

# Capstone Project

**Creating a Testing Framework for Spotify Shoes Website**

**GITHUB URL : <https://github.com/naveenmpc/JavaCodes1/tree/master/src>**

**GITHUB – Capstone url - <https://github.com/naveenmpc/CAPSTONE-PROJECT>**

**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.

## **1.Rest Assure**

1.1 Getting all Shoes using restassure ..having base uri and path , we can fetch the data.

```

1 package capstone.Sporty.shoes.RestAssureScripts;
2
3 import org.testng.annotations.Test;
4
5 import io.restassured.RestAssured;
6
7 Run All
7 public class GetAllAShoes {
8
9     @Test
10    Run | Debug
11    public void get_all_shoes()
12
13    {
14        RestAssured.given()
15        .baseUri("http://localhost:9010")
16        .basePath("/get-shoes")
17        .when()
18        .get()
19        .then()
20        .statusCode(200)
21        .log()
22        .all();
23    }
24
25    @Test
26    Run | Debug
27    public void get_all_users()
28
29    {
30        RestAssured.given()
31        .baseUri("http://localhost:9010")
32        .basePath("/get-users")
33        .when()
34        .get()
35        .then()
36        .log()
37        .all();
38    }
39
40 }
41
42 }
43
44 }

```

## Fetched 12 Shoes Successfully

The screenshot shows the Eclipse IDE interface with the following details:

- Left Panel (Code View):** Displays the Java code for the `GetAllAShoes` and `GetAllUsers` classes.
- Right Panel (Output View):** Shows the terminal output of the test execution. It includes the command run, the URL, and the response body, which is a JSON array of shoe objects.

```

1 package capstone.Sporty.shoes.RestAssureScripts;
2
3 import org.testng.annotations.Test;
4
5 import io.restassured.RestAssured;
6
7 Run All
7 public class GetAllAShoes {
8
9     @Test
10    Run | Debug
11    public void get_all_shoes()
12
13    {
14        RestAssured.given()
15        .baseUri("http://localhost:9010")
16        .basePath("/get-shoes")
17        .when()
18        .get()
19        .then()
20        .statusCode(200)
21        .log()
22        .all();
23    }
24
25    @Test
26    Run | Debug
27    public void get_all_users()
28
29    {
30        RestAssured.given()
31        .baseUri("http://localhost:9010")
32        .basePath("/get-users")
33        .when()
34        .get()
35        .then()
36        .log()
37        .all();
38    }
39
40 }
41
42 }
43
44 }

```

**Terminal Output (Right Panel):**

```

<terminated> GetAllAShoes [TestNG] C:\Users\mnave\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further
HTTP/1.1 200
Content-Type: application/json
Transfer-Encoding: chunked
Date: Wed, 15 Nov 2023 13:24:32 GMT
Keep-Alive: timeout=60
Connection: keep-alive

{
  "code": 101,
  "message": "12 Shoes Fetched Successfully.",
  "shoes": [
    {
      "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
    }
  ]
}

```

## 1.2 Updating and Adding the data to SpotyShoes using Put and Post methods in rest assure.

Updation done via Query Parameters , the name and price has been updated and added.

```

eclipse-workspace - ATE_Capstone-Project/src/test/java/capstone/spoty/shoes/RestAssureScripts/Post_Put_Methods.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer   Run All   Delete.meth... GetAllShoe... Post_Put_M...
src/main/java capstone.spoty.shoes.RestAssureScripts;
import org.testng.annotations.Test;
public class Post_Put_Methods {
    @Test(priority='1')
    Run | Debug
    public void add_new_product() {
        RestAssured.given()
            .baseUri("http://localhost:9010")
            .basePath("/add-shoe")
            .queryParam("id", "101")
            .queryParam("image", "www.imgur.com")
            .queryParam("name", "Bata")
            .queryParam("category", "Running")
            .queryParam("sizes", "5,6,7,8")
            .queryParam("price", "3000")
        .when()
            .post()
        .then()
            .log().all();
    }
    @Test(priority='2')
    Run | Debug
    public void update_a_product() {
        RestAssured.given()
            .baseUri("http://localhost:9010")
            .basePath("/update-shoe")
            .queryParam("id", "101")
            .queryParam("image", "www.imgur123.com")
            .queryParam("name", "Reebok")
            .queryParam("category", "Running")
            .queryParam("sizes", "5,6,7,8")
            .queryParam("price", "2500")
        .when()
            .put()
        .then()
            .log().all();
    }
}

```

The screenshot shows the Eclipse IDE interface with the Java code for the `Post_Put_Methods` class. The code uses RestAssured to make HTTP requests to a local server at port 9010. It includes two test methods: `add_new_product` and `update_a_product`. The `add_new_product` method adds a new product with ID 101, name "Bata", category "Running", sizes "5,6,7,8", and price 3000. The `update_a_product` method updates an existing product with ID 101, name "Reebok", category "Running", sizes "5,6,7,8", and price 2500. The right side of the screen shows the terminal output of the test execution, which includes the API responses and the test results.

## 1.3 Deleting the data using ID = 101

```

Delete.meth... GetAllShoe... Post_Put_M...
package capstone.spoty.shoes.RestAssureScripts;
import org.testng.annotations.Test;
import io.restassured.RestAssured;
Run All
public class Delete_method {
    @Test(priority='1')
    Run | Debug
    public void delete_product() {
        RestAssured.given()
            .baseUri("http://localhost:9010")
            .basePath("/delete-shoe")
            .queryParam("id", "101")
        .when().delete()
        .then().log().all();
    }
}

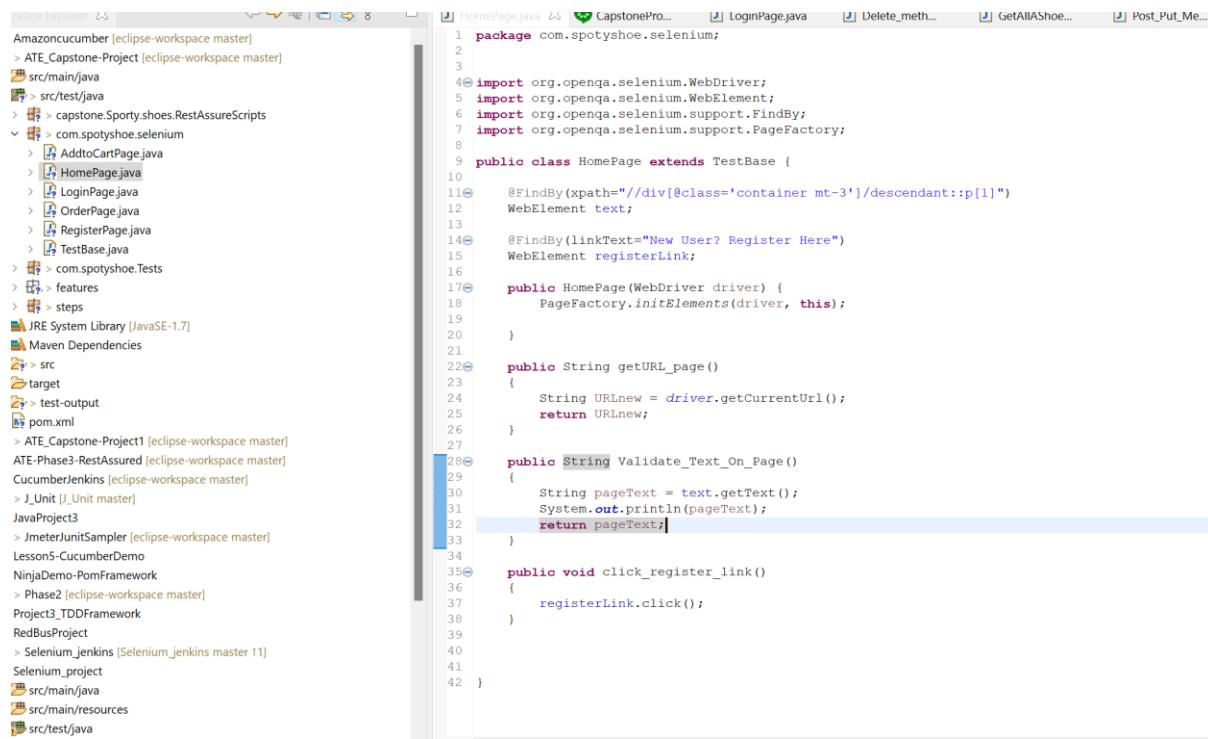
```

The screenshot shows the Eclipse IDE interface with the Java code for the `Delete_method` class. The code uses RestAssured to make a DELETE request to the "/delete-shoe" endpoint of the local server at port 9010, specifying the ID 101. The right side of the screen shows the terminal output of the test execution, which includes the API response and the test results.

## 2 . Creating the Browser-based end-user testing using Selenium WebDriver with TestNG Framework

Lets us Quickly add the selenium scripts of

Homepage:



The screenshot shows the Eclipse IDE interface with the 'Amazoncucumber' project selected. The left pane displays the project structure, including 'src/main/java', 'src/test/java', and several Java files like 'AddtoCartPage.java', 'LoginPage.java', 'OrderPage.java', 'RegisterPage.java', and 'TestBase.java'. The right pane shows the 'HomePage.java' code, which is a part of the 'com.spotyshoe.selenium' package. The code uses Selenium WebDriver to interact with a web page, specifically targeting elements with class 'container mt-3' and link text 'New User? Register Here'. It includes methods for getting the current URL and validating text on the page.

```
1 package com.spotyshoe.selenium;
2
3
4 import org.openqa.selenium.WebDriver;
5 import org.openqa.selenium.WebElement;
6 import org.openqa.selenium.support.FindBy;
7 import org.openqa.selenium.support.PageFactory;
8
9 public class HomePage extends TestBase {
10
11     @FindBy(xpath="//div[@class='container mt-3']/descendant::p[1]")
12     WebElement text;
13
14     @FindBy(linkText="New User? Register Here")
15     WebElement registerLink;
16
17     public HomePage(WebDriver driver) {
18         PageFactory.initElements(driver, this);
19     }
20
21     public String getURL_page()
22     {
23         String URLnew = driver.getCurrentUrl();
24         return URLnew;
25     }
26
27     public String Validate_Text_On_Page()
28     {
29         String pageText = text.getText();
30         System.out.println(pageText);
31         return pageText;
32     }
33
34
35     public void click_register_link()
36     {
37         registerLink.click();
38     }
39
40
41
42 }
```

Login Page:

```

1 package com.spotyshoe.selenium;
2
3
4
5@ import org.openqa.selenium.WebDriver;
6 import org.openqa.selenium.WebElement;
7 import org.openqa.selenium.support.FindBy;
8 import org.openqa.selenium.support.PageFactory;
9
10 public class LoginPage {
11
12@     @FindBy(xpath="//input[@id='email']")
13     WebElement loginEmail;
14
15@     @FindBy(xpath="//input[@id='password']")
16     WebElement loginpassword;
17
18@     @FindBy(xpath="//button[@type='submit']")
19     WebElement loginbtn;
20
21@     @FindBy(linkText="Cart")
22     WebElement clickCart;
23
24
25@     public LoginPage(WebDriver driver) {
26         PageFactory.initElements(driver, this);
27
28     }
29
30
31@     public void user_login()
32     {
33         loginEmail.sendKeys("mnavneenkumar.mpc@gmail.com");
34         loginpassword.sendKeys("Naveen@123");
35         loginbtn.click();
36     }
37
38
39@     public void click_cart()
40     {
41         clickCart.click();
42     }
43
44
45

```

Added to cart:

```

1 package com.spotyshoe.selenium;
2
3@ import org.openqa.selenium.JavascriptExecutor;
4 import org.openqa.selenium.WebDriver;
5 import org.openqa.selenium.WebElement;
6 import org.openqa.selenium.support.FindBy;
7 import org.openqa.selenium.support.PageFactory;
8
9 public class AddtoCartPage {
10
11
12@     @FindBy(xpath="//a[@id='shoe101']")
13     WebElement viewShoeBTN;
14
15@     @FindBy(xpath = "//a[@id='cart101']")
16     WebElement addtocartBTN;
17
18@     @FindBy(xpath="//div@class='alert alert-success']/descendant::p[1]")
19     WebElement successText;
20
21     JavascriptExecutor executor;
22
23@     public AddtoCartPage(WebDriver driver) {
24         PageFactory.initElements(driver, this);
25         executor = (JavascriptExecutor) driver;
26     }
27
28
29@     public void add_product_to_cart() throws InterruptedException
30     {
31
32         executor.executeScript("arguments[0].click()", addtocartBTN);
33
34
35
36
37
38@     public String validate_success_message()
39     {
40
41         String successtext = successText.getText();
42         return successtext;
43
44
45

```

Register Page:

```

2
3@ import org.openqa.selenium.WebDriver;
4 import org.openqa.selenium.WebElement;
5 import org.openqa.selenium.support.FindBy;
6 import org.openqa.selenium.support.PageFactory;
7
8 public class RegisterPage extends TestBase{
9     @FindBy(xpath="//input[@id='name']")
10    WebElement registerusername;
11
12    @FindBy(xpath="//input[@id='email']")
13    WebElement registeremail;
14
15    @FindBy(xpath="//input[@id='password']")
16    WebElement registerpassword;
17
18    @FindBy(xpath="//button[@type='submit']")
19    WebElement registerBtn;
20
21    @FindBy(xpath="//div[@class='mt-4 p-5 bg-primary text-white rounded']/descendant::p[3]")
22    WebElement userText;
23
24    public RegisterPage(WebDriver driver) {
25        PageFactory.initElements(driver, this);
26    }
27
28    public void register_user()
29    {
30        registerusername.sendKeys("Naveen");
31        registeremail.sendKeys("mnaveenkumar.mpc@gmail.com");
32        registerpassword.sendKeys("Naveen@123");
33        registerBtn.click();
34    }
35
36    public String validate_registration_URL()
37    {
38        String register_url = driver.getCurrentUrl();
39        return register_url;
40    }
41
42    public String validate_registration_Text()
43    {
44        String user_name = userText.getText();
45        return user_name;
46    }

```

## Order Page:

```

1 package com.spotyshoe.selenium;
2
3
4@ import org.openqa.selenium.WebDriver;
5 import org.openqa.selenium.WebElement;
6 import org.openqa.selenium.support.FindBy;
7 import org.openqa.selenium.support.PageFactory;
8
9 public class OrderPage {
10
11    @FindBy(linkText="Orders")
12    WebElement orderlink;
13
14
15    public OrderPage(WebDriver driver) {
16        PageFactory.initElements(driver, this);
17    }
18
19    public void click_orderPage()
20    {
21        orderlink.click();
22    }
23
24
25 }

```

The screenshot shows the Eclipse IDE interface with the OrderPage.java file open in the editor. The code defines a class OrderPage with a single method click\_orderPage() that clicks on an element with the linkText "Orders". The editor tab bar shows other files like HomePage.java, LoginPage.java, RegisterPage.java, AddtoCartPage.java, Delete\_method.java, and POST\_PUT\_Method.java. On the left, the Project Explorer view shows a complex workspace structure with multiple projects and sub-folders, including ATE\_Capstone-Project, capstone.Spoty.shoes.RestAssureScripts, com.spotyshoe.selenium, and various test and feature files.

## Selenium Webdriver with TestNg:

### Register Page

Amazoncucumber [eclipse-workspace master]

```

11 import com.spotyshoe.selenium.TestBase;
Run All
12 public class RegisterPageTest extends TestBase {
13
14     HomePage hp;
15     RegisterPage rp;
16     @BeforeTest
17     public void start_browser()
18     {
19         OpenBrowser("Chrome");
20         hp = new HomePage(driver);
21         rp = new RegisterPage(driver);
22     }
23 @Test(priority='1')
Run | Debug
24     public void test_click_register_link() throws InterruptedException
25     {
26         Thread.sleep(1500);
27         hp.click_register_link();
28     }
29
30 @Test(priority='2')
Run | Debug
31     public void test_getTitle_page()
32     {
33         String expected = "http://localhost:9010/register";
34         String Actual = hp.getURL_page();
35         Assert.assertEquals(Actual, expected);
36     }
37
38 @Test(priority='3')
Run | Debug
39     public void Test_register_user()
40     {
41         rp.register_user();
42     }
43
44 @Test(priority='4')
Run | Debug
45     public void Test_validate_registration_URL()
46

```

Writable Smart Insert 12 : 1 : 317

28°C ENG 19:09

LoginPage.java HomePageTes...

```

11 import com.spotyshoe.selenium.TestBase;
Run All
12 public class RegisterPageTest extends TestBase {
13
14     HomePage hp;
15     RegisterPage rp;
16     @BeforeTest
17     public void start_browser()
18     {
19         OpenBrowser("Chrome");
20         hp = new HomePage(driver);
21         rp = new RegisterPage(driver);
22     }
23 @Test(priority='1')
Run | Debug
24     public void test_click_register_link()
25     {
26         Thread.sleep(1500);
27         hp.click_register_link();
28     }
29
30 @Test(priority='2')
Run | Debug
31     public void test_getTitle_page()
32     {
33         String expected = "http://localhost:9010/register";
34         String Actual = hp.getURL_page();
35         Assert.assertEquals(Actual, expected);
36     }
37
38 @Test(priority='3')
Run | Debug
39     public void Test_register_user()
40     {
41         rp.register_user();
42     }
43
44 @Test(priority='4')
Run | Debug
45     public void Test_validate_registration_URL()
46

```

<terminated> RegisterPageTest [TestNG] C:\Users\mnavi\p2\pool\plugins\org.eclipse.jdt.openjdk.hotspot.jre.full.win32.x86\_64\_17.0.0
at org.testng.SuiteRunnerWorker.run(SuiteRunnerWorker.java:95)
at org.testng.TestNG.runSuitesSequentially(TestNG.java:1256)
at org.testng.TestNG.runSuitesLocally(TestNG.java:1176)
at org.testng.TestNG.runSuites(TestNG.java:1099)
at org.testng.TestNG.run(TestNG.java:1067)
at org.testng.remote.AbstractRemoteTestNG.run(AbstractRemoteTestNG.java:115)
at org.testng.remote.RemoteTestNG.initAndRun(RemoteTestNG.java:251)
at org.testng.remote.RemoteTestNG.main(RemoteTestNG.java:77)

=====
Default test
Tests run: 5, Failures: 1, Skips: 0
=====
Default suite
Total tests run: 5, Passes: 4, Failures: 1, Skips: 0
=====

Service connected to port 0 ::19010 [localhost:9010].

## LoginPage :

```

6 import com.spotyshoe.selenium.HomePage;
7 import com.spotyshoe.selenium.LoginPage;
8 import com.spotyshoe.selenium.RegisterPage;
9 import com.spotyshoe.selenium.TestBase;
Run All
.0 public class LoginPageTest extends TestBase {

.1     HomePage hp;
.2     RegisterPage rp;
.3     LoginPage lp;
.4
.5     @BeforeTest
.6     public void start_browser()
.7     {
.8         OpenBrowser("Chrome");
.9         hp = new HomePage(driver);
.0         rp = new RegisterPage(driver);
.1         lp = new LoginPage(driver);
.2     }
.3@ @Test(priority='1')
Run | Debug
.4     public void test_login()
.5     {
.6         lp.user_login();
.7     }
.8
.9
.10    @Test(priority='2')
Run | Debug
.11    public void test_getTitle_page()
.12    {
.13        String expected = "http://localhost:9010/login";
.14        String Actual = hp.getURL_page();
.15        Assert.assertEquals(Actual, expected);
.16    }
.17
.18
.19    @Test(priority='3')
Run | Debug
.20    public void Test_validate_registration_Text()
.21    {

```

<terminated> LoginPageTest [TestNG] C:\Users\mnave\p2\pool\plugins\org.eclipse.jdt.openjdk.hotspot.jre.full.win32.x86\_64\_17.0.8.v [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.  
Nov 15, 2023 7:12:25 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch  
WARNING: Unable to find an exact match for CDP version 119, so returning the closest v  
PASSED: com.spotyshoe.Tests.LoginPageTest.test\_login  
PASSED: com.spotyshoe.Tests.LoginPageTest.test\_click\_cart  
PASSED: com.spotyshoe.Tests.LoginPageTest.test\_getTitle\_page  
PASSED: com.spotyshoe.Tests.LoginPageTest.Test\_validate\_registration\_Text  
=====  
Default test  
Tests run: 4, Failures: 0, Skips: 0  
=====  
Default suite  
Total tests run: 4, Passes: 4, Failures: 0, Skips: 0  
=====

## OrderPage:

```

Amazoncucumber
|> ATE_Capstone
src/main/java
|> src/test/java
|> capstor
|> com.sp
|> AddtoC
|> Homef
|> LoginP
|> OrderF
|> Registe
|> TestBa
|> > featur
|> > steps
JRE System L
Maven Depe
|> src
target
|> test-output
pom.xml
|> ATE_Capstone
ATE-Phase3-Res
CucumberJenkin
|> JUnit J_ Unit
JavaProject3
|> JmeterUnitSa
Lessons-Cucum
NinjaDemo-Pon
|> Phase2 [eclip
Project3 TDDFr

```

```

1 package com.spotyshoe.Tests;
2@ import org.testng.Assert;
3 import org.testng.annotations.BeforeTest;
4 import org.testng.annotations.Test;
5 import com.spotyshoe.selenium.HomePage;
6 import com.spotyshoe.selenium.LoginPage;
7 import com.spotyshoe.selenium.OrderPage;
8 import com.spotyshoe.selenium.RegisterPage;
9 import com.spotyshoe.selenium.TestBase;
10
11     Run All
12     public class OrderpageTest extends TestBase {HomePage hp;
13
14         RegisterPage rp;
15         LoginPage lp;
16         OrderPage op;
17
18         @BeforeTest
19         public void start_browser(){
20             OpenBrowser("Chrome");
21             hp = new HomePage(driver);
22             rp = new RegisterPage(driver);
23             lp = new LoginPage(driver);
24             op = new OrderPage(driver);
25
26         }
27@ @Test(priority='1')
Run | Debug
28         public void test_login(){
29             lp.user_login();
30         }
31
32@ @Test(priority='2')
Run | Debug
33         public void test_click_orders()
34         {
35             op.click_orderPage();
36         }
37@ @Test(priority='3')
Run | Debug
38         public void test_getTitle_page(){String expected = "http://localhost:9010/orders";
39
40             String Actual = hp.getURL_page();
41             Assert.assertEquals(Actual, expected);}
42
43     }

```

```

<terminated> OrderpageTest [TestNG] C:\Users\mnave\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v
|[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.
Nov 15, 2023 7:15:34 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 119, so returning the closest v
PASSED: com.spotyshoe.Tests.OrderpageTest.test_login
PASSED: com.spotyshoe.Tests.OrderpageTest.test_click_orders
PASSED: com.spotyshoe.Tests.OrderpageTest.test_getTitle_page

=====
Default test
Tests run: 3, Failures: 0, Skips: 0
=====

=====
Default suite
Total tests run: 3, Passes: 3, Failures: 0, Skips: 0
=====


```

### Added to Cart Test:

```

LoginPage.java HomePage... LoginPageTe... OrderpageTes... RegisterPag... *AddtoCartPa... RegisterPag... OrderPage.java 12
1 import org.testng.Assert;
2 import org.testng.annotations.BeforeTest;
3 import org.testng.annotations.Test;
4 import org.openqa.selenium.WebDriver;
5 import com.spotyshoe.selenium.AddtoCartPage;
6 import com.spotyshoe.selenium.HomePage;
7 import com.spotyshoe.selenium.LoginPage;
8 import com.spotyshoe.selenium.RegisterPage;
9 import com.spotyshoe.selenium.TestBase;
10
11 public class AddtoCartPageTest extends TestBase {
12     HomePage hp;
13     LoginPage lp;
14     AddtoCartPage ac;
15     @BeforeTest
16     public void start_browser() {
17         OpenBrowser("Chrome");
18         hp = new HomePage(driver);
19         rp = new RegisterPage(driver);
20         lp = new LoginPage(driver);
21         ac = new AddtoCartPage(driver);
22     }
23
24     @Test(priority=1)
25     public void test_login() {
26         lp.user_login();
27     }
28
29
30     @Test(priority=2)
31     public void test_add_product_to_cart() throws InterruptedException {
32         ac.add_product_to_cart();
33     }
34
35     @Test(priority=3)
36     public void test_validate_success_message() {
37         String expected = "Message:Shoe BlueWave Running Shoes Added Successfully to Cart";
38         String actualText= ac.validate_success_message();
39         Assert.assertEquals(actualText, expected);
40     }

```

```

<terminated> AddtoCartPageTest [TestNG] C:\Users\mnav\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.1\lib\testng-7.7.1.jar
[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.
Nov 15, 2023 7:21:36 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 119, so returning the closest version 118
PASSED: com.spotyshoe.Tests.AddtoCartPageTest.test_login

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

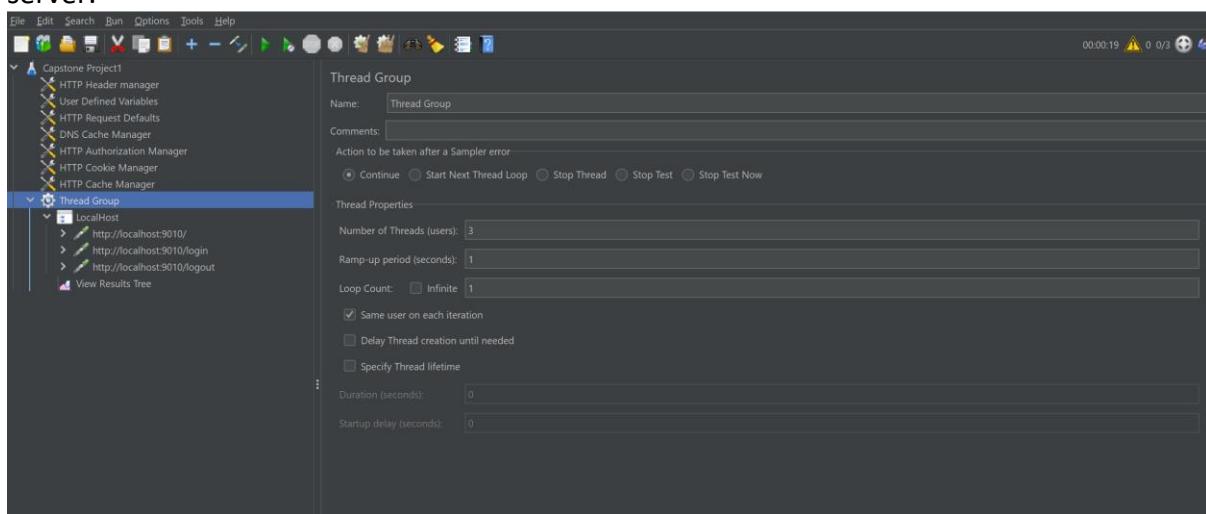
=====
Default suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
=====


```

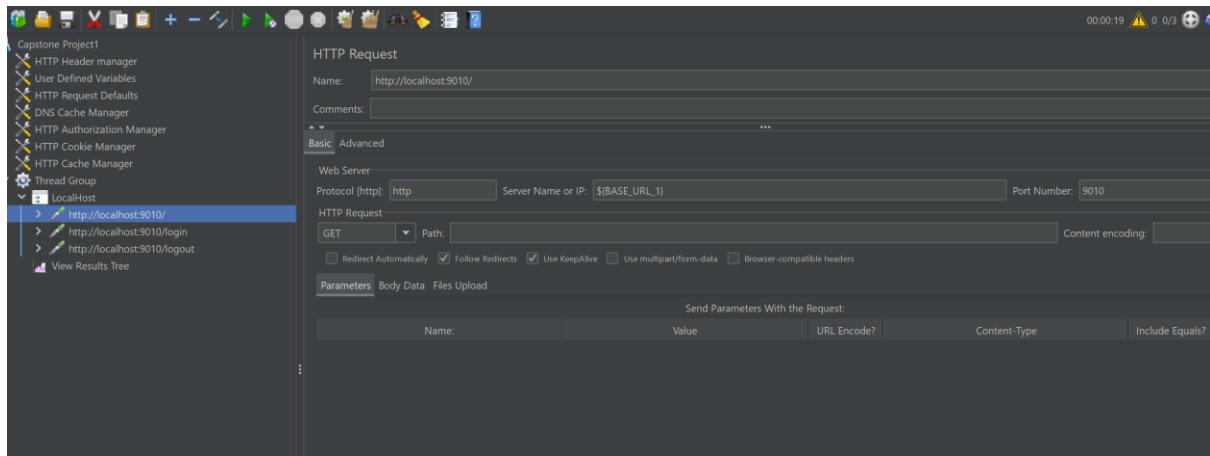
### 3.JMETER SCRIPT

Recording of script has been done via blazemeter of local host 9010. Login and logout .

Script has been recorded and we create a thread group , to load the Thread users to run on server.

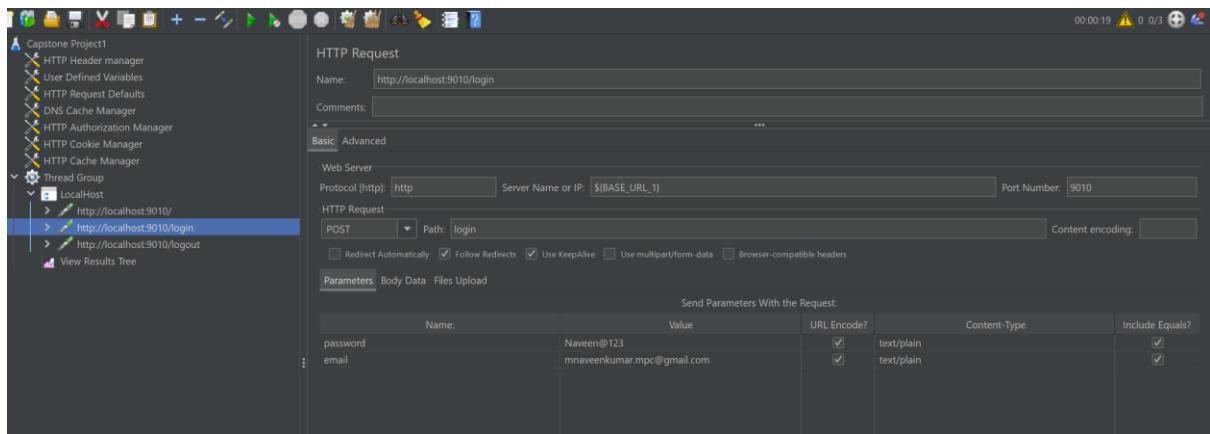


Added three requests - home page, login page , logoutpage.

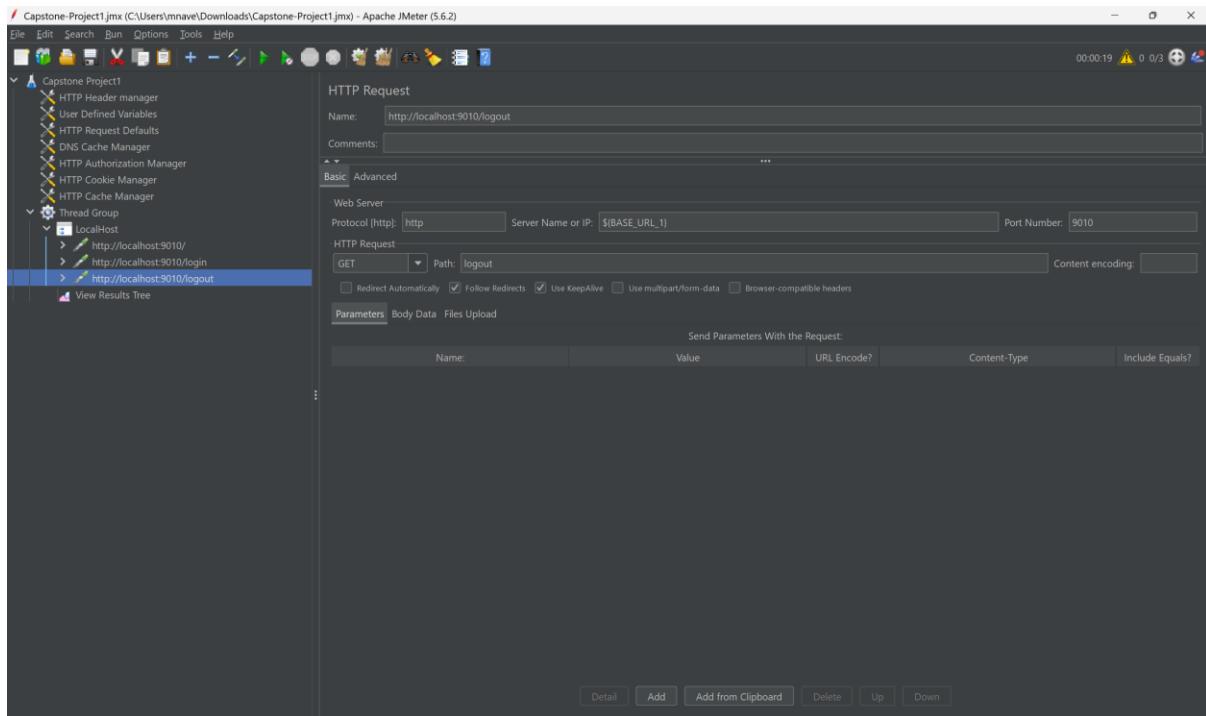


## Login Page :

User Email  
Password

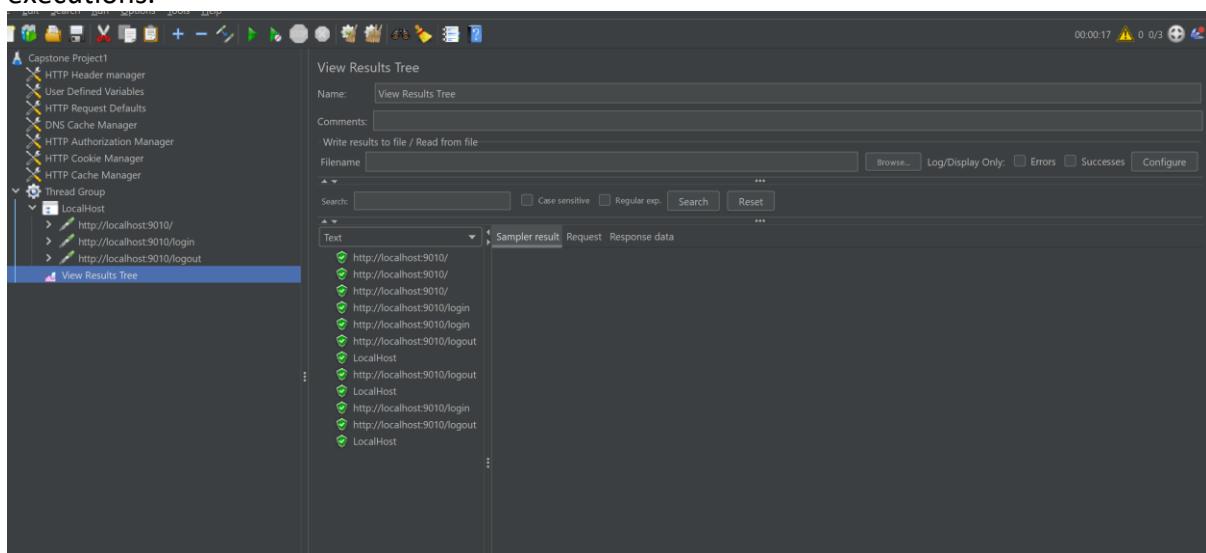


## Logout :



## Output:

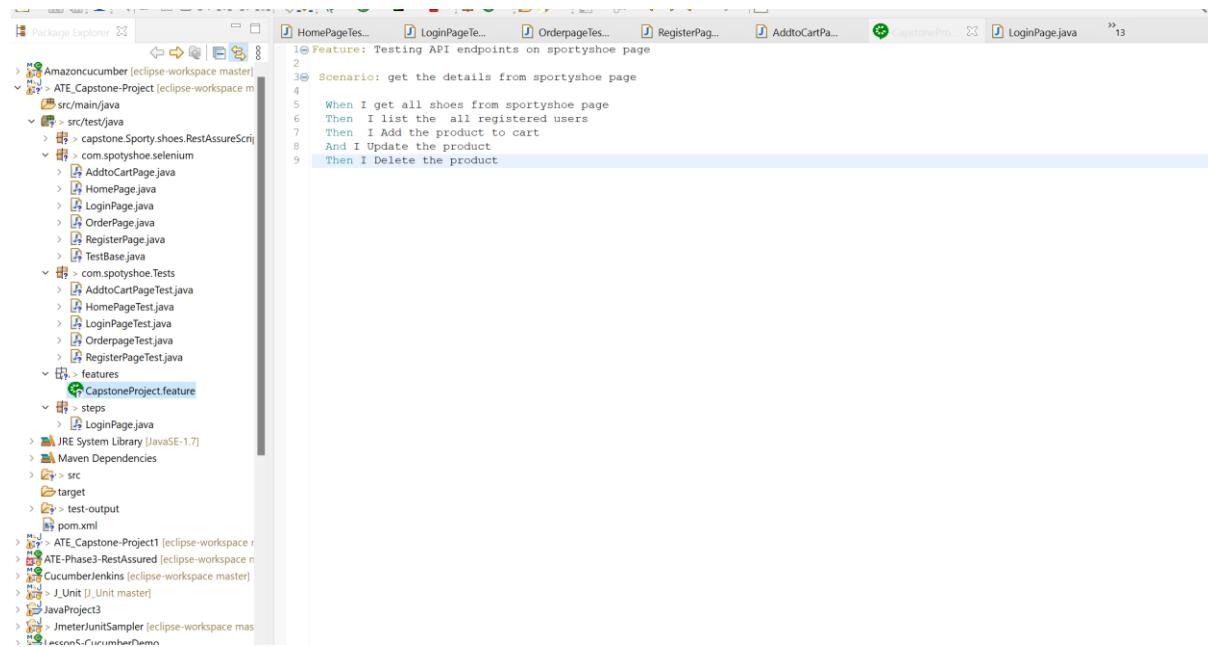
Thread Load given with 3 users , so each request executes 3 times overall 3 requests with 9 executions.



## 4 . CUCUMBER

### Feature File:

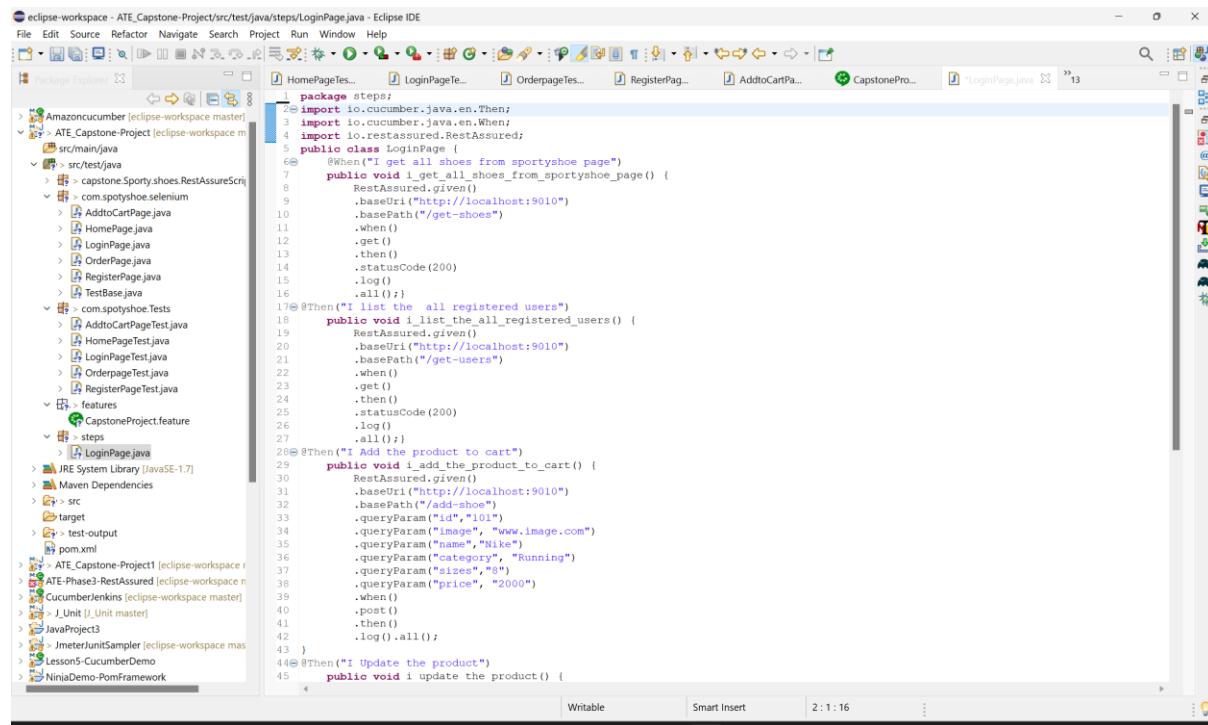
Cucumber file with feature file and scenario.



The screenshot shows the Eclipse IDE interface with the 'Feature File' open. The 'features' directory contains the file 'CapstoneProject.feature'. The code in the file is as follows:

```
Feature: Testing API endpoints on sportyshoe page
  Scenario: get the details from sportyshoe page
    When I get all shoes from sportyshoe page
    Then I list the all registered users
    Then I Add the product to cart
    And I Update the product
    Then I Delete the product
```

### STEPS:



The screenshot shows the Eclipse IDE interface with the 'LoginPage.java' file open. The file contains step definitions for the 'LoginPage' class. The code is as follows:

```
package steps;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;
import io.restassured.RestAssured;
public class LoginPage {
    @When("I get all shoes from sportyshoe page")
    public void i_get_all_shoes_from_sportyshoe_page() {
        RestAssured.given()
            .baseUri("http://localhost:9010")
            .basePath("/get-shoes")
            .when()
            .get()
            .then()
            .statusCode(200)
            .log()
            .all();
    }
    @Then("I list the all registered users")
    public void i_list_the_all_registered_users() {
        RestAssured.given()
            .baseUri("http://localhost:9010")
            .basePath("/get-users")
            .when()
            .get()
            .then()
            .statusCode(200)
            .log()
            .all();
    }
    @Then("I Add the product to cart")
    public void i_add_the_product_to_cart() {
        RestAssured.given()
            .baseUri("http://localhost:9010")
            .basePath("/add-shoe")
            .queryParam("id", "101")
            .queryParam("image", "www.image.com")
            .queryParam("name", "Nike")
            .queryParam("category", "Running")
            .queryParam("sizes", "8")
            .queryParam("price", "2000")
            .when()
            .post()
            .then()
            .log().all();
    }
    @Then("I Update the product")
    public void i_update_the_product() {
    }
}
```

```

    package com.spotyshoe.sportyshoes;
    import io.restassured.RestAssured;
    import io.restassured.response.Response;
    import org.junit.*;
    import static io.restassured.RestAssured.given;
    import static org.hamcrest.Matchers.*;

    public class HomePageTest {
        @Test
        public void i_add_the_product_to_cart() {
            RestAssured.given()
                    .baseUri("http://localhost:9010")
                    .basePath("/add-shoe")
                    .queryParam("id", "101")
                    .queryParam("image", "www.image.com")
                    .queryParam("name", "Nike")
                    .queryParam("category", "Running")
                    .queryParam("sizes", "8")
                    .queryParam("price", "2000")
                    .when()
                    .post()
                    .then()
                    .log().all();
        }

        @Then("I Update the product")
        public void i_update_the_product() {
            RestAssured.given()
                    .baseUri("http://localhost:9010")
                    .basePath("/update-shoe")
                    .queryParam("id", "101")
                    .queryParam("image", "www.image123.com")
                    .queryParam("name", "Reebok")
                    .queryParam("category", "walking")
                    .queryParam("sizes", "8,9,10")
                    .queryParam("price", "1500")
                    .when()
                    .put()
                    .then()
                    .log().all();
        }

        @Then("I Delete the product")
        public void i_delete_the_product() {
            RestAssured.given()
                    .baseUri("http://localhost:9010")
                    .basePath("/delete-shoe")
                    .queryParam("id", "101")
                    .when()
                    .delete()
                    .then().log().all();
        }
    }

```

## OUTPUT:

```

    Feature: Testing API endpoints on sportyshoe page
      Scenario: get the details from sportyshoe page
        When I get all shoes from sportyshoe page
        Then I list the all registered users
        Then I Add the product to cart
        And I Update the product
        Then I Delete the product
    
```

Output in the terminal:

```

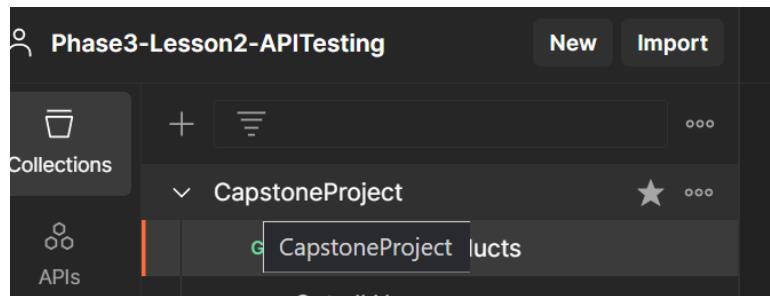
<terminated> CapstoneProject.feature [Cucumber Feature] C:\Users\mnavet\p2\pool\plugins\org.eclipse.jst.j2ee\_fragments\src\test\java\features\CapstoneProject.feature
    Then I Add the product to cart # steps.LoginPage.i_add_
HTTP/1.1 200
Set-Cookie: JSESSIONID=DD1FF667C3098674C32BFFCAC2DE3AD; Path=/
Content-Type: application/json
Transfer-Encoding: chunked
Date: Wed, 15 Nov 2023 14:07:34 GMT
Keep-Alive: timeout=60
Connection: keep-alive

{
    "code": 101,
    "message": "Reebok Updated Successfully.",
    "shoe": [
        {
            "id": 101,
            "image": "www.image123.com",
            "name": "Reebok",
            "category": "walking",
            "sizes": "8,9,10",
            "price": 1500
        }
    ]
}
    And I Update the product # steps.LoginPage.i_upd...
HTTP/1.1 200
Content-Type: application/json
Transfer-Encoding: chunked
Date: Wed, 15 Nov 2023 14:07:34 GMT
Keep-Alive: timeout=60
Connection: keep-alive

{
    "code": 101,
    "message": "Shoe with ID 101 Deleted Successfully."
}
    Then I Delete the product # steps.LoginPage.i_dele...
1 Scenarios (1 passed)
5 Steps (5 passed)
0m1.862s
    
```

## 5 .API Testing With Postman And Rest Assured

In PostMan First we need to create Collection and name it as Capstone Project and followed with some methods to test.



Methods Used:

Get  
Post  
Put  
Delete

## 1. Getting the list of all products in the store :

- Now Getting list of all products has done with GET Method And Url is (  
<http://localhost:9010/get-shoes>)

OutPut :

A detailed screenshot of the Postman interface showing a specific API call. In the center, there's a 'Request' pane with the method 'GET', URL 'http://localhost:9010/get-shoes', and a 'Send' button. Above the request, the collection 'CapstoneProject' is selected, and under it, the specific endpoint 'GET Get all List Products' is chosen. To the right of the request, there's a 'Body' tab showing a JSON response with three items of shoes. The response body is as follows:

```
1 "code": 101,
2 "message": "12 Shoes Fetched Successfully.",
3 "shoes": [
4     {
5         "id": 101,
6         "image": "1.png",
7         "name": "Bluewave Running Shoes",
8         "category": "Sports",
9         "sizes": "7, 8, 9",
10        "price": 2000
11    },
12    {
13        "id": 201,
14        "image": "2.png",
15        "name": "Elegant Leather Loafers",
16        "category": "Formal",
17        "sizes": "7, 8, 9",
18        "price": 3000
19    },
20    {
21        "id": 301,
22        "image": "3.png",
23        "name": "NooFlex Athletic Shoes",
24        "category": "Sports",
25        "sizes": "7, 8, 9, 10",
26        "price": 4500
27    },
28    {
29        "id": 401
30    }
31]
```

The bottom of the interface shows various status indicators and navigation buttons.

## **2 . Getting all Users**

- Get method used to get all users from sporty shoes
- Status code Ok with 200.

The screenshot shows the Postman application interface. On the left, there's a sidebar with project navigation (Collections, APIs, Environments, Monitors, History) and a list of recent projects. The main area displays a collection named 'CapstoneProject' with a single item: 'GET Get all Users'. This item is highlighted in orange. Below it, there are other items like 'Get all List Products', 'PUT Put Update the data', and 'DELETE Delete id = 101'. The right side of the screen shows the request details for 'GET Get all Users'. The URL is 'http://localhost:9010/get-users'. The response status is 'Status: 200 OK' with a 'Time: 30 ms' and a 'Size: 440 B'. The response body is displayed in a JSONpretty-printed format:

```

1
2   "code": 101,
3   "message": "3 Users Fetched Successfully.",
4   "users": [
5     {
6       "name": "John Watson",
7       "email": "john@example.com",
8       "password": "john123"
9     },
10    {
11      "name": "Fionna Flynn",
12      "email": "fionna@example.com",
13      "password": "fionna123"
14    },
15    {
16      "name": "Sia Sen",
17      "email": "sia@example.com",
18      "password": "sia123"
19    }
20  ]
21

```

### 3 . Add the Product

- Product has been added to list using method POST.
- Product id=101 and name ="SampleShoe" has been added.
- These Post method adds the data to the server or given url database.
- Post method Defaultly gives Query Parameters to add the data to sporty shoes.

OutPut:

The screenshot shows the Postman application interface. On the left, the 'Collections' sidebar lists several projects and environments. The 'CapstoneProject' collection is expanded, showing four items: 'Get all List Products', 'GET Get all Users', 'POST Post add Url', and 'PUT Put Update the data'. The 'POST Post add Url' item is selected. The main workspace displays a POST request to 'http://localhost:9010/add-shoe?id=101&image=image\_url&name=SampleShoe&category=Running&sizes=9&price=1000'. The 'Params' tab shows query parameters: id (101), image (image\_url), name (SampleShoe), category (Running), sizes (9), and price (1000). The 'Body' tab shows the JSON response received: a code of 101 and a message 'SampleShoe Added Successfully.' followed by a shoe object with the same details. The status bar at the bottom indicates 'Status: 200 OK'.

## Bugs :

- Updating data using put method is not fine ..gives error with given url shown below.
- So to overcome I used Post Method url and Post method output data as body in put method.

The screenshot shows a Postman interface with a PUT request to `http://localhost:9010/update-shoe`. The 'Params' tab is selected. The 'Body' tab shows an empty table with columns 'Key' and 'Value'. The 'Headers' tab shows 8 headers. The 'Tests' and 'Settings' tabs are also visible. In the 'Body' section, the 'Pretty' tab is selected, showing a JSON response:

```
1  {
2      "timestamp": "2023-11-14T08:10:08.775+00:00",
3      "status": 400,
4      "error": "Bad Request",
5      "path": "/update-shoe"
6  }
```

## 4 . Update the Data

- Data has been updated to Spoty shoes using Put method.
- Body given as below to update spoty shoes

```
{
    "id": 101,
    "name": "Updated Shoe Name",
    "category": "Updated Category",
    "sizes": "8,9,10",
    "price": 1500
}
```

The screenshot shows the Postman interface with the following details:

- Project:** CapstoneProject
- Method:** PUT
- URL:** `http://localhost:9010/update-shoe?id=101&image=image_url&name=Updated Shoe Name &category=Updated Category&sizes=8,9,10&price=1500`
- Params:**
  - id: 101
  - image: image\_url
  - name: Updated Shoe Name
  - category: Updated Category
  - sizes: 8,9,10
  - price: 1500
- Body:** (Pretty) JSON response:

```

1  "code": 101,
2  "message": "Updated Shoe Name Updated Successfully.",
3  "shoe": {
4    "id": 101,
5    "image": "image_url",
6    "name": "Updated Shoe Name",
7    "category": "Updated Category",
8    "sizes": "8,9,10",
9    "price": 1500
11  }

```
- Status:** 200 OK
- Time:** 15 ms
- Size:** 357 B

## 5. Delete ID

- Id has been Deleted using DELETE Method having ID number .
- Now deleting id with number ID = 101

Above ID is deleted successfully:

The screenshot shows the Postman interface with the following details:

- Project:** CapstoneProject
- Method:** DELETE
- URL:** `http://localhost:9010/delete-shoe?id=101`
- Params:**
  - id: 101
- Body:** (Pretty) JSON response:

```

1  "code": 101,
2  "message": "Shoe with ID 101 Deleted Successfully."

```
- Status:** 200 OK
- Time:** 13 ms
- Size:** 227 B