Amazon Scrapper

Project Overview

Objective

The objective is to develop a Python script to scrape product details from Amazon for a set of search queries and display the results in a React frontend. The React app is deployed on Vercel.

Requirements

1. **Scrape product details** from Amazon for a list of search queries.
2. **Extracted details**: Title, Total reviews, Price, Image URL.
3. **Save data** into JSON files named after each search query.
4. **React frontend** to display the scraped data in a tabular format.
5. **Deploy the frontend** on Vercel.

Backend

To write backend i chose , Flask to write apis and scrapy to run the spider and crawl through the web pages.

Dependencies

- Python
- Scrapy
- Flask
- Flask-CORS

Installation

1. **Clone the repository or Download the code file**:

```
git clone https://github.com/your-repo/amazon-scraper.gi
t
cd amazon-scraper
```

2. **Create a virtual environment and activate it**:

```
python3 -m venv venv
source venv/Scripts/activate
```

3. **Install the required packages**:

```
pip install scrapy flask flask-cors
```

Directory Structure

- `app.py` : Contains the flask app and the API
- `user_queries.json` : Contains the json of keywords
- `amazonSpider.py` : Contains the Scrapy spider for scraping Amazon.

Running the Script

1. **Start the Flask server**:

```
flask run
```

Flask API

The Flask server runs an API endpoint at `http://localhost:3/scrape` to handle POST requests for scraping.

Frontend

Dependencies

- React
- Axios
- Material-UI

Installation

1. **Navigate to the frontend directory**:

   ```
   cd amazon-scraper
   ```

2. **Install the required packages**:

   ```
   npm install
   ```

Running the Frontend

1. **Start the React development server**:

   ```
   npm start
   ```

Deployment on Vercel

Vercel link is given below :

https://amazon-scrapper-amber.vercel.app/

Detailed Explanation

Design Decisions

1. **Modularity**: The codebase is divided into multiple modules to adhere to the Single Responsibility Principle.
2. **Scrapy**: Used for web scraping due to its robustness and efficiency.
3. **Flask**: Provides an API endpoint for the frontend to trigger the scraping process.
4. **React**: Used for the frontend to create a dynamic and responsive UI.
5. **Material-UI**: For a consistent and professional look for the React component.

Sample Output

- Each keyword will generate a JSON file named after the keyword, e.g., `headphones.json`.
- The JSON files contain an array of product details.