**TASK 1**

**AutomateAmazonFeature.feature:**

Feature: Shopping Cart Functionality

Scenario: Add item to cart and validate

Given I am an unregistered user on Amazon website

When I search for "TP-Link N450 WiFi Router - Wireless Internet Router for Home (TL-WR940N)"

And I add the corresponding item to the cart

**ShoppingCartSteps.cs:**

using System;

using TechTalk.SpecFlow;

using NUnit.Framework;

using OpenQA.Selenium;

using OpenQA.Selenium.Chrome;

using System.Collections.Generic;

using WebDriverManager.DriverConfigs.Impl;

using WebDriverManager;

using OpenQA.Selenium.Interactions;

using System.Collections.ObjectModel;

namespace AmazonTests

{

[Binding]

public class ShoppingCartSteps

{

private AmazonPage amazonPage;

private ShoppingCartPage shoppingCartPage;

public string searchedItem = "TP-Link N450 WiFi Router - Wireless Internet Router for Home (TL-WR940N)";

[Given(@"I am an unregistered user on Amazon website")]

public void GivenIAmAnUnregisteredUserOnAmazonWebsite()

{

amazonPage = new AmazonPage();

amazonPage.NavigateToAmazon();

}

[When(@"I search for ""(.\*)""")]

public void WhenISearchFor(string itemName)

{

amazonPage.SearchForItem(itemName);

}

[When(@"I add the corresponding item to the cart")]

public void WhenIAddTheCorrespondingItemToTheCart()

{

amazonPage.AddItemToCart(searchedItem);

}

[When(@"I navigate to the cart")]

public void WhenINavigateToTheCart()

{

shoppingCartPage.NavigateToCart();

}

[Then(@"I should see the correct item and amount displayed in the cart")]

public void ThenIShouldSeeTheCorrectItemAndAmountDisplayedInTheCart()

{

List<String> cartItemName = shoppingCartPage.GetCartItemNames();

List<String> cartItemPrice = shoppingCartPage.GetCartItemPrices();

decimal cartItemZeroPrice = decimal.Parse(cartItemPrice[0]);

// Assert.That(ShoppingCartSteps, cartItemName, "Incorrect item name in the cart");

Assert.That(cartItemZeroPrice > 0, "Invalid item price in the cart");

}

}

public class AmazonPage

{

private IWebDriver driver;

public AmazonPage()

{

new DriverManager().SetUpDriver(new ChromeConfig());

// Initialize ChromeDriver

driver = new ChromeDriver();

// Example usage: navigate to a webpage

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

// Close the browser

// driver.Quit();

}

// Method to navigate to Amazon website

public void NavigateToAmazon()

{

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

driver.Navigate().GoToUrl("https://www.amazon.com/dp/B096N2MV3H?th=1");

// driver.Navigate().GoToUrl("https://www.amazon.com/dp/B096N2MV3H?th=1");

driver.Navigate().GoToUrl("https://www.amazon.com/dp/B096N2MV3H?th=1");

driver.Navigate().Refresh();

driver.Manage().Window.Maximize();

}

// Method to search for a product on Amazon

public void SearchForItem(string productName)

{

try

{

IWebElement captchaBox = driver.FindElement(By.Id("captchacharacters"));

if (captchaBox != null)

{

if (captchaBox.Displayed && captchaBox.Enabled)

{

Assert.AreEqual(true, false, "[FAILED] Captcha box was shown, Automation tests cannot proceed on this website with unregistered user.");

}

}

}

catch (Exception ex){

Console.WriteLine("Captcha was not found so the test can proceed.");

IWebElement searchBox = driver.FindElement(By.Id("twotabsearchtextbox"));

searchBox.SendKeys(productName);

searchBox.SendKeys(Keys.Enter);

}

}

// Method to add a product to the cart

public void AddItemToCart(string productTitle)

{

// Click on the product

// IWebElement productLink = driver.FindElement(By.PartialLinkText(productTitle));

// productLink.Click();

// ReadOnlyCollection<IWebElement> addToCartButton = driver.FindElements(By.LinkText("Add to cart"));

//addToCartButton.

// IList<IWebElement> selectElements = driver.FindElements(By.TagName("select"));

// var displayedSelectElements = selectElements.get;

// var element = driver.FindElement(By.Id("a-autoid-1-announce"));

// Actions actions = new Actions(driver);

// actions.MoveToElement(element);

// actions.Perform();

// driver.FindElement(By.Id("a-autoid-2-announce")).Click();

// element = driver.FindElement(By.Id("a-autoid-2-announce"));

// actions = new Actions(driver);

// actions.MoveToElement(element);

// actions.Perform();

driver.FindElement(By.Id("a-autoid-3-announce")).Click();

// id = "a-autoid-3-announce"

// Click on the "Add to Cart" button

// IWebElement addToCartButton = driver.FindElement(By.Id("add-to-cart-button"));

// addToCartButton.Click();

// var element = driver.FindElement(By.XPath("//\*[@id=\"nav-cart-count-container\"]/span[2]"));

// Actions actions = new Actions(driver);

// actions.MoveToElement(element);

// actions.Perform();

// driver.FindElement(By.XPath("//\*[@id=\"nav-cart-count-container\"]/span[2]")).Click();

Thread.Sleep(3000);

driver.FindElement(By.LinkText("Go to Cart")).Click();

IWebElement productLink = driver.FindElement(By.PartialLinkText("TP-Link"));

//\*[@id="sc-active-f6f5759a-c5f7-48d3-89af-76dc36c5c8b5"]/div[4]/div/div[2]/ul/li/span/a/span[1]/span/span[2]

IWebElement productLink2 = driver.FindElement(By.PartialLinkText("$"));

String x = "TP-Link";

if (productLink.Text.Contains(x)){

Assert.AreEqual(true, true, "[FAILED] Doesn't contain text 'TP-LINK', text: "+productLink.Text);

Console.WriteLine("[PASSED] Contain text 'TP-LINK', text: "+productLink.Text);

}

else

{

Console.WriteLine("");

Assert.AreEqual(true, false, "[FAILED] Doesn't contain text 'TP-LINK', text: "+productLink.Text);

}

if (productLink2.Text.Contains("$")){

Assert.AreEqual(true, true, "[FAILED] Doesn't contain $ sign, text: "+productLink2.Text);

Console.WriteLine("[PASSED] Contain $ sign, text: "+productLink2.Text);

}

else

{

Console.WriteLine("");

Assert.AreEqual(true, false, "[FAILED] Doesn't contain $ sign, text: " + productLink2.Text);

}

//\*[@id="sc-active-252e5ed1-69fe-4040-a85a-7563615ca171"]/div[4]/div/div[2]/ul/li/span/a/span[1]/span/span[2]

//\*[@id="sc-active-252e5ed1-69fe-4040-a85a-7563615ca171"]/div[4]/div/div[2]/ul/div[1]/div[1]/div/div/span

// driver.FindElement(By.LinkText("Go to Cart")).Click();

}

// Method to close the browser

public void CloseBrowser()

{

driver.Quit();

}

}

public class ShoppingCartPage

{

private IWebDriver driver;

public ShoppingCartPage(IWebDriver driver)

{

this.driver = driver;

}

// Method to navigate to the shopping cart

public void NavigateToCart()

{

IWebElement cartIcon = driver.FindElement(By.Id("nav-cart-count"));

cartIcon.Click();

}

// Method to get names of items in the shopping cart

public List<string> GetCartItemNames()

{

// Navigate to the shopping cart

NavigateToCart();

// Initialize a list to store item names

List<string> itemNames = new List<string>();

// Get list of items in the cart

IReadOnlyCollection<IWebElement> itemRows = driver.FindElements(By.CssSelector(".a-spacing-mini.sc-list-item"));

// Iterate through each item row and extract name

foreach (IWebElement itemRow in itemRows)

{

// Get item name

IWebElement itemNameElement = itemRow.FindElement(By.CssSelector(".a-text-left.sc-product-link"));

string itemName = itemNameElement.Text;

// Add item name to the list

itemNames.Add(itemName);

}

return itemNames;

}

// Method to get prices of items in the shopping cart

public List<string> GetCartItemPrices()

{

// Navigate to the shopping cart

NavigateToCart();

// Initialize a list to store item prices

List<string> itemPrices = new List<string>();

// Get list of items in the cart

IReadOnlyCollection<IWebElement> itemRows = driver.FindElements(By.CssSelector(".a-spacing-mini.sc-list-item"));

// Iterate through each item row and extract price

foreach (IWebElement itemRow in itemRows)

{

// Get item price

IWebElement itemPriceElement = itemRow.FindElement(By.CssSelector(".a-color-price"));

string itemPrice = itemPriceElement.Text;

// Add item price to the list

itemPrices.Add(itemPrice);

}

return itemPrices;

}

// Method to get the total price of items in the shopping cart

public double GetTotalPrice()

{

IWebElement totalPriceElement = driver.FindElement(By.CssSelector("[data-name='Subtotals'] .a-color-price"));

string totalPriceText = totalPriceElement.Text;

// Parse the total price string into a double

double totalPrice = double.Parse(totalPriceText.Replace("$", ""));

return totalPrice;

}

}

}

/\*

\*

[Test]

public void TheUntitledTestCaseTest()

{

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

driver.FindElement(By.Id("captchacharacters")).Click();

driver.FindElement(By.Id("captchacharacters")).Clear();

driver.FindElement(By.Id("captchacharacters")).SendKeys("xbulfj");

driver.FindElement(By.XPath("//button[@type='submit']")).Click();

driver.FindElement(By.Id("twotabsearchtextbox")).Click();

driver.FindElement(By.Id("twotabsearchtextbox")).Clear();

driver.FindElement(By.Id("twotabsearchtextbox")).SendKeys("TP-Link N450 WiFi Router - Wireless Internet Router for Home (TL-WR940N)");

driver.FindElement(By.Id("nav-search-submit-button")).Click();

driver.FindElement(By.Id("a-autoid-2-announce")).Click();

driver.FindElement(By.Id("a-autoid-4-announce")).Click();

driver.FindElement(By.LinkText("Go to Cart")).Click();

driver.FindElement(By.XPath("//div[@id='sc-active-9bd18247-ab42-4656-a9c8-dc748b131746']/div[4]/div/div[2]/ul/li/span/a/span/span/span[2]")).Click();

// ERROR: Caught exception [ERROR: Unsupported command [selectWindow | win\_ser\_1 | ]]

// ERROR: Caught exception [ERROR: Unsupported command [selectWindow | win\_ser\_local | ]]

driver.FindElement(By.XPath("//div[@id='sc-active-9bd18247-ab42-4656-a9c8-dc748b131746']/div[4]/div/div[2]/ul/li/span/a/span/span/span[2]")).Click();

// ERROR: Caught exception [ERROR: Unsupported command [selectWindow | win\_ser\_2 | ]]

// ERROR: Caught exception [ERROR: Unsupported command [selectWindow | win\_ser\_local | ]]

driver.FindElement(By.XPath("//div[@id='sc-active-9bd18247-ab42-4656-a9c8-dc748b131746']/div[4]/div/div[2]/ul/li/span/a/span/span/span[2]")).Click();

// ERROR: Caught exception [ERROR: Unsupported command [selectWindow | win\_ser\_3 | ]]

// ERROR: Caught exception [ERROR: Unsupported command [selectWindow | win\_ser\_local | ]]

driver.FindElement(By.XPath("//div[@id='sc-active-9bd18247-ab42-4656-a9c8-dc748b131746']/div[4]/div/div[2]/ul/div/div/div/div/span")).Click();

driver.FindElement(By.XPath("//div[@id='sc-active-265468e9-3d7a-40c7-9230-ec5439d4453b']/div[4]/div/div[2]/ul/li/span/a/span/span/span[2]")).Click();

// ERROR: Caught exception [ERROR: Unsupported command [selectWindow | win\_ser\_4 | ]]

// ERROR: Caught exception [ERROR: Unsupported command [selectWindow | win\_ser\_local | ]]

driver.FindElement(By.XPath("//div[@id='sc-active-265468e9-3d7a-40c7-9230-ec5439d4453b']/div[4]/div/div[2]/ul/div/div/div/div/span")).Click();

driver.FindElement(By.XPath("//span[@id='sc-subtotal-amount-activecart']/span")).Click();

}

private bool IsElementPresent(By by)

{

try

{

driver.FindElement(by);

return true;

}

catch (NoSuchElementException)

{

return false;

}

}

private bool IsAlertPresent()

{

try

{

driver.SwitchTo().Alert();

return true;

}

catch (NoAlertPresentException)

{

return false;

}

}

private string CloseAlertAndGetItsText() {

try {

IAlert alert = driver.SwitchTo().Alert();

string alertText = alert.Text;

if (acceptNextAlert) {

alert.Accept();

} else {

alert.Dismiss();

}

return alertText;

} finally {

acceptNextAlert = true;

}

}

\*/

**ImplicitUsing.cs:**

global using FluentAssertions;

global using NUnit;

global using TechTalk.SpecFlow;

**chromedriver.exe:**

**TASK 2:**

**BOOKS COLLECTION.postman\_collection.json:**

{

"info": {

"\_postman\_id": "c0ce61a6-06fd-4156-aaf4-f5ff3b32c6da",

"name": "Books Collection",

"schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json",

"\_exporter\_id": "9111715"

},

"item": [

{

"name": "Get Book Details",

"event": [

{

"listen": "test",

"script": {

"exec": [

"\r",

"var responseBody = pm.response.json();\r",

"var firstItemName = responseBody[0].name;\r",

"var secondItemType = responseBody[1].type;\r",

"\r",

"pm.test(\"Status code is 200\", function () {\r",

" pm.response.to.have.status(200);\r",

"});\r",

"\r",

"pm.test(\"Response body should contain name: 'The Russian'\", function () {\r",

" pm.expect(responseBody[0].name).to.equal(\"The Russian\");\r",

"});\r",

"\r",

"pm.test(\"Response body should contain type: 'non-fiction'\", function () {\r",

" pm.expect(responseBody[1].type).to.equal(\"non-fiction\");\r",

"});\r",

""

],

"type": "text/javascript",

"packages": {}

}

}

],

"request": {

"method": "GET",

"header": [],

"url": {

"raw": "https://simple-books-api.glitch.me/books",

"protocol": "https",

"host": [

"simple-books-api",

"glitch",

"me"

],

"path": [

"books"

]

}

},

"response": []

}

]

}

**RestSharp Program.cs file:**

using System;

using System.Diagnostics;

using System.Net;

using System.Text.Json.Nodes;

using Newtonsoft.Json;

using Newtonsoft.Json.Linq;

using RestSharp;

namespace RestSharpExample2

{

class Program2

{

static void Main(string[] args)

{

var client = new RestClient("https://simple-books-api.glitch.me/");

var request = new RestRequest("/books", Method.Get);

RestResponse response = client.Execute(request);

Console.WriteLine("Response Content:");

Console.WriteLine(response.Content);

if (response.StatusCode == System.Net.HttpStatusCode.OK)

{

Console.WriteLine("[PASSED] Status code is 200");

JArray jsonArray = JArray.Parse(response.Content);

String name = jsonArray[0]["name"].ToString();

String type = jsonArray[1]["type"].ToString();

Console.WriteLine("Response body should contain name: 'The Russian'");

Trace.Assert(name.Equals("The Russian"), "[FAILED] Response body should contain name: 'The Russian'");

Console.WriteLine("[PASSED] Response body should contain name: 'The Russian'");

Console.WriteLine("Response body should contain type: 'non-fiction'");

Trace.Assert(type.Equals("non-fiction"), "[FAILED] Response body should contain type: 'non-fiction'");

Console.WriteLine("[PASSED] Response body should contain type: 'non-fiction'");

}

else

{

Console.WriteLine("Error: " + response.ErrorMessage);

Console.WriteLine("Status code is NOT 200");

}

}

}

}

Test Scripts Postman:

var responseBody = pm.response.json();

var firstItemName = responseBody[0].name;

var secondItemType = responseBody[1].type;

pm.test("Status code is 200", function () {

    pm.response.to.have.status(200);

});

pm.test("Response body should contain name: 'The Russian'", function () {

    pm.expect(responseBody[0].name).to.equal("The Russian");

});

pm.test("Response body should contain type: 'non-fiction'", function () {

    pm.expect(responseBody[1].type).to.equal("non-fiction");

});

API Response POSTMAN:  
[

    {

        "id": 1,

        "name": "The Russian",

        "type": "fiction",

        "available": **true**

    },

    {

        "id": 2,

        "name": "Just as I Am",

        "type": "non-fiction",

        "available": **false**

    },

    {

        "id": 3,

        "name": "The Vanishing Half",

        "type": "fiction",

        "available": **true**

    },

    {

        "id": 4,

        "name": "The Midnight Library",

        "type": "fiction",

        "available": **true**

    },

    {

        "id": 5,

        "name": "Untamed",

        "type": "non-fiction",

        "available": **true**

    },

    {

        "id": 6,

        "name": "Viscount Who Loved Me",

        "type": "fiction",

        "available": **true**

    }

]