CS172 - Final Project

Team Member	Name	Contribution
1	Renzo Olivares	42%
2	Phyllis Chen	29%
3	Joshua Torres	29%

Overview:

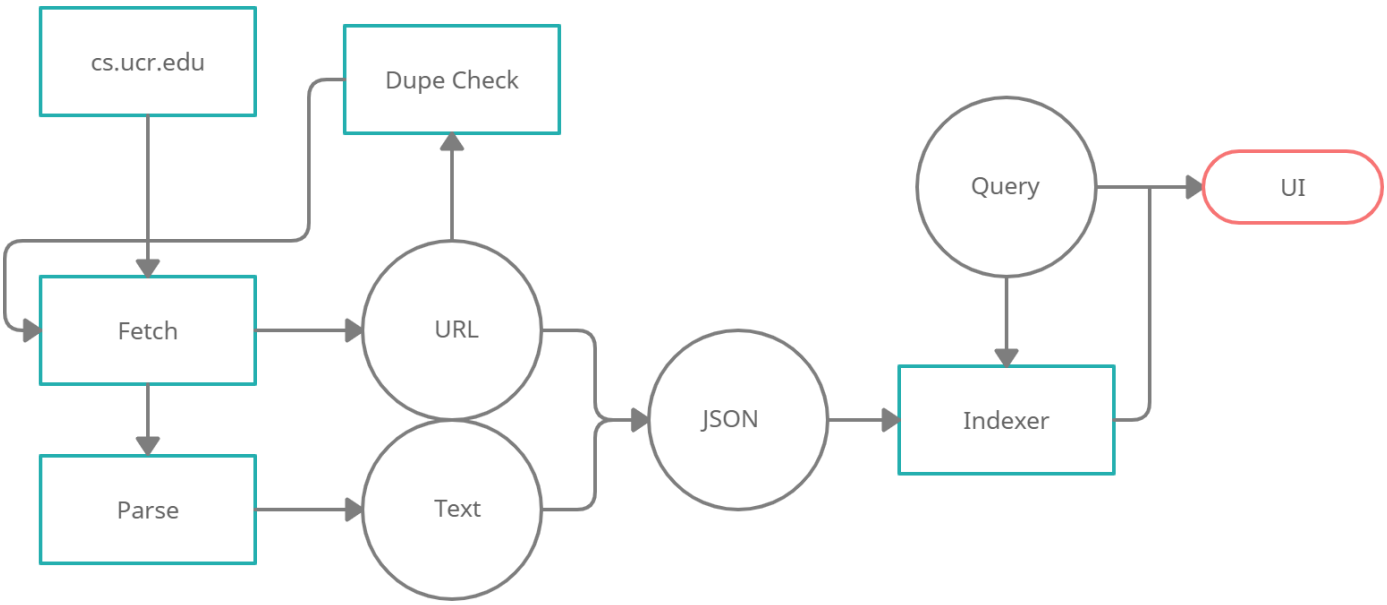
Using URLs to create documents, and utilizing Elasticsearch to index the documents, Croogle is a web crawler for cs.ucr.edu.

In order to run Croogle:

- Python 3.7
- Flutter https://flutter.dev/docs/get-started/install
- Dart
- Android Studio/VSCode (To launch Flutter app)
- Flutter & Dart Plugins for chosen IDE (Android Studio/VS Code)
- ElasticSearch local https://www.elastic.co/downloads/elasticsearch
- `pip install requests`
- `pip install lxml`
- `pip install BeautifulSoup4`

1. Make sure to start elasticsearch service before doing anything.
2. Run the builder.sh file to run crawler.py and indexer.py `$ source ./builder.sh` .
    - In order to run with a user-specified number of hops and pages, run:
      `$ python crawler.py --hops 2 --pages 40`
3. Run the indexer.sh file to run the indexer on its own `$ source ./indexer.sh` (can skip if ran 1.).
4. Running runner.sh will allow the user to search through the index `$ source ./runner.sh` .
    - For example, when prompted "Enter search: ", the user can query "CSE" to return results.
5. To run front-end import flutter_front_end/bubbles as a project into Android Studio and run in there, must have Flutter installed.

Architecture



Limitations:

- While Croogle is capable of crawling, it does run exponentially slower after designating a number of hops greater than 3.

Part 1: Crawler

The crawler collects a list of initial URLs from seedurls.txt into a queue. It keeps a counter on the URLs and assigns a document ID to it as well. As the crawler finds new URLs, each URL is added to the queue. From the queue, it continues crawling after the seed URLs have been crawled.

Using the requests library, the crawler can get the text from the URL and parse it using BeautifulSoup. The crawler stores the full document ID (each full document ID has a prefix "RJP" and ends in its assigned document ID), the url, and the textual content into an object, which is then added to a list. The crawler then iterates through the list of objects and writes each object to a JSON file.

The crawler uses SimHash to check for previously visited URLs, and is also able to check if an href link is an extension of the current URL it is on. If the URL has been previously seen, then the crawler will skip the link.

In an attempt to prevent devation from content relevancy, the user can set the depth of the crawler through `--hops` and set the number of pages crawled through `--pages`. Whichever criteria the crawler reaches first is where it will stop at.

An example of the crawler in action:

```
PS C:\Users\Torre\Documents\GitHub\CS172_FinalProject\Crawler> python crawler.py --hops 2 --pages 40
https://www1.cs.ucr.edu/
https://www1.cs.ucr.edu/
https://www.engr.ucr.edu/covid-19-information-page

https://www1.cs.ucr.edu/news/cse-professor-received-doctoral-dissertation-faculty-awar
https://www1.cs.ucr.edu/news/cse-professor-papalexakis-received-the-nsf-career-awar
https://www1.cs.ucr.edu/news/icpc-team-advances-to-north-american-division-championshi
https://www1.cs.ucr.edu/new
https://www1.cs.ucr.edu/about/event
https://www1.cs.ucr.edu/about/distinguished-lecture
https://library.ucr.edu
https://campusstatus.ucr.edu/
https://campusstore.ucr.edu/

https://diversity.ucr.edu/
https://campusmap.ucr.edu/

https://www.ucr.edu/about/visitors

https://campusmap.ucr.edu/?loc=CHUNG
mailto:contact@cs.ucr.edu

https://www1.cs.ucr.edu/department-intrane
https://twitter.com/ucr_cse
```

Part 2: Indexer

The indexer takes all the JSON files that the crawler has collected and uses Elasticsearch to index. Since the requests library is able to provide commands equivalent to the curl commands used in Linux, we are able to post the index to the local server.

Part 3: Extension

Search

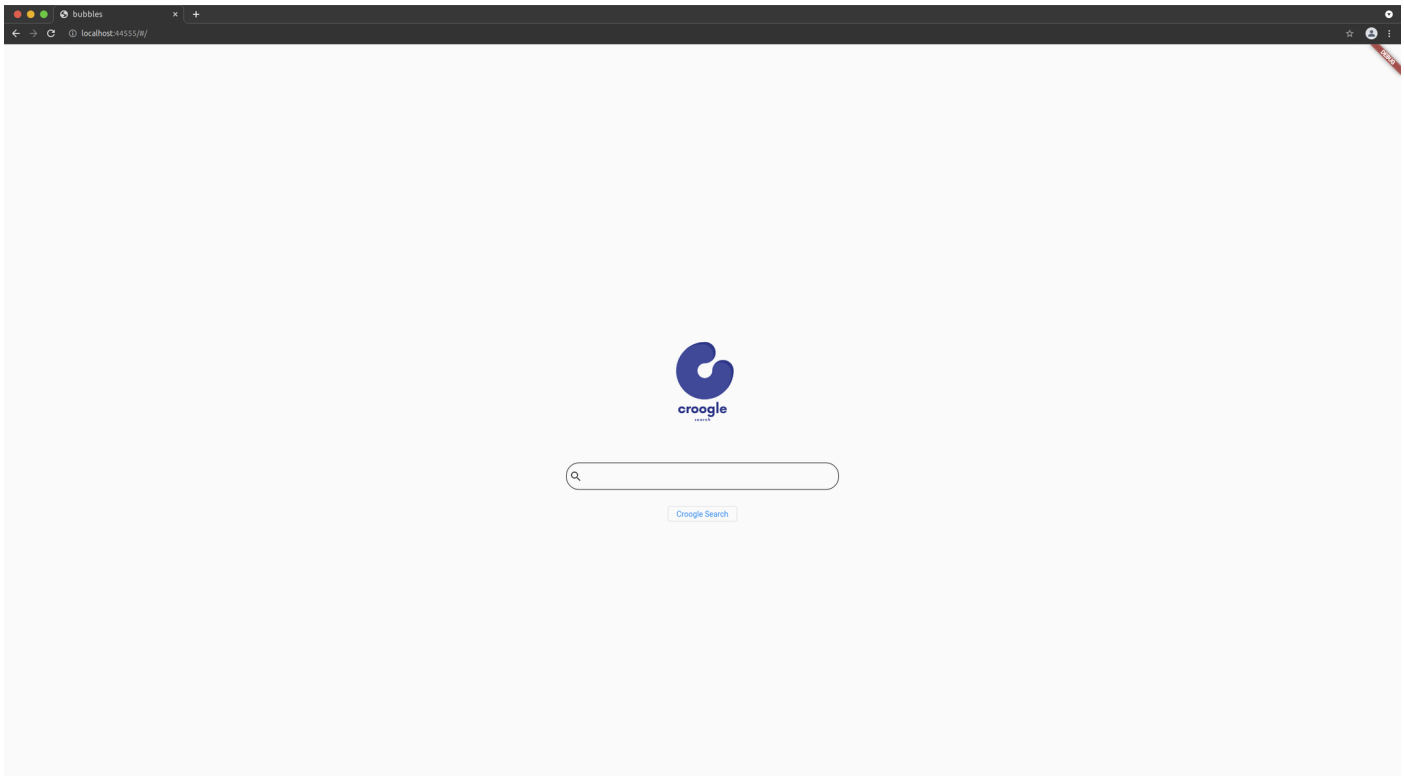
The user is prompted to enter a query. Using the requests library, Elasticsearch will return the top 10 results from the user's query.

UI

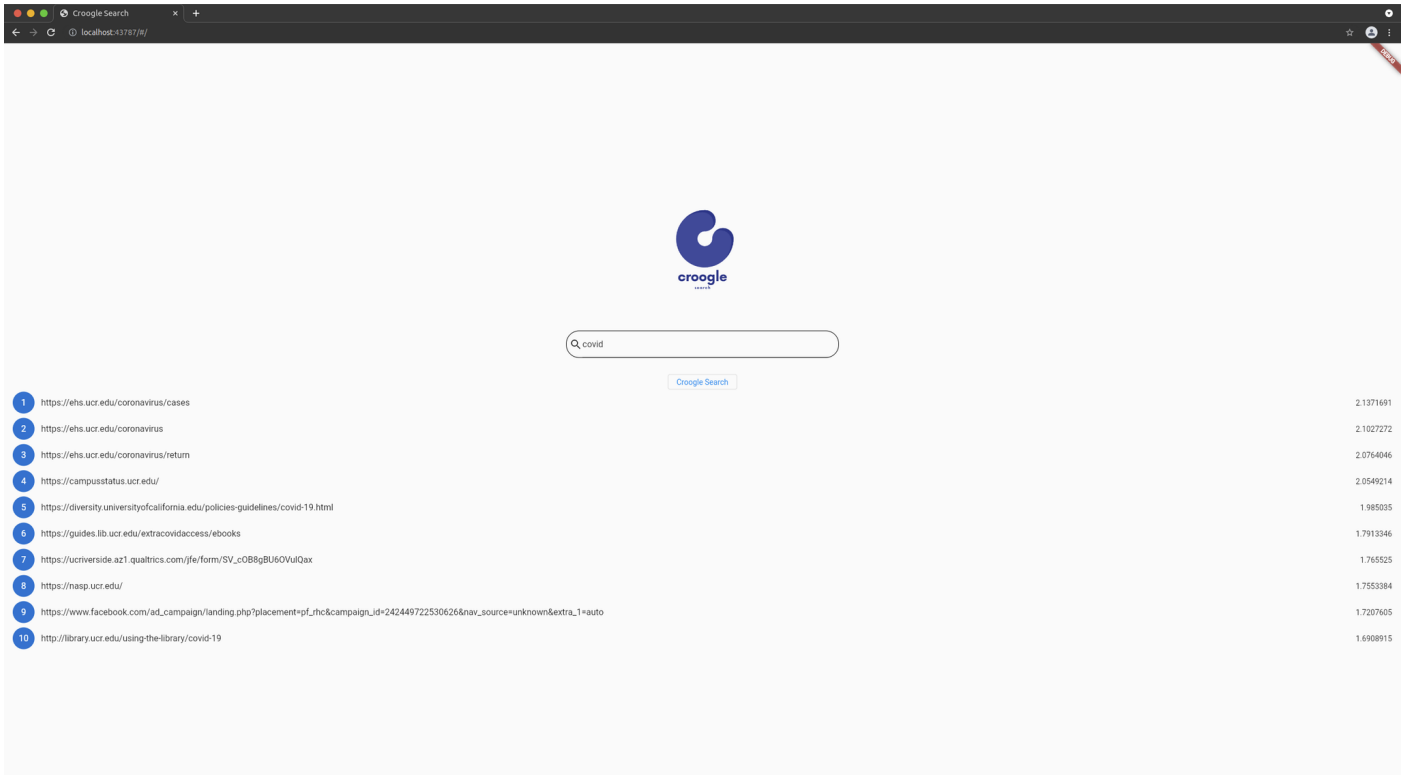
We have implemented a web-based interface using Flutter to display the user's query and the list of results returned by Elasticsearch.

Croogle UI:

Croogle UI:



Croogle UI with a query, displaying the first 10 results:



Extra Notes:

- If bash scripts do not work try editing the python version argument `python indexer.py` to `python3.9 indexer.py` for example, or the version you are currently on.

- There is a 0.5 second wait time for implicit politeness.
- There is a filtering for robots.txt for explicit politeness.
- We used this stack overflow post to assist in learning how to overcome cors policy issue when connecting our front end to the elasticsearch backend:
    - https://stackoverflow.com/questions/37384380/cross-origin-request-blocked-elasticsearch.