Performance Testing for Service Now Family Expense

Date	1-11-2025
Team id	NM2025TMID00838
Project name	Calculating Family Expenses using service now
Team member	4

Performance Testing for Service Now Family Expense Management System

---

Objective

To verify that the **Family Expense Management application** performs efficiently under expected and peak loads, ensuring:

- Fast response times for expense entries and dashboard updates.
- Quick execution of automated flows (totals, alerts, referrals).
- Scalability with multiple family members or households using the app.
- Stable system behavior under stress (bulk records, concurrent users).

---

1. Key Modules to Test

Module	Functionality to Test	Performance Goals
Expense Entry Form	Add/update/delete expense records	Response < 3 sec
Automated Totals Flow	Auto recalculation after each expense	Executes within 2–5 sec
Alert Notification Flow	Triggered alerts on thresholds	Alert delivered within 5 sec
Referral System	Referral creation, approval, and reward update	Processes within 3–4 sec
Dashboard / Reports	Load charts, totals, and history	Page load < 4 sec
AI Forecasting (if added)	Generate predictions	< 10 sec for 1-month forecast

---

2. Performance Test Types

Performance Testing for Service Now Family Expense

Test Type	Purpose	Example Scenario
Load Testing	Measure performance under expected load.	50 users entering expenses simultaneously.
Stress Testing	Determine limits of system stability.	200 concurrent expense submissions.
Spike Testing	Check response to sudden user increase.	Burst of 100 new users in 1 min (via referral).
Endurance Testing	Ensure stability over long usage periods.	24-hour simulation of ongoing expense activity.
Scalability Testing	Check how system handles growth.	Double number of members/records monthly.
Component Testing	Validate specific flows or notifications.	Test alert generation flow in isolation.

---

3. Test Environment Setup

Component	Description
Platform	ServiceNow Personal Developer Instance (PDI) or sub-production instance
Version	Latest ServiceNow release (e.g., Washington DC / Vancouver)
Tables	`x_family_expense`, `x_family_budget`, `x_family_alert`, `x_family_referral`, `x_family_member`
Tools	- ServiceNow **ATF (Automated Test Framework)** - **LoadRunner / JMeter** for API-level tests - **Performance Analytics** for dashboards
Data Set	1,000+ expense records, 10+ members, 5 categories, 50+ alerts, 20 referrals

---

Performance Testing for Service Now Family Expense

4. Test Scenarios

Test Case ID	Scenario	Input / Trigger	Expected Result	Success Criteria
PT-01	Add Expense Record	Insert 100 expenses in 1 min	Totals auto-update	Update < 5 sec
PT-02	Budget Limit Alert	Simulate overspending	Alert generated automatically	Alert < 3 sec
PT-03	Referral Approval	Approve 10 referrals	Points updated instantly	Reward update < 4 sec
PT-04	Monthly Summary	Trigger scheduled flow	Dashboard refreshed	Execution < 10 sec
PT-05	Dashboard Load	Open with 1000+ records	Charts load smoothly	Load time < 4 sec
PT-06	Concurrent Users	50 users submit expenses	No errors / slowdown	< 10% delay
PT-07	Stress Test	500 random submissions	System remains stable	No crash or data loss

---

□ 5. Tools and Techniques

Tool	Usage
ServiceNow Automated Test Framework (ATF)	For simulating user interactions and verifying UI/flow performance.
ServiceNow Performance Dashboard	Track average transaction times, script execution, and flow durations.
Postman / JMeter	Test REST API performance (if APIs used).
Instance Diagnostics (via System Diagnostics → Stats)	Monitor DB query speed, transaction time, and system logs.
Script Logs & Flow Metrics	Measure execution time of Flow Designer automations.

Performance Testing for Service Now Family Expense

6. Key Performance Metrics

Metric	Description	Target Value
Average Transaction Time	Time to save a record	< 2 sec
Flow Execution Time	Total + Alert + Referral flows	< 5 sec average
Notification Latency	Time from trigger → user alert	< 3 sec
CPU / Memory Usage	Instance resource utilization	< 70%
Throughput	Requests processed per minute	≥ 100
Error Rate	Failed transactions	< 1%