Automating Web Data Collection: Challenges, Solutions, and Python-Based Strategies for Effective Web Scraping

Mutaz Abdel Wahed
Faculty of Information Technology
Jadara University
Irbid, Jordan
mutaz@jadara.edu.jo

Mowafaq Salem Alzboon
Faculty of Information Technology
Jadara University
Irbid, Jordan
malzboon@jadara.edu.jo

Muhyeeddin Alqaraleh
Faculty of Information Technology
Zarqa University,
Alzarqa, Jordan
malqaraleh@zu.edu.jo

Jaradat Ayman
Faculty of Information Technology
Jadara University
Irbid, Jordan
ay.jaradat@jadara.edu.jo

Mohammad Al-Batah
Faculty of Information Technology
Jadara University
Irbid, Jordan
albatah@jadara.edu.jo

Ahmad Fuad Bader
Faculty of Engineering
Jadara University
Irbid, Jordan
abader@jadara.edu.jo

2024 7th International Conference on Internet Applications, Protocols, and Services (NETAPPS) | 979-8-3315-1951-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/NETAPPS63333.2024.10823528

***Abstract*— The process of collecting and retrieving such a massive amount of data is difficult, especially when manual approach is the only option. Instead, we can use web scraping to automate the process of collecting web data using bots or automated scripts known as web scrapers. Web scraping as data mining technology is gaining popularity among data scientists and analysts, it is also causing controversy due to ethical concerns about the ability of get data due to privacy and copyright using automated programs or tools. This paper aims to classify and estimate the various challenges and solutions associated with web scraping algorithms, specifically those utilizing python libraries, and addressing the ethical implication associated with this technology.**

Keywords— Web Scraping, data mining, big data, Bots

I. INTRODUCTION

Information has been the most important resource for corporate development during the last decade, with the internet serving as the primary source, billions of internet users contribute new data every second [1]. Companies establish their business plans and attain their goals by obtaining and analyzing web data. The process of collecting and retrieving such a massive amount of data is difficult, especially when manual approach is the only option. Companies and individuals can use web scraping to automate the process of collecting web data using bots or automated scripts known as web scrapers, and then download the data in database, Excel, CSV, or XML format for further analysis and processing [2].

Web scraping can be divided into two categories [3]. The first is screen scraping, which involves using an HTML parser or regular expression matching to extract data from a website's source code. The second method is to employ application programming interfaces (APIs). When a website provides a set of structured HTTP requests that return CSV, Excel, JSON, or XML files, this is known as structured HTTP requests [4]. In this article, I focus on the challenges due to code and execute the Python web scraping script, however, before building the script, it is recommended to check if a website offers an API to access their data [5]. While Web scraping is gaining popularity among data scientists and analysts, it is also causing controversy due to ethical concerns about the ability of get data due to privacy and copyright using automated programs or tools [6].

Contact scraping, pricing change monitoring or evaluation, product assessment collection, actual property list collecting, climate facts statement, website exchange detection, and internet information integration are only few of the scenarios that web scraping may be utilized for [7]. At a microscale, for instance, the rate of a inventory may be scraped on an ordinary foundation to visualize rate trade over time [8], and social media feeds may be scraped in bulk to study public opinion and identify opinion leaders [9].

II. RELATED WORKS

With the growing demand for big data, web scraping has become a prominent issue among researchers and individuals. Most companies increasingly demand for data from multiple websites to use in the development of their business. When online scraping procedures scale, many challenges issues may arise, such as legality, web site structure, dynamic contents, blocking mechanisms, which can prevent scrapers from receiving data. Obtaining unstructured data from a variety of online sources, then structuring and turning the data into a structured format that can be saved and analyzed in a database [10].

Table 1, illustrate a number of challenges that researchers identify when attempting to automate the collecting of internet data.

Most researchers consider some of the difficulties and challenges in online scraping, but often overlook additional challenges that could have resulted in different outcomes. In highlighted the issues of web scraping in light of the website's interface being updated on a regular basis to boost user activities.

TABLE 1. OVERVIEW OF THE MAJOR WEB SCRAPING CHALLENGES

Challenge	Description
Ethical and Legal Risks	Check if the website permits Web scraping
HTML structure	Websites periodically update the content
IP blocking	Stop web scrapers from accessing data of a website
Access barriers (Login, Captcha)	Requirement of a login or a puzzle to gain access.
Big Data	Gathering and use of big amounts of data
Real-time data scraping	Scraper needs to monitor the websites all the time

Because the data is not static, scraping process will output different results if the source data on the website has been modified, even if using the same code to scrape the data at a later time. In [11] the authors identify challenges that are typically faced when scraping the Web such as, HTML structure of the web site, and access barriers for web scraping is deemed criminal. It's also possible that your IP will be permanently blacklisted by a website. In [12] the author highlights challenges like ethical, registration requirement, login from different IPs, big data, and real time data.

Aside from the technical difficulties, online scraping also poses ethical and legal issues. Websites' terms of service frequently explicitly forbid automatic scanning of their material (e.g., Facebook), but even if they don't, it's often questionable whether specific content is ethically allowed to scrape [13]. Web scraping is a powerful and valuable technology that comes with its own set of problems and challenges. This study tries to identify the difficulties that have been experienced in earlier studies which summarized in table 2, and Python is used to solve difficulties.

TABLE 2. WEB SCRAPING CHALLENGES ENCOUNTERED IN PREVIOUS STUDIES

work	Challenges					
	Ethical and legal Risks	**HTML Structure**	**IP blocking**	**Login and captcha**	**Big Data**	**Real-time data**
[2]	√	√	√	X	√	X
[4]	√	√	√	√	X	X
[5]	√	X	X	X	√	X
[6]	√	√	√	√	√	√
[8]	X	√	√	√	√	X

Web scraping software utilities is built by programming companies to simplify the process of collecting data from websites.

While web scraping tools like Scraper API, Octoparse, DataOx, Scraping-Bot.io, Webhose.io, and Diffbot offer powerful features such as CAPTCHA bypass, automatic IP rotation, and real-time data retrieval, they also come with certain disadvantages. One of the main concerns is the potential for legal issues, as scraping data from websites without permission may violate terms of service or intellectual property rights, leading to legal disputes. Additionally, these tools often require a significant amount of configuration and technical expertise to optimize for specific use cases, which can be a barrier for non-technical users.

Moreover, despite features like IP rotation, there's always a risk of being blocked or flagged by target websites, especially if scraping is done excessively or the website employs sophisticated anti-bot measures. Lastly, the reliance on third-party APIs or services for scraping also introduces vulnerabilities related to data privacy, security, and dependency on external providers, which may impact the reliability and consistency of the data collection process.

Table 3 contain overview of the major Popular Web Scraping Tools.

TABLE 3. OVERVIEW OF THE MAJOR POPULAR WEB SCRAPING TOOLS

Tool	Description
Scraper API	Scraper API can easily work with browsers and proxy servers and bypass CAPTCHA verification codes. It's almost impossible to get blocked with this tool, as it changes IP addresses on every request, automatically retries failed attempts, and solves captchas.
Octoparse	Extracts data from the Internet and turn web pages into structured data in just one click. With automatic IP rotation to prevent blocking and the ability to schedule subsequent scraping.
DataOx	The tool enables large-scale data collection and provide comprehensive solutions fit to customer needs.
Scraping-Bot.io	Offers APIs for data collection in retail (product description, price, currency, recall) and real estate (purchase or rental price, area, location).
Webhose.io	An advanced API service for real-time web data retrieval. This tool is very commonly used for historical data extraction, media monitoring, business intelligence, financial analysis, and academic research.
Diffbot	Automates the extraction of web data using artificial intelligence and makes it easy to get various data from any site.

The fact that these web data extraction solutions may not always work as planned is a significant disadvantage. They regularly miss certain online pages or are just unable of determining how to obtain the information they seek from a website. Many of these tools have subscription costs, forcing many industry researchers to create their own proprietary Web scraping tools using powerful programming languages like Python.

Web scraping using Python libraries has become an essential technique for data extraction, but it often comes with challenges such as website blocking, dynamic content, CAPTCHA protection, and legal issues. However, several solutions can be implemented within Python to overcome these challenges.

III. METHODOLOGY

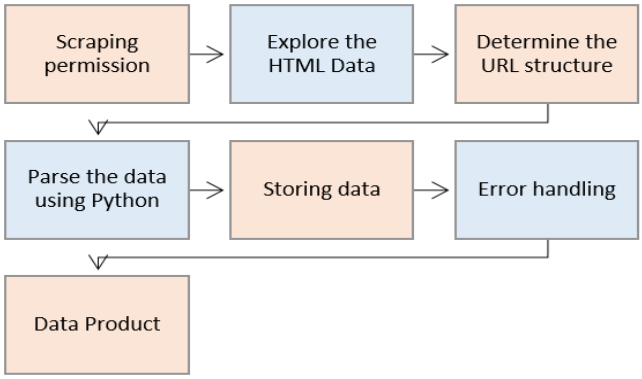
An alternative method of getting site data is through API calls. Interaction with the API is a method officially provided by the site owner to get data directly from the database or regular files. This usually requires permission from the site owner and a special token. However, the API is not always available, which is why scraping is so attractive, but its legality is questionable [14].

Creating a solution for mentioned challenges using Python for pulling data out of HTML and XML files, scraping can violate copyright or site usage policies, especially if

requested data is large. Scraping, on the other hand, is freely available and can be used for personal, academic, or other non-commercial purposes [15]. Python is usually recognized as the best programming language due to its ease of use. Python is also well suited to web scraping for the same reason. The Python web scraping libraries and tools like (**Request, LXML, BeautifulSoup, Scrapy, and Selenium**) are recommended, the proposed process of web scraping includes seven stages, Figure 1 describes the stages required to create a web scraper [16]–[18].

This paper aims to provide detailed instructions on how to code a web scraper and to propose solutions to the challenges highlighted in previous studies by providing an intuitive description of what a web scraper does technically and how this tool can be implemented to generate any data sets for further use [19], [20].

FIG. 1 STAGES OF WEB SCRAPING APPROACH



3.1 Basic system environment requirements

Operating System: Windows 10-11
Programming Language: Python version 3.12
SDK Environment: Jupyter on Anaconda 2024.1
Python modules: requests, csv, pandas, and beautiful Soup
Hardware: CPU Core i7, DDDR4 8GB, NVIDIA GPU
Data Storage: Local SSD 1TB
URL for the webpage: http://examplexx.xxx

3.2 Website Scraping Permission

Scraping on such a large scale has the potential to be criminal. A higher number of queries per second can cause damage to the infrastructure of the scraped website. It is permissible to scrape all publicly available data, but it is advised that web scraping be done in accordance with the web scraping rules. This can be done by looking at the robot.txt file [6], sitemap file, and the website's terms and conditions.

https://www.myexample.com/
https://www.myexample.com/robots.text
https://www.myexample.com/pages/robots.text

3.3 Ethical and legal Risks

Apart from legal constraints, ethical concerns should be discussed. Large websites, in particular, usually require user registration and can detect scrapers [21], [22]. Although this method is technically feasible, the question of whether it is ethically appropriate to use it in scientific, often publicly financed research remains unanswered. Obtaining the explicit authorization of the website operator is unquestionably the best approach [23], [24].

3.4 HTML Structure

Understanding the HTML structure of a website is crucial for successful internet scraping. The structure determines how statistics is prepared and provided at the web page, and this information permits scrapers to correctly target and extract the preferred data. Websites usually use tags like , , and <a> to arrange content material, and attributes along with class and identity to style or identify particular factors. Scrapers frequently depend upon those tags and attributes to locate the records they need. Tools like 'BeautifulSoup' and 'lxml' may be used in Python to navigate the HTML structure and extract information [23], [25].

3.5 IP Blocking

IP blocking off is a commonplace countermeasure used by websites to prevent unauthorized internet scraping. When a site detects an uncommon quantity of requests from a single IP deal with, it could block that IP to protect its resources. To avoid IP blocking, scrapers can implement techniques inclusive of rotating IP addresses, using proxy servers, and reducing the frequency of requests. Additionally, it is vital to adhere to the website's robots.Txt document, which outlines the permissible scraping conduct [26], [27].

3.6 Login and CAPTCHA

Websites often require customers to log in or remedy a CAPTCHA to get admission to sure content material, growing demanding situations for scrapers. CAPTCHAs are designed to differentiate human customers from bots and can be a widespread hurdle for computerized scraping. However, with the help of Python libraries like Selenium, it's miles feasible to automate login tactics and, in a few instances, bypass CAPTCHA challenges the usage of 1/3-celebration services or system learning techniques like schooling a Convolutional Neural Network (CNN) to resolve CAPTCHAs [28].

3.7 Big Data

Web scraping at scale frequently entails coping with big volumes of facts, which can be tough to control and examine. Big facts strategies and gear inclusive of Hadoop, Spark, and cloud-based answers may be employed to keep, manner, and analyze the scraped facts correctly. When scraping large facts, it's critical to don't forget factors like facts storage, retrieval instances, and processing power to make certain the system can manage the load [29].

3.8 Real-Time Data Scraping

Real-time facts pose precise demanding situations, specifically with regard to retaining statistics freshness and coping with common updates. To control actual-time scraping, scrapers can be set up to run continuously or at normal periods, pulling within the today's information because it becomes to be had. Python libraries including

Selenium can automate the actual-time scraping system, whilst equipment like Celery and RabbitMQ can be used to control challenge queues and make sure well timed information series [30].

Table 4 provides an overview of the common obstacles encountered during web scraping and the corresponding strategies to overcome them using Python-based tools and techniques [31].

TABLE 4: CHALLENGES AND SOLUTIONS IN WEB SCRAPING

Section	Description	Solutions/Tools
Website Scraping Permission	Scraping should be done in accordance with the website's terms and conditions and **robots.txt** file.	Review **robots.txt**, sitemap, and terms and conditions; adhere to rules set by the website.
Ethical and Legal Risks	Scraping can pose ethical and legal concerns, especially on sites requiring user registration. Explicit permission from website operators is recommended.	Seek permission from website operators, follow legal guidelines, and use scraping ethically.
HTML Structure	Understanding the HTML structure of a webpage is key to successfully targeting and extracting data.	Use **BeautifulSoup**, **lxml** to navigate and parse the HTML structure.
IP Blocking	Websites may block IPs that make too many requests.	Implement IP rotation, use proxy servers, reduce request frequency, and adhere to **robots.txt**.
Login and CAPTCHA	CAPTCHA and login requirements protect websites from bots, but can be challenging for scrapers.	Use **Selenium** for automation, machine learning models for CAPTCHA solving, and third-party CAPTCHA-solving services.
Big Data	Managing and analyzing large-scale scraped data requires big data techniques.	Use tools like Hadoop, Spark, and cloud storage solutions for handling big data.
Real-Time Data	Scraping real-time data requires maintaining data freshness and handling frequent updates.	Implement continuous scraping with **Selenium**, manage task queues with **Celery**, **RabbitMQ**, and ensure timely data collection.
Parse the Data using Python	After receiving an HTML document, parsing focuses on extracting only the needed data.	Use Python libraries such as **BeautifulSoup**, **Scrapy**, and **lxml** for parsing.
Storing Data	Extracted data needs to be stored efficiently for future use or analysis.	Store data in local files (CSV, JSON), SQL databases (SQLite, MySQL), or cloud storage solutions (AWS S3, Google Cloud Storage) depending on the project's scale.

3.9 Parse the Data

The usage of Python Parsing is the technique of studying the HTML content material of a web site and extracting applicable statistics. Python gives several powerful libraries for parsing web records, which include BeautifulSoup, Scrapy, and lxml. After sending a request to a web server and receiving the HTML response, those libraries help to navigate via the report and extract precise elements based on tags, training, or IDs. This parsed information can then be saved, analyzed, or utilized in numerous packages.

3.10 Storing Data

After parsing, the extracted statistics have to be stored efficiently for future evaluation or use. Depending on the character of the records and the scale of the scraping mission, numerous storage solutions may be employed. For small to medium-sized projects, records may be saved in nearby files (e.g., CSV, JSON) or SQL databases like SQLite or MySQL. For large-scale operations, cloud storage solutions along with AWS S3, Google Cloud Storage, or distributed databases like Cassandra and MongoDB are more appropriate. These storage alternatives permit for scalability, ease of access, and the capacity to perform complex queries at the statistics [3].

IV. RESULTS

The proposed web scraping technique using Python libraries such as BeautifulSoup, Scrapy, and Selenium become examined throughout numerous web sites with one-of-a-kind tiers of safety, which includes CAPTCHA, login necessities, and anti-scraping mechanisms like IP blockading. The experiments tested the performance of these equipment in overcoming not unusual internet scraping challenges.

Website Scraping Permission: The use of robots.Txt and adherence to the internet site's phrases and conditions ensured that the scraping was performed within criminal and ethical obstacles. Websites with honestly defined robots.Txt files allowed smoother and legally compliant scraping.

Ethical and Legal Risks: By acquiring permission from website operators, the scraping system was ethically performed, minimizing the hazard of prison repercussions.

HTML Structure: The parsing segment the use of BeautifulSoup and lxml efficiently extracted the focused facts, confirming the effectiveness of those libraries in navigating complicated HTML systems.

IP Blocking: Implementing IP rotation and decreasing request frequency notably reduced the instances of IP blockading, making sure the continuous operation of the scraper.

Login and CAPTCHA: The use of Selenium, blended with CAPTCHA-solving techniques, allowed for a success facts extraction even from websites with state-of-the-art bot protection measures.

V. DISCUSSION

The effects imply that at the same time as Python-based internet scraping can be notably effective, it requires cautious planning and the usage of superior techniques to conquer various demanding situations. The significance of respecting

website permissions and criminal barriers cannot be overstated, as failure to accomplish that can result in extreme moral and criminal issues. Additionally, the complexity of HTML structures and the presence of anti-scraping mechanisms consisting of IP blockading and CAPTCHA gift widespread challenges. However, with the right gear and strategies, those barriers can be efficaciously managed.

The use of Python libraries like BeautifulSoup and Scrapy for parsing and Selenium for automation proved to be powerful answers in the context of net scraping. IP rotation and CAPTCHA-fixing techniques had been important in managing web sites that rent anti-bot strategies. Moreover, ethical concerns performed a critical function in ensuring that the scraping technique turned into performed responsibly.

VI. Conclusion

In conclusion, web scraping using Python is a robust method for collecting and analyzing large amounts of data from the web. By employing appropriate libraries and tools, it is possible to overcome common challenges such as IP blocking, CAPTCHA, and complex HTML structures. However, it is essential to perform web scraping within the legal and ethical guidelines set by the target websites. Future work could focus on enhancing the efficiency and speed of these scraping tools and developing more advanced methods for overcoming sophisticated anti-scraping technologies.

References

[1] S. Kapoor, A. Sharma, A. Verma, V. Dhull, and C. Goyal, "A comparative study on deep learning and machine learning models for human action recognition in aerial videos.," Int. Arab J. Inf. Technol., vol. 20, no. 4, pp. 567–574, 2023.

[2] B. Zhao, "Web Scraping," in Encyclopedia of Big Data, 2017, pp. 1–3.

[3] S. Kapoor, V. Dhull, A. Sharma, C. Goyal, and A. Verma, "A Comparative Study on Deep Learning and Machine Learning Models for Human Action Recognition in Aerial Videos," Int. Arab J. Inf. Technol., vol. 20, no. 4, pp. 567–574, 2023, doi: 10.34028/iajit/20/4/2.

[4] S. Sharma, R. K. Challa, and R. Kumar, "An ensemble-based supervised machine learning framework for android ransomware detection," Int. Arab J. Inf. Technol., vol. 18, no. 3 Special Issue, pp. 422–429, 2021, doi: 10.34028/IAJIT/18/3A/5.

[5] M. Dogucu and M. Çetinkaya-Rundel, "Web Scraping in the Statistics and Data Science Curriculum: Challenges and Opportunities," J. Stat. Data Sci. Educ., vol. 29, no. S1, pp. S112–S122, 2021, doi: 10.1080/10691898.2020.1787116.

[6] V. Krotov, L. Johnson, and L. Silva, "Tutorial: Legality and ethics of web scraping," Commun. Assoc. Inf. Syst., vol. 47, no. 1, pp. 539–563, 2020, doi: 10.17705/1CAIS.04724.

[7] E. Vargiu and M. Urru, "Exploiting web scraping in a collaborative filtering- based approach to web advertising," Artif. Intell. Res., vol. 2, no. 1, pp. 44–54, 2012, doi: 10.5430/air.v2n1p44.

[8] E. L. Nylen and P. Wallisch, "Web Scraping," in Neural Data Science, 2017, pp. 277–288.

[9] S. J. Mooney, D. J. Westreich, and A. M. El-Sayed, "Epidemiology in the era of big data," Epidemiology, vol. 26, no. 3. pp. 390–394, 2015, doi: 10.1097/eDe.0000000000000274.

[10] S. de S. Sirisuriya, "A Comparative Study on Web Scraping," in Proceedings of 8th International Research Conference, KDU, 2015, pp. 135–140.

[11] P. Meschenmoser, N. Meuschke, M. Hotz, and B. Gipp, "Scraping scientific web repositories: Challenges and solutions for automated content extraction," D-Lib Mag., vol. 22, no. 9–10, 2016, doi: 10.1045/september2016-meschenmoser.

[12] J. Hillen, "Web scraping for food price research," Br. Food J., vol. 121, no. 12, pp. 3350–3361, 2019, doi: 10.1108/BFJ-02-2019-0081.

[13] F. Speckmann, "Web Scraping: A Useful Tool to Broaden and Extend Psychological Research," Zeitschrift fur Psychol. / J. Psychol., vol. 229, no. 4, pp. 241–244, 2021, doi: 10.1027/2151-2604/a000470.

[14] M. Alqaraleh, M. S. Alzboon, M. S. Al-Batah, M. A. Wahed, A. Abuashour, and F. H. Alsmadi, "Harnessing Machine Learning for Quantifying Vesicoureteral Reflux: A Promising Approach for Objective Assessment.," Int. J. Online \& Biomed. Eng., vol. 20, no. 11, 2024.

[15] M. Allam and N. Malaiyappan, "Hybrid Feature Selection based on BTLBO and RNCA to Diagnose the Breast Cancer," Int. Arab J. Inf. Technol., vol. 20, no. 5, pp. 727–737, 2023, doi: 10.34028/iajit/20/5/5.

[16] M. S. Alzboon, M. Al-Batah, M. Alqaraleh, A. Abuashour, and A. F. Bader, "A Comparative Study of Machine Learning Techniques for Early Prediction of Diabetes," in 2023 IEEE 10th International Conference on Communications and Networking, ComNet 2023 - Proceedings, 2023, pp. 1–12, doi: 10.1109/ComNet60156.2023.10366688.

[17] M. S. Alzboon and M. S. Al-Batah, "Prostate Cancer Detection and Analysis using Advanced Machine Learning," Int. J. Adv. Comput. Sci. Appl., vol. 14, no. 8, pp. 388–396, 2023, doi: 10.14569/IJACSA.2023.0140843.

[18] M. S. Alzboon, M. S. Al-Batah, M. Alqaraleh, A. Abuashour, and A. F. H. Bader, "Early Diagnosis of Diabetes: A Comparison of Machine Learning Methods," Int. J. online Biomed. Eng., vol. 19, no. 15, pp. 144–165, 2023, doi: 10.3991/ijoe.v19i15.42417.

[19] M. S. Alzboon, S. Qawasmeh, M. Alqaraleh, A. Abuashour, A. F. Bader, and M. Al-Batah, "Machine Learning Classification Algorithms for Accurate Breast Cancer Diagnosis," 2023, doi: 10.1109/eSmarTA59349.2023.10293415.

[20] M. S. Alzboon, S. Qawasmeh, M. Alqaraleh, A. Abuashour, A. F. Bader, and M. Al-Batah, "Pushing the Envelope: Investigating the Potential and Limitations of ChatGPT and Artificial Intelligence in Advancing Computer Science Research," 2023, doi: 10.1109/eSmarTA59349.2023.10293294.

[21] M. S. Alzboon, A. F. Bader, A. Abuashour, M. K. Alqaraleh, B. Zaqaibeh, and M. Al-Batah, "The Two Sides of AI in Cybersecurity: Opportunities and Challenges," 2023, doi: 10.1109/ICNGN59831.2023.10396670.

[22] M. S. Al-Batah, M. S. Alzboon, and R. Alazaidah, "Intelligent Heart Disease Prediction System with Applications in Jordanian Hospitals," Int. J. Adv. Comput. Sci. Appl., vol. 14, no. 9, pp. 508–517, 2023, doi: 10.14569/IJACSA.2023.0140954.

[23] M. Alzboon, "Semantic Text Analysis on Social Networks and Data Processing: Review and Future Directions," Inf. Sci. Lett., vol. 11, no. 5, pp. 1371–1384, 2022, doi: 10.18576/isl/110506.

[24] M. S. Alzboon, "Survey on Patient Health Monitoring System Based on Internet of Things," Inf. Sci. Lett., vol. 11, no. 4, pp. 1183–1190, 2022, doi: 10.18576/isl/110418.

[25] M. S. Alzboon, E. Aljarrah, M. Alqaraleh, and S. A. Alomari, "Nodexl Tool for Social Network Analysis," 2021.

[26] M. Al-Batah, B. Zaqaibeh, S. A. Alomari, and M. S. Alzboon, "Gene Microarray Cancer classification using correlation based feature selection algorithm and rules classifiers," Int. J. online Biomed. Eng., vol. 15, no. 8, pp. 62–73, 2019, doi: 10.3991/ijoe.v15i08.10617.

[27] M. Banikhalaf, S. A. Alomari, and M. S. Alzboon, "An advanced emergency warning message scheme based on vehicles speed and traffic densities," Int. J. Adv. Comput. Sci. Appl., vol. 10, no. 5, pp. 201–205, 2019, doi: 10.14569/ijacsa.2019.0100526.

[28] S. A. Alomari, M. S. Alzboon, B. Zaqaibeh, and M. S. Al-Batah, "An effective self-adaptive policy for optimal video quality over heterogeneous mobile devices and network discovery services," Appl. Math. Inf. Sci., vol. 13, no. 3, pp. 489–505, 2019, doi: 10.18576/AMIS/130322.

[29] M. S. Alzboon, "Internet of things between reality or a wishing - list : a survey," Int. J. Eng. \& Technol., vol. 7, no. June, pp. 956–961, 2019.

[30] I. Al-Oqily, M. Alzboon, H. Al-Shemery, and A. Alsarhan, "Towards autonomic overlay self-load balancing," in 2013 10th International Multi-Conference on Systems, Signals and Devices, SSD 2013, Mar. 2013, pp. 1–6, doi: 10.1109/SSD.2013.6564018.

[31] H. Alawneh and A. Hasasneh, “Survival Prediction of Children after Bone Marrow Transplant Using Machine Learning Algorithms,” Int. Arab J. Inf. Technol., vol. 21, no. 3, pp. 394–407, 2024, doi: 10.34028/iajit/21/3/4.