**Automated Web Scraping for Industry News Monitoring**

**Surya Sanjeeva Pravarsha Erodula**

**Project Overview:**

In today’s fast-paced industry landscape, staying updated with the latest news is crucial. However, **manually checking** multiple websites, filtering articles, and reading through irrelevant content **consumes valuable time**. To automate and optimize this process, I have developed a Python-based **web scraping solution** that extracts relevant articles from multiple industry news websites. Filters news using a predefined list of industry keywords (LNG, Offshore, Drilling, etc.). **Eliminates manual effort**, allowing employees to focus on critical decision-making instead of searching for news. Provides structured output with article titles and direct links.

**Future Enhancements:**

* Retrieve only the past 10 days news for up-to-date insights.
* Implement AI-based article summarization for quick information retrieval.  
  Code Deployment on GitHub:
* The entire project will be hosted on GitHub, ensuring code persistence even if the local system is lost or damaged.

This automated pipeline transforms the way industry news is monitored, making it faster, smarter, and more efficient.

**Key Business Benefits:**

|  |  |
| --- | --- |
| **Benefit** | **Impact on Company** |
| Significant Time Savings | Eliminates hours of manual work spent browsing multiple news sites. |
| Increased Productivity | Employees can focus on decision-making instead of searching for news. |
| Real-Time Monitoring | Ensures only the latest and most relevant articles are captured. |
| Eliminates Redundant Reading | Articles are pre-filtered, reducing the time wasted on irrelevant content. |
| Automated Summarization (Future) | Will provide quick insights instead of requiring full article reading. |
| Scalability | Can be expanded to cover more websites and industries as needed. |
| Secure Code Storage on GitHub | Ensures the project is never lost, even if the local system fails. |

**Work Completed:**

**Initial Web Scraping Implementation**

* Developed a Python-based scraper using Selenium WebDriver and BeautifulSoup.
* Implemented a robust mechanism to extract headlines and article URLs.
* Ensured correct navigation across multiple news websites.

**Keyword-Based Filtering**

* Created a comprehensive keyword list, including:
  + LNG, FID, Offshore, Shell, Transocean, Floating Wind, Pipelay Vessel, etc.
* The scraper only extracts and displays relevant articles, eliminating unrelated news.

**Handling Web Scraping Challenges**

|  |  |
| --- | --- |
| **Challenge** | **Solution Implemented** |
| Lazy-Loading Websites | Added scrolling functionality to load hidden elements. |
| Stale Element Errors | Implemented retry logic to refresh elements if they disappear. |
| Slow-Loading Websites | Increased explicit wait time and improved error handling. |
| Extracting Duplicate URLs | Added deduplication logic to ensure clean results. |
| Websites Blocking Automation | Used headers & user-agent spoofing to mimic real browsing. |

**Multi-Site Scraping Implementation**

The scraper currently supports multiple sources, ensuring a wide coverage of industry news. The targeted websites include:

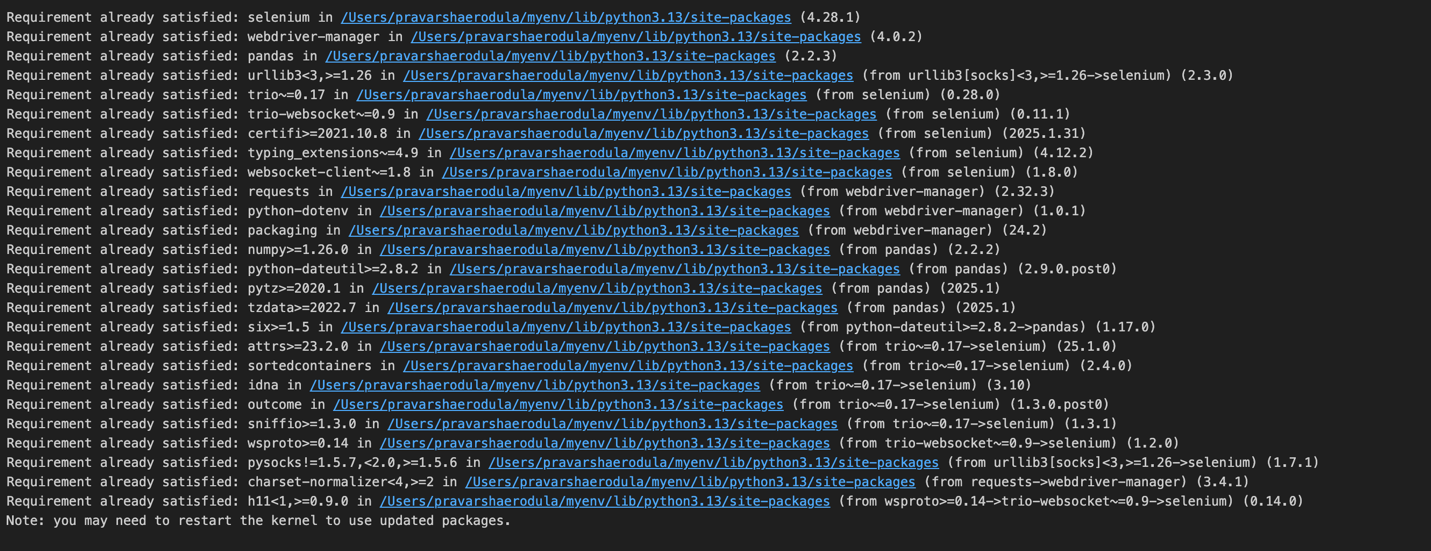
1. Upstream Online
2. Offshore Wind
3. Offshore Energy
4. gCaptain
5. OE Digital
6. Maritime Executive
7. MarineLink
8. TradeWinds News

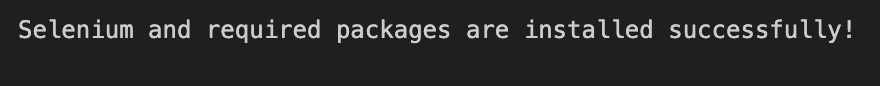
The script extracts, filters, and formats the latest articles from these sources.

**Performance Optimization**

* Enabled headless browsing for faster execution.
* Improved page load handling to accommodate slow or JavaScript-heavy sites.
* Optimized error handling for seamless execution across all websites.

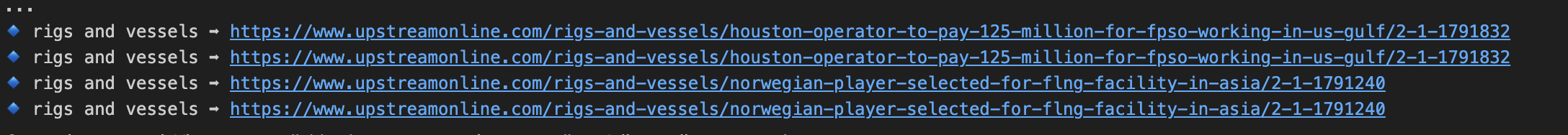
**Output:** (There are many trials and error codes and outputs. Only attaching those that have accurate data)

****

****

**A screenshot of a computer

AI-generated content may be incorrect.**

****

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A black and blue text

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

Only relevant articles are displayed, reducing manual reading efforts.

**Technical Requirements**

**Python Version**

* Python 3.8+ (Recommended: Python 3.10+)

**Required Python Libraries (Dependencies)**

Run the below command in terminal and install packages

**pip install selenium beautifulsoup4 requests webdriver-manager python-docx transformers**

|  |  |  |
| --- | --- | --- |
| **Dependency** | **Version (Recommended)** | **Purpose** |
| selenium | 4.10.0+ | Automates browser interactions for web scraping |
| beautifulsoup4 | 4.12.2+ | Parses and extracts HTML data from websites |
| requests | 2.31.0+ | Fetches website content for scraping |
| webdriver-manager | 4.0.1+ | Automatically manages WebDriver for Selenium |
| python-docx | 0.8.11+ | Exports extracted data into Word documents |
| transformers | 4.29.2+ | Used for AI-based text summarization (Future Enhancement) |

**Web Drivers (For Selenium)**

* Google Chrome
* Chrome WebDriver (managed automatically using webdriver-manager)

**Tools for Development & Debugging**

* Jupyter Notebook (if running interactively): pip install notebook
* VS Code / PyCharm (Recommended for debugging and editing)

**Future Work**

**Implementing Date Filtering**

* Modify the script to only retrieve articles from the past 10 days.
* Ensures only the most recent news is captured.

**Article Summarization**

* Implement AI-based summarization to condense articles into key insights.
* Possible approaches:
  + Use Hugging Face Transformers (e.g., BART, T5).
  + Utilize spaCy or Gensim for extractive summarization.
* Expected Benefit: Saves employees from reading full articles by providing concise summaries.

**Storing & Exporting Data**

* Store extracted data in CSV/Excel databases.

**Deployment on GitHub**

* The entire project will be hosted on GitHub, ensuring:
  + Code persistence even if the local system crashes.
  + Version control for future updates and improvements.
  + Collaboration opportunities for multiple contributors.

**Potential Use Cases**

|  |  |
| --- | --- |
| **Scenario** | **How This Solution Helps** |
| Company executives need quick news updates | Provides only the most relevant news, eliminating unnecessary reading. |
| Analysts spend hours manually searching | Automates this process, saving time and effort. |
| Some websites block automation | Uses error handling & user-agent spoofing for smooth extraction. |
| Need for daily news reports | Can be scheduled to run automatically every morning. |
| Employees waste time on unrelated articles | Ensures only relevant articles are extracted based on keywords. |

**Conclusion**

This project automates industry news extraction, saving time and improving efficiency. Future enhancements will include date-based filtering and AI summarization for better insights. The code will be deployed on GitHub to ensure persistence and prevent data loss. Additionally, extracted data can be automatically formatted into tables in Word using Python, making reports structured and easy to analyze. These improvements will make the system faster, smarter, and highly efficient for real-time news monitoring.