PROJECT REPORT
WEB SCRAPPER

Omar ALLAL
Hamza BOUKHATEM
Skander RADHOUANE
Mohamed Ali BEN GHARBIA

Fourth deliverable: Tools and Development phases

Contents

I. Tools used.

Web scraping projects typically involve a combination of programming languages, libraries, and tools:

1. Programming Languages:

Python: Python is a popular language for web scraping due to its simplicity and the availability of various libraries.

2. Web Scraping Libraries:

BeautifulSoup: BeautifulSoup is a Python library for parsing HTML and XML documents. It provides easy-to-use methods for navigating and extracting data from web pages.

Selenium: Selenium is a web testing framework that can also be used for web scraping. It allows you to automate browser interactions and extract data from websites that heavily rely on JavaScript.

3. HTTP Client Libraries:

Requests: Requests is a popular Python library for making HTTP requests. It simplifies the process of sending HTTP requests and handling responses in web scraping projects.

4. Data Parsing and Manipulation:

Pandas: Pandas is a powerful data manipulation library in Python. It provides convenient data structures and functions for cleaning, transforming, and analyzing scraped data.

JSON/XML parsers: Depending on the format of the data you're scraping; you may need libraries like json or xml.etree.ElementTree to parse and extract relevant information from JSON or XML responses.

5. Proxy Tools:(conditional)

Proxies: In some cases, you may need to use proxies to scrape websites without getting blocked or to bypass certain restrictions. Proxy tools like ProxyMesh, Scraper API, or ProxyCrawl can be helpful in managing and rotating proxies.

6. Captcha Solving:(conditional)

CAPTCHA solving services: When dealing with websites that implement CAPTCHA, you may need to use third-party services like 2Captcha or Anti-Captcha to automate the CAPTCHA solving process.

7. Data Storage:

store the scraped data in simple file formats like CSV, JSON, or XML.

II. Project development phases

1. Project Planning:
   - Define the project objectives and requirements.
   - Identify the websites or data sources to scrape.
   - Determine the data fields to extract.
   - Consider legal and ethical aspects of web scraping, such as the website's terms of service and the legality of scraping the targeted data.

2. Data Source Analysis:
   - Analyze the structure and layout of the target website(s).
   - Identify the HTML structure or API endpoints from which the data can be extracted.
   - Determine if any authentication or session management is required.

3. Implementation:
   - Set up the development environment.
   - Choose the programming language and web scraping libraries/tools.
   - Write code to make HTTP requests and retrieve web pages.
   - Parse the HTML/XML responses and extract the desired data using libraries like BeautifulSoup.
   - Handle pagination, dynamic content, or JavaScript-driven pages using tools like Selenium if necessary.
   - Implement data cleaning and transformation routines if needed.

4. Data Storage and Processing:
    - Decide on the storage format, such as databases (MySQL, PostgreSQL, MongoDB) or file formats (CSV, JSON, XML).
    - Set up the database or file system for storing the scraped data.
    - Develop the code to store the extracted data in the chosen format.

5. Visualization and Analysis:
    - Determine the key insights or metrics you want to extract from the scraped data.
    - Select appropriate visualization tools , such as power BI for creating charts, graphs, or interactive visualizations.
    - Preprocess and clean the scraped data, if necessary, to ensure it is in a suitable format for visualization.
    - Generate visual representations of the data, such as line charts, bar graphs, scatter plots, heatmaps, or interactive dashboards.