

Capstone Project

Name: Aditya Pawar

USN: Aditya_72233061J

Question 1

You are tasked with writing unit tests for a small e-commerce application. The application has two classes: Product and ShoppingCart.

- The Product class represents an individual item that can be added to the cart
- The ShoppingCart class represents a shopping cart containing multiple products.

1. Java Files

- Product Class

```
package com.aditya;

public class Product {
    private String name;
    private double price;
    private int quantity;

    public Product(int quantity, String name, double price) {
        if (price < 0 || quantity < 0) {
            throw new IllegalArgumentException("Price and Quantity must be non-negative");
        }
        if (name == null || name.trim().isEmpty()) {
            throw new IllegalArgumentException("Product name must not be null");
        }
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

    public String getName() {
        return name;
    }

    public double getPrice() {
        return price;
    }

    public int getQuantity() {
        return quantity;
    }

    public void addQuantity(int quantity) {
        this.quantity += quantity;
    }

    public void removeQuantity(int quantity) {
        this.quantity -= quantity;
    }

    public void clearCart() {
        quantity = 0;
    }

    public boolean isEmpty() {
        return quantity == 0;
    }

    public void calculateTotalPrice() {
        System.out.println("Total Price: " + (price * quantity));
    }
}
```

```
I or empty");
    }
    this.name = name;
    this.quantity = quantity;
    this.price = price;
}

public double getTotalPrice() {
    return price * quantity;
}

public String getName() {
    return name;
}

public double getPrice() {
    return price;
}

public int getQuantity() {
    return quantity;
}
}
```

- ShoppingCart Class

```
package com.aditya;

import java.util.List;
import java.util.ArrayList;

public class ShoppingCart {

    private List<Product> products;

    public ShoppingCart() {
        this.products = new ArrayList<>();
```

```
}

public void addProduct(Product product) {
    this.products.add(product);
}

public double getTotalPrice() {
    double total = 0.0;
    for (Product product : products) {
        total += product.getTotalPrice();
    }
    return total;
}

public int getProductCount() {
    return products.size();
}

public List<Product> getProducts() {
    return products;
}
}
```

2. Test File —>

```
package com.aditya;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

public class TestFile {

    private Product product1;
    private Product product2;
```

```
private ShoppingCart cart;

@BeforeEach
void setup(){
    product1 = new Product(3, "Laptop", 60000.0);
    product2 = new Product(2, "Smartphone", 20000.0);
    cart = new ShoppingCart();
}

// Product Class Test
@Test
void testProductTotalPrice(){
    assert product1.getTotalPrice() == 180000.0;
    assert product2.getTotalPrice() == 40000.0;
}

// ShoppingCart Class Tests
// 1. Add Product Test
@Test
void testAddProduct(){
    cart.addProduct(product1);
    assertTrue(cart.getProducts().contains(product1));
    assertEquals(1, cart.getProductCount());
}

// 2. Total Price Test
@Test
void testCartTotalPrice(){
    cart.addProduct(product1);
    cart.addProduct(product2);
    assertEquals(220000.0, cart.getTotalPrice());
}

// 3. Product Count Test
@Test
void testProductCount(){
    cart.addProduct(product1);
    cart.addProduct(product2);
```

```
        assertEquals(2, cart.getProductCount());
    }

// Edge Case Tests
// 1. empty Cart
@Test
void testEmptyCart() {
    assertEquals(0, cart.getProductCount());
    assertEquals(0.0, cart.getTotalPrice());
}

// 2. Zero price and quantity product
@Test
void testZeroPriceAndQuantityProduct() {
    Product freeProduct = new Product(0, "Free Item", 0.0);
    cart.addProduct(freeProduct);
    assertEquals(1, cart.getProductCount());
    assertEquals(0.0, cart.getTotalPrice());
}

// Tests for validity of product information
// 1. Negative price → Invalid Input
@Test
void testNegativePriceException(){
    assertThrows(IllegalArgumentException.class, () → {
        new Product(1, "Invalid Product", -100.0);
    });
}

// 2. Negative quantity → Invalid Input
@Test
void testNegativeQuantityException(){
    assertThrows(IllegalArgumentException.class, () → {
        new Product(-2, "Invalid Product", 100.0);
    });
}

// 3. Null or empty product name
```

```

@Test
void testNullOrEmptyProductNameException(){
    assertThrows(IllegalArgumentException.class, () -> {
        new Product(1, null, 100.0);
    });
    assertThrows(IllegalArgumentException.class, () -> {
        new Product(1, "", 100.0);
    });
}

```

3. Outputs

- Product Class Test

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "CapstoneProject". It contains a "src" directory with "main" and "test" packages. "main" contains "Product.java" and "ShoppingCart.java". "test" contains "TestFile.java".
- Code Editor:** The "TestFile.java" file is open, showing Java code for testing the "Product" and "ShoppingCart" classes.
- Run Tool Window:** The "Run" tool window is open, showing the results of a test run. It lists a single test: "TestFile (com.aditya) 48 ms" with "1 test passed" and "1 test total, 48 ms". The output shows the command: "/usr/lib/jvm/default-java/bin/java ...".
- Status Bar:** The status bar at the bottom right shows "9:1 LF UTF-8 4 spaces".

- ShoppingCart Class Tests
 - Add Product Test

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** CapstoneProject [CapstoneProj] contains src, test, and target directories. src contains main, java (with com.aditya, Product, ShoppingCart), and test (with java, com.aditya, TestFile). test contains java and com.aditya.
- Code Editor:** The TestFile.java tab is active, showing Java code for testing a ShoppingCart class. It includes methods for testing product addition, total price, and product count.
- Run Tab:** Shows a successful test run for TestFile (com.aditya) with one test passed: testAddProduct() in 38ms.
- Console:** Displays the command used to run the test and the message "Process finished with exit code 0".

o Total Price Test

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** CapstoneProject [CapstoneProj] contains src, test, and target directories. src contains main, java (with com.aditya, Product, ShoppingCart), and test (with java, com.aditya, TestFile). test contains java and com.aditya.
- Code Editor:** The TestFile.java tab is active, showing Java code for testing a ShoppingCart class. It includes methods for testing product addition, total price, product count, and edge cases.
- Run Tab:** Shows a successful test run for TestFile (com.aditya) with one test passed: testCartTotalPrice() in 57ms.
- Console:** Displays the command used to run the test and the message "Process finished with exit code 0".

o Product Count Test

The screenshot shows a Java development environment with the following details:

- Project Structure:** CapstoneProject [CapstoneProj] > src > main > com.aditya > TestFile
- Code Editor:** The active file is `TestFile.java`. The code contains several test methods for `Product` and `ShoppingCart`.
- Run Output:** The terminal shows the results of running the test `testProductCount()`. It indicates 1 test passed in 53 ms.
- Bottom Status Bar:** Shows the current time as 44:1, file type as LF, encoding as UTF-8, and a 4 spaces indentation setting.

```

public class Testfile {
    void testCartTotalPrice(){
        assertEquals(expected: 220000.0, cart.getTotalPrice());
    }

    // 3. Product Count Test
    @Test
    void testProductCount(){
        cart.addProduct(product1);
        cart.addProduct(product2);
        assertEquals(expected: 2, cart.getProductCount());
    }

    // Edge Case Tests
    // 1. empty Cart
    @Test
    void testEmptyCart() {
        assertEquals(expected: 0, cart.getProductCount());
        assertEquals(expected: 0.0, cart.getTotalPrice());
    }
}

```

- Edge Case Tests

- empty cart

The screenshot shows a Java development environment with the following details:

- Project Structure:** CapstoneProject [CapstoneProj] > src > main > com.aditya > TestFile
- Code Editor:** The active file is `TestFile.java`. The code includes additional test methods for edge cases like an empty cart and zero price/quantity products.
- Run Output:** The terminal shows the results of running the test `testEmptyCart()`. It indicates 1 test passed in 36 ms.
- Bottom Status Bar:** Shows the current time as 55:10, file type as LF, encoding as UTF-8, and a 4 spaces indentation setting.

```

public class Testfile {
    void testProductCount(){
    }

    // Edge Case Tests
    // 1. empty Cart
    else{
        void testEmptyCart() {
            assertEquals(expected: 0, cart.getProductCount());
            assertEquals(expected: 0.0, cart.getTotalPrice());
        }
    }

    // 2. Zero price and quantity product
    else{
        void testZeroPriceAndQuantityProduct() {
            Product freeProduct = new Product(quantity: 0, name: "Free Item", price: 0.0);
            cart.addProduct(freeProduct);
            assertEquals(expected: 1, cart.getProductCount());
            assertEquals(expected: 0.0, cart.getTotalPrice());
        }
    }
}

```

- zero price and zero quantity product

The screenshot shows an IDE interface with the following details:

- Project:** CapstoneProject [CapstoneProject]
- File:** TestFile.java
- Code:**

```

public class TestFile {
    void testEmptyCart() {
        assertEquals(expected: 0, cart.getProductCount());
        assertEquals(expected: 0.0, cart.getTotalPrice());
    }

    // 2. Zero price and quantity product
    @Test
    void testZeroPriceAndQuantityProduct() {
        Product freeProduct = new Product(quantity: 0, name: "Free Item", price: 0.0);
        cart.addProduct(freeProduct);
        assertEquals(expected: 1, cart.getProductCount());
        assertEquals(expected: 0.0, cart.getTotalPrice());
    }

    // Tests for validity of product information
    // 1. Negative price --> Invalid Input
    @Test
    void testNegativePriceException(){
        assertThrows(IllegalArgumentException.class, () -> {
            new Product(quantity: 1, name: "Invalid Product", price: -100.0);
        });
    }
}

```
- Run:** TestFile.testZeroPriceAndQuantityProduct
- Output:**
 - 1 test passed 1 test total, 53 ms
 - /usr/lib/jvm/default-java/bin/java ...
 - Process finished with exit code 0

- Test for validity of product information

- Invalid input → Negative price

The screenshot shows an IDE interface with the following details:

- Project:** CapstoneProject [CapstoneProject]
- File:** TestFile.java
- Code:**

```

public class TestFile {
    void testZeroPriceAndQuantityProduct() {
    }

    // Tests for validity of product information
    // 1. Negative price --> Invalid Input
    @Test
    void testNegativePriceException(){
        assertThrows(IllegalArgumentException.class, () -> {
            new Product(quantity: 1, name: "Invalid Product", price: -100.0);
        });
    }

    // 2. Negative quantity --> Invalid Input
    @Test
    void testNegativeQuantityException(){
        assertThrows(IllegalArgumentException.class, () -> {
            new Product(quantity: -2, name: "Invalid Product", price: 100.0);
        });
    }
}

```
- Run:** TestFile.testNegativePriceException
- Output:**
 - 1 test passed 1 test total, 39 ms
 - /usr/lib/jvm/default-java/bin/java ...
 - Process finished with exit code 0

- Invalid input → Negative Quantity

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows the project structure under "CapstonProject [CapstoneProject]".
- Code Editor:** Displays `TestFile.java` with three test methods: `testNegativePriceException`, `testNegativeQuantityException`, and `testNullOrEmptyProductNameException`.
- Run Tab:** Shows the results of the run. It indicates 1 test passed (1 test total) in 47 ms. The command used is `/usr/Lib/jvm/default-java/bin/java ...`. The output shows "Process finished with exit code 0".
- Bottom Status Bar:** Shows file statistics: 77.6 LF, UTF-8, 4 spaces.

o Null Value

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows the project structure under "CapstonProject [CapstoneProject]".
- Code Editor:** Displays `TestFile.java` with three test methods: `testNegativePriceException`, `testNegativeQuantityException`, and `testNullOrEmptyProductNameException`.
- Run Tab:** Shows the results of the run. It indicates 1 test passed (1 test total) in 47 ms. The command used is `/usr/Lib/jvm/default-java/bin/java ...`. The output shows "Process finished with exit code 0".
- Bottom Status Bar:** Shows file statistics: 77.6 LF, UTF-8, 4 spaces.

```

1 package com.aditya;
2
3 import org.junit.jupiter.api.BeforeEach;
4 import org.junit.jupiter.api.Test;
5
6 import static org.junit.jupiter.api.Assertions.*;
7
8 public class TestFile {
9
10 }

```

Run TestFile (com.aditya) 87 ms ✓ 9 tests passed 9 tests total, 87 ms
 ✓ testNullOrEmptyProductName() 61 ms ✓ testCartTotalPrice() 4 ms
 ✓ testNegativeQuantityException() 6 ms ✓ testZeroPriceAndQuantityPrice() 2 ms
 ✓ testEmptyCart() 3 ms ✓ testProductTotalPrice() 1 ms
 ✓ testNegativePriceException() 1 ms ✓ testAddProduct() 6 ms
 ✓ testProductCount() 3 ms

Question 2

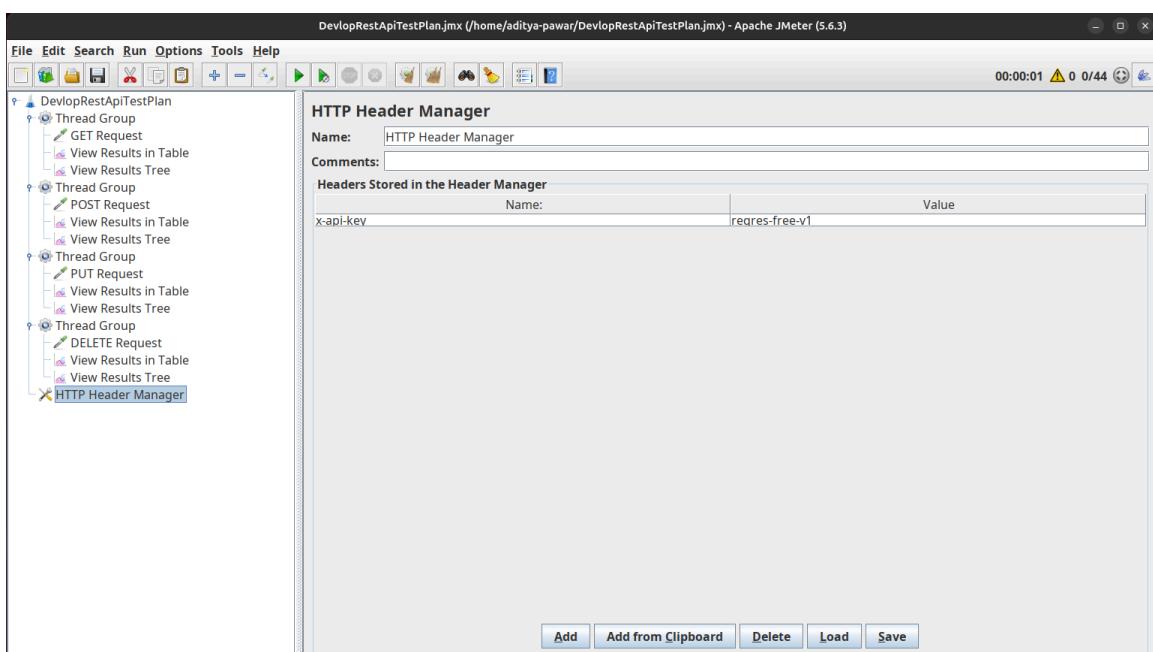
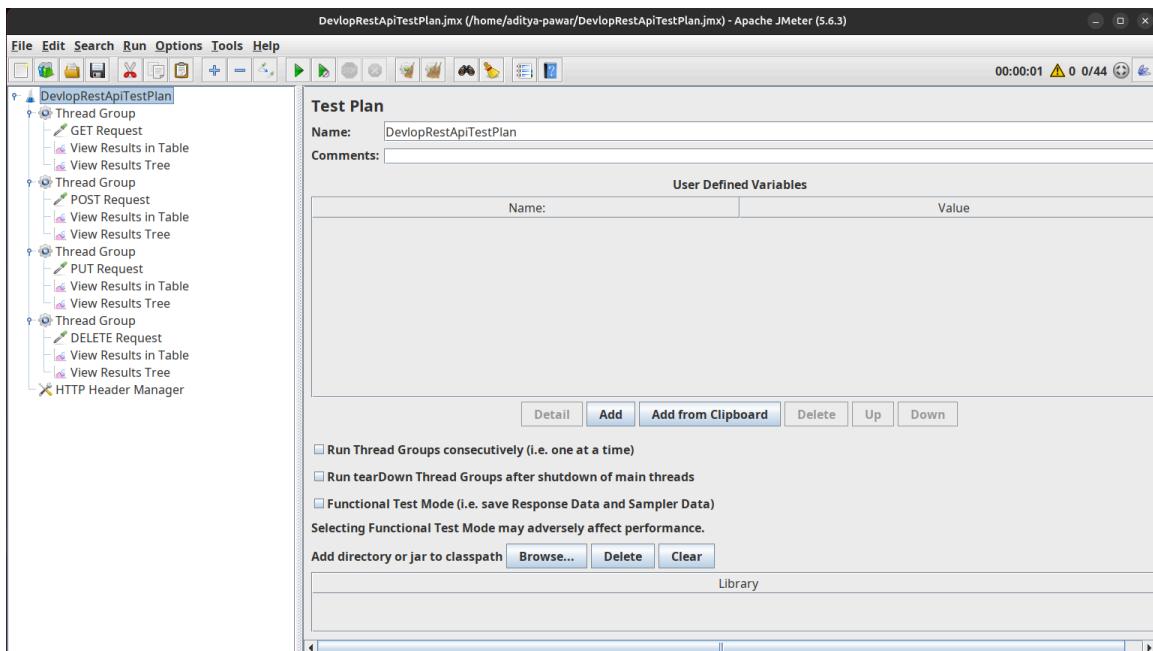
Create a Test Plan in JMeter for simulating traffic to the rest api use appropriate

- listeners and samplers
- Name: DevlopRestApiTestPlan
- URL: <https://regres.in>
- Users: 20 for following requests methods
 - Get list of all objects
 - Post
 - Update
 - Delete

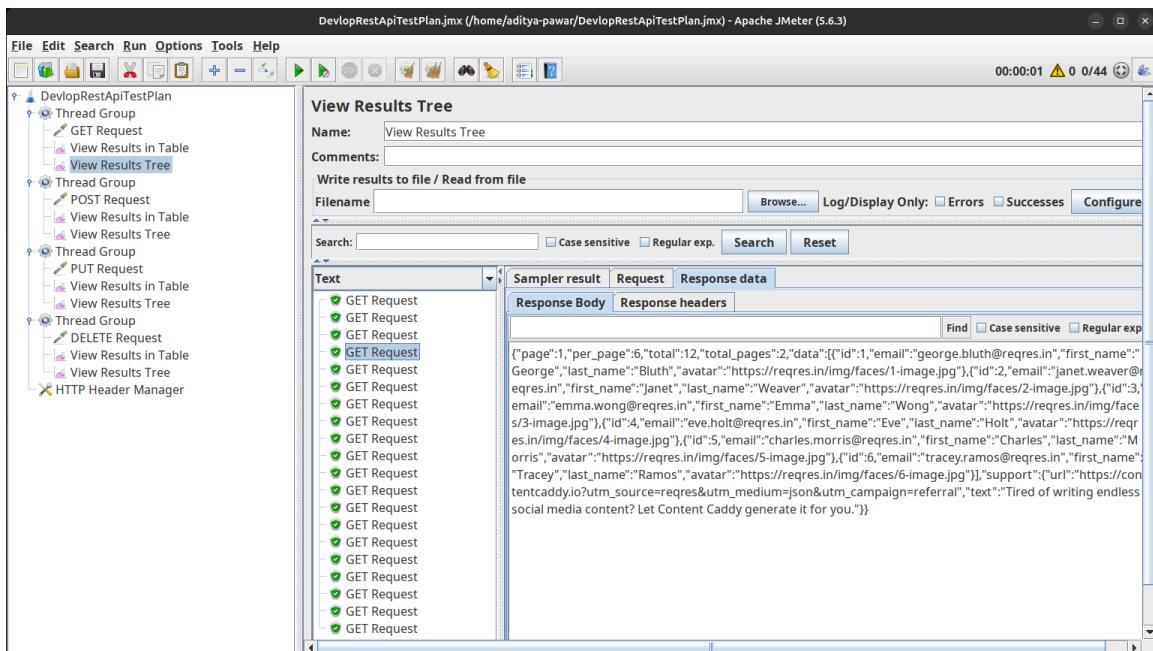
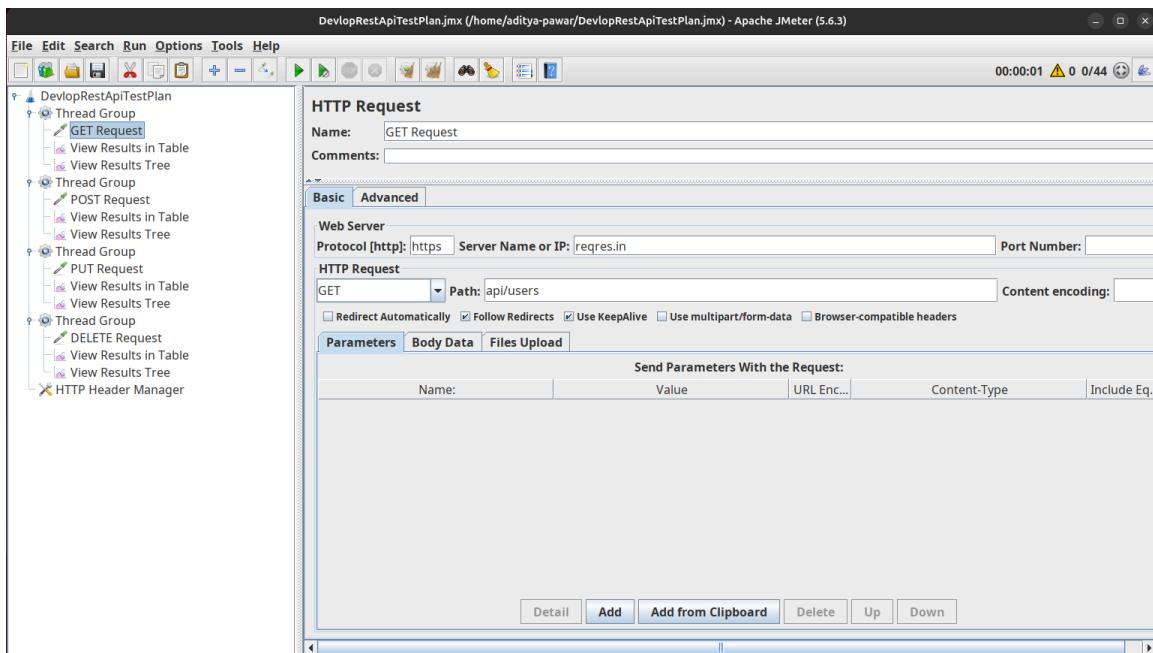
1. Check for valid jmx for above plan

Solution :

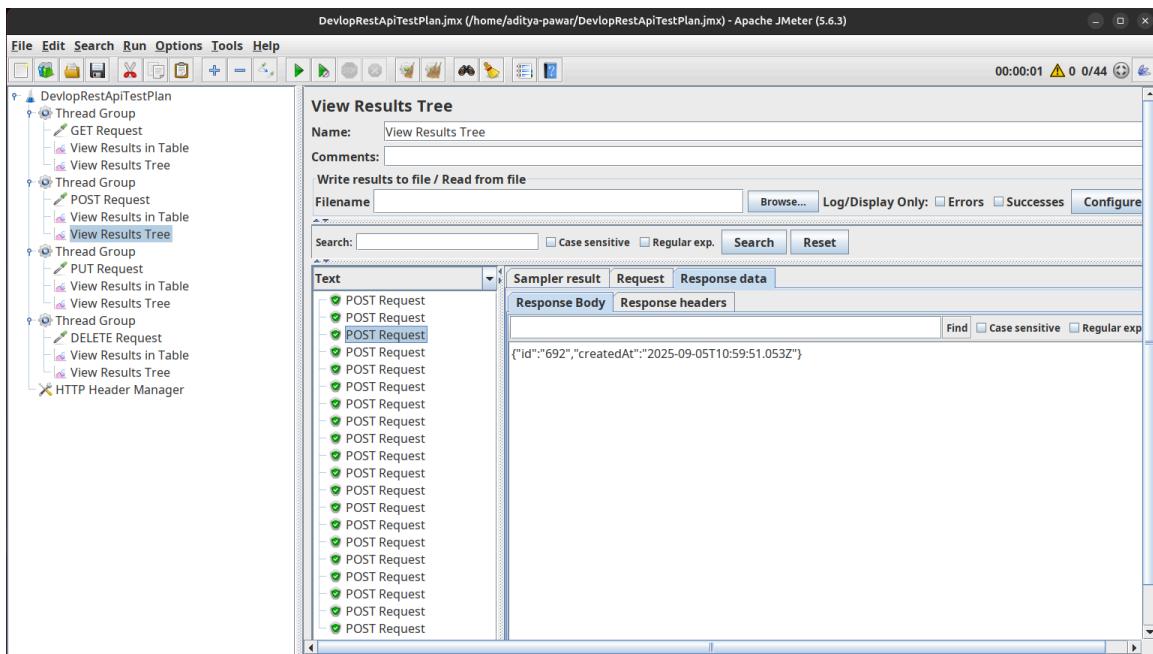
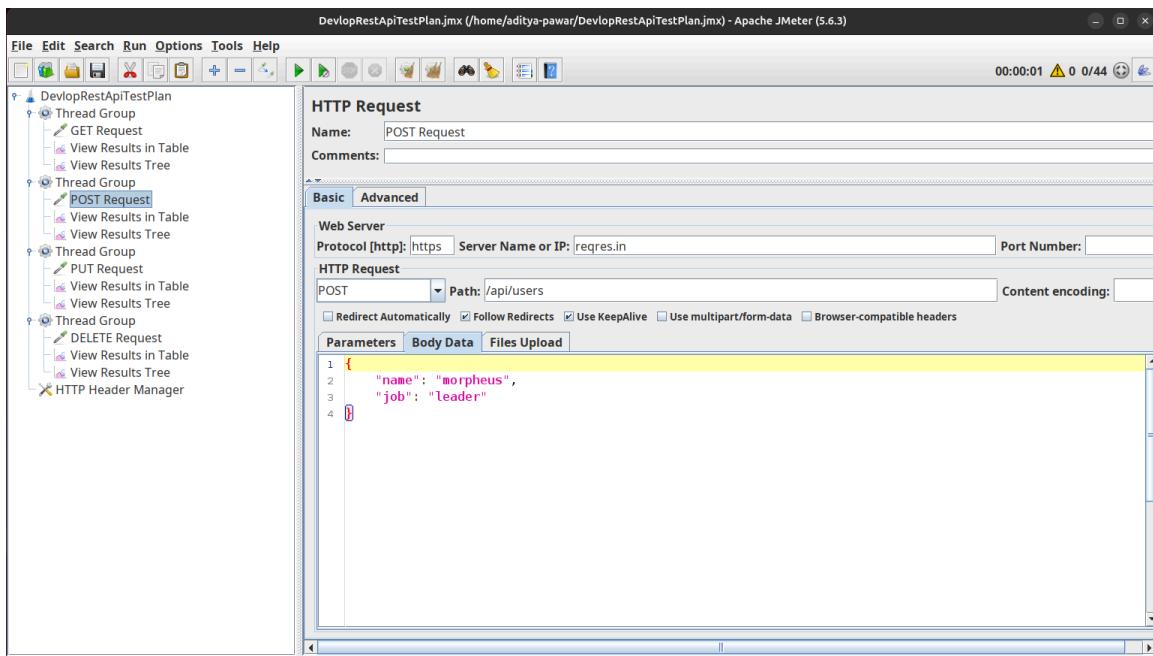
- jmeter Structure



- Get Request



- Post Request



- Put Request

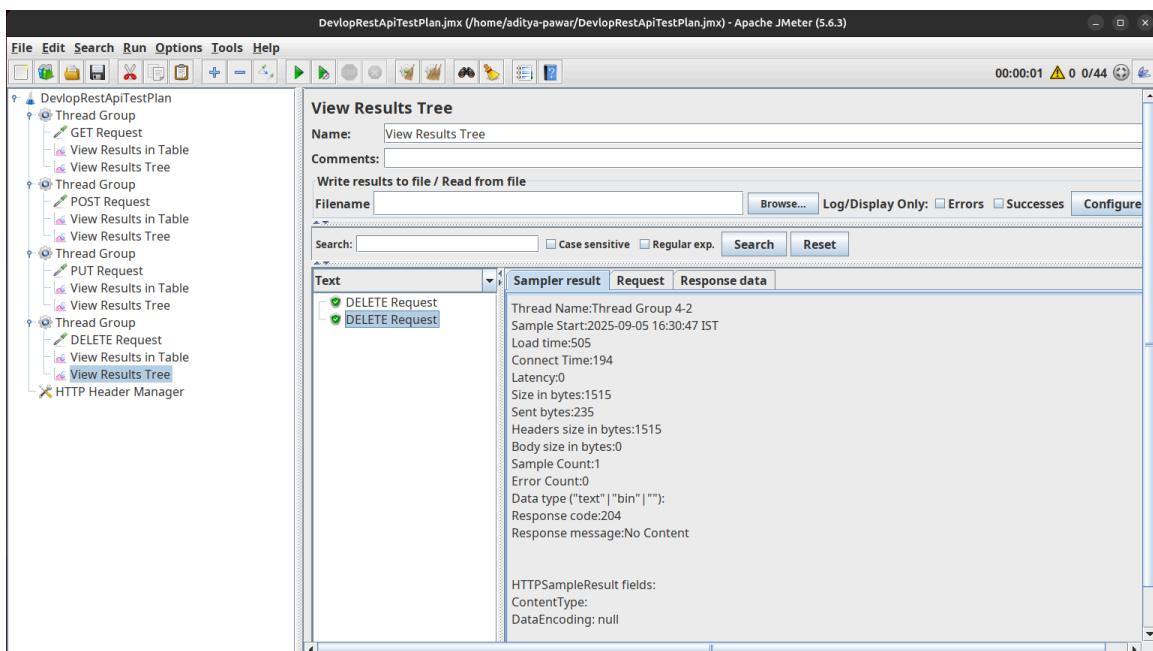
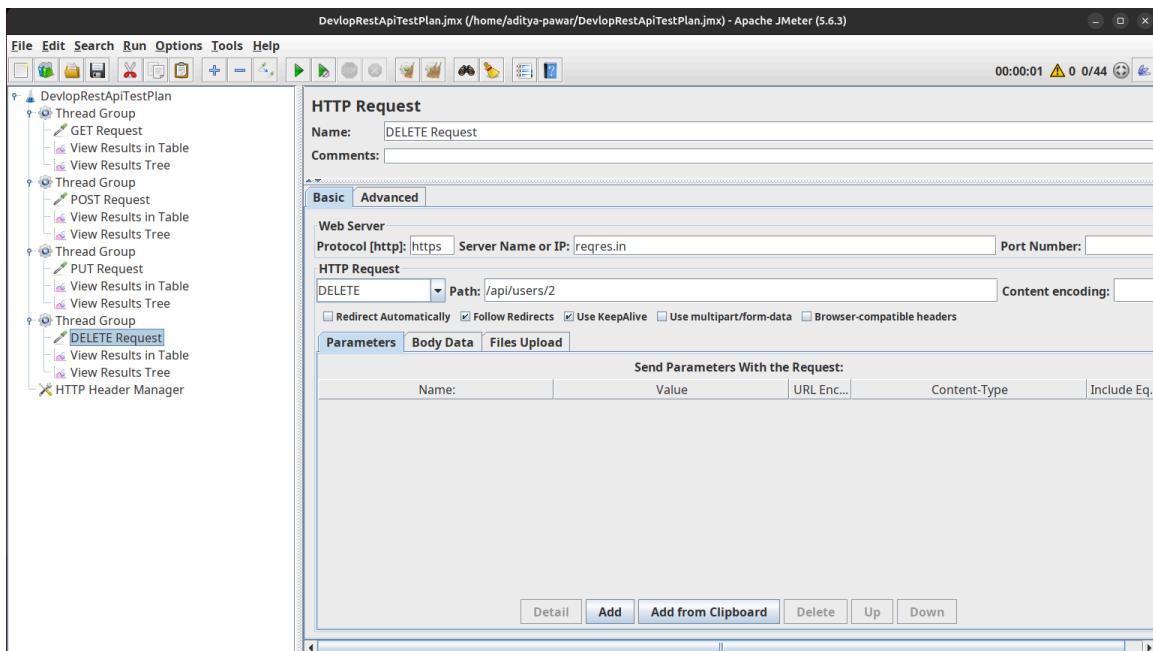
Screenshot of Apache JMeter 5.6.3 showing a test plan named "DevelopRestApiTestPlan.jmx". The left pane displays the test plan structure with several Thread Groups and various request types (GET, POST, PUT, DELETE). The right pane shows a detailed view of a "PUT Request" sampler. The "Basic" tab is selected, displaying the "Web Server" configuration with Protocol set to https, Server Name or IP to reqres.in, and Port Number left blank. The "HTTP Request" section shows the method as POST and the path as /api/users/2. Under the "Parameters" tab, there is a JSON payload:

```
1 {
2     "name": "morpheus",
3     "job": "zion resident"
4 }
```

Screenshot of Apache JMeter 5.6.3 showing the same test plan. The left pane remains the same. The right pane shows a "View Results Tree" listener. The "Text" panel lists two entries under "PUT Request": "PUT Request" and "PUT Request". The "Response Body" panel displays the JSON response from the previous PUT request:

```
{"id":418,"createdAt":"2025-09-05T11:00:35.405Z"}
```

- Delete Request

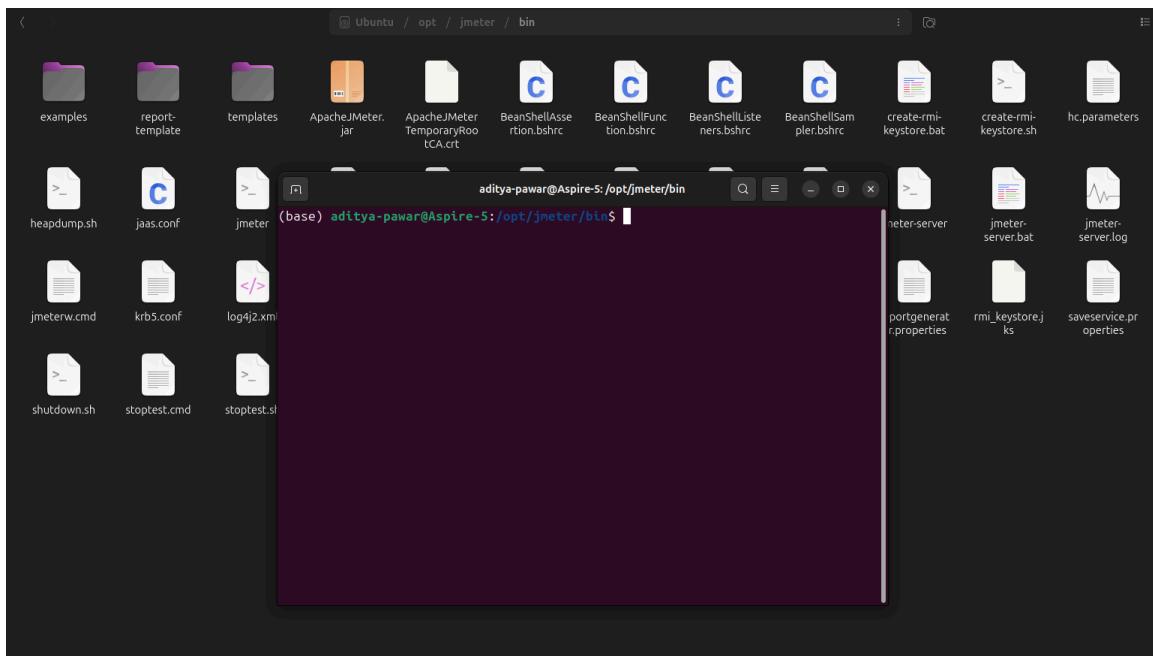


2. Generate HTML Report for the above test plan using non Gui mode of JMeter

STEPS:

1. Open Jmeter in Non-GUI Mode →

- /opt/jmeter/bin — path in terminal



2. use command

```
jmeter -n -t "/home/aditya-pawar/Documents/Jmeter/JDBC Connection Configuration.jmx" \
-I "/home/aditya-pawar/Documents/Jmeter/Html report/jdbc_test_html_report.csv"
```

- to generate the .csv log file of the .jmx jmeter file

```

(base) aditya-pawar@Aspire-5:/opt/jmeter/bin$ jmeter -n -t "/home/aditya-pawar/Documents/Jmeter/DevlopRestApiTestPlan.jmx" -l "/home/aditya-pawar/Documents/JmeterReports/logFile/DevlopRestApitest.csv"
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
Creating summariser <summary>
Created the tree successfully using /home/aditya-pawar/Documents/Jmeter/DevlopRestApiTestPlan.jmx
Starting standalone test @ 2025 Sep 5 16:43:49 IST (1757070829642)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary = 44 in 00:00:02 = 23.0/s Avg: 1032 Min: 467 Max: 1668 Err: 0 (0.00%)
Tidying up ... @ 2025 Sep 5 16:43:52 IST (1757070832346)
... end of run
(base) aditya-pawar@Aspire-5:/opt/jmeter/bin$ 

```

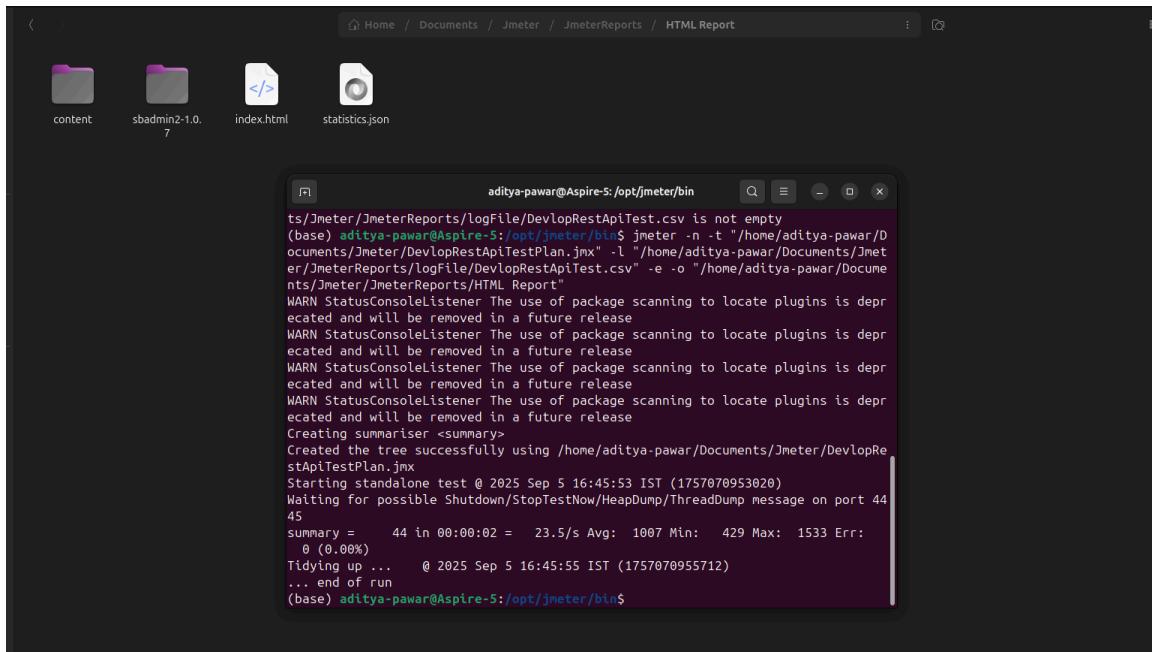
- DevlopRestApiTest csv file

	A1	timeStamp	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	timeStamp		elapsed	label	responseCode	responseMessage	threadName	dataType	success	failureMessage	bytes	sentBytes	grpThreads	allThreads	URL	Latency	idleTime	Connect	
2	1757070954427		804	GET Request	200	OK	Thread Group 1-1	text	TRUE		2874	148	20	44	https://regres.in/	803	0	668	
3	1757070954408		1145	GET Request	200	OK	Thread Group 1-1	text			2873	148	20	44	https://regres.in/	1137	0	1013	
4	1757070954085		1146	GET Request	200	OK	Thread Group 1-1	text	TRUE		2873	148	20	44	https://regres.in/	1137	0	1009	
5	1757070954479		750	GET Request	200	OK	Thread Group 1-1	text	TRUE		2873	148	20	44	https://regres.in/	747	0	619	
6	1757070954320		640	GET Request	200	OK	Thread Group 1-1	text	TRUE		2873	148	20	44	https://regres.in/	641	0	519	
7	1757070954320		965	GET Request	200	OK	Thread Group 1-1	text	TRUE		2873	148	20	44	https://regres.in/	890	0	768	
8	1757070954680		551	GET Request	200	OK	Thread Group 1-1	text	TRUE		2873	148	20	44	https://regres.in/	543	0	416	
9	1757070954098		1146	GET Request	200	OK	Thread Group 1-1	text	TRUE		2875	148	20	44	https://regres.in/	1137	0	1011	
10	1757070954095		1145	GET Request	200	OK	Thread Group 1-1	text	TRUE		2874	148	20	44	https://regres.in/	1141	0	1009	
11	1757070954180		1051	GET Request	200	OK	Thread Group 1-1	text	TRUE		2874	148	20	44	https://regres.in/	1042	0	918	
12	1757070954377		854	GET Request	200	OK	Thread Group 1-1	text	TRUE		2873	148	20	44	https://regres.in/	849	0	719	
13	1757070954628		606	GET Request	200	OK	Thread Group 1-1	text	TRUE		2873	148	20	44	https://regres.in/	606	0	468	
14	1757070954129		1109	GET Request	200	OK	Thread Group 1-1	text	TRUE		2875	148	20	44	https://regres.in/	1108	0	974	
15	1757070954529		709	GET Request	200	OK	Thread Group 1-1	text	TRUE		2873	148	20	44	https://regres.in/	709	0	571	
16	1757070954701		460	GET Request	200	OK	Thread Group 1-1	text	TRUE		2873	148	6	30	https://regres.in/	481	0	340	
17	1757070954276		962	GET Request	200	OK	Thread Group 1-1	text	TRUE		2873	148	5	20	https://regres.in/	881	0	834	
18	1757070954727		532	GET Request	200	OK	Thread Group 1-1	text	TRUE		2873	148	5	29	https://regres.in/	532	0	387	
19	1757070954230		1030	GET Request	200	OK	Thread Group 1-1	text	TRUE		2873	148	5	29	https://regres.in/	1030	0	886	
20	1757070954831		429	GET Request	200	OK	Thread Group 1-1	text	TRUE		2873	148	5	29	https://regres.in/	429	0	284	
21	1757070954098		1175	GET Request	200	OK	Thread Group 1-1	text	TRUE		2874	148	5	29	https://regres.in/	1175	0	1020	
22	1757070954317		1069	POST Request	201	Created	Thread Group 2-1	text	TRUE		1643	260	20	24	https://regres.in/	1069	0	781	
23	1757070954098		1314	POST Request	201	Created	Thread Group 2-1	text	TRUE		1643	260	19	23	https://regres.in/	1314	0	1017	
24	1757070954086		1318	DELETE Request	204	No Content	Thread Group 4-1	text	TRUE		1529	236	2	22	https://regres.in/	0	0	1016	
25	1757070954410		1160	PUT Request	201	Created	Thread Group 2-1	text	TRUE		1643	260	18	22	https://regres.in/	1186	0	881	
26	1757070954411		965	PUT Request	201	Created	Thread Group 2-1	text	TRUE		1643	260	18	20	https://regres.in/	860	0	665	
27	1757070954098		1331	POST Request	201	Created	Thread Group 2-1	text	TRUE		1644	260	17	19	https://regres.in/	1331	0	1011	
28	1757070954097		1321	POST Request	201	Created	Thread Group 2-1	text	TRUE		1643	260	16	18	https://regres.in/	1321	0	1017	
29	1757070954766		652	POST Request	201	Created	Thread Group 2-1	text	TRUE		1643	260	16	18	https://regres.in/	652	0	351	
30	1757070954433		991	DELETE Request	204	No Content	Thread Group 4-2	text	TRUE		1528	236	1	16	https://regres.in/	0	0	668	
31	1757070954626		802	POST Request	201	Created	Thread Group 2-1	text	TRUE		1643	260	14	15	https://regres.in/	802	0	475	
32	1757070954717		722	POST Request	201	Created	Thread Group 2-1	text	TRUE		1643	260	13	14	https://regres.in/	722	0	390	
33	1757070954567		954	POST Request	201	Created	Thread Group 2-1	text	TRUE		1643	260	12	13	https://regres.in/	954	0	441	
34	1757070954267		1299	POST Request	201	Created	Thread Group 2-1	text	TRUE		1645	260	11	12	https://regres.in/	1298	0	441	

3. Generate HTML report

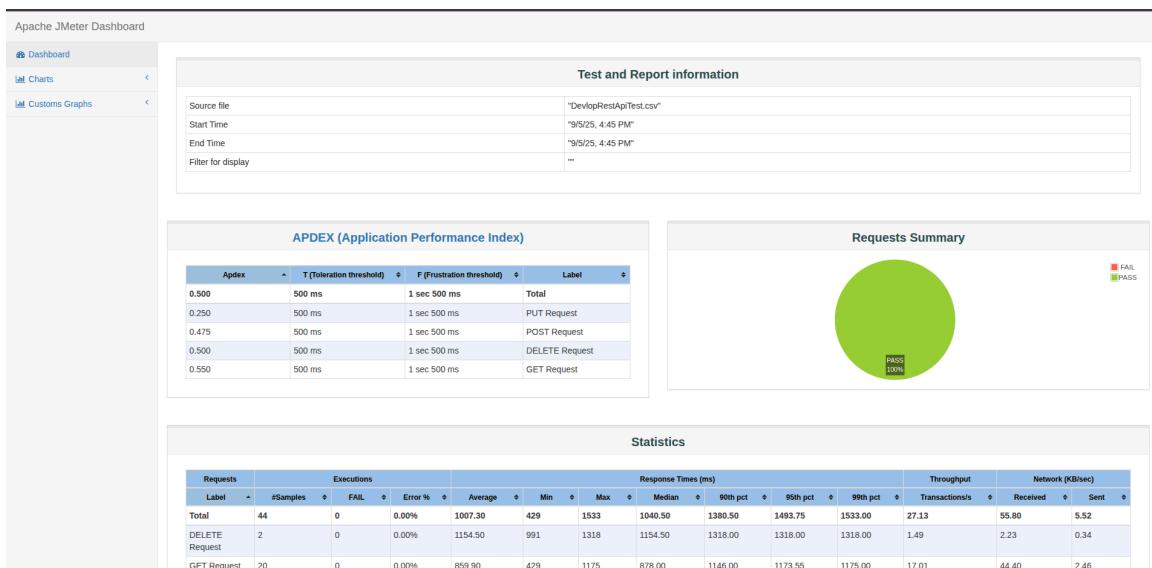
```
jmeter -n -t "/home/aditya-pawar/Documents/Jmeter/DevlopRestApiTe
stPlan.jmx"
-I "/home/aditya-pawar/Documents/Jmeter/JmeterReports/logFile/Devl
opRestApiTest.csv"
```

```
-e -o "/home/aditya-pawar/Documents/Jmeter/JmeterReports/HTML Report"
```



```
ts/Jmeter/JmeterReports/logFile/DevelopRestApiTest.csv is not empty
(base) aditya-pawar@Aspire-5:/opt/jmeter/bin$ jmeter -n -t "/home/aditya-pawar/Documents/Jmeter/DevelopRestApitestPlan.jmx" -l "/home/aditya-pawar/Documents/Jmeter/JmeterReports/logfile/DevelopRestApitest.csv" -e -o "/home/aditya-pawar/Documents/Jmeter/JmeterReports/HTML Report"
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
Creating summariser <summary>
Created the tree successfully using /home/aditya-pawar/Documents/Jmeter/DevelopRestApitestPlan.jmx
Starting standalone test @ 2025 Sep 5 16:45:53 IST (1757070953020)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 44
45
summary = 44 in 00:00:02 = 23.5/s Avg: 1007 Min: 429 Max: 1533 Err: 0 (0.00%)
Tidying up ... @ 2025 Sep 5 16:45:55 IST (1757070955712)
... end of run
(base) aditya-pawar@Aspire-5:/opt/jmeter/bin$
```

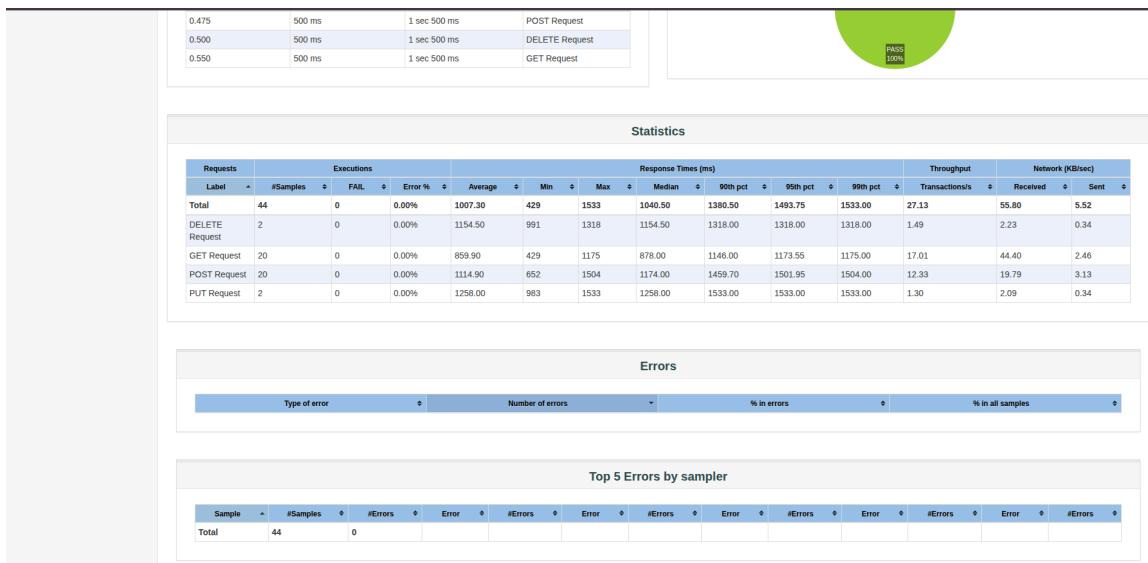
- HTML Report output



The screenshot shows the Apache JMeter Dashboard with several sections:

- Test and Report information:** Displays the source file as "DevelopRestApiTest.csv", start time as "9/5/25, 4:45 PM", and end time as "9/5/25, 4:45 PM".
- APDEX (Application Performance Index):** A table showing APDEX values, toleration thresholds, frustration thresholds, and labels for various requests.
- Requests Summary:** A large green circle indicating successful requests.
- Statistics:** A detailed table of statistics for different requests, including requests, executions, response times, throughput, and network usage.

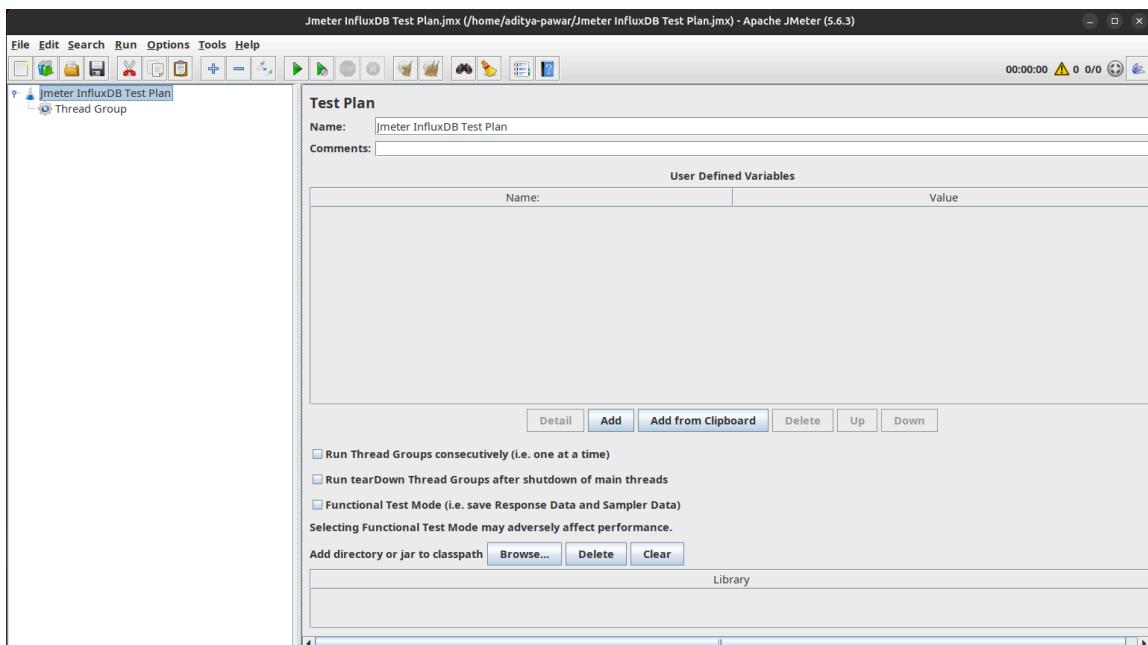
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Throughput	Network (KByte/s)	
Total	44	0	0.00%	1007.30	429	1533	1040.50	1380.50	1493.75	1533.00	27.13	55.80	5.52
DELETE Request	2	0	0.00%	1154.50	991	1318	1154.50	1318.00	1318.00	1318.00	1.49	2.23	0.34
GET Request	20	0	0.00%	859.90	429	1175	878.00	1146.00	1173.55	1175.00	17.01	44.40	2.46



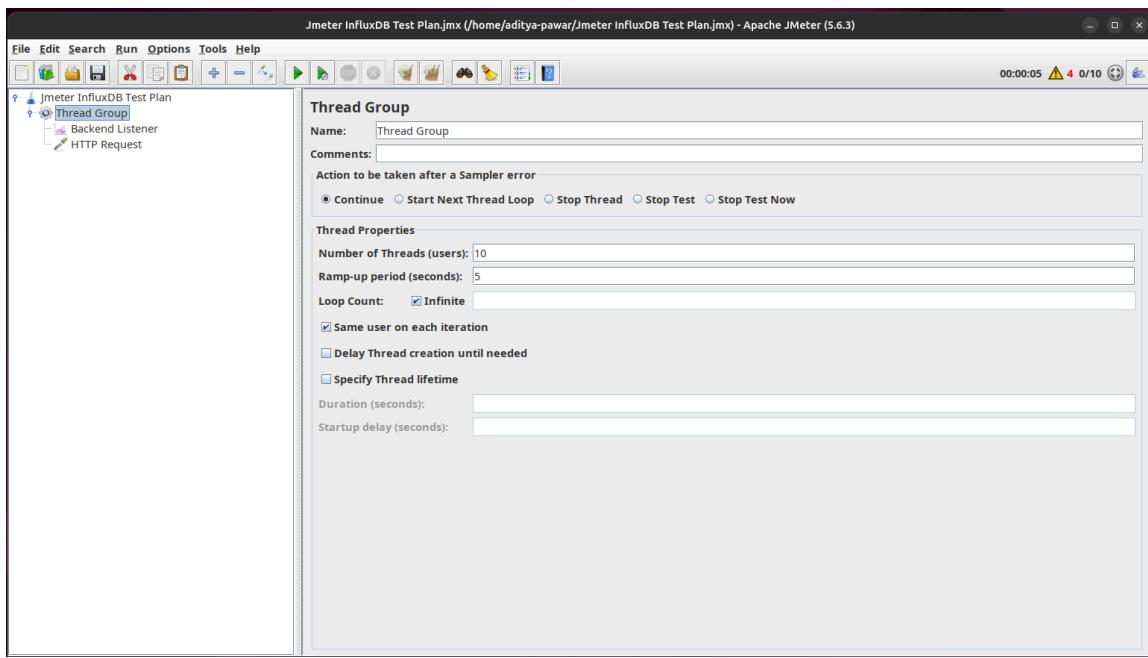
3. Connect JMeter with influx db. and store data in Database DBName : DevInfluxDB

STEPS

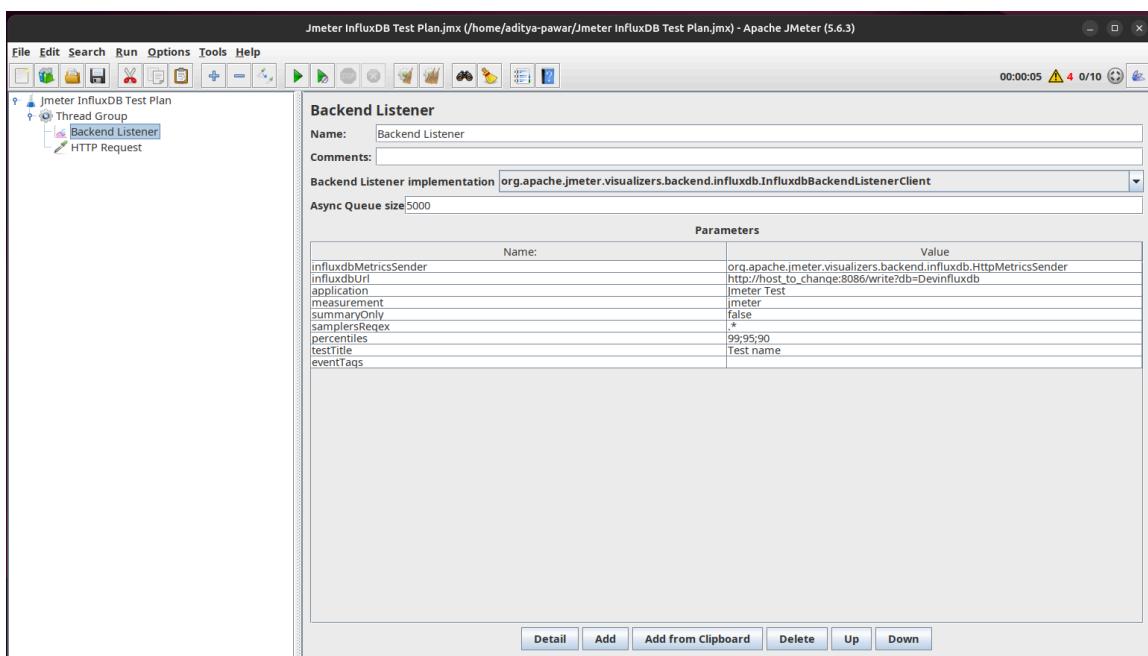
1. create new test Plan



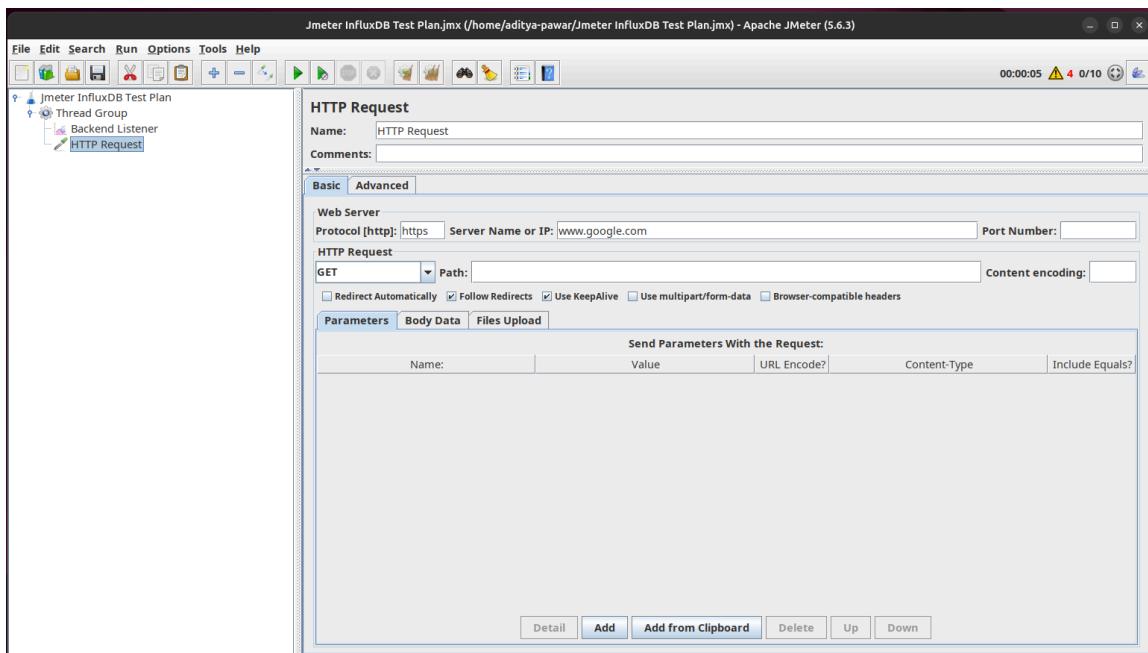
2. add thread group



3. add Backend Listener



4. Add http Request



5. InfluxDB Creating Database DevInfluxdb :

```
aditya-pawar$ influx
Connected to http://localhost:8086 version 1.6.7~rc0
InfluxDB shell version: 1.6.7~rc0
> show databases;
name: databases
name
-----
jmeter
_internal
DevInfluxdb
> █
```

6. running jmeter jmx on terminal to generate responses

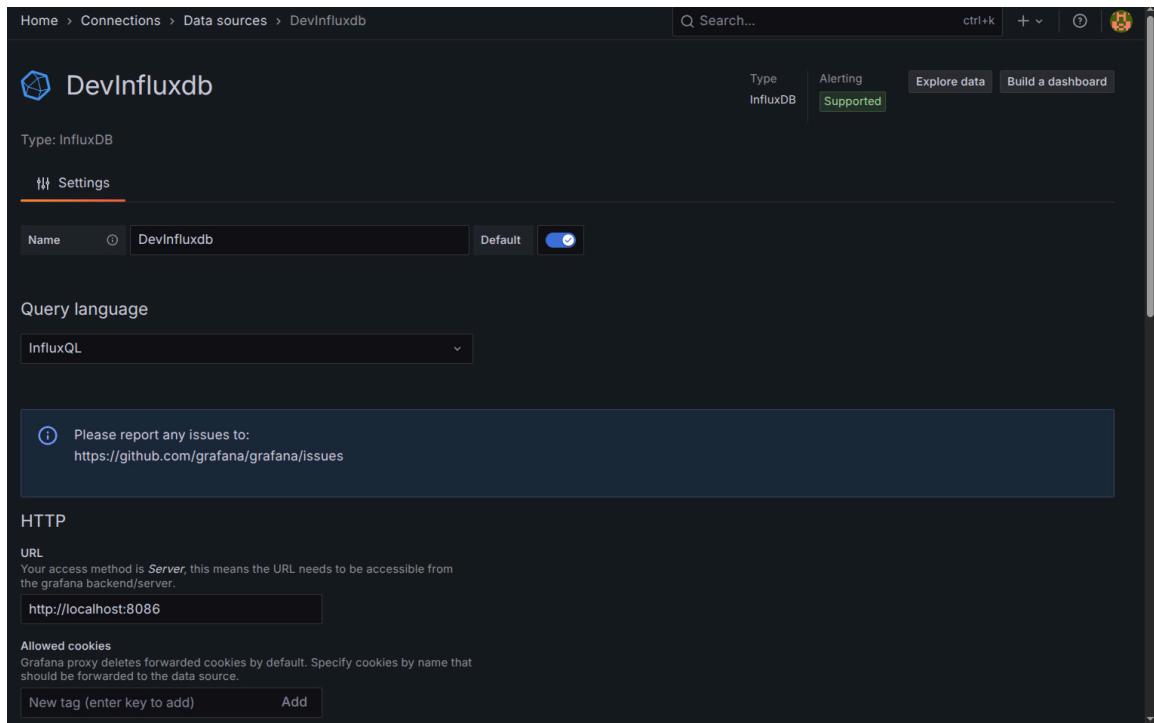
```
aditya-pawar$ jmeter -n -t /home/aditya/Documents/Capstone/Jmeter\ InfluxDBTest\ Plan.jmx
```

7. Jmeter Pushed Data to InfluxDB :

```
Starting standalone test @ 2025 Sep 5 18:19:00 IST (1757076540301)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary +      1 in 00:00:01 =    0.9/s Avg:   972 Min:   972 Max:   972 Err:     0 (0.00%) Active: 3 Started: 3 Finished: 0
summary +  278 in 00:00:28 =   9.8/s Avg:   928 Min:   464 Max:  3309 Err:     0 (0.00%) Active: 10 Started: 10 Finished: 0
Summary =  279 in 00:00:29 =   9.5/s Avg:   928 Min:   464 Max:  3309 Err:     0 (0.00%)
```

4. Connect influxdb with Grafana and create a dashboard for showing the result of test

1. Graphana DataSource :



The screenshot shows the Grafana interface for managing data sources. The top navigation bar includes 'Home', 'Connections', 'Data sources', and 'DevInfluxdb'. A search bar and various control buttons are also present. The main panel displays the 'DevInfluxdb' data source settings. Key details include:

- Type:** InfluxDB
- Name:** DevInfluxdb (set as Default)
- Query language:** InfluxQL
- HTTP:** URL: http://localhost:8086
- Allowed cookies:** A note states: "Grafana proxy deletes forwarded cookies by default. Specify cookies by name that should be forwarded to the data source." A text input field for "New tag (enter key to add)" and a "Add" button are shown.
- A message at the bottom encourages reporting issues to <https://github.com/grafana/grafana/issues>.

2. Importing Dashboard For InfluxDB in Graphana :

Home > Dashboards > Import dashboard

Import dashboard from file or Grafana.com

Options

Name: Apache JMeter Dashboard using Core InfluxdbBackendListenerClient

Folder: Dashboards

Unique identifier (UID)
The unique identifier (UID) of a dashboard can be used for uniquely identify a dashboard between multiple Grafana installs. The UID allows having consistent URLs for accessing dashboards so changing the title of a dashboard will not break any bookmarked links to that dashboard.

DB name: Create a Datasource in Grafana that points to jmeter database

Measurement name: jmeter

Backend send interval: 5

Import | **Cancel**

3. run jmx file

4. this is final output

