Performance Test Strategy

To effectively performance test the DemoBlaze website we have to consider the following major API calls flows which can be segregated in below 4 types:

1. Log In Flow
2. Sign up Flow.
3. Load products by each category
4. End to End Purchase flow

API details :

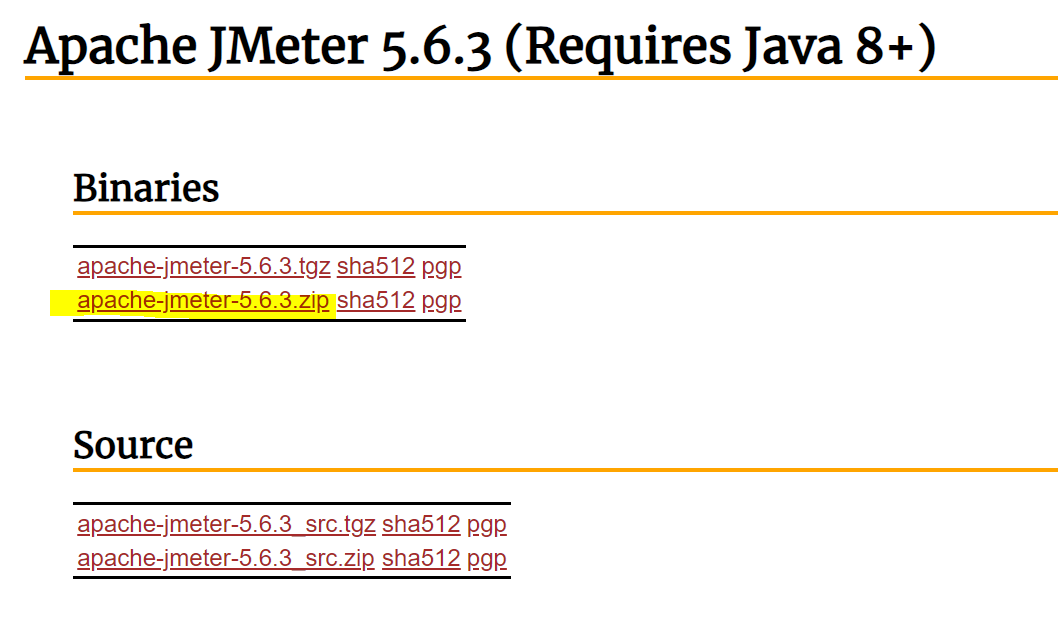
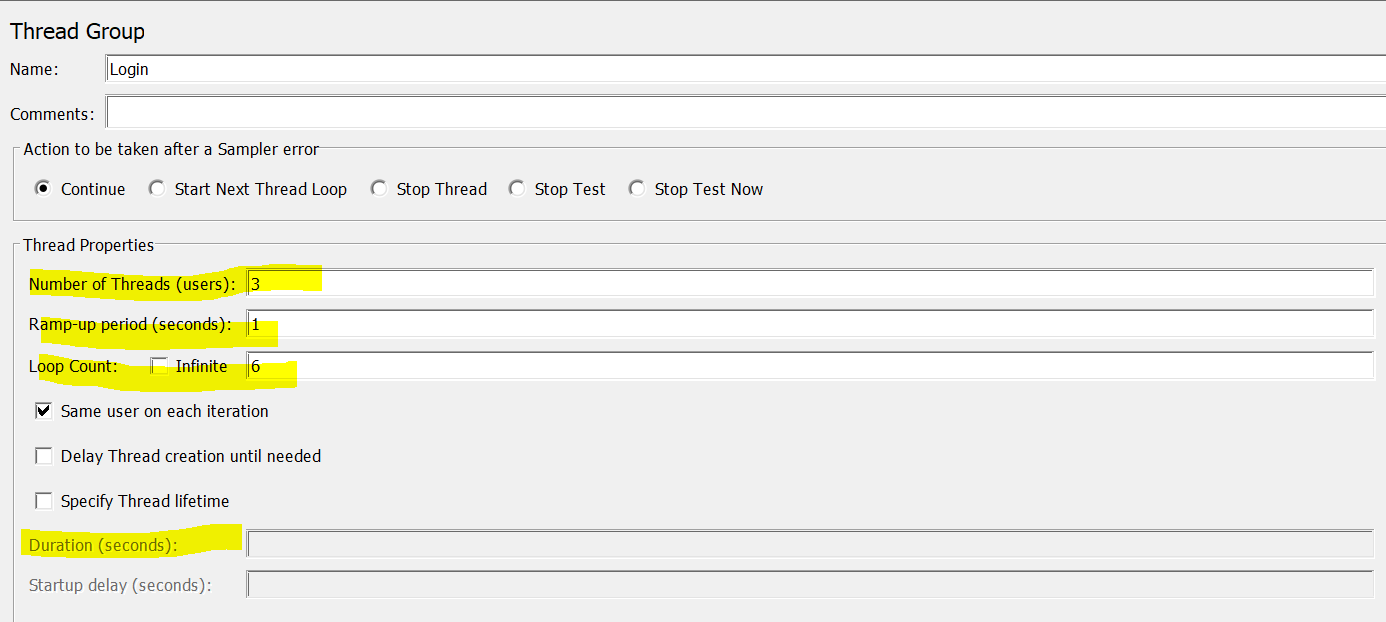
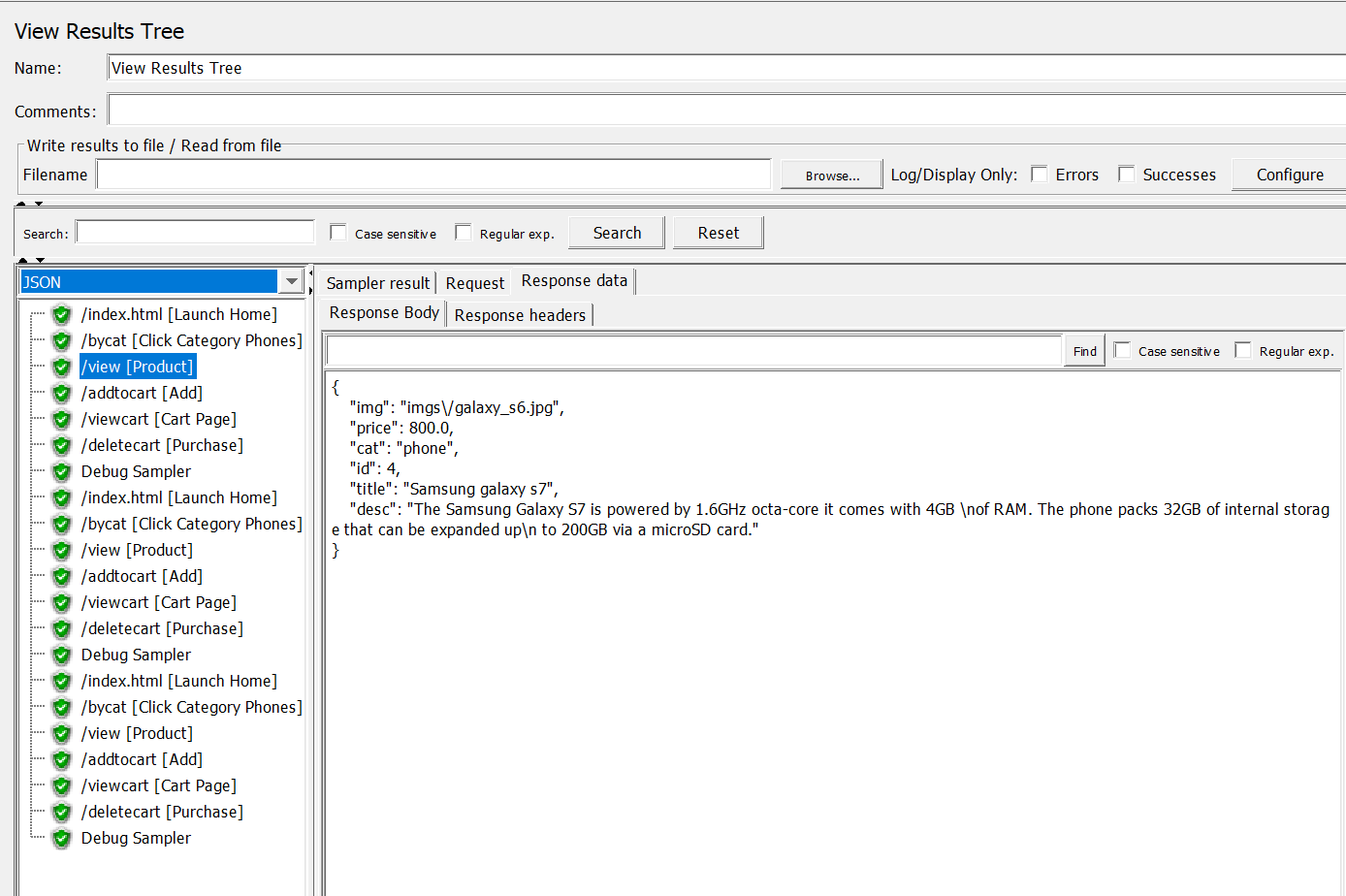
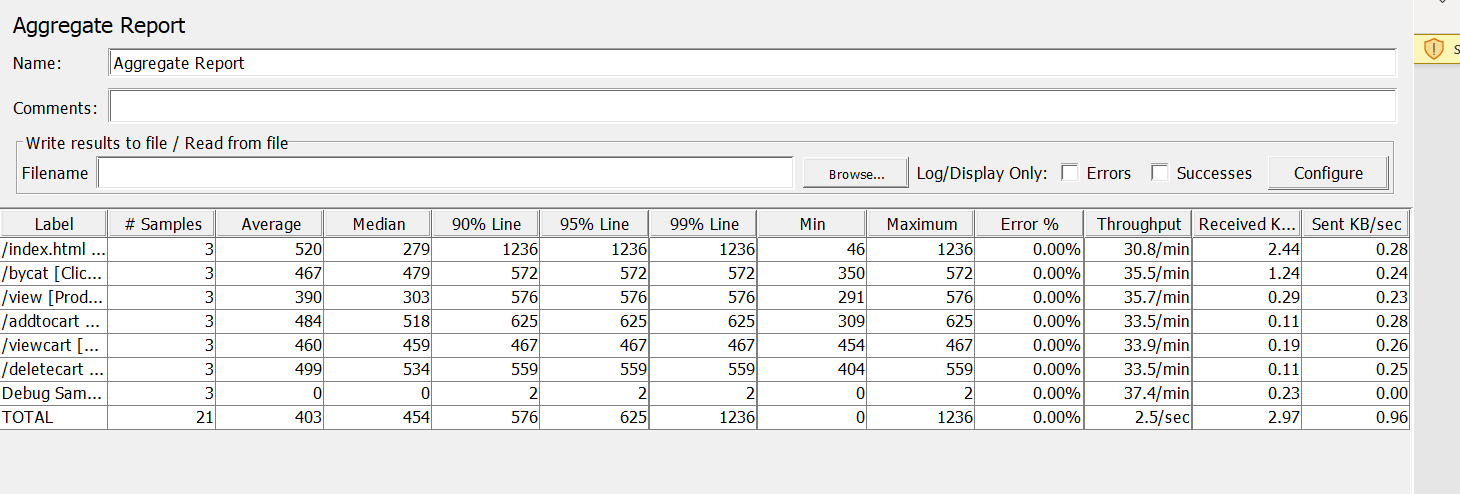
1. Login
   1. POST https:*//api.demoblaze.com/login*  
        
      POST data:  
      {"username":"test","password":"dGVzdA=="}
2. Sign up.
   1. POST https:*//api.demoblaze.com/signup*  
        
      POST data:  
      {"username":"BXVh7","password":"bmV3cnQ0NQ=="}

1. Load Products by Category
   1. Phones
      1. POST https:*//api.demoblaze.com/bycat*  
           
         POST data:  
         {"cat":"phone"}
   2. Laptops
      1. POST https:*//api.demoblaze.com/bycat*  
           
         POST data:  
         {"cat":"notebook"}
   3. Monitors
      1. POST https:*//api.demoblaze.com/bycat*  
           
         POST data:  
         {"cat":"monitor"}
2. Purchase Flow
   1. Launch DemoBlaze Home
      1. GET https:*//www.demoblaze.com/index.html*
   2. Click Category Phones
      1. POST https:*//api.demoblaze.com/bycat*  
           
         POST data:  
         {"cat":"phone"}
   3. View Product
      1. POST https:*//api.demoblaze.com/view*  
           
         POST data:  
         {"id":"4"}
   4. Add to Cart
      1. POST https:*//api.demoblaze.com/addtocart*  
           
         POST data:  
         {"id":"b961685e-4fce-eece-0ba9-f39a28b60925","cookie":"user=ac542990-1835-67ed-bb86-7a280e9cb45f","prod\_id":4,"flag":**false**}
   5. View Cart
      1. POST https:*//api.demoblaze.com/viewcart*  
           
         POST data:  
         {"cookie":"user=ac542990-1835-67ed-bb86-7a280e9cb45f","flag":**false**}
   6. Purchase Product (basically Demo blaze just deletes the cart)
      1. POST https:*//api.demoblaze.com/deletecart*  
           
         POST data:  
         {"cookie":"user=ac542990-1835-67ed-bb86-7a280e9cb45f"}

Performance Test Script

1. The performance test script has been automated using **JMeter V5.6.2.**
2. The performance test automation script in .jmx format can be found in **PerformanceTest** folder
3. The script name is: **DemoBlazePerformanceTest.jmx**
4. I have implemented all the major test scenarios as mentioned above.

Set Up

1. Go to Apache official website: <https://jmeter.apache.org/download_jmeter.cgi> to download the latest version of JMeter
2. 
3. Download the zip file and unzip it anywhere you want in your system.
4. Go to bin folder and then open **jmeter.batch** file
5. Make sure your machine has Java installed.
6. Once JMeter UI is launched you can click on **file>open and browse the DemoBlazePerformanceTest.jmx** in the Git repository and open and execute
7. There are 4 thread groups each for the scenarios mentioned above.
8. Adjust the below values as per your load test configuration.
   1. 
9. The test results can be seen in
   1. **View results in Tree**
      1. 
   2. **Aggregate report format**
      1. 

Task : Execute Performance Tests (3 days)

**Load Test:** Demo blaze can handle expected user traffic.

Output:

1. Get the anticipated traffic on Demo blaze website for every day in hits per second from the Product Owner/ Business stakeholders.
2. Perform the Load test to check if the application can handle the expected traffic.

**Stress Test:** To check for 20% more than anticipated traffic on the demo blaze website.

Output:

1. To find the breakpoint of the application
2. To check if application under stress can be scaled either horizontally or vertically.

**Spike Test:** This is to simulate a sudden spike or peak load on the application for APIs like add to cart and checkout. This will be useful when we have any mega sale event like 10.10 or 11.11 sale when the users will suddenly rush to the website to buy at discounted price.

Output:

1. To check that the application can handle a sudden burst in traffic.
2. To check if we need to scale up the infra and deployment before the sale.
3. To check that the response time is not impacted.
4. To check the failure rate is <2% and user experience is not dropped.

**Soak Test:** This to run a performance testing on demo blaze for >24 hours to 48 hours.

Output:

1. To Determine if the application have any memory leaks or dead logs over long run.
2. To check the response time for critical APIs, remain same over period of longer time.
3. To check there is no sudden crash or application failures.

Test Metrices to be Collected in each Performance Run

1. From **JMeter: Collect**
   1. Response times (min, max, avg)
   2. 95% response time
   3. 99% response time
   4. Error %
   5. Throughput/ hits per second
   6. Graphs:
      1. Response codes per second
      2. Hits per second
      3. Response time over time
      4. Active thready over time
2. From **Server: Collect (or APM tool like Dynatrace)**
   1. CPU utilization
   2. Memory utilization

Exit Criteria

1. For the performance tests to be **successful the 95% response time** for all the APIs should be **<2 seconds**
2. The **overall error rate** for the performance test scenarios **should be < 2%**
3. **No Exceptions, Errors and Warnings** should be seen in the server logs during and after tests
4. **The CPU and Memory utilization** of application servers **should be <80%**
5. Any **Data bases and caches like Redis** should have optimum server stats like CPU and memory utilization in healthy range.