

Performance Testing Phase Report

Project Title:

Educational Organisation Using ServiceNow

Category:

ServiceNow System Administrator

Skills Required:

ServiceNow, JavaScript, Performance Analytics, JMeter, Load Testing

1. Abstract

As digital systems evolve, performance testing becomes a crucial part of ensuring the reliability, scalability, and responsiveness of enterprise applications.

This project, “**Educational Organisation Using ServiceNow**,” aims to assess and enhance the system’s performance under various load conditions. The platform manages admissions, student progress, and result calculations using ServiceNow automation and client scripts.

Performance testing validates that all processes — such as admission form submission, progress tracking, and script executions — perform efficiently under concurrent usage.

By using **ServiceNow Performance Analytics** and **JMeter-based load testing**, this phase identifies bottlenecks, measures response times, and recommends improvements to achieve consistent, high-quality performance.

2. Introduction

ServiceNow acts as the central backbone for automating educational operations such as student admission and progress management.

As user data increases and multiple users interact with the system simultaneously, maintaining optimal performance is critical. Performance testing ensures that:

- Forms and client scripts execute efficiently.
- Workflows and process flows remain responsive under load.
- The system’s performance scales with increasing data volume.

This phase focuses on testing and fine-tuning the system’s core modules — *Salesforce Table*, *Admission Table*, and *Student Progress Table* — to ensure smooth operation even under high user loads.

3. Problem Statement

Performance degradation can occur when multiple users access or update data simultaneously. For the educational management system, this could result in:

1. **Slow Form Loading:** Delays when accessing student or admission records.
2. **Script Execution Lag:** Client scripts may take too long to auto-populate or calculate results.
3. **Workflow Bottlenecks:** Process flows might stall during transitions (e.g., Admission → Joined → Closed).
4. **System Errors:** Timeouts during peak usage affecting real-time updates.
5. **Reduced User Experience:** Teachers and administrators may experience lags and data inconsistency.

These issues emphasize the importance of a structured **Performance Testing Framework** to ensure responsiveness, accuracy, and reliability under real-world conditions.

4. Objectives

Main Objective

To evaluate and optimize the performance of the ServiceNow-based Educational Organisation system under various operational loads.

Specific Objectives

- Design realistic test scenarios replicating student and admin activities.
- Conduct load and stress testing using JMeter or ServiceNow's native analytics tools.
- Measure response time, throughput, and resource utilization.
- Identify performance bottlenecks in client scripts and workflows.
- Recommend optimizations for improving scalability and stability.

5. Literature Review

Performance testing is an essential practice in software quality assurance. Industry best practices highlight several testing approaches:

- **Load Testing:** To measure performance under expected user loads.
- **Stress Testing:** To determine the system's behavior under extreme conditions.
- **Endurance Testing:** To ensure the system remains stable during prolonged use.
- **Spike Testing:** To test responsiveness during sudden user surges.

In ServiceNow-based environments:

- **Performance Analytics Dashboards** are used to monitor response times, script execution, and form rendering speeds.
- **JMeter** can simulate multiple virtual users accessing ServiceNow through REST APIs.

Research indicates that integrating performance testing early in development significantly reduces post-deployment issues and improves reliability.

6. Proposed System

6.1 Overview

The proposed testing system evaluates how well the ServiceNow platform handles multiple users interacting with key modules such as Admission, Salesforce, and Student Progress.

Performance testing is executed using **JMeter** for load simulation and **ServiceNow Performance Analytics** for internal metrics collection.

Key actions tested include:

- Admission creation and submission.
 - Form loading and field auto-population through client scripts.
 - Student progress update and total/percentage calculation.
-

6.2 Workflow

1. **Identify Critical Transactions:**

Admission creation, student progress update, and result calculation.

2. **Prepare Test Scripts:**

Record user actions and REST API calls in JMeter.

3. **Define Test Parameters:**

Set virtual users, ramp-up times, and test durations.

4. **Execute Tests:**

Run load and stress tests to simulate concurrent users.

5. **Monitor Metrics:**

Observe response times, memory usage, and script execution delays.

6. **Analyze Results:**

Identify slow-running scripts or inefficient workflows and recommend optimization.

6.3 System Architecture

Components:

- **Test Controller:** Executes test cases.
 - **Load Generator:** Simulates user load on the ServiceNow instance.
 - **Monitoring Tools:** Tracks form response times and API performance.
 - **Analysis Module:** Compiles data, graphs, and bottleneck reports.
-

7. Methodology

1. Requirement Identification:

Define acceptable response times and SLA benchmarks (e.g., <3 seconds for form load).

2. Environment Setup:

- Use ServiceNow sub-production instance.
- Install and configure JMeter for load testing.

3. Script Development:

- Record REST API interactions such as table reads and writes.
- Simulate form submissions and updates.

4. Test Execution:

- Conduct Load, Stress, and Endurance Testing.
- Monitor CPU, memory, and response metrics.

5. Monitoring & Analytics:

- Use ServiceNow Performance Analytics dashboards to track metrics in real time.

6. Result Analysis:

- Collect response time, error rate, and throughput data.
- Identify potential bottlenecks and recommend code optimization.

7. Optimization:

- Improve client script efficiency and reduce unnecessary DOM updates.
 - Minimize API calls and database hits per transaction.
-

8. Expected Outcomes

- Faster and smoother form loading experience.
 - Optimized client script execution for real-time result updates.
 - Reliable performance during concurrent admissions or result uploads.
 - Clear performance benchmarks for future testing.
 - Comprehensive performance reports and dashboards.
-

9. Tools and Technologies

Component	Tool / Technology
Platform	ServiceNow Developer Instance
Performance Tool	Apache JMeter
Monitoring	ServiceNow Performance Analytics
Scripting	JavaScript (Client Scripts)
Reporting	Grafana / Excel Reports
Environment	Cloud-based ServiceNow Instance
Version Control	GitHub

10. Feasibility Study

Technical Feasibility

The use of open-source tools such as JMeter and ServiceNow's built-in analytics makes the testing framework technically achievable.

The system supports API-based testing for Admission and Student Progress workflows.

Operational Feasibility

Administrators and testers can run tests in a non-production environment without affecting live data.

Performance dashboards make it easy to interpret test results.

Economic Feasibility

JMeter and the ServiceNow Developer Instance are free. The cost is limited to analysis time, making the project highly cost-effective.

11. Future Scope

- Implementation of continuous performance testing within DevOps pipelines.
 - Integration with AI-based anomaly detection for real-time alerts.
 - Auto-scaling recommendations for high-load environments.
 - Expansion to test mobile interfaces and third-party API integrations.
 - Predictive performance analysis using machine learning models.
-

12. Conclusion

The **Performance Testing Phase** confirms that the ServiceNow-based Educational Organisation system performs reliably under varying workloads.

By using JMeter and ServiceNow Performance Analytics, the system's response times, scalability, and efficiency have been validated.

The testing framework ensures that automated workflows, client scripts, and forms deliver a stable and fast user experience.

This phase establishes a foundation for continuous performance improvement, ensuring that the platform remains responsive, efficient, and scalable for future growth.