**Assignment on Web Scraping**

## **Introduction**

Web scraping plays a vital role in automating data collection, particularly in situations where APIs are insufficient, unavailable, or when databases lack comprehensive information. This assignment highlights the importance of web scraping for developers in automating repetitive processes for data extraction.

## **Objective**

The objective of this assignment is to develop a robust web scraping script to enrich a dataset of product part numbers with additional product information like description, category, etc. It focuses on overcoming real-world challenges such as dynamic content loading, cookie handling, error recovery, pagination, and rate-limiting, also highlighting the significance of web scraping in modern software development.

## **Assignment Instructions**

### **1. Prerequisites**

Before starting the assignment, ensure the following tools and libraries are installed in your local virtual environment:

* **Python 3.9 or above**
* **Playwright** for browser automation:
  + Install using **pip install playwright** & Run **playwright install**
* **BeautifulSoup** for HTML parsing:
  + Install using **pip install beautifulsoup4**
* **Pandas** for data manipulation:
  + Install using **pip install pandas**
* **Tenacity** for retry mechanisms:
  + Install using **pip install tenacity**
* Any other library your solution may require.
* Download the input sample\_part\_number.csv [here](https://datagrokranalytics-my.sharepoint.com/:x:/g/personal/anurag_n_datagrokr_com/EeqN6QYSwVhIn1gjc9Zfd44Bn-0P3PkrGScmnPM29ROrWw?e=TQxnMe) – 997 part numbers are there

### **2. Assignment Scope**

Develop a Python-based web scraper that can:

* **Scrape Product Data:**
  + Develop a Python scraper that reads product part numbers from a CSV file and fetches detailed information for each part from a given website, handling dynamic content like cookie consent popups and structured HTML elements (e.g., tables, lists).
* **Download Assets and Handle Errors:**
  + The scraper should download product-related assets (e.g., images, datasheets) if available and implement error handling with retry mechanisms to ensure reliable data extraction.
* **Save and Log Data:**
  + After scraping the necessary product information for all the part numbers, save the results in an Excel file. Also, implement logging to capture errors, retries, and processing status for each product in a log file.

### **3. Detailed Requirements**

### 3.1. Target Website

You will extract product details from a webpage by utilizing the specified URL:

‘https://mall.industry.siemens.com/mall/en/vn/Catalog/Product/{Product Part Number}’

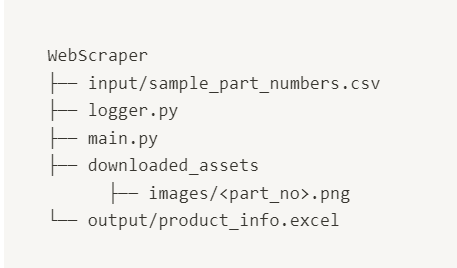
### 3.2. Key Functionalities

1. **Cookie Handling**:
   1. Automatically detect and click the cookie consent popup.
   2. Retry if the popup does not appear or fails to interact.
2. **Navigation**:
   1. Navigate to the main page.
   2. Handle pagination by clicking "Next" until all pages are scraped.
   3. When searching for certain part numbers, you may be redirected to a page containing multiple related parts (a family of parts with different suffixes). In this case, you need to scrape the required information for each individual part number and collect data for all parts in that family.
3. **Data Extraction**:
   1. Parse data from HTML tables or div-based content and extract details like:
      * Article Number (Market Facing Number)
      * Product Description
      * Product family
      * Product Lifecycle (PLM)
      * PLM Effective Date
      * Notes
      * Product Image
4. **Error Handling**:
   1. Implement retry logic to handle failures due to network issues or dynamic page load delays.
   2. Log errors and retries for debugging purposes.
5. **Downloading Assets**:
   1. If the website contains downloadable content (e.g., images) related to the product, implement logic to download them and save them locally with filenames as the part number.
6. **Output**:
   1. Save the extracted data in a well-structured Excel file.
   2. Store the product images under downloaded\_assets/images/ directory.

### **4. Steps to Complete the Assignment**

### Step 1: Setup and Initialization

* Install all necessary libraries.
* Create a folder structure for the project:



### Step 2: Website Cookie Handling

* Identify common selectors for cookie popups (Accept, Allow All, etc.).
* Write a function that clicks these buttons using Playwright.
* Test this on the target website.

### Step 3: Page Navigation

* Write a function to navigate through multiple pages.
* Ensure the function waits for all elements to load before proceeding to the next page.

### Step 4: Data Parsing

* Use BeautifulSoup to extract data from HTML tables or other structured elements.
* Parse the data into a dictionary or Pandas DataFrame for further processing.

### Step 5: Downloading Assets

* If the page contains downloadable items (e.g., images) and save the items in downloaded\_assets/.

### Step 6: Error Handling and Logging

* Use the **Tenacity** library to retry failed actions (e.g., navigation, scraping).
* Log all activities and errors into logs/ with timestamps for debugging.

### Step 7: Save Data

* Save the extracted data in an Excel file under output/ folder.

### **6. Deliverables**

1. **Python Script**: A standalone script that meets the assignment's objectives.
2. **Logs**: Logs showing the status of the progress and error handling for each part number.
3. **Scraped Data**:
   1. Excel file with the extracted information.
   2. Product images.
4. **Documentation**: Detailed instructions for setting up and running the pipeline.