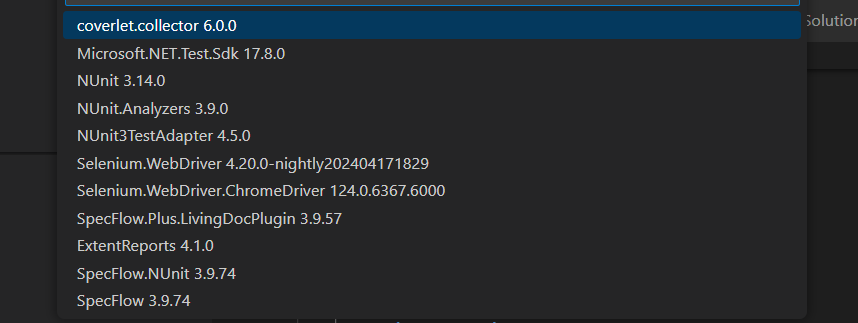
Let's modify the code to ensure the file is correctly created. Here's the updated code:

Step 1: Install Required Packages



Step 2: Configure ExtentReports

ExtentReports.cs:

Create a class to configure ExtentReports.

using AventStack.ExtentReports;

using AventStack.ExtentReports.Reporter;

using System;

using System.IO;

namespace MyTestProject

{

public class ExtentReportsConfig

{

public static ExtentReports extent;

public static ExtentHtmlReporter htmlReporter;

public static ExtentReports GetExtentReports()

{

if (extent == null)

{

string reportDirectory = AppDomain.CurrentDomain.BaseDirectory;

string reportPath = reportDirectory.Replace("bin\\Debug\\net8.0","TestResults");

if (!Directory.Exists(reportDirectory))

{

Directory.CreateDirectory(reportDirectory);

}

htmlReporter = new ExtentHtmlReporter(reportPath);

htmlReporter.Config.Theme = AventStack.ExtentReports.Reporter.Configuration.Theme.Standard;

extent = new ExtentReports();

extent.AttachReporter(htmlReporter);

}

return extent;

}

}

}

Step 3: Update the Hooks under Hooks Folder

Hooks.cs:

Update the hooks class to start and close the ExtentReports.

using AventStack.ExtentReports;

using AventStack.ExtentReports.Gherkin.Model;

using AventStack.ExtentReports.Reporter;

using NUnit.Framework;

using TechTalk.SpecFlow;

namespace MyTestProject

{

[Binding]

public sealed class Hooks

{

private static ExtentReports extent;

private static ExtentTest featureName;

private static ExtentTest scenario;

[BeforeTestRun]

public static void InitializeReport()

{

extent = ExtentReportsConfig.GetExtentReports();

}

[BeforeFeature]

public static void BeforeFeature(FeatureContext featureContext)

{

featureName = extent.CreateTest<Feature>(featureContext.FeatureInfo.Title);

}

[BeforeScenario]

public void BeforeScenario(ScenarioContext scenarioContext)

{

scenario = featureName.CreateNode<Scenario>(scenarioContext.ScenarioInfo.Title);

}

[AfterStep]

public void InsertReportingSteps(ScenarioContext scenarioContext)

{

var stepType = ScenarioStepContext.Current.StepInfo.StepDefinitionType.ToString();

if (scenarioContext.TestError == null)

{

if (stepType == "Given")

scenario.CreateNode<Given>(ScenarioStepContext.Current.StepInfo.Text);

else if (stepType == "When")

scenario.CreateNode<When>(ScenarioStepContext.Current.StepInfo.Text);

else if (stepType == "Then")

scenario.CreateNode<Then>(ScenarioStepContext.Current.StepInfo.Text);

else if (stepType == "And")

scenario.CreateNode<And>(ScenarioStepContext.Current.StepInfo.Text);

}

else if (scenarioContext.TestError != null)

{

if (stepType == "Given")

scenario.CreateNode<Given>(ScenarioStepContext.Current.StepInfo.Text).Fail(scenarioContext.TestError.InnerException);

else if (stepType == "When")

scenario.CreateNode<When>(ScenarioStepContext.Current.StepInfo.Text).Fail(scenarioContext.TestError.InnerException);

else if (stepType == "Then")

scenario.CreateNode<Then>(ScenarioStepContext.Current.StepInfo.Text).Fail(scenarioContext.TestError.Message);

}

}

[AfterTestRun]

public static void TearDownReport()

{

extent.Flush();

}

}

}

Step 4: Update the Test Class

AmazonAddItemToCartSteps.cs:

Update the test class to include the hooks.

using System;

using OpenQA.Selenium;

using OpenQA.Selenium.Chrome;

using TechTalk.SpecFlow;

using NUnit.Framework;

namespace MyTestProject

{

[Binding]

public class AmazonAddItemToCartSteps

{

private IWebDriver driver;

private HomePage homePage;

private ProductPage productPage;

private CartPage cartPage;

[BeforeScenario]

public void Setup()

{

driver = new ChromeDriver();

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

driver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(10);

homePage = new HomePage(driver);

homePage.GoTo();

}

[Given(@"I am on the Amazon home page")]

public void GivenIAmOnTheAmazonHomePage()

{

// This step is already handled in the BeforeScenario hook

}

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

public void WhenISearchFor(string itemName)

{

// Search for the item

homePage.SearchItem(itemName);

}

[When(@"I click on the first search result")]

public void WhenIClickOnTheFirstSearchResult()

{

// Click on the first item

productPage = homePage.ClickFirstItem();

}

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

public void WhenIAddTheItemToTheCart()

{

// Add the item to the cart

productPage.AddToCart();

}

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

public void WhenIGoToTheCart()

{

// Navigate to the cart

cartPage = productPage.GoToCart();

}

[Then(@"I validate the correct item and amount")]

public void ThenIValidateTheCorrectItemAndAmount()

{

// Validate the correct item and amount

Assert.AreEqual("TP-Link Tri-Band BE9300 WiFi 7 Router Archer BE550 | 6-Stream 9.2Gbps | 𝗙𝘂𝗹𝗹 𝟮.𝟱𝗚 Ports | USB 3.0 | 6 Smart Internal Antennas | VPN Clients & Server | Easy Mesh, HomeS…", cartPage.GetItemName().Trim(), "Incorrect item in the cart.");

Assert.AreEqual("$249.00", cartPage.GetItemPrice().Trim(), "Incorrect item price in the cart.");

}

[AfterScenario]

public void Cleanup()

{

if (driver != null)

{

driver.Quit();

driver.Dispose();

}

}

}

}

Step 5: Run the Tests and Generate the Report

Open a terminal in Visual Studio Code (Terminal > New Terminal).

Run the following command to execute the tests:

bash

Copy code

dotnet test

After running the tests, you can find the generated report in the TestResults directory. Open ExtentReport.html to view the test report.