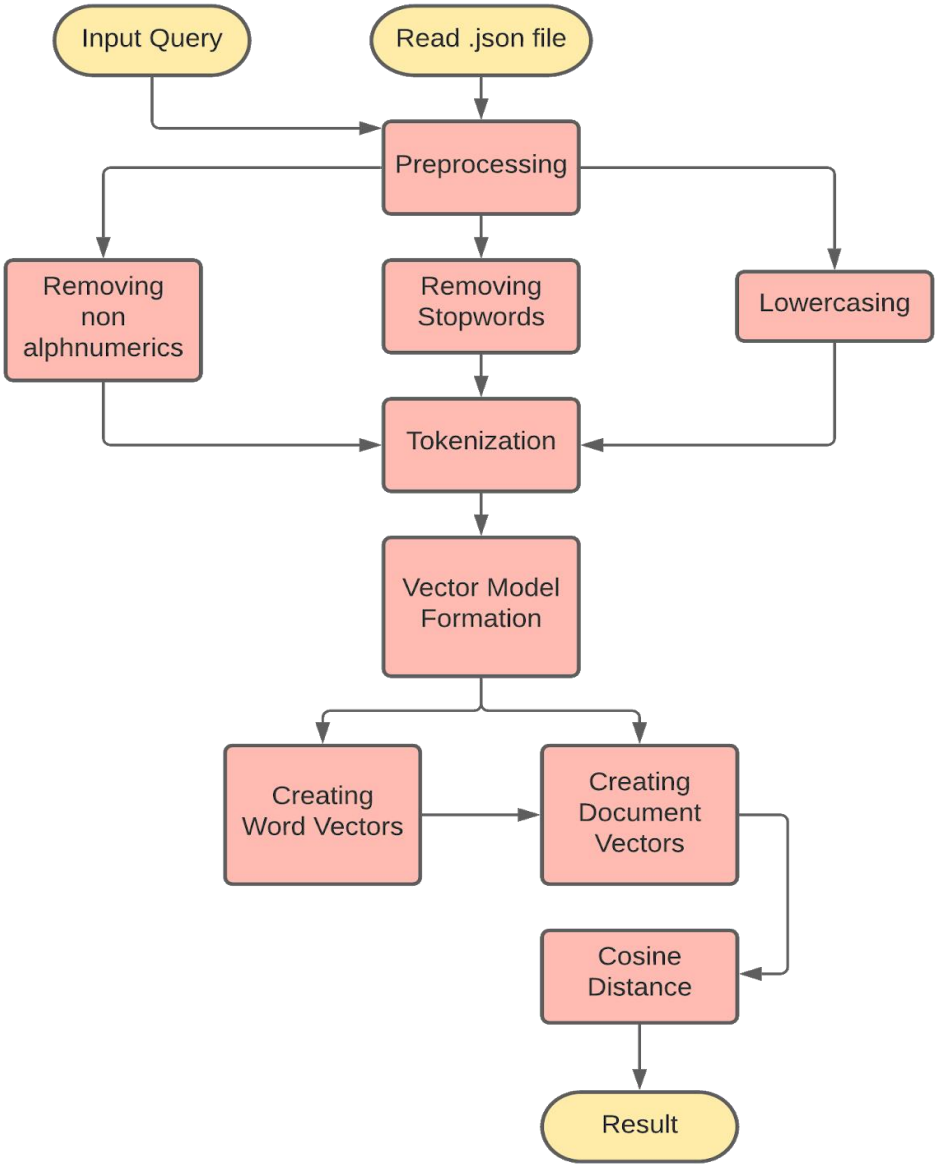
# Design Documentation




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:

This json file contains lists which are further enclosed within a list. This file contains the stemmed tokens, which are present in each file or document in our corpus, as individual lists. All these token are appended or added together to make a list. Example:

[["aaron", "watson", "angel", "&", "outlaw", "honki", "tonk", "rollercoast", "ride"], ["aaron", "watson", "angel", "&", "outlaw", "honki", "tonk", "sweet", "contradict"]]

## Vocabulary

This json file contains all the unique words which are present in our corpus. Example:


```
{"aaron": -1, "watson": 0, "angel": 1, "&": 2, "outlaw": 3, "honki": 4, "tonk": 5,}
```


## Prime Dictionary

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


```
{"aaron": {"0": {"1": 0.15200309344505006, "2": 7.98584193700334, "3": 1.2138726781877183}, "1": {"1": 0.15200309344505006, "2": 7.98584193700334, "3": 1.2138726781877183}, "2": {"1": 0.15200309344505006, "2": 7.98584193700334, "3": 1.2138726781877183}}
```


*Index '1' refers to TF*

*Index '2' refers to IDF*

*Index '3' refers to TF-*

*IDF*

## Scores:

This json file A contains the scores of the documents after the user has given or inputted the query and cosine similarity algorithm has been run to give score to the document w.r.t. the query. 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.

The user can search for the songs using the full name or part of full name of the song, artist or band name or genre using the search box present in the homepage of our Search Engine. The result displayed will contain the top 10 most relevant songs along with their Artist or Band Names, Album Names, Genres and Duration.

**Runtime for creating the vector space model** - Approximately 1-5 minutes(for all files)
**Runtime for returning query results** – Around 10-30 seconds