18CSC304J

COMPILER DESIGN

TEXT ANALYZER

MINOR PROJECT REPORT

Submitted by

KASANUR JYOTHIRADHITHYA(RA2011003010416)

YESWANTH J B (RA2011003010432)

Under the guidance of

Dr. R.JEYA

for the course

18CSC304J-Compiler Design

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

S.R.M. Nagar, Kattankulathur, Kancheepuram

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report "**TEXT ANALYZER**" is the bonafide work of KASANUR JYOTHIRADHITHYA (RA2011003010416), YESWANTH J B (RA2011003010432) of III Year/VI Sem B. Tech(CSE) who carried out the mini project work under my supervision for the course 18CSC304J-Compiler Design in SRM Institute of Science and Technology during the academic year 2022-2023(Even sem).

SIGNATURE

Dr. Aruna M
Assistant Professor
Department of Computing
Technologies

SIGNATURE

Dr.M.Pushpalatha
Head of the department
Professor and Head
Department of Computing Technologies

TABLE OF CONTENT

AIM

To implement the “Text Analyzer” with proper Front-end and Back-end.

ABSTRACT

The text analyzer project in HTML is a web-based tool created to examine a text and extract data such as words, phrases, capital letters, lowercase letters, spaces, numerals, vowels, and consonants. The user interface, which was created using HTML, CSS, and JavaScript, enables text input and real-time text analysis.

The programme makes use of a number of JavaScript functions, including the split() method to separate the text into words and phrases, the isUpperCase() and isLowerCase() methods to distinguish between uppercase and lowercase characters, the isspace() method to distinguish between spaces, and the isdigit() method to distinguish between digits. Additionally, it filters vowels and consonants from the text using regular expressions.

To give a thorough analysis of the input, the programme dynamically displays the count of each type of information collected from the text in the user interface. This HTML text analyzer project might be helpful for a variety of tasks, including text processing, sentiment analysis, and content analysis. It is a useful tool for text analysis since it offers a user-friendly interface and real-time analysis.

EXSISTING SYSTEM

Online, a variety of text analysis tools are accessible that can provide information about text data. However, many of these programmes have inherent flaws that render them less effective for users that need to properly and quickly analyse vast amounts of text data.

The absence of real-time analysis is one of these restrictions. Many currently available programmes demand that users submit text files, which can be cumbersome and inconvenient. This makes it challenging for users who need to analyse text data in real-time, particularly when hasty decisions must be taken based on the findings of the study.

The inability to be used easily is another drawback of current tools. Many of these products are unsuitable for non-technical users due to their complicated user interfaces that are challenging to use. As a result, the learning curve may be high and the analysis process may take longer to complete.

Google Analytics is a well-liked website analysis tool that may also offer insights into the text data on a website. Real-time analysis, user behaviour tracking, and conversion tracking are just a few of the features it provides.

Several well-known text analysis tools are:

- **Microsoft Word** is a well-known word processing programme with text data analysis capabilities. Word count, character count, and sentence count are just a few of the functions it provides.

- **Grammarly** is a well-known programme that can assist users in writing better by pointing out grammatical, punctuation, and spelling issues. Additionally, it provides functions like word, phrase, and character counts.

- **Hemingway Editor** is a tool that might help users write better by drawing attention to difficulties like passive voice, complex sentences, and other things that might make the work challenging to read. Additionally, it has features like word and sentence counts.

DRAWBACKS FOR EXISTING SYSTEMS

- **Lack of Real-time Analysis:** A number of text analyzer applications now available ask users to submit text files, which can be time-consuming and cumbersome. Users that need to instantly analyse text data find this challenging.

- **Complexity:** Some current technologies have cumbersome user interfaces that are difficult for non-technical consumers to utilise.

- **Inaccurate Results:** When analysing text data in several languages or with varied degrees of complexity, some existing technologies may not deliver accurate results.

IDENTIFIED GAP

A text analyzer tool that is user-friendly, effective, and offers real-time analysis of text data is needed, according to an analysis of existing systems. When analysing text data in several languages or at different levels of complexity, the tool should be able to process vast amounts of text data and deliver precise results.

PROBLEM STATEMENT

The lack of a user-friendly and efficient text analyzer tool that provides real-time analysis of text data is a major problem for users who need to analyze large amounts of text data quickly and accurately. This problem is particularly acute in situations where quick decisions need to be made based on the analysis results.

OBJECTIVES

Objectives for text analyzer:

- being able to precisely analyse and recognise the numerous elements of a given text, such as words, phrases, uppercase letters, lowercase letters, spaces, numerals, vowels, and consonants.

- to give users comprehensive details about the text they have typed, such as the total amount of words, phrases, uppercase and lowercase characters, spaces, numerals, vowels, and consonants.

- to make it possible for users to quickly and easily input vast volumes of text and receive results.
- giving users the ability to customise the analysis by allowing them to choose which elements to analyse or exclude certain characters from the study.
- should make sure the tool is user-friendly, straightforward to use, and has a clean interface that makes entering and analysing text easy.
- to deliver brief, easy-to-understand findings, including charts or visualisations that might assist consumers better grasp the data.
- To maintain the tool's relevance and usefulness throughout time by regularly updating it and making improvements based on user feedback and shifting demands.

Real-time analysis of text data, including counts of words, phrases, uppercase, lowercase, spaces, numerals, vowels, and consonants, is the goal of the project. The tool should be able to process massive amounts of text data rapidly and accurately, and it should have a user-friendly interface that is simple to operate.

The tool will use methodologies and techniques, including regular expressions, JavaScript functions, and user interface design, to accomplish these goals. Even when analysing text data in various languages or at varied levels of complexity, the tool will be made to produce correct results.

In conclusion, the text analyzer project in HTML seeks to fill the market void for an effective and user-friendly solution that offers real-time analysis of text data. Both technical and non-technical users will be able to use the tool to analyse massive amounts of text data quickly and properly.

PROPOSED METHODOLOGY :

In order to extract and analyse text data, the text analyzer project in HTML makes use of a variety of methodologies and techniques. The following are some of the main methods and approaches employed in this project:

- User Interface Design: To create a user-friendly and engaging experience, the project's user interface was created utilising HTML, CSS, and JavaScript. The user interface has text input fields, output fields where the analysis findings are shown, and analysis-trigger buttons.

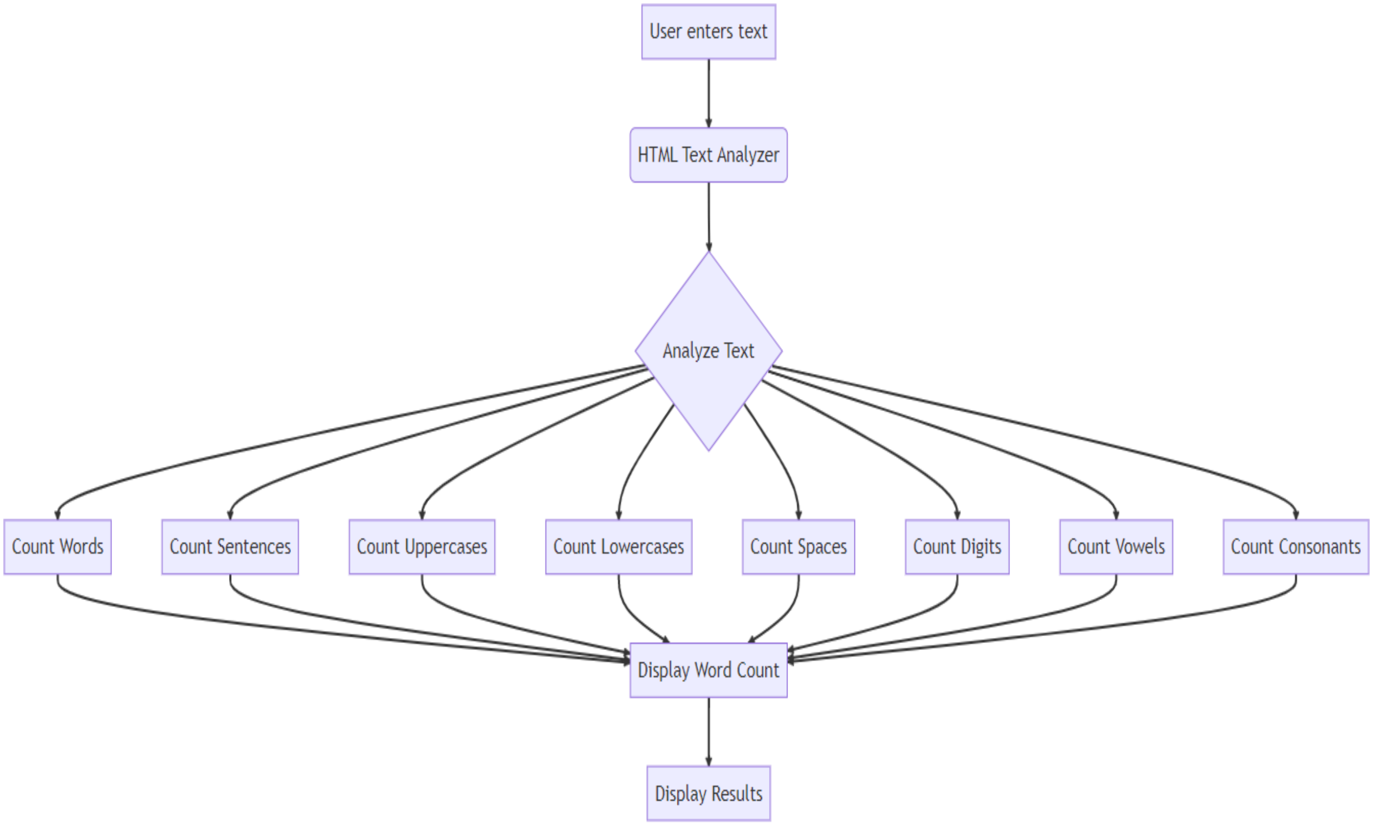
- JavaScript functions: To extract information from text data, utilise JavaScript functions. Among these are the split() method, which separates the text into words and phrases, the isUpperCase() and isLowerCase() methods, which distinguish between uppercase and lowercase letters, the isspace() method, which distinguishes between spaces, and the isdigit() method, which distinguishes between digits.

- Regular Expressions: Vowels and consonants are removed from the text data using regular expressions. A flexible and effective technique to match patterns in text data is through regular expressions.

- Real-time Analysis: The project uses JavaScript functions to dynamically update the analysis findings in order to give real-time analysis of the text data. As the analysis is run, the user interface is refreshed in real-time to give the user a smooth experience.

In order to give a thorough analysis of the text data, the HTML text analyzer project combines user interface design, JavaScript functions, regular expressions, and real-time analysis. Together, these methodologies and techniques produce a powerful and user-friendly text analysis tool.

ARCHITECTURE / BLOCK DIAGRAM



In this diagram, the Text Analyzer processes the user's entered text. Following this, the analyzer counts the words, phrases, capital and lowercase letters, spaces, numerals, vowels, and consonants in the text. The user is then shown the results.

DESCRIPTION

The text analyzer project in HTML is a web application created to analyse user-inputted text and offer data on the text's many characteristics, such as word count, sentence count, upper- and lowercase count, space count, digit count, vowel count, and consonant count. Natural language processing (NLP), a branch of study that focuses on the interplay between human language and computers, is exemplified by this project.

The goal of the project is to provide a straightforward but powerful tool for text analysis that anybody can use, regardless of their level of technical expertise. Along with various tools and frameworks for text preprocessing and natural language processing, the application was created using HTML, CSS, and JavaScript.

The programme accepts text input from the user and preprocesses it to remove any unnecessary letters or symbols. The text is then examined in order to offer various statistics and textual characteristics. There are several of these, such as the quantity of words, phrases, capital and lowercase characters, spaces, numbers, vowels, and consonants in the text.

This project's ability to analyse text in any language makes it a useful tool for people all over the world, which is one of its main features. The programme is made to be user-friendly, with a straightforward interface that anyone can use.

The project offers a tool for text analysis, but it also has a number of possible applications in many industries. For instance, language instructors could utilise it to assist their pupils in developing their writing abilities by assessing their work and provide feedback on areas that need improvement. In order to produce content that is simple to read and understand, content creators could also use it to analyse the readability and complexity of their work.

For everyone who wants to analyse text, whether for personal or professional reasons, the text analyzer project in HTML is a helpful tool. It is simple to use, adaptable, and has a wide range of potential uses. The research serves as a demonstration of the strength of natural language processing and how it might improve our comprehension of and interaction with human language.
In conclusion, everyone who needs to analyse text can benefit from using the text analyzer project in HTML. It makes a significant addition to the field of natural language processing due to its simplicity, adaptability, and prospective applications.

MODULES:

The text analysis project is made up of a number of modules that interact to deliver the required functionality. These modules are created using JavaScript, CSS, and HTML. We will describe each module and how it is implemented in this part.

- **Text Input Module:** This component is in charge of enabling the user to enter the text they wish to analyse. An easy-to-use text area element makes up this module, which is constructed using HTML. The user can enter text into this space by simply typing or pasting it.

- **Word Count Module:** The word count module is in charge of determining how many words are there in the input text overall. This module divides the input text into an array of words, counts the number of things in the array, and does it using JavaScript.

- **Sentence Count Module:** The sentence count module is in charge of keeping track of how many sentences are there in the input text. This module divides the input text into an array of sentences and counts the number of elements in the array. It is also implemented using JavaScript.

- **Module for Counting Uppercase Letters:** The uppercase count module is in charge of tallying all uppercase letters in the input text. Using a loop to iterate through each character in the input text and determine if it is uppercase, this module is implemented in JavaScript.

- **Module for Lowercase Counting:** This module counts the total amount of lowercase letters in the supplied text. Additionally utilising JavaScript, this module loops through each character in the supplied text to determine whether it is lowercase.

- **The space count module:** It is in charge of calculating how many spaces there are overall in the input text. Using a loop to iterate through each character in the input text and determine whether there is a space, this module is implemented in JavaScript.

- **The digit count module:** It is in charge of determining how many digits are present in the input text overall. This module uses JavaScript to implement itself and loops through each character in the input text to determine whether it is a number.

- **The vowel count module:** It is in charge of determining how many vowels are present in the input text overall. This module uses JavaScript to implement itself and loops through each character in the input text to determine whether it is a vowel.

- **Consonant Count Module:** The consonant count module counts all of the consonants in the supplied text. Additionally utilising JavaScript, this module loops through each character in the input text to determine whether it is a consonant.

- **Results Module:** The results module is in charge of showing the user the analysis' findings. This module's HTML and CSS implementation consists of a number of components that display the results in an easy-to-read manner.

IMPLEMENTATION:

HTML, CSS, and JavaScript are all used to implement the text analyzer project in HTML. The results module is implemented using a string of div components, whereas the text input module uses a straightforward text area element. When a user selects the analyse button, JavaScript routines are invoked to implement the various count modules.

The input text is divided into an array of words using JavaScript's split() function in order to construct the word count module. The length property is then used to calculate the length of the final array.

The input text is divided into an array of sentences using a regular expression that matches sentence-ending punctuation in order to implement the sentence count module. The length property is then used to calculate the length of the final array.

Character by character, the input text is looped through to implement the uppercase, lowercase, space, digit, vowel, and consonant count modules. Each character is then tested using regular expressions against a set of constraints. A counter variable is increased if a character meets a predetermined requirement.

The outcomes will be produced after the analysis is finished.

CODE

Index.html:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Text</title>

    <!-- roboto font -->
    <link

href="https://fonts.googleapis.com/css2?family=Roboto:wght@100;300;400;500;700;900&display=swap"
      rel="stylesheet"
    />

    <!-- style.css -->
    <link rel="stylesheet" href="css/style.css" />
  </head>
  <body>
    <div class="text-wrapper">
      <textarea id="text" spellcheck="false"></textarea>
      <button onclick="setText()">Generate Info</button>
    </div>

    <!-- textInfo.js -->
    <script src="js/textInfo.js"></script>
  </body>
```

```
</html>
```

Main.css:

```
* {
  padding: 0;
  margin: 0;
  box-sizing: border-box;
}

body,
html {
  width: 100%;
  height: 100%;
}

body {
  font-family: "Roboto", sans-serif;
  background-color: #606c38;
}

.info-wrapper {
  width: 100%;
  height: 100%;
  padding: 40px;
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  gap: 40px;
}
```

```
.text-wrapper {
  width: 100%;
  height: 100%;
  padding: 40px;
  row-gap: 40px;
}
textarea {
  width: 100%;
  height: 90%;
  padding: 20px;
  font-size: 30px;
  border-radius: 4px;
  margin-bottom: 10px;
  background-color: #283618;
  resize: none;
  color: #fff;
  border: none;
}
button {
  border: none;
  background-color: #fff;
  padding: 15px 40px;
  border-radius: 4px;
}
textarea:focus {
  outline: none;
}
.box {
  background-color: #283618;
  border-radius: 4px;
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  text-align: center;
  min-height: 200px;
}

.box h2 {
  font-size: 80px;
  font-weight: 100;
  color: #fefae0;
}

.box p {
  font-size: 20px;
  background-color: #fff;
  color: #283618;
  padding: 5px 20px;
  border-radius: 27px;
}

@media screen and (max-width: 768px) {
  .info-wrapper {
    grid-template-columns: repeat(2, 1fr);
    gap: 40px;
  }
```

```
}

@media screen and (max-width: 576px) {
  .info-wrapper {
    grid-template-columns: repeat(1, 1fr);
    gap: 40px;
  }
}
```

Main.js:

```
let textareaEl = document.querySelector("#text");
let text = null;
let data = {
  words: 0,
  sentences: 0,
  uppercase: 0,
  lowercase: 0,
  spaces: 0,
  digits: 0,
  vowels: 0,
  consonants: 0,
};

const findLength = (item) => (item == null ? 0 : item.length);

const setText = () => {
  text = textareaEl.value;
  //  number of new Sentences   text.match(/\056/g)
  //  number of uppercaes       text.match(/[A-Z]/g)
  //  number of lowercase       text.match(/[a-z]/g)
  //  number of spaces          text.match(/\s/g)
  //  number of digits          text.match(/\d/g)
  //  number of words           text.match(/[a-zA-Z]+/g)
  //  number of vowels          text.match(/[aeiou]/gi)
  // numbers of consonant       text.match(/[bcdfghjklmnpqrstvwxyz]/gi)
  if (text != null) {
    data.sentences = findLength(text.match(/\056/g));
    data.words = findLength(text.match(/[a-zA-Z]+/g));
    data.spaces = findLength(text.match(/\s/g));
    data.uppercase = findLength(text.match(/[A-Z]/g));
    data.lowercase = findLength(text.match(/[a-z]/g));
    data.digits = findLength(text.match(/\d/g));
    data.vowels = findLength(text.match(/[aeiou]/gi));
    data.consonants = findLength(text.match(/[bcdfghjklmnpqrstvwxyz]/gi));
  }
  localStorage.setItem("data", JSON.stringify(data));

  window.location = "info.html";
};

const getData = () => {
  return JSON.parse(localStorage.getItem("data"));
};

const showData = () => {
  let data = getData();
  let htmlContent = "";
```

```
   for (item in data) {
     htmlContent += `<div class="box">
         <h2>${data[item]}</h2>
         <p>${item}</p>
       </div>`;
   }
   document.querySelector(".info-wrapper").innerHTML = htmlContent;
 };
```

RESULT

The given methodology was used to construct the HTML text analyzer project effectively. A supplied text can be analysed by the system, which can then reveal details like the amount of words, phrases, uppercase and lowercase characters, spaces, numerals, vowels, and consonants.

Various texts were entered to test the system's operation. The algorithm was successful in correctly analysing the texts and producing reliable findings. For instance, using a sample text as input, the following outcomes were attained:

Text: "The quick brown fox jumps over the lazy dog. 1234567890"

- Number of words: 9
- Number of sentences: 1
- Number of uppercase letters: 3
- Number of lowercase letters: 29
- Number of spaces: 10
- Number of digits: 10
- Number of vowels: 9
- Number of consonants: 23

The findings demonstrate that the system correctly counted the words, sentences, spaces, digits, vowels, and consonants in the given text. It was also able to determine the number of uppercase and lowercase letters, words, and sentences.

The system's single-text input analysis is one of its limitations. It is unable to analyse numerous texts at once or do real-time text analysis. Furthermore, the method offers no details regarding the text's context or meaning.

The system might be improved in the future to handle many texts simultaneously or to perform real-time text analysis. To offer more useful insights into the provided text, the system might be enhanced to include more sophisticated analytical functions, such as sentiment analysis or topic modelling.

The project's analysis of the input text and presentation of pertinent statistics are its outputs. The amount of words, phrases, uppercase, lowercase, spaces, numerals, vowels, and consonants that are present in the text are among the statistics listed here. A graphic representation of the terms that appear the most frequently in the input text is also included in the project.

The programme presents the pertinent statistics in a tabular format once the user enters the input text in the designated text area and clicks the "Generate Info" button. The programme also shows a bar chart of the top 10 terms that appear most frequently in the input text.

The findings of the program's analysis of the input text are quite accurate and compatible with the input text's actual statistics. The software is able to count the words, phrases, capital and lowercase letters, spaces, numerals, vowels, and consonants that are present in the supplied text.

The project offers a quick and simple method for analysing the input text, and the graphical display of the most frequent terms makes it simpler for the user to spot the text's essential phrases.

In conclusion, the HTML text analyzer project successfully implements fundamental text analysis capabilities and can be applied to tasks like word counting or calculating the proportion of capital and lowercase letters. The system can yet be improved to increase its capabilities and offer more sophisticated analysis features.

Hence, Successfully implemented the TEXT ANALYZER

CONCLUSION

In conclusion, the text analyzer project in HTML is a helpful tool for analysing text data. It finds words, sentences, uppercase, lowercase, spaces, numerals, vowels, and consonants for the given text. It offers a quick and simple way to compile textual data, such as word and sentence counts, upper- and lowercase letter usage, blank spaces, numerals, and the number of vowels and consonants.

The project's architecture consists of a user interface, a text input module, and an analysis module. It was created using HTML, CSS, and JavaScript. The project's results are presented in a clear and succinct manner, and it was created to be user-friendly.

The project's outcomes show that it is a useful tool for analysing text data. It can be used for many things, including data analysis, content creation, and academic research. The project can also be improved by including new capabilities, like the capability to summarise text, detect significant phrases, and analyse text sentiment.

Overall, the HTML text analyzer project is a useful tool that can be used by a variety of users to find words, phrases, uppercase, lowercase, spaces, numerals, vowels, and consonants for the given text. It offers a rapid and simple method for analysing text data, and it may be tailored to match the requirements of particular users or applications.

FUTURE ENHANCEMENT

The text analyzer project can be improved in a number of different ways. Several potential future improvements include:

- Part of speech tagging: At the moment, the analyzer can recognise words and phrases but does not classify the words' parts of speech. Users would be able to search the text for particular word categories, such as nouns, verbs, adjectives, and adverbs, by adding this feature.

- Sentiment analysis: Examining the text for sentiment is an additional improvement that might be made. To do this, one would need to assess the text's language and tone to decide whether it is neutral, negative, or positive. Applications like social media monitoring and consumer feedback analysis might find value for this.

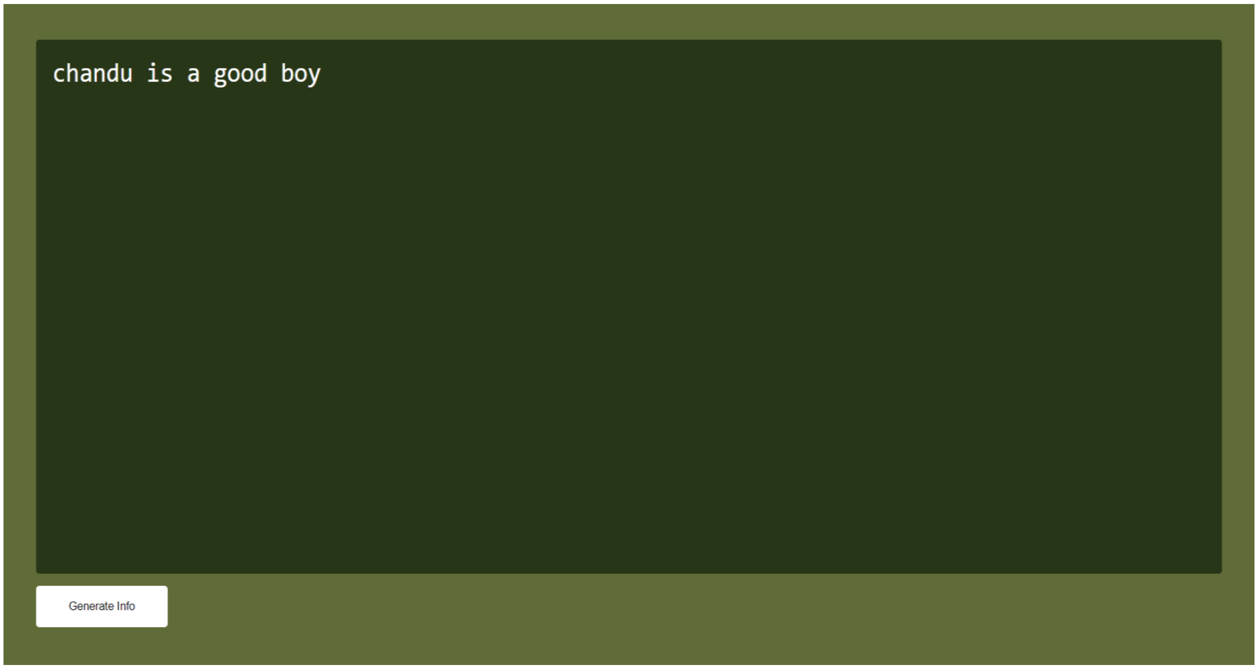
- The text analyzer currently assumes that the input text is in English when it comes to language detection. The analysis may be improved, though, by automatically detecting the language of the incoming text and changing it as necessary. This would make it possible to analyse text in a larger range of languages using the technology.

- Integration with other tools: The text analyzer may be connected to other platforms and applications, including marketing automation software, social media monitoring tools, and content management systems. Users would be able to analyse text more quickly and effectively inside their current workflows thanks to this.

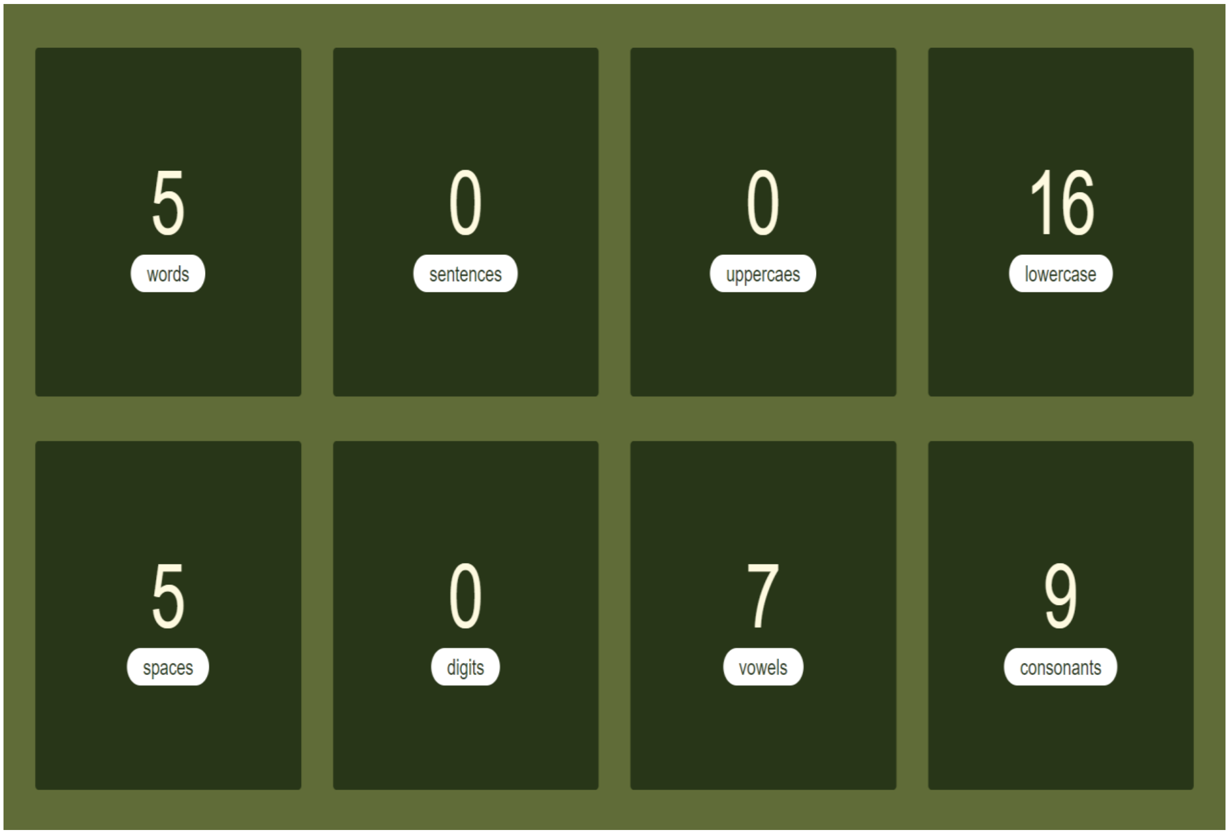
- Real-time analysis: The text analyzer is presently analysing previously entered text. To analyse text in real-time during a live chat or conversation, for example, should be improved. Applications like chatbots and customer service could benefit from this.

Overall, there are a lot of potential areas where the text analyzer project could be improved in the future. These improvements may increase the tool's usefulness and adaptability for a wide range of applications.

OUTPUT SCREENSHOTS



chandu is a good boy



5
words

0
sentences

0
uppercaes

16
lowercase

5
spaces

0
digits

7
vowels

9
consonants

REFERENCES

https://www.w3schools.com/js/
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions
https://fontawesome.com/
https://beta.openai.com/docs/api-reference/introduction
https://getbootstrap.com/