Coding Scraper Challenge

Create a Python/Django web scraper to scrape organic and sponsored search results from Amazon.co.uk, store the results in a database, and design an API for data access.

Requirements

Web Scraper (Python/Django)

- Write a Python/Django web scraper to scrape search results from Amazon.co.uk.
- Accept a list of keywords as input, e.g. cat food, dog food, car parts, gym clothes men.
- Scrape organic and sponsored results snippet information, storing the **Title**, **Description**, **Price**, and **Rating**.
- Store the results in the correct order and differentiate between **Organic** and **Sponsored** results.

Database (SQLite)

- Design a database schema to store the scraped data.
- Use Django as your base project so you can use Django model directly
- The scraper should accept the list of keywords as input, which the user can edit (add/remove keywords).
- Scrape jobs should plan to run daily and store the results for each day.

API Design (Django)

- Design an API to access the data stored in the database.
- Create an endpoint to trigger the scraper for today.

Instructions

Implement the web scraper using Python and Django. You may use libraries such as **BeautifulSoup** or **Scrapy** to help with the scraping process.

Design a database schema to store the scraped data. Consider using Django's built-in **models** and **migrations** to create and manage your database schema.

You should be able to add keywords using Django Admin.

Create an endpoint that triggers the scraper.

Implement an API to access the data stored in the database. The API should support querying data based on keyword and date.

Write unit tests to ensure the correctness of your scraper, database schema, and API.

Document your code and write a README file explaining how to set up and run your project, as well as how to use the API.

Ensure your code is clean, well-organized, and follows best practices.

Submission

Submit a GitHub repository link containing your complete project, including the source code for the scraper, API implementation, unit tests, and documentation.

END