Performance Test Plan

Performance Test Plan

> **Project:** {{PROJECT_NAME}} **Version:** {{VERSION}} **Date:** {{DATE}}
> **Author:** {{AUTHOR}} **Status:** Draft | In Review | Approved **Reviewers:** {{REVIEWERS}}

Document History

Version	Date	Author	Changes
0.1	{{DATE}}	{{AUTHOR}}	Initial draft

---

1. Performance Testing Objectives

1. Validate that {{PROJECT_NAME}} meets the performance SLAs defined in the NFR document under normal operating conditions
2. Determine the maximum capacity the system can handle before performance degrades
3. Identify bottlenecks (database, application, infrastructure) before production release
4. Establish a performance baseline for regression comparison in future releases

Reference NFRs: {{NFR_DOC_LINK}}

---

2. Performance Requirements Reference

Endpoint / Feature	P50	P90	P95	P99	Error Rate	Throughput

`GET /` (homepage)	< {{P50}}ms	< {{P90}}ms	< {{P95}}ms	< {{P99}}ms	< {{ERR}}%	{{RPS}} req/s
`POST /api/auth/login`	< {{P50}}ms	< {{P90}}ms	< {{P95}}ms	< {{P99}}ms	< {{ERR}}%	{{RPS}} req/s
`GET /api/{{RESOURCE}}`	< {{P50}}ms	< {{P90}}ms	< {{P95}}ms	< {{P99}}ms	< {{ERR}}%	{{RPS}} req/s
`POST /api/{{RESOURCE}}`	< {{P50}}ms	< {{P90}}ms	< {{P95}}ms	< {{P99}}ms	< {{ERR}}%	{{RPS}} req/s
Database query (P95)	< {{DB_P95}}ms					
Page load (First Contentful Paint)			< {{FCP}}ms			

---

3. Test Types

3.1 Load Testing — Normal Load Simulation

Objective: Confirm the system meets SLAs under expected normal load **Normal load definition:** {{NORMAL_USERS}} concurrent users / {{NORMAL_RPS}} requests/second **Duration:** {{LOAD_DURATION}} minutes (after ramp-up) **Pass criteria:** All SLA targets in Section 2 met with ≤ {{LOAD_ERROR_RATE}}% error rate

3.2 Stress Testing — Beyond Normal Capacity

Objective: Find the maximum capacity and understand failure behavior **Starting point:** {{STRESS_START_USERS}} users, increasing by {{STRESS_INCREMENT}} users every {{STRESS_STEP}}min **Stop condition:** Error rate > {{STRESS_STOP_ERROR}}% or service crashes **Pass criteria:** System fails gracefully (circuit breakers, meaningful error messages), data integrity maintained

3.3 Spike Testing — Sudden Traffic Surges

Objective: Verify system handles sudden traffic spikes without crashing **Baseline:** {{SPIKE_BASELINE}} users **Spike to:** {{SPIKE_PEAK}} users ({{SPIKE_MULTIPLIER}}× baseline) **Spike duration:** {{SPIKE_DURATION}} minutes **Pass criteria:** System recovers to baseline performance within {{SPIKE_RECOVERY}} minutes after spike ends

3.4 Endurance / Soak Testing — Sustained Load

Objective: Identify resource leaks and degradation under sustained load **Load level:** {{SOAK_USERS}} users ({{SOAK_PERCENT}}% of normal load) **Duration:** {{SOAK_DURATION}} hours **Metrics to watch:** Memory usage trend, response time trend, disk space, database connection pool **Pass criteria:** No upward trend in memory/response time over the soak period

3.5 Scalability Testing — Increasing Load

Objective: Verify linear (or better) scaling as infrastructure grows **Test:** Step load from {{SCALE_START}} to {{SCALE_MAX}} users while scaling from {{MIN_INSTANCES}} to {{MAX_INSTANCES}} instances **Pass criteria:** Throughput increases proportionally (≥ {{SCALE_EFFICIENCY}}% efficiency) with added instances

---

4. Load Profiles

Scenario	Virtual Users	Ramp-Up	Hold	Ramp-Down	Think Time	Iterations
Smoke (quick sanity)	{{SMOKE_VU}}	{{SMOKE_RAMP}}s	{{SMOKE_HOLD}}min	30s	1s	1
Load (normal)	{{LOAD_VU}}	{{LOAD_RAMP}}min	{{LOAD_HOLD}}min	5min	{{THINK}}s	—
Stress	{{STRESS_VU}} (increasing)	{{STRESS_RAMP}}min	Until failure	—	{{THINK}}s	—
Spike	{{SPIKE_VU}} (instant)	0s	{{SPIKE_HOLD}}min	0s	{{THINK}}s	—
Soak	{{SOAK_VU}}	{{SOAK_RAMP}}min	{{SOAK_HOLD}}h	10min	{{THINK}}s	—

Think time simulation: Random between {{THINK_MIN}}s and {{THINK_MAX}}s (realistic user behavior)

---

5. Test Environment

CRITICAL: Performance tests run on a dedicated environment with the same infrastructure sizing as production.

Component	Test Environment	Production
App instances	{{TEST_APP_INSTANCES}} × {{TEST_INSTANCE_TYPE}}	{{PROD_APP_INSTANCES}} × {{PROD_INSTANCE_TYPE}}
Database	{{TEST_DB_SPEC}}	{{PROD_DB_SPEC}}
Cache	{{TEST_CACHE_SPEC}}	{{PROD_CACHE_SPEC}}
CDN	Disabled (direct hit)	Enabled

Load generator infrastructure:

- Tool: {{PERF_TOOL}}
- Load generator location: {{LG_LOCATION}} (same region as app, separate VPC)
- Load generator sizing: {{LG_SPEC}}

---

6. Test Data Requirements

Data Type	Volume	Generation Method
Users	{{USER_COUNT}}	Script + factory
{{DATA_TYPE_1}}	{{VOLUME}}	Bulk import script
{{DATA_TYPE_2}}	{{VOLUME}}	Bulk import script
Search index	Production-sized	Seeded from anonymized production data

Database size at test time: {{DB_SIZE}}GB **Data preparation:** `bash scripts/perf-seed.sh` (estimated time: {{SEED_TIME}}min)

---

7. Tools & Infrastructure

Tool	Version	Purpose	Config
{{PERF_TOOL}}	{{VERSION}}	Load generation	`perf-tests/`
{{MONITOR_TOOL}}	{{VERSION}}	Real-time metrics during test	Dashboard link
{{APM_TOOL}}	{{VERSION}}	Application performance profiling	Dashboard link
{{DB_MONITOR}}	{{VERSION}}	Database query analysis	Dashboard link

Script location: `{{PERF_SCRIPT_PATH}}`

8. Key Metrics to Capture

Response Time

Metric	Description	Tool
P50 (median)	Half of requests faster than this	{{TOOL}}
P90	90% of requests faster than this	{{TOOL}}
P95	95% of requests faster than this	{{TOOL}}
P99	99% of requests faster than this	{{TOOL}}
Max	Worst single request	{{TOOL}}

Throughput

Metric	Description
Requests/second	Total throughput at peak load
Transactions/second	Successful business transactions/second
Data transferred	Total MB/s in + out

Error Metrics

Metric	Target
HTTP error rate (5xx)	< {{HTTP_ERR}}%
Timeout rate	< {{TIMEOUT_ERR}}%
Connection refused rate	0%

Resource Utilization

Resource	Warning	Critical
App CPU	> {{CPU_WARN}}%	> {{CPU_CRIT}}%
App Memory	> {{MEM_WARN}}%	> {{MEM_CRIT}}%
DB CPU	> {{DB_CPU_WARN}}%	> {{DB_CPU_CRIT}}%
DB Connections	> {{DB_CONN_WARN}}% of pool	> {{DB_CONN_CRIT}}%

Resource	Warning	Critical
Cache hit ratio	< {{CACHE_HIT}}%	< {{CACHE_CRIT}}%

Database Query Performance

Metric	Target
Average query time	< {{AVG_QUERY}}ms
Slow queries (> {{SLOW_THRESHOLD}}ms)	< {{SLOW_COUNT}} per minute
Deadlocks	0

---

9. SLA Targets Per Endpoint

Endpoint	Method	P95 SLA	Error Rate SLA	Notes
`/`	GET	{{P95}}ms	< {{ERR}}%	Static or cached
`/api/auth/login`	POST	{{P95}}ms	< {{ERR}}%	Auth critical path
`/api/{{RESOURCE}}`	GET	{{P95}}ms	< {{ERR}}%	{{NOTE}}
`/api/{{RESOURCE}}`	POST	{{P95}}ms	< {{ERR}}%	{{NOTE}}

---

10. Baseline Establishment

Baseline criteria: System in stable state, seeded with production-realistic data, running load test scenario for {{BASELINE_DURATION}} minutes

Baseline metrics to record:

Metric	Baseline Value	Date Recorded
P95 latency (key endpoints)	TBD	{{DATE}}
Throughput at normal load	TBD	{{DATE}}
Error rate at normal load	TBD	{{DATE}}
CPU utilization at normal load	TBD	{{DATE}}
Memory utilization at normal load	TBD	{{DATE}}

Regression threshold: Alert if any metric degrades > {{REGRESSION_THRESHOLD}}% vs baseline

11. Test Execution Schedule

Run Type	Trigger	Environment	Frequency
Smoke	Every deployment	Staging	Per deploy
Load (baseline)	Release candidate	Production-sized	Per release
Stress	Major feature releases	Production-sized	Quarterly
Soak	Before major releases	Production-sized	Per release
Scalability	Infrastructure changes	Production-sized	As needed

12. Results Analysis Template

Test Run ID: {{RUN_ID}} **Date:** {{DATE}} **Tester:** {{TESTER}} **Scenario:** {{SCENARIO}}
Build / Version: {{VERSION}}

Endpoint	P50 (ms)	P90 (ms)	P95 (ms)	P99 (ms)	Error %	RPS	Status vs SLA
{{ENDPOINT}}	—	—	—	—	—	—	□ Pass / □ Fail

Summary:

- Peak concurrent users: {{PEAK_USERS}}
- Peak throughput: {{PEAK_RPS}} req/s
- Test duration: {{DURATION}} min
- Total requests: {{TOTAL_REQUESTS}}
- Total errors: {{TOTAL_ERRORS}}

Notable findings:

- {{FINDING_1}}
- {{FINDING_2}}

Comparison to baseline:

- {{COMPARISON}}

Recommendation: Pass / Fail / Conditional pass with {{CONDITIONS}}

13. Bottleneck Identification Process

```
1. Check application error rate → Is the app returning errors, or just slow?
2. Check P99 vs P50 spread → Large spread = outlier requests (DB queries, locks, GC pauses)
3. Check CPU saturation → CPU > 80% sustained = compute bottleneck
4. Check memory → Memory growing during test = leak / GC pressure
5. Check DB metrics → Slow queries, high connections, deadlocks
6. Check cache hit rate → Low cache hit = DB overloaded unnecessarily
7. Check external calls → Third-party API latency or rate limiting
8. Check network → Bandwidth saturation, packet loss
```

---

14. Remediation Tracking

Issue	Found In	Severity	Root Cause	Fix	Fixed In	Verified
{{ISSUE}}	{{SCENARIO}}	{{SEVERITY}}	{{CAUSE}}	{{FIX}}	{{VERSION}}	{{DATE}}

Related Documents

- Test Strategy
- Monitoring & Observability
- SLA Report

Approval

Role	Name	Date	Signature
Author			
Reviewer			
Approver			

Revision #6
Created 2026-02-23 12:06:06 UTC by John
Updated 2026-05-25 07:34:16 UTC by John