International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 11 | Nov – 2025 DOI: 10.55041/ISJEM05191
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

Modular Web Scraping Pipeline for Systematic Multi-Site Data Extraction

Mrs. G Hema Prabha[1], Vijay A[2]

[1] Professor, Sri Shakthi Institute of Engineering and Technology

[2] Student, Sri Shakthi Institute of Engineering and Technology

Abstract

The rapid expansion of digital ecosystems and the increasing reliance on data-driven decision-making have created a critical need for efficient, scalable, and adaptable mechanisms to extract structured information from diverse online sources. Traditional web scraping solutions, while sufficient for single-site or small-scale tasks, often fail to meet the complexities associated with multi-site data extraction, evolving web architectures, and dynamic content rendering. To address these challenges, the Modular Web Scraping Pipeline presents a comprehensive and configurable framework engineered to facilitate systematic, multi-source data retrieval with high reliability and minimal manual intervention. Designed using Python and powered by modular components, the pipeline offers a structured methodology for collecting URLs, fetching raw HTML or API responses, parsing and normalizing content, managing storage, and integrating with analytical dashboards or downstream systems.

A key strength of the pipeline lies in its modular design philosophy, which decomposes the scraping process into six independent yet interconnected components: URL Collector, File Fetcher, Data Extractor, Automated File Cleanup, Database Management, and Dashboard Integration. This separation not only enhances maintainability but also enables site-specific customization without altering the overall workflow. The use of YAML configuration files allows extraction logic to be defined declaratively, making it possible to adapt quickly to new website structures or modifications in existing ones. This approach significantly reduces the burden of rewrites, increases pipeline flexibility, and ensures that the system remains resilient in the face of frequent web interface updates

Introduction

In the contemporary digital landscape, data has emerged as one of the most valuable organizational assets. Enterprises, researchers, and analysts consistently rely on large volumes of structured and unstructured data to derive insights, make strategic decisions, and build predictive systems. As online platforms continue to expand, the internet has evolved into a massive repository of information spanning e-commerce, social media, news outlets, product catalogs, academic portals, and numerous domain-specific platforms. This explosive growth has resulted in an increasing demand for efficient and systematic methods to extract relevant information from multiple web sources. Web scraping, therefore, has become a foundational technique for acquiring such data, enabling automated retrieval and transformation of online content into structured datasets.

Problem Statement

The rapid expansion of digital platforms has created unprecedented opportunities for organizations, researchers, and analysts to leverage web-based data for insights, forecasting, and strategic decision-making. E-commerce platforms, review portals, pricing aggregators, product catalogs, and content-driven websites collectively host billions of data points that change frequently and unpredictably. However, accessing this information in a structured and automated manner remains a significant challenge. Although web scraping is a widely adopted method for gathering such data, existing scraping solutions are often fragile, highly specific, and lacking in scalability. The absence of a standardized, modular, and adaptable framework creates several technical, operational, and analytical barriers for large-scale, multi-site data acquisition. This research addresses these gaps by identifying the core problems in current scraping practices and proposing a systematic solution through the Modular Web Scraping Pipeline.

A major issue in traditional web scraping lies in the **tight coupling between scraping logic and website structure**. Most scripts are built with hard-coded selectors, fixed tag identifiers, and rigid functions that cater only to one particular website. Such scripts typically work well for a single project but quickly break when websites are updated, redesigned, or modified. Modern websites frequently introduce dynamic elements, content loading through JavaScript, or non-standard HTML patterns. As a result, scraping scripts require constant maintenance and manual updates, making them unsustainable for long-term or multi-site operations. The lack of abstraction and modularity results in high maintenance costs and operational inefficiencies.

I. Existing Solution

Over the past decade, several web scraping tools, libraries, and platforms have been developed to extract data from online sources. These solutions attempt to simplify aspects of web scraping, automate repetitive tasks, and provide higher-level abstractions for data extraction. However, despite advancements, most existing solutions still fall short when applied to large-scale, multi-site, configurable, and continuously running scraping workflows. The following section reviews the widely used scraping approaches and highlights their limitations in the context of multi-source data extraction.

The most commonly adopted category of existing solutions includes traditional scripting libraries, such as BeautifulSoup, Requests, Selenium, and Scrapy. BeautifulSoup and Requests are widely used for HTML parsing and HTTP communication. While these libraries provide fine-grained control over extraction, they require developers to hard-code logic for each site. This results in scripts that are brittle, site-specific, and difficult to maintain. Selenium, on the other hand, allows scraping of dynamically rendered content but is computationally expensive and not suitable for large-scale or long-running scraping tasks due to browser overhead. Although powerful, these libraries serve as low-level tools rather than complete scraping frameworks, leaving developers responsible for building pagination logic, data normalization, file management, error handling, and workflow orchestration manually.

II. Literature Survey

Web scraping as a method for automated web data extraction has gained substantial attention in academic research and industry applications. Numerous studies have explored techniques, frameworks, tools, and challenges associated with scraping structured, semi-structured, and unstructured data from the web. This literature survey examines prior work in the fields of web data extraction, modular scraping architectures, workflow automation, error resilience, and configuration-driven data acquisition systems. The objective is to identify existing gaps and how modern research contextualizes the need for scalable, multi-site, and maintainable scraping pipelines.

Early academic studies on web scraping primarily focused on HTML parsing and DOM traversal. Gupta and Kaiser (2005) highlighted the challenges of extracting data from semi-structured web documents, emphasizing the limitations of regex-based scrapers and proposing wrapper-based extraction techniques. Their work demonstrated initial attempts to introduce template-driven extraction, yet these systems lacked robustness when dealing with frequently changing web layouts. As websites became more dynamic, static templates proved insufficient, laying the foundation for more adaptive scraping solutions.

In the following years, research shifted toward **wrapper induction systems**, such as WIEN and STALKER, which attempted to automate extraction rule generation. These systems relied on machine learning or pattern recognition to identify key content blocks within web pages. While effective in controlled environments, studies found that wrapper induction struggles with noisy, deeply nested HTML or JavaScript-generated content (Kushmerick, 1997). Furthermore, wrappers often failed when websites underwent even minor structural changes, requiring frequent retraining and manual intervention. These findings emphasized the need for more flexible architectures that combine rule-based systems with modularity and resilience.

III. Proposed Solution

To overcome the limitations identified in current web scraping practices, this research proposes a **Modular Web Scraping Pipeline**—a configurable, extensible, and automated framework designed to support large-scale, multi-site data extraction. Unlike traditional scraping scripts or rigid frameworks, the proposed solution is architected around modular components, declarative extraction rules, and workflow automation. It aims to provide a unified system capable of reliable, continuous, and maintainable scraping across diverse websites and data formats.

The architecture of the proposed pipeline is divided into independent but interconnected functional modules: **URL Collector, File Fetcher, Data Extractor, Automated File Cleanup System, Database Manager, and Dashboard Integration Layer**. Each module is designed with a clear responsibility, enabling ease of maintenance, reusability, and scalability. This modular approach ensures that any component can be modified, extended, or replaced without affecting the rest of the system.

IV. Executive Summary

The rapid growth of digital platforms has made web-based data a critical asset for businesses, researchers, and decision-makers. However, most existing scraping tools and ad-hoc scripts lack scalability, adaptability, and long-term maintainability—especially when dealing with multiple websites, dynamic layouts, and continuously changing online data. To address these challenges, the Modular Web Scraping Pipeline is proposed as a comprehensive, configuration-driven, and automated framework capable of performing systematic multi-site data extraction with high accuracy, efficiency, and resilience.

The system is built around a modular architecture that separates scraping tasks into independent functional stages: URL collection, file fetching, data extraction, automated file cleanup, database management, and dashboard integration. This decoupling ensures flexibility, enabling each module to be customized or upgraded without affecting the entire workflow. A key innovation of the solution is its YAML-based configuration layer, which allows extraction rules to be defined declaratively rather than through code. This makes it easy to adapt the pipeline to new websites or changes in existing site structures, significantly reducing maintenance overhead.

The pipeline incorporates robust error-handling mechanisms, retry logic, structured logging, and rate-limit–aware request handling to ensure reliability in real-world conditions. Extracted data undergoes normalization to maintain consistent formats across multiple websites, making it suitable for analytics, machine learning, and business intelligence applications. The system also includes automated storage in CSV files and a MySQL database, ensuring both portability and query-ready structured data.

V. Business Viability

The increasing digitization of commerce, services, and information systems has made data one of the most crucial strategic resources for modern organizations. Businesses today rely heavily on real-time insights to drive pricing strategies, monitor competitors, optimize operations, and understand consumer behavior. As a result, the demand for scalable, automated, and cost-efficient data extraction systems has grown significantly. The Modular Web Scraping Pipeline presents a strong business case due to its ability to address these needs through a flexible, maintainable, and high-performance architecture that can operate across multiple online platforms.

From a commercial perspective, the solution offers substantial cost advantages compared to manual data collection or reliance on expensive third-party scraping services. Traditional data-gathering methods often involve subscription fees, usage-based billing, or limited customization options. By deploying an in-house scraping pipeline, organizations can eliminate recurring costs, gain full control over the scraping logic, and avoid vendor lock-in. The system's modularity also reduces long-term operational expenses, as individual components can be upgraded or replaced without requiring a full system overhaul. This positions the pipeline as a highly economical solution for companies seeking sustainable, long-term data extraction capabilities.

VI. Advantages

The proposed Modular Web Scraping Pipeline provides a wide range of strategic, technical, and operational advantages that make it far superior to traditional web scraping approaches. It is built to be robust, scalable, and future-ready, capable of supporting multi-site data extraction in environments where websites frequently update their structures. One of the most important strengths of the system is its modular architecture, where every component such as URL collection, HTML fetching, data extraction, cleanup, database insertion, and dashboard integration functions independently. This ensures easy debugging, smoother upgrades, and high reusability across different projects, unlike monolithic scripts that are hard to maintain. The configuration-driven design of the pipeline, especially its use of YAML files for extraction rules, further reduces dependency on developers by allowing non-technical users to adjust scraping logic quickly whenever websites change. This improves adaptability, lowers maintenance costs, and supports large-scale multi-site operations.

The system also demonstrates strong scalability for scraping thousands of URLs and supports high-frequency data collection across multiple websites without needing major rewrites. Its robust error-handling capabilities—such as configurable retry logic, structured logs, and graceful handling of unexpected HTML changes—make the pipeline stable even in unpredictable conditions. Another major advantage is end-to-end automation, enabling the entire workflow from URL collection to dashboard publishing to run without manual intervention. This reduces human effort, improves consistency, and supports scheduled scraping for businesses that require hourly or daily updates.

Data normalization is another important feature, ensuring that information collected from different websites is transformed into a unified structure suitable for analytics, comparisons, and machine learning. The pipeline's direct integration with MySQL allows data to be stored in a structured format, enabling efficient querying, long-term analysis, duplicate detection, and seamless BI workflows. Automated file cleanup mechanisms prevent long-term storage issues by removing outdated temporary files, thereby keeping the system fast and healthy. Additionally, the Google Sheets integration ensures that the scraped data is instantly available on dashboards, allowing stakeholders to visualize insights in real time without manually uploading files.

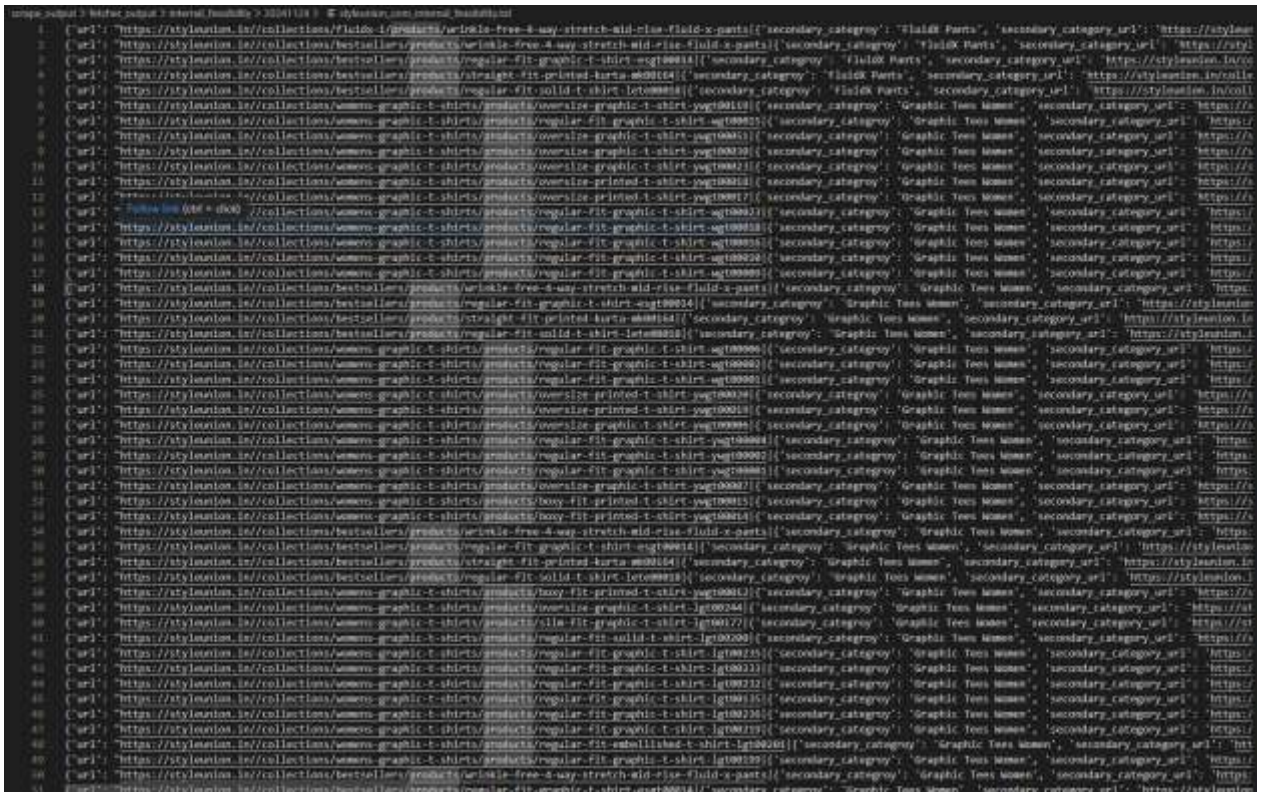
VII. Modules

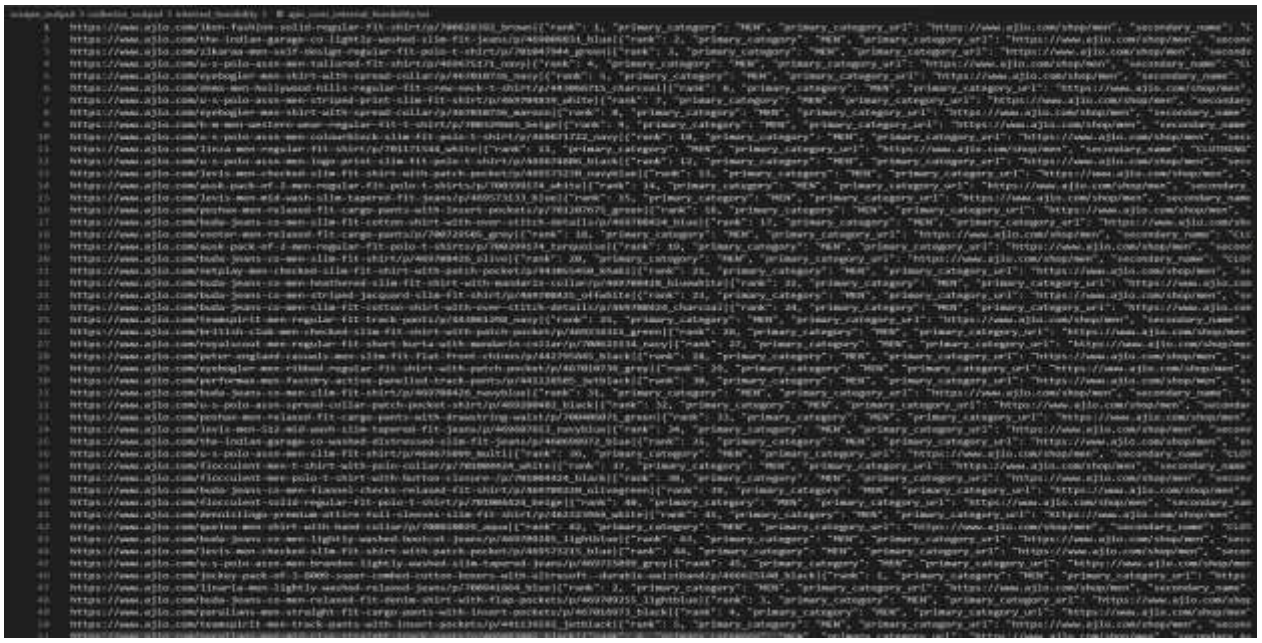
The Modular Web Scraping Pipeline is built using a set of well-structured and independent modules that work together to ensure efficient, scalable, and reliable multi-site data extraction. The workflow begins with the **URL Collector Module**, which identifies and retrieves all relevant product or content URLs from listing pages using pagination-aware logic and duplicate filtering. The collected URLs are passed to the **File Fetcher Module**, which handles HTTP requests, retry mechanisms, rate-limit awareness, and the storage of raw HTML files or API responses. Once the data is fetched, the **Data Extractor Module** processes these files using YAML-based extraction rules, allowing the system to adapt easily to different website structures. The extracted fields are standardized by the **Normalization Module**, ensuring consistent data formats across multiple sources. To support long-term storage and data accessibility, the **Data Storage Module** saves the processed information in CSV formats and inserts it into a MySQL database for structured querying and analytics.

Several support modules enhance reliability, maintainability, and automation within the pipeline. The **Automated File Cleanup Module** manages storage by deleting outdated temporary files, preventing clutter and optimizing performance. The **Error Handling and Logging Module** ensures operational stability by capturing errors, implementing retry logic, and recording detailed logs for debugging. For real-time visualization, the **Dashboard Integration Module** automatically uploads extracted data to Google Sheets, enabling instant access for analysis and decision-making. Finally, the **Scheduler and Automation Module**

allows the entire scraping workflow to run continuously or at scheduled intervals without manual intervention. Together, these modules form a comprehensive, flexible, and future-ready scraping system capable of handling diverse web sources with minimal maintenance.

VIII. **Output**





XII. CONCLUSION

The Modular Web Scraping Pipeline provides a scalable, flexible, and highly maintainable solution to the challenges associated with multi-site data extraction. By adopting a modular architecture and configuration-driven design, the system effectively overcomes the limitations of traditional scraping methods, which often struggle with

fragility, high maintenance, and poor adaptability. Each module—ranging from URL collection to dashboard integration—works independently yet cohesively, ensuring smooth data flow, robust error handling, and reliable automation. The inclusion of YAML-based extraction rules, automated cleanup routines, and real-time reporting mechanisms further enhances the system's ability to operate efficiently in dynamic web environments where site structures frequently change.

Overall, the proposed pipeline not only improves the accuracy and consistency of scraped data but also significantly reduces operational overhead, making it a highly viable solution for businesses, researchers, and data analysts. Its ability to integrate seamlessly with databases and visualization tools enables organizations to unlock valuable insights from diverse online sources while maintaining scalability for future expansion. The Modular Web Scraping Pipeline lays a strong foundation for advanced enhancements such as distributed scraping, ML-based extraction, and intelligent monitoring, positioning it as a robust, future-ready framework for systematic web data acquisition.

IX. Acknowledgment

We would like to express our profound appreciation to everyone who has helped to create and make the Modular Web Scraping Pipeline for Systematic Multi-Site Data Extraction a success. This project would never have happened without their commitment, knowledge, and help.

We also thank the administrators, patients, and medical experts who took part in the testing and offered insightful comments. Their feedback has been invaluable in helping to improve Modular Web Scraping Pipeline for Systematic Multi-Site Data Extraction and guarantee its applicability and efficacy in actual healthcare environments.

References

[1] Gupta, S., & Kaiser, G. (2005). Extracting content from semi-structured web pages. *Proceedings of the 13th International Conference on World Wide Web*.

[2] Kushmerick, N. (1997). Wrapper induction for information extraction. *Ph.D. Dissertation, University of Washington*.

[3] Lerman, K., Plangprasopchok, A., & Wong, C. (2010). A framework for attribute extraction from Amazon product pages. *Journal of Web Engineering*