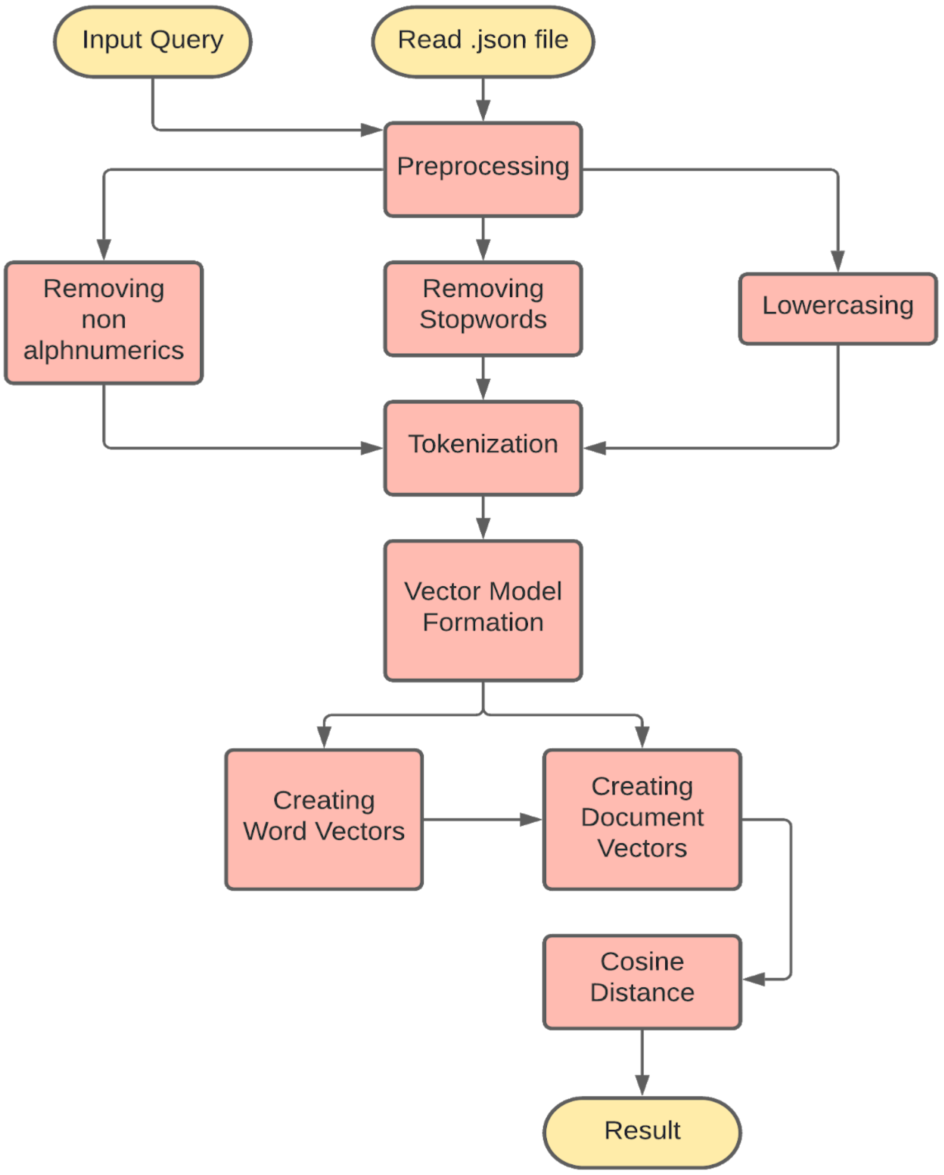
Design Documentation

Preprocessing on the .json file was performed and separate .json files were created for each entry. The list data structure has been used to store the tokenized words of each document and further each list is a list of lists.



DATA STRUCTURES USED:

Document_tokens_list

Contains lists enclosed within a list It will contain the stemmed tokens from each file in the corpus as individual lists. All are appended to make a list. Example:

```
[[‘i’,’play’,’cricket’],[‘sachin’,’tendulkar’],[‘india’,’is’,’best’]]
```

Vocabulary

Will contain a dictionary of all the unique words in the corpus. Example:

```
{‘i’: 1, ‘play’:2, ‘cricket’:3, ‘sachin’:4, ‘tendulkar’ :5, ‘india’:6 ,
‘is’ :7, ‘best’:8]
```

Prime Dictionary

A nested dictionary containing the following structure explained through the following example:(Numbers are just representational)

```
{‘i’:{‘0’: {‘1’:1 ,“2”:0.8, ‘3’ : 0.8} , ‘1’:{‘1’: 2 ,‘2’: 0.4,
‘3’:0.8}, ‘2’:{‘1’: 0 ,‘2’: 0.78, ‘3’:0.8}} , ‘cricket’ :{‘0’: {‘1’:2
,“2”:0.6, ‘3’ :1.2} , ‘1’:{‘1’: 0 ,‘2’: 0.4, ‘3’:1.2}, ‘2’:{ ‘1’:
1,‘2’: 0.4, ‘3’:1.2}}}
```

Index ‘1’ refers to TF

Index ‘2’ refers to IDF

Index ‘3’ refers to TF-IDF

Scores

A dictionary which will contain the scores of the documents after inputting the query and running cosine similarity algorithm. Example :

```
{‘0’: 0.2323 , ‘1’: 0.3125 , ‘2’ : 0.467 }
```

Index ‘0’, ‘1’, ‘2’ etc refer to document numbers

Creating the GUI

Flask Framework V-1.0.2 has been used to create the GUI. It is a web application framework written in Python. It contains boilerplate code consisting of HTML, CSS and bootstrap files for easy front-end development.

On the homepage, the user can search for the books using the title, author's name or even categories using the search box. Top 10 most relevant quotes along with their authors will be displayed over the next page.

Runtime for creating the vector space model - Approx 20 seconds
Runtime for returning query results - Around 2 seconds