Project Title:

“Performance Testing on WebTours Application”

Website/Endpoint:

- **Website URL:** WebTours is a sample travel booking application, typically available on **localhost.**
- **Key Pages/Modules for Testing:**
    - Home Page: /WebTours/home.html
    - Login Page: /cgi-bin/login.pl
    - Flights Search: /cgi-bin/reservations.pl
    - Flight Booking: /cgi-bin/reservations.pl
    - Itinerary Review: /cgi-bin/itinerary.pl
    - Sign Off: /cgi-bin/welcome.pl?signOff=true

Project Overview:

This project aims to test the performance of the **WebTours** application—a web-based travel booking system. The primary goal is to evaluate the system’s ability to handle different user loads, perform travel searches, book flights, and manage itineraries. The performance test will cover all key actions of a typical user journey within the application, ensuring stability and responsiveness under various traffic conditions.

Project Scope:

The performance testing of WebTours includes:

1. **Load Testing:** To simulate typical user load scenarios such as login, searching for flights, booking tickets, and viewing itineraries.
2. **Stress Testing:** To identify the application's breaking point by pushing it beyond its designed load capacity.
3. **Spike Testing:** To observe how the system handles sudden surges in traffic and recovers afterward.
4. **Endurance Testing:** To evaluate performance over an extended time period to detect memory leaks and performance degradation.
5. **Volume Testing:** To test the application’s performance with a large volume of booking data.

Project Requirements:

1. Setup and Access:

- Host the WebTours application on **localhost**.
- Focus on key modules:
    - **Login/Logout Workflow**
    - **Flight Search and Booking**
    - **Itinerary Management**
    - **Flight Cancellation**
    - **User Sign Off**

2. Tools Required:

- **Apache JMeter** for creating and executing the performance test scripts.
- **GitHub** for version control of the test scripts and test plan.
- **Jenkins** for continuous integration and automating the performance tests.

Detailed Benchmarks to Measure:

A. Response Time Metrics:

1. **Average Response Time:**
    - **Description:** Measure the average time taken for pages to load such as search results, booking confirmation, and itinerary review.
    - **Benchmark: Less than 2.5 seconds** for 90% of requests.
2. **95th Percentile Response Time:**
    - **Description:** Response time for the slowest 5% of requests.
    - **Benchmark:** Should be **under 4 seconds** for 95% of requests.

B. Throughput and Load Handling:

3. **Requests per Second (Throughput):**
    - **Description:** Measure how many requests can be handled by the system per second.
    - **Benchmark:** The system should handle **500-1,000 requests per second** under normal load.
4. **Concurrent Users:**
    - **Description:** Number of simultaneous users who can perform key actions like searching for flights and booking.
    - **Benchmark:** The application should handle **200-500 concurrent users** for core actions.

C. Error Rate:

5. **Error Rate under Load:**
    - **Description:** Measure the percentage of failed requests as the user load increases.
    - **Benchmark: Less than 2%** error rate at normal load, and **below 5%** during stress testing.

D. Network Performance:

6. **Latency:**
    - **Description:** Measure the time it takes for a user's request to reach the server and return the response.
    - **Benchmark: Less than 200ms** latency under normal conditions.

Test Scenarios to Execute:

1. **Scenario 1: Load Testing (Simulating Normal User Traffic)**

- Simulate **200 concurrent users** logging in, searching for flights, booking tickets, and viewing itineraries over a **1-hour period**.
- Measure response time, throughput, and error rates for these typical user journeys.

2. **Scenario 2: Stress Testing (Pushing System Limits)**
    - Gradually increase the number of concurrent users from **200 to 1,000** to determine when the system becomes unstable.
    - Capture the point at which performance begins to degrade (increased error rates and slower response times).
3. **Scenario 3: Spike Testing (Handling Traffic Spikes)**
    - Simulate a sudden burst of **500 users** within **1 minute**, performing key tasks like searching for flights and booking.
    - Measure how well the system handles the spike and its recovery after the burst.
4. **Scenario 4: Endurance Testing (Long-duration Testing)**
    - Simulate **100 users** booking flights, updating itineraries, and signing off over a **6-hour period**.
    - Check for resource exhaustion, or performance degradation over time.
5. **Scenario 5: Volume Testing (Handling Large Data Volumes)**
    - Test the system with **50,000+ booking records**, searching and updating multiple itineraries at the same time.
    - Measure how well the system handles bulk data and booking transactions.
6. **Scenario 6: Itinerary Management Load Testing**
    - Simulate **300 users** simultaneously accessing their itineraries, modifying bookings, and cancelling flights.
    - Measure response times and errors while managing a large volume of booking data.
7. **Scenario 7: Flight Booking Stress Testing**
    - Simulate **500 users** concurrently searching and booking flights.
    - Measure the application's ability to process booking transactions and the point at which failures occur.
8. **Scenario 8: Sign Off Stress Testing**
    - Simulate **100 users** logging out/signing off from the system simultaneously after completing their transactions.
    - Test how the system performs under simultaneous session terminations.

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

Deliverables:

1. **Performance Test Plan:**
    - Detailed strategy, tools, and test scenarios for load, stress, spike, endurance, and volume testing.
2. **Performance Testing Report:**

- Summary of test results including:
    - Average and 95th percentile response times.
    - Throughput (requests per second).
    - Error rates (failed requests).
    - Latency and network performance.
    - Performance bottlenecks.
    - Recommendations for performance improvements.

3. **JMeter Test Scripts:**
    - All the JMeter scripts used for the performance testing (.jmx files).
4. **Graphical Analysis:**
    - Charts and graphs showing trends in response times, throughput, error rates, and system performance under load.

Report Submission Template:

1. **Introduction:**
    - Overview of the WebTours application and the importance of conducting performance testing.
    - Objectives: Ensure WebTours can handle high user traffic while maintaining performance.
2. **Test Setup:**
    - Tools: **JMeter**, **Jenkins**, etc.
    - Environment: Detailed description of the server, virtual user setup, and testing configurations.
3. **Test Results:**
    - **Response Times:** Average, 95th percentile, and maximum response times for critical user journeys.
    - **Throughput:** Requests per second across different scenarios.
    - **Error Rates:** The percentage of failed requests at various load levels.
    - **System Limits:** The maximum load the application could handle before performance degraded.
    - **Latency:** The network latency recorded during the test.
4. **Conclusion:**
    - Summary of performance against benchmarks.
    - Insights into how the system performs under different load scenarios.
    - Recommendations for system optimization and potential improvements.
5. **Appendix:**
    - Attachments: JMeter scripts, logs, and graphical reports of performance metrics.

Additional Notes:

- **Project Duration:** 7-10 days.
    - **Days 1-2:** Set up tools and environment.

- **Days 3-4:** Create and validate JMeter scripts.
- **Days 5-7:** Execute test scenarios, capture results.
- **Day 8:** Analyze results and prepare performance reports.