**Hello**

In this word document I will explain my thought process behind this project,

I focused on not only "making it work" but on making it OOP as possible and also very reusable for example:

In part 2 of the project, multithreading part, I initialized the Worker threads with the Wikipedia class, which the solo purpose of it is to return request from a wiki page, I can easily replace it with a different date source, or a diffract site whatsoever and make very minimal changes to the code to use that different data source. We can do the same thing with the search algorithm in the future if for example we will try to work on a more efficient one or something like that.

I know there is basically no interface in python and an interface only acts as a blueprint to design a class. Still I tried my best to make it as OOP and abstract as possible.

So at first I made a class diagram of the project and the vision I have for the making of it.

And after I understood what I had to do I went straight to work of curse with some trial and error, it took me some time to get everything to work, nonetheless I had a great time doing it.

**Class explanation: Part 1**

We are creating a single word list from any keyword, processing it with regex and with an algorithm we check if each word is in the data we receive (in this case input string).

Search folder consists of:

**Keyword**-*This class is for saving list of single keywords*

**KeywordPool**-All of the keywords, list[list] also it processes the string it gets to remove any non-alphanumerical characters and lowercases everything so we can then check it with the string efficiently.

**KMSSearchAlg**-uses both previous class to search the string with the keywords from the file, this algorithm gets called from KeyWordSearch to do the actual searching.

**KeyWordSearch**-Takes the searching algorithm class and uses it to make (sends the string of data) after that append all the results that matched to one list for output

Isearch Folder consists of:

**IKeyword**-interface Keyword

**IKeywordPool**-interface KeywordPool

**IKeywordSearch**-interface KeywordSearch

**ISearchAlg**-interface KMSSearchAlg in our case, we can replace algorithm at any time without changing a lot of code

**Class explanation:Part 2**

Uses all of the classes from part 1 too, but instead uses multithreading to extract and search for keywords in data taken from Wikipedia pages.

MultithreadingKeywordSearch folder consists of:

**DataSearchWorker**-*worker thread class gets data from wiki page and uses our alg to search through* the data with our keywords, limited to a given number of request per second by the rate limit class.

**WikiWorkerPool**-initialization of all the threads including the rate limit thread. Sub class of a threading class he activates the worker threads and the rate limit thread.

**RateLimiting**-consist of 2 classes one is the thread subclass that notify other threads when needed and also the only thread that sleeps in the entire program, the other class is a locking class to prevent exceeding the number of requests per second.

**Wikipedia**-here we make requests to Wikipedia and return the response we get back from it, used by the worker thread to get data from, sent to worker thread class through wikiWorkerPool.

IMultithreadingKeywordSearch Folder consists of:

**IDataProvider**-interface for Wikipedia class, can be abstract class for any other data provider source like the example I gave previously in the intro

**IRateLimiting**-interface the locking class to prevent burden on the wiki servers, and the thread sub class that sleeps and notifyall, we can later replace current method for some thing else.