**Assignment 1-5(PostMan.Json File)**

{

    "info": {

        "\_postman\_id": "703cabb2-6955-41ab-a2ad-f34678980bce",

        "name": "Pet\_ID\_Testing",

        "schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json",

        "\_exporter\_id": "31715611",

        "\_collection\_link": "https://lively-trinity-694168.postman.co/workspace/MyWorkSpace1~260fbfc2-5763-495f-8b10-1b1f4eb58c24/collection/31715611-703cabb2-6955-41ab-a2ad-f34678980bce?action=share&source=collection\_link&creator=31715611"

    },

    "item": [

        {

            "name": "CreatePetID",

            "event": [

                {

                    "listen": "test",

                    "script": {

                        "exec": [

                            "pm.test(\"Status code is 200\", function () {\r",

                            "    pm.response.to.have.status(200);\r",

                            "});\r",

                            "\r",

                            "pm.test(\"Body matches string\", function () {\r",

                            "    pm.expect(pm.response.text()).to.include(\"available\");\r",

                            "});"

                        ],

                        "type": "text/javascript"

                    }

                },

                {

                    "listen": "prerequest",

                    "script": {

                        "exec": [

                            ""

                        ],

                        "type": "text/javascript"

                    }

                }

            ],

            "request": {

                "method": "POST",

                "header": [],

                "body": {

                    "mode": "raw",

                    "raw": "{\r\n    \"id\": {{petID}},\r\n    \"category\": {\r\n        \"id\": 0,\r\n        \"name\": \"string\"\r\n    },\r\n    \"name\": \"{{petName}}\",\r\n    \"photoUrls\": [\r\n        \"string\"\r\n    ],\r\n    \"tags\": [\r\n        {\r\n            \"id\": 0,\r\n            \"name\": \"string\"\r\n        }\r\n    ],\r\n    \"status\": \"available\"\r\n}",

                    "options": {

                        "raw": {

                            "language": "json"

                        }

                    }

                },

                "url": {

                    "raw": "https://petstore.swagger.io/v2/pet",

                    "protocol": "https",

                    "host": [

                        "petstore",

                        "swagger",

                        "io"

                    ],

                    "path": [

                        "v2",

                        "pet"

                    ]

                }

            },

            "response": []

        },

        {

            "name": "GetPetID",

            "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(\"available\");\r",

                            "});"

                        ],

                        "type": "text/javascript"

                    }

                }

            ],

            "request": {

                "method": "GET",

                "header": [

                    {

                        "key": "accept",

                        "value": "application/json"

                    },

                    {

                        "key": "api\_key",

                        "value": "12345"

                    }

                ],

                "url": {

                    "raw": "https://petstore.swagger.io/v2/pet/{{petID}}",

                    "protocol": "https",

                    "host": [

                        "petstore",

                        "swagger",

                        "io"

                    ],

                    "path": [

                        "v2",

                        "pet",

                        "{{petID}}"

                    ]

                }

            },

            "response": []

        },

        {

            "name": "DeletePet",

            "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(\"unknown\");\r",

                            "});"

                        ],

                        "type": "text/javascript"

                    }

                }

            ],

            "request": {

                "method": "DELETE",

                "header": [

                    {

                        "key": "accept",

                        "value": "application/json"

                    },

                    {

                        "key": "api\_key",

                        "value": "12345"

                    }

                ],

                "url": {

                    "raw": "https://petstore.swagger.io/v2/pet/{{petID}}",

                    "protocol": "https",

                    "host": [

                        "petstore",

                        "swagger",

                        "io"

                    ],

                    "path": [

                        "v2",

                        "pet",

                        "{{petID}}"

                    ]

                }

            },

            "response": []

        },

        {

            "name": "Assignmen002",

            "request": {

                "method": "PUT",

                "header": [],

                "body": {

                    "mode": "raw",

                    "raw": "{\r\n\"id\": 9223372016900013000, \"category\": {\r\n\"id\": 20021,\r\n\"name\": \"string\" },\r\n\"name\": \"doggie\", \"photoUrls\": [\r\n\"string\"\r\n], \"tags\": [\r\n{\r\n\"id\": 0,\r\n\"name\": \"string\"\r\n}\r\n],\r\n\"status\": \"{{status}}\" \r\n}",

                    "options": {

                        "raw": {

                            "language": "json"

                        }

                    }

                },

                "url": {

                    "raw": "{{testUrl}}",

                    "host": [

                        "{{testUrl}}"

                    ]

                }

            },

            "response": []

        },

        {

            "name": "Assignment003",

            "event": [

                {

                    "listen": "test",

                    "script": {

                        "exec": [

                            "pm.test(\"Status code is 200\", function () {\r",

                            "    pm.response.to.have.status(200);\r",

                            "});\r",

                            "pm.test(\" Validate UserName\", function () {\r",

                            "    var jsonData = pm.response.json();\r",

                            "    pm.expect(jsonData.username).to.eql(\"Uname001\");\r",

                            "});\r",

                            "pm.test(\" Validate Email \", function () {\r",

                            "    var jsonData = pm.response.json();\r",

                            "    pm.expect(jsonData.email).to.eql(\"Positive@Attitude.com\");\r",

                            "});\r",

                            "pm.test(\"Your test name\", function () {\r",

                            "    var jsonData = pm.response.json();\r",

                            "    pm.expect(jsonData.userStatus).to.eql(1);\r",

                            "});"

                        ],

                        "type": "text/javascript"

                    }

                }

            ],

            "request": {

                "method": "GET",

                "header": [],

                "url": {

                    "raw": "https://petstore.swagger.io/v2/user/{{UserName}}",

                    "protocol": "https",

                    "host": [

                        "petstore",

                        "swagger",

                        "io"

                    ],

                    "path": [

                        "v2",

                        "user",

                        "{{UserName}}"

                    ]

                }

            },

            "response": []

        },

        {

            "name": "Assignment004",

            "event": [

                {

                    "listen": "test",

                    "script": {

                        "exec": [

                            "pm.test(\"Status code is 200\", function () {\r",

                            "    pm.response.to.have.status(200);\r",

                            "});\r",

                            "\r",

                            "pm.test(\"All pets are available\", function () {\r",

                            "    let responseJson = pm.response.json();\r",

                            "    responseJson.forEach((pet) => {\r",

                            "        pm.expect(pet.status).to.equal(\"available\");\r",

                            "    });\r",

                            "});"

                        ],

                        "type": "text/javascript"

                    }

                }

            ],

            "request": {

                "method": "GET",

                "header": [],

                "url": {

                    "raw": "https://petstore.swagger.io/v2/pet/findByStatus?status= Sold",

                    "protocol": "https",

                    "host": [

                        "petstore",

                        "swagger",

                        "io"

                    ],

                    "path": [

                        "v2",

                        "pet",

                        "findByStatus"

                    ],

                    "query": [

                        {

                            "key": "status",

                            "value": " Sold"

                        },

                        {

                            "key": "status",

                            "value": " pending",

                            "disabled": true

                        },

                        {

                            "key": "status",

                            "value": " sold",

                            "disabled": true

                        }

                    ]

                }

            },

            "response": []

        },

        {

            "name": "Assignment005",

            "event": [

                {

                    "listen": "test",

                    "script": {

                        "exec": [

                            "pm.test(\"Validate code\", function () {\r",

                            "    pm.response.to.have.status(200);\r",

                            "});\r",

                            "pm.test(\"Validate message\", function () {\r",

                            "    var jsonData = pm.response.json();\r",

                            "    pm.expect(jsonData.message).to.eql(\"ok\");\r",

                            "});\r",

                            "pm.test(\"Validate message\", function () {\r",

                            "    var jsonData = pm.response.json();\r",

                            "    pm.expect(jsonData.code).to.eql(200);\r",

                            "});"

                        ],

                        "type": "text/javascript"

                    }

                }

            ],

            "request": {

                "method": "GET",

                "header": [],

                "url": {

                    "raw": "https://petstore.swagger.io/v2/user/logout",

                    "protocol": "https",

                    "host": [

                        "petstore",

                        "swagger",

                        "io"

                    ],

                    "path": [

                        "v2",

                        "user",

                        "logout"

                    ]

                }

            },

            "response": []

        }

    ]

}

**#RestAssured:**

**#001**

**package** Assessment\_Project;

**import** java.io.File;

**import** org.apache.logging.log4j.LogManager;

**import** org.apache.logging.log4j.Logger;

**import** org.hamcrest.Matchers;

**import** org.testng.annotations.Test;

**import** io.restassured.RestAssured;

**import** io.restassured.http.ContentType;

**public** **class** RestAssure\_001 {

Logger logger = LogManager.*getLogger*(RestAssure\_001.**class**);

@Test(priority='1')

**public** **void** Assignment001Post() {

logger.info("Course End Project - Assignment001 - Post Request");

File file = **new** File("C:\\Users\\DELL\\Desktop\\Mphasis\\Selenium\\Phase-3-RestAssured\\src\\main\\resource\\demo.json");

**int** id = RestAssured.*given*()

.baseUri("https://petstore.swagger.io/v2/pet")

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

.body(file)

.when().post()

.then()

.statusCode(200)

.log().all()

.body("name", Matchers.*equalTo*("duck")).extract().path("id");

logger.trace("The status code is checked");

System.***out***.println(id);

logger.trace("ID has been captured and validated");

}

@Test(priority='2', dependsOnMethods="Assignment001Post")

**public** **void** assignment001Get() {

logger.info("Course End Project - Assignment001 - Get Request");

**int** id = RestAssured.*given*()

.baseUri("https://petstore.swagger.io/v2/pet/344")

.when().get()

.then().statusCode(200)

.log().all()

.body("status", Matchers.*equalTo*("available")).extract().path("category.id");

System.***out***.println(id);

logger.trace("ID and status has been captured and validated");

}

@Test(priority='3', dependsOnMethods="assignment001Get")

**public** **void** assignment001Delete() {

logger.info("Course End Project - Assignment001 - Delete Request");

RestAssured.*given*()

.baseUri("https://petstore.swagger.io/v2/pet/344")

.when().delete()

.then().statusCode(200)

.log().all()

.body("code", Matchers.*equalTo*(200))

.body("type", Matchers.*equalTo*("unknown"))

.body("message", Matchers.*equalTo*("344"));

}

}

**#002**

**package** Assessment\_Project;

**import** java.io.File;

**import** org.apache.logging.log4j.LogManager;

**import** org.apache.logging.log4j.Logger;

**import** org.hamcrest.Matchers;

**import** org.testng.annotations.Test;

**import** io.restassured.RestAssured;

**import** io.restassured.http.ContentType;

**public** **class** RestAssured\_002 {

Logger logger = LogManager.*getLogger*(RestAssured\_002.**class**);

@Test(priority = '1')

**public** **void** assignment002Post() {

logger.info("Course End project - Assignment002 - POST request");

File file = **new** File("C:\\Users\\DELL\\Desktop\\Mphasis\\Selenium\\Phase-3-RestAssured\\src\\main\\resource\\demo.json");

**int** id = RestAssured.*given*().baseUri("https://petstore.swagger.io/v2/pet").contentType(ContentType.***JSON***)

.body(file).when().post().then().statusCode(200).log().all().body("name", Matchers.*equalTo*("duck"))

.extract().path("id");

logger.trace("The status code is checked");

System.***out***.println(id);

logger.trace("Id has been captured and validated");

}

@Test(priority = '2', dependsOnMethods = "assignment002Post")

**public** **void** assignment002Put() {

File file = **new** File("C:\\Users\\DELL\\Desktop\\Mphasis\\Selenium\\Phase-3-RestAssured\\src\\main\\resource\\dataPut.json");

**int** id = RestAssured.*given*().baseUri("https://petstore.swagger.io/v2/pet/").contentType(ContentType.***JSON***)

.body(file).when().put().then().statusCode(200).log().all()

.body("status", Matchers.*equalTo*("available\_QA")).extract().path("id");

System.***out***.println(id);

}

}

**#03-4**

**package** Assessment\_Project;

**import** org.hamcrest.Matchers;

**import** org.testng.annotations.Test;

**import** io.restassured.RestAssured;

**public** **class** RestAssured\_3\_4 {

@Test(priority='1')

**public** **void** assignment003User()

{

RestAssured.*given*()

.baseUri("https://petstore.swagger.io/v2/user/Uname001")

.when()

.get()

.then()

.statusCode(200)

.log().all()

.body("username", Matchers.*equalTo*("Uname001"))

.body("email", Matchers.*equalTo*("Positive@Attitude.com"))

.body("userStatus", Matchers.*equalTo*(1))

;

}

@Test(priority='2')

**public** **void** assignment004login()

{

RestAssured.*given*()

.baseUri("https://petstore.swagger.io/v2/user/login")

.auth().preemptive().basic("Uname001", "@tt!tude")

.when()

.get()

.then()

.statusCode(200)

.log().all()

.body("message", Matchers.*anything*());

}

}

**#05-6**

package Assessment\_Project;

import org.hamcrest.Matchers;

import org.testng.annotations.Test;

import io.restassured.RestAssured;

public class RestAssure\_5\_6 {

@Test(priority='1')

public void assignment005FindByStatus()

{

RestAssured.given()

.baseUri("https://petstore.swagger.io/v2/pet/findByStatus")

//.queryParam("status", "available")

//.queryParam("status", "pending")

.queryParam("status", "sold")

.when()

.get()

.then()

.statusCode(200)

.log().all()

;

}

@Test(priority='2')

public void assignment006Logout()

{

RestAssured.given()

.baseUri("https://petstore.swagger.io/v2/user/logout")

.when()

.get()

.then()

.statusCode(200)

.log().all()

.body("code", Matchers.equalTo(200))

.body("type", Matchers.equalTo("unknown"))

.body("message", Matchers.equalTo("ok"))

;

}

}

**J-Meter**

**Source-Code**

**(Code From .Jmx File)**

<?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="ELearning\_Demo" 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" guiclass="ArgumentsPanel" testclass="Arguments" enabled="true">

<collectionProp name="Arguments.arguments"/>

</elementProp>

<boolProp name="TestPlan.tearDown\_on\_shutdown">false</boolProp>

</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;Google Chrome&quot;;v=&quot;119&quot;, &quot;Chromium&quot;;v=&quot;119&quot;, &quot;Not?A\_Brand&quot;;v=&quot;24&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">&quot;Windows&quot;</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/119.0.0.0 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-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">www.simplilearn.com</stringProp>

<stringProp name="Argument.metadata">=</stringProp>

</elementProp>

<elementProp name="BASE\_URL\_2" elementType="Argument">

<stringProp name="Argument.name">BASE\_URL\_2</stringProp>

<stringProp name="Argument.value">lms.simplilearn.com</stringProp>

<stringProp name="Argument.metadata">=</stringProp>

</elementProp>

<elementProp name="BASE\_URL\_3" elementType="Argument">

<stringProp name="Argument.name">BASE\_URL\_3</stringProp>

<stringProp name="Argument.value">accounts.simplilearn.com</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>

</DNSCacheManager>

<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="Phase\_3\_Project" 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">1</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>

<GenericController guiclass="LogicControllerGui" testclass="GenericController" testname="Simplilearn-Login-Simple Controller" enabled="true"/>

<hashTree>

<HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Entering Url-Simplilearn" 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="redirect\_url" elementType="HTTPArgument">

<boolProp name="HTTPArgument.always\_encode">true</boolProp>

<stringProp name="Argument.name">redirect\_url</stringProp>

<stringProp name="Argument.value">https://lms.simplilearn.com/</stringProp>

<stringProp name="Argument.metadata">=</stringProp>

<boolProp name="HTTPArgument.use\_equals">true</boolProp>

</elementProp>

</collectionProp>

</elementProp>

<stringProp name="HTTPSampler.domain">${BASE\_URL\_3}</stringProp>

<stringProp name="HTTPSampler.protocol">https</stringProp>

<stringProp name="HTTPSampler.path">user/login</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-site</stringProp>

</elementProp>

</collectionProp>

</HeaderManager>

<hashTree/>

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

<stringProp name="ConstantTimer.delay">780</stringProp>

</ConstantTimer>

<hashTree/>

<ResponseAssertion guiclass="AssertionGui" testclass="ResponseAssertion" testname="Response Assertion" enabled="true">

<collectionProp name="Asserion.test\_strings">

<stringProp name="49586">200</stringProp>

</collectionProp>

<stringProp name="Assertion.custom\_message"></stringProp>

<stringProp name="Assertion.test\_field">Assertion.response\_code</stringProp>

<boolProp name="Assertion.assume\_success">false</boolProp>

<intProp name="Assertion.test\_type">1</intProp>

</ResponseAssertion>

<hashTree/>

</hashTree>

<HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Entering Login Credentials" 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="user\_login" elementType="HTTPArgument">

<boolProp name="HTTPArgument.always\_encode">true</boolProp>

<stringProp name="Argument.name">user\_login</stringProp>

<stringProp name="Argument.value">gaurav121@gmail.com</stringProp>

<stringProp name="Argument.metadata">=</stringProp>

<boolProp name="HTTPArgument.use\_equals">true</boolProp>

</elementProp>

<elementProp name="user\_pwd" elementType="HTTPArgument">

<boolProp name="HTTPArgument.always\_encode">true</boolProp>

<stringProp name="Argument.name">user\_pwd</stringProp>

<stringProp name="Argument.value">1234Lord@</stringProp>

<stringProp name="Argument.metadata">=</stringProp>

<boolProp name="HTTPArgument.use\_equals">true</boolProp>

</elementProp>

<elementProp name="btn\_login" elementType="HTTPArgument">

<boolProp name="HTTPArgument.always\_encode">true</boolProp>

<stringProp name="Argument.name">btn\_login</stringProp>

<stringProp name="Argument.value">Login</stringProp>

<stringProp name="Argument.metadata">=</stringProp>

<boolProp name="HTTPArgument.use\_equals">true</boolProp>

</elementProp>

<elementProp name="calendar\_url" elementType="HTTPArgument">

<boolProp name="HTTPArgument.always\_encode">true</boolProp>

<stringProp name="Argument.name">calendar\_url</stringProp>

<stringProp name="Argument.value"></stringProp>

<stringProp name="Argument.metadata">=</stringProp>

<boolProp name="HTTPArgument.use\_equals">true</boolProp>

</elementProp>

<elementProp name="redirect\_url" elementType="HTTPArgument">

<boolProp name="HTTPArgument.always\_encode">true</boolProp>

<stringProp name="Argument.name">redirect\_url</stringProp>

<stringProp name="Argument.value">https://lms.simplilearn.com/</stringProp>

<stringProp name="Argument.metadata">=</stringProp>

<boolProp name="HTTPArgument.use\_equals">true</boolProp>

</elementProp>

</collectionProp>

</elementProp>

<stringProp name="HTTPSampler.domain">${BASE\_URL\_3}</stringProp>

<stringProp name="HTTPSampler.protocol">https</stringProp>

<stringProp name="HTTPSampler.path">user/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>

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

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

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

</elementProp>

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

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

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

</elementProp>

</collectionProp>

</HeaderManager>

<hashTree/>

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

<stringProp name="ConstantTimer.delay">10243</stringProp>

</ConstantTimer>

<hashTree/>

</hashTree>

<HTMLAssertion guiclass="HTMLAssertionGui" testclass="HTMLAssertion" testname="HTML Assertion" enabled="true">

<longProp name="html\_assertion\_error\_threshold">10000</longProp>

<longProp name="html\_assertion\_warning\_threshold">1000</longProp>

<stringProp name="html\_assertion\_doctype">omit</stringProp>

<boolProp name="html\_assertion\_errorsonly">false</boolProp>

<longProp name="html\_assertion\_format">0</longProp>

<stringProp name="html\_assertion\_filename">C:\apache-jmeter-5.6.2\bin\DemoScript\_SL\HTMLErrors\elearning\_errors.txt</stringProp>

</HTMLAssertion>

<hashTree/>

</hashTree>

<GenericController guiclass="LogicControllerGui" testclass="GenericController" testname="ExploreHomePage-Simple-Controller" enabled="true"/>

<hashTree>

<HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Home-Page" 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.protocol">https</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>

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

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

<stringProp name="Header.value">?1</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="Redirecting to AI-Machine Learning Program" 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.protocol">https</stringProp>

<stringProp name="HTTPSampler.path">iitk-professional-certificate-course-ai-machine-learning</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>

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

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

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

</elementProp>

</collectionProp>

</HeaderManager>

<hashTree/>

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

<stringProp name="ConstantTimer.delay">16014</stringProp>

</ConstantTimer>

<hashTree/>

</hashTree>

<HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Redirecting To HomePage" 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\_2}</stringProp>

<stringProp name="HTTPSampler.protocol">https</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-site</stringProp>

</elementProp>

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

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

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

</elementProp>

</collectionProp>

</HeaderManager>

<hashTree/>

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

<stringProp name="ConstantTimer.delay">7372</stringProp>

</ConstantTimer>

<hashTree/>

</hashTree>

</hashTree>

<GenericController guiclass="LogicControllerGui" testclass="GenericController" testname="LogOut-Simple Controller" enabled="true"/>

<hashTree>

<HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Navigating to Logout Window" 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\_2}</stringProp>

<stringProp name="HTTPSampler.protocol">https</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-site</stringProp>

</elementProp>

</collectionProp>

</HeaderManager>

<hashTree/>

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

<stringProp name="ConstantTimer.delay">5906</stringProp>

</ConstantTimer>

<hashTree/>

</hashTree>

<HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Log-Out" 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\_2}</stringProp>

<stringProp name="HTTPSampler.protocol">https</stringProp>

<stringProp name="HTTPSampler.path">session/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>

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

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

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

</elementProp>

</collectionProp>

</HeaderManager>

<hashTree/>

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

<stringProp name="ConstantTimer.delay">12258</stringProp>

</ConstantTimer>

<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/>

<ResultCollector guiclass="AssertionVisualizer" testclass="ResultCollector" testname="Assertion Results" enabled="true">

<boolProp name="ResultCollector.error\_logging">true</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/>

<DurationAssertion guiclass="DurationAssertionGui" testclass="DurationAssertion" testname="Duration Assertion" enabled="true">

<stringProp name="DurationAssertion.duration">3000</stringProp>

</DurationAssertion>

<hashTree/>

</hashTree>

</hashTree>

</hashTree>

</jmeterTestPlan>