**Load Testing Test Case Document**  
**Project:** Performance Testing for Simplita.ai  
**Prepared By:** Lakshman  
**Date:** 10/03/2025  
**Version:** 1.0

**1. Test Overview**

* **Objective:** To evaluate the performance, stability, and scalability of https://app.simplita.ai/ under different load conditions.
* **Scope:** Load test on the web application’s core functionalities, including authentication, dashboard access, and primary workflows.
* **Assumptions:**
  + The application is fully functional and accessible.
  + Test environment closely resembles the production environment.
  + Adequate monitoring tools are in place to track performance metrics.

**2. Test Environment**

* **Application URL:** https://app.simplita.ai/
* **Test Environment:** Pre-Production / Staging
* **Hardware Configuration:**
  + CPU: 8-Core Processor
  + RAM: 32GB
  + Disk: 500GB SSD
  + Network: 1Gbps bandwidth
* **Software Configuration:**
  + OS: Windows
  + Web Server: Apache
  + Database: Supabase
  + Load Testing Tool: JMeter

**3. Performance Test Scenarios**

**3.1. User Workflows for Load Testing**

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

**4. Test Execution Plan**

* **Ramp-Up Strategy:**
  + Start with 10 users, then increase by 50 users every 10 seconds until the target load is reached.
* **Steady State:**
  + Maintain peak load for 30 minutes to evaluate system stability.
* **Ramp-Down Strategy:**
  + Gradually decrease users over 5 minutes to measure recovery performance.
* **Test Duration:** Approx. 3 hour
* **Concurrency Levels:** 100, 500, 1000, 2000 users
* **Think Time:** 3-5 seconds per request

**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 |

**6. Monitoring & Reporting**

* **Tools Used:** Grafana, New Relic, Datadog, Prometheus
* **Monitored Parameters:**
  + CPU & Memory usage
  + Disk I/O & Network Bandwidth
  + Error Logs & Latency Reports
  + Database Query Performance
* **Reporting Format:**
  + Summary Report with charts & graphs
  + Detailed breakdown of each scenario
  + Recommendations for optimization

**7. Risks & Mitigation Plan**

| **Risk** | **Mitigation Strategy** |
| --- | --- |
| High Response Time | Optimize database queries, implement caching |
| Server Crashes | Increase server capacity, implement load balancing |
| High Error Rate | Debug logs, handle exceptions efficiently |
| Network Bottleneck | Upgrade network bandwidth, optimize requests |

**8. Conclusion & Recommendations**

* Ensure application can handle peak user load without significant degradation.
* Optimize database queries & API response times.
* Implement caching strategies to improve performance.
* Scale infrastructure dynamically based on traffic.

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