**Capstone Project: Create a Testing Framework for Sporty Shoes Website**

**Project objective:**

The objective is to develop a comprehensive QA and test suite for the Sporty Shoes website. The QA effort will require the following:

1. Browser-based end-user testing using Selenium WebDriver with TestNG Framework.
2. Load Testing using JMeter.
3. API Testing with Cucumber.
4. API Testing with Postman and Rest Assured.

The end deliverables will be executable scripts and modules that can be run on demand for testing the Sporty Shoes web app.

**Background of the problem statement:**

Sporty Shoes has an e-commerce website that has the following existing features in place:

* Users can view products.
* If users want to purchase something, they can first sign up and then log in.
* Users can add multiple items to their cart and do a checkout.
* Users have a dashboard that lets them edit their profile, view past purchases, and view their cart.
* Once users do a checkout, the items are cleared from their cart and an order is generated which is stored in their order history.

The above application is already functional. What is needed now is to add a testing layer that will ensure that everything is passed through QA.

**Implementation Requirements**

The following deliverables are expected:

1. Automate the below API endpoints using Rest-Assured

* Retrieve the list of all products in the store.
* Retrieve the list of all registered users.
* Add the product.
* Delete the product.
* Update the product.

1. Create Selenium scripts using TestNG to test all the pages in the web app that will automate:

* Login page
* Registration Page
* Add Product to cart page.
* Place Order Page

1. Create JMeter scripts to do load testing of the homepage and the product detail page.
2. Setup Cucumber in Java Project and write Feature Files using Gherkin to test the API endpoints mentioned in point 1 above.
3. Create Postman scripts to test the following API endpoints:

* Retrieve the list of all products in the store.
* Retrieve the list of all registered users.
* Add the product.
* Delete the product.
* Update the product.

**API Endpoints:**

|  |  |  |
| --- | --- | --- |
| Action | Method | Endpoint |
| Retrieve the list of all products in the store | GET | <http://localhost:9010/get-shoes> |
| Retrieve the list of all registered users | GET | <http://localhost:9010/get-users> |
| Add the product | POST | http://localhost:9010/add-shoe?id=101&image=image\_url&name=SampleShoe&category=Running&sizes=9&price=1000 |
| Delete the product | DELETE | <http://localhost:9010/delete-shoe?id=101> |
| Update the product | PUT | [http://localhost:9010/update-shoe](http://localhost:9010/add-shoe) |

**Add Product URL Sample with POST Method:**

<http://localhost:9010/add-shoe?id=101&image=image_url&name=SampleShoe&category=Running&sizes=9&price=1000>

**Update Product URL Sample with PUT Method:**

**Request body** ([http://localhost:9010/update-shoe](http://localhost:9010/add-shoe))

{

"id": 101,

"name": "Updated Shoe Name",

"category": "Updated Category",

"sizes": "8,9,10",

"price": 1500

"image": "updated\_image\_url",

}

**Delete Product URL Sample with DELETE Method:**

<http://localhost:9010/delete-shoe?id=101>

**You must use the following:**

1. Source code editing and modification: Eclipse IDE
2. End-User Black Box Testing: Selenium WebDriver (A Browser testing framework where only the Java version is used.) with TestNG Framework.
3. Load Testing: JMeter (A load testing application for Java applications.)
4. API Testing: Cucumber (The Gherkin syntax used in Cucumber allows you to define test scenarios in a natural language format)
5. API Testing: Postman (A standalone application that allows testing of API-based services.) and Rest-Assured.
6. Git: To connect and push files from the local system to GitHub
7. GitHub: To store the application code and track its versions.
8. Specification document: Any open-source document or Google Docs

**The following requirements should be met:**

1. All testing scripts and code should be pushed to your GitHub repository. You need to document the steps and write the algorithms in it.
2. The submission of your GitHub repository link is mandatory. In order to track your task, you need to share the link to the repository. You can add a section in your document.
3. Document the step-by-step process starting from creating test cases and then executing them and recording the results.
4. You need to submit the final specification document which should include:

* Project and tester details
* Concepts used in the project.
* Links to the GitHub repository to verify the project completion.
* Your conclusion on enhancing the application and defining the USPs (Unique Selling Points)

**Project to be tested:**

Download the project from <https://github.com/Simplilearn-Edu/SportyShoes>

Then execute:

**java -jar project\_name.jar**

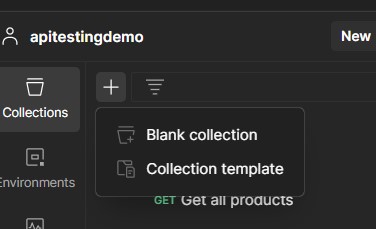
**Source code for capstone Project:**

**Creating a Testing Framework for sporty shoes website:**

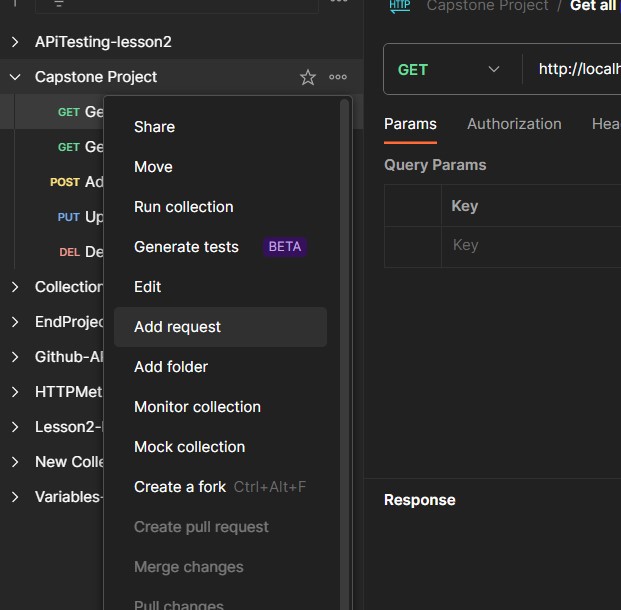
1. Create Postman scripts to test the following API endpoints: ● Retrieve the list of all products in the store.

* Retrieve the list of all registered users.
* Add the product.
* Delete the product.
* Update the product.

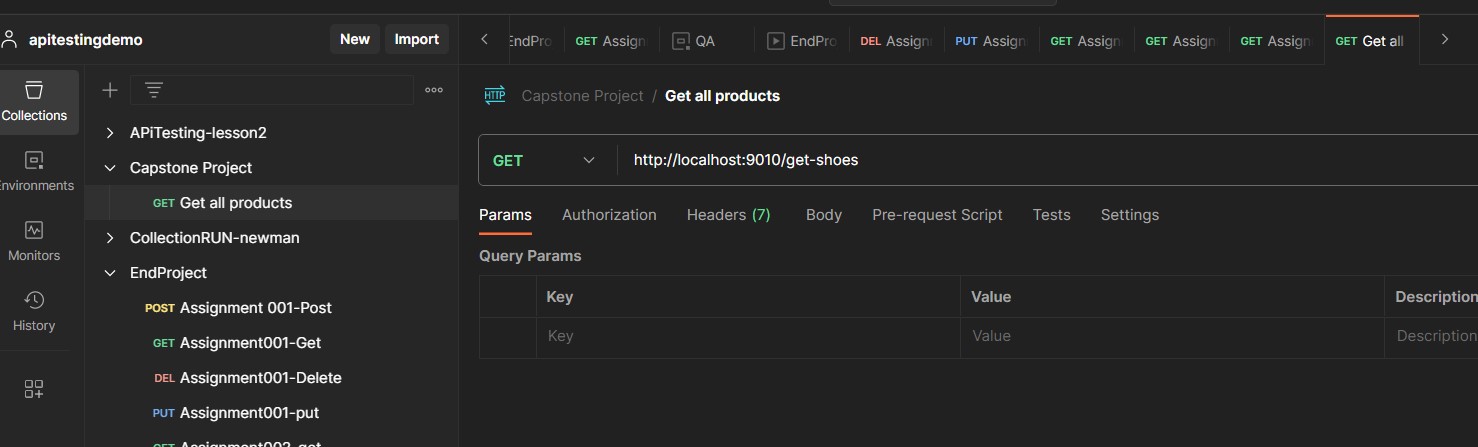
First go to postman application



Click on + symbol here show 2 options select blank collection and put name as capston e project.

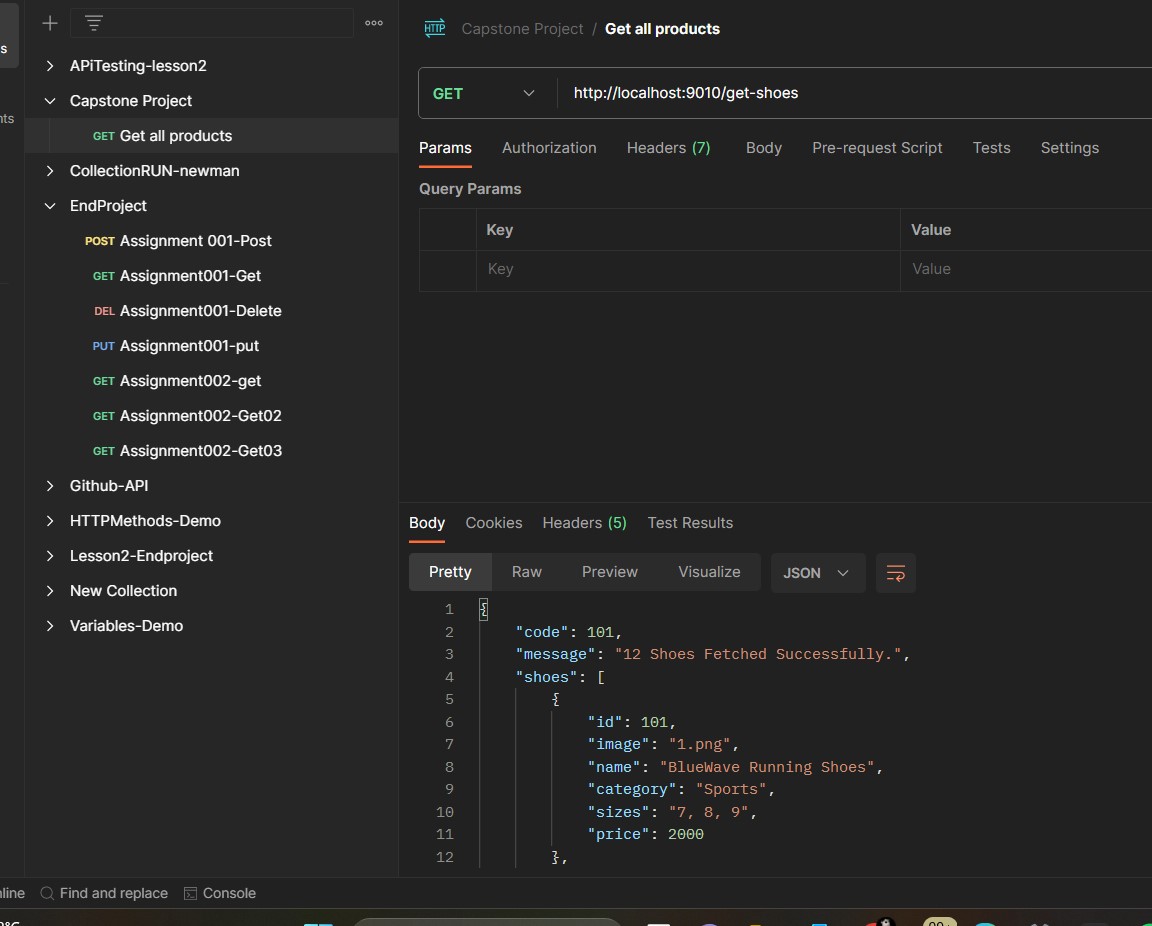


Next step capstone project have 3 dots click on 3 dots have option Add request Put name as Get all Products.



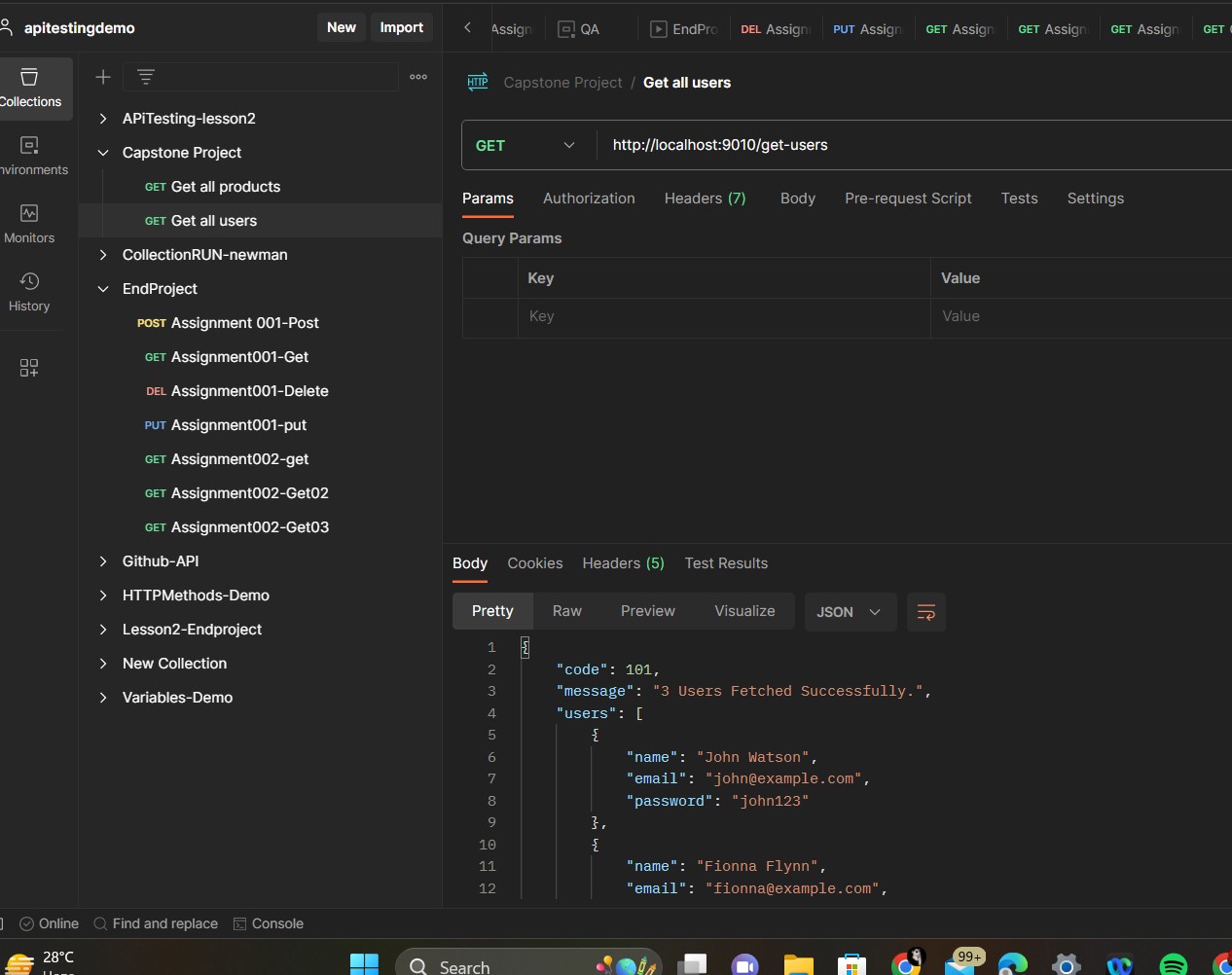
# Enter URL in GET method <http://localhost:9010/get-shoes>

After adding URL click on save button and run this URL



This above picture is the GET all products output

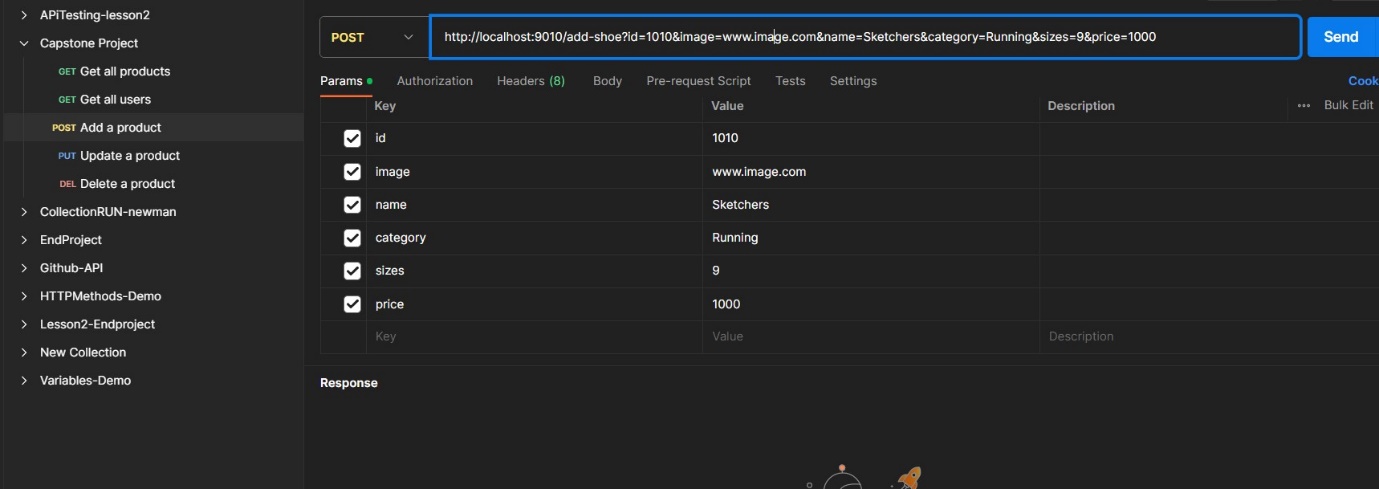
Next select another request put name as GET all users



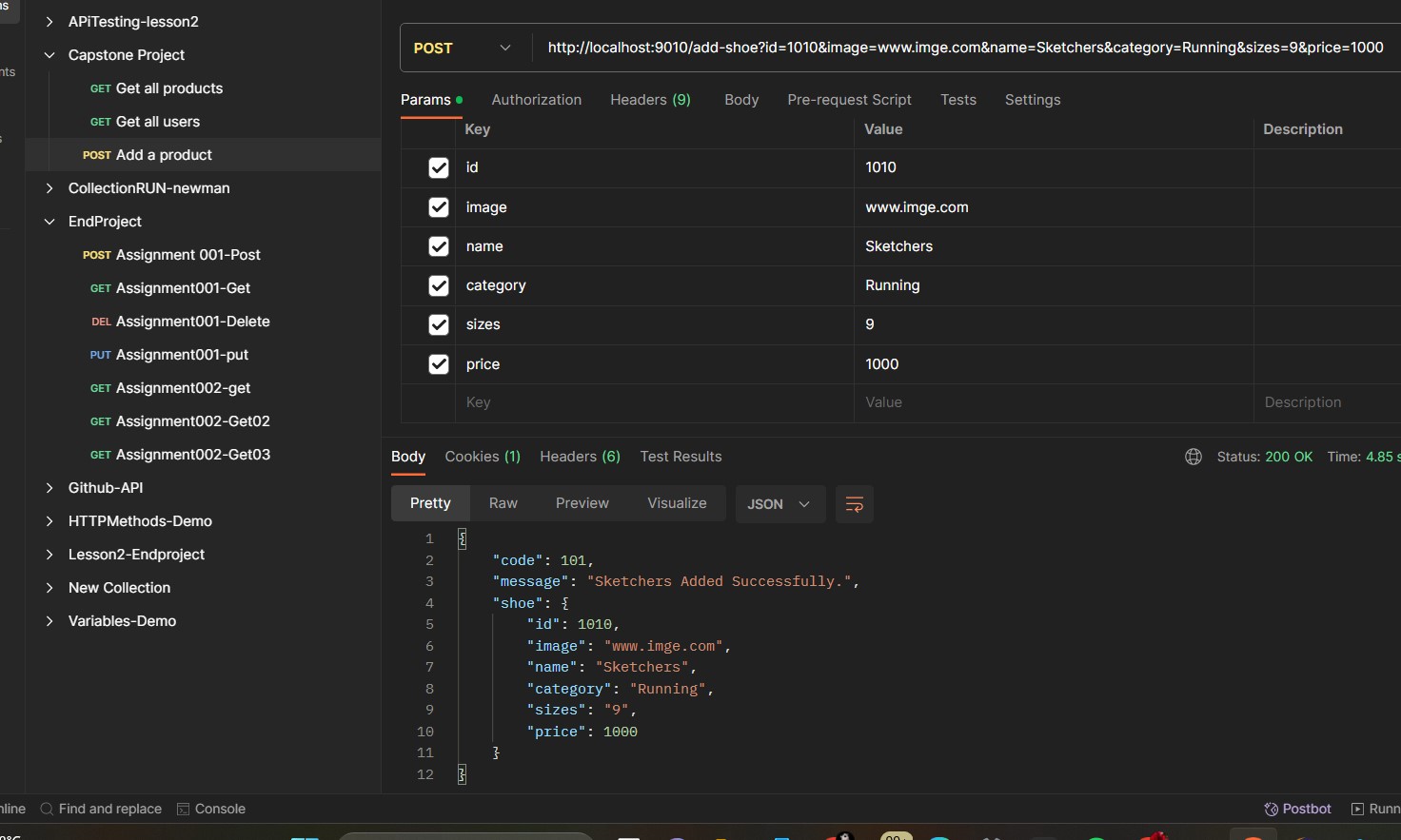
# In GET all users enter URL <http://localhost:9010/get-users>

And click on save and run this URL above picture is get all users output

Next add another Request and enter name as Post add a Product

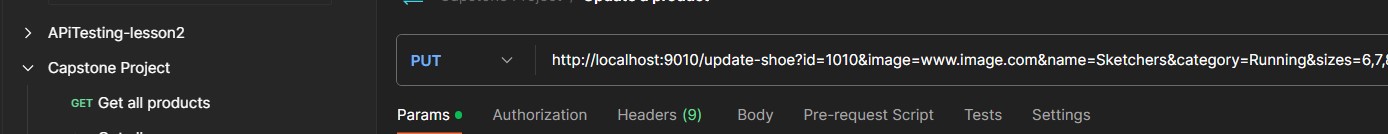


Enter URL select method as POST method click on save and send the request

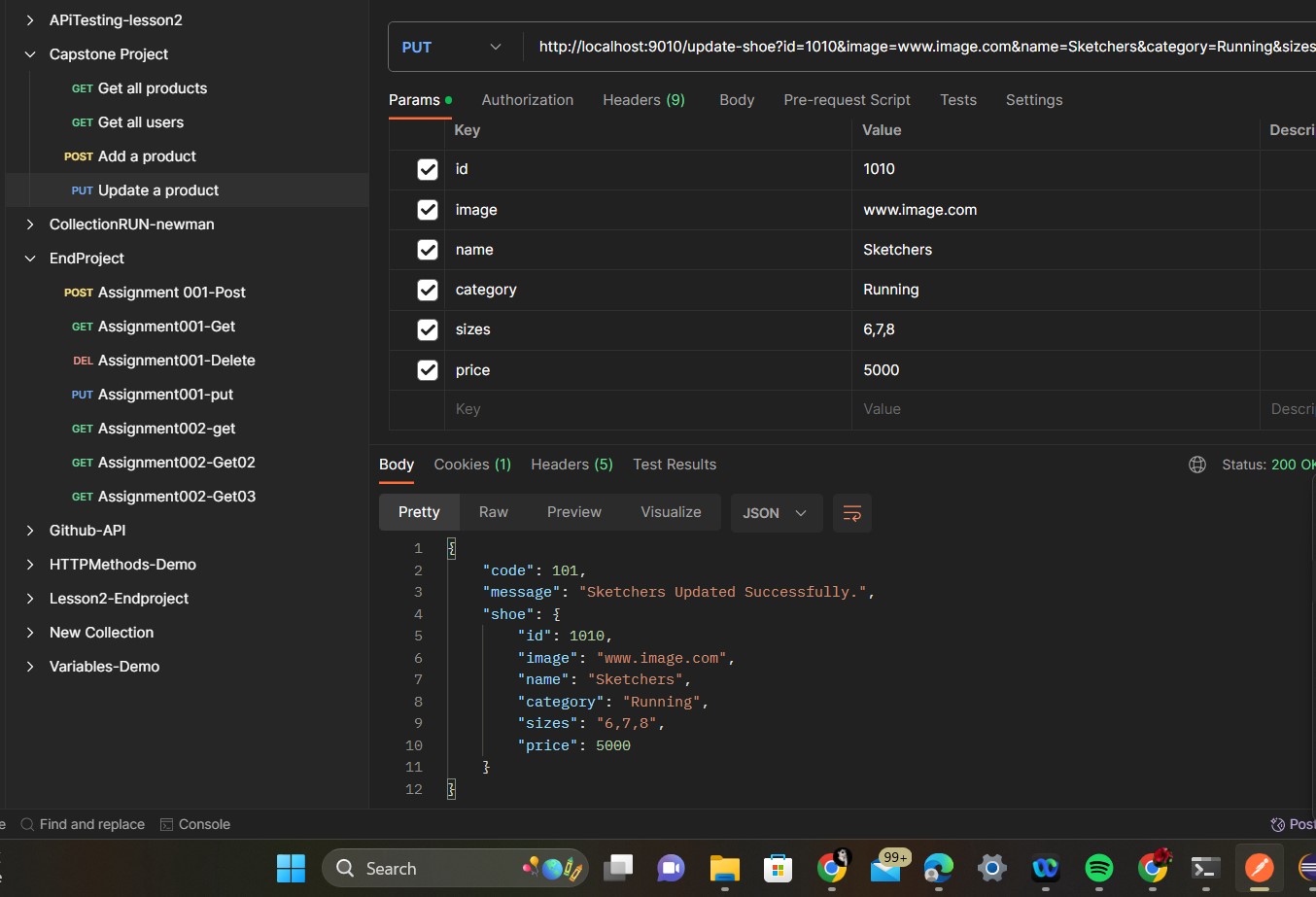


this is the output for POST add a Product.

And then select another method as Put update a Product

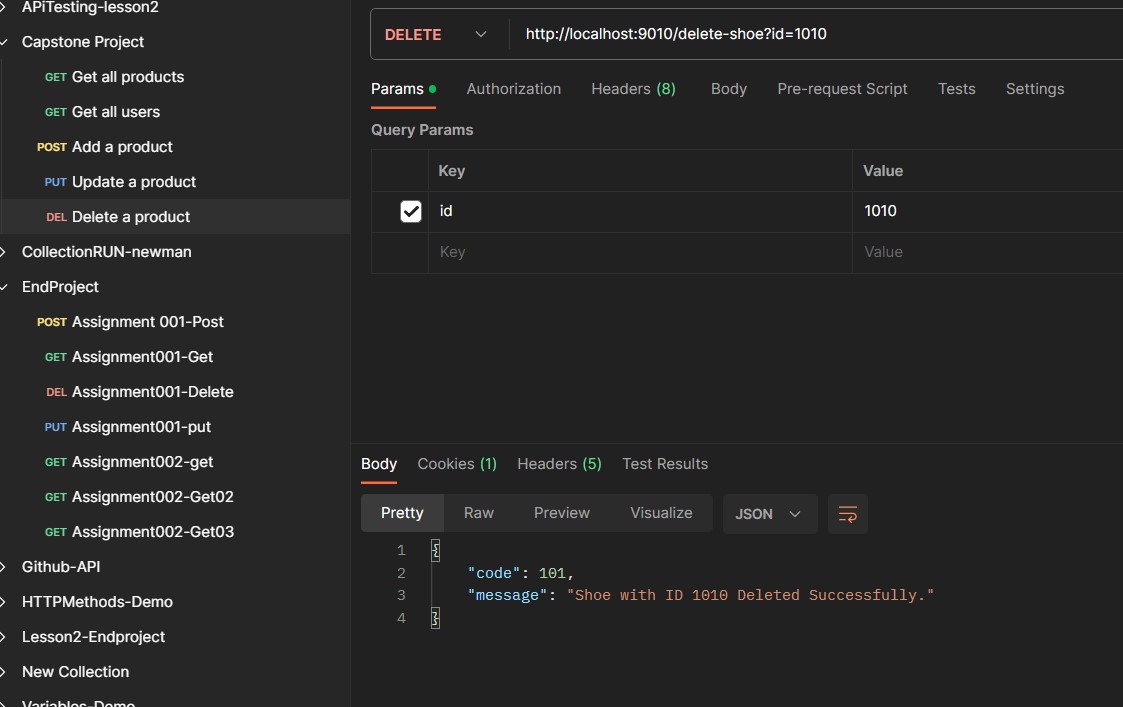


And insert the URL click on save and send Request



This is the output for PUT Update a product

Next select another request Delete a Product



# Enter link as <http://localhost:9010/delete-shoe?id=1010>

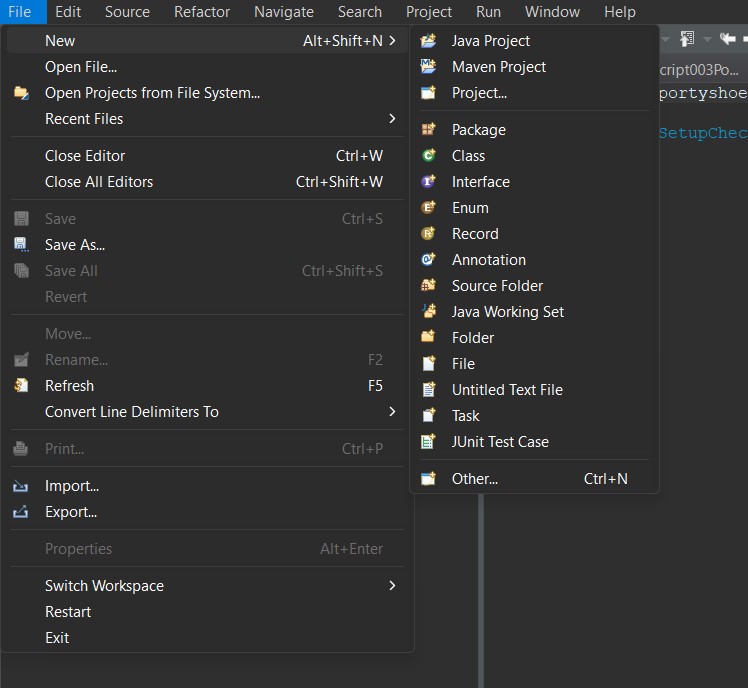
Click on save and run it

2.Automate the below API endpoints using Rest-Assured

● Retrieve the list of all products in the store.

* Retrieve the list of all registered users.
* Add the product.
* Delete the product.
* Update the product.

First go to Eclipse file >new>maven project



And enter name as capstone project

In capstone project add a package com.spotryshoe.restAssuredscripts

Put class name as getallShoes.java Code:

package com.sportyshoe.restAssuredScripts; import org.testng.annotations.Test;

import io.restassured.RestAssured;

public class GetAllshoes {

@Test(priority='1')

public void get\_all\_shoes() {

RestAssured.given()

.baseUri("http://localhost:9010")

.basePath("/get-shoes")

.when()

.get()

.then()

.statusCode(200)

.log().all();

}

@Test(priority='2')

public void get\_all\_users() {

RestAssured.given()

.baseUri("http://localhost:9010")

.basePath("/get-users")

.when()

.get()

.then()

.statusCode(200)

.log().all();

}

}

**Output:**

|  |
| --- |
| RemoteTestNG] detected TestNG version 7.7.1  SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".  SLF4J: Defaulting to no-operation (NOP) logger implementation  SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details. HTTP/1.1 200  Content-Type: application/json  Transfer-Encoding: chunked  Date: Tue, 14 Nov 2023 11:54:20 GMT  Keep-Alive: timeout=60  Connection: keep-alive    {  "code": 101,  "message": "12 Shoes Fetched Successfully.", "shoes": [  {  "id": 101,  "image": "1.png",  "name": "BlueWave Running Shoes",  "category": "Sports",  "sizes": "7, 8, 9",  "price": 2000  },  {  "id": 201,  "image": "2.png",  "name": "Elegant Leather Loafers",  "category": "Formal",  "sizes": "7, 8, 9", |

|  |
| --- |
| "price": 3000  },  {  "id": 301,  "image": "3.png",  "name": "NeoFlex Athletic Shoes",  "category": "Sports",  "sizes": "7, 8, 9, 10",  "price": 4500  },  {  "id": 401,  "image": "4.png",  "name": "PowerStride Training Shoes",  "category": "Sports",  "sizes": "6, 7, 8, 9",  "price": 6000  },  {  "id": 501,  "image": "5.png",  "name": "StreetRunner Urban Sneakers",  "category": "Sports",  "sizes": "7, 9",  "price": 4000  },  {  "id": 601,  "image": "6.png",  "name": "VentureHike Trail Shoes",  "category": "Sports",  "sizes": "5, 8, 9",  "price": 2500  },  {  "id": 701,  "image": "7.png",  "name": "EnduraGrip Sports Sneakers",  "category": "Sports",  "sizes": "4, 6, 9",  "price": 5000  },  {  "id": 801,  "image": "8.png",  "name": "LightStride Performance Shoes",  "category": "Sports",  "sizes": "7, 8",  "price": 1200  },  {  "id": 901,  "image": "9.png",  "name": "MaxFit Pro Sports Shoes",  "category": "Sports",  "sizes": "7, 8, 9, 10",  "price": 4700  },  {  "id": 111,  "image": "10.png",  "price": 4700  },  {  "id": 111,  "image": "10.png",  "name": "RapidFlex Running Shoes",  "category": "Sports",  "sizes": "4, 5, 6",  "price": 2500  },  {  "id": 211,  "image": "11.png",  "name": "ZenFlex Sports Sneakers",  "category": "Sports",  "sizes": "7, 8, 9",  "price": 6000  },  {  "id": 311,  "image": "12.png",  "name": "TrailBlaze Adventure Shoes",  "category": "Sports",  "sizes": "5, 6, 9",  "price": 7500  }  ]  }  HTTP/1.1 200  Content-Type: application/json  Transfer-Encoding: chunked  Date: Tue, 14 Nov 2023 11:54:20 GMT  Keep-Alive: timeout=60  Connection: keep-alive    {  "code": 101,  "message": "3 Users Fetched Successfully.",  "users": [  {  "name": "John Watson",  "email": "john@example.com",  "password": "john123"  },  {  "name": "Fionna Flynn",  "email": "fionna@example.com",  "password": "fionna123"  },  {  "name": "Sia Sen",  "email": "sia@example.com",  "password": "sia123"  }  ] }  PASSED: com.sportyshoe.restAssuredScripts.GetAllshoes.get\_all\_users PASSED: com.sportyshoe.restAssuredScripts.GetAllshoes.get\_all\_shoes  ===============================================  Default test  Tests run: 2, Failures: 0, Skips: 0  ===============================================    =============================================== |

|  |
| --- |
| Default suite  Total tests run: 2, Passes: 2, Failures: 0, Skips: 0  =============================================== |

And Create another class as postandputnew shoe Code: package com.sportyshoe.restAssuredScripts; import org.testng.annotations.Test;

import io.restassured.RestAssured;

public class PostandPutnewshoe { @Test(priority='1') public void add\_new\_product()

{

RestAssured.given()

.baseUri("http://localhost:9010")

.basePath("/add-shoe")

.queryParam("id","1020")

.queryParam("image", "www.imge.com")

.queryParam("name","Nike")

.queryParam("category", "Running")

.queryParam("sizes","5,6,7")

.queryParam("price", "2000")

.when()

.post()

.then()

.log().all();

}

@Test(priority='2') public void update\_a\_product()

{

RestAssured.given()

.baseUri("http://localhost:9010")

.basePath("/update-shoe")

.queryParam("id","1020")

.queryParam("image", "www.imge123.com")

.queryParam("name","Reebok")

.queryParam("category", "Running")

.queryParam("sizes","5,6,7")

.queryParam("price", "2500")

.when()

.put()

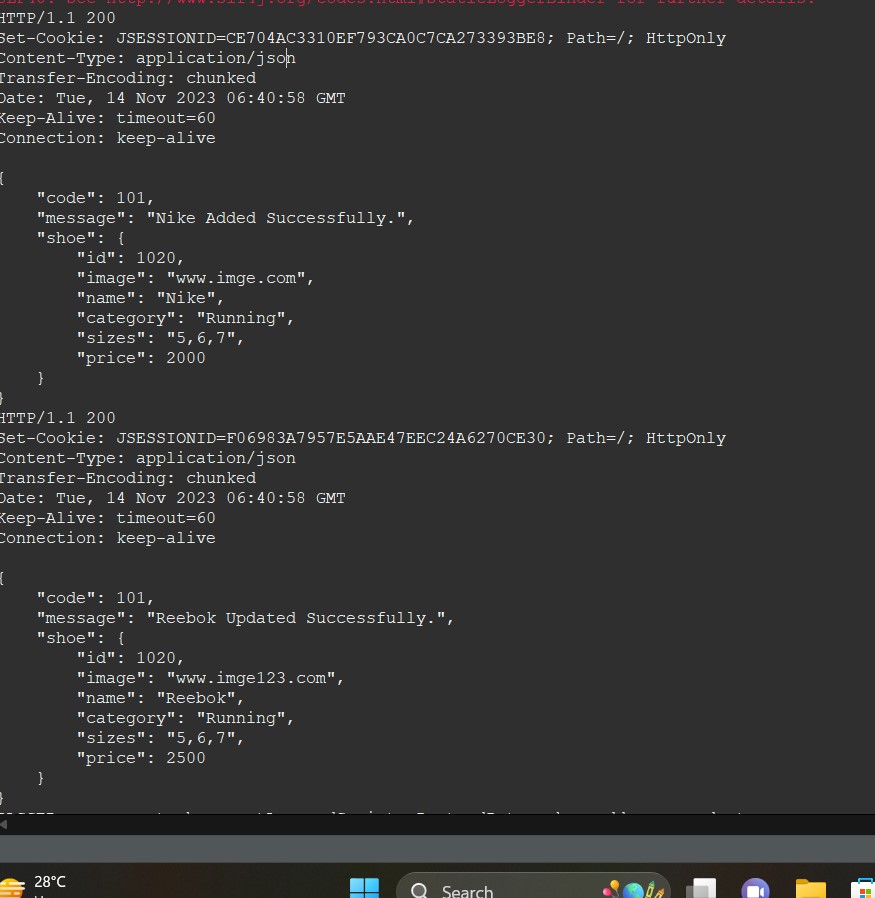
.then()

.log().all();

}

}

Output:



Next add delete Rest assured code

Code:

package com.sportyshoe.restAssuredScripts;

import org.testng.annotations.Test;

import io.restassured.RestAssured;

public class Deleteshoe {

@Test(priority='1') public void delete\_product()

{

RestAssured.given()

.baseUri("http://localhost:9010")

.basePath("/delete-shoe")

.queryParam("id", "1010")

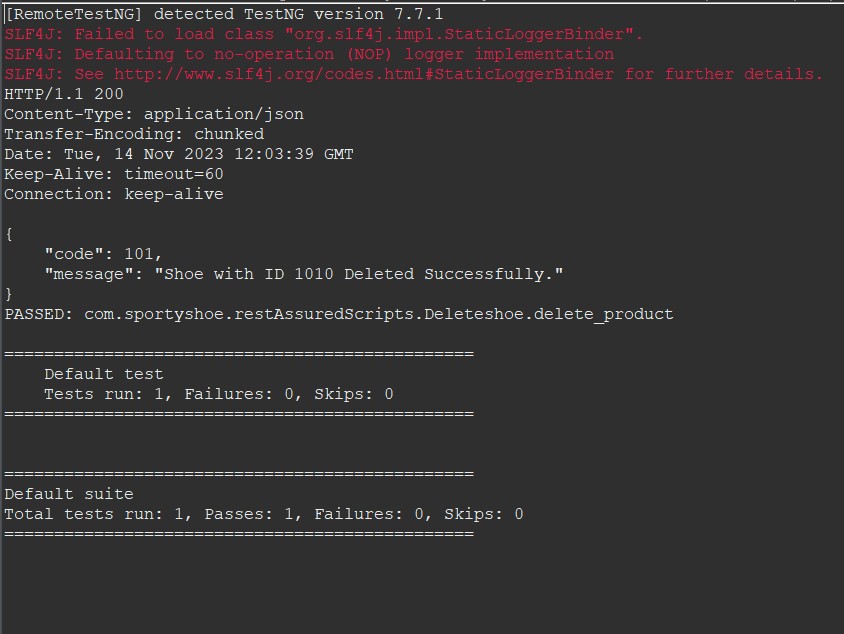
.when().delete()

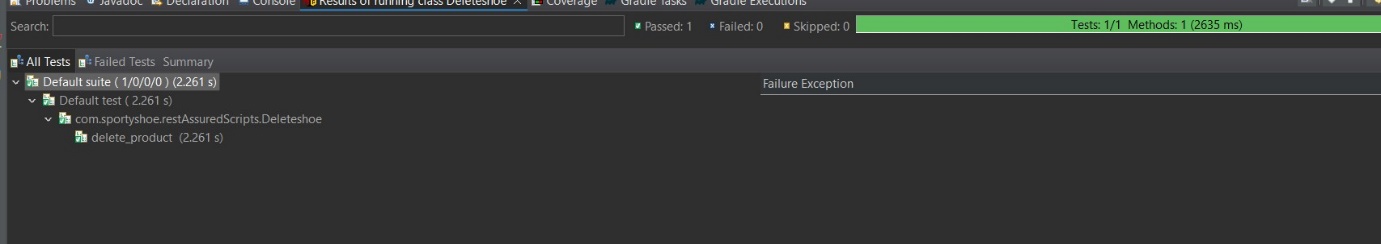
.then().log().all();

}

}

Output:





Selenium Scripts:

3.Create Selenium scripts using TestNG to test all the pages in the web app that will automate:

* Login page
* Registration Page
* Add Product to cart page.
* Place Order Page

First go to eclipse create a package com.sportyshoe.seleniumcucumberscripts Under class setupcheck.java

Code: package com.sportyshoe.SeleniumCucumberScripts; import org.openqa.selenium.By; import org.openqa.selenium.WebDriver; import org.openqa.selenium.chrome.ChromeDriver; import org.testng.annotations.Test;

public class SetupCheck {

@Test

public void test\_setup\_selenium()

{

WebDriver driver = new ChromeDriver();

driver.manage().window().maximize(); driver.get("http://localhost:9010/");

String text = driver.findElement(By.xpath("//div[@class='container mt-3']/descendant::p[1]")).getText();

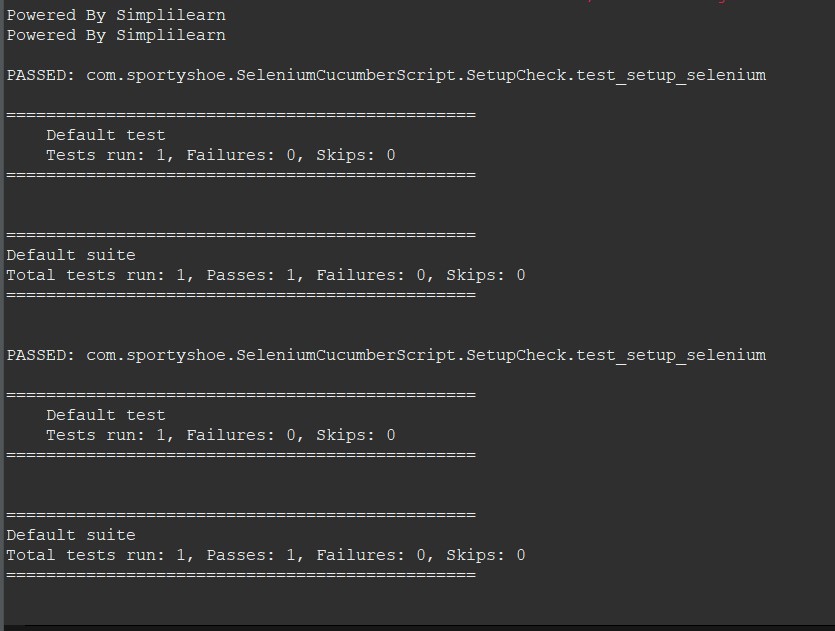
System.out.println(text);

System.out.println(driver.getTitle());

}

}

Output:



Next go to another class name as TestBase

Code: package com.sportyshoe.SeleniumCucumberScripts;

import org.openqa.selenium.WebDriver; import org.openqa.selenium.chrome.ChromeDriver; import org.openqa.selenium.firefox.FirefoxDriver;

public class TestBase {

public static WebDriver driver;

public static void OpenBrowser(String browser)

{

if(browser == "Chrome")

{

driver = new ChromeDriver();

}

if(browser == "FireFox")

{

driver = new FirefoxDriver();

}

driver.manage().window().maximize(); driver.manage().deleteAllCookies(); driver.get("http://localhost:9010/");

}

public static void closebrowser()

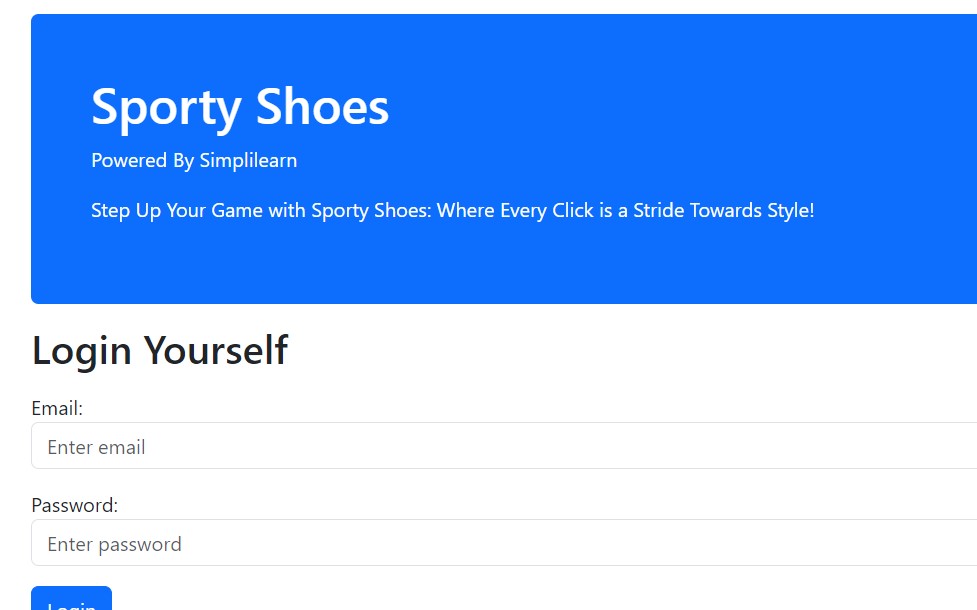
{

driver.close();

}

}

**Output:**



Next go to another class name as HomePage

Code:

package com.sportyshoe.SeleniumCucumberScripts;

import org.openqa.selenium.WebDriver; import org.openqa.selenium.WebElement; import org.openqa.selenium.support.FindBy; import org.openqa.selenium.support.PageFactory;

public class HomePage extends TestBase {

@FindBy(xpath="//div[@class='container mt-3']/descendant::p[1]")

WebElement text;

@FindBy(linkText="New User? Register Here")

WebElement registerLink;

public HomePage(WebDriver driver) {

PageFactory.initElements(driver, this);

}

public String getURL\_page()

{

String URLnew = driver.getCurrentUrl();

return URLnew;

}

public String Validate\_Text\_On\_Page()

{

String pageText = text.getText(); System.out.println(pageText); return pageText;

}

public void click\_register\_link()

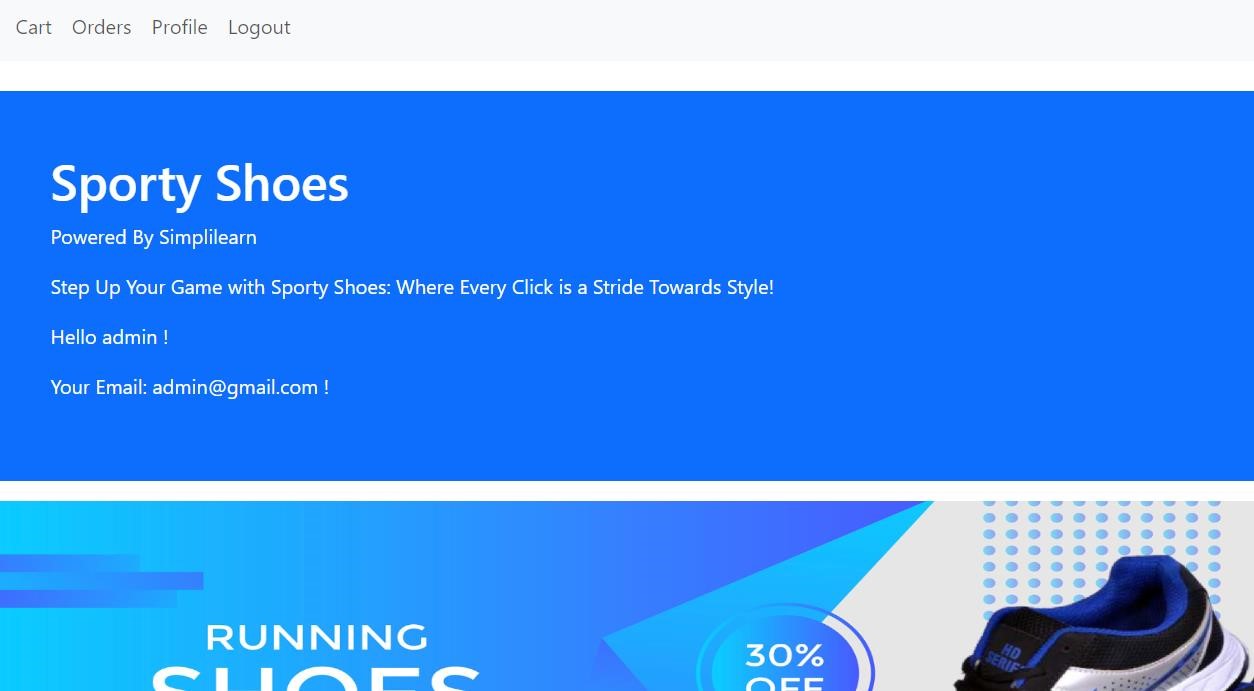
{

registerLink.click();

}

}

**Output:**



Next add another class Login Page under same package:

Code:

package com.sportyshoe.SeleniumCucumberScripts;

import org.openqa.selenium.WebDriver; import org.openqa.selenium.WebElement; import org.openqa.selenium.support.FindBy; import org.openqa.selenium.support.PageFactory;

public class LoginPage {

@FindBy(xpath="//input[@id='email']")

WebElement loginEmail;

@FindBy(xpath="//input[@id='password']")

WebElement loginpassword;

@FindBy(xpath="//button[@type='submit']")

WebElement loginbtn;

@FindBy(linkText="Cart")

WebElement clickCart;

public LoginPage(WebDriver driver) {

PageFactory.initElements(driver, this);

}

public void user\_login()

{

loginEmail.sendKeys("admin@gmail.com"); loginpassword.sendKeys("rishikesh@123"); loginbtn.click();

}

public void click\_cart()

{

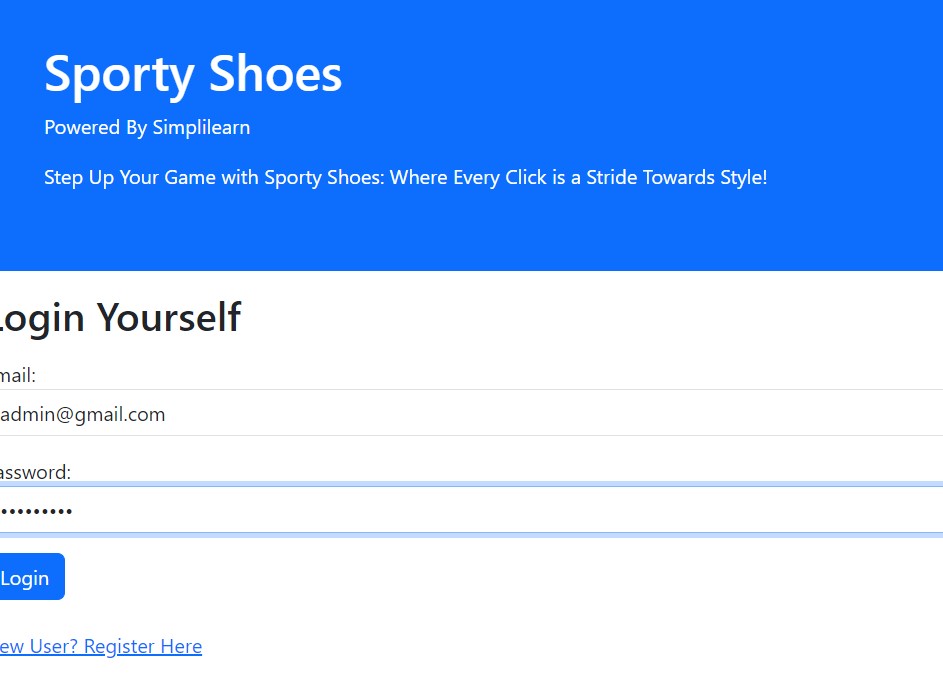
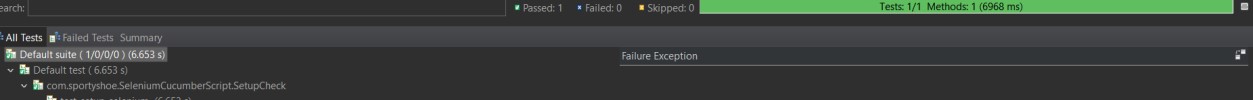
clickCart.click();

}

}

}

**Output:**



Next add another class under same package name as Register Page package com.sportyshoe.SeleniumCucumberScripts;

import org.openqa.selenium.WebDriver; import org.openqa.selenium.WebElement; import org.openqa.selenium.support.FindBy; import org.openqa.selenium.support.PageFactory;

public class RegisterPage extends TestBase{

@FindBy(xpath="//input[@id='name']")

WebElement registername;

@FindBy(xpath="//input[@id='email']")

WebElement registeremail;

@FindBy(xpath="//input[@id='password']")

WebElement registerpassword;

@FindBy(xpath="//button[@type='submit']")

WebElement registerBtn;

@FindBy(xpath="//div[@class='mt-4 p-5 bg-primary text-white rounded']/descendant::p[3]")

WebElement userText;

public RegisterPage(WebDriver driver) {

PageFactory.initElements(driver, this);

}

public void register\_user()

{

registername.sendKeys("rishikesh"); registeremail.sendKeys("rishikesh@gmail.com"); registerpassword.sendKeys("rishikesh@123"); registerBtn.click();

}

public String validate\_registration\_URL()

{

String register\_url = driver.getCurrentUrl();

return register\_url;

}

public String validate\_registration\_Text()

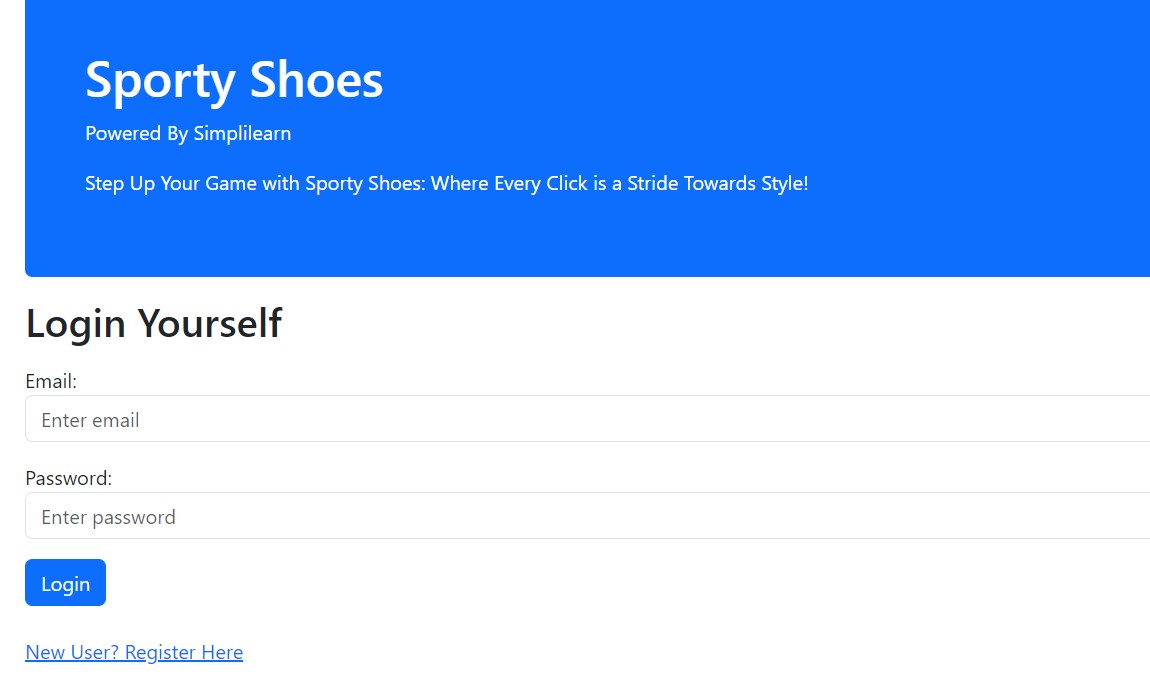
{

String user\_name = userText.getText();

return user\_name;

}

Output:



Next add another class name as orderPage under the same package package com.sportyshoe.SeleniumCucumberScripts;

import org.openqa.selenium.WebDriver; import org.openqa.selenium.WebElement; import org.openqa.selenium.support.FindBy; import org.openqa.selenium.support.PageFactory;

public class OrderPage {

@FindBy(linkText="Orders")

WebElement orderlink;

public OrderPage(WebDriver driver) {

PageFactory.initElements(driver, this);

}

public void click\_orderPage()

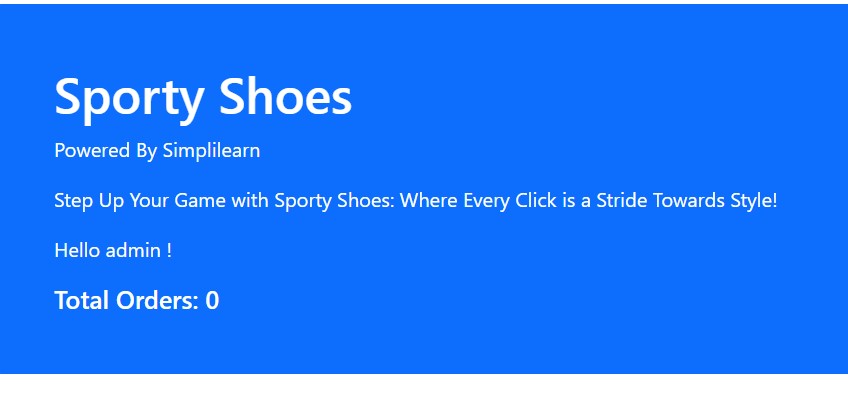
{

orderlink.click();

}

}

**Output:**



Next step add another class name as AddCart Page under same package package com.sportyshoe.SeleniumCucumberScripts;

import org.openqa.selenium.JavascriptExecutor; import org.openqa.selenium.WebDriver; import org.openqa.selenium.WebElement; import org.openqa.selenium.support.FindBy; import org.openqa.selenium.support.PageFactory; public class AddtoCartPage {

@FindBy(xpath="//a[@id=\"shoe101\"]")

WebElement viewShoeBTN;

@FindBy(xpath = "//a[@id='cart101']")

WebElement addtocartBTN;

@FindBy(xpath="//div[@class='alert alert-success']/descendant::p[1]")

WebElement successText;

JavascriptExecutor executor;

public AddtoCartPage(WebDriver driver) { PageFactory.initElements(driver, this); executor = (JavascriptExecutor) driver;

}

public void add\_product\_to\_cart() throws InterruptedException

{

executor.executeScript("arguments[0].click();", addtocartBTN);

}

public String validate\_success\_message()

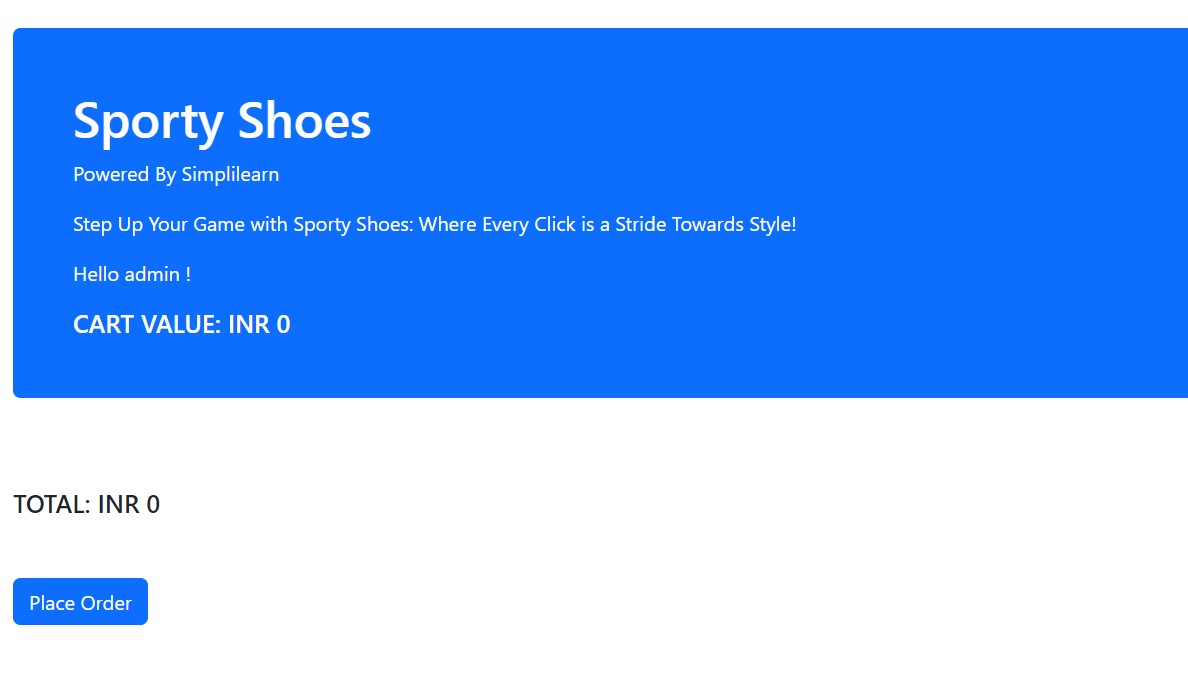
{

String successtext = successText.getText(); return successtext;

}

}

**Output:**



Next go to create another package name com.sportyshoe.Tests

Under class name as orderPageTests

Code:

package com.sportyshoe.Tests;

import org.testng.Assert; import org.testng.annotations.BeforeTest; import org.testng.annotations.Test; import com.sportyshoe.SeleniumCucumberScripts.HomePage; import com.sportyshoe.SeleniumCucumberScripts.LoginPage; import com.sportyshoe.SeleniumCucumberScripts.OrderPage; import com.sportyshoe.SeleniumCucumberScripts.RegisterPage; import com.sportyshoe.SeleniumCucumberScripts.TestBase;

public class OrderpageTest extends TestBase {

HomePage hp;

RegisterPage rp;

LoginPage lp;

OrderPage op;

@BeforeTest

public void start\_browser()

{

OpenBrowser("Chrome"); hp = new HomePage(driver); rp = new RegisterPage(driver); lp = new LoginPage(driver); op = new OrderPage(driver);

}

@Test(priority='1')

public void test\_login()

{

lp.user\_login();

}

@Test(priority='2')

public void test\_click\_orders()

{

op.click\_orderPage();

}

@Test(priority='3') public void test\_getTitle\_page()

{

String expected = "http://localhost:9010/orders";

String Actual = hp.getURL\_page();

Assert.assertEquals(Actual, expected);

}

}

Next class AddCartPage under same package package com.sportyshoe.Tests;

import org.testng.Assert; import org.testng.annotations.BeforeTest; import org.testng.annotations.Test;

import com.sportyshoe.SeleniumCucumberScripts.AddtoCartPage; import com.sportyshoe.SeleniumCucumberScripts.HomePage; import com.sportyshoe.SeleniumCucumberScripts.LoginPage; import com.sportyshoe.SeleniumCucumberScripts.RegisterPage; import com.sportyshoe.SeleniumCucumberScripts.TestBase;

public class AddtoCartPageTest extends TestBase {

HomePage hp;

RegisterPage rp;

LoginPage lp;

AddtoCartPage ac;

@BeforeTest

public void start\_browser()

{

OpenBrowser("Chrome"); hp = new HomePage(driver); rp = new RegisterPage(driver); lp = new LoginPage(driver); ac = new AddtoCartPage(driver);

}

@Test(priority='1') public void test\_login()

{

lp.user\_login();

}

@Test(priority='2') public void test\_add\_product\_to\_cart() throws InterruptedException

{

ac.add\_product\_to\_cart();

}

@Test(priority='3')

public void test\_validate\_success\_message()

{

String expected = "Message:Shoe BlueWave Running Shoes Added Successfully to

Cart";

String actualText= ac.validate\_success\_message();

Assert.assertEquals(actualText, expected);

}

}

Next go to another class name HomePageTests

Code:

package com.sportyshoe.Tests;

import org.testng.annotations.AfterTest; import org.testng.annotations.BeforeTest; import org.testng.annotations.Test;

import com.sportyshoe.SeleniumCucumberScripts.HomePage; import com.sportyshoe.SeleniumCucumberScripts.TestBase;

import static org.testng.Assert.assertEquals;

import org.testng.Assert; import org.testng.Assert.\*;

public class HomePageTest extends TestBase {

HomePage hp;

@BeforeTest public void start\_browser()

{

OpenBrowser("Chrome"); hp = new HomePage(driver);

}

@Test(priority='1') public void test\_getTitle\_page()

{

String expected = "http://localhost:9010/";

String Actual = hp.getURL\_page();

Assert.assertEquals(Actual, expected);

}

@Test(priority='2')

public void Test\_Validate\_Text\_On\_Page()

{

String expected = "Powered By Simplilearn";

String actualText = hp.Validate\_Text\_On\_Page();

Assert.assertEquals(actualText, expected);

}

@Test(priority='3') public void test\_click\_register\_link() throws InterruptedException

{

Thread.sleep(1500);

hp.click\_register\_link();

}

}

Next go to add another class LoginPage Test under same Package

Code: package com.sportyshoe.Tests;

import org.testng.Assert; import org.testng.annotations.BeforeTest; import org.testng.annotations.Test;

import com.sportyshoe.SeleniumCucumberScripts.HomePage; import com.sportyshoe.SeleniumCucumberScripts.LoginPage; import com.sportyshoe.SeleniumCucumberScripts.RegisterPage; import com.sportyshoe.SeleniumCucumberScripts.TestBase;

public class LoginPageTest extends TestBase {

HomePage hp;

RegisterPage rp;

LoginPage lp;

@BeforeTest

public void start\_browser()

{

OpenBrowser("Chrome"); hp = new HomePage(driver); rp = new RegisterPage(driver); lp = new LoginPage(driver);

}

@Test(priority='1') public void test\_login()

{

lp.user\_login();

}

@Test(priority='2') public void test\_getTitle\_page()

{

String expected = "http://localhost:9010/login";

String Actual = hp.getURL\_page();

Assert.assertEquals(Actual, expected);

}

@Test(priority='3')

public void Test\_validate\_registration\_Text()

{

String expected = "Hello rishikesh !";

String actualText = rp.validate\_registration\_Text();

Assert.assertEquals(actualText, expected);

}

@Test(priority='4')

public void test\_click\_cart()

{

lp.click\_cart();

}

}

Next go to another class add registerPagetests

Code: package com.sportyshoe.Tests;

import static org.testng.Assert.assertEquals;

import org.testng.Assert; import org.testng.annotations.BeforeTest; import org.testng.annotations.Test;

import com.sportyshoe.SeleniumCucumberScripts.HomePage; import com.sportyshoe.SeleniumCucumberScripts.RegisterPage; import com.sportyshoe.SeleniumCucumberScripts.TestBase;

public class RegisterPageTest extends TestBase {

HomePage hp;

RegisterPage rp;

@BeforeTest

public void start\_browser()

{

OpenBrowser("Chrome"); hp = new HomePage(driver); rp = new RegisterPage(driver); }

@Test(priority='1')

public void test\_click\_register\_link() throws InterruptedException

{

Thread.sleep(1500); hp.click\_register\_link();

}

@Test(priority='2') public void test\_getTitle\_page()

{

String expected = "http://localhost:9010/register";

String Actual = hp.getURL\_page();

Assert.assertEquals(Actual, expected);

}

@Test(priority='3') public void Test\_register\_user()

{

rp.register\_user();

}

@Test(priority='4')

public void Test\_validate\_registration\_URL()

{

String expected = "http://localhost:9010/register-user"; String Actual = rp.validate\_registration\_URL(); assertEquals(Actual, expected);

}

@Test(priority='5')

public void Test\_validate\_registration\_Text()

{

String expected = "Hello rishikesh !";

String actualText = rp.validate\_registration\_Text();

Assert.assertEquals(actualText, expected);

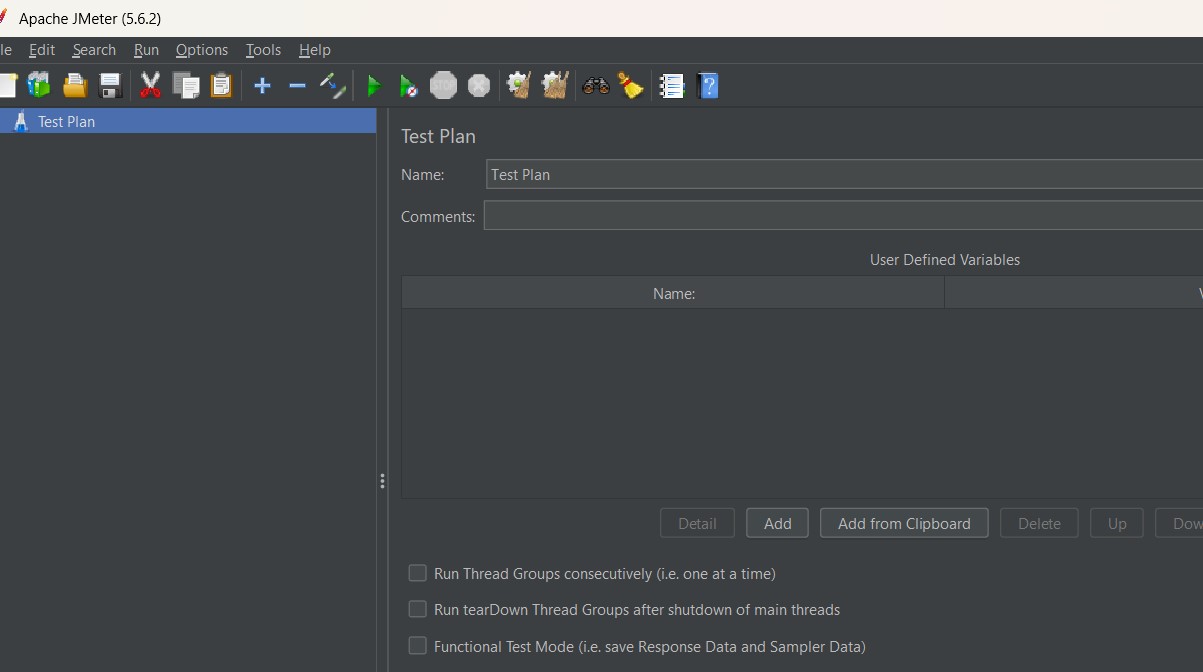
}

}

Jmeter:

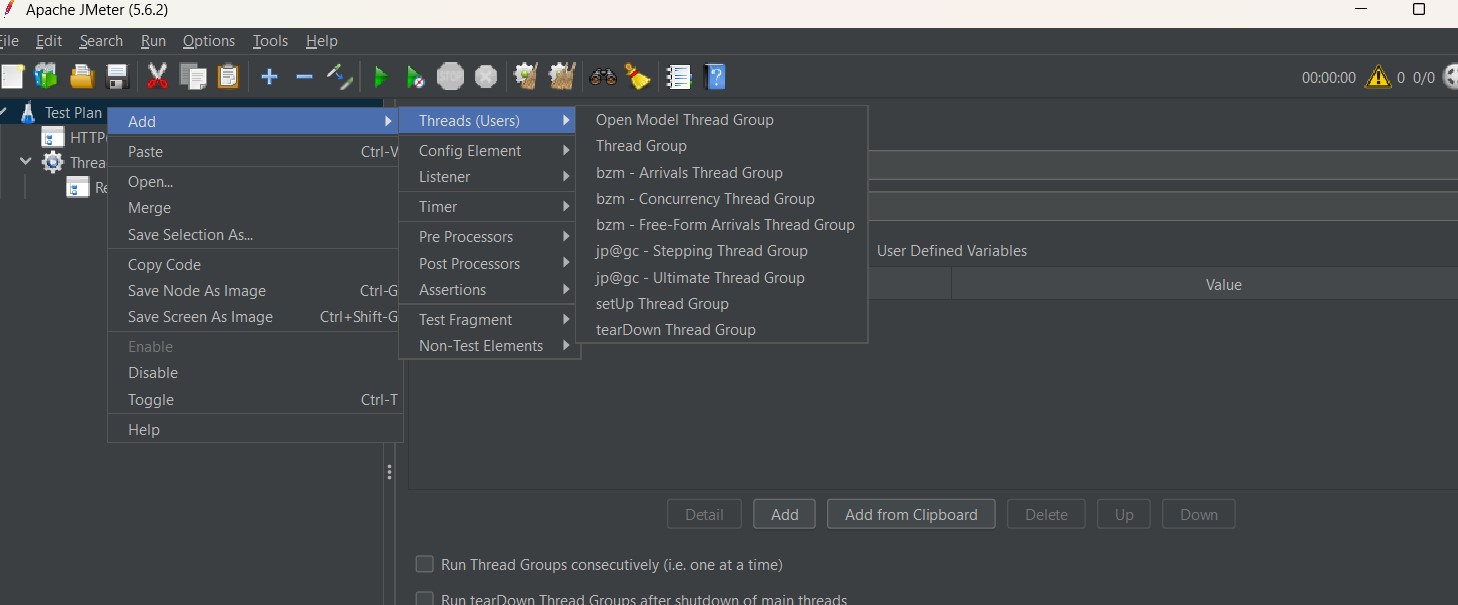
1. Create JMeter scripts to do load testing of the homepage and the product detail page.

First go to Apache Jmeter

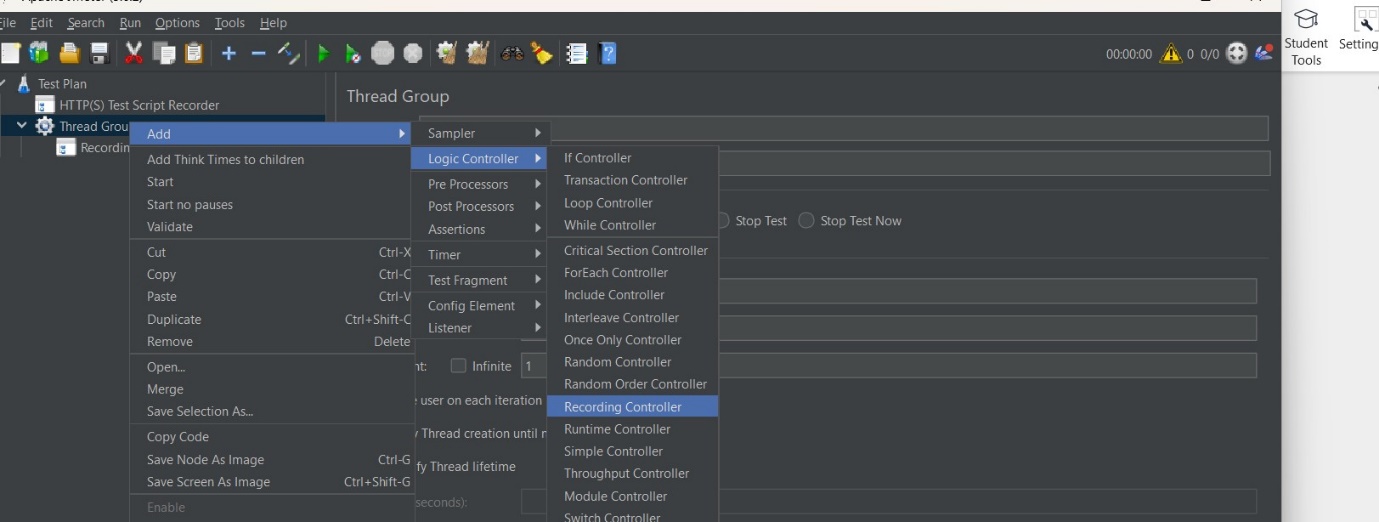


In this test plan add HTTP(s) Test Script Recorder

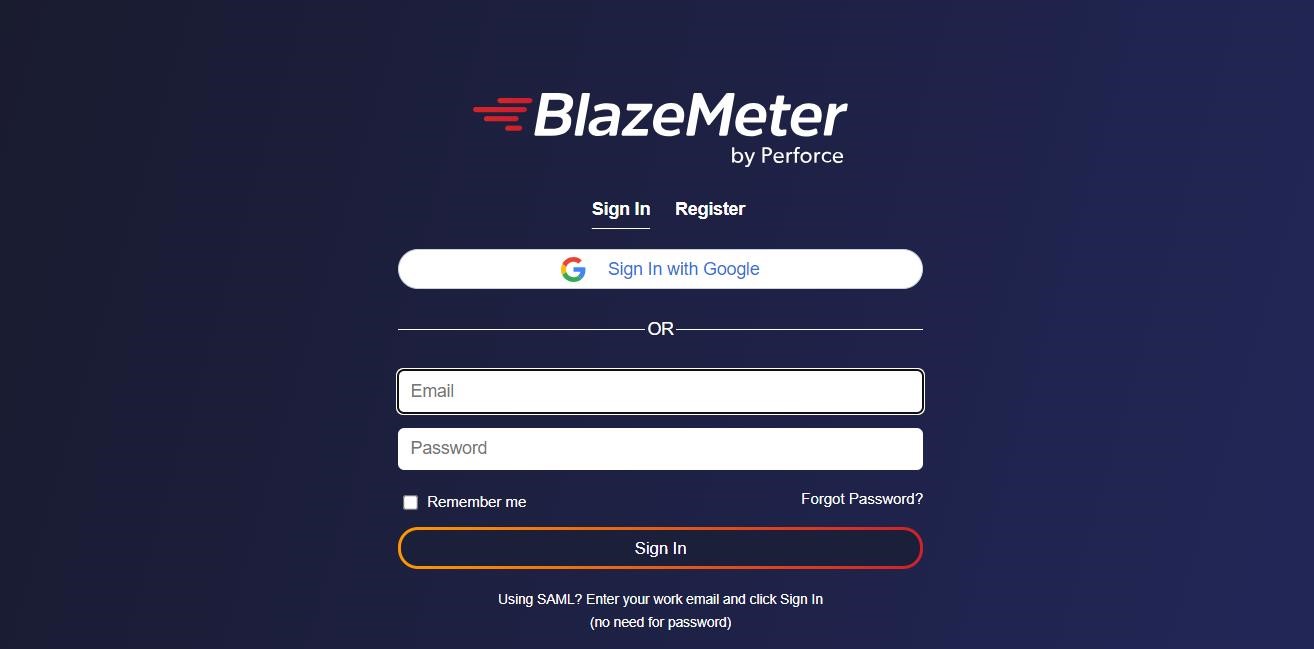
Next Add thread group

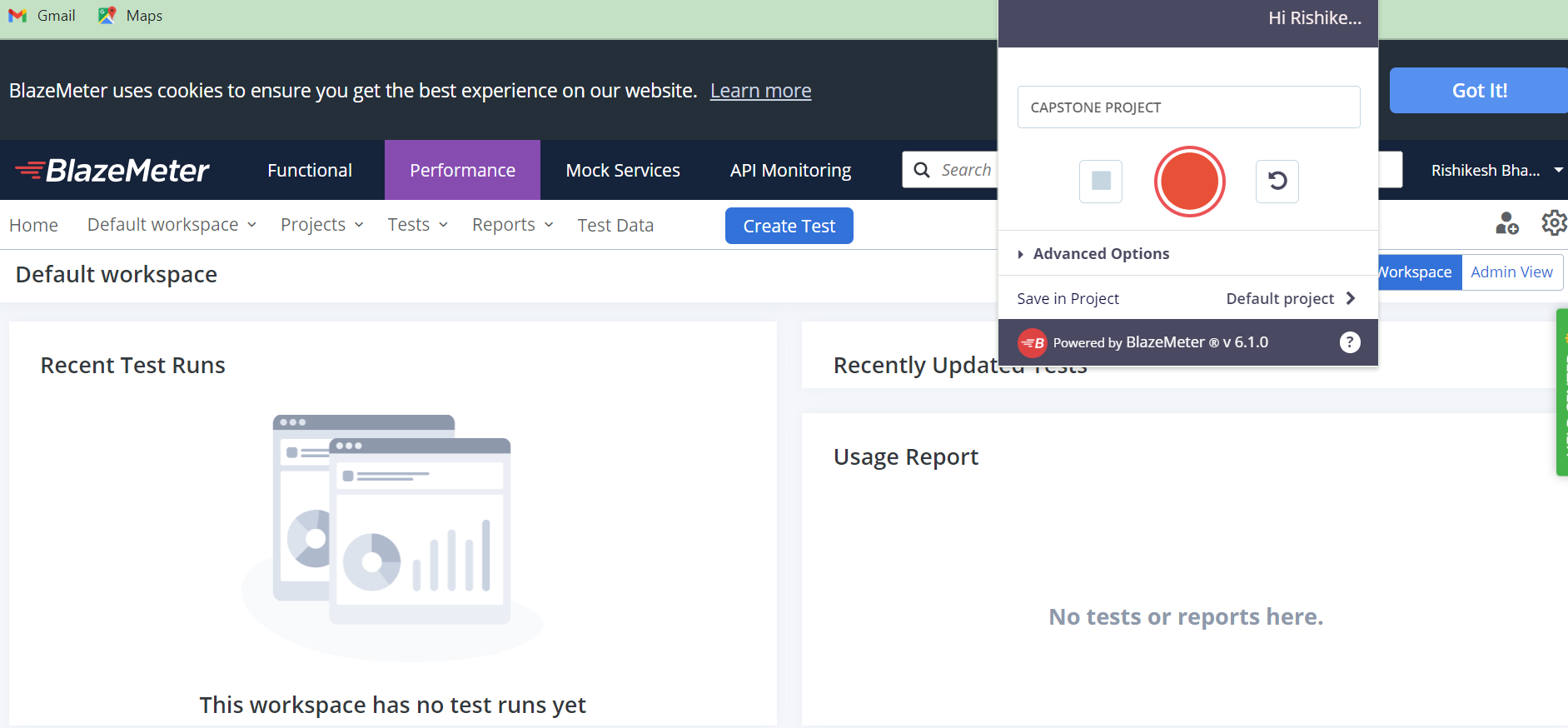


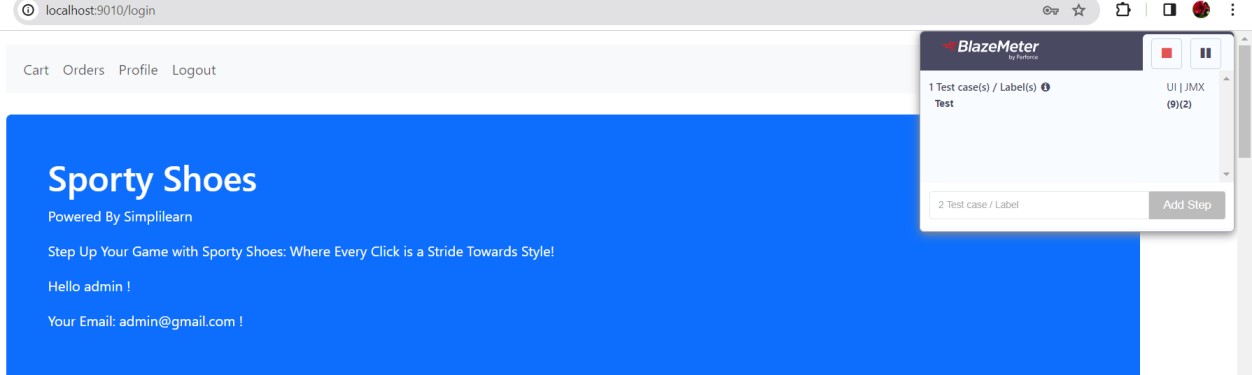
In thread group add controller >logic controller>Recording controller



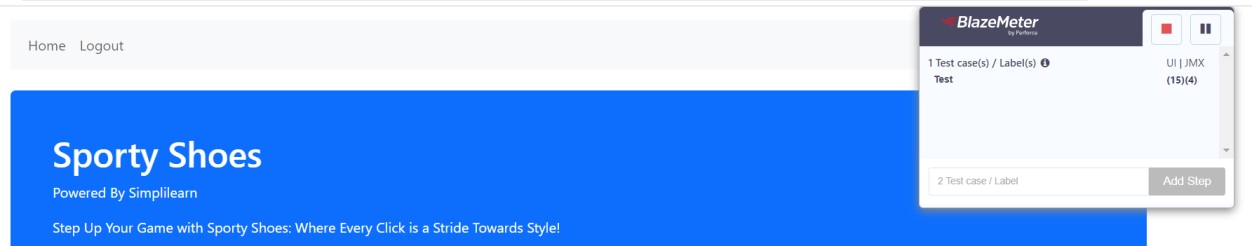
Next go to blazemeter for recording Scripts





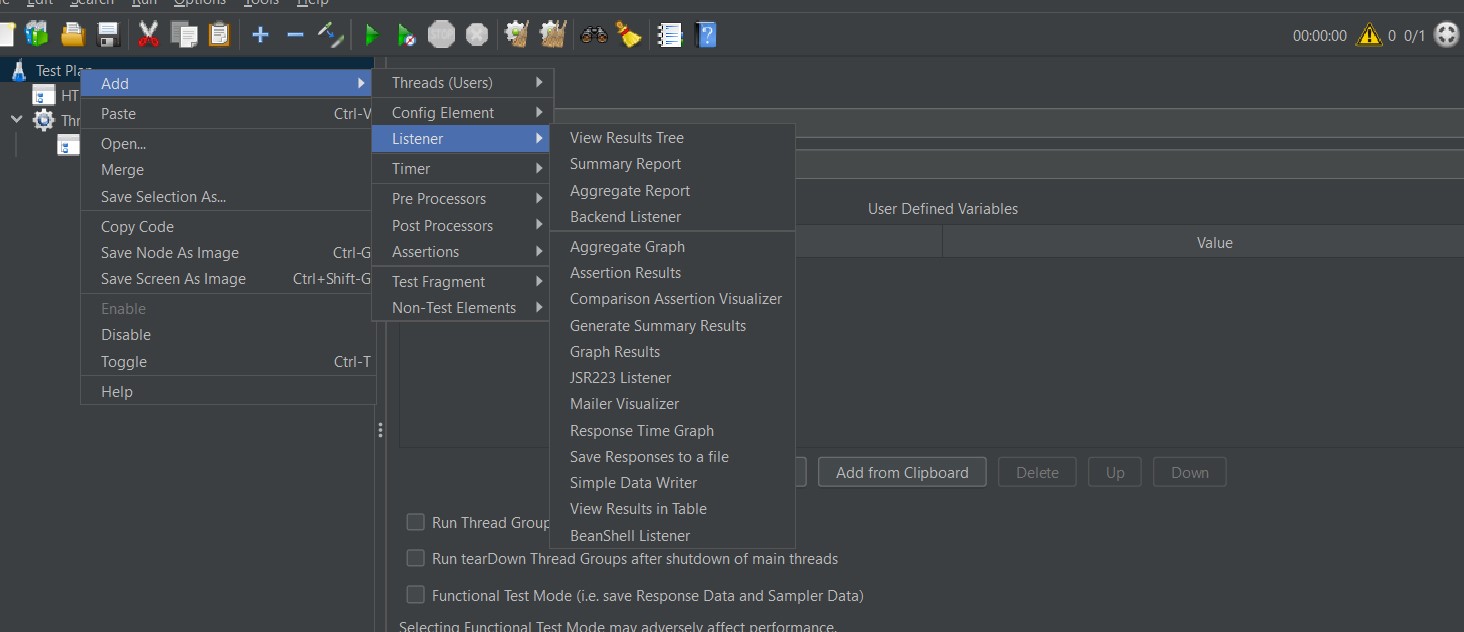


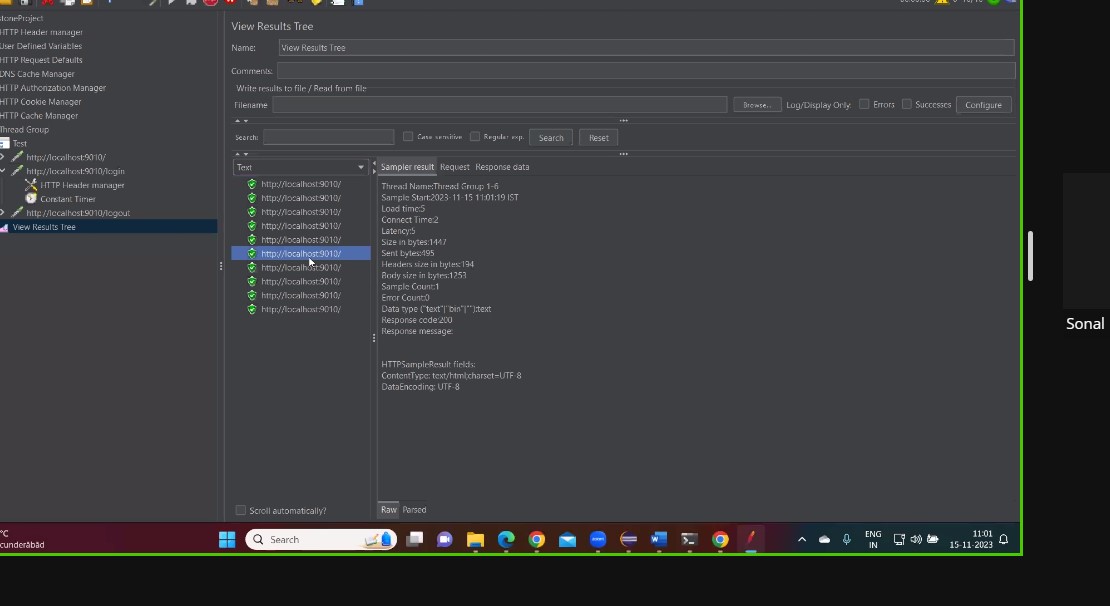
Logout Page



And save the recorder controller in jmx format

Add Listener to see results





Setup Cucumber in Java Project and write Feature Files using Gherkin to test the API endpoints mentioned in point 1 above.

First go to eclipse create a java project add feature file convert into step defintion file

Feature: first login to the spotyshoe. login page using local host:9010

Background:

Given I open the browser and enter URL

Then I capture the title of the Page

When I enter username and password

Then I click on login button

Scenario: User has to test if login on sportyshoe.login is successful or not

And I should see an Error message

Then I click on Click Here link

And I close the browser

Scenario:

Then I click on Create a sportyshoe. account

Then I capture the title of the Page

And I close the browser

Steps:

package com.sportyshoe.steps;

import org.openqa.selenium.WebDriver; import org.openqa.selenium.WebElement; import org.openqa.selenium.support.FindBy; import org.openqa.selenium.support.PageFactory;

public class LoginPage {

@FindBy(xpath="//input[@id='email']")

WebElement loginEmail;

@FindBy(xpath="//input[@id='password']")

WebElement loginpassword;

@FindBy(xpath="//button[@type='submit']")

WebElement loginbtn;

@FindBy(linkText="Cart")

WebElement clickCart;

public LoginPage(WebDriver driver) { PageFactory.initElements(driver, this);

}

public void user\_login()

{

loginEmail.sendKeys("rishikesh@gmail.com"); loginpassword.sendKeys("rishikesh@123");

loginbtn.click();

}

public void click\_cart()

{

clickCart.click();

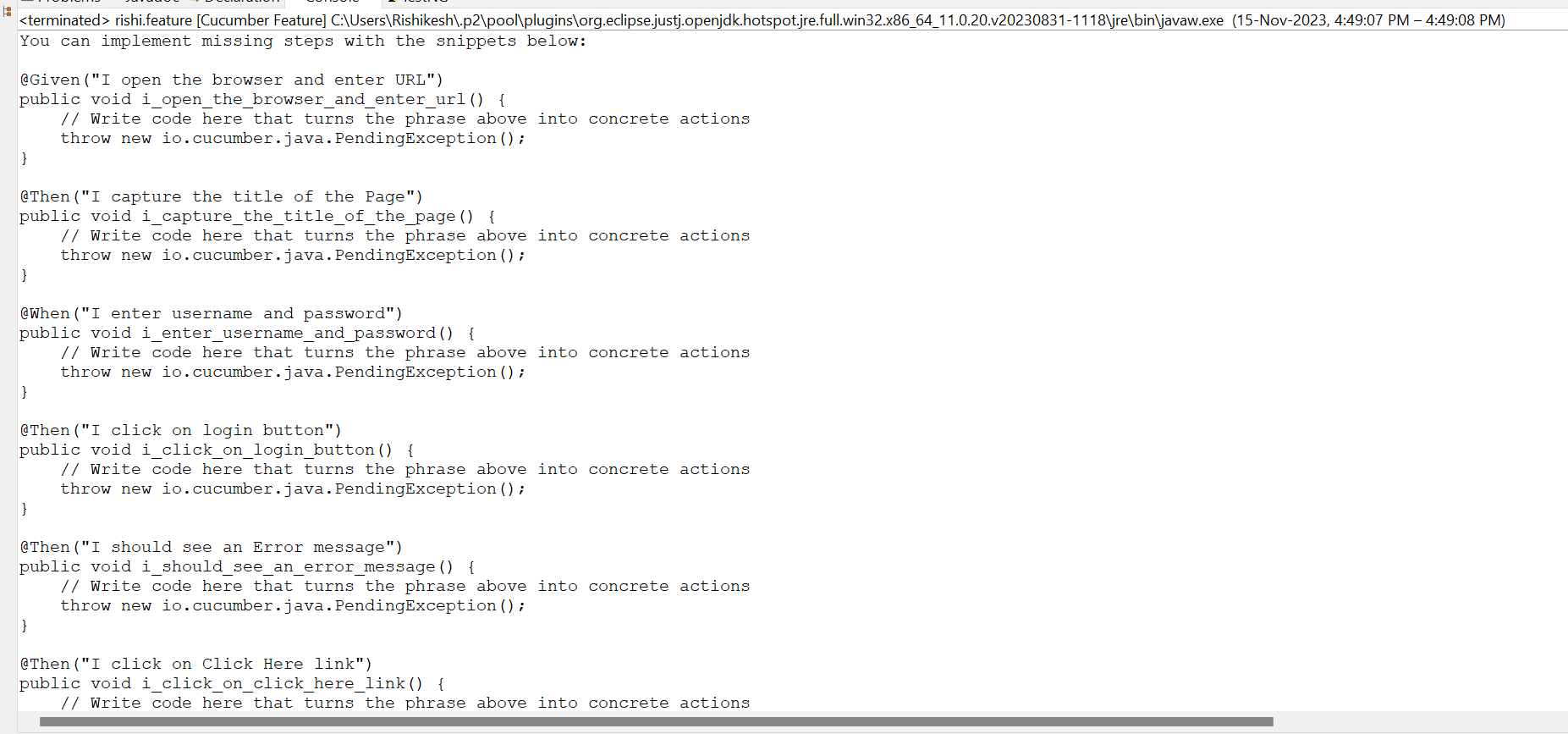
}

}

}

Output:





Gherkin feature file

