Project Title:

“Weather Data API Performance Testing Using OpenWeatherMap”

Website/Endpoint:

- **URL:** https://openweathermap.org/
- **API Endpoint for Testing:** Please use OpenWeatherMap's public API for accessing weather data. Register for a free API key at https://home.openweathermap.org/users/sign_up.

Project Overview:

In this project, you will conduct performance testing on OpenWeatherMap's public API, which provides weather data for cities worldwide. The goal is to evaluate the API's performance under various conditions, identify potential bottlenecks, and ensure that it can handle real-world traffic scenarios.

Project Scope:

The project involves:

- **Load Testing** the API to measure performance under normal and peak traffic.
- **Stress Testing** to push the API beyond its normal operating conditions and observe how it behaves under extreme loads.
- **Endurance Testing** to evaluate how well the API performs over a sustained period of time.
- **Spike Testing** to determine how the API handles sudden traffic surges.
- **Benchmark Testing** to set performance standards that will help in real-world deployments.

Project Requirements:

1. Registration & Access:

- **Sign up** on OpenWeatherMap to obtain an API key.
- Familiarize yourself with the OpenWeatherMap API documentation.
- Use the **Current Weather Data** API for testing the performance.

2. Tools Required:

- **Apache JMeter** for load testing of API.
- **Apache JMeter** for API testing
- **GitHub** for Version Control
- **Jenkins** for Continuous Integration and Continuous Deployment

Detailed Benchmarks to Measure:

You are required to test the API based on the following 12 **detailed benchmarks**:

A. Response Time Metrics

1. **Average Response Time:**
    - Description: Measure the average time taken by the API to respond to requests across multiple cities.
    - **Benchmark:** Should be **< 500 ms** for 90% of requests.
2. **Response Time (95th Percentile):**
    - Measure the time for the slowest 5% of requests to be completed.
    - **Benchmark:** Should be **< 700 ms** for 95% of requests.
3. **Response Time (99th Percentile):**
    - Measure the response time for the slowest 1% of requests.
    - **Benchmark:** Should be **< 1 second**.

B. Throughput and Load Handling

4. **Throughput (Requests per Second):**
    - Measure how many requests the API can handle in one second.
    - **Benchmark:** Should handle **1500 requests per second** at minimum under standard conditions.
5. **Concurrent User Handling Capacity:**
    - Measure the maximum number of users that can make requests concurrently.
    - **Benchmark:** API should support at least **5000 concurrent users**.

C. Error Handling Metrics

6. **Error Rate (Normal Load):**
    - Measure the percentage of requests that fail when subjected to 1500 requests per second.
    - **Benchmark:** Error rate should be **< 2%** under normal load.
7. **Error Rate (Under Stress):**
    - Measure the percentage of failed requests when the number of users is pushed to 5000.
    - **Benchmark:** Error rate should remain **< 5%**.
8. **HTTP Status Codes Analysis:**
    - Track the frequency of HTTP status codes (200, 400, 500, etc.) under different loads.
    - **Benchmark:** At least **98%** of responses should return **HTTP 200 (OK)** status.

D. Network Performance

9. **Latency:**
    - Measure the time taken by the server to process and respond to the request (server-side latency).
    - **Benchmark:** Should be **< 300 ms** for most requests.

E. Data Consistency and Accuracy

10. **Data Consistency Under Load:**
    - Ensure the weather data returned is accurate and consistent across multiple requests.
    - **Benchmark:** Must be **100% accurate** under all load conditions.

F. Spike and Recovery Testing

11. **Spike Testing:**
    - Perform a sudden spike in the number of requests (5000 requests within 5 seconds) and measure system recovery.
    - **Benchmark: System should recover within 60 seconds after the spike.**
12. **Time to Recover After Spike:**
    - Measure how quickly the system stabilizes after a sudden traffic spike.
    - **Benchmark:** System should return to normal within **1 minute**.

Test Scenarios to be Executed:

- **Scenario 1:** Load Testing with 1500 requests per second for 30 minutes.
- **Scenario 2:** Stress Testing with 5000 concurrent users.
- **Scenario 3:** Spike Testing with sudden traffic surges (4000 requests within 5 seconds).
- **Scenario 4:** Endurance Testing with 1500 requests per second for 2 hours.

P.S. Run all .JMX file from Jenkins, configured with Git.

Deliverables:

1. **Performance Test Plan:**
    - Clearly outline the tools used, testing strategy, benchmarks to be measured, and test execution timeline.
2. **Performance Testing Report:**
    - Include the following sections:
        - Summary of all tests performed.
        - Graphs showing response times, throughput, CPU, and memory usage.
        - Errors with details.
        - Bottleneck identification and improvement suggestions.
        - Recommendations for further optimization.
3. **JMeter Test Scripts:**
    - Submit the JMeter .jmx file with all configured tests.

    P.S. API Key**:** Use your registered OpenWeatherMap API key in all requests. Do **not** share it in the final report.

Report Submission Template:

1. **Introduction:**
    - Overview of the OpenWeatherMap API and the project's objectives.
    - Brief description of the performance testing goals (e.g., identifying bottlenecks, evaluating data consistency, etc.).
    - Summary of test types conducted (load, stress, spike, endurance).
2. **Test Setup:**
    - Tools used for testing (e.g., Apache JMeter, Git, Jenkins).
    - Detailed environment setup, including:
        - Hardware configuration (CPU, RAM, disk space).
        - Network configuration and geolocation of test servers (if applicable).
        - API key setup and any specific configurations made in the API requests (e.g., city, unit parameters).
    - Testing duration, virtual users, and concurrency levels.
    - Description of API endpoints being tested (e.g., /data/2.5/weather for Current Weather Data).
    - Any third-party monitoring tools (e.g., Grafana, New Relic) integrated for analysis.
3. **Test Results:**
    - **Response Time Analysis:**
        - Graphs showing **average response time**, **95th percentile response time**, and **99th percentile response time** across different loads.
        - Analysis of response time trends over different user loads and request volumes.
    - **Throughput:**
        - Graphs showing throughput (requests per second) for each test scenario (normal load, stress, spike, etc.).
        - Comparison of throughput vs. user concurrency levels.
    - **Error Rates:**
        - Graph showing the error rate under different test scenarios.
        - Summary of HTTP status codes (200, 400, 500, etc.) encountered during tests.
        - List and analysis of any errors returned by the API, with possible explanations (e.g., request timeouts, server overload).
    - **Network Latency:**
        - Latency comparison charts showing server response times.
        - Summary of how network conditions affected performance.
    - **Benchmark Comparison:**
        - Table comparing actual test results against the predefined benchmarks for each scenario.

        - Mark whether the API passed or failed for each benchmark (e.g., response time < 500ms, etc.,).
4. **Conclusion:**
    - Summary of key findings:
        - Whether the API met or exceeded performance benchmarks.
        - Insights into bottlenecks or areas where performance degraded (e.g., under stress or spike conditions).
        - Identified API inefficiencies, such as slow response times under specific conditions.
    - Recommendations for improvement:
        - Possible areas for horizontal or vertical scaling to handle higher concurrency.
        - Recommendations for optimizing code to handle high-throughput or reduce latency.
5. **Appendix:**
    - Attach JMeter .jmx test scripts for all the test scenarios.
    - Screenshots of test results or key moments (e.g., spike test results, API errors).
    - List of API endpoints and their descriptions that were tested.
    - Include a list of tools and plugins used (e.g., JMeter plugins for monitoring or advanced reporting).

Additional Notes:

- **Project Duration:** The project is designed to be completed in **7 days**.
    - **Days 1-2:** Set up environment, tools, and API keys.
    - **Day 3:** Familiarize yourself with the API and develop JMeter scripts.
    - **Day 4-5:** Execute performance tests for different scenarios (load, stress, endurance, spike).
    - **Day 6:** Analyze results and generate reports.
    - **Day 7:** Submit the final report with detailed analysis.
- **Performance Testing Tools:** Apache JMeter.
- **API Quota Limit:** Keep in mind the free-tier API rate limits provided by OpenWeatherMap. Design your test plan in a way that optimizes API usage without exceeding the quota (e.g., schedule tests to minimize API key exhaustion).
- **Important Reminders:**
    - Ensure that you apply proper **parameterization** in your test scripts for dynamic data such as city IDs or query parameters and **correlation** if necessary.