Timeline report

Phase 1:

I divided phase 1 work into the following steps:

(i) By building a crawler/scraper, scraping the data from the medium website.

(ii) Building a web page using the frontend containing a search bar, where we can search the keywords or tags for the blog.

(iii) Building a database where we can store the scraped data from the medium.

(iv) Building backend using Django for taking the data from the database.

(v) Building APIs which will send requests from frontend to backend and will bring back the scraped data and display it to the user.

(vi) Integrating all these modules.

**FIRST DAY:**

It took me one day to build the scrapper.

I first started searching for a scraper and read many articles and blogs. I gave 2-3 hours to the research and reading part. After gathering information, I decided to build a basic project on the scraper. After completing it, I learned to solve many practical issues.

For building the scraper for the assignment:

I inspected the medium website to understand the working of the web app. Then I found some logic from which I can scrap the details of the website as mentioned in the assignment.

So, firstly what I found is the working of the URL of the medium, such that if a user wants to search some blogs then how the website will react and how the link works, so I found a common link there which I declared in my program(default\_url). After that, I declared a route so that if a user wants to search anything he can go through it.

Here, I used bs4 and imported beautifulsoup.

When I was inspecting the website of the medium I found some scripts tag in the blogs as well as in the index page of the medium, so then I used script tag as a medium of extraction of data with the help of indexing and I found that if I use indexing then after 9th place there was the title of the blog so I declare it as (y = x[9]) and created it as a string.

After that, I declare the flag and count to use it while looping to print the title. Then simply use if to trace the values, and the good thing in the website is after the title, the content of the blog is present, so simply used my flag there to print the title and the content of the website.

With that logic now I have to print or scrap the author of the blog, here I used two conditions if the title contains the name of the author and the username of the blog author.

I simply use indexing there to print the username and author’s username there, you can check it out!

Now, for finding the name of the blog writer, I used previous output as a medium to find the writers name, I inspected the site and I found it can be done with strings, and I did it ( colonFind2 = t\_temp.find(new\_string[0], name + 9))

With this ease, I was also able to find the writer’s name.

Here comes in the mind how I was able to work it through it! simply I just scrap the script as the output on my jupyter notebook.

Here is the code of inspection:

#playground

x = soup.find\_all("script")

y = x[9]

f = 0

y = y.string

u = 0

while u != -1:

u = y.find("title", f)

f = u + 4

#print(str(u)+ " " + y[u:u + 50])

print(y[u-600:u-150])

print("\n")

**NEXT HALF DAY:**

Step(ii)

It took me half-day to build the front end. I used the React app for the frontend. On the web page, I included a search bar and a search icon for the user to search the tags or keywords related to the blog. After clicking the button, the data is to be fetched and displayed on the screen by using the onclick function. It displayed a total of 10 blogs per search.

**OTHER HALF-DAY:**

The other half-day I tried to build a database as well as work on the backend part but wasn't able to complete it.

Phase 2:

In advanced features, I included a sort by drop down on my web page where the user can sort the extracted data in many ways. For example: displaying the author's name in alphabetical order, latest blogs, mostly views blogs, increasing the time of min read.

I also included a filter section as an additional feature. In this user can filter the data displayed according to his/her needs. Here, I included blog-type, which is for the user to define what type of blog data he/she wants. Whether it's science, news, economics, etc. I also included a slider for the min read so that the user can define a time span of the min read.

I also tried to build the backend part for the filter section. It was easier than I thought. I made the backend part for the filter section using Django.

Phase 3:

In phase 3, I decided to display the content scraped from the site to be displayed in cards. I also included pagination and decided to display the data such that if we want to display data of 10 blogs then the first five cards should appear then after clicking on the arrow for the next page next five cards are displayed.