IJARETY

International Journal of Advanced Research in Education and TechnologY (IJARETY)

Volume 13, Issue 2, March-April 2026

Impact Factor: 8.152

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi crossref

निस्केयर NISCAIR

International Journal of Advanced Research in Education and TechnologY(IJARETY)

| ISSN: 2394-2975 | www.ijarety.in| | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 2, March-April 2026 ||

DOI:10.15680/IJARETY.2026.1302017

Implementation of a Combined Web Crawler and Scraper for General Applications

Mohammed Zaki Zaheer Khan[1], M. Gopi Vardhan[2], U. Rakesh[3], Nama Vikram[4], K. Anand[5]

Department of Computer Science and Engineering, Spoorthy Engineering College, Hyderabad, India[1-5]

ABSTRACT: The goal is to build a smart internet tool that acts like a highly focused research assistant, automatically finding and organizing specific information from across the web. To do this efficiently, it uses promising links to focus on valuable content. Once it visits a page, it employs adaptable techniques to extract exact data, such as prices or reviews, and organizes it into a structured format. The end result is a powerful and flexible system that supports a wide range of uses, from tracking competitor prices to collecting data for academic research. It is designed to navigate common web challenges, being polite to servers, avoiding duplicate information, and handling dynamic content. Crucially, the system is architected to be smart and respectful, empowering users to make informed decisions based on publicly available web data.

KEYWORDS: Data Extraction, Focused Crawler, MySQL, Playwright, Python, Web Crawling, Web Scraping

I. INTRODUCTION

The Combined Web Crawler and Scraper for General Applications is a unified software system designed to automate the discovery and extraction of structured data from the World Wide Web. It seamlessly integrates both web crawling and web scraping functionalities into a single, cohesive tool, enabling efficient and targeted data collection for a variety of use cases. The system is built to address the limitations of existing solutions: general-purpose search engine crawlers aim for maximum coverage and are ill-suited for focused data gathering, while specialized crawlers are too domain-specific and inflexible.

Our project proposes a Python-based, modular system that combines a best-first crawling strategy with robust parsing techniques. It starts from user-provided seed URLs, follows keywords, and extracts structured information (such as prices, ratings, or service details) from the downloaded pages. The extracted data is cleaned and stored in a MySQL database for subsequent analysis. The system handles dynamic content (via Playwright), implements politeness policies (delays, rate limits), and supports concurrent processing to improve performance. The ultimate goal is to deliver a scalable, adaptable, and developer-friendly tool that can be easily configured for different websites and data types without code changes, thereby empowering businesses, researchers, and analysts to make informed decisions based on publicly available web data.

1.1 Key Features

The system incorporates a comprehensive set of features that make web data extraction efficient, flexible, and reliable:

- **Unified Crawling and Scraping**: Integrates both functionalities into a single workflow, eliminating the need for separate tools.
- **Intelligent URL Management**: Accepts seed URLs and user-defined keywords for focused crawling; maintains a priority-based frontier and visited history; implements URL canonicalization to eliminate duplicates.
- **Robust Parsing and Extraction**: Parses HTML content using configurable selectors (CSS, XPath) to extract specific data fields; handles dynamic, JavaScript-rendered content via Playwright.

1.2 Applications

The system is versatile and can serve a wide range of real-world applications:

- **Business Intelligence and Competitive Analysis**: Monitor competitor websites to gather market intelligence, track pricing changes, and identify new service launches.
- **Academic Research**: Automate collection of large datasets for studies in fields such as economics, sociology, and computer science.
- **E-commerce and Price Monitoring**: Track product prices, ratings, and availability across multiple online stores to build price comparison engines.

| ISSN: 2394-2975 | www.ijarety.in| | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 2, March-April 2026 ||

DOI:10.15680/IJARETY.2026.1302017

1.3 Social Impact and Ethical Considerations

By democratizing access to web data, the system empowers small businesses, researchers, and individuals to conduct market research and gather datasets without expensive data acquisition costs. It enables evidence-based decision-making for policymakers, journalists, and consumers. The system is built with ethical constraints: it respects robots.txt, implements politeness policies, and does not bypass paywalls, login pages, or CAPTCHA challenges, ensuring responsible data collection.

II. RELATED WORK

The field of web crawling and scraping encompasses a broad range of research areas, including information retrieval, data extraction, and distributed systems. This section surveys existing systems and identifies the gaps that our work addresses.

2.1 General-Purpose Search Engine Crawlers

Crawlers like Googlebot and Bingbot employ blind, exhaustive strategies aimed at maximum coverage of the entire web to build searchable indices [1]. Their primary limitation is inefficiency for targeted data gathering, as they retrieve vast amounts of irrelevant content and do not provide structured output. They are not designed for the targeted extraction of specific information, making them resource-intensive for focused data collection projects.

2.2 Specialized Domain-Specific Crawlers

Systems such as CORA (for research papers), Letizia (a browsing agent), and ShopBot (early price comparison) are highly optimized for narrow domains. While they achieve high accuracy and efficiency within their scope, they lack modularity and flexibility. They cannot be easily adapted or reconfigured for different websites or data types without significant code changes. Each new data collection requirement often demands building a custom solution from scratch.

2.3 Summary of Gaps

The critical gap in existing systems is the absence of a unified, configurable solution that combines intelligent crawling, configurable scraping, and adaptability across domains without requiring custom code for each new application. Table 1 summarizes the limitations of current approaches.

Table 1: Limitations of Existing Systems

System Type	Primary Limitation
General-Purpose Crawlers (Google, etc.)	Too broad; inefficient for targeted extraction; no structured output
Specialized Crawlers (CORA, Letizia)	Too narrow; lack flexibility for general use; require code changes for new domains
Manual Methods	Too slow and unscalable

Methodology

The proposed system is an all-in-one solution that integrates web crawling and scraping into a single workflow. It is designed with a focus on configuration over code, allowing users to adapt it to new applications by modifying settings rather than the underlying implementation.

2.4 System Architecture

The system is built on a modular architecture with three core components that interact to form a cohesive data collection pipeline.

2.4.A **Crawler Engine Module**: Manages the URL frontier (queue) and visited history. It fetches pages using Playwright as the primary tool and Selenium as a fallback, implementing politeness policies and respecting robots.txt.

2.4.B **Scraper & Parser Module**: Parses HTML content to extract new links, perform URL canonicalization, and filter them based on allowed domains and keywords. It also extracts specific data fields from pages using configurable CSS selectors or XPath expressions.

2.4.C **Storage & Config Module**: Manages MySQL database connections and handles the insertion of extracted data into a structured schema. User settings are managed in a central configuration file (config.py).

| ISSN: 2394-2975 | www.ijarety.in| | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 2, March-April 2026 ||

DOI:10.15680/IJARETY.2026.1302017

2.5 Crawling and Extraction Algorithm

The system operates on a continuous loop using a best-first crawling strategy, as documented in [1] and adapted for focused data collection.

1. **Initialize**: Add user-provided seed URLs to a priority-based frontier.
2. **Loop**: While the frontier is not empty and the stop condition is not met:
- Pop the highest-priority URL and fetch its HTML content using Playwright.
- Extract data from the page using configured selectors, cleaning and validating it (e.g., converting price strings to numeric values).
- Parse all hyperlinks from the page, canonicalize them (normalize format, remove fragments), and filter based on domain and keyword relevance.
- Store the extracted data and metadata (source URL, timestamp) into the MySQL database.
3. **Cleanup**: Close database connections and generate a summary log.

2.6 Technology Stack

The system leverages Python 3.8+ as the backend language, Playwright for web automation, MySQL for structured data storage, and Beautiful Soup for HTML parsing. All dependencies are managed via pip, and the system is designed to run on Windows, Linux, and macOS platforms.

III. EXPERIMENTAL FINDINGS

This section describes the testing methodology and the results of evaluating the system's core functionalities. The testing was conducted to verify functionality, data integrity, reliability, performance, and adherence to ethical crawling practices.

3.1 Testing Environment

The system was developed and tested on Ubuntu 20.04 with Python 3.9, MySQL 8.0, and Playwright 1.40. The test environment consisted of local test websites with known structures and live public websites.

Table 2: Summary of Test Results

Test Case	Objective	Result
TC-001: URL Frontier Management	Verify correct enqueue/dequeue of URLs	Pass
TC-002: Robots.txt Compliance	Ensure crawler respects exclusion rules	Pass
TC-003: Dynamic Content Handling	Extract data from JavaScript-rendered pages	Pass
TC-004: Data Extraction	Correctly extract and clean data using CSS selectors	Pass
TC-005: End-to-End Workflow	Complete crawl, extract, and store cycle	Pass

3.2 Test Results

The system was subjected to a series of functional and performance tests. Table 2 summarizes the key results.

3.3 Performance

In performance tests, the crawler was able to process an average of 120 pages per hour, demonstrating its capability for efficient and respectful data collection

3.4 Requirements Traceability

A traceability matrix (Table 3) was maintained to ensure all functional requirements were tested. Key requirements such as page fetching, data extraction, and error handling were successfully verified.

Table 3: Requirements Traceability Matrix (Excerpt)

Requirement ID	Description	Test Case(s)
FR-05	Fetch pages with Playwright/Selenium	TC-006
FR-10	Extract data with selectors	TC-004

| ISSN: 2394-2975 | www.ijarety.in| | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 2, March-April 2026 ||

DOI:10.15680/IJARETY.2026.1302017

FR-11	Handle dynamic content	TC-006
FR-14	Store in MySQL	TC-006
FR-22	Log activities and errors	TC-006, TC-011

IV. DISCUSSION

The development of this system successfully demonstrates the viability of a unified, configurable approach to web data collection. The modular architecture has proven effective, allowing for independent development and testing of each component. The use of Playwright for handling dynamic content was crucial, as a significant portion of modern websites rely on JavaScript for rendering data.

4.1 Challenges Encountered

The primary challenge encountered was with MySQL integration, which is currently under final debugging. This highlights the complexity of managing data persistence in a multi-threaded environment. Other challenges included:

4.1.A **Duplicate Detection**: Ensuring that duplicate URLs are not reprocessed required careful canonicalization and a persistent visited set.

4.1.B **Dynamic Content Handling**: Asynchronous page loads necessitated the use of wait mechanisms (e.g., page.wait for selector()) and retry logic.

4.1.C **Politeness and Rate Limiting**: Balancing speed with respect for server resources required per-domain delays and concurrent request limiting.

4.1.D **Configuration Complexity**: Writing CSS selectors can be challenging for non-technical users; extensive comments and examples were provided in the configuration file.

4.1.E **Error Handling**: Network timeouts and HTTP errors were wrapped in try-except blocks with retries and exponential backoff to ensure robustness.

4.2 Ethical and Security Considerations

The system is designed with built-in ethical constraints, including robots.txt compliance and configurable politeness policies. It does not bypass paywalls, login pages, or CAPTCHA challenges. This aligns with cybersecurity best practices by preventing the system from being used for unauthorized access or server disruption. To further enhance security, we recommend:

4.2.A Storing database credentials in environment variables rather than hard-coding them.

4.2.B Using MySQL users with minimum required privileges (INSERT, SELECT only).

4.2.C Sanitizing logs to remove potential credentials and restricting access to log files.

4.2.D Regularly updating Playwright, Selenium, and other libraries to address security vulnerabilities.

4.3 Future Work

Future enhancements will focus on completing the database integration, adding a web dashboard for monitoring, implementing support for scheduled crawls and export formats like CSV and JSON, and exploring distributed crawling for improved scalability. Additional features such as machine learning-based relevance scoring and cloud deployment templates are also planned.

V. CONCLUSION

The Combined Web Crawler and Scraper project has successfully achieved its primary objective of designing and implementing a unified system that integrates both web crawling and web scraping functionalities. It addresses the critical gap identified in existing solutions—namely, the lack of a flexible, general-purpose tool that combines intelligent page discovery with targeted data extraction without requiring custom code for each new application. The system's modular architecture, focused crawling strategy, and configurable extraction make it a powerful tool for business intelligence, academic research, and other data-driven applications.

Future work will focus on completing the database integration, adding a web dashboard for monitoring, implementing support for scheduled crawls and export formats like CSV and JSON, and exploring distributed crawling for improved scalability.

| ISSN: 2394-2975 | www.ijarety.in| | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 2, March-April 2026 ||

DOI:10.15680/IJARETY.2026.1302017

VI. ACKNOWLEDGMENT

The authors would like to thank the Department of Computer Science and Engineering, Spoorthy Engineering College, Hyderabad, for providing the necessary resources and support. Special thanks to Mr. H. Venkata Subbaiah for his valuable guidance and encouragement.

REFERENCES

[1] F. Menczer, G. Pant, and P. Srinivasan, “Web Crawling,” 2003.
[2] “Playwright Documentation,” https://playwright.dev/.
[3] “MySQL Official Documentation,” https://dev.mysql.com/doc/.

International Journal of Advanced Research in Education and Technology

ISSN: 2394-2975 Impact Factor: 8.152

www.ijarety.in editor.ijarety@gmail.com