**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.

**GITHUB-URL:** [**https://github.com/Vijay-Vel/CapstoneProject-SportyShoes.git**](https://github.com/Vijay-Vel/CapstoneProject-SportyShoes.git)

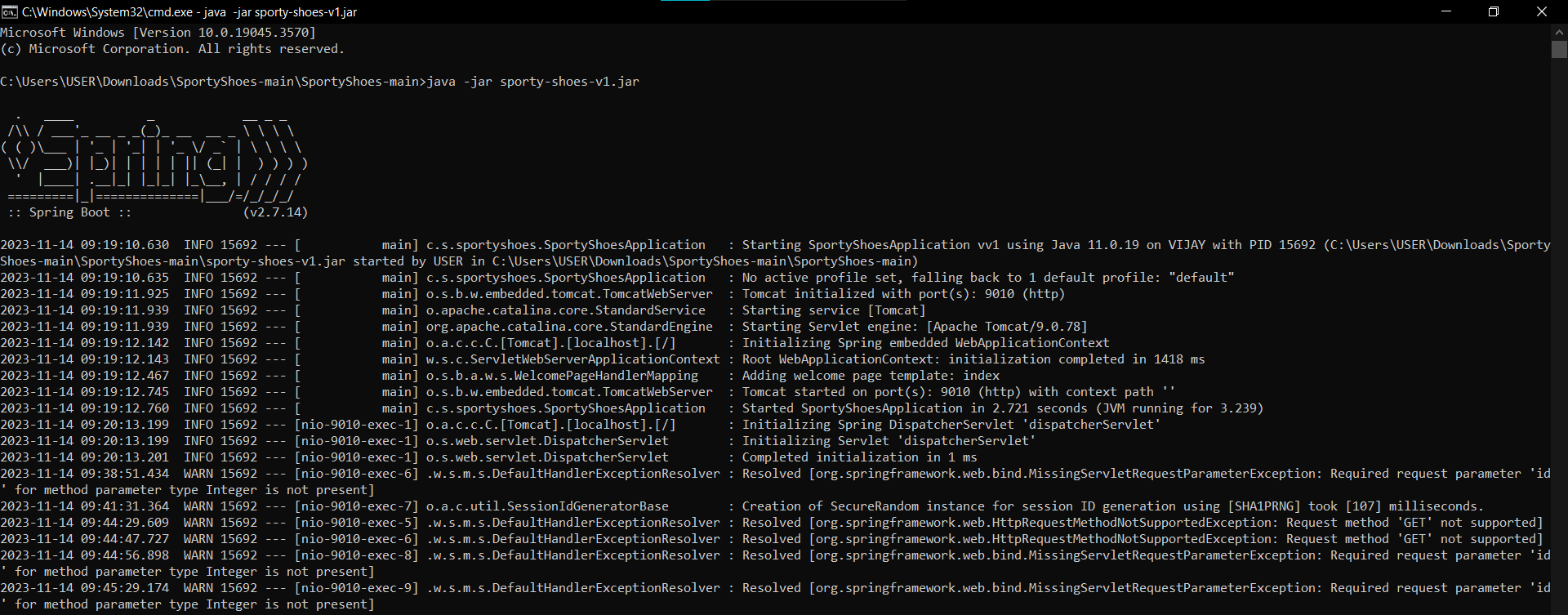
**IMPLEMENTATION REQUIREMENTS**

The following deliverables are expected:

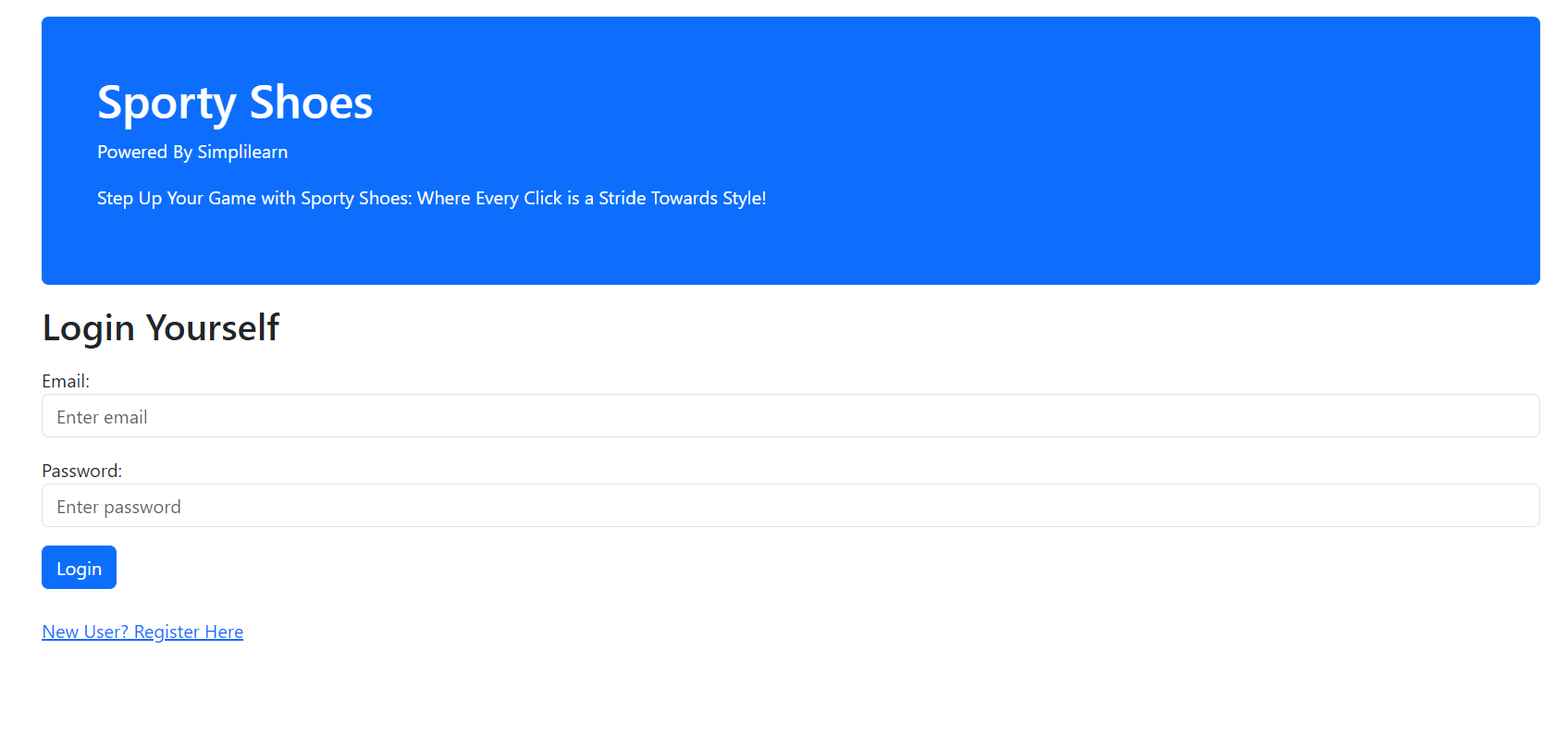
1. Automate the below api endpoints using rest-assured
2. Create Selenium scripts using TestNG to test all the pages in the web app
3. Create JMeter scripts to do load testing of the homepage and the login page.
4. Setup Cucumber in Java Project and write Feature Files using Gherkin to test the API endpoints
5. Create Postman scripts to test the API endpoints

**EXPLANATION:**

Download the JAR file from the github and open command prompt and run the command #java -jar sporty-shoes.jar



Now open the chrome browser and on the address bar enter <http://localhost:9010>



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.
* **TO RETRIEVE THE LIST OF ALL PRODUCTS IN THE STORE.**

Open Eclipse

Create a Maven project

Add Maven dependency for TestNG and REST Assured

Create a TestNG test with below REST API execution details -

GET CALL

URL: <http://localhost:9010/get-shoes>

Service Type: GET

**SOURCE CODE:**

package com.sporty.shoe.RestAssuredScripts;

import org.hamcrest.Matchers;

import org.testng.annotations.Test;

import io.restassured.RestAssured;

import io.restassured.http.ContentType;

public class GetAllShoes {

@Test()

public void get\_all\_shoes() {

RestAssured.given()

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

.basePath("/get-shoes")

.contentType(ContentType.JSON)

.when()

.get()

.then().statusCode(200)

.log()

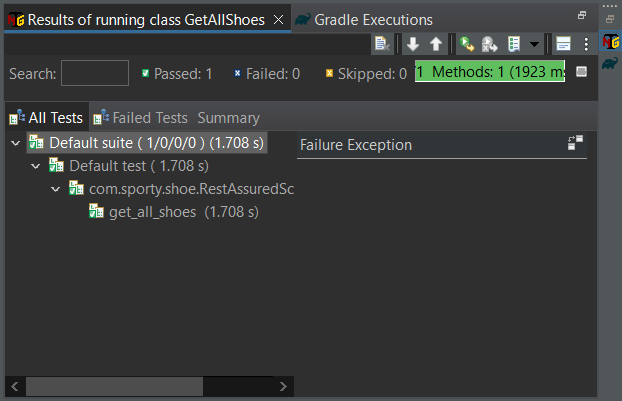
.all()

.body("code", Matchers.equalTo(101));

}

}

**TESTNG TEST WINDOW:**



**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 14:27:23 GMT

Keep-Alive: timeout=60

Connection: keep-alive

{

"code": 101,

"message": "17 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",

"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

},

{

"id": 100,

"image": "www.imge123.com",

"name": "Adidas",

"category": "Running",

"sizes": "5,6,7",

"price": 2500

},

{

"id": 102,

"image": "img102.src",

"name": "Adidas",

"category": "Sports+Fashion",

"sizes": "9,10,11,12",

"price": 5000

},

{

"id": 100,

"image": "www.img.com",

"name": "Nike",

"category": "Sports",

"sizes": "9,10",

"price": 3000

},

{

"id": 105,

"image": "img105.src",

"name": "Adidas",

"category": "Sports+Fashion",

"sizes": "11,12",

"price": 6000

},

{

"id": 106,

"image": "img106.src",

"name": "Adidas",

"category": "Fashion",

"sizes": "11,12",

"price": 8000

},

]

}

PASSED: com.sporty.shoe.RestAssuredScripts.GetAllShoes.get\_all\_shoes

===============================================

Default test

Tests run: 1, Failures: 0, Skips: 0

===============================================

===============================================

Default suite

Total tests run: 1, Passes: 1, Failures: 0, Skips: 0

===============================================

* **RETRIEVE THE LIST OF ALL REGISTERED USERS.**

Create a TestNG test with below REST API execution details -

GET CALL

URL: <http://localhost:9010/get-users>

Service Type: GET

**SOURCE CODE:**

package com.sporty.shoe.RestAssuredScripts;

import org.hamcrest.Matchers;

import org.testng.annotations.Test;

import io.restassured.RestAssured;

import io.restassured.http.ContentType;

public class GetAllUsers {

@Test()

public void get\_all\_users() {

RestAssured.given()

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

.basePath("/get-users")

.contentType(ContentType.JSON)

.when()

.get()

.then()

.statusCode(200)

.log()

.all()

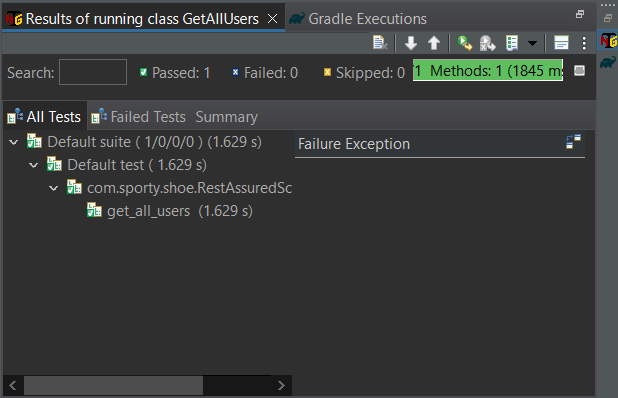
.body("code", Matchers.equalTo(101))

.body("message", Matchers.equalTo("3 Users Fetched Successfully."));

}

}

**TESTNG TEST WINDOW:**



**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 14:32:07 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.sporty.shoe.RestAssuredScripts.GetAllUsers.get\_all\_users

===============================================

Default test

Tests run: 1, Failures: 0, Skips: 0

===============================================

===============================================

Default suite

Total tests run: 1, Passes: 1, Failures: 0, Skips: 0

===============================================

* **ADD THE PRODUCT.**

Create a TestNG test with below REST API execution details -

POST CALL

URL: <http://localhost:9010/add-shoe?id=100&image=www.img.com&name=Nike&category=Sports&sizes=9,10&price=3000>

Service Type: POST

**Params:**

id=100

image=www.img.com

name=Nike

category=Sports

sizes=9,10

price=3000

**SOURCE CODE:**

package com.sporty.shoe.RestAssuredScripts;

import org.hamcrest.Matchers;

import org.testng.annotations.Test;

import io.restassured.RestAssured;

import io.restassured.http.ContentType;

public class PostNewShoe {

@Test

public void AddNewShoe() {

String Shoe\_Name = "Adidas";

RestAssured

.given()

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

.basePath("/add-shoe")

.queryParam("id", "106")

.queryParam("image", "img106.src")

.queryParam("name", Shoe\_Name)

.queryParam("category", "Fashion")

.queryParam("sizes", "11,12")

.queryParam("price", "8000")

.contentType(ContentType.JSON)

.when()

.post()

.then().statusCode(200)

.log()

.all()

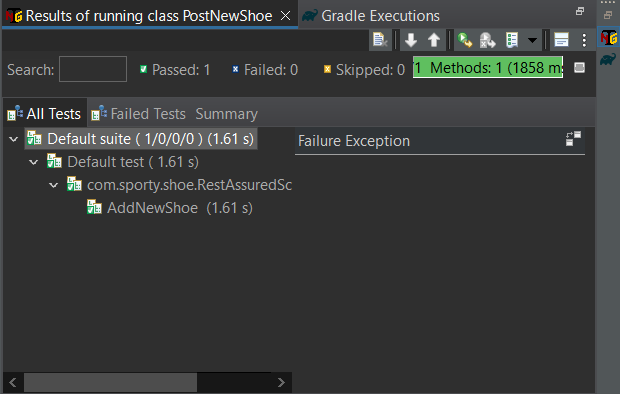
.body("code", Matchers.equalTo(101))

.body("message", Matchers.equalTo(Shoe\_Name +" Added Successfully."));

}

}

**TESTNG TEST WINDOW:**



**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

Set-Cookie: JSESSIONID=980A66E59C7F25ADD6577968BCDC4E3C; Path=/; HttpOnly

Content-Type: application/json

Transfer-Encoding: chunked

Date: Tue, 14 Nov 2023 14:24:30 GMT

Keep-Alive: timeout=60

Connection: keep-alive

{

"code": 101,

"message": "Adidas Added Successfully.",

"shoe": {

"id": 106,

"image": "img106.src",

"name": "Adidas",

"category": "Fashion",

"sizes": "11,12",

"price": 8000

}

}

PASSED: com.sporty.shoe.RestAssuredScripts.PostNewShoe.AddNewShoe

===============================================

Default test

Tests run: 1, Failures: 0, Skips: 0

===============================================

===============================================

Default suite

Total tests run: 1, Passes: 1, Failures: 0, Skips: 0

===============================================

* **UPDATE THE PRODUCT.**

Create a TestNG test with below REST API execution details -

PUT CALL

URL: <http://localhost:9010>/update-shoe?id=100&image=image.src&name=Nike Sports&category=Sports-Running&sizes=9,11&price=4000

Service Type: PUT

**Params:**

id=100

image=image.src

name=Nike Sports

category=Sports-Running

sizes=9,11

price=4000

**SOURCE CODE:**

package com.sporty.shoe.RestAssuredScripts;

import org.hamcrest.Matchers;

import org.testng.annotations.Test;

import io.restassured.RestAssured;

public class PutUpdateShoe {

@Test()

public void update\_a\_product()

{

String Shoe\_Name = "Adidas";

RestAssured.given()

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

.basePath("/update-shoe")

.queryParam("id","100")

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

.queryParam("name", Shoe\_Name)

.queryParam("category", "Running")

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

.queryParam("price", "2500")

.when()

.put()

.then()

.log().all()

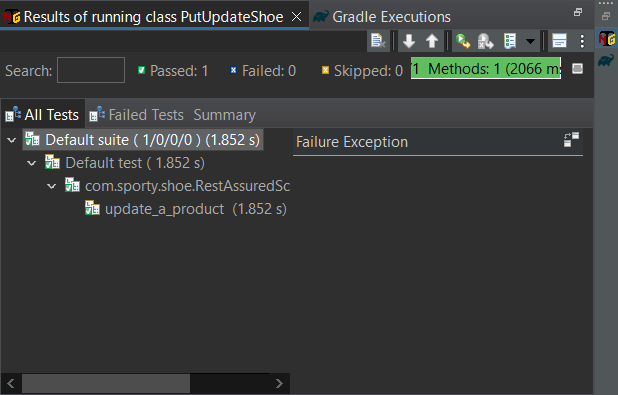
.body("code", Matchers.equalTo(101))

.body("message", Matchers.equalTo(Shoe\_Name +" Updated Successfully."));

}

}

**TESTNG TEST WINDOW:**



**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

Set-Cookie: JSESSIONID=07D98B2B0E8F460D09581678EA5DFF42; Path=/; HttpOnly

Content-Type: application/json

Transfer-Encoding: chunked

Date: Tue, 14 Nov 2023 14:19:03 GMT

Keep-Alive: timeout=60

Connection: keep-alive

{

"code": 101,

"message": "Adidas Updated Successfully.",

"shoe": {

"id": 100,

"image": "www.imge123.com",

"name": "Adidas",

"category": "Running",

"sizes": "5,6,7",

"price": 2500

}

}

PASSED: com.sporty.shoe.RestAssuredScripts.PutUpdateShoe.update\_a\_product

===============================================

Default test

Tests run: 1, Failures: 0, Skips: 0

===============================================

===============================================

Default suite

Total tests run: 1, Passes: 1, Failures: 0, Skips: 0

===============================================

* **DELETE THE PRODUCT.**

Create a TestNG test with below REST API execution details -

**DELETE CALL**

URL: <http://localhost:9010/delete-shoe?id=100>

Service Type: DELETE

**SOURCE CODE:**

package com.sporty.shoe.RestAssuredScripts;

import org.hamcrest.Matchers;

import org.testng.annotations.Test;

import io.restassured.RestAssured;

public class DeleteShoe {

@Test()

public void delete\_product()

{

String Shoe\_id = "100";

RestAssured.given()

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

.basePath("/delete-shoe")

.queryParam("id", Shoe\_id)

.when().delete()

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

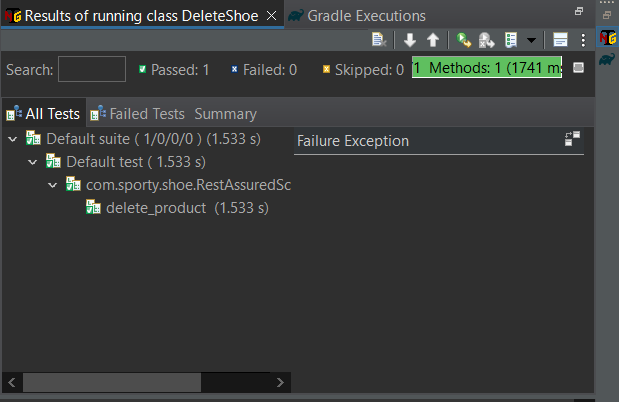
.body("code", Matchers.equalTo(101))

.body("message", Matchers.equalTo("Shoe with ID " + Shoe\_id + " Deleted Successfully."));

}

}

**TESTNG TEST WINDOW:**



**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 14:35:38 GMT

Keep-Alive: timeout=60

Connection: keep-alive

{

"code": 101,

"message": "Shoe with ID 100 Deleted Successfully."

}

PASSED: com.sporty.shoe.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

===============================================

1. **CREATE SELENIUM SCRIPTS USING TESTNG TO TEST ALL THE PAGES IN THE WEB APP THAT WILL AUTOMATE:**

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

**CREATE A MAVEN PROJECT:**

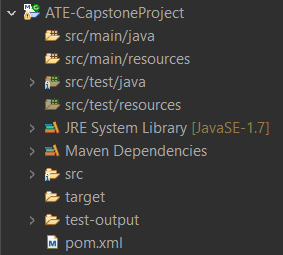
Go to File 🡪 New 🡪 Maven Project.

Enter the archetype as maven-archetype-quickstart to create a maven quickstart project and select the template with version 1.4.

Enter the Group Id and Artifact Id as **ATE-CapstoneProject**.

After completing this right click on the project and click on configure and convert to cucumber project.

This will be the basic structure of the cucumber project.



In the **pom.xml** file add the required dependencies like **Selenium Web Driver, TestNG, Cucumber, Rest Assured, Hamcrest** etc.

Save the POM file.

Now under the src/test/java folder in the project add a package by right clicking on the project 🡪 New 🡪 Package and enter the name for the package.

Now create a class under the package by right clicking on the package 🡪 New 🡪 Class and enter a name for the class.

We are create a POM (Page Object Model) Framework using the Page Factory class

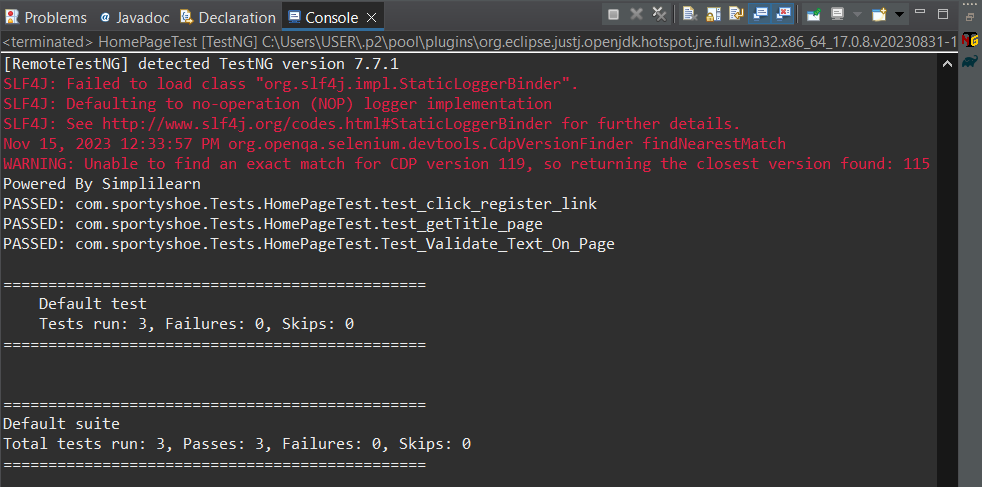
**HomePage**

Create a **HomePage** class and find the required xpath for the locators and enter the xpath using the @findby annotation.

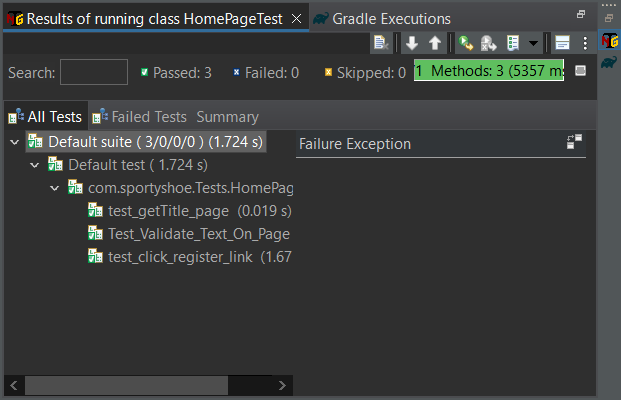
Now create a **HomePageTest** class in another package and this class extends the base class and import the HomePage class.

Validate the Selenium script with then necessary TestNG assertions and use @Test to test each method

**OUTPUT:**



**TESTNG TEST WINDOW**



**RegisterPage**

Create a **RegisterPage** class and find the required xpath for the locators and enter the xpath using the @findby annotation.

Now create a **RegisterPageTest** class in another package and this class extends the base class and import the HomePage class and RegisterPage class.

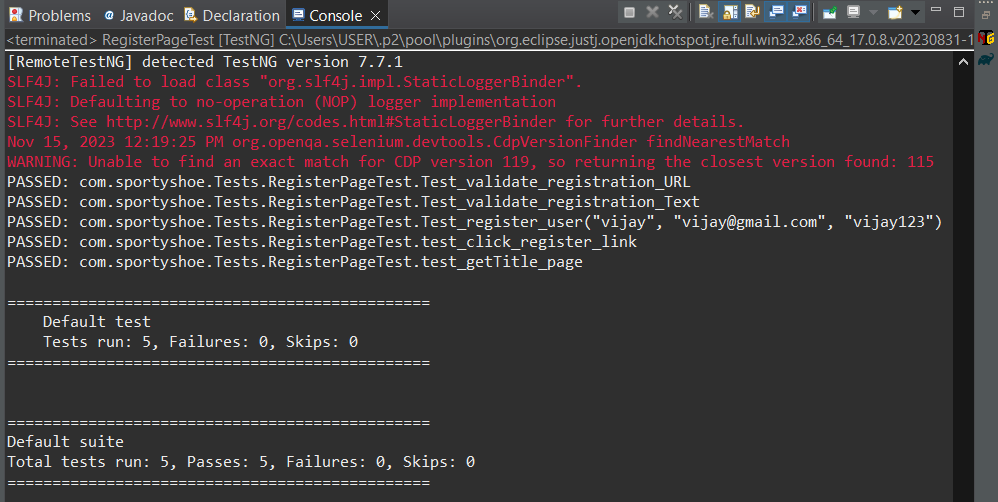
Validate the Selenium script with then necessary TestNG assertions and use @Test to test each method

For test\_register\_user rather than hardcoding the name, email and password a Xls\_DataProvider class is created where it is used for retrieving the data form Excel sheet

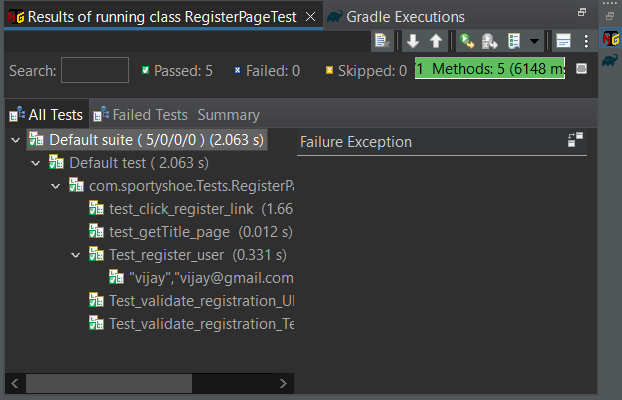
**LoginDetails.xlsx – Sheet1**

|  |  |  |
| --- | --- | --- |
| name | email | password |
| vijay | [vijay@gmail.com](mailto:vijay@gmail.com) | vijay123 |

**OUTPUT:**



**TESTNG TEST WINDOW:**



**LoginPage**

Create a **LoginPage** class and find the required xpath for the locators and enter the xpath using the @findby annotation.

Now create a **LoginPageTest** class in another package and this class extends the base class and import the HomePage class, RegisterPage class, LoginPage.

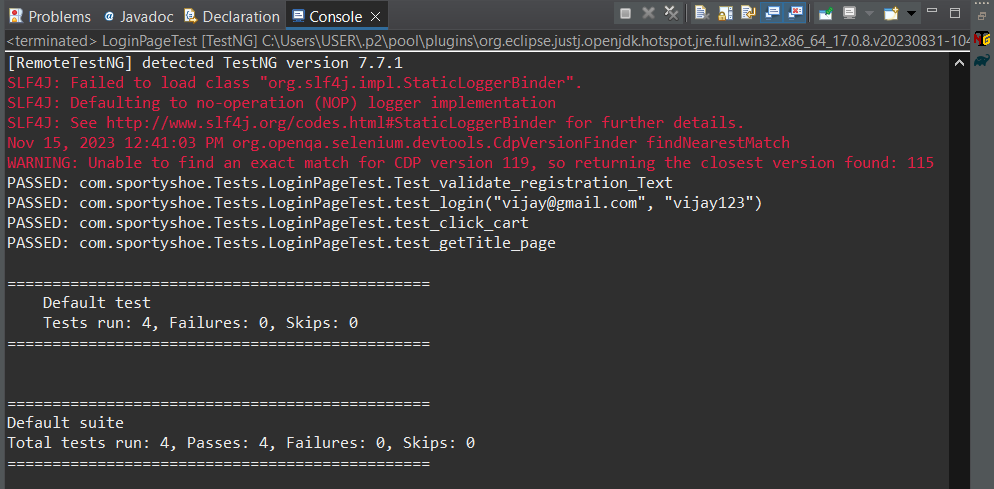
Validate the Selenium script with then necessary TestNG assertions and use @Test to test each method

For test\_register\_user rather than hardcoding the email and password a Xls\_DataProvider class is created where it is used for retrieving the data form Excel sheet

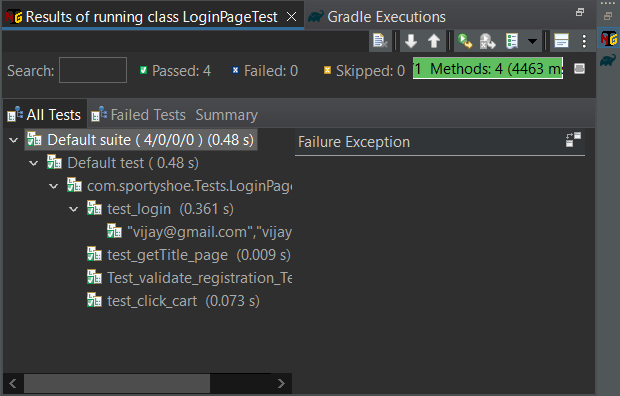
**LoginDetails.xlsx – Sheet2**

|  |  |
| --- | --- |
| email | password |
| [vijay@gmail.com](mailto:vijay@gmail.com) | vijay123 |

**OUTPUT:**



**TESTNG TEST WINDOW**



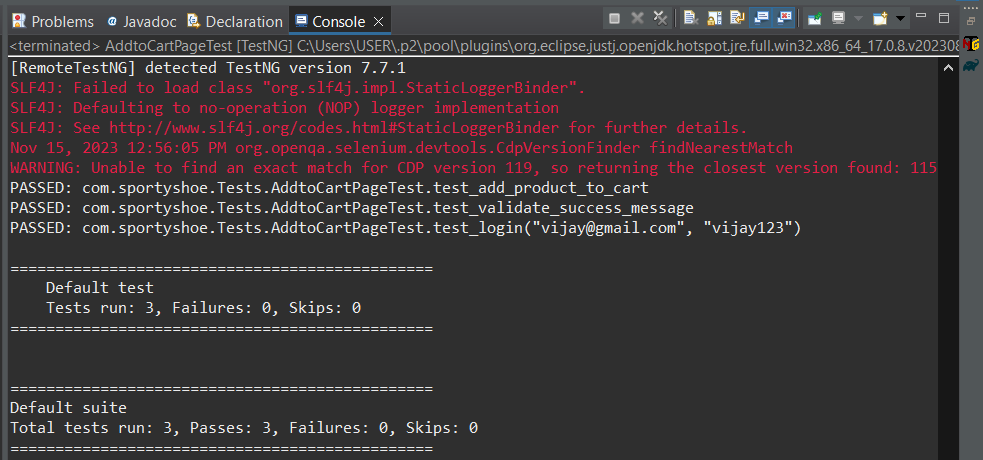
**AddtoCartPage**

Create a **AddtoCartPage** class and find the required xpath for the locators and enter the xpath using the @findby annotation.

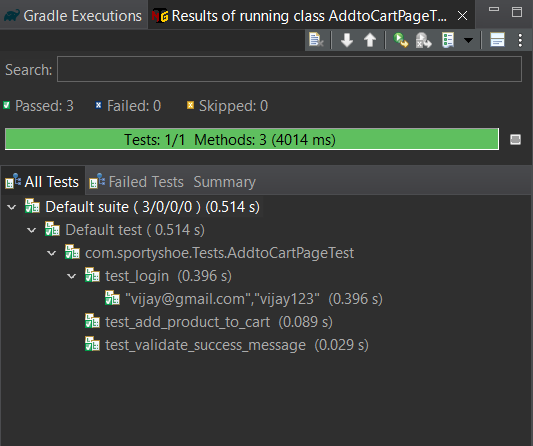
Now create a **AddtoCartTest** class in another package and this class extends the base class and import the OrderPage class, RegisterPage, LoginPage and AddtoCartPage.

Validate the Selenium script with then necessary TestNG assertions and use @Test to test each method

**OUTPUT:**



**TESTNG TEST WINDOW**



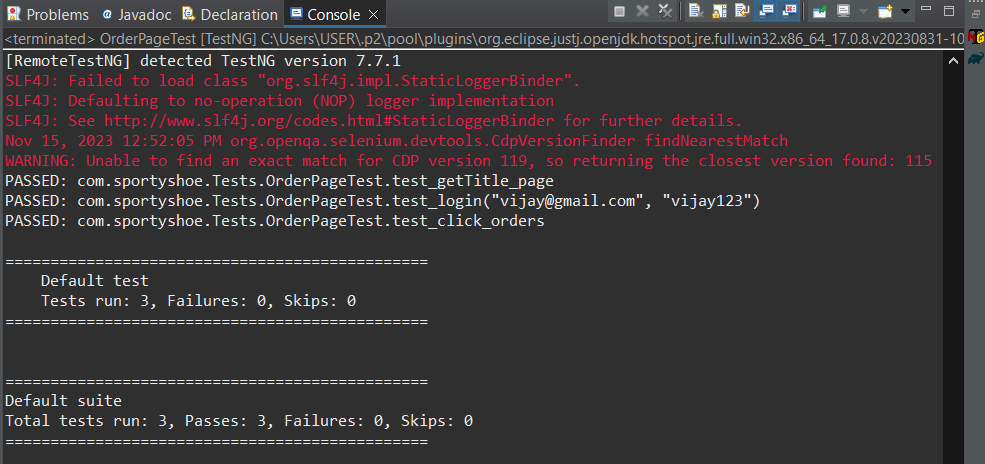
**OrderPage**

Create a **OrderPage** class and find the required xpath for the locators and enter the xpath using the @findby annotation.

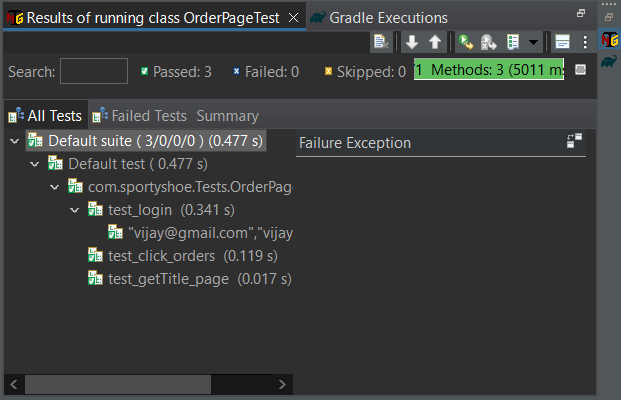
Now create a **OrderPageTest** class in another package and this class extends the base class and import the OrderPage class, RegisterPage, LoginPage and OrderPage.

Validate the Selenium script with then necessary TestNG assertions and use @Test to test each method

**OUTPUT:**



**TESTNG TEST WINDOW**



**TESTNG RUNNER FILE:**

Create a package runner and inside the package create a file and name the file as **Runner.xml**

<suite name="test1" verbose="1">

<listeners>

<listener class-name="com.sportyshoe.Tests.ItestListenerClass"></listener>

</listeners>

<test name="POMframework-SportyShoe-HomePageAndRegisterPage">

<classes>

<class name="com.sportyshoe.Tests.HomePageTest"></class>

<class name="com.sportyshoe.Tests.RegisterPageTest"></class>

</classes>

</test>

<test name="POMframework-SportyShoe-LoginPage">

<classes>

<class name="com.sportyshoe.Tests.LoginPageTest"></class>

</classes>

</test>

<test name="POMframework-SportyShoe-OrderPageAndAddtoCartPage">

<classes>

<class name="com.sportyshoe.Tests.OrderPageTest"></class>

<class name="com.sportyshoe.Tests.AddtoCartPageTest"></class>

</classes>

</test>

</suite>

Right click and run the file as TestNg suite.

**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.

Testing has started

Nov 15, 2023 2:19:34 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch

WARNING: Unable to find an exact match for CDP version 119, so returning the closest version found: 115

Nov 15, 2023 2:19:36 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch

WARNING: Unable to find an exact match for CDP version 119, so returning the closest version found: 115

Test method has been started

Test method is success

Test method has been started

Powered By Simplilearn

Test method is success

Test method has been started

Test method is success

Test method has been started

Test method is success

Test method has been started

Test method is success

Test method has been started

Test method is success

Test method has been started

Test method is success

Test method has been started

Test method is success

Testing has ended

Testing has started

Nov 15, 2023 2:19:42 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch

WARNING: Unable to find an exact match for CDP version 119, so returning the closest version found: 115

Test method has been started

Test method is success

Test method has been started

Test method is success

Test method has been started

Test method is success

Test method has been started

Test method is success

Testing has ended

Testing has started

Nov 15, 2023 2:19:44 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch

WARNING: Unable to find an exact match for CDP version 119, so returning the closest version found: 115

Nov 15, 2023 2:19:45 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch

WARNING: Unable to find an exact match for CDP version 119, so returning the closest version found: 115

Test method has been started

Test method is success

Test method has been started

Test method is success

Test method has been started

Test method is success

Test method has been started

Test method is success

Test method has been started

Test method is success

Test method has been started

Test method is success

Testing has ended

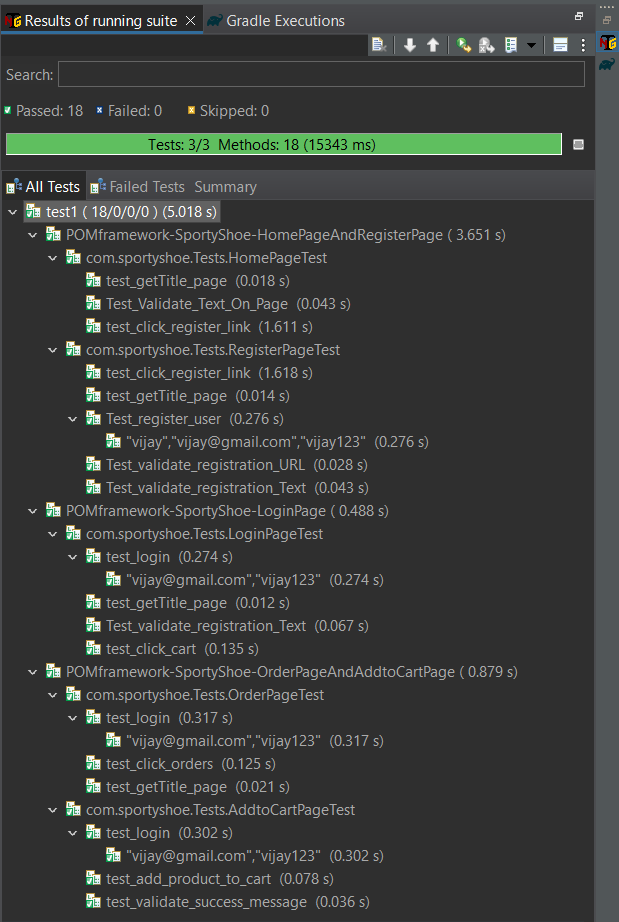
===============================================

test1

Total tests run: 18, Passes: 18, Failures: 0, Skips: 0

===============================================

**TESTNG TEST WINDOW:**

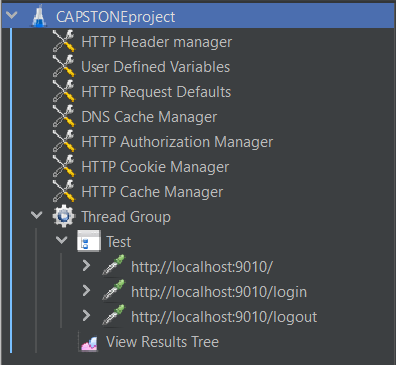


1. **CREATE JMETER SCRIPTS TO DO LOAD TESTING OF THE HOME PAGE AND THE LOGIN PAGE.**

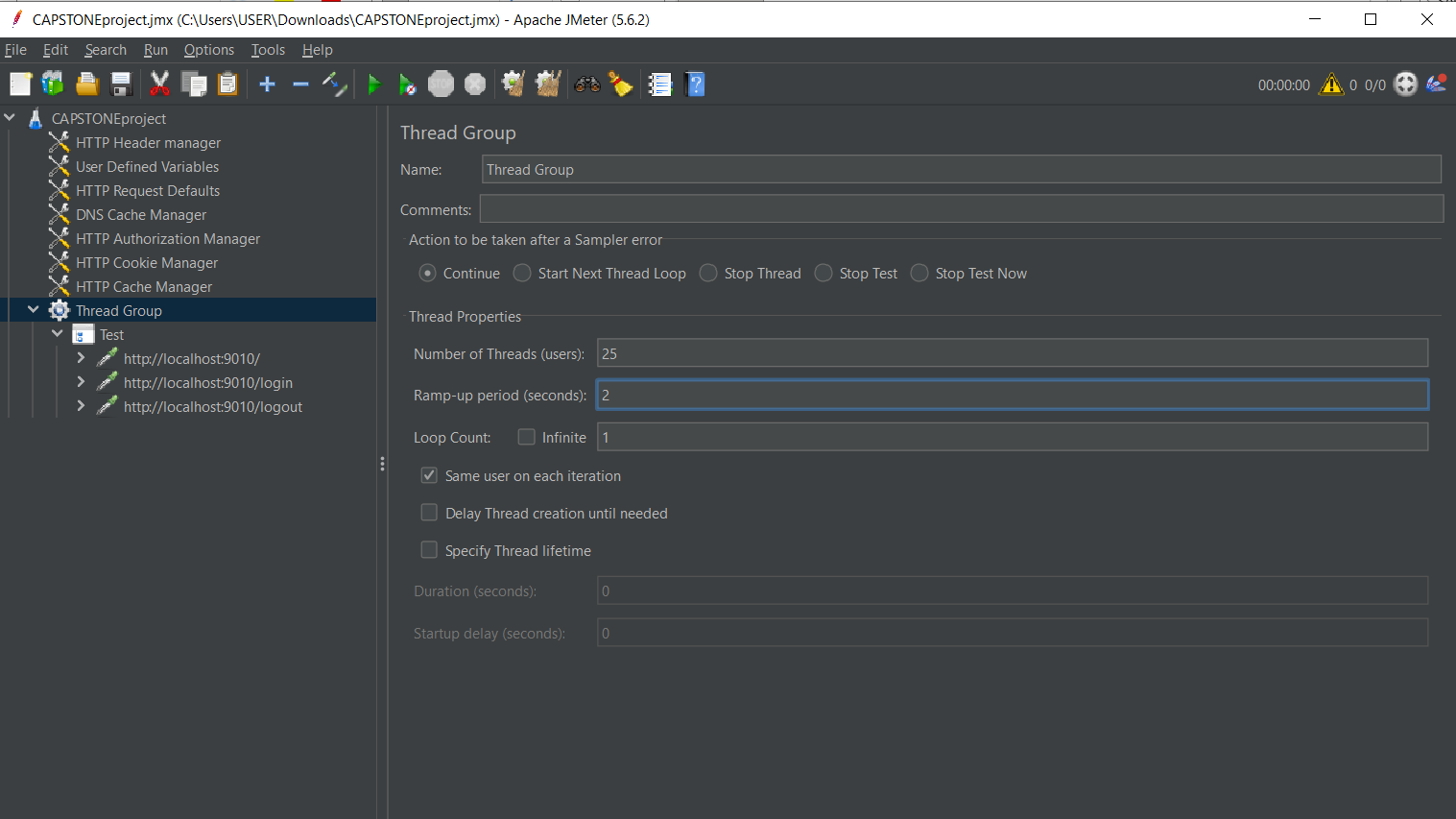
Open Apache Jmeter

Record the Script using the Blazemeter and click on the save button and click on the jmx file to download the recorded script

Open the downloaded on the Jmeter application.



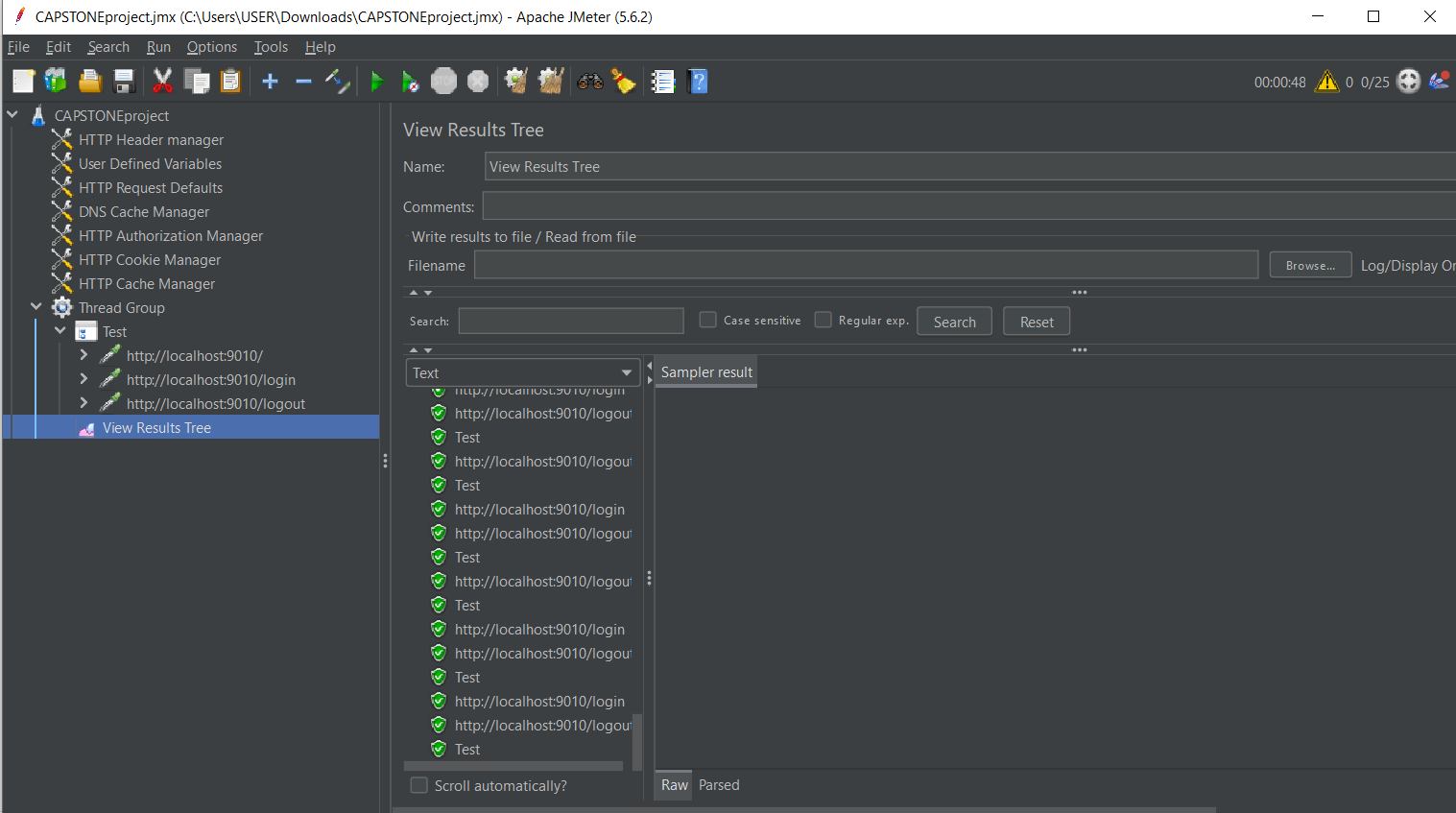
In the thread group give the number of threads and user as 25 to perform load Testing. And in the Ramp-up period as 2 for each second.



Right click on the thread group to add lister 🡪 View Result Tree to see the output.

And click on the Run button to start the load testing using Jmeter.

**OUTPUT:**



1. **SETUP CUCUMBER IN JAVA PROJECT AND WRITE FEATURE FILES USING GHERKIN TO TEST THE 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.

Create a package with the name feature and create feature file by right clicking the package 🡪 New 🡪 File

Enter the file name as **sportyshoe.feature**

Feature: Implementation of API of Sporty shoes using Cucumber

Scenario: Rest API Testing on Sporty shoes

Given User sends a get request to get list of users and validates status

Given User send a Post request to create a new shoe and validates status

Given User sends a get request to get list of shoes and validates status

Given User sends a Put request to update a shoe and validates status

Given User sends a delete request to delete shoe and validates status

Run the feature file by right clicking on the file and Run the file as Cucumber feature

The Feature file has the steps in the console

Add the required Rest Assured scripts for each steps and use the @Test to validate the test using TestNg

Add the step definition for the below steps,

* 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.

**SportyShoeStepDefinition.java**

package com.sportyshoe.steps;

import org.hamcrest.Matchers;

import org.testng.annotations.Test;

import io.cucumber.java.en.Given;

import io.restassured.RestAssured;

import io.restassured.http.ContentType;

public class SportyShoeStepDefinition {

*@Test*

*@Given*("User sends a get request to get list of users and validates status")

public void user\_sends\_a\_get\_request\_to\_get\_list\_of\_users\_and\_validates\_status() {

RestAssured.*given*()

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

.basePath("/get-users")

.contentType(*ContentType*.***JSON***)

.when()

.get()

.then()

.statusCode(200)

.log()

.all()

.body("code", Matchers.*equalTo*(101))

.body("message", Matchers.*equalTo*("3 Users Fetched Successfully."));

}

*@Test*

*@Given*("User send a Post request to create a new shoe and validates status")

public void user\_send\_a\_post\_request\_to\_create\_a\_new\_shoe\_and\_validates\_status() {

String Shoe\_Name = "Adidas";

RestAssured

.*given*()

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

.basePath("/add-shoe")

.queryParam("id", "106")

.queryParam("image", "img106.src")

.queryParam("name", Shoe\_Name)

.queryParam("category", "Fashion")

.queryParam("sizes", "11,12")

.queryParam("price", "8000")

.contentType(*ContentType*.***JSON***)

.when()

.post()

.then().statusCode(200)

.log()

.all()

.body("code", Matchers.*equalTo*(101))

.body("message", Matchers.*equalTo*(Shoe\_Name +" Added Successfully."));

}

*@Test*

*@Given*("User sends a get request to get list of shoes and validates status")

public void user\_sends\_a\_get\_request\_to\_get\_list\_of\_shoes\_and\_validates\_status() {

RestAssured.*given*()

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

.basePath("/get-shoes")

.contentType(*ContentType*.***JSON***)

.when()

.get()

.then().statusCode(200)

.log()

.all()

.body("code", Matchers.*equalTo*(101));

}

*@Test*

*@Given*("User sends a Put request to update a shoe and validates status")

public void user\_sends\_a\_put\_request\_to\_update\_a\_shoe\_and\_validates\_status() {

String Shoe\_Name = "Adidas";

RestAssured.*given*()

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

.basePath("/update-shoe")

.queryParam("id","100")

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

.queryParam("name", Shoe\_Name)

.queryParam("category", "Running")

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

.queryParam("price", "2500")

.when()

.put()

.then()

.log().all()

.body("code", Matchers.*equalTo*(101))

.body("message", Matchers.*equalTo*(Shoe\_Name +" Updated Successfully."));

}

*@Test*

*@Given*("User sends a delete request to delete shoe and validates status")

public void user\_sends\_a\_delete\_request\_to\_delete\_shoe\_and\_validates\_status() {

String Shoe\_id = "100";

RestAssured.*given*()

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

.basePath("/delete-shoe")

.queryParam("id", Shoe\_id)

.when().delete()

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

.body("code", Matchers.*equalTo*(101))

.body("message", Matchers.*equalTo*("Shoe with ID " + Shoe\_id + " Deleted Successfully."));

}

}

**OUTPUT:**

Nov 15, 2023 2:50:09 PM cucumber.api.cli.Main run

WARNING: You are using deprecated Main class. Please use io.cucumber.core.cli.Main

Scenario: Rest API Testing on Sporty shoes # src/test/java/com/sportyshoe/features/sportyshoe.feature:3

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: Wed, 15 Nov 2023 09:20:11 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"

}

]

}

Given User sends a get request to get list of users and validates status # com.sportyshoe.steps.SportyShoeStepDefinition.user\_sends\_a\_get\_request\_to\_get\_list\_of\_users\_and\_validates\_status()

HTTP/1.1 200

Set-Cookie: JSESSIONID=EA52BDC31F051B7F27EE2FEA4CA6AD78; Path=/; HttpOnly

Content-Type: application/json

Transfer-Encoding: chunked

Date: Wed, 15 Nov 2023 09:20:11 GMT

Keep-Alive: timeout=60

Connection: keep-alive

{

"code": 101,

"message": "Adidas Added Successfully.",

"shoe": {

"id": 106,

"image": "img106.src",

"name": "Adidas",

"category": "Fashion",

"sizes": "11,12",

"price": 8000

}

}

Given User send a Post request to create a new shoe and validates status # com.sportyshoe.steps.SportyShoeStepDefinition.user\_send\_a\_post\_request\_to\_create\_a\_new\_shoe\_and\_validates\_status()

HTTP/1.1 200

Content-Type: application/json

Transfer-Encoding: chunked

Date: Wed, 15 Nov 2023 09:20:11 GMT

Keep-Alive: timeout=60

Connection: keep-alive

{

"code": 101,

"message": "13 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",

"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

},

{

"id": 106,

"image": "img106.src",

"name": "Adidas",

"category": "Fashion",

"sizes": "11,12",

"price": 8000

}

]

}

Given User sends a get request to get list of shoes and validates status # com.sportyshoe.steps.SportyShoeStepDefinition.user\_sends\_a\_get\_request\_to\_get\_list\_of\_shoes\_and\_validates\_status()

HTTP/1.1 200

Set-Cookie: JSESSIONID=8A2D366F6F5545FE8CAB34AE38784EE7; Path=/; HttpOnly

Content-Type: application/json

Transfer-Encoding: chunked

Date: Wed, 15 Nov 2023 09:20:11 GMT

Keep-Alive: timeout=60

Connection: keep-alive

{

"code": 101,

"message": "Adidas Updated Successfully.",

"shoe": {

"id": 100,

"image": "www.imge123.com",

"name": "Adidas",

"category": "Running",

"sizes": "5,6,7",

"price": 2500

}

}

Given User sends a Put request to update a shoe and validates status # com.sportyshoe.steps.SportyShoeStepDefinition.user\_sends\_a\_put\_request\_to\_update\_a\_shoe\_and\_validates\_status()

HTTP/1.1 200

Content-Type: application/json

Transfer-Encoding: chunked

Date: Wed, 15 Nov 2023 09:20:12 GMT

Keep-Alive: timeout=60

Connection: keep-alive

{

"code": 101,

"message": "Shoe with ID 100 Deleted Successfully."

}

Given User sends a delete request to delete shoe and validates status # com.sportyshoe.steps.SportyShoeStepDefinition.user\_sends\_a\_delete\_request\_to\_delete\_shoe\_and\_validates\_status()

1 Scenarios (1 passed)

5 Steps (5 passed)

0m2.833s

┌───────────────────────────────────────────────────────────────────────────────────┐

│ Share your Cucumber Report with your team at https://reports.cucumber.io │

│ Activate publishing with one of the following: │

│ │

│ src/test/resources/cucumber.properties: cucumber.publish.enabled=true │

│ src/test/resources/junit-platform.properties: cucumber.publish.enabled=true │

│ Environment variable: CUCUMBER\_PUBLISH\_ENABLED=true │

│ JUnit: @CucumberOptions(publish = true) │

│ │

│ More information at https://cucumber.io/docs/cucumber/environment-variables/ │

│ │

│ Disable this message with one of the following: │

│ │

│ src/test/resources/cucumber.properties: cucumber.publish.quiet=true │

│ src/test/resources/junit-platform.properties: cucumber.publish.quiet=true │

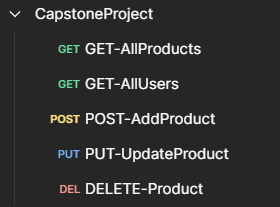
└───────────────────────────────────────────────────────────────────────────────────┘

1. **CREATE POSTMAN SCRIPTS TO TEST THE FOLLOWING API ENDPOINTS:**

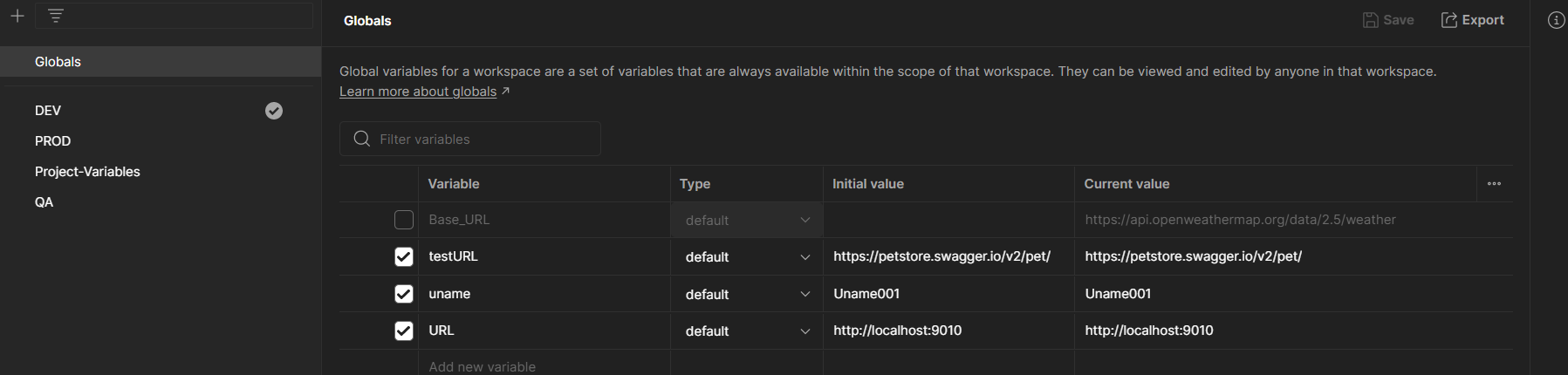
* Retrieve the list of all products in the store.
* Retrieve the list of all registered users.
* Add the product.
* Delete the product.
* Update the product.

**RETRIEVE THE LIST OF ALL PRODUCTS IN THE STORE.**

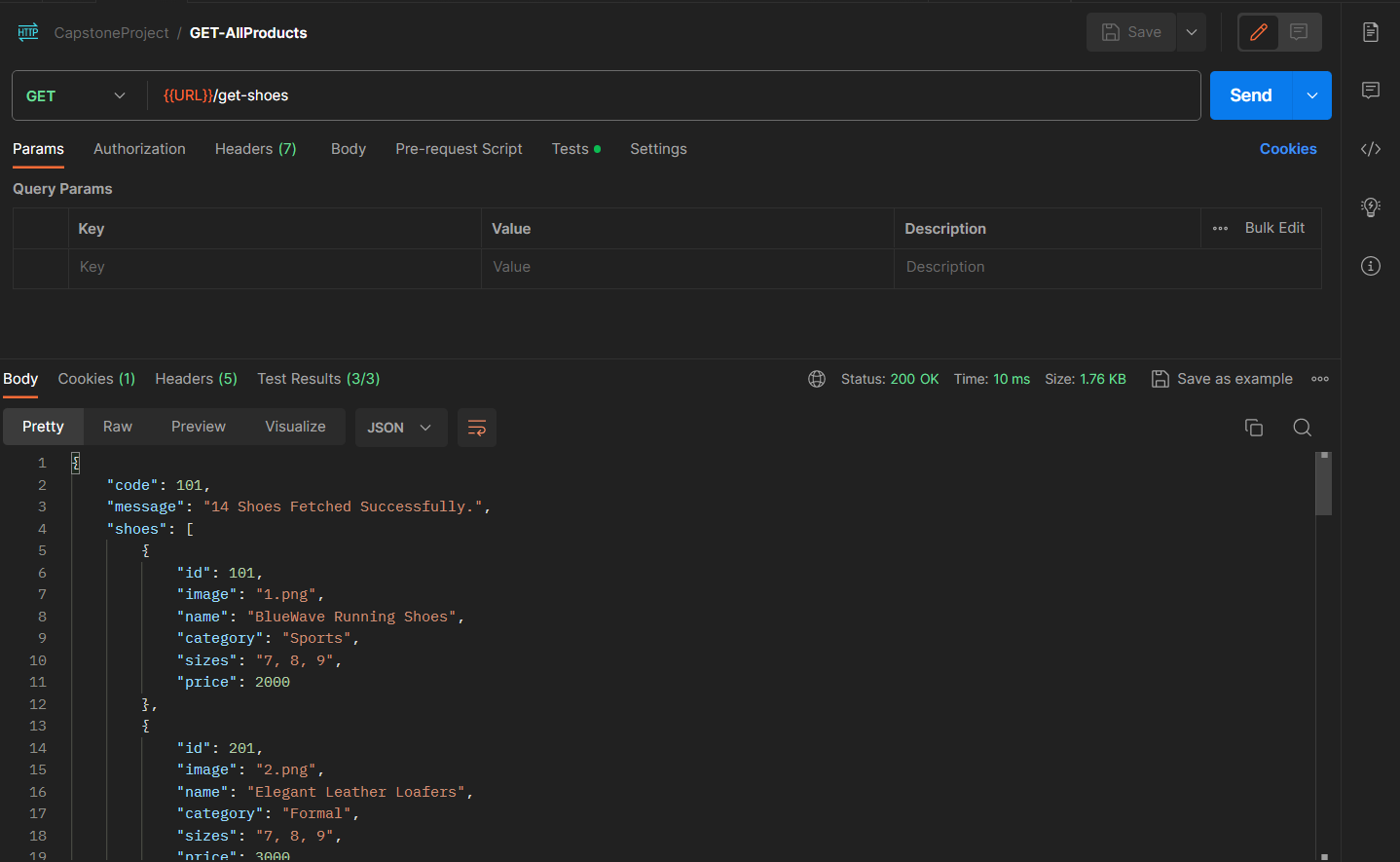
* Open Postman Application
* Create a new collection called **CapstoneProject** and arrange three end-to-end services POST



* Create a global variable of this URL where **URL** is the variable name and its value is <http://localhost:9010>
* Use this variable while hitting the URL in Postman.

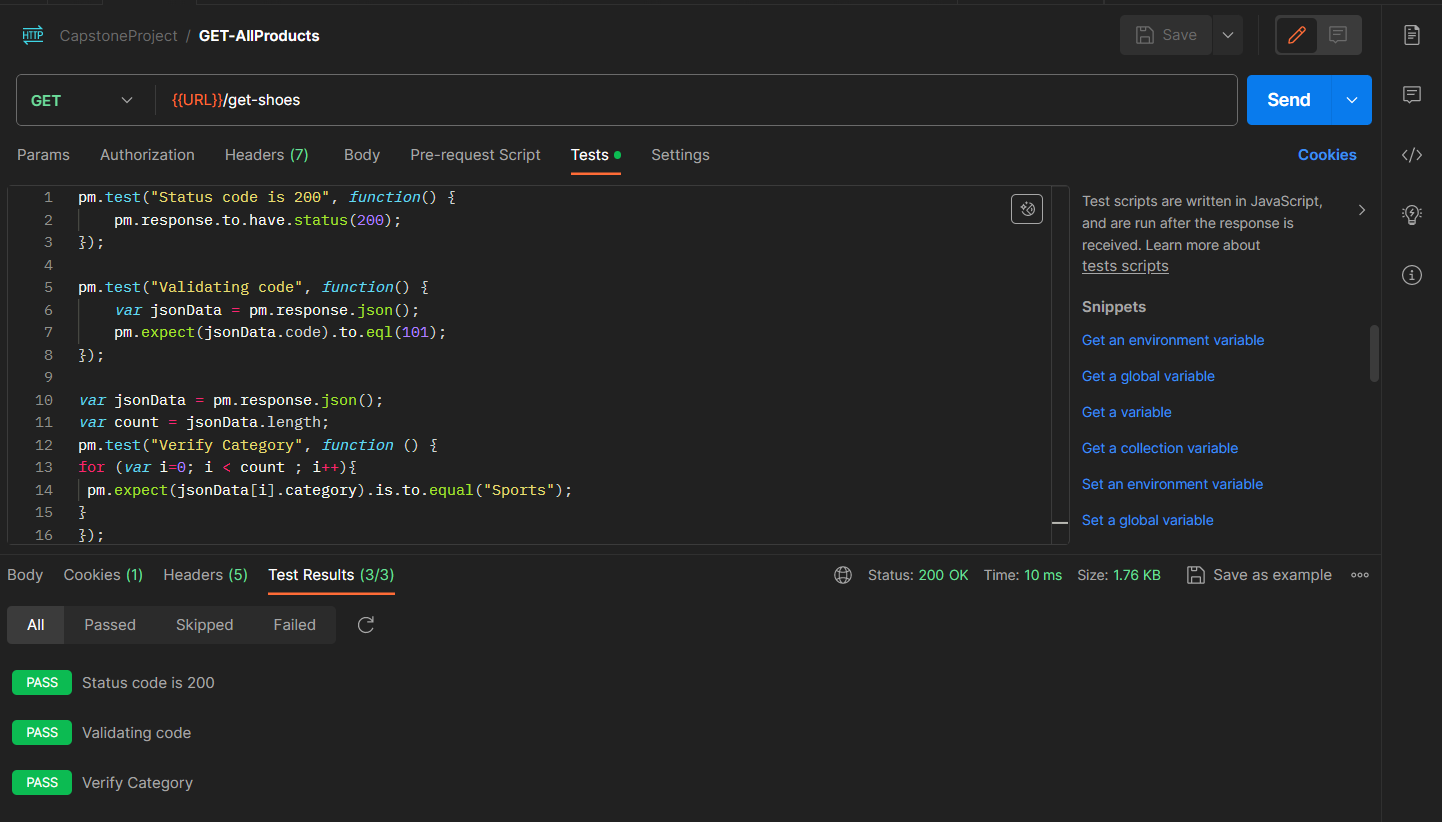


* In the collection, add a new GET service request with following details:
* URL: <http://localhost:9010/get-shoes>
* Click save button and click on the send button to send the request.



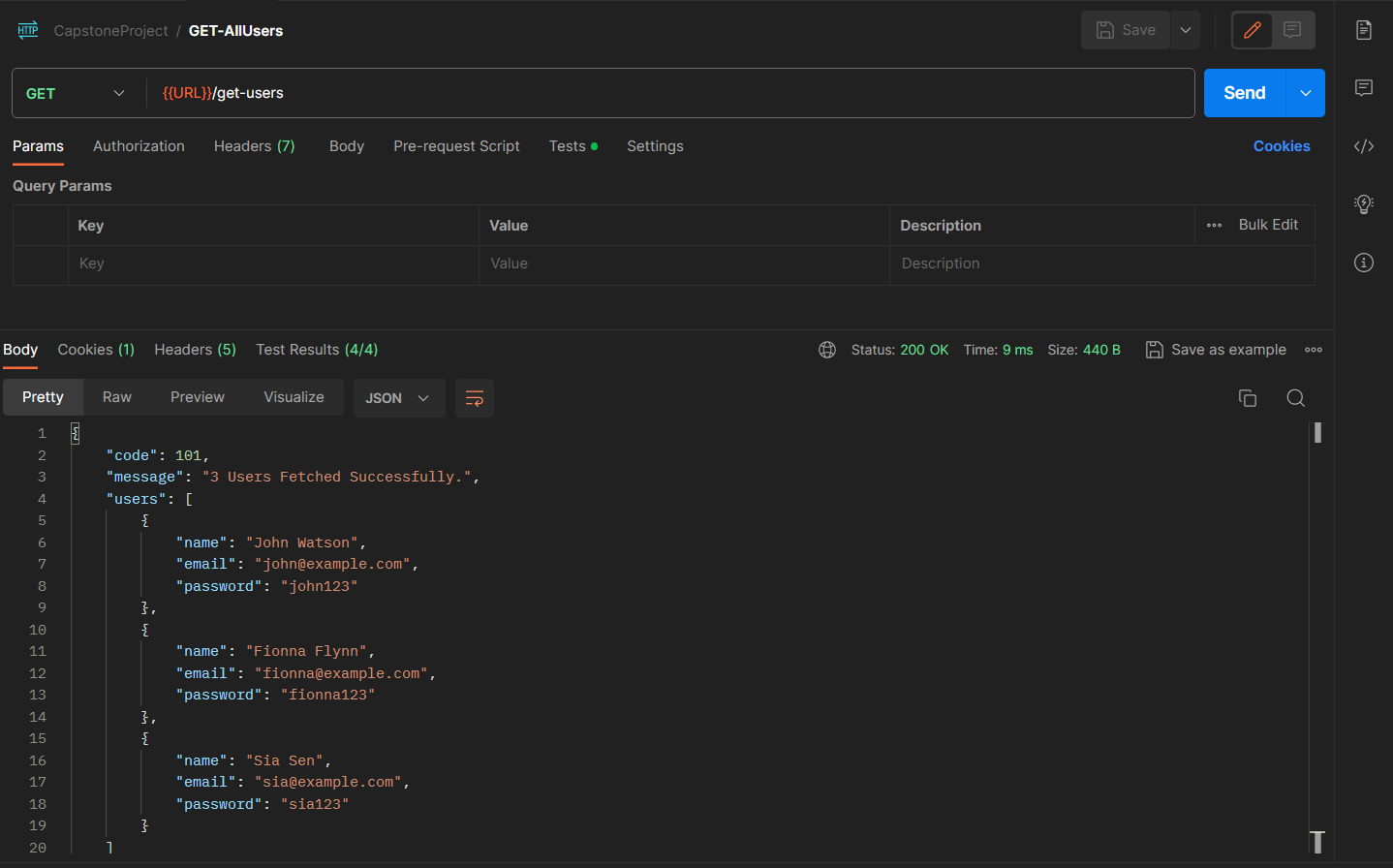
**TESTS:**

* In the Postman Tests section, validate for this GET call. Response status code should be 200.
* In the Postman Tests section validate for this GET call. Response body should contain code as **101**.
* In the Postman Tests section validate for this GET call. Response body should contain category as **“Sports”**.



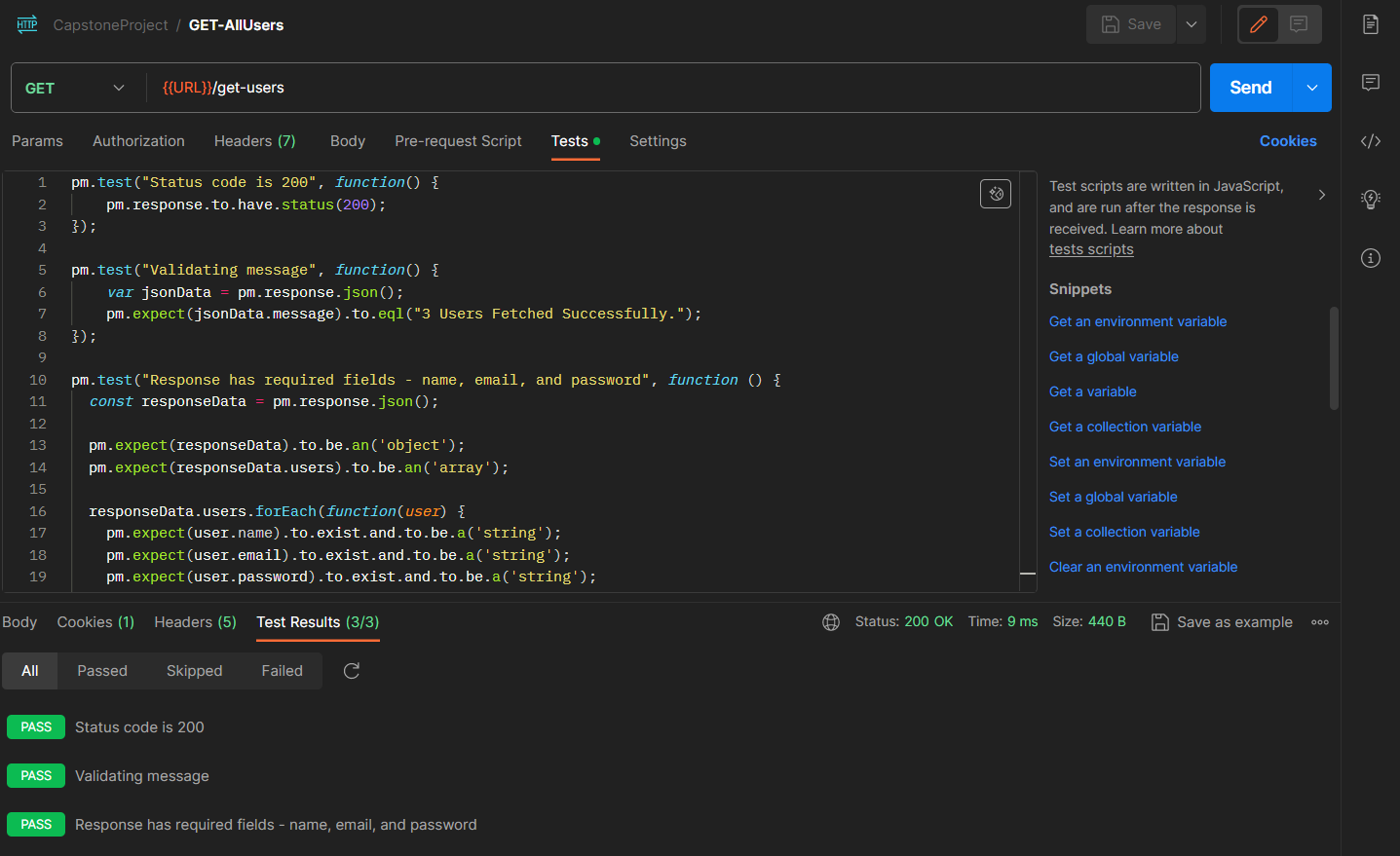
**RETRIEVE THE LIST OF ALL REGISTERED USERS.**

* In the collection, add a new GET service request with following details:
* URL: <http://localhost:9010/get-users>
* Click save button and click on the send button to send the request.



**TESTS:**

* In the Postman Tests section, validate for this GET call. Response status code should be 200.
* In the Postman Tests section validate for this GET call. Response body should contain message as **“3 Users Fetched Successfully”**.
* In the Postman Tests section validate for this GET call. Response body should contain name, email and password.



**ADD THE PRODUCT.**

* In the collection, add a new POST service request with following details:
* URL: <http://localhost:9010/add-shoe>
* Click save button and click on the send button to send the request.
* POST Call request Params:

Id: 100

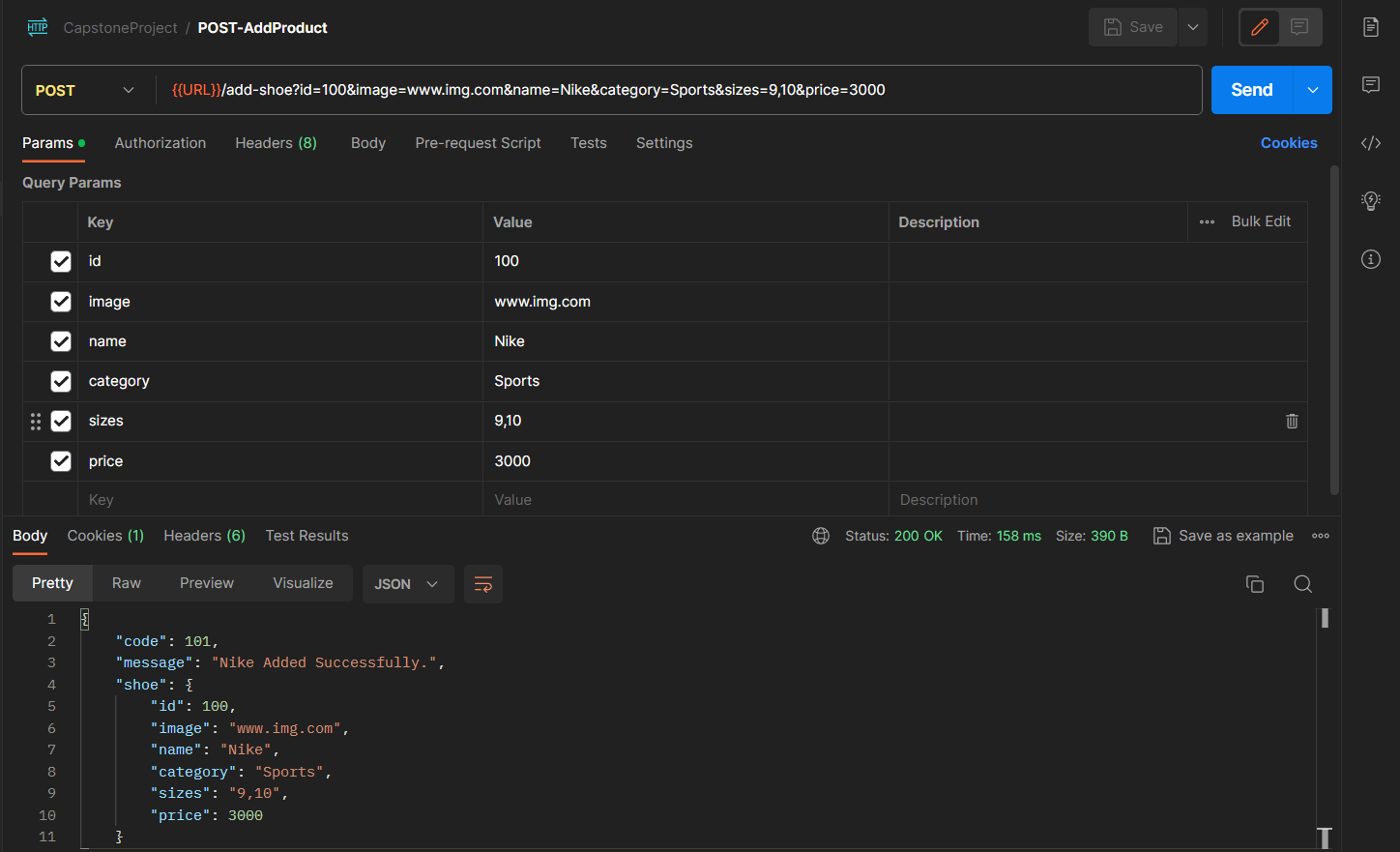
Image: www.img.com

Name: Nike

Category: Sports

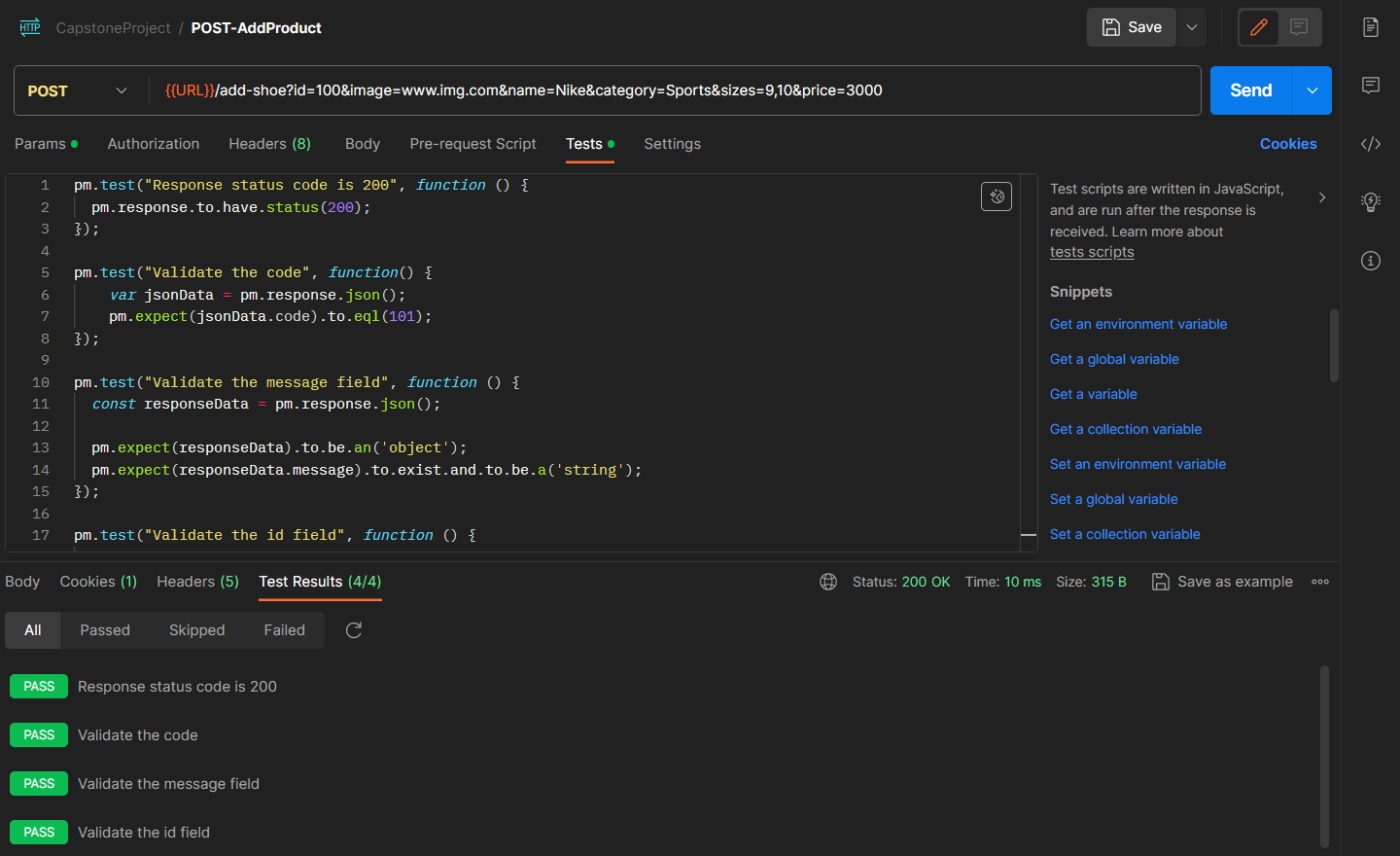
Size: 9,10

Price: 3000



**TESTS:**

* In the Postman Tests section, validate for this POST call. Response status code should be 200.
* In the Postman Tests section validate for this POST call. Response body should contain code as **101**.
* In the Postman Tests section validate for this POST call. Response body should contain message field as String
* In the Postman Tests section validate for this POST call. Response body should contain id field as Integer



**UPDATE THE PRODUCT.**

* In the collection, add a new PUT service request with following details:
* URL: <http://localhost:9010/update-shoe>
* Click save button and click on the send button to send the request.
* PUT Call request Params:

Id: 100

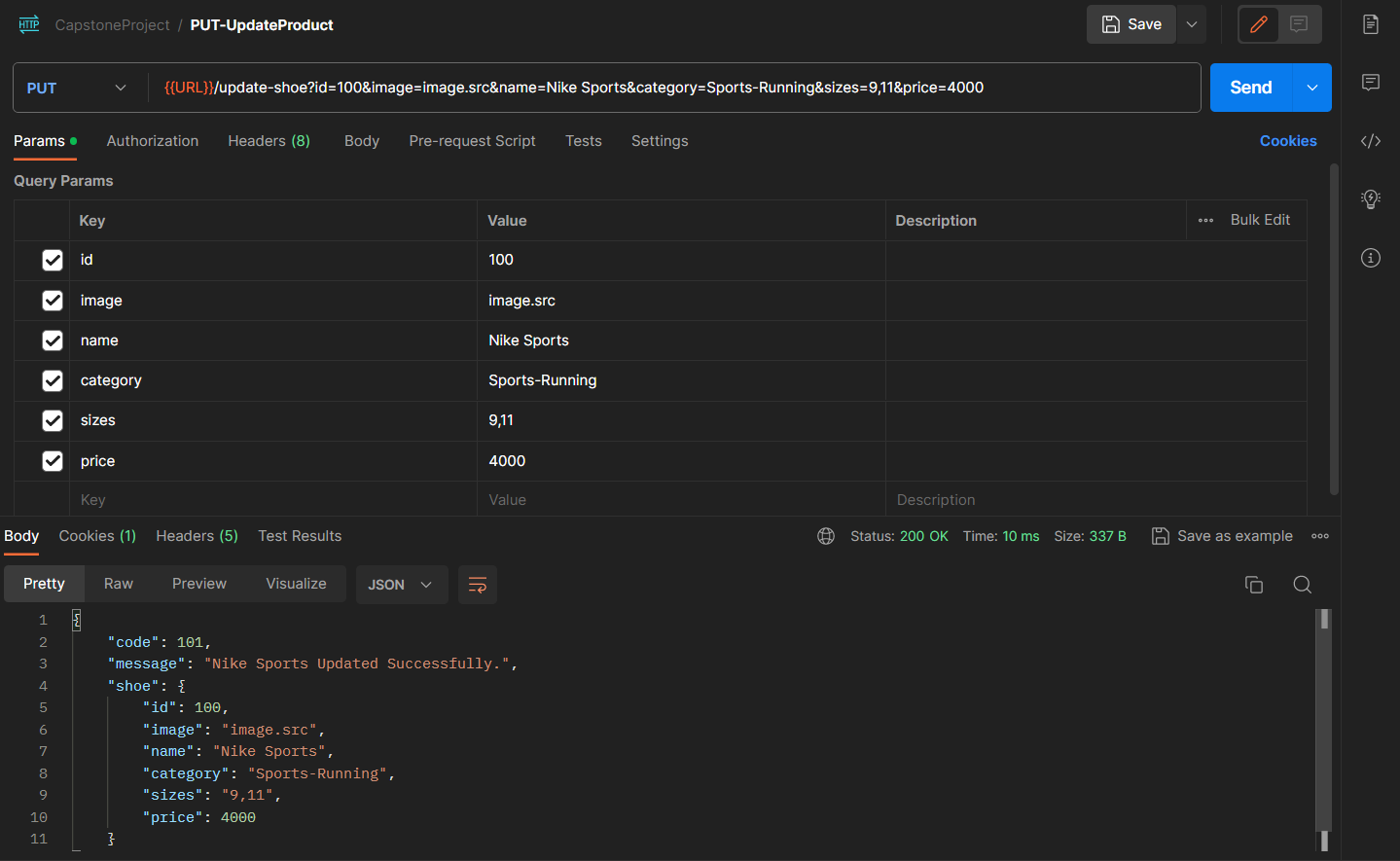
Image: img.src

Name: Nike Sports

Category: Sports+Running

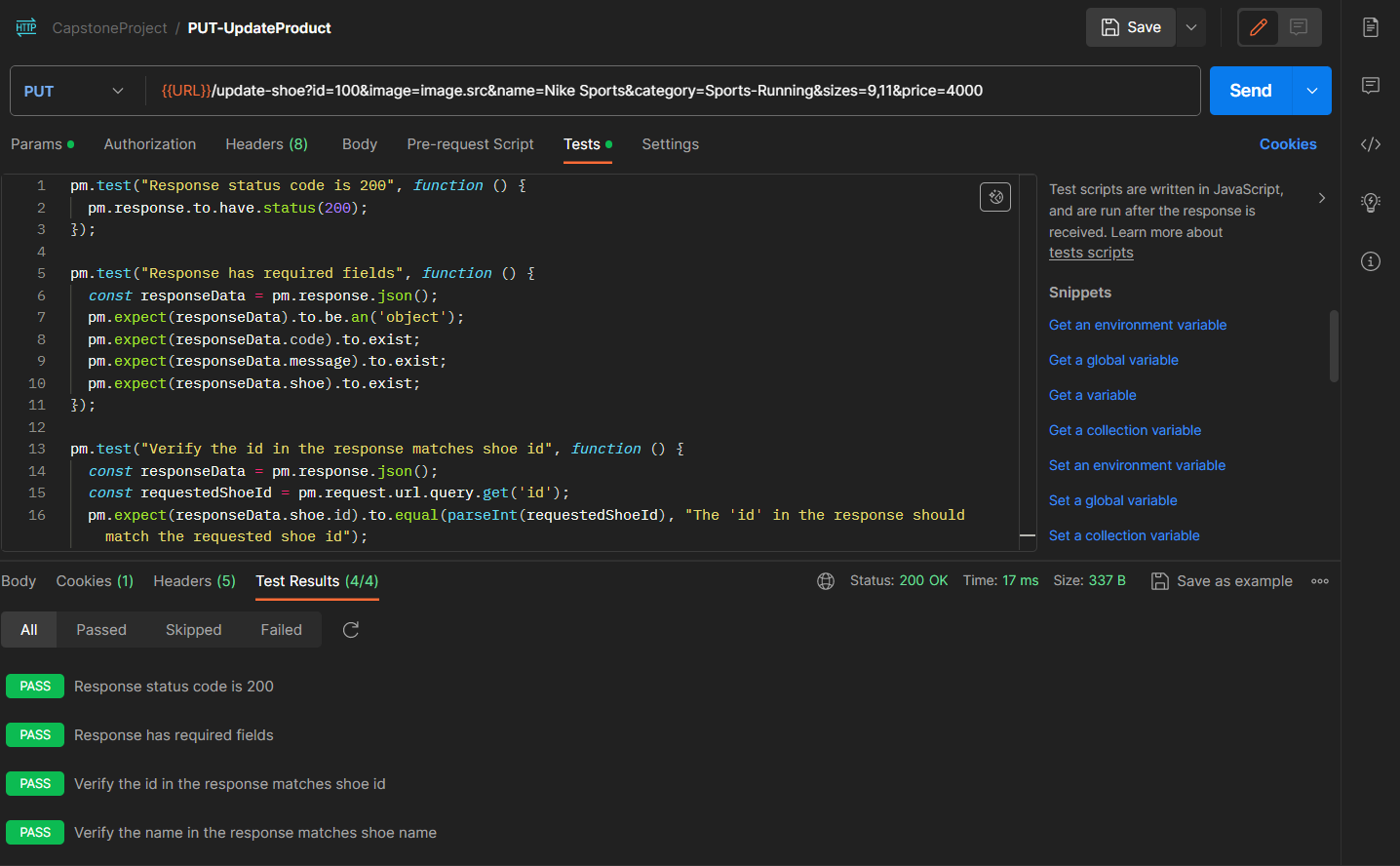
Size: 9,11

Price: 4000



**TESTS:**

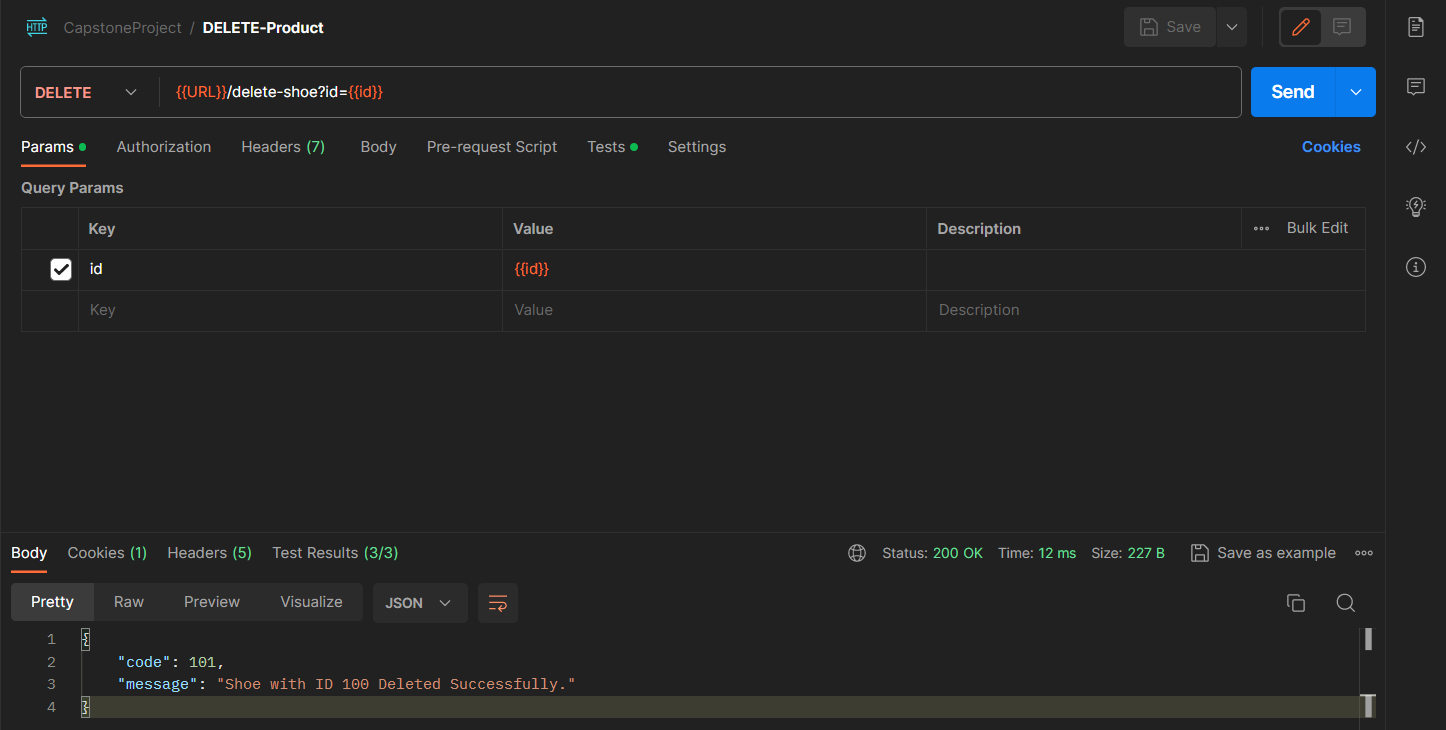
* In the Postman Tests section, validate for this PUT call. Response status code should be 200.
* In the Postman Tests section validate for this PUT call. Response body should contain required fields like code, message, shoe.
* In the Postman Tests section validate for this PUT call. Response body should contain id in response should be same as the given id.
* In the Postman Tests section validate for this PUT call. Response body should contain name in response should be same as the given name.



**DELETE THE PRODUCT.**

* In the collection, add a new PUT service request with following details:
* URL: <http://localhost:9010/delete-shoe>
* Click save button and click on the send button to send the request.
* PUT Call request Params:

Id: 100



**TEST:**

* In the Postman Tests section, validate for this GET call. Response status code should be 200.
* In the Postman Tests section validate for this GET call. Response body should contain code as **101**.
* In the Postman Tests section validate for this GET call. Response body should contain message as **“Shoe with ID 100 Deleted Successfully”**.

