Performance Evaluation Web Scraping BeautifulSoup and Lxml in the ConvexView Application

Achmad Ulul Azmi Wafiqi[1,*], Mochamad Taufiqurrochman Abdul Aziz Zein [2], Tri Anggoro [3]

[1*,2,3] Faculty of Mathematics and Computer Science, Informatics, Universitas Nahdlatul Ulama Al Ghazali, Cilacap, Indonesia

Email: [1,*]azmiwafiqi2@gmail.com, [2]zein@unugha.id, [3]trianggoro1103@unugha.id

[*)] Email Penulis Utama

Abstrak—Cuaca merupakan faktor yang krusial di kehidupan manusia, terutama dalam sektor pertanian, pelayaran, dan aktivitas lainnya. Salah satu indikator utama dari kondisi cuaca ekstrem adalah awan konvektif, yaitu awan yang terbentuk akibat konveksi udara hangat dan berpotensi menyebabkan hujan lebat, petir, dan angin kencang. Pemantauan awan konvektif saat ini dapat dilakukan melalui citra satelit himawari-9 yang menyediakan data citra inframerah untuk mendeteksi keberadaan awan berdasarkan suhu puncaknya. Namun, data ini masih berbentuk mentah dan sulit diinterpretasikan oleh masyarakat umum. Untuk menjawab masalah tersebut, penelitian ini bertujuan merancang dan membangun aplikasi *ConvexView* berbasis web yang mampu memvisualisasikan citra satelit Himawari-9 secara otomatis, khususnya untuk wilayah Cilacap. Penelitian ini juga membandingkan dua pendekatan teknik *Web Scraping* yaitu *BeautifulSoup* dan *lxml*, untuk dibandingkan untuk menentukan teknik pengambilan data yang paling optimal. Bahasa pemrograman yang digunakan adalah Python, dengan dukungan *FastAPI* untuk *backend* dan React JS untuk *frontend.* Metode pengembangan yang digunakan adalah *Prototyping*, karena memungkinkan pengembangan dilakukan secara bertahap dan melibatkan pengguna dalam proses perancangan. Aplikasi ini dirancang agar pengguna dapat melihat visualisasi awan konvektif secara *real-time* serta pengguna dapat mengunduh hasil gambar. Diharapkan, hasil dari penelitian ini tidak hanya memberikan kontribusi dalam bentuk aplikasi berbasis web, tetapi juga sebagai referensi dalam pengembangan aplikasi meteorologi berbasis citra satelit di masa depan.

Kata Kunci: *Web Scraping*, citra satelit himawari-9, awan konvektif, aplikasi *web*, *prototyping*

Abstract— Weather is a crucial factor in human life, especially in agriculture, shipping, and other activities. One of the main indicators of extreme weather conditions is convective clouds, which are clouds formed by warm air convection and have the potential to cause heavy rain, lightning, and strong winds. Monitoring convective clouds can currently be done using Himawari-9 satellite imagery, which provides infrared imagery data to detect the presence of clouds based on their peak temperature. However, this data is still raw and difficult for the general public to interpret. To address this issue, this study aims to design and develop a web-based ConvexView application capable of automatically visualizing Himawari-9 satellite imagery, specifically for the Cilacap region. This study also compares two Web Scraping techniques, BeautifulSoup and lxml, to determine the most optimal data extraction technique. The programming language used is Python, with FastAPI support for the backend and React JS for the frontend. The development method used is Prototyping, as it allows development to be carried out in stages and involves users in the design process. This application is designed so that users can view real-time visualizations of convective clouds and download the images. It is hoped that the results of this research will not only contribute in the form of a web-based application, but also serve as a reference for the development of satellite image-based meteorological applications in the future.

Keywords: Web Scraping, Himawari-9 satellite imagery, convective clouds, web applications, prototyping.

1. INTRODUCTION

Weather plays a critical role in influencing a wide range of human activities across various sectors. In agriculture, for example, weather conditions determine the optimal time for planting and harvesting. In aviation and maritime transport, weather forecasts are essential for ensuring the safety of operations and minimizing the risk of accidents due to adverse atmospheric conditions. Accurate and timely weather information is thus vital for effective decision-making. One of the key indicators used in determining weather conditions is the presence of convective clouds. [1], [2].

Convective clouds are a type of cloud formed through the process of convection, where warm air rises from the surface into the atmosphere and cools down as it ascends, eventually reaching the saturation point and condensing into cloud particles. This process typically occurs in areas with strong surface heating, leading to the formation of towering clouds that extend vertically through the atmosphere. Convective clouds, particularly cumulonimbus (Cb) clouds, are closely associated with extreme weather events such as heavy rainfall, thunderstorms, and in some cases, tornadoes. As a result, the identification and monitoring of convective clouds are fundamental components of modern meteorological information systems.

A primary source of information for detecting convective cloud activity is satellite infrared imagery, specifically from the Himawari-9 satellite operated by the Japan Meteorological Agency (JMA) [3], [4]. Himawari-9 provides high-resolution, near real-time weather imagery that is crucial for analyzing atmospheric conditions. Through Enhanced Infrared (IR) imagery, the cloud-top temperature can be measured, allowing for

more accurate identification of convective clouds. Generally, lower cloud-top temperatures indicate taller clouds, which are a strong sign of convective development. [5], [6].

Although satellite imagery data from Himawari-9 is publicly available through official websites such as the Meteorological, Climatological, and Geophysical Agency (BMKG) and the JMA, the data is usually provided in raw image formats. These images require further processing and interpretation before they can be meaningfully understood by the general public. Moreover, not everyone has the technical expertise to analyze raw satellite data. Therefore, there is a clear need for a system that can automatically process and visualize satellite imagery in a way that is accessible and understandable to non-experts. Such a system would not only benefit the general public but also aid agencies such as BMKG and local disaster management authorities like BPBD in making timely and informed decisions.[7]

To meet this need, a web-based application called ConvexView has been developed. This application is designed to provide real-time visualizations of convective cloud activity, specifically targeting the Cilacap Regency area in Indonesia. The main objective of ConvexView is to make satellite-based weather data more user-friendly and accessible, especially in regions that are prone to extreme weather events. One of the fundamental challenges in developing this application lies in the continuous retrieval of satellite imagery data. Since the relevant data is updated every 10 minutes on the BMKG website, the application must rely on an automated system to fetch the latest images regularly and reliably.

This is where web scraping becomes an essential component. Web scraping refers to the process of extracting data from websites by analyzing the structure of web pages and retrieving specific elements of interest. Unlike Application Programming Interfaces (APIs), which are not always available or may impose access limitations, web scraping offers a flexible approach to data acquisition. According to various studies, web scraping is a practical and efficient method for collecting data from online sources that do not provide formal APIs [8]. In the context of ConvexView, web scraping is used to automate the process of acquiring satellite imagery from BMKG's website, ensuring that the data displayed is always up-to-date without requiring manual downloads.

In the Python programming environment, two widely used libraries for web scraping are BeautifulSoup and lxml. BeautifulSoup is well-known for its ease of use and flexibility in parsing both HTML and XML documents. It is particularly suitable for beginners and for handling documents with irregular structures. On the other hand, lxml is recognized for its speed and efficiency, making it more appropriate for high-performance tasks[9], [10]. It also supports XPath, a powerful query language for selecting elements in XML and HTML documents, which enhances its capability for structured data extraction. For scheduled and periodic scraping, Python's requests library is used to retrieve web pages, while the schedule library handles time-based task execution, allowing for continuous operation without human intervention.

Despite the popularity of these libraries, there is currently a lack of comparative studies specifically examining the performance of BeautifulSoup versus lxml in the context of dynamic data retrieval, such as satellite imagery. For this reason, this research focuses on the implementation of web scraping techniques in the ConvexView application and presents a comparative analysis between BeautifulSoup and lxml. The comparison includes several key metrics, such as parsing speed, accuracy of data extraction, consistency of results, ease of implementation, and memory efficiency[10], [11].

Through this study, it is expected that concrete recommendations can be made regarding the most optimal web scraping method for automatically retrieving satellite imagery data. These findings can not only contribute to the improvement of the ConvexView application but also serve as a reference for future development of weather monitoring systems based on satellite imagery. In particular, the research aims to support efforts in early weather warning and disaster risk mitigation by providing timely and actionable meteorological information to stakeholders in vulnerable regions like Cilacap.

2. RESEARCH METHODS

2.1 Web Scraping

Web scraping is a technique for automatically extracting data from websites by using program code to access and retrieve information available in the HTML structure of web pages[6]. This technique is widely used in web development research for information retrieval[12][13]. In the ConvexView application, web scraping is used to retrieve Enhanced satellite images from the official BMKG website and process them further to display convective clouds. By applying the appropriate scraping techniques, the application can automatically access the latest data from reliable sources and process it into information that can be used in meteorological research and extreme weather studies.

2.2 Scraping Image

In the implementation of web scraping, there are several commonly used libraries for extracting data from the web, including BeautifulSoup and lxml. BeautifulSoup is a Python library used for easily parsing HTML and

XML documents. Since BeautifulSoup uses CSS Selector-based parsing, it is highly effective in handling semi-structured data by searching for elements based on tags, attributes, and DOM hierarchy [10], while Lxml is a faster library compared to BeautifulSoup in terms of parsing HTML and XML documents. Lxml uses XPath-based parsing, which allows elements within the document structure to be searched with high efficiency. However, according to K. Dwicahyo and C. Indah Ratnasari, using Lxml requires more configuration than BeautifulSoup [6].

The scraping process is carried out using two methods, namely the BeautifulSoup and lxml libraries. Both are used to retrieve image elements from the HTML structure of the target site, then extract the URL and temporarily store it in the system for visualization. The flowchart of the scraping process applied to the ConvexView application can be seen in Figures 1 and 2 [12].

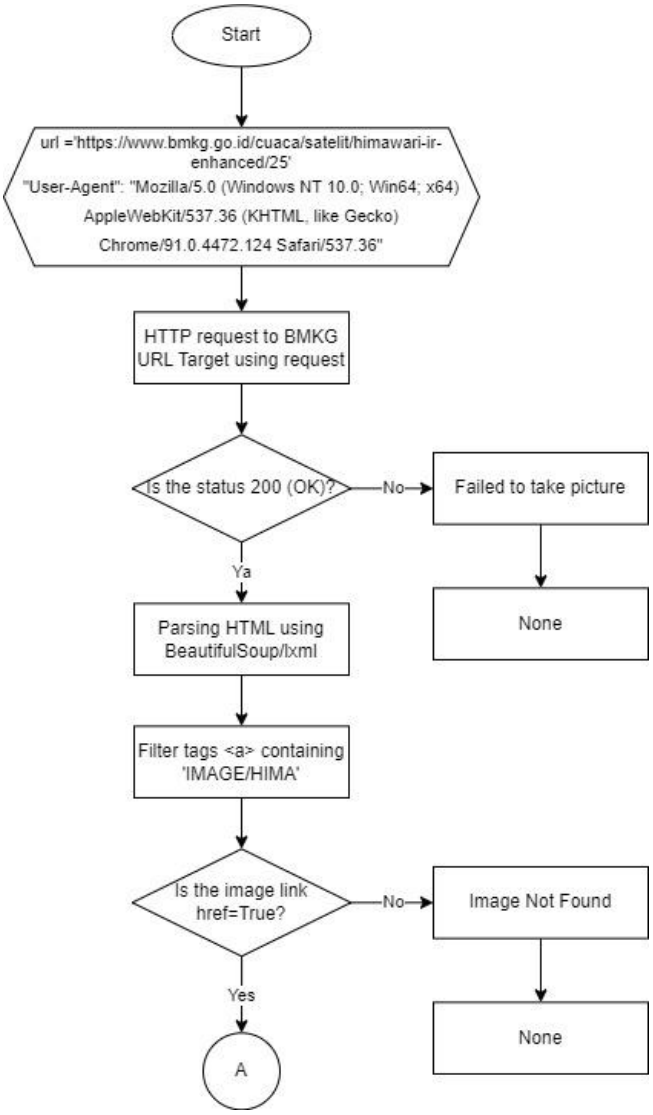

Figure 1. scraping process flowchart part 1

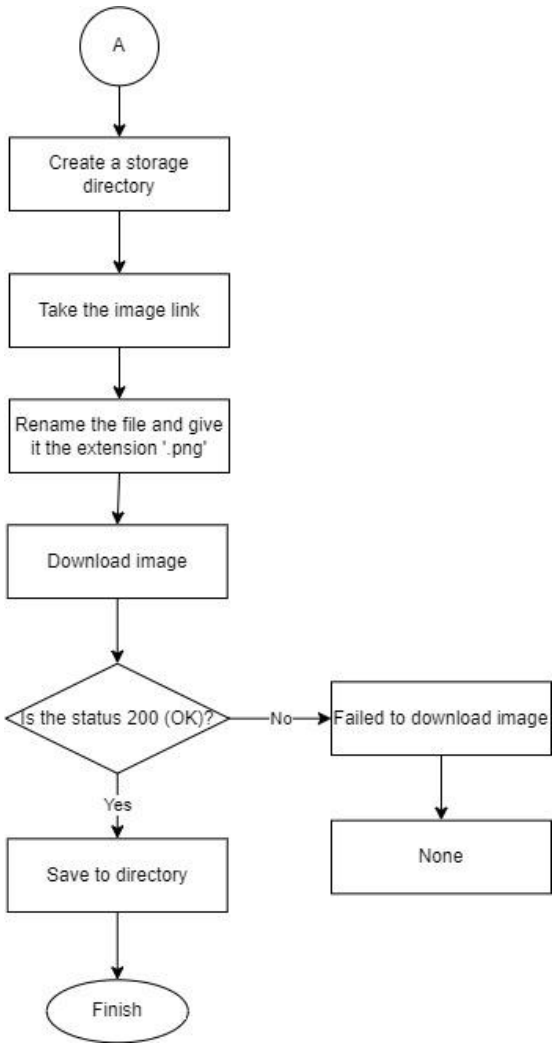

Figure 2. scraping process flowchart part 1

2.3 Prototype Method

Prototyping Model, which is a software development model that focuses on creating prototypes as an initial representation of the system to be developed. This approach allows users to interact directly with the initial system in order to provide feedback that is then used to refine the system in subsequent stages.

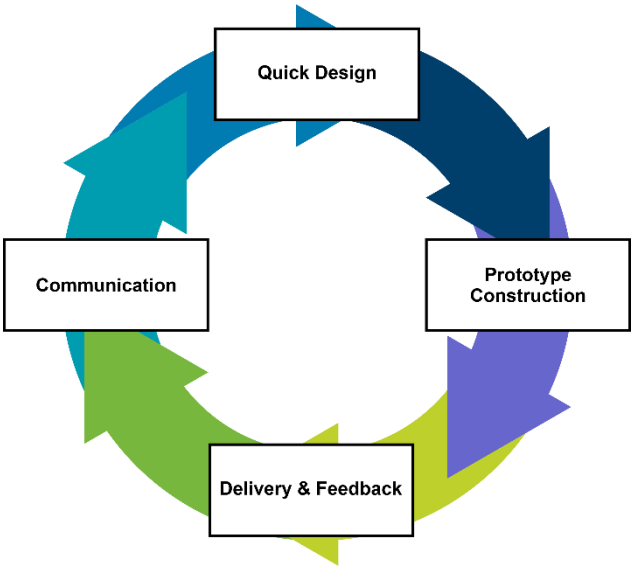

Figure 3. Prototyping Method

The prototyping development model is appropriate when user requirements are not yet fully clear at the outset, as it allows for repeated brainstorming between users and developers until the system meets user expectations. The main stages in this method include communication, quick design, prototype development, delivery, feedback, and final product development [16].

2.2 Research Procedure

This research was conducted in several stages to ensure accurate results and to meet the research objectives. The stages involved are shown in Figure 4

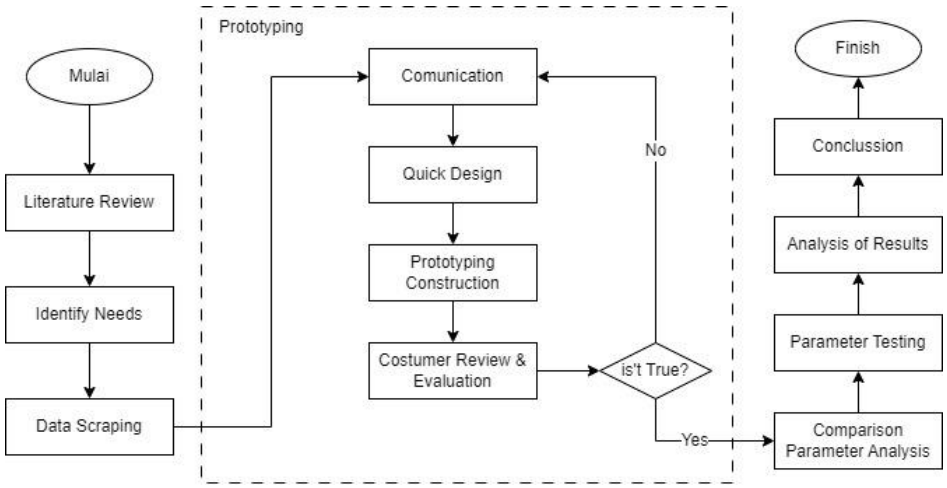

Figure 4. Research Procedure

In identifying a research problem, a thorough literature review is essential as it provides a theoretical, conceptual, and methodological foundation for the development and execution of the research. The literature study in this project focuses on techniques related to Web Scraping for satellite image acquisition, the use of Python programming language, FastAPI, ReactJS, and the prototyping method in application development. Previous studies highlight that manually collecting data from provider websites is time-consuming and inefficient, especially for routine satellite image acquisition tasks. Therefore, Web Scraping emerges as a suitable approach to automate the retrieval of satellite imagery, streamlining the data acquisition process. This method helps avoid human error and supports real-time data-driven systems, particularly in the monitoring of convective clouds. Web Scraping can be implemented using several Python libraries, such as BeautifulSoup and lxml. BeautifulSoup is appreciated for its simple and readable syntax, making it accessible to beginners, while lxml offers faster parsing speeds and support for XPath [11]. Both techniques were applied in the development of the ConvexView web application, which visualizes Himawari-9 Enhanced satellite images. These tools significantly enhance efficiency

and ensure reliability in automated image retrieval for atmospheric monitoring. Python was chosen for its straightforward syntax and extensive ecosystem that supports various tasks including Web Scraping, image processing, and data visualization. Additionally, Python integrates seamlessly with FastAPI, a modern web framework used in this project to develop a RESTFULbackend service. The frontend of the application is developed using ReactJS, which allows for a responsive and interactive user interface [17], [18].

In the needs identification phase, essential tools and resources were defined, including Python with BeautifulSoup, lxml, and Requests libraries for HTTP-based data extraction. A stable internet connection and a foundational understanding of HTML structure were also considered necessary for successful implementation of Web Scraping. During the data scraping phase, Himawari-9 IR Enhanced images were extracted from the official BMKG website. Both BeautifulSoup and lxml methods were employed to retrieve image elements from the site's HTML, extract their URLs, and store them temporarily for visualization purposes. These two methods were compared to assess their performance and determine the most effective for full integration into the ConvexView system. The communication phase, which is the first step in the prototyping methodology, involved discussions between developers and users (or clients) to gather requirements, both functional and non-functional.

This two-way communication ensures mutual understanding of expectations and lays the foundation for the next design phase. The quick design phase involved the creation of an initial system design using Unified Modeling Language (UML) tools such as Use Case Diagrams to define application functionality [15], the following Use Case Diagram ConvexView can be seen in Figure 5.

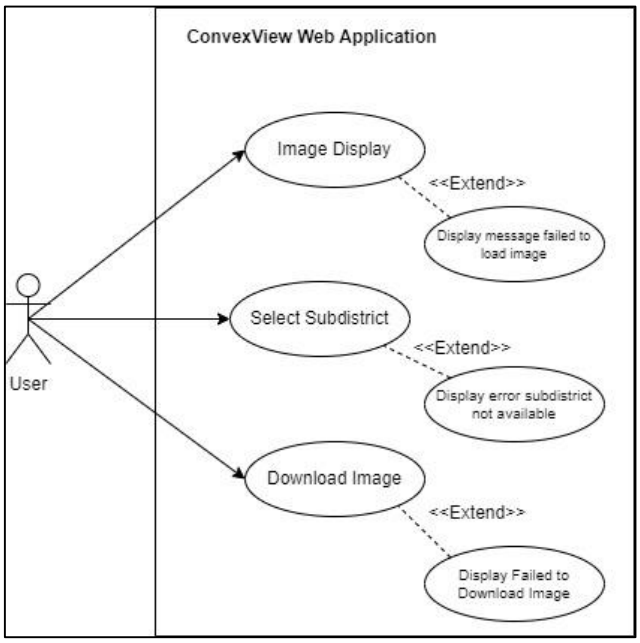

Figure 5. Use Case

The Use Case Diagram above illustrates the interaction between users and the ConvexView application system. Each oval represents a key function provided by the system that can be accessed directly by users. The diagram shows that the system is designed so that users can select a subdistrict, view a visualization of convective clouds, and download the image. The arrows from the actor to each feature illustrate the user's role in performing actions, indicating that the ConvexView application is interactive and responsive to user input. This use case helps in understanding the scope of features and interactions between users and the application.

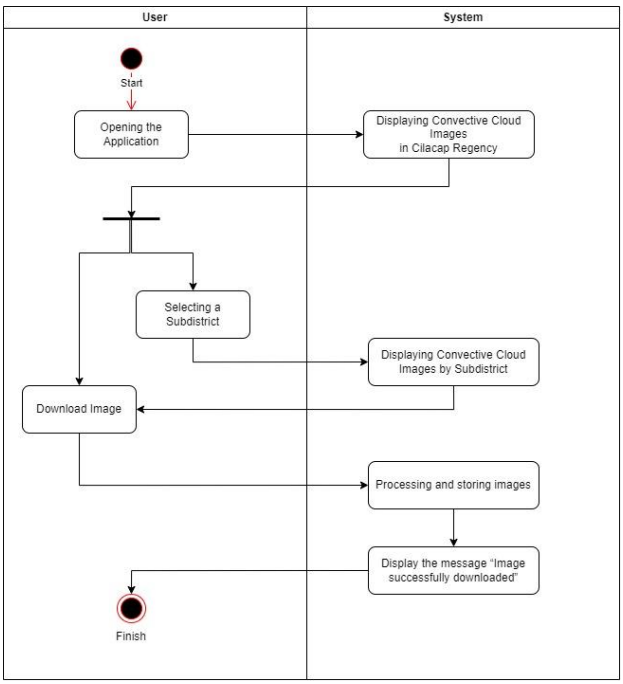

Figure 6. Activity Diagram

This Activity Diagram illustrates the user's workflow when using the ConvexView application to view and download convective cloud imagery. The process begins when the application is opened, displaying an image of the Cilacap region's clouds. Users can select a specific district for visualization, and if they wish to download the image, the system will process and save it to the device, followed by a successful notification. This diagram illustrates the simple interaction between the user and the system in the interactive visualization and distribution of cloud image data.

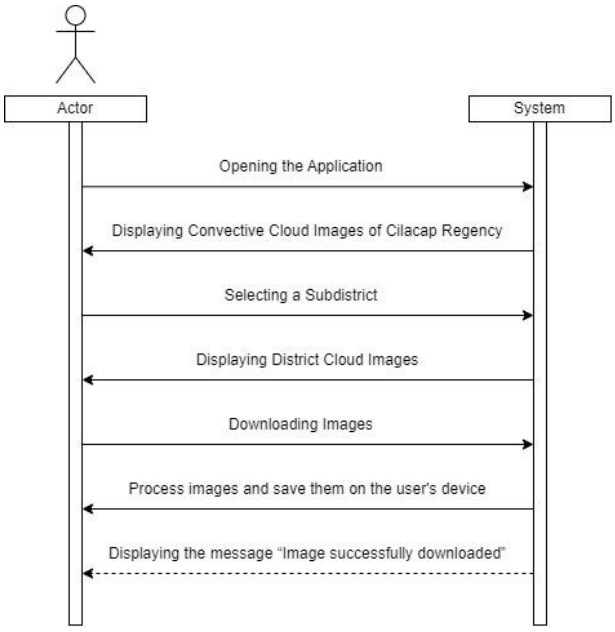

Figure 7. Sequence Diagram

The sequence diagram in Figure 7 explains that when the user opens the application, the system immediately responds by displaying an image of convective clouds in the Cilacap district. The user can then select a subdistrict to see a more specific image according to the location. Once the image is displayed, the user has the option to download it. The system then processes the request to save the image to the user's device and provides feedback in the form of a message that the image has been successfully downloaded.

In the prototyping construction phase, an initial working prototype was built using Python, FastAPI, and ReactJS. This prototype focused on key features: automatic satellite image retrieval through Web Scraping, image processing, and result visualization. Development steps included implementing Web Scraping scripts using both BeautifulSoup and lxml, integrating scraping with processing and visualization, building REST API endpoints with FastAPI, and designing the frontend to display processed imagery to users. The customer review and evaluation phase allowed users to assess the initial prototype and provide feedback [19]. If the prototype did not meet expectations or required additional features, the process would return to the communication phase. Once approved, the project would proceed to the next stages. The comparison parameter analysis phase aimed to evaluate the performance of the two Web Scraping methods. Key parameters included parsing speed, data extraction accuracy, result consistency, ease of implementation, and resource efficiency. These benchmarks helped identify the more suitable method for integration into the final application version.

Following this, the parameter testing phase involved applying identical test scenarios to both scraping methods to ensure fair and comparable results. Performance data from these tests were then collected and systematically analyzed in the results analysis phase. Findings were presented in comparison tables to highlight each method's strengths and weaknesses across the predefined parameters. The conclusion phase summarized the overall research and identified the optimal Web Scraping technique for ConvexView development. This final evaluation not only guided the method selection but also provided insights for future enhancements to the application.

3. RESULTS AND DISCUSSIONS

3.1 Parameter Testing

Scraping speed, data retrieval accuracy, consistency of results, ease of implementation, resource efficiency. On Thursday, June 5, 2025, at 8:10 PM to 9:40 PM WIB/1:10 PM to 2:40 PM UTC, testing was conducted on the 5 parameters. This testing was performed 10 times, and the scraping times for Beautifulsoap and lxml can be seen in Table 1.

Tabel 1. Scraping Speed

	Scraping Speed (s)	
Experiment -	*BeautifulSoup*	*lxml*
1	23,29	23,38
2	25,55	29,78
3	23,54	23,46
4	22,87	22,78
5	23,45	23,49
6	22,81	22,76
7	23,57	23,63
8	22,55	22,19
9	21,89	22,69
10	22,52	22,64
Average	23,20	23,68

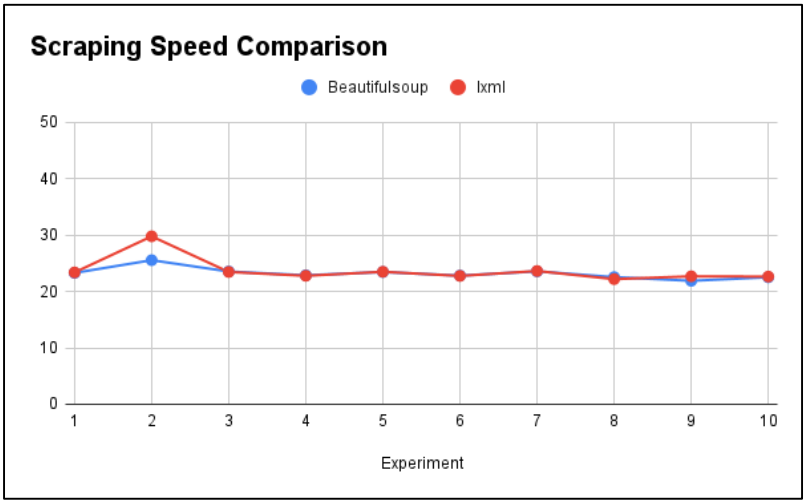

Figure 8. Scraping Speed Chart

From the initial testing of scraping speed parameters, BeautifulSoup proved to be slightly faster than lxml. The average scraping time using BeautifulSoup was 23.2 seconds, while lxml recorded an average time of 23.68 seconds, a difference of 0.48 seconds. Although this time difference is not significant, the results still indicate that BeautifulSoup is slightly more efficient in terms of scraping speed and could be the right choice if speed is a critical factor in systems requiring real-time data retrieval.

Tabel 2. Data Collection Accuracy

Data Collection Accuracy			
Experiment	Time	*BeautifulSoup*	*lxml*
1	13.10UTC	✓	✓
2	13.20UTC	x	x
3	13.30UTC	✓	✓
4	13.40UTC	✓	✓
5	13.50UTC	✓	✓
6	14.00UTC	✓	✓
7	14.10UTC	✓	✓
8	14.20UTC	✓	✓
9	14.30UTC	✓	✓
10	14.40UTC	✓	✓
Amount		9	9

For the second parameter, data retrieval accuracy, testing was conducted by counting the number of recent images successfully retrieved by each method. Both methods were tested under the same conditions and time to ensure fairness in the test results. The results showed that both BeautifulSoup and lxml successfully retrieved image data with the same accuracy, namely 90%. This indicates that both methods are equally reliable in extracting actual information from the target site, and there are no significant differences in their ability to retrieve the latest images. The cause of failure in the second experiment was due to outdated image data on the BMKG website.

Tabel 3. Consistency Of Result

Consistency Of Results		
Experiment	*BeautifulSoup*	*lxml*
1	✓	✓
2	✓	✓
3	✓	✓

4	✓	✓
5	✓	✓
6	✓	✓
7	✓	✓
8	✓	✓
9	✓	✓
10	✓	✓
Amount	10	10

The next test is for consistency of results, which is how stable the method is in producing the same output in multiple trials. This consistency is tested by comparing the number of images successfully captured repeatedly in the same scraping scenario. The test results show that both methods have a very high level of consistency, with a success rate of 100%. This means that both BeautifulSoup and lxml are able to work consistently without any data loss or significant parsing failures during the scraping process.

The comparison of implementation ease parameters was assessed semi-quantitatively by counting the number of main syntax lines in image scraping on the BMKG website, excluding library imports, comments, and schedulers. The following are the results of the calculation of the number of BeautifulSoup (1) and lxml lines (2):

$$Beautifulsoup\ Sintaks\ Score = 34 \quad (1)$$

$$lxml\ Sintaks\ Score = 36 \quad (2)$$

In terms of ease of implementation, this parameter is measured by counting the number of main syntax lines used in each method when performing scraping. The results show that BeautifulSoup is superior, requiring only 34 lines of code, while lxml requires 36 lines of code. Although the difference is only two lines, this can have an impact on application development, especially for developers who prioritize simplicity, code readability, and limited development time. A more concise syntax also allows for faster debugging.

Tabel 4. Memory Usage

Memory Usage		
Experiment	*BeautifulSoup*	*lxml*
1	67,15	50,66
2	68,93	51,44
3	70,03	51,52
4	70,96	51,46
5	72,33	51,68
6	73,48	51,53
7	72,2	44,7
8	70,54	41,08
9	70,46	41,16
10	70,36	40,79
Average	70,6	47,6

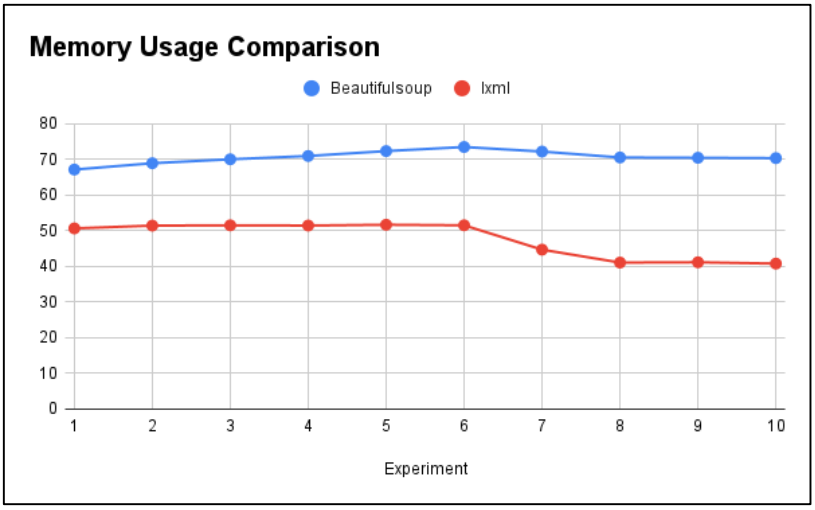

Figure 9. Memory Usage Chart

The last parameter tested was resource usage efficiency, specifically memory usage during the scraping process. Memory usage was measured using profiling tools during the execution process. From the test results, it was found that BeautifulSoup averages 70.6 MB of memory usage, while lxml only uses 47.6 MB. This indicates that lxml is more efficient in terms of memory usage, making it more suitable for applications with resource constraints or when the application is run in a server environment with limited specifications.

After testing five parameters, the results of the five comparison parameters that can be used to determine the Web Scraping method can be seen in Table 5

Tabel 5. Comparison Result

Comparison Results		
Comparison Parameters	BeautifulSoup	lxml
Scraping Speed (s)	23,2	23,68
Data Collection Accuracy (%)	90%	90%
Konsistensi Hasil (%)	100%	100%
Ease of Implementation	34	36
Resource Efficiency (Mb)	70,6	47,6

The results of testing two Web Scraping methods, namely BeautifulSoup and lxml, show that both have their own advantages and characteristics. The testing was conducted based on five main comparison parameters: scraping speed, data extraction accuracy, consistency of results, ease of implementation, and memory resource efficiency. This evaluation aims to determine which method is most optimal for use in the development of the ConvexView application, particularly in automating the extraction of Himawari-9 satellite images from the official BMKG website. Overall, although the differences between the two methods are not significant, each has advantages in certain parameters. BeautifulSoup excels in speed and ease of implementation, while lxml is more efficient in memory usage. Both methods are equally accurate and consistent in the data retrieval process.

Test results show that BeautifulSoup excels in speed and ease of implementation, while lxml is more memory efficient. These findings provide a basis for decision-making in the next stage of development. In the context of scaling up applications or integrating systems running on servers with limited resources, lxml may be the preferred choice. Conversely, for rapid development and lightweight prototyping, BeautifulSoup remains the ideal option.

3.2 User Interface

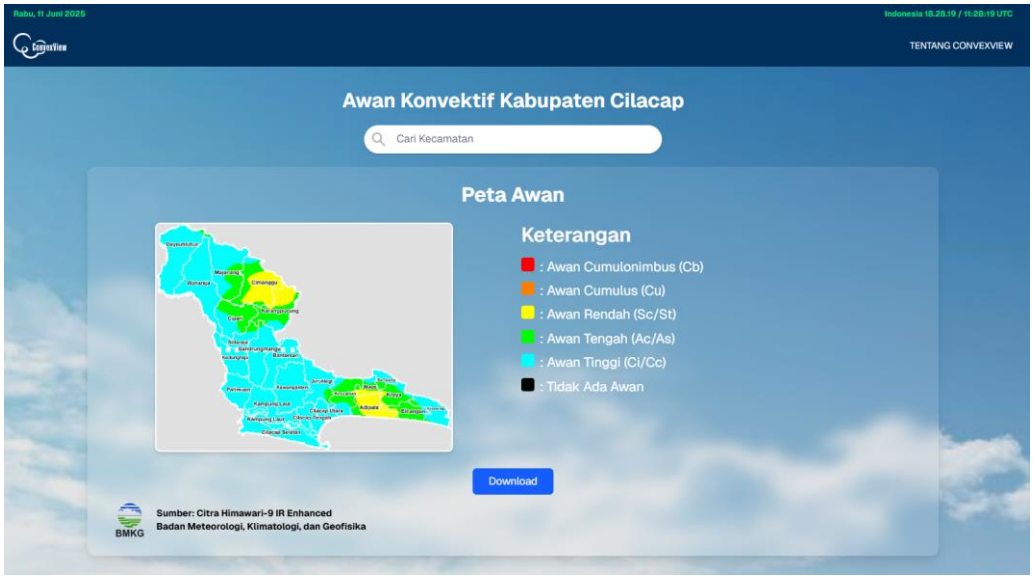

Figure 10. ConvexView User Interface

Figure 8 shows the initial display on the ConvexView application. Users can select the subdistrict they want to display in the subdistrict search.

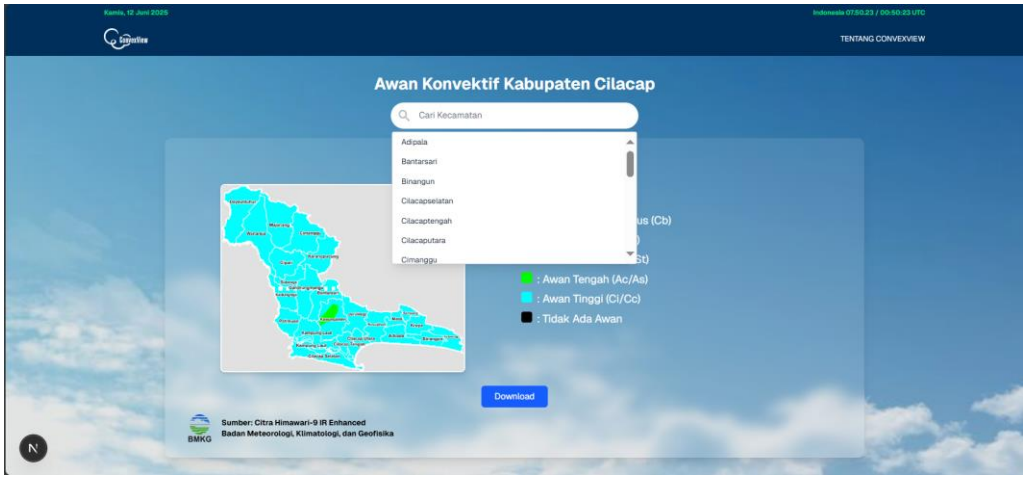

Figure 11. Subdistrict Search Column

When the user selects a subdistrict, an image of the subdistrict cloud map will appear.

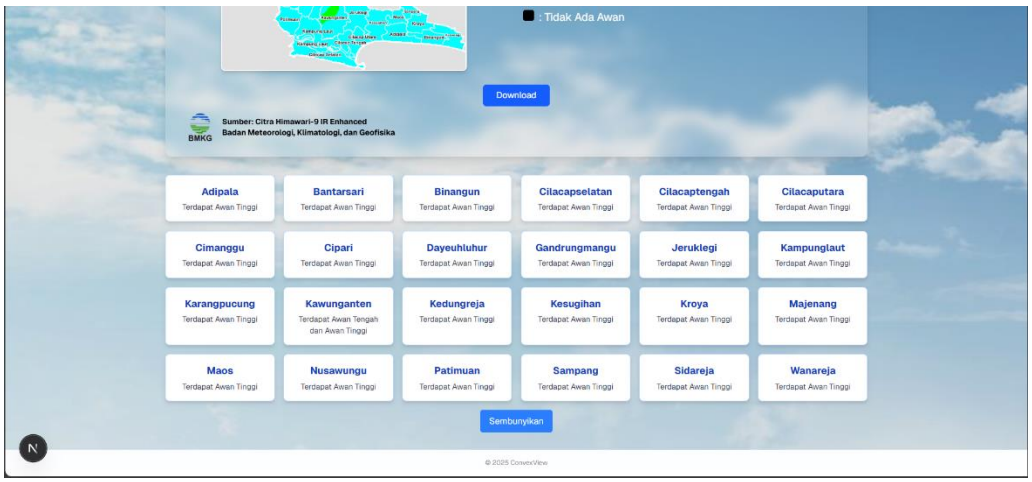

Figure 12. Subdistrict Card Information

The Subdistrict Card Information displays the types of clouds found in the subdistrict. In the implementation of Web Scraping in the ConvexView application, this is done through the image_scrap.py module, which sends HTTP requests to the target page using the request library, then parses the HTML using lxml.HTML to extract the latest image links. The images are then downloaded, converted to PNG format, and saved in a local directory for further processing.

The next step is the classification process based on peak cloud temperature, which is executed using the identifikasinocrop.py code. The results of the classification process are then cropped in the Cilacap area and subdistricts in Cilacap Regency using the kabupaten_crop.py and Adipala_crop.py codes, as well as other subdistrict crops. The results of the cropping process based on the regency and subdistrict areas are stored in the folder directory of each subdistrict. The entire process is performed automatically by scheduling the execution time using the schedule library, with a 10-minute interval to match the satellite data update time on the BMKG website. The entire pipeline is run using the run_pipeline function. The ConvexView application integrates the scraping process into the backend using FastAPI to display all the resulting images.

4. CONCLUSION

Based on comparative testing between BeautifulSoup and lxml across five evaluation parameters scraping speed, data extraction accuracy, result consistency, ease of implementation, and memory efficiency it can be concluded that both libraries offer robust web scraping capabilities, each excelling in different areas. BeautifulSoup stands out for its faster scraping speed and simpler syntax, making it ideal for projects that require rapid development and ease of maintenance. Its readable code structure provides flexibility, especially for developers seeking quick deployment without deep configuration. In contrast, lxml offers superior memory efficiency, which becomes a significant advantage in large-scale applications or when working in memory-constrained environments. Despite these differences, both libraries achieved the same levels of accuracy (90%) and consistency (100%) when extracting image data from the BMKG (Indonesian Meteorology, Climatology, and Geophysical Agency) website. This indicates that either library can be reliably used for accurate and consistent data extraction. Therefore, the choice between the two depends largely on the specific technical requirements and constraints of the application in question.

In the ConvexView application, the web scraping module plays a critical role in the automated data processing pipeline. The image_scrap.py module, built using lxml.html, is responsible for retrieving the latest Himawari-9 satellite images at 10-minute intervals. These images are then analyzed through classification algorithms implemented in the identifikasinocrop.py module. Following classification, the images are cropped according to administrative boundaries at both the regional and subdistrict levels using kabupaten_crop.py and additional subdistrict-specific cropping modules. Each output is saved in its respective directory for organized access. The integration of the schedule library ensures the system operates continuously and in sync with BMKG's data updates. FastAPI serves as the backend framework, delivering the processed images to a React-based frontend interface. This seamless integration of web scraping, image processing, and web service delivery enables ConvexView to function as a fully automated, efficient, and user-oriented application for real-time cloud

monitoring in Cilacap Regency. Its modular design also ensures scalability, allowing for easy adaptation to different regions and future expansion of system functionalities.

REFERENCES

[1] A. R. S. Maghriza, Y. D. Haryanto, Munawar, O. Yosafat, and N. F. Silalahi, "ANALISIS KEJADIAN HUJAN LEBAT DI WILAYAH CILACAP," vol. 5, no. 2, pp. 144–154, 2024, doi: 10.53682/gjppg.v5i2.8499.

[2] M. O. R. Hutagalung, "ANALISIS KEJADIAN HUJAN LEBAT MENGGUNAKAN DATA CITRA SATELIT HIMAWARI-8 (Studi Kasus Kota Manado, 16 Januari 2021)," *Fisitek J. Ilmu Fis. dan Teknol.*, vol. 6, no. 2, pp. 16–22, 2023, doi: 10.30821/fisitekfisitek.v6i2.14445.

[3] R. F. Ramdani, "Analisis Kejadian Hujan Lebat dan Banjir Kabupaten Pati Menggunakan Metode Cloud Convective Overlays dan Red Green Blue Convective Storms pada Satelit Himawari 8," *J. Penelit. Sains*, vol. 23, no. 3, p. 150, 2021, doi: 10.56064/jps.v23i3.647.

[4] R. Kurniati, S. L. H. Pakpahan, and A. Mulya, "Analisis Kejadian Hujan Lebat Menggunakan Citra Satelit HIMAWARI-8 (Studi Kasus Pulau Bintan, 7 November 2020)," *Pros. Semin. Nas.*, vol. 3, no. 1, pp. 130–141, 2021.

[5] A. Stevan Yondra, D. Triyanto, and S. Bahri, "Implementasi Web Scraping Untuk Mengumpulkan Informasi Produk Dari Situs E-Commerce," *J. Komput. Dan Apl.*, vol. 10, no. 01, 2022.

[6] K. Dwicahyo and C. Indah Ratnasari, "Perbandingan Metode Web Scraping Dalam Pengambilan Data: Kajian Literatur," *Automata*, vol. 4, 2023.

[7] dan G. Badan Meteorologi, Klimatologi, "Citra Himawari-9 IR Enhanced - Indonesia." https://www.bmkg.go.id/cuaca/satelit/himawari-ir-enhanced

[8] U. Khandelwal and A. KB, "FastAPI vs. The Competition: A Security Feature Showdown with a Proposed Model for Enhanced Protection," *Interantional J. Sci. Res. Eng. Manag.*, vol. 08, no. 06, pp. 1–5, 2024, doi: 10.55041/ijsrem35726.

[9] F. Sembiring and D. P. Sari, "Penerapan teknik scraping python pada website marketplace indonesia," *Integr. (Journal Inf. Technol. Vocat. Educ.*, vol. 2, no. 1, pp. 15–22, 2020, doi: 10.17509/integrated.v2i1.28243.

[10] K. Sharma and G. M. Borkar, "Comparative Analysis of Dynamic Web Scraping Strategies: Evaluating Techniques for Enhanced Data Acquisition," *Adv. Commun. Syst.*, pp. 241–252, 2024, doi: 10.56155/978-81-955020-7-3-22.

[11] E. Uzun, T. Yerlikaya, and O. Kırat, "Comparison of Python Libraries used for Web Data Extraction," *J. Tech. Univ. - Sofia Plovdiv branch, Bulg.*, vol. 24, no. May, pp. 87–92, 2018, [Online]. Available: https://erdincuzun.com/wp-content/uploads/download/plovdiv_journal_2018_01.pdf

[12] M. D. Al Farizi, R. Hidayat, and M. Abdi, "OPTIMASI METODE SCRAPING DATA PRODUK DARI PLATFORM OPTIMIZATION OF DATA SCRAPING METHODS FOR PRODUCTS FROM TOKOPEDIA . COM PLATFORM," vol. 2, no. September, pp. 1172–1181, 2023.

[13] S. T. Aji, S. Rosad, and M. Abror, "Penerapan Teknik Scraping Pada Prototype Papan," *CENTIVE*, vol. 3, no. Dec, 2023.

[14] D. Ardiyansah, O. Pahlevi, and T. Santoso, "Implementasi Metode Prototyping Pada Sistem Informasi Pengadaan Barang Cetakan Berbasis Web," *Hexag. J. Tek. dan Sains*, vol. 2, no. 2, pp. 17–22, 2021, doi: 10.36761/hexagon.v2i2.1083.

[15] A. Fikriyya and R. T. Dirgahayu, "Implementasi Prototyping dalam Perancangan Sistem Informasi Sekolah Desa Pendar Foundation Yogyakarta," *J. UII Autom.*, vol. 1, no. 2, pp. 1–9, 2020.

[16] E. W. Fridayanthie, H. Haryanto, and T. Tsabitah, "Penerapan Metode Prototype Pada Perancangan Sistem Informasi Penggajian Karyawan (Persis Gawan) Berbasis Web," *Paradig. - J. Komput. dan Inform.*, vol. 23, no. 2, pp. 151–157, 2021, doi: 10.31294/p.v23i2.10998.

[17] S. A. Rismawan and Y. Syahidin, "Implementasi Website Berita Online Menggunakan Metode Crawling Data Dengan Bahasa Pemrograman Python," *J. Tek. Inform. dan Sist. Inf.* , vol. 10, no. 3, pp. 167–178, 2023, [Online]. Available: http://jurnal.mdp.ac.id

[18] R. Yusuf Azhari, "Web Service Framework : Flask Dan Fastapi," *Technol. Informatics Insight J.*, vol. 1, no. 1, pp. 58–65, 2022, doi: 10.32639/tiij.v1i1.54.

[19] J. Chen, "Application Technology of Atmospheric Dispersion Models Based on FastAPI+Vue," *Acad. J. Eng. Technol. Sci.*, vol. 6, no. 11, pp. 120–126, 2023, doi: 10.25236/ajets.2023.061118.