**Web Crawler Project**

**Submitted By**

Sajjad Ali

22K-8729

Hasan Abdul Rehman

22K-8727

**Department of Computer Science**

**National University of Computer & Emerging Sciences**

1. **Motivation**

The explosion of content on the web has created a massive, unstructured information space. Navigating and organizing this data manually is inefficient and infeasible. This project aims to develop a simple yet powerful web crawler to automate the process of data extraction and organization from online news platforms.

1. **Overview**
   1. **Significance of the Project**

This project addresses the growing need for structured data from unstructured web content. It is practical for academic research, journalism, data analytics, and digital archiving. Automating web data collection saves manual labor and provides real-time access to fresh information. The approach has academic value in demonstrating core concepts of HTTP requests, HTML parsing, recursion, and data filtering.

* 1. **Description of the Project**

The system is a web crawler that fetches HTML pages from a seed URL, parses them using BeautifulSoup, and extracts specific content elements such as links, headings, metadata, and images. It supports recursive crawling to a specified depth, ensuring comprehensive site coverage while filtering external links. The output includes structured JSON-like data.

* 1. **Background of the Project**

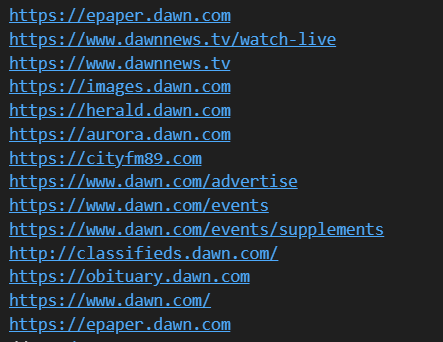
The web crawler uses Python libraries: requests for HTTP communication and BeautifulSoup for HTML parsing. It builds upon foundational knowledge of the Document Object Model (DOM), recursive traversal, and URL resolution. Inspired by search engine crawlers and research tools, it aims for lightweight, domain-specific extraction

* 1. **Project Category**

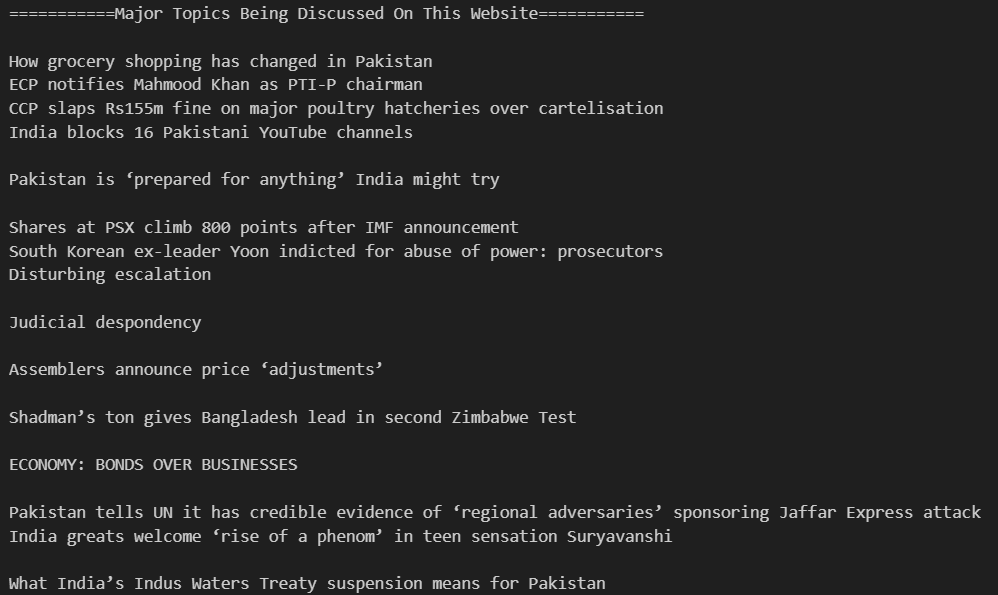
This is a **Product-based** project.

1. **Features / Scope / Modules**

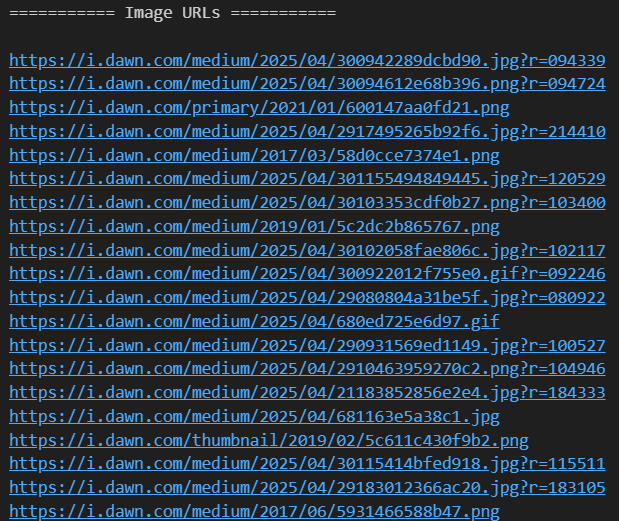
• **Link Extraction** Extracts all hyperlinks (<a href=...>) from the given seed URL, resolving relative paths.



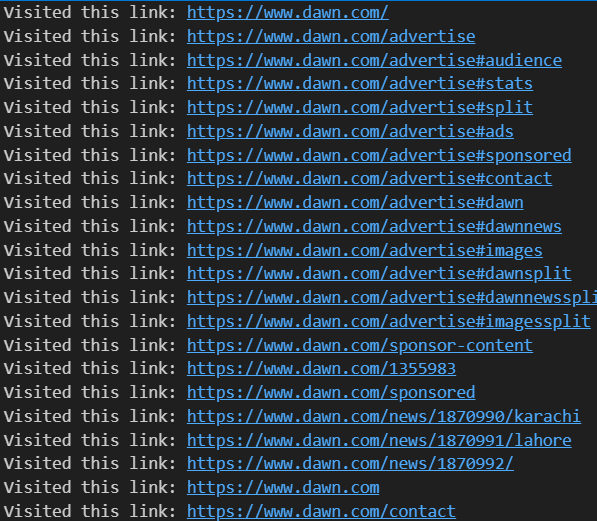
• **Heading Extraction** Identifies and stores all <h2> headings for summary-level content.



• **Image Extraction** Collects image sources (<img src=...>) to document visual content.



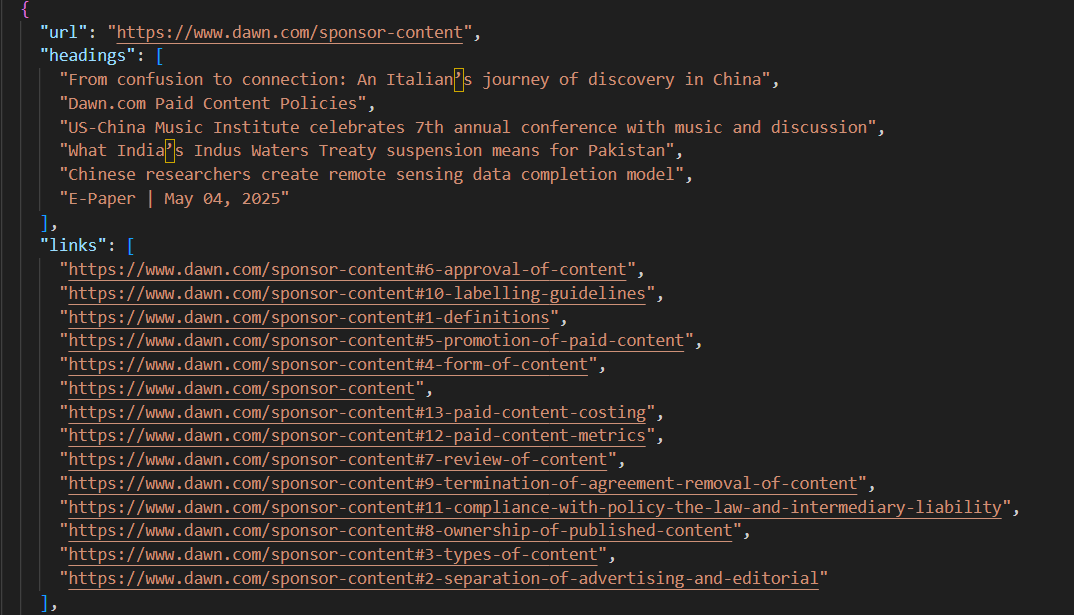
• **Recursive Crawling** Crawls internal links up to a configurable depth to simulate multi-level site traversal.

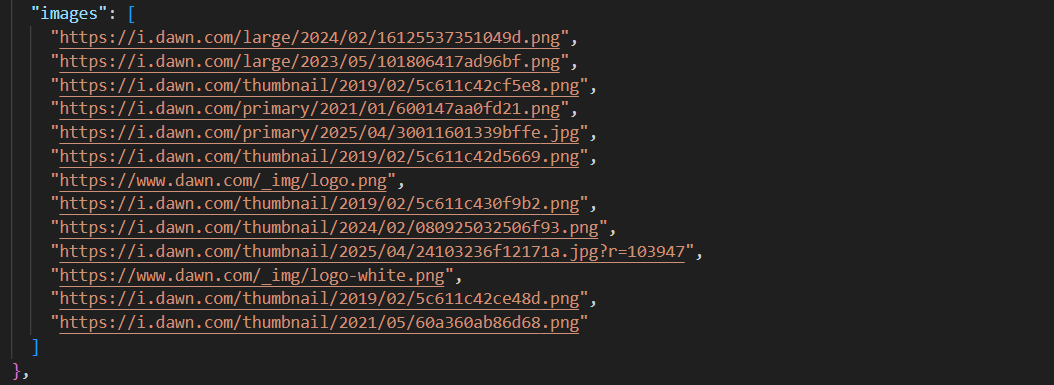


• **Content Filtering** Ignores duplicate links and external domains for cleaner, domain-specific scraping.

• **Storing the Results in a structured json form**

The data is saved in the json format, in which a specific page’s url is first saved, then the heading present in that link are saved, then the hyperlinks in that url are saved, and then the images in that are saved.





1. **Project Planning**

| 1. Research on web crawling, HTML structure (Sajjad Ali) |
| --- |
| 1. Setup environment and initial crawling tests (Sajjad Ali) |
| 1. Implement link and heading extraction (Sajjad Ali) |
| 1. Add metadata and image extraction   (Sajjad Ali) |
| 1. Add recursive crawling with depth limit (Hasan Abdul Rehman) 2. Save the data in a structured json format (Hasan Abdul Rehman) |
| 1. Test edge cases and clean outputs (Hasan Abdul Rehman) |
| 1. Finalize report and prepare video (Hasan Abdul Rehman) |

1. **Project Feasibility**

• **Technical Feasibility** The project uses mature libraries (requests, BeautifulSoup) and standard networking protocols, ensuring stability and compatibility.

• **Economic Feasibility** There are no additional hardware or license costs. It runs on freely available software with modest system requirements.

• **Schedule Feasibility** The timeline was followed successfully over 1 week with incremental integration and parallel development.

1. **Hardware and Software Requirements**

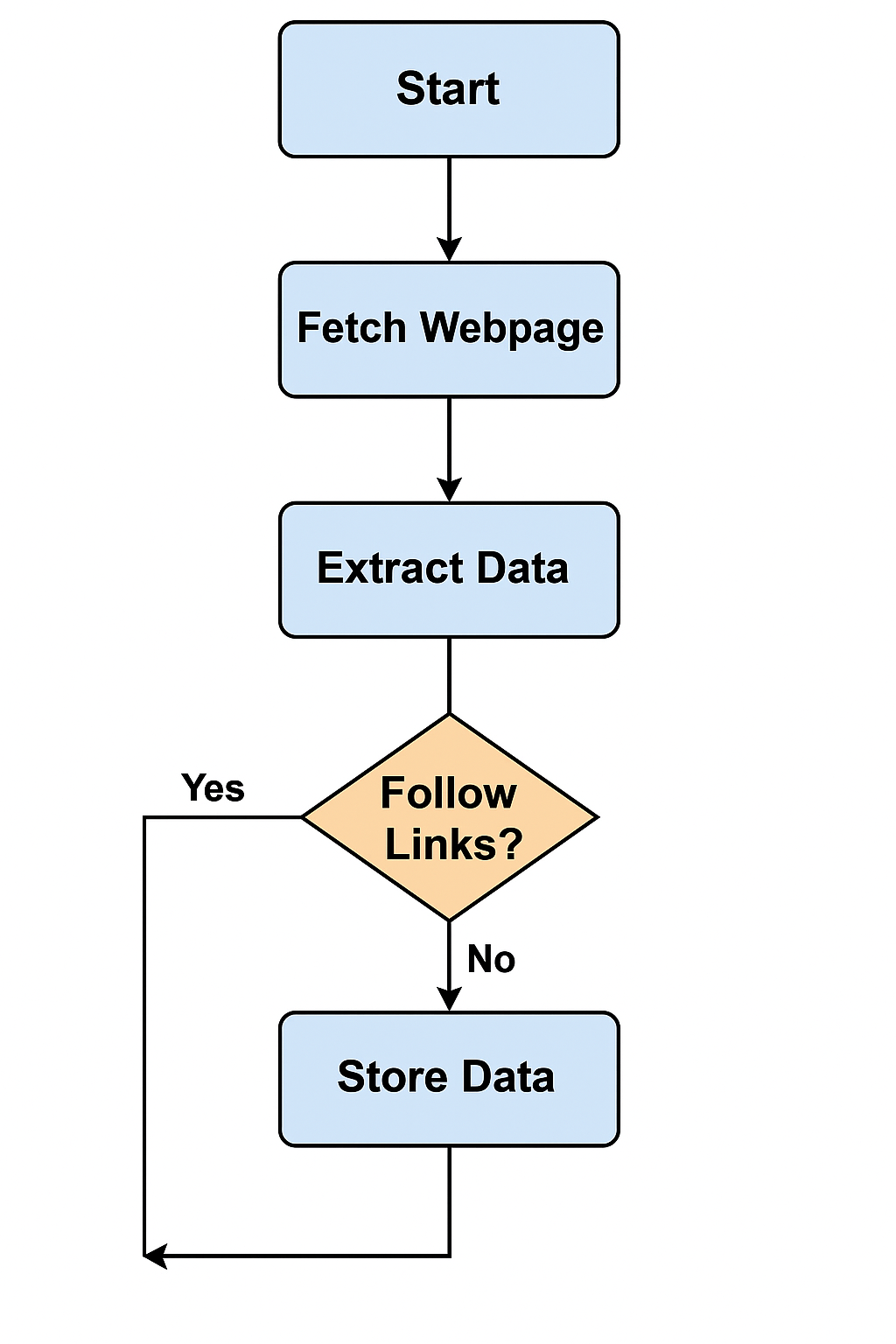
**Software:**

* Python 3.9+
* Jupyter Notebook
* requests, bs4,json python libraries

**Hardware:**

1. 4GB+ RAM
2. Stable Internet Connection
3. Any OS with Python support

1. **Diagrammatic Representation of the Overall System**

****

[Figure 1. Architecture of Web Crawler System]

* + User provides Seed URL
  + Requests made to server
  + HTML content parsed by BeautifulSoup
  + Content filtered and categorized
  + Structured data stored/displayed

1. **References**

[1] M. Richardson, A. Y. Halevy, and M. J. Franklin, "Principles of Web Crawling," in *WWW*, ACM, 2002.

[2]BeautifulSoupDocumentation,<https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

[3] Python Requests Documentation,<https://docs.python-requests.org/en/latest/>

[4] Kumar, V. "Web Crawlers: Theory and Practice", Springer, 2018.

[5] <https://www.woorank.com/en/blog/how-a-crawler-works-back-to-the-basics>