**Performance Test Plan for PhotoApp Website**

**Test Plan Version**: 1.0  
**Date**: March 14, 2025  
**Tester**: Dominic Rutto  
**Project**: PhotoApp Website  
**Environment**: Production (or appropriate environment)

**1. Introduction**

The purpose of this Performance Test Plan is to outline the strategy for evaluating the performance of the PhotoApp website. This plan will focus on conducting Load Testing, Stress Testing, and Response Time Testing to verify the website's ability to handle varying levels of traffic and ensure that it meets performance expectations.

**2. Test Objectives**

* **Load Testing**: Determine the maximum number of users the website can handle under normal conditions without significant performance degradation.
* **Stress Testing**: Identify how the website behaves under extreme conditions such as traffic spikes or high concurrent user activity.
* **Response Time Testing**: Measure how quickly the website responds to user actions (e.g., page load times, API responses).

**3. Test Scope**

* **Functional Areas to be Tested**:
  + Homepage
  + Album Listing Page
  + Photo Viewing Page
  + User Profile Pages
  + Search Functionality
  + Login and Logout processes
* **Exclusions**:
  + Non-performance related tests (e.g., functional, security, or usability tests).

**4. Test Approach**

1. **Load Testing**:
   * **Objective**: To determine the performance of the PhotoApp website under a normal, expected load (e.g., 100, 500, 1,000 users).
   * **Tools**: Apache JMeter, Artillery, LoadRunner, or similar load testing tools.
   * **Test Method**:
     + Simulate users accessing the website simultaneously.
     + Gradually increase the load to simulate expected traffic levels.
     + Monitor and record system performance, including server response time, throughput, and resource utilization.
2. **Stress Testing**:
   * **Objective**: To test how the website behaves under extreme conditions (e.g., more than the maximum expected load).
   * **Tools**: Apache JMeter, Artillery, or similar tools.
   * **Test Method**:
     + Gradually increase the load beyond the normal expected number of users.
     + Monitor the server for signs of failure, including slow response times, crashes, or resource exhaustion.
     + Identify the breaking point of the system.
3. **Response Time Testing**:
   * **Objective**: To measure how quickly the website responds to user actions (page loads, API responses).
   * **Tools**: Apache JMeter, Artillery, Browser Developer Tools (for frontend).
   * **Test Method**:
     + Measure response times for important user actions like page loads, album selection, photo browsing, etc.
     + Conduct tests with a variety of user load levels to understand performance at both normal and peak conditions.

**5. Performance Metrics**

The following metrics will be recorded during testing:

* **Response Time**: The time taken for the server to respond to a request (in milliseconds).
* **Throughput**: The number of requests the server can handle per second (requests/second).
* **Error Rate**: The percentage of failed requests during testing.
* **Resource Utilization**: CPU, memory, disk, and network utilization of the server.
* **Concurrency Level**: The number of users interacting with the website simultaneously.

**6. Test Environment**

* **Hardware Configuration**:
  + Web Server: Apache or Nginx
  + Database: MySQL or PostgreSQL
  + OS: Linux or Windows
  + Server Specifications: [Insert hardware specs]
* **Network Configuration**:
  + Test environment should simulate a production-like network with reasonable latency and bandwidth limitations.

**7. Success Criteria**

* **Load Testing**: The website should be able to handle a load of 1,000 concurrent users without performance degradation beyond an acceptable threshold (e.g., page load time > 2 seconds).
* **Stress Testing**: The system should gracefully degrade under high load, either by slowing down or showing meaningful error messages. The system should recover once the load is reduced.
* **Response Time Testing**: The website should maintain a response time of under 2 seconds for 95% of all requests, even with moderate traffic.

**8. Tools for Performance Testing**

* **JMeter**: For simulating user activity and measuring load and stress.
* **Artillery**: An alternative lightweight tool for load testing, especially useful for testing APIs.
* **Browser Developer Tools**: For frontend response time analysis.
* **Datadog/New Relic**: For real-time monitoring of server resource usage during tests.

**9. Test Schedule**

* **Preparation Phase**:
  + Setup and configuration of the test environment
  + Creation of test data
* **Execution Phase**:
  + **Load Testing**
  + **Stress Testing**
  + **Response Time Testing**
* **Reporting Phase**:
  + **Analysis and reporting**

**10. Risk Management and Mitigation**

* **Risk**: The test environment may not reflect the production environment.
  + **Mitigation**: Ensure the test environment closely matches the production environment in terms of configuration, hardware, and network conditions.
* **Risk**: Data inconsistency during testing.
  + **Mitigation**: Use standardized test data, and verify that the test data is consistently applied throughout all test cases.