SHM PUBLISHER
SURYA HIJAU MANFAAT

Journal of Soft Computing Exploration

Homepage: https://shmpublisher.com/index.php/joscex

JOSCEX

p-ISSN: 2746-7686
e-ISSN: 2746-0991

.

Optimizing the Implementation of the BFS and DFS algorithms using the web crawler method on the kumparan site

Amirul Mustaqim[1], Dony Benaya Dinova[2], Muhammad Syafiq Fadhilah[3], Ravenia Arinka Seivany[4], Budi Prasetiyo[5], Much Aziz Muslim[6]
[1,2,3,4,5]Faculty of Mathematics and Natural Sciences, Semarang State University, Indonesia
[6]Faculty of Technology Management, Universiti Tun Hussein Onn Malaysia, Malaysia

Article Info

Article history:

Received March 30, 2024
Revised June 25, 2024
Accepted July 2, 2024

Keywords:

Depth first search (DFS)
Breadth first search (BFS)
Web crawling

ABSTRACT

Efficient access to timely information is critical in today's digital era. Web crawlers, automated programs that navigate the Internet, play an important role in collecting data from websites such as Kumparan, a leading news site in Indonesia. This research shows the effectiveness of the Breadth-First Search (BFS) and Depth-First Search (DFS) algorithms in indexing Kumparan content. The results of the research show that BFS consistently indexes more files comprehensively but with longer execution times compared to DFS, which provides faster initial results but with fewer files. For example, at depth 4 BFS indexed 949 files in 886.94 seconds, while DFS indexed 470 files in 233.02 seconds. These findings highlight the balance between precision and speed when selecting a crawling algorithm tailored to the needs of a particular website. This research provides insights into optimizing web crawler technology for complex websites such as Coil and suggests avenues for further research to improve permission efficiency and adaptability across a variety of crawling scenarios.

Corresponding Author:

Ravenia Arinka Seivany,
Faculty of Mathematics and Natural Sciences,
Universitas Negeri Semarang,
Kampus Sekaran Gunungpati, Semarang, 50229.
Email: raveniaarinka@students.unnes.ac.id

https://doi.org/10.52465/joscex.v5i2.309

1. INTRODUCTION

In today's information-rich digital era, fast and efficient access to news and actual information is very important for Internet users. One way to ensure this access is through the use of a web crawler. A web crawler is a computer program that automatically explores the Internet [1], playing a key role in the collection of information from various websites. Kumparan, as one of the leading news sites in Indonesia, provides a variety of the latest information to users [2]. However, with the complexity of its web page structure, which includes many cross-links between pages, a major challenge in efficiently indexing Kumparan sites arises. This is why choosing the right search algorithm in web crawler development is very important.

Web Crawler is a tool that functions as a search process and retrieves web pages from the Internet [3] to be indexed which will then be saved in a search database. When visiting a document, this web crawler performs three steps, namely detecting when a document has been visited, detecting links contained in the document, and indexing its contents [4]. However, due to hardware, bandwidth, and other network limitations, Web crawlers cannot download every page of search engine query results. Web crawlers must also avoid loading the Web server and comply with all server policies, including those specified in the robot.txt file [5]. Web Crawler is useful for a website so that it is easy for other people to visit. The indexing process is a very important process considering that it really helps users to quickly find relevant answer results related to searches. It is the same when indexing a book by looking for page numbers through the table of contents [6].

In the development of web crawlers, the efficiency and effectiveness of indexing web pages are highly dependent on the choice of search algorithm. Two search algorithms commonly used in this context are breadth-first search (BFS) and Depth-First Search (DFS). The Breadth First Search algorithm is a broadening search method that visits node n before moving to node n+1 [7]. This algorithm takes nodes from the beginning of the queue after inserting the end nodes into the queue. The search will end if the node is the desired search result. The search will enter all nodes that are near the node if it is not the desired search result. The search is considered complete when all nodes successfully complete the checking stage and the queue is empty. In this case, the search continues from the first node in the queue [8]. The Deep First Search algorithm is a deep search technique that starts at the first node and continues to the leftmost child node at the next level [9]. The performance of this algorithm is achieved by adding a root node to the stack. The search will then complete, and the results returned if the first node at the top level is the desired search result. All nodes next to the node will be added to the stack if the node is not the desired search result. The search is considered complete when all nodes successfully complete the checking stage and the queue is empty. If this happens, the search is restarted from the first node in the queue [8].

Previous research has investigated the use of BFS and DFS in web crawler development. Some studies use BFS for even exploration at the same level, while DFS tends to unfold vertically to a certain depth before returning [10]. Understanding the strengths and weaknesses of each of these algorithms is the key to choosing the right search strategy in developing a Web crawler for the Kumparan site.

Further complicating the development of web crawlers are issues such as handling dynamic content, managing duplicate content, and respecting the website's robots.txt policies. Dynamic content, often generated by JavaScript, presents a significant challenge because it requires the crawler to execute scripts to retrieve the content [11]. Handling duplicate content is another critical aspect because crawlers must identify and avoid indexing identical pages that appear under different URLs [12]. Additionally, respecting the robots.txt file ensures that the crawler does not overload the website and adheres to the site owner's restrictions on which parts of the site can be crawled [13].

Web crawlers also play a crucial role in various applications beyond search engines. They are used in data mining, web archiving, and web monitoring. In data mining, web crawlers help to collect vast amounts of data for analysis [14]. Web archiving involves preserving snapshots of websites over time, which is essential for historical research and legal purposes [15]. Monitoring website changes is critical for businesses that rely on up-to-date information about competitors, market trends, and customer feedback [16].

Therefore, this research aims to investigate the optimal application of the BFS and DFS algorithms in developing a web crawler for the Kumparan site. By understanding how these two algorithms behave in a specific context, it is hoped that the most efficient and effective search strategy can be found to collect information from the Kumparan site accurately and in a timely manner. This research will provide valuable insights for web crawler developers looking to improve the indexing performance of complex news sites like Kumparan, as well as providing a foundation for further research in this area.

2. METHOD

In this research, the type of research used is experimental research. Experimental research that optimizes the application of the BFS and DFS algorithms using the web crawler method on the Kumparan site aims to increase the efficiency of the indexing and browsing process for the website's content. The research methodology will involve testing and performance comparison stages between BFS and DFS algorithms that have been optimized in collecting and compiling data from the Kumparan website. The results of this research are expected to enhance web crawler technology, making it more efficient and adaptable for collecting information from news websites in diverse contexts.

a) Problem Identification

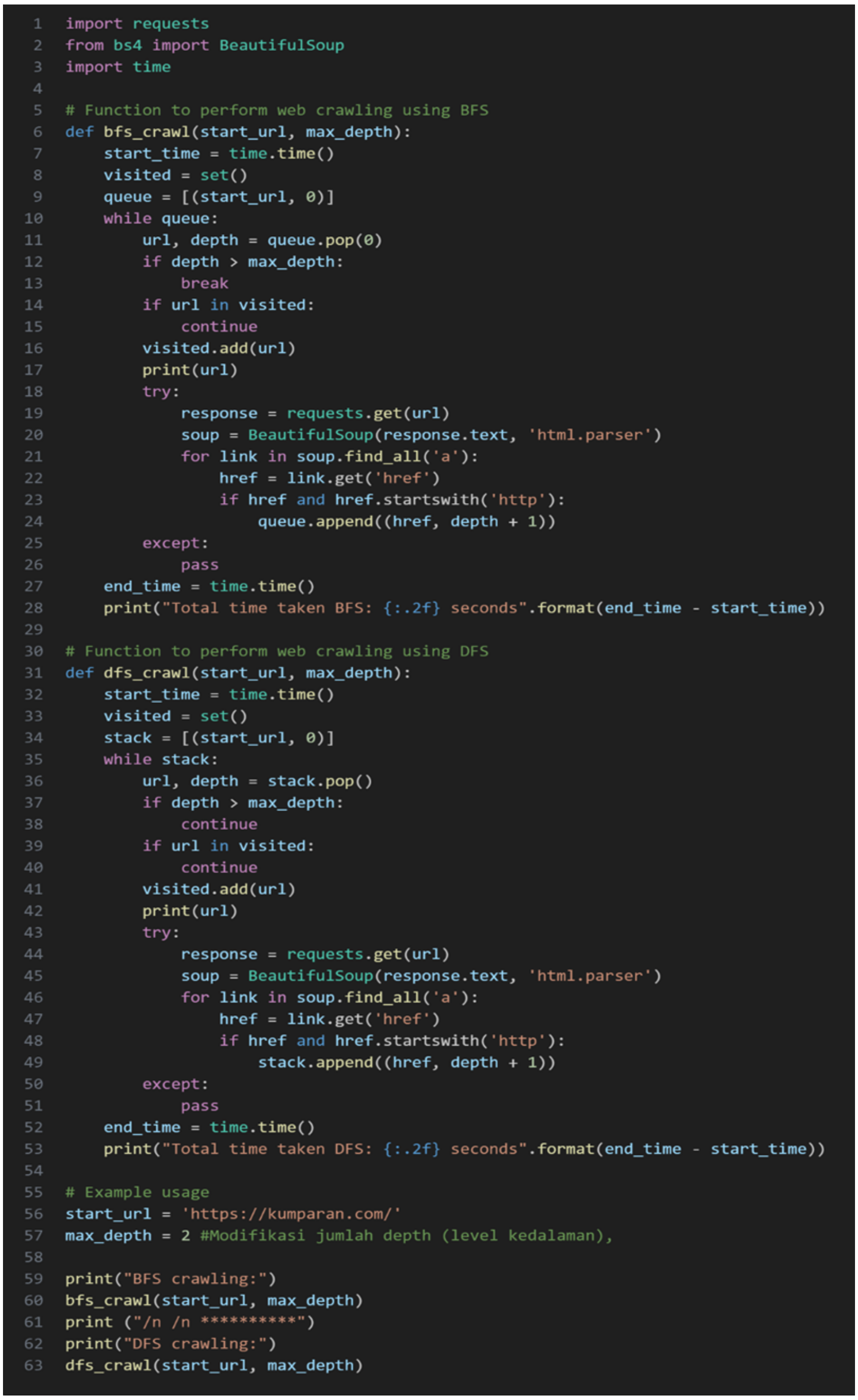
The initial stage in this research is to identify the problem to be solved, namely increasing the efficiency of indexing and searching the content of the Kumparan website by optimizing the BFS and DFS algorithms. Identification of this problem is carried out through an initial analysis of the existing search and indexing process, as well as observing the time required in the process.

b) Experiment Design

This research requires the selection of relevant parameters to measure the performance of the BFS and DFS algorithms. The parameters used are the different depths, the time required to complete the search, and the number of files indexed at each depth. Experimental design involves setting up various test scenarios to observe how changing these parameters affects the performance of the algorithm. Each algorithm will be run at various depth levels to see its effectiveness and efficiency in indexing content and completing searches. The data collected from this experiment will be analyzed to determine which algorithm is the most optimal on the Kumparan website.

c) Web Crawler Development

In this research, Python code is used to implement functions that perform web crawling using BFS and DFS algorithms. This web crawler will search and index the content of the Kumparan website based on its URL structure. The following is the Python code used to implement this algorithm.



```python
import requests
from bs4 import BeautifulSoup
import time

# Function to perform web crawling using BFS
def bfs_crawl(start_url, max_depth):
    start_time = time.time()
    visited = set()
    queue = [(start_url, 0)]
    while queue:
        url, depth = queue.pop(0)
        if depth > max_depth:
            break
        if url in visited:
            continue
        visited.add(url)
        print(url)
        try:
            response = requests.get(url)
            soup = BeautifulSoup(response.text, 'html.parser')
            for link in soup.find_all('a'):
                href = link.get('href')
                if href and href.startswith('http'):
                    queue.append((href, depth + 1))
        except:
            pass
    end_time = time.time()
    print("Total time taken BFS: {:.2f} seconds".format(end_time - start_time))

# Function to perform web crawling using DFS
def dfs_crawl(start_url, max_depth):
    start_time = time.time()
    visited = set()
    stack = [(start_url, 0)]
    while stack:
        url, depth = stack.pop()
        if depth > max_depth:
            continue
        if url in visited:
            continue
        visited.add(url)
        print(url)
        try:
            response = requests.get(url)
            soup = BeautifulSoup(response.text, 'html.parser')
            for link in soup.find_all('a'):
                href = link.get('href')
                if href and href.startswith('http'):
                    stack.append((href, depth + 1))
        except:
            pass
    end_time = time.time()
    print("Total time taken DFS: {:.2f} seconds".format(end_time - start_time))

# Example usage
start_url = 'https://kumparan.com/'
max_depth = 2 #Modifikasi jumlah depth (level kedalaman),

print("BFS crawling:")
bfs_crawl(start_url, max_depth)
print ("/n /n **********")
print("DFS crawling:")
dfs_crawl(start_url, max_depth)
```

Figure 1. Python web crawler code

Figure 1 shows a Python script to crawl the web using two different algorithms, namely, breadth-first search (BFS) and depth-first search (DFS). This script uses the ‘requests’ library to retrieve web pages and the ‘BeautifulSoup’ library from the ‘bs4’ library to parse HTML content. The functions ‘bfs_crawl’ and ‘dfs_crawl’ perform web crawling with BFS and DFS, respectively. Both functions accept ‘start_url’ as the starting URL and ‘max_depth’ as the maximum crawl depth. They start by initializing the time, sets for visited URLs, and data structures (queue for BFS, stack for DFS). Both processes the URL until the data structure is empty or the maximum depth is reached, fetches the page, and extracts links to add to the data structure if they are valid. The total crawl time is printed after the process is complete. In the example usage section, start_url is set to 'https://kumparan.com/' and max_depth is set to 2. Then, the functions 'bfs_crawl' and 'dfs_crawl' are executed to compare the performance and behavior of the BFS and DFS algorithms in context web crawling. The maximum depth parameter controls how deep the crawler goes into the web page hierarchy, limiting the breadth and depth of searches to prevent over-crawling.

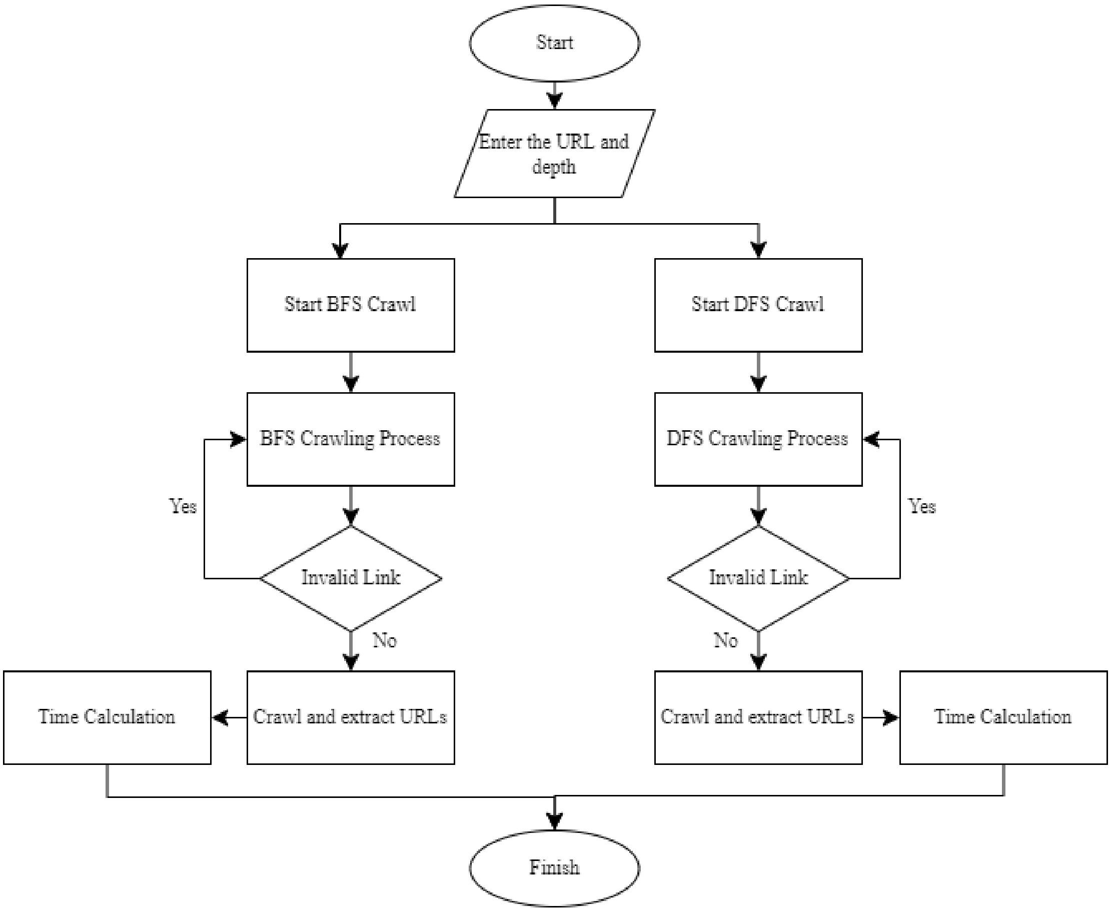


Figure 2. Web crawler process flowchart

d) Data Collection

The data collected will include the results of both the BFS and DFS algorithms in indexing and searching the content of the Kumparan website, available at https://kumparan.com/ [17]. The data obtained will be described in the results and discussion section.

e) Data Analysis

The collected data will then be analyzed to evaluate the performance of each algorithm, such as evaluating execution time, number of files indexed, and content browsing efficiency.

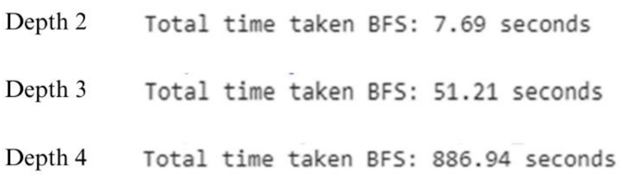
3. RESULTS AND DISCUSSIONS

After carrying out the crawling process, data were obtained from both the BFS and DFS algorithms in indexing and searching the content of the Kumparan website at different depths.

Calculating The Controllability System

Controlability is a test for determining how many states the system can be controlled with. The results of this stage show that control states can be determined.

.



Depth 2

```
Total time taken BFS: 7.69 seconds
```

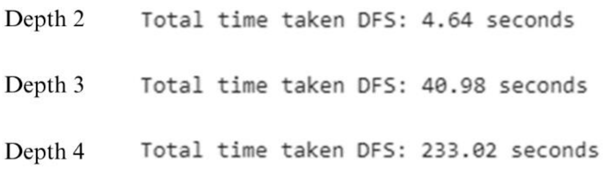
Depth 3

```
Total time taken BFS: 51.21 seconds
```

Depth 4

```
Total time taken BFS: 886.94 seconds
```

Figure 3. Crawling data in BFS



Depth 2

```
Total time taken DFS: 4.64 seconds
```

Depth 3

```
Total time taken DFS: 40.98 seconds
```

Depth 4

```
Total time taken DFS: 233.02 seconds
```

Figure 4. Crawling data in DFS

In Figure 3 and Figure 4, you can see the results of performance testing using the BFS and DFS algorithms in the development of web crawlers on the Kumparan site. There are significant differences in the number of files indexed and the time required to run the crawling process. These figures illustrate how each algorithm performs under varying depths, showcasing their strengths and weaknesses in different scenarios.

At level 2 search depth, both algorithms show consistency in the number of files indexed, that is, 21 files. However, there is a striking difference in execution time, where BFS takes around 7.69 seconds, while DFS only takes around 4.64 seconds. At depth level 3, although DFS shows slightly higher speed than BFS, with an execution time of around 40.98 seconds compared to BFS's 51.21 seconds, the number of files indexed by DFS is much smaller, namely 124 files compared to 136 files indexed. by BFS. At depth level 4, the differences between the two algorithms become more striking. BFS succeeded in indexing 949 files, but it took a very long time, namely around 886.94 seconds or around 15 minutes. Meanwhile, DFS succeeded in indexing 470 files with a much shorter execution time, namely around 233.02 seconds or around 4 minutes. These results show that, although DFS has faster performance in some cases, BFS is able to index more files in the same number. Therefore, the choice of search algorithm must consider both the number of files indexed and the time required to perform the crawling process. This highlights the importance of choosing a search strategy that suits the specific needs of the website and the working environment at hand.

Table 1 Comparison of the performance of the BFS and DFS algorithms

Depth	Number of Files		Times	
	BFS	DFS	BFS	DFS
2	21	21	5.35 s	4.50 s
3	136	124	49.41 s	15.57 s
4	949	470	886.94 s	233.02 s

4. CONCLUSION

Web crawlers have become essential tools in the information-rich digital era to ensure quick and easy access to online content, including breaking news from sites like Kumparan. In this case, this research aims to increase the efficiency of indexing and searching for content on the Kumparan site by optimizing the Breadth-First Search (BFS) and depth-first search (DFS) algorithms. The experimental method was carried out by comparing the performance of the two algorithms in indexing and searching website content at various search depths. The research results show that BFS is able to index more files with deeper details, although it takes longer. On the contrary, DFS shows faster performance, especially at finer search depths. For example, at depth 4 BFS indexed 949 files in 886.94 seconds, while DFS indexed 470 files in 233.02 seconds. Selecting the optimal call depends on the specific needs of a website's content indexing task, with BFS being better suited for deep exploration and DFS for fast, entry-level browsing. This research provides important information for the development of web crawler technology, although it is necessary to consider constraints such as dynamic content and compliance with site policies.

ACKNOWLEDGEMENTS

The authors express our sincere thanks to all those who contributed to the preparation of this article. This article would not have been possible without your dedication, efforts, and support. Thanks to the research team who worked hard to collect data, analyze information and formulate findings. This article is not just a

collection of information, but is a great resource for anyone interested in using BFS and DFS algorithms for web crawlers.

We hope that this article will be a useful guide on the necessary procedures and in-depth insight into the efficient implementation of algorithms. Thank you for your cooperation and dedication to make this article a success. We greatly value his role in advancing the understanding of web technologies and algorithm development.

REFERENCES

[1] M. Batari, "'Web Crawler: Pengertian, Cara Kerja, Fungsi, dan Contohnya,'" *Exabytes*. .

[2] Stekom, "'Kumparan (situs web),'" *Universitas STEKOM Semarang.* https://p2k.stekom.ac.id/ensiklopedia/Kumparan_(situs_web) (diakses Mar 04, 2024).

[3] Sulastri dan E. Zuliarso, "Aplikasi Web crawler Berdasarkan Breadth First Search dan Back-Link," *J. Teknol. Inf. Din.*, vol. XV, no. 1, hal. 52–56, 2010.

[4] C. Kustanto, R. M. S, dan P. Viqarunnisa, "Penerapan Algoritma Breadth-first Search dan Depth-first Search Pada FTP Search Engine for ITB Network," *Bandung Inst. Teknol. Bandung*, hal. 1–3.

[5] D. T. Yuwono dan S. Abdul Fadlil, "Perbandingan Algoritma Breadth First Search dan Depth First Search Sebagai Focused Crawler," *Pros. Annu. Res. Semin. 2016*, vol. 2, no. 1, hal. 106–110, 2016.

[6] J. Beel, B. Gipp, dan E. Wilde, "Engine Optimization (ASEO): Optimizing Scholarly Literature for Google Scholar & Co .," *J. Sch. Publ.*, vol. 41, no. May 2014, hal. 2, 2010, doi: 10.1353/scp.0.0082.

[7] A. Muhardono, "Penerapan Algoritma Breadth First Search dan Depth First Search pada Game Angka," *J. Minfo Polgan*, vol. 12, hal. 171–182, 2023, doi: https://doi.org/10.33395/jmp.v12i1.12340.

[8] P. Kumari dan G. Kakhani, "Comparative Analysis of Web PageRank Algorithm using DFS and BFS Crawling," *Int. J. Sci. Res.*, vol. 4, no. 6, hal. 859–863, 2015.

[9] A. E. Wibowo, "Perbandingan Performansi Terhadap Algoritma Breadth First Search (BFS) & Depth First Search (DFS) Pada Web Crawler," *e-Proceeding Eng. Bandung Univ. Telkom*, 2019.

[10] M. Parmar dan H. J. Kaur, "Comparative Analysis of Secured Hash Algorithms for Blockchain Technology and Internet of Things," *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 3, hal. 282–289, 2021.

[11] M. Soulemane dan H. Mahmud, "Crawling the Hidden Web : An Approach to Dynamic Web Indexing Crawling the Hidden Web : An Approach to Dynamic Web Indexing," *Int. J. Comput. Appl.*, no. October, 2016, doi: 10.5120/8717-7290.

[12] B. Pedroza, J. M. G. Calleros, J. G. García, dan C. A. Collazos, "Continuous Evaluation of the Learning Process of Algebra Through a Semi-Automated Tool," *J. Inf. Technol. Res.*, vol. 12, no. 3, hal. 1–20, 2019, doi: 10.4018/JITR.2019070101.

[13] Y. Sun, I. Councill, dan C. L. Giles, "The Ethicality of Web Crawlers," *Conf. Web Intell. Intell. Agent Technol. (WI-IAT), 2010*, vol. 1, 2010, doi: 10.1109/WI-IAT.2010.316.

[14] N. N. S. Faraj, A. Al Azzawi, S. Darwish, dan H. Al Deeb, "The Balance Between Social Life and Work and its Relationship with Work Stress – An Applied Study on the Ministry of Youth and Sports Affairs In the Kingdom of Bahrain," *Int. J. Data Min. Knowl. Manag. Process*, vol. 9, no. January, 2019, doi: 10.5121/ijdkp.2019.9102.

[15] A. Anthony, K. Onasoga, D. U. Ike, dan O. Ajayi, "Council for Innovative Research," *Int. J. Manag. Inf. Technol.*, vol. 5, no. 3, hal. 598–603, 2013.

[16] K. C. Cox, J. Lortie, D. R. Marshall, dan R. E. Kidwell, "Beyond the balance Sheet: The effects of family influence on social performance," *J. Bus. Res.*, vol. 143, hal. 318–330, 2022, doi: https://doi.org/10.1016/j.jbusres.2022.01.013.

[17] "'Kumparan.'" https://kumparan.com/ (diakses Jun 24, 2024).

.