Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Advanced Web Scraping Tool: for Static and Dynamic Content Extraction

Mr. Tahir Khan[1], Mr. Rizwan Ansari[2], Mr. Aman Shaikh[3], Mr. Aditya Pisal[4], Prof. Amit Chakrawarti[5]
[1,2,3,4]Department of Artificial Intelligence & Machine Learning Dilkap
Research Institute of Engineering & Management Studies Mamdapur,
Neral, Karjat, Raigad – 410101, Maharashtra, India University of
Mumbai | Academic Year 2025–26

***Abstract*- Web scraping has become an essential technique for extracting structured information from web sources. However, many existing tools are limited to either static or dynamic content, reducing their effectiveness in modern web environments. This paper presents the design and development of an advanced web scraping system capable of handling both static and dynamic web pages within a unified interface.**

The proposed system integrates multiple technologies to provide efficient data extraction. Static content is processed using lightweight parsing techniques, while dynamic content is handled through automated browser interaction. The system allows users to define custom extraction parameters, manage multiple web sources, and store extracted data in various formats including structured and document-based outputs.

A graphical user interface is implemented to improve usability and accessibility, enabling users with minimal technical expertise to perform complex scraping tasks. The modular architecture ensures scalability, maintainability, and flexibility for future enhancements.

Experimental usage demonstrates that the system effectively extracts diverse data types such as text, links, images, and tabular data from different categories of websites. The proposed solution provides a practical and efficient approach to modern web data extraction challenges.

***Keywords*- Web Scraping; Static Content; Dynamic Content; Data Extraction; Automation; Graphical User Interface; Modular System**

I. INTRODUCTION

The rapid growth of the internet has led to an exponential increase in the availability of data across various web platforms. Extracting this data efficiently has become essential for applications such as data analysis, market research, content aggregation, and decision-making systems. Web scraping serves as a fundamental technique for collecting structured information from web pages in an automated manner.

Traditional web scraping approaches primarily focus on static web pages, where data is directly available in the page source. However, modern websites increasingly rely on dynamic content generated through client-side technologies, making data extraction more complex. Many existing tools either lack support for dynamic content or require advanced technical expertise, creating a gap between functionality and usability.

In addition, users often face challenges in managing multiple data sources, customizing extraction requirements, and storing data in usable formats. The absence of an integrated system that combines flexibility, ease of use, and support for both static and dynamic content limits the effectiveness of current solutions.

To address these challenges, this paper presents the design and development of an advanced web scraping system that supports both static and dynamic web content within a unified framework. The system provides a user-friendly graphical interface that allows users to manage URLs, define extraction parameters, and perform scraping operations without requiring deep technical knowledge.

The proposed solution adopts a modular architecture, separating core functionalities such as URL management, static scraping, dynamic scraping, and data storage. This design improves maintainability, scalability, and adaptability to different web structures. Furthermore, the system enables users to extract various types of data including text, links, images, tables, and metadata, and store the results in multiple formats for further analysis.

The remainder of this paper discusses the system design, implementation details, and performance evaluation, demonstrating the effectiveness of the proposed approach in handling real-world web scraping scenarios.

II. SYSTEM ARCHITECTURE

The proposed Advanced Web Scraping Tool is designed using a modular and layered architecture to ensure flexibility, scalability, and ease of maintenance. The system integrates multiple components that work together to perform efficient data extraction from both static and dynamic web sources. Each module is responsible for a specific functionality, enabling clear separation of concerns and improved system organization.

The overall architecture consists of five primary components: User Interface Module, URL Management Module, Static Scraping Module, Dynamic Scraping Module, and Data Storage Module. These components interact sequentially to execute the complete scraping workflow.

The User Interface Module provides an interactive graphical environment through which users can access all system functionalities. It enables users to manage input URLs, select scraping options, define extraction parameters, and initiate scraping processes. The interface is designed to be user-friendly, allowing even non-technical users to perform complex operations efficiently. Additionally, threading mechanisms are incorporated to ensure that the interface remains responsive during long-running scraping tasks.

The URL Management Module handles the addition, storage, and retrieval of multiple web addresses. It maintains a structured list of URLs, allowing users to save and load them for repeated use. This module improves usability by eliminating the need to manually re-enter frequently used web sources.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

The Static Scraping Module is responsible for extracting data from web pages with fixed content. It processes the page source and retrieves elements such as headings, paragraphs, links, images, tables, and metadata based on user-selected options. Efficient parsing techniques are employed to ensure fast and reliable extraction of structured data.

The Dynamic Scraping Module extends the system's capability to handle modern web applications that rely on client-side rendering. It utilizes automated browser control to load web pages, execute scripts, and retrieve dynamically generated content. The module supports pagination and allows users to

define custom extraction fields, including element tags, attributes, and extraction types. This flexibility enables precise data collection from complex web structures.

The Data Storage Module manages the organization and export of extracted data. It supports multiple output formats, including spreadsheet, structured text, and document-based formats, ensuring compatibility with various data analysis tools. The module intelligently handles different data types such as tabular data, lists, and key-value metadata, providing structured and readable outputs.

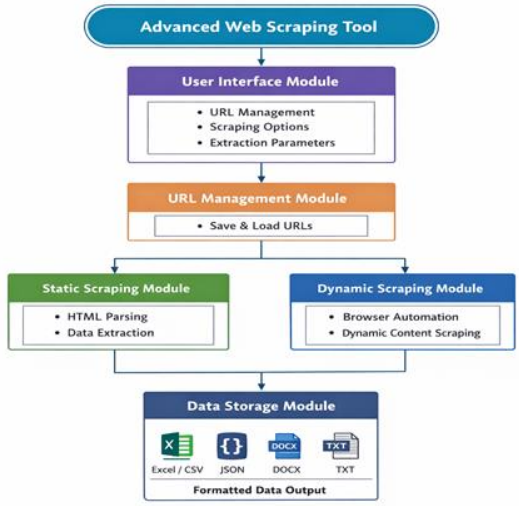

Fig.1 System Architecture of proposed Web Scraping Tool

The system workflow begins with user input through the interface, followed by URL selection and configuration of extraction parameters. Depending on the type of web content, the appropriate scraping module is invoked. The extracted data is then processed and forwarded to the storage module for saving in the desired format.

The modular design of the system allows independent enhancement of each component without affecting the overall functionality. This architecture not only improves system reliability but also provides a strong foundation for future extensions such as automated scheduling, advanced filtering, and integration with data analysis pipelines.

The overall system architecture is illustrated in fig 1.

III. WORKING FLOW / FLOWCHART

A. Working Flow of the System

The working flow of the Advanced Web Scraping Tool is designed to provide a structured and user-friendly process for extracting data from web sources. The system follows a step-by-step execution model, beginning from the main interface and proceeding through different functional modules based on user selection. The complete workflow ensures efficient handling of both static and dynamic web content while maintaining flexibility and ease of use.

B. Main Menu (Entry Point)

The system execution begins with the main menu interface, which serves as the central control panel. Upon launching the application, users are presented with multiple options including URL Management, Static Scraping, Dynamic Scraping, and View Saved Files.

The user selects the desired operation, and the system redirects to the corresponding module. This modular navigation ensures clarity and ease of interaction, allowing users to perform specific tasks without unnecessary complexity.

C. URL Management Workflow

When the user selects the URL Management option, the system provides functionality to add, store, and retrieve web addresses. The user can input one or more URLs, which are temporarily stored in memory and can be permanently saved into a file.

The system also allows previously saved URLs to be loaded, enabling reuse of frequently accessed web sources. This reduces redundancy and improves workflow efficiency. Once the URLs are managed, the user can return to the main menu for further operations.

D. Static Scraper Workflow

In the Static Scraper module, the system processes web pages with fixed content. The workflow begins with the selection of a target URL from the stored list. The user then selects specific data extraction options such as headings, paragraphs, links, images, tables, and metadata.

After initiating the scraping process, the system sends a request to the target web page and retrieves its HTML content. The retrieved content is parsed using structured parsing techniques to extract the selected elements.

If the extraction process is successful, the data is organized into a structured format and passed to the storage module. In case of errors, appropriate messages are displayed to inform the user. The system then allows the user to return to the main menu.

E. Dynamic Scraper Workflow

The Dynamic Scraper module is designed to handle web pages that rely on client-side rendering. The workflow begins with user input of a target URL, including support for pagination through dynamic placeholders. The user specifies the number of pages to be processed and defines extraction parameters such as HTML tags, attributes, and data types.

The system then launches an automated browser instance to load the web page. It waits for dynamic content to be fully rendered before extracting the page source. The content is parsed, and user-defined fields are extracted accordingly.

The module supports iterative processing across multiple pages, enabling large-scale data extraction. Extracted data is validated and organized before being forwarded to the storage module. If no data is found or an error occurs, the system notifies the user accordingly. After completion, control is returned to the main menu.

F. View Saved File Workflow

The View File module allows users to access previously saved data. The system prompts the user to select a file from the local storage. Once selected, the file is opened using the default application associated with its format.

This feature provides quick access to extracted data without

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

requiring manual navigation, improving overall usability and efficiency.

G. Data Storage and Output

Regardless of the scraping method used, all extracted data is processed by the storage module. The system provides multiple output formats, including spreadsheet, structured text, and document-based formats.

The data is formatted based on its type, ensuring readability and compatibility with external tools. Once saved, the file is automatically opened for user verification.

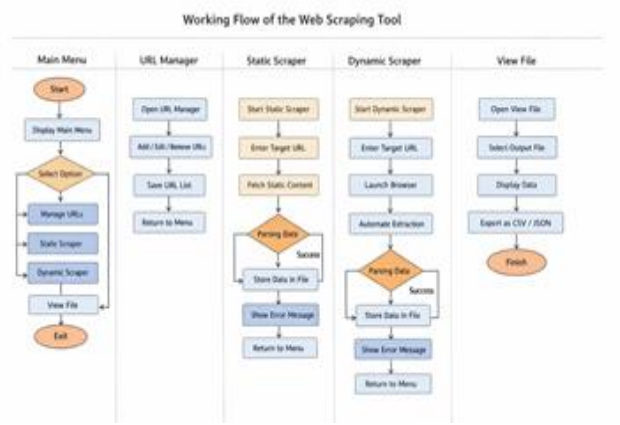

Fig.2 Working Flow of the Proposed Tool

IV. USER INTERFACE SCREENS

The Advanced Web Scraping Tool is equipped with a graphical user interface designed to provide a seamless and user-friendly interaction experience. The interface is structured into multiple screens, each dedicated to a specific functionality. This modular design improves usability and allows users to perform complex operations through simple interactions. The following sections describe each interface screen in detail.

A. Main Dashboard Screen

The Main Dashboard Screen serves as the entry point of the application. It provides access to all major functionalities through clearly defined options.

The interface includes navigation controls for URL management, static scraping, dynamic scraping, and file viewing. Each option is represented as an interactive button, allowing users to quickly navigate to the desired module.

The dashboard is designed with a clean layout and intuitive structure, ensuring ease of use for both technical and non-technical users.

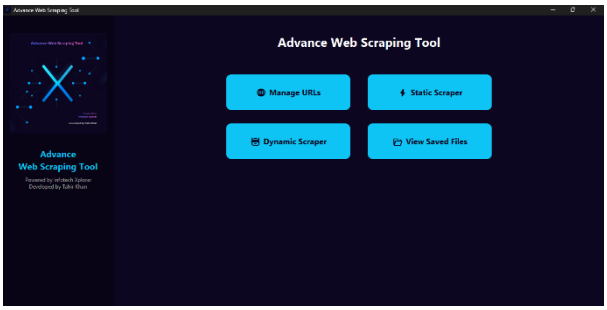

Fig.3 Main Dashboard Screen (Main Menu UI)

B. URL Management Screen

The URL Management Screen allows users to add, store, and retrieve web addresses for scraping operations.

Users can input URLs into a text field and add them to a managed list. The interface also provides options to save the list to a file and load previously stored URLs.

This screen simplifies the process of handling multiple web sources and eliminates the need for repeated manual input.

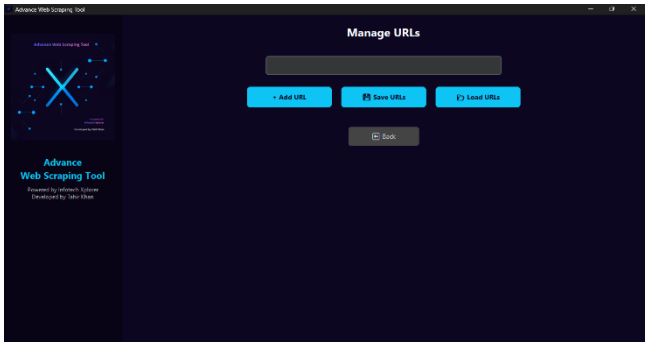

Fig.4 URL Manager Screen

C. Static Scraper Configuration Screen

The Static Scraper Configuration Screen enables users to define data extraction options for static web pages.

The interface provides a dropdown menu for selecting target URLs and a set of selectable options for extracting specific elements such as headings, paragraphs, links, images, tables, and metadata.

A dedicated action button is provided to initiate the scraping process, along with a status display area to inform users about the progress and results of the operation.

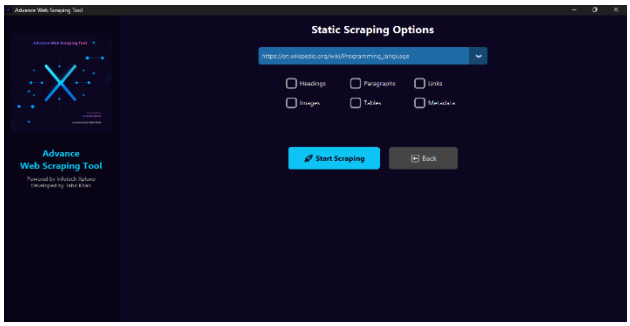

Fig.5 Static *Scraper* UI

D. Dynamic Scraper Configuration Screen

The Dynamic Scraper Configuration Screen is designed for handling web pages with dynamic content.

This interface allows users to input a target URL with pagination support, specify the number of pages to scrape, and define detailed extraction parameters. Users can configure container elements and dynamically add multiple fields by specifying HTML tags, attributes, and extraction types.

The screen also includes controls for initiating the scraping process and monitoring its progress, providing flexibility and precision in data extraction.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

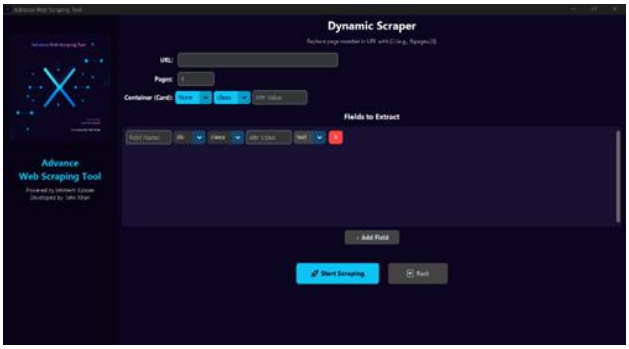

Fig.6 Dynamic Scraper UI

E. File Viewer Screen

The File Viewer Screen provides functionality to access and view previously saved data files.

Users can select a file from the local system, which is then opened using the default associated application. This feature allows quick verification and analysis of extracted data without requiring additional steps.

The interface is kept minimal to ensure straightforward interaction and fast access.

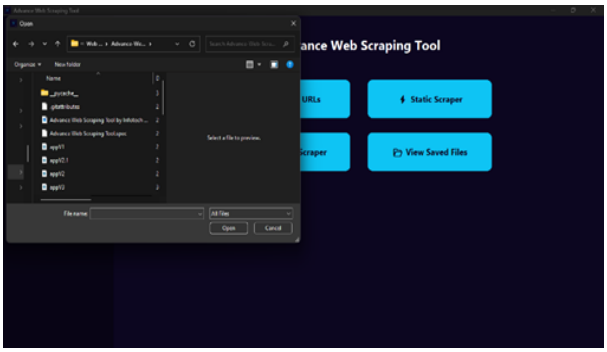

Fig.7 File Viewer UI

V. IMPLEMENTATION

The Advanced Web Scraping Tool is implemented using a modular approach, integrating multiple technologies to handle different aspects of the system efficiently. The application is developed in Python, leveraging its extensive ecosystem of libraries for web scraping, data processing, and graphical interface design.

The graphical user interface is built using a modern GUI framework, providing an interactive and responsive environment for user operations. The system incorporates event-driven programming and multithreading to ensure smooth execution of long-running scraping tasks without affecting user interaction.

The static scraping functionality is implemented using HTTP request handling and HTML parsing techniques. Web pages are fetched through network requests and processed to extract structured data elements such as text, links, images, tables, and metadata. Efficient parsing ensures quick data retrieval with minimal resource consumption.

For dynamic content extraction, the system utilizes browser automation to load and interact with web pages that rely on client-side rendering. The automated browser executes scripts, waits for content to load, and retrieves the fully rendered page source. This approach enables accurate extraction from modern web applications.

The system allows user-defined extraction by specifying element tags, attributes, and data types. This flexibility is achieved through configurable input parameters, enabling precise targeting of required data fields across different websites.

Data handling and storage are implemented using structured data processing techniques. Extracted data is organized into appropriate formats and exported into multiple file types, ensuring compatibility with various analysis tools.

Overall, the implementation focuses on modularity, usability, and efficiency, ensuring that each component operates independently while contributing to the complete system workflow.

VI. CONCLUSION

This paper presented the design and development of an Advanced Web Scraping Tool capable of extracting data from both static and dynamic web sources within a unified system. The proposed solution addresses the limitations of traditional scraping tools by integrating multiple functionalities into a single, user-friendly interface.

The system demonstrates effective handling of diverse data types, including text, links, images, tables, and metadata,

while providing flexible configuration options for customized data extraction. The incorporation of dynamic scraping techniques enables the tool to process modern web applications that rely on client-side rendering.

The modular architecture ensures scalability,

maintainability, and ease of enhancement, making the system suitable for real-world applications. Additionally, the availability of multiple output formats improves the usability of extracted data for further analysis and processing.

Overall, the proposed tool provides a practical and efficient solution for web data extraction, bridging the gap between functionality and usability. Future enhancements may include automated scheduling, advanced filtering mechanisms, and integration with data analytics platforms to further extend the system's capabilities.

VII. ACKNOWLEDGMENT

We would like to express my sincere gratitude to my respected professors, **[Prof. Akshata Laddha (Project Guide)]** and **[Prof. Amit Chakrawarti (Head of Department, AI & ML)]**, for their continuous guidance, valuable suggestions, and encouragement throughout the development of this project and research paper. Their support and insights played a crucial role in shaping the direction and successful completion of this work.

I am also thankful to my institution for providing the necessary resources and environment to carry out this project effectively.

Additionally, I would like to extend my appreciation to all those who directly or indirectly contributed to this work, including peers and well-wishers who provided support and motivation during the development process.

Finally, I express my gratitude to my family for their constant encouragement and support, which helped me stay focused and complete this project successfully.

VIII. REFERENCES

[1] Python Software Foundation, *Python Documentation*. Available: https://docs.python.org

[2] Requests Library, *HTTP for Humans*. Available: https://docs.python-requests.org

[3] Beautiful Soup Documentation, *HTML Parsing Library*. Available: https://www.crummy.com/software/BeautifulSoup/

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

[4] Selenium Documentation, *Web Browser* Automation. Available: https://www.selenium.dev/documentation/
[5] Pandas Documentation, *Data Analysis and Manipulation Tool*. Available: https://pandas.pydata.org/docs/
[6] OpenPyXL Documentation, *Excel File Handling in Python*. Available: https://openpyxl.readthedocs.io
[7] CustomTkinterDocumentation, *Modern GUI Framework for Python*. Available: https://customtkinter.tomschimansky.com/
[8] Tkinter Library, *Standard GUI Toolkit for Python*. Available: https://docs.python.org/3/library/tkinter.html
[9] W3C, *HTML and Web Standards*. Available: https://www.w3.org
[10] Mozilla Developer Network, *Web Technologies Documentation*.Available: