SourceCode:

(#1.RestAssured)

```
package Capstone_Project;

import java.util.HashMap;

import org.testng.annotations.Test;

import io.restassured.RestAssured;

public class RestAssured_Testing {

	@Test(priority=1)
	public void get_all_products()
	{

		RestAssured.given()
		.baseUri("http://localhost:9010")
		.basePath("/get-products")
		.when().get()
		.then().statusCode(200)
		.log().all();
	}
	@Test(priority=2)
	public void get_all_users() {

		RestAssured.given()
		.baseUri("http://localhost:9010")
		.basePath("/get-users")
		.when().get()
		.then().statusCode(200)
		.log().all();


	}

	@Test(priority=3)
	public void add_Product() {


		HashMap<String, String> map = new HashMap<String, String>();

		map.put("id", "999");
		map.put("image", ".png");
		map.put("name", "Disprin");
		map.put("category", "medicine");
		map.put("brand", "BZ Medico");
		map.put("status", "1");
		map.put("price", "100");

		RestAssured.given()
		.baseUri("http://localhost:9010")
		.basePath("/add-product")
		.contentType("application/json")
		.body(map)
		.when().post()
		.then().statusCode(200).log().all();



	}
```

```
	@Test(priority=4)
	public void update_ProductName() {


		HashMap<String, String> map = new HashMap<String, String>();

		map.put("id", "999");
		map.put("image", "2.png");
		map.put("name", "Disprin+");
		map.put("category", "medicine");
		map.put("brand", "BZ Medico");
		map.put("status", "1");
		map.put("price", "120");

		RestAssured.given()
		.baseUri("http://localhost:9010")
		.basePath("/update-product")
		.contentType("application/json")
		.body(map)
		.when().put()
		.then().statusCode(200).log().all();



	}

	@Test(priority=5)
	public void update_ProductStatus() {


		HashMap<String, String> map = new HashMap<String, String>();

		map.put("id", "999");
		map.put("image", "2.png");
		map.put("name", "Disprin+");
		map.put("category", "medicine");
		map.put("brand", "BZ Medico");
		map.put("status", "0");
		map.put("price", "120");

		RestAssured.given()
		.baseUri("http://localhost:9010")
		.basePath("/update-product-status")
		.contentType("application/json")
		.body(map)
		.when().put()
		.then().statusCode(200).log().all();

	}
	@Test(priority=6)
	public void deleteProduct() {

		RestAssured.given()
		.baseUri("http://localhost:9010")
		.basePath("/delete-product")
		.queryParam("id", "101")
		.when().delete()
		.then().statusCode(200)
		.log().all();


	}

}
```

#2 Selenium TestNG:

(TestBase.java)

```
package Base;

import java.time.Duration;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class TestBase {
public static WebDriver driver;
public static void openBrowser(String browser) {
driver = new ChromeDriver();
driver.manage().window().maximize();
driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(5));
driver.get("http://localhost:9010/");
}

}
```

(HomePage.java)

```
package HomePage;

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 {


        @FindBy(xpath = "//input[@id=\"email\"]")
        WebElement Email;

        @FindBy(id = "password")
        WebElement password;

        @FindBy(xpath = "//button[contains(text(),'Login')]")
        WebElement login;

        @FindBy(id = "cart102")
        WebElement AddToCart;

        @FindBy(linkText = "Home")
        WebElement Home;

        @FindBy(linkText = "Cart")
        WebElement Cart;

        @FindBy(linkText = "Place Order")
        WebElement PlaceOrder;
```

```
@FindBy(id = "search-product")
WebElement Search;

@FindBy(id = "filter-button")
WebElement Filter;

@FindBy(xpath = "//ul[@class=\"dropdown-menu show\"]/descendant::a[@id=\"lth\"]")
WebElement Filter_option;

@FindBy(id = "search-product-button")
WebElement SearchButton;

public HomePage(WebDriver driver) {

	PageFactory.initElements(driver, this);
}

// Actions

public void EnterEmail() {

	Email.click();
	Email.sendKeys("gaurav@medicare.com");
}

public void EnterPassword() {

	password.click();
	password.sendKeys("123456");;

}

public void clickOnLogin() {

	login.click();
}

public void clickOnAddToCart() {

	AddToCart.click();
}

public void Home() {

	Home.click();
}

public void ClickOnCart() {

	Cart.click();
}

public void ClickOnPlaceOrder() {

	PlaceOrder.click();
}

public void ClickOnFilter() {
	System.out.println("Filter");
	Filter.click();
}

public void ClickOnFilter_option() {
```

```
		System.out.println("Filter_option");
		Filter_option.click();
	}

	 public void ClickOnSearch() {

	 Search.click();
	 Search.sendKeys("Hamdard Safi Natural Blood Purifier Syrup");
	 }

	 public void ClickOnSearchButton() {

				SearchButton.click();
		}


}
```

(TestAllPage.java)

```
package HomePageTest;

import org.openqa.selenium.JavascriptExecutor;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;

import Base.TestBase;
import HomePage.HomePage;


public class TestAll_Page extends TestBase {

	HomePage hp;

	@BeforeClass
	public void OpenApp() {

		openBrowser("Chrome");
		hp = new HomePage(driver);
	}

	@Test(priority = '1')
	public void Registration_Page() throws InterruptedException {

		Thread.sleep(2000);
		hp.EnterEmail();;
		Thread.sleep(2000);
		hp.EnterPassword();;
		Thread.sleep(2000);
		hp.clickOnLogin();
		((JavascriptExecutor) driver).executeScript("window.scrollBy(0,650)");
		Thread.sleep(2000);

	}
	@Test(priority = '2')
	public void AddToCart_Page() throws InterruptedException{
		hp.clickOnAddToCart();
		Thread.sleep(2000);
		hp.Home();
		Thread.sleep(2000);
		hp.ClickOnCart();
```

```
                ((JavascriptExecutor) driver).executeScript("window.scrollBy(0,6500)");
                Thread.sleep(5000);
        }
        @Test(priority = '3')
        public void PlaceOrder_Page() throws InterruptedException{
                hp.ClickOnPlaceOrder();
                Thread.sleep(2000);
                hp.Home();
                Thread.sleep(1000);
        }
        @Test(priority = '4')
        public void SearchProduct_Page() throws InterruptedException{
                hp.ClickOnSearch();
                hp.ClickOnSearchButton();
                ((JavascriptExecutor) driver).executeScript("window.scrollBy(0,650)");
                //Thread.sleep(1000);
                hp.Home();

        }

        @Test(priority = '5')
        public void Filter_Page() throws InterruptedException{
                ((JavascriptExecutor) driver).executeScript("window.scrollBy(0,650)");
                Thread.sleep(4000);
                hp.ClickOnFilter();
                Thread.sleep(2000);
                hp.ClickOnFilter_option();

        }

         @AfterClass public void CloseApp() {
                 driver.quit();
                 }

}
```

#3 JMeter Scripts:

(.jmx File Code)

```
<?xml version="1.0" encoding="UTF-8"?>
<jmeterTestPlan version="1.2" properties="5.0" jmeter="5.6.2">
  <hashTree>
    <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Second" enabled="true">
      <stringProp name="TestPlan.comments">This test plan was created by the BlazeMeter converter v.3.1.23.
Please contact support@blazemeter.com for further support.</stringProp>
      <boolProp name="TestPlan.functional_mode">false</boolProp>
      <boolProp name="TestPlan.serialize_threadgroups">false</boolProp>
      <elementProp name="TestPlan.user_defined_variables" elementType="Arguments">
        <collectionProp name="Arguments.arguments"/>
      </elementProp>
      <stringProp name="TestPlan.user_define_classpath"></stringProp>
    </TestPlan>
    <hashTree>
      <HeaderManager guiclass="HeaderPanel" testclass="HeaderManager" testname="HTTP Header manager"
enabled="true">
        <collectionProp name="HeaderManager.headers">
          <elementProp name="sec-ch-ua" elementType="Header">
            <stringProp name="Header.name">sec-ch-ua</stringProp>
            <stringProp name="Header.value">&amp;quot;Not_A Brand&amp;quot;;v=&amp;quot;8&amp;quot;,
&amp;quot;Chromium&amp;quot;;v=&amp;quot;120&amp;quot;, &amp;quot;Google Chrome&amp;quot;;v=&amp;quot;120&amp;quot;</stringProp>
          </elementProp>
          <elementProp name="sec-ch-ua-mobile" elementType="Header">
            <stringProp name="Header.name">sec-ch-ua-mobile</stringProp>
```

<stringProp name="Header.value">?0</stringProp>
</elementProp>
<elementProp name="Accept" elementType="Header">
<stringProp name="Header.name">Accept</stringProp>
<stringProp
name="Header.value">text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,
/;q=0.8,application/signed-exchange;v=b3;q=0.7</stringProp>
</elementProp>
<elementProp name="Upgrade-Insecure-Requests" elementType="Header">
<stringProp name="Header.name">Upgrade-Insecure-Requests</stringProp>
<stringProp name="Header.value">1</stringProp>
</elementProp>
<elementProp name="sec-ch-ua-platform" elementType="Header">
<stringProp name="Header.name">sec-ch-ua-platform</stringProp>
<stringProp name="Header.value">"Windows"</stringProp>
</elementProp>
<elementProp name="User-Agent" elementType="Header">
<stringProp name="Header.name">User-Agent</stringProp>
<stringProp name="Header.value">Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36</stringProp>
</elementProp>
</collectionProp>
</HeaderManager>
<hashTree/>
<Arguments guiclass="ArgumentsPanel" testclass="Arguments" testname="User Defined Variables"
enabled="true">
<collectionProp name="Arguments.arguments">
<elementProp name="BASE_URL_1" elementType="Argument">
<stringProp name="Argument.name">BASE_URL_1</stringProp>
<stringProp name="Argument.value">localhost</stringProp>
<stringProp name="Argument.metadata">=</stringProp>
</elementProp>
</collectionProp>
</Arguments>
<hashTree/>
<ConfigTestElement guiclass="HttpDefaultsGui" testclass="ConfigTestElement" testname="HTTP Request
Defaults" enabled="true">
<elementProp name="HTTPsampler.Arguments" elementType="Arguments"
guiclass="HTTPArgumentsPanel" testclass="Arguments" enabled="true">
<collectionProp name="Arguments.arguments"/>
</elementProp>
<boolProp name="HTTPSampler.image_parser">true</boolProp>
<boolProp name="HTTPSampler.concurrentDwn">true</boolProp>
<stringProp name="HTTPSampler.concurrentPool">6</stringProp>
</ConfigTestElement>
<hashTree/>
<DNSCacheManager guiclass="DNSCachePanel" testclass="DNSCacheManager" testname="DNS Cache
Manager" enabled="true">
<collectionProp name="DNSCacheManager.servers"/>
<boolProp name="DNSCacheManager.clearEachIteration">true</boolProp>
<boolProp name="DNSCacheManager.isCustomResolver">false</boolProp>
</DNSCacheManager>
<hashTree/>
<AuthManager guiclass="AuthPanel" testclass="AuthManager" testname="HTTP Authorization Manager"
enabled="true">
<collectionProp name="AuthManager.auth_list"/>
<boolProp name="AuthManager.controllerByThreadGroup">false</boolProp>
</AuthManager>
<hashTree/>
<CookieManager guiclass="CookiePanel" testclass="CookieManager" testname="HTTP Cookie Manager"
enabled="true">
<collectionProp name="CookieManager.cookies"/>
<boolProp name="CookieManager.clearEachIteration">true</boolProp>
<boolProp name="CookieManager.controllerByThreadGroup">false</boolProp>
</CookieManager>
<hashTree/>

```
    <CacheManager guiclass="CacheManagerGui" testclass="CacheManager" testname="HTTP Cache Manager"
enabled="true">
      <boolProp name="clearEachIteration">true</boolProp>
      <boolProp name="useExpires">false</boolProp>
      <boolProp name="CacheManager.controlledByThread">false</boolProp>
    </CacheManager>
    <hashTree/>
    <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup" testname="Thread Group"
enabled="true">
      <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
      <elementProp name="ThreadGroup.main_controller" elementType="LoopController"
guiclass="LoopControlPanel" testclass="LoopController" enabled="true">
        <stringProp name="LoopController.loops">2</stringProp>
        <boolProp name="LoopController.continue_forever">false</boolProp>
      </elementProp>
      <stringProp name="ThreadGroup.num_threads">5</stringProp>
      <stringProp name="ThreadGroup.ramp_time">1</stringProp>
      <boolProp name="ThreadGroup.scheduler">false</boolProp>
      <stringProp name="ThreadGroup.duration">0</stringProp>
      <stringProp name="ThreadGroup.delay">0</stringProp>
      <boolProp name="ThreadGroup.delayedStart">false</boolProp>
      <boolProp name="ThreadGroup.same_user_on_next_iteration">true</boolProp>
    </ThreadGroup>
    <hashTree>
      <TransactionController guiclass="TransactionControllerGui" testclass="TransactionController"
testname="Test" enabled="true">
        <boolProp name="TransactionController.includeTimers">false</boolProp>
        <boolProp name="TransactionController.parent">false</boolProp>
      </TransactionController>
      <hashTree>
        <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Login-
Registration" enabled="true">
          <boolProp name="HTTPSampler.postBodyRaw">false</boolProp>
          <elementProp name="HTTPsampler.Arguments" elementType="Arguments"
guiclass="HTTPArgumentsPanel" testclass="Arguments" enabled="true">
            <collectionProp name="Arguments.arguments">
              <elementProp name="password" elementType="HTTPArgument">
                <boolProp name="HTTPArgument.always_encode">true</boolProp>
                <stringProp name="Argument.name">password</stringProp>
                <stringProp name="Argument.value">123456</stringProp>
                <stringProp name="Argument.metadata">=</stringProp>
                <boolProp name="HTTPArgument.use_equals">true</boolProp>
              </elementProp>
              <elementProp name="email" elementType="HTTPArgument">
                <boolProp name="HTTPArgument.always_encode">true</boolProp>
                <stringProp name="Argument.name">email</stringProp>
                <stringProp name="Argument.value">gaurav@medicare.com</stringProp>
                <stringProp name="Argument.metadata">=</stringProp>
                <boolProp name="HTTPArgument.use_equals">true</boolProp>
              </elementProp>
            </collectionProp>
          </elementProp>
          <stringProp name="HTTPSampler.domain">${BASE_URL_1}</stringProp>
          <stringProp name="HTTPSampler.port">9010</stringProp>
          <stringProp name="HTTPSampler.protocol">http</stringProp>
          <stringProp name="HTTPSampler.path">login</stringProp>
          <stringProp name="HTTPSampler.method">POST</stringProp>
          <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
          <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
          <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
          <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
          <boolProp name="HTTPSampler.BROWSER_COMPATIBLE_MULTIPART">false</boolProp>
          <boolProp name="HTTPSampler.image_parser">false</boolProp>
          <boolProp name="HTTPSampler.concurrentDwn">false</boolProp>
          <stringProp name="HTTPSampler.concurrentPool">6</stringProp>
          <boolProp name="HTTPSampler.md5">false</boolProp>
          <intProp name="HTTPSampler.ipSourceType">0</intProp>
```

```
        </HTTPSamplerProxy>
        <hashTree>
          <HeaderManager guiclass="HeaderPanel" testclass="HeaderManager" testname="HTTP Header manager"
enabled="true">
            <collectionProp name="HeaderManager.headers">
              <elementProp name="Content-Type" elementType="Header">
                <stringProp name="Header.name">Content-Type</stringProp>
                <stringProp name="Header.value">application/x-www-form-urlencoded</stringProp>
              </elementProp>
            </collectionProp>
          </HeaderManager>
          <hashTree/>
          <UniformRandomTimer guiclass="UniformRandomTimerGui" testclass="UniformRandomTimer"
testname="Uniform Random Timer" enabled="true">
            <stringProp name="RandomTimer.range">0.0</stringProp>
            <stringProp name="ConstantTimer.delay">0</stringProp>
            <stringProp name="TestPlan.comments">Recorded time was 0 milliseconds</stringProp>
          </UniformRandomTimer>
          <hashTree/>
        </hashTree>
        <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy"
testname="AddElementToCart" enabled="true">
          <boolProp name="HTTPSampler.postBodyRaw">false</boolProp>
          <elementProp name="HTTPsampler.Arguments" elementType="Arguments"
guiclass="HTTPArgumentsPanel" testclass="Arguments" enabled="true">
            <collectionProp name="Arguments.arguments">
              <elementProp name="id" elementType="HTTPArgument">
                <boolProp name="HTTPArgument.always_encode">false</boolProp>
                <stringProp name="Argument.name">id</stringProp>
                <stringProp name="Argument.value">101</stringProp>
                <stringProp name="Argument.metadata">=</stringProp>
                <boolProp name="HTTPArgument.use_equals">true</boolProp>
              </elementProp>
            </collectionProp>
          </elementProp>
          <stringProp name="HTTPSampler.domain">${BASE_URL_1}</stringProp>
          <stringProp name="HTTPSampler.port">9010</stringProp>
          <stringProp name="HTTPSampler.protocol">http</stringProp>
          <stringProp name="HTTPSampler.path">add-to-cart</stringProp>
          <stringProp name="HTTPSampler.method">GET</stringProp>
          <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
          <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
          <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
          <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
          <boolProp name="HTTPSampler.BROWSER_COMPATIBLE_MULTIPART">false</boolProp>
          <boolProp name="HTTPSampler.image_parser">false</boolProp>
          <boolProp name="HTTPSampler.concurrentDwn">false</boolProp>
          <stringProp name="HTTPSampler.concurrentPool">6</stringProp>
          <boolProp name="HTTPSampler.md5">false</boolProp>
          <intProp name="HTTPSampler.ipSourceType">0</intProp>
        </HTTPSamplerProxy>
        <hashTree>
          <UniformRandomTimer guiclass="UniformRandomTimerGui" testclass="UniformRandomTimer"
testname="Uniform Random Timer" enabled="true">
            <stringProp name="RandomTimer.range">7159.0</stringProp>
            <stringProp name="ConstantTimer.delay">3579.5</stringProp>
            <stringProp name="TestPlan.comments">Recorded time was 7159 milliseconds</stringProp>
          </UniformRandomTimer>
          <hashTree/>
        </hashTree>
        <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy"
testname="BackToHomePage" enabled="true">
          <boolProp name="HTTPSampler.postBodyRaw">false</boolProp>
          <elementProp name="HTTPsampler.Arguments" elementType="Arguments"
guiclass="HTTPArgumentsPanel" testclass="Arguments" enabled="true">
            <collectionProp name="Arguments.arguments"/>
          </elementProp>
```

```
          <stringProp name="HTTPSampler.domain">${BASE_URL_1}</stringProp>
          <stringProp name="HTTPSampler.port">9010</stringProp>
          <stringProp name="HTTPSampler.protocol">http</stringProp>
          <stringProp name="HTTPSampler.path">home</stringProp>
          <stringProp name="HTTPSampler.method">GET</stringProp>
          <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
          <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
          <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
          <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
          <boolProp name="HTTPSampler.BROWSER_COMPATIBLE_MULTIPART">false</boolProp>
          <boolProp name="HTTPSampler.image_parser">false</boolProp>
          <boolProp name="HTTPSampler.concurrentDwn">false</boolProp>
          <stringProp name="HTTPSampler.concurrentPool">6</stringProp>
          <boolProp name="HTTPSampler.md5">false</boolProp>
          <intProp name="HTTPSampler.ipSourceType">0</intProp>
        </HTTPSamplerProxy>
        <hashTree>
          <UniformRandomTimer guiclass="UniformRandomTimerGui" testclass="UniformRandomTimer"
testname="Uniform Random Timer" enabled="true">
            <stringProp name="RandomTimer.range">3322.0</stringProp>
            <stringProp name="ConstantTimer.delay">1661</stringProp>
            <stringProp name="TestPlan.comments">Recorded time was 3322 milliseconds</stringProp>
          </UniformRandomTimer>
          <hashTree/>
        </hashTree>
        <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy"
testname="SearchForParticularProduct" enabled="true">
          <boolProp name="HTTPSampler.postBodyRaw">false</boolProp>
          <elementProp name="HTTPsampler.Arguments" elementType="Arguments"
guiclass="HTTPArgumentsPanel" testclass="Arguments" enabled="true">
            <collectionProp name="Arguments.arguments">
              <elementProp name="name" elementType="HTTPArgument">
                <boolProp name="HTTPArgument.always_encode">true</boolProp>
                <stringProp name="Argument.name">name</stringProp>
                <stringProp name="Argument.value">Hamdard Safi Natural Blood Purifier Syrup</stringProp>
                <stringProp name="Argument.metadata">=</stringProp>
                <boolProp name="HTTPArgument.use_equals">true</boolProp>
              </elementProp>
            </collectionProp>
          </elementProp>
          <stringProp name="HTTPSampler.domain">${BASE_URL_1}</stringProp>
          <stringProp name="HTTPSampler.port">9010</stringProp>
          <stringProp name="HTTPSampler.protocol">http</stringProp>
          <stringProp name="HTTPSampler.path">search-product</stringProp>
          <stringProp name="HTTPSampler.method">GET</stringProp>
          <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
          <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
          <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
          <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
          <boolProp name="HTTPSampler.BROWSER_COMPATIBLE_MULTIPART">false</boolProp>
          <boolProp name="HTTPSampler.image_parser">false</boolProp>
          <boolProp name="HTTPSampler.concurrentDwn">false</boolProp>
          <stringProp name="HTTPSampler.concurrentPool">6</stringProp>
          <boolProp name="HTTPSampler.md5">false</boolProp>
          <intProp name="HTTPSampler.ipSourceType">0</intProp>
        </HTTPSamplerProxy>
        <hashTree>
          <UniformRandomTimer guiclass="UniformRandomTimerGui" testclass="UniformRandomTimer"
testname="Uniform Random Timer" enabled="true">
            <stringProp name="RandomTimer.range">17733.0</stringProp>
            <stringProp name="ConstantTimer.delay">8866.5</stringProp>
            <stringProp name="TestPlan.comments">Recorded time was 17733 milliseconds</stringProp>
          </UniformRandomTimer>
          <hashTree/>
        </hashTree>
        <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy"
testname="BackToHomePage" enabled="true">
```

```
          <boolProp name="HTTPSampler.postBodyRaw">false</boolProp>
          <elementProp name="HTTPsampler.Arguments" elementType="Arguments"
guiclass="HTTPArgumentsPanel" testclass="Arguments" enabled="true">
            <collectionProp name="Arguments.arguments"/>
          </elementProp>
          <stringProp name="HTTPSampler.domain">${BASE_URL_1}</stringProp>
          <stringProp name="HTTPSampler.port">9010</stringProp>
          <stringProp name="HTTPSampler.protocol">http</stringProp>
          <stringProp name="HTTPSampler.path">home</stringProp>
          <stringProp name="HTTPSampler.method">GET</stringProp>
          <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
          <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
          <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
          <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
          <boolProp name="HTTPSampler.BROWSER_COMPATIBLE_MULTIPART">false</boolProp>
          <boolProp name="HTTPSampler.image_parser">false</boolProp>
          <boolProp name="HTTPSampler.concurrentDwn">false</boolProp>
          <stringProp name="HTTPSampler.concurrentPool">6</stringProp>
          <boolProp name="HTTPSampler.md5">false</boolProp>
          <intProp name="HTTPSampler.ipSourceType">0</intProp>
        </HTTPSamplerProxy>
        <hashTree>
          <UniformRandomTimer guiclass="UniformRandomTimerGui" testclass="UniformRandomTimer"
testname="Uniform Random Timer" enabled="true">
            <stringProp name="RandomTimer.range">6059.0</stringProp>
            <stringProp name="ConstantTimer.delay">3029.5</stringProp>
            <stringProp name="TestPlan.comments">Recorded time was 6059 milliseconds</stringProp>
          </UniformRandomTimer>
          <hashTree/>
        </hashTree>
        <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy"
testname="ApplyFilter" enabled="true">
          <boolProp name="HTTPSampler.postBodyRaw">false</boolProp>
          <elementProp name="HTTPsampler.Arguments" elementType="Arguments"
guiclass="HTTPArgumentsPanel" testclass="Arguments" enabled="true">
            <collectionProp name="Arguments.arguments">
              <elementProp name="name" elementType="HTTPArgument">
                <boolProp name="HTTPArgument.always_encode">false</boolProp>
                <stringProp name="Argument.name">name</stringProp>
                <stringProp name="Argument.value">0</stringProp>
                <stringProp name="Argument.metadata">=</stringProp>
                <boolProp name="HTTPArgument.use_equals">true</boolProp>
              </elementProp>
            </collectionProp>
          </elementProp>
          <stringProp name="HTTPSampler.domain">${BASE_URL_1}</stringProp>
          <stringProp name="HTTPSampler.port">9010</stringProp>
          <stringProp name="HTTPSampler.protocol">http</stringProp>
          <stringProp name="HTTPSampler.path">search-product</stringProp>
          <stringProp name="HTTPSampler.method">GET</stringProp>
          <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
          <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
          <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
          <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
          <boolProp name="HTTPSampler.BROWSER_COMPATIBLE_MULTIPART">false</boolProp>
          <boolProp name="HTTPSampler.image_parser">false</boolProp>
          <boolProp name="HTTPSampler.concurrentDwn">false</boolProp>
          <stringProp name="HTTPSampler.concurrentPool">6</stringProp>
          <boolProp name="HTTPSampler.md5">false</boolProp>
          <intProp name="HTTPSampler.ipSourceType">0</intProp>
        </HTTPSamplerProxy>
        <hashTree>
          <UniformRandomTimer guiclass="UniformRandomTimerGui" testclass="UniformRandomTimer"
testname="Uniform Random Timer" enabled="true">
            <stringProp name="RandomTimer.range">5543.0</stringProp>
            <stringProp name="ConstantTimer.delay">2771.5</stringProp>
            <stringProp name="TestPlan.comments">Recorded time was 5543 milliseconds</stringProp>
```

```
          </UniformRandomTimer>
          <hashTree/>
        </hashTree>
        <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Logout"
enabled="true">
          <boolProp name="HTTPSampler.postBodyRaw">false</boolProp>
          <elementProp name="HTTPsampler.Arguments" elementType="Arguments"
guiclass="HTTPArgumentsPanel" testclass="Arguments" enabled="true">
            <collectionProp name="Arguments.arguments"/>
          </elementProp>
          <stringProp name="HTTPSampler.domain">${BASE_URL_1}</stringProp>
          <stringProp name="HTTPSampler.port">9010</stringProp>
          <stringProp name="HTTPSampler.protocol">http</stringProp>
          <stringProp name="HTTPSampler.path">logout</stringProp>
          <stringProp name="HTTPSampler.method">GET</stringProp>
          <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
          <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
          <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
          <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
          <boolProp name="HTTPSampler.BROWSER_COMPATIBLE_MULTIPART">false</boolProp>
          <boolProp name="HTTPSampler.image_parser">false</boolProp>
          <boolProp name="HTTPSampler.concurrentDwn">false</boolProp>
          <stringProp name="HTTPSampler.concurrentPool">6</stringProp>
          <boolProp name="HTTPSampler.md5">false</boolProp>
          <intProp name="HTTPSampler.ipSourceType">0</intProp>
        </HTTPSamplerProxy>
        <hashTree>
          <UniformRandomTimer guiclass="UniformRandomTimerGui" testclass="UniformRandomTimer"
testname="Uniform Random Timer" enabled="true">
            <stringProp name="RandomTimer.range">10322.0</stringProp>
            <stringProp name="ConstantTimer.delay">5161</stringProp>
            <stringProp name="TestPlan.comments">Recorded time was 10322 milliseconds</stringProp>
          </UniformRandomTimer>
          <hashTree/>
        </hashTree>
      </hashTree>
      <ResultCollector guiclass="ViewResultsFullVisualizer" testclass="ResultCollector" testname="View Results
Tree" enabled="true">
        <boolProp name="ResultCollector.error_logging">false</boolProp>
        <objProp>
          <name>saveConfig</name>
          <value class="SampleSaveConfiguration">
            <time>true</time>
            <latency>true</latency>
            <timestamp>true</timestamp>
            <success>true</success>
            <label>true</label>
            <code>true</code>
            <message>true</message>
            <threadName>true</threadName>
            <dataType>true</dataType>
            <encoding>false</encoding>
            <assertions>true</assertions>
            <subresults>true</subresults>
            <responseData>false</responseData>
            <samplerData>false</samplerData>
            <xml>false</xml>
            <fieldNames>true</fieldNames>
            <responseHeaders>false</responseHeaders>
            <requestHeaders>false</requestHeaders>
            <responseDataOnError>false</responseDataOnError>
            <saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
            <assertionsResultsToSave>0</assertionsResultsToSave>
            <bytes>true</bytes>
            <sentBytes>true</sentBytes>
            <url>true</url>
            <threadCounts>true</threadCounts>
```

```
            <idleTime>true</idleTime>
            <connectTime>true</connectTime>
          </value>
        </objProp>
        <stringProp name="filename"></stringProp>
      </ResultCollector>
      <hashTree/>
      <ResultCollector guiclass="SummaryReport" testclass="ResultCollector" testname="Summary Report"
enabled="true">
        <boolProp name="ResultCollector.error_logging">false</boolProp>
        <objProp>
          <name>saveConfig</name>
          <value class="SampleSaveConfiguration">
            <time>true</time>
            <latency>true</latency>
            <timestamp>true</timestamp>
            <success>true</success>
            <label>true</label>
            <code>true</code>
            <message>true</message>
            <threadName>true</threadName>
            <dataType>true</dataType>
            <encoding>false</encoding>
            <assertions>true</assertions>
            <subresults>true</subresults>
            <responseData>false</responseData>
            <samplerData>false</samplerData>
            <xml>false</xml>
            <fieldNames>true</fieldNames>
            <responseHeaders>false</responseHeaders>
            <requestHeaders>false</requestHeaders>
            <responseDataOnError>false</responseDataOnError>
            <saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
            <assertionsResultsToSave>0</assertionsResultsToSave>
            <bytes>true</bytes>
            <sentBytes>true</sentBytes>
            <url>true</url>
            <threadCounts>true</threadCounts>
            <idleTime>true</idleTime>
            <connectTime>true</connectTime>
          </value>
        </objProp>
        <stringProp name="filename"></stringProp>
      </ResultCollector>
      <hashTree/>
      <ResultCollector guiclass="StatVisualizer" testclass="ResultCollector" testname="Aggregate Report"
enabled="true">
        <boolProp name="ResultCollector.error_logging">false</boolProp>
        <objProp>
          <name>saveConfig</name>
          <value class="SampleSaveConfiguration">
            <time>true</time>
            <latency>true</latency>
            <timestamp>true</timestamp>
            <success>true</success>
            <label>true</label>
            <code>true</code>
            <message>true</message>
            <threadName>true</threadName>
            <dataType>true</dataType>
            <encoding>false</encoding>
            <assertions>true</assertions>
            <subresults>true</subresults>
            <responseData>false</responseData>
            <samplerData>false</samplerData>
            <xml>false</xml>
            <fieldNames>true</fieldNames>
```

```
            <responseHeaders>false</responseHeaders>
            <requestHeaders>false</requestHeaders>
            <responseDataOnError>false</responseDataOnError>
            <saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
            <assertionsResultsToSave>0</assertionsResultsToSave>
            <bytes>true</bytes>
            <sentBytes>true</sentBytes>
            <url>true</url>
            <threadCounts>true</threadCounts>
            <idleTime>true</idleTime>
            <connectTime>true</connectTime>
          </value>
        </objProp>
        <stringProp name="filename"></stringProp>
      </ResultCollector>
      <hashTree/>
      <ResultCollector guiclass="GraphVisualizer" testclass="ResultCollector" testname="Graph Results"
enabled="true">
        <boolProp name="ResultCollector.error_logging">false</boolProp>
        <objProp>
          <name>saveConfig</name>
          <value class="SampleSaveConfiguration">
            <time>true</time>
            <latency>true</latency>
            <timestamp>true</timestamp>
            <success>true</success>
            <label>true</label>
            <code>true</code>
            <message>true</message>
            <threadName>true</threadName>
            <dataType>true</dataType>
            <encoding>false</encoding>
            <assertions>true</assertions>
            <subresults>true</subresults>
            <responseData>false</responseData>
            <samplerData>false</samplerData>
            <xml>false</xml>
            <fieldNames>true</fieldNames>
            <responseHeaders>false</responseHeaders>
            <requestHeaders>false</requestHeaders>
            <responseDataOnError>false</responseDataOnError>
            <saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
            <assertionsResultsToSave>0</assertionsResultsToSave>
            <bytes>true</bytes>
            <sentBytes>true</sentBytes>
            <url>true</url>
            <threadCounts>true</threadCounts>
            <idleTime>true</idleTime>
            <connectTime>true</connectTime>
          </value>
        </objProp>
        <stringProp name="filename"></stringProp>
      </ResultCollector>
      <hashTree/>
    </hashTree>
  </hashTree>
 </hashTree>
</jmeterTestPlan>
```

#4 Cucumber-Gerkins Keyword For Api Testing:

```
Feature: TO Create A Test Plan For Medicare Application Capstone
Project

Scenario: To Retrieve the list of all products in the store
Given User Enters Medicare base URI and Base Path
When User Executes Get Request Of HTTP method
Then Validate The Response Status Code

Scenario: To Retrieve the list of all Registered users
Given User Enters Medicare base URI and Base Path
When User Executes Get Request Of HTTP method
Then Validate The Response Status Code

Scenario: Add the product
Given User Enters Medicare base URI and Base Path
When User Executes Post Request Of HTTP method
Then Validate The Response Status Code

Scenario: Update the product
Given User Enters Medicare base URI and Base Path
When User Executes Put Request Of HTTP method
Then Validate The Response Status Code

Scenario: Update the product status
Given User Enters Medicare base URI and Base Path
When User Executes Put Request Of HTTP method To Update Status Code
Then Validate The Response Status Code

Scenario: Delete the product
Given User Enters Medicare base URI and Base Path
When User Executes Delete Request Of HTTP method
Then Validate The Response Status Code
```

(StepDefination.java)

```
package Capstone_Project;

import java.util.HashMap;

import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;
import io.restassured.RestAssured;

public class StepDefinationFile{
@Given("User Enters Medicare base URI and Base Path")
public void user_enters_medicare_base_uri_and_base_path() {
RestAssured.given()
.baseUri("http://localhost:9010")
.basePath("/get-products")
.when().get()
.then().statusCode(200)
.log().all();
}
```

```
@When("User Executes Get Request Of HTTP method")
public void user_executes_get_request_of_http_method() {
RestAssured.given()
.baseUri("http://localhost:9010")
.basePath("/get-products")
.when().get()
.then().statusCode(200)
.log().all();
}
@Then("Validate The Response Status Code")
public void Validate_the_Response_status_code() {
RestAssured.given()
.baseUri("http://localhost:9010")
.basePath("/get-products")
.when().get()
.then().statusCode(200)
.log().all();
}
@When("User Executes Post Request Of HTTP method")
public void user_executes_post_request_of_http_method() {
HashMap<String, String> map = new HashMap<String, String>();
map.put("id", "999");
map.put("image", ".png");
map.put("name", "Disprin");
map.put("category", "medicine");
map.put("brand", "BZ Medico");
map.put("status", "1");
map.put("price", "100");
RestAssured.given()
.baseUri("http://localhost:9010")
.basePath("/add-product")
.contentType("application/json")
.body(map)
.when().post()
.then().statusCode(200).log().all();
}
@When("User Executes Put Request Of HTTP method")
public void user_executes_put_request_of_http_method() {
HashMap<String, String> map = new HashMap<String, String>();
map.put("id", "999");
map.put("image", "2.png");
map.put("name", "Disprin+");
map.put("category", "medicine");
map.put("brand", "BZ Medico");
map.put("status", "1");
map.put("price", "120");
RestAssured.given()
.baseUri("http://localhost:9010")
.basePath("/update-product")
.contentType("application/json")
.body(map)
.when().put()
.then().statusCode(200).log().all();
}
@When("User Executes Put Request Of HTTP method To Update Status
Code")
public void
user_executes_put_request_of_http_method_to_update_status_code() {
HashMap<String, String> map = new HashMap<String, String>();
map.put("id", "999");
map.put("image", "2.png");
```

```
map.put("name", "Disprin+");
map.put("category", "medicine");
map.put("brand", "BZ Medico");
map.put("status", "0");
map.put("price", "120");
RestAssured.given()
.baseUri("http://localhost:9010")
.basePath("/update-product-status")
.contentType("application/json")
.body(map)
.when().put()
.then().statusCode(200).log().all();
}

@When("User Executes Delete Request Of HTTP method")
public void user_executes_delete_request_of_http_method() {
RestAssured.given()
.baseUri("http://localhost:9010")
.basePath("/delete-product")
.queryParam("id", "101")
.when().delete()
.then().statusCode(200)
.log().all();
}
}
```

#5 Postman-scripts to test:

(.json file code)

```
{
    "info": {
        "_postman_id": "f383143a-31b1-41c6-a770-b3da6a3ea20c",
        "name": "Medicare_Project",
        "schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json",
        "_exporter_id": "31715611",
        "_collection_link": "https://lively-trinity-
694168.postman.co/workspace/MediCare_Project~3d1a577c-e05c-4cce-8b8d-
52ef79244652/collection/31715611-f383143a-31b1-41c6-a770-
b3da6a3ea20c?action=share&source=collection_link&creator=31715611"
    },
    "item": [
        {
            "name": "RetrieveAll_Products",
            "event": [
                {
                    "listen": "test",
                    "script": {
                        "exec": [
                            "pm.test(\"Status code is 200\", function () {\r",
                            "    pm.response.to.have.status(200);\r",
                            "});\r",
                            "\r",
                            "pm.test(\"BodyCode_Check\", function () {\r",
                            "    pm.expect(pm.response.text()).to.include(101);\r",
                            "});"
                        ],
                        "type": "text/javascript"
```

```
                }
            }
        ],
        "request": {
            "method": "GET",
            "header": [],
            "url": {
                "raw": "{{BaseUrl}}/get-products",
                "host": [
                    "{{BaseUrl}}"
                ],
                "path": [
                    "get-products"
                ]
            }
        },
        "response": []
    },
    {
        "name": "RetrieveAll_Users",
        "event": [
            {
                "listen": "test",
                "script": {
                    "exec": [
                        "pm.test(\"Status code is 200\", function () {\r",
                        "    pm.response.to.have.status(200);\r",
                        "});\r",
                        ""
                    ],
                    "type": "text/javascript"
                }
            }
        ],
        "request": {
            "method": "GET",
            "header": [],
            "url": {
                "raw": "{{BaseUrl}}/get-users",
                "host": [
                    "{{BaseUrl}}"
                ],
                "path": [
                    "get-users"
                ]
            }
        },
        "response": []
    },
    {
        "name": "AddThe_Product",
        "request": {
            "method": "POST",
            "header": [],
            "body": {
                "mode": "raw",
                "raw": "{\r\n        \"id\": 999,\r\n        \"image\":
\"1.png\",\r\n        \"name\": \"Disprin\",\r\n        \"category\":
\"medicine\",\r\n        \"brand\": \"BZ Medico\",\r\n        \"status\":
1,\r\n        \"price\": 100\r\n}",
                "options": {
```

```
                    "raw": {
                        "language": "json"
                    }
                }
            },
            "url": {
                "raw": "{{BaseUrl}}/add-product",
                "host": [
                    "{{BaseUrl}}"
                ],
                "path": [
                    "add-product"
                ]
            }
        },
        "response": []
    },
    {
        "name": "UpdateThe_Product",
        "event": [
            {
                "listen": "test",
                "script": {
                    "exec": [
                        "pm.test(\"Status code is 200\", function () {\r",
                        "    pm.response.to.have.status(200);\r",
                        "});\r",
                        "pm.test(\"Body matches string\", function () {\r",
                        "    pm.expect(pm.response.text()).to.include(\"Disprin+\");\r",
                        "});\r",
                        "pm.test(\"Body matches string\", function () {\r",
                        "    pm.expect(pm.response.text()).to.include(101);\r",
                        "});"
                    ],
                    "type": "text/javascript"
                }
            }
        ],
        "request": {
            "method": "PUT",
            "header": [],
            "body": {
                "mode": "raw",
                "raw": "{\r\n        \"id\": 999,\r\n        \"image\":
\"2.png\",\r\n        \"name\": \"Disprin+\",\r\n        \"category\":
\"medicine\",\r\n        \"brand\": \"BZ Medico\",\r\n        \"status\":
1,\r\n        \"price\": 120\r\n}",
                "options": {
                    "raw": {
                        "language": "json"
                    }
                }
            },
            "url": {
                "raw": "{{BaseUrl}}/update-product",
                "host": [
                    "{{BaseUrl}}"
                ],
                "path": [
                    "update-product"
                ]
```

```
                }
            },
            "response": []
        },
        {
            "name": "UpdateThe_ProductStatus",
            "event": [
                {
                    "listen": "test",
                    "script": {
                        "exec": [
                            "pm.test(\"Status code is 200\", function () {\r",
                            "    pm.response.to.have.status(200);\r",
                            "});\r",
                            "pm.test(\"Body matches string\", function () {\r",
                            "    pm.expect(pm.response.text()).to.include(\"Disprin+ Status
Updated Successfully.\");\r",
                            "});"
                        ],
                        "type": "text/javascript"
                    }
                }
            ],
            "request": {
                "method": "PUT",
                "header": [],
                "body": {
                    "mode": "raw",
                    "raw": "{\r\n        \"id\": 999,\r\n        \"image\":
\"2.png\",\r\n        \"name\": \"Disprin+\",\r\n        \"category\":
\"medicine\",\r\n        \"brand\": \"BZ Medico\",\r\n        \"status\":
0,\r\n        \"price\": 120\r\n}",
                    "options": {
                        "raw": {
                            "language": "json"
                        }
                    }
                },
                "url": {
                    "raw": "{{BaseUrl}}/update-product-status",
                    "host": [
                        "{{BaseUrl}}"
                    ],
                    "path": [
                        "update-product-status"
                    ]
                }
            },
            "response": []
        },
        {
            "name": "DeleteThe_Product",
            "event": [
                {
                    "listen": "test",
                    "script": {
                        "exec": [
                            "pm.test(\"Status code is 200\", function () {\r",
                            "    pm.response.to.have.status(200);\r",
                            "});\r",
                            "pm.test(\"Body matches string\", function () {\r",
```

```
                            "    pm.expect(pm.response.text()).to.include(\"Product with ID
101 Deleted Successfully.\");\r",
                            "});"
                        ],
                        "type": "text/javascript"
                    }
                }
            ],
            "request": {
                "method": "DELETE",
                "header": [],
                "url": {
                    "raw": "{{BaseUrl}}/delete-product?id=101",
                    "host": [
                        "{{BaseUrl}}"
                    ],
                    "path": [
                        "delete-product"
                    ],
                    "query": [
                        {
                            "key": "id",
                            "value": "101"
                        }
                    ]
                }
            },
            "response": []
        }
    ],
    "event": [
        {
            "listen": "prerequest",
            "script": {
                "type": "text/javascript",
                "exec": [
                    ""
                ]
            }
        },
        {
            "listen": "test",
            "script": {
                "type": "text/javascript",
                "exec": [
                    ""
                ]
            }
        }
    ],
    "variable": [
        {
            "key": "BaseUrl",
            "value": "http://localhost:9010",
            "type": "string"
        }
    ]
}
```