**Lab Assignment 1: Getting Started with Selenium WebDriver**

**Objective**

* Set up the automation environment.
* Develop a basic Selenium script to launch a browser, navigate to a URL, and interact with elements.
* Document the setup and any challenges faced.

**Tasks**

1. **Environment Setup (6 Marks)**
   * **Installation & Configuration (4 Marks):**
     + Install your preferred IDE.
     + Set up Selenium WebDriver libraries (via Maven/Gradle for Java, pip for Python, etc.).
     + Install and configure the appropriate browser driver (e.g., ChromeDriver or GeckoDriver).
   * **Verification (2 Marks):**
     + Run a simple command/script to verify that Selenium and the driver are working correctly.
2. **Basic Script Creation (12 Marks)**
   * **Script Requirements (8 Marks):**
     + Write a script that launches a browser and navigates to a specified URL (e.g., https://www.example.com).
     + Print the page title and current URL to the console.
     + Close the browser session gracefully.
   * **Element Interaction (4 Marks):**
     + Locate at least one web element (e.g., a search box) using a locator (ID, Name, etc.).
     + Perform a basic interaction (e.g., send keys) and print an attribute value (e.g., placeholder text).
3. **Documentation and Reflection (12 Marks)**
   * **Write-Up (8 Marks):**
     + Provide a detailed write-up that explains each step of the environment setup.
     + Describe the challenges encountered during configuration and how you resolved them.
   * **Code Comments & Clarity (4 Marks):**
     + Ensure your script is well-commented and follows good coding practices.

**Lab Assignment 2: Implementing Functional Test Cases**

**Objective**

* Develop and automate functional test cases using Selenium.
* Practice using various locators and simulate common web application flows.
* Document your approach and outcomes.

**Tasks**

1. **Web Element Locators & Interactions (10 Marks)**
   * **Locator Diversity (6 Marks):**
     + Use at least three different locator strategies (ID, CSS Selector, XPath, etc.) to identify elements.
     + Demonstrate interactions such as clicking a button, entering text, and selecting from a dropdown.
   * **Explanation (4 Marks):**
     + Provide inline comments or a separate document explaining why each locator was chosen.
2. **Automating a Login Flow (10 Marks)**
   * **Test Case Development (6 Marks):**
     + Automate a login process on a demo site (e.g., https://the-internet.herokuapp.com/login).
     + Include steps: open the login page, enter valid credentials, submit the form.
   * **Validation (4 Marks):**
     + Verify a successful login by asserting the presence of a logout button or a success message.
3. **Data Entry and Form Submission (10 Marks)**
   * **Form Automation (6 Marks):**
     + Write a script that navigates to a sample form page.
     + Enter data into at least three input fields (e.g., name, email, message).
     + Submit the form.
   * **Result Verification (4 Marks):**
     + Verify the form submission by checking for a confirmation message or change in page content.
     + Document your test case flow and include screenshots if possible.

**Lab Assignment 3: Assertions and Test Reporting**

**Objective**

* Integrate a test framework with Selenium.
* Use assertions to validate test outcomes.
* Generate a test report to summarize the test results.

**Tasks**

1. **Test Framework Integration (8 Marks)**
   * **Setup (5 Marks):**
     + Integrate a test framework suitable for your language (JUnit/TestNG for Java, PyTest/unittest for Python).
     + Organize your tests into proper test methods or classes.
   * **Basic Test Structure (3 Marks):**
     + Ensure at least one test method is created that encapsulates a specific test case.
2. **Using Assertions (12 Marks)**
   * **Assertion Implementation (8 Marks):**
     + Write at least two test methods using assertions:
       - Example 1: A login test that asserts the appearance of a success message.
       - Example 2: A form submission test that asserts a confirmation message is displayed.
   * **Test Coverage & Accuracy (4 Marks):**
     + Demonstrate that your assertions accurately verify the expected outcomes.
     + Include error handling where necessary.
3. **Test Report Generation and Documentation (10 Marks)**
   * **Report Generation (6 Marks):**
     + Configure the test framework to produce a report (HTML, XML, or console-based report).
     + Include details such as pass/fail status and any error messages.
   * **Documentation and Reflection (4 Marks):**
     + Provide a brief report on how the test reports help in continuous integration and debugging.
     + Explain how assertions improve test reliability.

**Submission Guidelines for All Assignments**

* **Source Code:** Submit all script files organized by assignment.
* **Documentation:** Include a separate document or README with each assignment detailing:
  + Environment setup
  + Step-by-step explanations
  + Screenshots or output logs, if applicable
* **Test Reports:** For Assignment 3, attach the generated test report.