of a spell corrector for Hausa language. *international Journal of computational Linguistics (IJCL)*, *5*(2), pp.14-26.

Shaalan, K., Attia, M., Pecina, P., Samih, Y. and van Genabith, J., 2012, May. Arabic word generation and modelling for spell checking. In *LREC* (pp. 719-725).

Shannon, C.E., 1951. Prediction and entropy of printed English. *Bell system technical journal*, *30*(1), pp.50-64.

Sinclair, J., 1991. Corpus, concordance, collocation. *(No Title)*.

Sydenham, P.H. and Thorn, R. eds., 2005. *Handbook of measuring system design* (Vol. 3, pp. 901-908). Chichester, UK: John Wiley & Sons.

Uysal, A.K. and Gunal, S., 2014. The impact of preprocessing on text classification. *Information processing & management*, *50*(1), pp.104-112.

Uzzaman, N., Khan, M., 2004. A Bangla Phonetic Encoding for Better Spelling Suggestions.

Venditama, D., 2020. Levenshtein Distance [WWW Document]. Levenshtein Distance for Dummies. URL https://medium.com/analytics-vidhya/levenshtein-distance-for-dummies-dd9eb83d3e09 (accessed 2.25.23).

Verberne, S., 2002. Context-sensitive spell checking based on word trigram probabilities. *Unpublished master's thesis, University of Nijmegen*.

Vienney, S. ed., 2004. *Correction automatique: bilan et perspectives*. Presses Univ. Franche-Comté.

Wieds, G., 2007. Bioinformatics explained: Blast versus smith-waterman. *CLCBio.* *http://www.clcbio.com/index.php*.

Zhao, C. and Sahni, S., 2019. String correction using the Damerau-Levenshtein distance. *BMC bioinformatics*, *20*(11), pp.1-28.