

Source Code

RestAssured

APIEndpoints:

```
package medicare.testassured;

import java.util.HashMap;

import org.testng.annotations.Test;

import io.restassured.RestAssured;

public class APIEndpoints {

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

        RestAssured.given()
            .baseUrl("http://localhost:9010")
            .basePath("/get-products")
            .when().get()
            .then().statusCode(200)
            .log().all();
    }
}
```

```
@Test(priority=2)
public void get_all_users() {

    RestAssured.given()
        .baseUrl("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()
```

```

        .baseUrl("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()
        .baseUrl("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()
        .baseUrl("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()  
        .baseUrl("http://localhost:9010")  
        .basePath("/delete-product")  
        .queryParams("id", "101")  
        .when().delete()  
        .then().statusCode(200)  
        .log().all();
```

```
}
```

```
}
```

TestNG

Base:

```
package medicare.base;

import java.awt.Desktop;
import java.io.File;
import java.io.IOException;
import java.time.Duration;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.ITestContext;
import org.testng.ITestResult;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.AfterSuite;
import org.testng.annotations.BeforeSuite;
import org.testng.annotations.BeforeTest;

import com.aventstack.extentreports.ExtentReports;
import com.aventstack.extentreports.ExtentTest;
import com.aventstack.extentreports.reporter.ExtentSparkReporter;
import com.github.dockerjava.transport.DockerHttpClient.Request.Method;

public class Base {

    public static WebDriver driver;
```

```
public static ExtentReports extentreports;
```

```
public static ExtentTest extentTest;
```

```
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/");
```

```
}
```

```
@BeforeTest
```

```
public void getNameMethod(ITestContext context) {
```

```
    extentTest = extentreports.createTest(context.getName());
```

```
}
```

```
@BeforeSuite // this method will be excuted before suite begins its execution
```

```
public void InitalizeExtentReport() {
```

```
    ExtentSparkReporter sparkreporter = new  
ExtentSparkReporter("register_user_report.html");
```

```
    extentreports = new ExtentReports();
```

```
    extentreports.attachReporter(sparkreporter);
```

```
    // on the report display more information about OS, browser, java etc
```

```
    extentreports.setSystemInfo("OS", System.getProperty("os.name"));
```

```
    extentreports.setSystemInfo("JAVA", System.getProperty("java.version"));
```

```
}
```

@AfterSuite

```
public void generateReports() throws IOException {  
    extentreports.flush();  
    Desktop.getDesktop().browse(new File("register_user_report.html").toURI());  
    driver.close();  
}
```

@AfterMethod

```
public void generateTestStatus(Method m, ITestResult result) {  
    if (result.getStatus() == ITestResult.FAILURE) {  
        System.out.println("Capture Screenshot");  
        extentTest.fail(result.getThrowable());  
    } else if (result.getStatus() == ITestResult.SUCCESS) {  
        //extentTest.pass(m.getName() + " is passed");  
    }  
}
```

```
}
```


RegisterPage:

```
package medicare.pages;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;

public class RegisterPage {

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

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

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

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

    @FindBy(xpath="//button[@class='btn btn-success']")
    WebElement Registerbutton;

    public RegisterPage(WebDriver driver) {
```

```
        PageFactory.initElements(driver, this);
    }

    public void new_user_Register_here() {

        Registration.click();

    }

    public void RegisterUser(String name1, String email1, String password1) {

        // Actions

        name.clear();
        name.sendKeys(name1);

        email.clear();
        email.sendKeys(email1);

        password.clear();
        password.sendKeys(password1);

        Registerbutton.click();
    }
}
```

LoginPage:

```
package medicare.pages;
```

```
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 {
```

```
    //WebDriver driver;
```

```
    // Create PageFactory
```

```
    @FindBy(id = "email")
```

```
    WebElement email;
```

```
    @FindBy(id = "password")
```

```
    WebElement password;
```

```
    @FindBy(xpath = "//button[@class=\"btn btn-success\"]")
```

```
    WebElement login;
```

```
    @FindBy(linkText = "Logout")
```

```
    WebElement logout;
```

```
public LoginPage(WebDriver driver) {
```

```
    PageFactory.initElements(driver, this);
```

```
}
```

```
/*public void LoginUser(String email1, String password1) {
```

```
    // Actions
```

```
    email.clear();
```

```
    email.sendKeys(email1);
```

```
    password.clear();
```

```
    password.sendKeys(password1);
```

```
    login.click();
```

```
*/
```

```
public void LoginUser()
{
    email.sendKeys("saikiran1234@gmail.com");
    password.sendKeys("123456");
    login.click();
}

public void Logout() {
    logout.click();
}

}
```

AddProductToCartPage:

```
package medicare.pages;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;

public class AddProductToCartPage {

    // Find the locator with the help of PageFactory

    /*@FindBy(id = "search-product")
    WebElement search;

    @FindBy(id = "cart101")
    WebElement AddtoCart;

    @FindBy(id = "search-product-button")
    WebElement submit;*/

    @FindBy(xpath="//a[@id='product103']")
    //@FindBy(id = "product103")
    //@FindBy(linkText = "View Product")
    //@FindBy(xpath = "//a[@href='view-product?id=106']")
    WebElement ViewProductbtn;

    @FindBy(xpath = "//a[@class='btn btn-success']")
```

```
//@FindBy(id = "cart103")
//@FindBy(xpath = "//a[@id='cart110']")
//@FindBy(linkText = "Add to Cart")
WebElement AddtoCartbtn;
```

```
//WebElement AddtoCart;
```

```
@FindBy(xpath = "//a[@class='nav-link text-success']")
WebElement Cartbtn;
```

```
public AddProductToCartPage(WebDriver driver) {
```

```
    PageFactory.initElements(driver, this);
```

```
}
```

```
// Actions
```

```
/*public void SearchProduct(String productname1) {
```

```
    search.clear();
```

```
    search.sendKeys(productname1);
```

```
*/
```

```
public void ViewProducts() {
```

```
ViewProductbtn.click();
```

```
}
```

```
public void AddToCart() {
```

```
    AddtoCartbtn.click();
```

```
}
```

```
public void Cart() {
```

```
    Cartbtn.click();
```

```
}
```

```
}
```


PlaceOrderPage:

```
package medicare.pages;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;

public class PlaceOrderPage {

    @FindBy(xpath = "//a[@class='nav-link text-success']")
    WebElement Cart;

    @FindBy(xpath = "//a[@id='place-order']")
    WebElement PlaceOrderbtn;

    public PlaceOrderPage(WebDriver driver) {
        PageFactory.initElements(driver, this);
    }

    /*public void ViewProducts() {
        ViewProduct.click();
    }
    public void AddToCart() {
        AddtoCart.click();
    }
    */

    public void Cart() {
        Cart.click();
    }

    public void PlaceOrders() {
        PlaceOrderbtn.click();
    }

}
```

SearchPage:

```
package medicare.pages;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;

public class SearchPage {

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

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

    public SearchPage(WebDriver driver) {

        PageFactory.initElements(driver, this);
    }

    public void Search() {

        //Search.click();
        Search.sendKeys("Limcee Chewable Tablet Orange");
        Submit.click();
    }

}
```

RegisterPageTest:

```
package medicare.testpages;

import java.io.IOException;

import org.testng.annotations.BeforeMethod;
import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;

import Utilities.Xls_DataProvider;
import medicare.base.Base;

//import medicare.pages.HomePage;
import medicare.pages.RegisterPage;

public class RegisterPageTest extends Base {

    //    HomePage hp;
    //    RegisterPage rp;

    @BeforeMethod
    public void openApplication() throws InterruptedException {

        openBrowser("Chrome");

        //hp = new HomePage(driver);

        Thread.sleep(2000);

        rp = new RegisterPage(driver);
    }

    @Test(priority = '1', dataProvider = "testdata")
    public void RegisterUser(String name1, String email1, String password1)
    {

        rp.new_user_Register_here();

        rp.RegisterUser(name1,email1,password1);

    }

    @DataProvider(name = "testdata")
    public Object[][] datasupplier() throws IOException {

        Object[][] input = Xls_DataProvider.getTestData("Sheet1");
        return input;
    }

}
```

LoginPageTest:

```
package medicare.testpages;

//import java.io.IOException;

import org.testng.annotations.BeforeClass;
//import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;

//import Utilities.Xls_DataProvider2;

import medicare.pages.LoginPage;
import medicare.base.Base;

public class LoginPageTest extends Base {

    LoginPage lp;

    @BeforeClass
    public void OpenApp() throws InterruptedException {

        openBrowser("Chrome");
        Thread.sleep(2000);
        lp = new LoginPage(driver);

    }

    /*@Test(priority = '1', dataProvider = "testdata")
    public void LoginUser(String email1, String password1) throws
    InterruptedException {

        lp.LoginUser(email1,password1);
        lp.Logout();

    }

    @DataProvider(name = "testdata")
    public Object[][] datasupplier() throws IOException {

        Object[][] input = Xls_DataProvider2.getTestData("Sheet1");
        return input;

    }*/

    @Test(priority='1')
    public void test_login()
    {
        lp.LoginUser();
    }

}
```

AddProductToCartTest:

```
package medicare.testpages;

//import java.io.IOException;

//import org.testng.annotations.BeforeMethod;
import org.testng.annotations.BeforeTest;
//import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;

//import Utilities.Xls_DataProvider2;
import medicare.base.Base;
import medicare.pages.AddProductToCartPage;
import medicare.pages.LoginPage;
//import medicare.pages.RegisterPage;

public class AddProductToCartTest extends Base {

    LoginPage lp;
    //RegisterPage rp;
    AddProductToCartPage apc;

    @BeforeTest
    public void openApplication() throws InterruptedException {

        openBrowser("Chrome");

        lp = new LoginPage(driver);

        apc = new AddProductToCartPage(driver);

    }

    /*@Test(priority=1)

    public void ViewProduct() {
        apc.ViewProducts();
    }*/

    /* @Test(priority = '1', dataProvider = "testdata")
    public void LoginUser(String email1, String password1) throws
    InterruptedException {

        lp.LoginUser(email1,password1);
        Thread.sleep(2000);
        apc.ViewProducts();
        Thread.sleep(2000);
        apc.AddToCart();
```

```

        Thread.sleep(2000);
        apc.Cart();

    }

    @DataProvider(name = "testdata")
    public Object[][] datasupplier() throws IOException {

        Object[][] input = Xls_DataProvider2.getTestData("Sheet1");
        return input;
    }

    /*
    @Test(priority='2')

    public void AddToCart() {
        apc.AddtoCart;
    }*/

    @Test(priority='1')
    public void test_login()
    {
        lp.LoginUser();
    }

    @Test(priority='2')
    public void ViewProducts() throws InterruptedException
    {
        apc.ViewProducts();
    }

    @Test(priority='3')
    public void test_add_product_to_cart() throws InterruptedException
    {
        apc.AddToCart();
    }

    @Test(priority='4')
    public void cart() throws InterruptedException
    {
        apc.Cart();
    }

}

```

PlaceOrderPageTest:

```
package medicare.testpages;

import org.testng.annotations.BeforeTest;
import org.testng.annotations.Test;

import medicare.pages.AddProductToCartPage;
import medicare.pages.LoginPage;
import medicare.pages.PlaceOrderPage;
import medicare.base.Base;

public class PlaceOrderPageTest extends Base {

    LoginPage lp;

    AddProductToCartPage apc;

    PlaceOrderPage po;

    @BeforeTest
    public void openApplication() throws InterruptedException {

        openBrowser("Chrome");

        lp = new LoginPage(driver);

        apc = new AddProductToCartPage(driver);
        po = new PlaceOrderPage(driver);

    }

    @Test(priority='1')
    public void test_login()
    {
        lp.LoginUser();
    }

    @Test(priority='2')
    public void cart() throws InterruptedException
    {
        apc.Cart();
    }

    @Test(priority='3')
    public void PlaceOrder() throws InterruptedException
    {
        po.PlaceOrders();
    }

}
```

SearchPageTest:

```
package medicare.testpages;

//import java.io.IOException;

import org.testng.annotations.BeforeMethod;
//import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;

//import Utilities.Xls_DataProvider2;
import medicare.base.Base;
import medicare.pages.LoginPage;
import medicare.pages.SearchPage;

public class SearchPageTest extends Base {

    LoginPage lp;

    SearchPage sp;

    @BeforeMethod

    public void start_browser() throws InterruptedException {

        openBrowser("Chrome");

        //rp = new RegisterPage(driver);

        lp = new LoginPage(driver);
```



```
//Thread.sleep(2000);
```

```
sp = new SearchPage(driver);
```

```
}
```

```
@Test(priority='1')
```

```
public void test_login() throws InterruptedException
```

```
{
```

```
    lp.LoginUser();
```

```
    Thread.sleep(2000);
```

```
    sp.Search();
```

```
}
```

```
/*@Test(priority='2')
```

```
public void Search() throws InterruptedException
```

```
{
```

```
    sp.Search();
```

```
*/
```

```
}
```

ItestListenerClass:

```
package medicare.testpages;

import org.testng.ITestContext;
import org.testng.ITestListener;
import org.testng.ITestResult;

public class ItestListenerClass implements ITestListener {

    public void onTestStart(ITestResult result) {

        System.out.println("Test Method start");

    }

    public void onTestSuccess(ITestResult result) {

        System.out.println("Test Method success");

    }


    public void onTestFailure(ITestResult result) {

        System.out.println("Test Method Failed");

    }


    public void onTestSkipped(ITestResult result) {

        System.out.println("Test Method skipped");

    }

}
```

```
}

public void onTestFailedWithTimeout(ITestResult result) {

    System.out.println("Test Method fail with timeout");

}

public void onStart(ITestContext context) {

    System.out.println("Testing has started");

}

public void onFinish(ITestContext context) {

    System.out.println("Testing has finished");

}

}
```

Xls DataProvider:

```
package Utilities;
```

```
import java.io.FileInputStream;
```

```
import java.io.IOException;
```

```
import org.apache.poi.ss.usermodel.Sheet;
```

```
import org.apache.poi.ss.usermodel.Workbook;
```

```
import org.apache.poi.ss.usermodel.WorkbookFactory;
```

```
public class Xls_DataProvider {
```

```
    static Workbook book;
```

```
    static Sheet sheet;
```

```
    public static String testdata_sheet_path =  
    "C:\\Users\\Sai\\OneDrive\\Desktop\\mytestdata\\RegisterMedicare.xlsx";
```

```
    public static Object[][] getTestData(String sheetName) throws IOException {
```

```
        FileInputStream file = null;
```

```
        file = new FileInputStream(testdata_sheet_path);
```

```
        book = WorkbookFactory.create(file);
```

```
        sheet = book.getSheet(sheetName);
```

```
        Object[][] inputdata = new  
        Object[sheet.getLastRowNum()][sheet.getRow(0).getLastCellNum()];
```

```
for (int i = 0; i < sheet.getLastRowNum(); i++) {  
  
    for (int j = 0; j < sheet.getRow(0).getLastCellNum(); j++) {  
  
        inputdata[i][j] = sheet.getRow(i + 1).getCell(j).toString();  
    }  
}  
  
return inputdata;  
  
}  
  
}
```

TestNGRunner.xml:

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

    <listeners>

        <listener class-name="medicare.testpages.ItestListenerClass"></listener>
    </listeners>
    <test name="POMFramework">
        <classes>

            <class name="medicare.testpages.RegisterPageTest"></class>
            <class name="medicare.testpages.LoginPageTest"></class>
            <class name="medicare.testpages.AddProductToCartTest"></class>
            <class name="medicare.testpages.PlaceOrderPageTest"></class>

        </classes>

    </test>

</suite>
```

JMeter

Medicare JMeter.jmx:

```
<?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="capstoneproject"
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">&quot;Not_A Brand&quot;;v=&quot;8&quot;;
&quot;Chromium&quot;;v=&quot;120&quot;; &quot;Google
Chrome&quot;;v=&quot;120&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>
      </collectionProp>
    </HeaderManager>
  </hashTree>
</jmeterTestPlan>
```

```

</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>

<elementProp name="Sec-Fetch-Dest" elementType="Header">

  <stringProp name="Header.name">Sec-Fetch-Dest</stringProp>

  <stringProp name="Header.value">document</stringProp>

</elementProp>

<elementProp name="Sec-Fetch-User" elementType="Header">

  <stringProp name="Header.name">Sec-Fetch-User</stringProp>

  <stringProp name="Header.value">?1</stringProp>

</elementProp>

<elementProp name="Sec-Fetch-Mode" elementType="Header">

```



```

    <stringProp name="Header.name">Sec-Fetch-Mode</stringProp>
    <stringProp name="Header.value">navigate</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>
</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>

```

```

</DNSSCacheManager>

<hashTree/>

<AuthManager guiclass="AuthPanel" testclass="AuthManager" testname="HTTP
Authorization Manager" enabled="true">

  <collectionProp name="AuthManager.auth_list"/>

  <boolProp name="AuthManager.controlledByThreadGroup">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.controlledByThreadGroup">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">1</stringProp>

    <boolProp name="LoopController.continue_forever">false</boolProp>

  </elementProp>

```

```

<stringProp name="ThreadGroup.num_threads">5</stringProp>
<stringProp name="ThreadGroup.ramp_time">20</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="http://localhost:9010/" 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.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>
    <HeaderManager guiclass="HeaderPanel" testclass="HeaderManager" testname="HTTP
Header manager" enabled="true">
        <collectionProp name="HeaderManager.headers">
            <elementProp name="Sec-Fetch-Site" elementType="Header">
                <stringProp name="Header.name">Sec-Fetch-Site</stringProp>
                <stringProp name="Header.value">none</stringProp>
            </elementProp>
        </collectionProp>
    </HeaderManager>
</hashTree/>
    <ConstantTimer guiclass="ConstantTimerGui" testclass="ConstantTimer"
testname="Constant Timer" enabled="true">
        <stringProp name="ConstantTimer.delay">0</stringProp>
    </ConstantTimer>
</hashTree/>
</hashTree>
    <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy"
testname="http://localhost:9010/register" 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">register</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>
    <HeaderManager guiclass="HeaderPanel" testclass="HeaderManager" testname="HTTP
Header manager" enabled="true">
    <collectionProp name="HeaderManager.headers">
    <elementProp name="Sec-Fetch-Site" elementType="Header">
    <stringProp name="Header.name">Sec-Fetch-Site</stringProp>
    <stringProp name="Header.value">same-origin</stringProp>
    </elementProp>

```

```
</collectionProp>
</HeaderManager>
<hashTree/>
  <ConstantTimer guiclass="ConstantTimerGui" testclass="ConstantTimer"
testname="Constant Timer" enabled="true">
    <stringProp name="ConstantTimer.delay">2773</stringProp>
  </ConstantTimer>
<hashTree/>
</hashTree>
  <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy"
testname="http://localhost:9010/register-user" 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">12345</stringProp>
          <stringProp name="Argument.metadata">=</stringProp>
          <boolProp name="HTTPArgument.use_equals">true</boolProp>
        </elementProp>
        <elementProp name="name" elementType="HTTPArgument">
          <boolProp name="HTTPArgument.always_encode">true</boolProp>
          <stringProp name="Argument.name">name</stringProp>
          <stringProp name="Argument.value">user</stringProp>
          <stringProp name="Argument.metadata">=</stringProp>
          <boolProp name="HTTPArgument.use_equals">true</boolProp>
        </elementProp>
      </collectionProp>
    </elementProp>
  </HTTPSamplerProxy>
</hashTree>
```

```

    <elementProp name="email" elementType="HTTPArgument">
      <boolProp name="HTTPArgument.always_encode">true</boolProp>
      <stringProp name="Argument.name">email</stringProp>
      <stringProp name="Argument.value">user@medicure.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">register-user</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>
      <elementProp name="Sec-Fetch-Site" elementType="Header">
        <stringProp name="Header.name">Sec-Fetch-Site</stringProp>
        <stringProp name="Header.value">same-origin</stringProp>
      </elementProp>
    </collectionProp>
  </HeaderManager>
</hashTree/>

<ConstantTimer guiclass="ConstantTimerGui" testclass="ConstantTimer"
testname="Constant Timer" enabled="true">
  <stringProp name="ConstantTimer.delay">22566</stringProp>
</ConstantTimer>
</hashTree/>
</hashTree>

<HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy"
testname="http://localhost:9010/view-product?id=104" 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">104</stringProp>
        <stringProp name="Argument.metadata">=</stringProp>
      </elementProp>
    </collectionProp>
  </elementProp>
</HTTPSamplerProxy>

```



```

    <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">view-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>
  <HeaderManager guiclass="HeaderPanel" testclass="HeaderManager" testname="HTTP
Header manager" enabled="true">
    <collectionProp name="HeaderManager.headers">
      <elementProp name="Sec-Fetch-Site" elementType="Header">
        <stringProp name="Header.name">Sec-Fetch-Site</stringProp>
        <stringProp name="Header.value">same-origin</stringProp>
      </elementProp>

```

```
</collectionProp>
</HeaderManager>
<hashTree/>
  <ConstantTimer guiclass="ConstantTimerGui" testclass="ConstantTimer"
testname="Constant Timer" enabled="true">
    <stringProp name="ConstantTimer.delay">12418</stringProp>
  </ConstantTimer>
<hashTree/>
</hashTree>
  <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy"
testname="http://localhost:9010/add-to-cart?id=104" 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">104</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>
    <HeaderManager guiclass="HeaderPanel" testclass="HeaderManager" testname="HTTP
Header manager" enabled="true">
        <collectionProp name="HeaderManager.headers">
            <elementProp name="Sec-Fetch-Site" elementType="Header">
                <stringProp name="Header.name">Sec-Fetch-Site</stringProp>
                <stringProp name="Header.value">same-origin</stringProp>
            </elementProp>
        </collectionProp>
    </HeaderManager>
</hashTree/>
    <ConstantTimer guiclass="ConstantTimerGui" testclass="ConstantTimer"
testname="Constant Timer" enabled="true">
        <stringProp name="ConstantTimer.delay">4081</stringProp>
    </ConstantTimer>
</hashTree/>
</hashTree>

```

```
<HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy"
testname="http://localhost:9010/home" 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>
```

```
<HeaderManager guiclass="HeaderPanel" testclass="HeaderManager" testname="HTTP
Header manager" enabled="true">
```

```
<collectionProp name="HeaderManager.headers">
```

```
<elementProp name="Sec-Fetch-Site" elementType="Header">
```

```
<stringProp name="Header.name">Sec-Fetch-Site</stringProp>
```

```
<stringProp name="Header.value">same-origin</stringProp>
</elementProp>
</collectionProp>
</HeaderManager>
<hashTree/>
<ConstantTimer guiclass="ConstantTimerGui" testclass="ConstantTimer"
testname="Constant Timer" enabled="true">
  <stringProp name="ConstantTimer.delay">3472</stringProp>
</ConstantTimer>
<hashTree/>
</hashTree>
<HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy"
testname="http://localhost:9010/cart" 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">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>
    <HeaderManager guiclass="HeaderPanel" testclass="HeaderManager" testname="HTTP
Header manager" enabled="true">
      <collectionProp name="HeaderManager.headers">
        <elementProp name="Sec-Fetch-Site" elementType="Header">
          <stringProp name="Header.name">Sec-Fetch-Site</stringProp>
          <stringProp name="Header.value">same-origin</stringProp>
        </elementProp>
      </collectionProp>
    </HeaderManager>
  </hashTree/>
  <ConstantTimer guiclass="ConstantTimerGui" testclass="ConstantTimer"
testname="Constant Timer" enabled="true">
    <stringProp name="ConstantTimer.delay">6909</stringProp>
  </ConstantTimer>
</hashTree/>
</hashTree>
  <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy"
testname="http://localhost:9010/place-order" 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">place-order</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>
  <HeaderManager guiclass="HeaderPanel" testclass="HeaderManager" testname="HTTP
Header manager" enabled="true">
    <collectionProp name="HeaderManager.headers">
      <elementProp name="Sec-Fetch-Site" elementType="Header">
        <stringProp name="Header.name">Sec-Fetch-Site</stringProp>
        <stringProp name="Header.value">same-origin</stringProp>
      </elementProp>
    </collectionProp>
  </HeaderManager>
</hashTree/>
  <ConstantTimer guiclass="ConstantTimerGui" testclass="ConstantTimer"
testname="Constant Timer" enabled="true">

```

```
<stringProp name="ConstantTimer.delay">3924</stringProp>
</ConstantTimer>
<hashTree/>
</hashTree>
<HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy"
testname="http://localhost:9010/home" 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>
```


<HeaderManager guiclass="HeaderPanel" testclass="HeaderManager" testname="HTTP Header manager" enabled="true">

<collectionProp name="HeaderManager.headers">

<elementProp name="Sec-Fetch-Site" elementType="Header">

<stringProp name="Header.name">Sec-Fetch-Site</stringProp>

<stringProp name="Header.value">same-origin</stringProp>

</elementProp>

</collectionProp>

</HeaderManager>

<hashTree/>

<ConstantTimer guiclass="ConstantTimerGui" testclass="ConstantTimer" testname="Constant Timer" enabled="true">

<stringProp name="ConstantTimer.delay">4544</stringProp>

</ConstantTimer>

<hashTree/>

</hashTree>

<HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="http://localhost:9010/orders" 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">orders</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>
    <HeaderManager guiclass="HeaderPanel" testclass="HeaderManager" testname="HTTP
Header manager" enabled="true">
        <collectionProp name="HeaderManager.headers">
            <elementProp name="Sec-Fetch-Site" elementType="Header">
                <stringProp name="Header.name">Sec-Fetch-Site</stringProp>
                <stringProp name="Header.value">same-origin</stringProp>
            </elementProp>
        </collectionProp>
    </HeaderManager>
</hashTree/>
    <ConstantTimer guiclass="ConstantTimerGui" testclass="ConstantTimer"
testname="Constant Timer" enabled="true">
        <stringProp name="ConstantTimer.delay">1629</stringProp>
    </ConstantTimer>
</hashTree/>
</hashTree>
    <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy"
testname="http://localhost:9010/home" 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>
    <HeaderManager guiclass="HeaderPanel" testclass="HeaderManager" testname="HTTP
Header manager" enabled="true">
    <collectionProp name="HeaderManager.headers">
    <elementProp name="Sec-Fetch-Site" elementType="Header">
    <stringProp name="Header.name">Sec-Fetch-Site</stringProp>
    <stringProp name="Header.value">same-origin</stringProp>
    </elementProp>

```

```
</collectionProp>
</HeaderManager>
<hashTree/>
  <ConstantTimer guiclass="ConstantTimerGui" testclass="ConstantTimer"
testname="Constant Timer" enabled="true">
    <stringProp name="ConstantTimer.delay">4031</stringProp>
  </ConstantTimer>
  <hashTree/>
</hashTree>
  <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy"
testname="http://localhost:9010/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>
    <HeaderManager guiclass="HeaderPanel" testclass="HeaderManager" testname="HTTP
Header manager" enabled="true">
      <collectionProp name="HeaderManager.headers">
        <elementProp name="Sec-Fetch-Site" elementType="Header">
          <stringProp name="Header.name">Sec-Fetch-Site</stringProp>
          <stringProp name="Header.value">same-origin</stringProp>
        </elementProp>
      </collectionProp>
    </HeaderManager>
  </hashTree/>
  <ConstantTimer guiclass="ConstantTimerGui" testclass="ConstantTimer"
testname="Constant Timer" enabled="true">
    <stringProp name="ConstantTimer.delay">18117</stringProp>
  </ConstantTimer>
</hashTree/>
</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>
      </value>
    </objProp>
  </ResultCollector>
</hashTree>

```

```
<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>
<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/>
</hashTree>
</hashTree>
</hashTree>
</jmeterTestPlan>
```


Cucumber

Medicare Cucumber.feature:

Feature: To test medicare apis for capstone project

Scenario: Retrieve the list of all products in the store

Given User Enters Medicare base URL and Authorization

When User executes HTTP get method

Then Validate the response status code

Scenario: Retrieve the list of all registered users

Given User Enters Medicare base URL and Authorization

When User executes HTTP get method

Then Validate the response status code

Scenario: Add the product

Given User Enters Medicare base URL and Authorization

When User executes HTTP post method

Then Validate the response status code

Scenario: Update the product

Given User Enters Medicare base URL and Authorization

When User executes HTTP put method

Then Validate the response status code

Scenario: Update the product status

Given User Enters Medicare base URL and Authorization

When User executes HTTP put method

Then Validate the response status code

Scenario: Delete the product

Given User Enters Medicare base URL and Authorization

When User executes HTTP Delete method

Then Validate the response status code

MedicareAPIStepsTest:

```
package steps;

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 MedicareAPIStepsTest {

    @Given("User Enters Medicare base URL and Authorization")
    public void user_enters_medicare_base_url_and_authorization() {
        RestAssured.given()
            .baseUrl("http://localhost:9010")
            .basePath("/get-products")
            .when().get()
            .then().statusCode(200)
            .log().all();
    }

    @When("User executes HTTP get method")
    public void user_executes_http_get_method() {
        RestAssured.given()
            .baseUrl("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()
            .baseUrl("http://localhost:9010")
            .basePath("/get-products")
            .when().get()
            .then().statusCode(200)
            .log().all();
    }

    @When("User executes HTTP post method")
    public void user_executes_http_post_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()
            .baseUrl("http://localhost:9010")
            .basePath("/add-product")
            .contentType("application/json")
            .body(map)
            .when().post()
            .then().statusCode(200).log().all();
    }

    @When("User executes HTTP put method")
    public void user_executes_http_put_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()
            .baseUrl("http://localhost:9010")
            .basePath("/update-product")
            .contentType("application/json")
            .body(map)
            .when().put()
            .then().statusCode(200).log().all();
    }

    @When("User executes HTTP Delete method")
    public void user_executes_http_delete_method() {

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

```

POSTMAN

Medicare POSTMAN:

```
{
  "info": {
    "_postman_id": "4f275466-1aa4-481a-ab94-65e8a220bc70",
    "name": "Medicare_POSTMAN",
    "schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json",
    "_exporter_id": "31715054"
  },
  "item": [
    {
      "name": "get-products",
      "request": {
        "method": "GET",
        "header": [],
        "url": {
          "raw": "http://localhost:9010/get-products",
          "protocol": "http",
          "host": [
            "localhost"
          ],
          "port": "9010",
          "path": [
            "get-products"
          ]
        }
      }
    }
  ]
}
```

```
    },
    "response": []
  },
  {
    "name": "get-users",
    "request": {
      "method": "GET",
      "header": [],
      "url": {
        "raw": "http://localhost:9010/get-users",
        "protocol": "http",
        "host": [
          "localhost"
        ],
        "port": "9010",
        "path": [
          "get-users"
        ]
      }
    },
    "response": []
  },
  {
    "name": "add-product",
    "event": [
      {
        "listen": "test",
        "script": {
```

```

        "exec": [
            ""
        ],
        "type": "text/javascript"
    }
}

],
"request": {
    "method": "POST",
    "header": [],
    "body": {
        "mode": "raw",
        "raw": "{\r\n    \"id\": 999,\r\n    \"image\":\r\n    \"1.png\",\r\n    \"name\": \"Disprin\",\r\n    \"category\": \"medicine\",\r\n    \"brand\":\r\n    \"BZ Medico\",\r\n    \"status\": 1,\r\n    \"price\": 100\r\n}\r\n",
        "options": {
            "raw": {
                "language": "json"
            }
        }
    },
    "url": {
        "raw": "http://localhost:9010/add-product",
        "protocol": "http",
        "host": [
            "localhost"
        ],
        "port": "9010",
        "path": [

```

```

        "add-product"
    ]
}

},
"response": []
},
{
    "name": "update-product",
    "request": {
        "method": "PUT",
        "header": [],
        "body": {
            "mode": "raw",
            "raw": "{\r\n    \"id\": 999,\r\n    \"image\":\r\n\r\n    \"name\": \"Disprin+\", \r\n    \"category\": \"medicine\", \r\n    \"brand\":\r\n    \"BZ Medico\", \r\n    \"status\": 1, \r\n    \"price\": 120\r\n}\r\n",
            "options": {
                "raw": {
                    "language": "json"
                }
            }
        }
    },
    "url": {
        "raw": "http://localhost:9010/update-product",
        "protocol": "http",
        "host": [
            "localhost"
        ],
        "port": "9010",

```

```

        "path": [
            "update-product"
        ]
    },
    "response": []
},
{
    "name": "update-product-status",
    "request": {
        "method": "PUT",
        "header": [],
        "body": {
            "mode": "raw",
            "raw": "{\r\n    \"id\": 999,\r\n    \"image\":\r\n\r\n    \"name\": \"Disprin+\", \r\n    \"category\": \"medicine\", \r\n    \"brand\":\r\n    \"BZ Medico\", \r\n    \"status\": 0, \r\n    \"price\": 120\r\n}\r\n",
            "options": {
                "raw": {
                    "language": "json"
                }
            }
        },
        "url": {
            "raw": "http://localhost:9010/update-product-status",
            "protocol": "http",
            "host": [
                "localhost"
            ],

```



```
        "port": "9010",
        "path": [
            "update-product-status"
        ]
    },
    },
    "response": []
},
{
    "name": "delete-product",
    "request": {
        "method": "DELETE",
        "header": [],
        "url": {
            "raw": "http://localhost:9010/delete-product?id=101",
            "protocol": "http",
            "host": [
                "localhost"
            ],
            "port": "9010",
            "path": [
                "delete-product"
            ],
            "query": [
                {
                    "key": "id",
                    "value": "101"
                }
            ]
        }
    }
}
```

```
    ]
  }
},
"response": []
}
]
```