



Automated Web Scraping of Comics and Anime Platforms for Content Aggregation

A. Baranidharan¹, Divya Tej Atluri², Hari Prasad S³, Madan Mohan Reddy⁴, Harsha Sailesh B⁵

¹Assistant Professor, Department of IT, Kalasalingam university, Krishnankovil, Tamil Nadu, India.

^{2,3,4,5}UG Student, Department of IT, Kalasalingam University, Krishnankovil, Tamil Nadu, India.

Email ID: baranidharan@klu.ac.i¹, 9921008125@klu.ac.in², 9921008122@klu.ac.in³, 9921008127@klu.ac.in⁴, 9921008145@klu.ac.in⁵

Abstract

This paper discusses the development and application of Web Scraping Automation in aggregating and combining data on manga and anime available on online platforms. The exponential quantity of digital information generated within a single day, whether for fandom or research, prevents them from receiving accurate, up-to-date information relating to matters like release dates or times, running chapters of manga, episodes of anime, or critique reviews. One efficient way of automation in the retrieval of these data is through web scraping, not as a method for extracting information but as a technique that gives access to extensive databases for purposes of content analysis and discovery, aside from real-time updates. This automated web-scraping goes about handling manga and anime content search and arrangement by removing the required handling of manual data collection. Through this automated process, the system would become a depository that researchers, content curators, and anime enthusiasts can draw from, giving it the most contemporary well-organized representation of data.

Keywords: Web Scraping; Automation; Databases

1. Introduction

Having entered the modern digital age, web scraping has become a vital tool used to automatically gather lots of data from numerous websites by individuals and organizations. Especially with respect to manga and anime, web scraping offers a wide variety of resources in information regarding releases, character backgrounds, fan topics, ratings, and even images from other sources. This introduction will address the basic principles of web scraping, paying special attention to the automated extraction of data with relevance to manga and anime. Be it simply an enthusiast bent on producing a personal manga tracking system, the researcher investigating trends in anime, or the developer looking to design applications within this field, web scraping may be the best method through which data collection can be optimized. The manga and anime industries require a large amount of digital content. Information is freely spread, covering official databases, e-commerce platforms and fan sites, forums, and blogs. In order to keep track of the new chapters of manga or anime episodes one needs to look through many websites. It is recommended to provide a service that aggregates

user reviews, rankings, and recommendations from various sources. Another essential factor is tracking changes in prices for volumes of manga, DVDs, and other merchandise on different online retailers' portals. Gather data about character interpretations, popularity trends, or opinion analyses. Collect information systematically from different sources like character lists, story outlines, and data related to episodes. The process of systematically accessing and extracting particular information from web pages constitutes the core activity of web scraping. It often utilizes libraries such as BeautifulSoup, Scrapy, or Selenium along with programming languages like Python or JavaScript to parse the HTML and gather the information needed. The better one understands how web pages are constructed, the easier it is to find specific details, whether it be text, images, or hyperlinks. To find and extract a particular object from a web page, XPath and/or CSS selectors might be used. Sometimes, websites provide APIs that make it possible to fetch data without requesting the HTML content. In some cases, it is possible to interact with websites using JavaScript by



automatically loading the dynamic data contained within the page using tools like Selenium that act like a browser. However, ethical web scraping must always abide the copyright, intellectual property rights, and terms of service agreement stipulations that apply in those contexts. Review the scraping policies of sites you will target so you don't over-query their servers. Ethical scraping will make room not to disrespect others' efforts while maximizing on data used for projects within the very manga and anime internet domains where enthusiasts and content-creating artists thrive. By the end of this introduction, you should feel comfortable diving into creating a web scraper that will focus on content about manga and anime. At minimum, you will have some coding experience, familiarity with frameworks such as BeautifulSoup or Scrapy, and knowledge of the structural design of the websites in which you are interested in scraping.

2. Literature Review

This manuscript details the development of web scraping as a powerful tool to automatically retrieve information from internet sources, with a sharp increase in usage over the last couple of years even in niche fields such as manga and anime. Based on the automation aspect, this literature review examines the present research landscape, methodologies, resources, and challenges of web scraping and how it is applied in the manga and anime fields. The practice of web scraping has been widely researched in different fields, since it provides access and retrieval of data from the unstructured internet. There are numerous methodologies involved in web scraping that Mitra and Acharya (2019) present in their detailed analysis using BeautifulSoup, Scrapy, and Selenium tools. These tools, allowing one to navigate through the HTML architectures and interact dynamically with web pages, have become an integral part of any scraping activity. BeautifulSoup: Nice and easy, fast and efficient when parsing HTML/XML documents, which makes working with a webpage's DOM structure quite effortless. Scrapy: A more heavy-duty framework with integrated capability for request management, link tracing, and structured data parsing, built for heavy-duty scraping operations. Selenium is a widely employed

instrument for retrieving content rendered by JavaScript, as it simulates the behavior of web browsers and facilitates interaction with dynamic web pages. [1] In the light of processing large datasets, such as those found in significant industrial applications, automation of web scraping is necessary. According to Simpson et al. (2020), using cron jobs along with Python-based frameworks enables automation of the scraping process so that systematic recovery of new data could be achieved without human assistance. This aspect of automation is highly applicable in the light of manga and anime where regular new chapters or episodes are published by various parties. [2] Web scraping opportunities abound in the entertainment industry. Ghosh et al. (2018) explored the aggregation of metadata on movies from web pages such as IMDb, Rotten Tomatoes, and Metacritic through scraping. The study highlights the difficulties in extracting data from dynamic web pages, an issue that is especially common on websites devoted to anime, like My Anime List and Ani List, where JavaScript-based content may need the use of specialized methods like Selenium's headless browsing. [3]. Although there are few studies directly on web scraping in the context of manga and anime, many open-source applications have emerged that explain how scraping is performed in these industries. Web scraping is applied in various projects to gather data from online platforms, such as Manga-Dex, Crunchyroll, and Anime News Network, that have images, ratings, and even details on manga and anime. For instance, one of the most popular community-driven projects hosted on GitHub utilizes Python scripts in collecting information on the latest manga chapters to display it on personal dashboards. Huang et al. (2020) explored some other scraping methods for the metadata acquisition process from streaming services like Netflix and Hulu. This approach is particularly useful in anime streaming where scraper approaches can be applied to obtain metadata on sites like Funimation and Crunchyroll in order to consolidate the release and streaming timetables of new episodes [4]. Several works have showcased the ethical and legal problems of web scraping. Specifically for entertainment portals like manga and anime portal which secure



their copyrights, Madhavan et al. 2017 emphasizes that one should respect the terms of service of a website. The authors advise using an API when possible, such as My Anime List API that can offer access to information about anime and manga and warns developers against aggressive scraping that may result in payloads on systems.[5]. Thus, extraction of data using manga and anime websites is always hard because of anti-scraping technologies in the form of CAPTCHA systems, user-agent restrictions, or JavaScript-based rendering techniques. According to Liu et al. (2019), a few tactics to avoid these challenges are: a headless browser like Puppeteer; user-agent rotation; use of proxy servers. However, though the abovementioned methods work well, authors stress the fact that knowing the ethical and legal risks is crucial [6]. A multitude of user-generated applications make use of extracted data from manga and anime to aggregate information about favorite series or to generate personalized recommendation systems. These programs gather information like user ratings, episode guides, and release dates by web scraping data from manga and anime databases. Online stores selling items associated with manga have applied web scraping. Tanaka et al. (2021) has produced a study which "explores how web scraping can be applied to monitor and compare prices of niche manga retailers and marketplaces for manga merchandise that exist on Amazon and eBay". This will enable the fans to find the cheap DVDs, toys, and manga volumes. [7]. Use of web scraping in the manga and anime industry will present many benefits, such as tracking product prices, combing fan reviews, and enabling web scraping for new releases. Though applied, general techniques and available tools for scraping are very well noted in the scientific literature. The specific application in manga and anime thus still emerges mainly through contributions from the community. However, ethical issues are still of paramount importance, especially on compliance with the prevailing legal regulations and respect for intellectual property rights. The present literature review, coupled with community-driven initiatives, may serve as a foundational basis for further investigations into scraping automation

within this rapidly evolving entertainment sector.

3. Proposed Methodology

This technique intends to explain the process for designing and implementing an automated web scraping system especially for aggregating and reconciling content from platforms such as those that post comics and anime. One must collect the structure as well as combined data to develop a centralized database for analysis, tracking, or developing applications such as dashboards or recommendation systems. The goal remains collecting structured data like release dates, title, reviews, character information, and price.

3.1. Scrape and Database Check:

This is the python code of scraping and data updation from the Jikan API, offering information about some animated shows, manga, cartoons, or people. Explaining all parts of the script here:

1. Global Variables and Setup (<https://api.jikan.moe/v4>). wait: Time interval in between API requests to not hit the rate limits. data_dir: The directory where scraped data will be kept. The script checks whether the directory (data_dir) exists and creates it if it does not exist
2. Function: scrape_page(endpoint, page) This function fetches data from the Jikan API for a specific endpoint (such as 'anime', 'manga', 'people', or 'characters') and page. It returns the data part of the response, which includes the requested information
3. Function: scrape_jikan_db(database_list, db_path) This function scrapes data from one or more Jikan API endpoints and stores merged results in file (db_path). Checking for existing data in the file db_path not to override something, that has already been loaded. For every endpoint in the database_list there is, for example, ['anime'], ['manga'], it calculates how many pages are inside by using Jikan API. Then, at page by page scraping of data from the corresponding pages with scrape_page, adds the freshly obtained data into the list of merged_db and saves to the file in JSON format.
4. Function: merge_files(database_list, db_path)



Load the JSON data from db_path and merge all entries into a list. The merged data is saved back to the same file.

5. Function: check_and_update(database_list, db_path) This function checks if the data for a given set of endpoints (database_list) is up to date. It compares the current number of pages in the API obtained using requests.get() with the number of entries in the already available file. If there are new entries or no available file at all, it invokes scrape_jikan_db() that updates or scrapes the data.
6. Function: run_operations() This is the main function coordinating all the scraping and updating operations of four types: anime, manga, people, and characters. It initializes paths to their corresponding json files (anime.json, manga.json, people.json, characters.json) and checks for updates. After checking and possibly updating each category, merge each category data by the merge_files.
7. Running the Script: The function run_operations() is now called at the end, which initiates the full scraping, updating, and merging of all the four databases.

3.2. Automation and Date Time

This Python code is creating an automated task, that runs the function, run_operations(), at the next occurrence of 12 AM IST (Indian Standard Time), and then runs it every 24 hours. Here is a step by step of what the code does:

1. Imports and Setup threading: This module will be used to create a timer that will schedule a task to run at specified intervals. time: Used for measuring time or intervals between tasks. datetime and timedelta: Used to calculate the current time and to determine the time until the next midnight. pytz: A library that enables dealing with timezones. It is used here to fetch the current time in IST, or shortcuts for Asia/Kolkata timezone.
2. Function: time_until_next_midnight_ist()
3. This function determines how many seconds are left until next 12 AM/12 noon in IST, or Asia/Kolkata time zone. It fetches the current time in IST with the help of pytz. Next mid-

night is determined by adding one day to the date today and the clock to exactly 12:00 AM (midnight). The function returns a time difference in seconds between the current time and the next midnight.

4. Function: automation_run(interval) This function calls the run_operations() function and schedules the next run at the specified interval (in seconds). It prints a message reporting that it is running the automated task. The run_operations() function (from the previous code) is called to scrape, update and merge the anime/manga/People/characters data. After run_operations() is executed, the threading.Timer will call for a new execution of automation_run() after interval seconds have passed by, with interval set at 86400 seconds or as 24 hours.
5. The Core Logic It uses time_until_next_midnight_ist() to calculate time until next midnight. It prints the time left before the next 12 AM IST, converting from seconds to hours to make it easier to read. A threading.Timer is called, that will wait until midnight again 12 AM IST when the script will call automation_run(), passing 86400 seconds (24 hours) so that, after the first run at midnight itself, the task will be run every 24 hours; The script sees to it that the run_operations() function is automatically called at 12 AM IST one day, then at 12 AM IST the following day, and so on at the same time every 24 hours. It makes use of threading to run the scheduling in the background.

3.3. CMD Explanation

The command cd path_your_file is a terminal or command-line instruction where: cd stands for "change directory," which is utilized in order to change to another directory within your file system. path_your_file is a placeholder for the actual path to the directory where your file is located.

3.4. Server Hosting

The command python -m http.server 8000 is employed to start a simple HTTP server with Python. Here is a breakdown of what each part of the command does:



1. python: This calls up the Python interpreter. Make sure Python is installed on your system.
2. -m http.server: This starts the built-in HTTP server module (http.server that comes with Python. It serves files and directories from the current working directory over HTTP. Since Python 3.x.
3. 8000: It is specifying that port number on which server will listen for incoming request. Here, the server is going to run on port 8000. You can choose other ports (like 8080, 3000, etc.) as long as they are not in use by other services.
4. Connecting and fetching of Json: This JavaScript code fetches a list of anime from local file anime.json on the page. It has broken data up into pages and allowed for navigation through the list of anime based on these pagination divisions.

Here's how the Algorithm breaks down:

1. Global Variables animePerPage = 20: The number of anime entries on the page will be 20. currentPage = 1: The current page will be 1, which means the first page will load when you open it.
2. Function: fetchAndDisplayAnime(page = 1) It fetches the data from anime.json and displays it on the page: fetch('anime.json') does a GET request to fetch the file containing the list of anime anime.json. .then(response => response.json()) processes the response as JSON.

3.5. Clearing the Container

When the data is fetched, this function clears the HTML container (where anime cards are displayed) using `container.innerHTML = ''`. This ensures that only data for the current page are returned. Logic for pagination: start and end indices It calculates `startIndex` and `endIndex` by using the page parameter to determine which anime to render on the current page. For example, if on page `startIndex = (1 - 1) * 20 = 0` `endIndex = min(0 + 20, total_anime_count)` That is, anime from index 0 to 19 (first On page 1, see examples 20). For page 2, anime from index 20 to 39 will be shown, and so on. Generating the anime cards: For each anime in the current page range (from `startIndex` to `endIndex`), a card is dynamically

created with information such as:
`anime.images.jpg.image_url` : Anime thumbnail.
`anime.title` : Anime title. `anime.episodes` : Number of episodes. `anime.score` : User rating score.
`anime.synopsis` : A short description of the anime (first 150 characters). It also refers to a second page (`anime\details.html`) with more details about the anime by passing the `mal_id` (unique identifier of the card) as a URL parameter. Function: `renderPagination(totalItems, currentPage)` This function is used for calculating the pagination buttons in relation to the total number of anime items: To determine the total number of pages: `totalPages = Math.ceil(totalItems / animePerPage)` Calculates how many pages are needed for all entries in order to represent all anime, rounding up if there are more entries than the number of anime presented at one time on the page. To render the pagination buttons: For every page, a button is created and appended to pagination container. If the button is corresponding to the current page, it gets an active class to style them. Each button gets a click event listener, so that when this button is clicked, it calls `fetchAndDisplayAnime()` with the corresponding page number, updating the displayed anime. Page Fetch on page load: `fetchAndDisplayAnime(currentPage)`; The above ensures that when the anime data is loaded for the first time, default loaded when opening the page It first will render the first page, that is, `currentPage = 1`. The code dynamically fetches and displays anime from a local JSON file called anime.json. These will be responsible for making cards accompanied by the entire information of the anime. Pagination controls allow users to browse the different pages. Each card has a link to another page that would give information regarding the anime, passing the unique id of the anime as a parameter. Algorithm: The global configuration for the web scraping process begins with setting up the base API URL, which in this case is 'https://api.jikan.moe/v4'. To prevent rate limiting issues during scraping, a wait time of 1.2 seconds is implemented between requests. Additionally, a directory named data is designated for storing the scraped data. The next step involves verifying the existence of this data directory; if it does not already

exist, it is created. Scrape_page(endpoint, page), the primary scraping function, sends a GET request to the specified API endpoint for a given page and parses the response to return the contents in JSON format. Scrape_jikan_db(db_path, database_list) is used to manage the scraping of various datasets, such as anime, manga, etc. If the data is already available at the specified db_path, it is loaded directly; for each database type in the list, the total number of pages is calculated using the last_visible_page from the API; the scrape_page function is then called iteratively to retrieve data from each page, combining it with any previously available data before saving the updated content. The merge_files(database_list, db_path) function combines the scraped data into a single combined file by determining whether the data is present in the specified path. The check_and_update(database_list, db_path) function is in charge of detecting updates; it compares the most recent page count for each dataset with the existing data, updates the database using scrape_jikan_db if there are new entries, skips the update, or starts a new scraping process if no data is found in db_path. The main function, run_operations(), prepares filenames for various datasets, such as anime.json, manga.json, persons.json, and characters.json, and iterates through each database type, calling check_and_update to retrieve and update the data appropriately. The automation_run(interval) function calls run_operations() for scraping, printing a message at the specified interval; using threading.Timer, the first automated run is scheduled to start at the next 12 AM IST, followed by subsequent runs every 24 hours (86400 seconds). Modules like threading, time, datetime, and pytz are imported to handle time-based scheduling. The function time_till_next_midnight_ist() determines the amount of time until the next 12 AM IST using the Asia/Kolkata time zone, Shown in figure 1, figure 2, figure 3, figure 4, figure 5.

the system can adeptly manage variations in the API data without unnecessary scraping, ultimately enhancing performance. The utilization of functions such as 'merge_files' and 'check_and_update' plays a crucial role in overseeing the dataset's integrity, averting duplication, and guaranteeing that only fresh or modified entries are incorporated. Furthermore, incorporating pauses between requests minimizes the chances of facing rate limitations from the API, hence guaranteeing a seamless and reliable data gathering process. In general, this methodical approach enables efficient data scraping, establishing a dependable tool for collecting thorough anime and manga details from the Jikan API, Shown in figure 6, figure 7 & figure 8.

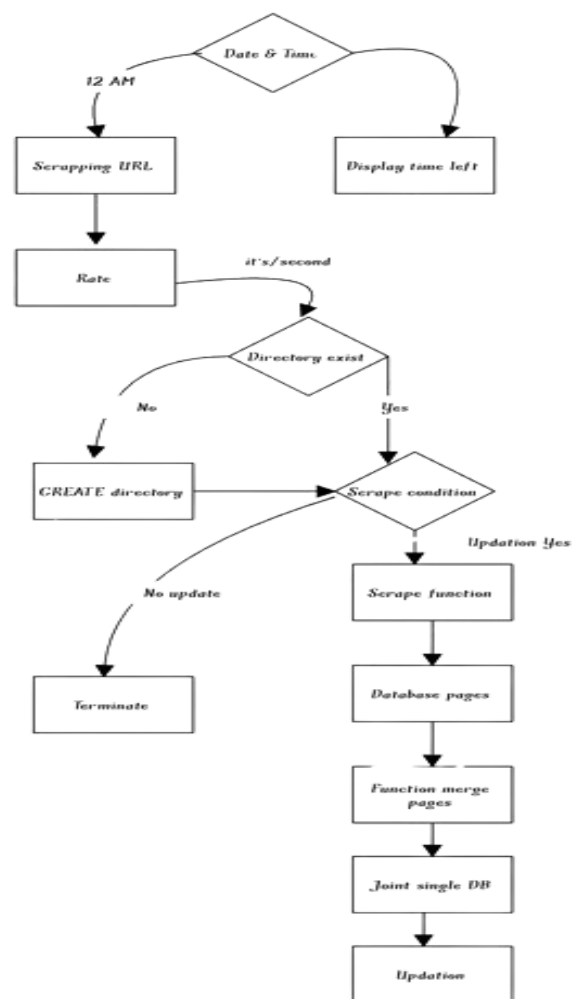


Figure 1 Process of Dataflow

4. Result

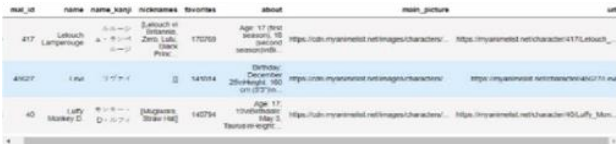


Figure 2 Characters.json Details

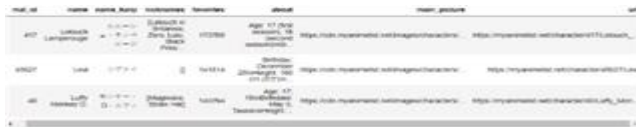


FIGURE 3 Process of Dataflow

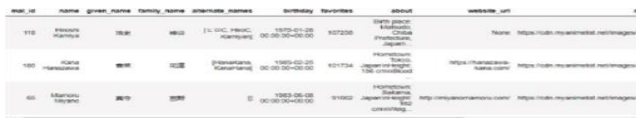


Figure 4 People.json Analysis



Figure 5 Manga.json Details (1)



Figure 6 Manga.json Details (2)

```

Successfully scraped page 1
Successfully scraped page 2
Successfully scraped page 3
Successfully scraped page 4
Successfully scraped page 5
Successfully scraped page 6
Successfully scraped page 7
Successfully scraped page 8
Successfully scraped page 9
Successfully scraped page 10
Successfully scraped page 11
Successfully scraped page 12
Successfully scraped page 13
Successfully scraped page 14
Successfully scraped page 15
Successfully scraped page 16
Successfully scraped page 17
Successfully scraped page 18
Successfully scraped page 19
Successfully scraped page 20
Successfully scraped page 21
Successfully scraped page 22
Successfully scraped page 23
Successfully scraped page 24
Successfully scraped page 25
Successfully scraped page 26
Successfully scraped page 27
Successfully scraped page 28
Failed to retrieve page 29. Status code: 404
Data successfully written to all_anime_data.json
  
```

Figure 7 Process of Scrapping

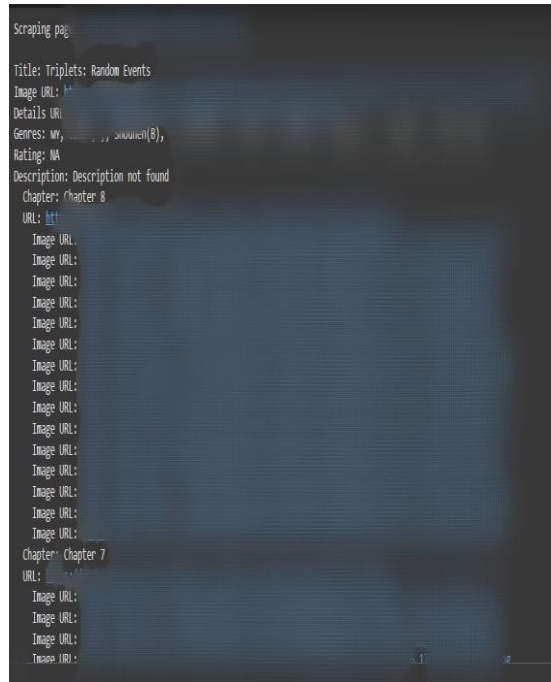


Figure 8 Scrapped Data Aggregation

5. Conclusion

In short, automatic web scraping integrates powerful avenues of collecting large volumes of information that may be present on various online sources and making it centralized. In this sense, it denies the possibility of overnight changes in manga and anime platforms since the data will be automatically generated and updated on each new instance of addition, say, anime episodes, manga chapters, character profiles, and information about artists. Thus, it minimizes the effort put into gathering data effort and goes ahead to make efficiency and accuracy in handling data. This system takes all the timely and comprehensive user rating, series episode dates, character details, or plot synopses and centralizes them, making it not just easier to aggregate content but also to provide an overall superior user experience for content enthusiasts, researchers, and curators. A centralized information repository ensures that up-to-date information is available to all users as they easily browse to discover new content, analyze trends, and track releases. This automation, therefore, provides a scalable framework for the management of massive volumes while offering support for informed content exploration and decision-making.



References

- [1]. Mitra, S., & Acharya, T. (2019). Web Scraping and Its Tools: A Survey. *International Journal of Computer Applications*, 178(32), 1-7.
- [2]. Simpson, M., Gupta, R., & Shenoy, A. (2020). Automated Data Collection and Processing Using Web Scraping for Market Intelligence. *Journal of Data Science and Artificial Intelligence*.
- [3]. Ghosh, A., Das, R., & Bhattacharyya, R. (2018). A Study on Web Scraping Techniques to Collect Data for Entertainment Industry. *International Journal of Data Science*.
- [4]. Huang, Z., Chen, T., & Patel, A. (2020). Scraping Streaming Platforms for Metadata Aggregation and Recommendation Systems. *ACM Transactions on Multimedia Computing, Communications, and Applications*.
- [5]. Madhavan, J., Halevy, A., & Ko, D. (2017). Ethical Considerations in Web Scraping for Digital Content. *Data & Knowledge Engineering Journal*.
- [6]. Liu, F., Zhang, C., & Wang, L. (2019). Overcoming Anti-Scraping Measures in Dynamic Web Environments. *International Journal of Web Engineering*.
- [7]. Tanaka, M., & Yoshida, K. (2021). Web Scraping for Manga Merchandising Price Comparison. *Journal of E-Commerce and Digital Marketing*.