Project Documentation : DealHunter : Scraper for Deals and Offers

Introduction

The "DealHunter" project aims to automate the extraction of deals and offers from the website [Deals Heaven](https://dealsheaven.in/) using Python. The project significantly reduces manual effort and time consumption, enabling users to access up-to-date deal information efficiently. By streamlining deal monitoring processes, this project helps users make informed decisions, potentially leading to cost savings and improved satisfaction. Real-time monitoring enhances user engagement and the overall experience.

The "DealHunter" project is a comprehensive Python-based tool developed to automate the process of extracting deals and offers from the website Deals Heaven. This project addresses the time-consuming nature of manually browsing deals by offering an automated and efficient solution.

The core objective of "DealHunter" is to streamline deal monitoring, providing users with up-to-date and accurate information about the latest offers. By integrating real-time monitoring capabilities, the project ensures users can quickly access relevant deals, aiding in faster and more informed decision-making.

In addition to enhancing efficiency, "DealHunter" helps users save money by ensuring they never miss out on significant offers. This not only improves user satisfaction but also fosters engagement by creating a seamless and reliable experience. With automation at its core, "DealHunter" redefines how users discover and interact with online deals, turning a tedious process into a straightforward and user-friendly experience.

Project Scope:

Included Features:
The project is designed to scrape essential deal-related information from Deals Heaven. This includes extracting headers, titles, descriptions, prices, ratings, seller details, images, and direct links to the deals. The extracted data is stored in a structured format, such as CSV or a database, enabling easy retrieval and further processing. To ensure reliability, the project includes robust error-handling mechanisms to manage unexpected issues during scraping, such as network errors or missing data fields. Comprehensive testing is also part of the scope to validate the tool's functionality under various scenarios, guaranteeing consistent and accurate performance.

Excluded Features:
The project is strictly focused on data extraction and does not extend to analyzing the scraped deals or providing recommendations to users. Similarly, integration with external applications or systems beyond the scope of data extraction, such as e-commerce platforms or analytics tools, is not included. This exclusion ensures the project remains lightweight and efficient in its core purpose.

Constraints:
The project faces certain constraints that may affect its functionality. Variations in the website's HTML structure, such as layout changes or updates, can disrupt the scraping process, requiring periodic adjustments to the tool. Additionally, the website may implement anti-scraping mechanisms like CAPTCHA, rate limiting, or IP blocking, which could limit the tool's ability to perform continuous scraping. These constraints require ongoing monitoring and potential enhancements to maintain the scraper's effectiveness.

By defining a clear scope, the "DealHunter" project maintains focus on its core objectives while acknowledging its limitations, ensuring a streamlined and purpose-driven approach.

Requirements

Functional Requirements:
The "DealHunter" project is designed to meet specific functional goals that enable efficient deal scraping from Deals Heaven. It must successfully extract key deal information, including headers, titles, descriptions, prices, ratings, seller details, images, and links. These details should be captured in a structured format for easy access and retrieval. Additionally, the scraper must adapt to variations in the website's HTML structure, ensuring continued functionality even if minor changes occur in the page layout or content structure.

Non-Functional Requirements:
The scraper must exhibit robustness and reliability in its operation, handling unexpected scenarios such as network interruptions, incomplete data, or website downtime. To ensure user trust, the extracted data must be validated for accuracy and completeness, avoiding errors or omissions. Moreover, the system should preserve data integrity during storage, ensuring that all retrieved information is stored securely and remains unaltered.

User Stories:

1. **Quick Retrieval of Deals:** As a user, I want the scraper to retrieve deal information quickly, allowing me to evaluate offers efficiently without delays. This ensures a seamless experience when accessing the latest deals.
2. **Accurate and Complete Data:** As a user, I want the scraper to provide accurate and complete information about each deal so that I can make well-informed purchase decisions, saving time and avoiding potential errors caused by missing or incorrect data.

These requirements ensure that "DealHunter" provides a dependable and user-centric tool for deal scraping, balancing functionality, reliability, and user satisfaction.

Technical Stack

Programming Languages:
The project is developed using Python due to its simplicity, versatility, and extensive library support for web scraping tasks.

Frameworks/Libraries:

- **BeautifulSoup :** A Python library used for parsing HTML and XML documents. It simplifies the process of navigating and extracting data from HTML structures.
- **Requests:** A library for making HTTP requests, enabling the scraper to retrieve web pages efficiently.

Databases:
The project employs file-based storage, such as text or CSV files, for storing the scraped data. This approach ensures simplicity in implementation and suffices for the project's scope.

Tools/Platforms:

- **Python IDEs:** Tools like PyCharm or Visual Studio Code are used for writing and debugging the Python code.
- **Git:** Version control system used for tracking changes and collaborative development.

Architecture/Design

System Architecture:
The system is divided into three key modules for clarity and maintainability:

1. **Website Scraping Module:** Responsible for sending HTTP requests to the target website and retrieving raw HTML content. This module handles network-related tasks and manages connections to the website.
2. **Data Extraction Module:** Parses the HTML content using BeautifulSoup to extract relevant information such as headers, titles, prices, and other deal details. This ensures data is structured and meaningful.
3. **Storage Module:** Stores the extracted data in a structured format, such as CSV or text files, ensuring easy access and portability.

Design Decisions:

- The project uses **BeautifulSoup** and **Requests** for their simplicity, reliability, and widespread community support. These tools streamline the scraping process and reduce complexity.
- A **modular design** is adopted to separate concerns, improving code readability, scalability, and ease of debugging. Each module focuses on a specific task (scraping, extraction, or storage), ensuring flexibility and maintainability.

Trade-offs:

- **File-based Storage:** Chosen for its simplicity and minimal setup requirements. While databases offer scalability, file-based storage aligns better with the project's scope and timeline, avoiding unnecessary overhead. This trade-off ensures quicker development while meeting current project needs.

Development

Technologies Used:

- The project uses **Python** for scripting due to its simplicity and robust libraries for web scraping.
- **BeautifulSoup** is used for parsing and navigating HTML, making it easy to extract data.
- **Requests** handles HTTP requests to fetch web page content efficiently.

Coding Standards:

- The code follows **PEP 8 guidelines** for readability and consistency, such as proper indentation and meaningful variable names.
- A **modular programming approach** is used, dividing the project into scraping, extraction, and storage modules for clarity and ease of maintenance.

Challenges and Solutions:

1. **HTML Variations:**
    - Websites may change their layout or structure over time.
    - The solution includes flexible parsing logic using BeautifulSoup and exception handling to manage unexpected changes without crashing.
2. **Anti-Scraping Measures:**
    - Websites often block bots with techniques like rate limiting or detecting requests without browser headers.
    - To address this, user-agent headers were added to mimic real browsers, and delays between requests were introduced to avoid detection.

This approach ensures the scraper is reliable, easy to maintain, and adaptable to changes or challenges.

Testing

Testing Approach:

1. **Unit Testing:**
    - **HTML Parsing:** Individual functions were tested to ensure they could correctly extract data (e.g., headers, titles, prices) from sample HTML snippets. Tests included scenarios with complete, partial, and missing elements to confirm reliable parsing.
    - **Data Validation:** Tested functions to ensure the extracted data met quality criteria, such as checking for non-empty values and valid formats (e.g., price as a numeric value). This step highlighted issues early in the scraping process.
2. **Integration Testing:**
    - End-to-end tests were conducted to validate the scraper's complete functionality, including retrieving HTML, extracting data, and storing it in a structured format (e.g., CSV).
    - Simulated real-world scenarios, such as intermittent network failures, unexpected HTML changes, and large volumes of data, to ensure the scraper performed reliably.
    - Verified the correctness of the stored data and its accessibility for further analysis or use.
3. **Edge Case Testing:**
    - Tested the scraper’s adaptability to uncommon or unexpected scenarios, such as:
        - Pages with unusual HTML structures or missing fields.
        - Data fields with unexpected values (e.g., empty descriptions, malformed links).
        - Large datasets with numerous deals to ensure efficient processing and storage.
    - Logged and handled exceptions gracefully, ensuring the scraper continued functioning without crashing.

Results:

- The scraper successfully extracted deals in various scenarios, demonstrating robust performance across diverse conditions.
- **Bugs Identified:** Minor issues in data validation logic were found, such as failing to handle empty price fields or incorrectly formatted URLs.

- **Resolutions:** Updated validation logic to account for these edge cases and improved error handling to log problematic entries while continuing the scraping process.

This rigorous testing approach ensured that the scraper operates reliably, even in challenging or unexpected conditions, while maintaining data accuracy and integrity.

Deployment

Deployment Process:

1. **Install Required Python Libraries:**
    - Before running the scraper, you need to install the necessary libraries. The core libraries required are **BeautifulSoup** for HTML parsing and **Requests** for making HTTP requests. These libraries can be installed using Python's package manager, pip.
    - To install them, run the following command in your terminal:

      bash
      Copy code
      pip install beautifulsoup4 requests

2. **Run the Main Python Script for Scraping and Storage:**
    - After installing the libraries, you can execute the main Python script. The script is responsible for scraping data from the target website, processing it, and storing the extracted information in the desired format (e.g., CSV or text files).
    - To execute the script, navigate to the directory where the script is stored and run the following command in the terminal:

      bash
      Copy code
      python dealhunter_scraper.py

    - This will trigger the scraping process, and the script will save the extracted deal data as specified in the storage format.

Deployment Instructions:

- **Ensure Internet Connectivity and Access to the Target Website:**
    - The scraper requires a stable internet connection to fetch data from the target website. Ensure that your internet connection is active and that there are no firewall or network restrictions preventing access to the website

(Deals Heaven). Without internet access or website availability, the scraper will not function.

- **Execute the Script from Terminal or IDE:**
  - The script can be executed either from the terminal or an integrated development environment (IDE) such as PyCharm or VS Code.
  - To run the script from the terminal, navigate to the script's directory and use the following command:

    bash
    Copy code
    python dealhunter_scraper.py

  - If you prefer using an IDE, open the script in your IDE and execute it directly from there. Ensure that the Python interpreter is properly set up in the IDE to avoid errors.

User Guide

Setup Instructions:

- **Install Python 3.x and Required Libraries:**
  Before using the scraper, ensure that you have **Python 3.x** installed on your machine. Python is the primary language for this project, and it provides the necessary environment to run the script.
    - To install Python, download it from the official Python website and follow the installation instructions for your operating system.
    - Once Python is installed, the required libraries for this project can be installed using a **requirements.txt** file. This file lists all the dependencies (such as BeautifulSoup and Requests) needed for the scraper.
    - To install the libraries, open your terminal, navigate to the project directory, and run the following command:

      bash
      Copy code
      pip install -r requirements.txt

- This command will automatically install all the necessary libraries listed in the requirements.txt file, ensuring that the environment is correctly set up to run the script.

Usage:

1. **Run the Script:**
   After setting up the environment and installing the required libraries, you can run the main Python script to start scraping.
    - Open your terminal or IDE (e.g., PyCharm, VS Code), navigate to the directory where the script is stored, and execute the script using the following command:

      bash
      Copy code
      python dealhunter_scraper.py

- The script will initiate the scraping process, sending HTTP requests to the target website and extracting deal information (such as headers, prices, descriptions, and links).

2. **Review Stored Data in the Output File:**
   Once the script completes its execution, the scraped data will be stored in an output file, usually in CSV or text format. You can review the data by opening this output file in a text editor or spreadsheet application (like Excel).
   - This file will contain all the extracted deals, including details such as titles, descriptions, prices, and URLs.

Troubleshooting:

1. **Issue: Script Fails to Fetch Data**
    - **Solution:** If the script fails to fetch data from the website, the first thing to check is your **internet connection**. Ensure that you are connected to the internet and that there are no issues with the network.
    - Additionally, verify that the target website (Deals Heaven) is accessible and has not been temporarily blocked or restricted. You can try opening the website manually in a browser to confirm its accessibility.
2. **Issue: Incomplete Data in the Output File**
    - **Solution:** If the output file contains missing or incomplete data, it may be due to parsing errors in the script. The scraper relies on the website's HTML structure to extract specific data fields, and any changes to this structure can cause issues.
    - In such cases, **review the log file** (if available) to check for any parsing errors or exceptions during the scraping process. The log file may provide clues, such as missing elements or unexpected HTML structures, which can be fixed by adjusting the parsing logic in the script.
    - Ensure that the scraper is up to date with the latest changes on the website, and adjust the CSS selectors or tags if needed.

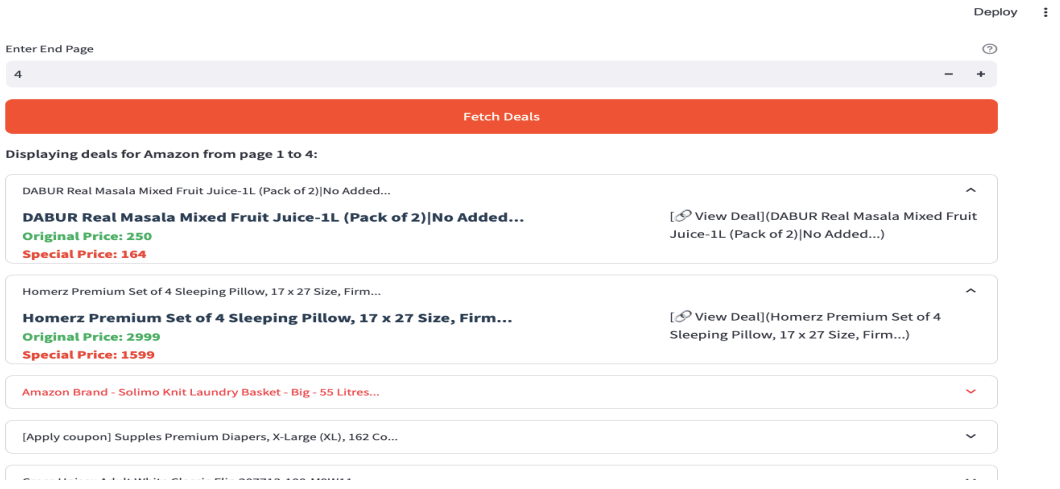
Conclusion

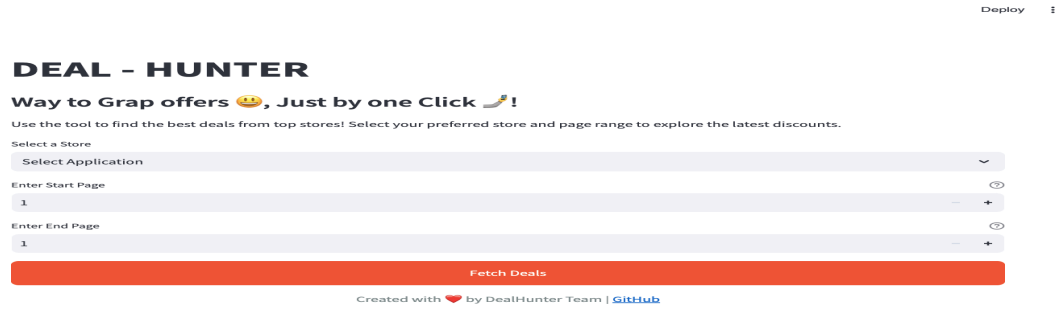
The **"DealHunter"** project successfully achieved its goal of automating the extraction of deals from the website Deals Heaven. This automation streamlines the deal monitoring process, significantly reducing the manual effort required to find and track offers. As a result, users can access up-to-date deals more efficiently, helping them make informed purchasing decisions with less effort.

One of the key learnings from the project includes handling **HTML variations**, where the website's structure might change, affecting the scraper's ability to correctly extract data. Overcoming this challenge involved creating flexible parsing logic and implementing error handling to adapt to these variations. Additionally, the project faced **anti-scraping measures** employed by the website, such as blocking or limiting automated requests. To address this, solutions like adding user-agent headers to requests and introducing delays between requests were implemented to mimic human-like behaviour and avoid detection.

Looking ahead, there are opportunities for future enhancements. One possible improvement is the integration of a **database** to store scraped data, making it easier to manage and query large volumes of deals. Additionally, **analytical features** could be added, such as trend analysis, deal comparison, or price history tracking, to provide users with deeper insights into the deals they are considering. These features could further enhance the user experience by offering more personalized and actionable data.

Appendices


Deploy
Enter End Page
4
Fetch Deals
Displaying deals for Amazon from page 1 to 4:
DABUR Real Masala Mixed Fruit Juice-1L (Pack of 2)|No Added...
DABUR Real Masala Mixed Fruit Juice-1L (Pack of 2)|No Added...
Original Price: 250
Special Price: 164
[View Deal](DABUR Real Masala Mixed Fruit Juice-1L (Pack of 2)|No Added...)
Homerz Premium Set of 4 Sleeping Pillow, 17 x 27 Size, Firm...
Homerz Premium Set of 4 Sleeping Pillow, 17 x 27 Size, Firm...
Original Price: 2999
Special Price: 1599
[View Deal](Homerz Premium Set of 4 Sleeping Pillow, 17 x 27 Size, Firm...)
Amazon Brand - Solimo Knit Laundry Basket - Big - 55 Litres...
[Apply coupon] Supples Premium Diapers, X-Large (XL), 162 Co...


Deploy
DEAL - HUNTER
Way to Grap offers 🤩, Just by one Click 🤳!
Use the tool to find the best deals from top stores! Select your preferred store and page range to explore the latest discounts.
Select a Store
Select Application
Enter Start Page
1
Enter End Page
1
Fetch Deals
Created with ❤ by DealHunter Team | GitHub


Public Libraries
Promoting local public libraries since 1999
Choose a State
Select
Select
State
Alabama
Alaska
Arizona
Arkansas
California


Public Libraries
Promoting local public libraries since 1999
Choose a State
Select

Public Libraries

Promoting local public libraries since 1999

Libraries in Alaska

	city	library	address	zip	phone
0	Akiachak	Akiachak School-Community Library	1 Main Street	99551	(907) 825-3600
1	Akiak	Akiak School-Community Library	1 Kilbuck Road	99552	(907) 765-4631
2	Anaktuvuk Pass	Nunamiut School-Community Library	114 Illinois Avenue		(907) 661-3226
3	Anchor Point	Anchor Point Public Library	72551 Milo Fritz Avenue	99556	(907) 235-5692
4	Anchorage	Anchorage Public Library	3600 Denali Street	99503	(907) 343-2975
5	Anchorage	Mountain View Branch Library	150 South Bragaw Street	99504	(907) 272-8467
6	Anchorage	Muldoon Branch Library	5530 East Northern Lights Boulevard	99504	(907) 338-4590
7	Anchorage	Z. J. Loussac Library	3600 Denali Street	99503	(907) 343-2983
8	Anderson	Anderson Village Library	101 First Street	99744	(907) 582-2628
9	Aniak	Aniak Public Library	270 Riverfront Drive	99557	(907) 675-4435

Mivi Fort H350 Soundbar, 350 Watts, 5.1 Channel, Multi-Input and EQ Modes, BT v5.1 350 W Bluetooth Soundbar(Black, 5.1 Channel)

Price: 34999.0

Special Price: 5999.0

Discount: 82%

Rating: 0.0 ⭐

View Deal

Deploy

🔢 Page Range

1

1

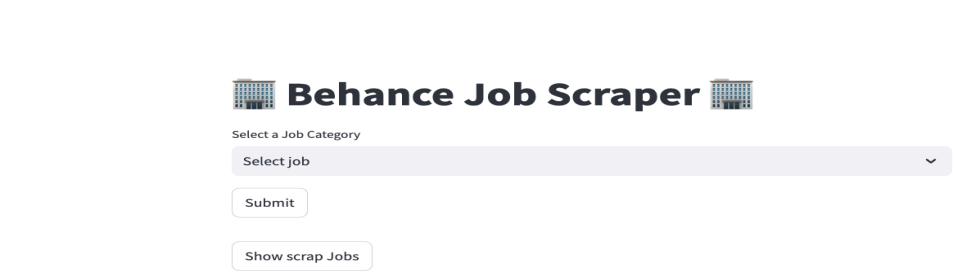
Deals Heaven Scraper

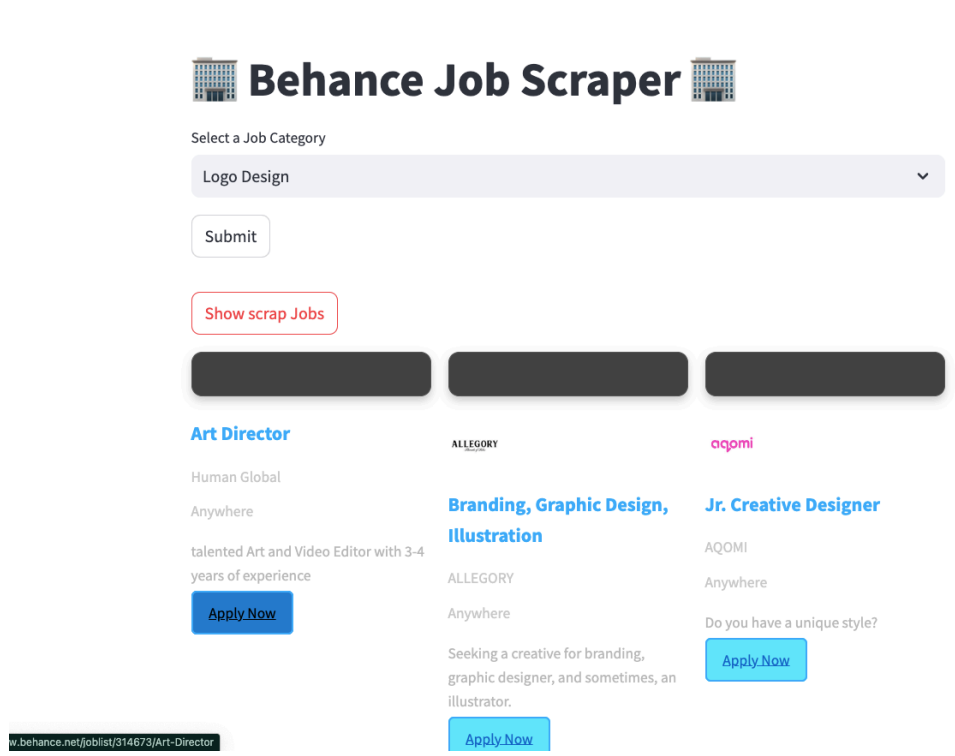
Choose a Store: Flipkart

Choose a Category: All Categories

✅ Data scraping and saving completed.

Scraped Deals


Deploy
Behance Job Scraper
Select a Job Category
Select job
Submit
Show scrap Jobs


Deploy
Behance Job Scraper
Select a Job Category
Logo Design
Submit
Show scrap Jobs
Art Director
Human Global
Anywhere
talented Art and Video Editor with 3-4 years of experience
Apply Now
ALLEGORY
Branding, Graphic Design, Illustration
ALLEGORY
Anywhere
Seeking a creative for branding, graphic designer, and sometimes, an illustrator.
Apply Now
aqomi
Jr. Creative Designer
AQOMI
Anywhere
Do you have a unique style?
Apply Now
w.behance.net/joblist/314673/Art-Director