**Selenium Automation Testing Project using Python**

**Overview**

This project aims to automate the testing of a web application using **Selenium** and **Python**. The goal is to test various functionalities of a web application, such as **Login**, **Signup**, **Add to Cart**, **Place Order**, **Contact Us**, and other core features, ensuring they work as expected. This project adheres to **best practices in automation**, with a focus on **modularity**, **reusability**, and **clean coding practices**.

**Project Features**

**1. Automation Scripts**

* Test automation is done using **Selenium WebDriver**.
* Python scripts control the browser to simulate real-world user interactions.
* The automation scripts are modular, reusable, and designed to be easy to maintain.

**2. Test Scenarios**

* Test scenarios are based on critical functionalities of the website:
  + **Signup**
  + **Login**
  + **Add to Cart**
  + **Place Order**
  + **Contact Us**

**3. Error Handling**

* The automation scripts handle unexpected alerts, pop-ups, and errors.
* Selenium waits for elements to appear and interact with them dynamically to prevent errors during execution.

**4. Reporting**

* **Test Reports**: HTML or other formats that provide an overview of test execution results.
* **Screenshot Capturing**: Screenshots are taken at critical steps or failures for visual evidence.
* **Traceability Matrix**: A mapping document connecting the features, test scenarios, manual test cases, and automated test scripts.

**Steps to Run the Project**

1. **Set up the Environment:**
   * Install **Python 3.x** from the official website: [python.org](https://www.python.org).
   * Install the required dependencies: pip install selenium webdriver-manager
2. **Install WebDriver:**
   * Use **webdriver-manager** to automatically manage the correct version of the WebDriver.
3. **Run Test Scripts:**
   * Test scripts are written to test different features such as signup, login, add to cart, and more.
   * To run a test script, use the command: python test\_script\_name.py
4. **Reports and Logs:**
   * After running the tests, you will get reports in an HTML format, showing the **Pass/Fail** status of each test.
   * Screenshots will be saved to the Screenshots/ directory.

**Details of Test Scenarios**

**1. Signup Test:**

* The script automates the signup process, ensuring the user can register successfully with unique credentials.
* If the user already exists, an appropriate alert message will appear.

**2. Login Test:**

* The script tests the login functionality by entering valid credentials and verifying that the user is logged in successfully.
* Invalid login attempts are also tested.

**3. Add to Cart Test:**

* The test automates the process of adding products to the cart.
* It verifies that the cart is updated after each product addition.

**4. Place Order Test:**

* The script automates the order placement process, ensuring the user can select products, fill out order details, and successfully place an order.

**5. Contact Us Test:**

* The script verifies that the "Contact Us" form is functional by filling out and submitting the form.

**Best Practices Followed**

1. **Modular Code Design**: Each functionality (e.g., login, signup, add to cart) is encapsulated in its own function for better readability, reusability, and maintainability.
2. **Explicit Waits**: Explicit waits are used to ensure elements are present and interactable before performing actions, reducing the likelihood of test failures due to page load times.
3. **Error Handling**: Proper handling of alerts, pop-ups, and unexpected issues during test execution.
4. **Reporting and Logging**: Detailed reports of test results, along with screenshots for failed tests, to assist in debugging and verification.
5. **Reuse of Code**: Functions such as login, logout, and taking screenshots are reused across multiple test scripts to maintain DRY (Don’t Repeat Yourself) principles.

**Final Remarks**

This project automates core features of a web application using **Selenium** and **Python**, ensuring the application behaves as expected. By adhering to industry best practices, including **modular coding**, **reusability**, and **reliable error handling**, the automation scripts are designed for scalability and maintenance.