J-INTECH (Journal of Information and Technology)
Accredited Sinta 4 Ministry of Higher Education, Science and Technology Republic of Indonesia SK No. 10/C/C3/DT.05.00/2025
E-ISSN: 2580-720X || P-ISSN: 2303-1425

Implementation of SerpApi-Based Google Scholar Monitoring System for Faculty Publication Evaluation

Farhan Nafi Emillul Fata[1], Danang Arbian Sulistyo[2*]

[1,2] Asia Institute of Technology and Business Malang, Faculty of Technology and Design, Department of Computer Science, Jl. Soekarno Hatta-Rembuksari No. 1A Mojolangu, Lowokwaru District, Malang City, Indonesia

Keywords
Accreditation; Django; Evaluation; Faculty Performance evaluation; Google Scholar; SerpApi

****Corresponding Author:***
dananga.arbian@asia.ac.id

Abstract
Evaluating faculty performance based on scientific publication metrics is a crucial tool in higher education quality assurance and accreditation reporting. However, conventional web scraping techniques based on HTML DOM parsing are highly vulnerable to layout changes and anti-bot blocking on Google Scholar. This study aims to develop a web-based Faculty Publication Monitoring System using the Django framework and SerpApi as a stable data extraction solution, following the Waterfall Software Development Life Cycle (SDLC) model. The results demonstrate that the API integration successfully bypasses anti-bot mechanisms with a 100% data extraction accuracy rate within the validated sample, validated through a cross-referencing protocol involving manual checks of 10 randomly sampled faculty profiles against the live Google Scholar database. Furthermore, mass data synchronization for 63 active faculty profiles was securely completed in approximately 180 seconds, while negative testing confirmed successful data normalization of null values. In conclusion, the system enhances institutional administrative efficiency in providing valid publication datasets to support institutional accreditation needs, delivering an estimated time-reduction of over 99%. Given the performance and stability delivered by this architecture, it can be adopted by other institutions facing similar challenges in automated bibliometric monitoring.

1. Introduction

Faculty performance evaluation is a fundamental tool for ensuring the quality of higher education institutions, particularly regarding the implementation of the Tridharma of Higher Education. Currently, scientific publication productivity and citation counts serve as the primary globally recognized quantitative indicators for measuring a researcher's academic impact. In this regard, Google Scholar has become the most widely cited bibliometric metrics platform due to its extensive indexing scope and open access [1],[2]. However, with the massive increase in the number of faculty members and publications, manually monitoring metrics such

as total citations, h-index, and i10-index has become highly inefficient, prone to human error, and complicates the process of compiling reports for institutional accreditation.

To address these inefficiencies, previous studies have attempted to develop automated compilation systems using web scraping techniques. Research implementing Google Scholar scraping has utilized Document Object Model (DOM) parsing methods based on HyperText Markup Language (HTML) [3] . Additionally, there are studies that employ automated scheduling using features such as Task Scheduler or cron jobs to perform periodic data [4] . Although these systems successfully collect data, they have significant architectural weaknesses. Conventional HTML DOM-based scraping techniques are highly dependent on the tag structure (such as class, id, or layout) of the target web page. If Google Scholar changes its layout structure, the script will immediately encounter errors. Furthermore, Google Scholar implements strict anti-bot mechanisms, where massive and repetitive access requests (continuous requests) from conventional scripts will trigger CAPTCHA challenges or IP address blocking (IP Ban). At the institutional level, while previous research has successfully developed a web-based lecturer research information system for the Research and Community Service Institution (LP2M) at the Asia Institute of Malang [5], that system focuses primarily on administrative documentation and lacks an automated pipeline for real-time bibliometric extraction.

Based on this research gap, a new approach is needed that does not rely on visual HTML elements, but rather on a stable raw data structure. Therefore, this study proposes the development of a web-based Faculty Research Publication Monitoring System by integrating SerpApi. SerpApi's primary advantage lies in its ability to extract search results in real-time into the standardized JavaScript Object Notation (JSON) format [6], while simultaneously bypassing the access blocking constraints experienced by conventional methods.

This study aims to design and develop a Scientific Publication Monitoring System for Faculty Members using the Django framework and the SQLite relational database. To demonstrate the reliability of the proposed method, the system was implemented using faculty data from the Asia Institute in Malang as a case study. The results of this study are expected to offer a novel approach in the form of a significantly more stable data extraction architecture for generating publication datasets to support accreditation.

To address the aforementioned gaps in conventional data extraction methods, the explicit contributions of this research are structured as follows:

1. Proposing a scalable API-based architectural design utilizing SerpApi to automate the extraction of Google Scholar bibliometric data, effectively bypassing the limitations of conventional DOM scraping, such as structural layout dependency and anti-bot interruptions.
2. Developing an automated data transformation mechanism to map unstructured, raw JSON payloads into a structured relational SQLite database within a backend framework environment.
3. Providing a validated, zero-variance extraction framework capable of achieving a 100% accuracy rate within the validated sample under collective data synchronization workloads, offering a reliable blueprint for institutional academic publication monitoring systems.

2. Research Method

This study utilizes secondary bibliometric data sourced from Google Scholar. Specifically, the dataset scope comprises the publication records of 63 active faculty members at the Asia Institute of Malang, with the primary data extraction period conducted in May 2026. This database was selected based on the breadth of its academic literature indexing and the availability of globally recognized researcher performance indicators. The details of the data collected and the data processing workflow are as follows:

2.1. Collected Data Objects

A data object is an entity or a set of attributes that represents the characteristics, features, or specific information of an object within a dataset. The data objects in this study consist of several main components [7] . First, lecturer profiles, which include the variables of lecturer name, affiliation, email, research interests, total citations, total citations since 2020, h-index, h-index since 2020, total i10-index, and i10-index since 2020. Second, scientific publication data, which includes research title, author name, year of publication, journal name, number of citations, and publication link. Third, visual data in the form of graphs used to illustrate the development of citations and the periodic academic performance of lecturers, as presented in Table 1.

Table 1. Categories and Attributes of Research Data

NO	Data Category	Specific Attributes/Variables
1	Faculty Profile	Faculty Name, Affiliation, Email, Research Interests, Total Citations (All), Total Citations (Since 2020), h-index (All), h-index (Since 2020), i10-index (All), i10-index (Since 2020)
2	Research Publications	Title, Author, Year, Journal, Citation, Link
3	Visual Data	Year, Total Citations per year

2.2. Data Processing Flow

The data processing flow consists of five sequential stages. First, in the inventory stage, the process begins by collecting the Google Scholar Author IDs of faculty members as the primary search key. Second, during data acquisition, raw data is retrieved based on those Author IDs using an Application Programming Interface (API) integration. Third, the system receives a JSON response, in which the retrieved data is returned in JavaScript Object Notation (JSON) format. Fourth, in the validation and cleansing stage, the extracted raw data is processed to ensure there are no anomalies or null values. Finally, in the storage stage, the validated data is stored in a relational SQLite database.

The software development approach follows the Waterfall Software Development Life Cycle (SDLC) model [8], which consists of Requirements Analysis, System Design, User Interface Implementation, and System Testing using Black Box Testing to validate the extraction performance. The Waterfall model was explicitly chosen over Agile or iterative approaches because the system requirements, particularly the JSON data schema mapped from the SerpApi endpoint and the institutional database structure, are strictly defined, static, and unlikely to evolve dynamically during the initial development phase.

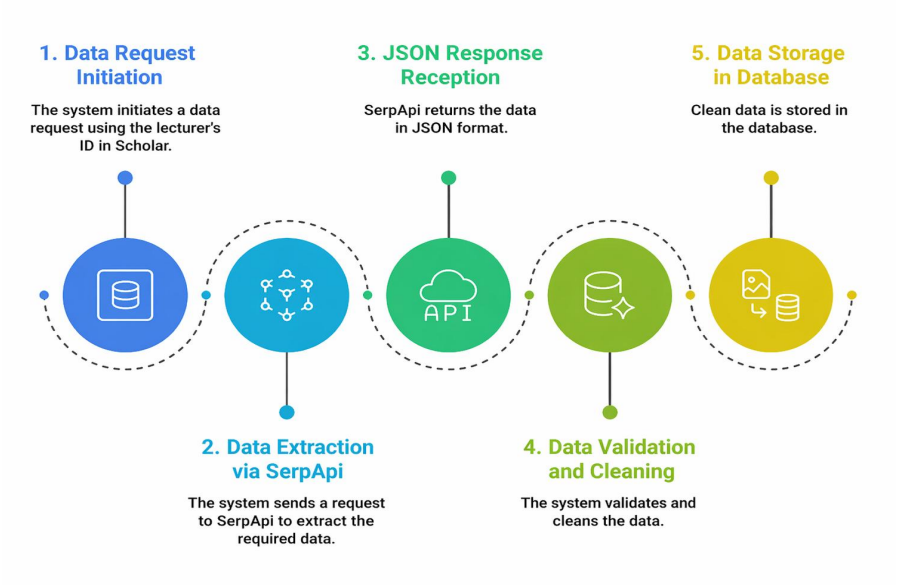


Figure 1. Data processing process

2.3. Requirements Analysis

The requirements analysis phase defines the functional boundaries, security constraints, and data specifications necessary for the system to operate systematically. Given that this system interacts directly with third-party services and manages institutional data, architectural security and access control are identified as core non-functional requirements[9], [10],[11].

The system mandates strict credential isolation, meaning that the SerpApi API key is not hardcoded into the source code but is securely stored within environment variables (.env) on the backend server. Furthermore, functional access requirements dictate that all operations related to the synchronization module (Update Data) and report downloads must be protected by the Django framework's built-in session-based authentication layer. To prevent unauthorized parties from forcibly executing data update scripts, the interface configuration requires the integration of Cross-Site Request Forgery (CSRF) protection tokens on every request form.

2.4. System Design

The system design phase translates the analyzed requirements into structural architectures, interactive behavioral workflows, and database mapping schemas before the coding phase begins.

2.4.1. Functional Modeling

A Use Case Diagram is used to visualize the functional interactions between actors and the system, thereby clearly illustrating the system's boundaries and context. This visual modeling approach, based on the Unified Modeling Language (UML), plays a crucial role in the system design phase, as it translates software requirements into structured interaction flows. In this system, there is one main actor, the Administrator, who has full access rights to carry out the primary objectives available within the system [12],[13],[14]. With a Use Case Diagram, the relationships between actors and system features can be identified in a structured manner, thereby facilitating the system design and development process, as shown in Figure 2.

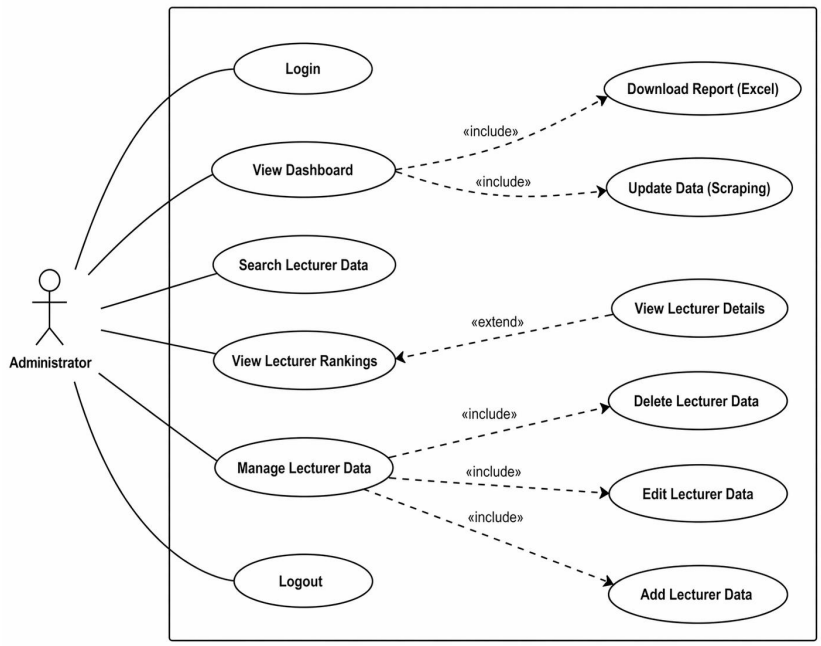


Figure 2. Use Case Diagram

2.4.2. Process Workflow Modeling

The Activity diagram is designed to map the backend process workflow, focusing on two main mechanisms [12] , [15] . Figure 3 illustrates the data extraction, response validation, and storage processes. Meanwhile, Figure 4 shows the reporting workflow, in which the system converts the aggregated data into an Excel format and packages it into a ZIP archive for download.

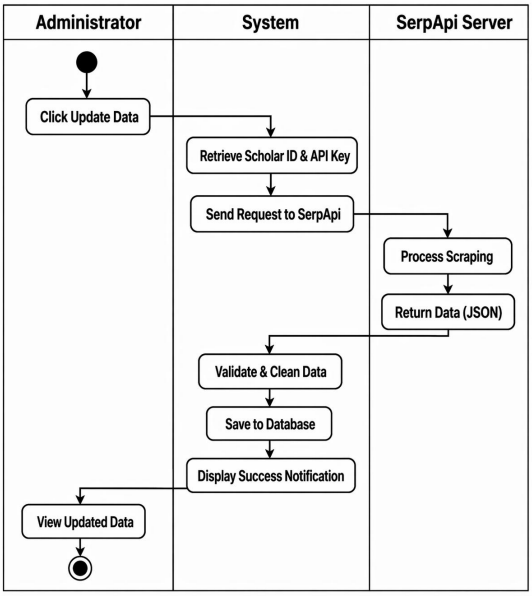


Figure 3. Data Update Activity Diagram

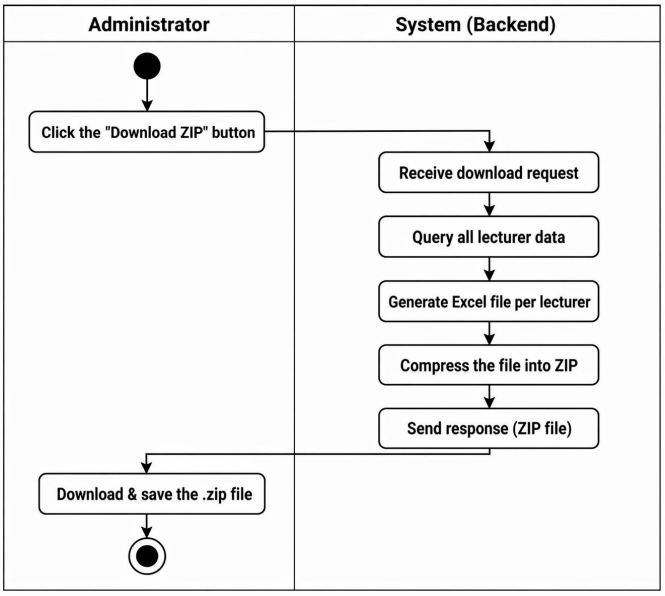


Figure 4. Activity Diagram Download Excel

2.4.3. Data Architecture Mapping

The main challenge in Application Programming Interface (API) integration is data exchange format transformation. SerpApi returns data from Google Scholar in JavaScript Object Notation (JSON) format, which is hierarchical and not strictly structured (NoSQL-like). Therefore, a mapping architecture is required so that the payload can be stored in a relational SQLite database.

In the design of this system, the main JSON object is split into two relational entities. The author_metrics array object (which contains h-index, total citations, and i10-index data) is extracted and mapped to the parent Faculty table. Meanwhile, the nested array articles is iterated using a loop on the Django backend side, then mapped as a child table Publications that has a foreign key relationship with the Faculty table. This mapping process is supplemented with sanitization functions to ensure that string and integer data types are stored in accordance with database constraints, thereby maintaining data integrity when exported to Excel format.

2.5. Implementation Environment

To ensure research reproducibility, this system was developed and tested within a specific computing environment. The hardware and software configurations utilized from the implementation (coding) through the testing stages are presented in Table 2.

Table 2. Hardware and Software Specifications of the Implementation Environment

Category	Component	Specification
Hardware	Processor	Intel Core i5 / AMD Ryzen 5 (or equivalent) with a 64-bit architecture
	RAM	8 GB DDR4, to ensure smooth processing of large-scale data and backend framework compilation
	Storage	Solid State Drive (SSD), to accelerate read/write queries on the local database
Software	Operating System	Windows 10/11 (64-bit) / macOS / Linux
	Environment & Backend	Python version 3.10.x and Django framework version 4.x
	Database	SQLite version 3 (built-in integration with Django)
	Third-party Integration	API Endpoint from SerpApi Google Scholar Engine
	IDE	Visual Studio Code

2.6. Testing Strategy

The testing phase establishes a structured evaluation strategy to validate system performance and stability before final deployment. The strategy incorporates a two-tier evaluation mechanism. First, functional validation is executed via Black Box Testing. This approach focuses exclusively on checking the interface execution against expected results—such as credential authentication inputs, record insertions, and ranking sorting dynamics ensuring the functional components operate error-free under user operations.

Second, a rigorous validation and verification protocol is defined to evaluate the reliability and precision of the data extraction architecture. This involves conducting multi-layered cross-validation trials where

randomly sampled faculty records are audited. The extracted bibliometric attributes (titles, citations, and years) stored in the local relational database are cross-referenced directly against the live Google Scholar web interface to statistically quantify the data integrity and margin of error under collective data synchronization workloads.

3. Result and Discussions

This web-based system for tracking faculty research publications has been successfully implemented, featuring administrator authentication and a basic data monitoring dashboard (Create, Read, Update, Delete) used for data management in the database[16],[17]. However, the primary focus of the testing and discussion in this section is on the performance of the system's core architecture, specifically the mechanism for extracting bibliometric data using SerpApi integration.

3.1. SerpApi-Based Data Extraction and Validation Implementation

The process of updating faculty performance metrics in this system no longer relies on conventional HTML DOM parsing methods; instead, it uses calls to the SerpApi endpoint. When an administrator triggers a data update command, the backend system built with Django extracts the Author ID parameter from the local database and sends it via an HTTP GET request to the SerpApi server. The Data Extraction and Validation screen can be seen in Figure 5.

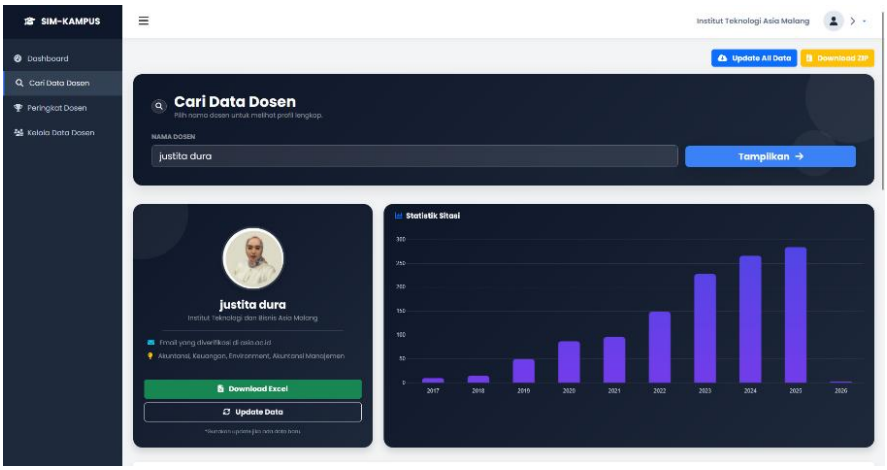


Figure 5. Data Extraction and Validation View

In this architecture, SerpApi acts as an intermediary agent that executes searches on Google Scholar, bypasses potential anti-bot blocks, and returns responses directly (on-demand) in the form of structured JSON objects. This automated extraction approach aligns with previous web-based application frameworks that successfully leverage automated web scraping to streamline specialized data retrieval for institutional needs [18] . After the JSON object is received, the system performs data validation and cleansing. The system's algorithm is designed to detect metric attributes (such as annual citation counts) with null or undefined values in the API response and automatically convert them to the integer 0. This is done to prevent data structure anomalies during faculty ranking calculations. The validated data is then committed to a relational SQLite database. It is subsequently displayed in the faculty member details menu, as shown in Figure 6.

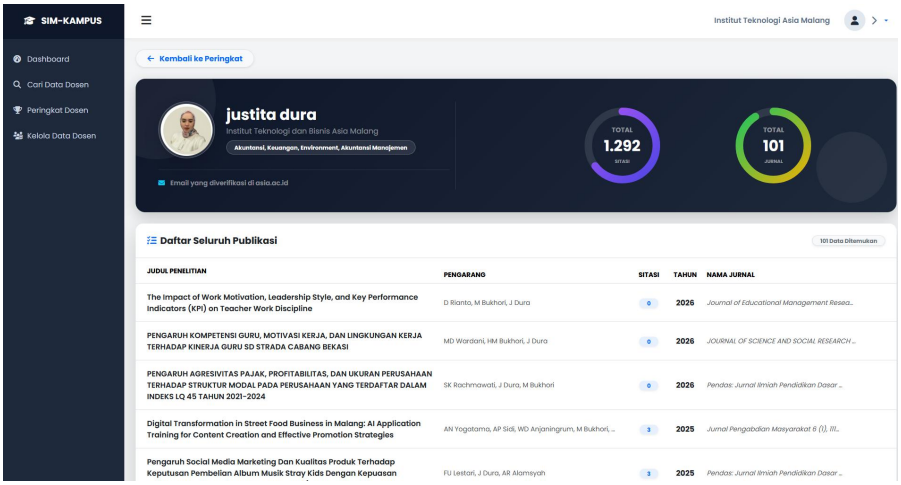


Figure 6. Lecturer Detail Menu Display

As previously explained, this system is equipped with a faculty ranking feature whose data is synchronized with Google Scholar through a data update process. Rankings are determined based on data extracted from each faculty member's publication count and citation count. This feature aims to provide the institution with an overview of faculty members' research productivity each year. The faculty ranking menu is shown in Figure 7.

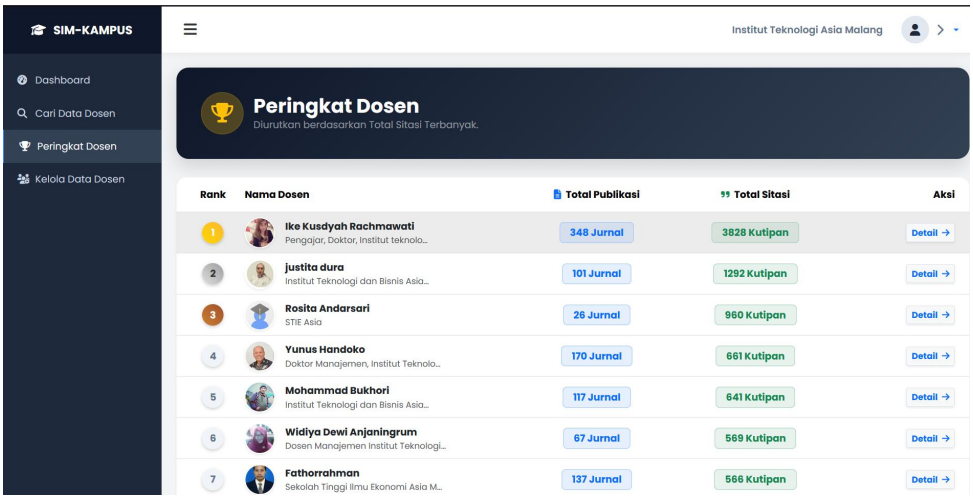


Figure 7. Lecturer Ranking Menu Display

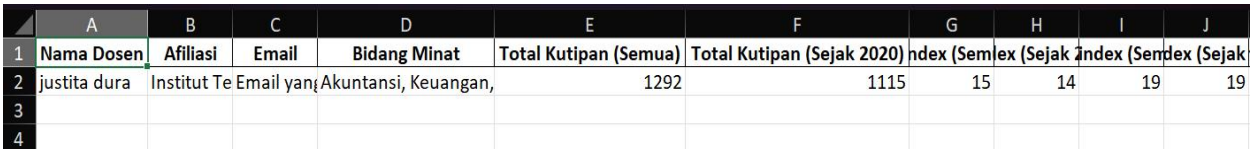
3.2. Implementation of Collective and On-Demand Data Synchronization

The developed system provides a data synchronization mechanism to ensure that stored information is always updated in accordance with the latest conditions in the data source. This synchronization process is carried out using two approaches: collective synchronization and on-demand synchronization. Collective synchronization is a comprehensive data update process applied to all faculty data stored in the database. In this process, the system retrieves data from external sources based on each faculty member's Author ID, then updates the existing information incrementally. This approach is used when large-scale data updates are required, such as to ensure all data is up-to-date.

Meanwhile, on-demand synchronization is a data update process performed specifically on certain data as needed by the user. This process occurs when a user accesses or selects specific faculty data, so the system

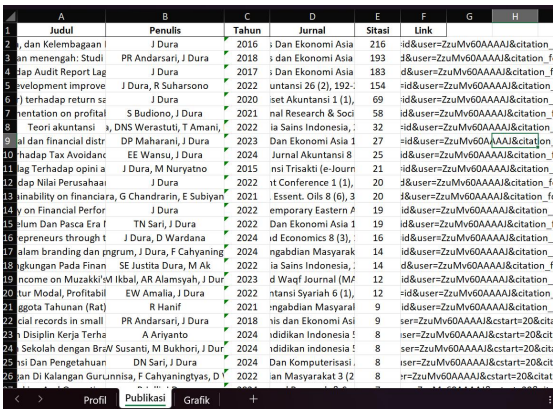
will only update the requested data without processing the entire dataset. This approach is more efficient as it reduces system load and processing time.

With these two mechanisms, the system provides flexibility in data management, whether for comprehensive updates or quick, selective updates based on user requests. After the data synchronization process is complete, the system provides a download feature that allows users to convert the data into a set of spreadsheet files (Excel) for reporting purposes. Visualizations of the results of this process are shown in Figures 7, 8, and 9.



	A	B	C	D	E	F	G	H	I	J
1	Nama Dosen	Afiliasi	Email	Bidang Minat	Total Kutipan (Semua)	Total Kutipan (Sejak 2020)	ndex (Sem	ex (Sejak	ndex (Sem	dex (Sejak
2	justita dura	Institut Te	Email yan	Akuntansi, Keuangan,	1292	1115	15	14	19	19
3										
4										

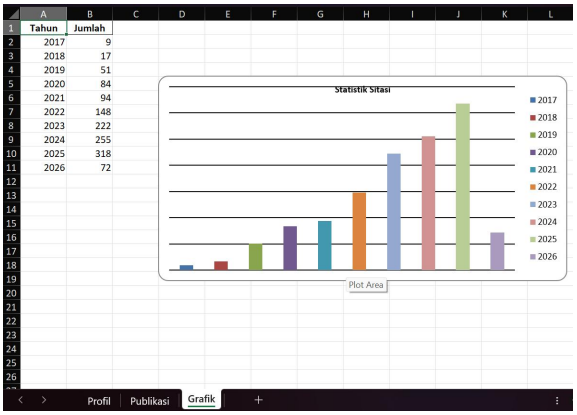
Figure 8. Profile sheet view



	A	B	C	D	E	F	G	H
1	Judul	Penulis	Tahun	Jurnal	Sitasi	Link		
2	y dan Kelembagaan	J Dura	2016	s Dan Ekonomi Asia	216	id&user=ZzuMv60AAAAJ&citation_		
3	an menengah: Studi	PR Andarsari, J Dura	2018	s dan Ekonomi Asia	193	d&user=ZzuMv60AAAAJ&citation_f		
4	dap Audit Report La	J Dura	2017	s Dan Ekonomi Asia	183	d&user=ZzuMv60AAAAJ&citation_f		
5	velopment improve	J Dura, R Suharsono	2022	untansi 26 (2), 192-	154	id&user=ZzuMv60AAAAJ&citation_		
6	) terhadap return sa	J Dura	2020	iset Akuntansi 1 (1),	69	id&user=ZzuMv60AAAAJ&citation_		
7	nentation on profital	S Budiono, J Dura	2021	nal Research & Soci	58	id&user=ZzuMv60AAAAJ&citation_		
8	Teori akuntansi	, DNS Werastuti, T Amani,	2022	ia Sains Indonesia,	32	id&user=ZzuMv60AAAAJ&citation_		
9	al dan financial distr	DP Maharani, J Dura	2023	Dan Ekonomi Asia 1	27	id&user=ZzuMv60AAAAJ&citation_		
10	hadap Tax Avoidanc	EE Wansu, J Dura	2024	Jurnal Akuntansi 8	25	id&user=ZzuMv60AAAAJ&citation_		
11	ag Terhadap opini a	J Dura, M Nuryatno	2015	nsi Trisakti (e-Journ	21	id&user=ZzuMv60AAAAJ&citation_		
12	dap Nilai Perusahaa	J Dura	2022	t Conference 1 (1),	20	d&user=ZzuMv60AAAAJ&citation_f		
13	inability on financiara, G	Chandrarin, E Subiyan	2021	Essent. Oils 8 (6), 3	20	d&user=ZzuMv60AAAAJ&citation_f		
14	y on Financial Perfor	J Dura	2022	emporary Eastern A	19	id&user=ZzuMv60AAAAJ&citation_		
15	lum Dan Pasca Era I	TN Sari, J Dura	2022	Dan Ekonomi Asia 1	19	id&user=ZzuMv60AAAAJ&citation_		
16	epreneurs through t	J Dura, D Wardana	2024	id Economics 8 (3),	16	id&user=ZzuMv60AAAAJ&citation_		
17	alam branding dan ngrum,	J Dura, F Cahyaning	2024	ngabdian Masyarak	14	id&user=ZzuMv60AAAAJ&citation_		
18	gkungan Pada Finan	SE Justita Dura, M Ak	2022	ia Sains Indonesia,	14	d&user=ZzuMv60AAAAJ&citation_f		
19	ncome on Muzakki'sM	Ikbal, AR Alamsyah, J Dur	2023	d Waqf Journal (MA	12	id&user=ZzuMv60AAAAJ&citation_		
20	tur Modal, Profitabil	EW Amalia, J Dura	2022	ntansi Syariah 6 (1),	12	id&user=ZzuMv60AAAAJ&citation_		
21	ggota Tahunan (Rat)	R Hanif	2021	ngabdian Masyaral	9	id&user=ZzuMv60AAAAJ&citation_		
22	cial records in small	PR Andarsari, J Dura	2018	is dan Ekonomi Asi	9	ser=ZzuMv60AAAAJ&cstart=20&cita		
23	Disiplin Kerja Terha	A Ariyanto	2024	ndidikan Indonesia	8	iser=ZzuMv60AAAAJ&cstart=20&cit		
24	Sekolah dengan Bra	Susanti, M Bukhori, J Dur	2024	ndidikan indonesia	8	ser=ZzuMv60AAAAJ&cstart=20&cita		
25	si Dan Pengetahuan	DN Sari, J Dura	2024	Dan Komputerisasi	8	iser=ZzuMv60AAAAJ&cstart=20&cit		
26	an Di Kalangan Gurunnisa,	F Cahyaningtyas, D	2022	an Masyarakat 3 (2	8	r=ZzuMv60AAAAJ&cstart=20&citat		

Profil | Publikasi | Grafik

Figure 9. Publication sheet view



	A	B
1	Tahun	Jumlah
2	2017	9
3	2018	17
4	2019	51
5	2020	84
6	2021	94
7	2022	148
8	2023	222
9	2024	255
10	2025	318
11	2026	72

Profil | Publikasi | Grafik

Figure 10. Graph sheet view

3.3. System Testing

Testing is a series of processes conducted to verify and validate whether a developed application meets user requirements and is free of errors (bugs) before implementation. This study uses the Blackbox Testing method, which plays a crucial role in software testing by ensuring that the overall system functions properly [19],[20]. This testing involves campus staff as the primary users of the application and assesses whether the developed scraping system has any issues by using data from the 63 active faculty members at the Asia Malang Institute. Refer to Table 3.

Table 3. Functional Interface Testing Results

Tested Feature	Testing Scenario	Test Results	Status
Administrator Login	Enter valid credentials on the authentication page./Incorrect password	The system grants access and redirects the user to the Dashboard page.	Valid
Add New Faculty Member	Enter Name, Faculty Scholar ID, then click Save.	The data is stored in the database and appears in the faculty list table.	Valid
Manage lecturer data	Verifying whether the data that has been successfully entered functions	The data changes, specifically in the number of lecturers in FTD and	Valid

	dynamically	FEB, after a new lecturer is added.	
Checking the Lecturer Details feature	Verify whether the citation data has changed or increased after the data update	The data increases according to what is available on Google Scholar	Valid
Faculty Rankings	Access the Faculty Rankings page	The system automatically sorts and displays the list of lecturers based on the highest number of citations and publications.	Valid
View Faculty Details	Ensure that the citation data and graph are updated accordingly after the data update process has been successfully completed.	Citation metrics numbers and graphs increase/change according to the data available on Google Scholar.	Valid

3.4. Performance and Accuracy Testing of the SerpApi Integration Extraction

The second phase constitutes the core testing of the study to evaluate the performance of the data extraction architecture [21]. This testing focuses on three key metrics: success rate against anti-bot systems, response time, and data accuracy (data validation).

To scientifically substantiate the 100% extraction accuracy claim, a rigorous validation methodology was implemented during the testing phase. The validation protocol utilized a multi-layered manual cross-referencing technique. Specifically, the evaluation involved 10 repeated trial iterations, where 10 faculty profiles were randomly sampled from the total dataset of 63 extracted profiles. This sample size, representing approximately 16% of the total population, is methodologically justified for deterministic API testing, as any structural extraction error in the JSON payload would be uniformly detected within this sample proportion. The comparison criteria strictly evaluated three primary data points: publication titles, individual citation metrics, and publication years, matching the extracted data stored in the SQLite database directly against the live Google Scholar web interface. Across all 10 validation iterations, the manual audit confirmed a perfect data match, resulting in a 0% margin of error. This zero-variance statistical reliability confirms that the SerpApi-based architecture extracts bibliometric data flawlessly without any loss or structural truncation, as shown in Table 4.

Table 4. Performance and Accuracy Testing of SerpApi Integration for Data Extraction

Tested Metric	Testing Scenario	Testing Result	Status
Data Extraction Accuracy (Cross Validation)	Extract the profiles of 10 Faculty randomly, then compare the h-index, total citations, and journal titles with Google Scholar.	The SerpApi JSON object shows 100% accuracy within the validated sample. No discrepancies in numbers or missing publication data were found.	Valid
Individual Extraction Time Performance (Single Request)	Updating a specific Faculty profile.	The backend system processes requests and displays data in a timeframe ranging from 2 to over 60 seconds per profile, depending entirely on the total volume of publication records extracted.	Valid
Mass Extraction Stability	Extract the profiles of 63 Faculty sequentially in a single run.	The system successfully processed all requests in approximately 180 seconds.	Valid

			The success rate was 100%, with no errors or CAPTCHA blocks.	
Data Conversion Validation (Data Cleansing)	Type	Extract Faculty profiles that contain null publication year values in the citation field.	The backend algorithm successfully detects and converts null values to the integer 0 before committing them to the database.	Valid

Furthermore, compared to previous studies, the SerpApi integration demonstrates superior structural stability and performance. HTML DOM techniques have been employed to implement Google Scholar scraping, enabling the extraction of bibliographic information from web pages [3]. However, DOM-based parsing is highly susceptible to script failures and extraction errors whenever Google Scholar alters its front-end HTML layout. The proposed API-based architecture overcomes this structural limitation and achieves a 100% extraction accuracy because it bypasses volatile visual elements entirely, directly extracting data into standardized JSON objects, rendering visual layout changes irrelevant.

Automated Google Scholar scraping has been reported to be susceptible to CAPTCHA challenges and IP bans during continuous requests, reducing its effectiveness for sustained data extraction [4]. The proposed SerpApi approach significantly improves upon this implementation regarding robustness and long-term maintainability by utilizing automatic proxy rotation to bypass these anti-bot mechanisms. In contrast to conventional methods, this proposed system successfully processed 63 sequential profile extractions without a single anti-bot interruption, thereby proving its reliability for institutional-scale data aggregation.

Analytically, completing this mass data synchronization for 63 faculty profiles in approximately 180 seconds is considered highly efficient compared to institutional manual procedures. The proposed architecture achieves this speed because the API bypasses the need to render heavy web-page graphical elements (such as CSS and JavaScript), focusing purely on raw data retrieval. Assuming a manual verification and data entry process takes roughly 10 minutes per profile, the conventional institutional approach would require approximately 10 working hours. Therefore, completing the extraction in 180 seconds translates to an estimated time-reduction efficiency of over 99%, providing a robust and scalable solution for institutional monitoring.

To address potential data inconsistencies from the SerpApi JSON response, the backend architecture implements a structured exception-handling and data normalization algorithm. This mechanism is crucial because missing metric attributes can trigger fatal errors during database operations. The core data cleansing logic is presented in Algorithm 1.

Algorithm 1. Data Validation and Normalization

```
Input: raw_json_response.publications
Output: validated_data

Initialize validated_data ← ∅
for each publication in raw_json_response.publications do
// Citation validation
if publication.citations is NULL or UNDEFINED then
publication.citations ← 0
end if
// Publication year validation
if publication.year is NULL or UNDEFINED then
```

```
9:   publication.year ← 0
10:  end if
11:  validated_data ← validated_data ∪ {publication}
12:  end for
13:  save validated_data to database
```

As shown in Algorithm 1, the system iteratively inspects key metric fields. If a null or undefined value is detected, the algorithm intercepts the anomaly and coercively assigns it an integer value of 0. This proactive cleansing mechanism guarantees structural integrity before the data reaches the relational database, successfully preventing system crashes during academic ranking calculations.

Beyond these technical safeguards, the urgency of transitioning from conventional scraping to API-based architectures is strongly supported by recent literature. Although Google Scholar remains an important source of bibliometric information, limitations in large-scale data acquisition necessitate the development of automated integration tools. Moreover, recent research has highlighted the importance of integrating bibliometric data from multiple scholarly sources to improve data coverage and support large-scale academic analysis and monitoring activities [22]. Building upon these findings, the SerpApi-based system proposed in this study provides a scalable and reliable approach for automated academic data extraction and publication monitoring. Furthermore, the system demonstrated stable performance and high extraction accuracy while mitigating common limitations associated with conventional scraping methods.

3.5. Negative Testing / Error Handling

In addition to testing functionality under ideal conditions, this study also conducted negative scenario testing to evaluate the system's robustness when faced with invalid inputs or technical glitches[23]. This is important to ensure that the system can provide informative feedback to users without crashing. the results are presented in Table 5.

Table 5. Negative Testing / Error Handling

Negative Scenario	Given Condition	Expected Result	Status
Input Fictitious Author ID	The administrator enters an unregistered Google Scholar ID (e.g., xyz123_invalid) and clicks the Update button.	The system detects a 404 / Not Found error response from SerpApi and displays a notification stating "Lecturer data/ID not found," without force-closing the application	Valid
Data Duplication (Primary Key Constraint)	The administrator is attempting to save faculty data using an Author ID that already exists in the database.	The database rejected the commit, and the interface displayed a warning message stating that the Scholar ID is already registered.	Valid
Network Failure (Time-out)	Intentionally disconnect the device's internet connection while the Update All Data process (63 lecturers) is in progress.	The system safely cancels the remaining requests in the queue, saves the data that has already been successfully retrieved, and displays a "Connection lost" message.	Valid

3.6. System Limitations and Data Disambiguation Challenges

Although the proposed architecture achieves a high extraction success rate, there are several inherent limitations in this study that need to be identified. First, the system is completely dependent (vendor lock-in)

on the availability of a third-party service (SerpApi). If there is a connectivity outage (downtime) on that API server, the metric update feature on the local system will be delayed.

Second, there are challenges related to author name disambiguation. Google Scholar often hosts researcher profiles with identical names. To mitigate the risk of misidentified profile extraction, this system is designed with Author ID-based locking parameters (a 12-character alphanumeric unique ID assigned by Google to each profile), rather than using standard character strings. During the testing phase involving 63 active faculty profiles, the system achieved a 0% false-positive rate, effectively preventing the inclusion of anomalous or irrelevant publication data that does not align with the actual identities of faculty members at the Asia Institute of Malang.

Third, there are operational limitations regarding data request quotas (API Quota). Currently, the system is used periodically for monthly data updates for 63 active faculty members. Using SerpApi on the free tier with a limit of 250 requests per month, the current quota utilization stands at exactly 25.2% (63/250 requests), demonstrating a sustainable operational buffer for the current institutional size. However, the critical threshold for upgrading to a paid commercial plan occurs when total monthly requests breach the 250-request limit. This ceiling will be exceeded if the active faculty population grows by more than 187 new members, or if the administration increases the synchronization frequency to four times a month (resulting in 252 requests). Therefore, system administrators must switch to a commercial subscription plan (paid tier) to ensure the availability of adequate extraction quotas in line with the growth in the number of faculty members at the institution.

4. Conclusions and Future Works

This study successfully implemented a Faculty Scientific Publication Monitoring System using SerpApi integration, which has proven to be more effective at overcoming anti-bot blocking than conventional methods. The system achieved 100% extraction accuracy within the validated sample, high synchronization efficiency (±180 seconds for 63 faculty members), and excellent error handling in normalizing data and preventing system crashes. The implementation of data synchronization features and Excel report generation enhances institutional administrative efficiency significantly. Specifically, compiling valid accreditation datasets for 63 faculty members conventionally required an estimated 10 working hours of manual data entry. This estimation is based on empirical observations of institutional administrative procedures, assuming an average of 10 minutes per faculty profile for manual search, verification, and data entry into spreadsheets. With the proposed system, this workload was reduced to approximately 180 seconds of automated processing, representing a time-reduction efficiency of over 99%. Given the performance and stability delivered by this architecture, it can be adopted by other institutions facing similar challenges in automated bibliometric monitoring.

For future research refinement and development, it is recommended that the scope of data extraction be expanded by integrating Application Programming Interfaces (APIs) from other academic indexing sources, such as Scopus, SINTA, or Garuda. Additionally, the application of Machine Learning algorithms to predict future trends in faculty publication productivity could be incorporated to support strategic decision-making by institutional leadership.

5. References

[1] Y. Rostiani and E. Tjandra, “Analisis Bibliometrik Studi Perkembangan Metode Service Quality pada Database Google Scholar Menggunakan Vosviewer (Studi Literatur Tahun 2016–2020),” *SMATIKA Jurna*l, vol. 12, no. 1, pp. 85–93, 2022, doi: 10.32664/smatika.v12i01.677.

[2] I. Muhammad, F. Marchy, H. K. Rusyid, and D. Dasari, “Analisis Bibliometrik: Penelitian Augmented Reality dalam Pendidikan Matematika,” *JIPM (Jurnal Ilmiah Pendidikan Matematika)*, vol. 11, no. 1, pp. 141–155, 2022, doi: 10.25273/jipm.v11i1.13818.

[3] A. F. Putri, G. Manik, F. Nabila, and N. Chamidah, "Implementasi Scraping Google Scholar Menggunakan HTML DOM untuk Pengumpulan Data Artikel Dosen UPN Veteran Jakarta Berbasis Web," *Prosiding Seminar Nasional Mahasiswa Bidang Ilmu Komputer dan Aplikasinya,* vol. 2, no. 1, pp. 668–678, Apr. 2021.

[4] M. Koprawi and W. S. Putra, "Web Scraping Implementation on Google Scholar for Lecturer Research Profiling," *Science Tech: Jurnal Ilmu Pengetahuan dan Teknologi*, vol. 9, no. 1, pp. 59–72, 2023, doi: 10.30738/st.vol9.no1.a14188.

[5] A. Nizar and L. Widayanti, "Sistem Informasi Penelitian Dosen pada LP2M ITB Asia Malang Berbasis Web," *Jikom: Jurnal Informatika dan Komputer*, vol. 13, no. 1, pp. 18–29, Apr. 2023, doi: 10.55794/jikom.v13i1.95.

[6] M. A. Khder, "Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application," *International Journal of Advances in Soft Computing and its Applications*, vol. 13, no. 3, 2021, doi: 10.15849/IJASCA.211128.11.

[7] R. Irawan and R. Kaestria, "Pemodelan Basis Data dengan Pendekatan Model Data Berorientasi Objek pada Native Apps Lokasi Tempat Ibadah di Kota Palangka Raya," J*urnal Sains Komputer dan Teknologi Informasi*, vol. 2, no. 2, pp. 36–43, 2020, doi: 10.33084/jsakti.v2i2.1479.

[8] I. Sommerville, *Software Engineering*, 10th ed. Boston, MA: Pearson, 2015.

[9] L. Adi Saputra, F. Muhammad Akbar, F. Cahyaningtias, M. Puspa Ningrum, and A. Fauzi, "Ancaman Keamanan pada Sistem Informasi Manajemen Perusahaan," Jurnal Pendidikan Siber Nusantara, vol. 1, no. 2, pp. 58–66, Aug. 2023, doi: 10.38035/jpsn.v1i2.48.

[10] T. D. Andini, P. Subekti, and M. Islamiyah, "Rancang Bangun Aplikasi Sarana Prasarana Institut Asia Malang," POSITIF: *Jurnal Sistem dan Teknologi Informasi,* vol. 7, no. 1, pp. 70–76, May 2021, doi: 10.31961/positif.v7i1.1060.

[11] A. Tedyyana, M. Fauzi, and F. Ratnawati, "Revamp Keamanan Web Service Milik PT XYZ Menggunakan REST API," *Digital Zone:* J*urnal Teknologi Informasi dan Komunikasi*, vol. 12, no. 1, pp. 1–10, May 2021, doi: 10.31849/digitalzone.v12i1.6378.

[12] F. S. Mukti and R. W. D. Anjasari, "Desain Unified Modeling Language untuk Sistem Informasi Unit Pelaksana Teknis Jaringan dan Komputer Institut Asia Malang Berbasis QR-Code," *Jurnal Ilmiah NERO*, vol. 7, no. 2, pp. 155–168, 2022, doi: 10.21107/nero.v7i2.212.

[13] B. Tri W. Utomo and A. Hadi, "Perancangan Knowledge Management System Dosen untuk Menunjang Kurikulum Pembelajaran pada Mahasiswa di Institut Asia Malang," vol. 15, no. 1, pp. 69–74, 2021, doi: 10.32815/jitika.v15i1.515

[14] T. A. Rospricilia and M. N. P. Ma'ady, "Pemodelan Integration Use Case (IUC): Perancangan Use Case Diagram (UML) untuk Sistem-sistem yang Terintegrasi," *INTEGER: Journal of Information Technology,* vol. 9, no. 2, pp. 165–172, 2024, doi: 10.31284/j.integer.2024.v9i2.6345.

[15] D. Akbar, T. D. Andini, Z. B. D. Asmoro, and A. Noercholis, "Pemanfaatan Metode Agile Development dalam Sistem Informasi Kepegawaian di Institut Asia Malang," *POSITIF: Jurnal Sistem dan Teknologi Informasi,* vol. 9, no. 2, pp. 75–83, Nov. 2023, doi: 10.31961/positif.v9i2.1995.

[16] Z. M. Anggraini, "Pengujian CRUD Link pada Aplikasi LinkStack Berbasis Web Menggunakan Black Box Testing," *Journal of Information Systems and Business Technology,* vol. 1, no. 1, pp. 28–34, 2025.

[17] F. Hermawan and A. F. O. Pasaribu, "Implementasi Web Service Sebagai Penyedia Informasi untuk Aplikasi Pengelolaan Jadwal Pemberian Pakan Ikan (Studi Kasus: Pokdakan Karya Bersama)," *Jurnal*

Informatika dan Rekayasa Perangkat Lunak (JATIKA), vol. 4, no. 3, pp. 335–341, 2023, doi: 10.33365/jatika.v4i3.2720.

[18] D. A. Prastiningtyas and M. N. Hidayat, "Design and Implementation of an Android-Based Scholarship Search Application Using Web Scraping Techniques," *J-INTECH (Journal of Information and Technology),* vol. 13, no. 2, pp. 396–400, 2025.

[19] H. Nurfauziah and I. Jamaliyah, "Perbandingan Metode Testing antara Blackbox dengan Whitebox pada Sebuah Sistem Informasi," *Jurnal Visualika*, vol. 8, no. 2, pp. 105–113, Oct. 2022.

[20] A. P. Putra, F. Andriyanto, T. D. M. Harti, and W. Puspitasari, "Pengujian Aplikasi Point of Sale Berbasis Web Menggunakan Black Box Testing," *Jurnal Bina Komputer*, pp. 74–79, 2014, doi : 10.33557/binakomputer.v2i1.757

[21] R. A. Sianturi et al., "Perancangan Pengujian Fungsional dan Non Fungsional Aplikasi," *JICON (Jurnal Informatika dan Komputer)*, vol. 9, no. 2, pp. 133–141, 2021, doi: 10.35508/jicon.v9i2.4706.

[22] E. M. Gavron, A. L. Pinto, F. L. do Canto, and M. Talau, "A Tool for Bibliometric Analysis of Journals Indexed in Google Scholar Metrics and OpenAlex," *Transinformação,* vol. 37, 2025, doi: 10.1590/2318-0889202537e2515501.

[23] W. Winata, A. W. R. Emanuel, and Herlina, "Pengujian Website EPOS PT XYZ Menggunakan Metode Black Box Testing," *Jurnal Informatika Atma Jogja*, vol. 3, no. 2, pp. 99–106, Nov. 2022.