**Web Scraping and Data Extraction Documentation**

**### Objective**

The objective of this script is to perform web scraping on a public register website to extract information related to applications within a specified date range. The extracted data is then stored in a Pandas DataFrame and saved to an Excel file.

**### Components**

**1. \*\*Web Driver Setup:\*\***

- The script uses Selenium with the Chrome web driver to automate browser interactions.

- The Chrome browser is initiated, and the target website is opened.

**2. \*\*Date Range Input:\*\***

- The script inputs a specified date range (from\_date to to\_date) into the search form on the web page.

**3. \*\*Search and Data Extraction:\*\***

- After entering the date range, it clicks the search button and enters a loop to navigate through multiple pages of search results.

- The BeautifulSoup library is used to parse the HTML content of each page.

- Relevant information (application number, type, parent application number, submitted date, and property address) is extracted from the table on each page.

**4. \*\*Pagination Handling:\*\***

- The script checks for the presence of a "Next" button and clicks it to navigate to the next page of results.

- The loop continues until there are no more pages to scrape.

**5. \*\*Data Storage:\*\***

- Extracted data is stored in a list of dictionaries.

- A Pandas DataFrame is created from the list of dictionaries.

- The DataFrame is printed to the console for debugging purposes.

- The DataFrame is saved to an Excel file named "output\_data\_jan01\_dec31\_2017.xlsx".

**6. \*\*Error Handling:\*\***

- The script includes exception handling for ElementClickInterceptedException and TimeoutException.

- If the "Next" button is not immediately clickable, it waits for a blocking element to disappear.

- If the "Next" button is not found within the expected time, a timeout exception is caught.

**7. \*\*WebDriver Closure:\*\***

- Finally, the web driver is closed.

**### Usage**

- Users should ensure they have the necessary Python libraries installed (Selenium, BeautifulSoup, Pandas).

- The Chrome web driver should be compatible with the installed Chrome browser.

- Adjustments to the XPaths may be needed if the website structure changes.

**### Notes**

- The script employs explicit waits to handle dynamic loading of elements on the web page.

- Sleep intervals are added to allow time for the next page to load.

- The script may need adjustments if the structure or behavior of the target website changes.