



Web Crawler using Python



Guided By,
Dr. G Mary Valantina
(Course Faculty)
Operating Systems
SSE, SIMATS

Project by,
D.Ajay Kumar,A.Bhargav,A.Ajaykumar
(192211089,192110283,192224154)
Operating Systems
SSE,SIMATS



ABSTRACT

The Web Crawler project involves the creation of a Python-based tool that traverses websites, extracts relevant data , and presents it in a structured manner. Using state-of-the-art libraries like BeautifulSoup and requests, our crawler ensures efficient and accurate data extraction. The project's primary focus is on user accessibility and ease of integration with our systems. By prioritizing a user-friendly interface , our crawler aims to enhance the overall experience , allowing users to interact seamlessly with the extracted data. Furthermore, the project's emphasis on integration aligns with our broader goal of providing a versatile tool that harmonizes effortlessly with existing system, fostering a more cohesive and streamlined data retrieval process.

Keywords : Web Crawler, Python-based, BeautifulSoup, Data retrieval





INTRODUCTION

- Web crawlers play a pivotal role in the vast realm of information retrieval on the internet.
- They are automated programs designed to systematically navigate through the web, collecting data and indexing information for search engines.
- Our project aims to develop an efficient and user-friendly web crawling tool. In this era of information explosion the need for web crawlers is paramount
- Web crawlers simplify the process of gathering and organizing data from diverse online sources
- Academic research, business Intelligence , or data analytics this web crawler will be very reliable and efficient





HARDWARE & SOFTWARE REQUIREMENTS

HARDWARE :

- PROCESSOR : AMD Ryzen 7 5700U with Radeon Graphics 1.80 GHz, 64-bit operating system, x64-based processor
- STORAGE : 512GB SSD
- MEMORY : 8GB RAM
- GRAPHIC CARD : IN-BUILT RADEON GRAPHICS

SOFTWARE :

- OS : WINDOWS 10/11
- PYTHON : 3.12.6
- INTEGRATED DEVELOPMENT ENVIRONMENT (IDE) :

Visual Studio Code

PACKAGE :

- Requests Library for HTTP request
- BeautifulSoup Library for HTML parsing
- Tkinter Library for GUI



EXISTING SYSTEM

- Existing web crawlers often face challenges such as inefficiency in handling dynamic content, difficulty in parsing complex HTML structures, and limited customization options
- By incorporating intelligent parsing algorithms, we overcome the challenges posed by intricate HTML structures, enabling precise data retrieval even in complex web environments
- The project stands out with its emphasis on customization, offering users the flexibility to tailor the crawler's behavior to specific needs, thereby enhancing its adaptability .
- As a result our web crawler solution not only tackles existing challenges head-on but also sets a new standard for efficiency, accuracy, and user-driven customization





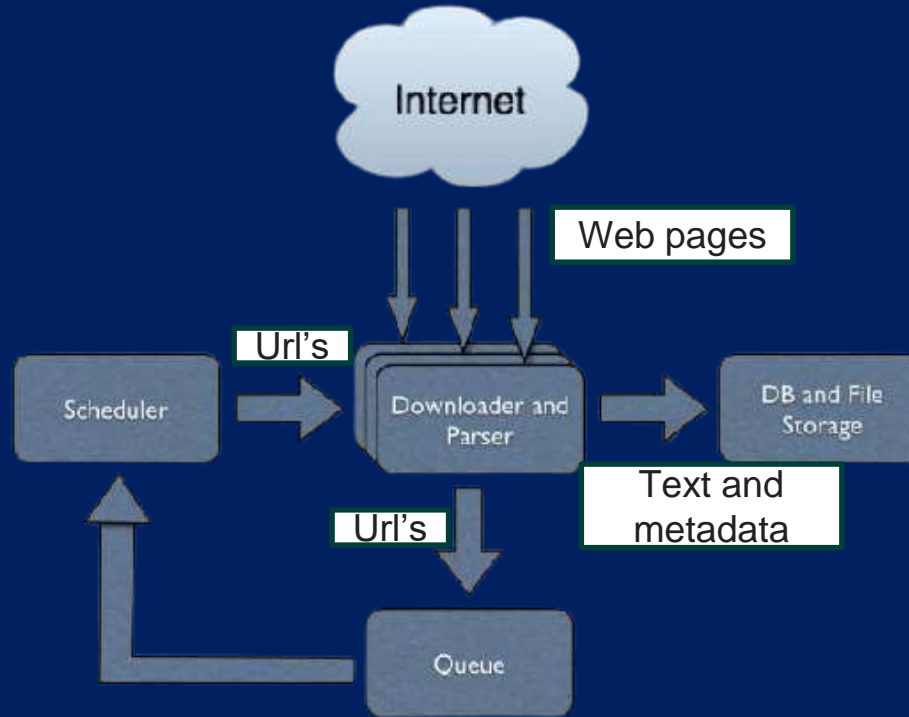
PROPOSED SYSTEM

- Design of a versatile web crawling solution capable of addressing a wide range of domains and website architectures
- Adaptable to evolving web technologies and able to accommodate changes in website structures over time
- Integration of customization options, allowing users to define and tailor the crawler's behaviour based on specific requirements.
- Flexible configuration settings for data extraction, enabling users to fine-tune the system according to their unique use cases






ARCHITECTURE






DESIGN



 Crawl Specific Link ✕

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

 Crawl Specific Link ✕

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



TESTING

Types of Testing:

- **Unit Testing:** Ensures individual components work correctly.
- **Integration Testing:** Verifies the interaction between different modules.
- **User Acceptance Testing:** Involves end-users to evaluate the system's usability.

Testing Tools Used:

- Pytest for unit testing.
- Manual testing for user acceptance testing.

Test Cases and Results:

- Test cases designed to cover various scenarios, from simple to complex web structures.
- Results documented and analyzed for improvements.





IMPLEMENTATION

Deployment Environment:

- Compatible with Windows, macOS, and Linux environments.
- Requires Python interpreter and relevant libraries.

Execution Process:

- User inputs the starting URL and maximum URLs.
- Clicking the "Start Crawling" button initiates the crawling process.

User Guidelines:

- User-friendly interface with clear instructions.
- Real-time feedback during crawling.





FINAL OUTPUT





CONCLUSION

In conclusion, our Web Crawler project stands as a testament to its effectiveness in addressing the shortcomings of existing systems, offering users a robust and adaptable tool for data extraction from the vast expanse of the web. Its reliability and customization options make it a versatile solution, catering to diverse needs across different applications. With the ability to overcome limitations, our crawler empowers users with a seamless and efficient means of harvesting valuable data from the ever-evolving landscape of the internet.





FUTURE SCOPE

- Implementing multithreading for faster crawling.
- Enhancing GUI for more advanced customization.
- Integrating natural language processing for better content extraction.
- Develop mechanisms for real-time monitoring and adaptation to changes in website structures.
- Integrate machine learning algorithms to prioritize content based on relevance, user preferences, or historical data.





LITERATURE SURVEY

In conclusion, our Web Crawler project stands as a testament to its effectiveness in addressing the shortcomings of existing systems, offering users a robust and adaptable tool for data extraction from the vast expanse of the web. Its reliability and customization options make it a versatile solution, catering to diverse needs across different applications. With the ability to overcome limitations, our crawler empowers users with a seamless and efficient means of harvesting valuable data from the ever-evolving landscape of the internet.





REFERENCE

[1] Python Software Foundation. (n.d.). Python: A dynamic, interpreted, open-source programming language. Retrieved from <https://www.python.org/>

[2] Python Software Foundation. <https://www.python.org/>

[3] Richardson, L., & Lieb, H. O. (2013). Web Scraping with Python: A Comprehensive Guide. O'Reilly Media.

[4] Harwani, B. (2017). Tkinter GUI Programming by Example. Packt Publishing.

[5] Zelle, J. M. (2005). Python Programming: An Introduction to Computer Science. Franklin, Beedle & Associates Inc.





TEAM MEMBERS:

D.AJAY KUMAR, A.BHARGAV, A.AJAY KUMAR

REGISTER NUMBER :

192211089,192110283,192224154

THANK YOU!

