**Lab Exercise 4 - Using BeautifulSoup for Web Scraping for RPA (Robotic Process Automation)**

**Objective**

In this exercise, you'll learn to use BeautifulSoup, a popular Python library, to scrape web data that can be used for automating tasks in RPA workflows.

**Objective**

* Scrape job postings from a job listing website.
* Extract useful information like job titles, company names, and locations.
* Save the extracted data in a CSV file for further use in an RPA process.

**Setup**

1. Install required libraries:

pip install requests beautifulsoup4 pandas

1. Import the necessary modules:

import requests

from bs4 import BeautifulSoup

import pandas as pd

**Steps**

**1. Identify a Target Website**

Choose a website with job postings. For this example, we'll use a hypothetical URL (replace it with an actual site, like a sample job board).

**2. Send an HTTP Request**

Use the requests library to fetch the HTML content of the webpage.

**3. Parse HTML Content**

Use BeautifulSoup to parse and navigate through the HTML.

**4. Extract Relevant Data**

Extract job titles, companies, and locations using appropriate HTML tags and classes.

**5. Save Data**

Store the extracted data in a structured format like a CSV file.

**Code Example**

import requests

from bs4 import BeautifulSoup

import pandas as pd

# URL of the job listing page (e.g., We Work Remotely)

url = "https://weworkremotely.com/remote-jobs"

# Send an HTTP request to the URL

response = requests.get(url)

# Check if the request was successful

if response.status\_code == 200:

# Parse the HTML content using BeautifulSoup

soup = BeautifulSoup(response.text, 'html.parser')

# Extract job postings

jobs = []

# Each job is in a 'section' with class 'jobs'

for job\_section in soup.find\_all('section', class\_='jobs'):

for job\_card in job\_section.find\_all('li', class\_='feature'):

# Extract job title

job\_title = job\_card.find('span', class\_='title').text.strip()

# Extract company name

company\_name = job\_card.find('span', class\_='company').text.strip()

# Extract job location (if available)

location\_tag = job\_card.find('span', class\_='region')

location = location\_tag.text.strip() if location\_tag else "Remote"

# Append to the jobs list

jobs.append({

'Job Title': job\_title,

'Company': company\_name,

'Location': location

})

# Convert to a DataFrame

df = pd.DataFrame(jobs)

# Save to CSV

df.to\_csv('job\_listings.csv', index=False)

print("Job listings have been saved to 'job\_listings.csv'")

else:

print(f"Failed to fetch the webpage. Status code: {response.status\_code}")

**Output**

The script will create a CSV file (job\_listings.csv) with the following structure:

| **Job Title** | **Company** | **Location** |
| --- | --- | --- |
| Software Engineer | Tech Corp | Remote |
| Data Scientist | DataWorks Inc. | Remote |

**RPA Use Cases for the Extracted Data**

1. **Automated Reporting:**
   * Create and share job market analysis reports with stakeholders.
2. **Email Automation:**
   * Use the CSV file to send automated application emails for specific roles.
3. **Data Ingestion:**
   * Integrate the data into an RPA pipeline for further processing or analysis.