**Github Repository Link:-** https://github.com/Kesavanagendra/Capstone\_Project.git

**Create a Testing Framework for Sporty Shoes Website**

**Rest Assured**

**Retrieve the list of all products in the store.**

package com.sportyshoe.RestAssuredScripts;

import org.hamcrest.Matchers;

import org.testng.annotations.Test;

import io.restassured.RestAssured;

public class GetAllShoes {

@Test()

public void getAllShoes() {

RestAssured.given()

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

.basePath("/get-shoes")

.when().get()

.then().statusCode(200)

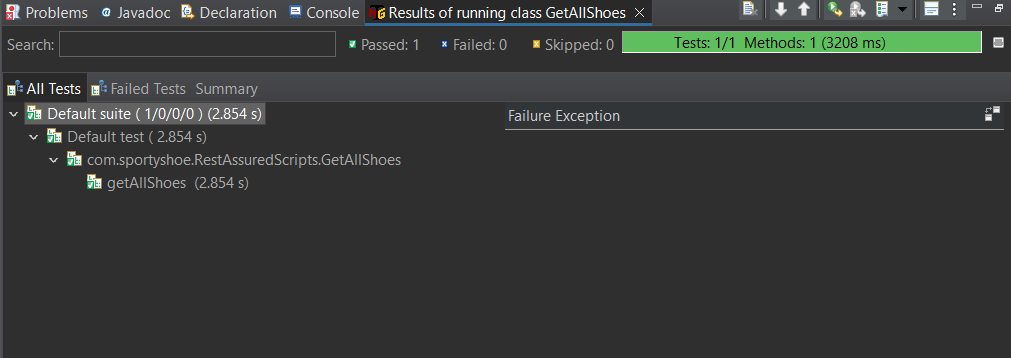
.body("shoes.id[0]", Matchers.equalTo(201))

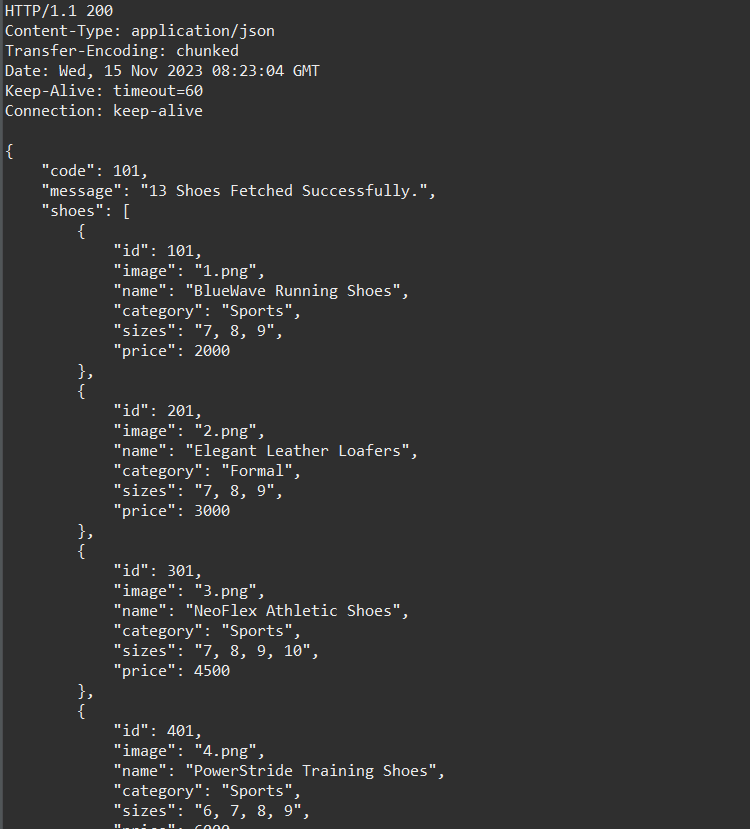
.log().all();

}

}

**Output:-**

****





**Retrieve the list of all registered users.**

package com.sportyshoe.RestAssuredScripts;

import org.hamcrest.Matchers;

import org.testng.annotations.Test;

import io.restassured.RestAssured;

public class GetUsers {

@Test

public void getAllRegisteredUsers() {

RestAssured.given()

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

.basePath("/get-users")

.when().get()

.then().statusCode(200)

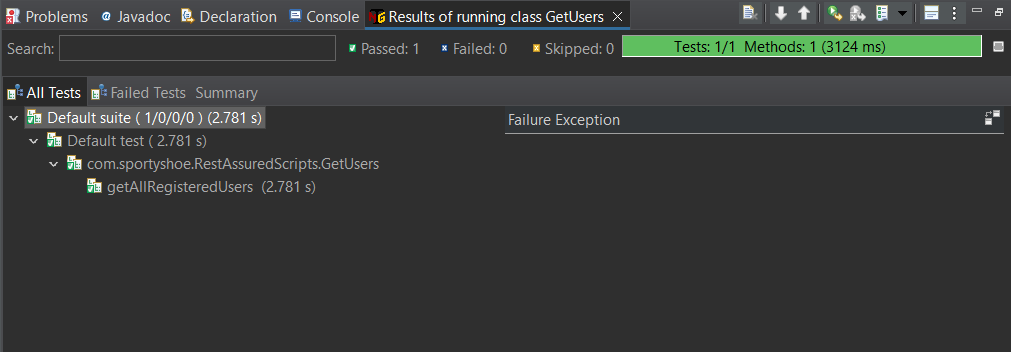
.body("code", Matchers.equalTo(101))

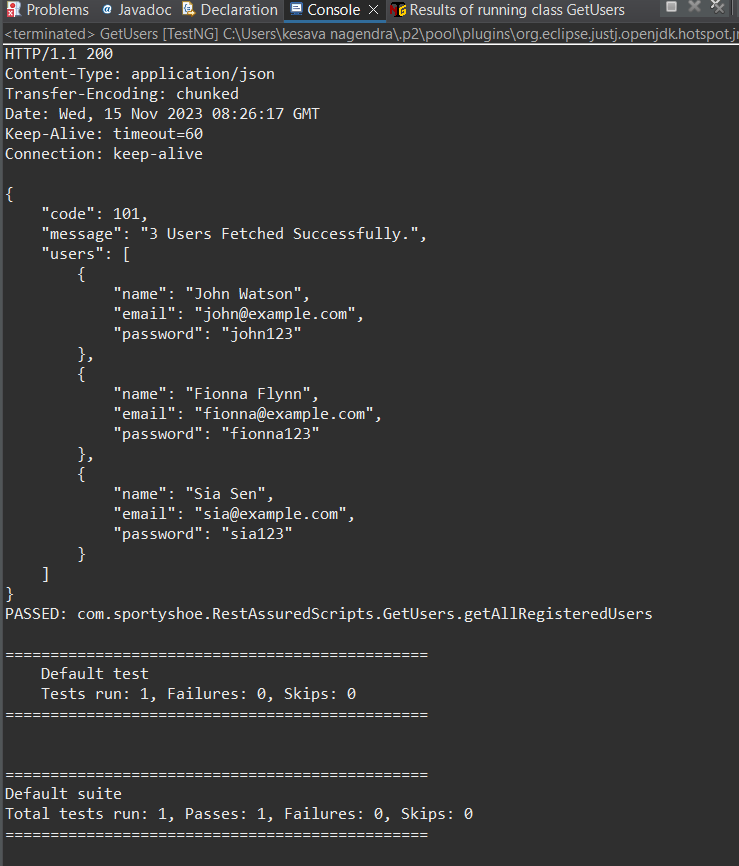
.log().all();

}

}

**Output:-**

****

****

**Add the product.**

package com.sportyshoe.RestAssuredScripts;

import org.hamcrest.Matchers;

import org.testng.annotations.Test;

import io.restassured.RestAssured;

public class AddShoeProduct {

@Test(priority = '1')

public void addProduct() {

RestAssured.given()

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

.basePath("/add-shoe")

.queryParam("id", "1103")

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

.queryParam("name", "Nike")

.queryParam("category", "Running")

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

.queryParam("price", "1000")

.when().post()

.then().statusCode(200)

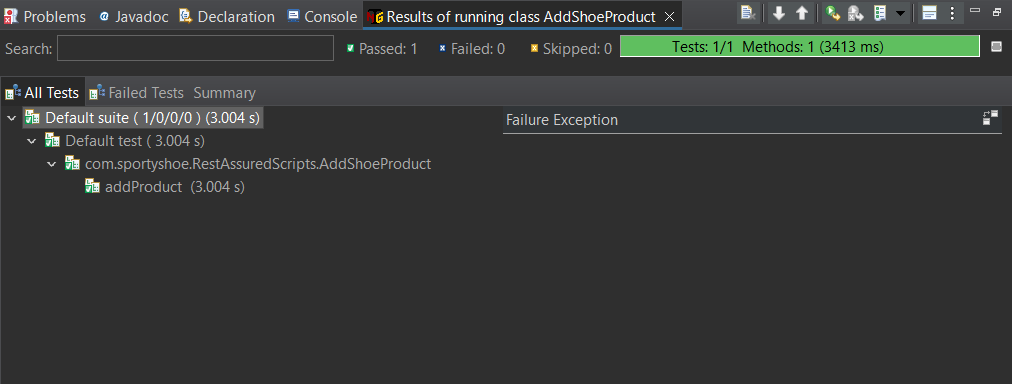
.body("shoe", Matchers.hasEntry("category","Running"))

.log().all();

}

}

**Output:-**





**Update the product.**

package com.sportyshoe.RestAssuredScripts;

import org.hamcrest.Matchers;

import org.testng.annotations.Test;

import io.restassured.RestAssured;

public class UpdateProduct {

@Test

public void updateShoeProduct() {

RestAssured.given()

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

.basePath("/update-shoe")

.queryParam("id", "1103")

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

.queryParam("name", "Puma")

.queryParam("category", "Sports")

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

.queryParam("price", "1500")

.when().put()

.then().statusCode(200)

.body("shoe.name", Matchers.equalTo("Puma"))

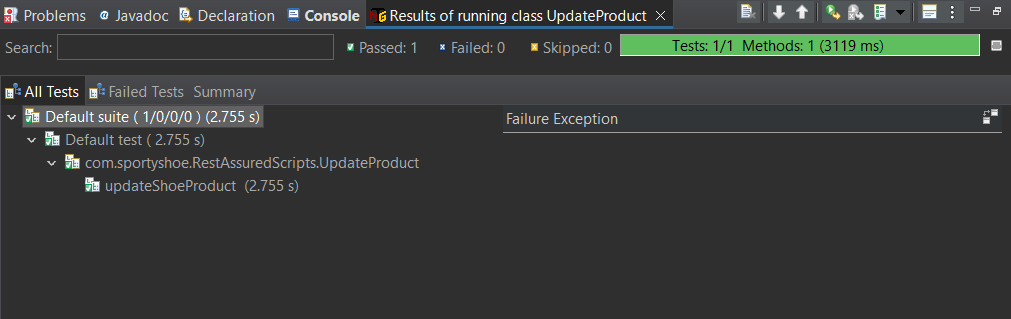
.body("shoe.sizes" ,Matchers.equalTo("5,6,7"))

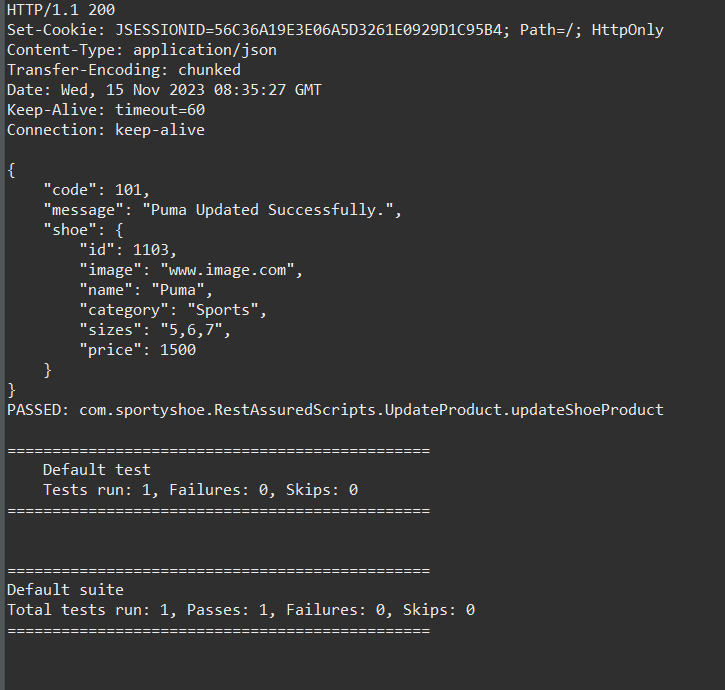
.log().all();

}

}

**Output:-**

****

****

**Delete the product.**

package com.sportyshoe.RestAssuredScripts;

import org.hamcrest.Matchers;

import org.testng.annotations.Test;

import io.restassured.RestAssured;

public class DeleteProduct {

@Test

public void deleteShoeProduct() {

RestAssured.given()

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

.basePath("/delete-shoe")

.queryParam("id", "1103")

.when().delete()

.then().statusCode(200)

.body("message", Matchers.equalTo("Shoe with ID 1103 Deleted Successfully."))

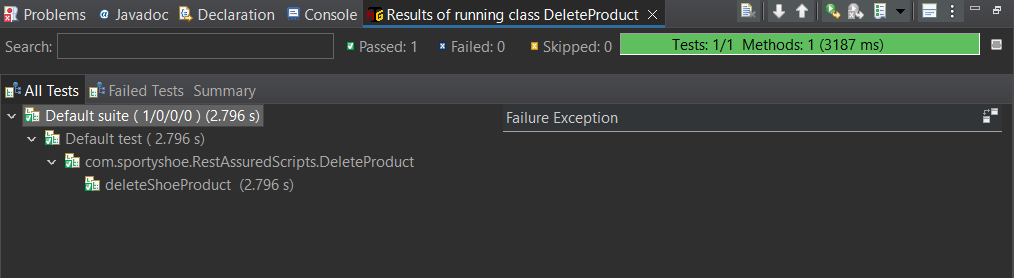
.log().all();

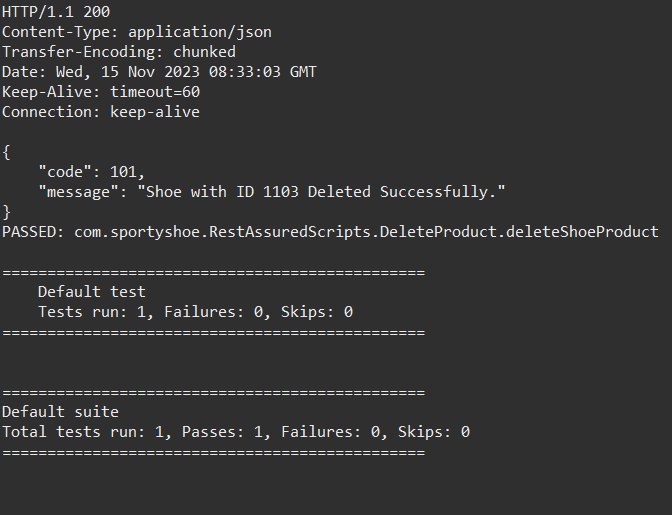
}

}

**Output:-**

Here After also deleting the Product from store . its shows product available in the store.





**Selenium scripts using TestNG**

package com.sportyshoeSeleniumCucumberScripts;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class TestBase {

public static WebDriver driver;

public void openBrowser() {

driver = new ChromeDriver();

driver.manage().window().maximize();

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

}

public static void closeBrowser() {

driver.close();

}

}

**HomePage**

package com.sportyshoeSeleniumCucumberScripts;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.support.FindBy;

import org.openqa.selenium.support.PageFactory;

public class HomePage extends TestBase{

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

WebElement text;

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

WebElement registerLink;

public HomePage(WebDriver driver) {

PageFactory.initElements(driver, this);

}

public String getURL\_page() {

String URLnew = driver.getCurrentUrl();

return URLnew;

}

public String getTextOnPage() {

String pagetext = text.getText();

System.out.println(pagetext);

return pagetext;

}

public void click\_register\_link()

{

registerLink.click();

}

}

**HomePageTest**

package com.sportyshoeTestPages;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

import org.testng.Assert;

import com.sportyshoeSeleniumCucumberScripts.HomePage;

import com.sportyshoeSeleniumCucumberScripts.TestBase;

public class HomePageTest extends TestBase {

HomePage hp;

@BeforeTest

public void startBrowser() {

openBrowser();

hp = new HomePage(driver);

}

@Test(priority = '1')

public void test\_GetUrl() {

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

String Actual = hp.getURL\_page();

Assert.assertEquals(Actual, expected);

System.out.println();

}

@Test(priority = '2')

public void test\_getTextOnPage() {

String ActualText = hp.getTextOnPage();

String ExpectedText = "Powered By Simplilearn";

Assert.assertEquals(ExpectedText, ActualText);

}

@Test(priority='3')

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

{

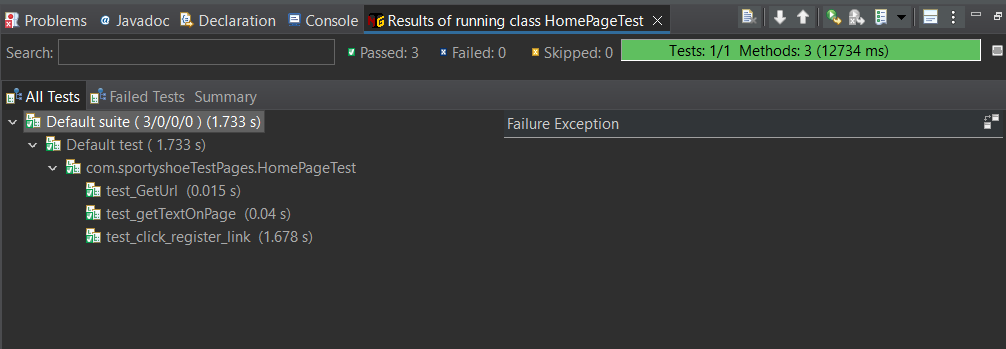
Thread.sleep(1500);

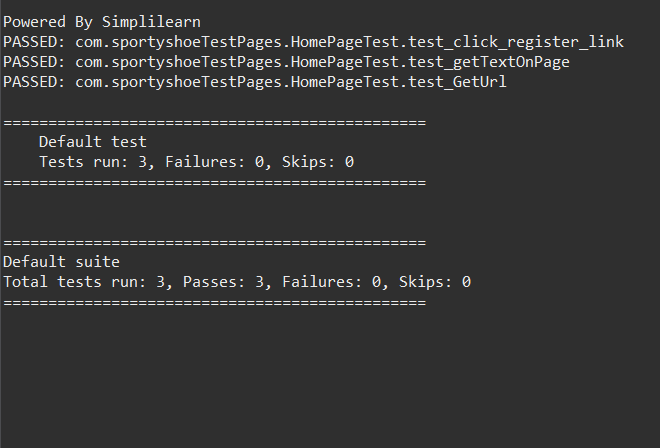
hp.click\_register\_link();

}

}

**Output:-**

****

****

**RegisterPage**

package com.sportyshoeSeleniumCucumberScripts;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.support.FindBy;

import org.openqa.selenium.support.PageFactory;

public class RegisterPage extends TestBase {

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

WebElement registername;

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

WebElement registeremail;

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

WebElement registerpassword;

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

WebElement registerBtn;

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

WebElement userText;

public RegisterPage(WebDriver driver) {

PageFactory.initElements(driver, this);

}

public void register\_user()

{

registername.sendKeys("Kesava");

registeremail.sendKeys("Kesava@gmail.com");

registerpassword.sendKeys("Kesava@123");

registerBtn.click();

}

public String validate\_registration\_URL()

{

String register\_url = driver.getCurrentUrl();

return register\_url;

}

public String validate\_registration\_Text()

{

String user\_name = userText.getText();

return user\_name;

}

}

**RegisterPageTest**

package com.sportyshoeTestPages;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

import org.testng.Assert;

import com.sportyshoeSeleniumCucumberScripts.HomePage;

import com.sportyshoeSeleniumCucumberScripts.RegisterPage;

import com.sportyshoeSeleniumCucumberScripts.TestBase;

public class RegisterPageTest extends TestBase {

HomePage hp;

RegisterPage rp;

@BeforeTest

public void start\_browser()

{

openBrowser();

hp = new HomePage(driver);

rp = new RegisterPage(driver);

}

@Test(priority='1')

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

{

Thread.sleep(1500);

hp.click\_register\_link();

}

@Test(priority='2')

public void test\_getTitle\_page()

{

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

String Actual = hp.getURL\_page();

Assert.assertEquals(Actual, expected);

}

@Test(priority='3')

public void Test\_register\_user()

{

rp.register\_user();

}

@Test(priority='4')

public void Test\_validate\_registration\_URL()

{

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

String Actual = rp.validate\_registration\_URL();

Assert.assertEquals(Actual, expected);

}

@Test(priority='5')

public void Test\_validate\_registration\_Text()

{

String expected = "Hello Kesava !";

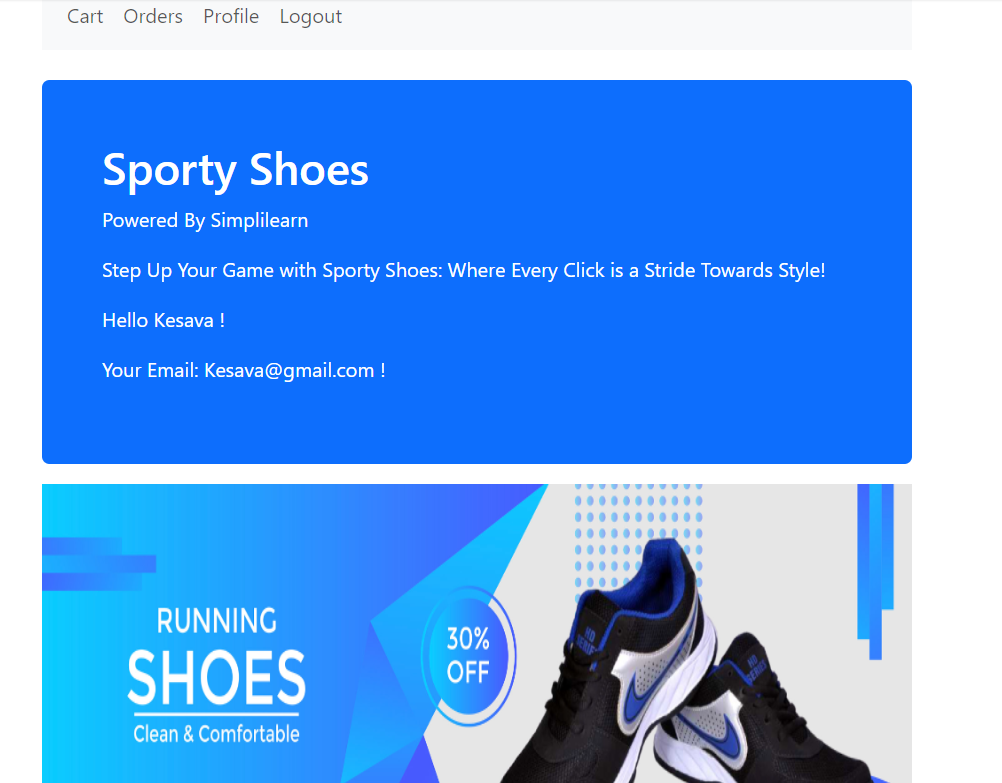
String actualText = rp.validate\_registration\_Text();

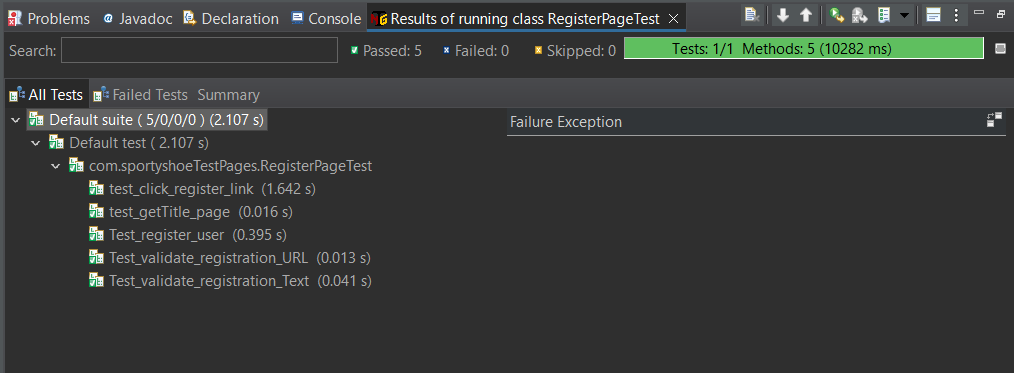
Assert.assertEquals(actualText, expected);

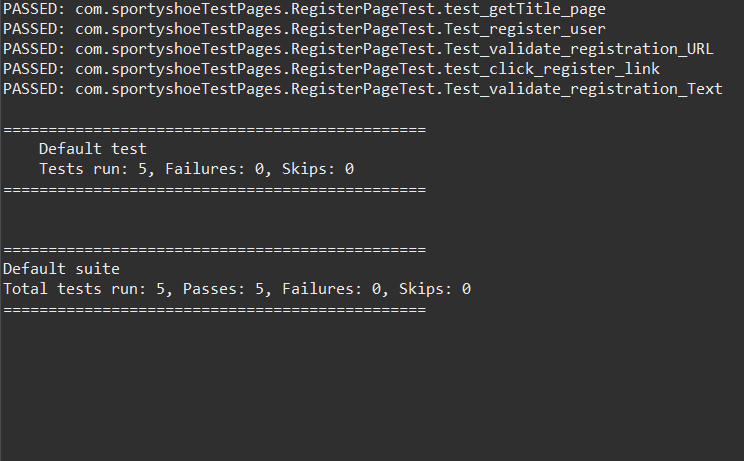
}

}

**Output:-**







**LoginPage**

package com.sportyshoeSeleniumCucumberScripts;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.support.FindBy;

import org.openqa.selenium.support.PageFactory;

public class LoginPage {

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

WebElement loginEmail;

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

WebElement loginpassword;

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

WebElement loginbtn;

@FindBy(linkText="Cart")

WebElement clickCart;

public LoginPage(WebDriver driver) {

PageFactory.initElements(driver, this);

}

public void user\_login()

{

loginEmail.sendKeys("Kesava@gmail.com");

loginpassword.sendKeys("Kesava@123");

loginbtn.click();

}

public void click\_cart()

{

clickCart.click();

}

}

**LoginPageTest**

package com.sportyshoeTestPages;

import org.testng.Assert;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

import com.sportyshoeSeleniumCucumberScripts.HomePage;

import com.sportyshoeSeleniumCucumberScripts.LoginPage;

import com.sportyshoeSeleniumCucumberScripts.RegisterPage;

import com.sportyshoeSeleniumCucumberScripts.TestBase;

public class LoginPageTest extends TestBase {

HomePage hp;

RegisterPage rp;

LoginPage lp;

@BeforeTest

public void start\_browser()

{

openBrowser();

hp = new HomePage(driver);

rp = new RegisterPage(driver);

lp = new LoginPage(driver);

}

@Test(priority='1')

public void test\_login()

{

lp.user\_login();

}

@Test(priority='2')

public void test\_getTitle\_page()

{

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

String Actual = hp.getURL\_page();

Assert.assertEquals(Actual, expected);

}

@Test(priority='3')

public void Test\_validate\_registration\_Text()

{

String expected = "Hello Kesava !";

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

Assert.assertEquals(actualText, expected);

}

@Test(priority='4')

public void test\_click\_cart()

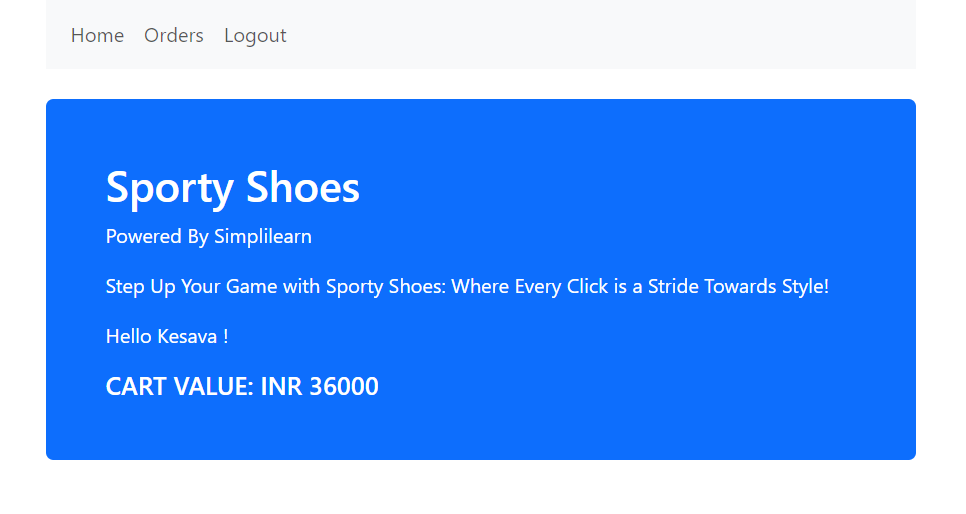
{

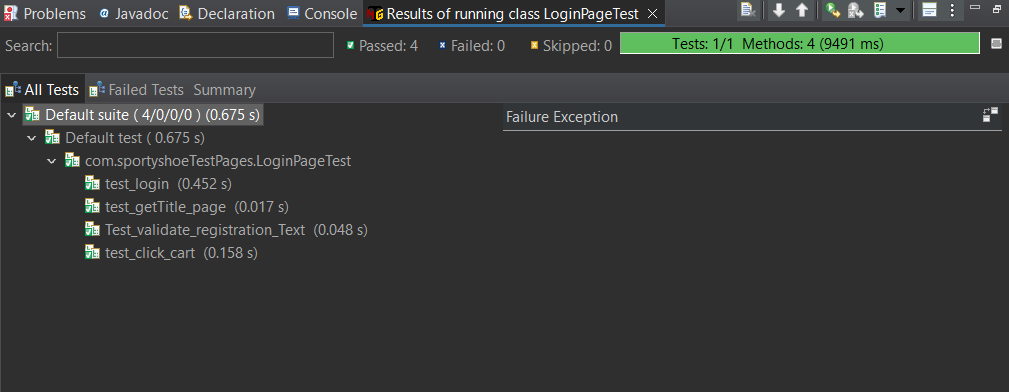
lp.click\_cart();

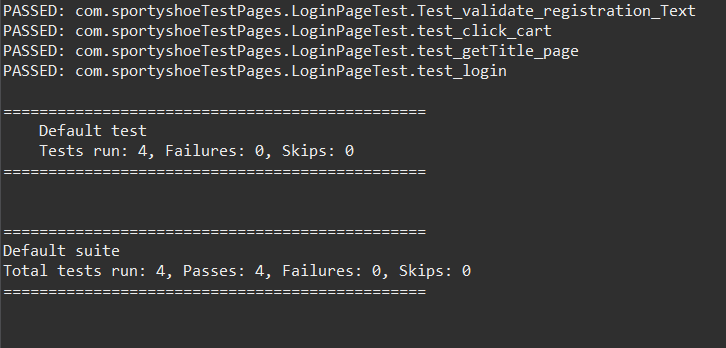
}

}

**Output:-**







**AddtoCart page**

package com.sportyshoeSeleniumCucumberScripts;

import org.openqa.selenium.JavascriptExecutor;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.support.FindBy;

import org.openqa.selenium.support.PageFactory;

public class AddtoCartPage {

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

WebElement viewShoeBTN;

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

WebElement addtocartBTN;

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

WebElement successText;

JavascriptExecutor executor;

public AddtoCartPage(WebDriver driver) {

PageFactory.initElements(driver, this);

executor = (JavascriptExecutor) driver;

}

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

{

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

}

public String validate\_success\_message()

{

String successtext = successText.getText();

return successtext;

}

}

**AddtoCartPageTest**

package com.sportyshoeTestPages;

import org.testng.Assert;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

import com.sportyshoeSeleniumCucumberScripts.AddtoCartPage;

import com.sportyshoeSeleniumCucumberScripts.HomePage;

import com.sportyshoeSeleniumCucumberScripts.LoginPage;

import com.sportyshoeSeleniumCucumberScripts.RegisterPage;

import com.sportyshoeSeleniumCucumberScripts.TestBase;

public class AddtoCartPageTest extends TestBase {

HomePage hp;

RegisterPage rp;

LoginPage lp;

AddtoCartPage ac;

@BeforeTest

public void start\_browser()

{

openBrowser();

hp = new HomePage(driver);

rp = new RegisterPage(driver);

lp = new LoginPage(driver);

ac = new AddtoCartPage(driver);

}

@Test(priority='1')

public void test\_login()

{

lp.user\_login();

}

@Test(priority='2')

public void test\_add\_product\_to\_cart() throws InterruptedException

{

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

}

@Test(priority='3')

public void test\_validate\_success\_message()

{

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

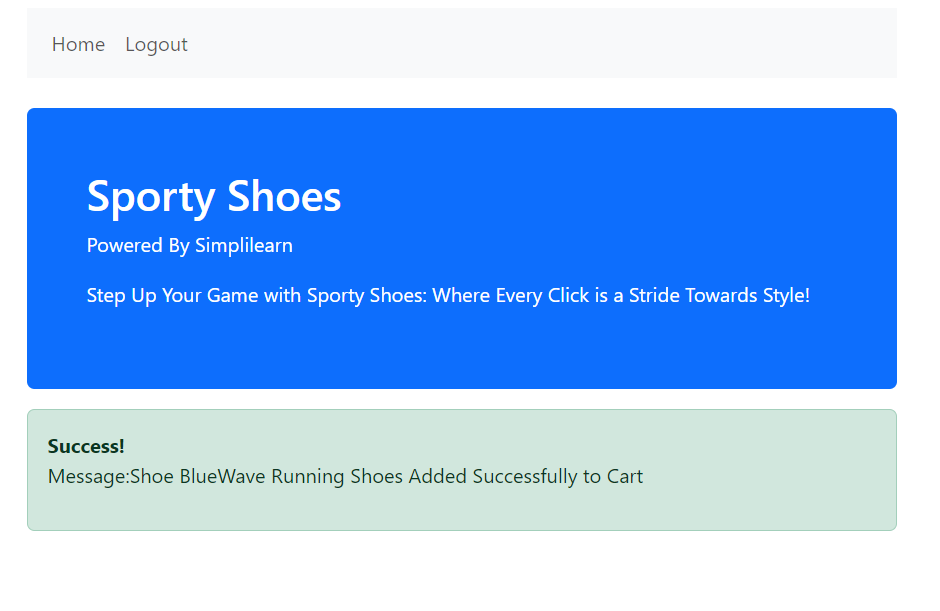
String actualText= ac.validate\_success\_message();

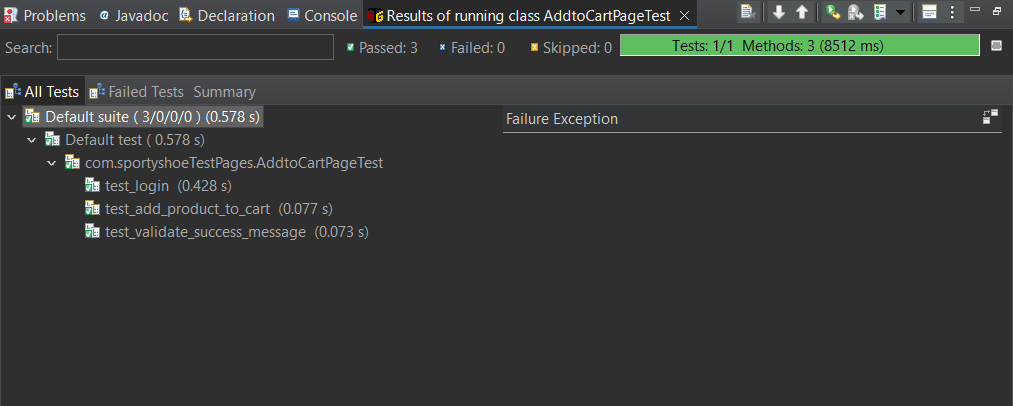
Assert.assertEquals(actualText, expected);

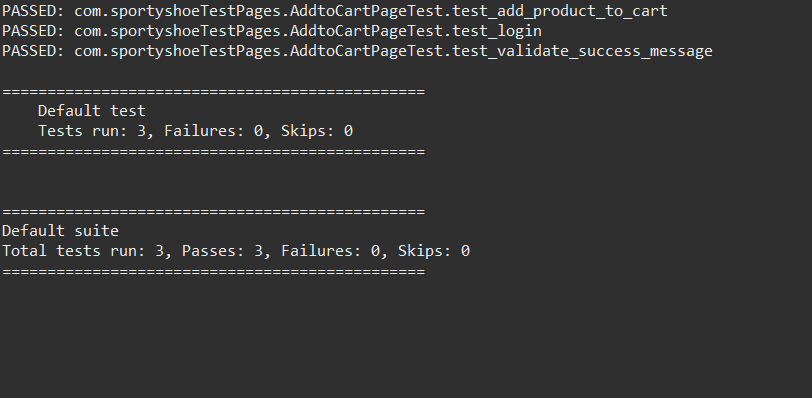
}

}

**Output:-**







**OrderPage**

package com.sportyshoeSeleniumCucumberScripts;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.support.FindBy;

import org.openqa.selenium.support.PageFactory;

public class OrderPage {

@FindBy(linkText="Orders")

WebElement orderlink;

public OrderPage(WebDriver driver) {

PageFactory.initElements(driver, this);

}

public void click\_orderPage()

{

orderlink.click();

}

}

**OrderPageTest**

package com.sportyshoeTestPages;

import org.testng.Assert;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

import com.sportyshoeSeleniumCucumberScripts.HomePage;

import com.sportyshoeSeleniumCucumberScripts.LoginPage;

import com.sportyshoeSeleniumCucumberScripts.OrderPage;

import com.sportyshoeSeleniumCucumberScripts.RegisterPage;

import com.sportyshoeSeleniumCucumberScripts.TestBase;

public class OrderPageTest extends TestBase {

HomePage hp;

RegisterPage rp;

LoginPage lp;

OrderPage op;

@BeforeTest

public void start\_browser()

{

openBrowser();

hp = new HomePage(driver);

rp = new RegisterPage(driver);

lp = new LoginPage(driver);

op = new OrderPage(driver);

}

@Test(priority='1')

public void test\_login()

{

lp.user\_login();

}

@Test(priority='2')

public void test\_click\_orders()

{

op.click\_orderPage();

}

@Test(priority='3')

public void test\_getTitle\_page()

{

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

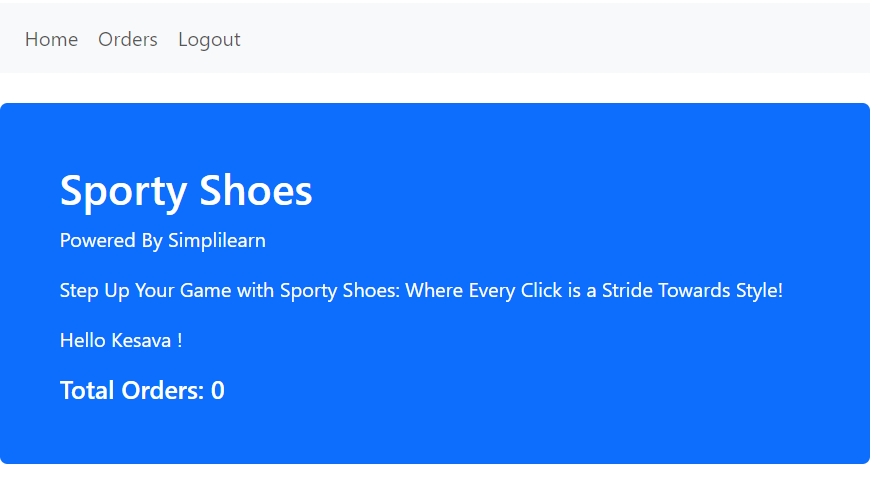
String Actual = hp.getURL\_page();

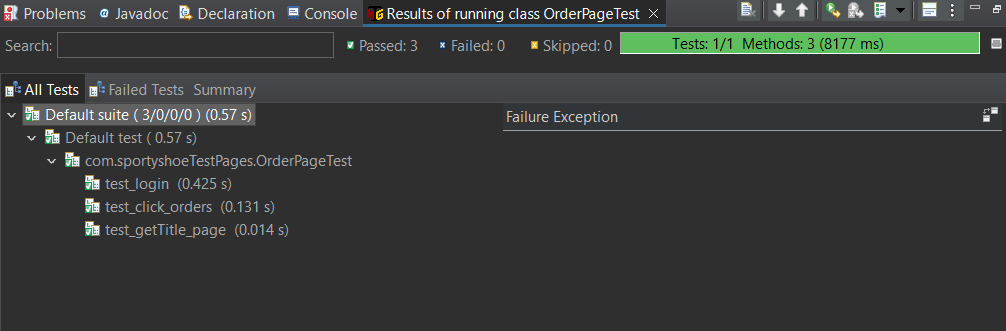
Assert.assertEquals(Actual, expected);

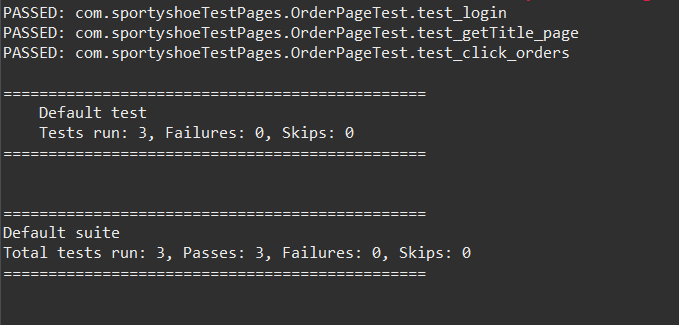
}

}

**Output:-**



****

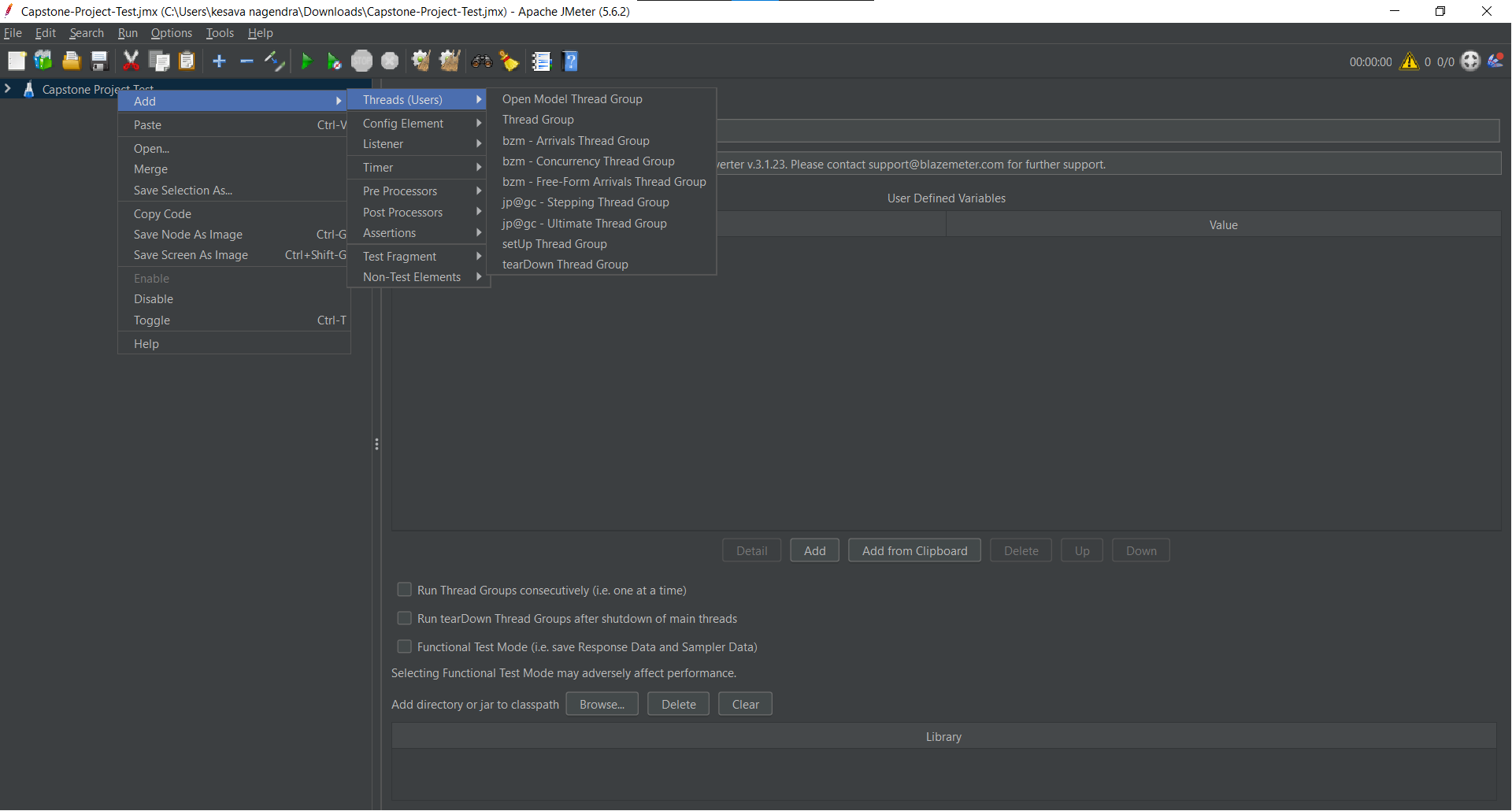
****

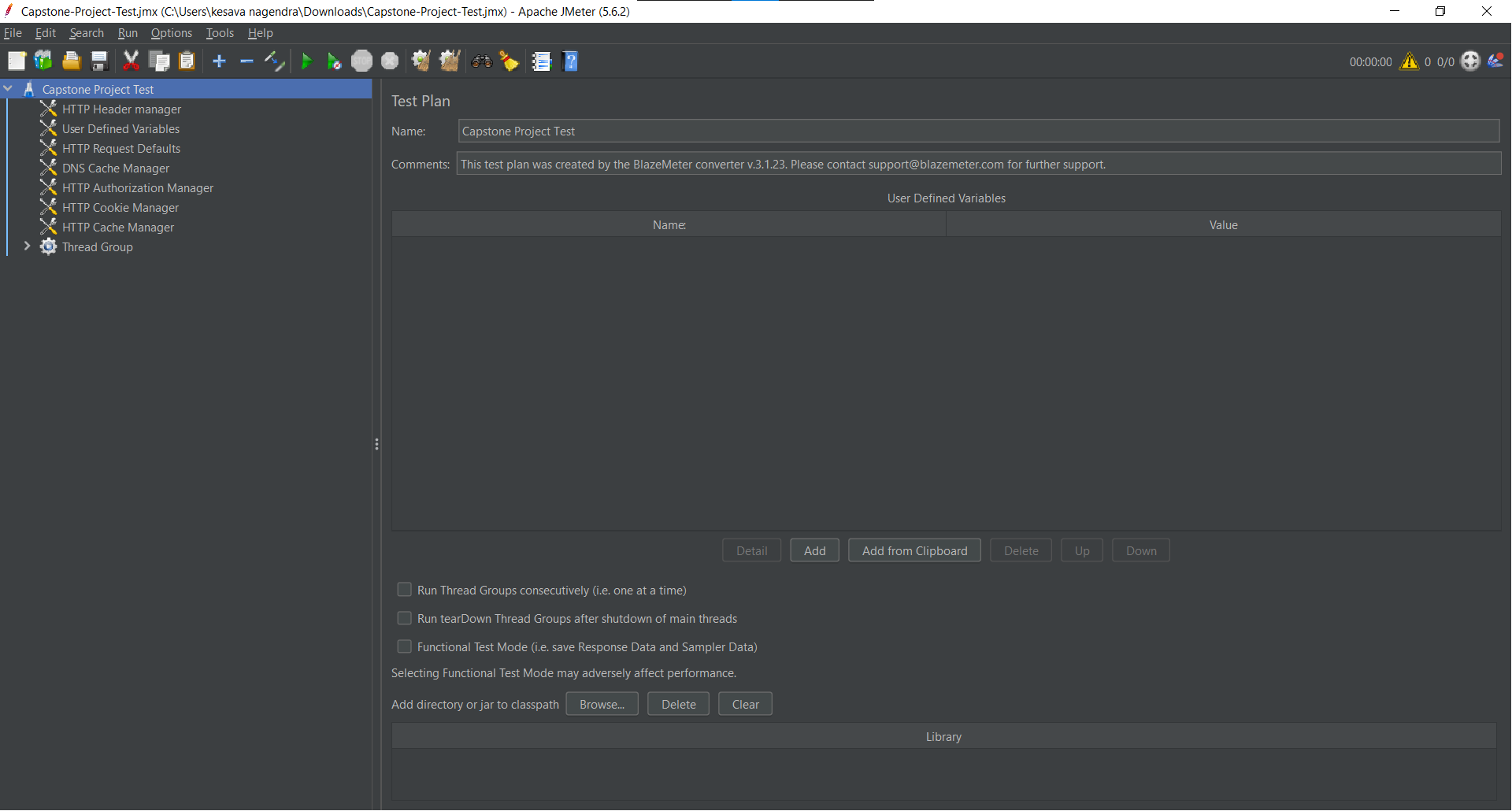
**JMeter Scripts**



In the Test plan , added the Thread Group

Right click on Test plan->Add->Thread Group

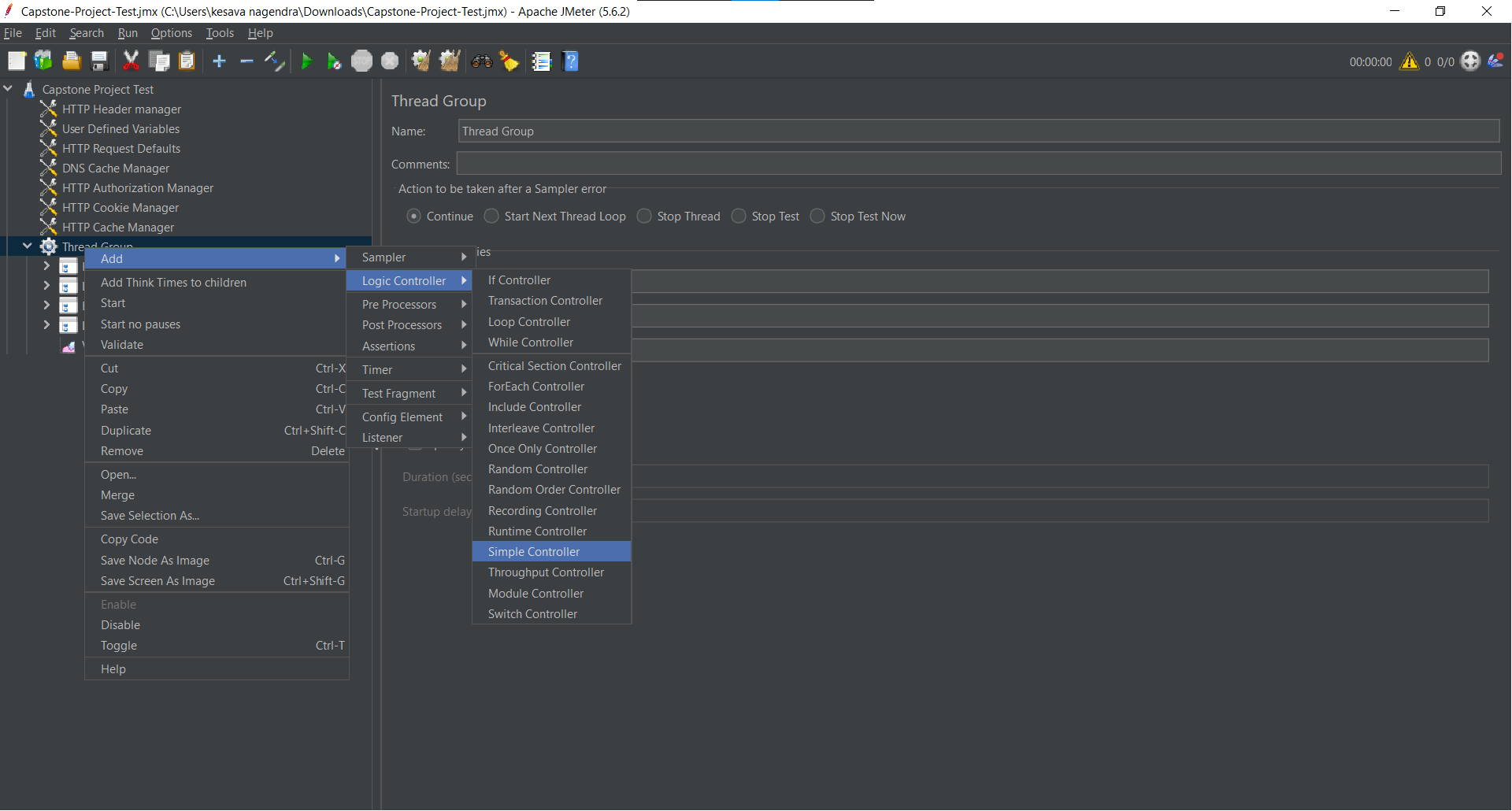


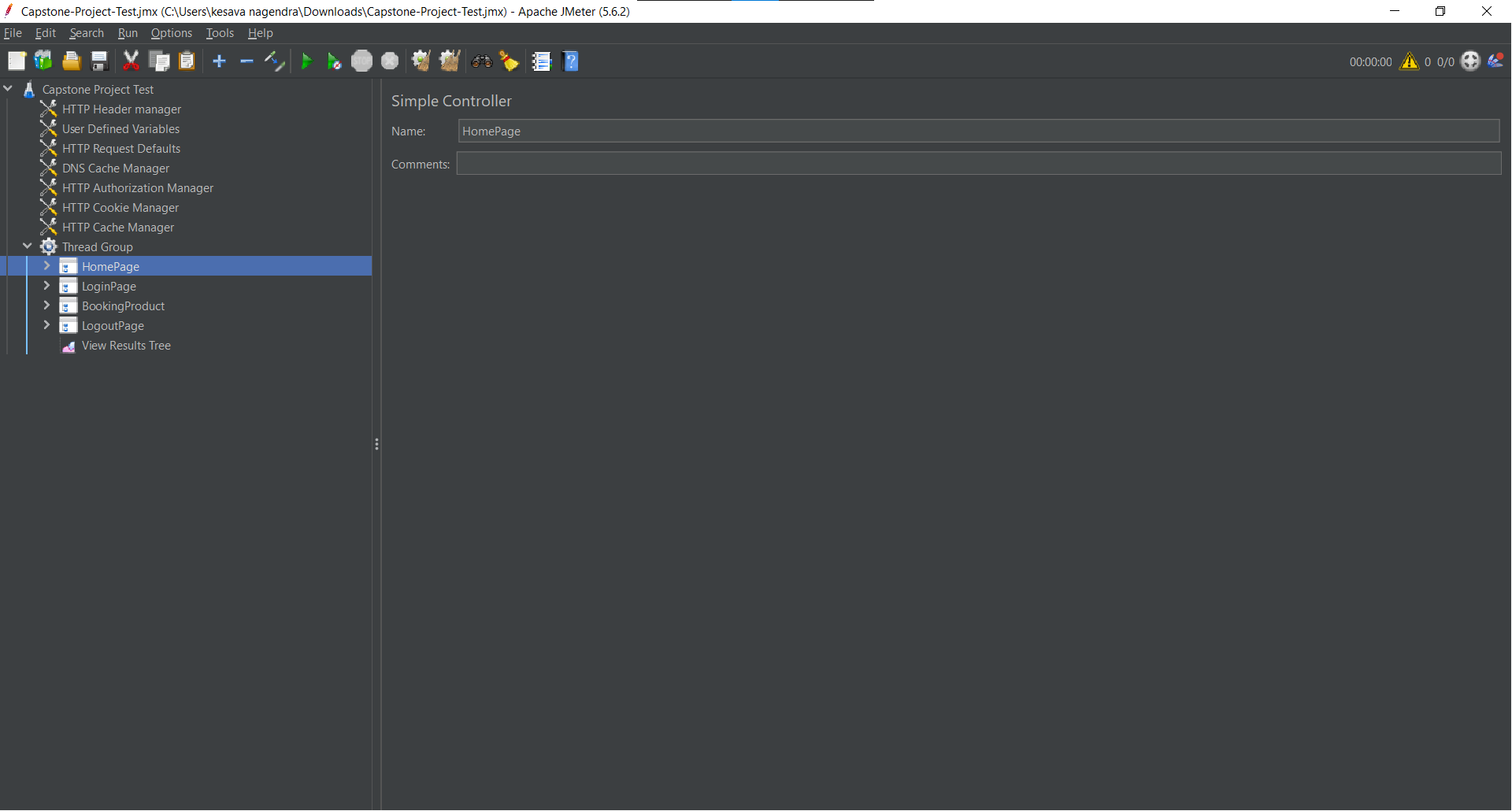


By using Blaze Meter recorder recorded the scripts.

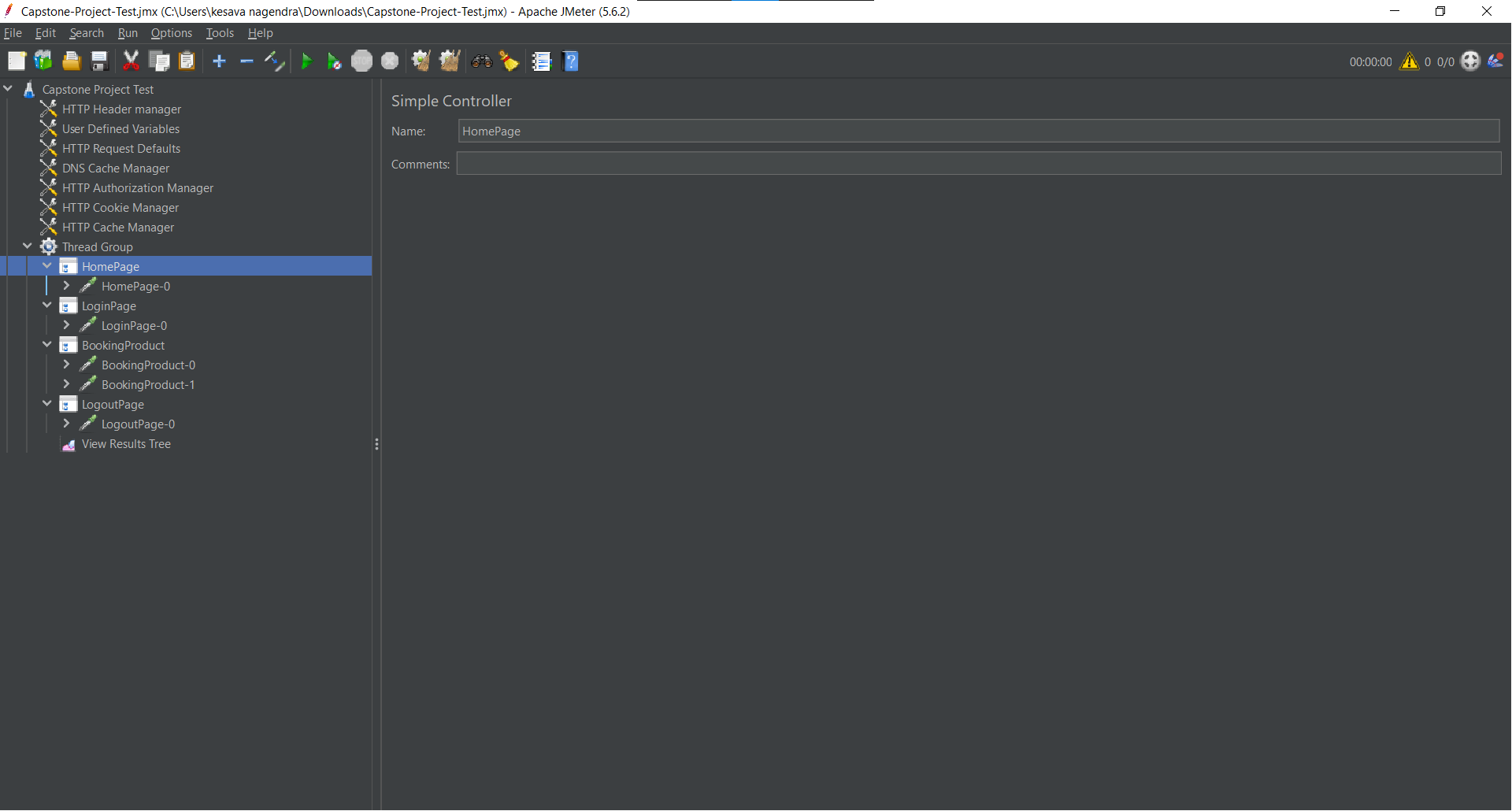
And then added the four Simple controller.

Right click on Thread Group->Logic Controller->Simple Controller



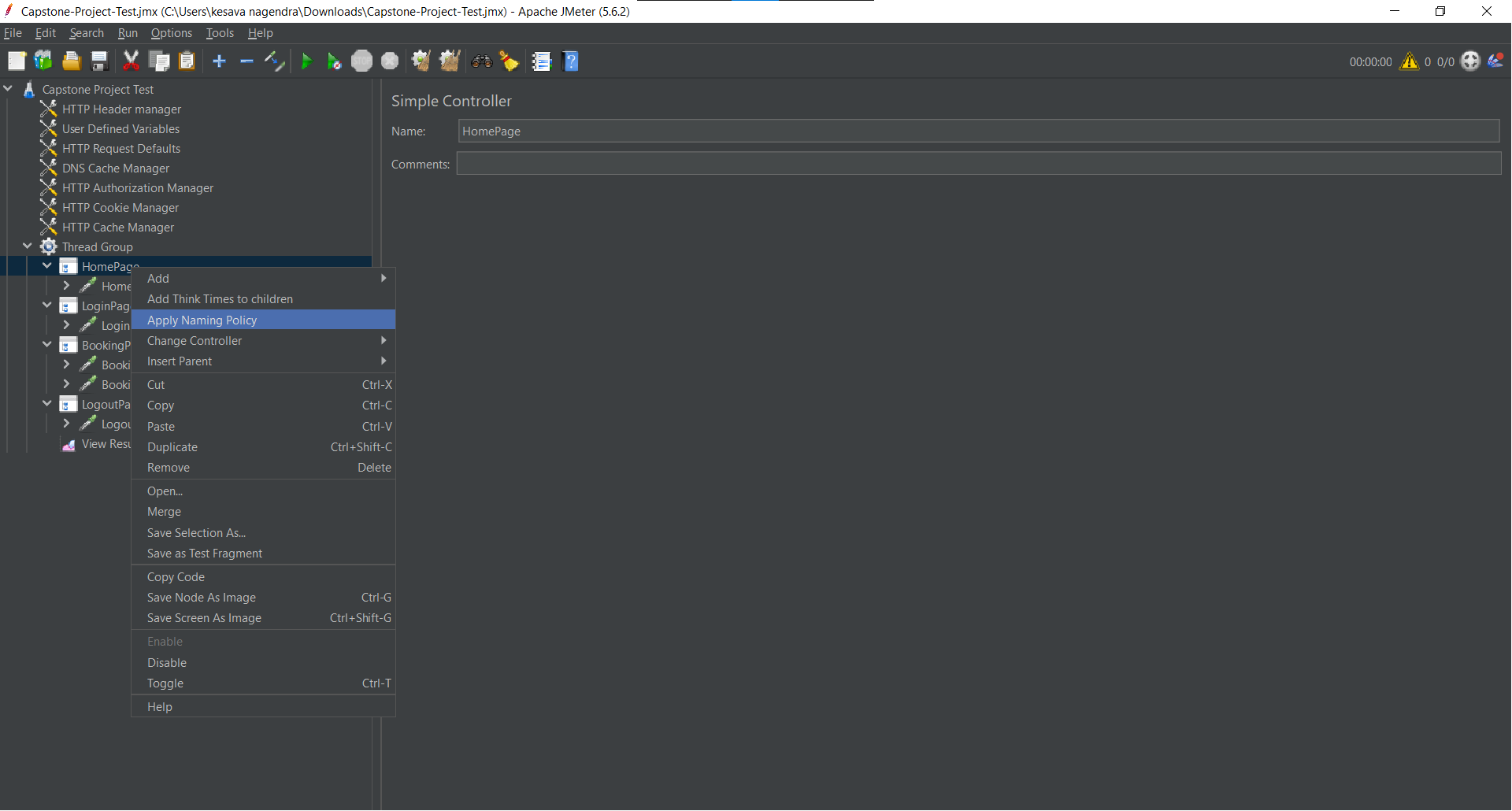


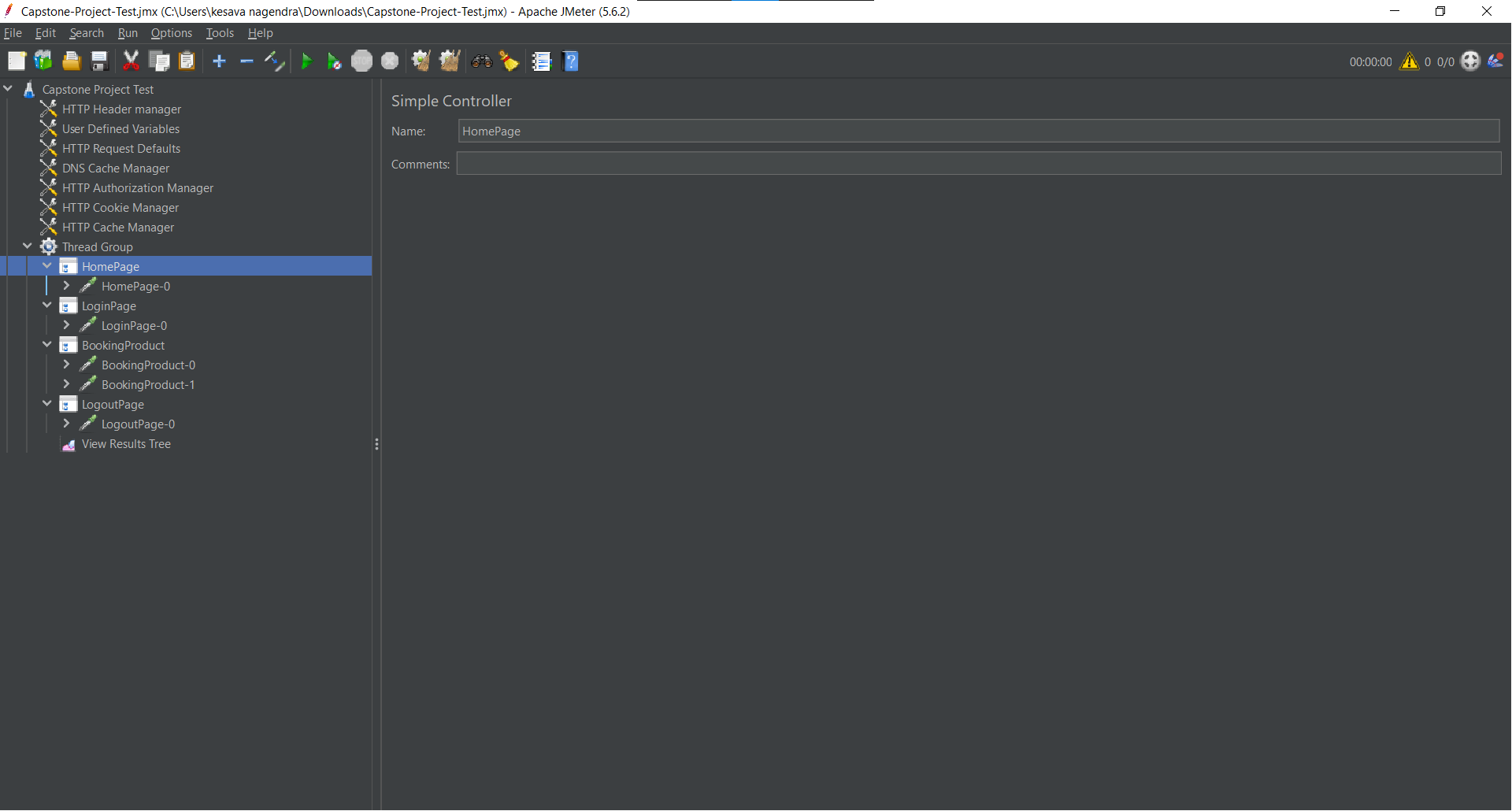
Arranged the requests based on action in simple controller.



Applied the naming policy to the each request based on the Controller name.

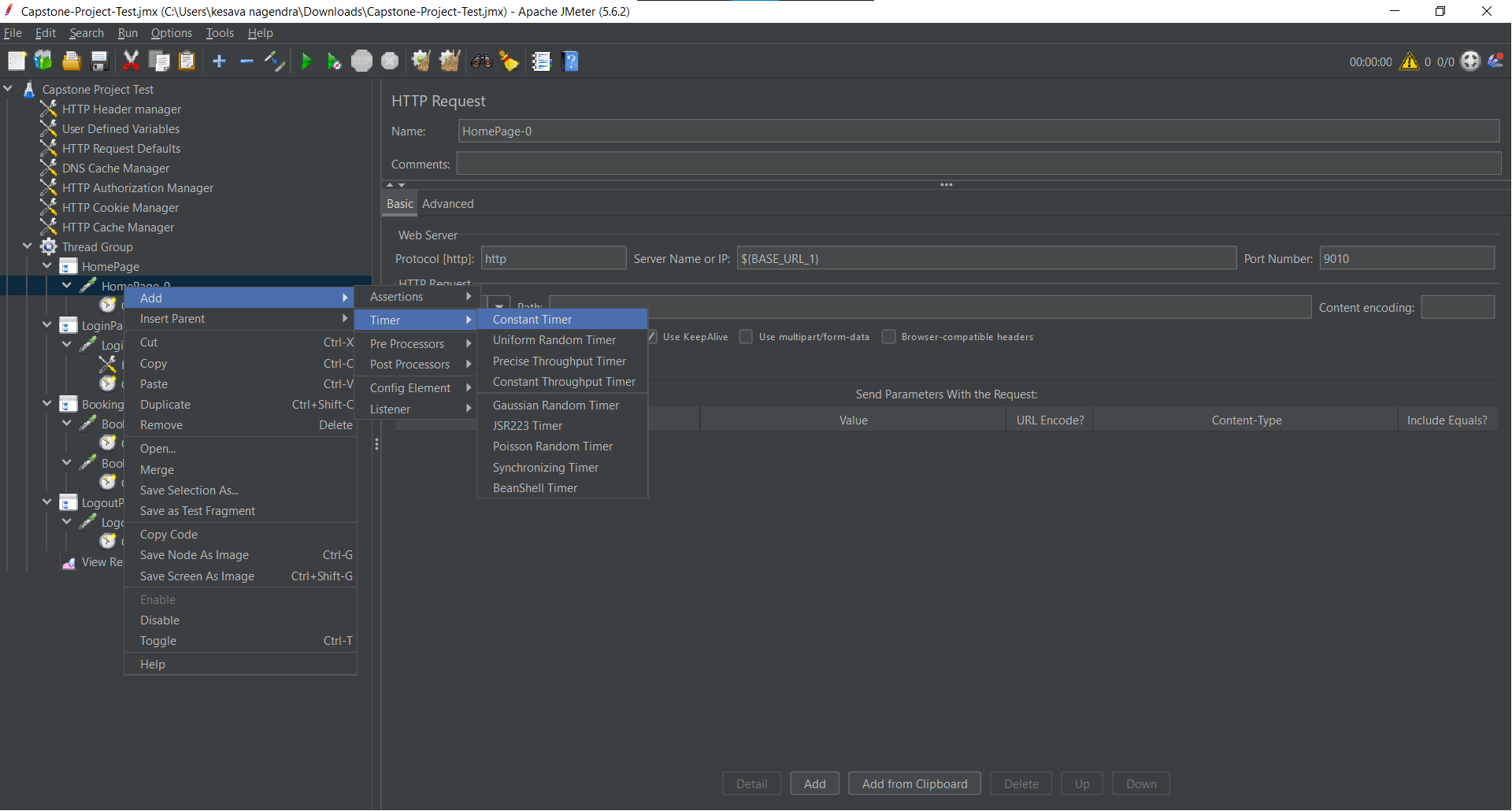
Right click on controller->Apply Naming Policy



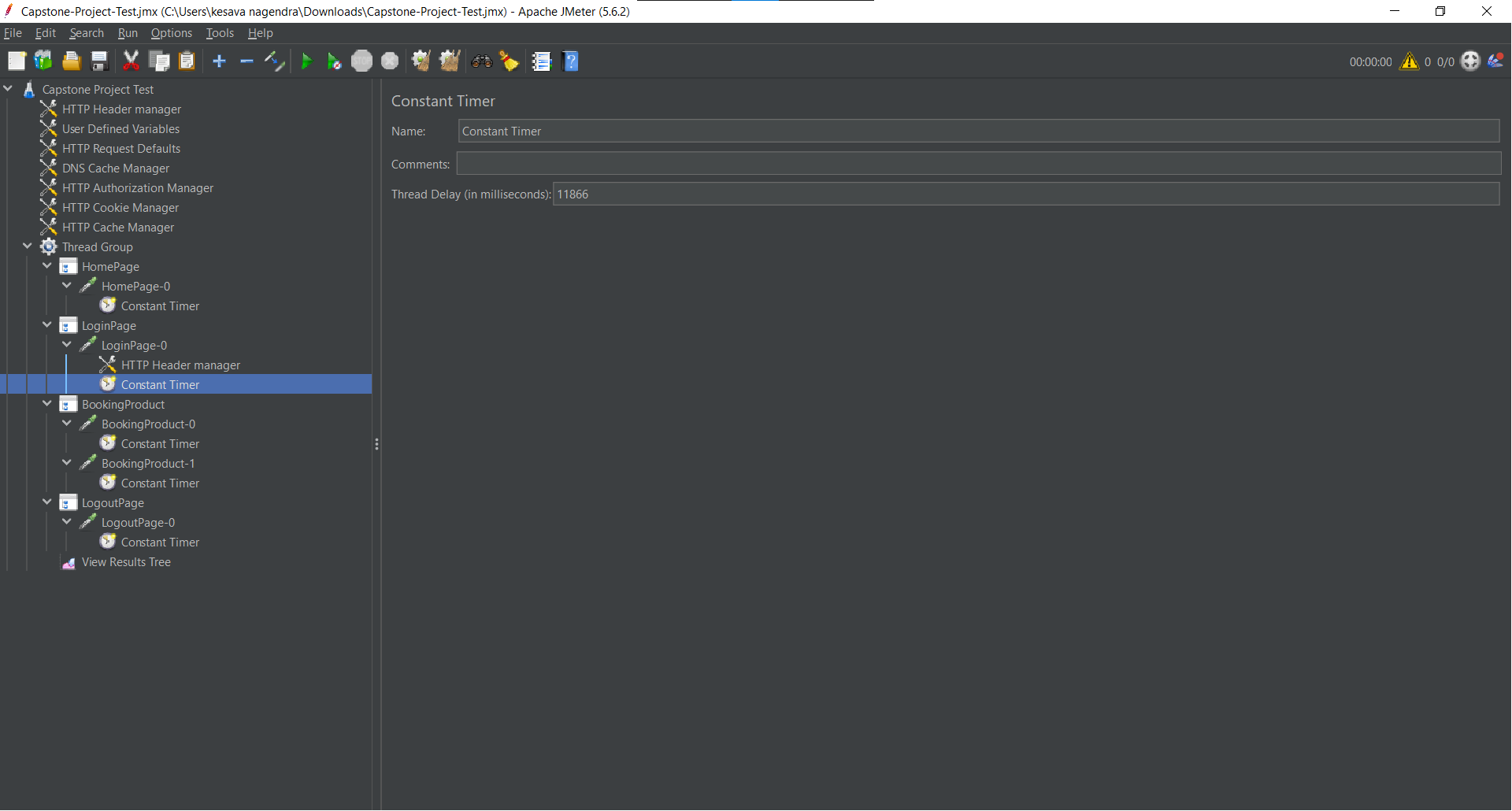


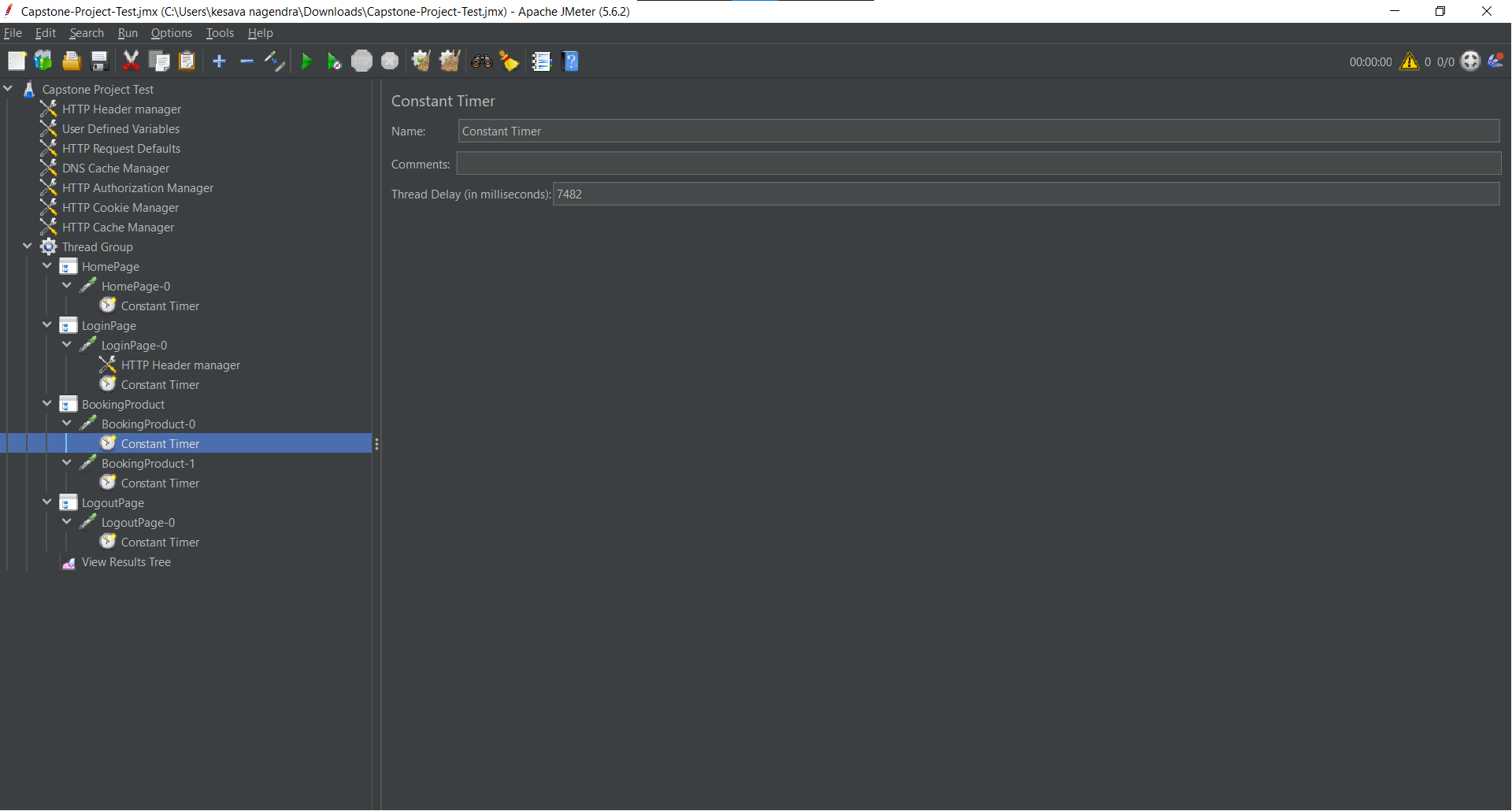
And then added the Constant Timer for each request in every controller.

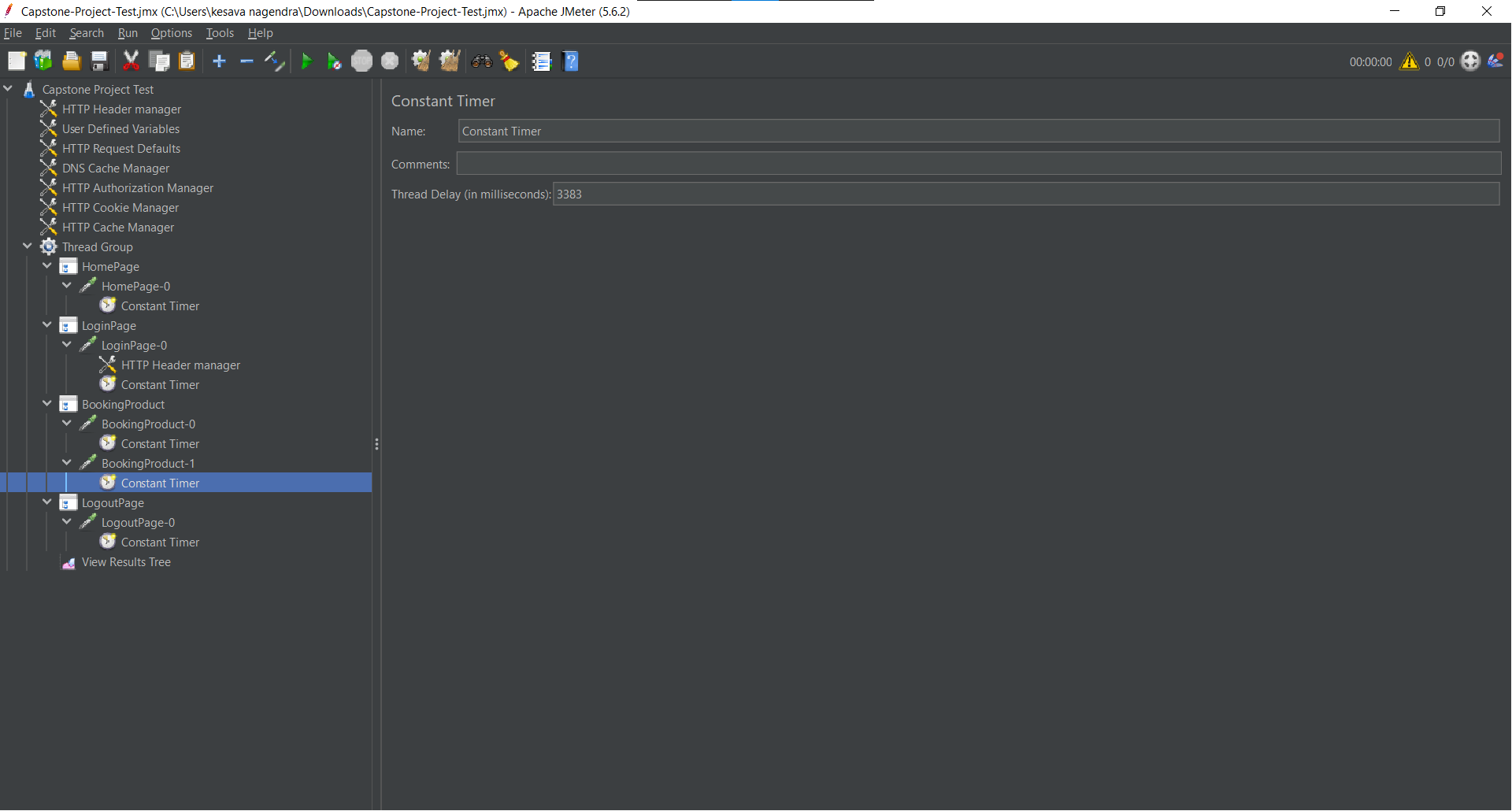
Right click on Request->Add->Timer->Constant Timer

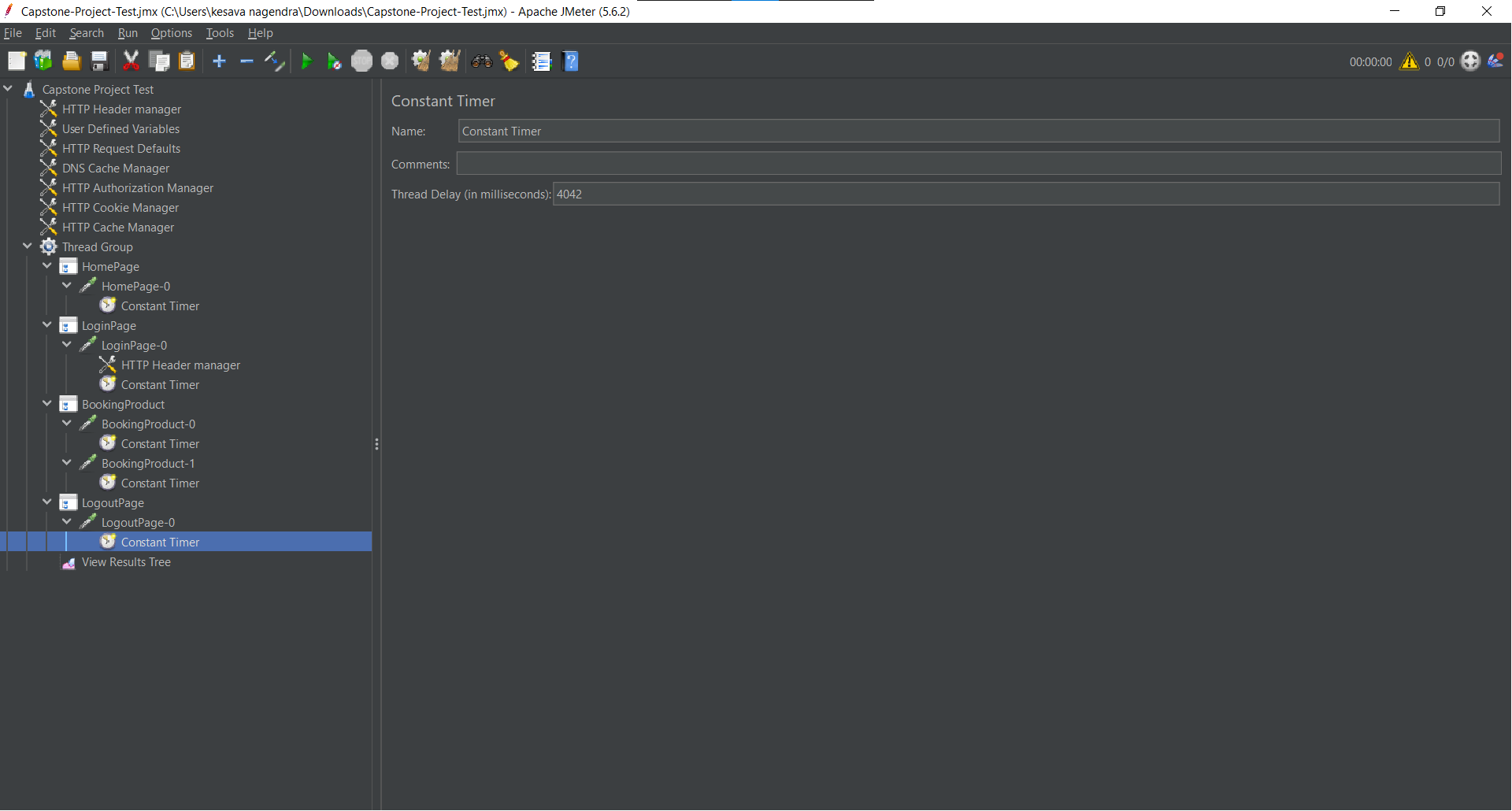


Added the some Thread Delay in every controller.



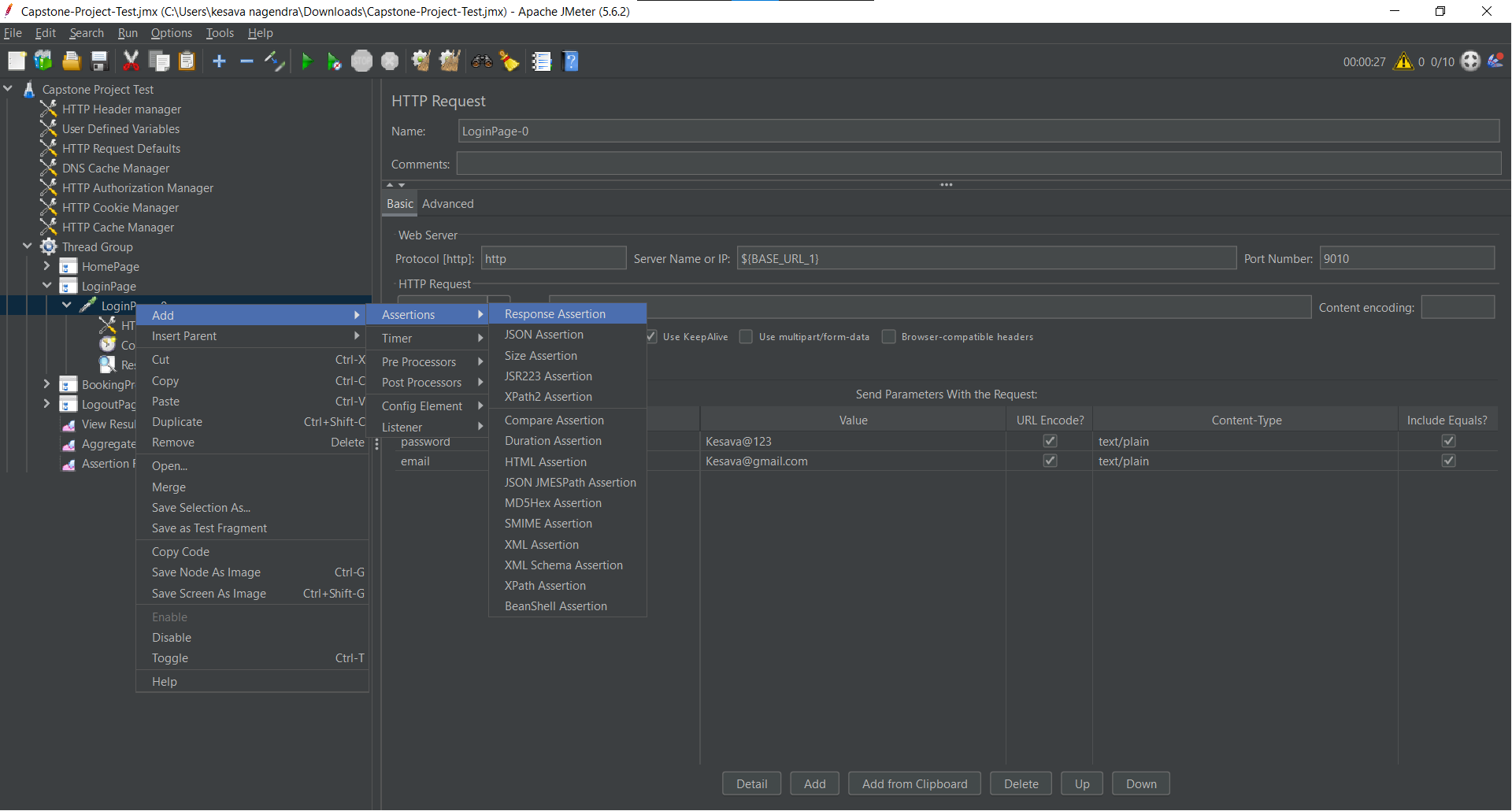


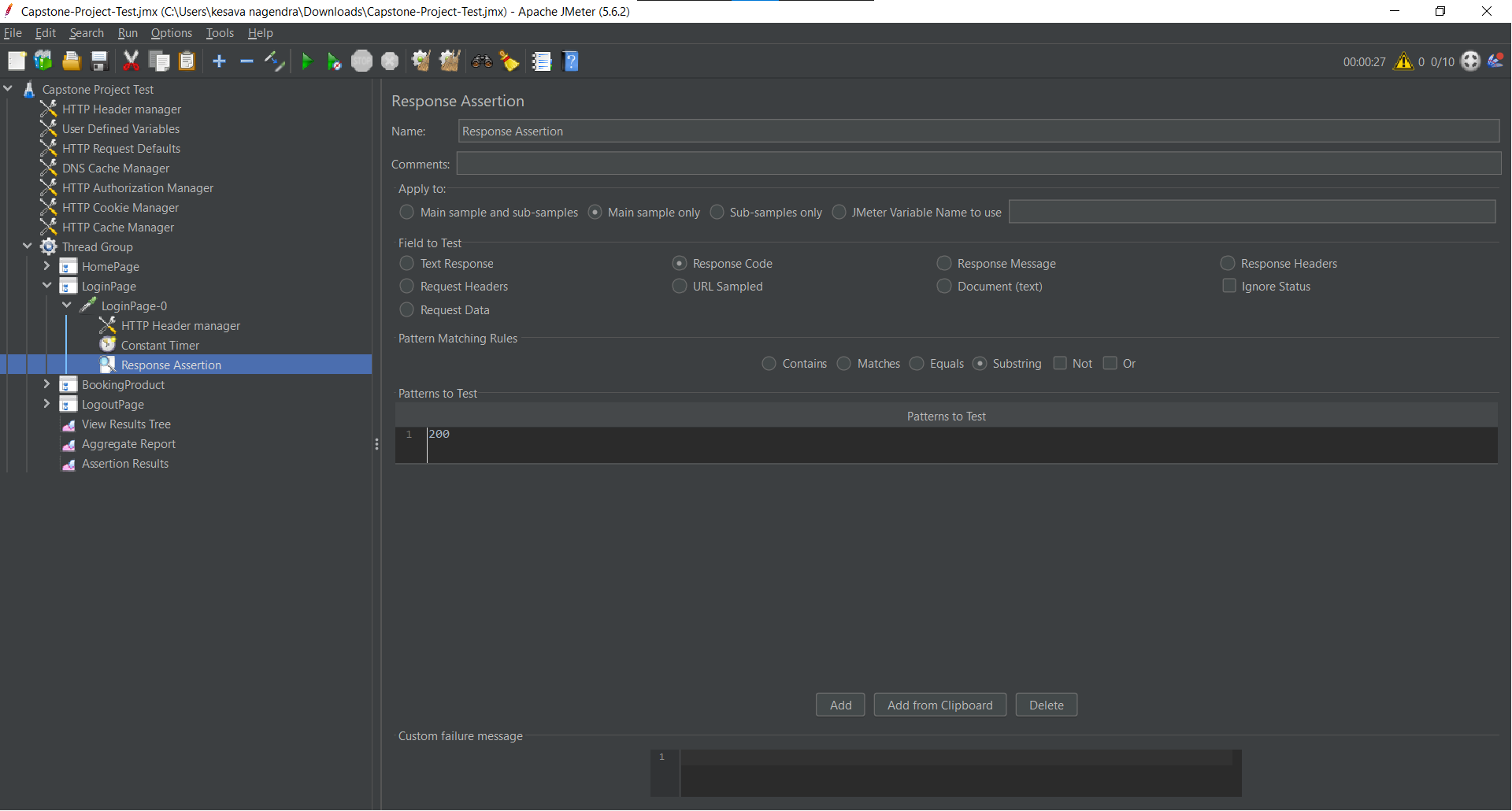




Added the Assertion to check the response code is 200 or not.

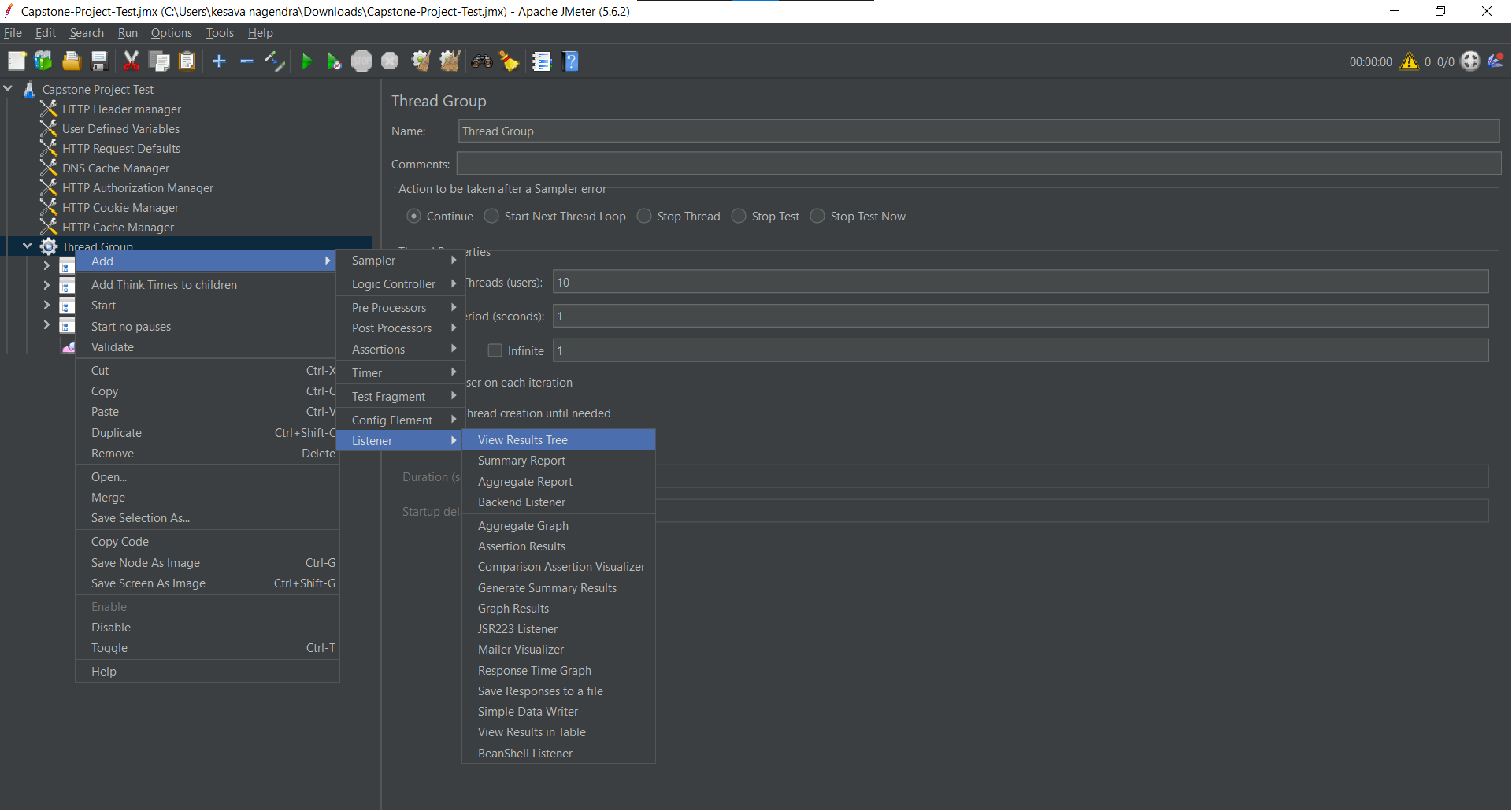
Right click on Request->Add->Assertions->Response Assertion



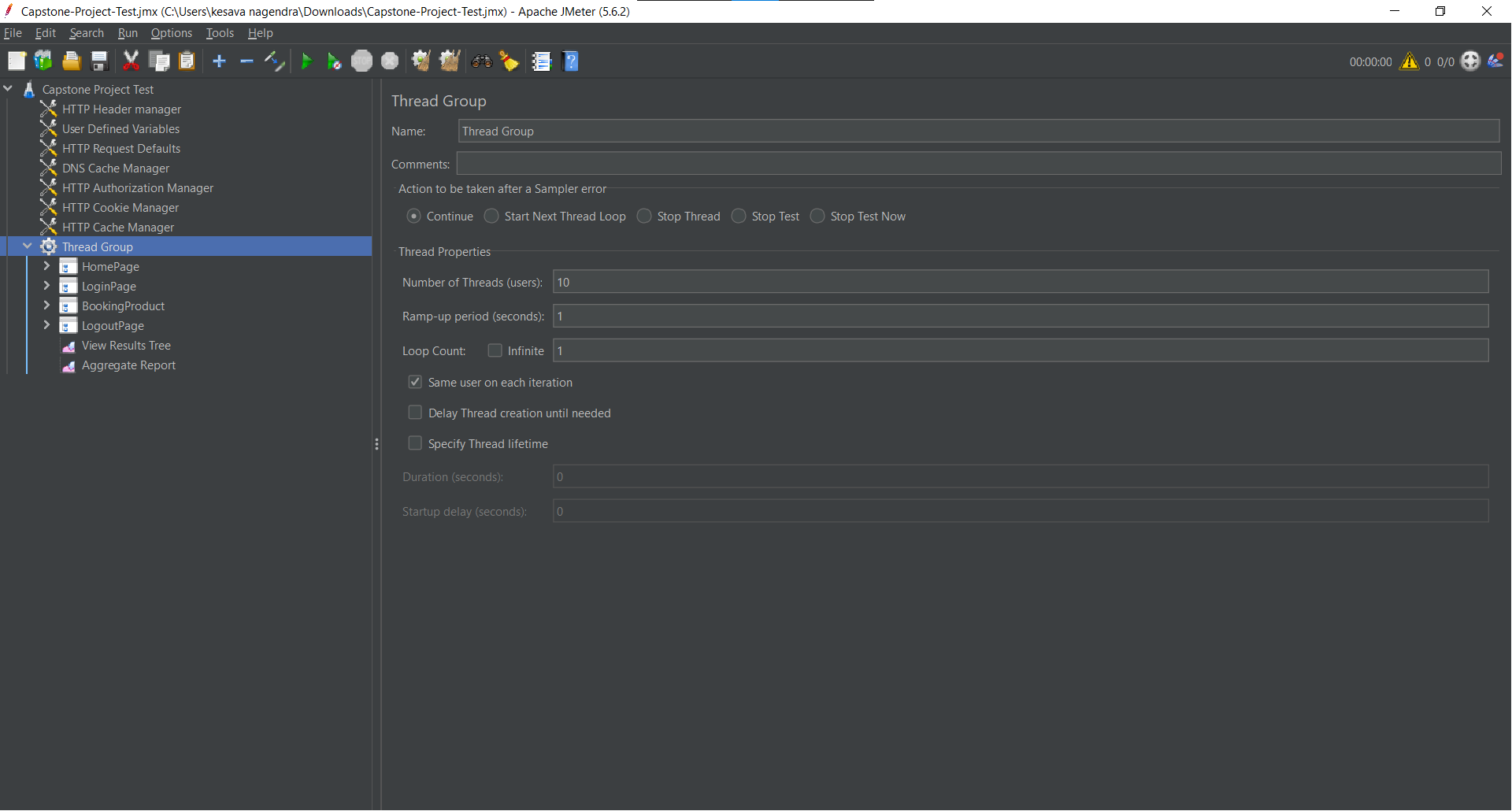


And then added the Listener to the Thread Group

Right click on Thread Group->Add->Listeners->View Result Tree and Aggregate Report

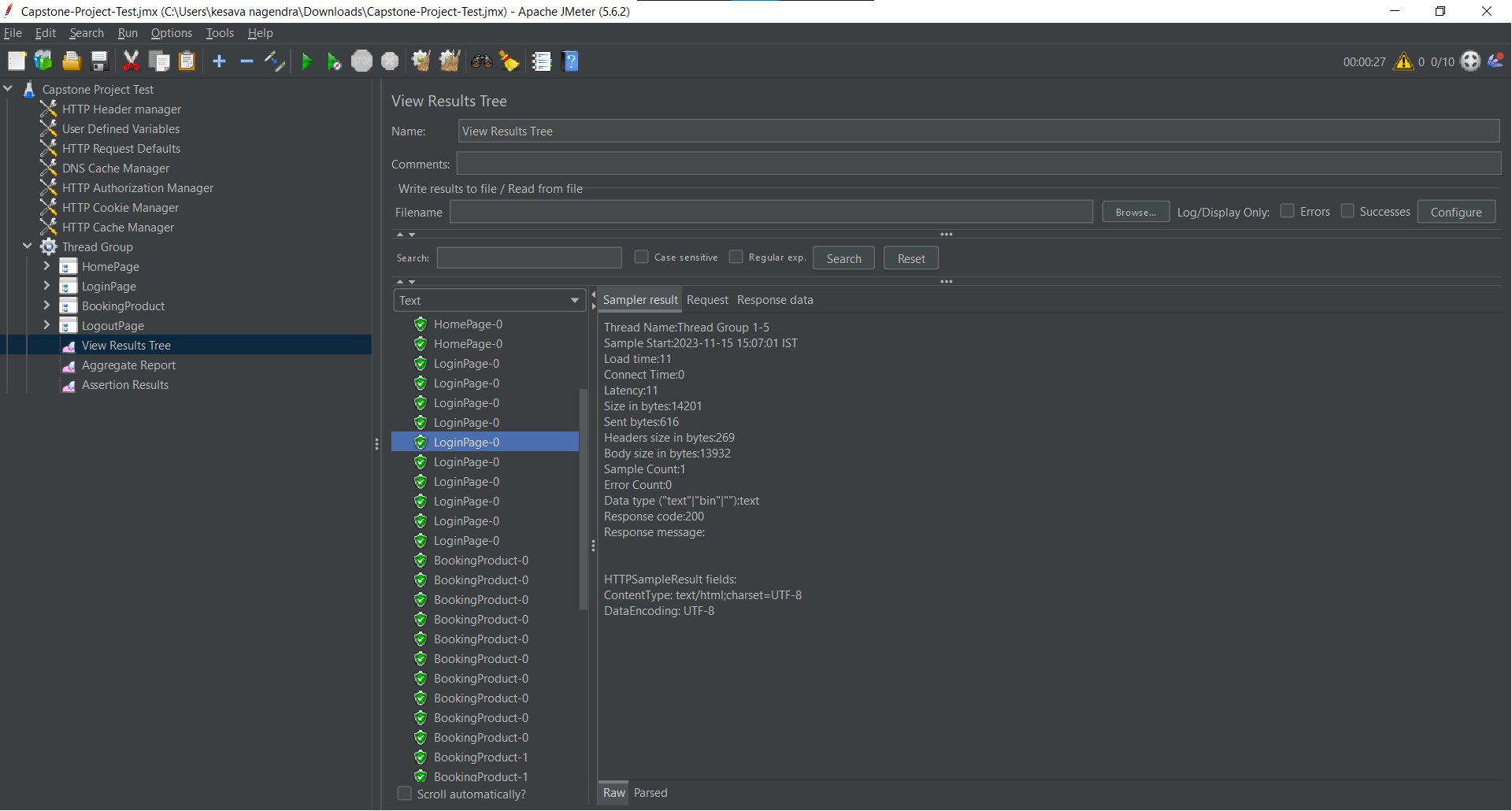


Added the load in the Thread Group by increasing the users.

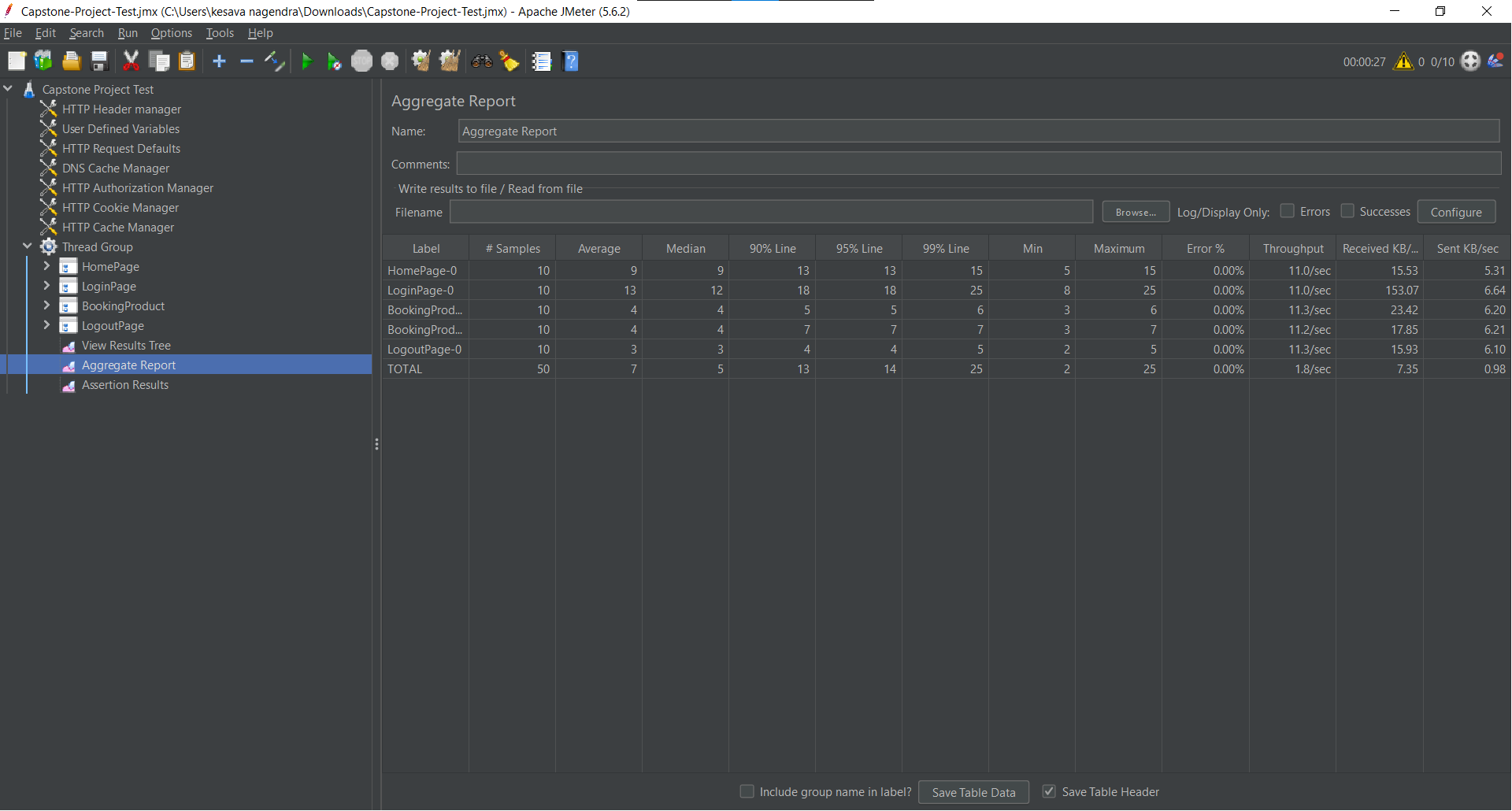


**Output:-**

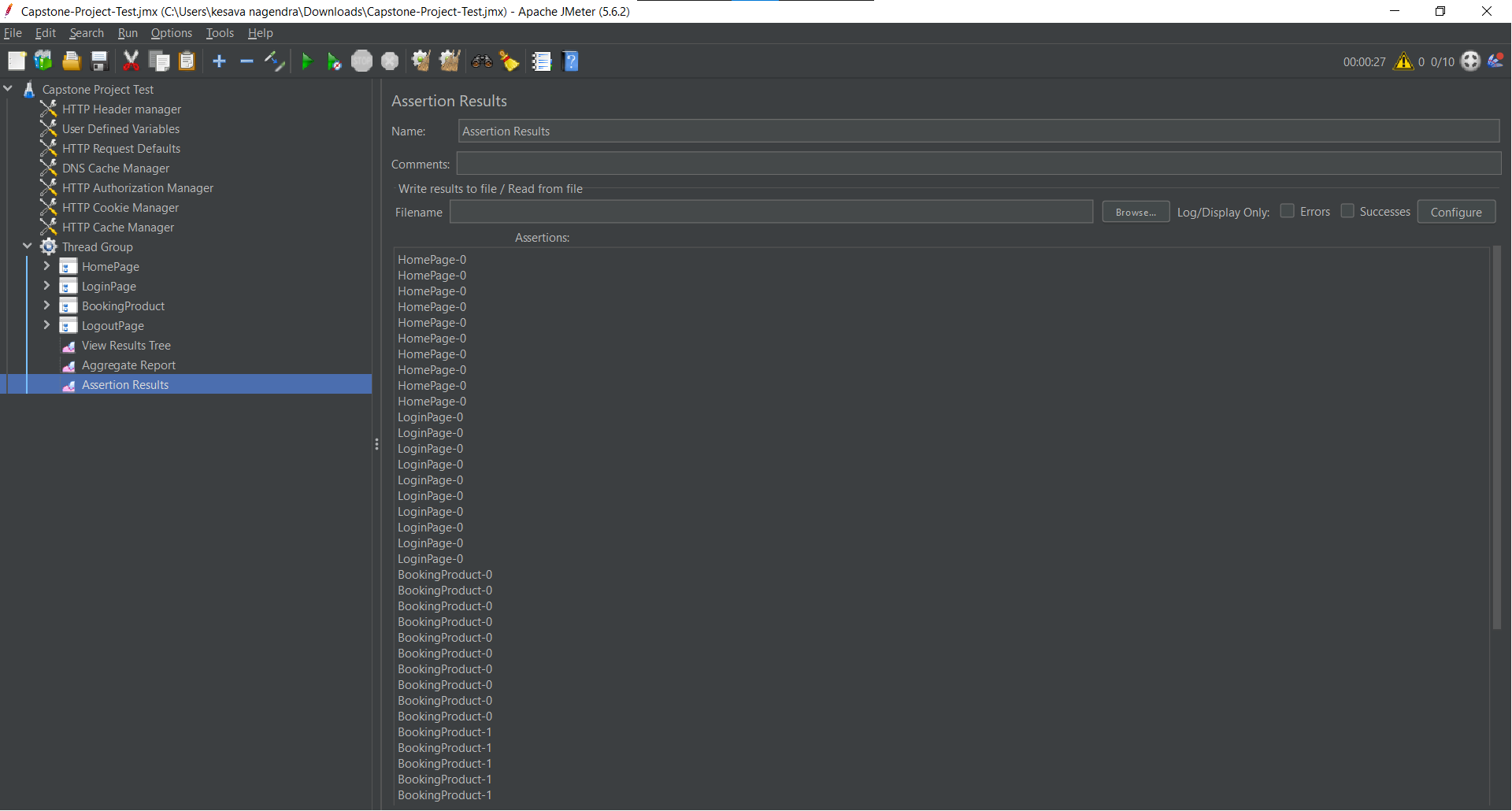
**View Result Tree:-**



**Aggregate Report**



**Assertion Results**

****

**Cucumber with Rest Assured**

**Feature file**

Feature: Implement the Capstone project on Sporty shoe website

Scenario: Rest API Testing on Sporty shoe website

Given User sends a Get Request to retrieve the list of products

Given User sends a Get Request to retrieve the all registered users

Given User sends a Post Request to add the product.

Given User sends a Put Request to update the product details

Given User sends a Delete Request to delete the product

**Step Definition file**

package steps;

import org.hamcrest.Matchers;

import io.cucumber.java.en.Given;

import io.restassured.RestAssured;

public class CapstoneProjectSteps {

@Given("User sends a Get Request to retrieve the list of products")

public void user\_sends\_a\_get\_request\_to\_retrieve\_the\_list\_of\_products() {

RestAssured.given()

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

.basePath("/get-shoes")

.when().get()

.then().statusCode(200)

.body("shoes.id[0]", Matchers.equalTo(101))

.log().all();

}

@Given("User sends a Get Request to retrieve the all registered users")

public void user\_sends\_a\_get\_request\_to\_retrieve\_the\_all\_registered\_users() {

RestAssured.given()

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

.basePath("/get-users")

.when().get()

.then().statusCode(200)

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

.log().all();

}

@Given("User sends a Post Request to add the product.")

public void user\_sends\_a\_post\_request\_to\_add\_the\_product() {

RestAssured.given()

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

.basePath("/add-shoe")

.queryParam("id", "1103")

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

.queryParam("name", "Nike")

.queryParam("category", "Running")

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

.queryParam("price", "1000")

.when().post()

.then().statusCode(200)

.body("shoe", Matchers.hasEntry("category","Running"))

.log().all();

}

@Given("User sends a Put Request to update the product details")

public void user\_sends\_a\_put\_request\_to\_update\_the\_product\_details() {

RestAssured.given()

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

.basePath("/update-shoe")

.queryParam("id", "1103")

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

.queryParam("name", "Puma")

.queryParam("category", "Sports")

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

.queryParam("price", "1500")

.when().put()

.then().statusCode(200)

.body("shoe.name", Matchers.equalTo("Puma"))

.body("shoe.sizes" ,Matchers.equalTo("5,6,7"))

.log().all();

}

@Given("User sends a Delete Request to delete the product")

public void user\_sends\_a\_delete\_request\_to\_delete\_the\_product() {

RestAssured.given()

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

.basePath("/delete-shoe")

.queryParam("id", "1103")

.when().delete()

.then().statusCode(200)

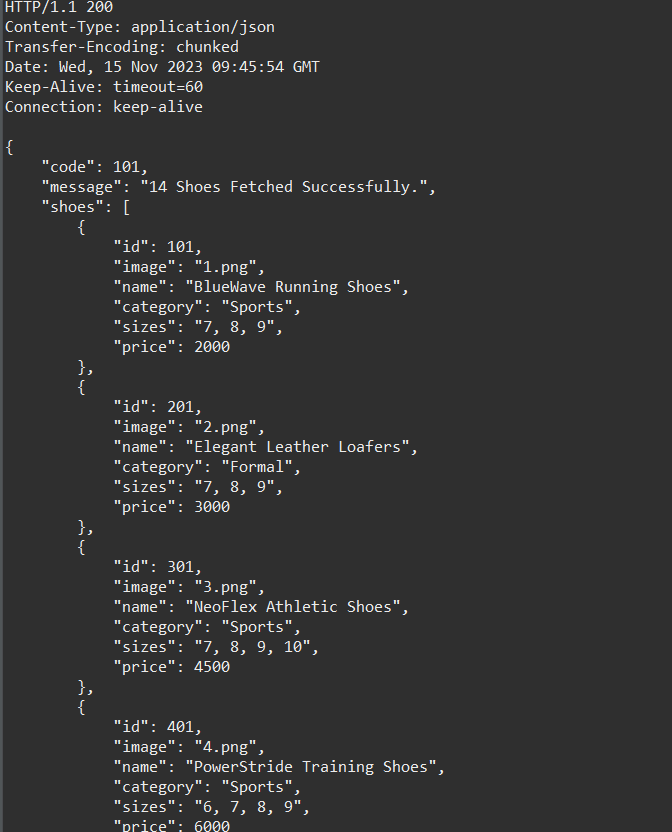
.body("message", Matchers.equalTo("Shoe with ID 1103 Deleted Successfully."))

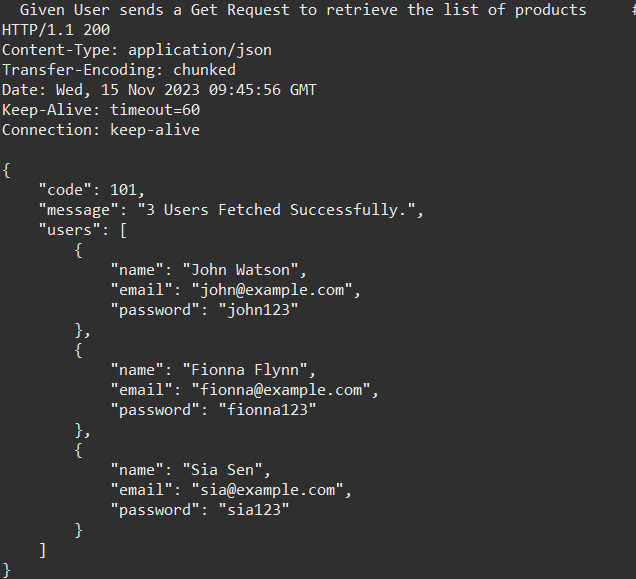
.log().all();

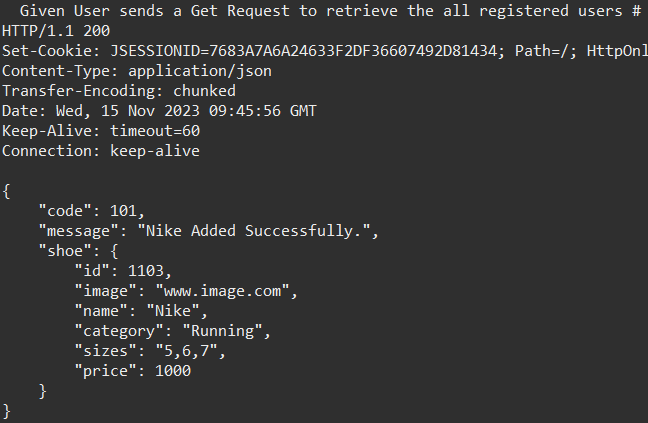
}

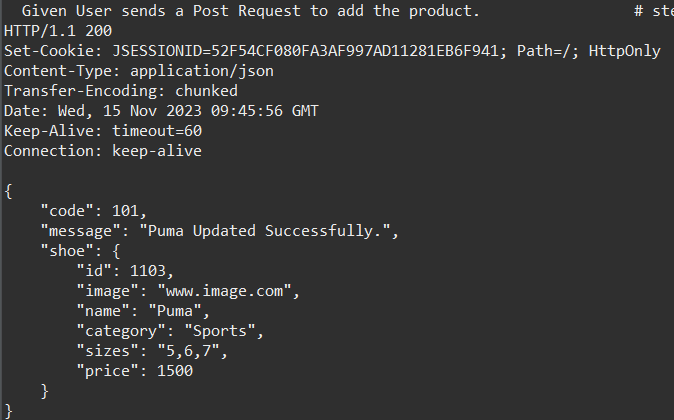
}

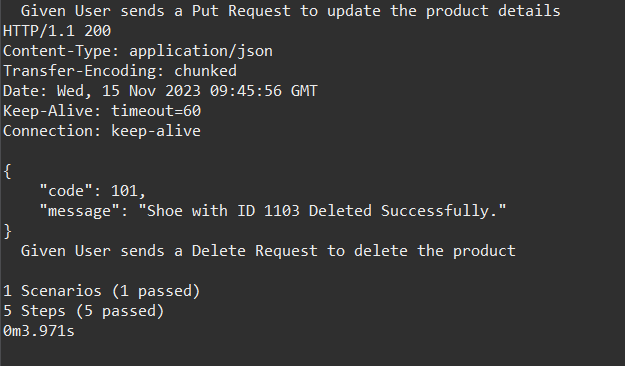
**Output:-**





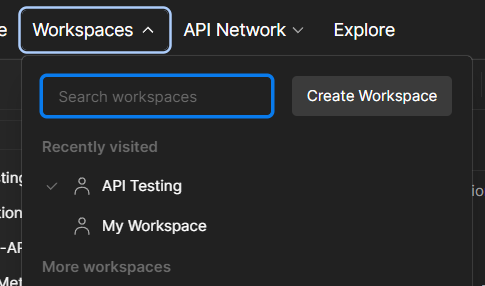




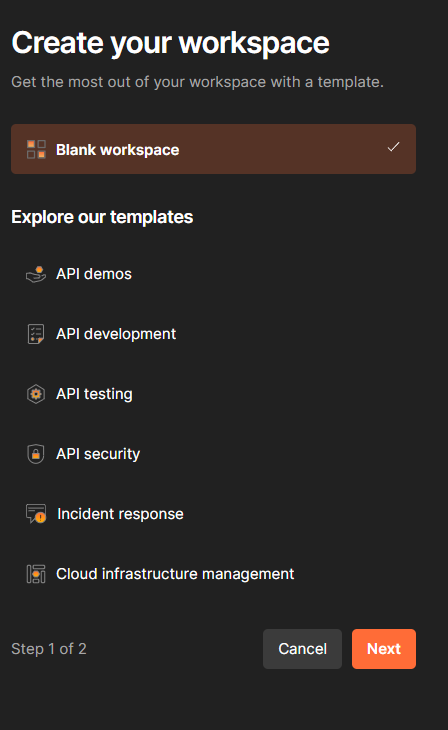


**Postman**

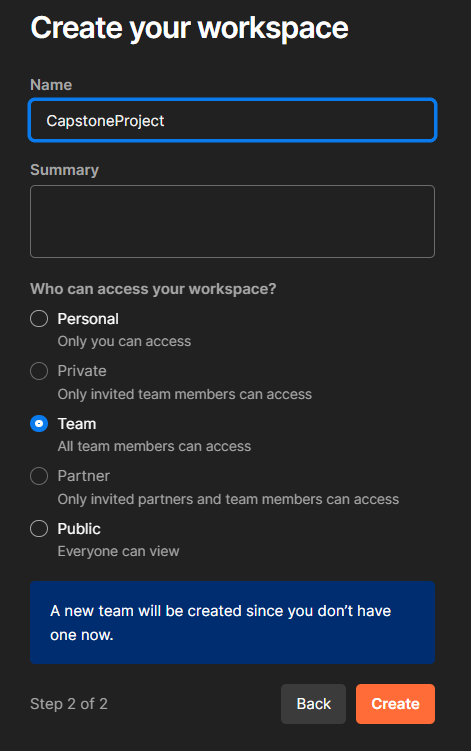
Click on workspace at top header section and create workspace



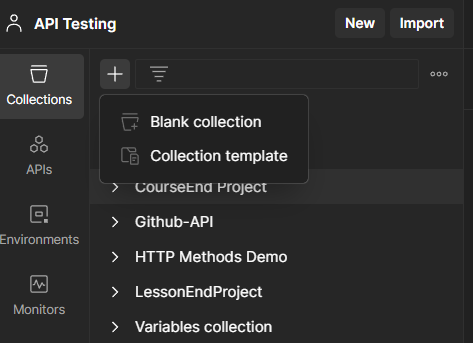
Select the Blank workspace and click on next

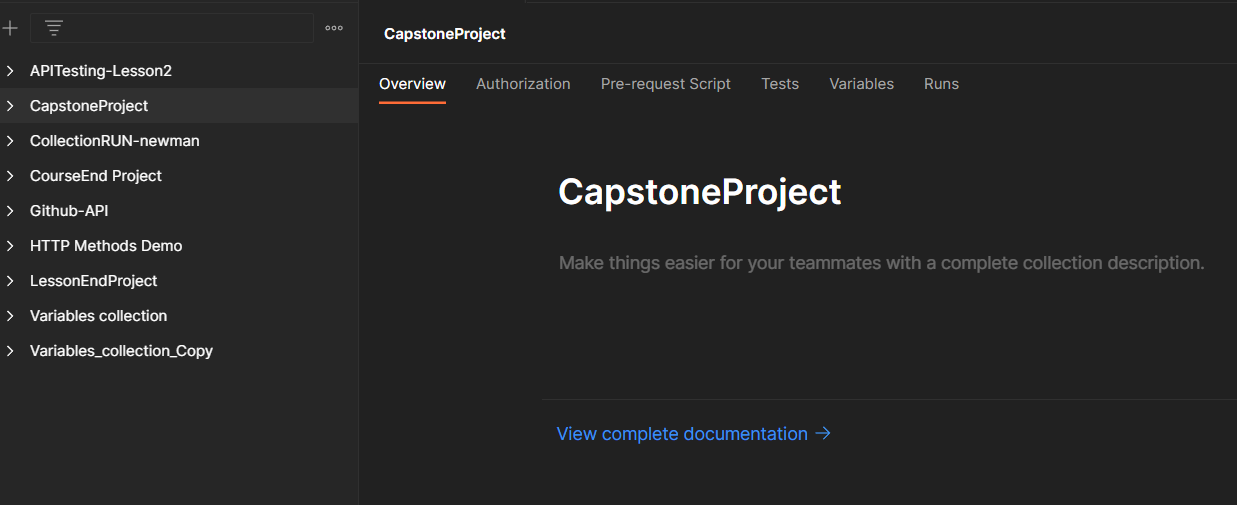


Give the name for workspace and choose the workspace type and click on create. It will create a new workspace.



To create a new collection click on collection at left side headers and then click on ‘+’ symbol and select blank collection then give the name to the collection.

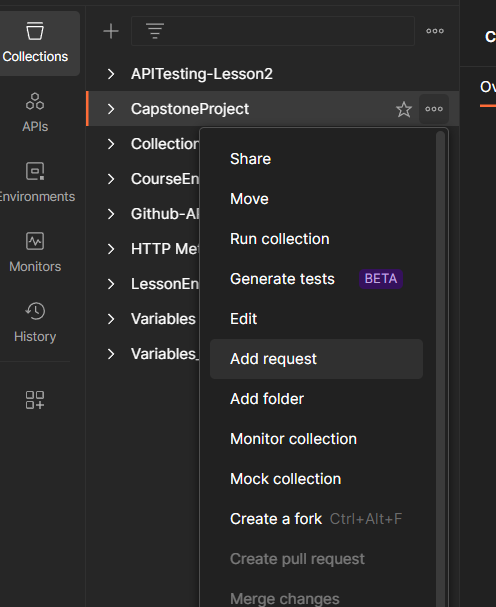




**Retrieve the list of all products in the store.**

To create a Request in the collection.

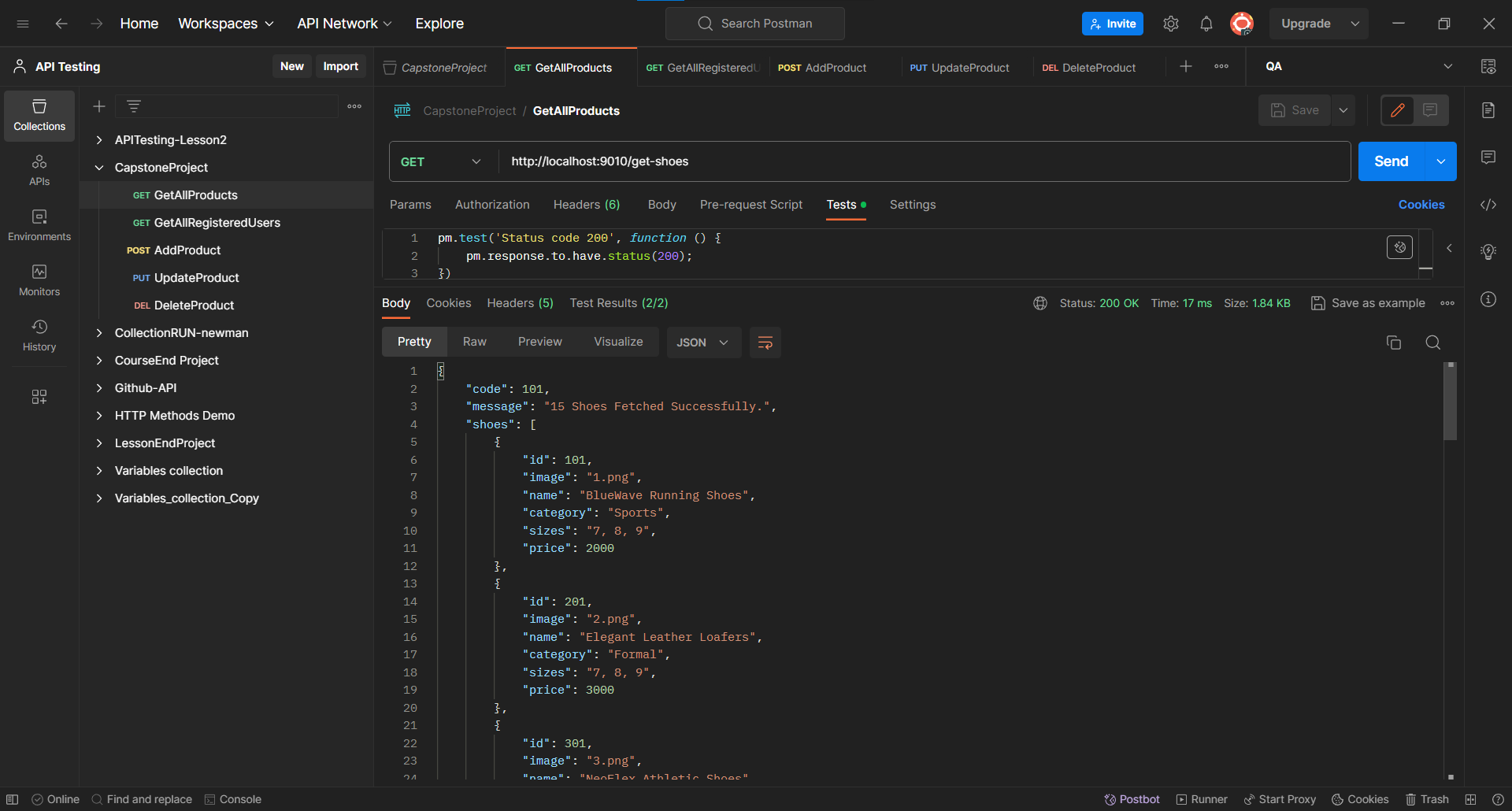
Right click on target collection->select Add Request. And then we can change the request name if we want.

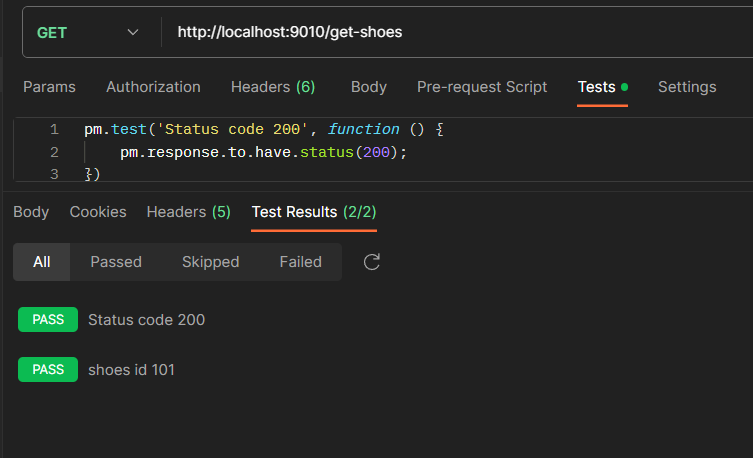


After creating a request. Given the https link in tab and selected the GET method.



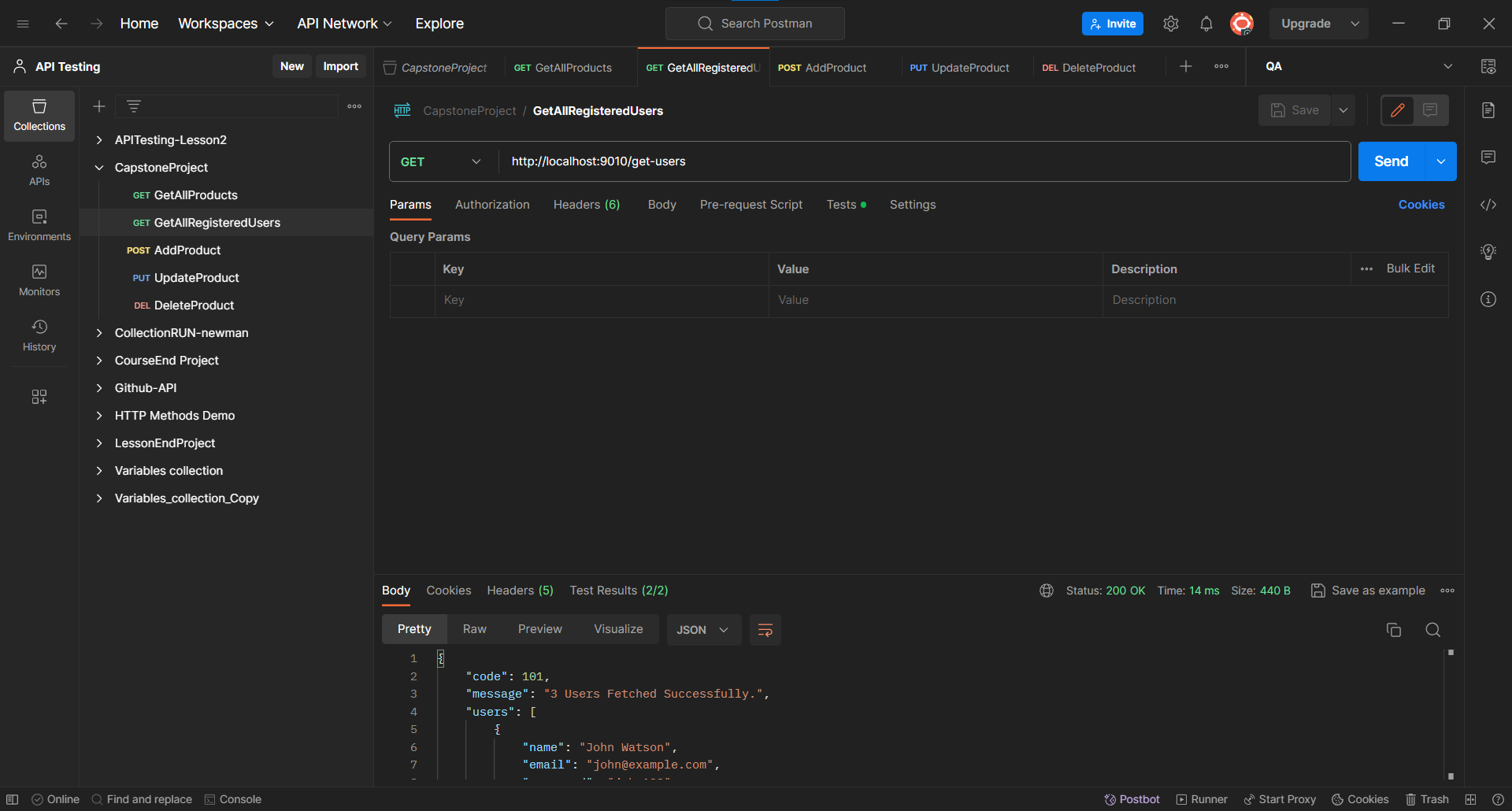


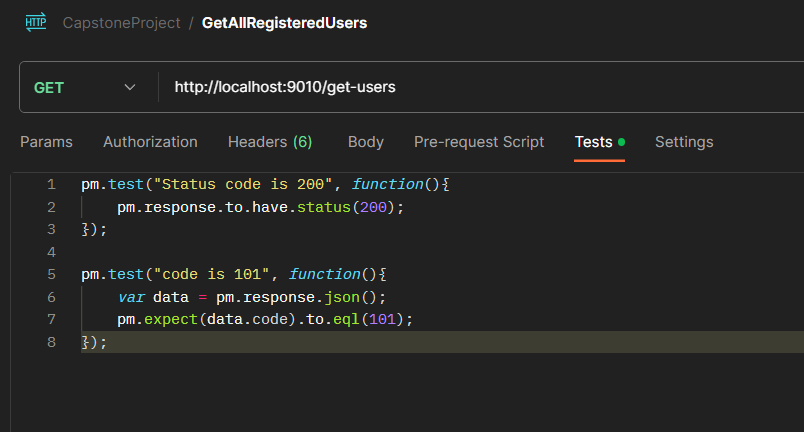


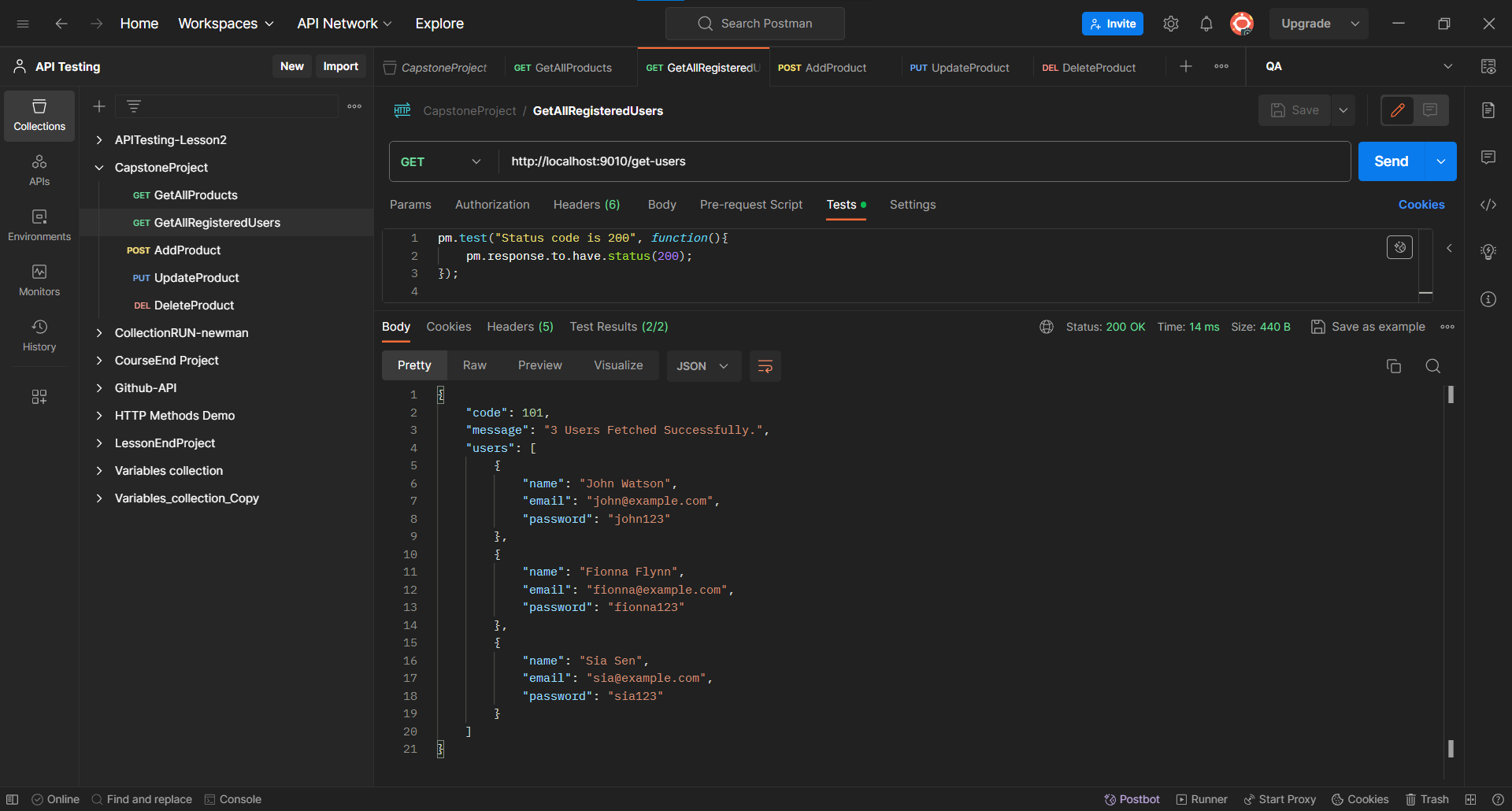


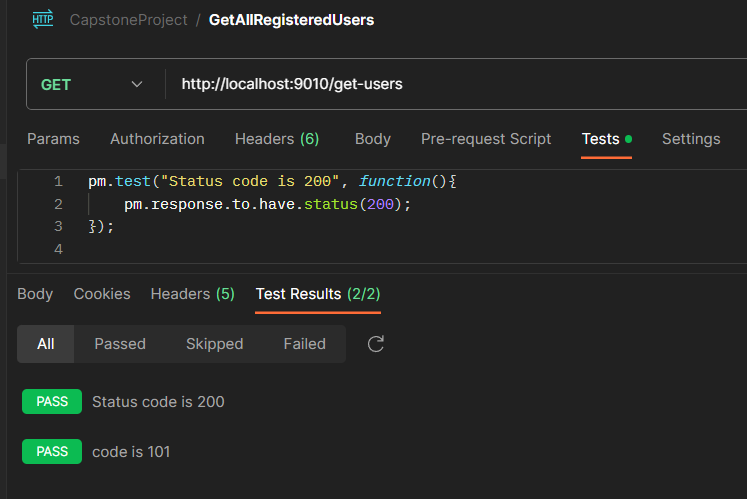
**Retrieve the list of all registered users.**

After creating a request. Given the https link in tab and selected the GET method.



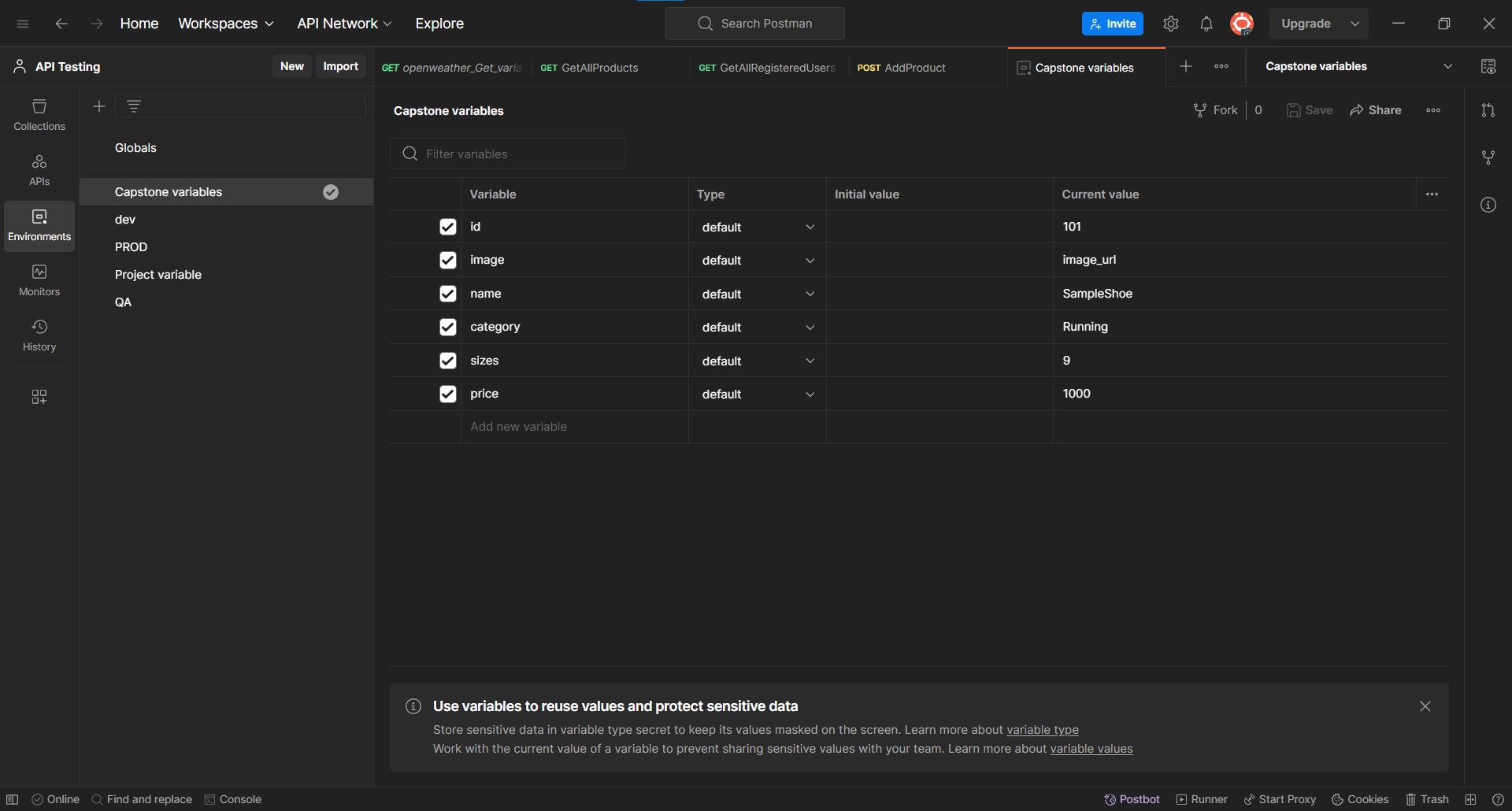






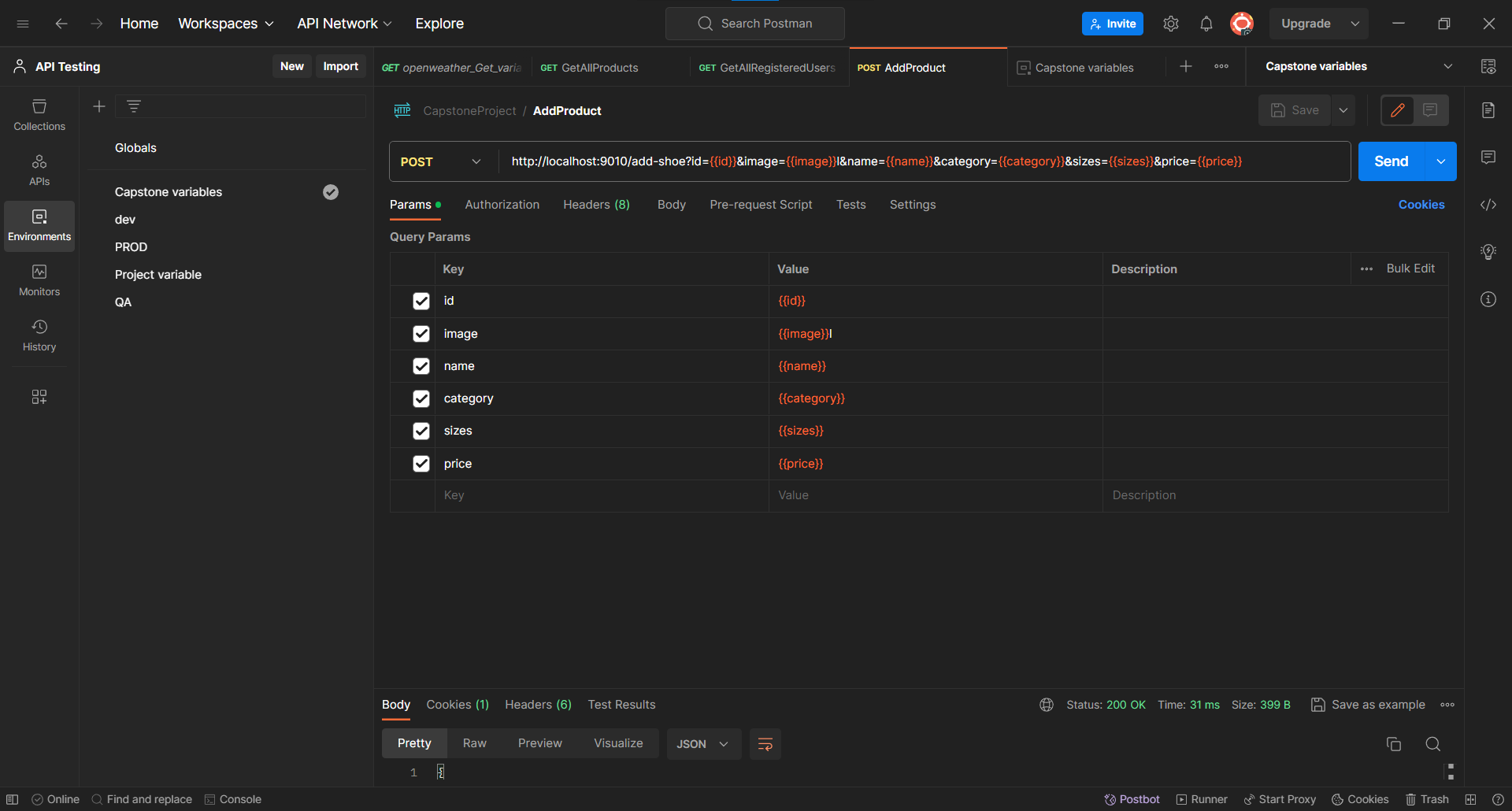
**Add the product.**

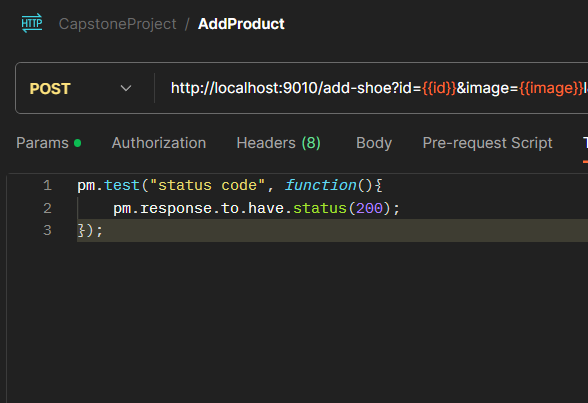
Created a Environment by click on Environment section and hen click on ‘+’ symbol then add the variables

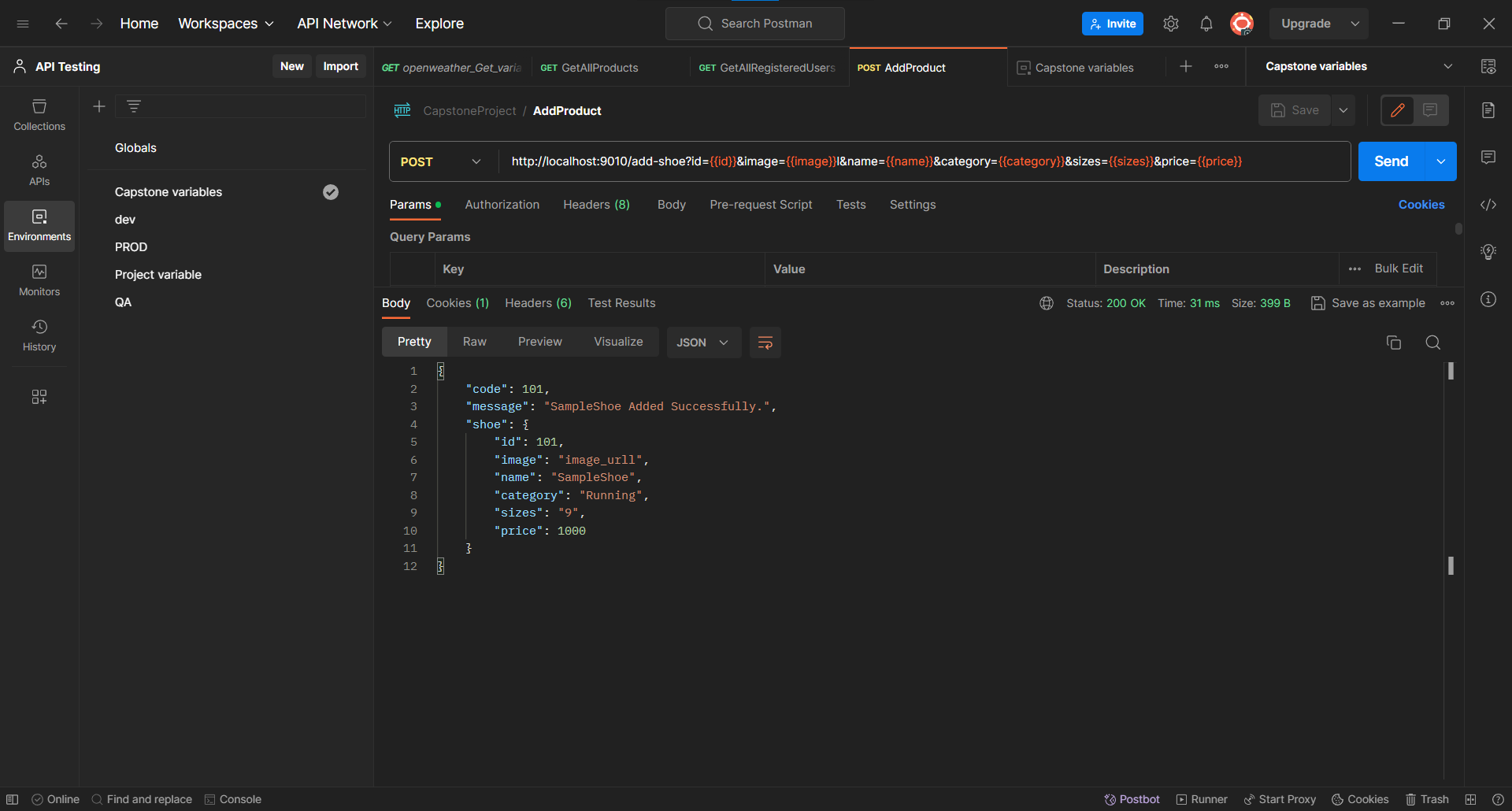


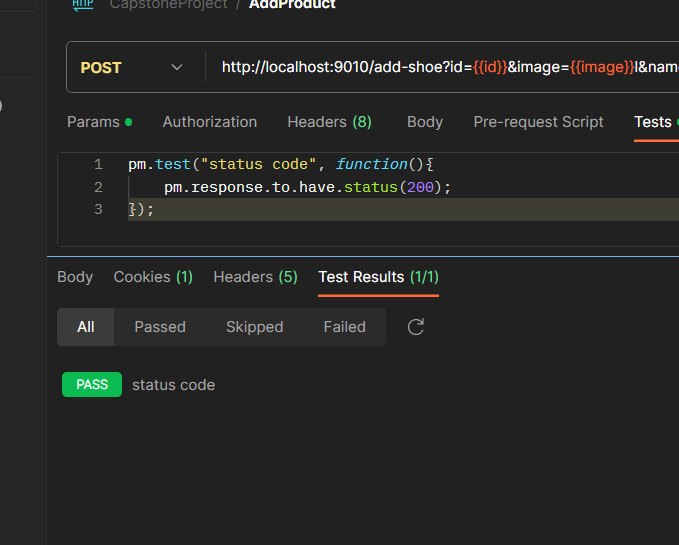
Created a request. Given the https link in tab and select the post method.

And selected the created environment.



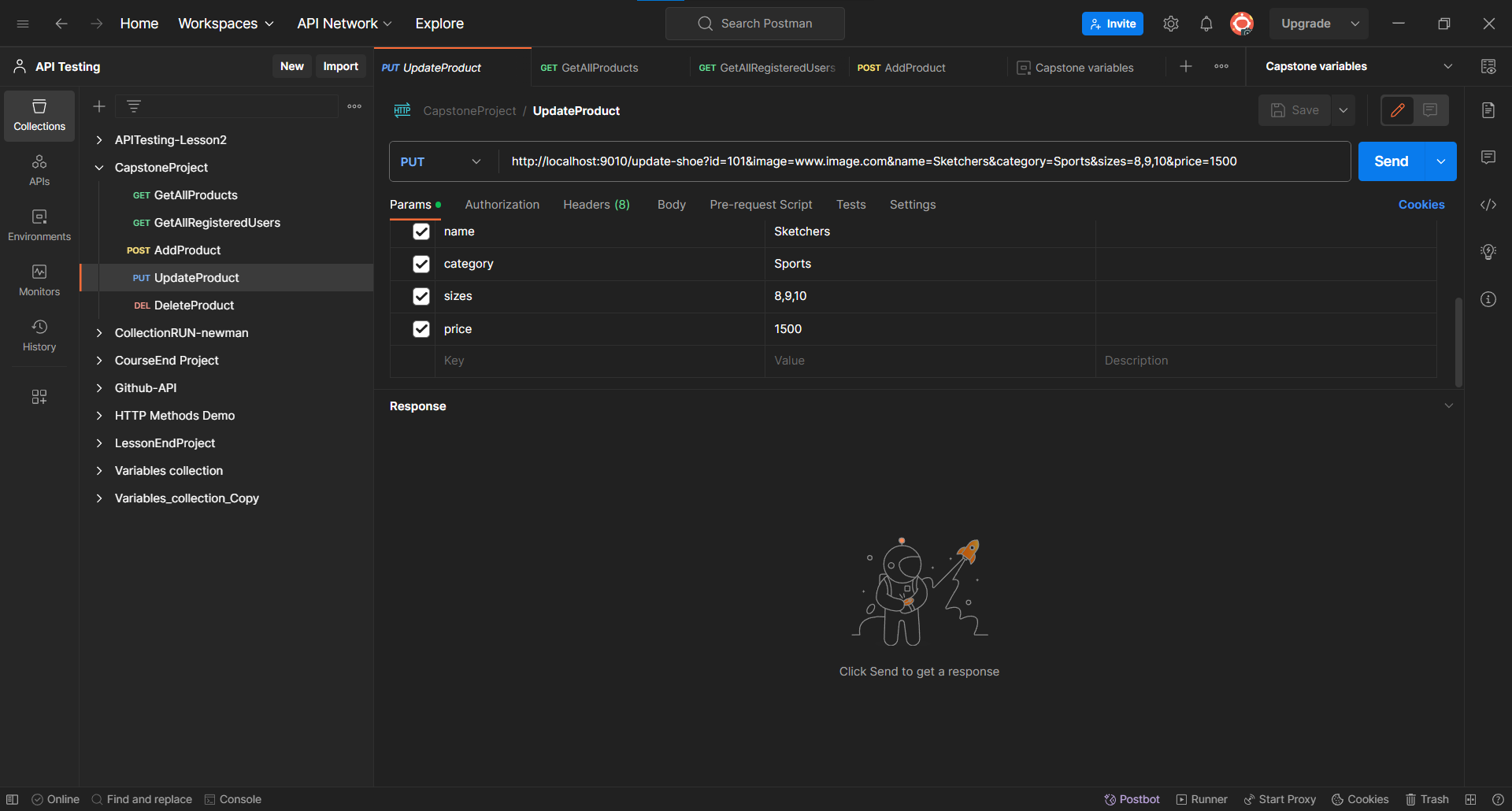


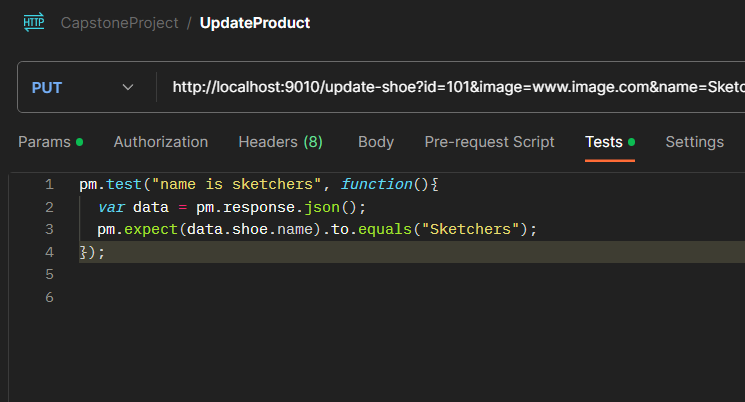


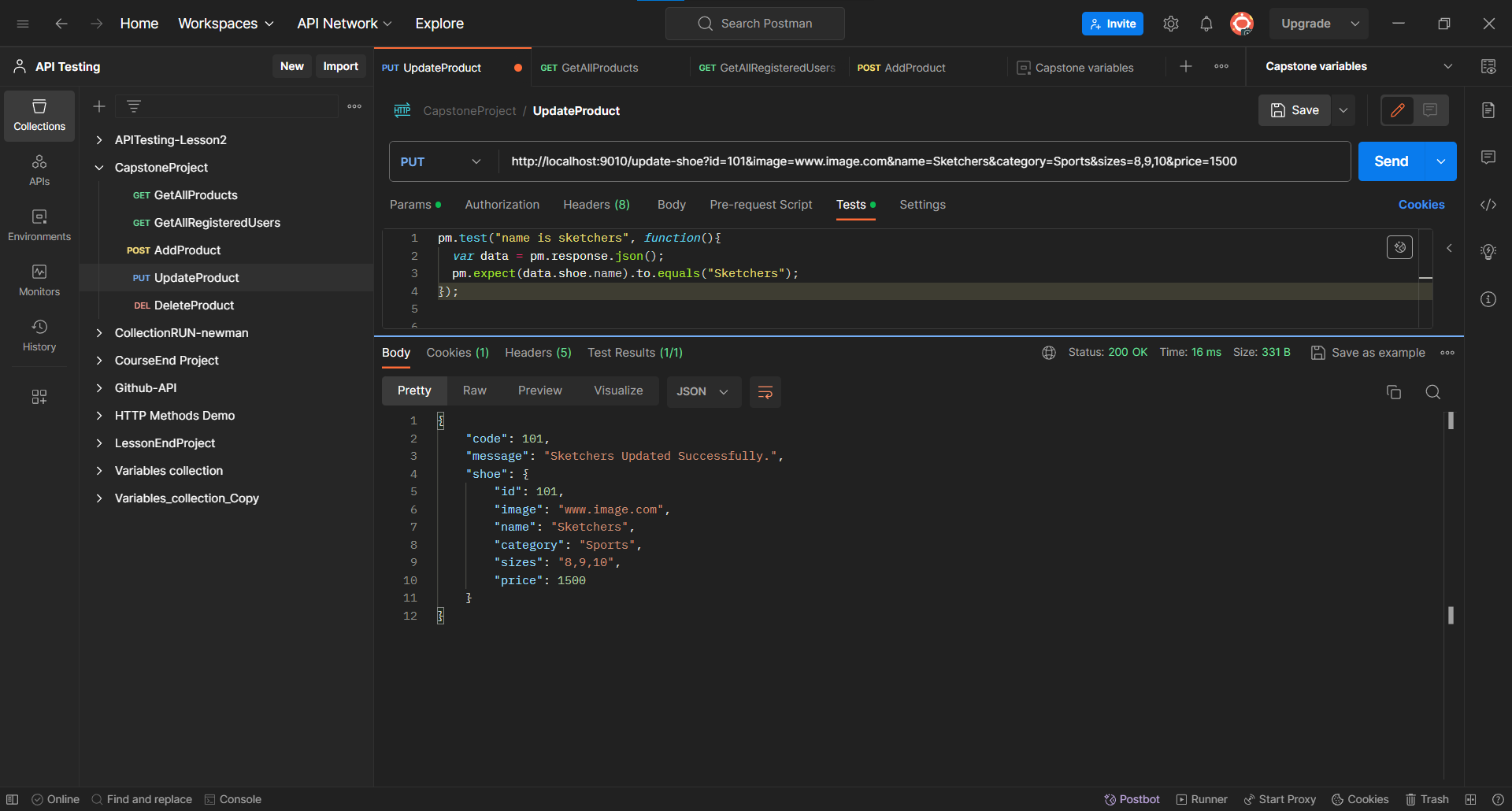


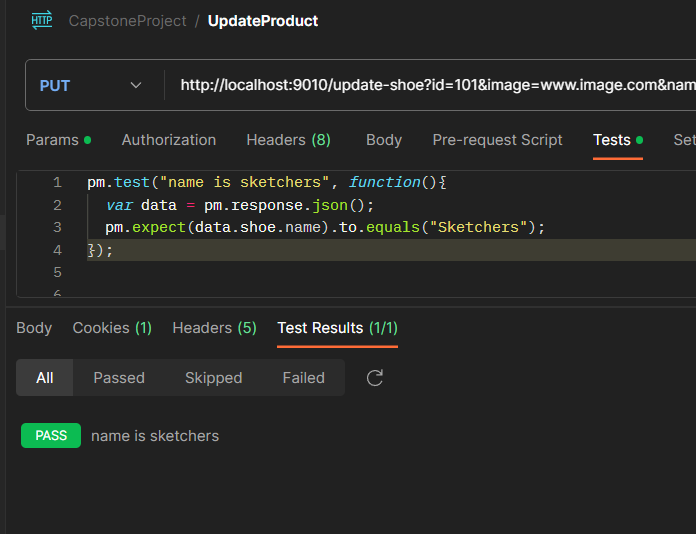
**Update the product**.

Created a request. Given the https link in tab and select the put method.









**Delete The Product**

Created a request. Given the https link in tab and select the Delete method.

