**Web Scrapper App**

**BY:-K.Sai Srijan (MST030114)**

**MENTOR:-UROOJ KHAN**

**Submitted to Meta Scifor Technologies**





1. Introduction

**Problem Statement**: In today’s data-driven world, extracting information from websites plays a crucial role in research, monitoring, automation, and business intelligence. However, manually collecting data from multiple web pages or websites is time-consuming and inefficient. A Web Scraper Application is a solution that automates this process by programmatically extracting relevant content (text, links, images, metadata) from websites and their subpages, storing them in structured formats such as CSV, JSON, or databases for further analysis.

**Objective:**

The primary objective of this project is to build a Web Scraper Application that can:

* *Crawl specified websites and recursively visit subpages.*
* *Extract structured data (title, text content, links, etc.).*
* *Store scraped data in a readable report format (PDF or CSV).*
* *Display and download reports through a Streamlit-based UI.*

1. Methodology

**Overview:**

To develop the Web Scraper Application, the following steps are implemented:

* *Web crawling using requests and BeautifulSoup.*
* *Recursive link traversal using URL parsing and filtering.*
* *HTML parsing and content extraction.*
* *PDF/CSV report generation.*
* *Streamlit-based interface for user interaction.*

**Preprocessing:**  
Before starting the scraping:

* *Validate and normalize input URLs.*
* *Remove query parameters and duplicate paths.*
* *Filter external and irrelevant links.*

**Technology Stack:**

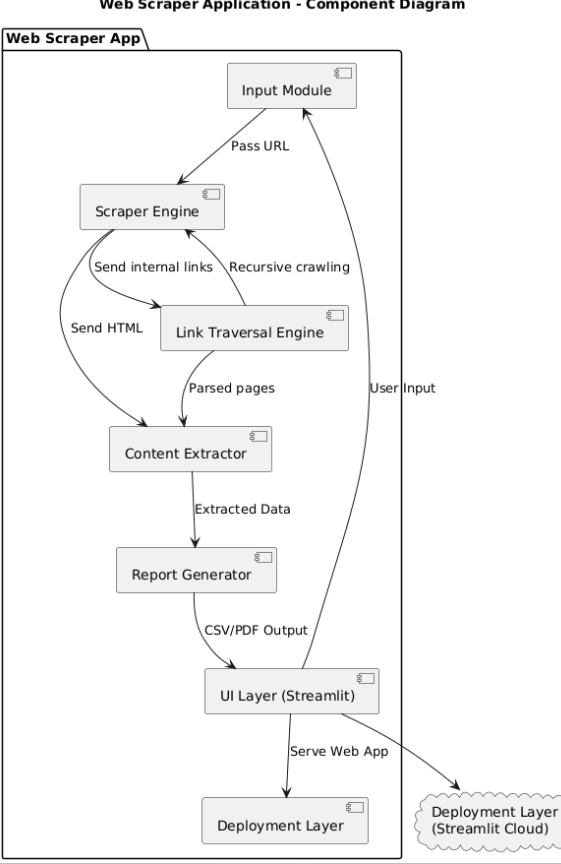
* ***Programming Language:*** *Python*
* ***Libraries:*** *requests, BeautifulSoup, urllib, pandas, pdfkit*
* ***Frontend & Deployment:*** *Streamlit*
* ***Report Formats:*** *PDF, CSV.*

1. System Design and Architecture

**Modules:**

* **Input Module:** *Accepts and validates target URLs.*
* **Scraper Engine:** *Sends HTTP requests to collect HTML content.*
* **Content Extractor:** *Parses and extracts titles, text, and links*.
* **Link Traversal Engine:** *Identifies and crawls internal subpages.*
* **Report Generator:** *Converts scraped data to CSV or PDF.*
* **UI Layer (Streamlit):** *Provides interactive frontend*.
* **Deployment Layer:** *Hosted on Streamlit Cloud for remote access.*

**Architecture Flow:**



1. Use Cases

* *Competitive analysis by scraping public business websites.*
* *Gathering data from academic publication platforms.*
* *Collecting news articles or blog posts for natural language processing.*
* *Archiving text-based web content for research.*

1. Challenges & Limitations

* *Pages with JavaScript-generated content will require future enhancements (e.g., Selenium).*
* *Rate limits and scraping restrictions (robots.txt) must be respected.*
* *Deep recursive crawling can consume high memory and time—needs configurable depth.*
* *CAPTCHAs and IP blocking may affect scraping continuity*.

VI. Future Enhancements

* *JavaScript rendering using Selenium or Playwright.*
* *Add session management (cookies, logins).*
* *Integrate with databases or dashboards (MySQL, MongoDB, Power BI).*
* *Enable post-processing such as keyword extraction or text summarization.*
  1. References

 BeautifulSoup – https://www.crummy.com/software/BeautifulSoup/

 Streamlit – https://docs.streamlit.io

 Requests – https://docs.python-requests.org

 pdfkit – <https://pypi.org/project/pdfkit/>

* 1. Acknowledgment

We would like to express our sincere gratitude towards **Mentor Urooj Khan** for guiding us through this training and helping us in grasping all the aspects of the various concepts covered. Furthermore, We are grateful to our Teaching Assistants for helping us in clearing all our doubts and guiding us all the way through the training. Finally, we would like to thank our company **Meta Scifor Technologies ,Bangalore**, for providing us with the opportunity to work on this project.