Web Scraping Project Report

Report Field	Details
Name	Achal Ghate
Date	[25 -11-2025]
Project Goal	The goal of this project is to develop a web scraper that automatically extracts structured data from selected websites , processes the information , and stores it in a usable format for analysis or reporting.
Target Website	https://books.toscrape.com/

1. Abstract

This project aimed to develop a web scraper to systematically extract structured data from Book to Scrape website. The primary objective was to collect key data points, including book tittle , price ,rating, availability status , product description, and category to facilitate market analysis. The project utilized Python programming language with the Requests and Beautiful Soup libraries. The resulting data was stored in a clean, structured [Format, e.g., CSV file] for easy analysis.

2. Introduction

2.1 Problem Statement

Manually collecting large volumes of data from websites is inefficient and prone to human error. This project addresses the need for an automated solution to gather up-to-date real estate information to identify market trends and support informed decision-making in property investment.

2.2 Scope and Constraints

The scraper was designed to target only the main listing pages of the specified website and did not delve into individual property detail pages. The project focused on extracting text-based data and did not handle images or complex media.

2.3 Ethics and Legality

Book to Scrap robots.txt file was reviewed and respected. The scraper was designed to mimic human browsing behaviour by introducing artificial delays between requests to avoid overloading the server or triggering anti-bot measures. The collected data is intended for non-commercial analysis only.

3. Methodology

The scraper operates in a sequence of steps:

1. **URL Request:** An HTTP GET request is sent to the target URL using the requests library.
2. **HTML Retrieval:** The server's response (HTML content) is received.
3. **Parsing:** The raw HTML is parsed into a navigable tree structure using Beautiful Soup.
4. **Extraction:** Specific data points are located and extracted using CSS selectors.
5. **Cleaning & Storage:** Extracted data is cleaned (e.g., removing currency symbols, extra spaces) and appended to a list of dictionaries, which is finally saved as a [File Format] file.

4. Implementation Details

4.1 Tools and Technologies

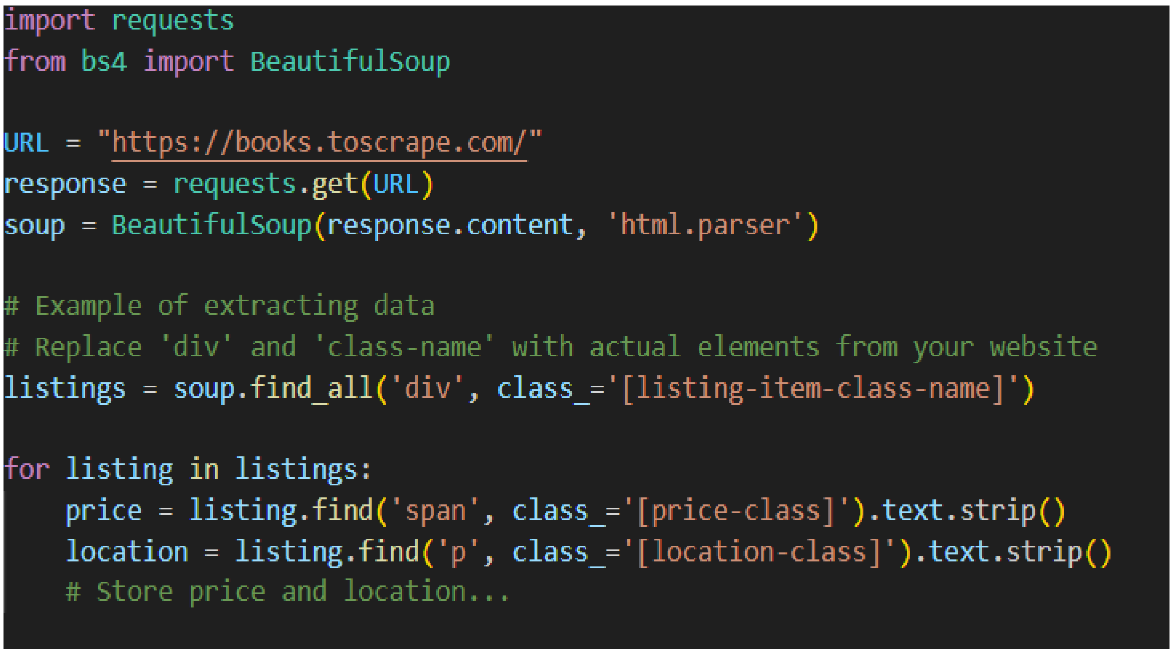
- **Language:** Python 3.14
- **Libraries:** requests, beautifulsoup4, pandas (for data storage)
- **Environment:** [e.g., Jupyter Notebook, VS Code]

4.2 Challenges and Solutions

Challenge	Solution Implemented
Data spread across multiple pages (Pagination)	Implemented a loop that automatically finds the "Next Page" button's URL and continues scraping until no next page is found.
Missing data for certain fields	Implemented try-except blocks to handle cases where an element is missing, defaulting to a placeholder value like "N/A" instead of crashing the script.

4.3 Code Snippet (Example Logic)

Python



```python
import requests
from bs4 import BeautifulSoup

URL = "https://books.toscrape.com/"
response = requests.get(URL)
soup = BeautifulSoup(response.content, 'html.parser')

# Example of extracting data
# Replace 'div' and 'class-name' with actual elements from your website
listings = soup.find_all('div', class_='[listing-item-class-name]')

for listing in listings:
    price = listing.find('span', class_='[price-class]').text.strip()
    location = listing.find('p', class_='[location-class]').text.strip()
    # Store price and location...
```

Use code with caution.

5. Results and Analysis

The project successfully extracted data from 50 pages, resulting in 1000 records of property data. The collected data revealed that the average property price in the target area is [e.g., $350,000]. The data has been saved in a file named books_data.csv.

Title	Price	Rating	Availability	Category
A Light in the Attic	$51.77	3	In stock	Poetry
Tipping the Velvet	$53.74	1	In stock	Historical

6. Conclusion and Future Work

6.1 Conclusion

The project successfully delivered an efficient and robust web scraper that automates the data collection process, meeting all initial objectives.

6.2 Future Enhancements

- **Deeper Scraping:** Modify the script to visit individual property pages to collect richer details (e.g., property descriptions, images).
- **Scheduling:** Integrate a task scheduler (e.g., using PythonAnywhere or Cron jobs) to run the scraper daily and keep the dataset updated automatically.
- **Database Integration:** Store data directly into a SQL database for easier querying and long-term storage.