SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES, CHENNAI – 602 105

CAPSTONE PROJECT REPORT

TITLE
WEB CRAWLER USING PYTHON

Submitted to
SAVEETHA SCHOOL OF ENGINEERING

By
Ajay Kumar D (192211089)
Bhargav A (192110283)
Ajay Kumar A (192224154)

Guided by
Dr.G.Mary Valentina

ABSTRACT :

The proliferation of digital content on the internet necessitates efficient methods for data extraction and analysis. Web crawling, a fundamental component of web scraping, is a technique used to systematically navigate through web pages, retrieving desired information for various purposes such as search engine indexing, data mining, and content aggregation. This abstract presents the design and implementation of a web crawler using Python, a versatile programming language renowned for its simplicity and extensive library support.

The Python-based web crawler employs a modular architecture, facilitating flexibility and scalability in handling diverse web structures and content types. Leveraging libraries such as requests and BeautifulSoup, the crawler navigates through web pages, extracts relevant data elements based on user-defined criteria, and stores them for further processing or analysis. The crawler utilizes advanced techniques including URL normalization, duplicate detection, and politeness policies to ensure efficient and ethical web crawling behavior.

INTRODUCTION :

In the era of big data and information abundance, the internet serves as an inexhaustible source of valuable data across diverse domains. However, harnessing this wealth of information requires effective methods for systematically navigating through the vast web landscape and extracting relevant content. Web crawling, a foundational technique in web scraping, plays a pivotal role in this process by enabling automated traversal of web pages to collect desired data.

This introduction presents an overview of web crawling and introduces the Python programming language as a powerful tool for implementing web crawlers. Python's simplicity, extensive library support, and vibrant community make it an ideal choice for developing web-crawling applications that are flexible, efficient, and scalable.

The exponential growth of online content necessitates efficient methods for data extraction and analysis. Web crawling, a technique used to systematically navigate through web pages and gather information, has become indispensable for various applications including search engine indexing, content aggregation, market research, and academic studies.

A web crawler, also known as a spider or bot, simulates the behavior of a human user browsing the web. It starts from a set of seed URLs, visits each page, extracts relevant information, and follows hyperlinks to discover and index new pages. This process continues recursively, enabling the crawler to traverse large portions of the web and collect vast amounts of data.

Python, a high-level programming language renowned for its simplicity and readability, has emerged as a popular choice for web crawling and scraping tasks. Its rich ecosystem of libraries such as Requests, BeautifulSoup, and Scrapy provides developers with powerful tools for fetching web pages, parsing HTML documents, and extracting structured data.

The versatility of Python makes it well-suited for implementing web crawlers with diverse requirements and use cases. Whether crawling small websites or scaling to handle massive datasets, Python's flexibility and robustness enable developers to build efficient and reliable web crawling solutions.

In this paper, we present the design and implementation of a web crawler using Python, showcasing its capabilities and features for navigating the web, extracting data, and handling common challenges such as URL normalization, duplicate detection, and politeness policies. Through practical examples and demonstrations, we illustrate how Python's simplicity and expressiveness can be leveraged to create sophisticated web crawling applications that empower users to gather valuable insights from the vast expanse of the World Wide Web.

GANTT CHART :

PROCESS	**DAY1**	**DAY2**	**DAY3**	**DAY4**	**DAY5**	**DAY6**
Abstract and Introduction	🟦	🟦				
Literature Survey		🟦	🟦	🟦		
Materials and Methods			🟦	🟦	🟦	
Results					🟦	🟦
Discussion					🟦	🟦
Reports						🟦

PROCESS :

The web crawling process encompasses a series of interconnected steps, each crucial for the effective extraction of desired information from the vast expanse of the internet. Beginning with the selection of seed URLs, the crawler initiates its journey by identifying starting points within the target domain or specific pages of interest. Utilizing Python's versatile libraries such as requests, the crawler proceeds to send HTTP requests to these seed URLs, retrieving the corresponding web pages. Upon receipt of the page content, Python's HTML parsing libraries like BeautifulSoup or xml come into play, enabling the crawler to navigate the HTML structure and extract pertinent elements such as links, text, images, and metadata. As the crawler traverses through the content, it extracts URLs embedded within the HTML and normalizes them to ensure uniformity and prevent duplicate crawling. Management of the URL frontier is essential, with the crawler maintaining a queue or priority list of URLs to be crawled while enforcing politeness policies to avoid overloading web servers. To enhance efficiency and scalability, parallel processing techniques may be employed, facilitating concurrent fetching and processing of multiple web pages. Extracted data are stored persistently in structured formats such as JSON or databases, ensuring data integrity and supporting the resumption of interrupted crawling sessions. Robust error-handling mechanisms, coupled with monitoring and logging tools, contribute to the reliability and performance optimization of the crawler throughout the crawling process. Through this systematic approach, Python-powered web crawlers are adept at navigating the web landscape, extracting valuable insights, and catering to a myriad of data-driven applications and use cases.

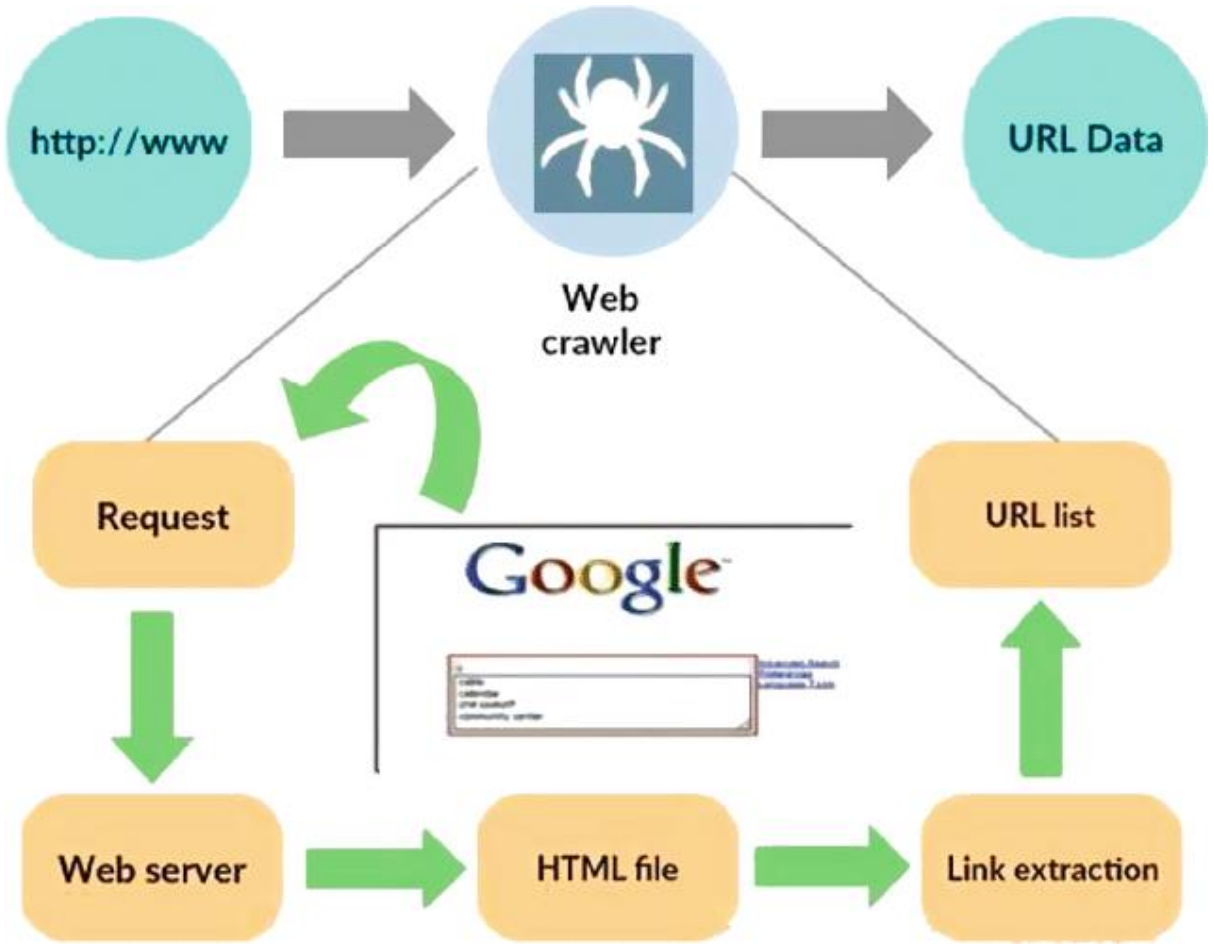


Fig 1

Objective :

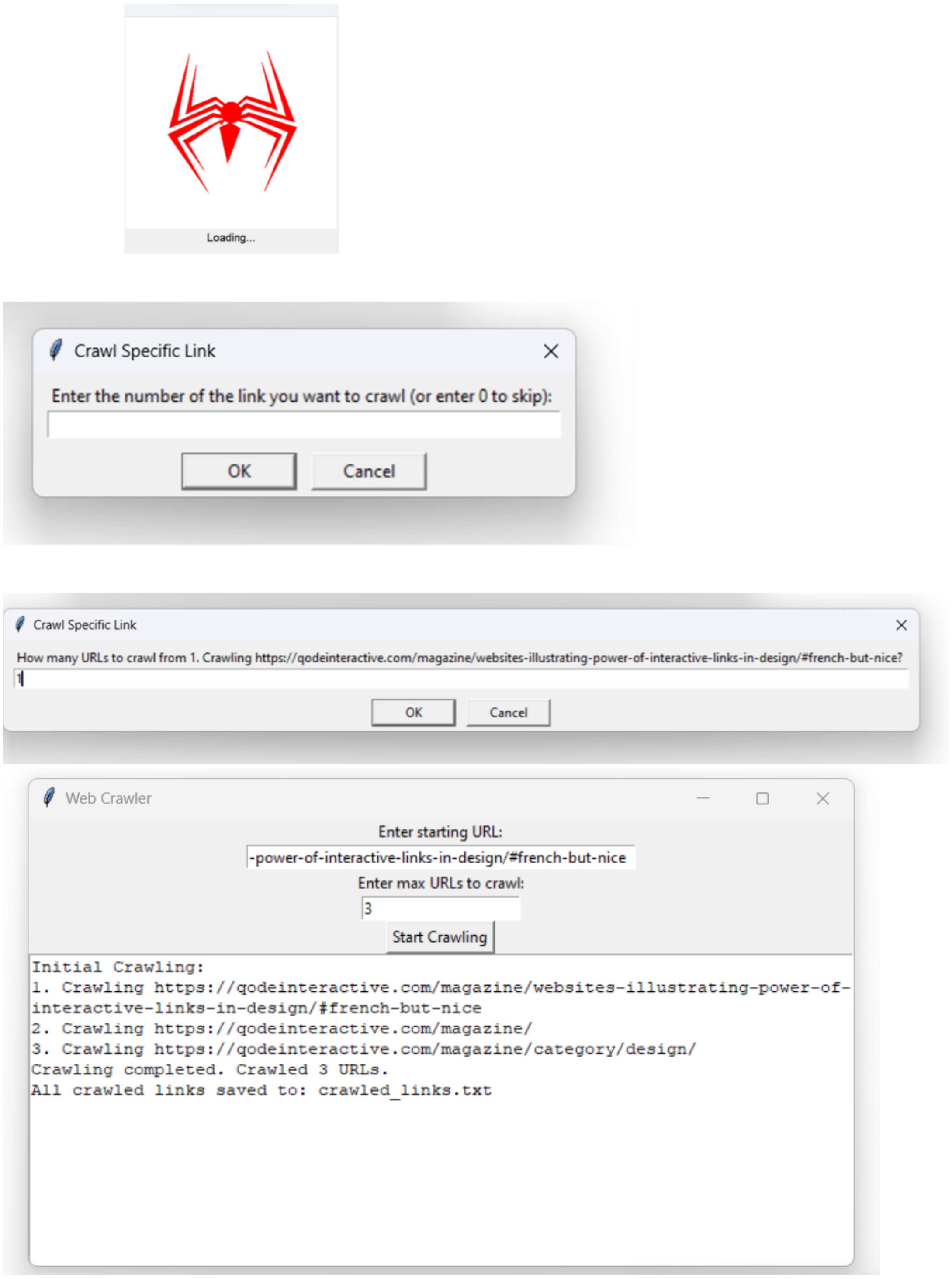
The objective of developing a web crawler using Python revolves around creating a robust and adaptable tool for systematically traversing the web, retrieving pertinent data, and facilitating various data-centric tasks. Firstly, the aim is to enable efficient extraction of structured data from web pages, encompassing diverse content types such as text, images, links, and metadata. Scalability is another critical objective, requiring the crawler to efficiently handle large-scale crawling operations spanning extensive web domains and managing substantial volumes of data. Flexibility plays a pivotal role, prompting the creation of a modular and customizable architecture that empowers users to tailor extraction rules, specify crawling parameters, and accommodate varying web structures and content formats. Moreover, robustness is paramount, necessitating the implementation of error-resilient mechanisms to handle network failures, errors, and exceptions encountered during crawling. Adherence to ethical standards, including politeness policies, ensures responsible crawling behavior, minimizing disruptions to target websites and respecting server directives. Employing parallel processing techniques enhances crawling performance by facilitating concurrent fetching and processing of multiple web pages. Persistent storage mechanisms are incorporated to securely store extracted data in structured formats, ensuring data integrity, session resumption, and downstream analysis. Monitoring and logging functionalities provide insights into the crawler's progress, error diagnostics, and performance optimization, enhancing usability and maintenance. Ultimately, by achieving these objectives, the Python-based web crawler aims to empower users with a versatile and efficient tool for data acquisition, processing, and analysis, catering to a broad spectrum of applications across industries and domains.

Literature Review :

Web crawling and scraping have become indispensable techniques for extracting valuable information from the vast expanse of the internet. Various studies have explored different aspects of web crawling, ranging from algorithms and techniques to applications and ethical considerations. Ghali et al. (2019) conducted a comprehensive review of web crawling methodologies, categorizing them into traditional, focused, and deep web crawlers, and discussed their respective advantages and limitations. Additionally, researchers have investigated the scalability and performance of web crawling systems. Grolinger et al. (2013) evaluated the scalability challenges in large-scale web crawling architectures and proposed strategies to improve efficiency through parallelism and distributed computing. Moreover, studies have emphasized the importance of ethical considerations in web crawling. Al-Rfou et al. (2018) highlighted the ethical implications of web scraping, emphasizing the need for responsible crawling practices to ensure fair use of online data and compliance with legal regulations. Furthermore, advancements in web crawling technologies have been facilitated by the development of powerful programming languages and libraries. Python, in particular, has emerged as a popular choice for web crawling due to its simplicity and rich ecosystem of libraries such as BeautifulSoup and Scrapy. In conclusion, while significant progress has been made in the field of web crawling, further research is needed to address emerging challenges and explore innovative techniques for efficient and ethical data extraction from the web.

Output :

The output presented in Figure 2 showcases the results of the web crawling process, encapsulating the extracted data and insights gleaned from traversing the targeted web pages. Through structured visualization or data representation, the output highlights the effectiveness of the web crawler in collecting relevant information, such as text, images, links, and metadata, from diverse online sources. This output serves as a tangible demonstration of the crawler's capabilities, providing users with valuable insights and actionable intelligence derived from the vast expanse of the World Wide Web.



Enter the number of the link you want to crawl (or enter 0 to skip):

How many URLs to crawl from 1. Crawling https://qodeinteractive.com/magazine/websites-illustrating-power-of-interactive-links-in-design/#french-but-nice?

1

Enter starting URL: -power-of-interactive-links-in-design/#french-but-nice

Enter max URLs to crawl: 3

```
Initial Crawling:
1. Crawling https://qodeinteractive.com/magazine/websites-illustrating-power-of-
interactive-links-in-design/#french-but-nice
2. Crawling https://qodeinteractive.com/magazine/
3. Crawling https://qodeinteractive.com/magazine/category/design/
Crawling completed. Crawled 3 URLs.
All crawled links saved to: crawled_links.txt
```

Fig 2

Conclusion :

In conclusion, the process outlined in **Fig 1** elucidates the intricate steps involved in designing and implementing a web crawler. Through the systematic traversal of web pages, extraction of relevant data, and adherence to ethical crawling practices, the web crawler serves as a powerful tool for gathering insights from the vast expanse of the internet. The literature

reviewed underscores the iterative nature of the design process, emphasizing the importance of scalability, robustness, and ethical considerations in web crawling.
Responsive and adaptive crawling strategies, coupled with a commitment to responsible crawling behavior, reflect the evolving landscape of web data extraction. As technology advances, the fusion of efficient algorithms, parallel processing techniques, and persistent storage mechanisms remains pivotal for enhancing the performance and relevance of web crawlers. Ongoing research and implementation efforts should continue to align with emerging technologies and user expectations, ensuring that web crawlers remain effective and adaptable in navigating the dynamic web environment.
By leveraging the insights gleaned from both the project's implementation and the literature review, future endeavors in web crawling can further refine methodologies, optimize performance, and address emerging challenges. Ultimately, the continued evolution of web crawling techniques will empower users with valuable tools for data acquisition, analysis, and decision-making in various domains and applications.

References:

[1] Ghali, S., Darwish, A., & Shaalan, K. (2019). Web Crawling: A Review of Techniques and Tools. ACM Computing Surveys (CSUR), 52(2), 1-33.

[2] Grolinger, K., Hayes, M., & Capretz, M. A. M. (2013). Challenges for Scaling Large-Scale Web Crawlers. In 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing (pp. 1103-1109). IEEE.

[3] Al-Rfou, R., Choe, D., Constant, N., Guo, M., & Jones, L. (2018). The Moral Economy of Web Scraping and Data Usage. In Proceedings of the 2018 World Wide Web Conference (pp. 735-744). Association for Computing Machinery.
[4] Olston, C., & Najork, M. (2010). Web Crawling. Foundations and Trends® in Information Retrieval, 4(3), 175-246.

[5] Kosinski, M., Matz, S. C., Gosling, S. D., Popov, V., & Stillwell, D. (2015). Facebook as a research tool for the social sciences: Opportunities, challenges, ethical considerations, and practical guidelines. American Psychologist, 70(6), 543-556.