**PERFORMANCE TESTING STRATEGY DOCUMENT**

**Performance Testing Strategy Document**

**Project Name:** Performance Testing for Simplita.ai  
**Prepared By:** Lakshman. M(QA)  
**Date:** 17/03/2025  
**Version:** 2.0

**1. TEST OBJECTIVES & SCOPE**

**1.1 Objectives:**

* Validate system performance under peak, normal, and stress conditions.
* Identify scalability limits and optimize infrastructure accordingly.
* Ensure system reliability and responsiveness for end-users.
* Detect potential bottlenecks in application performance.

**1.2 Scope:**

* Load test core functionalities including authentication, Engine’s Module navigation, data processing, and API responses.
* Assess concurrent user capacity and request handling efficiency.

**2. TEST ENVIRONMENT & INFRASTRUCTURE**

**2.1 Test Environment Details:**

* **Application URL:** <https://app.simplita.ai/>
* **Environment Type:** Testing Server
* **Web Server:** Apache
* **Database:** Supabase
* **Application Technology:** Node.js / Python FastAPI
* **Cloud Provider:** AWS

**2.2 Load Testing Tools:**

* Apache JMeter

**3. TEST SCENARIOS & WORKLOAD DESIGN**

**Iteration 1: Basic Workload:**

| **Scenario ID** | **Test Scenario** | **Target Load** | **Target Time** | **Loop Count** | **Expected Response Time** |
| --- | --- | --- | --- | --- | --- |
| TC-001 | User Login (Valid Credentials) | 500 concurrent users | 2 sec | 10 | < 2 sec |
| TC-002 | User Login (Invalid Credentials) | 500 concurrent users | 2 sec | 10 | < 2 sec |
| TC-003 | Navigate to existing Workspace | 1000 concurrent users | 3 sec | 10 | < 3 sec |
| TC-004 | Load any Engine module | 1000 concurrent users | 3 sec | 10 | < 3 sec |
| TC-005 | Execute Core Functionality (e.g., API Call) | 2000 concurrent users | 3 sec | 20 | < 3 sec |
| TC-006 | Log Out | 500 concurrent users | 2 sec | 10 | < 2 sec |

**Iteration 2: Middle level Workload:**

| **Scenario ID** | **Test Scenario** | **Target Load** | **Target Time** | **Loop Count** | **Expected Response Time** |
| --- | --- | --- | --- | --- | --- |
| TC-001 | User Login (Valid Credentials) | 1000 concurrent users | 2 sec | 10 | < 2 sec |
| TC-002 | User Login (Invalid Credentials) | 1000 concurrent users | 2 sec | 10 | < 2 sec |
| TC-003 | Navigate to existing Workspace | 2000 concurrent users | 3 sec | 10 | < 3 sec |
| TC-004 | Load any Engine module | 2000 concurrent users | 3 sec | 10 | < 3 sec |
| TC-005 | Execute Core API Call | 3000 concurrent users | 3 sec | 15 | < 3 sec |
| TC-006 | Log Out | 1000 concurrent users | 2 sec | 10 | < 2 sec |
| TC-007 | Data Fetch from Backend | 5000 concurrent users | 3 sec | 15 | < 3 sec |
| TC-008 | Multi-User Transactions | 5000 concurrent users | 5 sec | 20 | < 5 sec |
| TC-009 | File Upload (5MB) | 1000 concurrent users | 4 sec | 10 | < 4 sec |
| TC-010 | File Download (10MB) | 1000 concurrent users | 4 sec | 10 | < 4 sec |

**Iteration 3: Advanced Workload:**

| **Scenario ID** | **Test Scenario** | **Target Load** | **Target Time** | **Loop Count** | **Expected Response Time** |
| --- | --- | --- | --- | --- | --- |
| TC-001 | User Login (Valid) | 2000 concurrent users | 2 sec | 10 | < 2 sec |
| TC-002 | User Login (Invalid) | 2000 concurrent users | 2 sec | 10 | < 2 sec |
| TC-003 | Dashboard Navigation | 5000 concurrent users | 3 sec | 10 | < 3 sec |
| TC-004 | Data Fetch from Backend | 10,000 concurrent users | 3 sec | 10 | < 3 sec |
| TC-005 | API Call Performance | 20,000 requests/sec | 3 sec | 10 | < 3 sec |
| TC-006 | File Upload (10MB) | 3000 concurrent users | 5 sec | 5 | < 5 sec |
| TC-007 | File Download (20MB) | 3000 concurrent users | 5 sec | 5 | < 5 sec |
| TC-008 | Stress Test - Peak Load | 100,000 concurrent users | 5 sec | 5 | < 5 sec |
| TC-009 | Scalability Test (Auto Scaling) | 200,000 virtual users | 4 sec | 5 | < 4 sec |
| TC-010 | Database Query Performance | 1000 concurrent queries | 2 sec | 10 | < 2 sec |
| TC-011 | Multi-User Transactions | 50,000 concurrent users | 5 sec | 10 | < 5 sec |
| TC-012 | API Concurrent Request Handling | 25,000 requests/sec | 3 sec | 10 | < 3 sec |

**4. TEST EXECUTION STRATEGY**

**4.1 Load Test Phases:**

* **Baseline Performance Testing:** Establish performance benchmarks.
* **Load Testing:** Simulate concurrent user activity.
* **Stress Testing:** Test system behaviour under extreme loads.
* **Endurance Testing:** Run long-duration tests to assess stability.
* **Scalability Testing:** Evaluate dynamic scaling capabilities.
* **Spike Testing:** Simulate sudden traffic spikes.

**4.2 Load Distribution Plan:**

* **Ramp-Up Strategy:** Stat with 10 users, then increase by 50 users every 10 seconds until the target load is reached.
* **Steady-State Load:** Maintain peak user load for 60 minutes.
* **Ramp-Down Strategy:** Gradually reduce load over 10 minutes to measure recovery time.
* **Concurrency Levels for Itr-1: 100, 500, 1000, 2000 users**
* **Concurrency Levels for Itr-2:** 100, 500, 1000, 5000, 10,000, 100,000 users.
* **Think Time Between Requests:** 3-5 seconds.

**5. PERFORMANCE METRICS & BENCHMARKS**

| **Metric** | **Acceptable Threshold** |
| --- | --- |
| Average Response Time | < 3 sec |
| Peak Response Time | < 5 sec |
| Error Rate | < 1% |
| Throughput | Minimum 200 TPS |
| CPU Utilization | < 75% |
| Memory Utilization | < 80% |
| Database Query Response Time | < 2 sec |
| API Call Failure Rate | < 0.5% |

**6. MONITORING & REPORTING**

**6.1 Key Metrics to Monitor:**

* **Server Health:** CPU, Memory, Disk I/O, Network Traffic.
* **Application Performance:** Response time, request rate, error rate.
* **Database Performance:** Query response time, connection limits.
* **API Health:** Success rate, failure trends, latency analysis.

**6.2 Reporting Format:**

* **Performance Summary Report**
* **Charts & Graphs for Bottleneck Analysis**
* **Scalability & Capacity Planning Recommendations**

**7. RISKS & MITIGATION STRATEGY**

| **Risk** | **Mitigation Strategy** |
| --- | --- |
| High Response Time | Optimize DB queries, implement caching |
| Server Overload | Use load balancers, auto-scaling configurations |
| High API Error Rate | Improve exception handling, implement retries |
| Database Deadlocks | Optimize indexing, use read replicas |
| Unexpected Traffic Spikes | Implement auto-scaling policies |

**8. CONCLUSION & RECOMMENDATIONS**

* Ensure sustained application performance under peak and normal load conditions.
* Optimize API performance and minimize latency through caching and code optimizations.
* Implement scalable cloud infrastructure to handle traffic surges.
* Regular performance testing is recommended to pre-empt potential degradation.

**Approval Signatures:**

* **Performance Test Lead:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* **Project Manager:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_