

## **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.

2. 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
3. Create JMeter scripts to do load testing of the homepage and the product detail page.
4. Setup Cucumber in Java Project and write Feature Files using Gherkin to test the API endpoints mentioned in point 1 above.
5. 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	<a href="http://localhost:9010/get-shoes">http://localhost:9010/get-shoes</a>
Retrieve the list of all registered users	GET	<a href="http://localhost:9010/get-users">http://localhost:9010/get-users</a>
Add the product	POST	<a href="http://localhost:9010/add-shoe?id=101&amp;image=image_url&amp;name=SampleShoe&amp;category=Running&amp;sizes=9&amp;price=1000">http://localhost:9010/add-shoe?id=101&amp;image=image_url&amp;name=SampleShoe&amp;category=Running&amp;sizes=9&amp;price=1000</a>
Delete the product	DELETE	<a href="http://localhost:9010/delete-shoe?id=101">http://localhost:9010/delete-shoe?id=101</a>
Update the product	PUT	<a href="http://localhost:9010/update-shoe">http://localhost:9010/update-shoe</a>

**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](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>)

```
{
  "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 capstone 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>

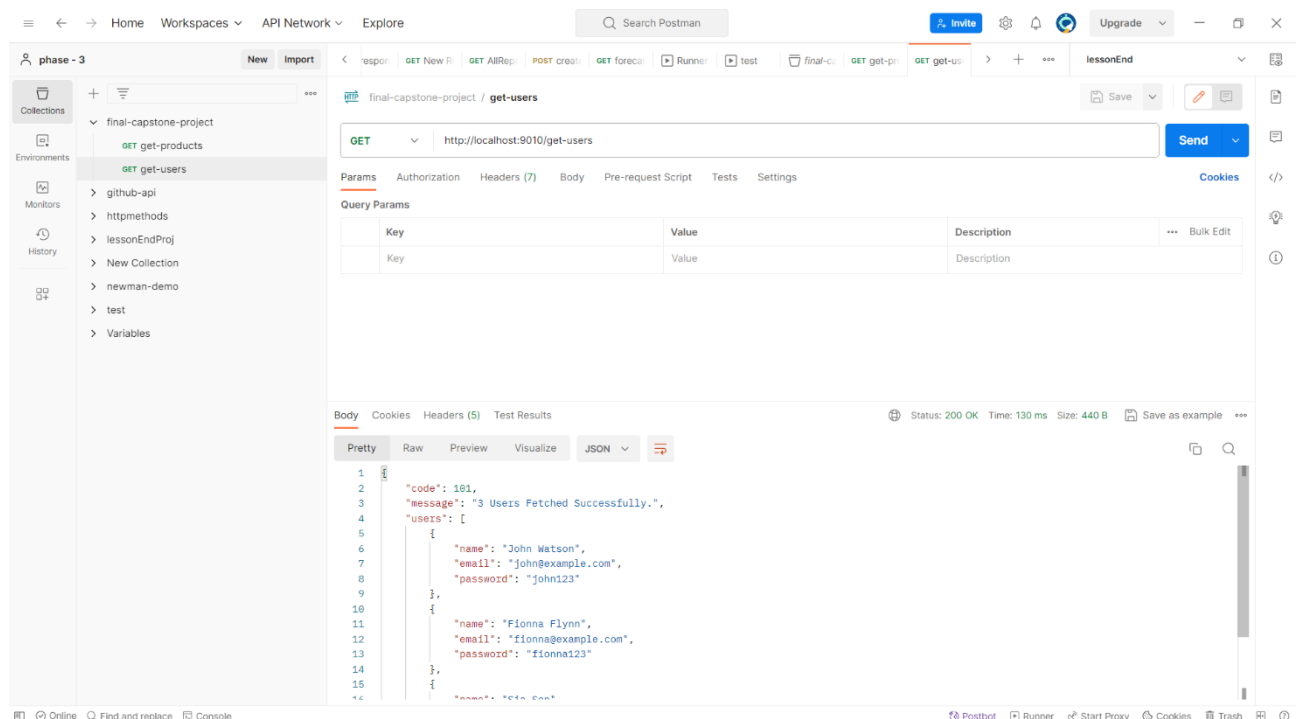
After adding URL click on save button and run this URL

This above picture is the GET all products output

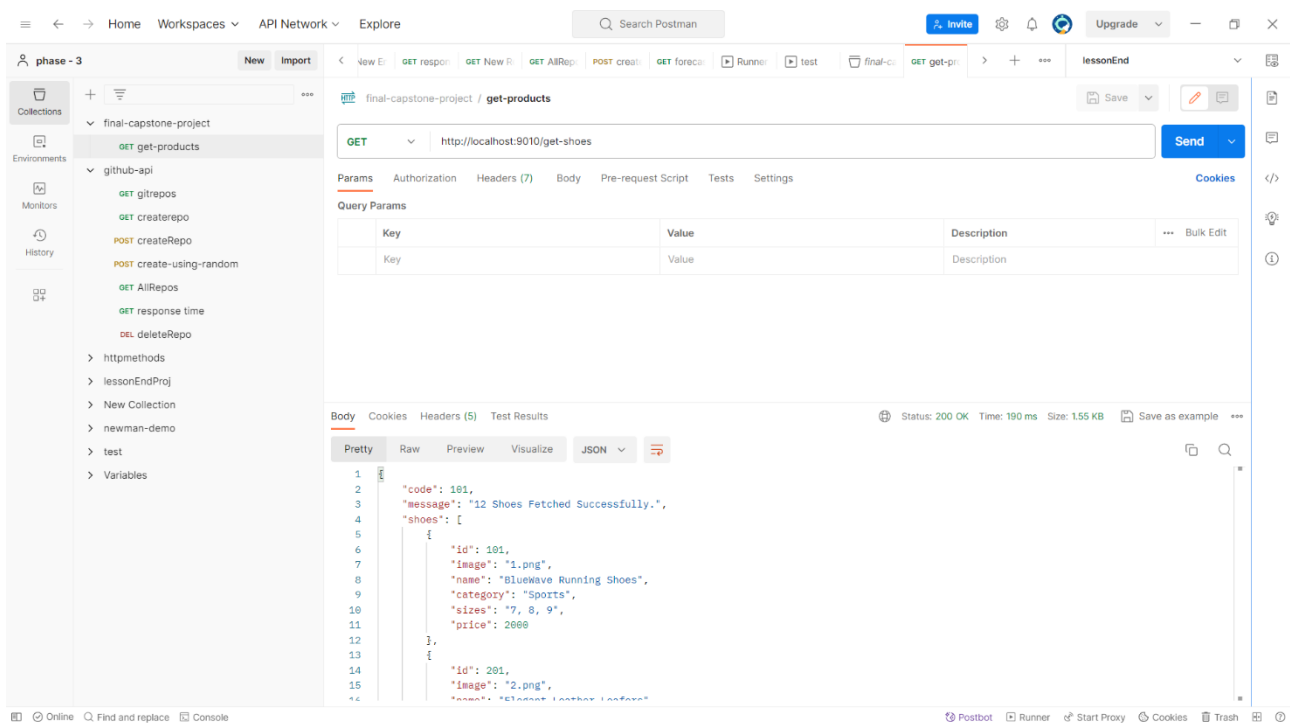
Next select another request put name as GET all users

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



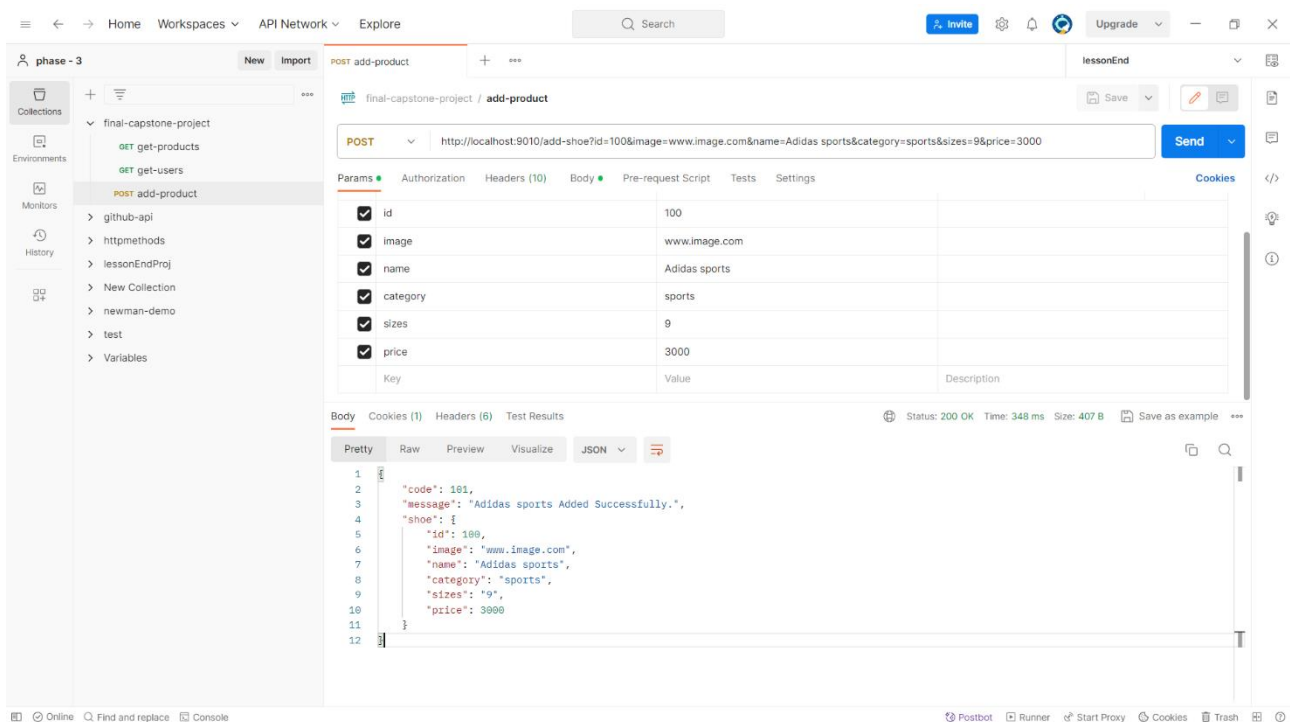
## Get products



The screenshot shows the Postman interface with a GET request to `http://localhost:9010/get-shoes`. The response is a JSON object with a status code of 200 OK, a message "12 Shoes Fetched Successfully.", and an array of shoe objects. Each shoe object contains `id`, `image`, `name`, `category`, `sizes`, and `price`.

```
1 {
2   "code": 101,
3   "message": "12 Shoes Fetched Successfully.",
4   "shoes": [
5     {
6       "id": 101,
7       "image": "1.png",
8       "name": "Bluewave Running Shoes",
9       "category": "Sports",
10      "sizes": "7, 8, 9",
11      "price": 2000
12    },
13    {
14      "id": 201,
15      "image": "2.png",
16      "name": "Elegant Leather Loafers"
17    }
18  ]
19 }
```

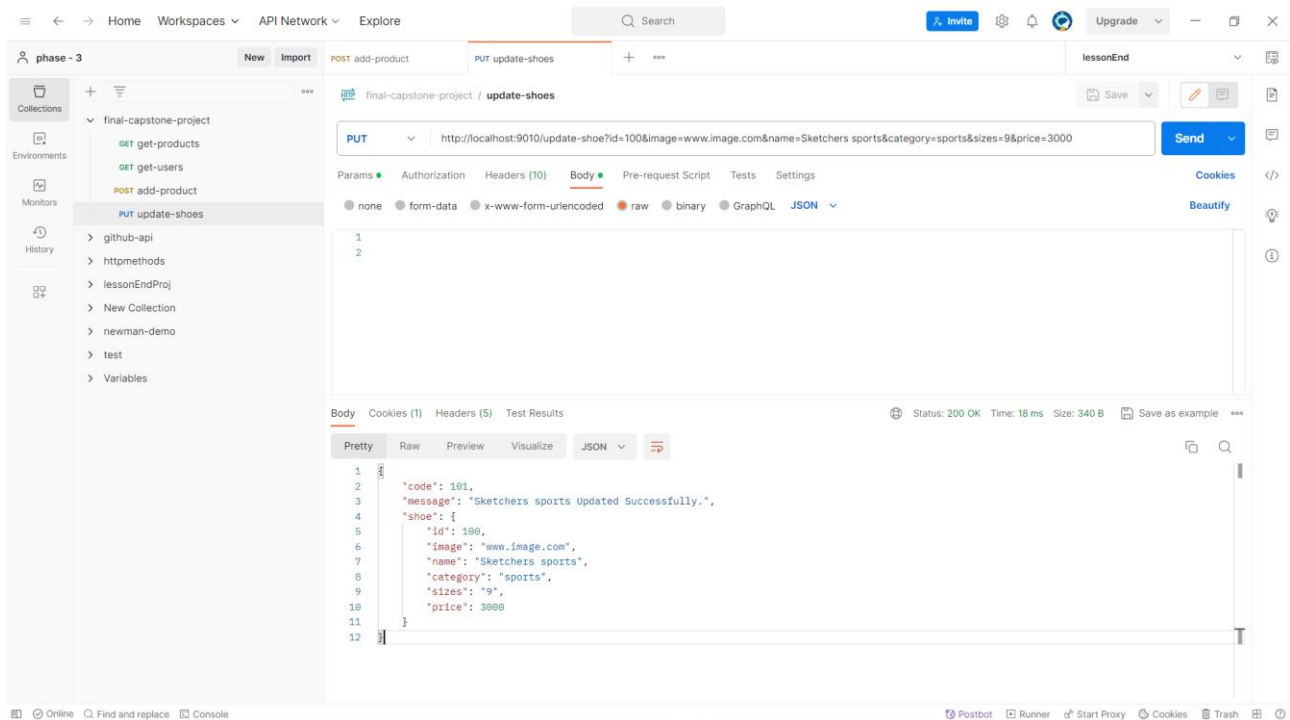
## Enter URL select method as POST method click on save and send the request



The screenshot shows the Postman interface with a POST request to `http://localhost:9010/add-shoe?id=100&image=www.image.com&name=Adidas sports&category=sports&sizes=9&price=3000`. The response is a JSON object with a status code of 200 OK, a message "Adidas sports Added Successfully.", and a shoe object. The shoe object contains `id`, `image`, `name`, `category`, `sizes`, and `price`.

```
1 {
2   "code": 101,
3   "message": "Adidas sports Added Successfully.",
4   "shoe": {
5     "id": 100,
6     "image": "www.image.com",
7     "name": "Adidas sports",
8     "category": "sports",
9     "sizes": "9",
10    "price": 3000
11  }
12 }
```

this is the output for POST add a Product.  
And then select another method as Put update a Product

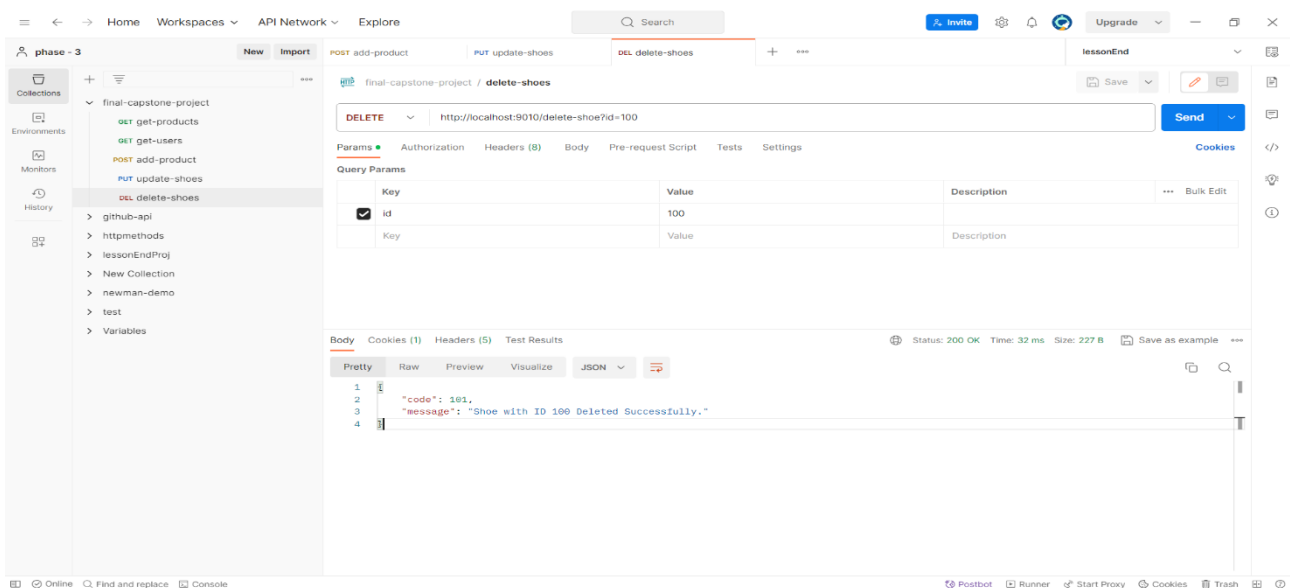


And insert the URL click on save and send Request

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=100>

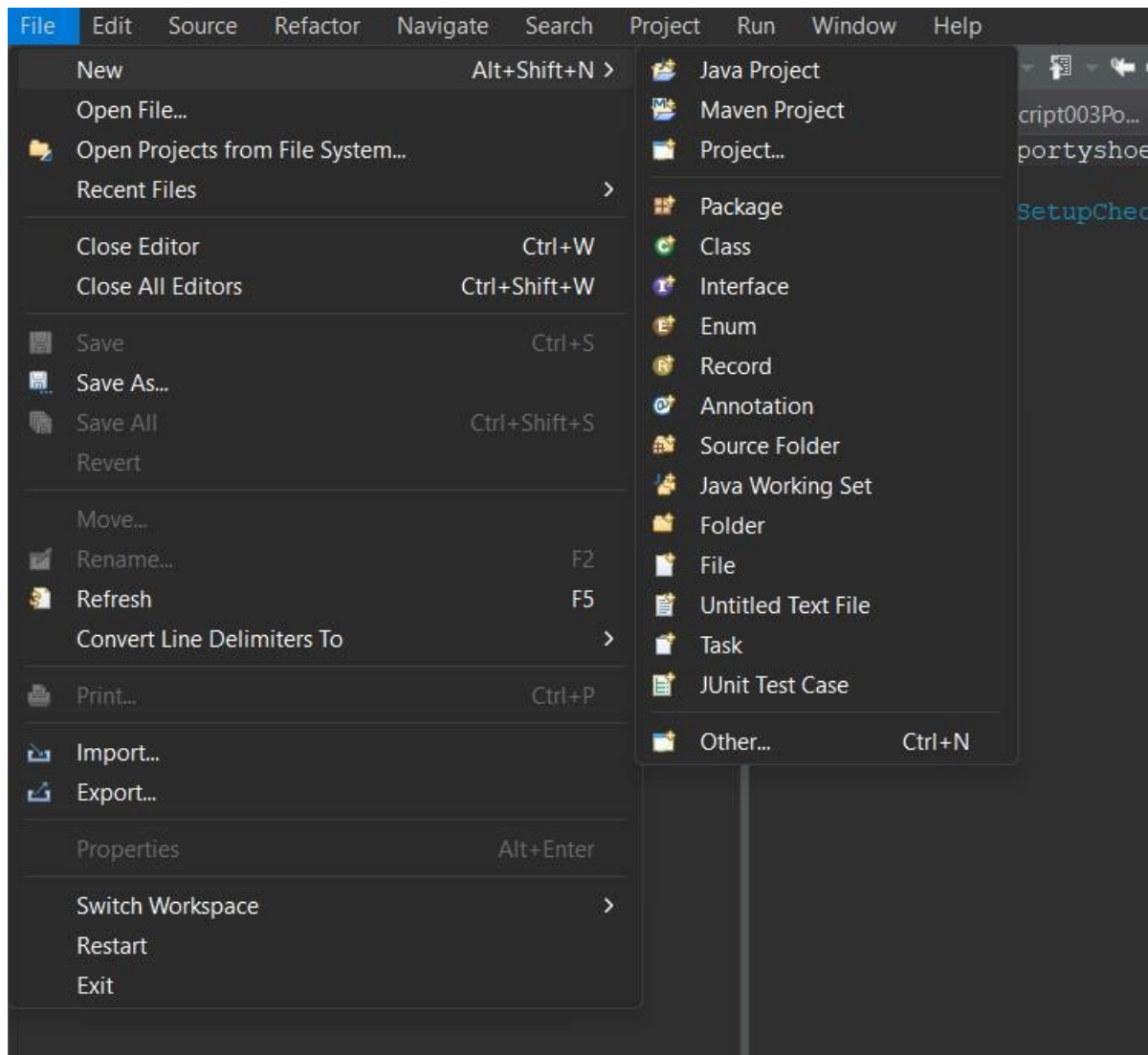


Click on save and run it

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()
            .baseUrl("http://localhost:9010")
            .basePath("/get-shoes")
            .when()
            .get()
            .then()
            .log().all()
            .body("shoes[2].id", equalTo(301),
                "shoes[2].name", equalTo("NeoFlex Athletic Shoes"),
                "shoes[2].category", equalTo("Sports"),
                "shoes[2].price", equalTo(4500))
            .log().all();

    }

    @Test(priority='2')
    public void get_all_users() {

        io.restassured.response.Response response = RestAssured
            .given()
            .baseUrl("http://localhost:9010")
            .basePath("/get-users")
            .when()
            .get()
            .then()
            .body("users[0].name", equalTo("John Watson"),
                "users[0].email", equalTo("john@example.com"),
                "users[0].password", equalTo("john123"))
            .extract()
            .response();

        System.out.println("The response is :"+response.getBody().asPrettyString());
        System.out.println("status code" + response.getStatusCode());
        System.out.println("response time" + response.getTime());
        int expectedstatus = 200;
        int actualstatus = response.getStatusCode();
        Assert.assertEquals(expectedstatus, actualstatus);

    }

}

```

**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()
    .baseUrl("http://localhost:9010")
    .basePath("/add-shoe")
    .queryParams("id","1020")
    .queryParams("image", "www.imge.com")
    .queryParams("name","Nike")
    .queryParams("category", "Running")
    .queryParams("sizes","5,6,7")
    .queryParams("price", "2000")
    .when()
    .post()
    .then()
    .log().all();
```

```
}
```

```
@Test(priority='2')    public void
```

```
update_a_product()
```

```
{
```

```
    RestAssured.given()
    .baseUrl("http://localhost:9010")
    .basePath("/update-shoe")
    .queryParams("id","1020")
    .queryParams("image", "www.imge123.com")
```

```
.queryParams("name","Reebok")  
.queryParams("category", "Running")  
.queryParams("sizes","5,6,7")  
.queryParams("price", "2500")  
.when()  
.put()  
.then()  
.log().all();
```

```
}
```

```
}
```

Output:

```
HTTP/1.1 200
Set-Cookie: JSESSIONID=CE704AC3310EF793CA0C7CA273393BE8; Path=/; HttpOnly
Content-Type: application/json
Transfer-Encoding: chunked
Date: Tue, 14 Nov 2023 06:40:58 GMT
Keep-Alive: timeout=60
Connection: keep-alive

{
  "code": 101,
  "message": "Nike Added Successfully.",
  "shoe": {
    "id": 1020,
    "image": "www.imge.com",
    "name": "Nike",
    "category": "Running",
    "sizes": "5,6,7",
    "price": 2000
  }
}

HTTP/1.1 200
Set-Cookie: JSESSIONID=F06983A7957E5AAE47EEC24A6270CE30; Path=/; HttpOnly
Content-Type: application/json
Transfer-Encoding: chunked
Date: Tue, 14 Nov 2023 06:40:58 GMT
Keep-Alive: timeout=60
Connection: keep-alive

{
  "code": 101,
  "message": "Reebok Updated Successfully.",
  "shoe": {
    "id": 1020,
    "image": "www.imge123.com",
    "name": "Reebok",
    "category": "Running",
    "sizes": "5,6,7",
    "price": 2500
  }
}
```

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()
    .baseUrl("http://localhost:9010")
```



```

        .basePath("/delete-shoe")
        .queryParams("id", "1010")
        .when().delete()
        .then().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 12:03:39 GMT
Keep-Alive: timeout=60
Connection: keep-alive

{
  "code": 101,
  "message": "Shoe with ID 1010 Deleted Successfully."
}
PASSED: com.sportyshoe.restAssuredScripts.Deleteshoe.delete_product

=====

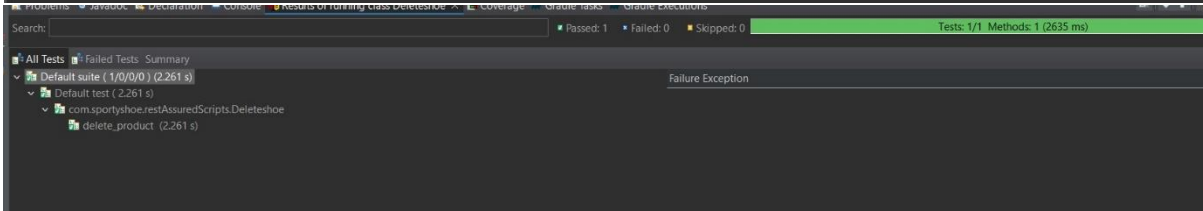
Default test
Tests run: 1, Failures: 0, Skips: 0
=====

=====

Default suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
=====

=====

```



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:

```

Powered By Simplilearn
Powered By Simplilearn

PASSED: com.sportyshoe.SeleniumCucumberScript.SetupCheck.test_setup_selenium

=====
    Default test
    Tests run: 1, Failures: 0, Skips: 0
=====

=====
Default suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
=====

PASSED: com.sportyshoe.SeleniumCucumberScript.SetupCheck.test_setup_selenium

=====
    Default test
    Tests run: 1, Failures: 0, Skips: 0
=====

=====
Default suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
=====

```

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:**

# Sporty Shoes

Powered By Simplilearn

Step Up Your Game with Sporty Shoes: Where Every Click is a Stride Towards Style!

## Login Yourself

Email:

Password:

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]")
```

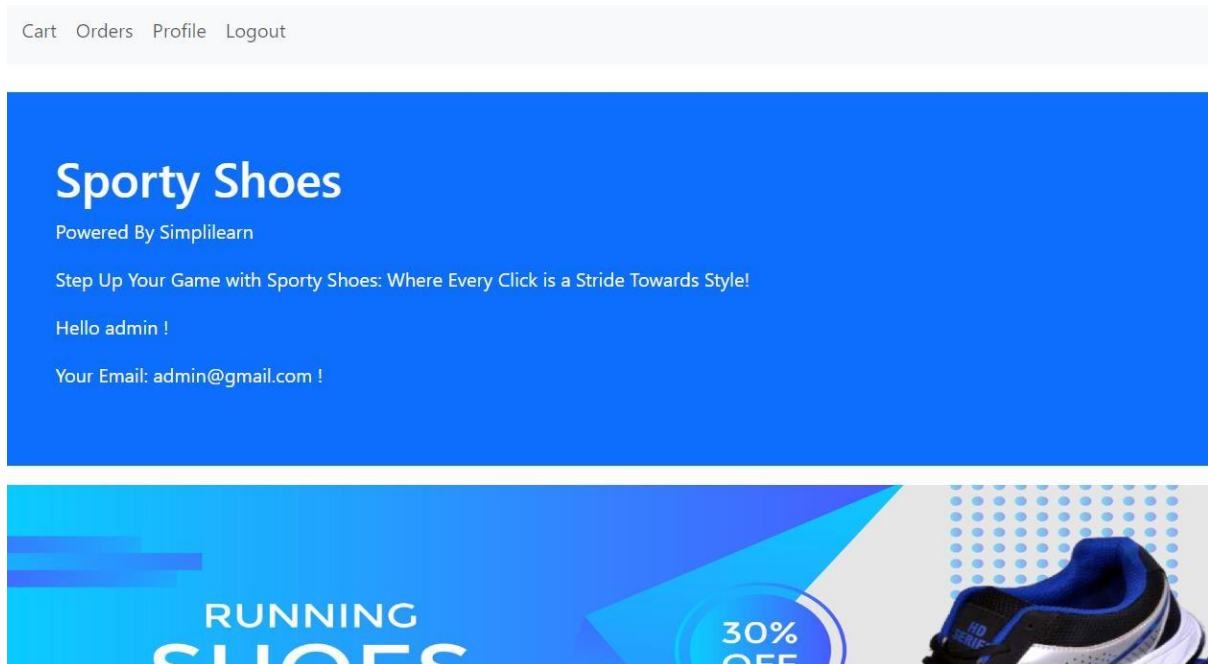
```
    WebElement text;
```

```
    @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("arpita@123");  
    loginbtn.click();  
}
```

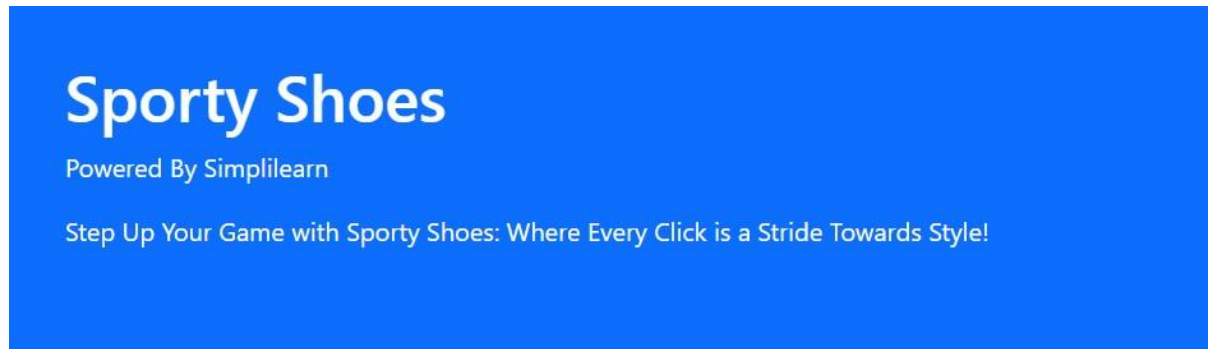
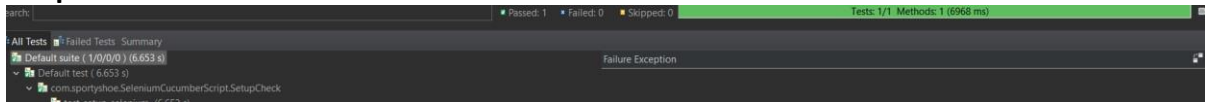
```
public void click_cart()  
{  
    clickCart.click();  
}
```

```
}
```

```
}
```



## Output:



## Login Yourself

mail:

admin@gmail.com

password:

\*\*\*\*\*

Login

[New User? Register Here](#)

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);
```

```
}
```

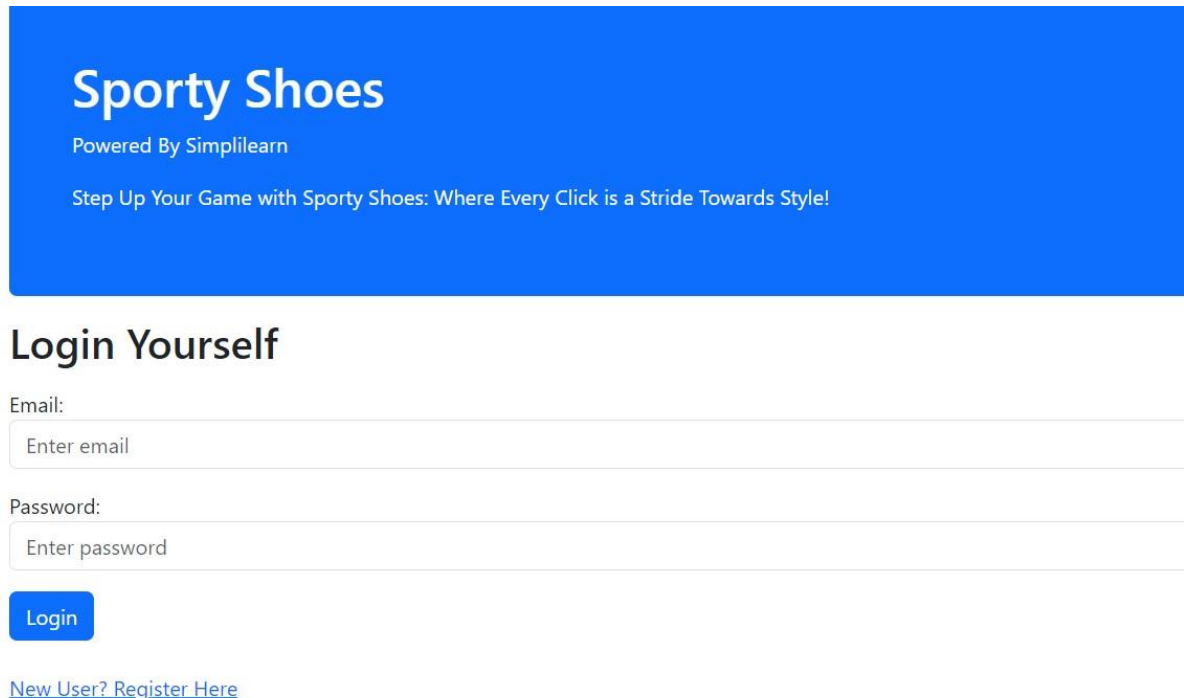
```
public void register_user()
{
    registername.sendKeys("rishikesh");
    registeremail.sendKeys("arpita@gmail.com");
    registerpassword.sendKeys("arpita@123");
    registerBtn.click();
}
```

```
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:



**Sporty Shoes**  
Powered By Simplilearn  
Step Up Your Game with Sporty Shoes: Where Every Click is a Stride Towards Style!

## Login Yourself

Email:

Password:

[Login](#)

[New User? Register Here](#)

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;
```

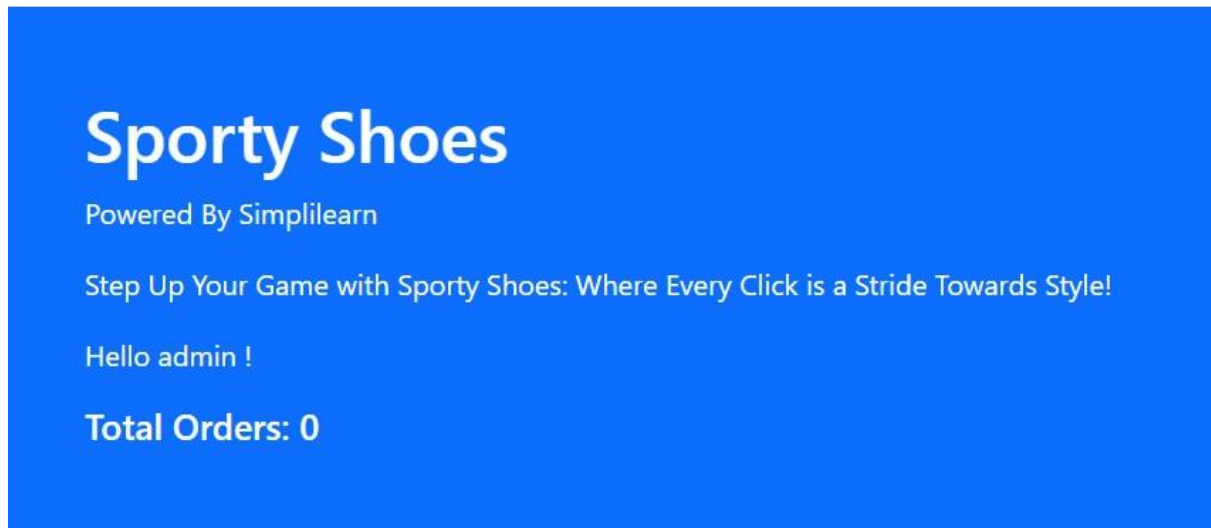
```
public class OrderPage {  
  
    @FindBy(linkText="Orders")  
    WebElement orderlink;  
  
    public OrderPage(WebDriver driver) {  
        PageFactory.initElements(driver, this);  
    }  
  
    public void click_orderPage()
```

```

    {
        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);
```

```
}
```

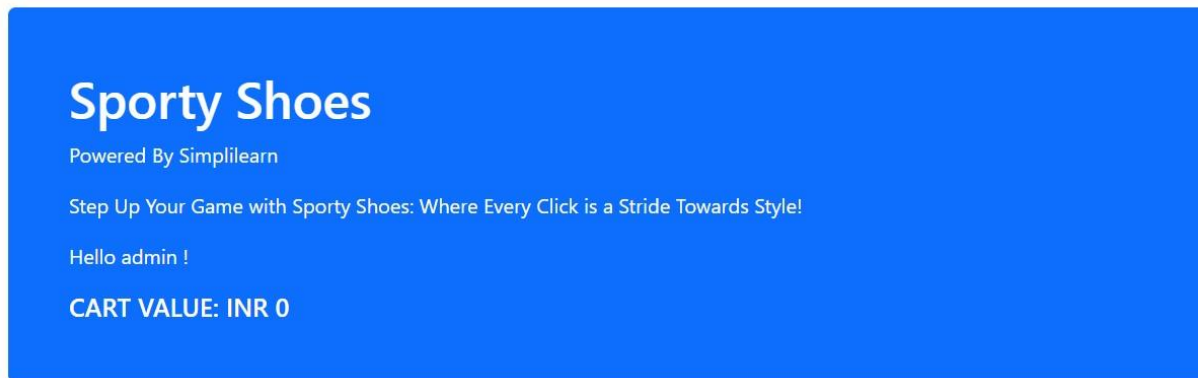
```
public String validate_success_message()
{
```

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

```
}
```

```
}
```

## Output:



TOTAL: INR 0

Place Order

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 Arpita!";  
        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
```

```
    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 Arpita !";
        String actualText = rp.validate_registration_Text();
        Assert.assertEquals(actualText, expected);
    }

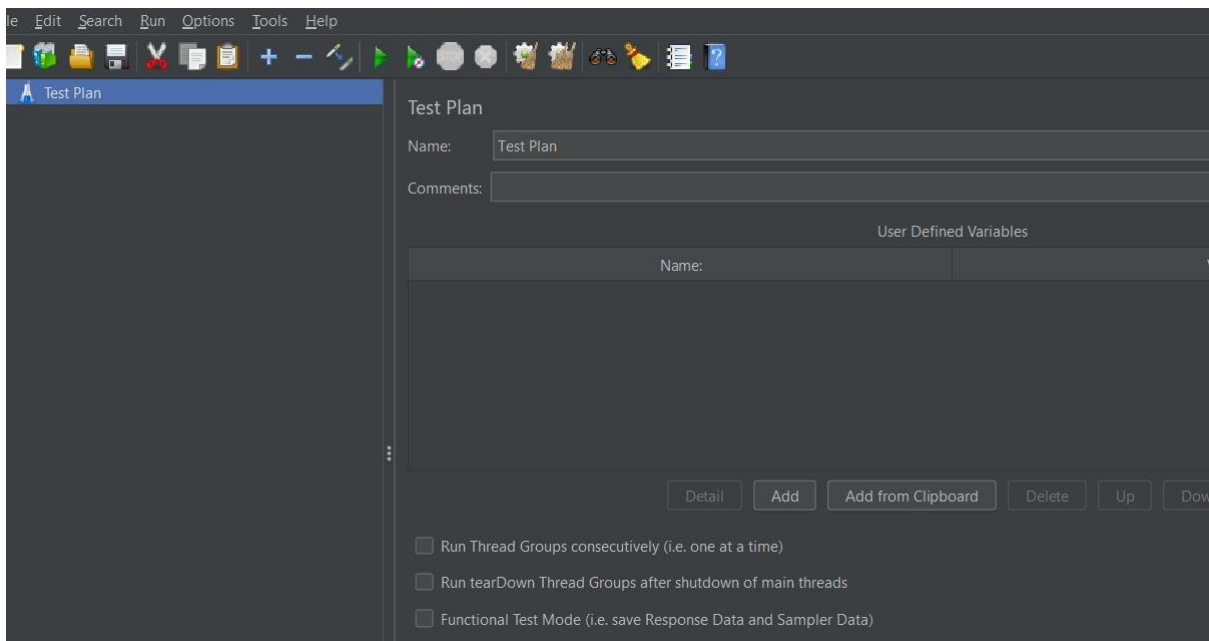
}

```

Jmeter:

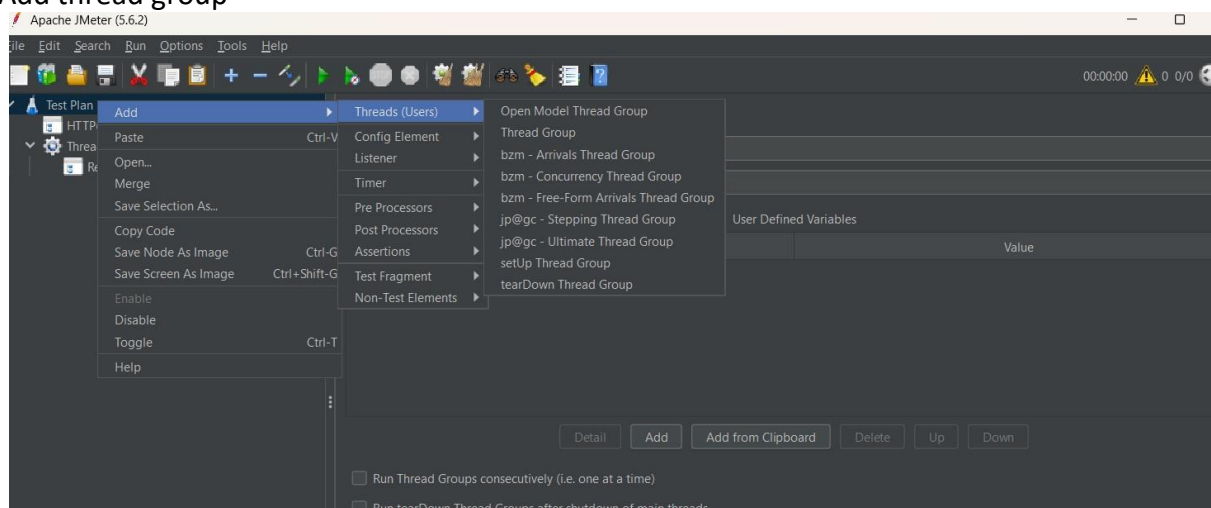
4. Create JMeter scripts to do load testing of the homepage and the product detail page.

First go to Apache Jmeter

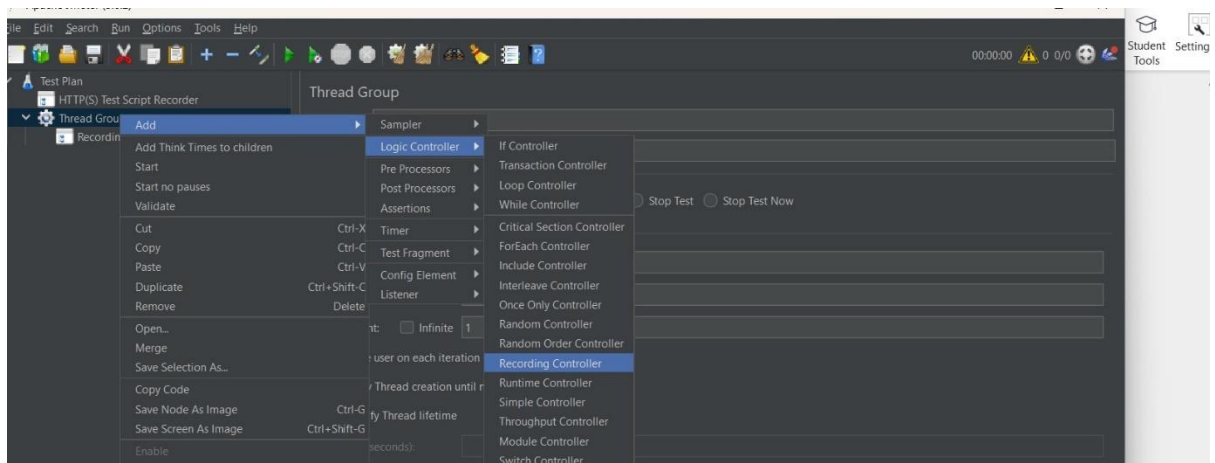


In this test plan add HTTP(s) Test Script Recorder

Next Add thread group



In thread group add controller >logic controller>Recording controller



Next go to blazemeter for recording Scripts

localhost:9010/register

← → ↻ 🏠 🔍

localhost:9010/register

Inbox (6,933) - arpita Jam Coursera Google Sheets chea... The Python Tutorial... ielts-academic | | simplilearn aws assignments javascript

Hi Arpita

CAPSTONE

🔴 🔵 🔁

Advanced Options

Save in Project Default project >

Powered by BlazeMeter v 6.1.0

## Sporty Shoes

Powered By Simplilearn

Step Up Your Game with Sporty Shoes: Where Every Click is a Stride Towards Style!

Its Wed Nov 15 16:07:16 IST 2023 !

### Register Yourself

Name:

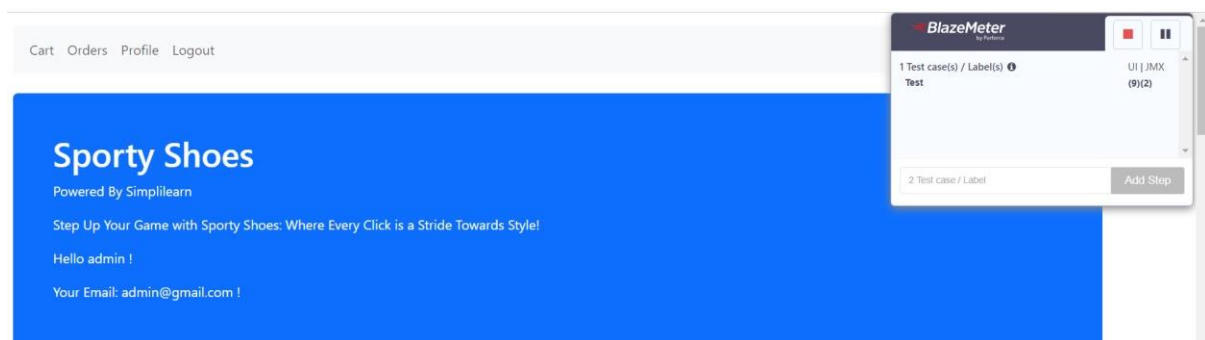
Email:

Password:

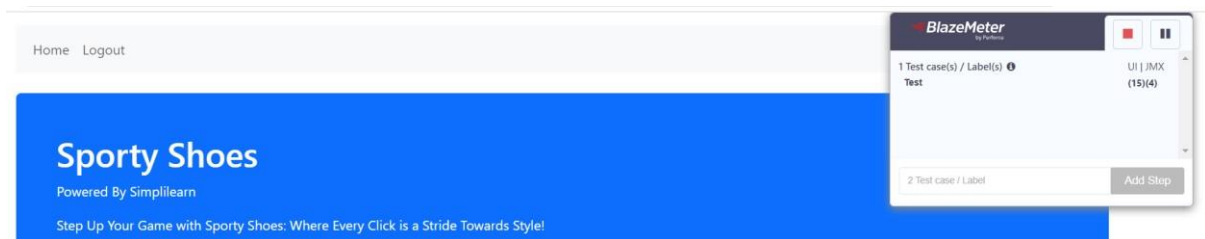
[Register](#)

[Existing User? Login Here](#)

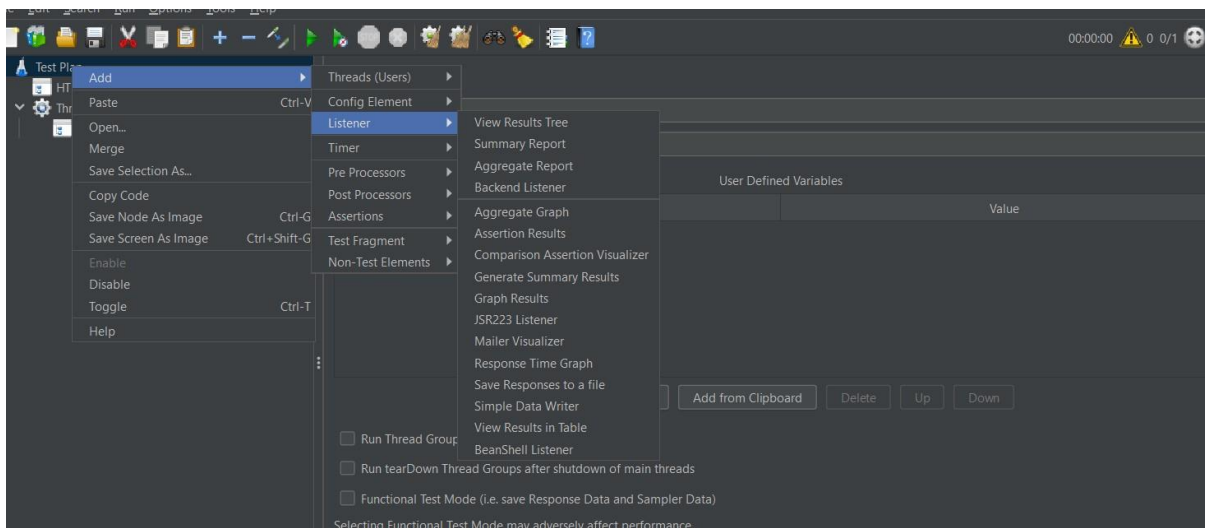


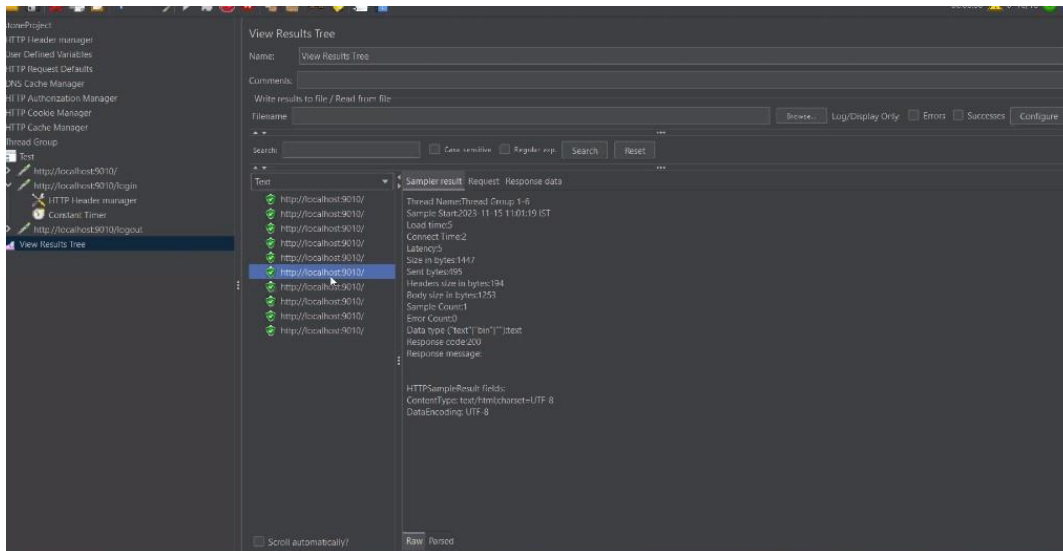


## Logout Page



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 definition file

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("arpita@gmail.com");  
loginpassword.sendKeys("arpita@123");  
        loginbtn.click();  
    }  
  
    public void click_cart()  
    {  
        clickCart.click();  
    }  
}  
  
}
```

```
<terminated> rishi.feature (Cucumber feature) C:\Users\rishikesh\p2\p00n\plugins\org.eclipse.jdt.launcher\org.eclipse.jdt.launcher.win32_x86_64_11.0.0.v20230611-1110\org.eclipse.jdt.launcher.exe (15-NOV-2023, 4:49:07 PM - 4:49:08 PM)
Nov 15, 2023 4:49:07 PM cucumber.api.cli.Main run
WARNING: You are using deprecated Main class. Please use io.cucumber.core.cli.Main

Scenario: User has to test if login on sportyshoe.login is successful or not # src/test/java/feature/rishi.feature:20
  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
  And I should see an Error message
  Then I click on Click Here link
  And I close the browser

Scenario: # src/test/java/feature/rishi.feature:24
  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
  Then I click on Create a sportyshoe. account
  Then I capture the title of the Page
  And I close the browser

Undefined scenarios:
file:///C:/Users/Rishikesh/Downloads/Phase3-PhaseEndProject-main/CapstoneProject2/src/test/java/feature/rishi.feature:20 # User has to test if login on sportyshoe
file:///C:/Users/Rishikesh/Downloads/Phase3-PhaseEndProject-main/CapstoneProject2/src/test/java/feature/rishi.feature:24 #

2 Scenarios (2 undefined)
14 Steps (12 skipped, 2 undefined)
0m0.334s

You can implement missing steps with the snippets below:

@Given("I open the browser and enter URL")
public void i_open_the_browser_and_enter_url() {
    // Write code here that takes the browser chrome data, opens the browser
}
```

[Cart](#) [Orders](#) [Profile](#) [Logout](#)

# Sporty Shoes

Powered By Simplilearn

Step Up Your Game with Sporty Shoes: Where Every Click is a Stride Towards Style!

Hello admin !

Your Email: admin@gmail.com !

