**Task #1 Gab Data Scraping using Python**

Given a website to work with, picking the right tool for web scraping depends mainly on what you’re trying to achieve, what kind of website you’re scraping, and what to do with the data. For this specific task, we are asked to extract the following information to monitor users from the social media platform Gab:

* Date joined
* User name
* User image
* Cover photo
* About
* Number of Gabs
* Number of followers
* Number of following
* Last 50 posts (including media in case it exists)
* Average engagement of the posts

Python is our main tool for accomplishing this task, with the limitation that we are not allowed to use any platform related third party code. This also includes the prohibition of any official platform API. With these in mind, we first look at the structure of the website. This was done using the Chrome browser with its built-in Developer tools.

At first glance, the user page for the Gab website doesn’t require you to login to see the content. This saves us the time to consider or give up credentials to access the site. Upon inspection using the Developer tools, it can be seen that the classes of the elements are unusual or very hard to comprehend. For example, we have classes with values likethose shown in the right side of Figure 1. Furthermore, by navigating to the Fetch/XHR filter from the network tab, we can see all the requests that use the Fetch API as shown in Figure 2. This allows to make HTTP request in JavaScript.

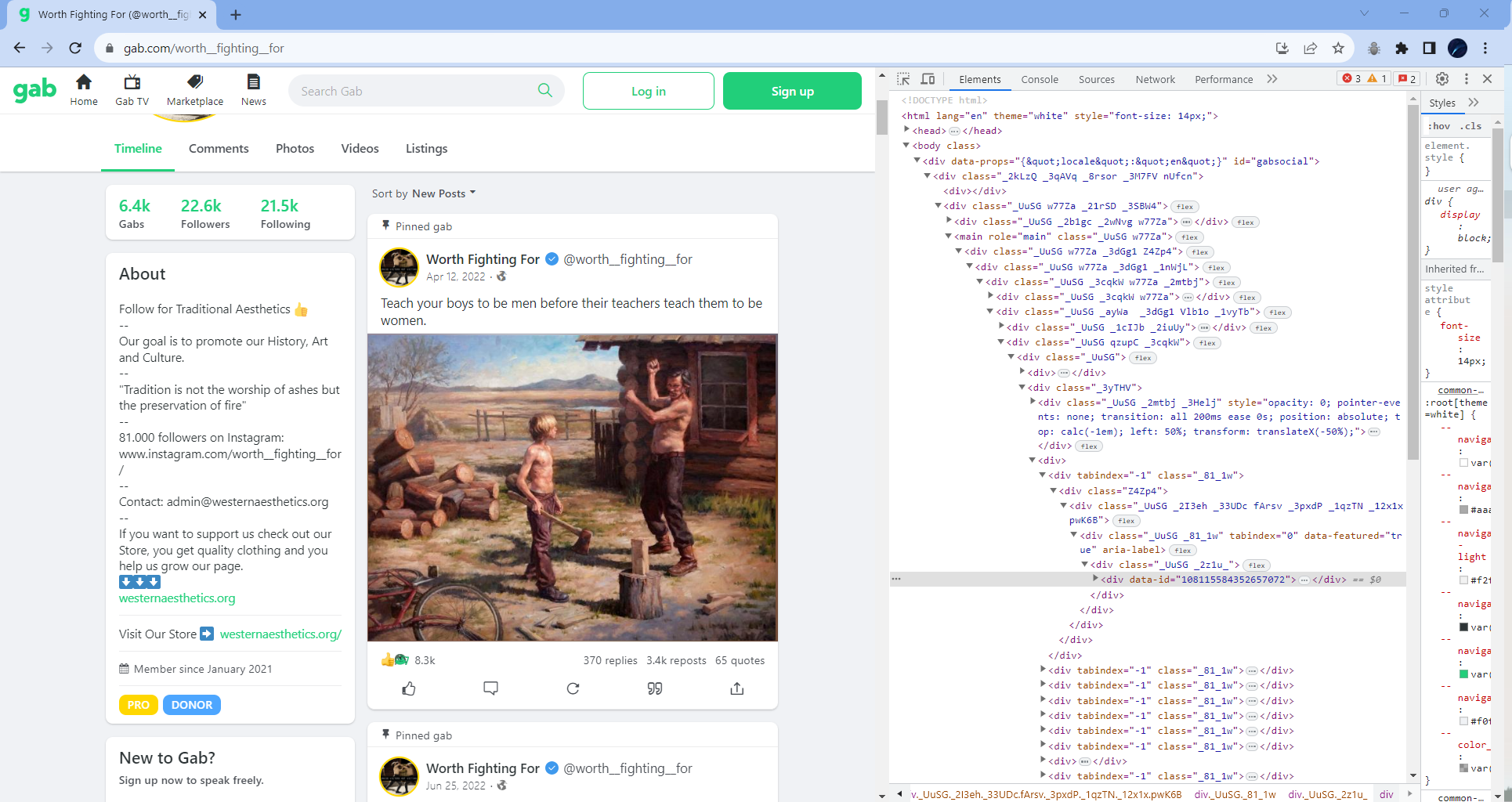


Figure 1. HTML structure of the profile page of a Gab user

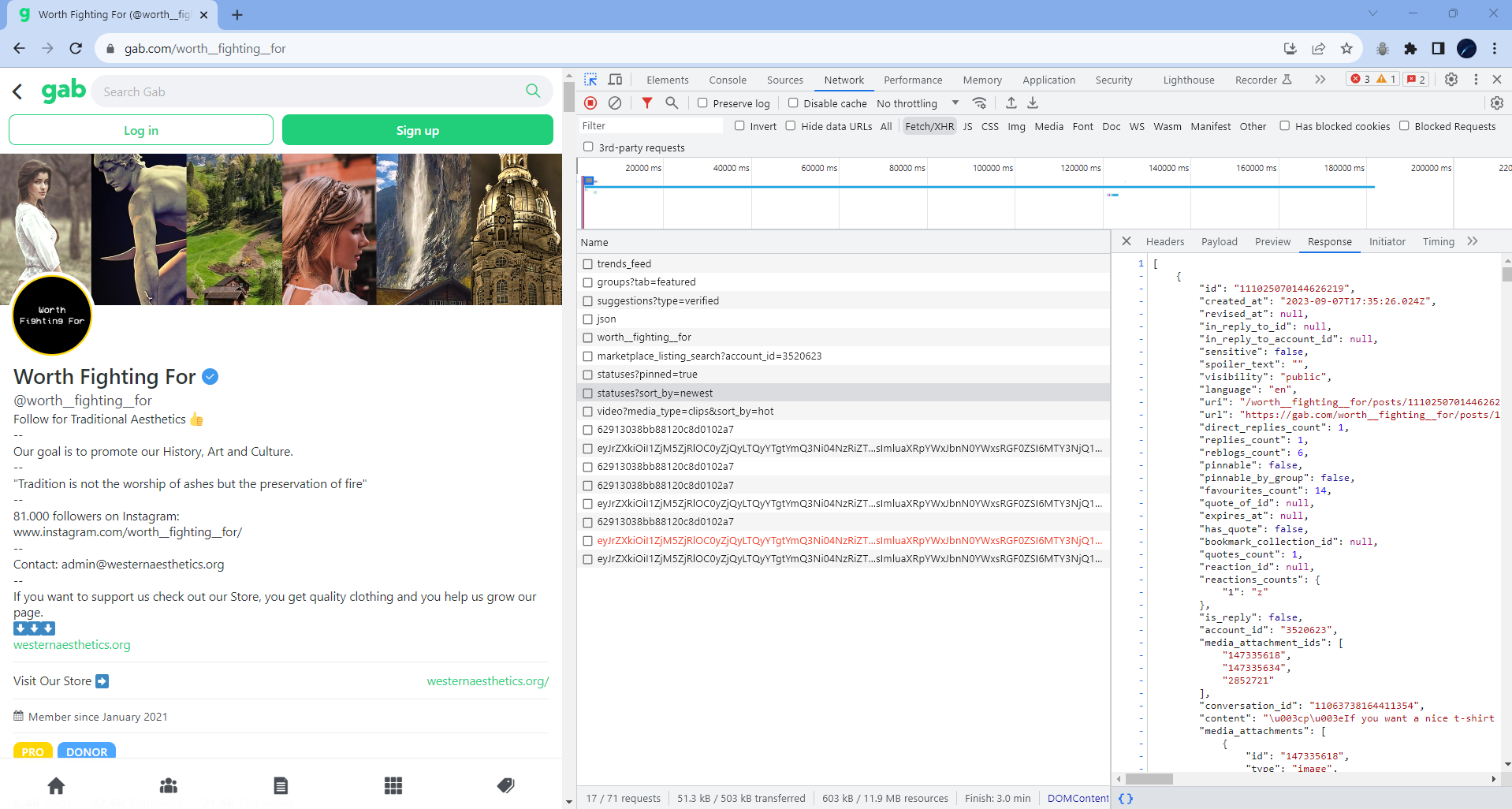


Figure 2. Filtered network responses of the website upon inspecting with Developer tools

The inspection made tells us that website is modern in nature allowing its data to be rendered heavily through JavaScript. This means that the data extraction from the front-end would be tricky. Generally, it would be easier to access the information from the back-end or through APIs.

Since we must build our own scraper in order to understand how data is structured in the platform, we ought to find another way. In this task, we used the selenium library in Python. The website was fully requested with a successful response using selenium, as opposed to when using BeautifulSoup or Scrapy, which may require more advanced stuff to perform.

First, we created a virtual environment to help keep the dependencies separated. VS Code was used as the IDE. Then, different modules/libraries were installed including selenium, pandas, and time. The list of dependencies are listed in the *requirements.txt* file. During the coding part, these modules were first imported and configured. Taking into account that approximately 100 users per day would be monitored, a user list was made in order to make the code scalable. The website was then inspected to see where the needed data can be located.

Data extraction was primarily done through CSS selectors. It was made sure that each selector is unique to get data accurately. Exception handling was also implemented to users and posts with missing data. This include users that are relatively new with no posts yet and posts with no engagements. For such cases, no values are explicitly stated.

The initial output generated by the program is a list object (*users\_info*) containing the info for each input user. Each user info is contained in a dictionary (*info*). This is similar to JSON in format so it can be viewed as in Figure 3 using a JSON viewer (). The final output is a csv file named *gab\_data.csv*.

The code and output can be accessed from my GitHub profile as indicated below.

https://github.com/benicastro/platform-research/tree/main/scrape\_gab/selenium



Figure 3. Structure of the output in JSON format for the first task

**Task #2 Data Collection from Discord using Python**

In attempting to scrape data from websites, it is important to note about any ethical and legal concerns in the process of extracting data. It must be ensured that the platform’s policies and applicable laws and regulations are followed. Given that these things are considered, here are some options in researching the platform.

* It is essential to read the Terms of Service and Privacy Policy of the given platform. This helps you identify key actions regarding the access and usage of data. It may also help in understanding what kind of data you are working with and how it can be used.
* One easy way of extracting data from websites is the utilization of APIs. It can come handy if you can research about the platform’s API endpoints, authentication methods, and data format in the developer documentation. This lets you plan on how you can access the data.
* The traditional but most straightforward way is to inspect the website. You can start by visiting the platform’s website to gain knowledge about its structure and features. Browser developer tools can also be used to view the page’s HTML source code, giving you enough information to locate the data you need.

Furthermore, in market researching for available APIs, data vendors, and other scraping tools, we can do the following:

* You can research for API marketplaces since some platforms host their APIs on marketplaces. They can let you browse, test, and subscribe to APIs. Online directories that specialize in datasets are always present.
* In your specific industry, you can research for well-known data vendors and providers. One sure way is to connect with professionals and seek recommendations for reliable and trusted data vendors.
* As for scraping tools, you can research for web scraping libraries and frameworks that are best suited for the job you are doing. Few examples include Beautiful Soup, Scrapy, and Selenium. This can be done through people’s reviews and comparisons of each tool available in many platforms. Consulting with developer communities can also help in this scenario.

In terms of monitoring and measuring the collection success of your scraping process, you can implement a comprehensive way of recording all actions, errors, and responses encountered. Error handling is one crucial component in this process, as well as alert systems which notifies you if there are any failures that may have occurred. The data collected should also be validated through some baseline info to aid in identifying potential discrepancies or missing data.

As proof of concept, we are tasked to collect data from this server: https://discord.com/invite/honkai-starrail. Upon visiting the website and joining the server, it was very clear that user authentication is required. Thus, we provided user details as part of the script. Similar to the previous task, we have employed here selenium. Given that this library is very powerful in web automation and testing, we can use it to interact with the website for logging in.

Same methodologies were utilized here in locating the data needed, including the use of CSS locators. The final output is stored in a csv file (*discord\_data.csv*). This include the following details:

* Server name
* Server ID
* Members count
* Channels count
* Last message from general chat
  + User
  + Text
  + Media
  + Time

A sample result in JSON format is shown in Figure 4. The code and output can be accessed from my GitHub profile as indicated below.

https://github.com/benicastro/platform-research/tree/main/scrape\_gab/selenium

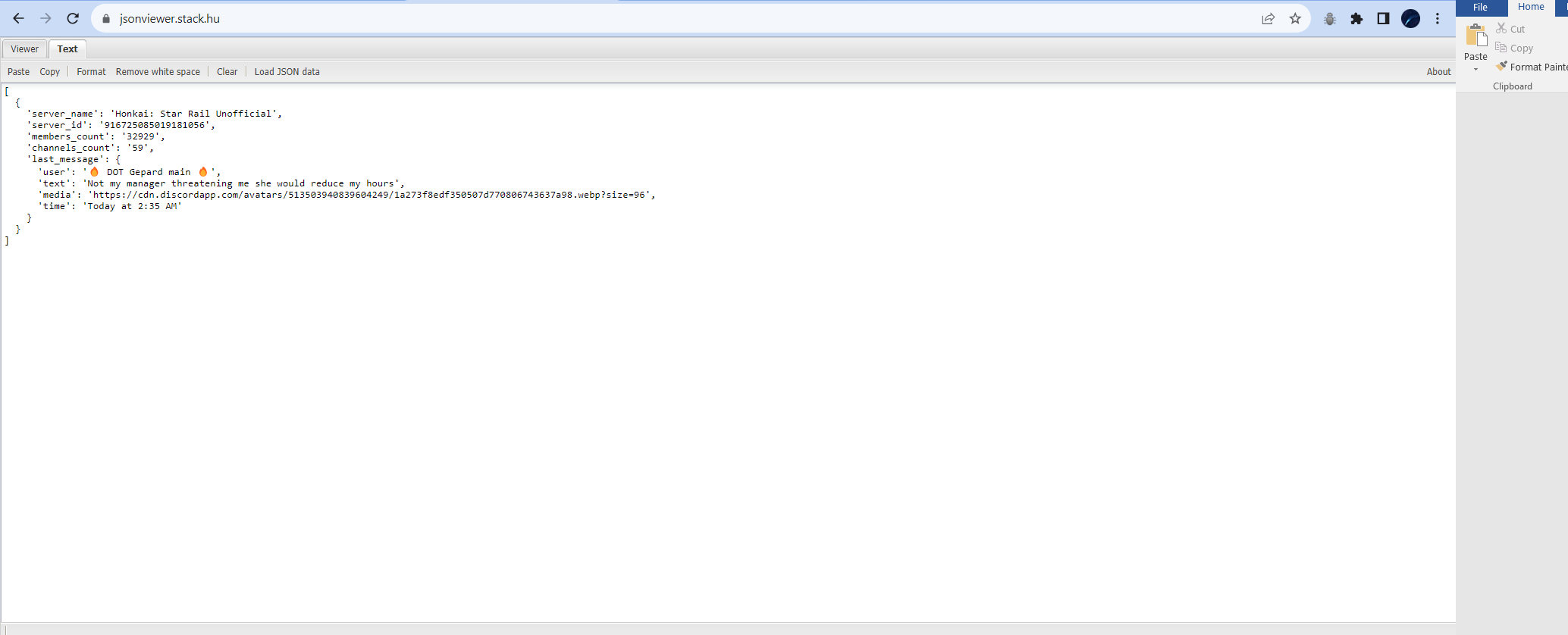


Figure 3. Structure of the output in JSON format for the second task

**Task #3: Collecting Violence Content from VK**

The task here involves finding 10 links from the social platform VK that contains different categories of abuse:

* Graphic Violence
* Nudity
* Selling illegal goods (drugs, prostration, and more)

The links below were found based on the workflow diagram shown in Figure 5.