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 n
on-negative");
        }
        if (name == null || name.trim().isEmpty()) {
        throw new IllegalArgumentException("Product name must not be nul
```

```
l 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;
```

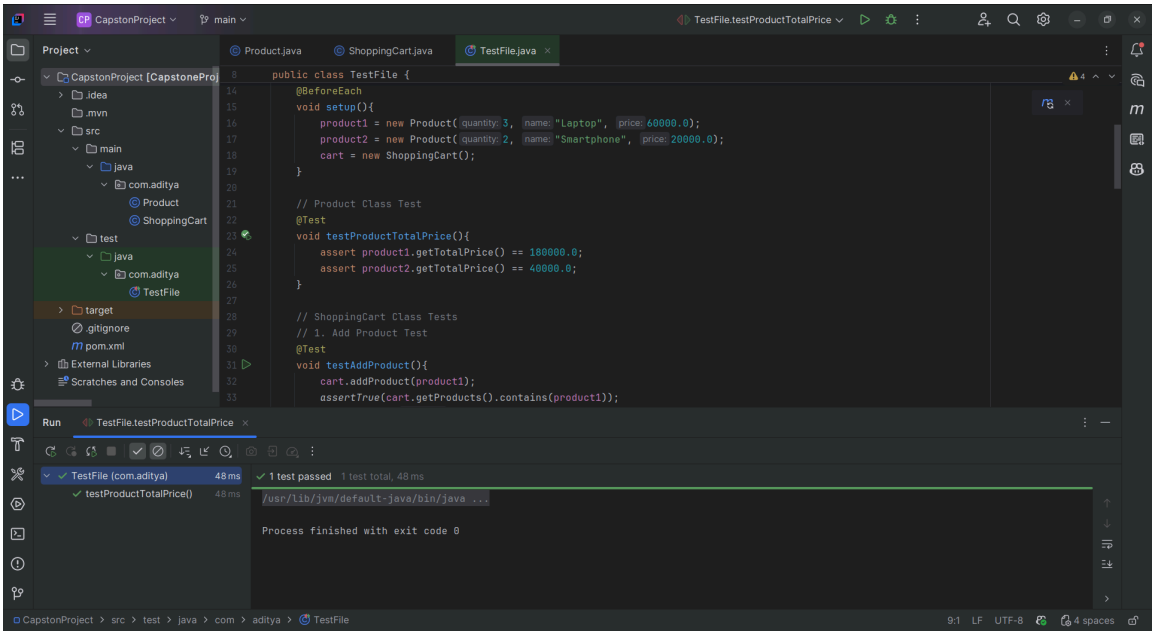
```
private ShoppingCart cart;

@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
```

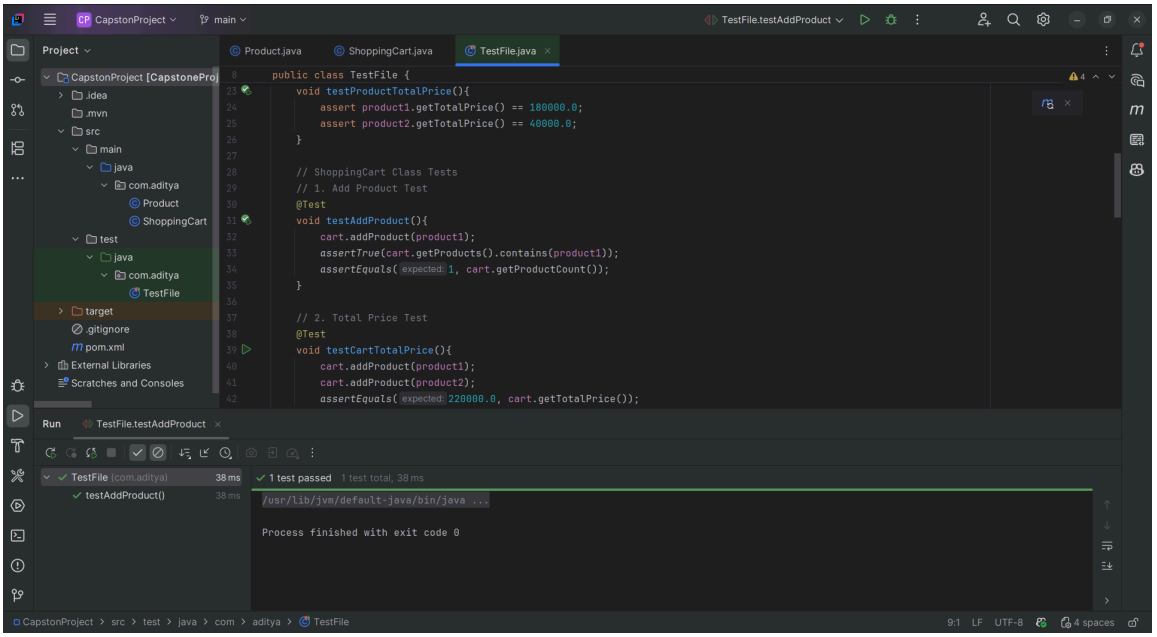
```
    @Test
    void testNullOrEmptyProductNameException(){
        assertThrows(IllegalArgumentException.class, () -> {
            new Product(1, null, 100.0);
        });
        assertThrows(IllegalArgumentException.class, () -> {
            new Product(1, "", 100.0);
        });
    }
}
```

3. Outputs

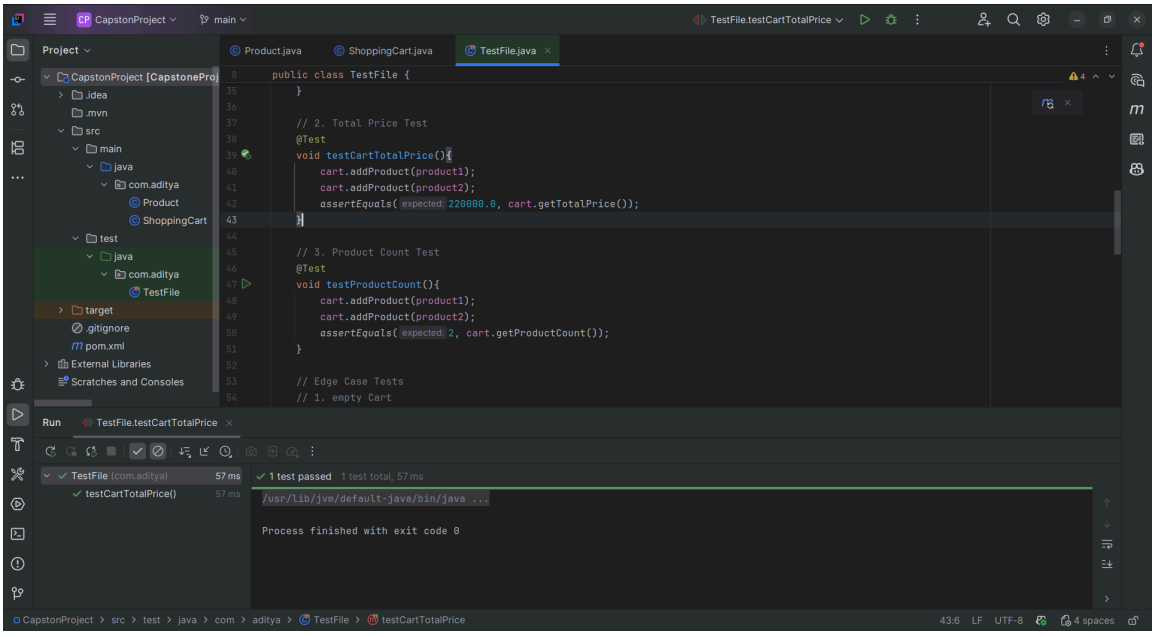
- Product Class Test



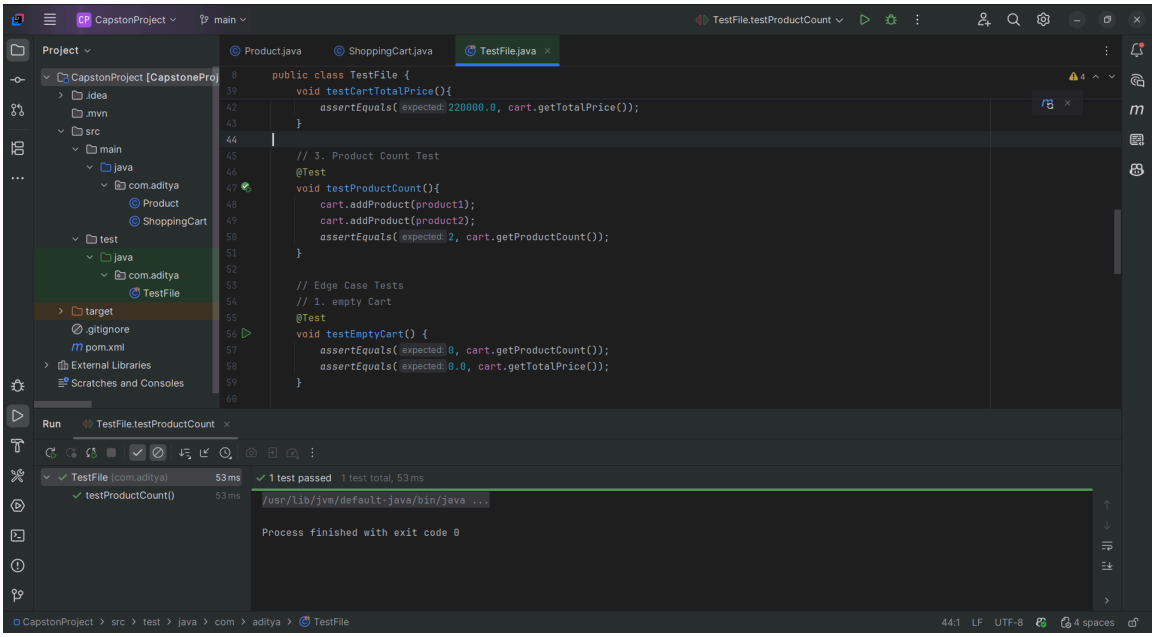
- ShoppingCart Class Tests
    - Add Product Test



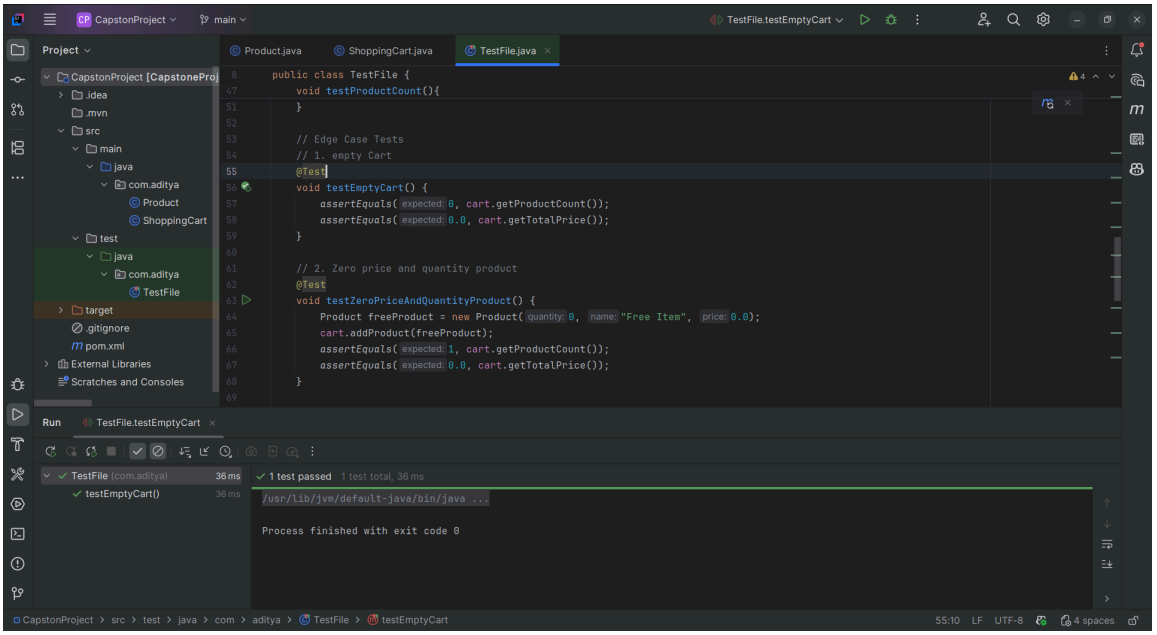
- Total Price Test



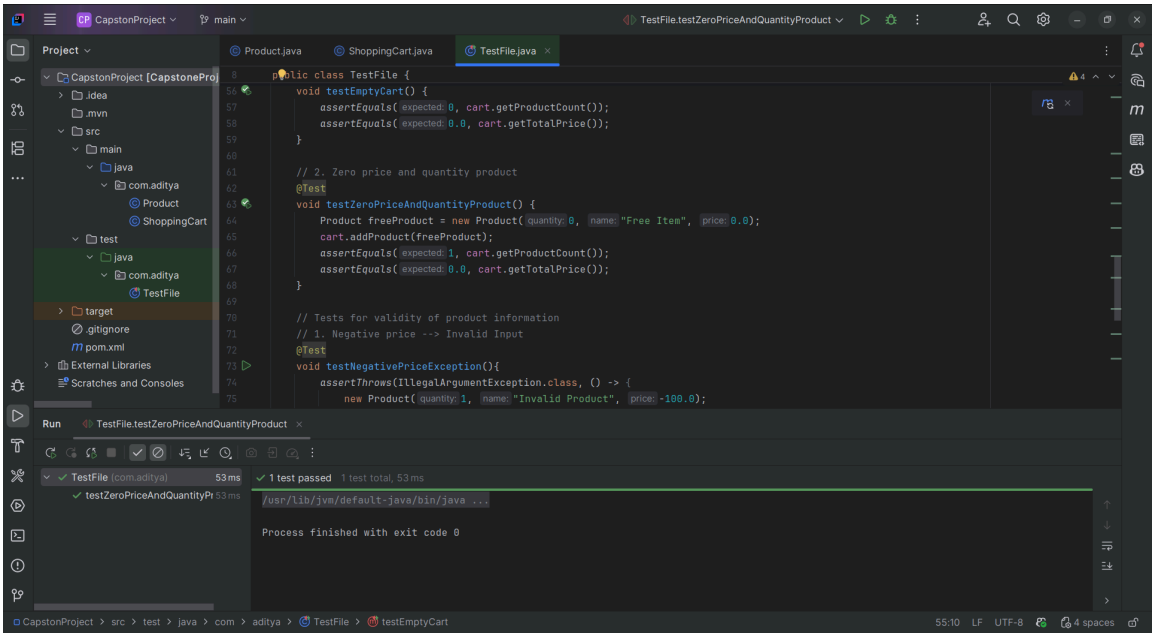
- Product Count Test



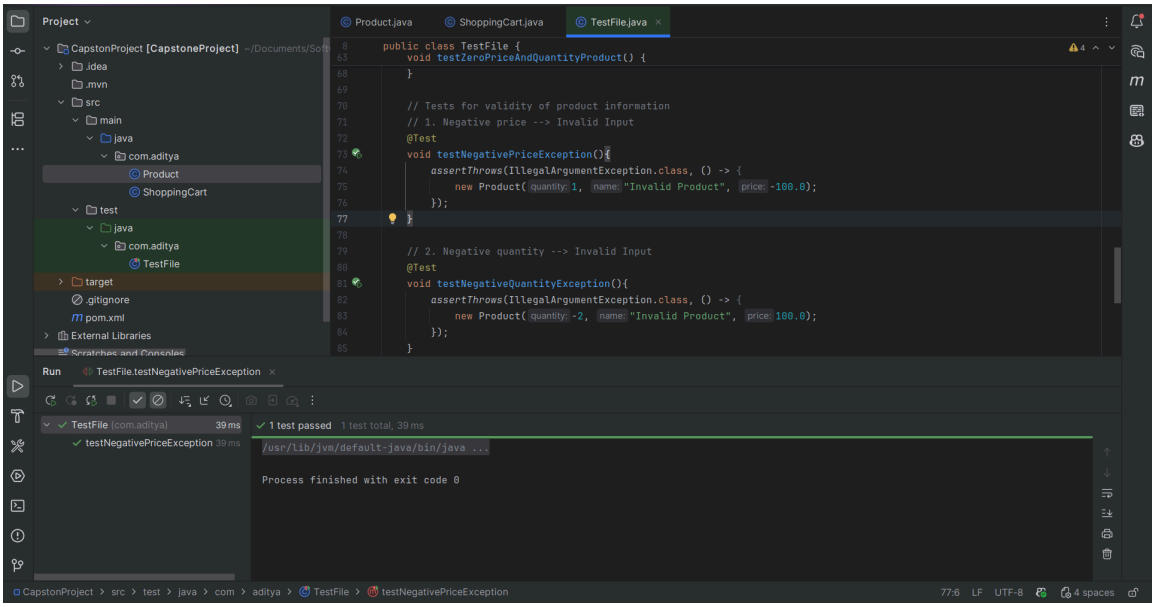
- Edge Case Tests
  - empty cart



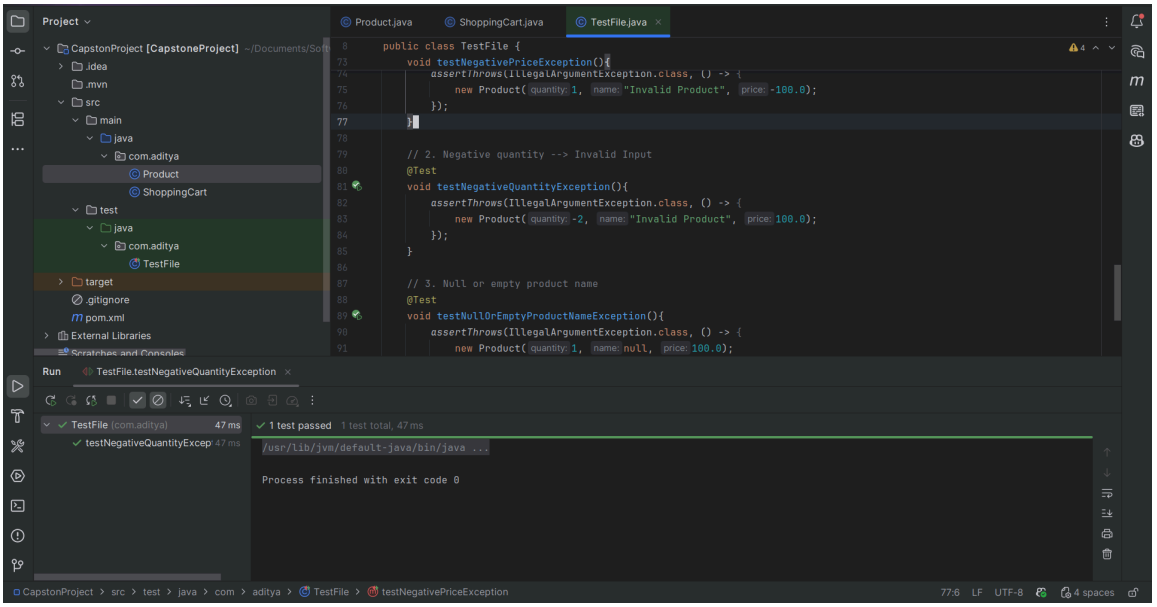
  - zero price and zero quantity product



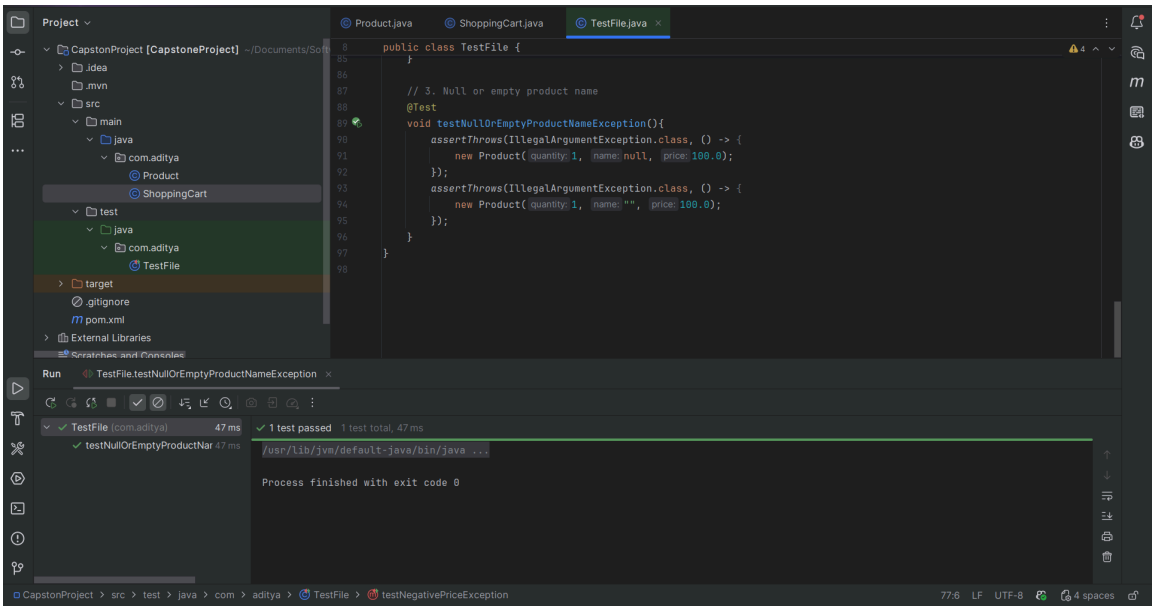
- Test for validity of product information
  - Invalid input → Ngeative price

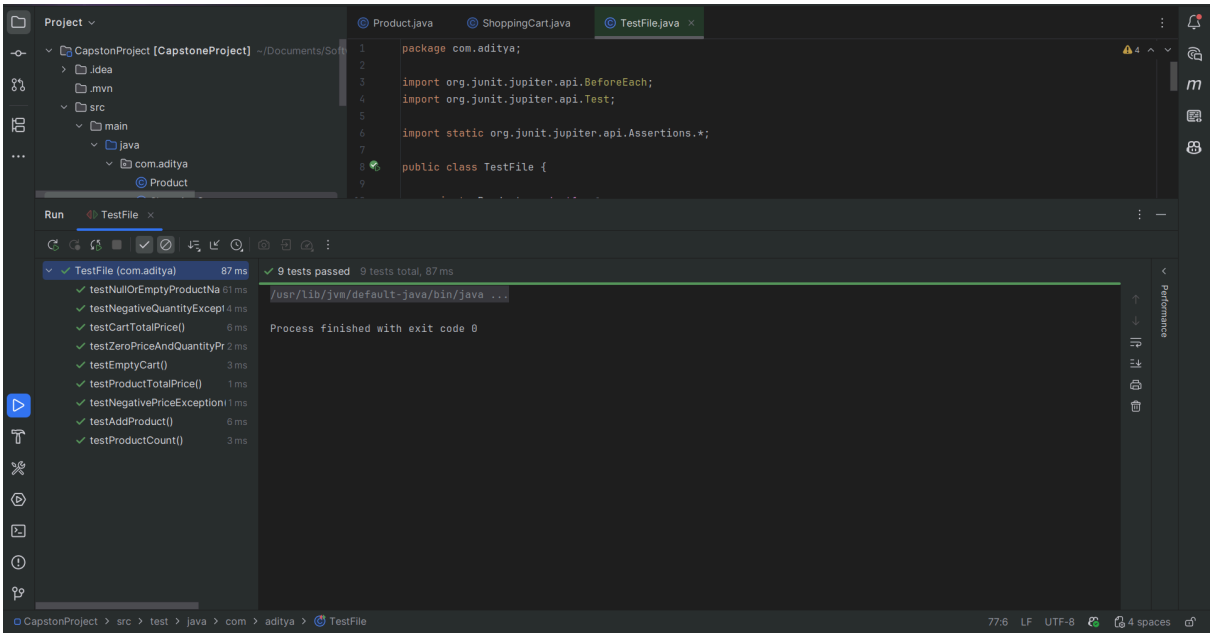


  - Invalid input → Nagtive Quantity


Project
Product.java
ShoppingCart.java
TestFile.java
public class TestFile {
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);
});
}
// 3. Null or empty product name
@Test
void testNullOrEmptyProductNameException(){
assertThrows(IllegalArgumentException.class, () -> {
new Product(quantity: 1, name: null, price: 100.0);
Run
TestFile.testNegativeQuantityException
1 test passed
/usr/lib/jvm/default-java/bin/java ...
Process finished with exit code 0

- Null Value


Project
Product.java
ShoppingCart.java
TestFile.java
public class TestFile {
}
// 3. Null or empty product name
@Test
void testNullOrEmptyProductNameException(){
assertThrows(IllegalArgumentException.class, () -> {
new Product(quantity: 1, name: null, price: 100.0);
});
assertThrows(IllegalArgumentException.class, () -> {
new Product(quantity: 1, name: "", price: 100.0);
});
}
}
Run
TestFile.testNullOrEmptyProductNameException
1 test passed
/usr/lib/jvm/default-java/bin/java ...
Process finished with exit code 0



Question 2

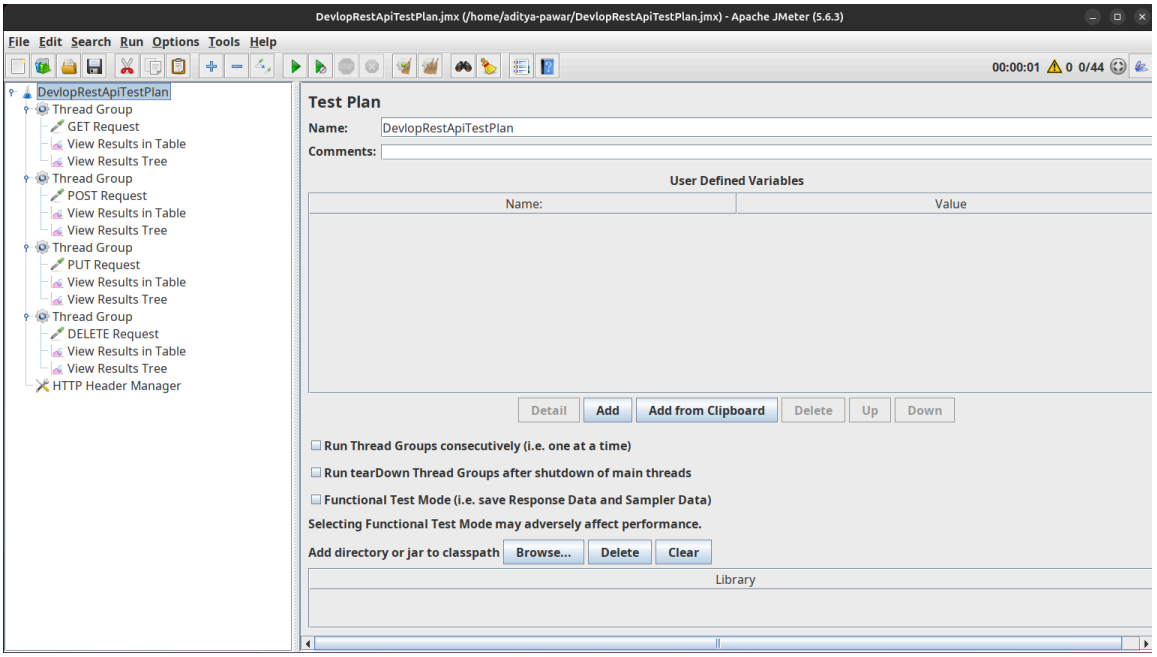
Create a Test Plan in JMeter for simulating traffic to the rest api use appropriate

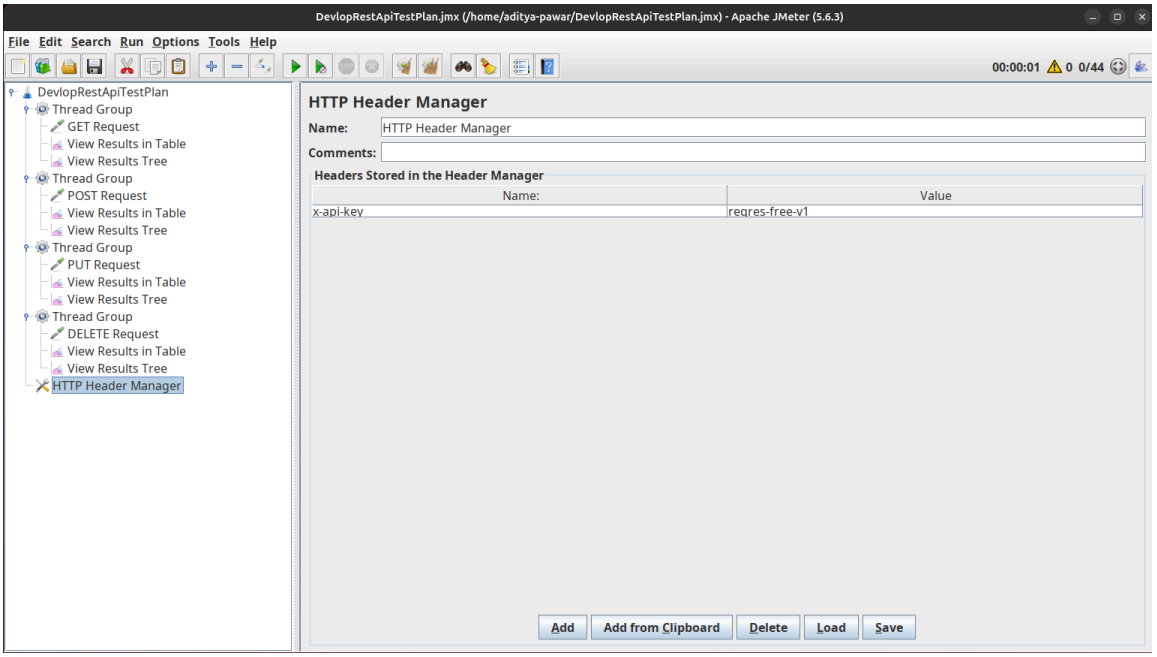
- listeners and samplers
- Name: DevlopRestApiTestPlan
- URL: https://reqres.in
- Users: 20 for following requests methods
  - Get list of all objects
  - Post
  - Update
  - Delete

1. Check for valid jmx for above plan

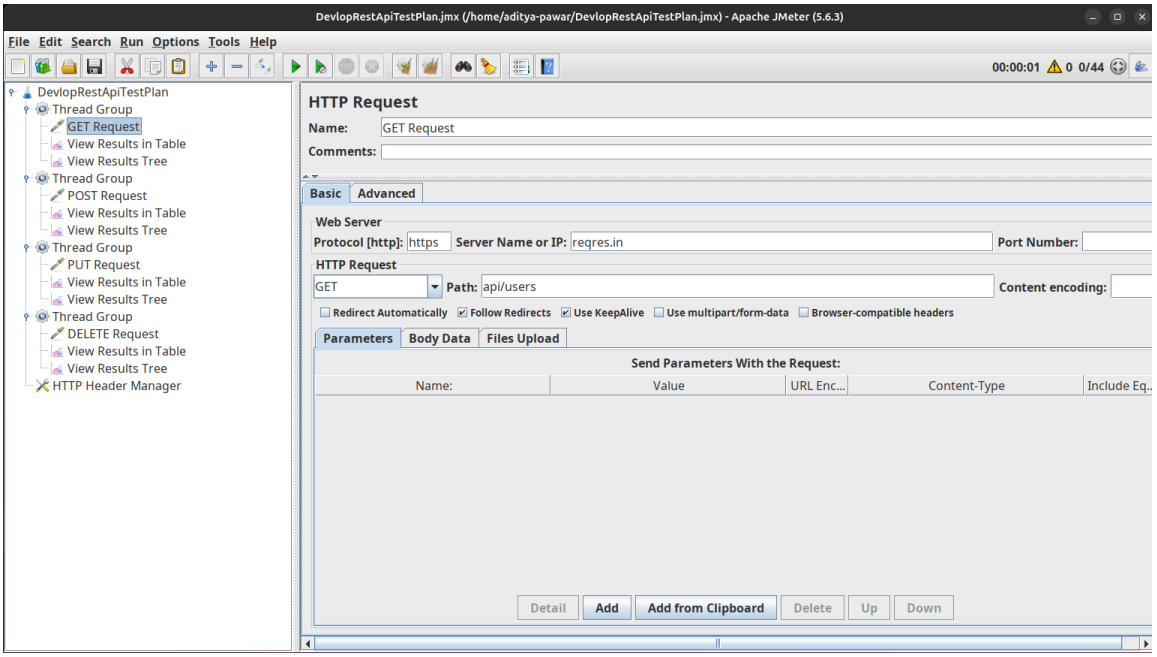
Solution :

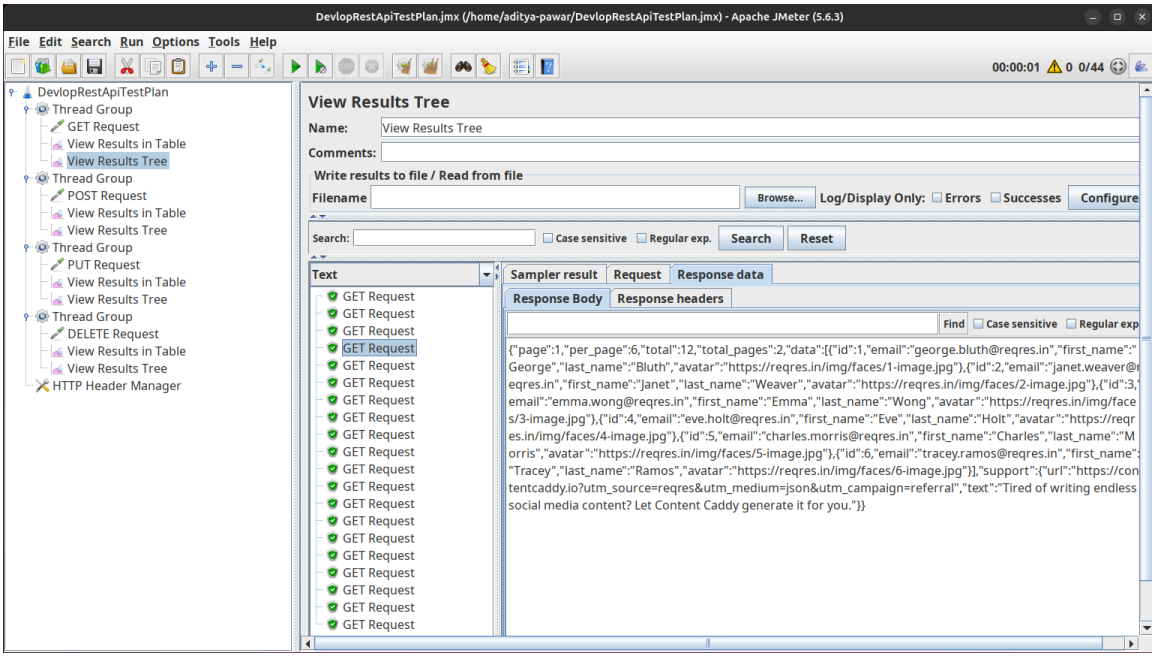
- jmeter Structure


DevlopRestApiTestPlan.jmx (/home/aditya-pawar/DevlopRestApiTestPlan.jmx) - Apache JMeter (5.6.3)
File Edit Search Run Options Tools Help
00:00:01 0 0/44
DevlopRestApiTestPlan
Thread Group
GET Request
View Results in Table
View Results Tree
Thread Group
POST Request
View Results in Table
View Results Tree
Thread Group
PUT Request
View Results in Table
View Results Tree
Thread Group
DELETE Request
View Results in Table
View Results Tree
HTTP Header Manager
Test Plan
Name: DevlopRestApiTestPlan
Comments:
User Defined Variables
Name:
Value
Detail
Add
Add from Clipboard
Delete
Up
Down
Run Thread Groups consecutively (i.e. one at a time)
Run tearDown Thread Groups after shutdown of main threads
Functional Test Mode (i.e. save Response Data and Sampler Data)
Selecting Functional Test Mode may adversely affect performance.
Add directory or jar to classpath
Browse...
Delete
Clear
Library

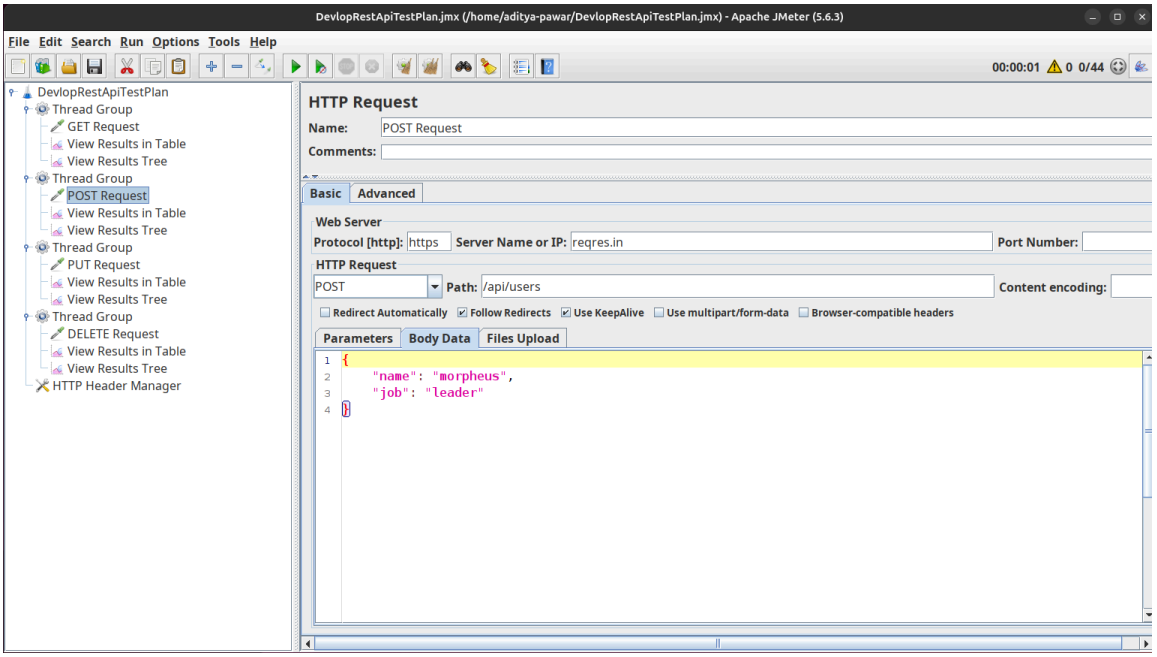

DevlopRestApiTestPlan.jmx (/home/aditya-pawar/DevlopRestApiTestPlan.jmx) - Apache JMeter (5.6.3)
File Edit Search Run Options Tools Help
00:00:01 0 0/44
DevlopRestApiTestPlan
Thread Group
GET Request
View Results in Table
View Results Tree
Thread Group
POST Request
View Results in Table
View Results Tree
Thread Group
PUT Request
View Results in Table
View Results Tree
Thread Group
DELETE Request
View Results in Table
View Results Tree
HTTP Header Manager
HTTP Header Manager
Name: HTTP Header Manager
Comments:
Headers Stored in the Header Manager
Name:
Value
x-api-key
reqres-free-v1
Add
Add from Clipboard
Delete
Load
Save

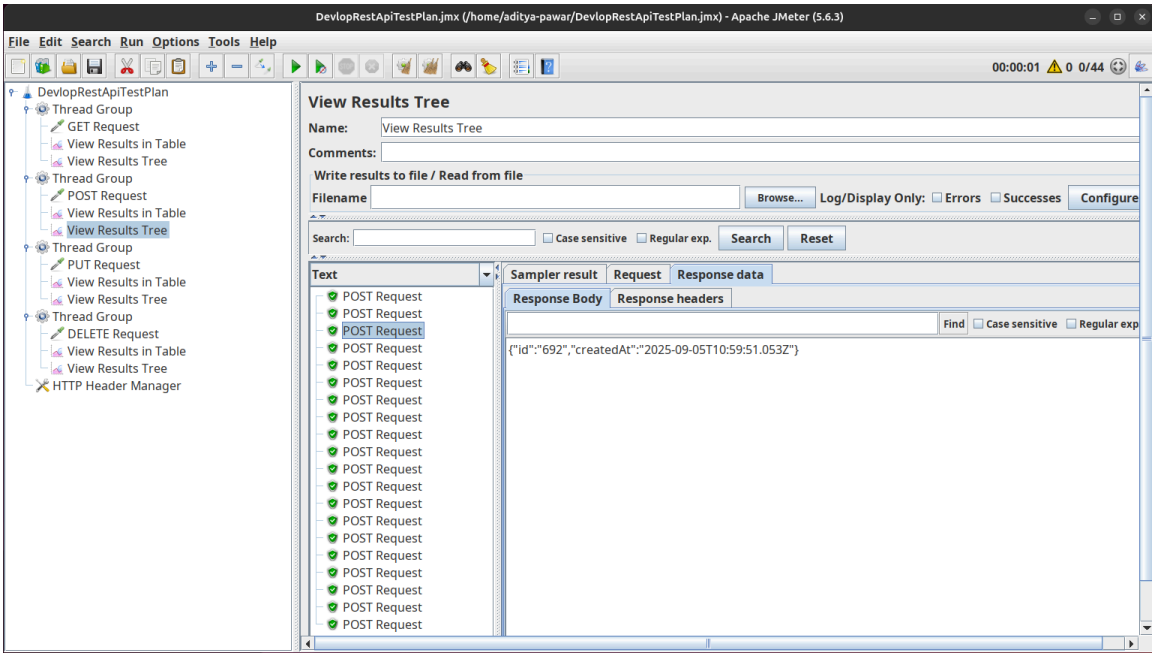
- Get Request



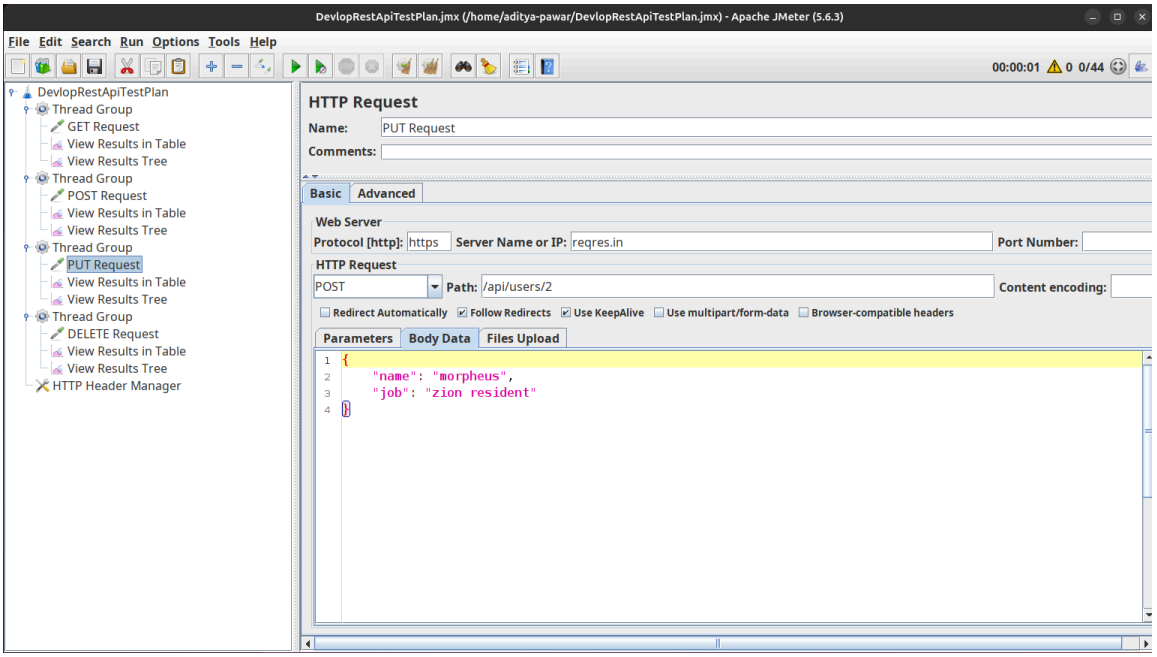


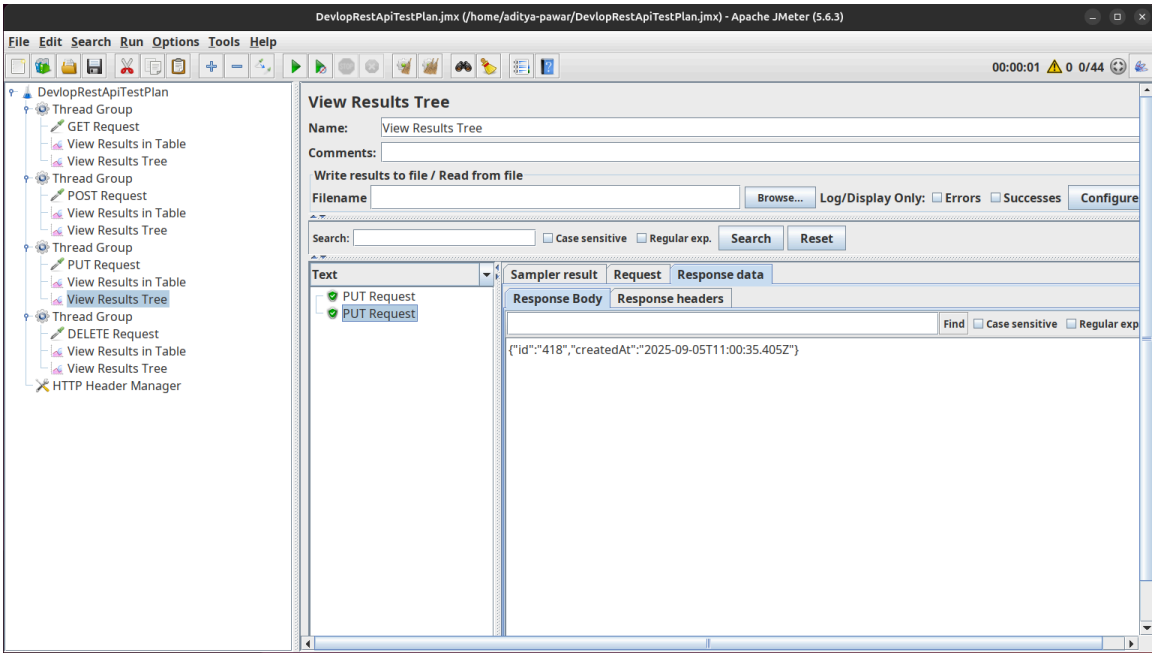
- Post Request


DevlopRestApiTestPlan.jmx (/home/aditya-pawar/DevlopRestApiTestPlan.jmx) - Apache JMeter (5.6.3)
File Edit Search Run Options Tools Help
00:00:01 0 0/44
DevlopRestApiTestPlan
Thread Group
GET Request
View Results in Table
View Results Tree
Thread Group
POST Request
View Results in Table
View Results Tree
Thread Group
PUT Request
View Results in Table
View Results Tree
Thread Group
DELETE Request
View Results in Table
View Results Tree
HTTP Header Manager
HTTP Request
Name: POST Request
Comments:
Basic Advanced
Web Server
Protocol [http]: https
Server Name or IP: reqres.in
Port Number:
HTTP Request
POST
Path: /api/users
Content encoding:
Redirect Automatically
Follow Redirects
Use KeepAlive
Use multipart/form-data
Browser-compatible headers
Parameters
Body Data
Files Upload
{
"name": "morpheus",
"job": "leader"
}

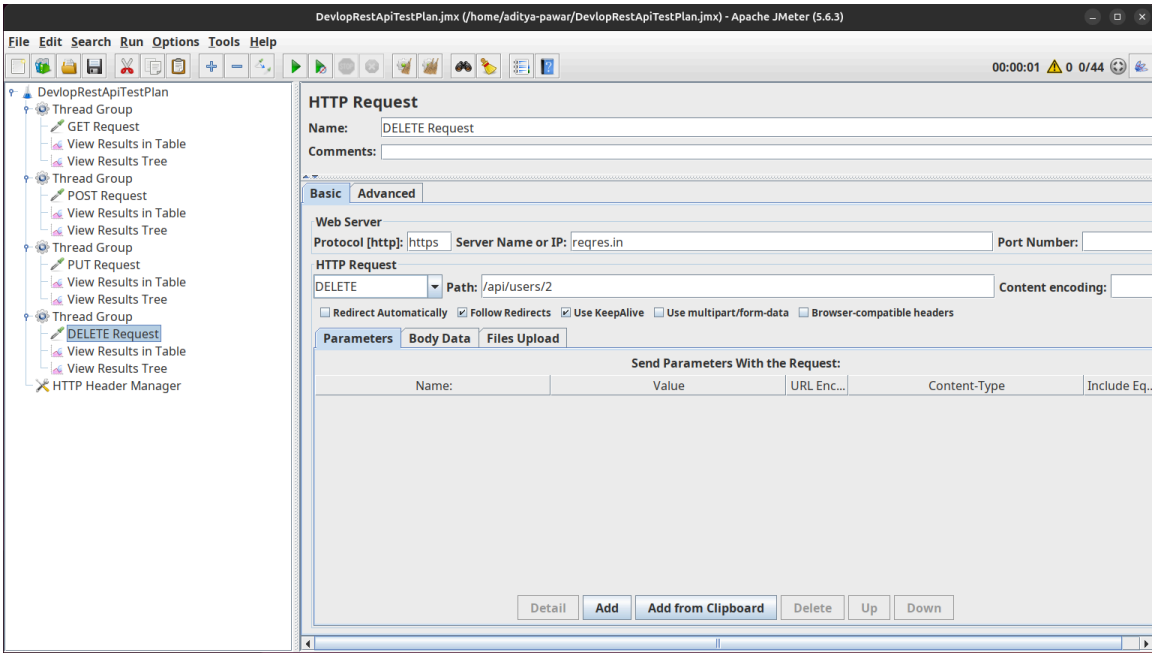

DevlopRestApiTestPlan.jmx (/home/aditya-pawar/DevlopRestApiTestPlan.jmx) - Apache JMeter (5.6.3)
File Edit Search Run Options Tools Help
00:00:01 0 0/44
DevlopRestApiTestPlan
Thread Group
GET Request
View Results in Table
View Results Tree
Thread Group
POST Request
View Results in Table
View Results Tree
Thread Group
PUT Request
View Results in Table
View Results Tree
Thread Group
DELETE Request
View Results in Table
View Results Tree
HTTP Header Manager
View Results Tree
Name: View Results Tree
Comments:
Write results to file / Read from file
Filename
Browse...
Log/Display Only: Errors Successes
Configure
Search:
Case sensitive
Regular exp.
Search
Reset
Text
POST Request
Sampler result
Request
Response data
Response Body
Response headers
Find
Case sensitive
Regular exp
{"id":"692","createdAt":"2025-09-05T10:59:51.053Z"}

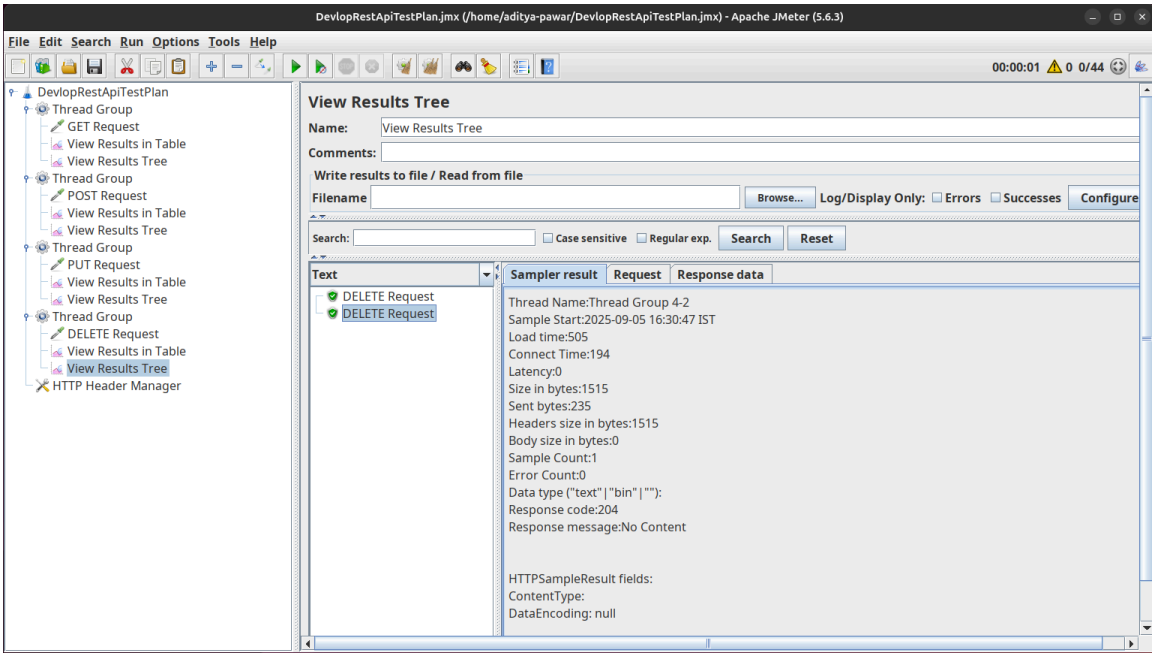
- Put Request





- Delete Request

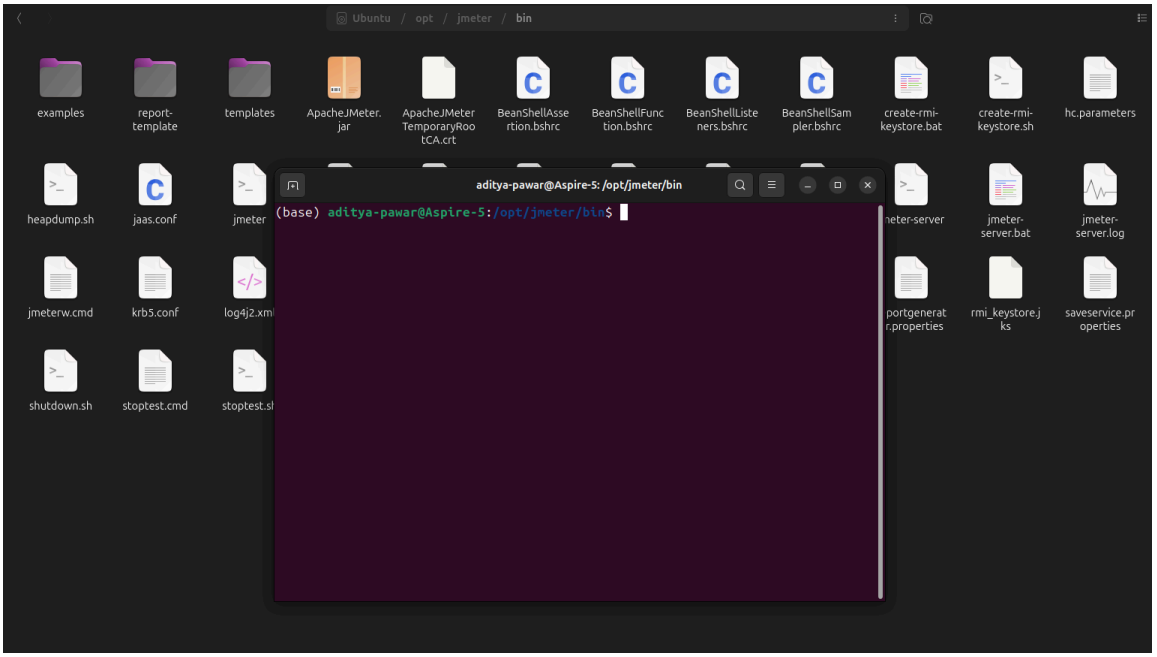




2. Generate HTML Report for the above test plan using non Gui mode of JMeter

STEPS:

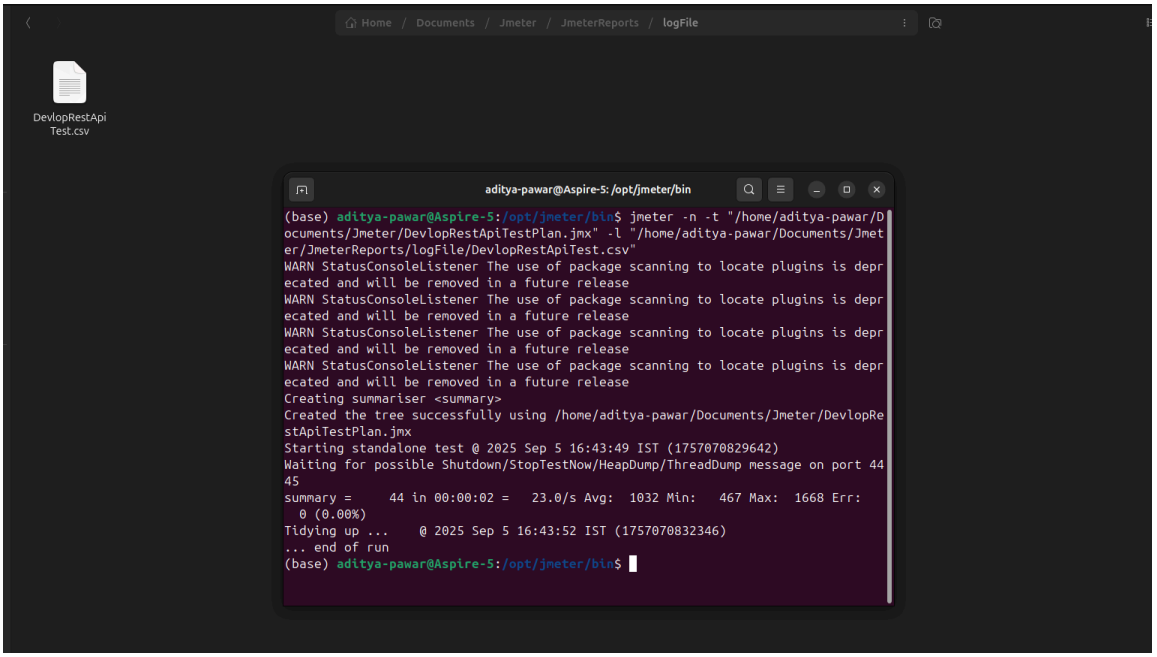
1. Open Jmeter in Non-GUI Mode →
    - /opt/jmeter/bin —- path in terminal



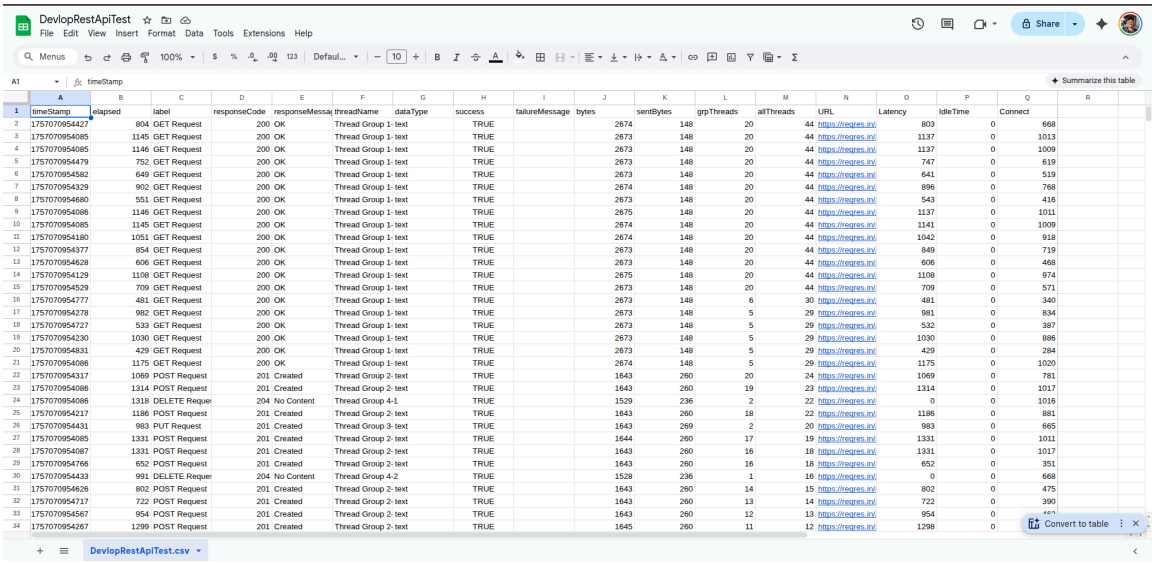
2. use command

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

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



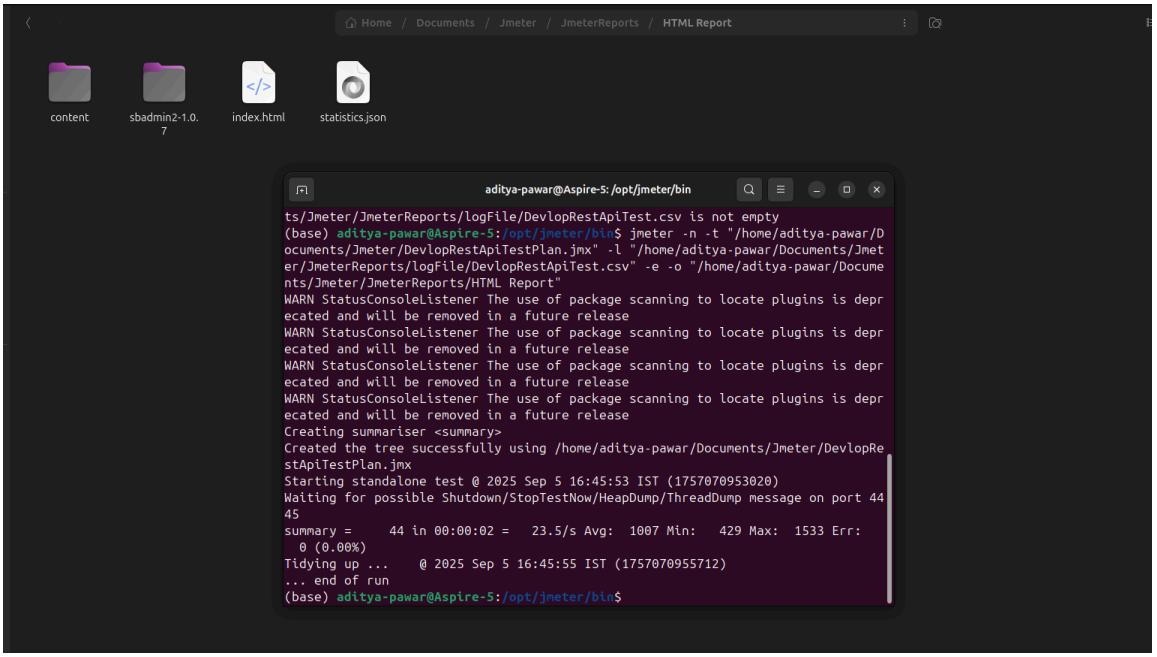
- DevlopRestApiTest csv file



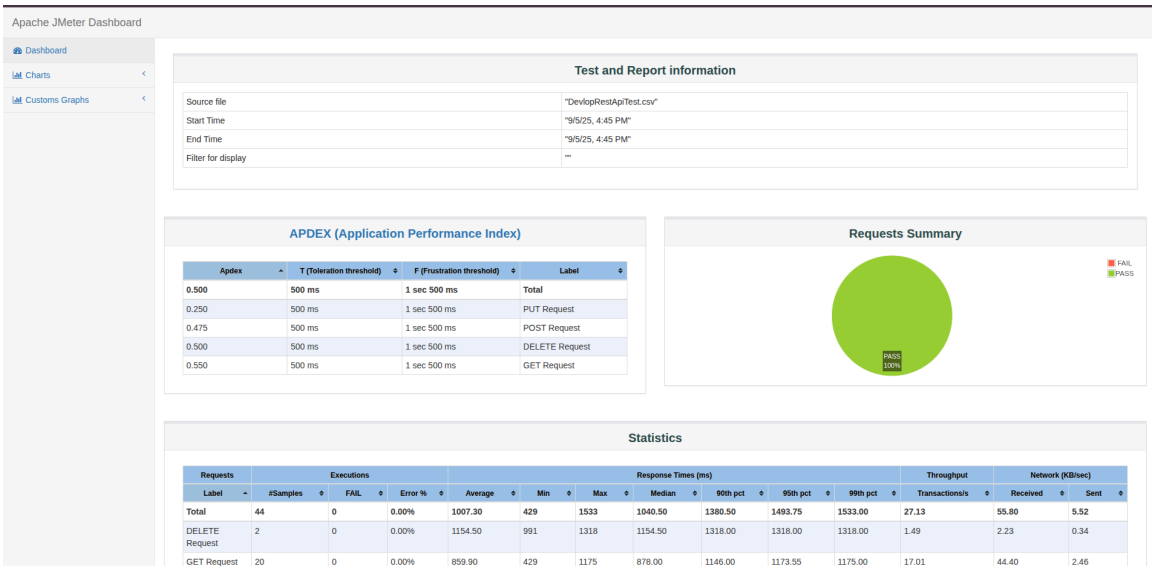
3. Generate HTML report

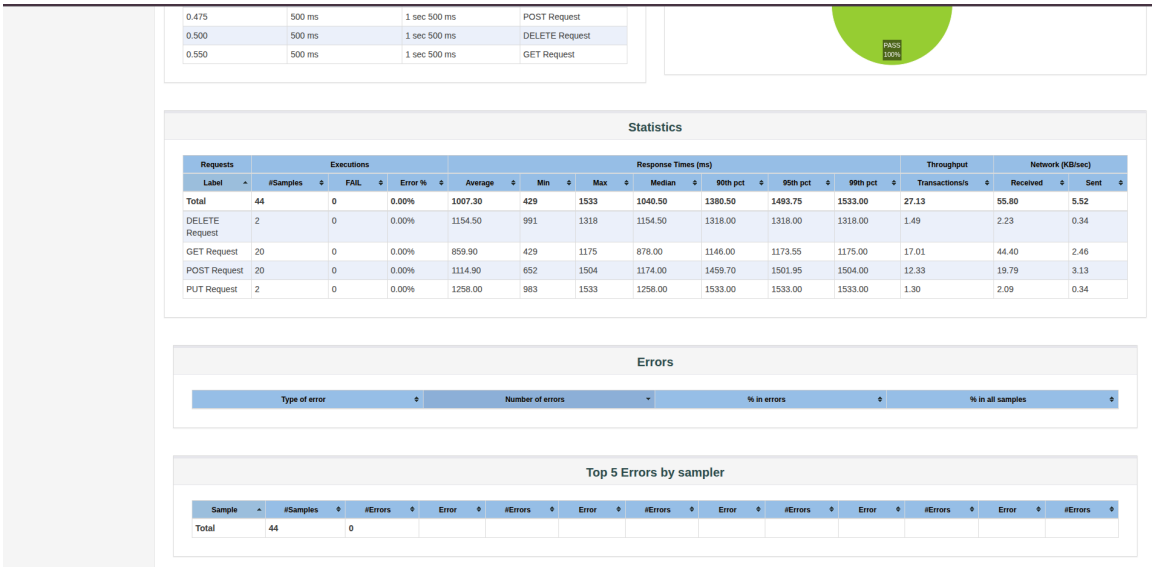
```
jmeter -n -t "/home/aditya-pawar/Documents/Jmeter/DevlopRestApiTestPlan.jmx"
-l "/home/aditya-pawar/Documents/Jmeter/JmeterReports/logFile/DevlopRestApiTest.csv"
```

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



- HTML Report output





0.475	500 ms	1 sec 500 ms	POST Request
0.500	500 ms	1 sec 500 ms	DELETE Request
0.550	500 ms	1 sec 500 ms	GET Request

PASS 100%

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
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
POST Request	20	0	0.00%	1114.90	652	1504	1174.00	1459.70	1501.95	1504.00	12.33	19.79	3.13
PUT Request	2	0	0.00%	1258.00	983	1533	1258.00	1533.00	1533.00	1533.00	1.30	2.09	0.34

Errors

Type of error	Number of errors	% in errors	% in all samples

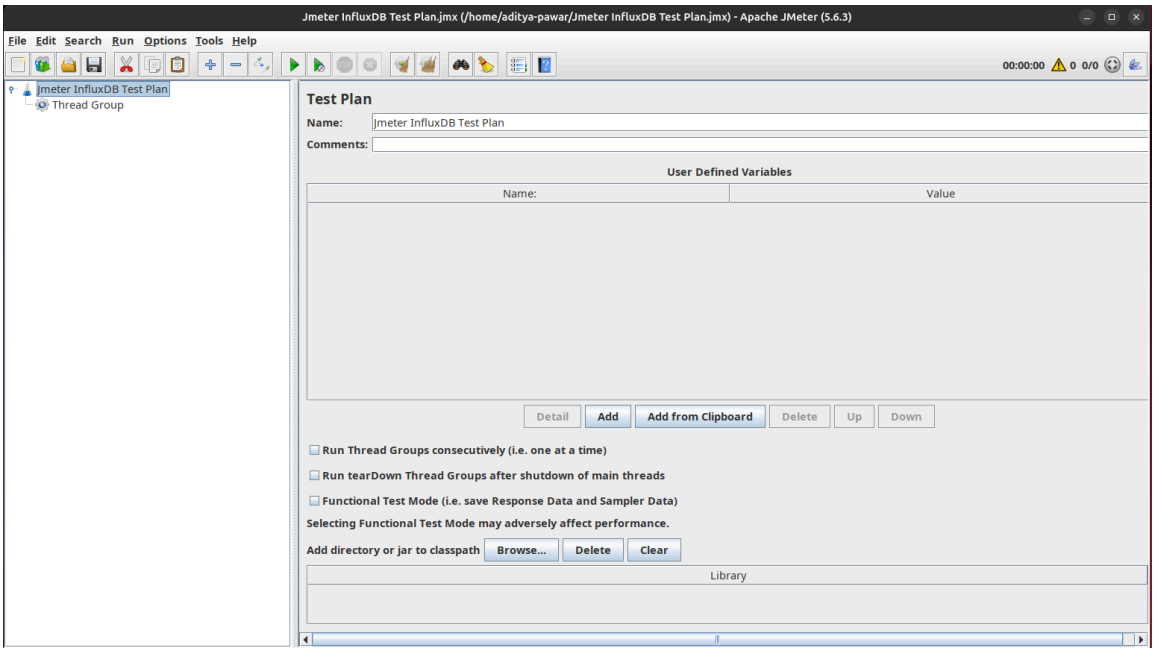
Top 5 Errors by sampler

Sample	#Samples	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors
Total	44	0										

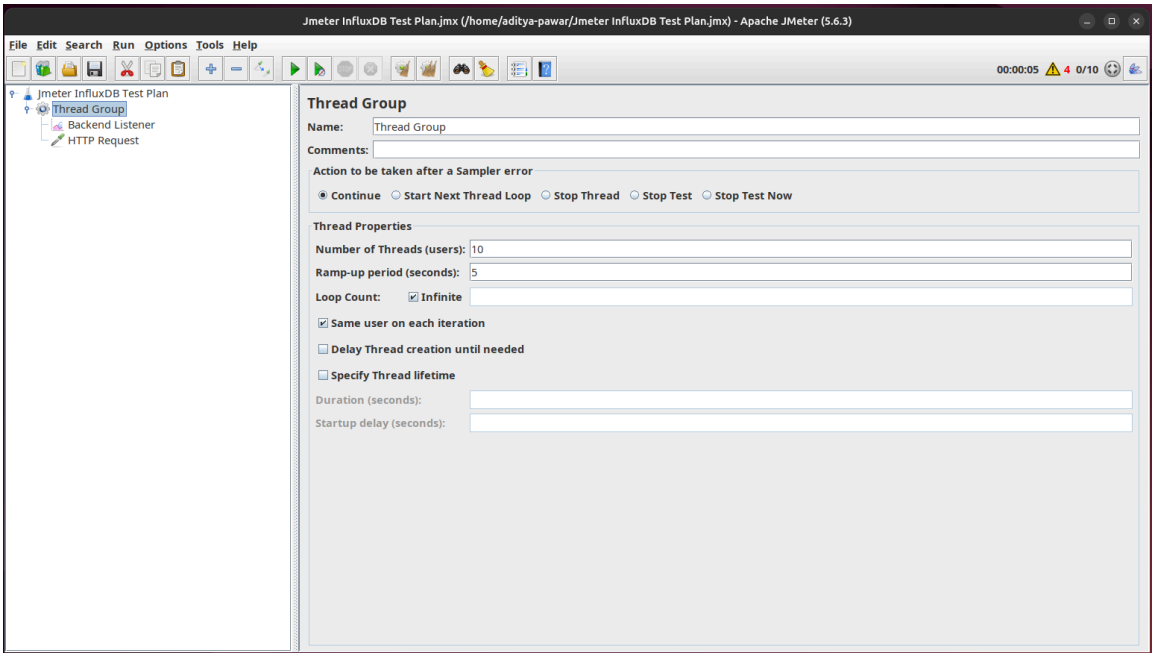
3. Connect JMeter with influx db. and store data in Database DBName : DevInfluxDB

STEPS

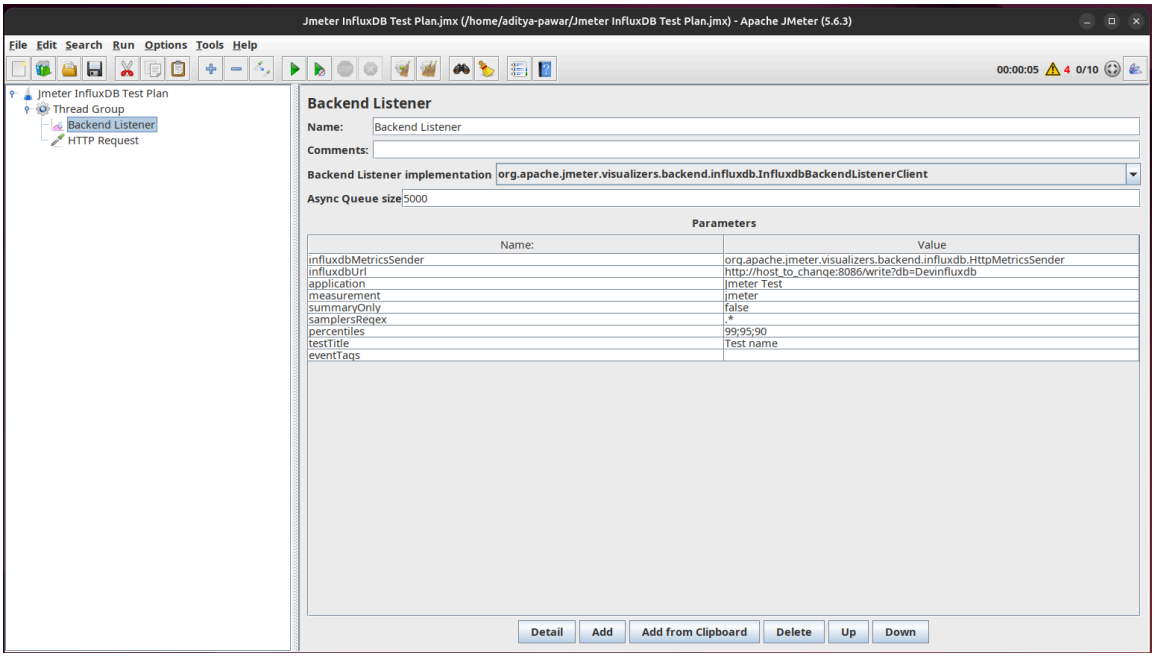
1. create new test Plan



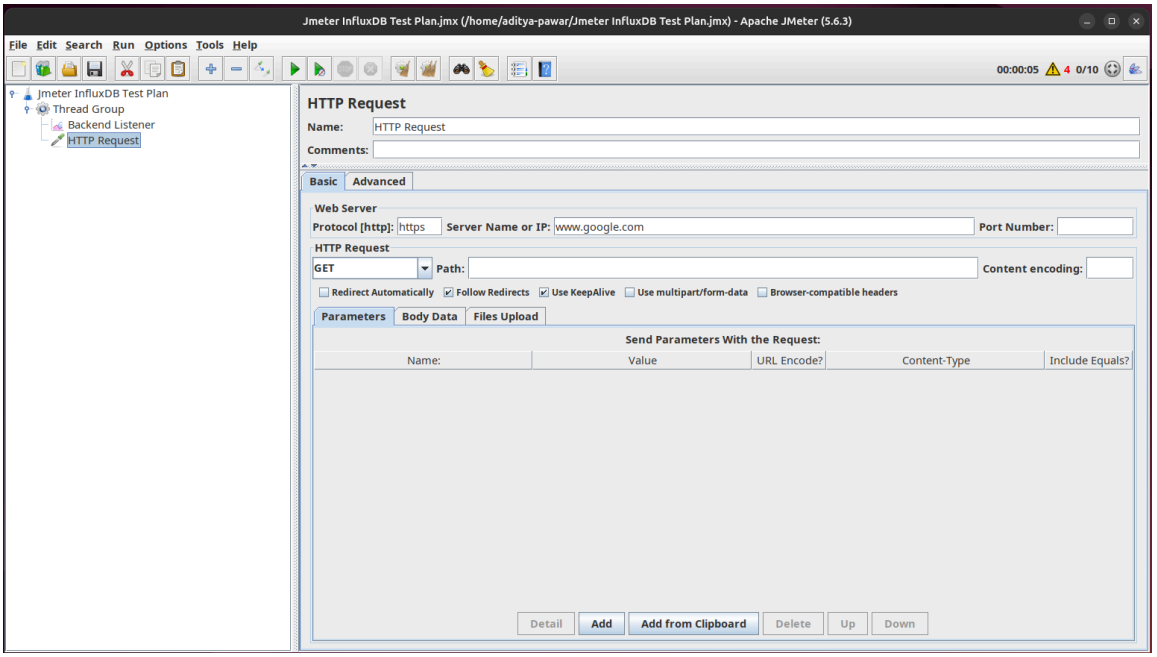
2. add thread group



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 genarate responses

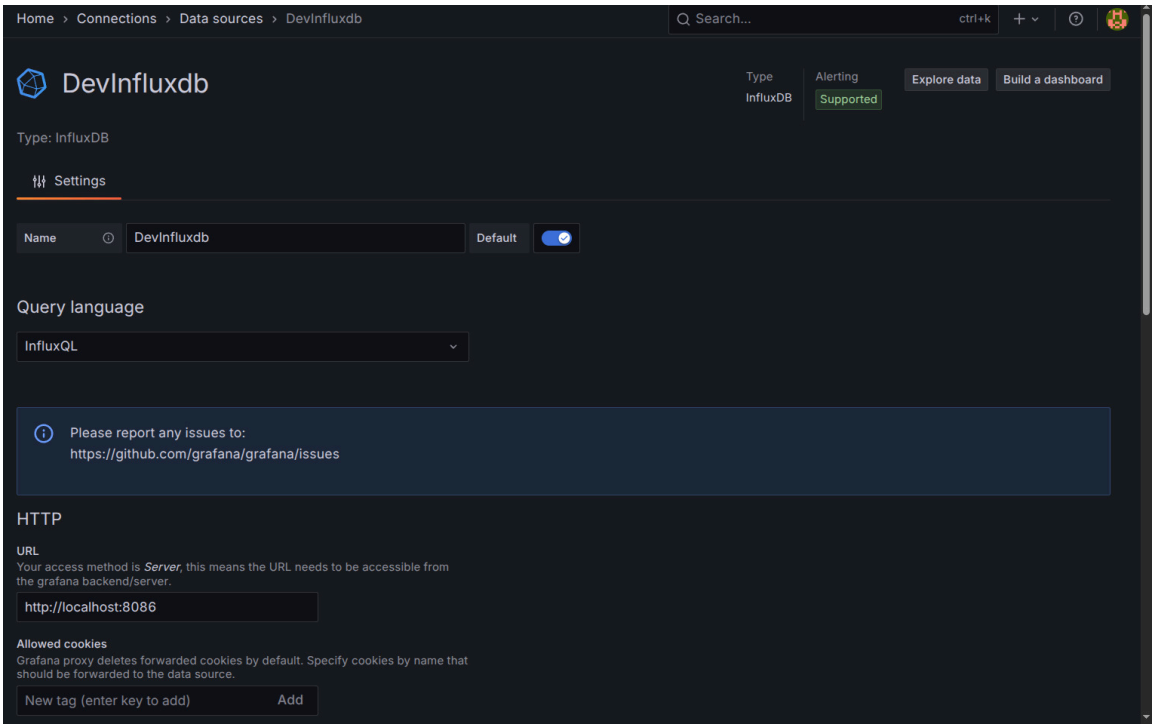
```
aditya-pawar$ jmeter -n -t /home/aditya/Documents/Capstone/Jmeter\ InfluxDBTest\ Plan.jmx
```

7. Jmeter Pushed Data to InfluxDB :

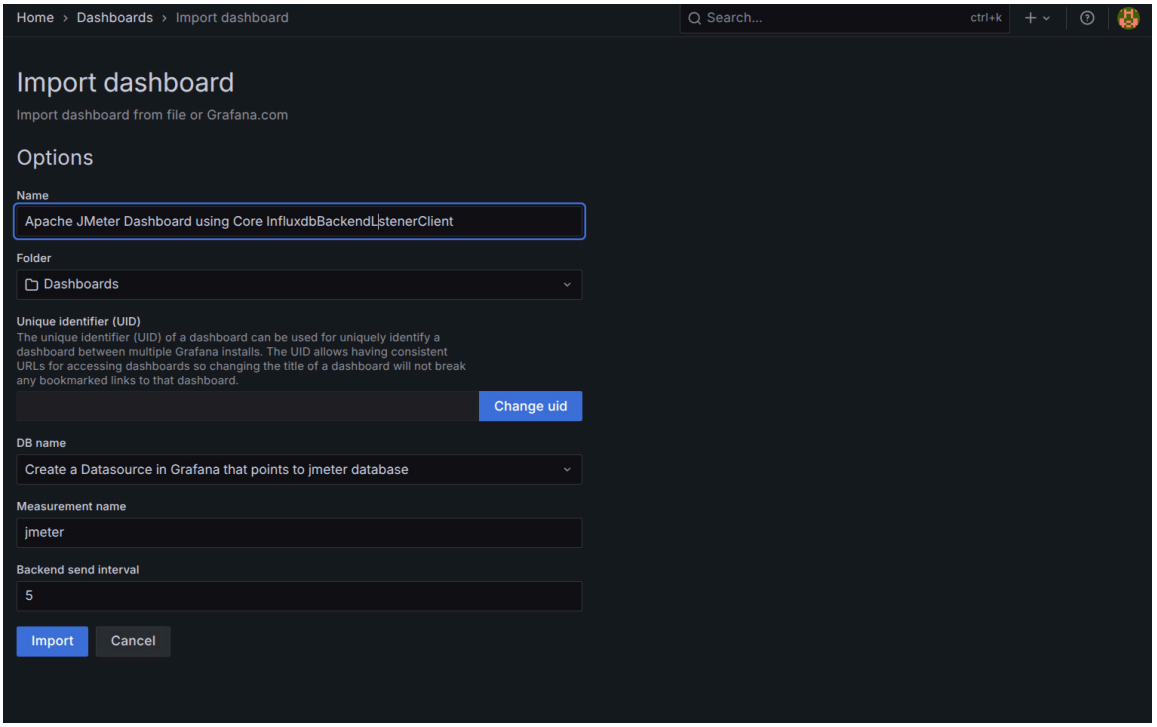
```
Starting standalone test @ 2025 Sep 5 18:19:00 IST (1757076540301)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary +      1 in 00:00:01 =    0.9/s Avg:   972 Min:   972 Max:   972 Err:     0 (0.00%) Active: 3 Started: 3 Finished: 0
summary +    278 in 00:00:28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

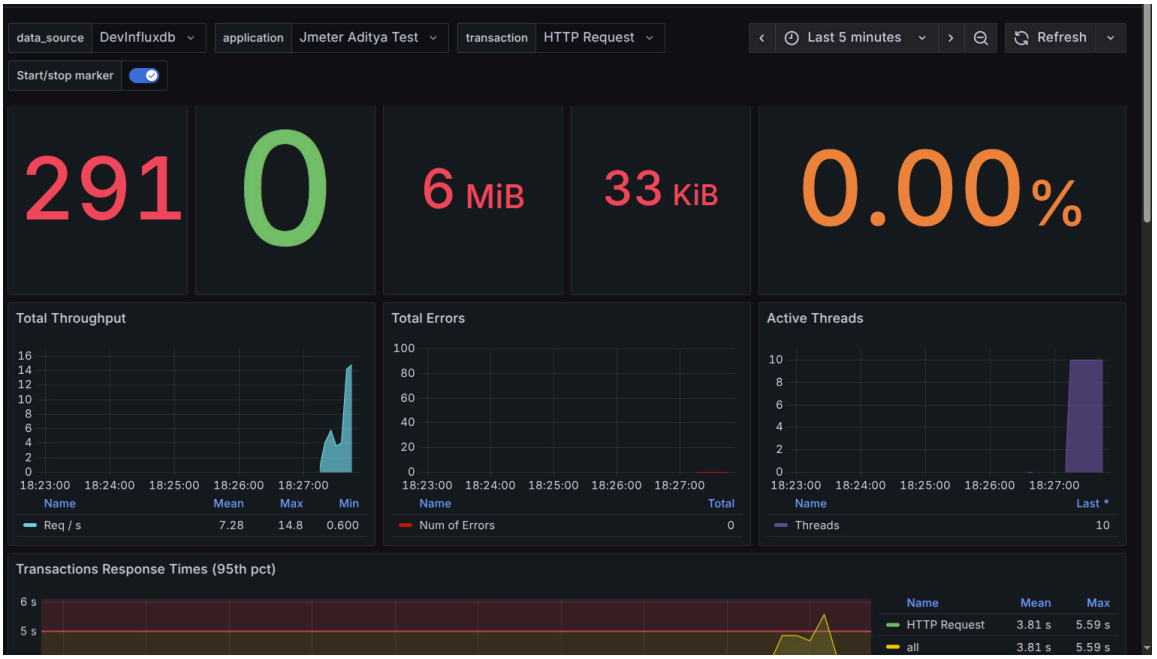


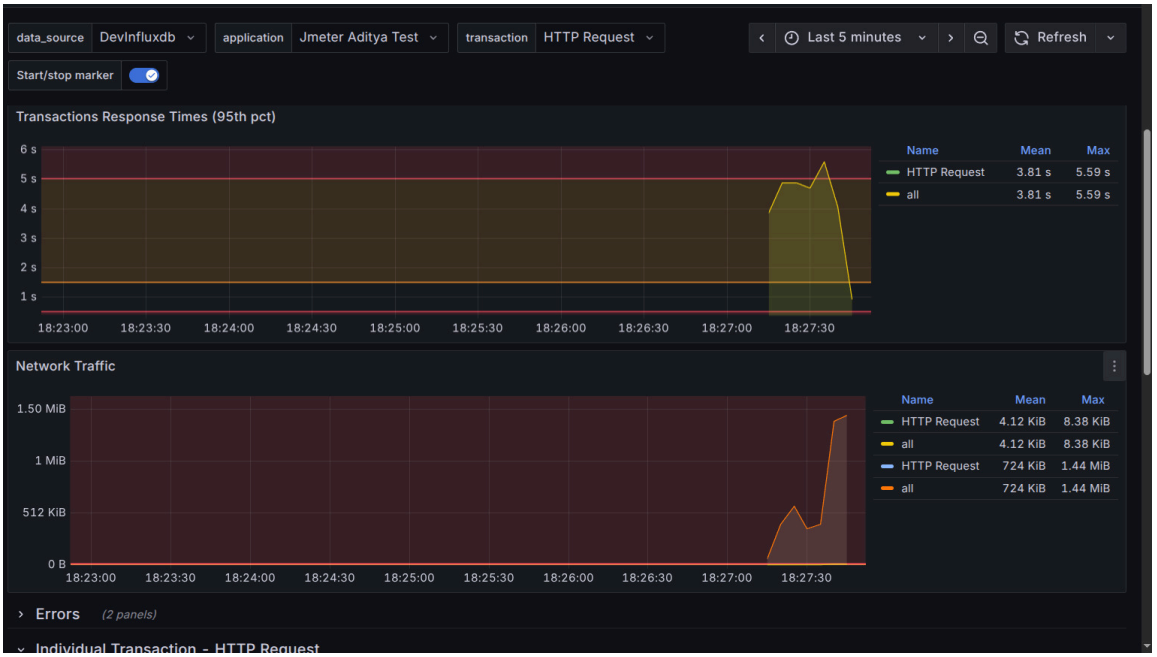
2. Importing Dashboard For InfluxDB in Graphana :

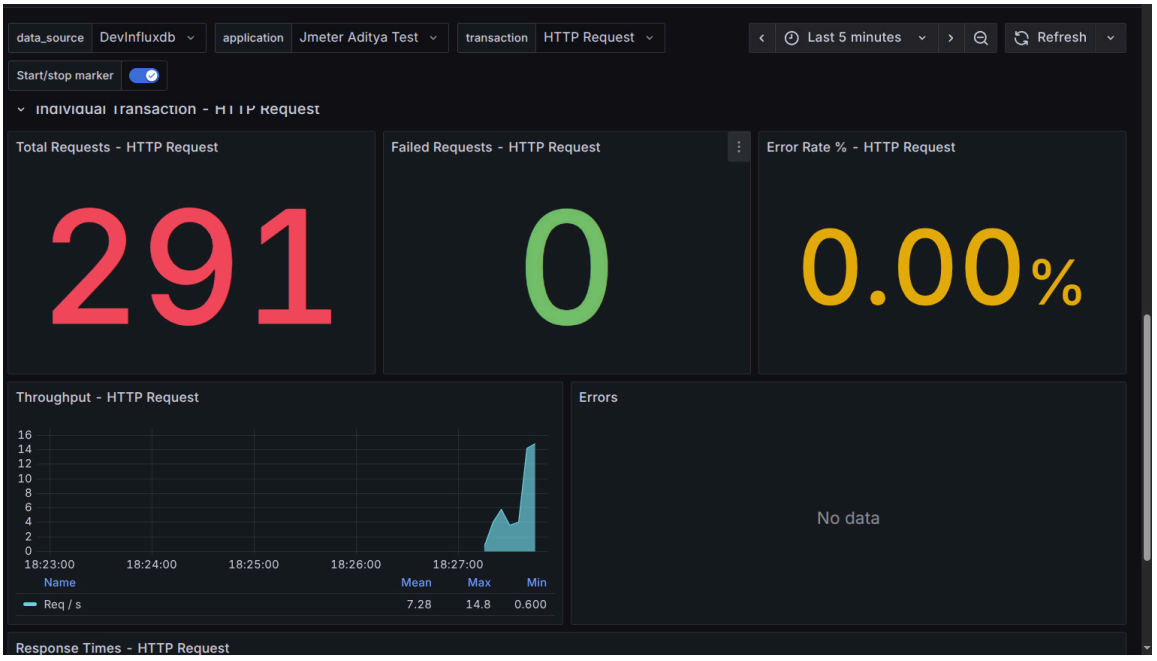


3. run jmx file

4. this is final output




data_source
DevInfluxdb
application
Jmeter Aditya Test
transaction
HTTP Request
Last 5 minutes
Refresh
Start/stop marker
Transactions Response Times (95th pct)
6 s
5 s
4 s
3 s
2 s
1 s
18:23:00
18:23:30
18:24:00
18:24:30
18:25:00
18:25:30
18:26:00
18:26:30
18:27:00
18:27:30
Name
Mean
Max
HTTP Request
3.81 s
5.59 s
all
3.81 s
5.59 s
Network Traffic
1.50 MiB
1 MiB
512 KiB
0 B
Name
Mean
Max
HTTP Request
4.12 KiB
8.38 KiB
all
4.12 KiB
8.38 KiB
HTTP Request
724 KiB
1.44 MiB
all
724 KiB
1.44 MiB
Errors
(2 panels)
Individual Transaction - HTTP Request


data_source
DevInfluxdb
application
Jmeter Aditya Test
transaction
HTTP Request
Last 5 minutes
Refresh
Start/stop marker
Individual Transaction - HTTP Request
Total Requests - HTTP Request
291
Failed Requests - HTTP Request
0
Error Rate % - HTTP Request
0.00%
Throughput - HTTP Request
16
14
12
10
8
6
4
2
0
18:23:00
18:24:00
18:25:00
18:26:00
18:27:00
Name
Mean
Max
Min
Req / s
7.28
14.8
0.600
Errors
No data
Response Times - HTTP Request