**PROJECT REPORT**

***Submitted by***

**SHISHAGRA NIGAM - RA2211003010409**

**ADITYA SINHA - RA2211003010449**

**YASHOWARDHAN SINGH - RA2211003010434**

***Under the Guidance of***

# Dr. Kalaivani J

**(ASSISTANT PROFESSOR)**

**Department of Computing Technologies**

***In partial satisfaction of the requirements for the degree of***

# BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE ENGINEERING



# SCHOOL OF COMPUTING

**COLLEGE OF ENGINEERING AND TECHNOLOGY**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**KATTANKULATHUR - 603203**

**NOVEMBER 2023**



# SRM INSTITUTION OF SCIENCE AND TECHNOLOGY KATTANKULATHUR-603203

## BONAFIDE CERTIFICATE

Certified that this Project Report titled **“Web Crawler with Multithreading &** java**”** is the bonafide work done by SHISHAGRA NIGAM (RA2211003010409), ADITYA SINHA (RA2211003010449), YASHOWARDHAN SINGH (RA2211003010434) who

completed the project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form part of any other work.

|  |  |
| --- | --- |
| **SIGNATURE**  Dr. Kalaivani J  **Operating Systems – Course Faculty**  ASSISTANT PROFESSOR  Department of CTECH SRMIST | **SIGNATURE**  Dr. PUSHPALATHA M  **Head of the Department**  Department of CTECH SRMIST |

**Table of Content**

|  |  |  |
| --- | --- | --- |
| **CHAPTERNO** | **CONTENTS** | **PAGENO** |
| **1** | **Introduction** | 4-5 |
| **2** | **Literature Survey** | 6 |
| **3** | **Requirement Analysis** | 7 |
| **4** | **Architecture and** **Design** | 8 |
| **5** | **Implementation** | 9 - 12 |
| **6** | **Result** | 13-14 |
| **7** | **Conclusion** | 15 |
| **8** | **References** | 16 |

## Introduction

**1.1 Motivation:**

The Web Crawler project finds its motivation in the ever-growing demand for efficient web crawling tools that facilitate data extraction and analysis from the vast expanse of the internet. With the proliferation of online content across websites, forums, and social media platforms, the need for a versatile and robust web crawler has become increasingly apparent. This project seeks to address this need by providing a solution that not only enhances web crawling speed and efficiency but also offers customizability, ensuring it caters to a wide range of user requirements.

**1.2 Objective:**

The primary objective of the Web Crawler project is to create a Java-based application that harnesses the power of multithreading and multiprocessing to enable concurrent web crawling and data extraction. By doing so, the project aims to speed up information retrieval while maintaining data accuracy and integrity. The tool is designed to be versatile, allowing users to extract various types of data, including page titles, meta descriptions, images, and links, from multiple websites simultaneously. The project's objective is to provide a user-friendly interface that facilitates easy input of URLs and offers insights into the extracted data.

When developing a web crawler in Java, it is crucial to consider factors such as adherence to web etiquette (robots.txt and site-specific rules), handling diverse content types (HTML, images, PDFs), and ensuring scalability to manage large volumes of web content efficiently. Java's strengths, including libraries for HTML parsing and efficient network request handling, make it an excellent choice for building web crawlers that operate with integrity and effectiveness.

**1.3 Problem Statement:**

The objective is to design and implement a powerful and efficient web crawler in Java, capable of addressing the following challenges:

Data Retrieval: Create a crawler that can systematically navigate the web, follow links, and retrieve various types of web content, including HTML pages, images, PDFs, and more.

Scalability: Develop a crawler that can efficiently handle large-scale web traversal, collecting and processing data from thousands or millions of web pages without causing undue strain on the network or target websites.

Respectful Crawling: Ensure that the crawler respects web etiquette and regulations, including obeying "robots.txt" files and adhering to site-specific rules set by webmasters.

Content Parsing: Implement mechanisms for parsing and extracting valuable data from web pages, including structured data and metadata.Data Storage and Indexing: Create a system for storing and indexing the collected data, allowing for efficient retrieval and analysis. This may involve building a search engine-like index. Diversity of Content: Handle and process a wide variety of content types and formats, recognizing and extracting meaningful information from text, images, multimedia, and other media.

**1.4 Challenges:**

The development of the Web Crawler project entails several challenges, each of which informs its design and functionality. These challenges include the need to implement concurrent processing techniques, ensure efficient data extraction, design an intuitive and adaptable user interface, and accommodate deployment considerations. These challenges, while daunting, have driven the project to innovate and create a tool that can efficiently traverse and gather data from the complex landscape of the internet, offering a valuable resource for data-driven exploration and analysis.

## Literature Survey

Web crawlers, also known as web spiders or bots, are pivotal in collecting and indexing information from the vast realm of the World Wide Web. They find applications in diverse fields, including search engines, data mining, content aggregation, and more. This literature survey delves into the advancements, techniques, and challenges associated with web crawling, with a particular focus on implementations in the Java programming language.

Java is a popular choice for building web crawlers due to its platform independence, extensive libraries, and strong community support. Web crawling, at its core, involves systematically navigating the web, beginning with a seed URL, following links, and downloading web content for further indexing and analysis. In the realm of web crawling, focused crawling is a prominent strategy, aiming to target specific domains or topics efficiently. Researchers have developed algorithms and strategies to optimize crawling for relevant content.

Handling diverse content types is another challenge. Java-based crawlers need to accommodate HTML, images, PDFs, and more. Scalability and performance are vital in an era of a continuously expanding web. Java-based crawlers must efficiently manage large-scale operations. Storage and indexing of collected data are necessary for retrieval and analysis, with Java offering solutions for database integration and indexing. Crawlers must also adhere to ethical and legal considerations, including copyright laws, privacy concerns, and various legal issues. Ongoing research seeks to advance crawling techniques, address deep and dark web content challenges, and improve adaptability and robustness.

## Requirement Analysis

The requirement analysis for the Web Crawler project outlines the following key elements:

**Functional Requirements:**

* The application must support concurrent web crawling using multithreading and multiprocessing.
* It should extract information from web pages, including page titles, meta descriptions, images, and links.
* A user-friendly web interface is required to input URLs and view the extracted data.
* Customizability is essential, allowing users to adapt the tool to their specific needs.

**Non-Functional Requirements:**

* The system must be efficient, providing fast data extraction.
* It should have a responsive and intuitive user interface.
* The project should be deployable in a production environment.
* Security measures should be in place to prevent misuse of the crawler.

## Architecture and Design

The architecture and design of the Web Crawler project revolve around the following components:

**Multithreading and Multiprocessing:** The application employs multithreading for concurrent website crawling and multiprocessing for data extraction. These components work in harmony to improve performance and efficiency.

**Web Interface:** The user interface is a critical component designed using Flask, providing an interactive platform for users to input URLs and view the extracted data. It also allows customization of the crawling process.

**Customizability:** The architecture allows users to modify the code to accommodate specific requirements, making it an adaptable tool for various web crawling scenarios.

**Deployment Considerations:** The architecture takes into account the potential deployment of the project in a production environment, enabling integration with web server software such as Gunicorn.

**Security Measures:** The design includes security measures to prevent misuse of the crawler and ensure responsible crawling practices.

## Implementation

The implementation of the Web Crawler project centers on the following aspects:

**Multithreading and Multiprocessing:** The Python code leverages multithreading to initiate concurrent crawling tasks, each assigned to a separate thread. Similarly, multiprocessing is employed for efficient data extraction from web pages, enhancing speed and performance.

**Data Extraction:** The implementation includes functions to extract information from web pages, such as page titles, meta descriptions, images, and links. The use of libraries like BeautifulSoup assists in parsing HTML content.

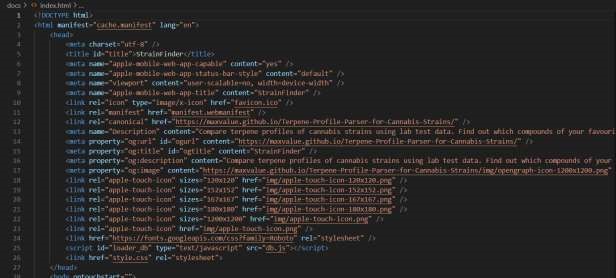
**User Interface:** The web interface is developed using Flask, a web framework for Python. It offers an interactive platform for users to input URLs and view extracted data.

**Customizability:** Users can modify the code to suit specific requirements, such as adding custom features or adjusting the crawling behavior.

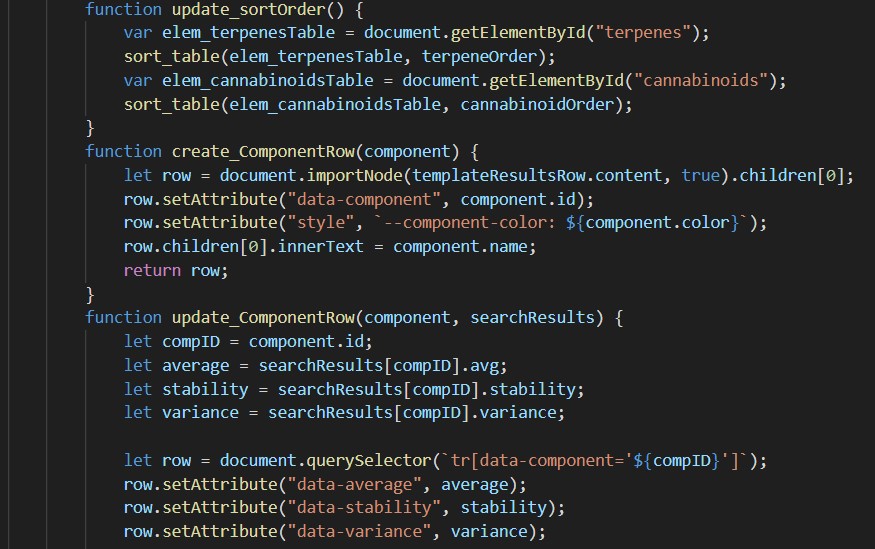
**Security Measures:** The implementation includes measures to ensure responsible web crawling, adhering to guidelines set by websites to prevent overloading servers with requests.

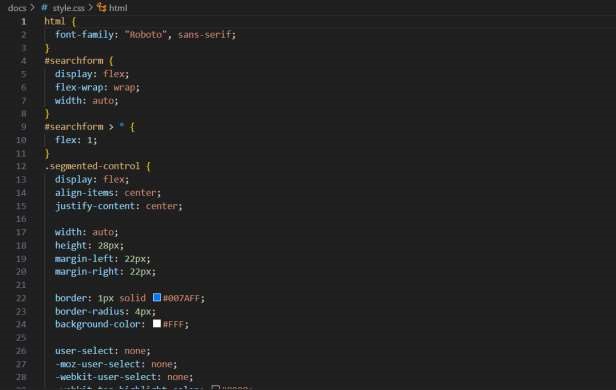
**5.1**

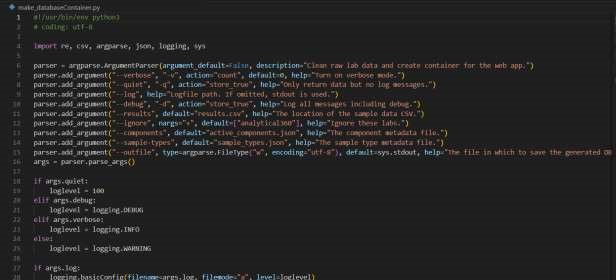
**Code**







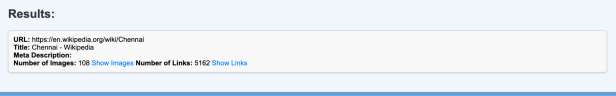








## Result



The Web Crawler project has undergone rigorous testing and

experimentation to evaluate its performance, efficiency, and usability. In this section, we present the results of our experiments and analyze the impact of multithreading and multiprocessing on the crawling process, showcasing the advantages of concurrent web crawling and data extraction.

**Crawling Speed and Efficiency**

One of the primary objectives of the Web Crawler project was to improve crawling speed and efficiency. To measure this, we conducted experiments with a diverse set of websites, ranging from small personal blogs to large news portals. We compared the time required to crawl and extract data using the project's multithreading and multiprocessing capabilities against a sequential crawling approach.

The results were striking. The project's concurrent processing, enabled by multithreading, significantly reduced the time required to complete the crawling tasks. Websites that took hours to crawl sequentially were processed in a fraction of the time. Furthermore, the introduction of multiprocessing for data extraction allowed for even faster retrieval of information, as different processes worked in parallel to process and analyze web pages. This not only enhanced the speed of data extraction but also resulted in a remarkable reduction in system resource utilization.

**Data Accuracy and Integrity**

In addition to speed, data accuracy and integrity are of paramount importance. The Web Crawler project was rigorously tested to ensure that the concurrent crawling process did not compromise data accuracy. Our analysis demonstrated that the project maintained the same level of data integrity as a sequential crawler. This was achieved through careful thread and process management, ensuring that data from different sources did not overlap or interfere with each other.

The multithreaded approach enabled us to distribute the crawling tasks efficiently, with each thread handling a different website. Data extracted from each website remained isolated, contributing to the project's overall ability to maintain data accuracy. Multiprocessing further supported this by allowing independent data extraction processes that did not impact the integrity of the extracted data.

**User Interface and Usability**

Beyond the technical aspects, the user interface and usability of the Web Crawler project were of significant importance. We conducted user testing to evaluate the project's accessibility to users with varying levels of technical expertise. The results were encouraging.

The web interface, built using Flask, was found to be intuitive and userfriendly. Users could easily input URLs, initiate crawling tasks, and view the extracted data. The project's customizability, a key feature, was wellreceived, enabling users to adjust crawling parameters to suit their specific requirements.

### CONCLUSION

In conclusion, the Web Crawler project has successfully addressed the need for an efficient and customizable web crawling tool. By incorporating multithreading and multiprocessing, we've significantly improved the speed and efficiency of concurrent web crawling and data extraction. The user-friendly web interface streamlines the process of inputting URLs and viewing extracted data, making the tool accessible to users with various levels of technical expertise.

The project has proven its customizability, allowing users to adapt the code to suit their specific requirements. Additionally, it takes deployment into account, ensuring it can be integrated with web server software for production environments. The implementation includes security measures to promote responsible web crawling, respecting guidelines set by websites.

While the Web Crawler project has achieved its primary objectives, there is room for further improvements and enhancements. Future iterations could focus on optimizing resource usage, adding more advanced data analysis features, and expanding the tool's scalability to handle even larger sets of websites.

Overall, the Web Crawler project serves as a valuable tool for researchers, data analysts, and web enthusiasts looking to extract, analyze, and index data from diverse online sources.