

Week 3 (Automation Testing)

→ Tools Used

- **Automation Tool:** Selenium WebDriver
- **Language:** C#
- **Test Framework:** NUnit
- **API Tool (Optional):** Postman
- **Test Application:** SauceDemo (<https://www.saucedemo.com>)

→ Automation Testing Overview

Automation testing is used to execute test cases automatically using scripts instead of manual effort. It helps save time, improve accuracy, and support regression testing.

In this project, Selenium WebDriver is used to automate browser actions such as login, navigation, and feature validation.

→ Automated Test Cases

Test Case 1: Login Validation

Objective:

Verify that a user can successfully log in using valid credentials.

Steps Automated:

1. Open SauceDemo website
2. Enter valid username
3. Enter valid password
4. Click Login button
5. Verify user is redirected to Products page

Expected Result:

User should log in successfully and see the Products page.

Status: Passed

Test Case 2: Feature Functionality Add Product to Cart

Objective:

Verify that a product can be added to the cart.

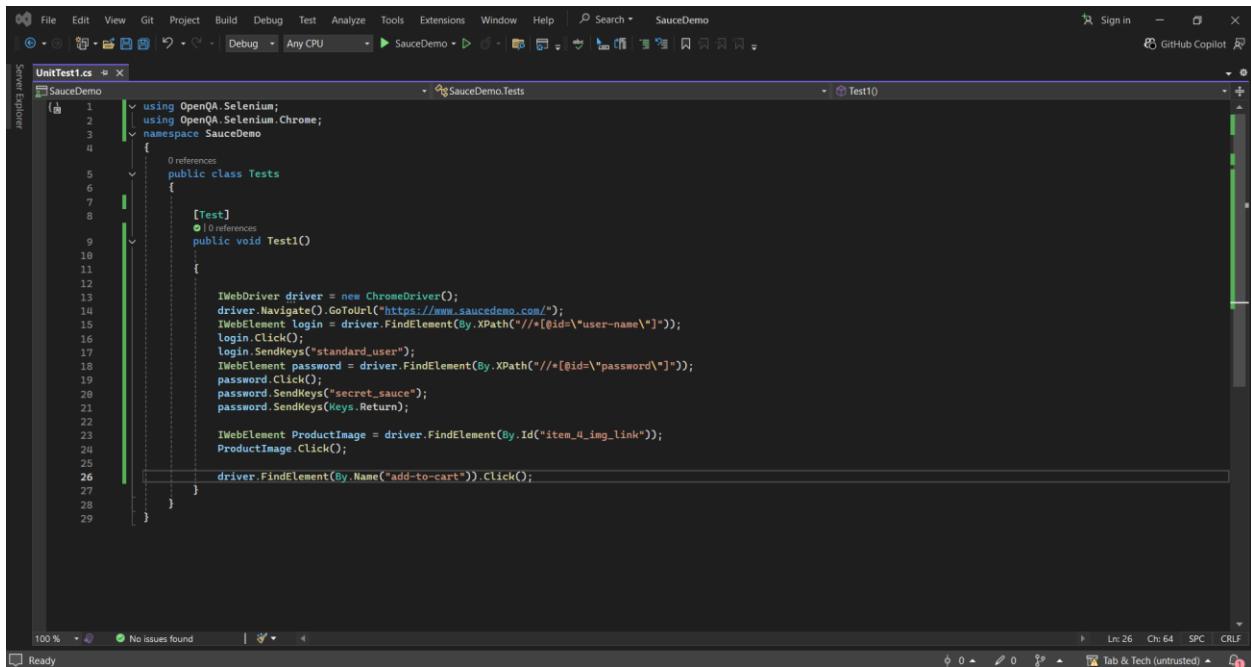
Steps Automated:

1. Login to the application
2. Click on a product
3. Click “Add to Cart” button
4. Verify cart icon shows item count

Expected Result:

Product should be added to cart successfully.

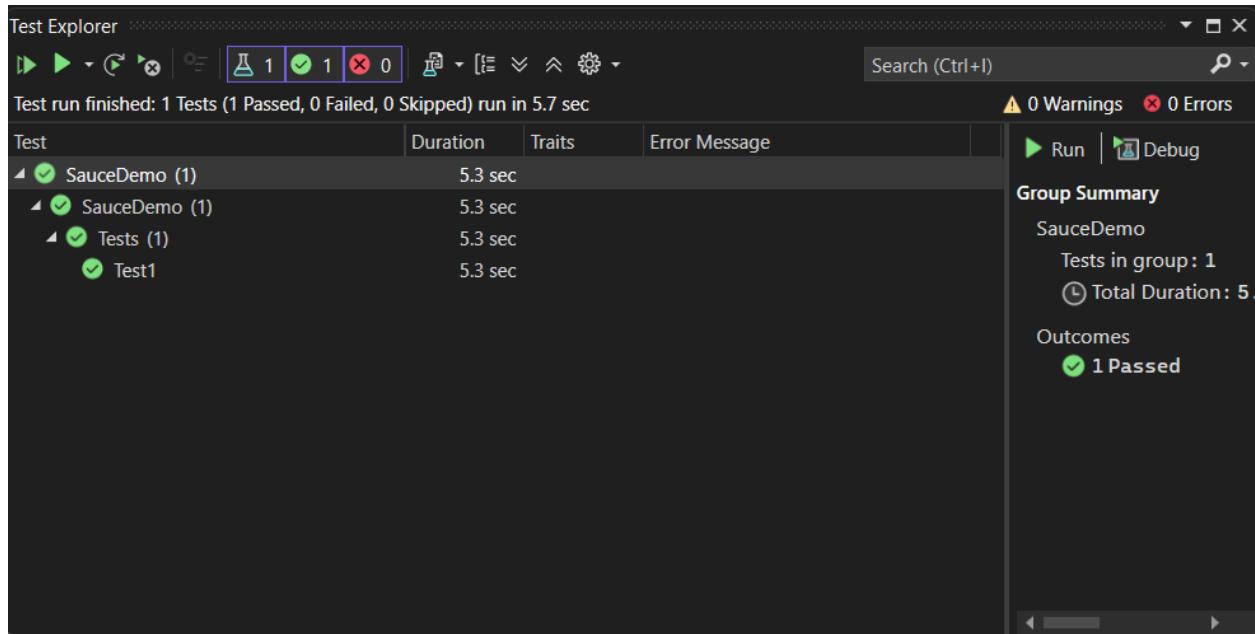
Status: Passed



The screenshot shows the Microsoft Visual Studio IDE interface. The top menu bar includes File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a GitHub Copilot icon. The title bar displays "UnitTest1.cs" and "SauceDemo". The main code editor window contains the following C# code:

```
1  using OpenQA.Selenium;
2  using OpenQA.Selenium.Chrome;
3  namespace SauceDemo
4  {
5      public class Tests
6      {
7          [Test]
8          public void Test1()
9          {
10
11             IWebDriver driver = new ChromeDriver();
12             driver.Navigate().GoToUrl("https://www.saucedemo.com/");
13             IWebElement login = driver.FindElement(By.XPath("//*[@id='user-name']"));
14             login.Click();
15             login.SendKeys("standard_user");
16             IWebElement password = driver.FindElement(By.XPath("//*[@id='password']"));
17             password.Click();
18             password.SendKeys("secret_sauce");
19             password.SendKeys(Keys.Return);
20
21             IWebElement ProductImage = driver.FindElement(By.Id("item_4_img_link"));
22             ProductImage.Click();
23
24             driver.FindElement(By.Name("add-to-cart")).Click();
25
26         }
27     }
28 }
```

The code implements a Selenium test named "Test1" that logs in with a standard user and adds a product from the fourth item in the inventory to the cart. The test uses the ChromeDriver and interacts with elements via XPath locators.



→ SCRIPT:

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
namespace SauceDemo
{
    public class Tests
    {
        [Test]
        public void Test1()
        {
            IWebDriver driver = new ChromeDriver();
            driver.Navigate().GoToUrl("https://www.saucedemo.com/");
            IWebElement login = driver.FindElement(By.XPath("//*[@id='user-name']"));
            login.Click();
            login.SendKeys("standard_user");
            IWebElement password = driver.FindElement(By.XPath("//*[@id='password']"));
            password.Click();
            password.SendKeys("secret_sauce");
            password.SendKeys(Keys.Return);

            IWebElement ProductImage = driver.FindElement(By.Id("item_4_img_link"));
            ProductImage.Click();

            driver.FindElement(By.Name("add-to-cart")).Click();
        }
    }
}
```

→ API Testing Using Postman (Simple Books API)

API Reference Used

For API testing, the **Simple Books API** was used.

This API provides endpoints for practicing common HTTP methods such as **GET, POST, PUT, and DELETE**.

API Documentation Source:

Simple Books API , Introduction to Postman Course

→ Post (Token Generation):

The screenshot shows the Postman application interface. At the top, there's a search bar and various navigation icons. Below the header, a sidebar lists 'Overview', 'GET New Request', 'GET https://simple...', 'POST New Request' (with a red dot), and 'POST Api token' (which is selected). A dropdown menu indicates 'No environment'. On the left, a tree view shows 'HTTP End-to-End Tests / Api token'. The main workspace shows a POST request to 'https://simple-books-api.click/api-clients/'. The 'Body' tab is active, showing raw JSON input:

```
1 {
2   "clientName": "Postman",
3   "clientEmail": "aisha@example.com"
4 }
```

Below the request, the response status is '201 Created', with details: 407 ms, 826 B. There are buttons for 'Save Response' and 'Beautify'. At the bottom, there are tabs for 'Body', 'Cookies', 'Headers (14)', and 'Test Results', along with preview and visualize options.

→ GET (List Of Books):

Postman interface showing a successful GET request to <https://simple-books-api.glitch.me/books>. The response is a 200 OK with 939 ms duration and 1.13 KB size. The JSON response body contains the following data:

	id	name	type	available
0	1	The Russian	fiction	true
1	3	The Vanishing Half	fiction	true
2	4	The Midnight Library	fiction	true
3	6	Viscount Who Loved Me	fiction	true
4	2	Just as I Am	non-fiction	false
5	5	Untamed	non-fiction	true

→ POST (Submit an Order):

Postman interface showing a successful POST request to <https://simple-books-api.click/orders>. The response is a 201 Created with 370 ms duration and 794 B size. The JSON response body is:

```

1 {
2   "created": true,
3   "orderId": "gydYtys08L6UeXp9Li9hh"
4 }

```

→ DELETE (Delete an order):

The screenshot shows the Postman application interface. At the top, there is a navigation bar with links for Overview, GET New Re, GET https://, POST Subr, POST Api tc, GET List of I, and DEL New R. To the right of these are buttons for '+', 'No environment', and a settings icon. Below the navigation is a toolbar with 'HTTP' (selected), 'End-to-End Tests / New Request', 'Save', 'Share', and a link icon.

The main workspace shows a 'DELETE' request to the URL <https://simple-books-api.click/orders/gydYtys08L6UeXp9Li9hh>. The 'Headers' tab is selected, showing 8 headers. The 'Body' tab is also visible.

Below the request details, the response section displays a status of '204 No Content'. Other response details include '517 ms', '746 B', and a 'Save Response' button. The response body is shown as a single character '1'.