OPENCART TESTING PROJECT

Using JMeter

This project focuses on performance testing of an OpenCart-based e-commerce application using Apache JMeter.

Pramodya Ilukpitiya

Introduction

This project focuses on performance testing of an OpenCart-based e-commerce application using Apache JMeter. The objective is to evaluate system behavior under different load conditions such as functional, load, stress, spike, and endurance testing. The results help identify performance bottlenecks and ensure application stability.

- ❖ OpenCart is an open-source PHP-based e-commerce platform.
- ❖ Used for product listing, cart, checkout, admin management.
- ❖ Performance is critical due to concurrent users, payments, and peak traffic.

Objectives

- To perform performance testing using Apache JMeter
- To analyze response time and throughput
- To identify system limitations under load
- To validate OpenCart stability under stress

JMeter

JMeter is a popular open-source tool for performance testing and load testing web applications, APIs, and servers. For QA purposes, it helps ensure that your application can handle expected load and performs reliably under stress. Here's a detailed overview and guide for QA testing using JMeter

What JMeter Can Test

- **Web applications**: Test performance of web servers (HTTP/HTTPS).
- **APIs**: Test REST or SOAP services.
- **Database queries**: JDBC connections for load testing.
- **FTP, LDAP, JMS**: Various protocols supported.

Key Concepts in JMeter

1. **Test Plan**: The container for your entire test.
2. **Thread Group**: Defines virtual users, ramp-up time, and loop count.

3. **Samplers**: Actions that JMeter performs (e.g., HTTP Request, JDBC Request).
4. **Listeners**: Collect and visualize results (e.g., graphs, tables).
5. **Config Elements**: Used to configure requests, like HTTP Headers or Cookies.
6. **Assertions**: Validate responses (e.g., check response code, content).
7. **Timers**: Add delays between requests to simulate real user behavior.
8. **Pre & Post-Processors**: Manipulate requests/responses, e.g., extract tokens.

Scope of testing

In Scope

- Homepage
- Login
- Product search
- Cart page

Out of Scope

- Payment gateway testing
- Security testing
- Mobile app testing

Test Environment

〉 OS: Windows 10

〉 Tool: Apache JMeter (version)

〉 Server: Localhost XAMPP

Detailed Test Scenarios

Test Case ID	Scenario	Request Type	Expected Result
TC_01	Home Page Load	GET	Status 200
TC_02	Login	POST	Login success
TC_03	Search Product	GET	Product list

Steps to Perform JMeter Testing

Step 1: Install JMeter

- Download from Apache JMeter
- Unzip and run jmeter.bat (Windows) or jmeter (Linux/Mac).

Step 2: Create a Test Plan

1. Open JMeter → Add a Test Plan.
2. Rename the Test Plan for clarity (e.g., QA_WebApp_Test).

Step 3: Add a Thread Group

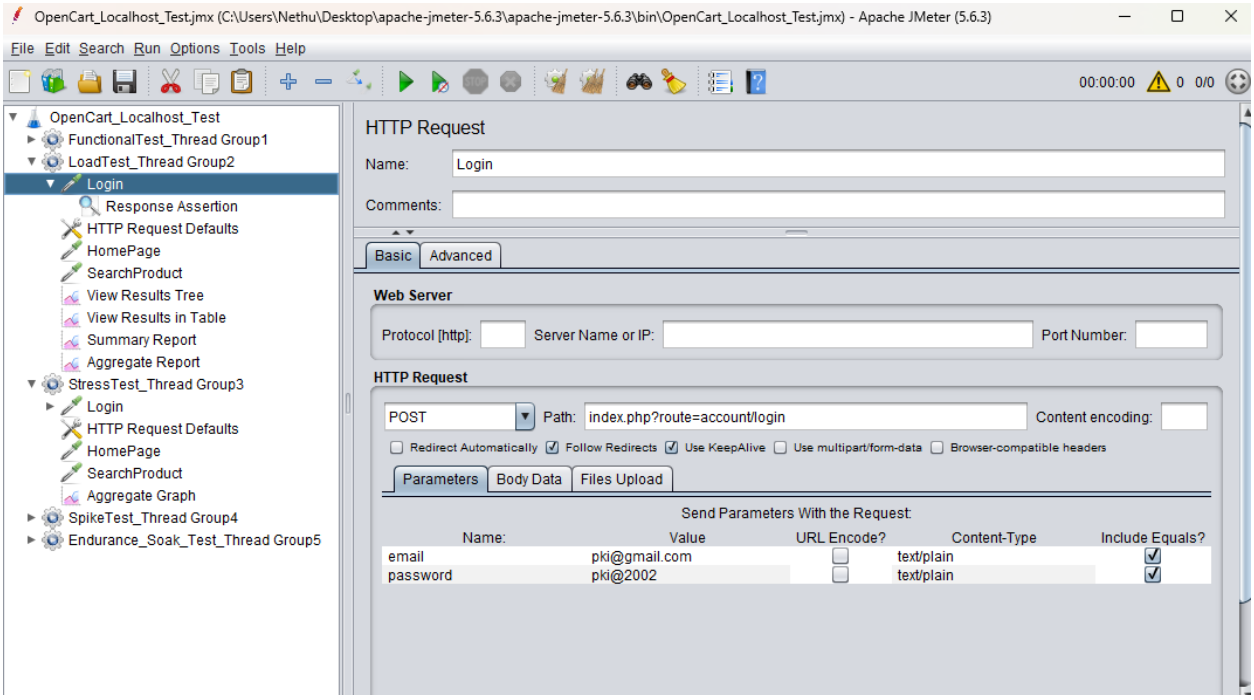
1. Right-click Test Plan → **Add → Threads → Thread Group**.
2. Configure:
    - **Number of Threads (Users)**: Number of virtual users.
    - **Ramp-Up Period**: Time to start all users.
    - **Loop Count**: Number of iterations per user.

Step 4: Add a Sampler

- Example: HTTP Request to test a website.

1. Right-click Thread Group → **Add → Sampler → HTTP Request**.
2. Enter:
    - **Server Name/IP** (e.g., example.com)
    - **Path** (e.g., /login)
    - **Method** (GET, POST, PUT, DELETE)

- **Parameters** (if POST request)



Step 5: Add Listeners

- Thread Group → Add → Listener → View Results Tree / Summary Report / Graph Results
- This will help QA analyze performance results.

Step 6: Add Assertions

- Example: Check if login returns 200 OK.
- HTTP Request → Add → Assertions → Response Assertion
- Set rules to validate server response.

Step 8: Run the Test

- Click the Start button (green triangle).
- Observe results in Listeners.

Step 9: Analyze Results

- Key metrics for QA:
    - **Response Time**: How long each request takes.
    - **Throughput**: Requests per second.

- **Error Rate**: % of failed requests.
- **Latency**: Delay between request sent and first response.

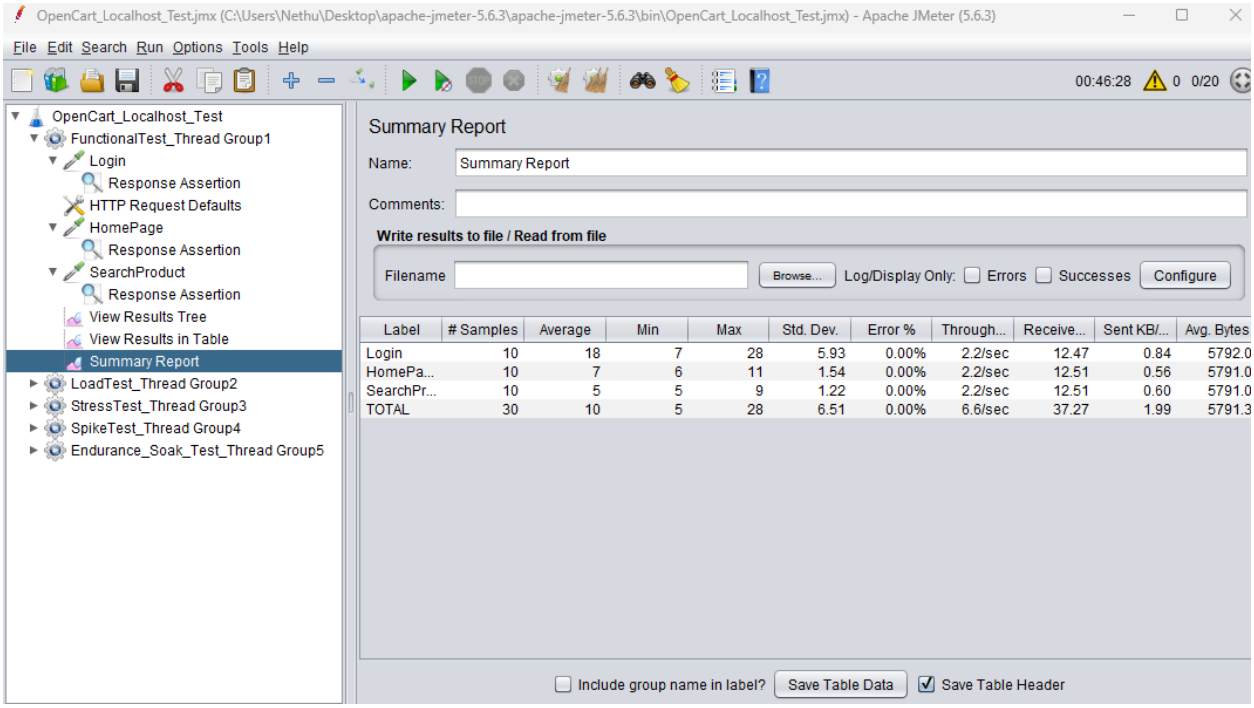
1. Functional Testing

Functional testing checks if pages work correctly.

Steps:

1. Add Thread Group → 10 users
2. Add HTTP Requests (Homepage, Login, Search)
3. Add Assertions
4. Add View Results Tree listener
5. Run test
6. Check if:
   - Response code = 200
   - Page loads correctly
   - Assertions pass

Good for verifying that your test plan works.



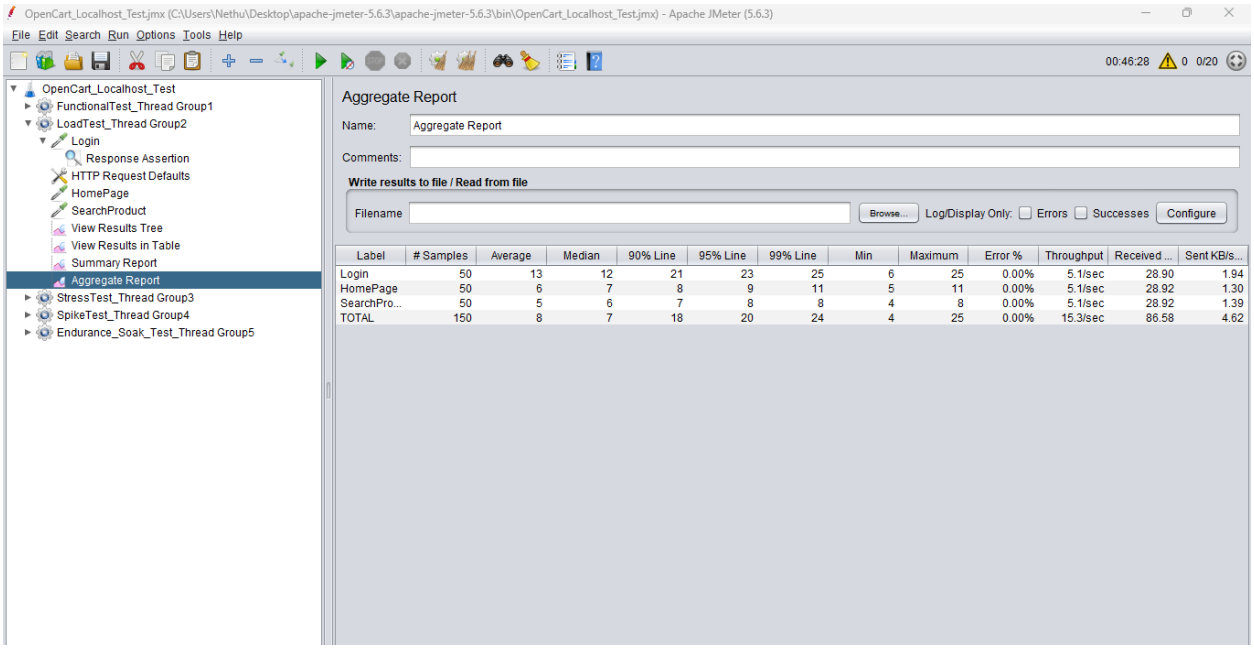
2. Load Testing

Load testing checks how your site performs under expected number of users.

Steps:

1. Thread Group:
    - Users (Threads): 50
    - Ramp-Up: 10 sec
2. Add all your important HTTP Requests
3. Add Summary Report and Aggregate Report
4. Run test
5. Monitor:
    - Average response time
    - Error rate
    - Throughput

Shows how many users your localhost can handle comfortably.



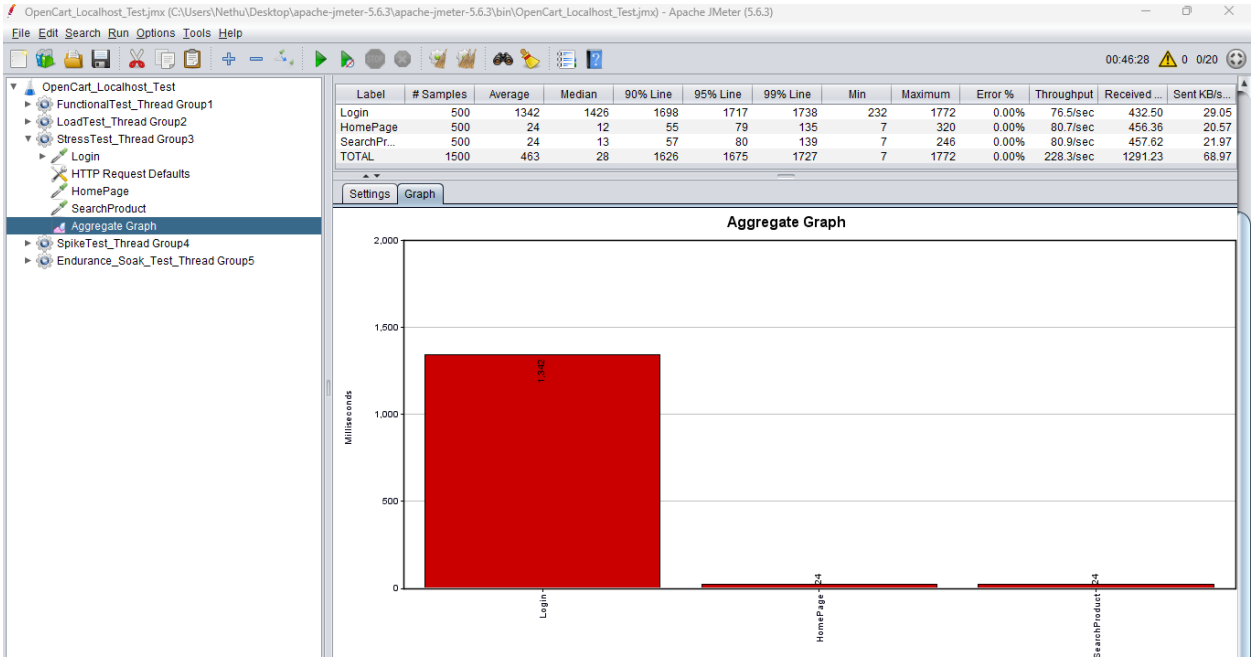
3. Stress Testing

Stress testing pushes the system beyond its limit.

Steps:

1. Thread Group:
    - Users: 100–200 (or more)
    - Ramp-Up: 5 seconds
2. Keep the same requests (Home, Login, Search, etc.)
3. Add Aggregate Report
4. Run test
5. Observe:
    - When errors start
    - When response time becomes too slow
    - When server crashes

Shows the breaking point of your local server.



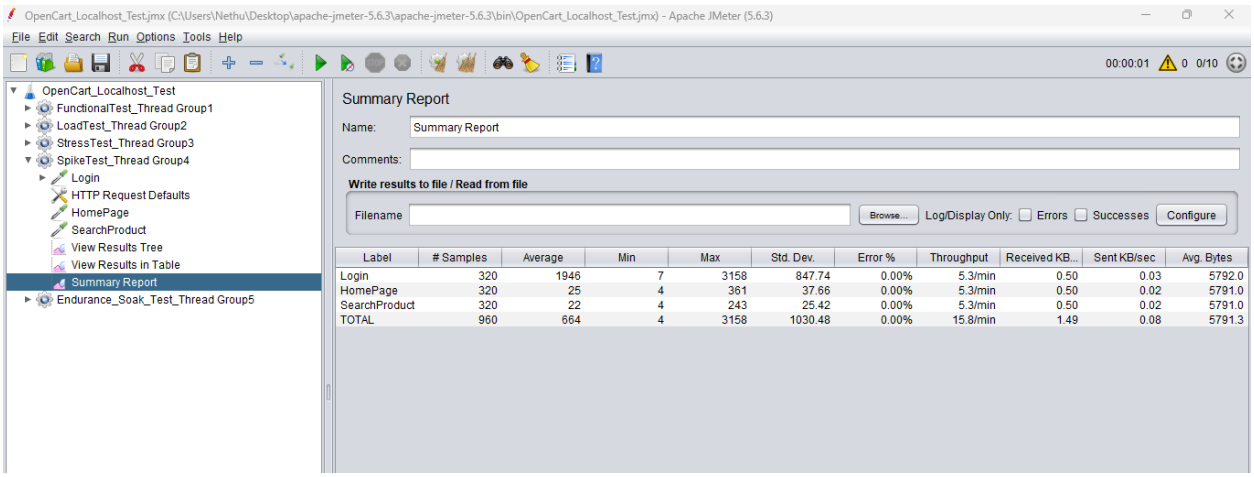
4. Spike Testing

Spike testing suddenly increases users.

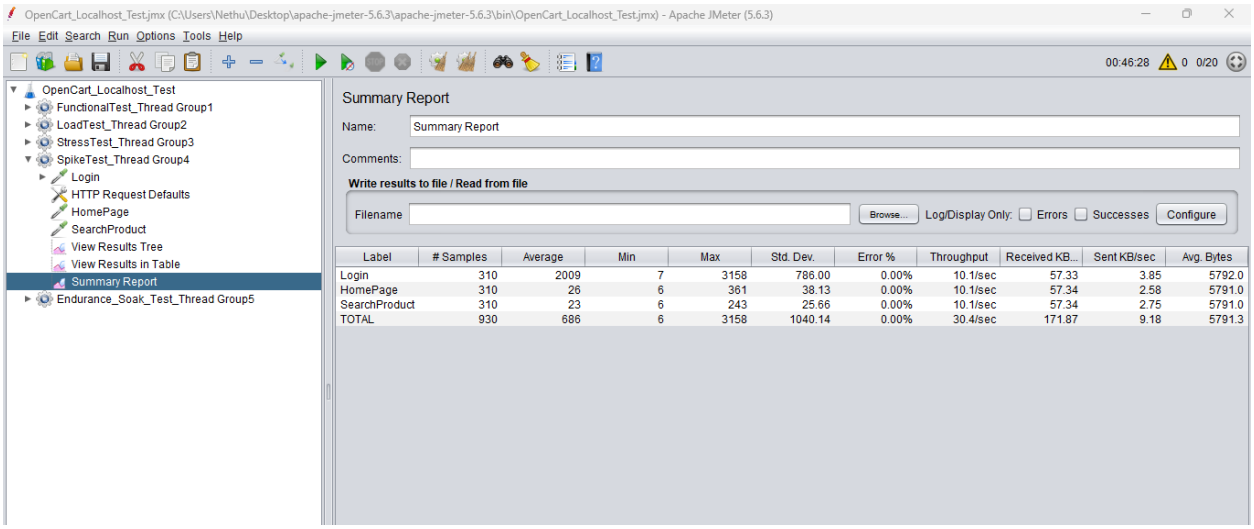
Steps:

1. Thread Group:
    - Users: 10
    - Ramp-Up: 10 sec
    - Run
2. Modify users as 300 and run
3. Compare results
4. Observe:
    - How fast the system recovers
    - How it handles sudden load

Tests how your site handles unexpected traffic spikes.



10 users


OpenCart_Localhost_Test.jmx (C:\Users\Nethu\Desktop\apache-jmeter-5.6.3\apache-jmeter-5.6.3\bin\OpenCart_Localhost_Test.jmx) - Apache JMeter (5.6.3)
File Edit Search Run Options Tools Help
00:46:28 0 0/20
OpenCart_Localhost_Test
FunctionalTest_Thread Group1
LoadTest_Thread Group2
StressTest_Thread Group3
SpikeTest_Thread Group4
Login
HTTP Request Defaults
HomePage
SearchProduct
View Results Tree
View Results in Table
Summary Report
Endurance_Soak_Test_Thread Group5
Summary Report
Name: Summary Report
Comments:
Write results to file / Read from file
Filename
Browse...
Log/Display Only: Errors Successes
Configure
Label # Samples Average Min Max Std. Dev. Error % Throughput Received KB... Sent KB/sec Avg. Bytes
Login 310 2009 7 3158 786.00 0.00% 10.1/sec 57.33 3.85 5792.0
HomePage 310 26 6 361 38.13 0.00% 10.1/sec 57.34 2.58 5791.0
SearchProduct 310 23 6 243 25.66 0.00% 10.1/sec 57.34 2.75 5791.0
TOTAL 930 686 6 3158 1040.14 0.00% 30.4/sec 171.87 9.18 5791.3

300 users

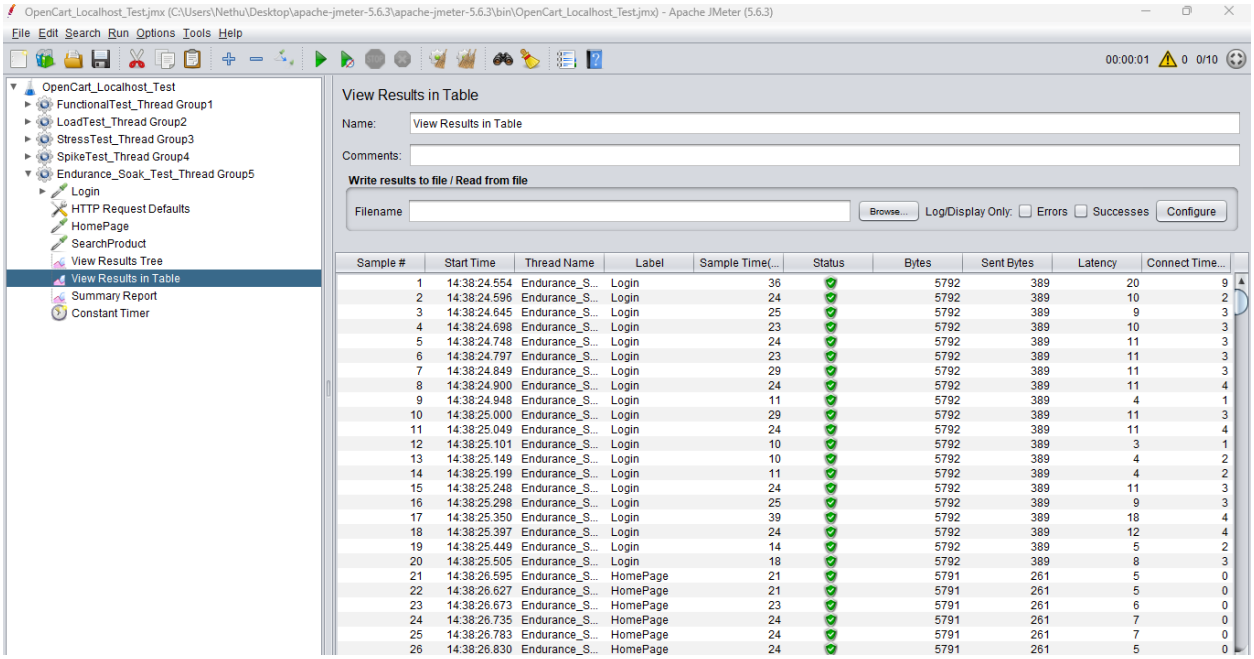
5. Endurance / Soak Testing

Long-duration tests (30 minutes – 2 hours or more).

Steps:

1. Thread Group:
    - Users: 10–20
    - Loop Count: Forever
2. Add a Constant Timer (2000 ms) to simulate real users
3. Add Summary Report
4. Run for a long duration
5. Observe:
    - Memory leak
    - CPU usage
    - Response time increase over time

Checks long-term system stability.



Performance Metrics

1. Response Time

Response Time is the total time taken from when a user sends a request to the server until the full response is received. How long the user waits to see the page or result.
In JMeter

- Measured in milliseconds (ms)
- Visible in:
    - View Results Tree
    - Summary Report
    - Aggregate Report

Slow response time = poor user experience
Response time represents the total duration taken by the server to process and return a response for a user request.

2. Latency

Latency is the delay before the server starts responding after receiving the request. How long the server takes to begin replying. Measured in milliseconds

Metric	Meaning
Latency	Time until first byte received
Response Time	Time until entire response received

Request sent at 10:00:00
Server starts responding at 10:00:00.400
Latency = 400 ms

High latency indicates:

- Network delay
- Server processing delay

Even if response time is acceptable, high latency can feel slow

3. Throughput

Throughput is the number of requests the server can handle per unit time. How many users or requests the system can handle at once.

In JMeter

- Measured as:
    - Requests per second
    - Requests per minute

If:

- 300 requests completed in 1 minute

Then: **Throughput = 300 requests/minute**

- ➢ Indicates system capacity
- ➢ High throughput = better scalability
- ➢ Very important for peak traffic (sales, offers)

4. Error Rate

Error Rate is the percentage of failed requests during testing. How many requests failed instead of succeeding. In JMeter,

Errors include:

- HTTP 4xx (client errors)
- HTTP 5xx (server errors)
- Assertion failures
- Timeouts

Example

- Total requests: 1000
- Failed requests: 50

  Error Rate = $(50/1000) \times 100 = 5\%$

- High error rate means system instability
- Indicates:
    - **Server overload**
    - **Incorrect configurations**
    - **Application bugs**

Limitations of the Testing

- Testing done on localhost only
- Network latency not considered
- Limited hardware resources

Conclusion

The performance testing of OpenCart using JMeter successfully identified system behavior under different loads. The application handled moderate load efficiently but showed performance degradation under high stress. This project enhanced practical understanding of QA performance testing.

Future Enhancements

- Cloud-based testing
- Distributed JMeter testing
- CI/CD integration
- Security & penetration testing

References

- Apache JMeter Official Documentation
- OpenCart Documentation
- Software Testing textbooks
- Online QA tutorials