|  |  |
| --- | --- |
| **Start Agile - Capstone Project** | **Student Name: Vijina P** |
| **Automation Testing Framework for bstackdemo.com using Java and Selenium** | |

**Question:**

**Automate Test Case Scenarios**

Login Tests

1. TC\_001: Login with valid credentials (username: demouser, password: testingisfun99)
2. TC\_002: Login with invalid credentials
3. TC\_003: Login with empty username/password ¯/.

Cart Tests

1. TC\_004: Add single item to cart
2. TC\_005: Add multiple items to cart and verify cart count
3. TC\_006: Remove item from cart .

Checkout Tests

1. TC\_007: Place order with valid details
2. TC\_008: Checkout flow without adding items (negative test)

**Github repository**

The source code has been uploaded to the following git repository;

**Project Structure**



**pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>Auto\_Testing\_BstackDemo</groupId>

<artifactId>Auto\_Testing\_BstackDemo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>4.34.0</version>

</dependency>

<dependency>

<groupId>com.aventstack</groupId>

<artifactId>extentreports</artifactId>

<version>5.0.9</version>

</dependency>

</dependencies>

<build>

<sourceDirectory>src</sourceDirectory>

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.13.0</version>

<configuration>

<release>13</release>

</configuration>

</plugin>

</plugins>

</build>

</project>

**testng.xml**

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name="BstackDemo">

<test name="BstackDemo\_Test">

<classes>

<class name="test.java.tests.LoginTests"/>

<class name="test.java.tests.AddToCartTests"/>

<class name="test.java.tests.CheckoutTests"/>

</classes>

</test>

</suite>

**\src\main\java\pages\CartPage.java**

package main.java.pages;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import main.java.utils.WaitUtils;

import java.util.List;

public class CartPage {

private WebDriver driver;

private WaitUtils waitUtil;

private By cartCount = By.*xpath*("/html/body/div/div/div/div[2]/div[2]/div[1]/span[1]/span");

private By removeButtons = By.*cssSelector*(".shelf-item\_\_del");

private By checkoutButton = By.*className*("buy-btn");

public CartPage(WebDriver driver, WaitUtils waitUtil) {

this.driver = driver;

this.waitUtil = waitUtil;

}

public int getCartItemCount() {

waitUtil.waitForVisibility(cartCount);

String countText = driver.findElement(cartCount).getText();

try {

return Integer.*parseInt*(countText);

} catch (NumberFormatException e) {

return 0;

}

}

public void removeItem(int index) {

List<WebElement> removes = driver.findElements(removeButtons);

if (index < removes.size()) {

removes.get(index).click();

}

}

public void clickCheckout() {

driver.findElement(checkoutButton).click();

}

}

**\src\main\java\pages\CheckoutPage.java**

package main.java.pages;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import main.java.utils.WaitUtils;

public class CheckoutPage {

private WebDriver driver;

private WaitUtils waitUtil;

private By firstNameInput = By.id("firstNameInput");

private By lastNameInput = By.id("lastNameInput");

private By addressLine1Input = By.id("addressLine1Input");

private By provinceInput = By.id("provinceInput");

private By postCodeInput = By.id("postCodeInput");

private By submitBtn = By.id("checkout-shipping-continue");

private By confirmationMessage = By.id("confirmation-message");

public CheckoutPage(WebDriver driver, WaitUtils waitUtil) {

this.driver = driver;

this.waitUtil = waitUtil;

}

public void clickPurchase() {

driver.findElement(submitBtn).click();

}

public String getConfirmationMessage() {

return driver.findElement(confirmationMessage).getText();

}

public void fillCheckoutForm(String firstName, String lastName, String addressLine1,

String province, String postCode) {

driver.findElement(firstNameInput).sendKeys(firstName);

driver.findElement(lastNameInput).sendKeys(lastName);

driver.findElement(addressLine1Input).sendKeys(addressLine1);

driver.findElement(provinceInput).sendKeys(province);

driver.findElement(postCodeInput).sendKeys(postCode);

}

}

**\src\main\java\pages\ProductPage.java**

package main.java.pages;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import main.java.utils.WaitUtils;

import java.util.List;

public class ProductPage {

private WebDriver driver;

private WaitUtils waitUtil;

private By products = By.cssSelector(".shelf-item");

private By addToCartButton = By.cssSelector(".shelf-item\_\_buy-btn");

private By cartCount = By.xpath("/html/body/div/div/div/div[2]/div[2]/div[1]/span[1]/span");

private By cartIcon = By.className("bag--float-cart-closed");

public ProductPage(WebDriver driver, WaitUtils waitUtil) {

this.driver = driver;

this.waitUtil = waitUtil;

}

public void addItemToCart(int index) {

List<WebElement> items = driver.findElements(products);

if (index < items.size()) {

items.get(index).findElement(addToCartButton).click();

}

}

public int getCartCount() {

waitUtil.waitForVisibility(cartCount);

String countText = driver.findElement(cartCount).getText();

try {

return Integer.parseInt(countText);

} catch (NumberFormatException e) {

return 0;

}

}

public void openCart() {

driver.findElement(cartIcon).click();

}

public String getLoggedInUserName() {

waitUtil.waitForVisibility(By.className("username"));

return driver.findElement(By.className("username")).getText();

}

}

**\src\main\java\pages\LoginPage.java**

package main.java.pages;

import org.openqa.selenium.By;

import org.openqa.selenium.Keys;

import org.openqa.selenium.WebDriver;

import main.java.utils.WaitUtils;

public class LoginPage {

private WebDriver driver;

private WaitUtils waitUtil;

private By usernameDropdown = By.id("username");

private By passwordDropdown = By.id("password");

private By usernameInput = By.cssSelector("#username input");

private By passwordInput = By.cssSelector("#password input");

private By loginButton = By.id("login-btn");

private By errorMessage = By.className("api-error");

public LoginPage(WebDriver driver, WaitUtils waitUtil) {

this.driver = driver;

this.waitUtil = waitUtil;

}

public String getErrorMessage() {

try {

return driver.findElement(errorMessage).getText();

} catch (Exception e) {

return "";

}

}

public void login(String username, String password) {

if(username != "")

{

driver.findElement(usernameDropdown).click();

driver.findElement(usernameInput).sendKeys(username + Keys.ENTER);

}

if(password != "")

{

driver.findElement(passwordDropdown).click();

driver.findElement(passwordInput).sendKeys(password + Keys.ENTER);

}

driver.findElement(loginButton).click();

}

}

**\src\main\java\utils\ConfigReader.java**

package main.java.utils;

import java.io.FileInputStream;

import java.io.IOException;

import java.util.Properties;

public class ConfigReader {

private static Properties *properties*;

// Static block executes only once when the class is loaded

static {

try {

FileInputStream fis = new FileInputStream("config.properties");

*properties* = new Properties();

*properties*.load(fis);

} catch (IOException e) {

System.*out*.println("Exception while loading config.properties: " + e.getMessage());

e.printStackTrace();

}

}

// Method to fetch value by key

public static String get(String key) {

return *properties*.getProperty(key);

}

// Method to fetch value with default if not found

public static String getOrDefault(String key, String defaultValue) {

return *properties*.getProperty(key, defaultValue);

}

}

**\src\main\java\utils\WaitUtils.java**

package main.java.utils;

import org.openqa.selenium.\*;

import org.openqa.selenium.support.ui.ExpectedConditions;

import org.openqa.selenium.support.ui.WebDriverWait;

import java.time.Duration;

public class WaitUtils {

private WebDriver driver;

private WebDriverWait wait;

// Constructor

public WaitUtils(WebDriver driver, int timeoutInSeconds) {

this.driver = driver;

this.wait = new WebDriverWait(driver, Duration.ofSeconds(timeoutInSeconds));

}

// Wait until element is visible

public WebElement waitForVisibility(By locator) {

return wait.until(ExpectedConditions.visibilityOfElementLocated(locator));

}

// Wait until element is clickable

public WebElement waitForClickable(By locator) {

return wait.until(ExpectedConditions.elementToBeClickable(locator));

}

// Wait until element is present in the DOM

public WebElement waitForPresence(By locator) {

return wait.until(ExpectedConditions.presenceOfElementLocated(locator));

}

// Wait until element is invisible

public boolean waitForInvisibility(By locator) {

return wait.until(ExpectedConditions.invisibilityOfElementLocated(locator));

}

// Wait until text is present in element

public boolean waitForText(By locator, String text) {

return wait.until(ExpectedConditions.textToBePresentInElementLocated(locator, text));

}

// Wait until alert is present

public Alert waitForAlert() {

return wait.until(ExpectedConditions.alertIsPresent());

}

}

**\src\main\java\utils\WebDriverFactory.java**

package main.java.utils;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.edge.EdgeDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import java.time.Duration;

public class WebDriverFactory {

public static WebDriver createDriver(String browser) {

WebDriver driver;

switch (browser.toLowerCase()) {

case "chrome":

driver = new ChromeDriver();

break;

case "firefox":

driver = new FirefoxDriver();

break;

case "edge":

driver = new EdgeDriver();

break;

default:

throw new IllegalArgumentException("Invalid browser: " + browser);

}

// Common WebDriver settings

driver.manage().window().maximize();

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(5));

return driver;

}

}

**Test Cases**

**\src\test\java\tests\BaseTest.java**

**package** test.java.tests;

**import** com.aventstack.extentreports.ExtentReports;

**import** com.aventstack.extentreports.ExtentTest;

**import** com.aventstack.extentreports.reporter.ExtentSparkReporter;

**import** org.openqa.selenium.WebDriver;

**import** org.testng.annotations.\*;

**import** main.java.utils.ConfigReader;

**import** main.java.utils.WaitUtils;

**import** main.java.utils.WebDriverFactory;

**public** **class** BaseTest {

**protected** WebDriver driver;

**protected** **static** ExtentReports *extent*;

**protected** ExtentTest test;

**protected** WaitUtils waitUtil;

@BeforeSuite

**public** **void** setupExtent() {

ExtentSparkReporter spark = **new** ExtentSparkReporter("test-output/ExtentReport.html");

*extent* = **new** ExtentReports();

*extent*.attachReporter(spark);

}

@AfterSuite

**public** **void** tearDownExtent() {

*extent*.flush();

}

@BeforeMethod

**public** **void** setupDriver() {

String browser = ConfigReader.*getOrDefault*("browser", "chrome");

System.*setProperty*(ConfigReader.*get*("driverClass"),

ConfigReader.*get*("driverPath"));

driver = WebDriverFactory.*createDriver*(browser);

driver.get(ConfigReader.*get*("baseUrl"));

waitUtil = **new** WaitUtils(driver,5);

}

@AfterMethod

**public** **void** quitDriver() {

**if**(driver != **null**) {

driver.quit();

}

}

}

**\src\test\java\tests\LoginTests.java**

package test.java.tests;

import com.aventstack.extentreports.Status;

import org.testng.Assert;

import org.testng.annotations.Test;

import main.java.pages.\*;

public class LoginTests extends BaseTest {

@Test

public void loginWithValidCredentials() {

test = extent.createTest("Login with valid credentials");

LoginPage loginPage = new LoginPage(driver, waitUtil);

loginPage.login("demouser", "testingisfun99");

// Check for successful login: dashboard displays logged in username

ProductPage homePage = new ProductPage(driver, waitUtil);

String loggedInUserName = homePage.getLoggedInUserName();

test.log(Status.INFO, "Logged in UserName: " + loggedInUserName);

Assert.assertEquals(loggedInUserName,"demouser", "Login probably failed, cart count is invalid");

test.log(Status.PASS, "Login with valid credentials passed");

}

@Test

public void loginWithInvalidCredentials() {

test = extent.createTest("Login with invalid credentials");

LoginPage loginPage = new LoginPage(driver, waitUtil);

loginPage.login("wronguser", "wrongpassword");

String error = loginPage.getErrorMessage();

test.log(Status.INFO, "Error message displayed: " + error);

Assert.assertTrue(error.contains("Invalid") || !error.isEmpty(), "Error message should be displayed");

test.log(Status.PASS, "Login with invalid credentials test passed");

}

@Test

public void loginWithEmptyUsernamePassword() {

test = extent.createTest("Login with empty username and password");

LoginPage loginPage = new LoginPage(driver, waitUtil);

loginPage.login("", "");

String error = loginPage.getErrorMessage();

test.log(Status.INFO, "Error message displayed: " + error);

Assert.assertTrue(error.contains("required") || !error.isEmpty(), "Error message should be displayed for empty input");

test.log(Status.PASS, "Login with empty credentials test passed");

}

}

**\src\test\java\tests\AddToCartTests.java**

package test.java.tests;

import com.aventstack.extentreports.Status;

import org.testng.Assert;

import org.testng.annotations.BeforeMethod;

import org.testng.annotations.Test;

import main.java.pages.\*;

public class AddToCartTests extends BaseTest {

@BeforeMethod

public void login() {

LoginPage loginPage = new LoginPage(driver, waitUtil);

loginPage.login("demouser", "testingisfun99");

}

@Test

public void addSingleItemToCart() {

test = extent.createTest("Add single item to cart");

ProductPage homePage = new ProductPage(driver, waitUtil);

homePage.addItemToCart(0);

int cartCount = homePage.getCartCount();

test.log(Status.INFO, "Cart count after adding one item: " + cartCount);

Assert.assertEquals(cartCount, 1);

test.log(Status.PASS, "Add single item to cart test passed");

}

@Test

public void addMultipleItemsToCartAndVerifyCount() {

test = extent.createTest("Add multiple items to cart and verify count");

ProductPage homePage = new ProductPage(driver, waitUtil);

homePage.addItemToCart(0);

homePage.addItemToCart(1);

int cartCount = homePage.getCartCount();

test.log(Status.INFO, "Cart count after adding two items: " + cartCount);

Assert.assertEquals(cartCount, 2);

test.log(Status.PASS, "Add multiple items to cart and verify count test passed");

}

@Test

public void removeItemFromCart() {

test = extent.createTest("Remove item from cart");

ProductPage homePage = new ProductPage(driver, waitUtil);

homePage.addItemToCart(0);

homePage.addItemToCart(1);

CartPage cartPage = new CartPage(driver, waitUtil);

int beforeRemoveCount = cartPage.getCartItemCount();

test.log(Status.INFO, "Cart items before removal: " + beforeRemoveCount);

cartPage.removeItem(0);

int afterRemoveCount = cartPage.getCartItemCount();

test.log(Status.INFO, "Cart items after removal: " + afterRemoveCount);

Assert.assertEquals(afterRemoveCount, beforeRemoveCount - 1);

test.log(Status.PASS, "Remove item from cart test passed");

}

}

**\src\test\java\tests\CheckoutTests.java**

package test.java.tests;

import com.aventstack.extentreports.Status;

import org.testng.Assert;

import org.testng.annotations.BeforeMethod;

import org.testng.annotations.Test;

import main.java.pages.\*;

import main.java.utils.WaitUtils;

public class CheckoutTests extends BaseTest {

@BeforeMethod

public void login() {

LoginPage loginPage = new LoginPage(driver, waitUtil);

loginPage.login("demouser", "testingisfun99");

}

@Test

public void placeOrderWithValidDetails() {

test = extent.createTest("Place order with valid details");

ProductPage homePage = new ProductPage(driver, waitUtil);

homePage.addItemToCart(2);

CartPage cartPage = new CartPage(driver, waitUtil);

cartPage.clickCheckout();

CheckoutPage checkoutPage = new CheckoutPage(driver, waitUtil);

checkoutPage.fillCheckoutForm("Vijina","P","Bangalore","KA","560068");

checkoutPage.clickPurchase();

String confirmation = checkoutPage.getConfirmationMessage();

test.log(Status.INFO, "Order confirmation message: " + confirmation);

Assert.assertEquals(confirmation,"Your Order has been successfully placed.", "Confirmation message not displayed");

test.log(Status.PASS, "Place order with valid details test passed");

}

@Test

public void checkoutWithoutAddingItems() {

test = extent.createTest("Checkout flow without adding items (negative test)");

ProductPage homePage = new ProductPage(driver, waitUtil);

homePage.openCart();

CartPage cartPage = new CartPage(driver, waitUtil);

cartPage.clickCheckout();

// Assuming that the checkout page won't allow purchase or shows error on empty cart

CheckoutPage checkoutPage = new CheckoutPage(driver, waitUtil);

// Fill the details form

try {

checkoutPage.fillCheckoutForm("Vijina","P","Bangalore","KA","560068");

checkoutPage.clickPurchase();

String confirmation = checkoutPage.getConfirmationMessage();

// If confirmation contains error message, assert that

test.log(Status.INFO, "Checkout confirmation or message: " + confirmation);

Assert.assertEquals(confirmation,"Your Order has been successfully placed.", "Confirmation message not displayed");

} catch (Exception e) {

test.log(Status.INFO, "Exception or failure as expected: " + e.getMessage());

Assert.assertTrue(true);

}

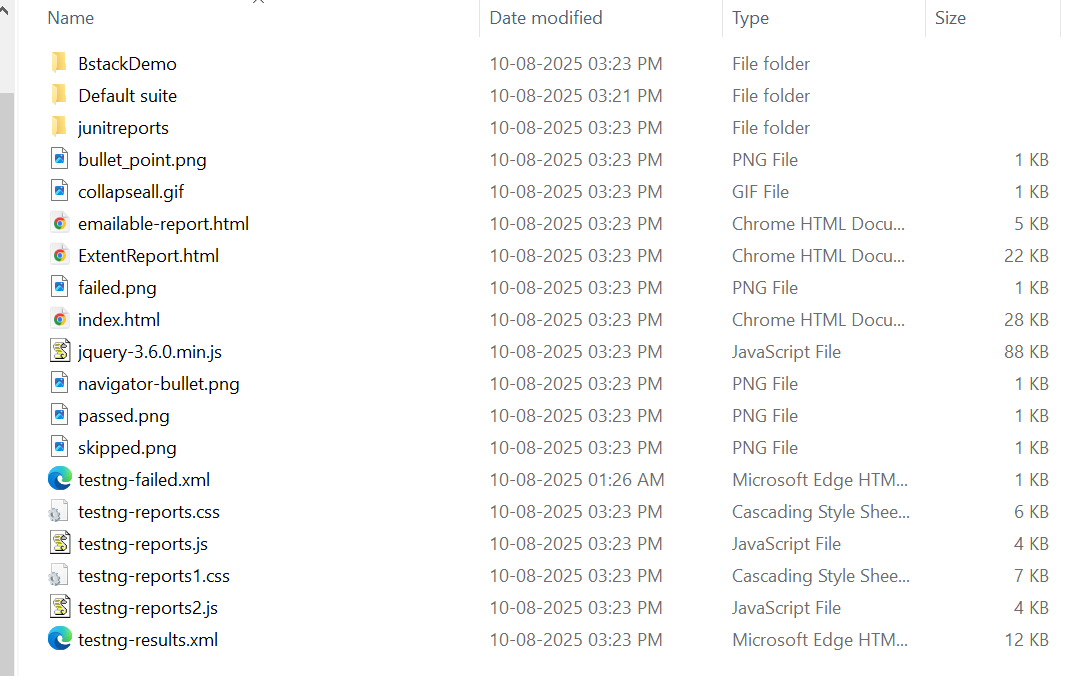
test.log(Status.PASS, "Checkout without adding items negative test passed");

}

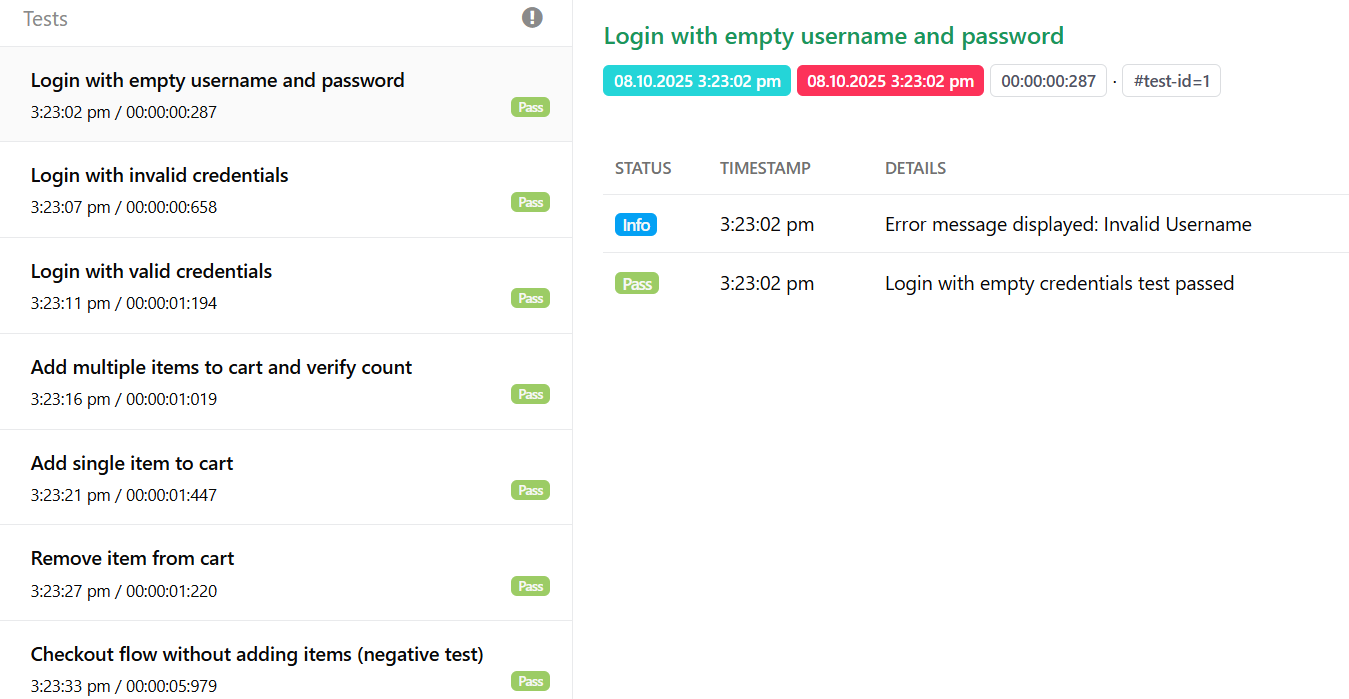
}

**Output**

**Test-output Folder**



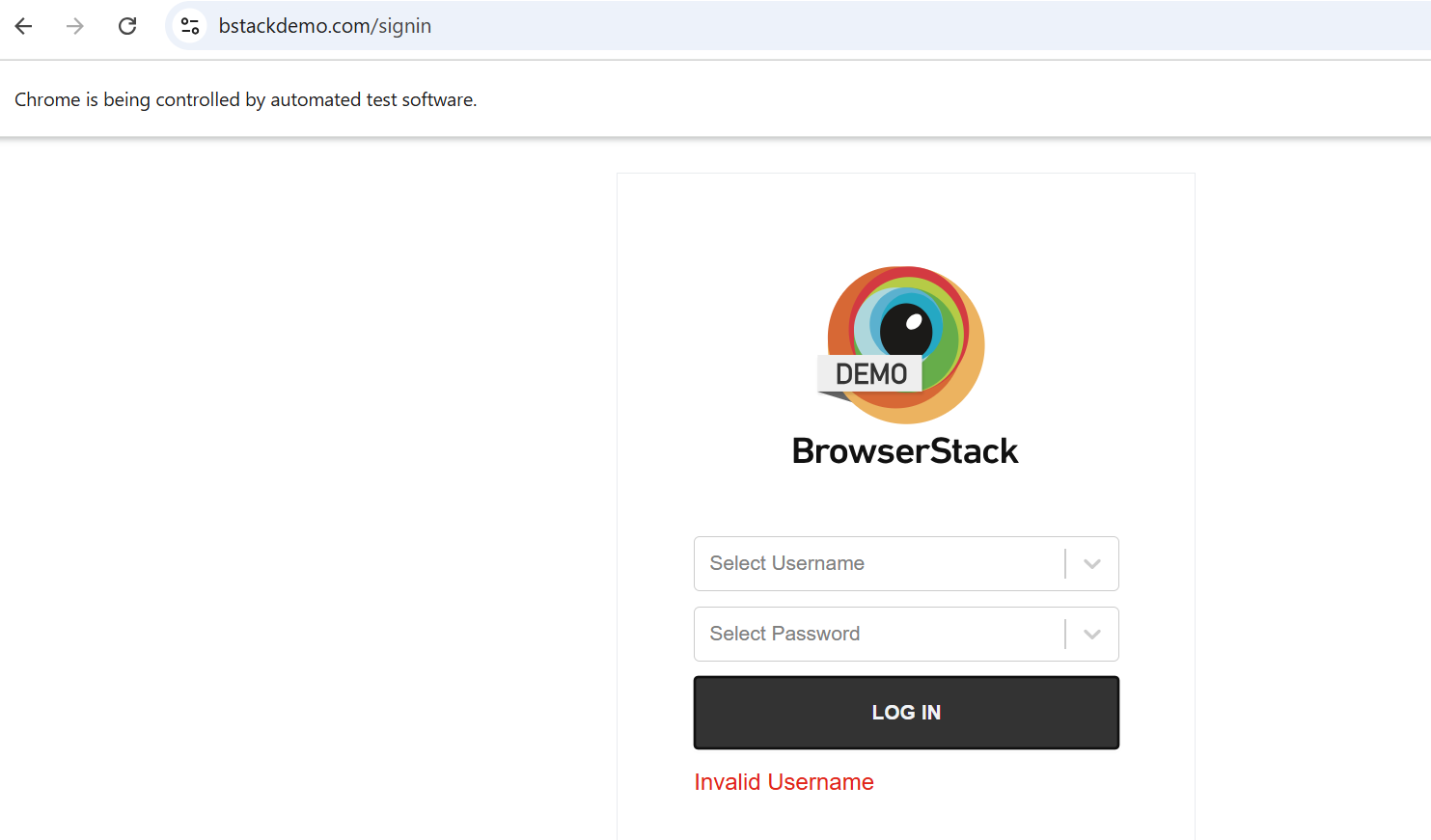
**Extent Report**

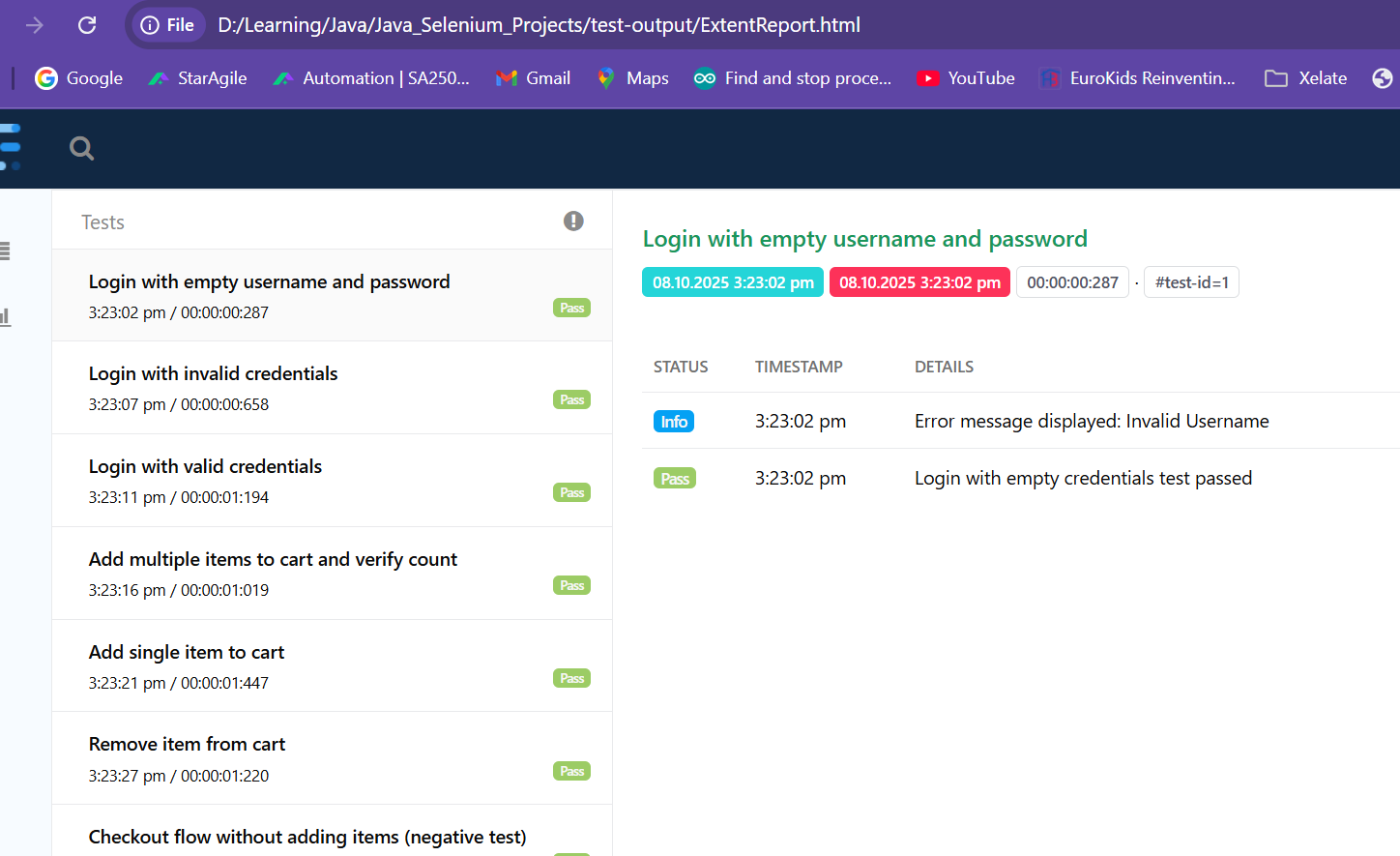


**Screenshots**

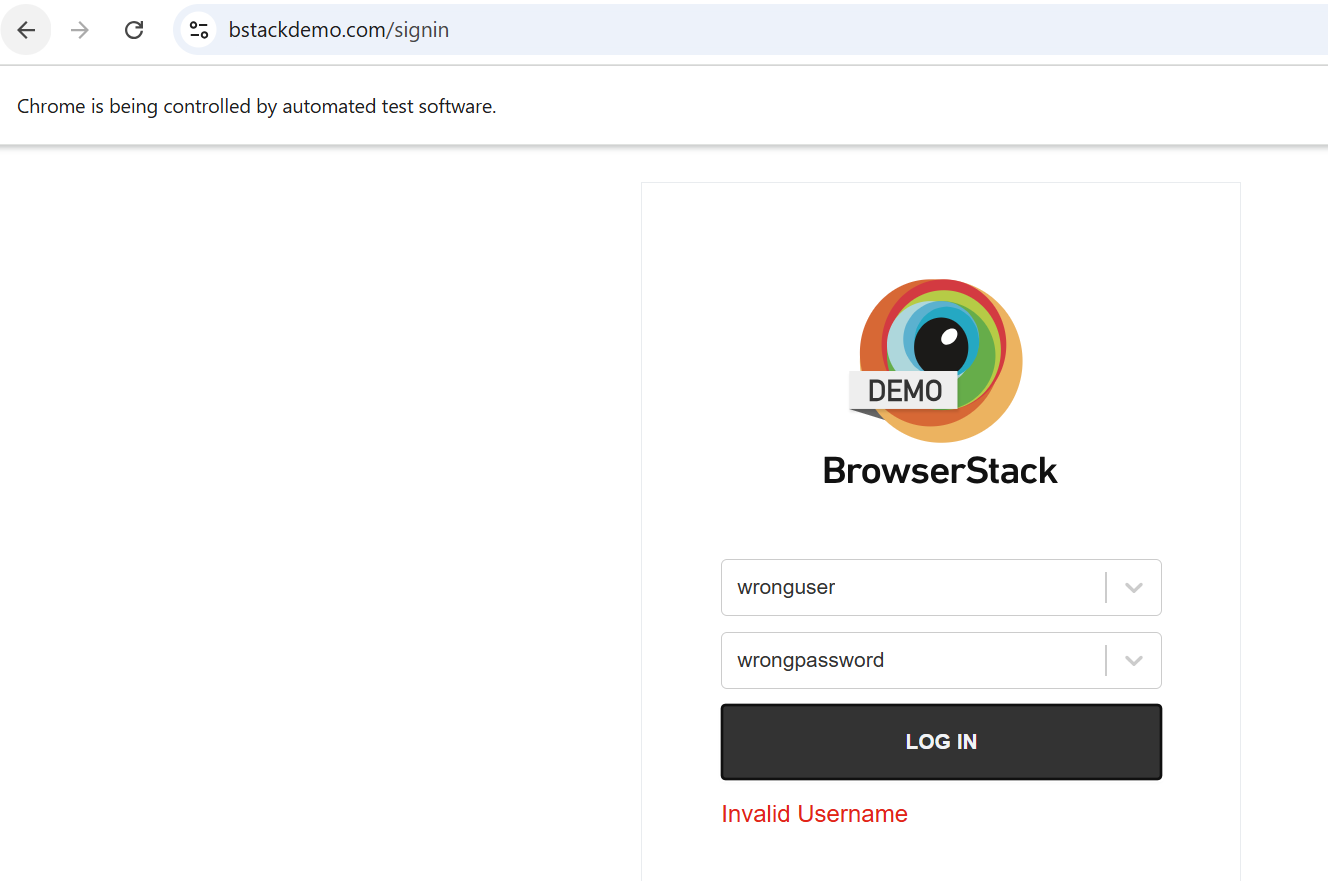
**Login Tests**

**TC\_003: Login with empty username and password**

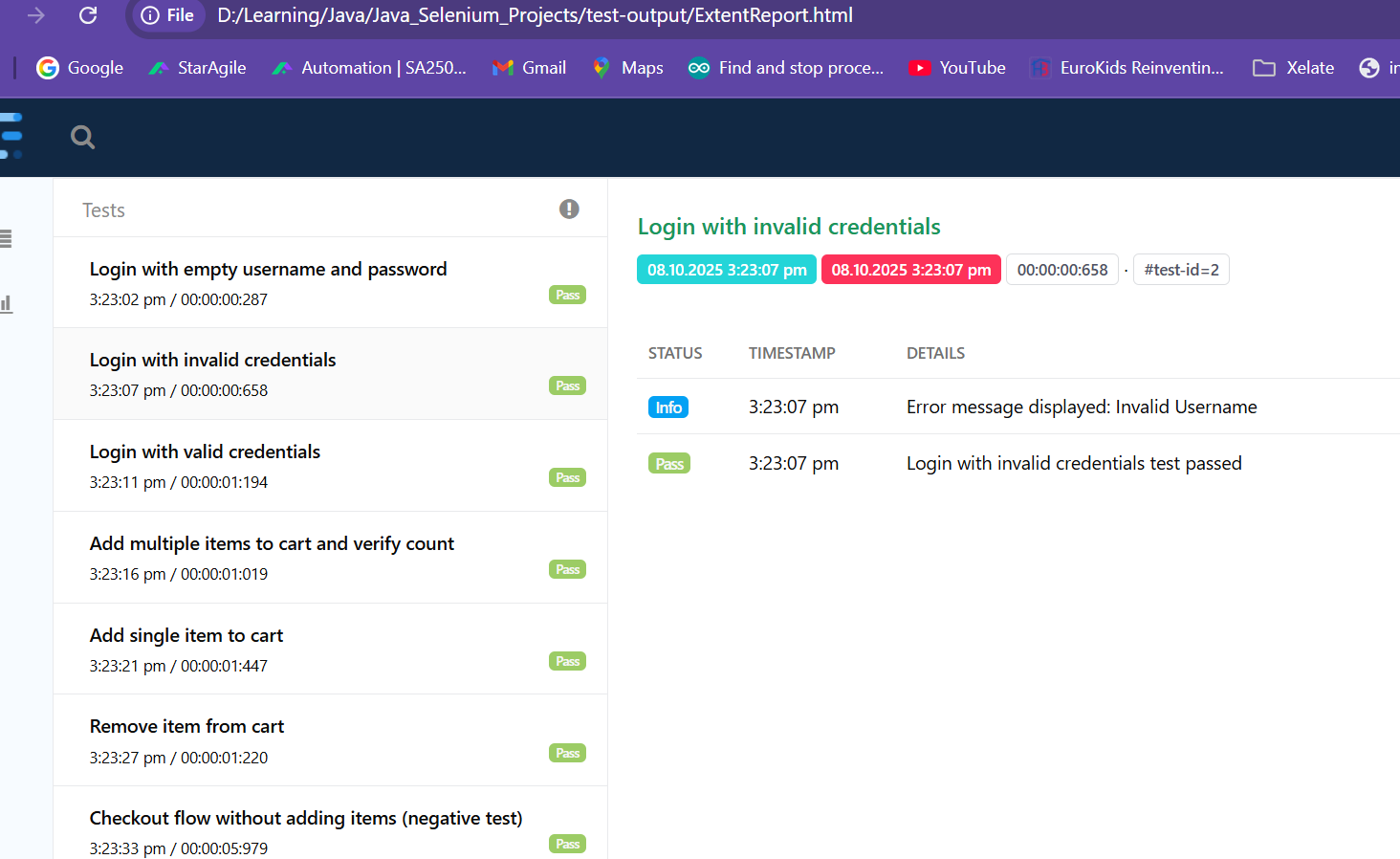


**ExtentReport Screenshot**

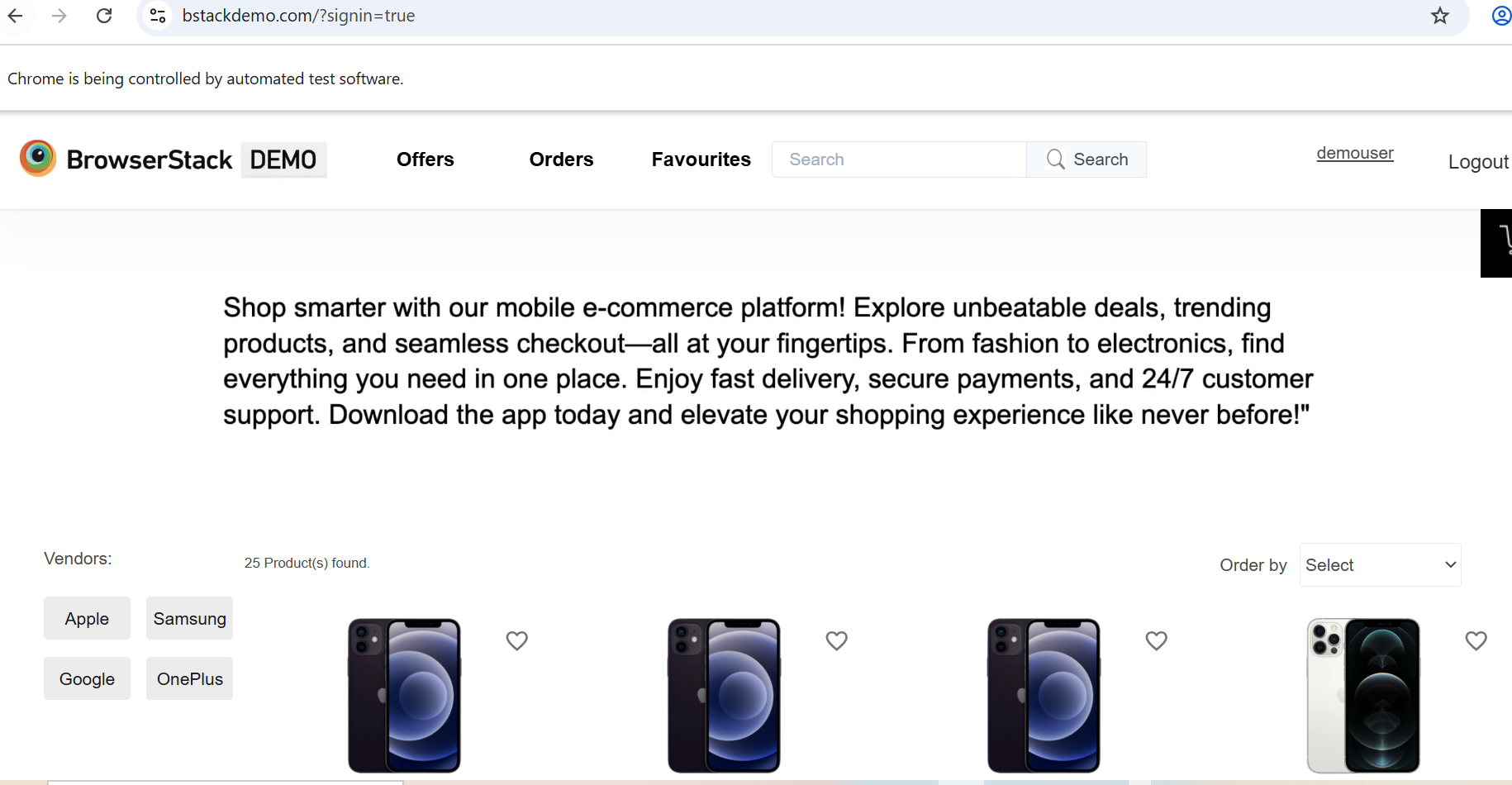
**TC\_002: Login with invalid credentials**



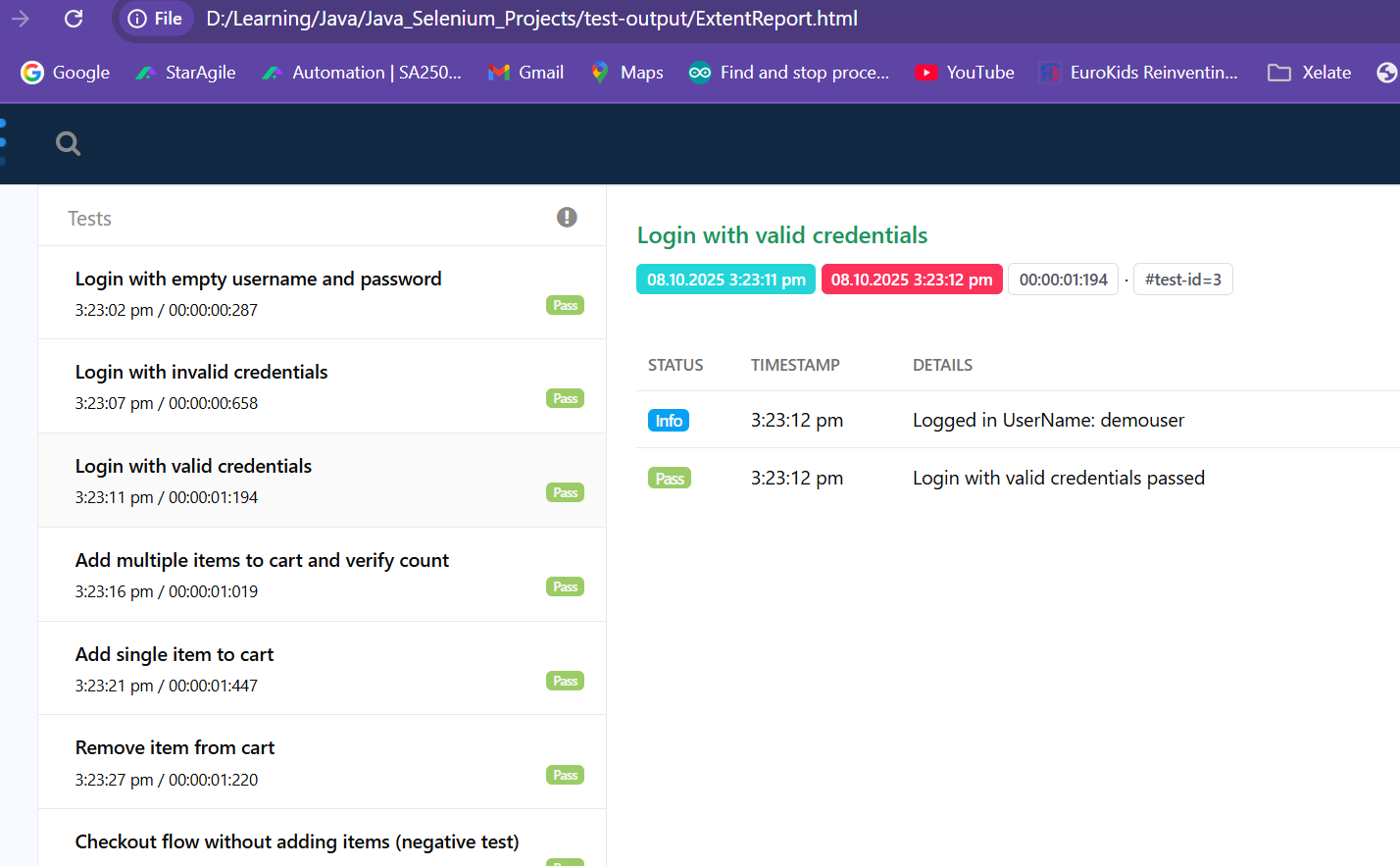
**ExtentReport Screenshot**



**TC\_001: Login with valid credentials (username: demouser, password: testingisfun99)**

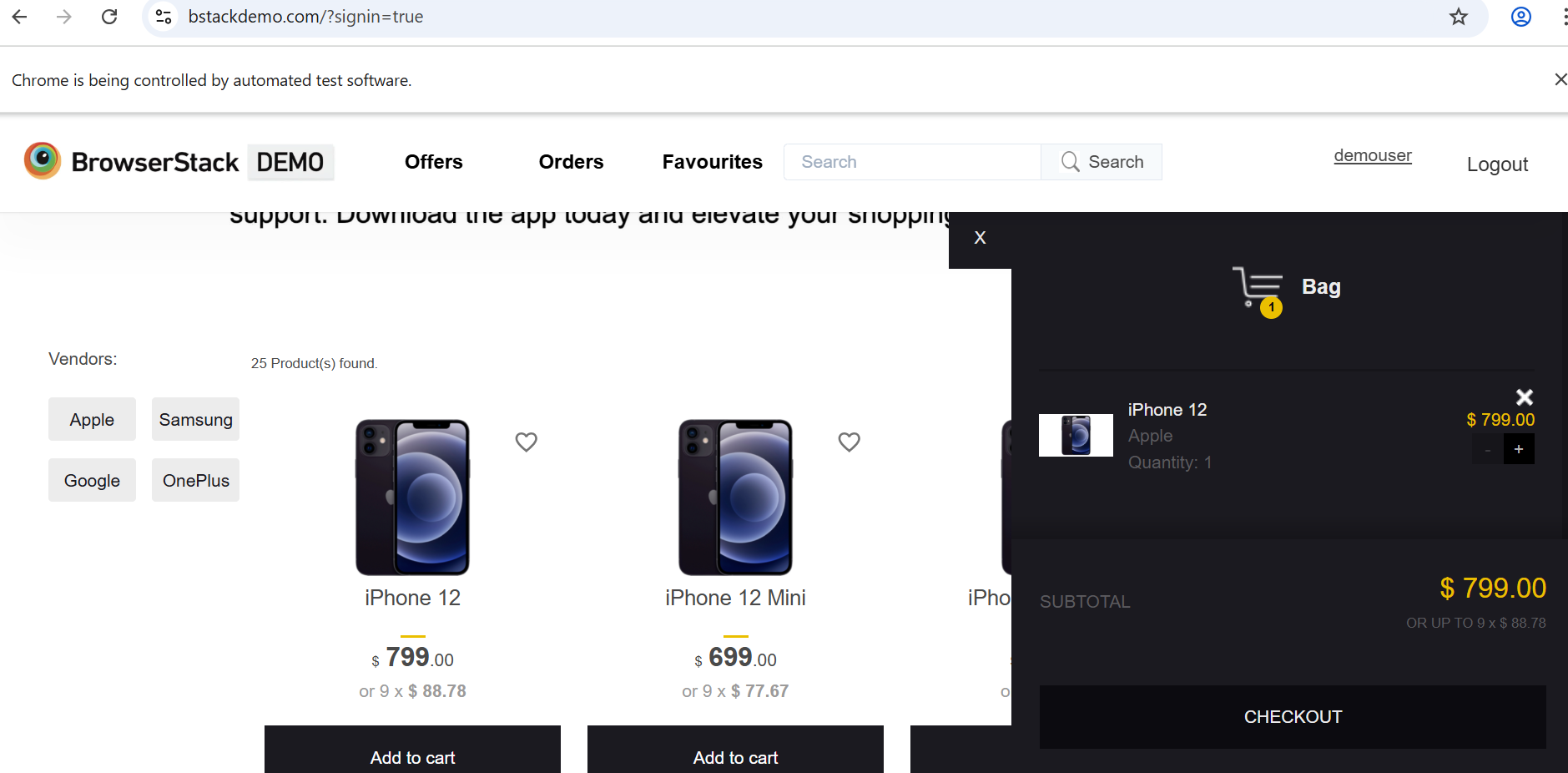


**ExtentReport Screenshot**

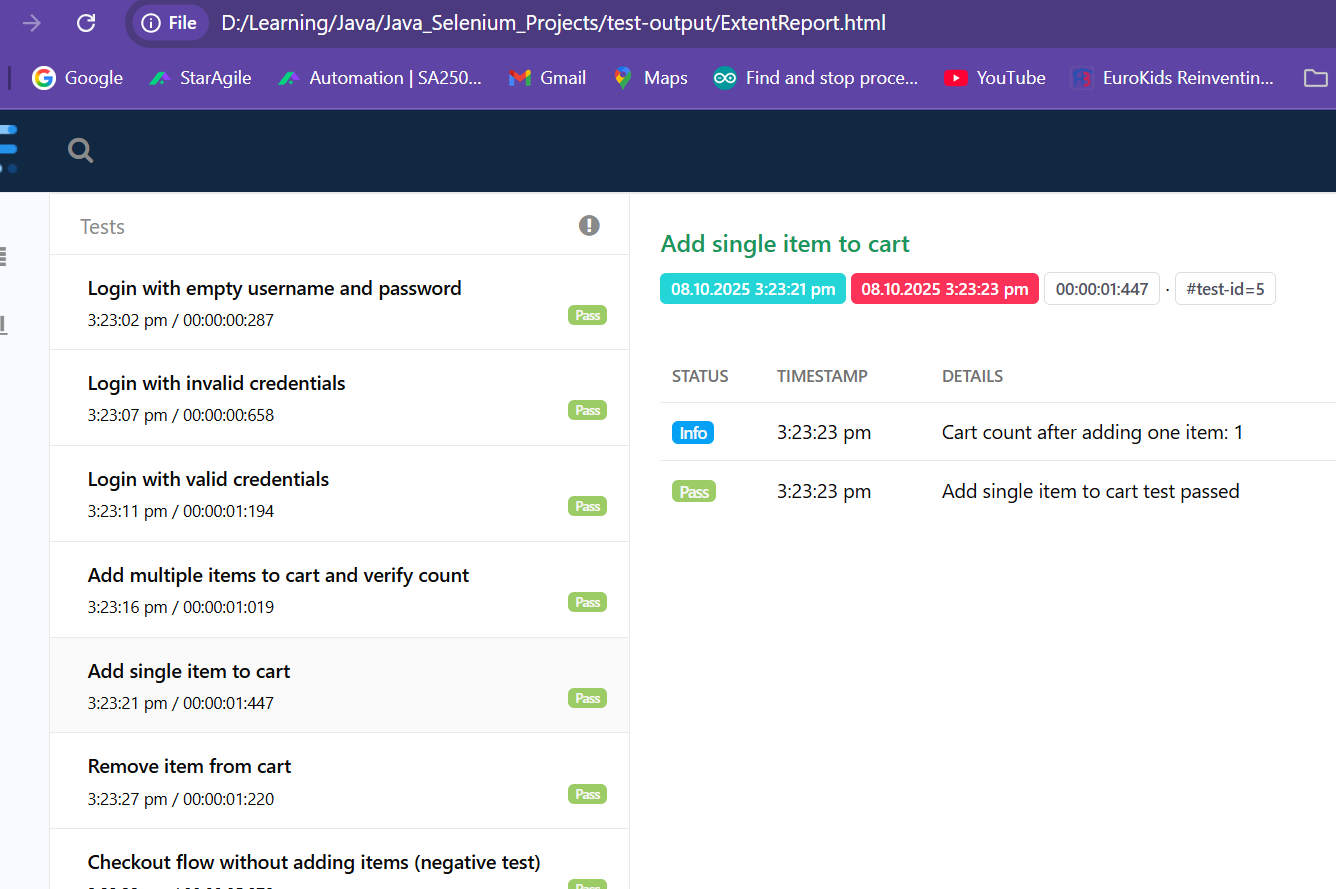


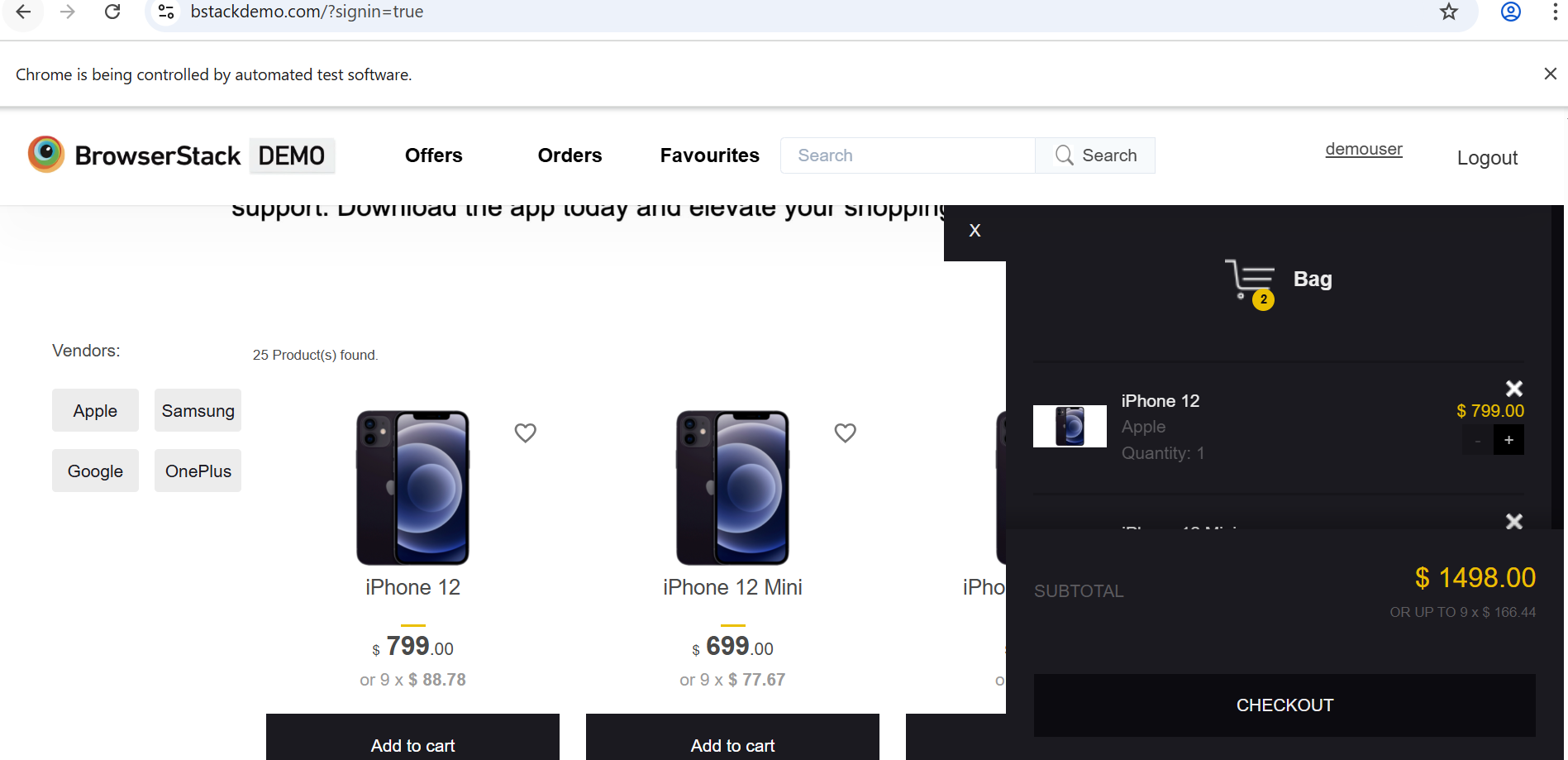
**Cart Tests**

**TC\_004: Add single item to cart**

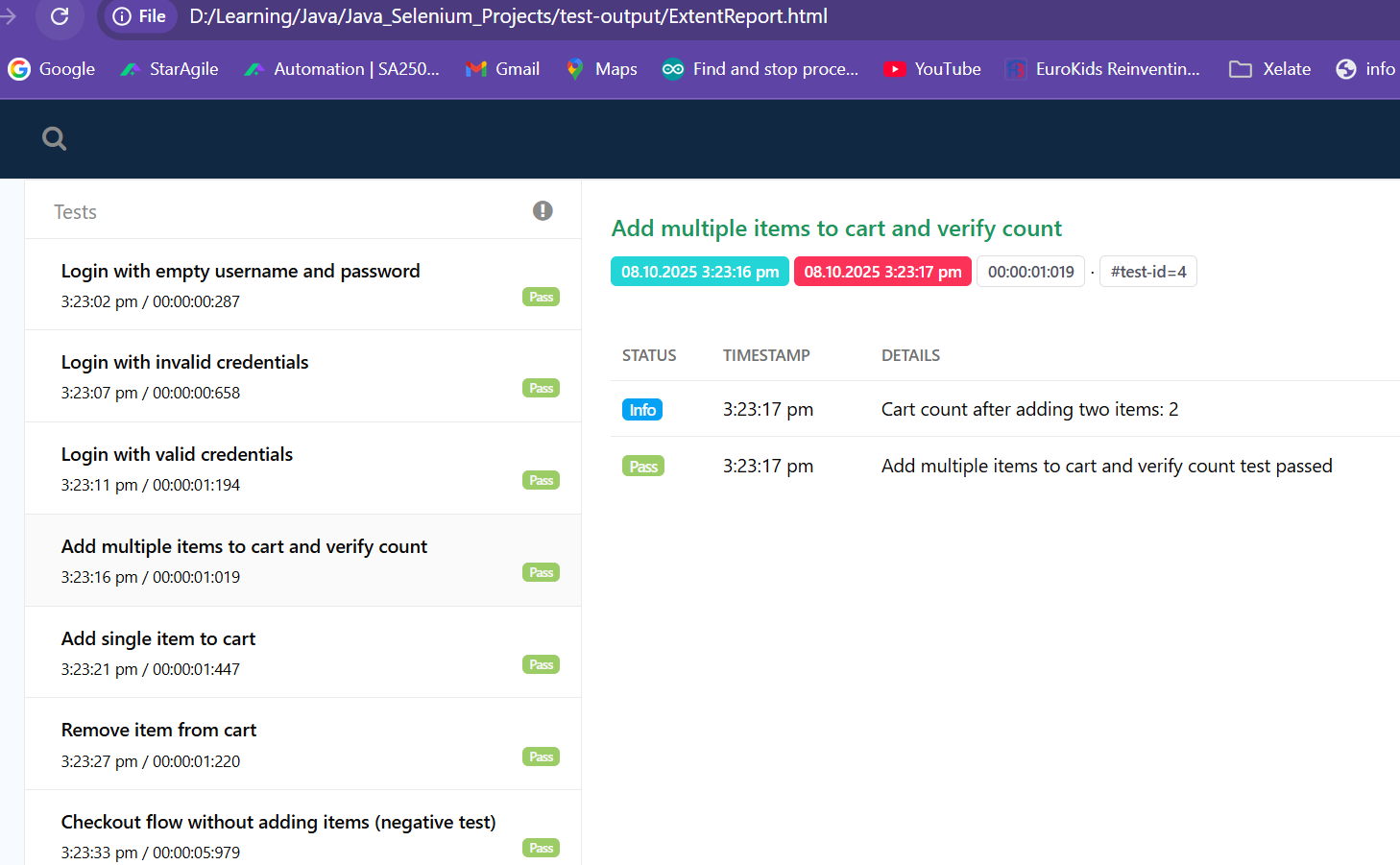


**ExtentReport Screenshot**



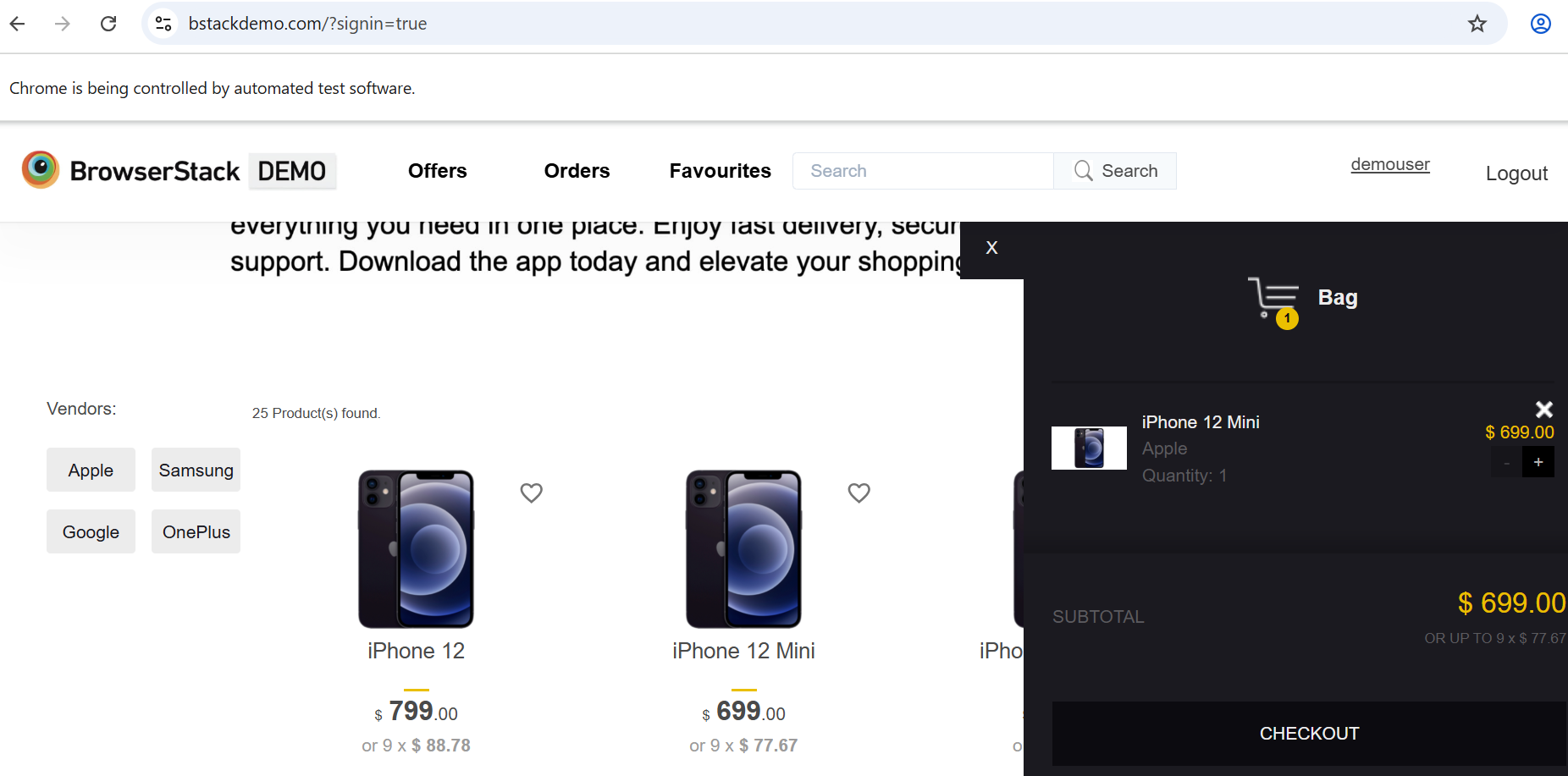
**TC\_005: Add multiple items to cart and verify cart count**

**ExtentReport Screenshot**

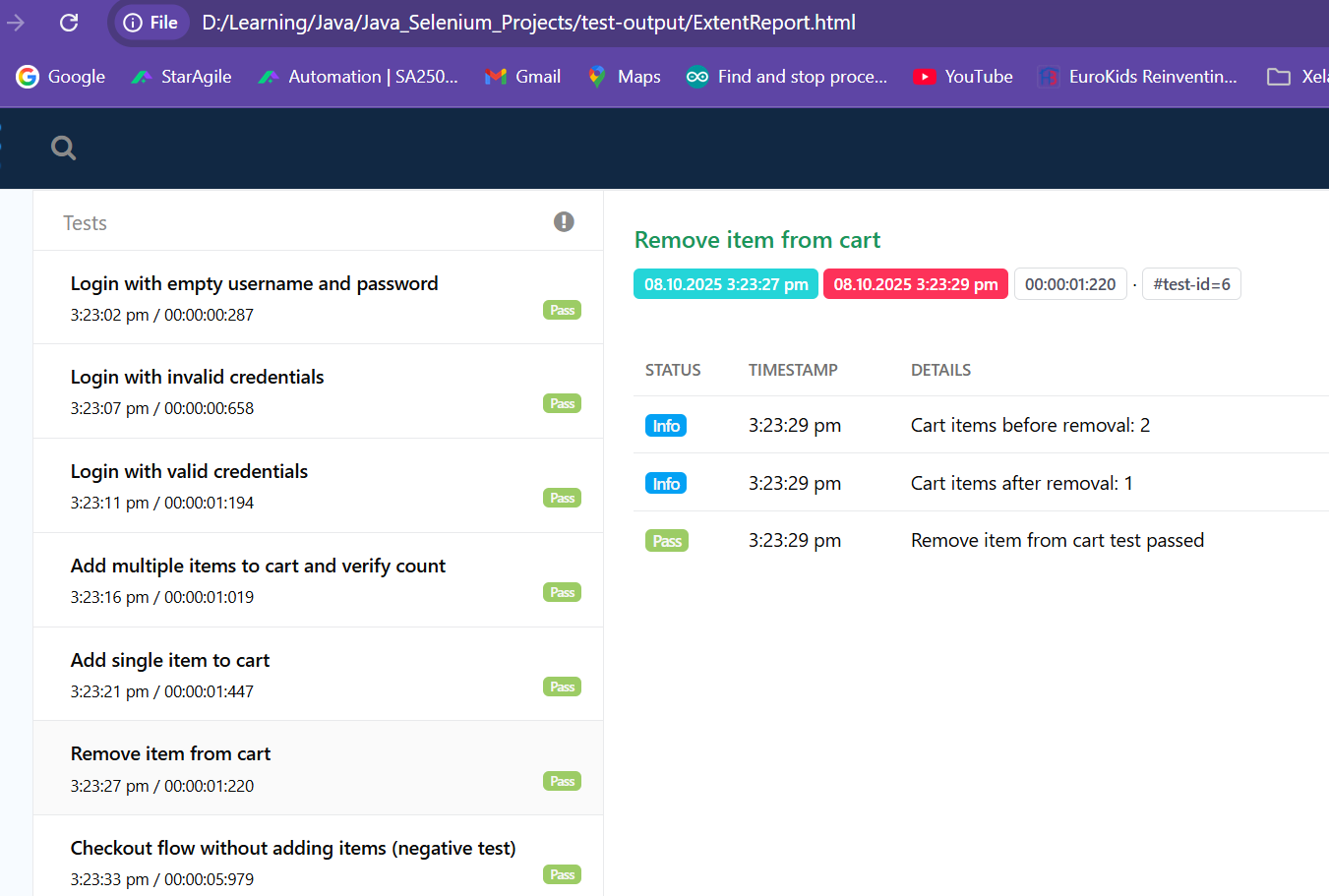


**TC\_006: Remove item from cart**

**(2 items added and one item removed)**

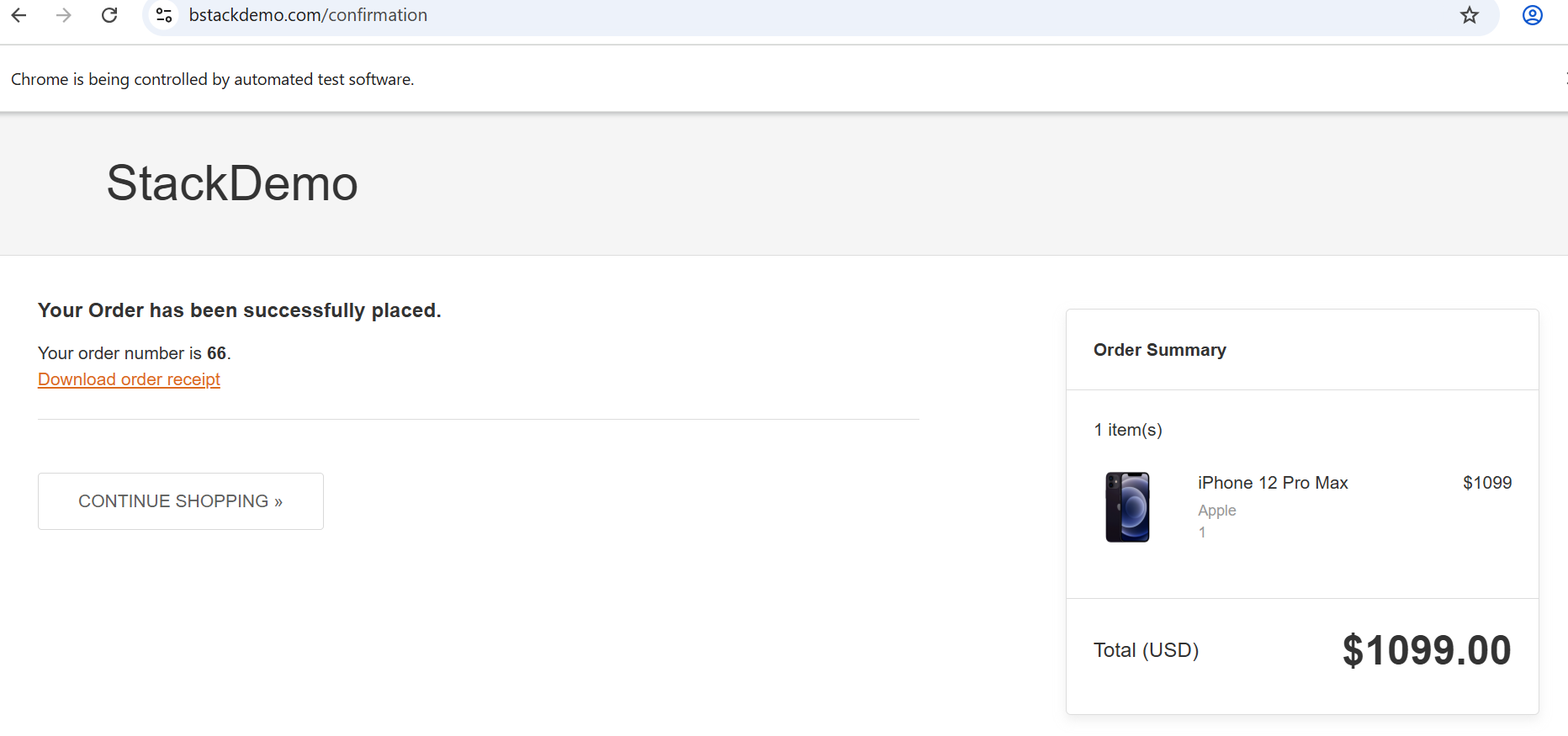


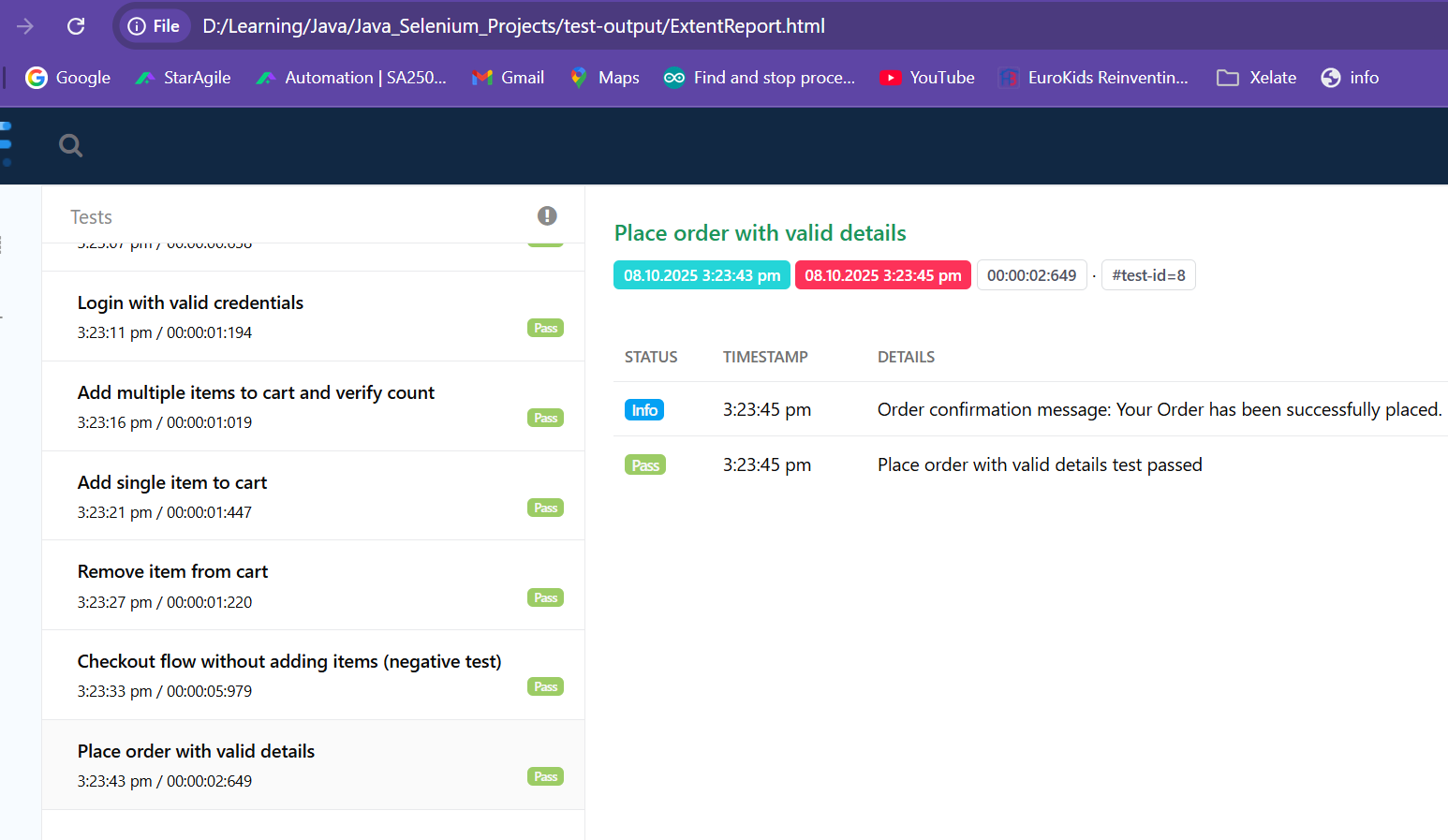
**ExtentReport Screenshot**



**Checkout Tests**

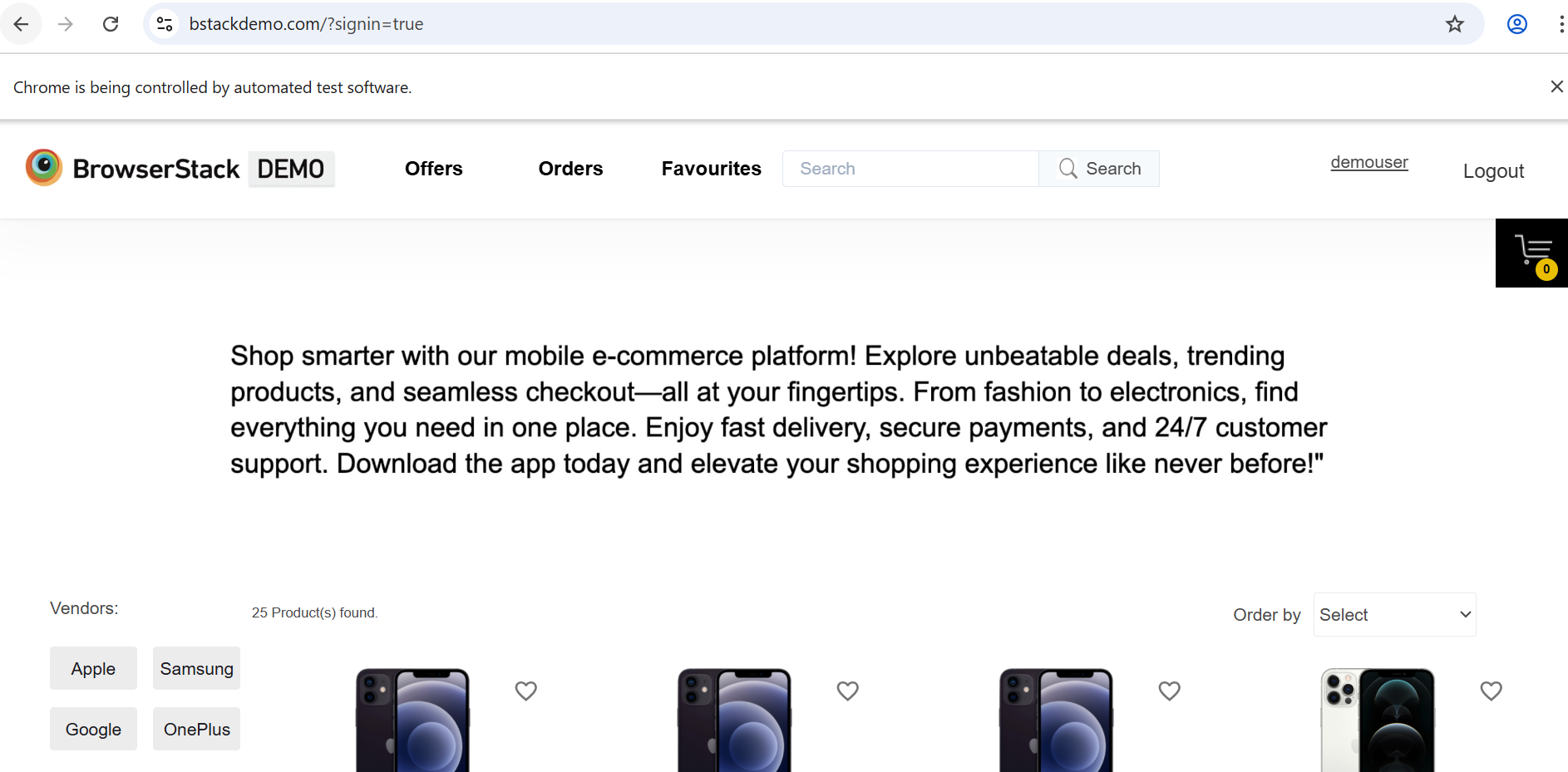
**TC\_007: Place order with valid details**



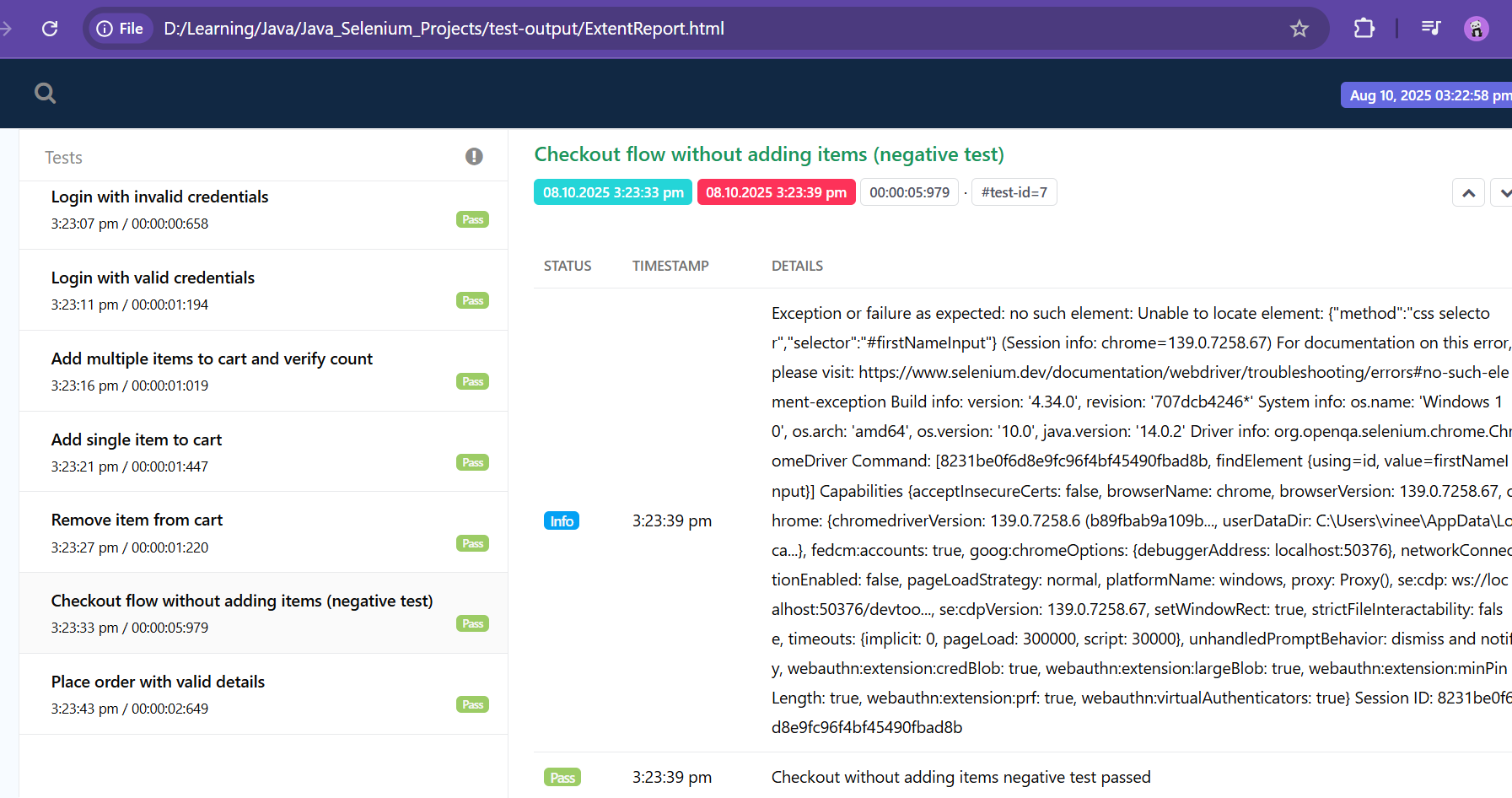
**ExtentReport Screenshot** 

**TC\_008: Checkout flow without adding items (negative test)**

**(Cart window collapses when empty, exception thrown in test case)**



**ExtentReport Screenshot**



**Maven Clean Screenshot**

