**Weekly Report**

Submitted By: Eman Shahid

**Topic**: Web Scraping Bloomberg with Selenium and BeautifulSoup

**Overview:**

This week, I created a Python-based web scraper to extract article information from the Bloomberg homepage. Since Bloomberg loads content dynamically using JavaScript, I used a combination of **Selenium** for page rendering and **BeautifulSoup** for HTML parsing. The scraper focuses on collecting article titles, URLs, and publication dates.

**Libraries Used:**

* **Selenium**: For rendering dynamic content (JavaScript-heavy)
* **BeautifulSoup**: For parsing and extracting structured HTML data
* **webdriver\_manager**: For automated ChromeDriver setup

**Approach:**

1. Opened Bloomberg homepage in **headless Chrome** (no UI).
2. Waited 5 seconds for the full page to load.
3. Parsed the rendered HTML using **BeautifulSoup**.
4. Collected:
   * Article **title** (h1, h2, h3)
   * Article **URL** (<a> tag)
   * **Date** (from <time> tag or fallback attribute)

**Code:**

from selenium import webdriver

from selenium.webdriver.chrome.service import Service

from selenium.webdriver.chrome.options import Options

from webdriver\_manager.chrome import ChromeDriverManager

from bs4 import BeautifulSoup

import time

def scrape\_bloomberg\_combined\_with\_date():

print(" Scraping Bloomberg with Selenium + BeautifulSoup (including Date)...")

options = Options()

options.add\_argument("--headless")

options.add\_argument("--disable-gpu")

options.add\_argument("--no-sandbox")

options.add\_argument("--window-size=1920,1080")

options.add\_argument("user-agent=Mozilla/5.0")

driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()), options=options)

try:

driver.get("https://www.bloomberg.com")

time.sleep(5)

soup = BeautifulSoup(driver.page\_source, 'html.parser')

articles = soup.find\_all('article')

print(f"Found {len(articles)} articles.\n")

for i, article in enumerate(articles, start=1):

title\_tag = article.find(['h1', 'h2', 'h3'])

title = title\_tag.get\_text(strip=True) if title\_tag else "No Title"

link\_tag = article.find('a', href=True)

link = link\_tag['href'] if link\_tag else "#"

full\_link = link if link.startswith("http") else f"https://www.bloomberg.com{link}"

date = "Date not found"

time\_tag = article.find('time')

if time\_tag:

if time\_tag.has\_attr('datetime'):

date = time\_tag['datetime']

else:

date = time\_tag.get\_text(strip=True)

else:

data\_time = article.get('data-time')

if data\_time:

date = data\_time

print(f" Article {i}:")

print(f" Title: {title}")

print(f" URL: {full\_link}")

print(f" Date: {date}\n")

print(" Finished scraping Bloomberg with date info.")

except Exception as e:

print(f" Error: {e}")

finally:

driver.quit()

if \_\_name\_\_ == "\_\_main\_\_":

scrape\_bloomberg\_combined\_with\_date()

**Learning:**

* Gained practical experience combining Selenium and BeautifulSoup to handle JavaScript-heavy websites.
* Learned how to extract structured content from <article> tags.
* Handled optional/missing fields (like dates) gracefully using fallback checks.
* Understood the role of user-agent strings and headless browser settings in avoiding bot detection**.**

**Questions :**

1. Is my fallback method for missing publication dates acceptable, or should I introduce logging or default placeholders?
2. Would you suggest writing the scraped output to a file (like CSV or JSON), or is printing to console fine for now?
3. Should I replace time.sleep(5) with Selenium’s WebDriverWait for better loading reliability?