Capstone Project: **Indexing & Searching for Hot Topics with ElasticSearch, Apache Spark, Kafka, Count-Min, and Heavy Hitters**

Description:

In this project, you are expected to use Apache Spark and Kafka, Python ElasticSearch to build keyword-based querying and top-K/heavy-hitter analysis over a real-time text stream, such a Twitter status updates or Wikipedia updates. You are expected to implement CountMin and HeavyHitters algorithms to dynamically update the statistics on the collected keywords.

Here are some high-level general guidelines for this project

Part A: Getting acquainted with ElasticSearch, and its Python's client APIs

You should figure out and document how to set up ElasticSearch and how to perform basic document insertion/deletion/querying. You are expected to submit a small tutorial that prepares you for the other parts of the project.

Part B: Connecting Apache Spark, Kafka, and ElasticSearch.

Ideally, you should aim to develop an end-to-end solution:

(a) to ingest real-world streaming text data from some source (e.g., Twitter, Wikipedia, etc.).
(b) to stream text from your Kafka/Spark into ElasticSearch
(c) to support DRPC queries over ElasticSearch, such as "Term query" (check ElasticSearch API to find supported query types): at least 3 query types should be supported

Part C: Computing Heavy-hitters/Top-K using Count-Min and Bloom Filters

In this part, you will implement Heavy-hitters/Top-K tracking (either one) for keywords in your real-time text stream, while ignoring "stop-words" using a Bloom Filter. Note: Dr. Samatova will provide access to the video lectures and notes, papers on this topic. You should provide the following functionality:

(a) Implementing and/or using CountMinSketch algorithm to continuously update a Heavy-hitters/Top-K state from the incoming text, with stop-word filtering. Note: you may choose any reasonable set of "stop-words" that you like; please note what they are in your README.
(b) to support querying of the current Heavy-hitters/Top-K state from the client using DRPC, and the client code to periodically perform this query and print the results.

Submission Requirements:

Your submission must include:

- Source code file(s) with detailed comments.

- Include NAMES.txt file with your name if you do not want it to be displayed. Make sure NOT to include your name any other file in this case. If this file is missing from the submission then we assume that you are OK that your name is publicly available in case we decide to post your submission as an example capstone project solution in this class or any other classes.
- All testing will be done using Ubuntu, similar to Spark/Kafka vcl image: provide shell script to install any missing packages, libraries.
- README file with detailed instructions. It must include the following information:
    - Software that needs to be installed (if any) with download and installation instructions.
    - Environment variable settings (if any).
    - Instructions on how to run the program.
    - Instructions on how to interpret the results.
    - Sample input and output files (if applicable).
    - References to any software you may have used (if applicable).

In short, the TA should be able to install any required software, set up the environment, execute your program, and obtain results without any prior knowledge about your project.

Your program must execute in a virtual machine hosting Ubuntu set up and the Spark/Kafka setup similar to the class Spark projects.