Web Crawling with Maximum Number of URLs

Documentation

Objective:

To create a web crawler that starts from the URL "https://www.bloomberg.com/" . And crawls up to a specified maximum number of URLs. The crawler will extract article titles and content using given XPaths, apply a regex pattern to filter valid child URLs, use multi-threading for concurrent requests, and store the crawled data in a database.

Requirements:

- Starting URL: "https://www.bloomberg.com/" .
- Regex Filter for Child URLs: Define a regex pattern to identify valid child URLs.
- Max Threads: Specify the maximum number of threads for concurrent requests.
- Max Number of URLs: Set a limit for the number of URLs to crawl (e.g., 100 URLs).
- Article Title XPath: XPath to extract the article title.
- Article Content XPath: XPath to extract the article content.
- Database: Use a database to store the crawled data, including URL, article title, article content, and timestamp of crawling.

Approach:

Crawler Initialization:

- Start from the given seed URL

Error Handling:

- Handle common HTTP errors (e.g., 404, 500).

Page Fetching:

- fetch the page content using playwright to handle JavaScript rendering.

Page Parsing:

- Parse the page content using playwright.
- Extract links using the regex filter.

Concurrency:

- Implement multi-threading.

Data Extraction:

- Use the provided XPaths to extract the article title and content.
- Store the data in a relational database.

URL Management:

- Maintain a set of visited URLs to avoid duplicates.
- Respect the maximum number of URLs parameter.

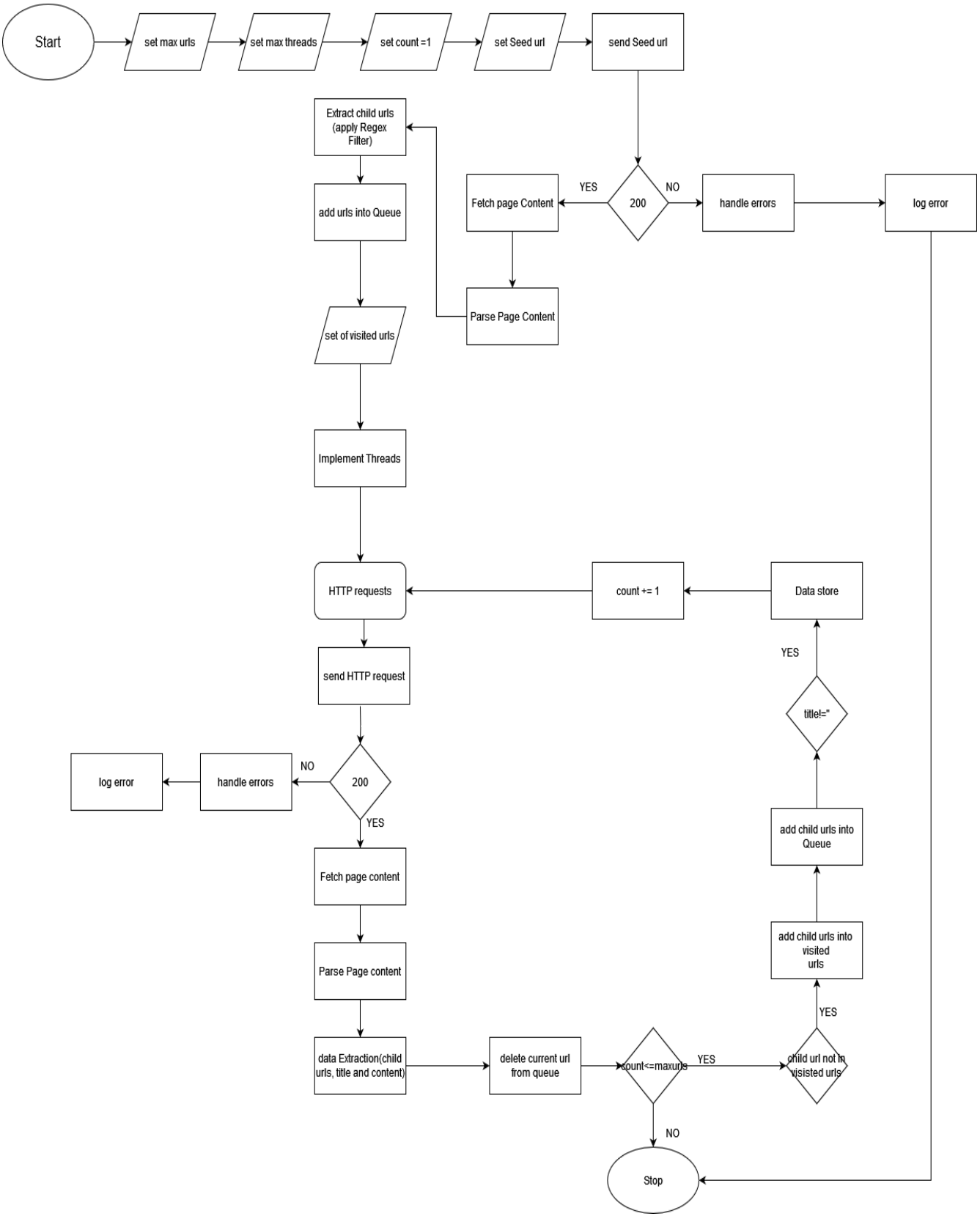
LOGIC:

1. Set the seed URL
2. Set the maximum threads for concurrent processing.
3. Set up a count and maximum URL limit.
4. Send seed URL
5. Check and handle errors (log, or skip).
6. Fetch the seed URL.
7. Parse the seed URLs content.
8. Extract valid URLs and add them to the queue.
9. Set of visited urls
10. Start threads:
    - Each thread will perform the following in parallel.
        - Send URL (send URL from queue)
        - if 200
            - Fetch the URL
            - Parse the URLs content
        - if not 200
            - Handle errors
            - log error
        - Extract the data(child URLs, title and content if it is present)
        - Check the condition for max URL limit if it is False stop the loop.
        - if True Continue until the max URL limit is reached.
        - If child URL is not in visited URLs then add into visited URLs and queue
        - If title and content is present store the extracted data into database
        - and then Increment the count of URLs

Tech-Stack:

- **Database:** MySQL
- **Language:** Python

Flowchart:


Start
set max urls
set max threads
set count =1
set Seed url
send Seed url
Extract child urls (apply Regex Filter)
add urls into Queue
set of visited urls
Implement Threads
HTTP requests
send HTTP request
200
NO
handle errors
log error
YES
Fetch page content
Parse Page content
data Extraction(child urls, title and content)
delete current url from queue
count<=maxurls
YES
NO
Stop
child url not in visisted urls
YES
add child urls into visited urls
add child urls into Queue
title!=''
YES
Data store
count += 1
Fetch page Content
200
YES
NO
handle errors
log error
Parse Page Content

Database Schema:

Table1: DataStor

column	Datatype	Constraints	Description
URL	VARCHAR(255)	Not Null	The URL of the crawled page
Article_Title	VARCHAR(255)		The title of the article
Article_Content	LONGTEXT(4,294,967,295)		The content of the article
Timestamp	TIMESTAMP	Not Null	The time when the page was crawled

Configuration Table:

Table1: DataConfig

config_names	config_values
Seed url	https://www.bloomberg.com/asia
Max threads	4
Max urls	100
article_title_xpath	//h1
article_content_xpath	//p[@data-component="paragraph"] | //h3[@data-component="subhead"] | //ul[@data-component="unodered-list"] | //li[@data-component="unordered-list-item"]
child_urls_xpath	//a
Count	1

Conclusion:

This documentation provides a guide to developing a web crawler for extracting article titles and content from Bloomberg.com, up to a specified maximum number of URLs. The crawler uses Playwright for efficient data extraction and handling dynamic content, and MySQL for structured data storage.