***Software Requirements Specification Document***

***Plagiarism Checker***

***Version 1.0 approved***

***Prepared by***

***Pavan R Kashyap (PES1UG20CS280)***

***Phaneesh R Katti (PES1UG20CS281)***

***Prajwal B S (PES1UG20CS287)***

***Pranav Kalwad (PES1UG20CS291)***

***Organization-***

***PES UNIVERSITY, BENGALURU- 61***

***Date created-***

***01/09/2022***

1. ***Introduction***

**1.1 Purpose:**

We often come under the ire of heat for having our work resemble something that’s already out there on the net. More often than not, it is not our fault. The lack of knowledge on the existence of other similar sources and the lack of availability of tools to check for plagiarism is the reason why we encounter accidental plagiarism (this problem). Therefore, the purpose of this project is to provide a tool/interphase to the user that allows him/her/them to upload their work and identify sources on the net that have similarities, so that they can minimise plagiarism in the work they publish/write.

**1.2 Intended Audience**

It is useful for anyone who is required to come up with ideas/thoughts of their own like students, document-writers, developers, researchers, etc. to make sure their “original” idea doesn’t already exist out there on the net.

**1.3 Product Scope**

Our product is intended to check for plagiarism practices in textual data, which helps original thinkers to avoid being plagiarized but also for others, for example teachers and professors to verify someone’s work to make sure that it hasn’t been plagiarized.

**1.4 References**

<https://towardsdatascience.com/build-a-plagiarism-checker-using-machine-learning-6538110ce162>

1. ***Overall Description***

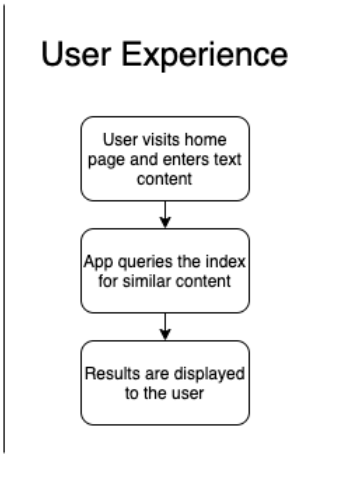
**2.1 Product Perspective**

The product at hand is new and self-contained. The software can be integrated into localised plagiarism checker platforms in educational institutions and research organisations. The integration can be done either by using links to the website or by uploading the software onto the organization’s cloud. The organization must have an Internet connection and/or localised cloud servers if they intend to include the software in their portals.

**2.2 Product Functions**

The product will perform the following functions-

* Allow the user to enter the text they intend to check for plagiarism on, in the text box provided. It also allows the users to upload files onto the platform to check for similarities on the net.
* A scale bar is displayed to the user on screen indicating the portion of similarity they want to check their documents on, with documents on. The user can move the scroll bar from 0 to 100 and all documents that have a similarity value greater than or equal to the current value set by the user, are displayed to the user.
* The user can look up the documents having similarities on the net and identify the changes they need to do in their document/ provide due credit to the authors of those documents.



**2.3 User Classes**

The most important user classes for this product are students, researchers and professors. They constitute the primary individuals who will be using this product the most.

Researchers primarily are going to use the tool/software to check for similarities in the research papers they are about to publish.

Students can use this tool to quote the sources they have obtained information from.It can also be used as a document engine to retrieve all documents that are present in the dataset , for a given set of sentences or words.

Professors can use this tool to verify student submissions and identify plagiarism.

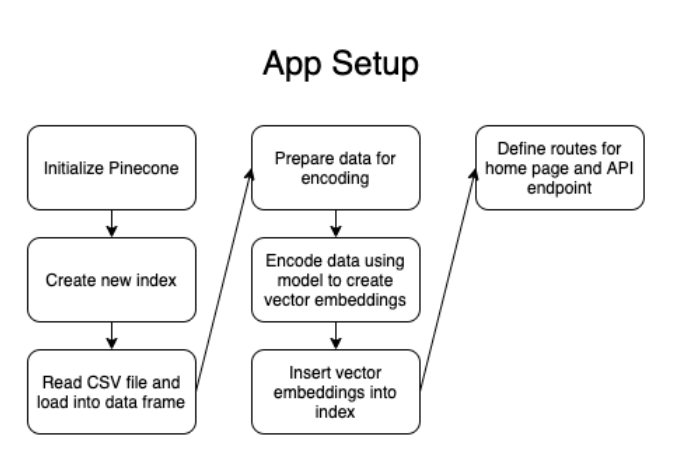
Professors may use this product more frequently than others to check for submissions and assignments, whilst research scholars may use it less frequently.

If the software is going to be integrated with a local plagiarism platform, then suitable security privileges can be provided( user parallelism,user authentication, logs of users accessing the tool).

**2.4 Operating Environment**

The product can be accessed by directly going to our website. Possession of our IP address/ URL and an Internet connection is sufficient to access and use our services. Our server will take in client entries , search the database/csv file that has been provided to it and display suitable outputs.

If the software is to be integrated to the cloud, then the cloud must have sufficient storage to hold the software (hold the files) that are uploaded by the user. The software will be stateful if integrated onto the cloud i.e it will be able to keep track of all the prior actions performed by users. In such a scenario, the cloud must be sufficiently large enough to store queries, documents and their results.



This displays the nature of how the software is set up at the backend so that it can provide suitable results on the front end.

**2.5 Design and Implementation Constraints**

Some of the constraints that might limit the use of this product to the customers are

1. Insufficient cloud storage-- Managing the state of software will become limited if sufficient memory is not allocated in the cloud storage space.
2. Entities outside our dataset-- The dataset that is being used is a subset of all the publications and documents that are available on the Internet. Expansion onto the actual net can be considered.
3. Document publishing timings -- There is no lower bound fixed on the latest documents on the web that the software/product must retrieve and show. This may be a constraint the client can impose, and discussion on its accommodation will be carried out in subsequent SCRUM meetings.
4. Expanding the document range-- The product at hand is currently restricted to documents that find an entry in the dataset. If the user asks to expand our search base to include other published work that is not readily accessible on the net( like novels, eBooks, older book versions),then those changes have to be brought about.

**2.6 Assumptions and Dependencies**

The app dependencies are as follows

dotenv for reading environment variables from the .env file

flask for the web application setup

json for working with JSON

os also for getting environment variables

pandas for working with the dataset

pinecone for working with the Pinecone SDK

re for working with regular expressions (RegEx)

requests for making API requests to download our dataset

statistics for some handy stats methods

sentence\_transformers for our embedding model

swifter for working with the pandas dataframe

***3. External Interface Requirements***

**3.1 User Interface**

The application will have a simple and easy-to-use user interface with which the user will interact and upload the files or enter the text to check for plagiarism. There are 3 buttons present on the screen. Once the user presses the RUN button then we internally run our program which takes the users file or the text as the input. The UPLOAD FILE button is provided to the user to input the files from the local system. The EXIT button is provided for the user to exit the application if he/she wishes to do so. There is also a text field provided so that the user can type in the text here.

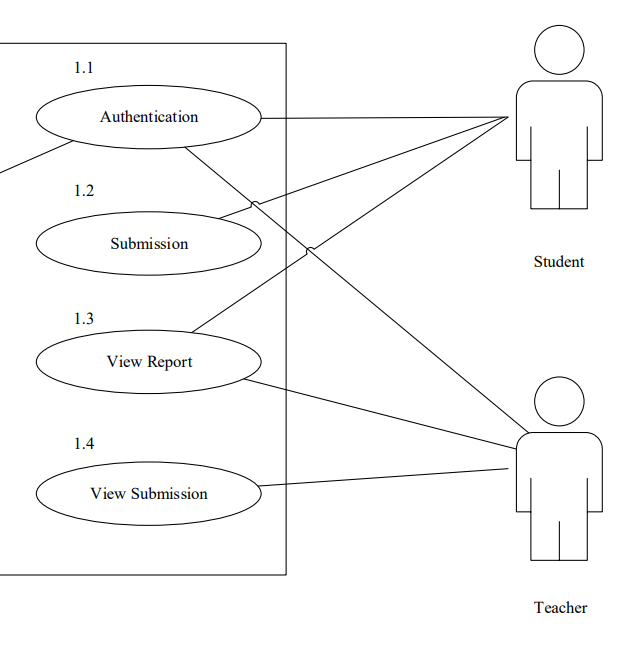
**3.2 Software Interface**

We are building this application using Python3. We are also using inbuilt libraries and packages of python which enable us to speed up the process of development of this application. We make use of the Python Flask app that uses Pinecone - a similarity search service -to find possibly plagiarized content. The frontend part of this application is built using html , css and static javascript.

**3.3 Communication Interface**

The communication interface mainly consists of the user interface through which the user interacts with the software application in the backend. The user interface is the only part of the application which is exposed to the user to communicate with the application. The functionality of the application is completely hidden from the user.

***4. Analysis Models***

******

*Use case diagram for student, teacher in a localised interphase.*

***5. System Features***

1. **Detection of Plagiarism**

Check whether the document has been copied from other documents or from any sources present in the dataset provided. It is a high priority requirement as it is our product’s main motive!

**User Actions:**

* Users can upload a file/ manually input text to check whether it has been copied from any sources present in the dataset.
* Suitable authentication can be provided to ensure that the professor/ teacher has more security actions and privileges than a student on the integrated plagiarism tool.
* In scenarios where the user uploads corrupted/ irrelevant (ex: completely numeric files) files to the checker, the checker must be capable of rejecting such files (it must not be stored in the state/cloud).

***6. Other Non-functional components***:

**6.1 Performance requirements**

Network connection may be required to check for plagiarism over the net. Minimum computer specs:

* 2 GB available RAM
* Sufficient space and basic server resources to store files and perform plagiarism check

(To be updated as per Scrum review meetings)

**6.2 Safety Requirements:**

Validated Backup of files (that are to be checked), to avoid potential data loss under unforeseen circumstances. Any files uploaded/used in the software will be checked for potential threats and appropriate actions will be taken as needed

(To be updated through Scrum review meetings)

**6.3 Software Quality Attributes:**

The software will be open to modifications as per the general user consensus. It is intended to be flexible, adaptable, portable, reusable, robust and easily maintainable

(To be updated through Scrum review meetings)

**6.4 Business Rules:**

(To be updated through Scrum review meetings)

***7. Other Requirements:***

For integrating our product as a part of the organization website, database access and Server shared resources must be made available for software use

Diagram, shape

Description automatically generated