HARSHITHA NA

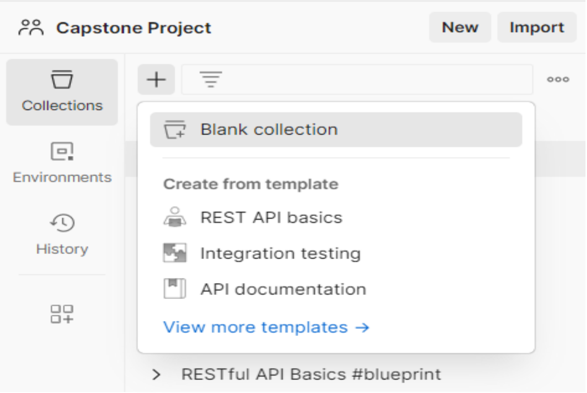
Capstone Project: Create a Testing Framework for Sporty Shoes Website

Source code for capstone Project:
Creating a Testing Framework for sporty shoes website:

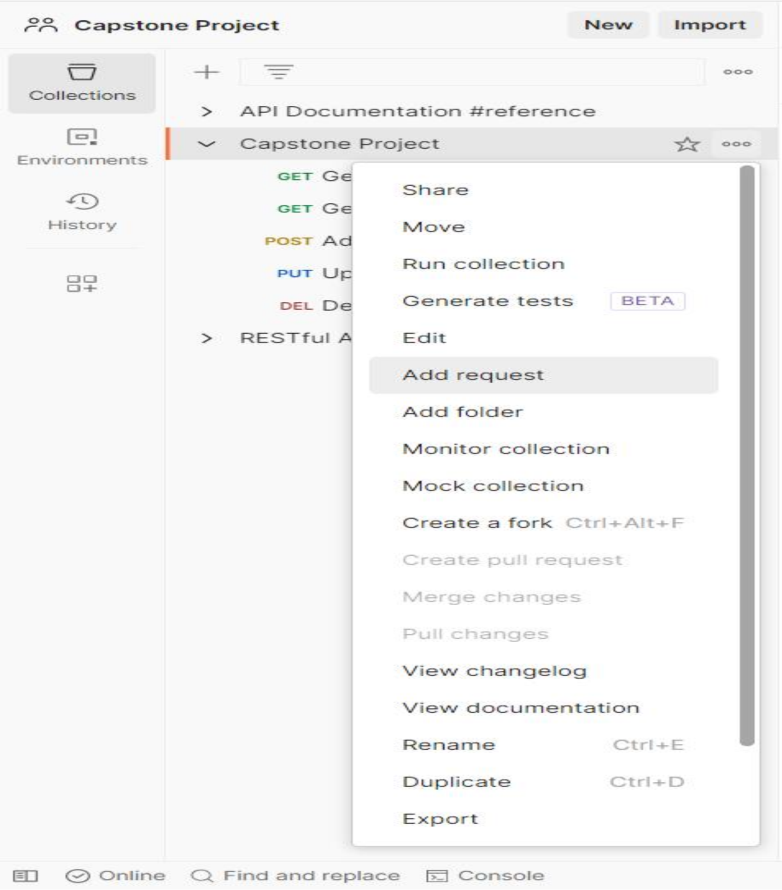
1. Create Postman scripts to test the following API endpoints:

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

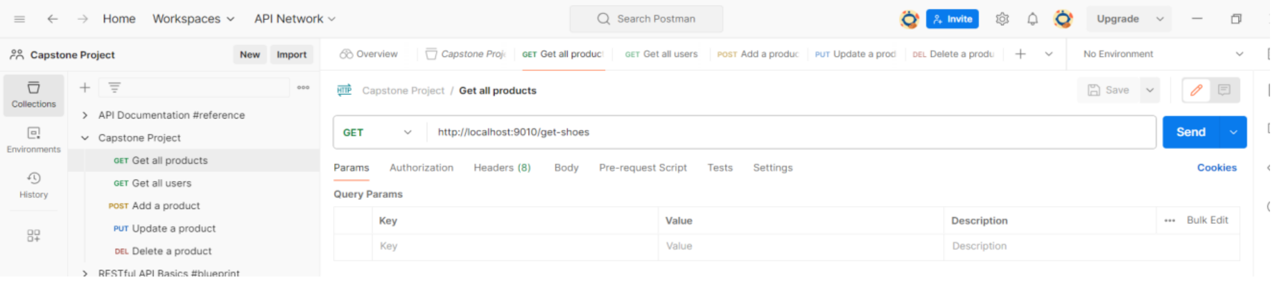
First go to postman application



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.

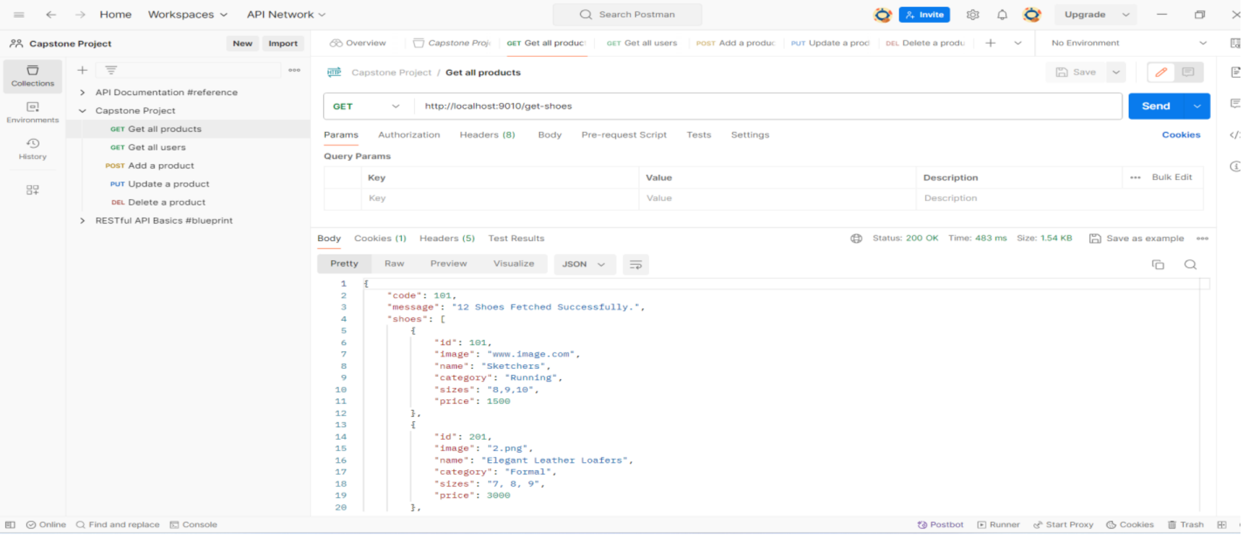


Capstone Project / Get all products

GET http://localhost:9010/get-shoes

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

After adding URL click on save button and run this URL



Capstone Project / Get all products

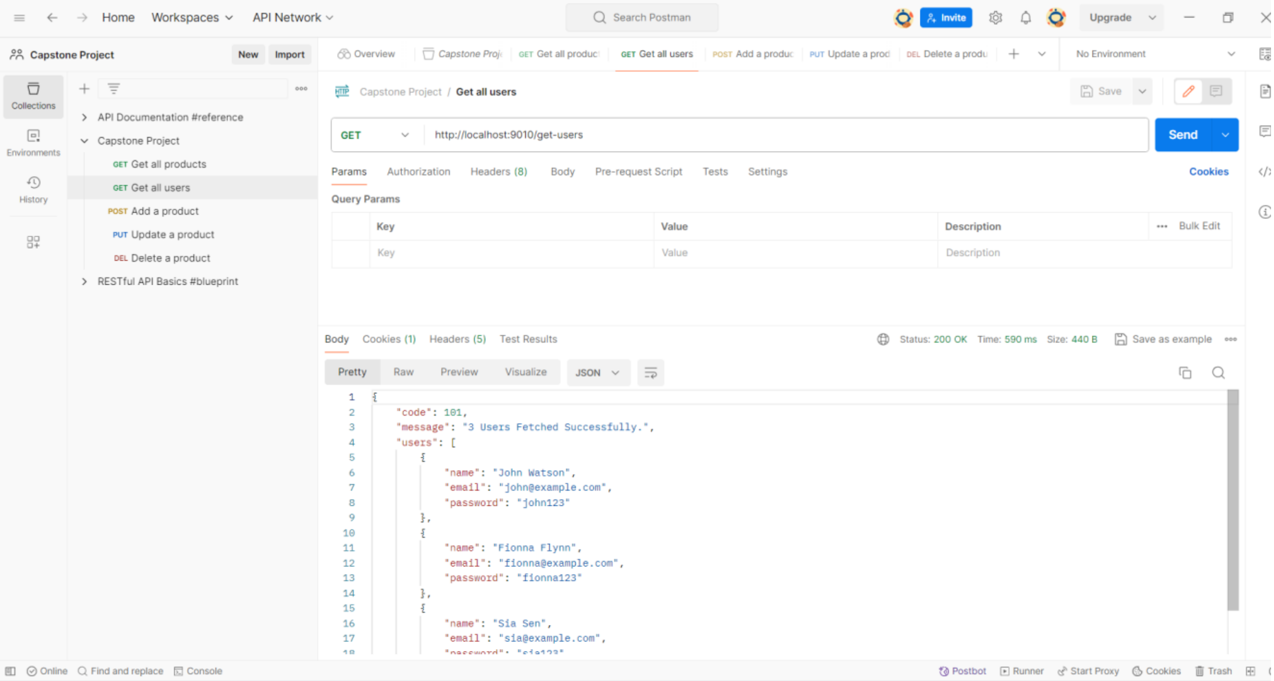
GET http://localhost:9010/get-shoes

Status: 200 OK Time: 483 ms Size: 1.54 KB

```
{
    "code": 101,
    "message": "12 Shoes Fetched Successfully.",
    "shoes": [
        {
            "id": 101,
            "image": "www.image.com",
            "name": "Sketchers",
            "category": "Running",
            "sizes": "8,9,10",
            "price": 1500
        },
        {
            "id": 201,
            "image": "2.png",
            "name": "Elegant Leather Loafers",
            "category": "Formal",
            "sizes": "7, 8, 9",
            "price": 3000
        },
```

This above picture is the GET all products output

Next select another request put name as GET all users



Capstone Project / Get all users

GET http://localhost:9010/get-users

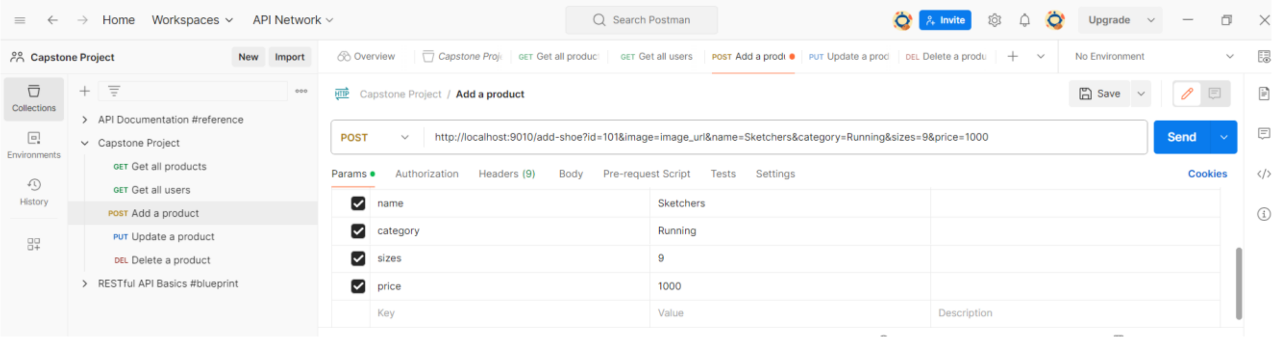
Status: 200 OK Time: 590 ms Size: 440 B

```
{
    "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"
```

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



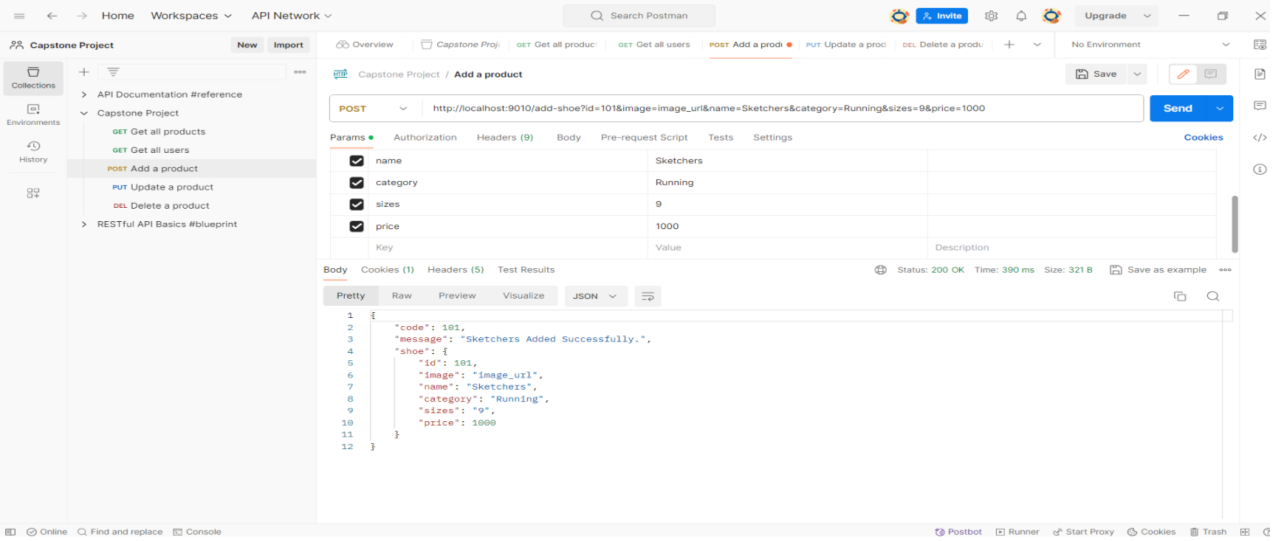
Capstone Project / Add a product

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

Params ● Authorization Headers (9) Body Pre-request Script Tests Settings

	Key	Value	Description
☑	name	Sketchers	
☑	category	Running	
☑	sizes	9	
☑	price	1000	
	Key	Value	Description

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



Capstone Project / Add a product

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

Params ● Authorization Headers (9) Body Pre-request Script Tests Settings

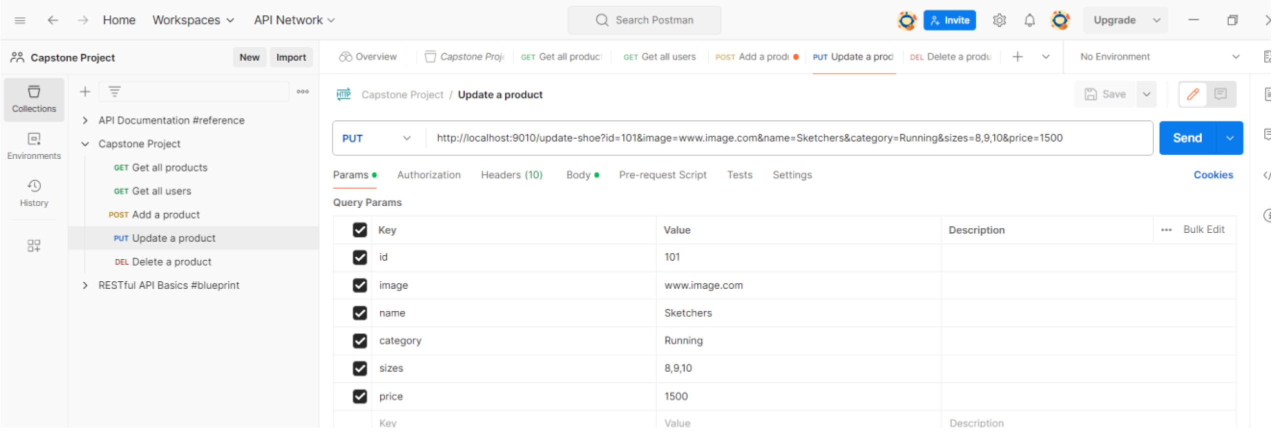
	Key	Value	Description
☑	name	Sketchers	
☑	category	Running	
☑	sizes	9	
☑	price	1000	
	Key	Value	Description

Body Cookies (1) Headers (5) Test Results — Status: 200 OK Time: 390 ms Size: 321 B Save as example

```
{
    "code": 101,
    "message": "Sketchers Added Successfully.",
    "shoe": {
        "id": 101,
        "image": "image_url",
        "name": "Sketchers",
        "category": "Running",
        "sizes": "9",
        "price": 1000
    }
}
```

this is the output for POST add a Product.

And then select another method as Put update a Product



Capstone Project / Update a product

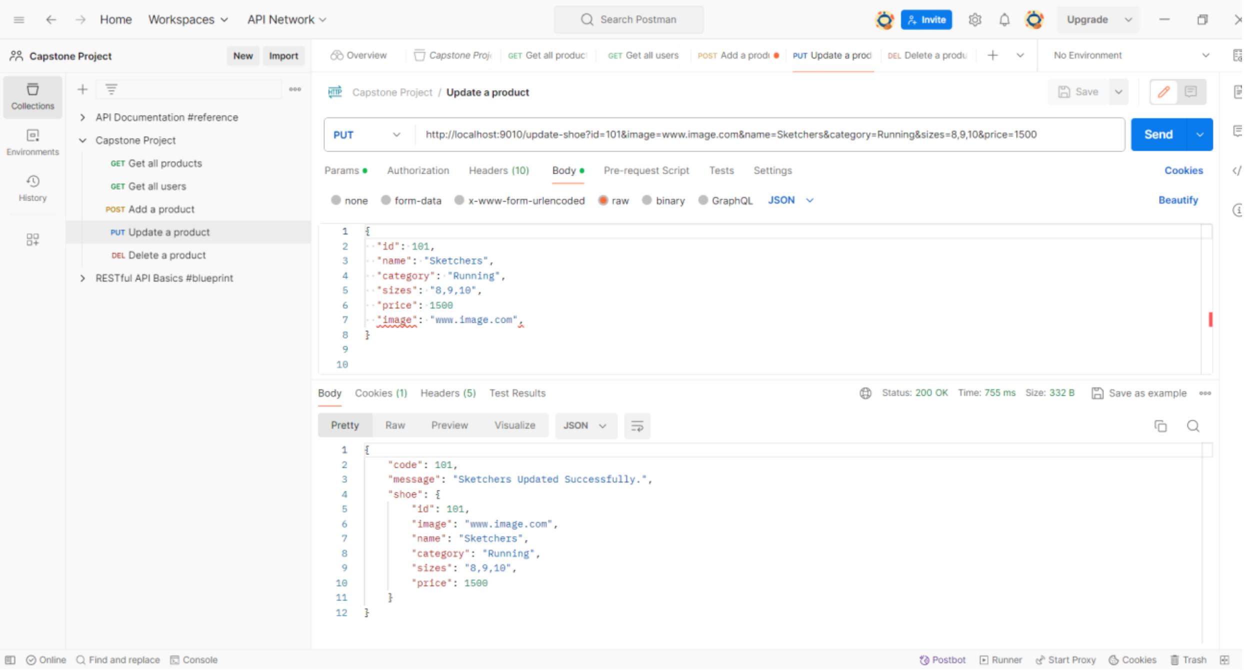
PUT http://localhost:9010/update-shoe?id=101&image=www.image.com&name=Sketchers&category=Running&sizes=8,9,10&price=1500

Params ● Authorization Headers (10) Body ● Pre-request Script Tests Settings

Query Params

☑	Key	Value	Description
☑	id	101	
☑	image	www.image.com	
☑	name	Sketchers	
☑	category	Running	
☑	sizes	8,9,10	
☑	price	1500	
	Key	Value	Description

And insert the URL click on save and send Request



Capstone Project / Update a product

PUT http://localhost:9010/update-shoe?id=101&image=www.image.com&name=Sketchers&category=Running&sizes=8,9,10&price=1500

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

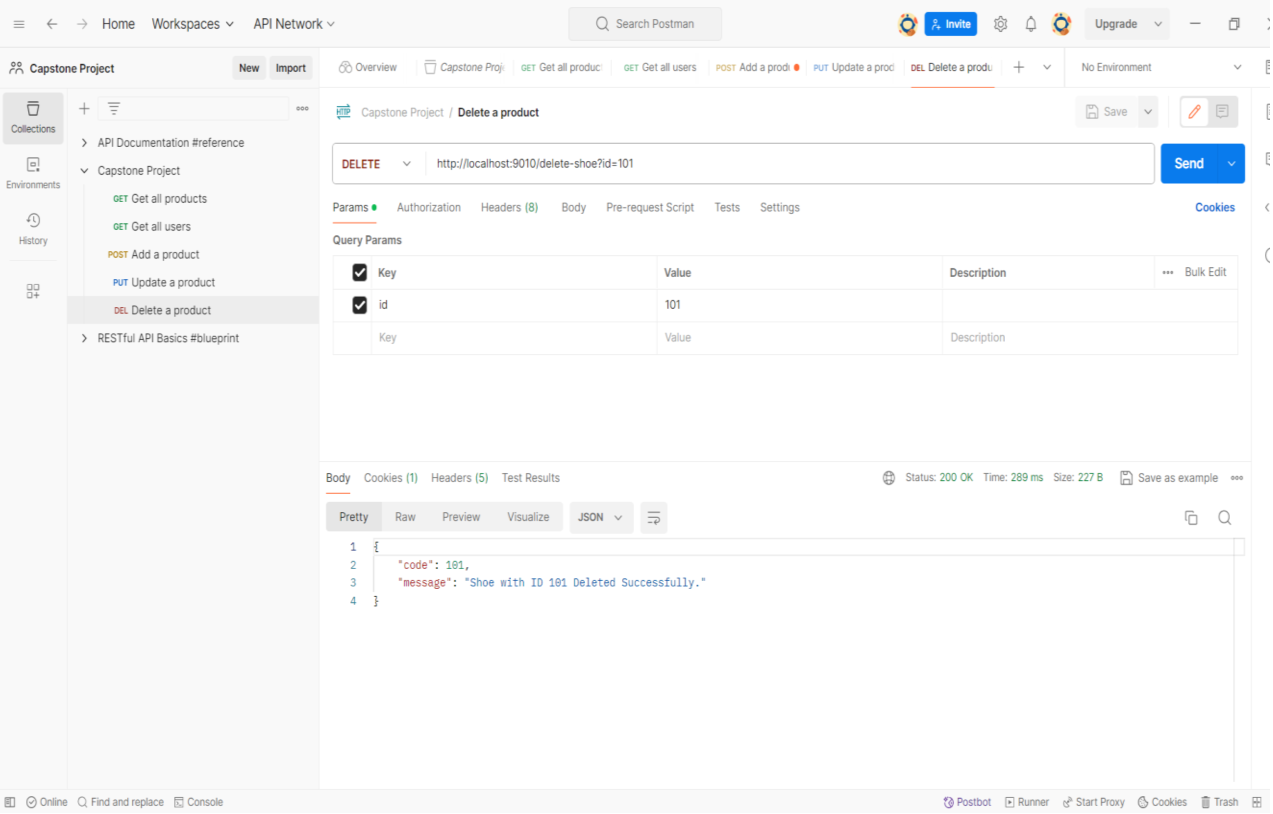
```
{
  "id": 101,
  "name": "Sketchers",
  "category": "Running",
  "sizes": "8,9,10",
  "price": 1500
  "image": "www.image.com",
}
```

Body Cookies (1) Headers (5) Test Results Status: 200 OK Time: 755 ms Size: 332 B Save as example

```
{
    "code": 101,
    "message": "Sketchers Updated Successfully.",
    "shoe": {
        "id": 101,
        "image": "www.image.com",
        "name": "Sketchers",
        "category": "Running",
        "sizes": "8,9,10",
        "price": 1500
    }
}
```

This is the output for PUT Update a product

Next select another request Delete a Product



Capstone Project / Delete a product

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

Query Params

	Key	Value	Description
☑	id	101	
	Key	Value	Description

Body Cookies (1) Headers (5) Test Results Status: 200 OK Time: 289 ms Size: 227 B Save as example

```
{
    "code": 101,
    "message": "Shoe with ID 101 Deleted Successfully."
}
```

Enter link as http://localhost:9010/delete-shoe?id=101

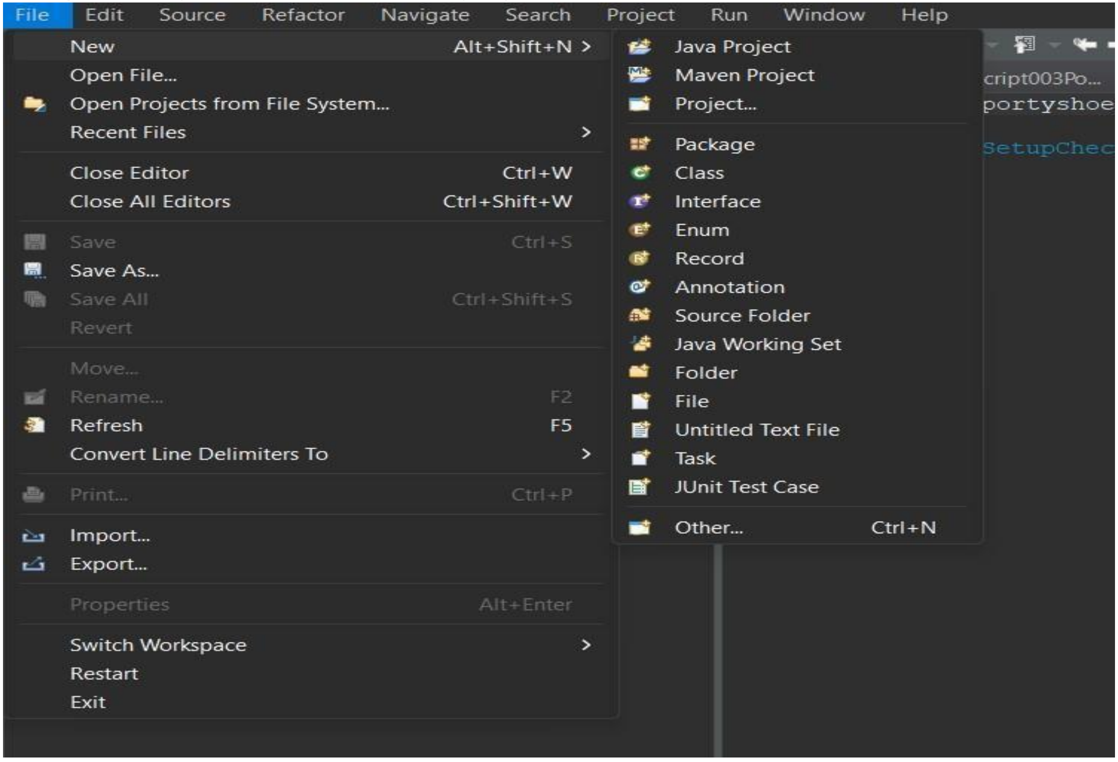
Click on save and run it

Rest-Assured:

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.

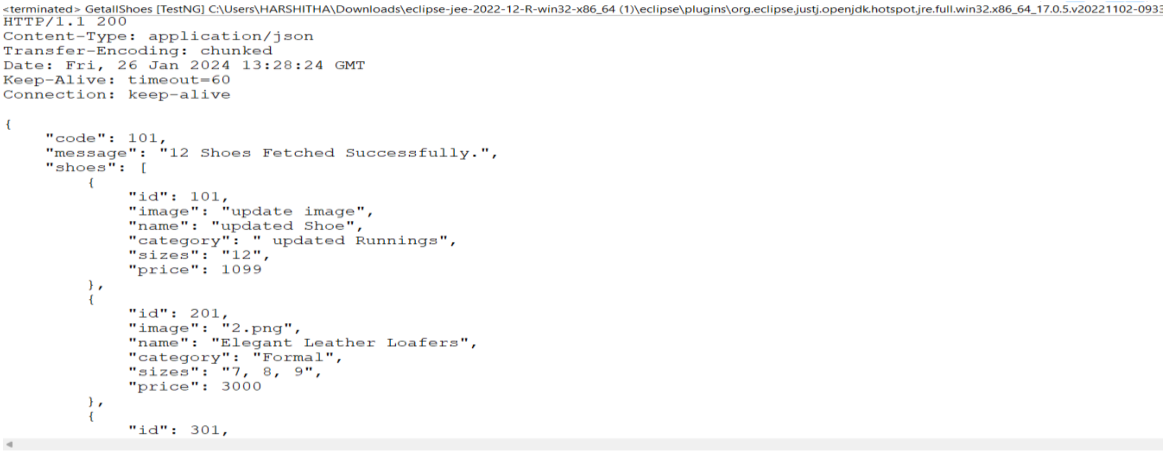
First go to Eclipse file >new>maven project



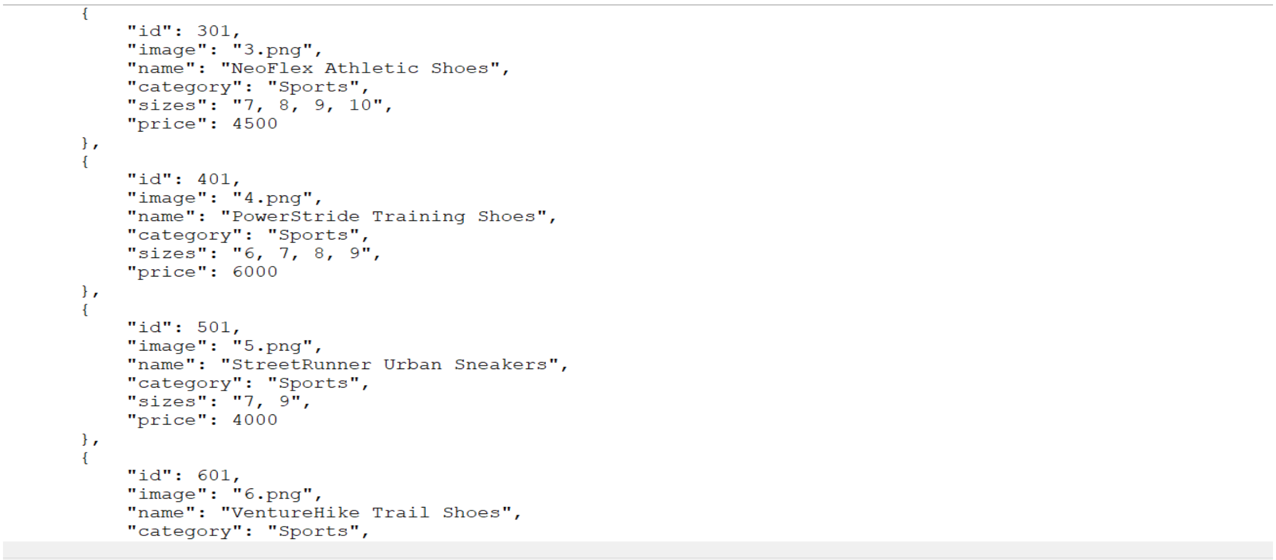
And enter name as capstone project
In capstone project add a package com.spotryshoe.restAssuredscripts
Put class name as getallShoes.java Code:

```
package com.sportyshoe.restAssuredScripts;

import org.testng.annotations.Test;
import io.restassured.RestAssured;
import static io.restassured.RestAssured.given;
import static org.hamcrest.Matchers.*;

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();
                }




}
```

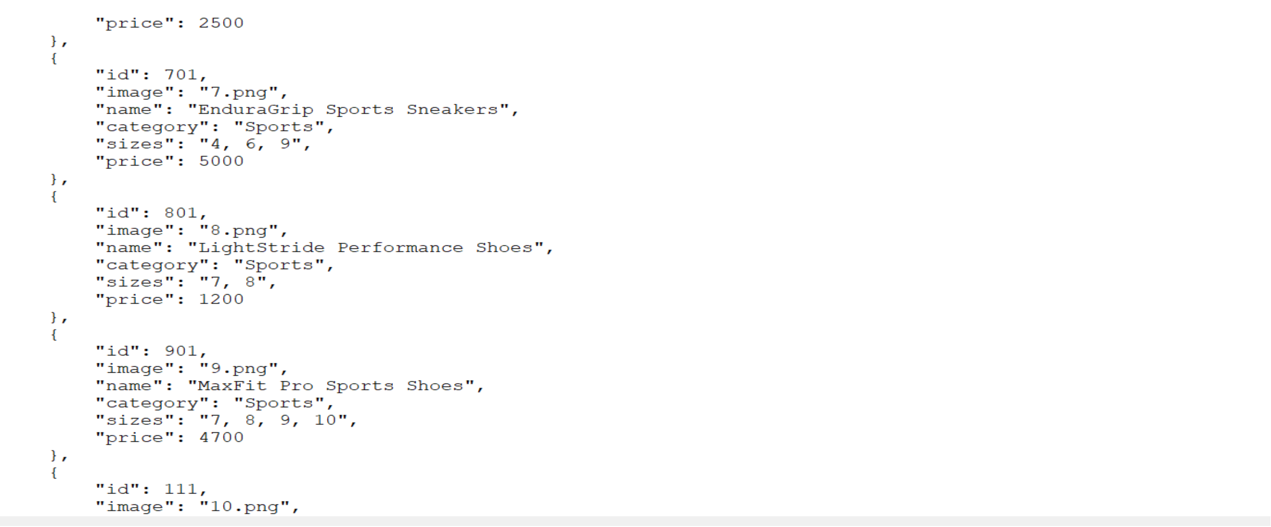
Output:



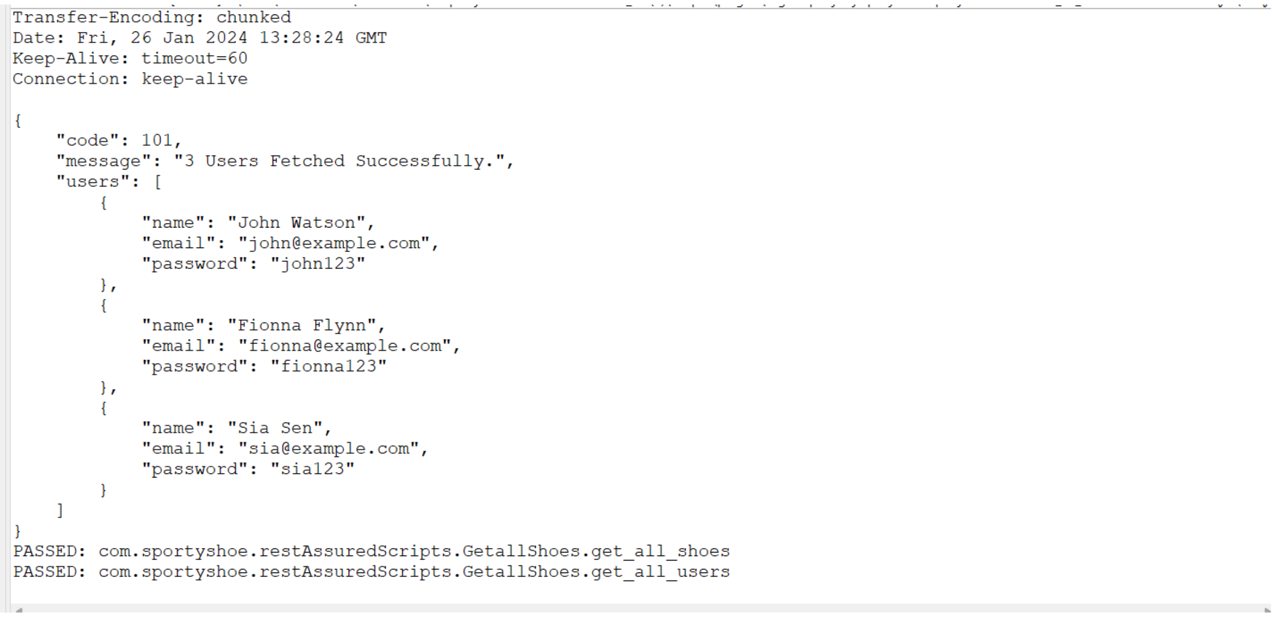
```
<terminated> GetallShoes [TestNG] C:\Users\HARSHITHA\Downloads\eclipse-jee-2022-12-R-win32-x86_64 (1)\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.5.v20221102-0933
HTTP/1.1 200
Content-Type: application/json
Transfer-Encoding: chunked
Date: Fri, 26 Jan 2024 13:28:24 GMT
Keep-Alive: timeout=60
Connection: keep-alive

{
    "code": 101,
    "message": "12 Shoes Fetched Successfully.",
    "shoes": [
        {
            "id": 101,
            "image": "update image",
            "name": "updated Shoe",
            "category": " updated Runnings",
            "sizes": "12",
            "price": 1099
        },
        {
            "id": 201,
            "image": "2.png",
            "name": "Elegant Leather Loafers",
            "category": "Formal",
            "sizes": "7, 8, 9",
            "price": 3000
        },
        {
            "id": 301,
```



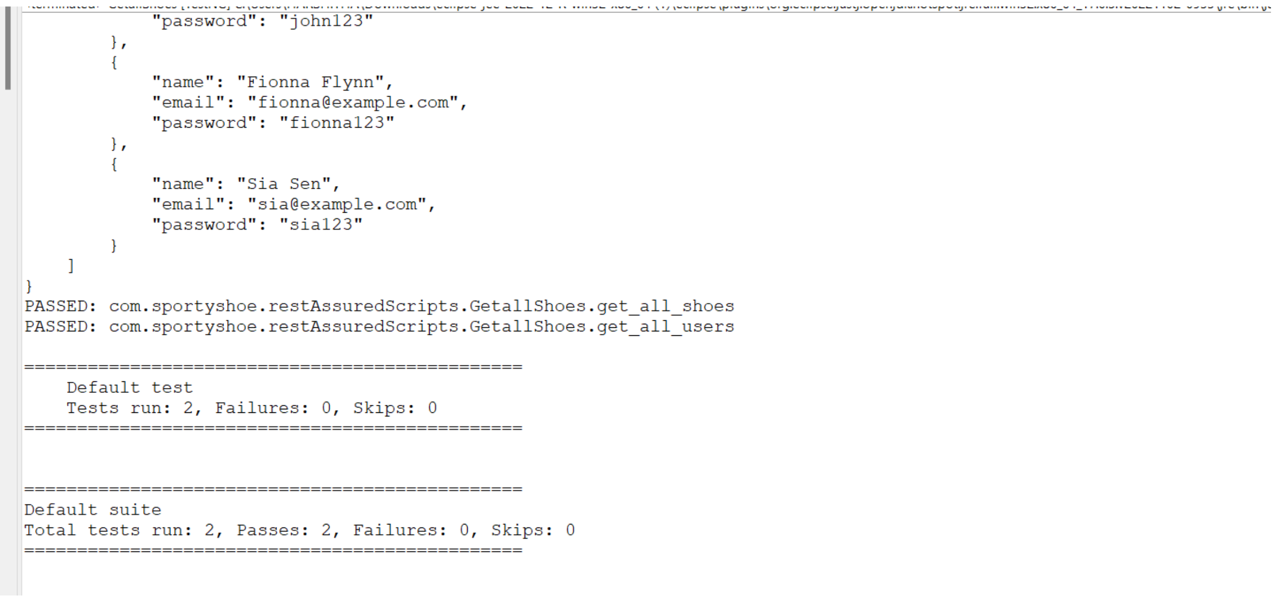
```
        {
            "id": 301,
            "image": "3.png",
            "name": "NeoFlex Athletic Shoes",
            "category": "Sports",
            "sizes": "7, 8, 9, 10",
            "price": 4500
        },
        {
            "id": 401,
            "image": "4.png",
            "name": "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",
```



```
        "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",
```



```
Transfer-Encoding: chunked
Date: Fri, 26 Jan 2024 13:28:24 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_shoes
PASSED: com.sportyshoe.restAssuredScripts.GetallShoes.get_all_users
```



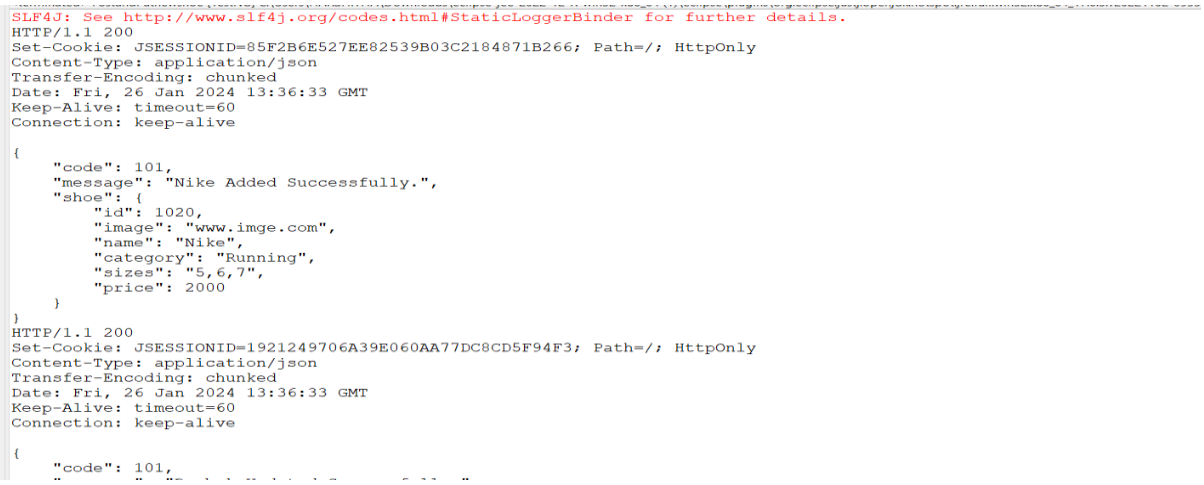
```
            "password": "john123"
        },
        {
            "name": "Fionna Flynn",
            "email": "fionna@example.com",
            "password": "fionna123"
        },
        {
            "name": "Sia Sen",
            "email": "sia@example.com",
            "password": "sia123"
        }
    ]
}
PASSED: com.sportyshoe.restAssuredScripts.GetallShoes.get_all_shoes
PASSED: com.sportyshoe.restAssuredScripts.GetallShoes.get_all_users
```

```
===============================================
    Default test
    Tests run: 2, Failures: 0, Skips: 0
===============================================


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

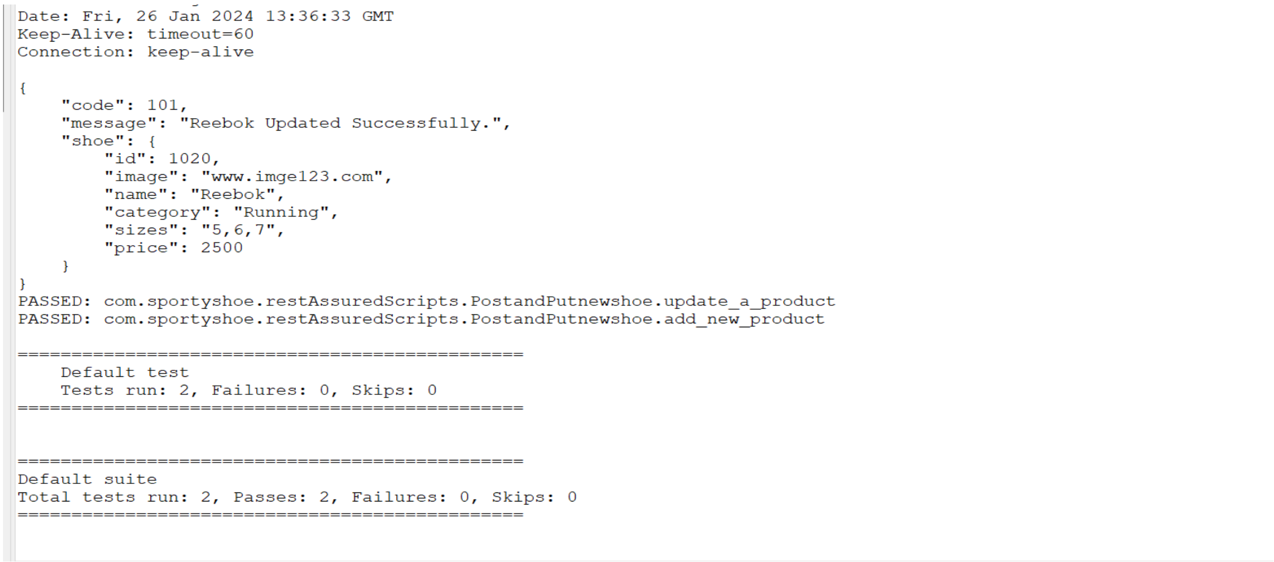
And Create another class as post and put new shoe Code:

```
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();

}

}
```

OUT PUT:



```
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
HTTP/1.1 200
Set-Cookie: JSESSIONID=85F2B6E527EE82539B03C2184871B266; Path=/; HttpOnly
Content-Type: application/json
Transfer-Encoding: chunked
Date: Fri, 26 Jan 2024 13:36:33 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=1921249706A39E060AA77DC8CD5F94F3; Path=/; HttpOnly
Content-Type: application/json
Transfer-Encoding: chunked
Date: Fri, 26 Jan 2024 13:36:33 GMT
Keep-Alive: timeout=60
Connection: keep-alive

{
    "code": 101,
```

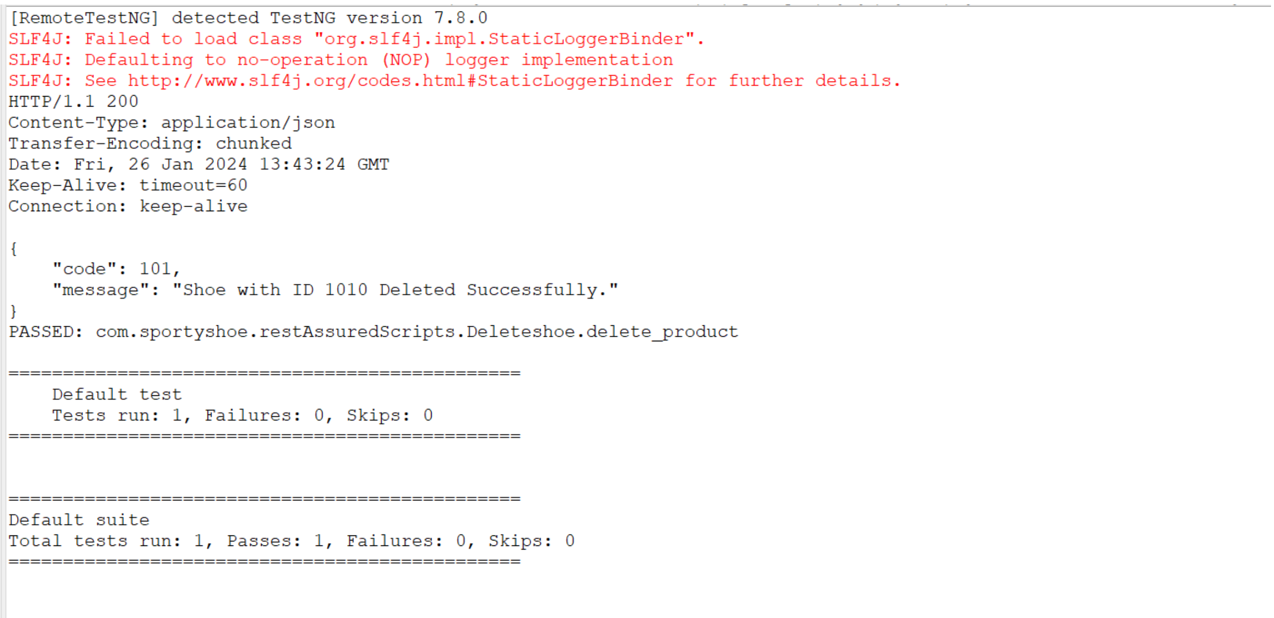


```
Date: Fri, 26 Jan 2024 13:36:33 GMT
Keep-Alive: timeout=60
Connection: keep-alive

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

===============================================
    Default test
    Tests run: 2, Failures: 0, Skips: 0
===============================================


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

Next add delete Rest assured codeCode:

```
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();

}

}
```

OUT PUT:



```
[RemoteTestNG] detected TestNG version 7.8.0
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: Fri, 26 Jan 2024 13:43:24 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:

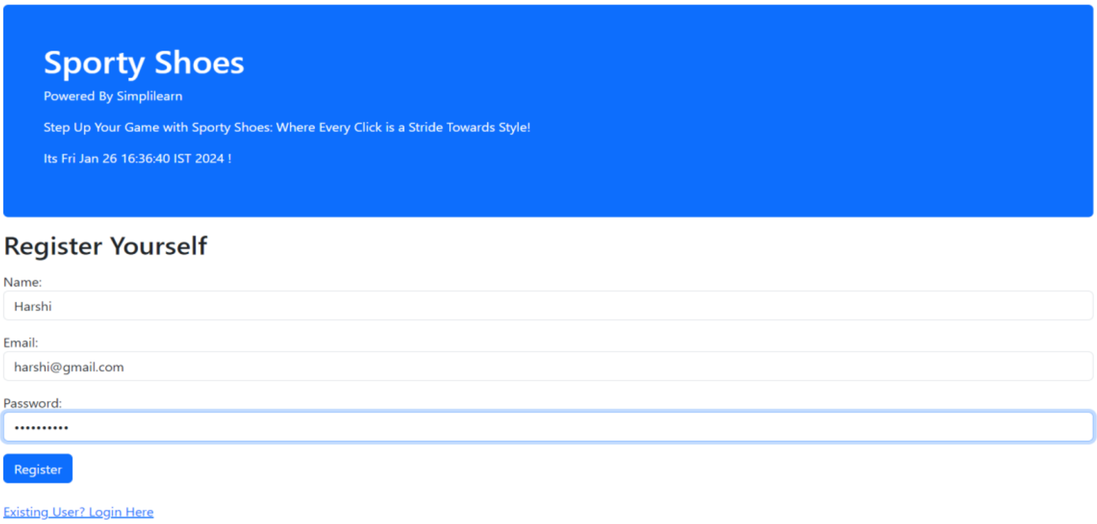
1. Create Selenium scripts using TestNG to test all the pages in the web app that will automate:
    - Login page
    - Registration Page
    - Add Product to cart page
    - Place Order Page

```
}
```

Registration Page

```
package com.sportyshoe.Tests;

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 {

@FindBy(css="body > div:nth-child(2) > form > a")

private WebElement register;

@FindBy(name="name")

private WebElement name;

@FindBy(name="email")

private WebElement email;

@FindBy(name="password")

private WebElement password;

@FindBy(xpath = "//button[contains(@class,'btn btn-primary')]")

private WebElement registerbtn;

public RegisterPage(WebDriver driver)

{

PageFactory.initElements(driver,this);

}

public void registerPage()

{

register.click();

}

public void EnterName()

{

name.sendKeys("Harshi");

}
public void EnterEmail() {

email.sendKeys("harshi@gmail.com");

}

public void EnterPassword()

{

password.sendKeys("Harshitha@12345");

}

public void clickRegisterBtn()

{

registerbtn.click();

}

}
```

OUT PUT:



Sporty Shoes

Powered By Simplilearn

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

Its Fri Jan 26 16:36:40 IST 2024 !

Register Yourself

Name: Harshi

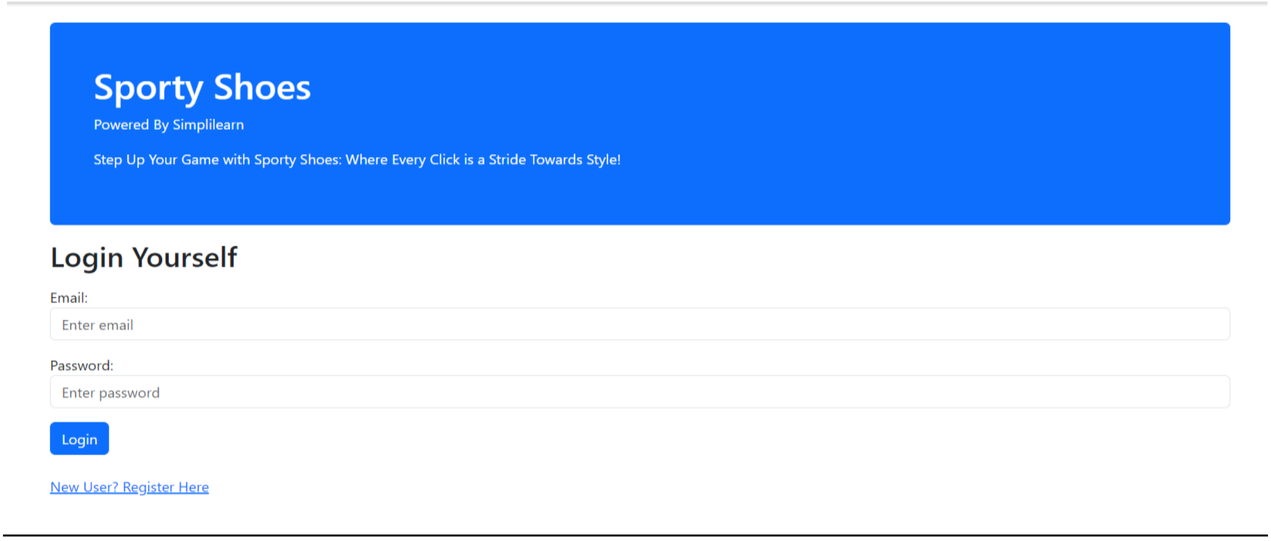
Email: harshi@gmail.com

Password: ••••••••••

Login page

```
package com.sportyshoe.Tests;

import org.testng.Assert;

import org.testng.annotations.Test;

public class TestPage extends BaseTest {

@Test

public void test() throws InterruptedException
{
RegisterPage page = new RegisterPage(driver);
page.registerPage();
page.EnterName();
page.EnterEmail();
page.EnterPassword();
Thread.sleep(3000);
page.clickRegisterBtn();
AddToCart add = new AddToCart(driver);
add.clickprofile();
add.clickHome();
add.clickAddToCart();
//add.addSecondItem();
String expectedErrMsg = "Message:Shoe Elegant Leather Loafers Added Successfully to Cart";
String acutalErrMsg = add.CheckVerifyCartMsg();
Assert.assertEquals(acutalErrMsg, expectedErrMsg);
ViewCartPage cart = new ViewCartPage(driver);
cart.clickHomeBtn();
cart.clickCartBtn();
PlaceOrder placeOrder = new PlaceOrder(driver);
placeOrder.clickPlaceOrderBtn();
String expectedMsg = "Success!";
String acutalMsg = placeOrder.CheckVerifyOrderMsg();
Assert.assertEquals(acutalMsg, expectedMsg);
}
}
```

OUTPUT:



Sporty Shoes

Powered By Simplilearn

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

Login Yourself

Email:

Enter email

Password:

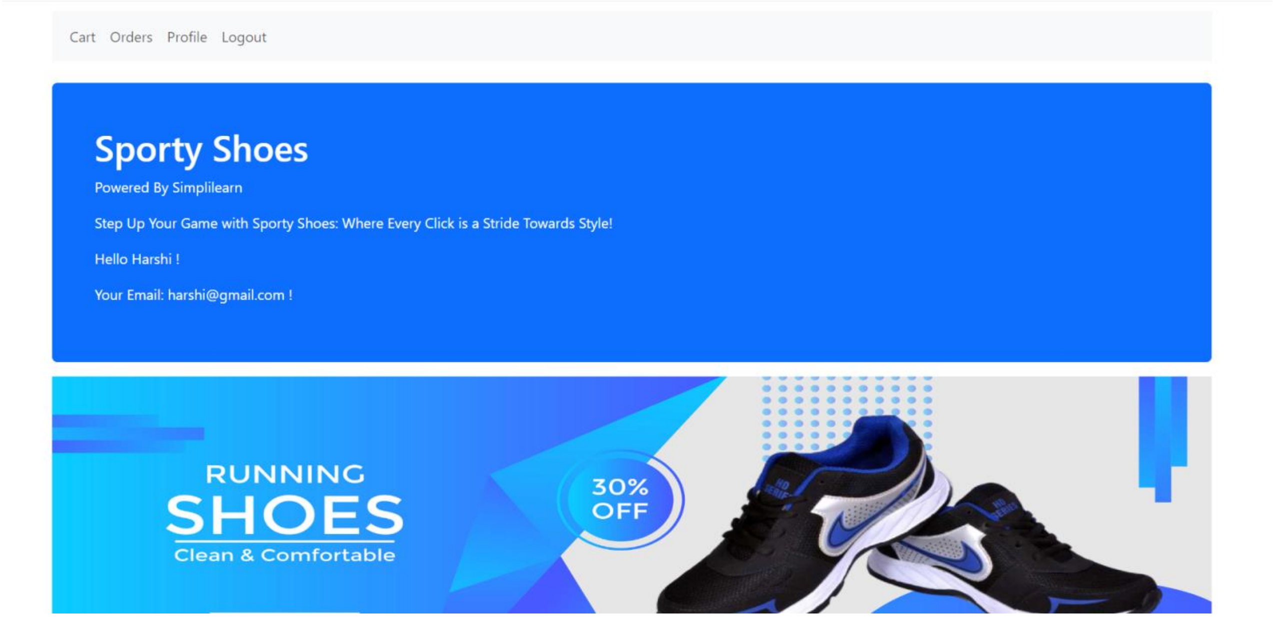
Enter password

Add Product to cart page

```
package com.sportyshoe.Tests;

import java.time.Duration;

import org.openqa.selenium.JavascriptExecutor;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.interactions.Actions;

import org.openqa.selenium.support.FindBy;

import org.openqa.selenium.support.PageFactory;

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

public class AddToCart {

Actions actions;

WebDriverWait wait;

JavascriptExecutor js;

public AddToCart(WebDriver driver)

{

js = (JavascriptExecutor) driver;

PageFactory.initElements(driver,this);

actions = new Actions(driver);

wait = new WebDriverWait(driver, Duration.ofSeconds(60));
}
@FindBy(css = "#mynavbar > ul > li:nth-child(3) > a")
private WebElement profile;
public void clickprofile()
{
profile.click();
}
@FindBy(css = "#mynavbar > ul > li:nth-child(1) > a")
private WebElement home;
public void clickHome()
{
home.click();
}
@FindBy(xpath = "(//a[contains(@class,'btn btn-primary')])[2]")
private WebElement firstItem;
public void clickAddToCart() throws InterruptedException
{
Thread.sleep(2000);
js.executeScript("arguments[0].scrollIntoView();", firstItem);
js.executeScript("arguments[0].click()", firstItem);
}
// @FindBy(xpath = "(//a[contains(@class,'btn btn-primary')])[2]")
// private WebElement secItem;
//
// public void addSecondItem()
// {
// js.executeScript("arguments[0].scrollIntoView();", firstItem);
// js.executeScript("arguments[0].click()", firstItem);
// }
```

```
//

// @FindBy(xpath = "//p[text()=\"Message:Shoe BlueWave Running Shoes Added Successfully to Cart\"]")

@FindBy(xpath = "/html/body/div[3]/div/p")

private WebElement verifycartmsg;

public String CheckVerifyCartMsg()

{

String textmsg = verifycartmsg.getText();

return textmsg;

}

}
```

OUT PUT:



Sporty Shoes

Powered By Simplilearn

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

Hello Harshi !

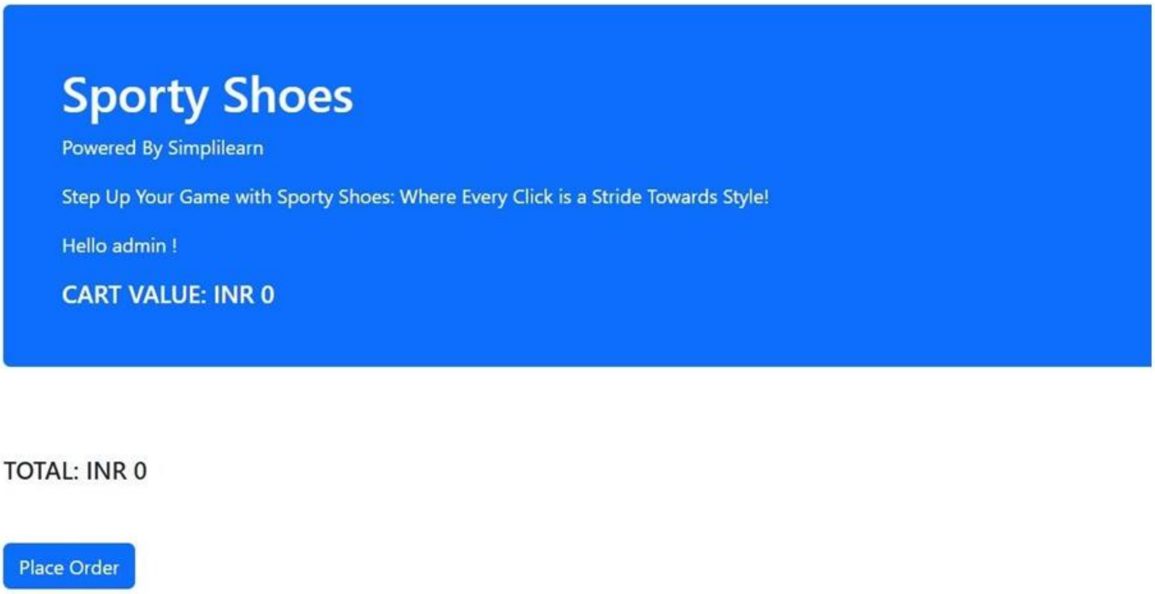
Your Email: harshi@gmail.com !

Place Order Page

```
package com.sportyshoe.Tests;

import java.time.Duration;

import org.openqa.selenium.JavascriptExecutor;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.interactions.Actions;

import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;
import org.openqa.selenium.support.ui.WebDriverWait;
public class PlaceOrder {
Actions actions;
WebDriverWait wait;
JavascriptExecutor js;
public PlaceOrder(WebDriver driver)
{
js=(JavascriptExecutor)driver;
PageFactory.initElements(driver,this);
actions =new Actions(driver);
wait = new WebDriverWait(driver,Duration.ofSeconds(60));
}
@FindBy(xpath = "(//a[contains(@class,'btn btn-primary')])[1]")
private WebElement placeOrderBtn;
public void clickPlaceOrderBtn()throws InterruptedException
{
Thread.sleep(2000);
js.executeScript("arguments[0].scrollIntoView()",placeOrderBtn);
js.executeScript("arguments[0].click()",placeOrderBtn);
}
@FindBy(xpath = "/html/body/div[3]/div/strong")
private WebElement verifyorderMsg;
public String CheckVerifyOrderMsg()
{
String message = verifyorderMsg.getText();
return message;
}
```

}

OUT PUT:



Sporty Shoes

Powered By Simplilearn

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

Hello admin !

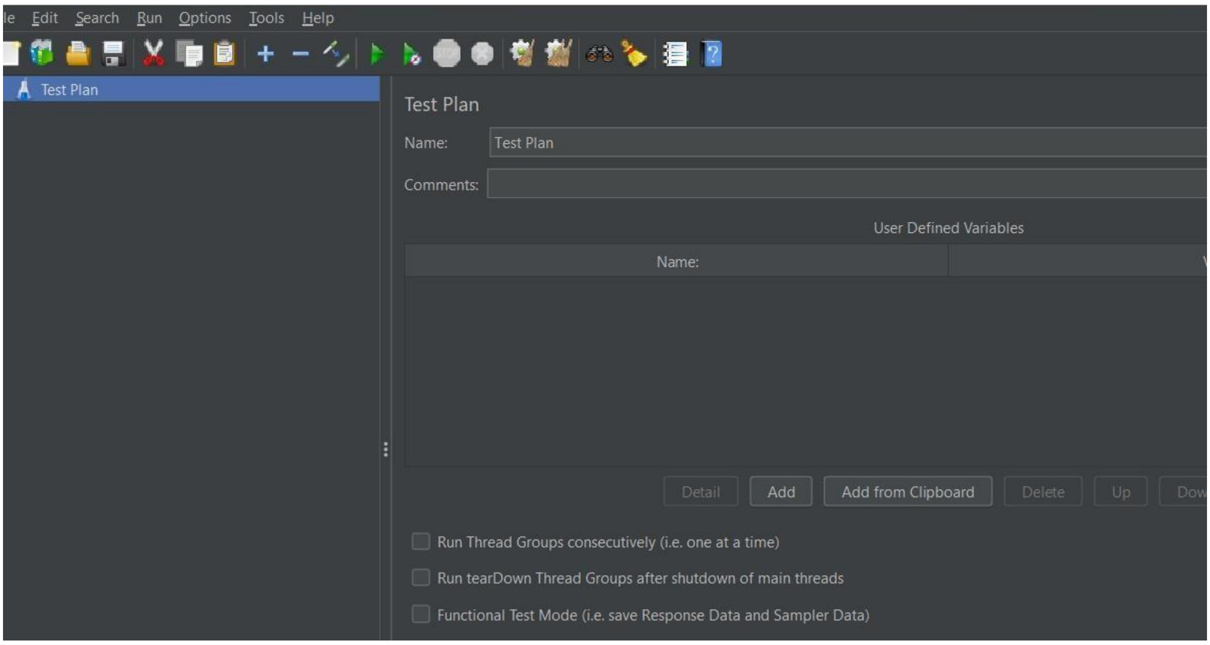
CART VALUE: INR 0

TOTAL: INR 0

Jmeter:

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

First go to Apache Jmeter



Test Plan

Name: Test Plan

Comments:

User Defined Variables

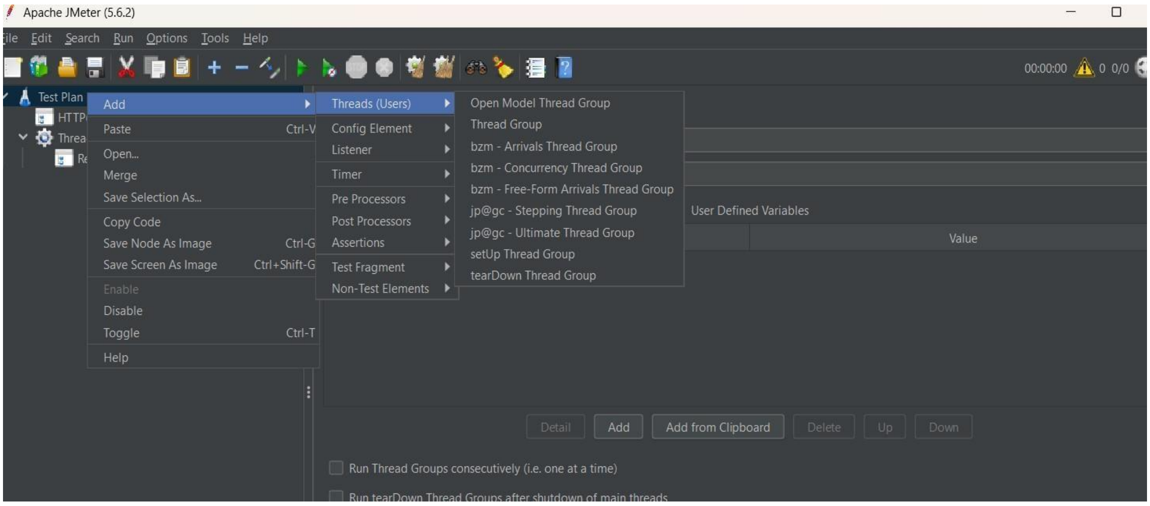
Name:

□ Run Thread Groups consecutively (i.e. one at a time)

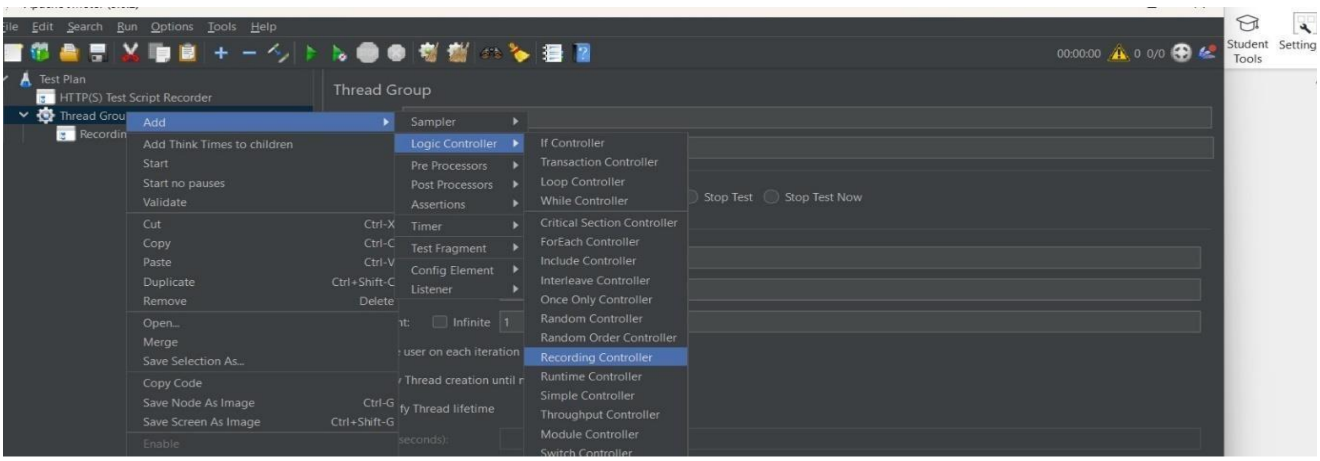
□ Run tearDown Thread Groups after shutdown of main threads

□ Functional Test Mode (i.e. save Response Data and Sampler Data)

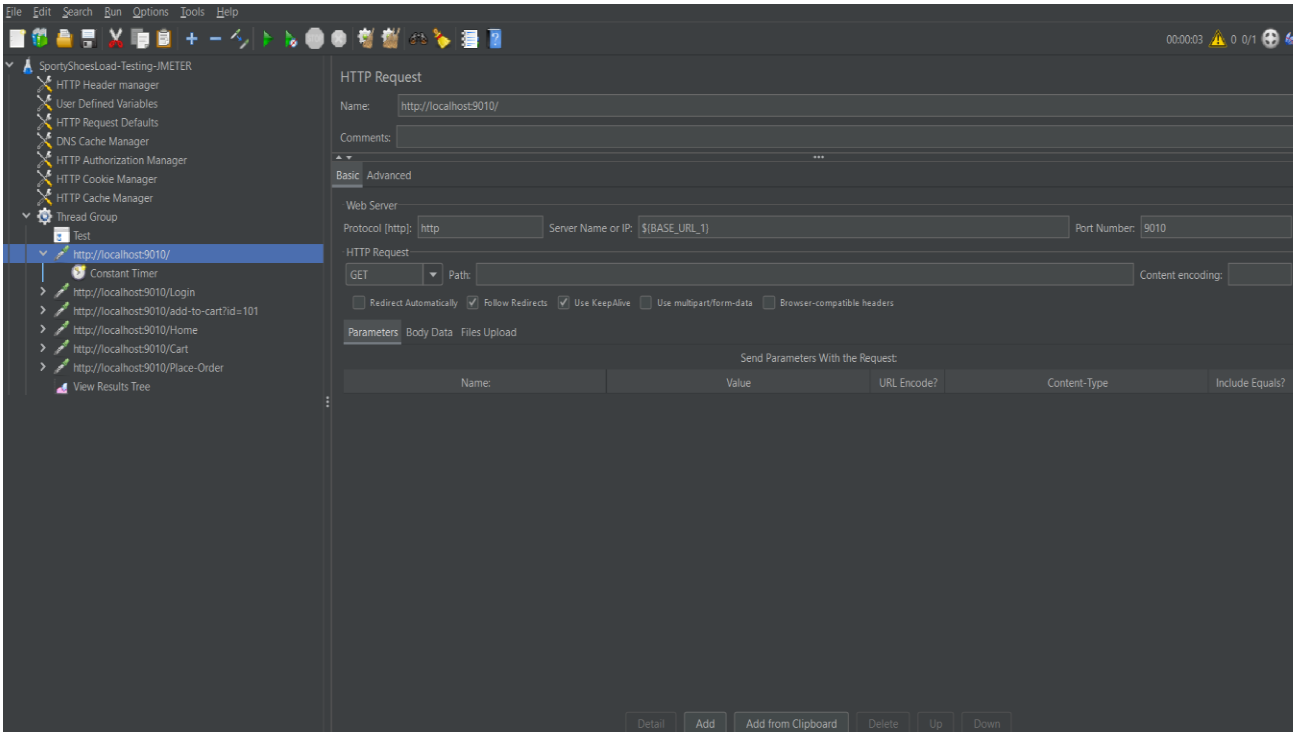
In this test plan add HTTP(s) Test Script Recorder
Next Add thread group



In thread group add controller >logic controller>Recording controller

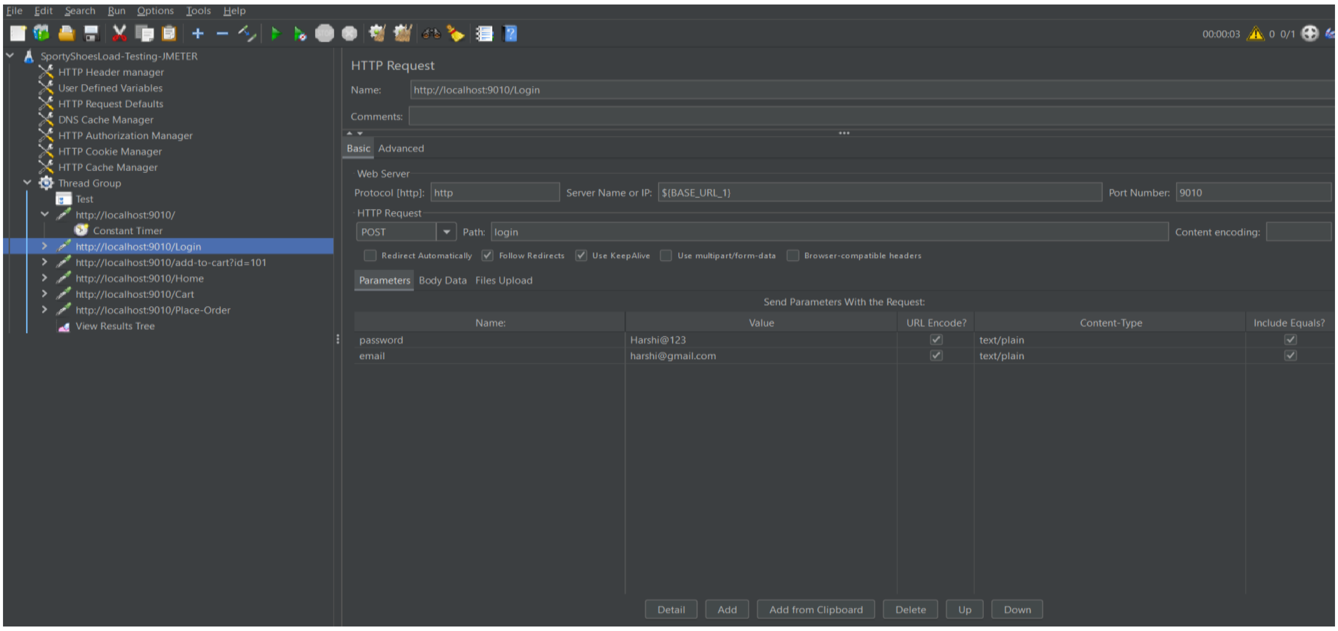


Create a http://localhost:9010/



Name: http://localhost:9010/
Comments:
Protocol [http]: http
Server Name or IP: ${BASE_URL_1}
Port Number: 9010
GET
Path:
Content encoding:
□ Redirect Automatically ☑ Follow Redirects ☑ Use KeepAlive □ Use multipart/form-data □ Browser-compatible headers

Login:



SportyShoesLoad-Testing-JMETER
- HTTP Header manager
- User Defined Variables
- HTTP Request Defaults
- DNS Cache Manager
- HTTP Authorization Manager
- HTTP Cookie Manager
- HTTP Cache Manager
- Thread Group
  - Test
  - http://localhost:9010/
    - Constant Timer
  - http://localhost:9010/Login
  - http://localhost:9010/add-to-cart?id=101
  - http://localhost:9010/Home
  - http://localhost:9010/Cart
  - http://localhost:9010/Place-Order
  - View Results Tree

HTTP Request

Name: http://localhost:9010/Login

Comments:

Basic | Advanced

Web Server

Protocol [http]: http   Server Name or IP: ${BASE_URL_1}   Port Number: 9010

HTTP Request

POST   Path: login   Content encoding:

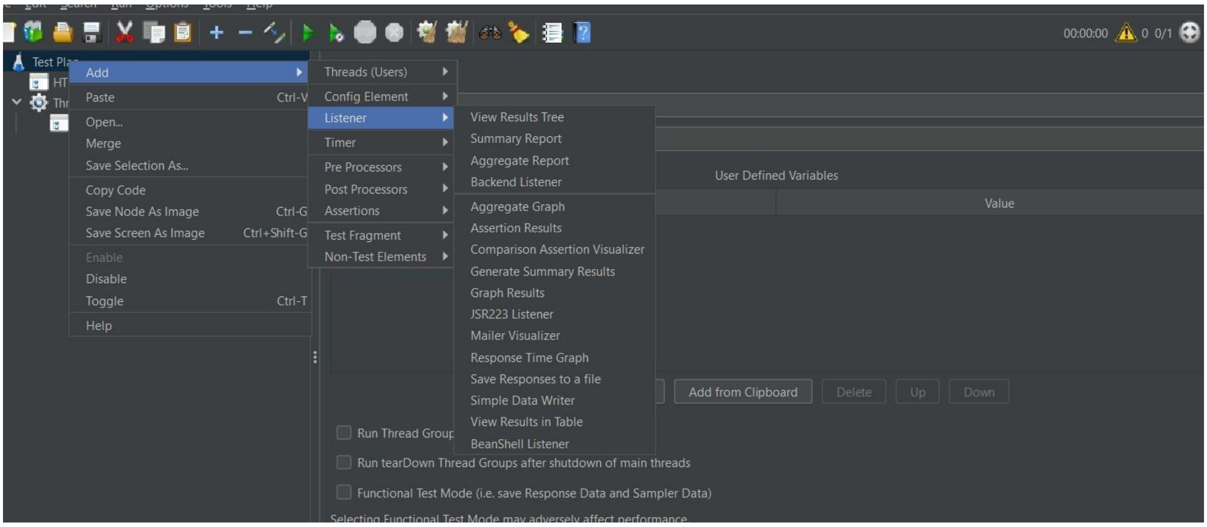
□ Redirect Automatically ☑ Follow Redirects ☑ Use KeepAlive □ Use multipart/form-data □ Browser-compatible headers

Parameters | Body Data | Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
password	Harshi@123	☑	text/plain	☑
email	harshi@gmail.com	☑	text/plain	☑

Add Listener to see results

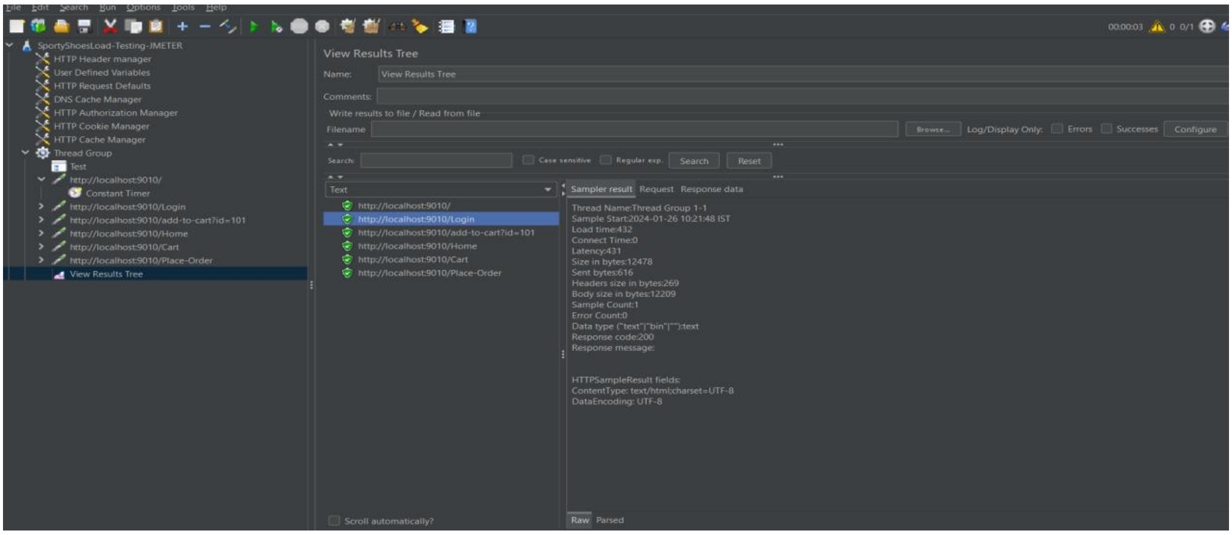


□ Run Thread Group

□ Run tearDown Thread Groups after shutdown of main threads

□ Functional Test Mode (i.e. save Response Data and Sampler Data)

Selecting Functional Test Mode may adversely affect performance



View Results Tree

Name: View Results Tree

Comments:

Write results to file / Read from file

Filename   Log/Display Only: □ Errors □ Successes

Search: □ Case sensitive □ Regular exp.

Text

- http://localhost:9010/
- http://localhost:9010/Login
- http://localhost:9010/add-to-cart?id=101
- http://localhost:9010/Home
- http://localhost:9010/Cart
- http://localhost:9010/Place-Order

```
Thread Name:Thread Group 1-1
Sample Start:2024-01-26 10:21:48 IST
Load time:432
Connect Time:0
Latency:431
Size in bytes:12478
Sent bytes:616
Headers size in bytes:269
Body size in bytes:12209
Sample Count:1
Error Count:0
Data type ("text"|"bin"|""):text
Response code:200
Response message:


HTTPSampleResult fields:
ContentType: text/html;charset=UTF-8
DataEncoding: UTF-8
```

□ Scroll automatically?

Source Code- Cucumber-Feature-Project:

CreateShoe.feature

```
@tag
Feature: Create a new shoe in the store
  I want to test the functionality of creating a new shoe

  @tag1
  Scenario: Create a new shoe on the website
    Given Open the API base URL
    And Enter the shoe details
    When Hit the POST request to create a new shoe
    Then The response code should be 200
    And Print the result of adding new shoe
```

DeleteProduct.feature

```
@tag
Feature: Delete a shoe from the store
  I want to test the functionality of deleting a shoe from the store

  @tag1
  Scenario: Delete a shoe from the website
    Given Open the API base URL
    And there is an existing shoe with ID 101
    When hit the DELETE request for the shoe with ID 101
    Then the response code should be 200
    And the response should indicate successful deletion
```

GetProducts.feature

```
Feature: Retrieve all products from the sports shoe store
  As a user,
  I want to retrieve the list of all available shoes from the sports shoe
store,
  So that I can browse and choose the desired products.

  @tag @GetProducts
  Scenario: Retrieve all products successfully
    Given I am interested in viewing the available shoes
    When I send a request to retrieve the list of products
    Then I should receive a successful response with status code 200
    And the response should contain a valid list of sports shoes
```

GetUsers.feature

```
Feature: Retrieve all registered users
  I want to retrieve the list of all available register users

  @tag @GetProducts
  Scenario: Retrieve all registered users
    Given I am interested in viewing the registered users
    When I send a request to retrieve the users
    Then I want to receive a successful response with status code 200
    And the response should contain a valid list of users
```

UpdateProduct.feature

```
@tag
Feature: Update a shoe in the store
  I want to test the functionality of updating a shoe in the store

  @tag1 @UpdateRequest
  Scenario: Update a shoe on the website
    Given open the update product base url
    When hit the PUT request to update the shoe with ID 101
    Then the response code should be equal to 200
    And the response should indicate successful update
```

DeleteProductSteps.java

```
package com.simplilearn.cucumber.stepdefinitions;

import static org.hamcrest.CoreMatchers.containsString;

import io.cucumber.java.en.Given;
```

```java
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;
import io.restassured.RestAssured;
import io.restassured.response.Response;
import io.restassured.response.ValidatableResponse;
import io.restassured.specification.RequestSpecification;

public class DeleteProductSteps {
    private Response response;
    private ValidatableResponse json;
    private RequestSpecification request;

    private String BASE_URL = "http://localhost:9010";



    @Given("there is an existing shoe with ID {int}")
    public void there_is_an_existing_shoe_with_id(Integer int1) {
       request=RestAssured.given().baseUri(BASE_URL);
    }

    @When("hit the DELETE request for the shoe with ID {int}")
    public void hit_the_delete_request_for_the_shoe_with_id(Integer shoeId) {
       response=request.delete("/delete-shoe?id="+shoeId);
    }

    @Then("the response code should be {int}")
    public void the_response_code_should_be(Integer code) {
       json=response.then().statusCode(code);
    }
@Then("the response should indicate successful deletion")
public void the_response_should_indicate_successful_deletion() {
   response.then().body("message", containsString("Shoe with ID 101 Deleted
Successfully"));
}


}
```

GetRequestSteps.java

```
package com.simplilearn.cucumber.stepdefinitions;


import org.hamcrest.Matchers;


import  io.cucumber.java.en.Given;
import   io.cucumber.java.en.Then;
import  io.cucumber.java.en.When;
import io.restassured.RestAssured;
import io.restassured.response.Response;
import       io.restassured.response.ValidatableResponse;
import io.restassured.specification.RequestSpecification;


public class GetRequestSteps {


            private Response response;
            private ValidatableResponse json;
```

```
        private RequestSpecification request;

        private String BASE_URL = "http://localhost:9010";

        @Given("I am interested in viewing the available shoes")
        public void i_am_interested_in_viewing_the_available_shoes() {
          request=RestAssured.given().baseUri(BASE_URL);
        }

        @When("I send a request to retrieve the list of products")
        public void i_send_a_request_to_retrieve_the_list_of_products() {
          response=request.when().get("/get-shoes");
        }

        @Then("I should receive a successful response with status code {int}")
        public void i_should_receive_a_successful_response_with_status_code(Integer
code) {
          json=response.then().statusCode(code);
          System.out.println(json.toString());
        }

        @Then("the response should contain a valid list of sports shoes")
        public void the_response_should_contain_a_valid_list_of_sports_shoes() {
                response.then().body("shoes", Matchers.notNullValue());
        }

}
```

GetUserSteps.java

```
package com.simplilearn.cucumber.stepdefinitions;


import org.hamcrest.Matchers;


import  io.cucumber.java.en.Given;
import   io.cucumber.java.en.Then;
import  io.cucumber.java.en.When;
import io.restassured.RestAssured;
import io.restassured.response.Response;
import      io.restassured.response.ValidatableResponse;
import io.restassured.specification.RequestSpecification;


public class GetUserSteps {



        private Response response;
        private ValidatableResponse json;
        private RequestSpecification request;


        private String BASE_URL = "http://localhost:9010";


        @Given("I am interested in viewing the registered users")
        public void i_am_interested_in_viewing_the_registered_users() {
           request=RestAssured.given().baseUri(BASE_URL);
        }


        @When("I send a request to retrieve the users")
```

```
    public void i_send_a_request_to_retrieve_the_users() {
      response=request.when().get("/get-users");
    }

    @Then("I want to receive a successful response with status code {int}")
    public void i_want_to_receive_a_successful_response_with_status_code(Integer code) {
      json=response.then().statusCode(code);
      System.out.println(json.toString());
    }


    @Then("the response should contain a valid list of users")
    public void the_response_should_contain_a_valid_list_of_users() {
          response.then().body("users", Matchers.notNullValue());
    }

}
```

PostRequestSteps.java

```
package com.simplilearn.cucumber.stepdefinitions;

import  io.cucumber.java.en.Given;
import   io.cucumber.java.en.Then;
import  io.cucumber.java.en.When;
import io.restassured.RestAssured;
import io.restassured.response.Response;
import io.restassured.response.ValidatableResponse;
import io.restassured.specification.RequestSpecification;

public class PostRequestSteps {



        private Response response;
        private ValidatableResponse json;
        private RequestSpecification request;
        private String requestBody;

        private String BASE_URL = "http://localhost:9010";

        @Given("Open the API base URL")
        public void open_the_api_base_url() {
          request=RestAssured.given().baseUri(BASE_URL);
        }

        @Given("Enter the shoe details")
        public void enter_the_shoe_details() {
                requestBody = "id=" + 101 + "&image=image_url&name=Sample
Shoes&category=Sports Shoe&sizes=9&price=1999";
        }

        @When("Hit the POST request to create a new shoe")
        public void hit_the_post_request_to_create_a_new_shoe() {
                int id = 101;
     String image_url = "image_url";
     String name = "sampleShoe";
     String category = "Runnings";
```

```
    int sizes = 9;
    int price = 1000;
        response=request.queryParam("id",id).queryParam("image",image_url)
       .queryParam("name",name).queryParam("category",category)
       .queryParam("sizes",sizes).queryParam("price",price).when().post("/add-shoe");
      }

      @Then("The response code should be {int}")
      public void the_response_code_should_be(Integer code) {
        json=response.then().statusCode(code);
        System.out.println(json);
      }

      @Then("Print the result of adding new shoe")
      public void print_the_result_of_adding_new_shoe() {
        System.out.println(response.getBody().asString());
      }
}
```

UpdateshoeSteps.java

```
package com.simplilearn.cucumber.stepdefinitions;


import org.hamcrest.Matchers;


import  io.cucumber.java.en.Given;
import   io.cucumber.java.en.Then;
import io.cucumber.java.en.When;
import  io.restassured.RestAssured;
import io.restassured.response.Response;
import io.restassured.response.ValidatableResponse;
import io.restassured.specification.RequestSpecification;

public class UpdateshoeSteps {



        private Response response;
        private ValidatableResponse json;
        private RequestSpecification request;

        private String BASE_URL = "http://localhost:9010";

        @Given("open the update product base url")
        public void open_the_update_product_base_url() {
           request=RestAssured.given().baseUri(BASE_URL);
        }



           @When("hit the PUT request to update the shoe with ID {int}")
           public void hit_the_put_request_to_update_the_shoe_with_id(Integer shoeId) {

                  int id = shoeId;
       String image= "update image";
       String name = "updated Shoe";
       String category = " updated Runnings";
       int sizes = 12;
       int price = 1099;
```

```
            response =request
                         .queryParam("id",id).queryParam("image",image)
                .queryParam("name",name).queryParam("category",category)
                .queryParam("sizes",sizes).queryParam("price",price)
                .when().put( "/update-shoe?id=" );
        }

        @Then("the response code should be equal to {int}")
        public void the_response_code_should_be_equal_to(Integer expectedStatusCode)
{
            json=response.then().statusCode(expectedStatusCode);
        }

        @Then("the response should indicate successful update")
        public void the_response_should_indicate_successful_update() {
              System.out.println(json.toString());
        }
}
```

TestRunner.java

```
package com.simplilearn.cucumber.testrunner;
import org.junit.runner.RunWith;

import io.cucumber.junit.Cucumber;
import io.cucumber.junit.CucumberOptions;

@RunWith(Cucumber.class)
```

```
@CucumberOptions(features = "src/test/resources/Features", glue =
{"com/simplilearn/cucumber/stepdefinitions" })

public class TestRunner {


}
```

Pom.xml

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

<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
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.simplilearn.cucumber</groupId>
  <artifactId>PhaseEnd_Cucumber_Restassured_Project</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <name>PhaseEnd_Cucumber_Restassured_Project</name>
  <!-- FIXME change it to the project's website -->
  <url>http://www.example.com</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.target>8</maven.compiler.target>
            <maven.compiler.source>8</maven.compiler.source>
            <rest.assured.version>5.4.0</rest.assured.version>
            <junit.version>4.13.2</junit.version>
            <cucumber.version>7.15.0</cucumber.version>
      </properties>

      <dependencies>
            <!-- rest-assured -->
            <dependency>
                  <groupId>io.rest-assured</groupId>
                  <artifactId>rest-assured</artifactId>
                  <version>${rest.assured.version}</version>
                  <scope>test</scope>
            </dependency>

            <!-- rest-assured/json-path -->
            <dependency>
                  <groupId>io.rest-assured</groupId>
                  <artifactId>json-path</artifactId>
                  <version>${rest.assured.version}</version>
                  <scope>test</scope>
            </dependency>

            <!-- rest-assured/json-schema-validator -->
            <dependency>
```

```
                <groupId>io.rest-assured</groupId>
                <artifactId>json-schema-validator</artifactId>
                <version>${rest.assured.version}</version>
            </dependency>

            <!-- io.rest-assured/xml-path -->
            <dependency>
                <groupId>io.rest-assured</groupId>
                <artifactId>xml-path</artifactId>
                <version>${rest.assured.version}</version>
            </dependency>

            <!-- cucumber-java -->
            <dependency>
                <groupId>io.cucumber</groupId>
                <artifactId>cucumber-java</artifactId>
                <version>${cucumber.version}</version>
            </dependency>

            <!-- cucumber-junit :: junit4-->
            <dependency>
                <groupId>io.cucumber</groupId>
                <artifactId>cucumber-junit</artifactId>
                <version>${cucumber.version}</version>
                <scope>test</scope>
            </dependency>

            <!-- junit4 -->
            <dependency>
                <groupId>junit</groupId>
                <artifactId>junit</artifactId>
                <version>${junit.version}</version>
                <scope>test</scope>
            </dependency>

        </dependencies>
        <build>
            <plugins>
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-surefire-plugin</artifactId>
                    <version>3.2.3</version>
                </plugin>
            </plugins>
        </build>
    </project>
```
