Performance Test Plan

Source File: generated-documents\quality-assurance\performance-test-plan.md
Generated: 30/07/2025 at 07:00:20
Generated by: Requirements Gathering Agent - PDF Converter

Performance Test Plan

Generated by adpa-enterprise-framework-automation v3.2.0
Category: quality-assurance
Generated: 2025-07-14T21:06:16.083Z
Description: Performance testing strategy and test plan

Performance Test Plan

Project: ADPA - Advanced Document Processing & Automation Framework
Version: 3.2.0
Date: July 2025
Prepared by: Performance Test & QA Engineering Team

---

1. Performance Test Overview

1.1 Objectives and Goals

- **Validate** that the ADPA framework (API, CLI, and admin web interface) meets enterprise-grade performance, scalability, and reliability standards under realistic and peak workloads.
- **Identify** system bottlenecks, breaking points, and degradation risks in API, document generation, AI provider orchestration, and integration flows (Confluence, SharePoint, Adobe).
- **Ensure** production readiness for high-concurrency, large-volume, and long-duration enterprise use cases as required by standards-compliant, Fortune 500 deployments.

1.2 Success Criteria & Benchmarks

- **Response Time:** < 2s for 95% of API requests under normal load; < 5s under peak load.
- **Throughput:** Support 250 concurrent users (API/CLI), > 1,500 document generation jobs/hour.
- **Resource Utilization:** CPU < 70%, RAM < 75%, Disk I/O < 60% saturation on primary nodes.
- **Scalability:** Linear throughput increase with additional API instances.
- **Availability:** 99.9% uptime during continuous 72-hour endurance testing.

1.3 Testing Scope & Limitations

- **In Scope:** REST API server (Express.js), CLI workflows, admin Next.js interface, AI orchestration, integration connectors, and document generation pipeline.
- **Out of Scope:** Non-production integrations (e.g., incomplete DMBOK features), legacy CLI, and experimental plugins.
- **Limitations:** Does not account for external failures (e.g., upstream AI provider outages), third-party API SLAs, or physical infrastructure failures outside the test environment.

1.4 Performance Risk Assessment

- **High:** AI provider latency, document template complexity, third-party integration limits.
- **Medium:** API authentication overhead, concurrent document creation, database/config file IO.
- **Low:** CLI command performance, admin portal under non-peak usage.

---

2. Performance Requirements

2.1 Response Time Requirements

Operation	Max Acceptable (Normal)	Max Acceptable (Peak)
Health Check (`/health`)	200ms	500ms
Document Generation (API/CLI)	2s (95th percentile)	5s (95th percentile)
Template Listing	500ms	1s
Integration Calls (Confluence etc)	3s	7s
Admin UI Page Load	1s	2s

2.2 Throughput Requirements

- **API:** ≥ 50 requests/sec sustained; burst to 200 requests/sec for 15 minutes.
- **Concurrent Users:** ≥ 250 users (API + admin UI + CLI combined).
- **Document Jobs:** ≥ 1,500 generated/hour with standard templates.

2.3 Resource Utilization Limits

- **CPU:** < 70% average per node (Node.js + dependent services)
- **Memory:** < 75% utilization during sustained load
- **Disk:** < 60% I/O utilization during peak document generation
- **Network:** < 80% of available bandwidth for API/data transfer

2.4 Scalability Targets

- **Horizontal Scaling:** Linear throughput scaling up to 5 API nodes
- **Scaling Lag:** < 60s to add/remove API instances (containerized deployments)
- **AI Providers:** Automatic failover and load balancing across at least 2 providers

2.5 Availability Requirements

- **Uptime:** ≥ 99.9% over a 30-day rolling window
- **Recovery:** < 2 minutes recovery time from single-node (API or worker) failure

---

3. Performance Test Types and Approach

Test Type	Objective	Approach
Load Testing	Validate performance under expected/typical usage	Simulate realistic user/API load; profile response time
Stress Testing	Identify system breaking points and graceful degradation	Incremental load increase past capacity

Test Type	Objective	Approach
Volume Testing	Validate handling of large document/data sets	Bulk upload/generation with large templates/files
Spike Testing	Test resilience to sudden traffic surges	Abruptly ramp up users/API calls
Endurance Testing	Ensure stability over extended periods	72-hour continuous load; monitor for leaks/degradation
Capacity Testing	Determine maximum sustainable throughput/user count	Incrementally increase concurrent jobs/users

---

4. Test Environment and Infrastructure

4.1 Environment Specifications

- **Environment:** Dedicated performance test cluster, isolated from production
- **API Servers:** 3x Node.js 18+ VMs/containers (8 vCPU, 16GB RAM each)
- **Admin UI:** 1x Next.js 14 instance (4 vCPU, 8GB RAM)
- **Load Generators:** 2x VMs (8 vCPU, 16GB RAM), distributed
- **Datastore:** Local JSON config, with simulated SQL backend for volume testing

4.2 Hardware/Software Requirements

- **Node.js:** v18.0.0+
- **TypeScript:** v5.7.2+

- **Redis (optional):** For caching and session management
- **Docker/Kubernetes:** For scalability and orchestration tests

4.3 Network Configuration

- **Internal VLAN:** 1Gbps+ between nodes
- **API Gateway/Load Balancer:** Simulate production routing

4.4 Test Data Requirements

- **Document Templates:** 10+ complex templates (BABOK, PMBOK, DMBOK)
- **Sample Documents:** 100k+ simulated business documents
- **AI Provider Credentials:** Valid API keys for all supported AI services
- **Integration Accounts:** Staging Confluence, SharePoint, and Adobe credentials

4.5 Monitoring & Instrumentation

- **APM:** Datadog, New Relic, or OpenTelemetry
- **System Metrics:** Prometheus & Grafana dashboards
- **Custom Logging:** Winston (Node.js), Express middleware
- **Network Monitoring:** Netdata, nload

---

5. Performance Test Scenarios

5.1 User Journey Scenarios

- **Standard User:** Log in (JWT), request project documentation, download output.
- **Power User:** Upload large templates, bulk-generate 100+ documents, export to Confluence/SharePoint.
- **Admin:** Monitor system health, approve/reject document jobs, manage templates.

5.2 Business Process Scenarios

- **End-to-End Document Generation:** API call → AI content generation → PDF/Word output → Publish to SharePoint.
- **Standards Compliance Check:** Submit requirements for compliance analysis → receive annotated report.

5.3 System Integration Scenarios

- **Multi-provider AI Orchestration:** Document generation with OpenAI fallback to Google AI.
- **Cross-platform Publishing:** Generate document → Publish to Confluence and SharePoint in parallel.

5.4 Background Process Scenarios

- **Nightly Batch Generation:** Automated creation of 1,000+ documents (scheduled job).
- **Template Sync:** Bulk import/export of templates from external repository.

5.5 Peak Load Scenarios

- **API Burst:** 200 concurrent users each submitting 10 document jobs within 5 minutes.
- **Simultaneous Integration:** 50 concurrent document exports to SharePoint and Confluence.

---

6. Test Tools and Technologies

6.1 Performance Testing Tools

- **Primary:** k6, Artillery, or JMeter for HTTP/API and WebSocket load generation

- **CLI Automation:** Custom Node.js scripts, shell scripts, or Artillery for CLI workflows

6.2 Monitoring Tools

- **APM:** Datadog, New Relic, or OpenTelemetry integration
- **System Metrics:** Prometheus, Grafana for resource monitoring
- **Custom Logs:** Winston, Express middleware for API logs

6.3 Data Analysis Tools

- **Reporting:** k6/Artillery HTML reports, Grafana dashboards, custom CSV/JSON export
- **Error Analysis:** ELK (Elasticsearch, Logstash, Kibana) stack for log aggregation

6.4 Load Generation & Distribution

- **Distributed Load:** Multiple geographically distributed load generators (e.g., Kubernetes jobs)
- **Network Emulation:** tc (Linux Traffic Control) for simulating latency/bandwidth constraints

6.5 Test Automation

- **CI Integration:** GitHub Actions, Azure DevOps, or Jenkins for automated nightly performance runs
- **Scripting:** TypeScript/Jest for pre/post-test automation (data setup, teardown)

---

7. Test Execution Strategy

7.1 Test Execution Schedule

Phase	Activities	Timeline
Environment Setup	Provision infra, configure monitoring	Day 1
Baseline Testing	Establish baseline metrics (low load)	Day 2
Load Testing	Gradual load increase, monitor KPIs	Day 3
Stress/Spike Tests	Rapid and incremental overload scenarios	Day 4
Volume/Endurance	Large data/batch jobs and 72-hour run	Day 5-7
Analysis & Tuning	Analyze results, optimize, retest if needed	Day 8+

7.2 Resource Allocation

- **Performance Lead:** Test design, oversight, reporting
- **Engineers (2-3):** Script development, execution, monitoring setup
- **DevOps:** Environment provisioning, scaling
- **Stakeholders:** Review and sign-off (as needed)

7.3 Test Data Management

- **Setup:** Automated scripts to seed templates/documents
- **Refresh:** Rollback/cleanup between test cycles
- **Security:** Mask or use synthetic data for integrations

7.4 Result Collection

- **Centralized Storage:** All metrics and logs stored in secured S3 bucket or Azure Blob
- **Automated Dashboard:** Real-time view in Grafana
- **Archival:** Retain raw and processed data for audit

7.5 Issue Management

- **Defect Tracking:** Jira or GitHub Issues with severity classification
- **Root Cause Analysis:** Link to log traces and metrics
- **Escalation:** Immediate notification for critical failures (>10% error rate, major outage)

---

8. Performance Metrics and KPIs

Metric Type	Description	Collection Tool
Response Time	Avg, median, 95th/99th percentile, min/max per endpoint	k6/Artillery, APM
Throughput	Requests/sec, jobs/hour	k6/Artillery
System Resource	CPU, memory, disk, network utilization (per node)	Prometheus, Grafana
Error Rate	% failed requests, error code distribution	k6/Artillery, logs
Availability	Uptime %, health check response times	Uptime monitor
Queue/Job Backlog	Pending jobs in document or integration queues	Custom API metrics

Metric Type	Description	Collection Tool
External Latency	AI provider, Confluence, SharePoint API response times	Custom probes

---

9. Success Criteria and Acceptance Thresholds

Category	Threshold/Criteria	Status
API Response Time	95% < 2s (normal), < 5s (peak), max 10s any time	Pass/Fail
Throughput	≥ 1,500 docs/hour, ≥ 50 req/sec sustained	Pass/Fail
Resource Usage	CPU < 70%, RAM < 75%, Disk I/O < 60% (per node)	Pass/Fail
Error Rate	< 1% overall, < 3% transient under spike	Pass/Fail
Uptime	≥ 99.9% during 72-hour endurance	Pass/Fail
Scalability	Linear scaling up to 5 nodes	Pass/Fail
Business Impact	No critical business function fails under test	Pass/Fail

- **Escalation Thresholds:**
    - Any sustained error rate > 5% triggers immediate investigation

    - Any job backlog > 10x average for over 10 minutes escalates to engineering

---

10. Risk Management and Contingency Planning

10.1 Performance Risks

- **AI Provider Latency/Outages:** May cause cascading delays in document generation.
- **Integration Service Throttling:** Rate limits from Confluence, SharePoint, or Adobe APIs.
- **Resource Exhaustion:** Memory leaks or unhandled async jobs in Node.js.
- **Scaling Misconfiguration:** Insufficient horizontal scaling or slow instance provisioning.
- **Template Complexity:** Large or malformed templates causing high CPU/memory usage.

10.2 Risk Mitigation Strategies

- **Provider Fallback:** Automatic switch to backup AI providers on timeout/failure.
- **Rate Limiting:** Internal throttling to avoid exceeding integration service quotas.
- **Health Checks:** Frequent, automated health checks with auto-remediation scripts.
- **Resource Monitoring:** Real-time alerts for high CPU/memory/disk with auto-scaling triggers.
- **Template Validation:** Pre-processing and validation of templates before job execution.

10.3 Contingency Plans

- **Job Queuing:** Queue and retry failed document jobs on transient errors.
- **Graceful Degradation:** Disable non-critical integrations under heavy load.
- **Rollback:** Roll back deployments if severe performance regressions detected.
- **Manual Intervention:** Engineering on-call for critical test windows.

10.4 Optimization Strategies

- **Code Profiling:** Node.js/TypeScript profiling to optimize hotspots.
- **Caching:** Redis or in-memory caching for frequent lookups and templates.
- **Horizontal Scaling:** Auto-scale API and worker instances in Kubernetes/Docker.
- **Async Processing:** Offload heavy/batch jobs to background workers.

10.5 Go/No-Go Decision Criteria

- **Go:** All pass/fail criteria met, no critical or high-severity unresolved defects, business impact assessed as acceptable.
- **No-Go:** Any critical failure in business processes, sustained resource exhaustion, or inability to scale as required.

Appendices

- **A. Test Data Sets:** Sample templates, synthetic documents, integration credentials (secured).
- **B. Load Profiles:** User concurrency, job submission rates, volume distributions.
- **C. Tool Configurations:** k6/Artillery scripts, monitoring dashboards, CI pipeline definitions.
- **D. Performance Test Reports:** Baseline, pre-optimization, and post-optimization result summaries.

End of Document
Prepared for: ADPA Engineering, DevOps, and Product Stakeholders
Contact: ADPA QA Team
Version: 1.0 – July 2025
