School of Computer Sciences & Engineering

Department of Computer Science & Application

Field Project Synopsis

On

'Web Scraping'

By

1. **Husain Faeez (027)**
2. **Huzaifa Rampurawala (031)**
3. **Idris Kapasi (042)**

Class & Semester: BCA (3rd SEM)

Under the Guidance of

Prof. Rohit S. Gupta

Academic Year: 2024-25 odd Semester

Web Scraper

1. Abstract

This project delves into the essentials of web scraping using Python, focusing on automating the data extraction process from websites. Using libraries such as Beautiful Soup and Requests, this project constructs a Python script that can systematically gather and store valuable information from web sources. The data scraped can be applied in areas like market analysis, academic research, and personal projects, making it accessible for users needing data that is frequently updated. By the end, this project provides an effective, foundational example of using Python to handle online data collection, bridging the gap between data availability and usability for analysis.

This project explores the essentials of web scraping using Python, focusing on extracting data from websites and managing it efficiently using MySQL. Utilizing libraries like Beautiful Soup and Requests, the project automates data collection, targeting product information such as names, prices, and ratings from Flipkart. Initially stored in a CSV file for preprocessing, the data is subsequently imported into a MySQL database using Python's MySQL connector. This integration ensures a structured, queryable format, enabling advanced analysis and real-time application in areas like market research and trend forecasting. By combining Python’s versatility with MySQL’s data management capabilities, the project bridges the gap between raw web data and actionable insights.

2. Introduction

This project explores the end-to-end workflow of extracting product data from Flipkart using Python, processing it into structured formats like CSV, and subsequently storing it in a MySQL database for further analysis and application. With the ever-growing importance of data-driven insights, this project demonstrates a comprehensive approach to automating data collection, organization, and storage for practical use in various domains, including market analysis, business intelligence, and trend forecasting.

The process begins with web scraping, implemented using the Python library BeautifulSoup. Product details such as names, prices, and ratings are systematically extracted by parsing Flipkart's HTML structure and identifying key elements using CSS selectors. The scraped data is initially saved in a CSV file for verification and preprocessing. This step ensures data accuracy and consistency, making it ready for database insertion.

Next, the focus shifts to integrating MySQL for storing the data in a structured and queryable format. Using Python's MySQL connector, the CSV data is programmatically imported into a MySQL database. This transformation enables efficient storage and retrieval, facilitating complex queries and analyses. The database schema is designed to organize the data logically, with attributes such as product name, price, and rating mapped to specific table columns.

This project not only highlights the technical aspects of data extraction and storage but also emphasizes their applications in real-world scenarios. By transitioning from CSV files to a relational database, the project demonstrates how to scale data management solutions for business-critical tasks.

The report concludes with insights into the challenges faced, including handling large datasets and ensuring seamless data insertion into MySQL. It also provides recommendations for enhancing the pipeline, such as automating periodic data updates and optimizing database queries for faster performance

3. System Architecture

3.1 Input Layer

• URL **Input**: Accepts user input for the URL of the webpage to be scraped.

• HTTP **Requests**: Uses Python's Requests library to establish a connection to the website and retrieve its HTML content.

3.2 Processing Layer

Processing Layer

1. **HTML Parsing**: Uses Beautiful Soup to interpret the HTML, allowing specific elements or data points to be isolated based on HTML tags, IDs, or classes.
2. **Data Filtering**: Filters out only the necessary information by setting criteria, such as CSS selectors or specific tags. This helps in extracting exactly what the user needs without clutter.
3. **Data Cleaning**: Post-extraction, the data is cleaned by removing extra spaces, unwanted tags, or characters that might interfere with data readability or usability.

3.3 Display Layer

- **Button Rendering:** Renders each calculator button on the screen using OpenCV, displaying numbers, operators, and the "Clear" button.
- **Equation & Result Display:** Displays the current equation or result on the screen in real-time. This is updated with each valid button press.

3.4 Output Layer

- Data **Storage**: Saves the retrieved data in a CSV file, providing a structured format for future analysis or integration with other data processing tools.

- Data **Visualization** : Basic charts or tables can be generated to display the data in a graphical format, aiding in a more visual interpretation of the results.

4. Objectives

- **Implement Web Scraping to Extract Data**: Build a functional scraper that collects data by parsing HTML and using tags and classes to locate relevant elements.
- **Learn and Use Beautiful Soup and Requests Libraries**: Understand and utilize these libraries, which are essential for efficiently extracting and processing web data.
- **Filter and Clean Extracted Data**: Ensure that only relevant data points are stored, removing any extraneous content and structuring it into a usable format.
- **Store Extracted Data Efficiently**: Provide an option to save extracted data in various file formats, like CSV or JSON, depending on user requirements.
- **Respect Website Terms and Policies**: Emphasize ethical web scraping, ensuring that requests are infrequent to avoid overloading servers and that all scraping adheres to the site's permissions.
- **Implement Dynamic Web Page Scraping**: Extend the project to scrape data from dynamic web pages that use JavaScript to load content. This could involve using Selenium or another tool to handle JavaScript-rendered elements.
- **Optimize for Efficiency and Speed**: Ensure that the web scraper runs efficiently by minimizing HTTP requests and optimizing parsing techniques, making it capable of handling larger websites without excessive delays.
- **Handle Pagination for Multi-Page Scraping**: Add functionality to handle paginated content, allowing the scraper to automatically navigate through multiple pages to gather all relevant data without manual input.
- **Implement Throttling and Delay Mechanisms**: Integrate mechanisms that add delays between requests to avoid overloading servers and reduce the risk of being blocked by the website, thus ensuring sustainable data extraction practices.
- **Integrate Extracted Data with Databases:** Extend the project by implementing a pipeline to store the extracted and cleaned data in relational databases such as MySQL, enabling seamless querying and data analysis for real-world applications.

5. System Requirements

5.1 Hardware Requirements

- **Processor**: An Intel Core i3 or higher is recommended to handle HTTP requests and process data smoothly.
- **Memory**: At least 4GB of RAM is recommended, though a higher capacity will improve performance, especially when handling large amounts of data.
- **Storage**: Allocate at least 500MB of storage for the necessary packages, Python environment, and output data storage.

6. Software Requirements

- **PyCharm:** PyCharm is an integrated development environment (IDE) specifically designed for Python programming. Developed by JetBrains, PyCharm provides a wide range of tools and features that enhance productivity and streamline the coding process for Python developers.
- **MySQL:** MySQL is an open-source relational database management system (RDBMS) known for its speed, reliability, and flexibility. It uses Structured Query Language (SQL) to manage and query data in tables, ensuring easy organization and retrieval. MySQL supports cross-platform compatibility.

7. Conclusion

This project demonstrates how Python can be utilized to automate the extraction and storage of online data, providing fast, systematic access to web-based information for individuals and businesses requiring regularly updated datasets. By employing web scraping techniques with libraries like Beautiful Soup and Requests, the project efficiently collects structured data, such as product names, prices, and ratings from Flipkart, and integrates it with MySQL for enhanced storage and retrieval capabilities.

The implementation begins with constructing a web scraper that navigates HTML content, identifies relevant elements using tags and classes, and extracts specific data points. Once the data is filtered and cleaned, it is initially stored in a CSV file for verification. The project then transitions to using MySQL, where the cleaned data is transferred into a relational database. This step allows for the organization of data into logical tables, providing users with robust query functionalities for in-depth analysis.

The use of MySQL elevates the project by enabling efficient handling of large datasets. The database is designed to allow seamless storage, indexing, and retrieval, supporting advanced use cases such as trend analysis, market research, and automated reporting. Furthermore, this integration ensures the scalability of the system, making it suitable for handling multiple scraping tasks simultaneously or managing data from numerous URLs.

8. References

8.1 https://www.flipkart.com/

8.2 https://www.youtube.com/watch?v=WHYMSkwJSYU

8.3 https://www.youtube.com/watch?v=hK-qUy3UfT8