Heliverse

Assignment:

The demo e-commerce site "http://automationpractice.com/" and set up an automation testing framework using Selenium.

1. Test Setup
   a. Web Application: Automation Practice
   b. Framework: Selenium with Java
   c. Dependencies:
      - Selenium WebDriver
      - TestNG
      - WebDriverManager (for managing browser drivers)
      - Allure (for reporting)

   d. Configuration:

      i. Install Java Development Kit (JDK)
      ii. Set up Maven or Gradle for dependency management
      iii. Install an IDE like IntelliJ IDEA or Eclipse
2. Test Cases
   a. Functional Test
      - Scenario: User searches for a product, adds it to the cart, and proceeds to checkout.

Java code:

```
import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;
```

```java
import io.github.bonigarcia.wdm.WebDriverManager;
import org.testng.Assert;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;

public class FunctionalTest {
    WebDriver driver;

    @BeforeClass
    public void setup() {
        WebDriverManager.chromedriver().setup();
        driver = new ChromeDriver();
        driver.get("http://automationpractice.com/");
    }

    @Test
    public void testAddToCart() {
        WebElement searchBox =
driver.findElement(By.id("search_query_top"));
        searchBox.sendKeys("dress");
        searchBox.submit();

        WebElement product =
driver.findElement(By.cssSelector(".product_img_link"));
```

```
        product.click();

        WebElement addToCartButton = 
driver.findElement(By.id("add_to_cart"));
        addToCartButton.click();

        WebElement proceedToCheckoutButton = 
driver.findElement(By.cssSelector(".button-medium[title='Proceed to 
checkout']"));
        proceedToCheckoutButton.click();

        WebElement cartSummary = 
driver.findElement(By.id("cart_summary"));
        Assert.assertTrue(cartSummary.isDisplayed(), "Cart summary is 
not displayed");
    }

    @AfterClass
    public void teardown() {
        driver.quit();
    }
}
```

b. Login Test
   - Scenario: Automate the login process.

Java Code

```
@Test
public void testLogin() {
    WebElement signInButton =
driver.findElement(By.className("login"));
    signInButton.click();

    WebElement emailField = driver.findElement(By.id("email"));
    WebElement passwordField =
driver.findElement(By.id("passwd"));
    WebElement submitButton =
driver.findElement(By.id("SubmitLogin"));

    emailField.sendKeys("valid_email@example.com");
    passwordField.sendKeys("valid_password");
    submitButton.click();

    WebElement accountPage =
driver.findElement(By.className("account"));
    Assert.assertTrue(accountPage.isDisplayed(), "Login failed with
valid credentials");
```

```
    // Test invalid credentials
    emailField.clear();
    passwordField.clear();
    emailField.sendKeys("invalid_email@example.com");
    passwordField.sendKeys("invalid_password");
    submitButton.click();

    WebElement errorMessage =
driver.findElement(By.cssSelector(".alert-danger"));
    Assert.assertTrue(errorMessage.isDisplayed(), "Error message not
displayed for invalid credentials");
}
```

c. UI Test
   - Scenario: Verify the presence of key UI elements on the homepage.

Java Code

```
@Test
public void testUIElements() {
    WebElement searchBar =
driver.findElement(By.id("search_query_top"));
    WebElement navigationMenu =
driver.findElement(By.id("block_top_menu"));
    WebElement footer = driver.findElement(By.id("footer"));
    Assert.assertTrue(searchBar.isDisplayed(), "Search bar is not
displayed");
    Assert.assertTrue(navigationMenu.isDisplayed(), "Navigation
menu is not displayed");
    Assert.assertTrue(footer.isDisplayed(), "Footer is not displayed");
}
```

d. Form Validation Test
   - Scenario: Automate a scenario where the user fills out a registration form.

Java Code

```
@Test
public void testFormValidation() {
    WebElement signInButton =
driver.findElement(By.className("login"));
    signInButton.click();

    WebElement emailCreateField =
driver.findElement(By.id("email_create"));
    WebElement createAccountButton =
driver.findElement(By.id("SubmitCreate"));

    emailCreateField.sendKeys("new_email@example.com");
    createAccountButton.click();
    WebElement firstNameField =
driver.findElement(By.id("customer_firstname"));
    WebElement lastNameField =
driver.findElement(By.id("customer_lastname"));
    WebElement passwordField =
driver.findElement(By.id("passwd"));
    WebElement registerButton =
driver.findElement(By.id("submitAccount"));

    firstNameField.sendKeys("");
    lastNameField.sendKeys("Doe");
    passwordField.sendKeys("password");
    registerButton.click();

    WebElement errorMessage =
driver.findElement(By.cssSelector(".alert-danger"));
    Assert.assertTrue(errorMessage.isDisplayed(), "Error message not
displayed for missing required fields");
}
```

3. Reporting
    a. Use Allure for generating test reports.

XML Code

```
<dependency>
    <groupId>io.qameta.allure</groupId>
    <artifactId>allure-testng</artifactId>
    <version>2.13.9</version>
</dependency>
```

4. Error Handling

   a. Log Errors:
      - Use logging libraries like Log4j or SLF4J to log errors.

5. Scalability

   a. Reusable and Scalable Tests:
      - Use Page Object Model (POM) to make tests reusable and scalable.

Expected Deliverables

Code Repository: GitHub Repository

Test Report: Generated Allure report.

Screen Recording: A short screen recording demonstrating the tests in action.

Evaluation Criteria

Correctness: Ensure tests validate the functionality as described.

Code Quality: Maintain well-structured, maintainable code.

Error Handling: Effectively handle and report errors.

Scalability: Make tests reusable and scalable.

Documentation: Provide clear documentation for setup and running tests.