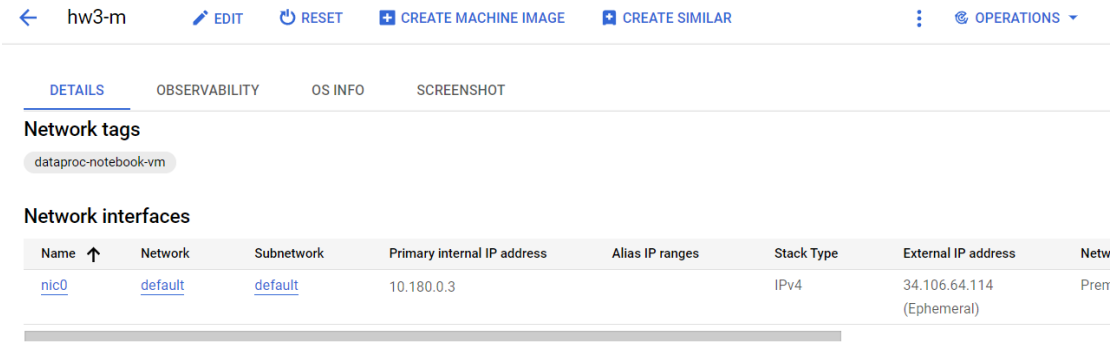
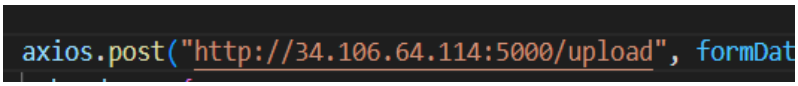
Final project report

Andrew ID: haichuax

0.How to run these applications

Step 1: Change external IP: 34.106.64.114 (it will change every time I restart the clusters) in:
sample\src\pages\landing.js
sample\src\component\TopN.js
sample\src\component\MiniSearchClient.js
Example:



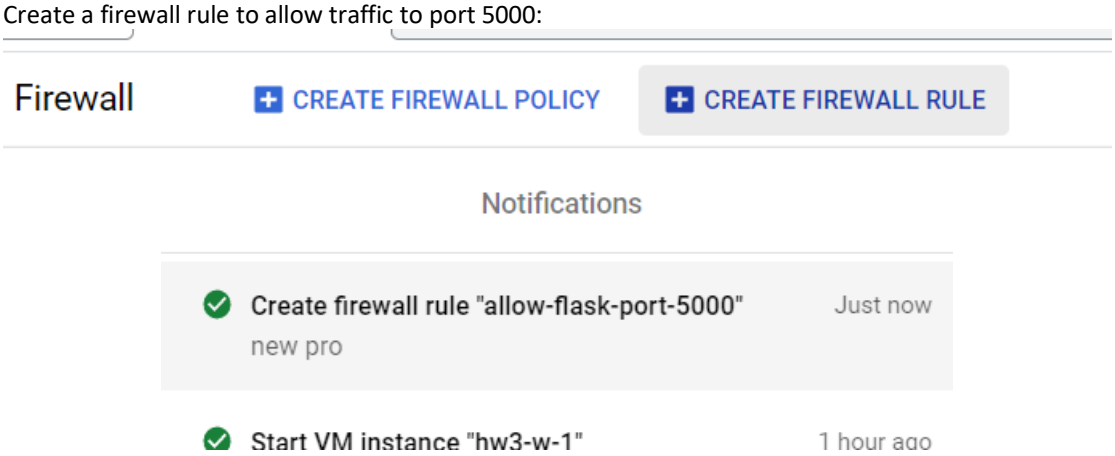


Step2: Start the first application on your PC:
cd sample
docker run -it --rm -v %cd%:/app -v /app/node_modules -p 3001:3000 -e CHOKIDAR_USEPOLLING=true tonyrays/dockerhub:projtimagepush2

Step3: Start the second application on GCP:
Connect to compute engine via SSH, move folder flask_second_app to the compute engine.
pip install Flask
pip install flask_cors
python app.py

Note1 If TCP connection is blocked:

Create a firewall rule to allow traffic to port 5000:

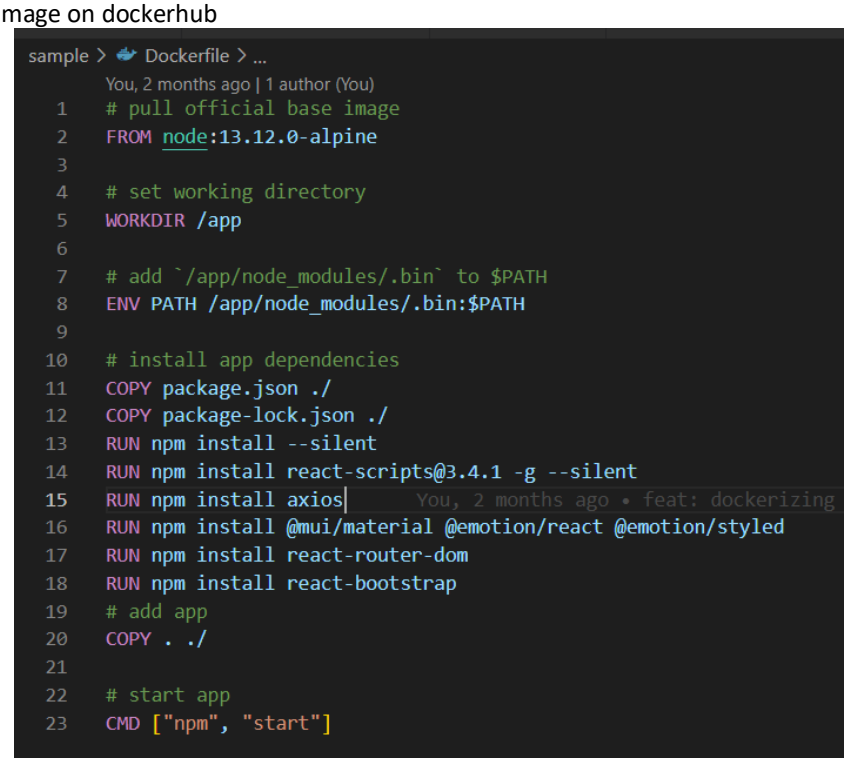


Note2 Data folder:

Create a 'Data' folder and insert the necessary data into it. Then, compress the folder into a zip file for uploading. Alternatively, you can use the zip file provided in the extra credit quiz.("./test_data/Data.zip")

1. Brief introduction to the first application

The first application contains four frontend pages:
(1) Landing.js: where you can upload a zip file to construct Inverted Indicies
(2) MiniSearchIndex.js: index page, proceed to (3) or (4), or go back to (1)
(3) MiniSearchClient.js: Search For Term
(4) Top_N.js: TOP-N Frequent Terms
(5) dropdown.js: implement a dropdown for better user experience
(6) Docker image on dockerhub



Watch the video for more details.

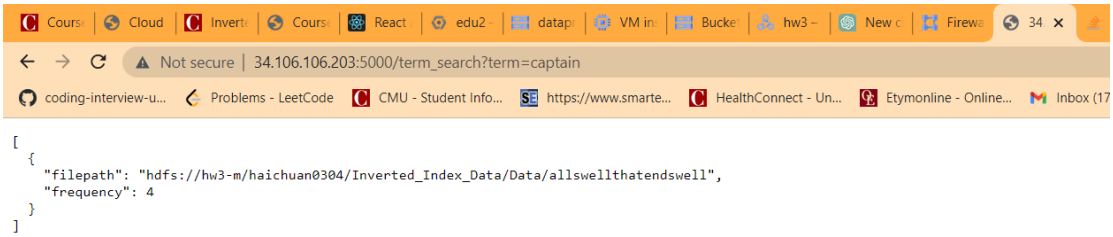
2. How I create the second application step by step

Since I have completed extra credit quiz, so I just modify the mapper.py and reducer.py I used for extra credits.

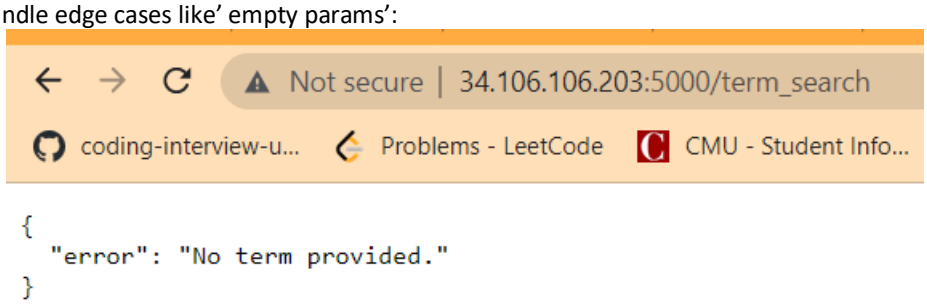
The second application contains:
(1) App.py
(2) Mapper_q.py
(3) Reducer_q.py
(4) Term_search.py
(5) Top_n_search.py

2.1 Test manually generate json file:
hadoop jar /usr/lib/hadoop/hadoop-streaming.jar -file mapper_q.py -mapper 'python mapper_q.py' -file reducer_q.py -reducer 'python reducer_q.py' -input /haichuan0304/Inverted_Index_Data/Data/ -output /haichuan0304/output_inverted_final6

hadoop fs -getmerge /haichuan0304/output_inverted_final6 ./flask/inverted_index.json

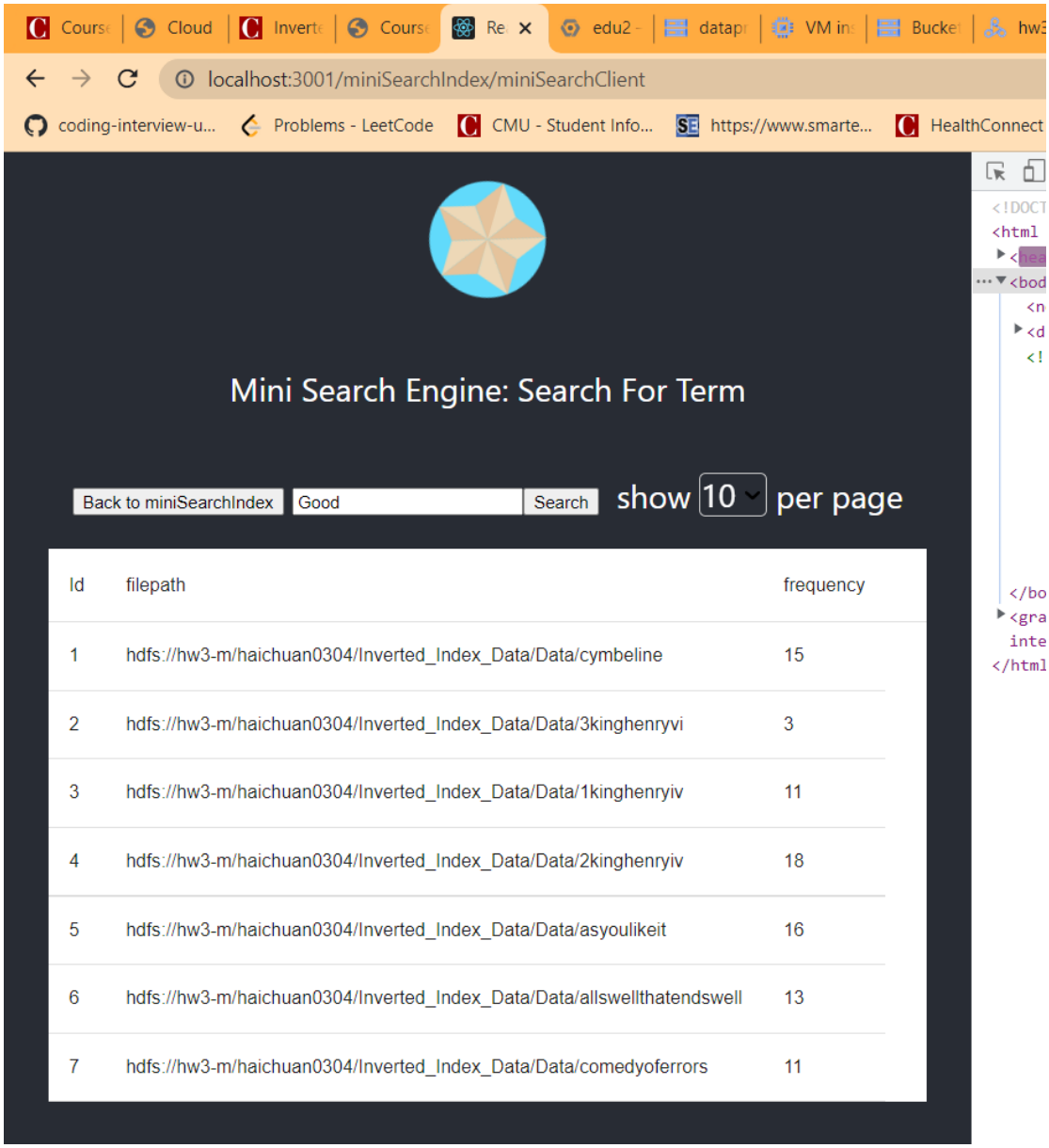


I also handle edge cases like' empty params':

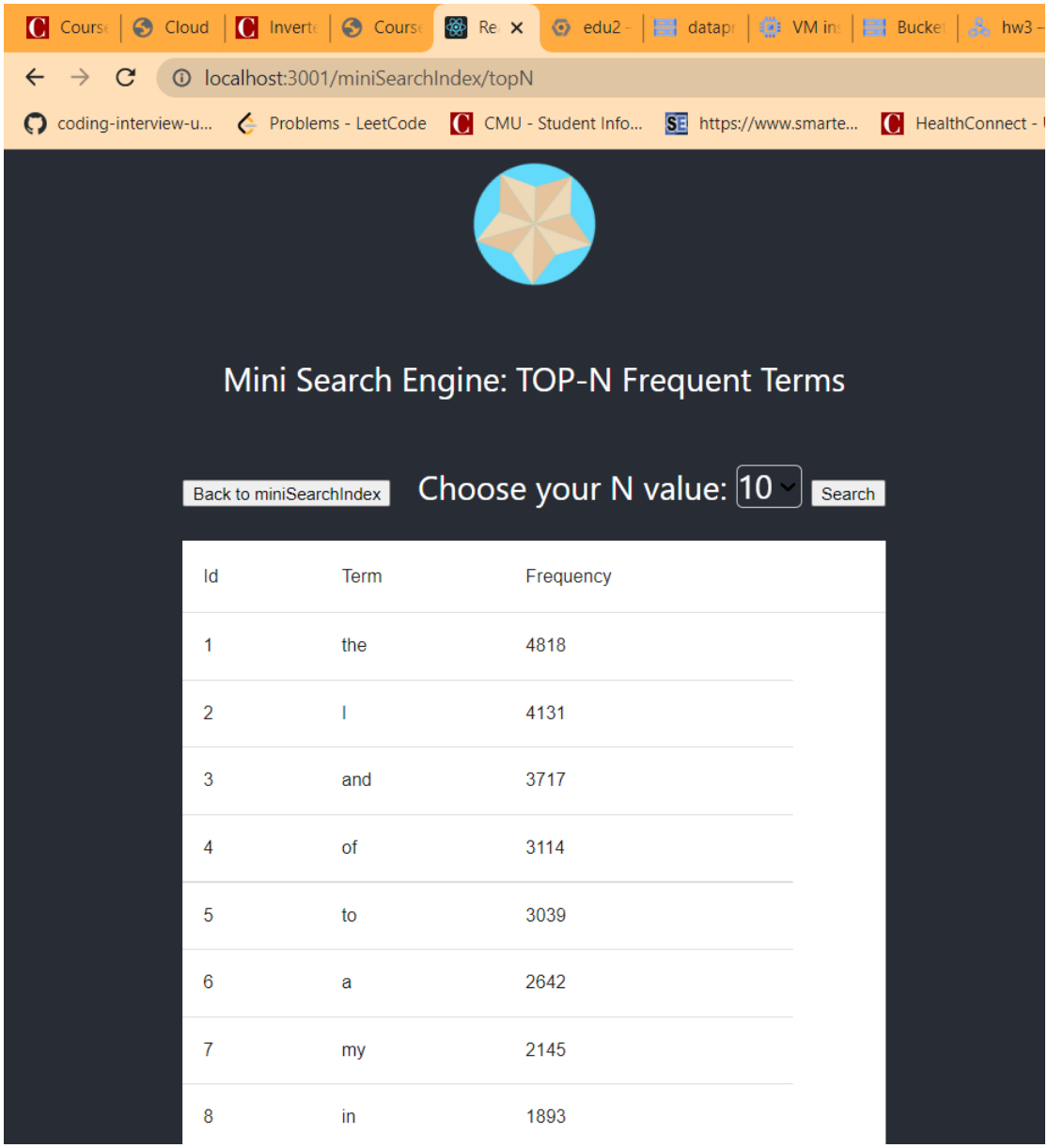


2.2 Test react app:

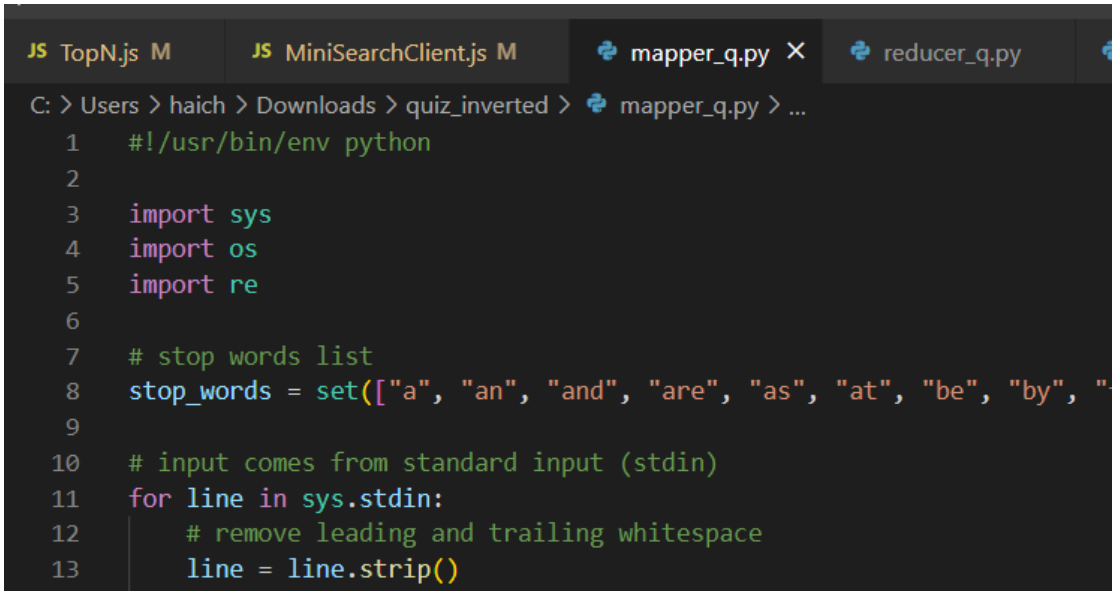
docker run -it --rm -v %cd%:/app -v /app/node_modules -p 3001:3000 -e CHOKIDAR_USEPOLLING=true tonyrays/dockerhub:projtimagepush2



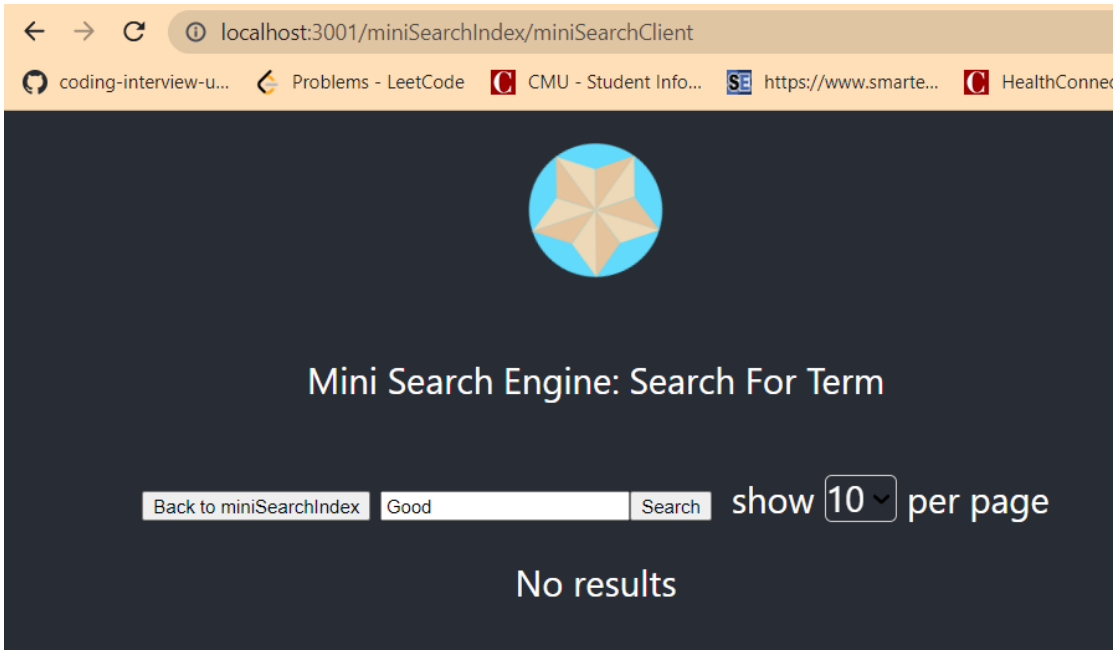
Also test out top_n_search:

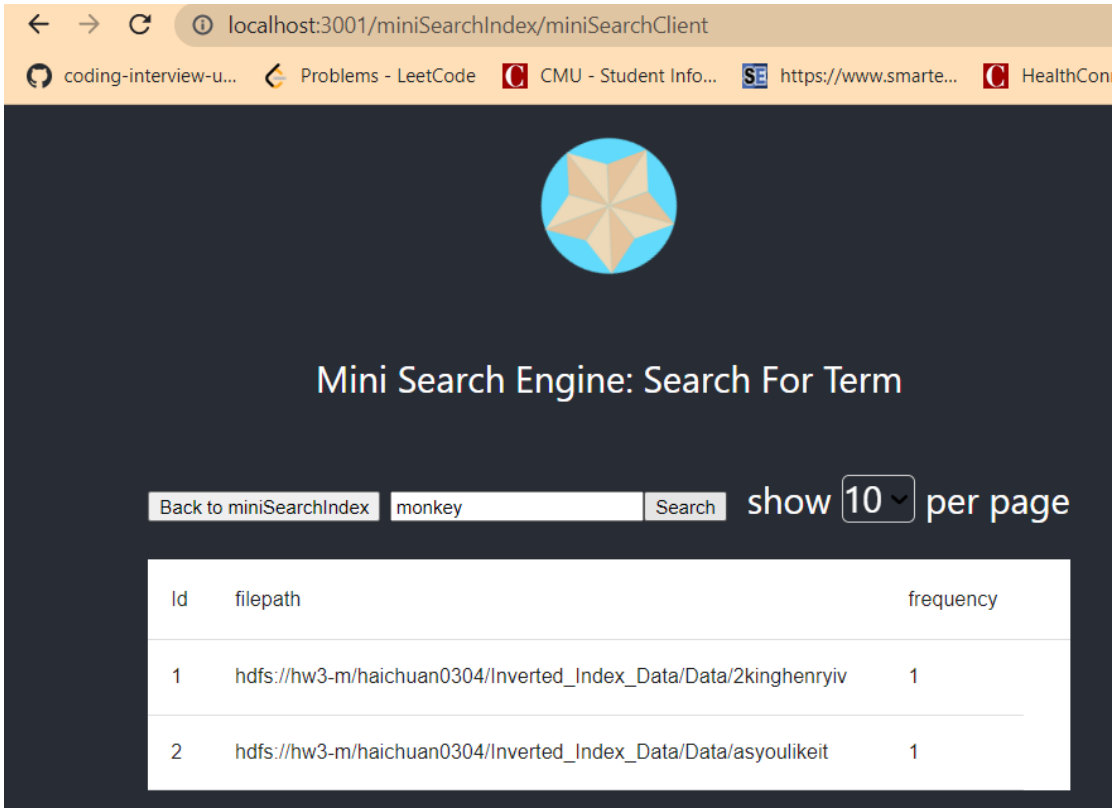


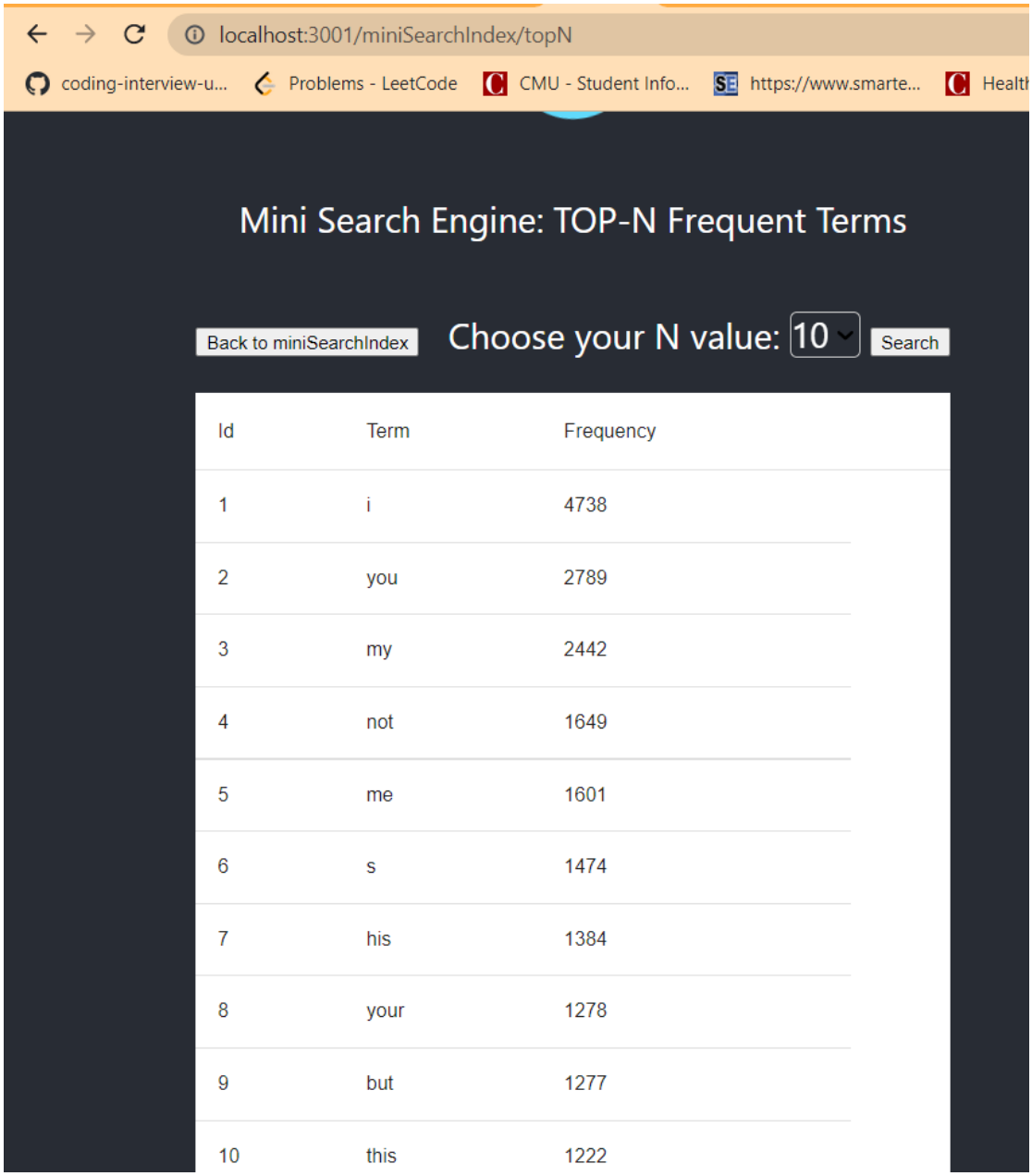
2.3 Add stop word list



2.4 Regenerate json file with stop words:
hadoop jar /usr/lib/hadoop/hadoop-streaming.jar -file mapper_q.py -mapper 'python mapper_q.py' -file reducer_q.py -reducer 'python reducer_q.py' -input /haichuan0304/Inverted_Index_Data/Data/ -output /haichuan0304/output_inverted_final7
hadoop fs -getmerge /haichuan0304/output_inverted_final7 ./inverted_index.json




localhost:3001/miniSearchIndex/miniSearchClient
coding-interview-u...
Problems - LeetCode
CMU - Student Info...
https://www.smarte...
HealthCon
Mini Search Engine: Search For Term
Back to miniSearchIndex
monkey
Search
show 10 per page
Id
filepath
frequency
1
hdfs://hw3-m/haichuan0304/Inverted_Index_Data/Data/2kinghenryiv
1
2
hdfs://hw3-m/haichuan0304/Inverted_Index_Data/Data/asyoulikeit
1



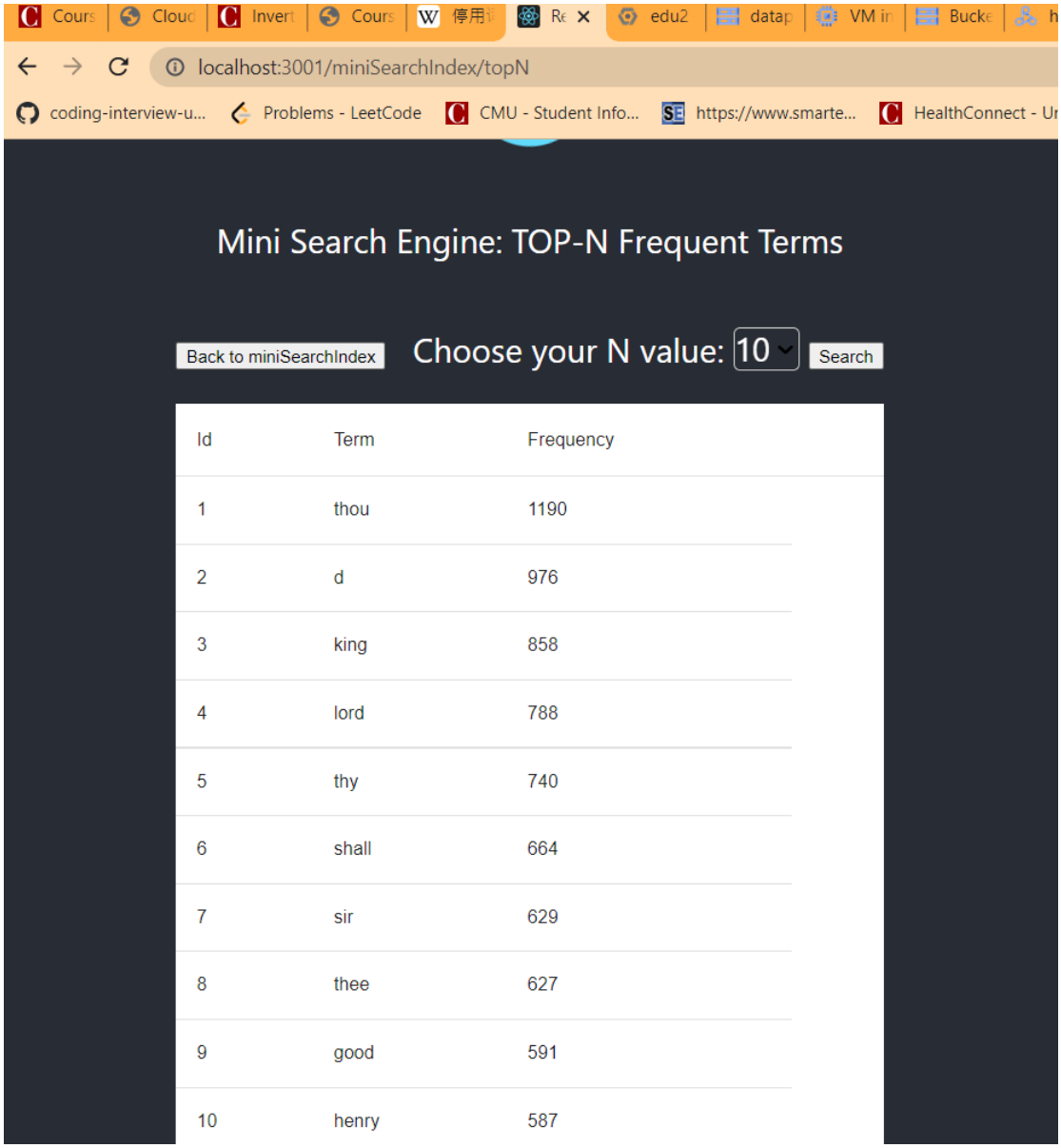
The top-N results are not satisfactory; therefore, it is necessary to include additional stop-words.

2.5 Regenerate json file with even more stop words:

Here I used stop words from library nltk.

```
import nltk
from nltk.corpus import stopwords
print(stopwords.words('english'))
```

{'ourselves', 'hers', 'between', 'yourself', 'but', 'again', 'there', 'about', 'once', 'during', 'out', 'very', 'having', 'with', 'they', 'own', 'an', 'be', 'some', 'for', 'do', 'its', 'yours', 'such', 'into', 'of', 'most', 'itself', 'other', 'off', 'is', 's', 'am', 'or', 'who', 'as', 'from', 'him', 'each', 'the', 'themselves', 'until', 'below', 'are', 'we', 'these', 'your', 'his', 'through', 'don', 'nor', 'me', 'were', 'her', 'more', 'himself', 'this', 'down', 'should', 'our', 'their', 'while', 'above', 'both', 'up', 'to', 'ours', 'had', 'she', 'all', 'no', 'when', 'at', 'any', 'before', 'them', 'same', 'and', 'been', 'have', 'in', 'will', 'on', 'does', 'yourselves', 'then', 'that', 'because', 'what', 'over', 'why', 'so', 'can', 'did', 'not', 'now', 'under', 'he', 'you', 'herself', 'has', 'just', 'where', 'too', 'only', 'myself', 'which', 'those', 'i', 'after', 'few', 'whom', 't', 'being', 'if', 'theirs', 'my', 'against', 'a', 'by', 'doing', 'it', 'how', 'further', 'was', 'here', 'than'}

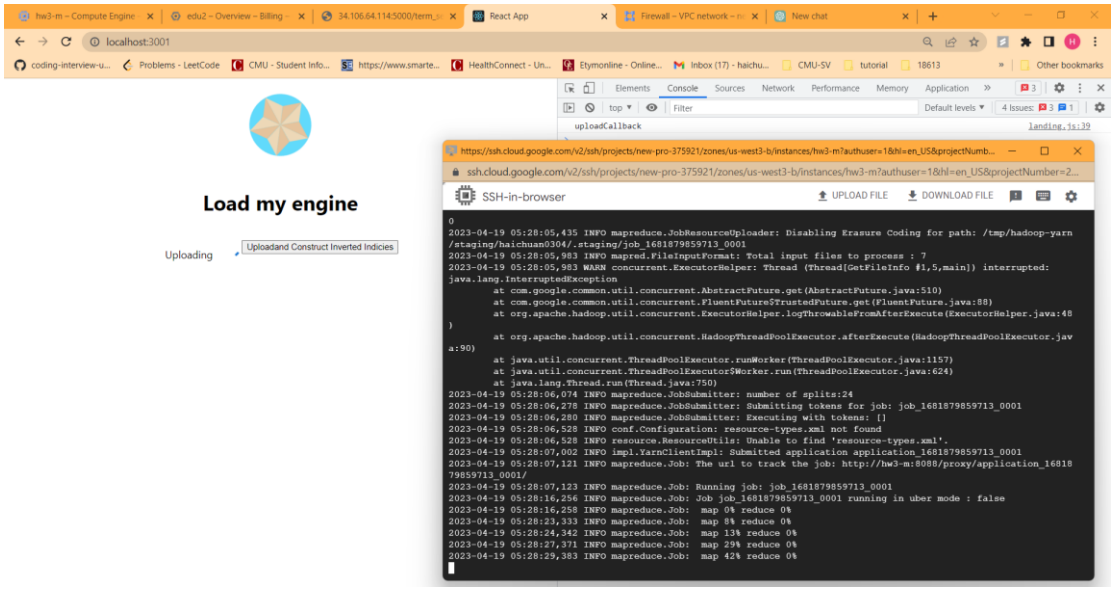


Looks much better now!

2.6 Allow user to upload a zip file and automatically generate json file.

To prevent auto reload, disable use_reloader.

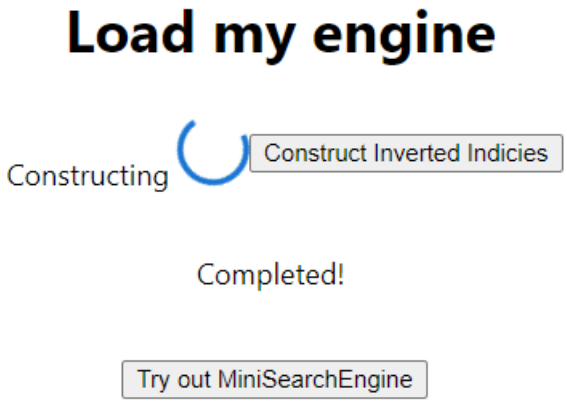
```
app.run(use_reloader=False, debug=True, host='0.0.0.0')
```



Delete ‘input’ and ‘output’ folder on HDFS to maintain a tidy environment.

```
2023-04-19 05:28:54,664 INFO streaming.StreamJob: Output directory: /haichuan0304/output
Deleted /haichuan0304/input
Deleted /haichuan0304/output
```

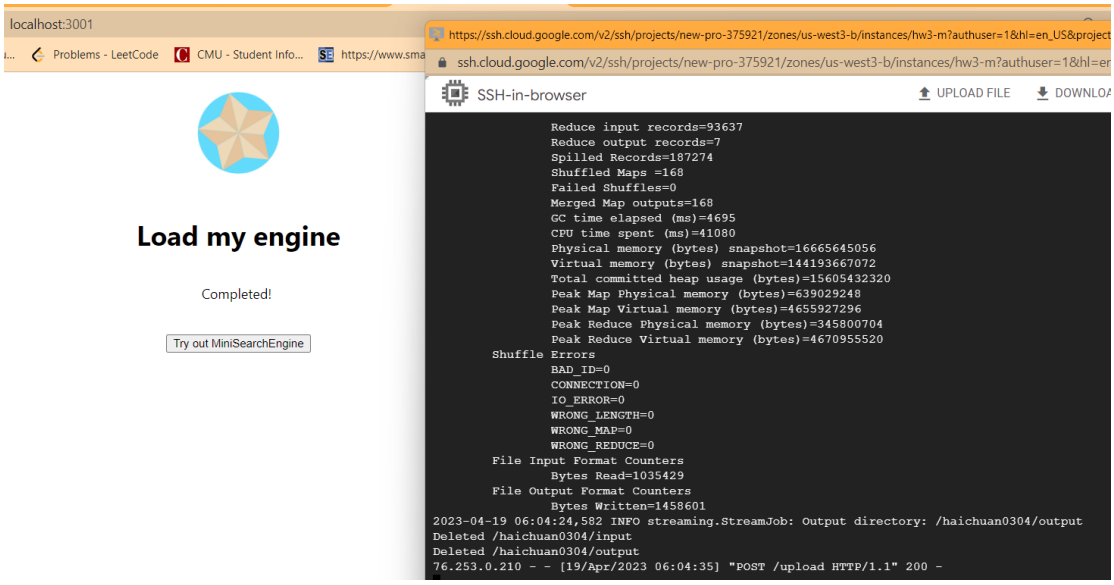
Rename the button:



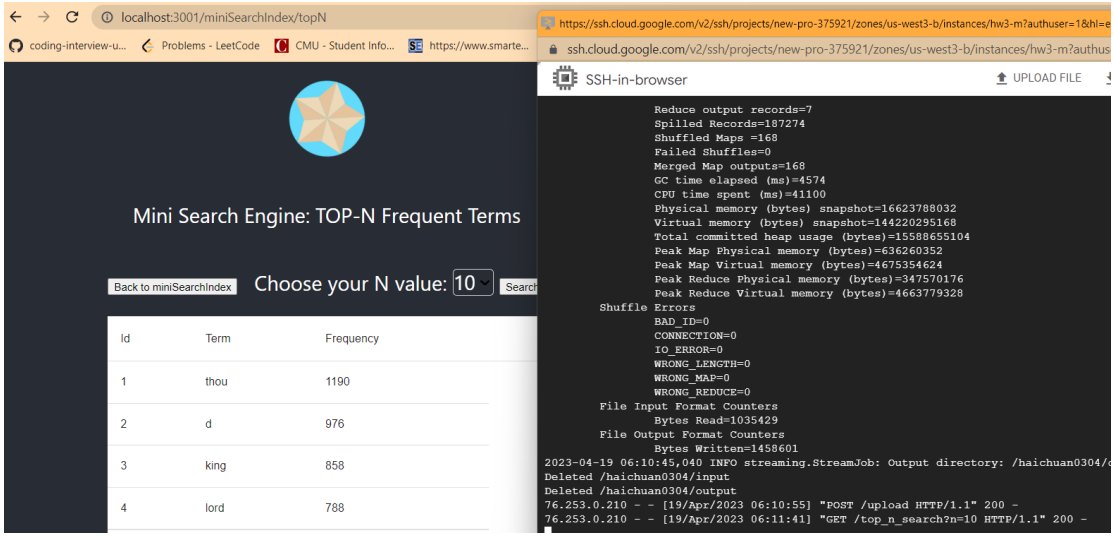
2.7 Double check:

Delete inverted_index.json

```
haichuan0304@hw3-m:~/flask$ rm inverted_index.json
haichuan0304@hw3-m:~/flask$ ls
__pycache__   mapper_q.py      reducer_q.py    top_n_search.py
app.py        reducer.py.save  term_search.py  uploaded_files
```



Works well:



Update:

I recently noticed that I forgot to include the execution time in the output, so I have updated my functions to return this information. Additionally, I have created a second video demonstration to showcase the newly execution time feature. Sorry for the inconvience.

e.g.

```python
import time

def term_search(term, word_postings):
    start_time = time.time()

    # Search for the term in the inverted index
    postings = word_postings.get(term, {})
    # results = [{"filepath": filepath, "frequen
    results = [{"id": idx, "filepath": filepath,

    execution_time = time.time() - start_time
    return results, execution_time
```

```
const Result = ({searchResults, execution_time}) => {
  return (<div>
    {/* <TextField/> */}
    <h3>Your search was executed in {execution_time} seconds</h3>
    <Table style={{backgroundColor:'white'}}>
      <TableHead>
```