|  |
| --- |
| Close-up image showing the leaf-sides of two oversized books side-by-side on a bookshelf, with additional books in soft focus background |
| **Test Cases**  According to Design Requirements of the Test Automation Plan |
| |  |  |  | | --- | --- | --- | | Weigand, Tibor | 5/25/24 | C# exercise | |

Table of Contents

[**1. Test Case: Verify Login Screen Display** 1](#_Toc167529717)

[**2. Test Case: Verify Username and Password Textboxes** 3](#_Toc167529718)

[**3. Test Case: Verify Login Button** 4](#_Toc167529719)

[**4. Test Case: Verify Error Message for Invalid Credentials** 5](#_Toc167529720)

[**5. Test Case 005: Verify Web Shop Test Page Loaded After Login** 6](#_Toc167529721)

[**Summary** 7](#_Toc167529722)

[**Composite Code** 7](#_Toc167529723)

## **1. Test Case: Verify Login Screen Display**

Documentation

* **Test Case ID**: TC001
* **Requirement Mark**: 1
* **Description**: Verify that the login screen is displayed when the application is started.
* **Preconditions**: None
* **Test Steps**:
  1. Open the application URL.
* **Expected Result**: The login screen should be displayed with the "Username" and "Password" fields and "Login" button.
* **Test Data**: Test usernames and the password are written on the page.
* **Priority**: High

**C# Code**

using OpenQA.Selenium;

using SeleniumExtras.PageObjects;

namespace AutomationTests.PageObjects {

public class LoginPage {

private IWebDriver driver;

public LoginPage(IWebDriver driver) {

this.driver = driver;

PageFactory.InitElements(driver, this);

}

[FindsBy(How = How.Id, Using = "user-name")]

public IWebElement UsernameField { get; set; }

[FindsBy(How = How.Id, Using = "password")]

public IWebElement PasswordField { get; set; }

[FindsBy(How = How.Id, Using = "login-button")]

public IWebElement LoginButton { get; set; }

[FindsBy(How = How.CssSelector, Using = "h3[data-test='error']")]

public IWebElement ErrorMessage { get; set; }

public void EnterUsername(string username) {

UsernameField.SendKeys(username);

}

public void EnterPassword(string password) {

PasswordField.SendKeys(password);

}

public void ClickLogin() {

LoginButton.Click();

}

}

}

## **2. Test Case: Verify Username and Password Textboxes**

Documentation

* **Test Case ID**: TC002
* **Requirement Mark**: 1.1
* **Description**: Verify that "Username" and "Password" textboxes are present, enabled, and empty.
* **Preconditions**: None
* **Test Steps**:
  1. Open the application URL.
  2. Check the "Username" textbox.
  3. Check the "Password" textbox.
* **Expected Result**: "Username" and "Password" textboxes should be enabled and empty.
* **Test Data**: None
* **Priority**: High

**C# Code**

[Test]

public void VerifyUsernameAndPasswordTextboxes()

{

// Test Case ID: TC002

// Requirement Mark: 1.1

driver.Navigate().GoToUrl("https://www.saucedemo.com/");

var usernameField = driver.FindElement(By.Id("user-name"));

var passwordField = driver.FindElement(By.Id("password"));

Assert.IsTrue(usernameField.Displayed && usernameField.Enabled && string.IsNullOrEmpty(usernameField.GetAttribute("value")));

Assert.IsTrue(passwordField.Displayed && passwordField.Enabled && string.IsNullOrEmpty(passwordField.GetAttribute("value")));

}

## **3. Test Case: Verify Login Button**

Documentation

* **Test Case ID**: TC003
* **Requirement Mark**: 1.2
* **Description**: Verify that the "Login" button is present and enabled on the login screen.
* **Preconditions**: None
* **Test Steps**:
  1. Open the application URL.
  2. Check the "Login" button.
* **Expected Result**: The "Login" button should be present and enabled.
* **Test Data**: None
* **Priority**: High

**C# Code**

[Test]

public void VerifyLoginButton()

{

// Test Case ID: TC003

// Requirement Mark: 1.2

driver.Navigate().GoToUrl("https://www.saucedemo.com/");

var loginButton = driver.FindElement(By.Id("login-button"));

Assert.IsTrue(loginButton.Displayed && loginButton.Enabled);

}

## **4. Test Case: Verify Error Message for Invalid Credentials**

Documentation

* **Test Case ID**: TC004
* **Requirement Mark**: 1.3
* **Description**: Verify that an error message is displayed when the user attempts to log in with invalid data.
* **Preconditions**: None
* **Test Steps**:
  1. Open the application URL.
  2. Enter invalid credentials.
  3. Click the "Login" button.
* **Expected Result**: An error message should be displayed indicating invalid login credentials.
* **Test Data**:
  1. **Invalid Username**: invalid\_user
  2. **Invalid Password**: invalid\_pass
* **Priority**: High

**C# Code**

[Test]

public void VerifyErrorMessageForInvalidCredentials()

{

// Test Case ID: TC004

// Requirement Mark: 1.3

driver.Navigate().GoToUrl("https://www.saucedemo.com/");

var usernameField = driver.FindElement(By.Id("user-name"));

var passwordField = driver.FindElement(By.Id("password"));

var loginButton = driver.FindElement(By.Id("login-button"));

usernameField.SendKeys("invalid\_user");

passwordField.SendKeys("invalid\_pass");

loginButton.Click();

var errorMessage = driver.FindElement(By.CssSelector("h3[data-test='error']"));

Assert.IsTrue(errorMessage.Displayed);

Assert.AreEqual("Epic sadface: Username and password do not match any user in this service", errorMessage.Text);

}

## **5. Test Case 005: Verify Web Shop Test Page Loaded After Login**

Documentation

* **Test Case ID**: TC005
* **Requirement Mark**: 1.4
* **Description**: Verify that the web shop test page is loaded after a successful login.
* **Preconditions**: Valid login credentials.
* **Test Steps**:
  1. Open the application URL.
  2. Enter valid login credentials.
  3. Click the "Login" button.
* **Expected Result**: The web shop test page should be loaded, and the user should see the product inventory.
* **Test Data**:
  1. **Valid Username**: standard\_user
  2. **Valid Password**: secret\_sauce
* **Priority**: High

**C# Code**

[Test]

public void VerifyWebShopTestPageLoadedAfterLogin()

{

// Test Case ID: TC005

// Requirement Mark: 1.4

driver.Navigate().GoToUrl("https://www.saucedemo.com/");

var usernameField = driver.FindElement(By.Id("user-name"));

var passwordField = driver.FindElement(By.Id("password"));

var loginButton = driver.FindElement(By.Id("login-button"));

usernameField.SendKeys("standard\_user");

passwordField.SendKeys("secret\_sauce");

loginButton.Click();

// Check if the web shop test page is loaded by verifying the presence of the inventory container

var inventoryContainer = driver.FindElement(By.Id("inventory\_container"));

Assert.IsTrue(inventoryContainer.Displayed);

}

## **Summary**

The above test cases cover the design requirements specified in the test automation plan. Each test case is documented with a detailed description, preconditions, test steps, expected results, test data, and priority.

The corresponding C# code implements these test cases using the Selenium WebDriver and NUnit framework, ensuring that the documentation aligns with the code. This ensures thorough testing of the login functionalities of the SauceDemo web application.

## **Composite Code**

using NUnit.Framework;

using OpenQA.Selenium;

using OpenQA.Selenium.Chrome;

namespace AutomationTests {

[TestFixture]

public class LoginTests {

private IWebDriver driver;

[SetUp]

public void SetUp() {

driver = new ChromeDriver();

}

[Test]

public void VerifyLoginScreenDisplay() {

// Test Case ID: TC001

// Requirement Mark: 1

driver.Navigate().GoToUrl("https://www.saucedemo.com/");

Assert.IsTrue(driver.FindElement(By.Id("user-name")).Displayed);

Assert.IsTrue(driver.FindElement(By.Id("password")).Displayed);

Assert.IsTrue(driver.FindElement(By.Id("login-button")).Displayed);

}

[Test]

public void VerifyUsernameAndPasswordTextboxes() {

// Test Case ID: TC002

// Requirement Mark: 1.1

driver.Navigate().GoToUrl("https://www.saucedemo.com/");

var usernameField = driver.FindElement(By.Id("user-name"));

var passwordField = driver.FindElement(By.Id("password"));

Assert.IsTrue(usernameField.Displayed && usernameField.Enabled && string.IsNullOrEmpty(usernameField.GetAttribute("value")));

Assert.IsTrue(passwordField.Displayed && passwordField.Enabled && string.IsNullOrEmpty(passwordField.GetAttribute("value")));

}

[Test]

public void VerifyLoginButton() {

// Test Case ID: TC003

// Requirement Mark: 1.2

driver.Navigate().GoToUrl("https://www.saucedemo.com/");

var loginButton = driver.FindElement(By.Id("login-button"));

Assert.IsTrue(loginButton.Displayed && loginButton.Enabled);

}

[Test]

public void VerifyErrorMessageForInvalidCredentials() {

// Test Case ID: TC004

// Requirement Mark: 1.3

driver.Navigate().GoToUrl("https://www.saucedemo.com/");

var usernameField = driver.FindElement(By.Id("user-name"));

var passwordField = driver.FindElement(By.Id("password"));

var loginButton = driver.FindElement(By.Id("login-button"));

usernameField.SendKeys("invalid\_user");

passwordField.SendKeys("invalid\_pass");

loginButton.Click();

var errorMessage = driver.FindElement(By.CssSelector("h3[data-test='error']"));

Assert.IsTrue(errorMessage.Displayed);

Assert.AreEqual("Epic sadface: Username and password do not match any user in this service", errorMessage.Text);

}

[Test]

public void VerifyWebShopTestPageLoadedAfterLogin() {

// Test Case ID: TC005

// Requirement Mark: 1.4

driver.Navigate().GoToUrl("https://www.saucedemo.com/");

var usernameField = driver.FindElement(By.Id("user-name"));

var passwordField = driver.FindElement(By.Id("password"));

var loginButton = driver.FindElement(By.Id("login-button"));

usernameField.SendKeys("standard\_user");

passwordField.SendKeys("secret\_sauce");

loginButton.Click();

// Check if the web shop test page is loaded by verifying the presence of the inventory container

var inventoryContainer = driver.FindElement(By.Id("inventory\_container"));

Assert.IsTrue(inventoryContainer.Displayed);

}

[TearDown]

public void TearDown() {

driver.Quit();

}

}

}