PBL

Report & Implementation

On

Crawler : within Search Engine

Submitted for partial fulfillment of Continuous Assessment

of

CSL0503 Design and Analysis of Algorithm

(July - Dec 2024)

Presented by

Submitted by

Rachit Kumar , Aditya raj savita
Betn1cs22136 , Betn1cs22056

Submitted to

Mr. H N VERMA
Associate Professor

(I m p l e m e n t a t i o n)

Prerequisites

- ***Node.js*** installed on your machine.
- ***MongoDB*** installed and running (or a MongoDB Atlas account).
- Create a ***.env*** file in the **server** directory and add your MongoDB connection URL:

1)- Implementation of code :

- Extract the attached ***Zip file*** named **" Crawler.7zip "** .
- Open the folder Crawler in ***IDE*** or ***Visual code Editor*** .
- You will see 2 folders , '***Frontend***' & '***Backend***' one folder for Client and another for Server respectively.
- Install the basic ***Dependencies*** that supports the running of the codes.
- Add Database (***mongodb***) to get it work fine.
- Now run the particular code in ***VScode Terminal*** by using following code:

~ path/Crawler/frontend **npm run start** .

~ path/Crawler/backend **npm run dev** .

- Now when the code runs Successfully , you will see the **UI** running on ***port 3000*** and ***Server on port 5000.***
- Now your setup is all set to run the ***Website or Crawler*** successfully

---

(R e p o r t)

Project Overview

This project is a web application built using the ***MERN stack*** (*MongoDB, Express, React, Node.js*) that allows users to perform ***web scraping*** or ***Crawling*** and search for specific content. The application utilizes *Axios* for making *HTTP* requests and Cheerio for parsing HTML, storing the extracted data in a ***MongoDB database.***

Objectives

- **Web Scraping**: Automatically crawl specified web pages to extract relevant data.
- **Data Storage**: Store the scraped data in a structured format in MongoDB.

- **Search Functionality**: Provide users with the ability to search for specific data using keywords.
- **User Interface**: Create a user-friendly frontend using React.

Technologies Used

- **Frontend**:
  - **React**: For building the user interface.
- **Backend**:
  - **Node.js**: Server-side JavaScript runtime.
  - **Express**: Web application framework for Node.js.
- **Database**:
  - **MongoDB**: NoSQL database for data storage.
- **Libraries**:
  - **Axios**: For making HTTP requests.
  - **Cheerio**: For parsing and manipulating HTML.
  - **CORS**: To handle cross-origin requests.
  - **Dotenv**: For managing environment variables.

Implementation Details

Backend:-

1. **Express Server**: The server is set up using Express, listening on a specified port.
2. **MongoDB Connection**: The application connects to a MongoDB database using Mongoose.
3. **Crawl Functionality**:
    - The `crawl` function uses Axios to fetch HTML content and Cheerio to extract the title, description, and links.
    - Data is saved to the database using Mongoose models.
4. **API Endpoints**:
    - **GET /search**: Accepts a query parameter and returns matching results from the database.
    - **POST /crawl**: Accepts a URL in the request body and initiates the crawling process.

Frontend:-

1. **React Application**:
    - The application provides an interface for users to input search queries and initiate web crawls.
    - The main component handles user inputs, displays search results, and interacts with the backend API.
2. **State Management**:

    - React hooks are used for managing input states and storing fetched data.
3. **Axios for API Calls**: Axios is used to make requests to the backend endpoints for searching and crawling.

Challenges Faced

- **Error Handling**: Implementing robust error handling for network requests and database operations.
- **Data Extraction**: Ensuring accurate data extraction from varying HTML structures of web pages.
- **CORS Issues**: Configuring CORS to allow cross-origin requests between the frontend and backend.

Results

The project successfully meets its objectives, allowing users to:

- Perform web scraping of specified URLs.
- Search through the scraped data effectively.
- Display results in a user-friendly interface.

Conclusion

This ***MERN web scraping*** *or* ***crawler*** project showcases the integration of multiple technologies to achieve automated data extraction and user interaction. Future enhancements could include advanced search capabilities, user authentication, and improved error handling for a more robust application.