Python Automation Assignment

Submitted by: Bhavya Srivastava

bhavya.srivastava1400@gmail.com

Introduction:

The objective of this Assignment is to gather and analyse data from senior living operators and community websites using Python-based web scraping techniques.

Approach:

The assignment is developed using Python as the core programming language, along with the following libraries and tools:

1. BeautifulSoup: HTML/XML parsing
2. Selenium: Web browser automation
3. Pandas: Data manipulation and analysis
4. webdriver_manager: Automated management of browser drivers

The website selected for scraping: "https://www.ashianahousing.com/senior-living-india" (provides information about various senior living projects across India.)

Output File : **senior_living_projects.csv**

Development Process:

1. Ensuring the site allows web scraping by checking its robots.txt file
2. Importing Required Libraries:_ The code begins with importing the necessary libraries, including requests, BeautifulSoup, pandas, selenium, and webdriver_manager.
3. Initialising Selenium WebDriver and Setup Chrome WebDriver: A function setup_driver() is created to initialise the Chrome WebDriver using Selenium. The webdriver_manager library is used to automatically handle the installation and management of the Chrome WebDriver.
4. Scraping Website Data: The scrape_website() function is implemented to scrape the website data. It utilises Selenium to navigate to the specified URL and extract relevant information, such as project name, address, price range, and project details URL(to fetch more information regarding the property such as Amenities). The extracted data is stored in a list of dictionaries.
5. Fetching Amenities: The fetch_amenities() function is developed to fetch the amenities for each project. It navigated to the project details page using Selenium and extracted the list of

amenities using BeautifulSoup. The amenities are then added to the corresponding project dictionary.

6. Cleaning Data and Exporting to CSV File: The clean_and_export_data() function is responsible for cleaning and formatting the scraped data. It utilises the pandas library to create a DataFrame, perform data cleaning operations (e.g., stripping whitespace, handling missing values), and export the cleaned data to a CSV file named "**senior_living_projects.csv**".

Challenges and Solutions:

During the development process, several challenges were encountered, and appropriate solutions were implemented:

1. Handling Dynamic Content: Some websites use JavaScript to dynamically load content, which can be challenging for web scraping. To overcome this, the assignment utilises Selenium, a web browser automation tool, to render the JavaScript and access the dynamic content.
2. Parsing HTML Structure: The HTML structure of the website varied across different sections, requiring careful analysis and custom parsing techniques. BeautifulSoup, a powerful HTML/XML parsing library, is employed to navigate and extract the desired data efficiently.
3. Error Handling: To ensure robust and reliable scraping, implemented error handling mechanisms. Exceptions were caught, and appropriate error messages were displayed to provide better visibility into potential issues during the scraping process.
4. Data Cleaning: The scraped data often required cleaning and formatting to ensure consistency and usability. The pandas library is utilised to perform data cleaning operations, such as handling missing values, stripping whitespace, and converting data types.

Main Highlights :

The Python code developed for this web scraping assignment adheres to industry best practices

1. Modular Design: Code is divided into distinct, reusable functions for better maintainability and extensibility.
2. Robust Error Handling: The code implements robust error handling mechanisms, ensuring that exceptions are caught and appropriate error messages are displayed. This enhances the reliability and stability of the scraper, making it more resilient to potential issues during the scraping process.
3. Efficient Data Extraction: The code leverages BeautifulSoup and Selenium to efficiently navigate and parse web pages, handling dynamic content and extracting structured data with precision.

4. Comprehensive Documentation: The code is well-documented, with clear comments and explanations throughout the codebase.
5. Scalability and Extensibility: The code is designed with scalability and extensibility in mind. The modular structure allows for easy addition of new features and adaptability.
6. Data Cleaning and Formatting: The code incorporates robust data cleaning and formatting techniques using pandas for consistency and usability.

Learning Potential and Future Enhancements:

To enhance the performance, scalability, and reliability, the following potential improvements could be considered:

1. Parallel Processing: Implement parallel processing techniques to scrape multiple websites or pages simultaneously, improving overall performance and efficiency.
2. Handling Captchas and Anti-Scraping Measures: Develop strategies to handle captchas and anti-scraping measures employed by some websites to prevent automated scraping.
3. Scheduling and Automation: Integrate the scraper into a scheduled task or pipeline to periodically update the scraped data, ensuring the availability of up-to-date information.
4. Error Logging and Monitoring: Implement comprehensive error logging and monitoring mechanisms to track and analyse errors, enabling proactive troubleshooting and maintenance.
5. User Interface and Reporting: Develop a user-friendly interface or reporting system to display the scraped data in a more accessible and visually appealing manner.

Conclusion:

This Python web scraping assignment has been an invaluable learning experience, allowing me to deepen my understanding of web scraping techniques, modular Python programming, error handling, data cleaning, and working with libraries like BeautifulSoup, Selenium, and pandas. I encountered challenges that tested my problem-solving abilities and prepared me for tackling complex problems in the future.