Assignment

1.0 Problem Statement:

Implement a command line program that can fetch web pages and saves them to disk for later retrieval and browsing.

Section 1

For example, if we invoked your program like this: `./fetch [https://www.google.com](https://www.google.com)` then in our current directory we should have a file containing the contents of `www.google.com`. (i.e. `/home/myusername/www.google.com.html`).

Section 2

Record metadata about what was fetched:

- What was the date and time of the last fetch
- How many links are on the page
- How many images are on the page

Modify the script to print this metadata.

For example (it can work differently if you like)

2.0 Approaches/Discussion

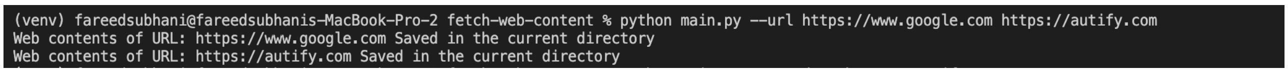
- There are many approaches to this problem

- For ***section-1***

- I have used Python reqeusts library to get the URL Content

- From the command line if you run this command

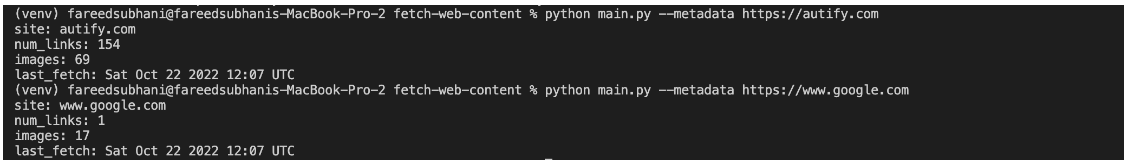
- > python main.py --url https://www.google.com https://autify.com < ... >

- Demo


```
(venv) fareedsubhani@fareedsubhanis-MacBook-Pro-2 fetch-web-content % python main.py --url https://www.google.com https://autify.com
Web contents of URL: https://www.google.com Saved in the current directory
Web contents of URL: https://autify.com Saved in the current directory
```

  - main fucntion calls downloadHTML class which args as an argument.

- Then it will process each URL and save it in the local disk with the filename of the website name + html.

- Note that I have used **stream = True** in the GET Method and have used chunk_size in each iter_contens
- *REASON* : This avoids reading the content all at once into memory for large responses. Especially if we have to deal with large images/video files.
- For ***section-2***
  - From the command line if we execute this command
  - > python main.py --metadata https://www.google.com
  - As shown in this image:
    - 

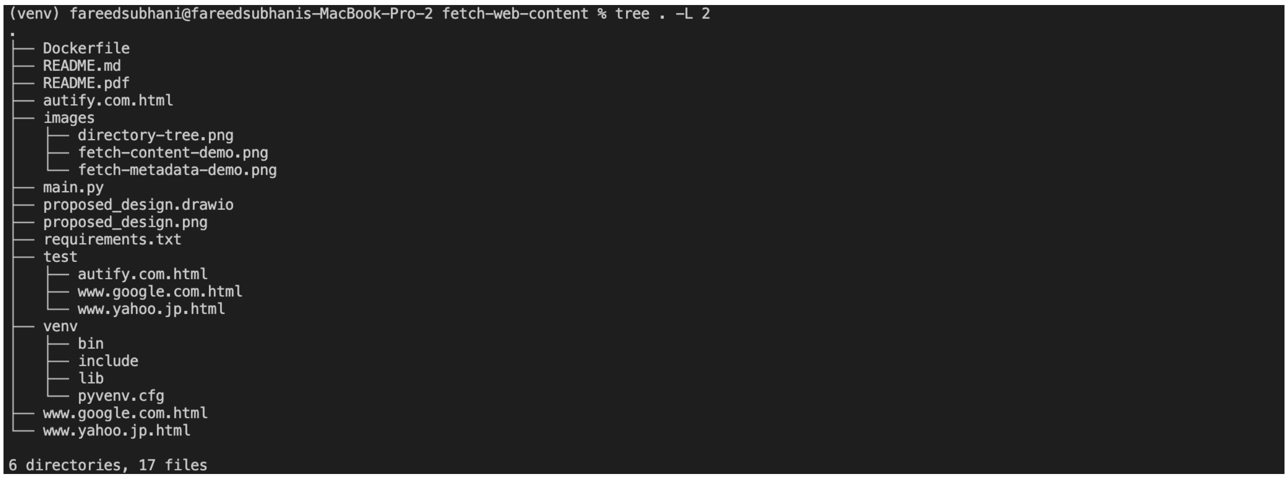
```
(venv) fareedsubhani@fareedsubhanis-MacBook-Pro-2 fetch-web-content % python main.py --metadata https://autify.com
site: autify.com
num_links: 154
images: 69
last_fetch: Sat Oct 22 2022 12:07 UTC
(venv) fareedsubhani@fareedsubhanis-MacBook-Pro-2 fetch-web-content % python main.py --metadata https://www.google.com
site: www.google.com
num_links: 1
images: 17
last_fetch: Sat Oct 22 2022 12:07 UTC
```

  - We needed to find mainly three pieces of information
    - date and time of the last fetch
    - Number of links in the content
    - Number of Images in the content
  - *Approach*
    - Last-Fetch:
      - I have used the Python os stats module to get the last date and time of the modified file.
      - Again I have used the datetime module and formatted the DateTime object using strftime in the desired output format.
    - Count of Images and URLs:
      - I have used the getMetadata class which again calls the ProcessMetadata class
      - I have used BeautifulSoup library to get the count of images and URL tags from the HTML file.

3.0 Testing and running

- System environment:
  - Python 3.9.0
  - MacBook-Pro(macOS Monterey v12.5)

- Directory Tree: -



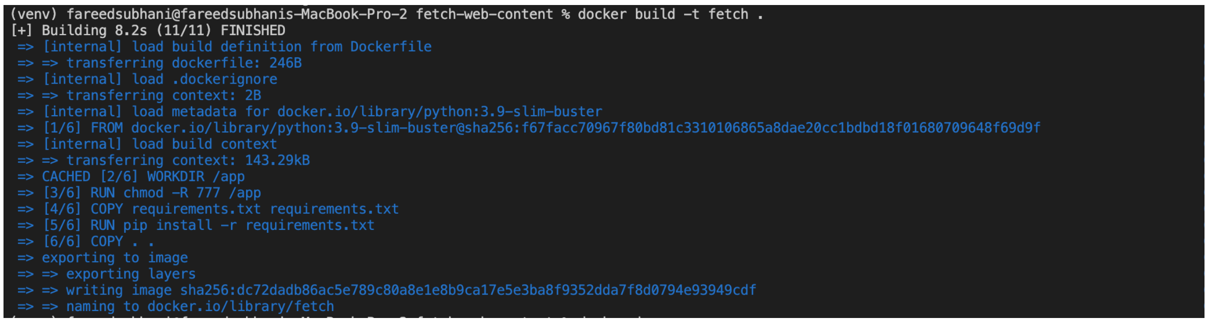
```
(venv) fareedsubhani@fareedsubhanis-MacBook-Pro-2 fetch-web-content % tree . -L 2
.
├── Dockerfile
├── README.md
├── README.pdf
├── autify.com.html
├── images
│   ├── directory-tree.png
│   ├── fetch-content-demo.png
│   └── fetch-metadata-demo.png
├── main.py
├── proposed_design.drawio
├── proposed_design.png
├── requirements.txt
├── test
│   ├── autify.com.html
│   ├── www.google.com.html
│   └── www.yahoo.jp.html
├── venv
│   ├── bin
│   ├── include
│   ├── lib
│   └── pyvenv.cfg
├── www.google.com.html
└── www.yahoo.jp.html

6 directories, 17 files
```

- ***Build and runing dockerfile***

  - Build a docker file using the command below

    > docker build -t fetch .

    ```
    (venv) fareedsubhani@fareedsubhanis-MacBook-Pro-2 fetch-web-content % docker build -t fetch .
    [+] Building 8.2s (11/11) FINISHED
     => [internal] load build definition from Dockerfile
     => => transferring dockerfile: 246B
     => [internal] load .dockerignore
     => => transferring context: 2B
     => [internal] load metadata for docker.io/library/python:3.9-slim-buster
     => [1/6] FROM docker.io/library/python:3.9-slim-buster@sha256:f67facc70967f80bd81c3310106865a8dae20cc1bdbd18f01680709648f69d9f
     => [internal] load build context
     => => transferring context: 143.29kB
     => CACHED [2/6] WORKDIR /app
     => [3/6] RUN chmod -R 777 /app
     => [4/6] COPY requirements.txt requirements.txt
     => [5/6] RUN pip install -r requirements.txt
     => [6/6] COPY . .
     => exporting to image
     => => exporting layers
     => => writing image sha256:dc72dadb86ac5e789c80a8e1e8b9ca17e5e3ba8f9352dda7f8d0794e93949cdf
     => => naming to docker.io/library/fetch
    ```

  - execute this command

    > docker run --rm -it fetch bash

  - now you are in

    > root@build-image:/app# directory

  - run the usual command i.e.

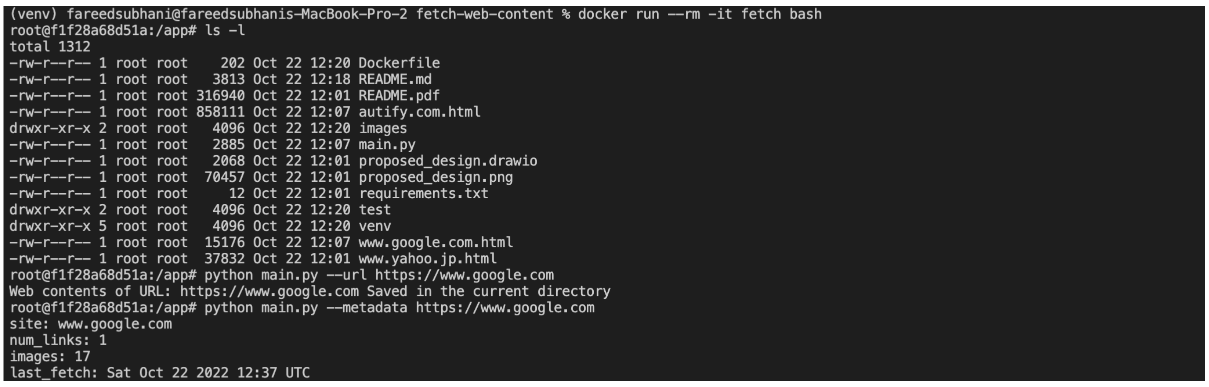
    > python main.py --url https://www.google.com

    to fetch URL content and

    > python main.py --metadata https://www.google.com

    to get metadata of the URL

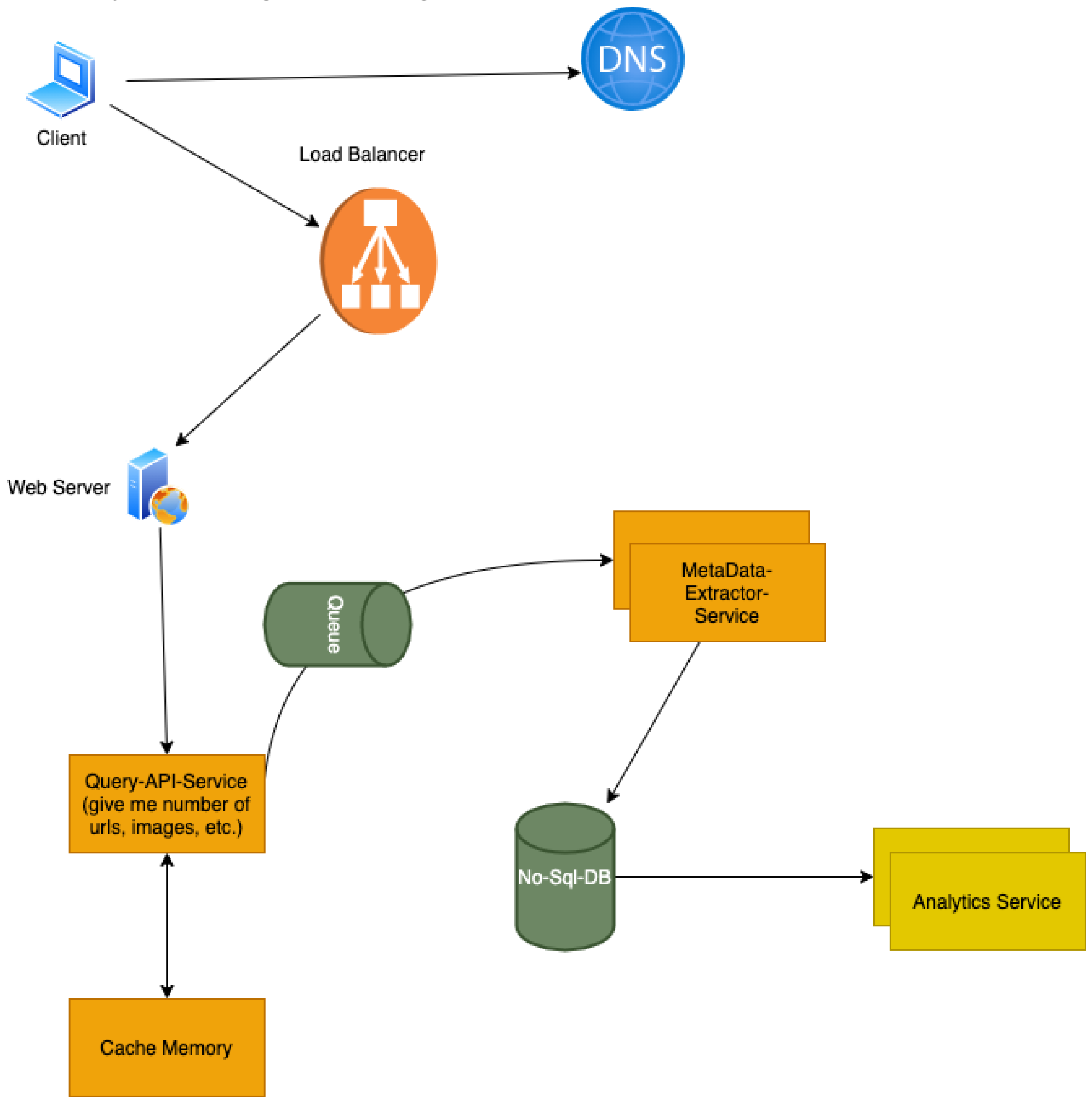
- here is a demo for the same



```
(venv) fareedsubhani@fareedsubhanis-MacBook-Pro-2 fetch-web-content % docker run --rm -it fetch bash
root@f1f28a68d51a:/app# ls -l
total 1312
-rw-r--r-- 1 root root    202 Oct 22 12:20 Dockerfile
-rw-r--r-- 1 root root   3813 Oct 22 12:18 README.md
-rw-r--r-- 1 root root 316940 Oct 22 12:01 README.pdf
-rw-r--r-- 1 root root 858111 Oct 22 12:07 autify.com.html
drwxr-xr-x 2 root root   4096 Oct 22 12:20 images
-rw-r--r-- 1 root root   2885 Oct 22 12:07 main.py
-rw-r--r-- 1 root root   2068 Oct 22 12:01 proposed_design.drawio
-rw-r--r-- 1 root root  70457 Oct 22 12:01 proposed_design.png
-rw-r--r-- 1 root root     12 Oct 22 12:01 requirements.txt
drwxr-xr-x 2 root root   4096 Oct 22 12:20 test
drwxr-xr-x 5 root root   4096 Oct 22 12:20 venv
-rw-r--r-- 1 root root  15176 Oct 22 12:07 www.google.com.html
-rw-r--r-- 1 root root  37832 Oct 22 12:01 www.yahoo.jp.html
root@f1f28a68d51a:/app# python main.py --url https://www.google.com
Web contents of URL: https://www.google.com Saved in the current directory
root@f1f28a68d51a:/app# python main.py --metadata https://www.google.com
site: www.google.com
num_links: 1
images: 17
last_fetch: Sat Oct 22 2022 12:37 UTC
```

4.0 Future Design and Thoughts

- Due to time constraints I was not able to code it in a better way
- I could have introduced a feature like saving all the contents
- We can make the code more modular for better code-reusability
- We can introduce various design patterns.
- Of course, this solution is not very practical if needs to handle very large requests from the user.
- I have come to a design if I were designing this service
  - Here is my idea for High-level design


Client
DNS
Load Balancer
Web Server
Queue
MetaData-Extractor-Service
Query-API-Service (give me number of urls, images, etc.)
No-Sql-DB
Analytics Service
Cache Memory

    - Again this is also not a perfect solution As we can improve a lot of components
- Personally, I really enjoyed solving this problem, particularly scaling part, when we have to design and implement this service we have to think about scaling and best practices as well, Thank you!