Project Title:

“Performance Testing on BlazeDemo Web Application”

Website/Endpoint:

- **Website URL:** https://blazedemo.com/
- **Pages for Testing:**
  - Home Page: /
  - Flight Search Page: /reserve.php
  - Purchase Page: /purchase.php
  - Confirmation Page: /confirmation.php

Project Overview:

The purpose of this project is to perform a detailed performance test on the **BlazeDemo** web application, which is designed for testing and simulating travel booking. This test will focus on critical functions such as searching for flights, selecting flights, and completing purchases to evaluate the system's performance under varying loads.

Project Scope:

The performance testing scope includes:

1. **Load Testing:** Determine how the application performs under normal user load by simulating flight searches and purchases.
2. **Stress Testing:** Identify the application's limits by increasing the load and assessing the application’s behavior at its breaking point.
3. **Spike Testing:** Test the application's response to sudden traffic bursts, simulating a surge in users (e.g., during promotions).
4. **Endurance Testing:** Evaluate the system's performance under sustained load over a long period.
5. **Scalability Testing:** Examine how well the application scales with increasing user numbers and requests.

Project Requirements:

1. Setup and Access:

- Access the **BlazeDemo** web application at https://blazedemo.com/.
- Focus on the core workflows (flight search, selection, and purchase) as primary scenarios for testing.

2. Tools Required:

- **Apache JMeter** for creating test scripts, simulating users, and generating traffic.
- **OctoPerf** for executing load tests and managing large-scale performance testing scenarios.
- **GitHub** for storing test plans and scripts.

- **Jenkins** for CI/CD integration to automate tests and schedule performance testing tasks.

Detailed Benchmarks to Measure:

A. Response Time Metrics:

1. **Average Response Time:**
    - **Description:** Measure the average time it takes for pages to load during critical operations (e.g., flight search, purchase).
    - **Benchmark:** Page load times should be under **2 seconds** for 90% of requests.
2. **Response Time (95th Percentile):**
    - **Description:** Measure the response time for the slowest 5% of transactions.
    - **Benchmark:** Response time should be **less than 3 seconds** for 95% of the requests.

B. Throughput and Load Handling:

3. **Requests per Second (Throughput):**
    - **Description:** Measure how many successful requests the web application can handle per second.
    - **Benchmark:** BlazeDemo should handle at least **800 requests per second** under load.
4. **Concurrent Users:**
    - **Description:** Measure how many users can simultaneously perform key actions (search flights, complete purchases).
    - **Benchmark:** The application should support **200-300 concurrent users** without major degradation in performance.

C. Error Rate:

5. **Error Rate under Load:**

- **Description:** Track the percentage of failed requests under varying load conditions.
- **Benchmark:** The error rate should remain below **1%** during normal load and not exceed **3%** under stress testing.

D. Network Performance:

6. **Latency:**

- **Description:** Measure the network latency from the user's browser to the server.
- **Benchmark:** Latency should remain below **200ms** for most requests.

Test Scenarios to Execute:

1. **Scenario 1: Load Testing**
    - Simulate **200 concurrent users** performing flight searches, selecting flights, and completing purchases for 30 minutes.

2. **Scenario 2: Stress Testing**
    - Gradually increase the load to **500 users** or more to determine the breaking point where the application starts failing or exhibiting performance issues.
3. **Scenario 3: Spike Testing**
    - Introduce a sudden spike in users (e.g., 1000 users within 10 seconds) to simulate a traffic surge and assess how the application recovers.
4. **Scenario 4: Endurance Testing**
    - Simulate **100-200 users** continuously interacting with the application for 2 hours to assess the application's stability under a sustained load.

Deliverables:

1. **Performance Test Plan:**
    - A document outlining the tools, strategy, scenarios, benchmarks, and test execution timeline.
2. **Performance Testing Report:**
    - A report summarizing test results including:
        - Response times (average, 95th percentile)
        - Throughput (requests per second)
        - Error rates (percentage of failed requests)
        - Network performance (latency)
        - Bottlenecks identified
        - Recommendations for improvements
3. **JMeter Test Scripts:**
    - Submit all the test plans created in JMeter (.jmx files).

Report Submission Template:

1. **Introduction:**
    - Overview of **BlazeDemo** and the purpose of performance testing.
    - Objectives: To assess BlazeDemo's ability to handle increasing traffic and provide smooth user experiences.
2. **Test Setup:**
    - Tools: **JMeter**, **OctoPerf**, **Jenkins**, **GitHub**.
    - Test Environment: Server specs, network conditions, number of users, etc.
    - User Journeys: Flight search, selection, and purchase paths to be tested.
3. **Test Results:**
    - **Response Time Analysis:** Compare average response times, 95th percentile, and 99th percentile times.
    - **Throughput:** Requests per second under different load scenarios.
    - **Error Rates:** Percentage of failed requests under varying loads.
    - **Latency:** Network latency and its impact on performance.
4. **Conclusion:**

- Summary of the performance test results.
- Whether the application met the performance benchmarks.
- Insights into bottlenecks and performance improvement suggestions.

5. **Appendix:**
    - Attachments: JMeter .jmx files, screenshots, graphs from performance tests.
    - Detailed configuration of the tools and environment.

Additional Notes:

- **Project Duration:** 5-6 days.
    - **Days 1-2:** Set up environment and tools.
    - **Day 3:** Create JMeter scripts and scenarios for OctoPerf.
    - **Days 4-5:** Execute tests, gather data, and compile the final report.