

# **B.Tech Project Report**

**ITIR32**

on

**Plagiarism detector**

**BY**

**Vikash Yadav(11913040)**

**Under the Supervision of**

**Mr. Mukesh Verma, Asst. Prof.**



**DEPARTMENT OF COMPUTER ENGINEERING  
NATIONAL INSTITUTE OF TECHNOLOGY  
KURUKSHETRA – 136119, HARYANA (INDIA)  
May, 2022**



## CERTIFICATE

I hereby certify that the work which is being presented in this B.Tech. Project report entitled in partial fulfillment of the requirements for the award of the **Bachelor of Technology in Computer Engineering** is an authentic record of our own work carried out during a period from January, 2022 to May, 2022 under the supervision of Mukesh Verma Sir ,Asst. Prof., Computer Engineering Department.

The matter presented in this project report has not been submitted for the award of any other degree elsewhere.

*Signature of Candidate*

**Vikash Yadav(11913040)**

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Date: 03.05.2022

*Signature of Supervisor*

**Mr Mukesh Verma , Asst.Prof.**

TABLE OF CONTENTS

Section	Title	Page No.
	Abstract	6
1	Introduction	7
1.1	Purpose	
1.2	Intended Audience	
1.3	Scope	
1.4	Rational	
2	Motivation	9
3	Literature Survey	10
4	Elicitation	13
4.1	Introduction	

4.2	Eliciting Requirements	
4.3	Collaborative Requirements Gathering	
4.4	Quality Function Deployment	
4.4.1	Normal Requirements	
4.4.2	Expected Requirements	
4.4.3	Exciting Requirements	
	Functional and Non-Functional Requirements	
4.5		
4.5.1	Functional Requirements	
	Non-Functional Requirements	
4.5.2		
	Usage Scenarios	
4.6		
4.7	Elicitation Work Product	
5.	Website Overview (Proposed Work) Working of Tika What is an API? How do APIs work? Home page How to run? How it works Folder Structure	17

6.	Results and Observations	34
7.	Conclusion and Future scope	35
8.	References	36

## Acronyms

<u>Acronym</u>	<u>Definition</u>
SRS	Software Requirement Specification
PDS	Plagiarism Detection System
QFD	Quality Function Deployment
ER	Entity Relationship
CRC	Class Responsibility Collaboration
DFD	Data Flow Diagram

## **Abstract**

This article discusses a trustworthy plagiarism detection system that is based on detecting copied content. Provides the functionality of auto checking document. Google search engine is used for querying. Pages returned are web scraped to extract meaningful info. Flask framework is used for process data. Along with the use of libraries function are created to process data and use it in proper way. API are created on cloud platform to use cloud services for translation and image text extraction. Potential copied content is mapped with its source URL, results are showed on web page.

## **Chapter 1: Introduction**

### **1.1 Purpose**

The Software Requirements Specification (SRS) for the Plagiarism Detection System is the document that is offered (PDS). It is the foundation of conditions for system development and encompasses comprehensive functional, non-functional, and support criteria. The SRS has discrete, distinctively numbered, and content-based groups of conditions. The SRS provides a common reference point for both the development platoon and the stakeholder community and acts as the functional method of expressing user requests to inventors. As users and creators collaborate to validate, define, and enhance SRS's content, SRS will get better over time.

### **1.2 Intended Audience**

This document is intended for several groups of users, including the customer, as well as the project manager, developers, and testers.

- The client will use this document to verify that the development team has created a product that is acceptable to the client.
- The development team's project manager will use this document to plan milestones and delivery dates and ensure that the development team is on track during system development.
- This document will serve as the designer's starting point while developing the system design.

To make sure that the system they are creating satisfies the client's needs, the designer will frequently refer back to this paper.

- The developer will use this document as a basis for developing system functionality. The developer will link the requirements defined in the document to the software they create to ensure that they have created software that meets all documented client requirements.
- This document will be used by the tester to create test plans and test cases for each requirement that has been documented. When portions of the software are finished, the tester runs his tests on those portions to make that the software complies with the specifications listed in this document. Once the system is finished, the tester will rerun his tests to make sure all the specifications outlined in this document have been met.

### 1.3 Scope

As a result of our effort, a program that can identify and assist in verifying text, article, or document duplication will be created. The task of document review is automated.



Fig:1 Plagiarism

### 1.4 Rational

Manual checking document has some problems. Some of these problems are –

- Students try to submit the same documents by changing the content or they can change only the title which is an unauthorized activity.
- It is very difficult for teachers to verify the similarity between documents and sometimes it causes wrong judgement.
- Sometimes students try to copy solutions from seniors.

This online plagiarism and application detection system will make it easy for teachers to stop these unauthorized activities.



## **Chapter2: Motivation**

Our main motivation for participating and working together on this project was to experiment with web scraping, use what we learnt to address any actual issues, and put a functional solution in place. The project mentor's consistent encouragement and specialised guidance eased the journey.

By assisting teachers in spotting plagiarism, a plagiarism detection system is essential to raising educational standards.

The most popular method for plagiarism detection tools is to apply a number of checks that are designed to find instances of plagiarism.

The web application uses the Flask framework to pre-process the raw data obtained through web scraping and displays associated filtered articles with appropriate semantics.

.

As a result of our effort, a program that can identify and assist in verifying text, article, or document duplication will be created.

The task of document review is automated.

## **Chapter 3: Literature Survey**

### **How it Works?**

It searches for matches between supplied text and already-existing texts using sophisticated database tools. It is used by the university to examine student work. Additionally, you can utilise professional plagiarism detectors to examine your own work before submission. In the background, plagiarism detectors index and crawl web information, looking for text that matches a database of previously published Internet content. The examination of keywords highlights exact matches. Inaccurate matches can also be found. The checker often provides the user with the proportion of plagiarism, highlights the plagiarism, and lists the sources.

### **Table of contents**

1. Differences between plagiarism checker.
2. What plagiarism checker can't identify.
3. Frequently asked questions about plagiarism.

## **Differences between plagiarism checkers**

Most of the plagiarism checkers operate similarly, there are some differences between them that affect the kinds of plagiarism they can detect. To help client choose the best tool, developers have done in-depth research to identify the best plagiarism checkers in 2022. Below are the key differences between checkers:-

### **Database size**

The database that each system has access to differs. It produces a wide range of outcomes. The databases of free programmes are frequently smaller. Large gaps in their ability to match emerge from this, especially for less readily available internet content. The ability to find a match is enhanced by the larger databases seen in the most effective applications.

### **Quality of scanning**

Scanning quality of software also varies widely. Many singles only recognize exact matches. If you paraphrased too closely or forgot to add a citation, these reviewers probably won't flag it. High-quality plagiarism checkers use a process called "fingerprinting" to find inaccurate matches between paraphrased or altered texts. Here, the software scans sentence fragments and looks for structural similarities. Like a real fingerprint, each fragment in the text should be completely unique and should not match fingerprints of existing documents. If there are matches, these ladies are able to identify them.

### **What it can't identify**

Although they are constantly evolving and improving, they cannot yet detect everything.

### **Ideas and non-text plagiarism**

If anything has been totally changed or translated, it can be challenging to detect plagiarism. They are also unable to assist with plagiarism that is not text-based, such as image plagiarism. Plagiarism of translated texts, concepts, ideas, images, or other non-textual content is still an issue and has the same repercussions as plagiarism that is more readily apparent.

**Text from internal databases**

Nearly every educational institution has its own internal database with uploaded works of current and previous students in addition to conventional plagiarism detectors. Normally, databases are not made available to outside parties. This means that their educational institution is likely to be the only one to find plagiarism from classmates. To better warn students of plagiarism and other forms of academic dishonesty, educational institutions may share their internal database with outside organisations. Self-plagiarism and submitting someone else's work as your own are both deemed plagiarism and have the same repercussions.

## **Chapter 4: Elicitation**

### **4.1 Introduction**

Elicitation aids in the client's definition of what is necessary. We had a lot of difficulties finishing it, including scope, volatility, and understanding issues. This is not a simple task, though. We used the requirements eliciting activities in an orderly and methodical manner to address relevant difficulties.

### **4.2 Eliciting Requirements**

Elicitation employs a specific format that incorporates aspects of problem-solving, elaboration, negotiation, and definition. To elicit requirements, a team of developers and end users must work together. Following our labour, we completed the requirements.

1. Collaborative requirements gathering
2. Quality function deployment
3. Usage scenarios
4. Elicitation work products

### **4.3 Collaborative Requirements Gathering**

Different approaches to gather collaborative requirements have been proposed. Each uses a slightly different scenario. For this we have done the following steps.

- Meetings were organized with our respected teacher. He was asked about his requirements and expectations from a plagiarism detection system.
- Teachers were asked about the problems they face with the current manual system.
- Finally, from the meetings with the teacher, we selected the final list of requirements.

## **4.4 Quality Function Deployment**

This technique translates client's needs into technical software requirements. Focuses on maximizing client satisfaction from the software engineering process. With respect to this project, the following requirements are identified in QFD.

### **4.4.1 Normal Requirements**

This consist of goals and objectives that are set when dealing with customers. Common requirements for this project are:

1. Accessible via the Internet.
2. Simple user interface.
3. Protection of user's personal data.
4. Allow all users to use.

### **4.4.2 Needed Requirements**

The requirements listed below may be so basic that the client does not expressly state them, but they are implicit to the system.

Unhappiness could be caused by their absence.

1. After all actions are finished, the database needs to clear the user-provided data.
2. The system ought to be accessible to all users.
3. The system's user interface must be simple to operate, and optional options should be used whenever possible in place of fields that demand data entry from the user.

### **4.4.3 Exciting Requirements**

These specifications speak of characteristics that, when offered, go above and beyond what the buyer might have reasonably expected.

1. For invalid input, the user interface should display the proper error messages.
2. The user interface should adhere to accepted web standards to ensure that it is compatible with common Internet apps.

## **4.5 Functional and Non-Functional Requirements**

All needs are divided into functional and non-functional categories.

### **4.5.1 Functional Requirements**

1. Web-based accessibility
2. Permit universal access.
3. For invalid input, the user interface should display the proper error warnings.

### **4.5.2 Non-Functional Requirements**

1. Simple layout.
2. The system will allow the user to use the services.
3. The system's user interface must be simple to operate, and optional options should be used whenever possible in place of fields that demand data entry from the user.
4. The user interface should follow standard web practices so that the web interface is consistent with typical Internet applications.

## **4.6 Usage Scenarios**

Any document type may be submitted. To submit a document, the user must first choose a file from their computer; if it is an image, they must click on the image input button, upload the picture, and

then wait for the results before pasting the copied text into the input text search section. After submission, the user will receive a report link on the document's legitimacy. Reports are available for download as PDF files for future reference, and if any copying is done, a resource link is also displayed.

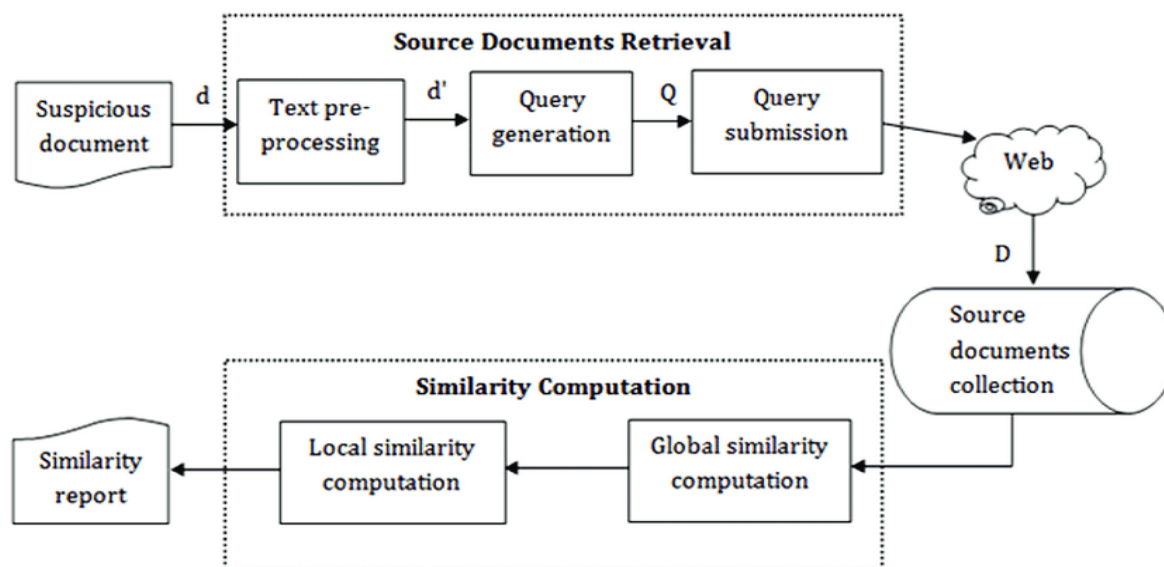


Fig: 2 Similarity report generation

## 4.7 Elicitation Work Product

The output of the elicitation task may vary depending on the size of the system or product to be built.

My elicitation work product includes:

- Make a statement about my requirements for the plagiarism detection system.
- Create a bounded scope statement for my system.
- Create a list of customers, users, and other stakeholders who participated in requirements elicitation.
- A set of usage scenarios.
- Description of the technical environment of the system



## Chapter 5: Website Overview (Proposed Work)

No login required .All users are allowed to use.Simple to use the user interface.

On the home page there are two sections one for pdf,txt,document and another one for text input.

Voice input is also available.Home page searches only english text ,no modification is done in input.Paragraphs are divided into sentences and then searched on google for each sentence.

Links are mapped and there frequencies are stored for providing optimal results. Top three links are showed based on their counts.Flask is used for backend work as it requires minimal libraries for working,if required any then can be easily imported.It is lightweight which increases efficiency of processing and executing tasks.

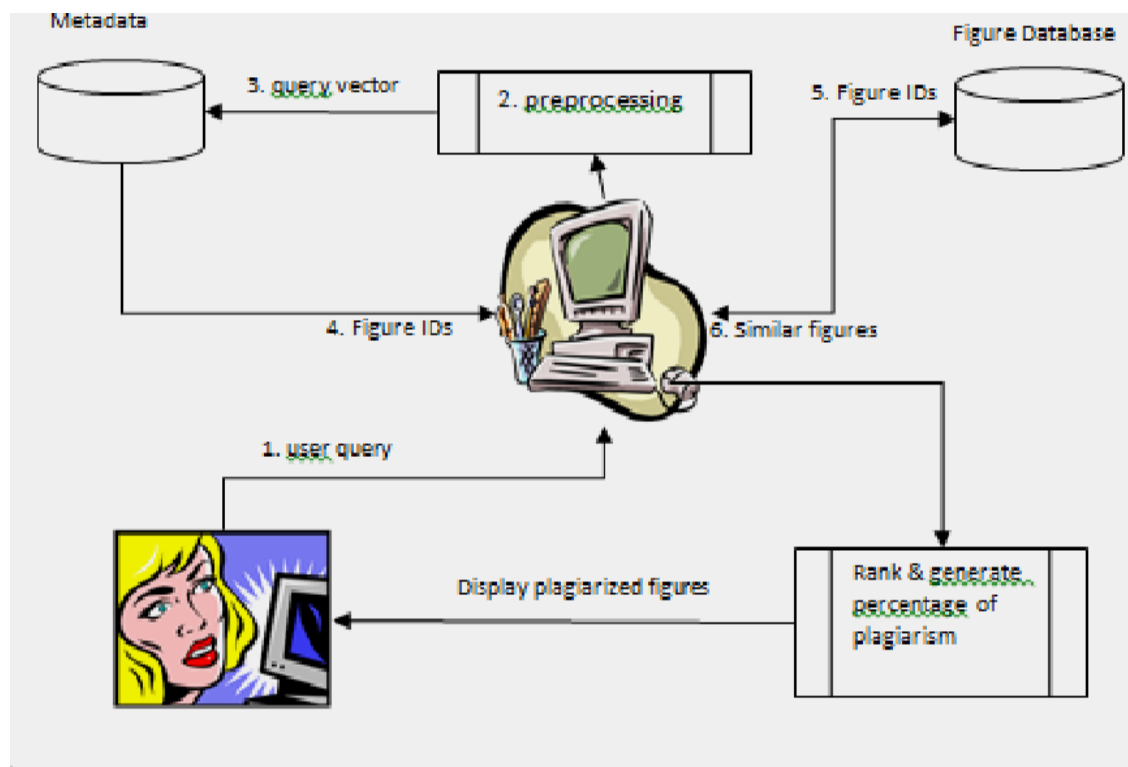


Fig:3 Flow of working

For different language search ,first request is sent to aws translate api for translation of text to required language then same process as of simple search is performed.

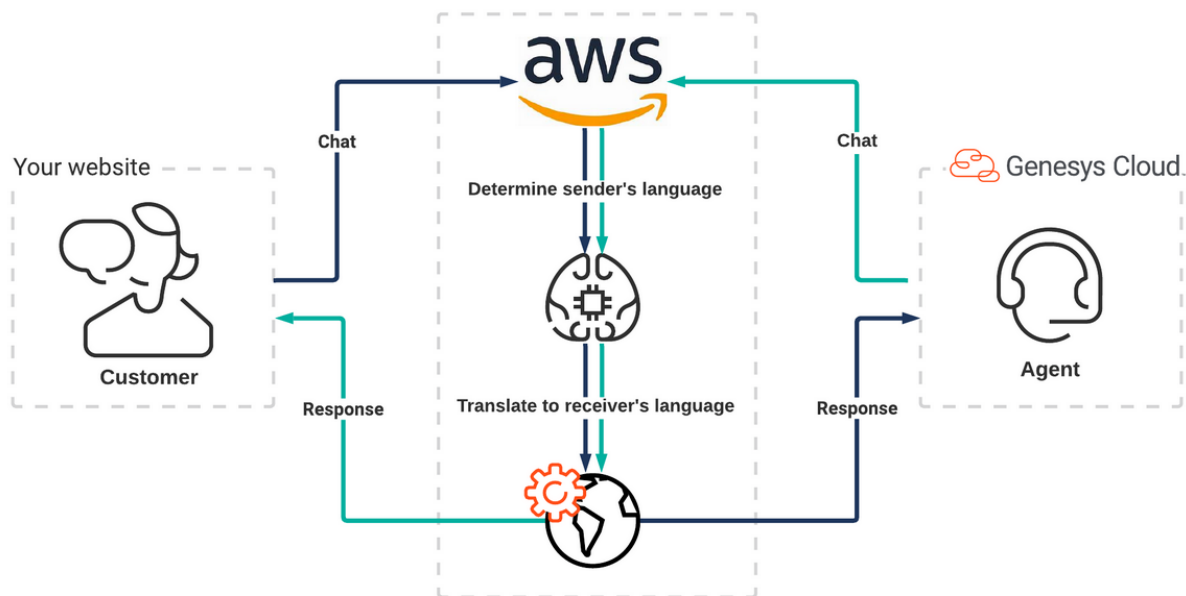
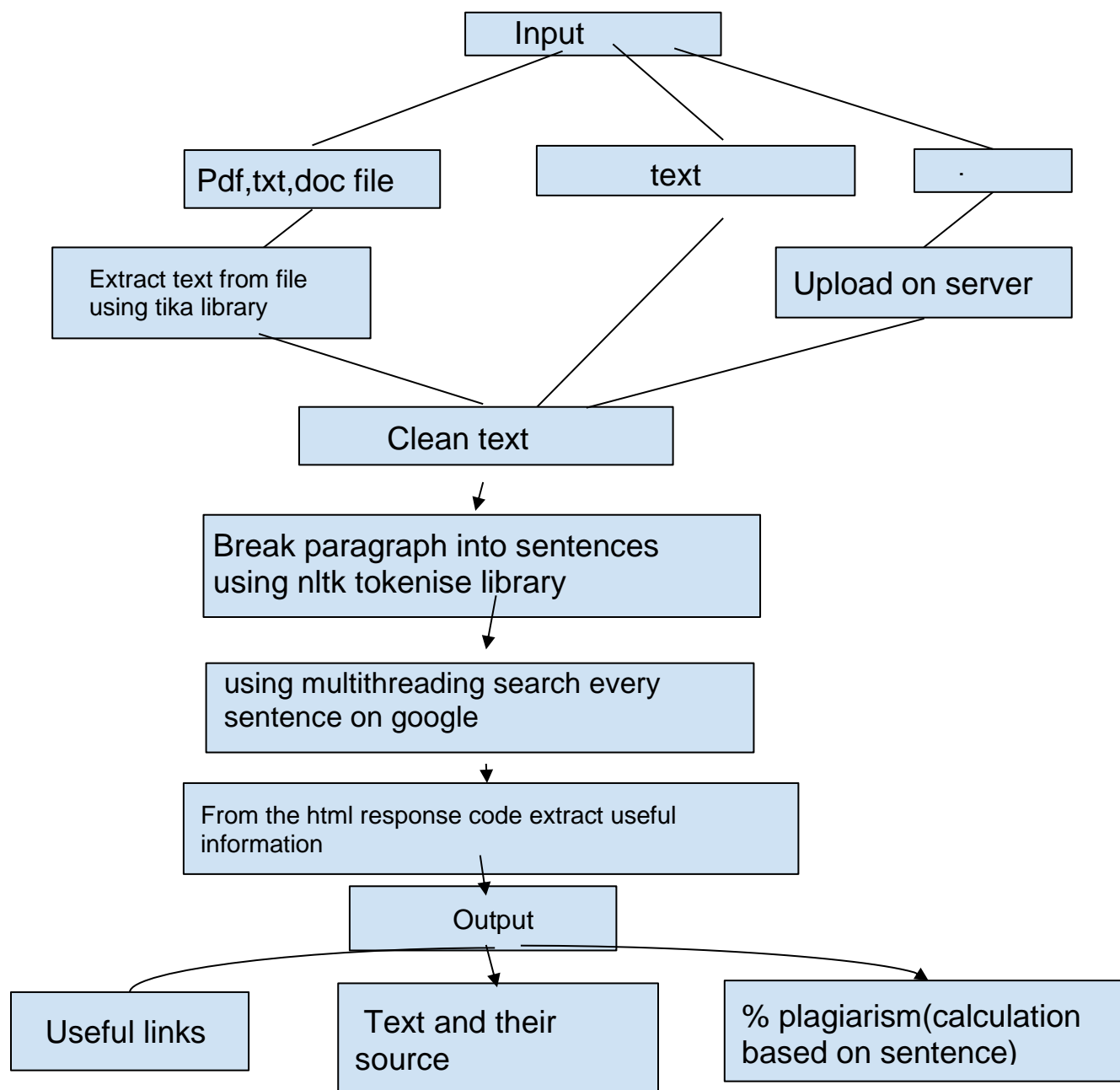


Fig:4 Aws translation api working

Direct different language can also be checked, it must follow selected language conditions.

Converting words to their synonym and then searching is in development phase. Different sentences will be formed from a single sentence then will be checked for any duplicacy.

Fig:5 Overview flow

## **How text extracted from document using Tika**

A library called Apache Tika can extract text from the majority of file types, including PDF, DOC, and PPT. Operating the library is simple with Tika's streamlined interface because it extracts the content. Its primary applications include translation, content analysis (for example, in journalism), and the indexing function in search engines (using paid APIs).

Metadata extraction is a component of the content analysis. The term "metadata" refers to data that describes a resource, in Tika's case, a file. The creation date, language, format, rights, subject, authors, title, and keywords are examples of data about the data. The examination of the file's content will be more precise the more metadata there are accessible.

Tika is a well-known Java library that is simple to use and is updated frequently.

It is therefore included into a variety of different programmes, like Apache Solr.

Since Tika is a FOSS (free and open-source software), you can add your own customised functionality using its flexible API.

The characteristics of Tika include content extraction as well as:

- Tika Server: provides access to its resources through the RESTful API, which is the focus of this article.
- Establishes the MIME type using the pattern type/subtype, for instance, image/png/.
- Identifies metadata, such as the metadata for a PDF, which is pdf:  
Language, doc: format, PDF Version, access permission, and Creation-Date (see more information below);
- Determines the text's language;
- Text translation: using a premium API from Lingo24, Google Cloud Translation, or Microsoft Translator Text.
- Tesseract OCR has been integrated to extract content from photos.

## **Advanced features**

Tika was initially just used for text extraction; more recently, it has been coupled with other libraries for more sophisticated usage (not covered in this article):

- For instance, computer vision can be used to create image captions.
- Integration of machine learning (ML) tools, such as TensorFlow and Mahout.
- Integration of natural language processing (NLP) tools like OpenNLP and NLTK.

## What is an API?

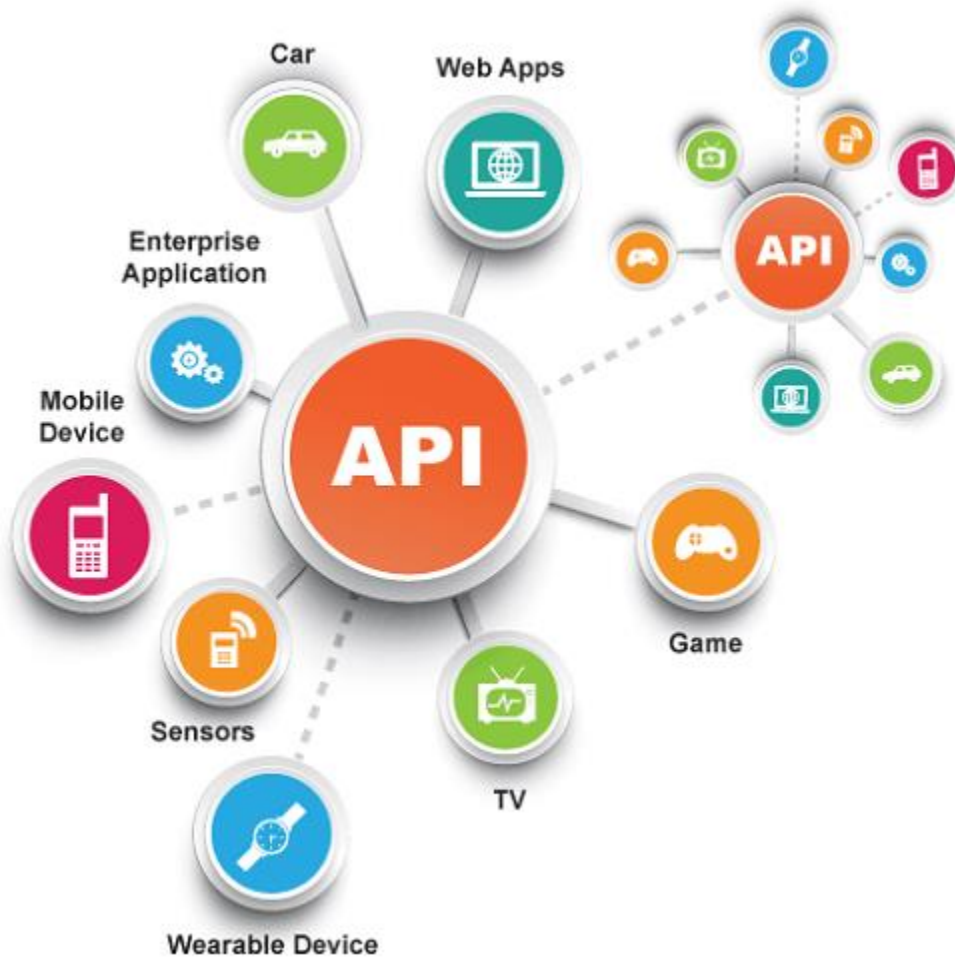


Fig:6 Uses of API

The use of a set of definitions and protocols to communicate between two software components is made possible by the use of APIs. The software system of the weather bureau, for instance, contains daily weather information. Your phone's weather app uses APIs to "speak" to this system and provides you with daily weather updates.

## **How do APIs work?**

Client and server architecture is typically defined in terms of APIs. Applications that transmit requests and responses are referred to as clients and servers, respectively. In the weather example, the mobile app is the client and the bureau's weather database is the server. Depending on when and why they were built, APIs can operate in one of four different ways.

### **SOAP APIs**

Simple Object Access Protocol is used by these APIs. XML is used by client and server to exchange messages. In the past, this more rigid API was more widely used.

### **RPC APIs**

Remote Procedure Calls are the name given to these APIs. A function (or operation) on the server is finished by the client, and the server then transmits the output back to the client.

### **Websocket APIs**

Another contemporary web API that uses JSON objects to convey data is the Websocket API. Client apps and the server can communicate in both directions using a WebSocket API. The server can communicate with connected clients via callback messages, making it more effective than REST API.

### **REST APIs**

These are the most widely used and adaptable APIs available right now online. Requests are sent to the server as data by the client. The server launches internal processes using this client input and sends the results back to the client. Here, we'll go through REST APIs in more detail.

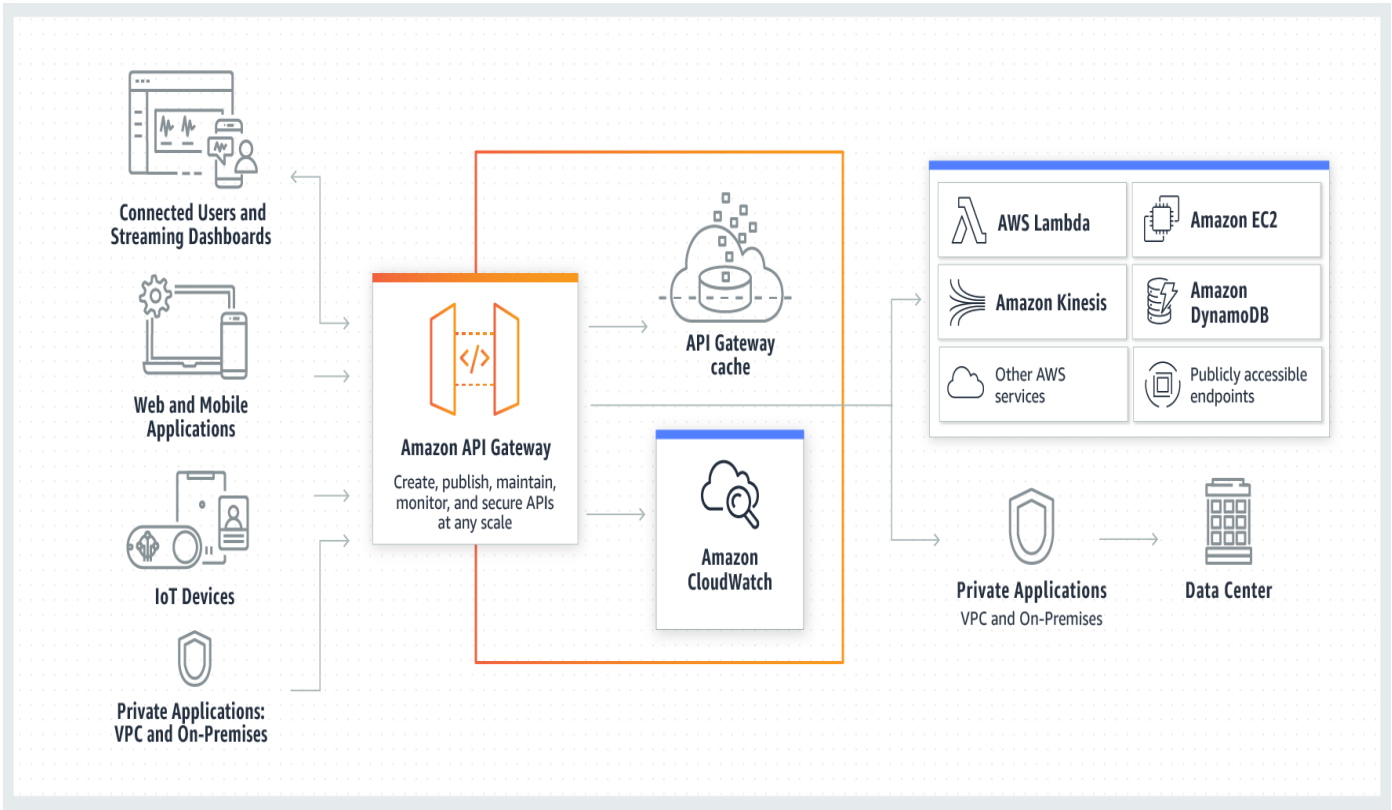


Fig: 7      Cloud Services



## Home page:

Click here for visiting: <https://plag-check-google.herokuapp.com/>

For file input click on browse and select file containing text only you want to check.

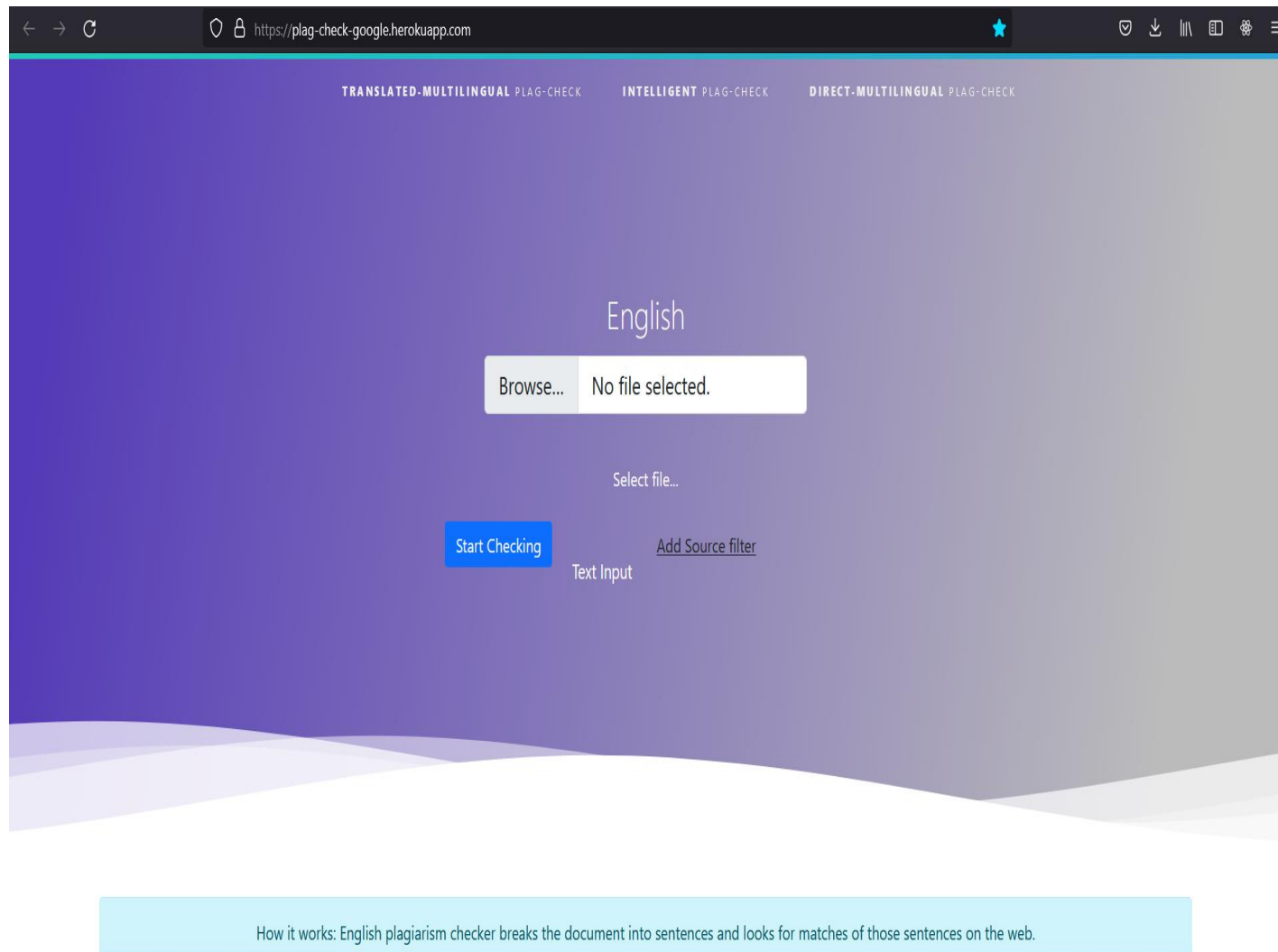



Fig:8 Home page

How it works: English plagiarism checker breaks the document into sentences and looks for matches of those sentences on the web.

Source filter URL (optional)

Enter the text

[For Image](#) [Submit](#) 

Press the mic

Fig:9 Text input

For text input copy it here ,if want to exclude a source add its link in the option given  
Click on mic for voice input.(Works on Chrome browser)

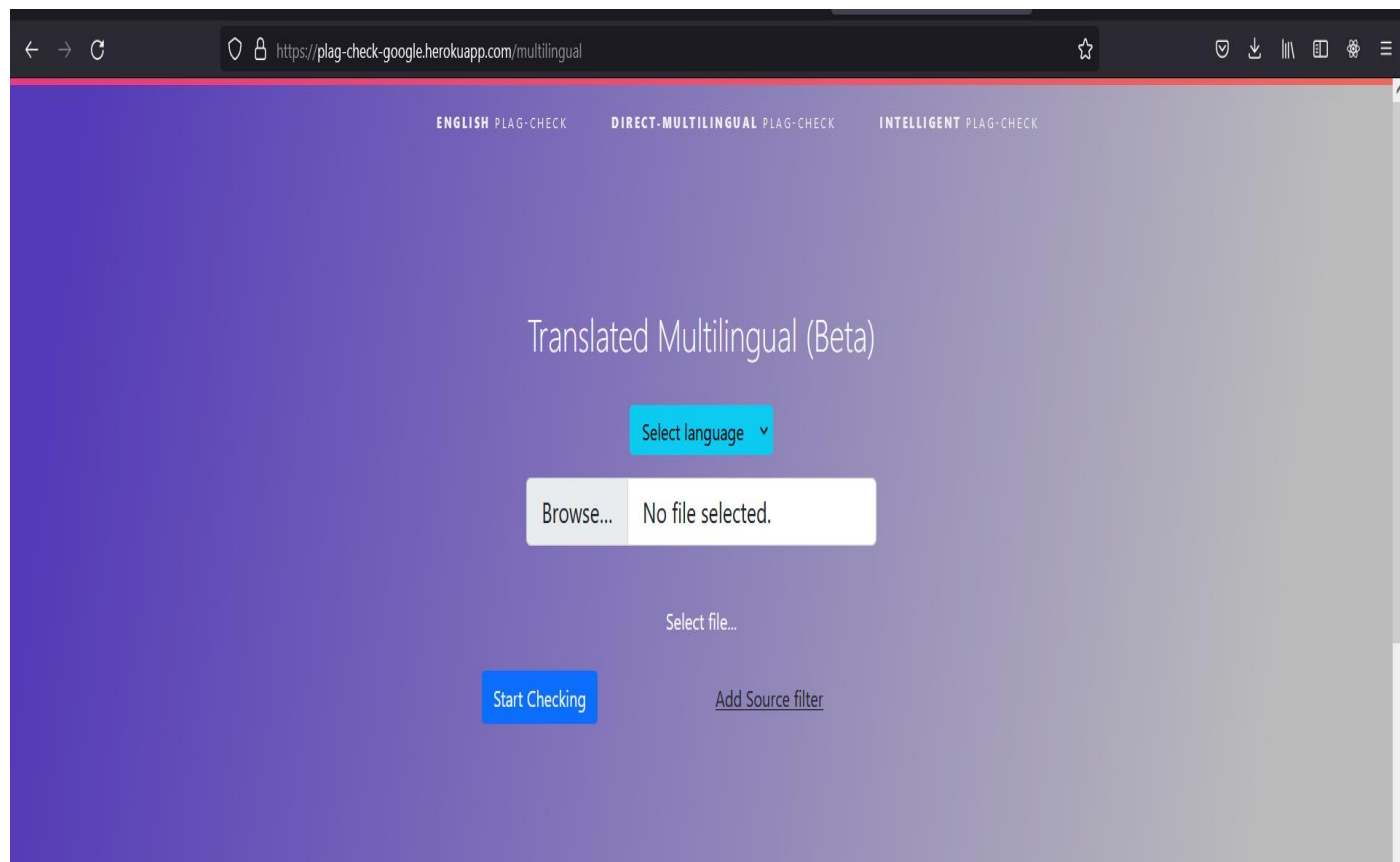


Fig:10 Select language along with file upload

Select text language for conversion to appropriate language ,search is performed in the selected language

The screenshot shows a web browser window with the URL <https://plag-check-google.herokuapp.com/multilingual>. The page has a light blue header with the text: "How it works: Translated multilingual plagiarism checker first translates the document into English and then check for plagiarism in the web". Below this, there is a section titled "Select a language" with a dropdown menu showing "Open this select menu". Underneath the dropdown is a text input field labeled "Source filter URL (optional)". A large text area labeled "Enter the text" is provided for input. At the bottom left, there is a "Submit" button and a microphone icon with the text "Press the mic" below it.

Fig:11 Direct check

Text is already provided in appropriate language, no conversion is done.

The screenshot shows the 'Direct Multilingual' section of the application. At the top, there are three tabs: "ENGLISH PLAG-CHECK", "TRANSLATED-MULTILINGUAL PLAG-CHECK" (which is active), and "INTELLIGENT PLAG-CHECK". The main heading is "Direct Multilingual". Below it is a "Select language" dropdown menu. A file selection area includes a "Browse..." button and a box stating "No file selected.". Below this is a "Select file..." label. At the bottom, there is a "Start Checking" button and a link labeled "Add Source filter".

Fig:12 Multilingual page

How it works: Direct multilingual plagiarism checker breaks the document into sentences and looks for matches of those sentences on the web.

Select a language

Open this select menu

Source filter URL (optional)

Enter the text

Submit

  
Press the mic

Fig:13      Multilingual text input

Words are changed to their most appropriate synonym in the text provide then search is performed

ENGLISH PLAG-CHECK   TRANSLATED-MULTILINGUAL PLAG-CHECK   DIRECT-MULTILINGUAL PLAG-CHECK

Intelligent (Beta)

Browse...

No file selected.

Select file...

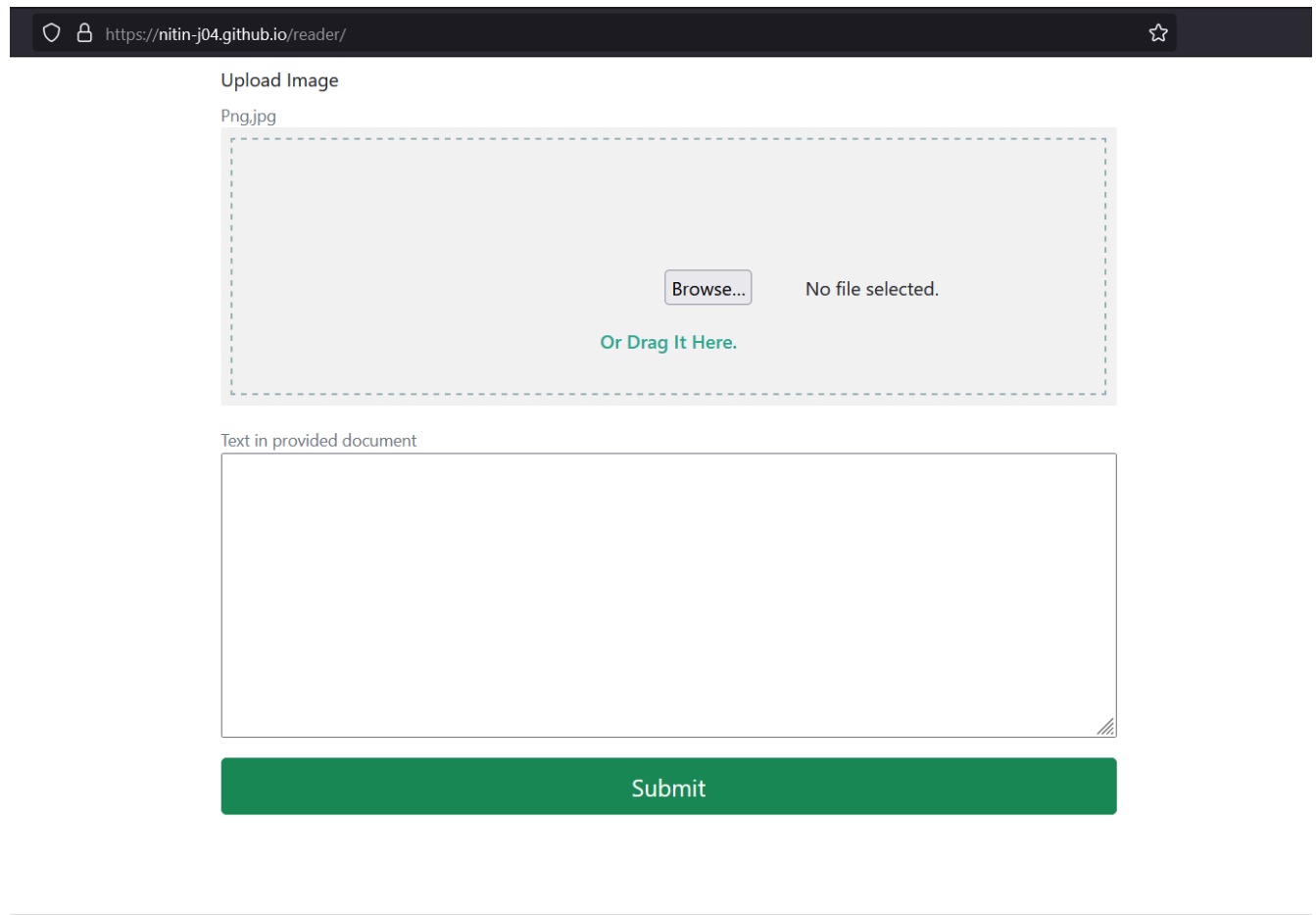
Start Checking

Add Source filter

Fig:14      Intelligent check

For image input :provide image by drag and drop or clicking on browse.Image is uploaded to aws s3 then request is sent to api created on aws for text.

Link for visit:<https://nitin-j04.github.io/reader/>



The screenshot shows a web browser window with the address bar displaying <https://nitin-j04.github.io/reader/>. The main content area is titled "Upload Image" and includes a file input area with a dashed border. Inside this area, there is a "Browse..." button, the text "No file selected.", and the instruction "Or Drag It Here." in green. Below the upload area, there is a section titled "Text in provided document" with a large, empty text area. At the bottom of the form is a green "Submit" button.

---

Fig:15 Text extractor

## How to run?

For source code(github):[https://github.com/nitin-j04/plag\\_check](https://github.com/nitin-j04/plag_check)

### To run the app

1. run command "pip install -r requirements.txt"
2. run command "python app.py" in the terminal
3. Open python console and execute the below code

```
import nltk,nltk.download('all')
```

**Note1 : Make sure that python 3.8+ is installed in your system.**

**Note2 : Make sure that Java is installed in your machine to use the tika package**

**Note3 : To use the Plag-Check, follow these steps :**

1. Get your API key and location to use the AWS translation API .
2. Create a .env file and write this code (replace "yourAPIkey" and "yourLocation" with the one that you obtained in step i.)

```
API_KEY=yourAPIkey
```

```
LOCATION=yourLocation
```

## How it works

- Based on the requirements there are four approaches to this project.
- It searches online using web scarapper designed to utilize Bing serarch engine for some queries. Queries are extracted from the source txt/pdf/odt/docx file.
- First approach works on English extracted data.
- Second approach (Tranlated-Multilingual) works on text data translated to English using AWS translator api and then translated data is fed to scrapper for match.
- Third approach works in a way extracted data is fed to a logic where synonyms are recognizied using NLTK Synset and then we replace the nouns and adjectives with its most commonly used synonym (obtained from NLTK brown corpus) after that matches are found using the same method.
- The fourth approach (Direct-Multilingual) directly looks for matches for the Multilingual data without any translation.
- Resulting URL, matched contents are checked for similarity with given text query.
- Finally the most probable sources are displayed on views with information showing plagiarised percentage and sentences.



## Folder Structure

- Static/ : Contains views functionality logics, script and styles.
- Templates/ : Contains html files for rendering on views.
- app.py : Main script file for running FLask application.
- modules.py : Consists of scrapper logic and required modules like tika logic for extracting text.
- requirements.txt: Contains required libraries ,install all these before running.

### Results/Observations/Future Scope:

Currently probability is calculated simply using the formula : (copied sentence)/(total sentence), to improve accuracy we will try to implement available algorithm like N-Gram or Jaccard or mixture of it to predict results accurately. Documents are not stored currently, creating and searching in our own database will be implemented in the near future.

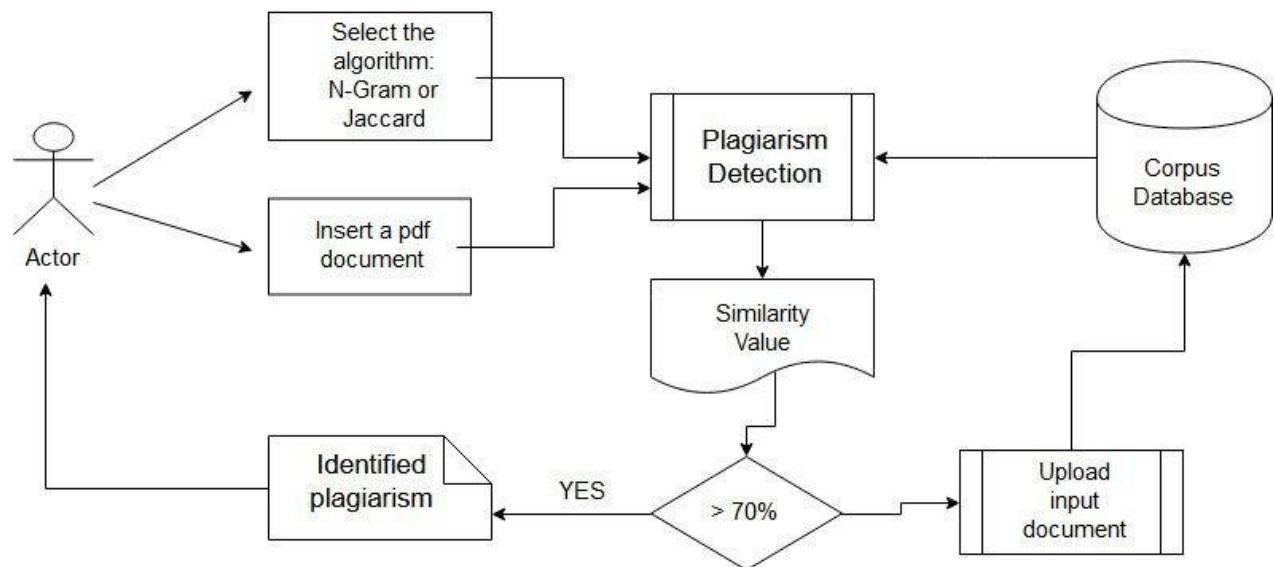


Fig:16 Use case diagram

## **Conclusion**

Plagiarism detection system is a much-needed system to stop the unfair activities by students. Through this document, readers will get a clear and easy view of plagiarism detection system. This SRS document can be effectively used to maintain the software development cycle. Management will be very easy for the entire project using this SRS. We tried my best to remove all dependencies and create an efficient and fully designed SRS. We believe the reader will find it alright.

## References:

### URLs

- <http://www.wikihow.com/Write-Software-Documentation> (Last Accessed: May 1, 2016)
- [https://en.wikipedia.org/wiki/Activity\\_diagram](https://en.wikipedia.org/wiki/Activity_diagram) (Last Accessed: May 1, 2016)
- [https://en.wikipedia.org/wiki/Use\\_Case\\_Diagram](https://en.wikipedia.org/wiki/Use_Case_Diagram) (Last Accessed: May 1, 2016)
- [https://en.wikipedia.org/wiki/State\\_diagram](https://en.wikipedia.org/wiki/State_diagram) (Last Accessed: May 1, 2016)
- <https://flask.palletsprojects.com/en/2.1.x/>
- <https://reactjs.org/>
- [https://en.wikipedia.org/wiki/Content\\_similarity\\_detection](https://en.wikipedia.org/wiki/Content_similarity_detection)
- <https://docs.aws.amazon.com/apigateway/latest/developerguide/http-api-develop.html>
- [https://docs.aws.amazon.com/translate/latest/dg/API\\_Reference.html](https://docs.aws.amazon.com/translate/latest/dg/API_Reference.html)
- <https://python-bloggers.com/2022/05/extract-text-from-image-using-python/>
- <https://www.python.org/>
- <https://betterprogramming.pub/deploy-your-app-for-free-in-7-easy-steps-thanks-to-heroku-dfd0f387edd0>
- <https://tika.apache.org/>
- <https://betterprogramming.pub/faster-web-scraping-in-python-using-multithreading-496da9eaf0c2>
- [https://en.wikipedia.org/wiki/Web\\_scraping](https://en.wikipedia.org/wiki/Web_scraping)