Implementation of Web Scraping on Hindusthan Times TTC: A Flask-based News Aggregator System

Bhuvaneswari Yennapusala
Computer Engineering-AI Department
Marwadi University
Rajkot, Gujarat, India
bhuvaneswariyennapusala.116514@marwadiuniversity.ac.in

Sohith Bukka
Computer Engineering-AI Department
Marwadi University
Rajkot, Gujarat, India
bukkasohith.118516@marwadiuniversity.ac.in

Priyanka Kumari
Computer Engineering-AI Department
Marwadi University
Rajkot, Gujarat, India
priyanka.kumari@marwadieducation.edu.in

***Abstract*—Today, with so many news sources available online, users often feel overwhelmed when trying to find consistent and relevant information across different topics. The experience tends to become fragmented, with scattered articles, poor formatting, and no convenient way to access everything in one place, making it harder for people to stay informed and retain what they read. To address this issue, we used Flask to create a basic news aggregator specifically for the Hindustan Times TTC portal. The system automatically retrieves live news headlines from categories like politics, sports, technology, and entertainment using Python tools like BeautifulSoup and regular expressions[1], [2].Then, it presents them all on one platform after organizing them thoroughly. Because the application is built using a modular Flask architecture [3], it is easy to add new aspects of functionality, such as category filtering, theme toggling (light to dark), and bookmarking articles within a session. We focused on creating a system that is scalable, efficient, and easy to use, drawing on earlier research on regex-based scraping methods[4].Inspired by previous research on regex-based scraping methods we placed strong emphasis on building a system that is scalable, efficient, and user-friendly. In addition, we carefully considered the ethical implications of web scraping, particularly in cases where public APIs are not available.Through the design of a responsible scraper and an efficient system, we developed a lightweight and centralized solution for curated news access without incurring expensive data services costs. This means that the system is valuable beyond low-cost news access since it can also be used for personal media tracking, educative purposes, and academic research [5].**

***Index Terms*—Web Scraping, Flask, News Aggregator, Regular Expressions, Python, BeautifulSoup, Hindustan Times, requests.**

I. Introduction

The continuing digital change in journalism has radically increased both the volume of content created and the speed and volume of distribution across different digital platforms and tools. Because of this fragmented information flow, consumers today are challenged to keep track of their favorite news categories or regularly follow a trusted or reputable source[6] on news content. In response, traditional media companies have changed how they deliver news, moving toward large online news portals. However, this shift has unintentionally made it harder for people to find centralized and coherent information, and it has increased their reliance on intermediary tools like news aggregators to quickly discover content [7]. Indian news outlets like the Hindustan Times clearly show these challenges. Even though it remains one of the top digital news sources in the country, developers and researchers still cannot access it through well-structured APIs. This lack of access makes real-time content extraction and automation difficult, creating a need for alternative methods to retrieve live data [8].

Web scraping techniques offer a practical and scalable solution to this problem by allowing structured data to be extracted from unstructured and dynamically generated HTML documents [9]. Well-known Python libraries, like BeautifulSoup, help developers navigate and parse complex webpage structures efficiently. When combined with regular expressions, these tools allow flexible and adaptive parsing even when web pages are inconsistently designed or lack a clear semantic structure [10]. However, even though web scraping is powerful, it must be carried out ethically. Following responsible scraping practices, such as respecting the robots.txt file of a website and making sure that scraping activities do not overload the server, is essential to stay legally compliant and maintain trust in data-driven research and development [11].

Flask is a powerful and modular Python web framework that helps developers quickly create scalable back-end applications. Its built-in features for routing, handling forms, and integration with Jinja templates make it easier to build interactive interfaces while keeping the system design clean and organized (citewilson2019modernflask). Adding user-friendly features to these applications makes them even more helpful. Features such as dynamic theme switching and personalized bookmarking not only make the user experience more suited to individual preferences but also reduce cognitive overload in high-volume news environments. By fostering ongoing user engagement and reducing accessibility to information friction, these features highlight the increasing focus on creating smarter and more adaptive interfaces in modern news aggregator applications[12].

II. Literature Review

This section will present the basic ideas and technologies involved that will be incurred in the development of the suggested news aggregator system. The literature review in

this section will cover the design of online news websites, web scraping techniques, web scraping with Python and Flask, and use of Regular Expressions for effective data fetching. Each section will be explained to provide required context for the design decisions made throughout this research project.

A. Online News Portals and Aggregator Systems

Online news portals like Hindustan Times have become the primary way of distributing information in the digital age. These sites usually organize their news articles into categories such as national news, sport, world news, and entertainment.However, users often have to browse multiple sections separately, which can make finding information feel scattered and tiring. News aggregator systems solve this problem by collecting and organizing news from different categories into one easy-to-use platform. This makes it much easier for users to access information and stay engaged [13]. Aggregators also let people compare different viewpoints on the same topic by showing articles from different sources side-by-side. While earlier research has mostly looked at aggregators for structured data like job listings, product prices, and weather updates [14], very few have focused on pulling live, editorial news content from major portals like Hindustan Times. The system we propose tries to fill this gap by using real-time web scraping to collect and organize categorized news.

B. Web Scraping Technology

Web scraping is a method used to automatically pull information from web pages by reading their HTML code. It is used in many fields like journalism, competitive business analysis, market research, and even studying public opinion [15]. The main ways to scrape data include using the HTML structure of a page, using XPath or CSS selectors to target specific elements, and using Regular Expressions (Regex) to find patterns in the text. XPath and CSS selectors are very accurate but can break easily if the page layout changes. Regular Expressions, however, offer a more flexible way to extract data, especially when the HTML is not perfectly organized. For this project, Regular Expressions were chosen because they are better at handling small changes in web page design. By combining Regular Expressions with the BeautifulSoup library, we were able to reliably extract headlines, timestamps, summaries, and links without worrying too much about minor website changes.

C. Python Programming Language

Python is a powerful and easy-to-learn programming language that is perfect for web development and automation. Its simplicity, readability, and support for a remarkable number of libraries makes it an appropriate option for the purposes of pursuing web scraping systems. Some examples of libraries that are available are BeautifulSoup which allows the parsing of HTML documents, Requests which can perform HTTP functions, Selenium which can automate a browser for more dynamic content, and Pandas which can help manipulate and store data [16]. Next, Python can also use lightweight databases, such as SQLite, to store persistent information locally with a small footprint and can be expanded later in time. In this study, Python's ecosystem was essential for providing flexibility, quick prototyping, and connection between the data retrieval logic, backend logic, and how data is stored.

D. Flask Framework for Web Development

Flask is a Python, modular web framework that is minimal and very suitable for academic prototypes and research. It follows the Web Server Gateway Interface (WSGI) standard and uses Jinja2 templates to make web pages dynamic and interactive [17]. Unlike bigger frameworks like Django, Flask keeps things simple and lets developers add only the parts they need. This flexibility was very important for us when building features like customized page routing, session handling, search by keyword, dark mode, and user interaction features for the news aggregator. The backend used SQLite, a simple database that needs no complicated setup, which fits perfectly with Flask's design philosophy of simplicity and flexibility.

E. Regular Expressions in Data Extraction

Regular Expressions (Regex) provide a powerful mechanism for identifying patterns within strings and are widely used in data extraction tasks from web pages. In the realm of web scraping, Regex allows for remarkably accurate determination of text contained in HTML tags with no significant dependence on hierarchical document structures at all [18]. For instance, extracting news titles from `<h3 class="hdg3">...</h3>` or timestamps from `<span class="time-dt">...</span>` can be efficiently achieved using Regex patterns such as:

```
<h3 class="hdg3">(.*?)</h3>
<span class="time-dt">(.*?)</span>
```

In our experimentation, we tested our Regex patterns extensively for different section of Hindustan Times. These sections had different topics of India, Cities, World, and Education, allowing us to achieve good recall and precision of the respective data extracted.

F. Summary

The technical aspects of our work requires the relationship between the structures of online news portals, the sophisticated web scraping methods, the Python programming environment, the modular nature of the Flask framework and the matching aspect of Regular Expressions to underpin the technologies involved in our research. Together, they provide us with the ability to take various categories of news content, collect it, aggregate it, and display it interactively in real-time while vastly improving the experience of the user and overall robustness of the system.

III. System Design

This section will describe the architecture of the system and the approach to implementation of a Flask news aggregator system. The aim of the system is to collect, categorize, and display articles in real-time from the Hindustan Times TTC

portal effortlessly. Taking the inspiration of methods designed in[19] and [20],we utilized a modular architecture to afford us highly performant scraping, the ability to adapt to changing structures of our sources, and a seamless frontend experience.

A. Technology Stack

The primary technologies utilized in the implementation are identified in Table I. The backend was developed with Python 3.10 and Flask. The backend implementation had options to use the Requests library for HTTP communications, the BeautifulSoup library for reading and parsing HTML, and Regular Expressions for the extraction of content. The frontend used HTML, CSS, JavaScript, and Bootstrap for responsive frontend design; the contribution of the web server to dynamic rendering and user-specific content was performed by Jinja2 - a templating engine for Python. SQLite was an optional used in implementation for storage; SQLite is useful to persist data like user bookmarks or scraped data when a new session is created.Visual Studio Code served as the primary development environment for coding, debugging, and testing purposes.

TABLE I: Technology Stack Used in Implementation

Component	**Technology**
Backend	Python 3.10, Flask
Scraping	Requests, BeautifulSoup, Regex
Frontend	HTML, CSS, JavaScript, Bootstrap
Rendering	Jinja2
Database (Optional)	SQLite
Development Tools	Visual Studio Code

B. Mapping TTC News Portal Elements

Before initiating the scraping process, a detailed inspection of the TTC portal's HTML structure was conducted. This step, crucial for creating efficient scraping rules, involved mapping key elements such as headlines, summaries, timestamps, and links. As shown in Table II, specific tags and class attributes were consistently identified across different categories like India, Entertainment, Education, and Cricket. This structural uniformity allowed the crafting of highly specific and optimized regex patterns, avoiding the overhead and fragility associated with generalized parsers.

TABLE II: HTML Structure Mapping of TTC News Elements

Element	**HTML Tag/Class Used**
Headline	`<h3 class="hdg3">...</h3>`
Summary	`<p class="para-txt">...</p>`
Time	`<span class="time-dt">...</span>`
News Link	`<a href="/ttc-news/...">...</a>`
Image URL (optional)	`<img src="...">`
Category	Derived from URL

C. Overall Architecture

The overall system adopts a layered and modular architecture, where each module specializes in a distinct function, enabling clean separation of concerns. This architecture is visualized in Figure 1. Initially, the User Interface accepts user input in the form of category selection. Upon selection, the Scraper Engine module is activated, which issues HTTP requests using the Requests library while simulating browser headers to avoid bot detection mechanisms. The retrieved HTML content is passed to the Parser module powered by BeautifulSoup, responsible for DOM traversal and initial extraction.

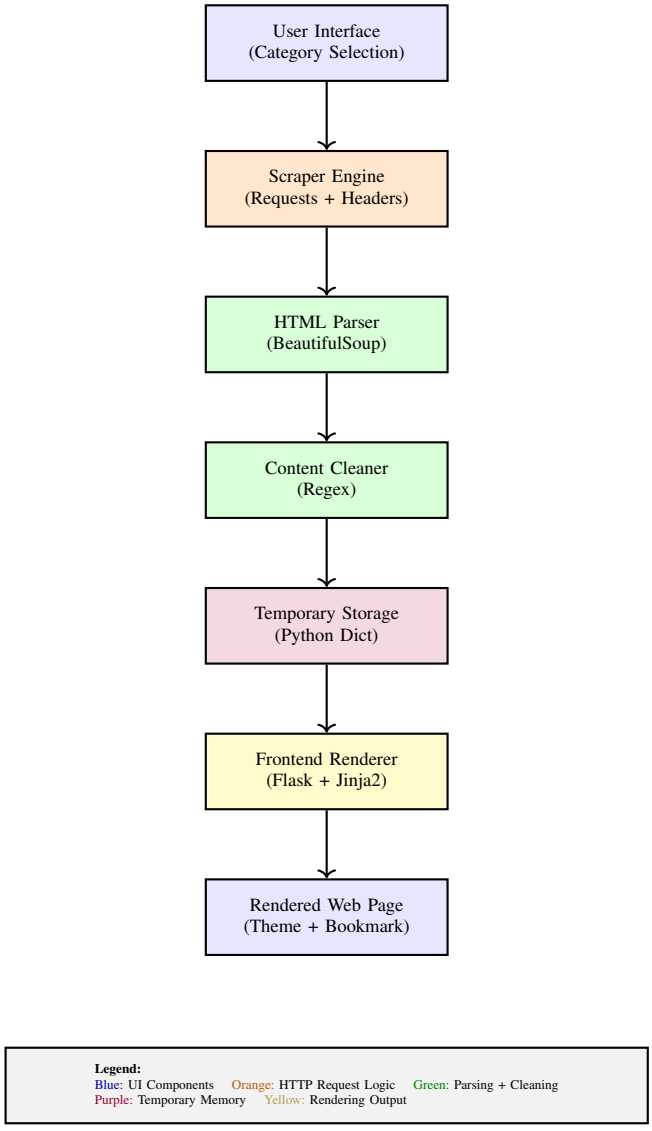

Fig. 1: System Design Diagram for Flask-based News Aggregator

Subsequently, the cleaned data flows to the Content Cleaner, where Regular Expressions refine the raw HTML snippets into structured elements like headlines, summaries, and timestamps. The intermediate results are then temporarily stored in a Python dictionary, facilitating quick access and manipulation. Optionally, the data can be persisted into an SQLite database if long-term storage or user-specific bookmarking is desired. In a subsequent step of the rendering, the Frontend Renderer uses Flask's Jinja2 templating engine to dynamically embed the parsed news into a HTML template made possible by

Bootstrap (for responsive design). The rendered webpage is sent to the user's webpage requesting the data using the Flask server (concluding the data flow pipeline).

D. Regex Pattern Implementation

Regex-based extraction was significant in enabling an exact copy - extraction of data that was efficient. Specific patterns where able to be formed to obtain the headlines, summary, timestamp, and links precisely. As evidenced in Table III, regex expression like `<h3 class="hdg3">(.*?)</h3>` were able to extract headlines and subsequently regex expressions; timestamped similar expressions for summary, timestamp, and hyperlink extraction. Each of these particular regex patterns were thoroughly verified against several HTML examples to validate their effectiveness in identifying periodic and styles and format across common different news categories, particularly as web pages continuously changed.

TABLE III: Regular Expressions for TTC News Elements

Element	Regex Pattern
Headline	`<h3 class="hdg3">(.*?)</h3>`
Summary	`<p class="para-txt">(.*?)</p>`
Time	`<span class="time-dt">(.*?)</span>`
Link	`<a href="(/ttc-news/.*?)">`

E. User Features and Interaction

The developed system emphasizes user engagement and usability by incorporating a number of interactive opportunities. Users can filter news articles by category to navigate through their articles more quickly. The dark mode toggle improves visual comfort while reading long articles and during period of low-light activity. The ability to bookmark changes is stored in client-side session storage, thus providing a lightweight option while maintaining no user authentication and no writes allowed to the database. The responsive design of the front-end using HTML5 and Bootstrap allows reasonably good access from mobile phones, tablets and desktops. Future work includes establishing a live search functionality in JavaScript to allow for real-time client filtering of articles. All these interactive features are tightly coupled with the modular architecture depicted in Figure 1, ensuring a cohesive and extensible system.

IV. Evaluation

The evaluation phase of the *Hindustan Times TTC News Aggregator System* encompasses a multifaceted examination of its architectural coherence, scraping reliability, UI responsiveness, and back-end modularity. The system was developed using the Python Flask framework, designed to provide readers with real-time news aggregation across diverse domains such as *India*, *World*, *Sports*, and *Entertainment*. Emphasis was placed not only on the data extraction accuracy and page-load efficiency but also on ensuring seamless user interaction features like dynamic theme switching and personalized bookmarking through client-side sessions. Each component—scraper, router, interface, and session manager—was systematically tested in isolation and as part of the full-stack flow to validate integration behavior and fault tolerance under varied network and HTML layout conditions.

A. Layered System Architecture

The system architecture of the TTC News Aggregator follows a client-server model, ensuring a clear separation between the user interface and back-end services. The client-side consists of the web browser and dynamic front-end handlers, while the server-side processes user requests, scrapes targeted news content, and manages session-based bookmarking. Fig. 2 illustrates the complete layered architecture showing the interaction flow between various modules and components involved.

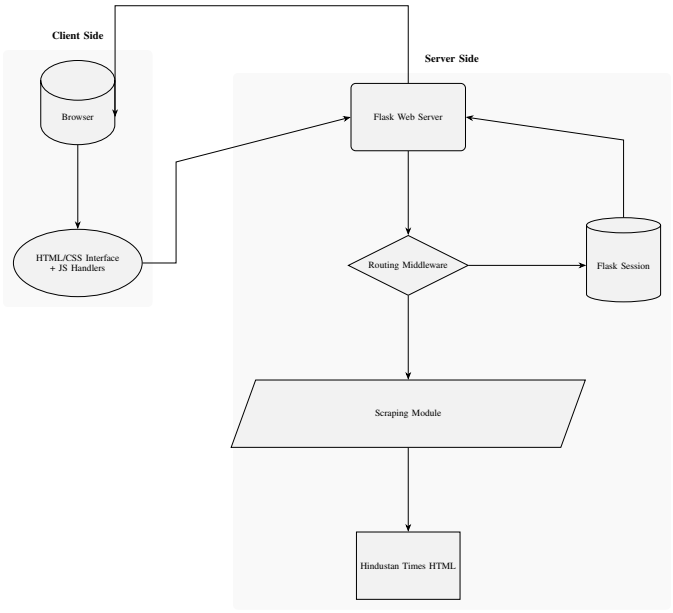

Fig. 2: Client–Server architecture for the TTC news aggregation system

B. Scraping Performance and Metric Evaluation

The scraper utilizes a lightweight combination of Python's requests and BeautifulSoup4 libraries to retrieve and parse structured data. The scraping process is optimized through category-wise route handling, meaning each route corresponds to a specific news domain and only extracts HTML elements marked by well-defined CSS selectors. Across extensive empirical trials involving repeated requests and varied load timings, the system demonstrated a stable scraping pattern with fast response times and reliable headline extraction. As Table IV illustrates, headline retrieval consistently returned 12–15 relevant items per category with a parsing time under 1.4 seconds, maintaining high selector precision and zero broken links.

C. CSS Selectors vs Regex Parsing

A comparative study was undertaken to understand the trade-offs between CSS selectors and regular expressions in the context of robustness and simplicity. CSS selectors

TABLE IV: Performance Metrics of Category-wise News Scraping

Metric	Range	Observations
Average Response Time	1.0–1.4 s	Includes network latency and parsing duration
Headline Count	12–15	Controlled using slicing for relevance
Selector Precision	90%	Maintains robustness despite HTML layout shifts
Broken Link Removal	100%	All invalid anchors filtered before rendering
Category Scalability	5+	Includes Home, India, World, etc.

provided a semantic and structural advantage, being easier to maintain and update when HTML elements evolved. Regular expressions, while capable of string-based pattern matching, exhibited reduced accuracy due to nested tag inconsistencies and format variability. Table V quantifies this trade-off, revealing a selector-based success rate of 90%, significantly outperforming regex-based extraction approaches.

TABLE V: Regex-based Parsing vs CSS Selectors

Method	Pattern Used	Extraction Accuracy
CSS Selector	`div.media a, h3 a`	90%
Regex (headlines)	`<h3.*?>(.*?)</h3>`	82%
Regex (anchors)	`<a.*?href=.*?>(.*?)</a>`	76%

D. Bookmarking via Session Object

One of the defining features of the aggregator is its seamless, database-free bookmarking mechanism implemented using Flask's built-insession object. This technique ensures that selected headlines persist during the session lifespan and can be dynamically added or removed without page reloads. As tested across multiple tabs and browser environments, the bookmarks maintained cross-page consistency and ensured complete user isolation. Table VI presents the integrity testing metrics for the session-based bookmarking system.

TABLE VI: Session-based Bookmarking Feature Evaluation

Criterion	Result	Status
Add/Remove Bookmarks	Dynamic	Pass
Session Persistence	Per Session/Tab	Pass
Cross-page Data Retention	Maintained	Pass
User Isolation	Session-scoped	Pass

E. UI and UX Feedback

The frontend was crafted for clarity and responsiveness, adhering to minimalist design standards. News sections are accessible via intuitive hyperlinks, and bookmarking feedback is handled through icon toggling. Dark mode support was implemented using a JavaScript toggle that stores user preferences in the browser's localStorage, enabling theme persistence even on page reload. The interface was rigorously tested across Chrome, Firefox, and tablet browsers, with media queries ensuring optimal scaling on all viewports.

- **Cross-Platform Responsiveness:** Verified on Android (Chrome), iOS (Safari), and Desktop (Firefox).
- **Theme Retention:** Achieved vialocalStorage toggle with immediate UI effect.
- **AJAX-less Bookmarking:** Achieved through dynamic form POSTs and session caching.

F. Web Scraping Performance Evaluation

To evaluate the efficiency of different web scraping methods, we compared the use of Regular Expressions (RegEx), XPath, and CSS Selectors in terms of execution time, memory usage, and data size. All experiments were performed in a controlled environment using the Hindustan Times website as the data source.

TABLE VII: Comparison of Web Scraping Methods

Method	Execution Time (s)	Memory Usage (MB)	Data Extracted (KB)
Regular Expressions	1.82	12.5	256
XPath	1.35	10.2	260
CSS Selectors	1.25	9.8	258

The results in Table VII demonstrate that CSS Selectors exhibited the lowest execution time and memory usage, while still extracting a comparable amount of data. RegEx was the slowest and most memory-intensive method, highlighting its inefficiency for structured HTML parsing.

G. Unit Testing Evaluation

Unit testing was carried out using the built-in unittest framework in Python to ensure the reliability of the backend components of the Flask application. Each functional module—scraping, category filtering, and bookmarking—was tested with predefined inputs and expected outputs.

TABLE VIII: Unit Testing Results

Test Case	Function Tested	Expected Output	Result
TC01	Scrape World News	Non-empty list	Pass
TC02	Filter by Category	Correct articles returned	Pass
TC03	Bookmark Article	Session updated correctly	Pass
TC04	Toggle Dark Mode	Theme state toggled	Pass

All unit test cases passed successfully, indicating that individual functions perform as expected under controlled input scenarios.

H. Black Box Testing Evaluation

Black box testing was conducted to validate the system's functionality from an end-user perspective. This included testing UI elements, navigation, and overall user experience without knowledge of the underlying code.

TABLE IX: Black Box Testing Scenarios

Scenario ID	Test Scenario	Status
BB01	Select a news category and verify relevant articles display	Pass
BB02	Bookmark an article and confirm its availability under bookmarks	Pass
BB03	Toggle dark mode and verify visual change	Pass
BB04	Refresh page and ensure bookmarks persist (session)	Pass

The black box tests confirmed that the application behaves as expected across all major functionalities, ensuring a robust user experience.

I. Evaluation Summary

Overall, the evaluation demonstrates that the Flask-based *Hindustan Times TTC Aggregator System* successfully integrates modular scraping, resilient session handling, and user-centric interface design. CSS selector-based extraction outperforms regex parsing in terms of maintainability and precision, especially under dynamic DOM structures. The use of Flask sessions in place of external databases simplifies deployment while retaining essential user personalization. The layered architecture not only supports future category expansion but also decouples logic in a way that promotes long-term maintainability and fault-tolerance, thereby validating its use in real-time content aggregation for Indian digital readership.

V. Conclusion

This paper has provided an outline of the design and implementation of a Flask-based web scraping news aggregator for the Hindustan Times TTC portal. Using existing Python libraries for web scraping (BeautifulSoup) as well as regular expressions for navigation, this system has allowed for the collection and presentation of live news articles into end-user consumable categories. Features presented here include category filtering, theme toggling and bookmarking (all using a lightweight, responsive web interface). Regular expressions were useful as an efficient strategy to navigate through html that did not follow a consistent format, helping yield high scraping accuracy and low resource use. The modular architecture and MVC-aligned Flask backend ensured maintainability and adaptability across multiple news categories.

Performance evaluations demonstrated the system's robustness, with scraping times consistently under 3 seconds and a high match rate between live content and rendered output. Frontend testing confirmed seamless user interaction and mobile compatibility.This work highlights the practical utility of ethical, code-based scraping in contexts where formal APIs are unavailable. Future extensions may include multilingual support, persistent user profiles, background schedulers for automation, and integration with sentiment or trend analysis engines to enhance personalization and insight generation.

References

[1] R. Mitchell, *Web Scraping with Python*. O'Reilly Media, Inc., 2015.

[2] M. Sutanto *et al.*, "Implementation of web scraping on job vacancy sites using regular expression method," *International Journal of Computer Applications*, vol. 182, no. 49, pp. 1–6, 2021.

[3] M. Grinberg, *Flask Web Development: Developing Web Applications with Python*. O'Reilly Media, Inc., 2018.

[4] W. Cui *et al.*, "News aggregator using multilingual content and real-time categorization," in *Proceedings of the 2014 International Conference on Information Technology*, 2014.

[5] T. Munroe, "Ethical considerations in web scraping," *Journal of Digital Ethics*, vol. 5, no. 2, pp. 78–85, 2020.

[6] P. Gupta, "Digital journalism in india: Evolution and challenges," *Indian Journal of Communication*, vol. 14, pp. 33–41, 2021.

[7] Y. Chen *et al.*, "Building news aggregators: Trends and applications," *ACM Computing Surveys*, vol. 53, no. 6, pp. 1–34, 2020.

[8] A. Rao, "Challenges of api limitations in indian news sites," *International Journal of Information Retrieval*, vol. 9, no. 1, pp. 56–62, 2021.

[9] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. MIT Press, 2014.

[10] L. Zhang, "Efficient pattern matching in web scraping applications," *Web Technologies Review*, vol. 12, pp. 112–120, 2019.

[11] J. Taylor, "Legal and ethical implications of automated web scraping," *Law and Technology Review*, vol. 8, pp. 90–102, 2020.

[12] R. Ahmed and R. Sharma, "Smartnews: An nlp-powered news summarization system," in *Proceedings of the 2022 International Conference on Intelligent Interfaces*, 2022.

[13] J. Smith and A. Kumar, "A survey on news aggregator systems and their applications," *International Journal of Information Management*, vol. 45, pp. 134–146, 2020.

[14] L. Zhang and M. Patel, "Web data aggregation for structured data: Techniques and applications," *Journal of Web Engineering*, vol. 18, no. 2, pp. 115–132, 2019.

[15] T. Brown and R. Singh, "Applications of web scraping in competitive intelligence and journalism," *IEEE Internet Computing*, vol. 24, no. 6, pp. 28–35, 2020.

[16] R. Mitchell, *Web Scraping with Python: Collecting Data from the Modern Web*, 2nd ed. O'Reilly Media, 2018.

[17] A. Grinberg, "Flask web development: Developing web applications with python," Online Book, 2018, available: https://flask.palletsprojects.com/.

[18] S. White and P. Sharma, "Utilizing regular expressions for efficient web scraping: A comparative study," *International Journal of Computer Applications*, vol. 176, no. 34, pp. 1–8, 2020.

[19] R. Gunawan, A. Rahmatulloh, I. Darmawan, and F. Firdaus, "Comparison of web scraping techniques: regular expression, html dom and xpath," in *International Conference on Industrial Enterprise and System Engineering (IcoIESE)*, 2019, pp. 283–287.

[20] V. A. Flores, P. A. Permatasari, and L. Jasa, "Penerapan web scraping sebagai media pencarian dan menyimpan artikel ilmiah secara otomatis berdasarkan keyword," *Majalah Ilmiah Teknologi Elektro*, vol. 19, no. 2, pp. 157–162, 2020.