Project WikiCrawler

CS 172 Section 001

Kevin Ni 862246587
Calvin TszFung 862085322
Aden Ghadimi 862153194

Credit

In completing this assignment we consulted:

- CS172 lectures and notes
- Python 3 Documentation
  https://docs.python.org/3/
- Stack Overflow
  https://stackoverflow.com/questions
- Elasticsearch
  https://elasticsearch-py.readthedocs.io/en/v7.13.1/api.html
- BeautifulSoup
  https://www.crummy.com/software/BeautifulSoup/bs4/doc/
- Django
  https://docs.djangoproject.com/en/3.2/
- ArgParse
  https://docs.python.org/3/library/argparse.html
- Web Crawler Wikipedia Page
  https://en.wikipedia.org/wiki/Web_crawler
- Script Output on Html Page Part2 by HACK ANONS for understanding how to build a website with the django framework
  https://www.youtube.com/watch?v=s6Xi7x4G7yg

All important code is original. Unimportant subroutines that are not completely original are…

- All subroutines used from **argparse**, to handle parsing the command line
- All subroutines used from **os**, to handle folder making
- All subroutines used from **glob**, to file pathing
- All subroutines used from **json**, to handle json files and formatting
- All subroutines used from **requests**, to handle server pinging
- All subroutines used from **urllib**, to manage html requests
- All subroutines used from **ssl**, to handle sockets
- All subroutines used from **re**, to use regular expressions
- All subroutines used from **copy**, to allow for deep copies

The coding was portioned as follows:

- Kevin Ni - **Part 1**
- Calvin TszFung Ng - **Part 2**
- Aden Ghadmi - **Part3**

Part 1 - Crawler

Architecture

- Wikipedia_crawler.py
- Wikepedia_page.py
- Sources.txt
- Run.py

Wikipedia_crawler.py This python file is responsible for the crawler class. The crawler starts from a given url and keeps track of the links in that html page. These links are then immediately crawled next, until the max depth is reached, in which they recursively return to the next link above that should be traversed. The class includes the following functions:

- register_page(self, url)
- download_page(self, url)
- parse_category(self, url, depth)
- parse_page(self, url, depth=0)
- crawl(self, initial_link, depth=0)

Wikipedia_page.py This python file is responsible for the wikipediapage class. The class both serves as a data structure and also carries a function to write itself to a given json file. This provides a useful way to store each page's data without worrying about complex vectors/arrays. The class includes the following data members:

- url
- title
- html
- table of contents
- graphics
- paragraphs
- links

Sources.txt This text file is responsible for storing all of the starting URLs should a url not be given,

Run.py This python file is responsible for parsing the command line and carrying out the crawler functions. This includes making a new folder to store the .json files should one not exist, cleaning the folder if the `--clean` flag is used, and initializing the crawler.

Crawling Strategy

Our crawler handles traversing by using recursion. The crawler will traverse the first link and continue to do so until it reaches a max depth, upon which it will back out to the next link on the level above. Since we built a simple crawler, the crawler does not run parallel and processes each html page one at a time. The crawler does handle duplicates as it stores the crawled pages in a set and checks to see if it has come across the page before. The crawler can be provided a max depth and max pages and is also capable of running multiple starting URLs.

Data structures employed:

- BeautifulSoup4
- Sets
- Lists
- Json
- Classes

Limitations

There are limitations of the system. We do not check for a robots.txt, but since we are only crawling wikipedia, there is only one robots.txt to obey. Additionally, this crawler is unable to adapt to other websites, and thus is limited to crawling wikipedia. Another limitation is that the crawler has to traverse down one depth link first, instead of going breadth first. This can lead to a rapid increase in unrelated traversed pages, since the crawler is not prioritizing the links on the initial starting page.

Setup Instructions

Step 1: Clone the repository.

```
> $git clone <repository url>
```

Step 2: Check Python Version. More information can be found at https://www.python.org/download/releases/3.0/

```
> $python --version
```

Step 3: Install the packages. Make sure you are in the folder while doing so

```
> $pip3 install -r requirements.txt
```

Step 4: Run the crawler with a specified depth level by typing `py run.py [#]`
Run `py run.py -h` to pull up a help GUI of possible commands.
Possible flags are:

```
> --url
> --dir
> --pages
> --clean
```

Run Instructions

The main command to run is `py run.py`

Depth - This number will determine how nested the crawler should go.
> If 0 is given, only the starting pages will be crawled
> If 1 is given, only the starting pages and the links on those pages will be crawled.
> If 2 is given, only the starting pages and 2 links in will be crawled.
> If -1 is given, there will be no maximum depth. WARNING: This is highly discouraged

Help - Will display the help message with descriptions on all the flags

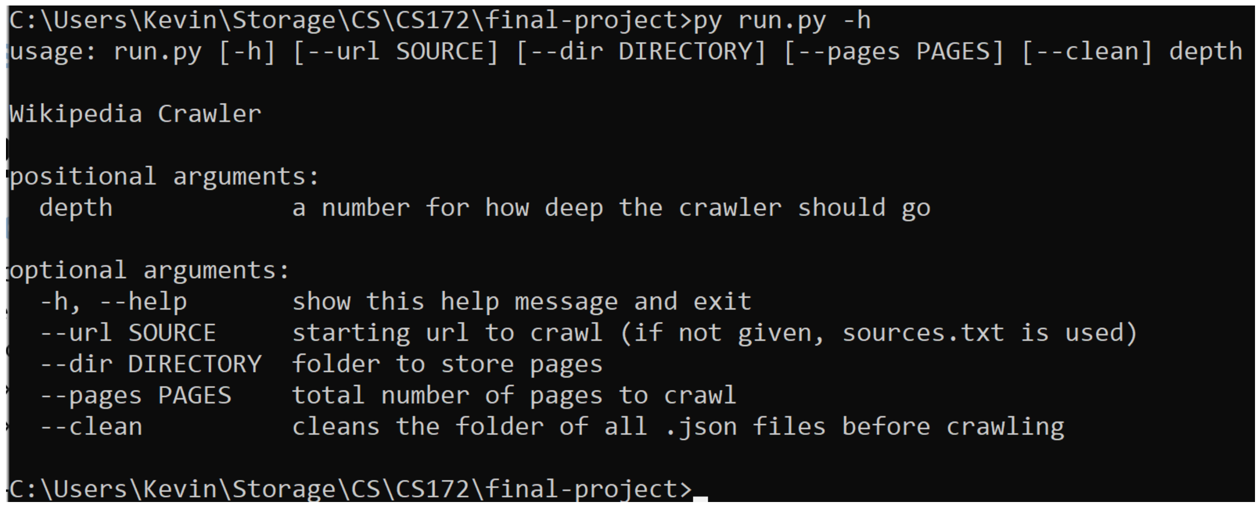
URL - Starting URL to crawl from. If this is not provided, sources.txt will automatically be used. Sources.txt can contain a URL on each new line, and the crawler will crawl through each.
> If `https://en.wikipedia.org/wiki/Web_crawler` is given, that will be the starting URL
> If nothing is given, sources.txt will be used to supply the URLs

Directory - This will determine the name for the folder where the .json files are stored in. If none is given, the name `data` is automatically set.
> If `sample_data` is given, then the folder will be named `sample_data`
> If none is given, then the folder will be named `data`

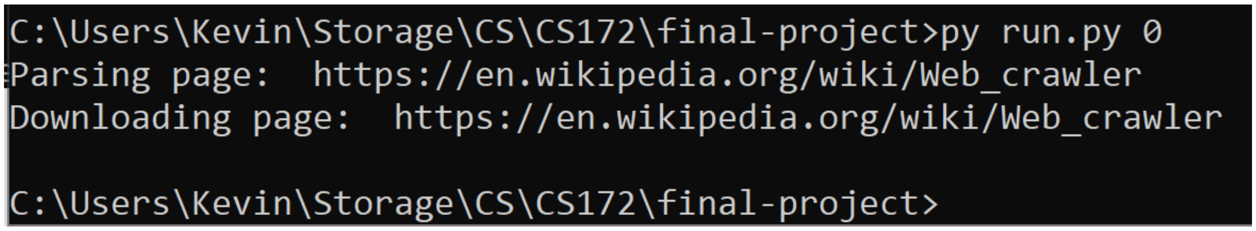
Pages - This will set the total number of pages to crawl. If none is provided, -1 will be set and there will be no maximum.
> If 0 is given, no pages will be crawled
> If 10 is given, 10 pages will be crawled
> If -1 is given, there will be no limit to how many pages is crawled

Clean - This is a flag that does not take on a value. If this flag is provided, the crawler will first clean the folder marked for the .json files before repopulating it.

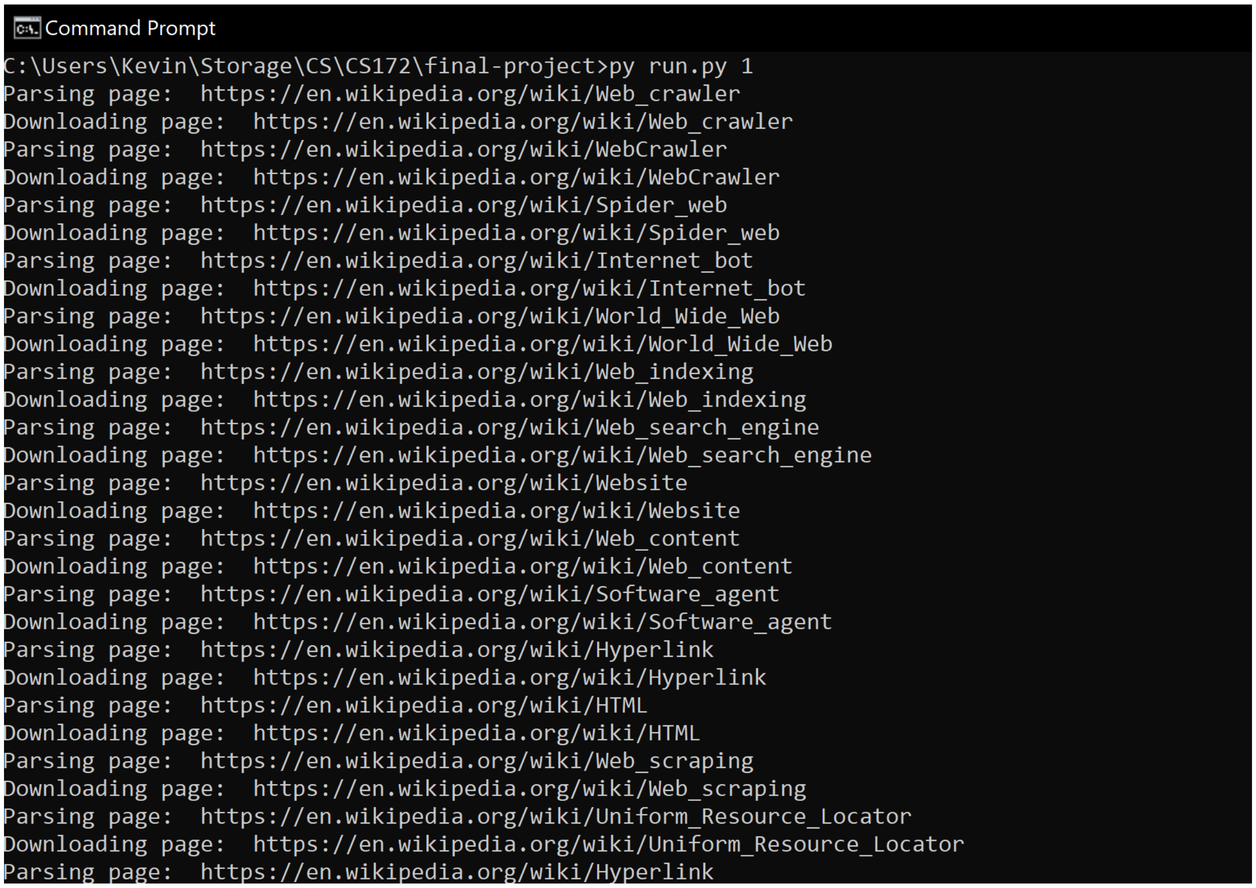
Screenshots



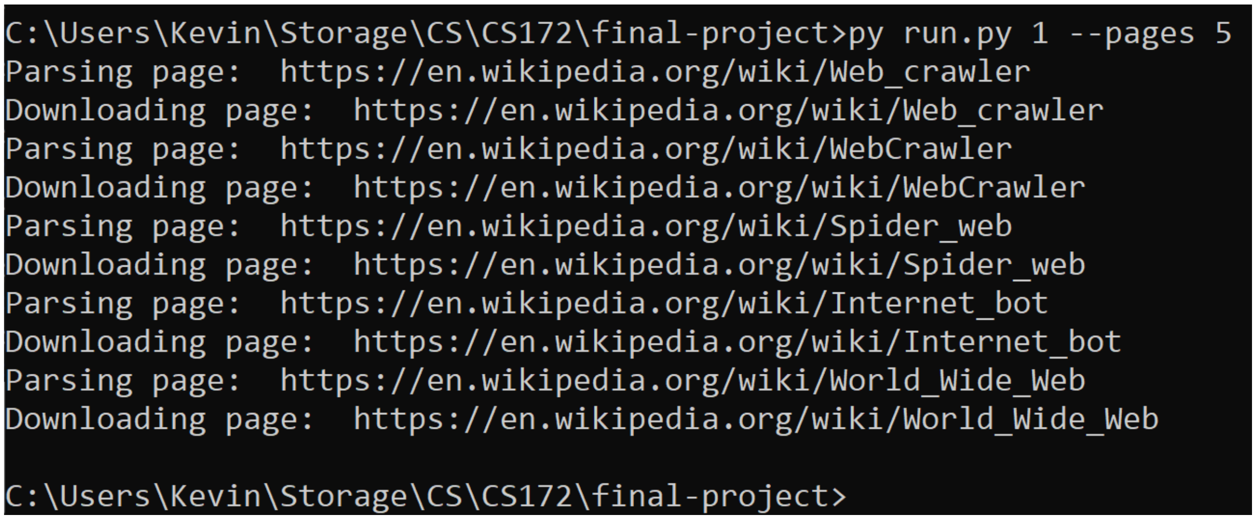
```
C:\Users\Kevin\Storage\CS\CS172\final-project>py run.py -h
usage: run.py [-h] [--url SOURCE] [--dir DIRECTORY] [--pages PAGES] [--clean] depth

Wikipedia Crawler

positional arguments:
  depth            a number for how deep the crawler should go

optional arguments:
  -h, --help       show this help message and exit
  --url SOURCE     starting url to crawl (if not given, sources.txt is used)
  --dir DIRECTORY  folder to store pages
  --pages PAGES    total number of pages to crawl
  --clean          cleans the folder of all .json files before crawling

C:\Users\Kevin\Storage\CS\CS172\final-project>
```



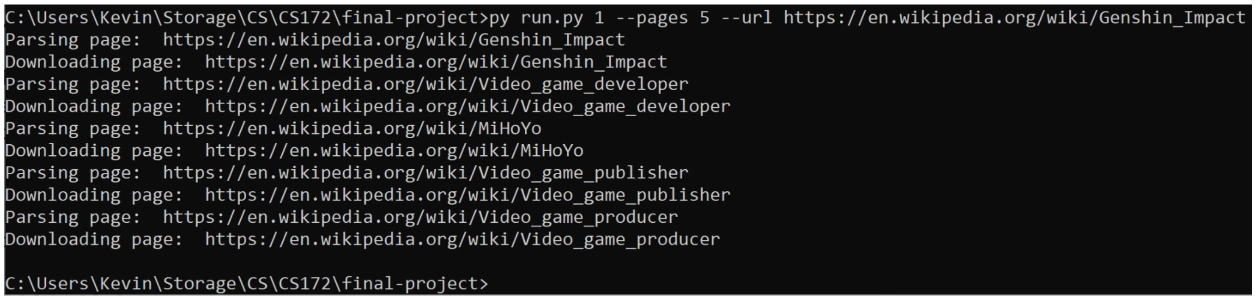
```
C:\Users\Kevin\Storage\CS\CS172\final-project>py run.py 0
Parsing page:  https://en.wikipedia.org/wiki/Web_crawler
Downloading page:  https://en.wikipedia.org/wiki/Web_crawler

C:\Users\Kevin\Storage\CS\CS172\final-project>
```



```
C:\Users\Kevin\Storage\CS\CS172\final-project>py run.py 1
Parsing page:  https://en.wikipedia.org/wiki/Web_crawler
Downloading page:  https://en.wikipedia.org/wiki/Web_crawler
Parsing page:  https://en.wikipedia.org/wiki/WebCrawler
Downloading page:  https://en.wikipedia.org/wiki/WebCrawler
Parsing page:  https://en.wikipedia.org/wiki/Spider_web
Downloading page:  https://en.wikipedia.org/wiki/Spider_web
Parsing page:  https://en.wikipedia.org/wiki/Internet_bot
Downloading page:  https://en.wikipedia.org/wiki/Internet_bot
Parsing page:  https://en.wikipedia.org/wiki/World_Wide_Web
Downloading page:  https://en.wikipedia.org/wiki/World_Wide_Web
Parsing page:  https://en.wikipedia.org/wiki/Web_indexing
Downloading page:  https://en.wikipedia.org/wiki/Web_indexing
Parsing page:  https://en.wikipedia.org/wiki/Web_search_engine
Downloading page:  https://en.wikipedia.org/wiki/Web_search_engine
Parsing page:  https://en.wikipedia.org/wiki/Website
Downloading page:  https://en.wikipedia.org/wiki/Website
Parsing page:  https://en.wikipedia.org/wiki/Web_content
Downloading page:  https://en.wikipedia.org/wiki/Web_content
Parsing page:  https://en.wikipedia.org/wiki/Software_agent
Downloading page:  https://en.wikipedia.org/wiki/Software_agent
Parsing page:  https://en.wikipedia.org/wiki/Hyperlink
Downloading page:  https://en.wikipedia.org/wiki/Hyperlink
Parsing page:  https://en.wikipedia.org/wiki/HTML
Downloading page:  https://en.wikipedia.org/wiki/HTML
Parsing page:  https://en.wikipedia.org/wiki/Web_scraping
Downloading page:  https://en.wikipedia.org/wiki/Web_scraping
Parsing page:  https://en.wikipedia.org/wiki/Uniform_Resource_Locator
Downloading page:  https://en.wikipedia.org/wiki/Uniform_Resource_Locator
Parsing page:  https://en.wikipedia.org/wiki/Hyperlink
```



```
C:\Users\Kevin\Storage\CS\CS172\final-project>py run.py 1 --pages 5
Parsing page:  https://en.wikipedia.org/wiki/Web_crawler
Downloading page:  https://en.wikipedia.org/wiki/Web_crawler
Parsing page:  https://en.wikipedia.org/wiki/WebCrawler
Downloading page:  https://en.wikipedia.org/wiki/WebCrawler
Parsing page:  https://en.wikipedia.org/wiki/Spider_web
Downloading page:  https://en.wikipedia.org/wiki/Spider_web
Parsing page:  https://en.wikipedia.org/wiki/Internet_bot
Downloading page:  https://en.wikipedia.org/wiki/Internet_bot
Parsing page:  https://en.wikipedia.org/wiki/World_Wide_Web
Downloading page:  https://en.wikipedia.org/wiki/World_Wide_Web

C:\Users\Kevin\Storage\CS\CS172\final-project>
```



```
C:\Users\Kevin\Storage\CS\CS172\final-project>py run.py 1 --pages 5 --url https://en.wikipedia.org/wiki/Genshin_Impact
Parsing page:  https://en.wikipedia.org/wiki/Genshin_Impact
Downloading page:  https://en.wikipedia.org/wiki/Genshin_Impact
Parsing page:  https://en.wikipedia.org/wiki/Video_game_developer
Downloading page:  https://en.wikipedia.org/wiki/Video_game_developer
Parsing page:  https://en.wikipedia.org/wiki/MiHoYo
Downloading page:  https://en.wikipedia.org/wiki/MiHoYo
Parsing page:  https://en.wikipedia.org/wiki/Video_game_publisher
Downloading page:  https://en.wikipedia.org/wiki/Video_game_publisher
Parsing page:  https://en.wikipedia.org/wiki/Video_game_producer
Downloading page:  https://en.wikipedia.org/wiki/Video_game_producer

C:\Users\Kevin\Storage\CS\CS172\final-project>
```

Part 2 - Indexer

Architecture

- Wikipedia_crawler.py
- Wikepedia_page.py
- Sources.txt
- Indexer.py
- Index.py

Indexer.py This python file is responsible for handling the indexing and searching. All of the elasticsearch code is handled, including checking if the server is running before attempting to index. The indexer also clears the index before a new one to eliminate unwanted data.

Index.py This python file is responsible for parsing the command line. It requires a query and also provides additional flags.

Setup Instructions

Step 1: Clone the repository.

> $git clone <repository url>

Step 2: Check Python Version. This project requires Python3. More information can be found at https://www.python.org/download/releases/3.0/

> $python --version

Step 3: Install the packages. Make sure you are in the folder while doing so

> $pip3 install -r requirements.txt

Step 4: Go to https://www.elastic.co/downloads/elasticsearch. Download and unzip the files

Step 5: Open a seperate terminal to run your elasticsearch. Go to the elasticsearch folder and start the elasticsearch server

> $bin/elasticsearch (or bin/elasticsearch.bat on windows)
> Example code: $elasticsearch-7.13.1/bin/elasticsearch.bat

Step 6: On the original terminal and check if your elasticsearch server is running. If you receive a response and not an error, it is working. More information can be found at https://www.elastic.co/downloads/elasticsearch

> $curl http://localhost:9200/

Step 4: Run the searcher with a specified query by typing py index.py [query]
Run py index.py -h to pull up a help GUI of possible commands.
Possible flags are:
> --depth
> --url
> --dir
> --pages
> --clean

Run Instructions

The main command to run is py index.py

Query - This string will be what the indexer will be searching for

Depth - This number will determine how nested the crawler should go.
> If 0 is given, only the starting pages will be crawled
> If 1 is given, only the starting pages and the links on those pages will be crawled.
> If 2 is given, only the starting pages and 2 links in will be crawled.
> If -1 is given, there will be no maximum depth. WARNING: This is highly discouraged

Help - Will display the help message with descriptions on all the flags

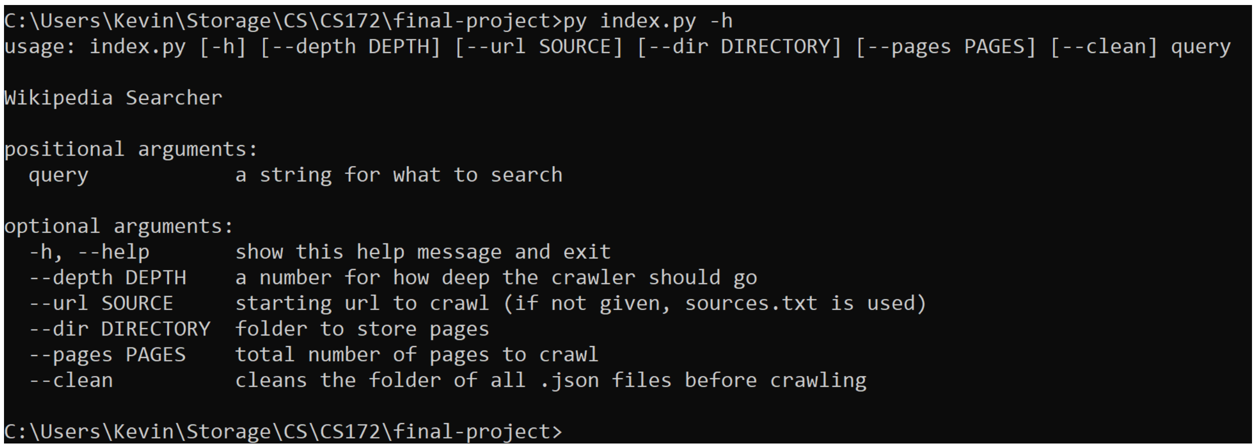
URL - Starting URL to crawl from. If this is not provided, sources.txt will automatically be used. Sources.txt can contain a URL on each new line, and the crawler will crawl through each.
> If `https://en.wikipedia.org/wiki/Web_crawler` is given, that will be the starting URL
> If nothing is given, sources.txt will be used to supply the URLs

Directory - This will determine the name for the folder where the .json files are stored in. If none is given, the name `data` is automatically set.
> If `sample_data` is given, then the folder will be named `sample_data`
> If none is given, then the folder will be named `data`

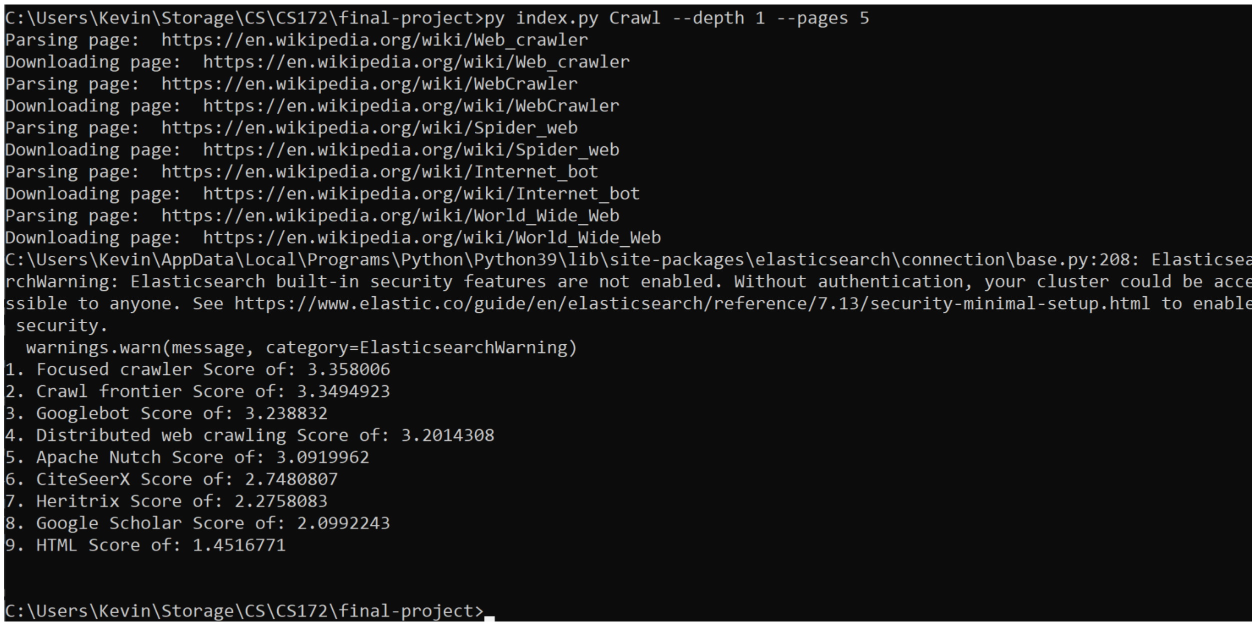
Pages - This will set the total number of pages to crawl. If none is provided, -1 will be set and there will be no maximum.
> If 0 is given, no pages will be crawled
> If 10 is given, 10 pages will be crawled
> If -1 is given, there will be no limit to how many pages is crawled

Clean - This is a flag that does not take on a value. If this flag is provided, the crawler will first clean the folder marked for the .json files before repopulating it.

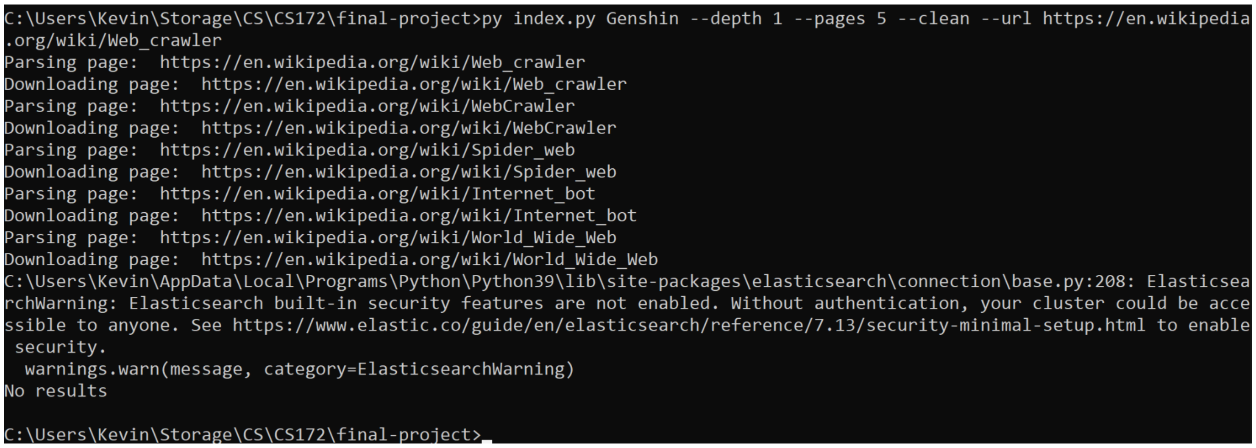
Screenshots



```
C:\Users\Kevin\Storage\CS\CS172\final-project>py index.py -h
usage: index.py [-h] [--depth DEPTH] [--url SOURCE] [--dir DIRECTORY] [--pages PAGES] [--clean] query

Wikipedia Searcher

positional arguments:
  query            a string for what to search

optional arguments:
  -h, --help       show this help message and exit
  --depth DEPTH    a number for how deep the crawler should go
  --url SOURCE     starting url to crawl (if not given, sources.txt is used)
  --dir DIRECTORY  folder to store pages
  --pages PAGES    total number of pages to crawl
  --clean          cleans the folder of all .json files before crawling

C:\Users\Kevin\Storage\CS\CS172\final-project>
```



```
C:\Users\Kevin\Storage\CS\CS172\final-project>py index.py Crawl --depth 1 --pages 5
Parsing page:  https://en.wikipedia.org/wiki/Web_crawler
Downloading page:  https://en.wikipedia.org/wiki/Web_crawler
Parsing page:  https://en.wikipedia.org/wiki/WebCrawler
Downloading page:  https://en.wikipedia.org/wiki/WebCrawler
Parsing page:  https://en.wikipedia.org/wiki/Spider_web
Downloading page:  https://en.wikipedia.org/wiki/Spider_web
Parsing page:  https://en.wikipedia.org/wiki/Internet_bot
Downloading page:  https://en.wikipedia.org/wiki/Internet_bot
Parsing page:  https://en.wikipedia.org/wiki/World_Wide_Web
Downloading page:  https://en.wikipedia.org/wiki/World_Wide_Web
C:\Users\Kevin\AppData\Local\Programs\Python\Python39\lib\site-packages\elasticsearch\connection\base.py:208: Elasticsea
rchWarning: Elasticsearch built-in security features are not enabled. Without authentication, your cluster could be acce
ssible to anyone. See https://www.elastic.co/guide/en/elasticsearch/reference/7.13/security-minimal-setup.html to enable
 security.
  warnings.warn(message, category=ElasticsearchWarning)
1. Focused crawler Score of: 3.358006
2. Crawl frontier Score of: 3.3494923
3. Googlebot Score of: 3.238832
4. Distributed web crawling Score of: 3.2014308
5. Apache Nutch Score of: 3.0919962
6. CiteSeerX Score of: 2.7480807
7. Heritrix Score of: 2.2758083
8. Google Scholar Score of: 2.0992243
9. HTML Score of: 1.4516771


C:\Users\Kevin\Storage\CS\CS172\final-project>_
```



```
C:\Users\Kevin\Storage\CS\CS172\final-project>py index.py Genshin --depth 1 --pages 5 --clean --url https://en.wikipedia
.org/wiki/Web_crawler
Parsing page:  https://en.wikipedia.org/wiki/Web_crawler
Downloading page:  https://en.wikipedia.org/wiki/Web_crawler
Parsing page:  https://en.wikipedia.org/wiki/WebCrawler
Downloading page:  https://en.wikipedia.org/wiki/WebCrawler
Parsing page:  https://en.wikipedia.org/wiki/Spider_web
Downloading page:  https://en.wikipedia.org/wiki/Spider_web
Parsing page:  https://en.wikipedia.org/wiki/Internet_bot
Downloading page:  https://en.wikipedia.org/wiki/Internet_bot
Parsing page:  https://en.wikipedia.org/wiki/World_Wide_Web
Downloading page:  https://en.wikipedia.org/wiki/World_Wide_Web
C:\Users\Kevin\AppData\Local\Programs\Python\Python39\lib\site-packages\elasticsearch\connection\base.py:208: Elasticsea
rchWarning: Elasticsearch built-in security features are not enabled. Without authentication, your cluster could be acce
ssible to anyone. See https://www.elastic.co/guide/en/elasticsearch/reference/7.13/security-minimal-setup.html to enable
 security.
  warnings.warn(message, category=ElasticsearchWarning)
No results

C:\Users\Kevin\Storage\CS\CS172\final-project>_
```

Part 3 - Website

For our part 3, we decided to create a Web-based interface. We also used Python Django Framework to make a basic user interface. The interface contains a Query, URL, Pages, and Depth textboxes. It also includes a button to search. Additionally, the Query and URL textboxes are required.

Architecture

```
Repository/
        website/
                templates/
                        Home.html
                Settings.py
                Urls.py
```

Views.py
Indexer.py
Wikipedia_crawler.py
Wikipedia_page.py
Manage.py

Home.html This html file is responsible for the layout of the webpage.

Settings.py This python file is responsible for the settings of the server. This was lightly touched with, only to manipulate file names.

Urls.py This python file is responsible for the different webpages the server hosts.

Views.py This python file is responsible for managing the functionality for the web page. The external function queries the user's input against the indexer, and displays the result onto the webpage.

Indexer.py This python file is responsible for searching the query. This was slightly modified compared to the indexer in part 2 to account for the lack of parameters given. This indexer always recrawls and fills in the data folder since each query is accompanied with a url.

Setup Instructions

Step 1: Clone the repository.

```
> $git clone <repository url>
```

Step 2: Check Python Version. This project requires Python3. More information can be found at https://www.python.org/download/releases/3.0/

```
> $python --version
```

Step 3: Install the packages. Make sure you are in the folder while doing so

```
> $pip3 install -r requirements.txt
```

Step 4: Go to https://www.elastic.co/downloads/elasticsearch. Download and unzip the files

Step 5: Open a seperate terminal to run your elasticsearch. Go to the elasticsearch folder and start the elasticsearch server

```
> $bin/elasticsearch (or bin/elasticsearch.bat on windows)
> Example code: $elasticsearch-7.13.1/bin/elasticsearch.bat
```

Step 6: On the original terminal and check if your elasticsearch server is running. If you receive a response and not an error, it is working. More information can be found at https://www.elastic.co/downloads/elasticsearch

```
> $curl http://localhost:9200/
```

Step 7: Open a separate terminal to run your website. Alternatively, you could use the same original terminal. Go the the repository folder and start the website server

```
> $py manage.py runserver
```

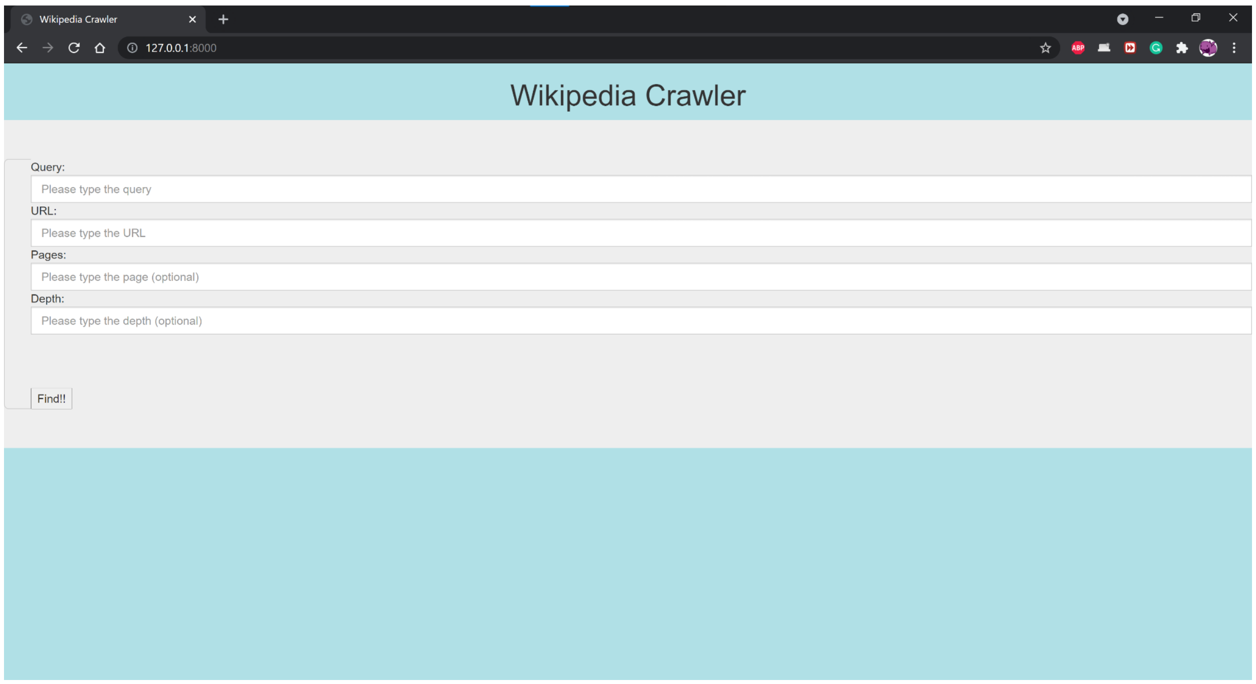
Run Instructions

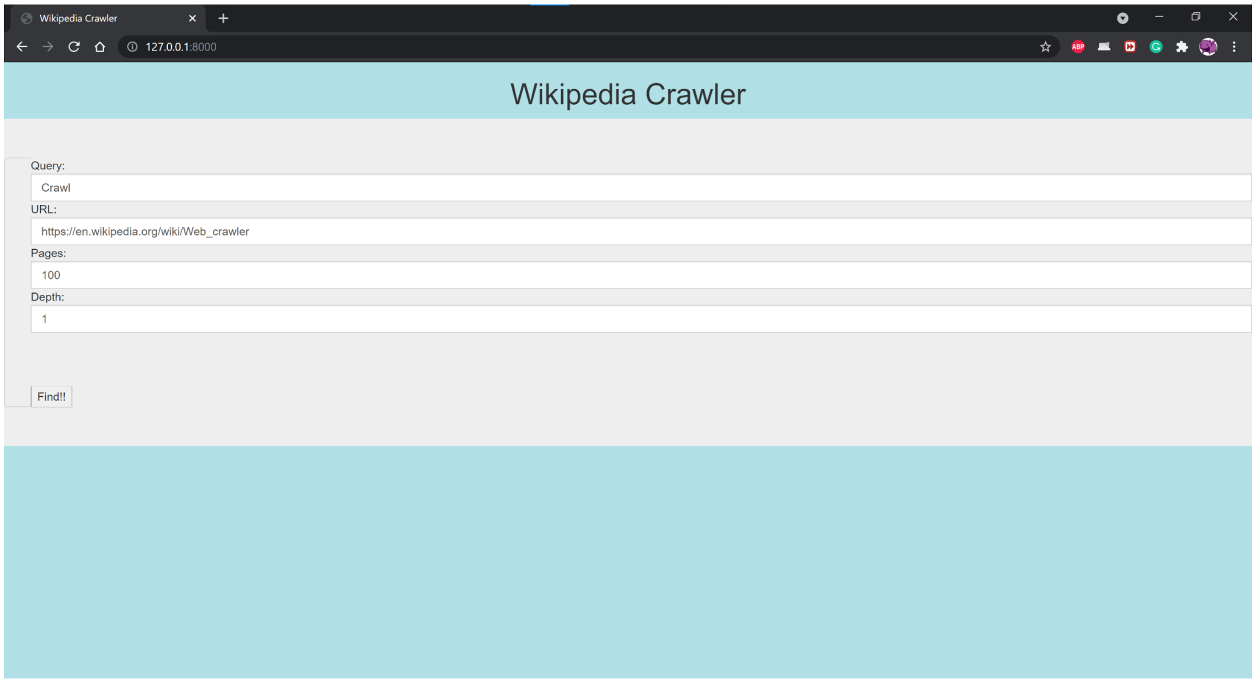
Go to a web browser and enter the URL: http://127.0.0.1:8000/

This will bring you to a webpage. There are 4 text boxes available, with Query and URL being mandatory.

Proceed to click the "find!!" button to start your search.

Screenshots





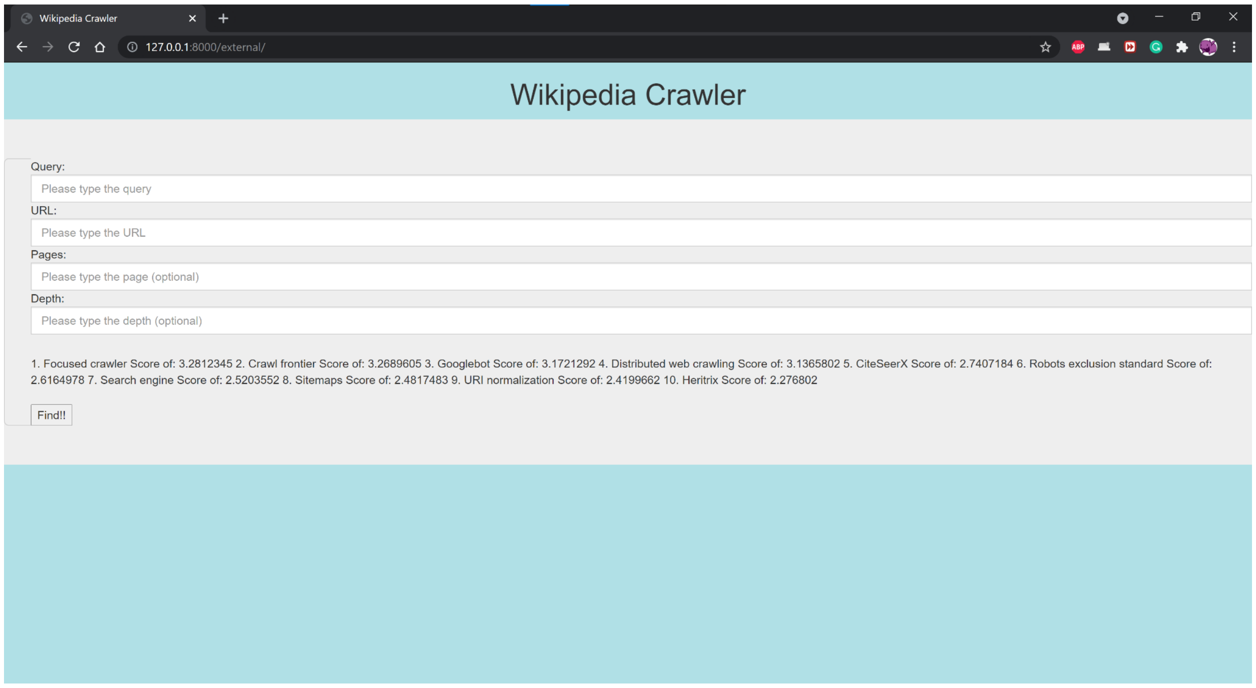
Wikipedia Crawler

Query: Crawl

URL: https://en.wikipedia.org/wiki/Web_crawler

Pages: 100

Depth: 1



Wikipedia Crawler

Query: Please type the query

URL: Please type the URL

Pages: Please type the page (optional)

Depth: Please type the depth (optional)

1. Focused crawler Score of: 3.2812345 2. Crawl frontier Score of: 3.2689605 3. Googlebot Score of: 3.1721292 4. Distributed web crawling Score of: 3.1365802 5. CiteSeerX Score of: 2.7407184 6. Robots exclusion standard Score of: 2.6164978 7. Search engine Score of: 2.5203552 8. Sitemaps Score of: 2.4817483 9. URI normalization Score of: 2.4199662 10. Heritrix Score of: 2.276802