Selenium Web Scraping Project: Python Dependencies and Installation Guide

This document outlines the Python libraries required for your Selenium web scraping project and provides the necessary installation commands.

Installation Commands

To install the essential libraries, open your terminal or command prompt and run the following commands:

- **Selenium**: This is the core library for browser automation.

pip install selenium

- **time**: This is a built-in Python module, so no separate pip install command is needed.
- **json**: This is a built-in Python module, so no separate pip install command is needed.
- **csv**: This is a built-in Python module, so no separate pip install command is needed.

Note on ChromeDriver: In addition to the Python libraries, you'll need **Google Chrome** installed on your system and the appropriate **ChromeDriver** executable. Selenium uses ChromeDriver to control the Chrome browser. Your script assumes chromedriver is in your system's PATH. If it's not, you'll need to download it manually from the official ChromeDriver website and either place it in a directory included in your system's PATH or specify its location directly in the webdriver.Chrome() call within your code.

Explanation of Libraries Used

- **selenium**: This powerful library enables your script to automate web browsers.
  - webdriver: Provides the interface to control web browsers like Chrome.
  - By: Used to locate elements on a web page using various strategies (e.g., by XPath, CSS selector, ID).
  - WebDriverWait: Allows your script to pause execution until a specific condition is met on the web page, which is crucial for handling dynamically loaded content.
  - expected_conditions as EC: Contains a set of common conditions used with WebDriverWait (e.g., waiting for an element to be clickable or visible).
  - TimeoutException, ElementClickInterceptedException, StaleElementReferenceException: These are specific exceptions caught by your script to handle common issues that arise during web automation, making your scraper more robust.
- **time**: A standard Python library used for adding delays in your script (time.sleep()). These delays can be important for allowing web page elements to load fully or to mimic more human-like Browse behavior.
- **json**: A standard Python library for working with **JSON (JavaScript Object Notation)** data. Your script uses this to save the scraped project details into a human-readable and machine-parseable .json file.
- **csv**: A standard Python library for handling **CSV (Comma Separated Values)** files. Your script utilizes this to export the scraped data into a .csv format, which is easily opened and analysed.