<project>

PERFORMANCE ENGINEERING RESULTS

Prepared By	{Name}, {Company}
Date	{Date}
Document No.	{Doc #}
Version	{version}
Status	{status}
Next Step	{Notes here}

Revision History

Date	Version	Description	Author

TABLE OF CONTENTS

1 EXECUTIVE SUMMARY

<Summarize the testing engagement. The summary should state if acceptance criteria were met, as well as the overall recommendation of the Performance Engineering Team.

The following may be used as an example where all criteria were NOT met:

The Performance Testing effort was completed on Monday, June 11th, 2001. The Performance Engineering Team discovered that on the hardware and software configuration tested, that a 150 concurrent user load through a LAN connection produced an acceptable user experience within the guidelines of <stakeholder> response time range. This number of 150 concurrent user maps out to approximately 50,000 unique sessions a month given a normal distribution pattern for United States web traffic.

During this effort, two potential issues were identified for <stakeholder> to pursue. The primary issue is a possible memory leak and the secondary issue concerns efficiency in database access. As part of the agreed upon process of performance testing, <stakeholder> has begun to take appropriate corrective action in line with <stakeholder> priorities.

The following may be used as an example where all criteria were met:

The Performance Testing effort was completed on Monday, June 11th, 2001. The Performance Engineering Team discovered that on the hardware and software configuration tested, that a 1500 concurrent user load through a LAN connection produced an acceptable user experience within the guidelines of <stakeholder's> response time range. Testing showed that 1500 concurrent users returned times that were slower than the acceptance criteria for dial-up modem speeds, but were ultimately deemed to be sufficient for this release of the application.
>

2 INTRODUCTION

2.1 Scope

This document includes a summary of performance engineering results, a tuning overview, and conclusions/recommendations for the <*application*>. This document does not address functional testing, nor does it address detailed application tuning. All data used to create this document resides in the Performance Testing folder in the <*fillin*>.

2.2 Purpose

The purpose of this document is to report on the <*application*> actual performance as compared to the acceptance criteria enumerated in the Performance Engineering Strategy document. Specifically, this document details the:

- Performance Acceptance Criteria
- Workload Distribution exercised
- Measurements gathered for the application
- Summary of Tests performed
- Summary of Measurements collected
- Summary of Tuning
- Results and Conclusions
- Appendices for supporting data

2.3 Related Documents

The following documents are referenced in this document.

Ref.	Name (Document No.)	Date
1.	Performance Engineering Methodology	
2.	<project>Performance Engineering Strategy	
3.	SOW	
4.	Project plan (if applicable)	
5.	Requirements doc (if applicable)	

3 PERFORMANCE ACCEPTANCE CRITERIA

3.1 Introduction

Performance efforts always have two sets of criteria associated with them. The first are performance criteria (requirements and goals), and the second are engagement completion criteria. In the sections below, both types of criteria are explained in general and in specific detail for the <*application*> performance engineering effort. The performance effort will be deemed complete when either all of the performance criteria are met, or any one of the engagement completion criteria is met, or <*stakeholder*> and the Performance Engineering Team agree that best possible performance has been achieved under the given project parameters.

The sections below reflect the criteria enumerated in the Performance Engineering Strategy Document. Each criteria has additionally been noted with "Pass", "Fail", "Not Tested", and/or "See section #.#".

3.2 Performance Criteria

Performance criteria are the specific target performance requirements and goals of the system under test. In the case of the <*application*>, <*Performance Team*> and <*stakeholders*> have worked collaboratively through mutual experience, conversations and workshops to develop the criteria enumerated below. The preferred result by both <*Performance Team* > and < *stakeholders*> of the performance engineering effort is to validate that the application meets all of these goals and requirements currently and/or tune the application until these goals are met. If this is not possible, at least one of the engagement completion criteria from the next section must be met for overall performance acceptance.

<The following examples indicate the type of acceptance criteria that should be listed in this section and at what level of detail it should be written. The actual performance acceptance criteria for the project should be included in the format of the examples seen here:

example 1

3.2.1 Requirements

Requirements are those criteria that must be met for the application to "go live" and become a production system.

3.2.1.1 General Requirements

1. *System stable and responsive with 250 hourly users accessing <system> via Intranet in accordance with the User Community model.*
2. *<system> exhibits not more than a 10 second response time (via Intranet) 90% of the time under a 250 hourly user load with less than 5% abandonment.*
3. *<system> stable and responsive under spike loads.*
4. *<system> stable and responsive under stress load.*
5. *<system> stable and responsive under expected load during scheduled maintenance/batch processing.*

3.2.1.2 Time Sensitive/High Profile Activity Requirements

1. *Check Out Item - 3 seconds*
2. *Check In Item - 3 seconds*

3.2.2 Goals

Goals are those criteria that are desired for the application which are in some way different than the previously stated requirements. Testing and tuning will continue until either all goals are met, or until time/money is at an end and the Requirements are met.

3.2.2.1 General Goals

1. *System stable and responsive with 500 hourly users accessing <system> via Intranet in accordance with the User Community model.*
2. *<system> exhibits not more than a 5 second response time (via Intranet) 85% of the time and not more than an 8 second response time 95% of the time under a 250 hourly user load with less than 5% abandonment.*
3. *All field validations exhibit not more than a 3 second response time (via Intranet) 95% of the time under maximum expected user loads.*

3.2.2.2 Time Sensitive/High Profile Activity Goals

1. *Check Out Item – 1.5 seconds*
2. *Check In Item – 1.5 seconds*

example 2

The objectives of the Performance Engineering Effort are:

- *To validate the scalability of the technical architecture and operability on a shared platform (up to 1000 concurrent users).*
- *To validate system performance of:*
  - *All user actions that require a page or screen to be loaded or refreshed will be fully displayed in 6 seconds 95% of the time when accessed over a 10 Mbs Lan while there is a 200 user load on the system.*
  - *To validate that the system does not exhibit any critical failures under stress (unrealistic load)*
  - *Identify and ensure that performance issues uncovered outside of the stated performance criteria are documented and/or addressed prior to deployment.*

example 3

The objectives of the Performance Engineering Effort are:

- *To validate the scalability of the technical architecture and operability on a shared platform (up to 1000 concurrent users).*
- *To validate system performance of the following average page load times over a 100 Mbs Lan with the distribution of account sizes described later in this document.*

Screen Name	*Timer Name*	*Page Load time, no Load*	*Page Load w/ 200 User Load*
Group Summary	*tmr_grp_sum*	*12 sec*	*20 sec*
Employee Listing	*tmr_emp_lst*	*12 sec*	*20 sec*
Search Results View	*tmr_srch_rslt*	*12 sec*	*20 sec*
Customer Adjustment Popup	*tmr_cust_adj*	*6 sec*	*10 sec*
Term Confirmation Popup	*tmr_term_conf*	*9 sec*	*15 sec*

>

3.3 Engagement Complete Criteria

In cases where performance requirements or goals cannot be achieved due to situations outside of the control of the Performance Engineering Team, the performance effort will be considered complete when any of the following conditions are met:

<This is an example of engagement completion criteria. This should be used as an example. The actual engagement acceptance criteria should be listed here in the format of the example below

example

- *All bottlenecks preventing the application from achieving the performance criteria are determined to be outside Performance Engineering Team control/contract.*
- *The pre-determined engagement end date is reached.*
- *The Performance Engineering Team and stakeholders agree that the application performs acceptably, although some performance requirements or goals have not been achieved.*

>

4 WORKLOAD DISTRIBUTION

4.1 Introduction

A Workload Distribution is a representation of the functions performed by a user community on a system. For example, during the course of a day on a retail-based website, most users are shopping, some are doing a search for a specific product, some are checking out and all this while a single administrator may be updating prices on products. A Workload Distribution is based on a percentage of users performing a specific function over a given period of time. Using the above example a Workload Distribution could be: shopping – 83%, searching - 5%, checking out – 10% and administration - 2%.

Performance engineering is separate from functional testing; therefore, it is not necessary to simulate all possible paths through the system. Specific functions that are expected to draw the largest number of users are scripted to exercise the critical system components. Less vital functions that may not be used as heavily are scripted to provide a varying workload, creating a more realistic mix. As the number of concurrent users grows, the load increase on the system will often expand exponentially; therefore, it is imperative to accurately simulate the expected workload of the system. Experience shows that modeling 80% of a user population provides accurate load tests.

The Workload Distribution for a series of tests is represented in Rational Test Manger suites that include individual scripts. One or more scripts are recorded to represent each unique function to be tested. The suite defines the percentage of users that will execute each script as well as how users enter the site and how many times each function will be performed by each user.

4.2 Workload Distribution for <application>

The sections below describe, in detail, the workload distributions to be used for testing the *<application>.* The diagram (figure 4.1) shows the overall workload distribution path. The vertical dashed lines show the common start point for the scripts within the testing of the site. One can see that a virtual user's activity in the site does not necessarily depend on how the user got to the site and that each virtual user may exercise a unique combination of functionality based on the determined rate of frequency discussed below.

The Workload Distribution percentages presented in this document were created in order to simulate actual user activity on the site. These numbers however, may be modified or changed during the testing effort based on request from *<stakeholders>* in order to simulate alternate system load.

<The following figure and section is an example of a sample Workload Distribution. They should be used as an example and the activities and percentages should be changed to meet the specific Workload Distribution for the application under test.

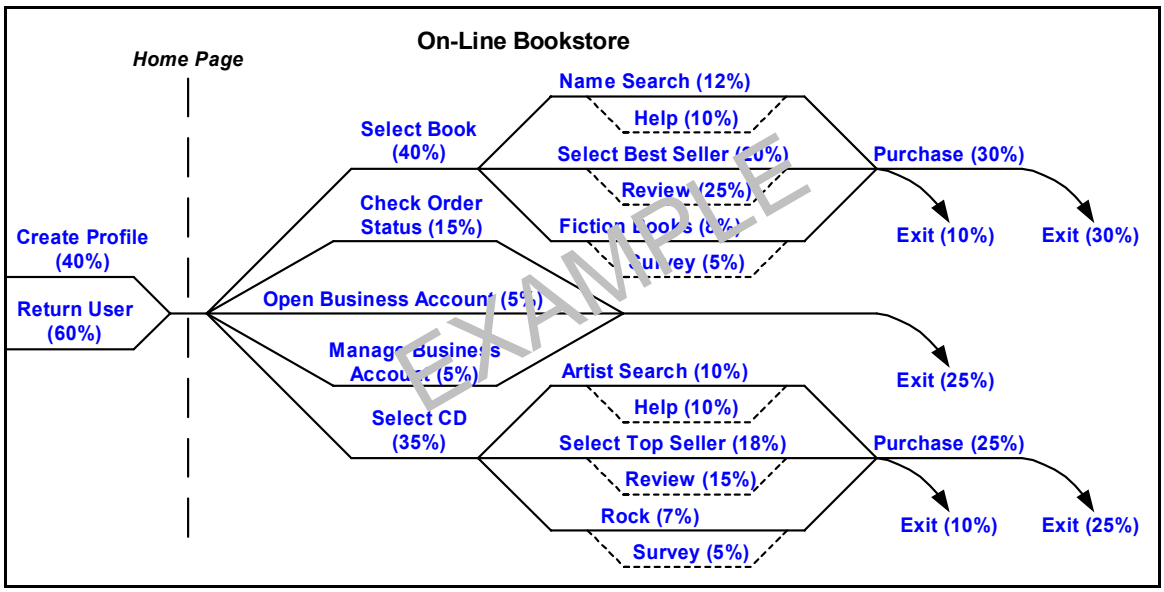


Figure 3.1 – Overall Workload Distribution

The Workload Distribution for <application> includes all of the functions to be executed during the performance testing effort. The dashed vertical line represents common start and/or end parts for the script. >

5 BASELINE RESULTS

5.1 Introduction

Baseline results represent each user activity being performed by a single user over multiple iterations. These baselines were used primarily to validate that the scripts have been developed correctly. All baselines were executed a minimum of 25 times. All reported times are statistical calculations (averages) of all 25 (or more) iterations. Each test execution run was separated by at least a minute, and every user wait time (time between user interactions with the system) was exactly 8 seconds to ensure baseline tests are identical. For these tests all stakeholder side caching was disabled, and each page was pre-compiled.

Due to the extreme number of times collected, only those with times greater than two (2) seconds or those activities that later exhibited poor performance are included in this document. The charts and figures below depict the times for the transactions described in the Performance Engineering Strategy document that either did not meet the stated goals of the system during the baseline, did not meet the stated goals under load, or were over 2 seconds. All of the activities (timers) listed in the Performance Engineering Strategy document and not listed here were never observed to not meet performance goals. The times below times reflect the system time to process various requests when only one user is accessing the application. All values are in seconds. For example, figure 5.1 shows that the time it took for the <*screenname*> screen to appear after clicking on the link was X.XX seconds. All charts scale up to a maximum response time of 60 seconds. Any response time in excess of the goal are noted in blue and those in excess of the requirements are noted in red.

A complete report of times can be found either in the Appendices or in the Rational Repository.

5.1.1 System Architecture

The baseline environment is a shared environment consisting of:
Load Balancer(s) - <*describe*>
Proxy – <*describe*>
Web Server(s) - <*describe*>
Appserver(s) - <*describe*>
Database - <*describe*>

5.2 Baseline Results

<Narrative description of the baseline test as actually conducted>

Screen Name	Timer Name	Avg Time (sec)	Standard Deviation (sec)	95th Percentile time (sec)
Group Summary	tmr_grp_sum			
Employee Listing	tmr_emp_lst			
Search Results View	tmr_srch_rslt			
Customer Adjustment Popup	tmr_cust_adj			
Term Confirmation Popup	tmr_term_conf			
Coverage Type View	tmr_cov_type			
Adjustment – Member View	tmr_adj_memb			
Adjustment – Other View	tmr_adj_other			
Service Fees View	tmr_svc_fees			
Detail by Account	tmr_dtl_acct			

Table 5.1 <sample>

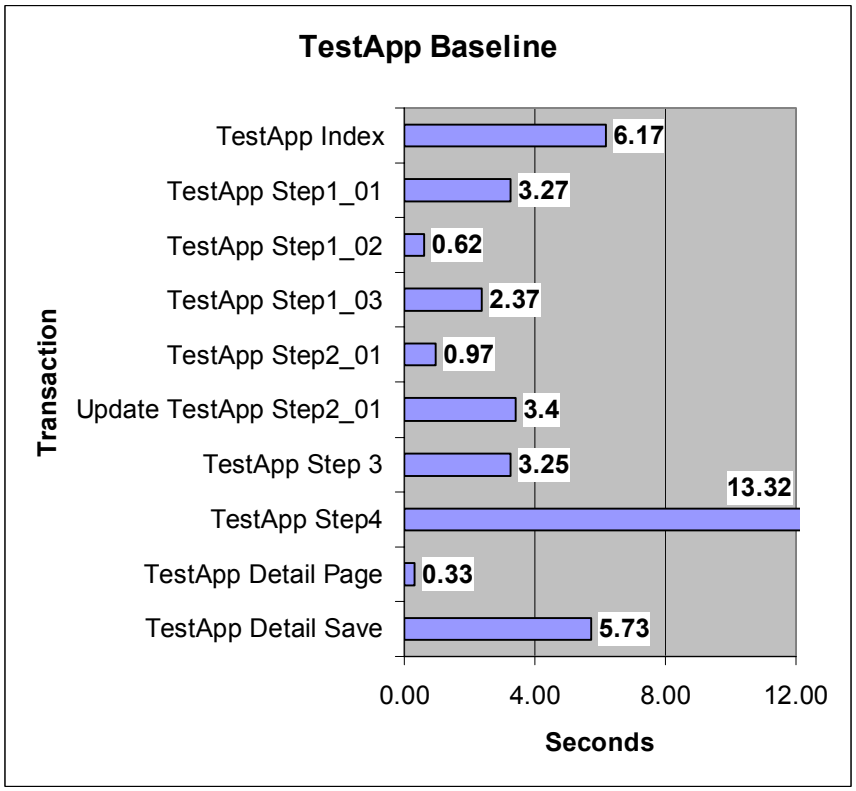


Figure 5.1 <sample>

6 BENCHMARK RESULTS

6.1 Introduction

A benchmark, or light load, scenario is generally a small community of users compared to the target load. This community of users must be large enough to represent a reasonable sample of the entire user community. Executing these tests ensures that the testing environment behaves as expected under light load before more demanding testing begins.

Additionally, the results of these tests are used as a benchmark to compare with all future tests results. Performance results obtained under the benchmark load should meet or exceed all indicated performance requirements; otherwise tuning must begin with the benchmark load. Assuming no performance problems are noticed during this scenario, the results obtained can be used as "best case" results. These results indicate how the system performs when it is not under noticeable stress, but is still performing all required functions, thus allowing conclusions to be drawn about the performance of the system during the higher load tests.

The <*application*> will be benchmarked, in the environments described below. This benchmark is intended to provide a basis of comparison for future testing. Tuning may occur during this benchmarking effort if critical bottlenecks are detected. Once any required tuning is complete, end-to-end page load times will be collected for each timer indicated in section 4 of this Performance Engineering Strategy document.

The <*application*> will be re-benchmarked each time an iteration of either tuning or development has been completed on a module. This ensures that there is always a known valid point of comparison for all scheduled tests. The Benchmark load for all modules will be 10 users, entering the system randomly over a 5-minute period and performing the tasks outlined in section 4 for either one hour or 5 complete iterations at a realistic user's pace.

If benchmark results do not meet the stated performance acceptance criteria, the Performance Engineering Team and <*stakeholder*> will work together to either resolve the bottlenecks or revise the test strategy. Continuing on to the next phase of testing without fixing the performance issues will not add value to the project.

6.1.1 System Architecture

The benchmark environment is a shared environment consisting of:
Load Balancer(s) - <*describe*>
Proxy – <*describe*>
Web Server(s) - <*describe*>
Appserver(s) - <*describe*>
Database - <*describe*>

6.2 Benchmark Results

Due to the extreme number of times collected, only those with times greater than two (2) seconds or those activities that later exhibited poor performance are included in this document. The charts and figures below depict the times for the transactions described in

the Performance Engineering Strategy document that either did not meet the stated goals of the system during the benchmark test, did not meet the stated goals under load, or were over 2 seconds. All of the activities (timers) listed in the Performance Engineering Strategy document and not listed here were never observed to not meet performance goals. All values are in seconds. For example, figure 6.1 shows that the time it took for the *<screenname>* screen to appear after clicking on the link was X.XX seconds. All charts scale up to a maximum response time of 60 seconds. Any response time in excess of the goal are noted in blue and those in excess of the requirements are noted in red.

A complete report of times can be found either in the Appendices or in the Rational Repository.

6.2.1 Benchmark Results

Screen Name	Timer Name	Avg Time (sec)	Standard Deviation (sec)	95th Percentile time (sec)
Group Summary	tmr_grp_sum			
Employee Listing	tmr_emp_lst			
Search Results View	tmr_srch_rslt			
Customer Adjustment Popup	tmr_cust_adj			
Term Confirmation Popup	tmr_term_conf			
Coverage Type View	tmr_cov_type			
Adjustment – Member View	tmr_adj_memb			
Adjustment – Other View	tmr_adj_other			
Service Fees View	tmr_svc_fees			
Detail by Account	tmr_dtl_acct			

Table 6.1 <sample>

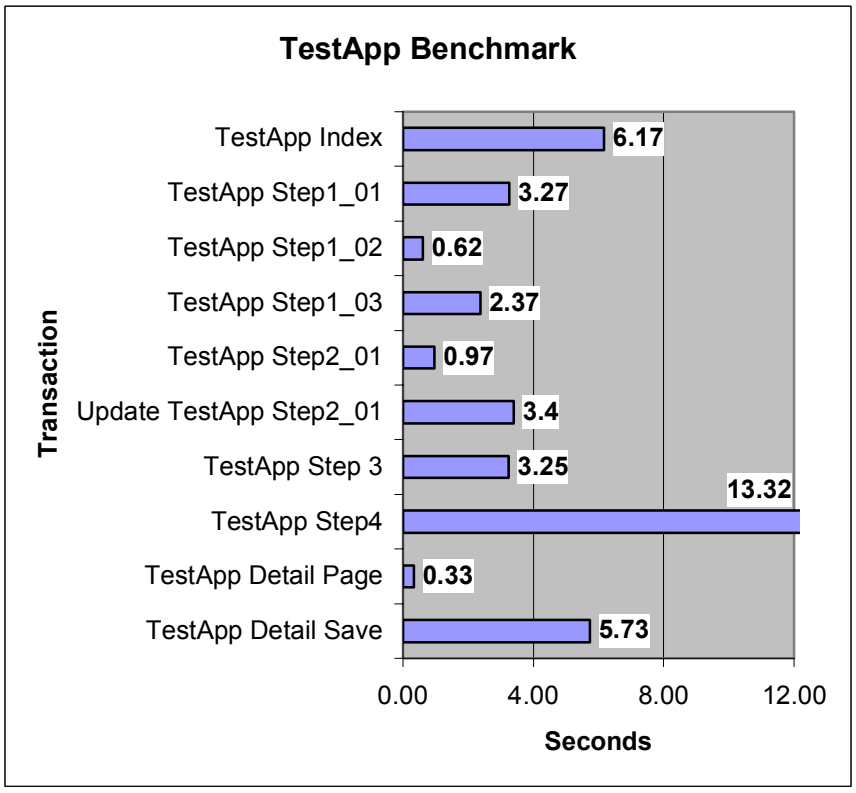


Figure 6.1 <sample>

7 OTHER SCHEDULED TEST RESULTS

7.1 Scheduled Tests

The Execute Scheduled Tests aspect includes those activities that are mandatory to validate the performance of the system. These tests need to be executed even if no performance issues are detected and no tuning is required. There are only two activities that can be conducted in this aspect. They are

- Execute User Experience Tests
- Execute Stability Tests
- Execute Production Validation Tests

All of the tests described below are considered to be Scheduled tests no matter how many times they are executed due to development or tuning iterations.

7.1.1 User Experience Tests

User Experience Tests constitute what are considered to be expected real-world loads, from best case to worst case. Applying less than the expected worst-case load is useful in identifying major failings in a system, but does so in a way that doesn't highlight many of the more minor failings, allowing an easier analysis of results. When the load is equivalent to the expected real-world worst-case load, actual performance of the system can be measured, and associated problems can be clearly identified. Executing Load Tests before moving to other types of testing allows for the more major problems of a system to be identified and corrected separately prior to the smaller issues.

Upon completing benchmarks, load tests were executed according to the Workload Distribution models outlined in section 4 of this document. These tests were designed to validate that the performance goals and requirements outlined in section 3 have been met. The results reported here represent the actual performance of the system upon conclusion of the Performance Engineering effort.

A summary of bottlenecks and tuning efforts is included in section 8 of this document.

Each module will be tested with up to their individual target load initially. When each individual module meets the acceptance criteria, the modules will be testing together at loads of 50, 100, 150, 200 and 250 virtual users.

7.1.1.1 System Architecture

The user experience tests environment is a shared environment consisting of:
Load Balancer(s) - <*describe*>
Proxy – <*describe*>
Web Server(s) - <*describe*>
Appserver(s) - <*describe*>
Database - <*describe*>

7.1.2 User Experience Test Results

Virtual users were gradually released into the system according to the arrival rate and duration of stay values documented in the Performance Engineering Strategy. Once the ramp up period was completed, each scenario iterated several times for a total of approximately an hour of relatively consistent load, then allowed to exit the system as the scenario completes for a total of roughly 90 minutes of load. User think times (time between user interactions with the system) were varied as described in the Performance Engineering Strategy document. This type of distribution, when spread across many users (statistically 100 or more) most accurately represents actual web-site activity. The page load times were measured in the same manner as they were under the Benchmark scenario to ensure consistency and validity between tests. Average times and 95th percentile times have been reported as well as standard deviations by page to ensure statistical validity of tests. For this test all stakeholder side caching was disabled, and each page was pre-compiled.

7.1.2.1 Collective User Experience Results

Screen Name	Timer Name	Number of Measurements	Avg time (sec)	Standard Deviation (sec)	95th Percentile time (sec)
Group Summary	tmr_grp_sum				
Employee Listing	tmr_emp_lst				
Search Results View	tmr_srch_rslt				
Customer Adjustment Popup	tmr_cust_adj				
Term Confirmation Popup	tmr_term_conf				
Coverage Type View	tmr_cov_type				
Adjustment – Member View	tmr_adj_memb				
FAQ	tmr_faq				

Table 7.1 <sample>

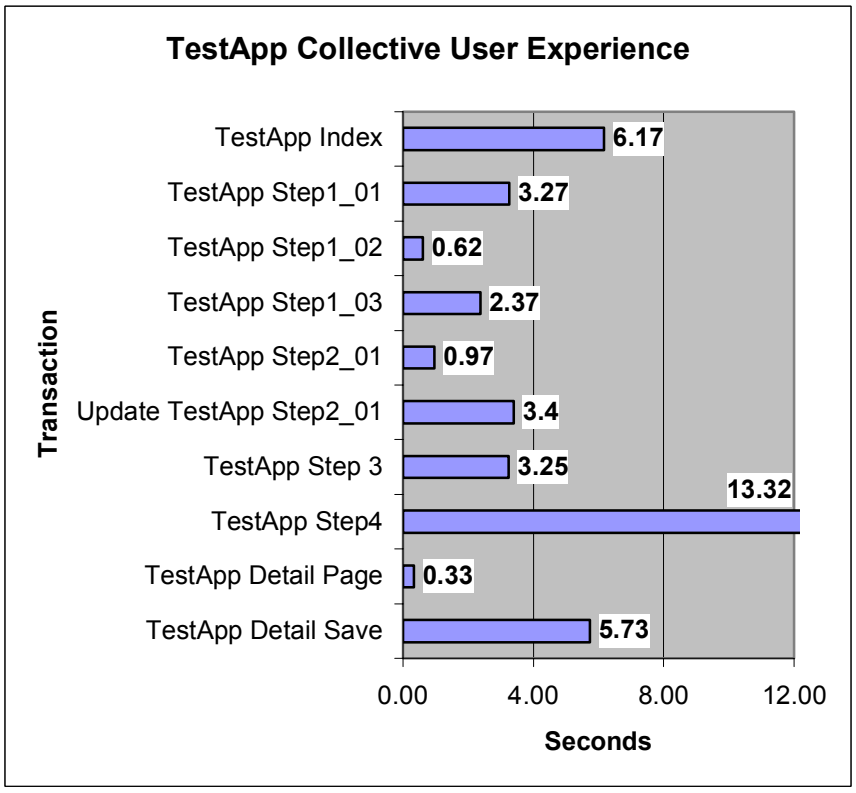


Figure 7.1 <sample>

7.1.3 Common Tasks Tests

<The first tests to be conducted weres not transaction specific. This test ensured that 100 users can log into the system in a 10 minute period and traverse the <application> navigation tree with no performance issues. This far exceeded any expected system load on the system.>

7.1.3.1 Common Tasks Results

Screen Name	Timer Name	Number of Measurements	Avg time (sec)	Standard Deviation (sec)	95th Percentile time (sec)
Group Summary	tmr_grp_sum				
Employee Listing	tmr_emp_lst				
Search Results View	tmr_srch_rslt				
FAQ	tmr_faq				

Table 7.9 <sample>

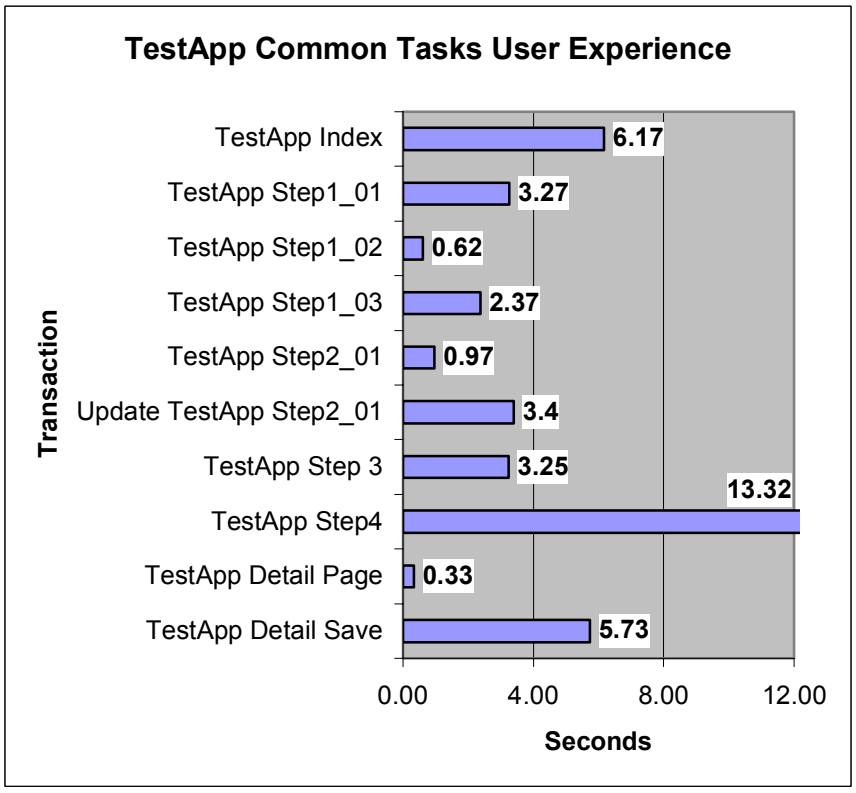


Figure 7.9 <sample>

7.1.4 Remote Location Tests

<The <application> will have geographically remote users. After scheduled tests were conducted locally and are determined to be generally acceptable, remote users were asked to test the performance of the system manually. This is not a formal test, but rather a sanity check. >

7.1.4.1 Remote Location Test Results

Screen Name	Timer Name	Number of Measurements	Avg time (sec)	Standard Deviation (sec)	95th Percentile time (sec)
Group Summary	tmr_grp_sum				
Employee Listing	tmr_emp_lst				
Search Results View	tmr_srch_rslt				
FAQ	tmr_faq				

Table 7.11 <sample>

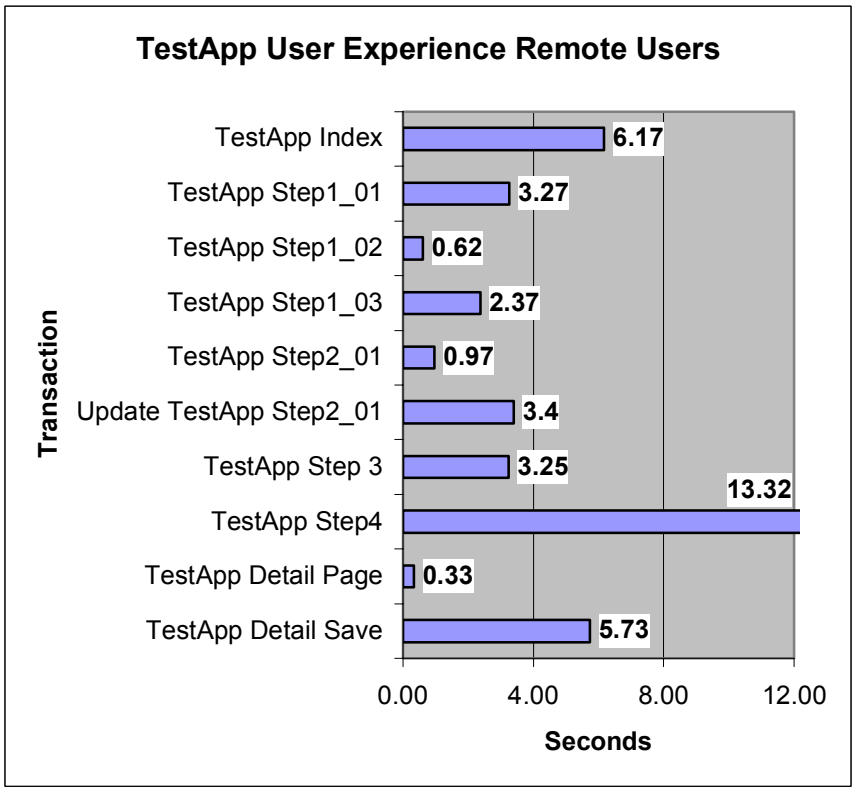


Figure 7.11 <sample>

7.1.5 Stability Tests

Stability scenarios test a system at and beyond the worse expected demand it is likely to face. The majority of critical deficiencies in the system will have already been identified during the execution of load tests, so this phase deals more with assessing the impact on performance and functionality under a heavy or unreasonable load. Stability scenarios will also identify many other system bottlenecks not previously noticed, which may in fact be partially responsible for earlier identified problems.

Heavy load scenarios are generally designed to be far more than a system can handle. They are used not to identify ***if*** a system fails, but ***where*** it fails first, how badly, and why. By answering the *why* question, it can be determined whether a system is as stable as it needs to be.

The analysis performed in this phase can vary depending on the exact goals and objectives of the load testing. Results obtained from the test automation tool are often used in conjunction with results obtained directly from the system, where white-box testing has been employed. These tests are designed to find more subtle performance issues, such as memory leaks, caching and database locking but are not designed to collect end-to-end user experience measurements in most cases.

7.1.5.1 Stress Tests

Stress Tests are tests that use real-world distributions and user communities, but under extreme conditions. It is common to execute stress tests that are 100% of expected peak

expected user-load sustained over 8-12 hours, and 150% expected peak user-load with normal ramp up and ramp down time.

7.1.5.1.1 Stress Test Descriptions

7.1.5.1.2 Stress Test Results

7.1.5.2 Spike Tests

Spike Tests are tests that used real-world distributions and user communities, but under extremely fast ramp up and ramp down times. It is common to execute stress tests that ramp up to 100% or 150% of expected peak expected user-load in a matter of minutes rather than about an hour.

7.1.5.2.1 Spike Test Descriptions

7.1.5.2.2 Spike Test Results

7.1.5.3 Hammer Tests

Hammer Tests have little or no resemblance to real-world distributions and user communities. These test take all existing load generation scripts and methods, eliminate user think times and increase load until something stops functioning properly. These tests are designed to make the system fail. Once failures occur, it can be decided how to mitigate the risk of that particular failure.

7.1.5.3.1 Hammer Test Descriptions

7.1.5.3.2 Hammer Test Results

7.1.6 Batch Baselines

Batch process testing will be treated as a validation exercise unless/until any batch process does not complete within its allotted time window during validation. Batch processes will be launched manually and timed. Any batch that completes in under 90% of its allotted time window will be considered acceptable. Any batches that do not complete within 90% of their time window will be evaluated, by component, and iteratively tuned to the greatest degree possible.

7.1.7 Production Validation Tests

Production Validation is conducted to ensure that the application performs as expected in the production environment after testing and tuning have been completed in a test environment.

Upon completing the tuning effort, a load test (250 concurrent users) will be executed according to the Workload Distribution model outlined in Section 3. This will take place in the production environment between deployment and going live. This test is designed to validate that the performance results obtained in the **<*??????*>** environment are also achieved in the production environment.

7.2 Exploratory (Specialty) Tests

Exploratory Tests are used to expose suspect bottlenecks or isolate performance concern areas that are to be tuned.

7.2.1 Concern/Issue 1

7.2.1.1 Description
7.2.1.2 Findings

7.2.2 Concern/Issue 2

7.2.2.1 Description
7.2.2.2 Findings

8 CONCLUSIONS AND RECOMENDATIONS

8.1 Consolidated Results

<The following is an example to show how conclusions are best displayed. In this section, a summary narrative should be inserted.

Summary Comparison								
Statistic/ Concurrent Users	LAN					128 kbs	56.6 kbs	28.8 kbs
	1	50	100	150	200	100	100	100
Times Recorded .	177	175	175	176	176	176	169	169
Times > Goal .	31	15	43	43	170	98	122	136
% Times > Goal .	17.5%	8.6%	24.6%	24.4%	96.6%	55.7%	72.2%	80.5%
Times > Max .	4	10	23	24	160	35	106	123
% Times > Max .	2.3%	5.7%	13.1%	13.6%	90.9%	19.9%	62.7%	72.8%
Typical Average Time By page .	1.23	1.44	1.70	1.61	30.63	6.49	6.15	16.10
Typical 95th Percentile Time By page .	3.42	2.98	4.67	4.06	53.51	7.68	8.74	17.72

Figure 8.1 – Summary Comparison

The chart below is a summary of some of the data from the chart above. The blue line is the total number of pages timed during the tests. Red is the number that responded faster than the stated maximum. Yellow is the number that responded faster than the stated goal. In general LAN users met the goal up to 150, and 128 and 56.6 kbs users did fair – particularly when the typical 95th Percentile Time is compared to the goal.

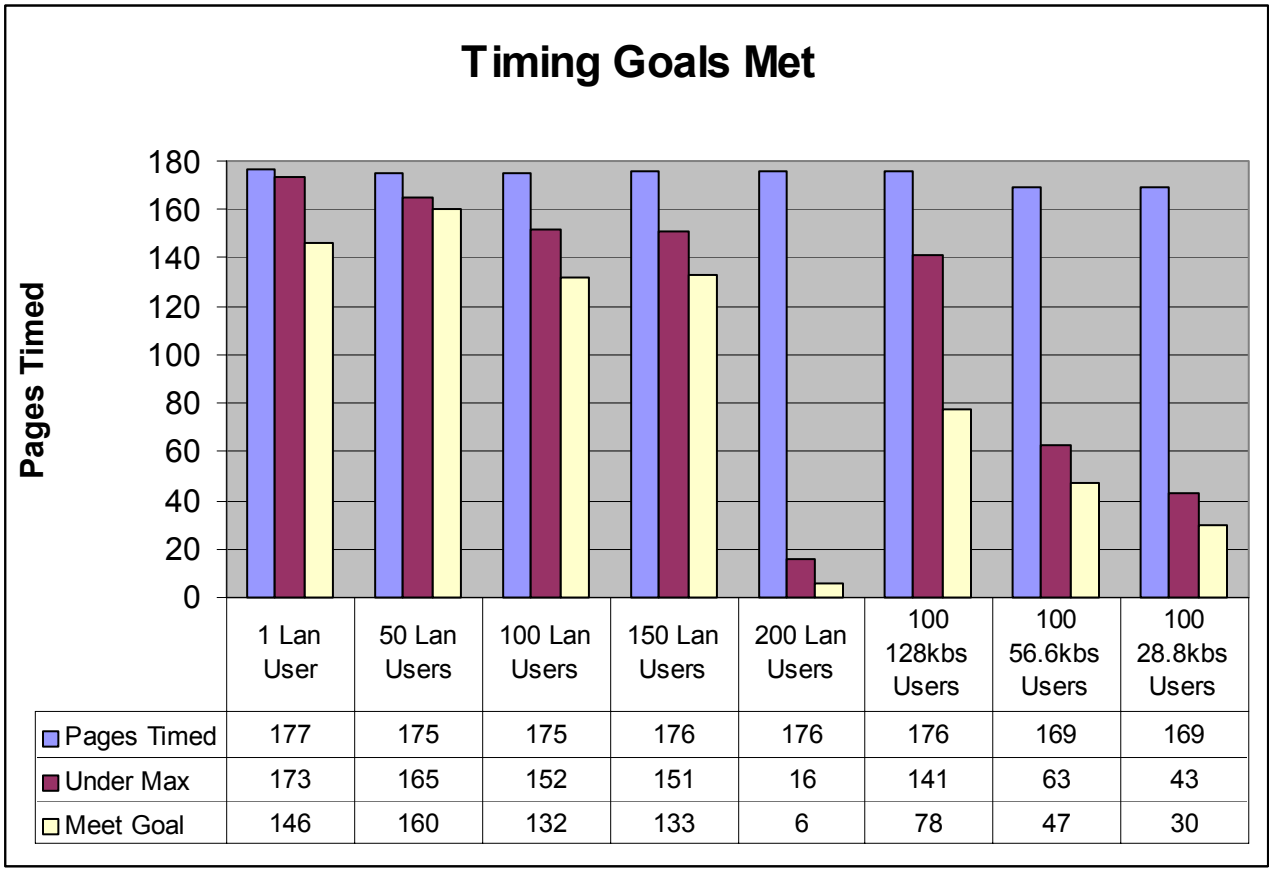


Figure 5.2 – Timing Goals Achieved

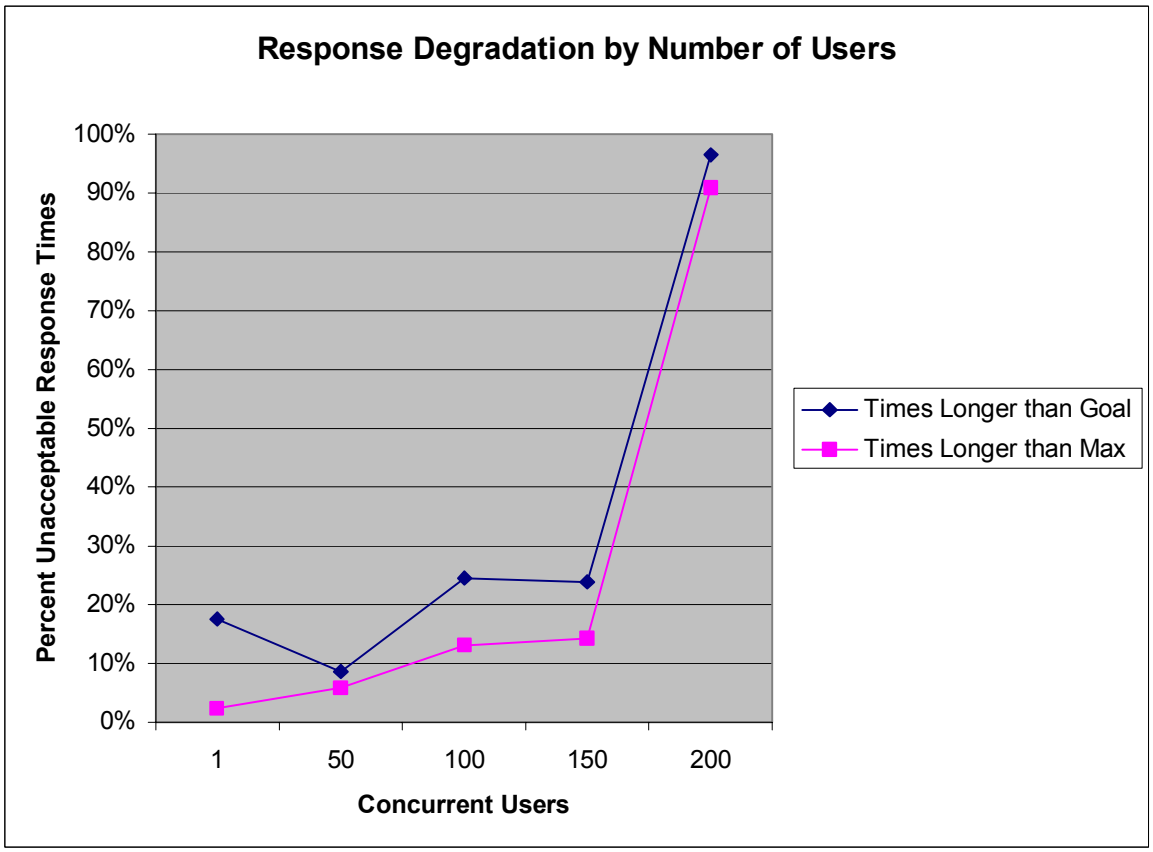
>

8.2 Tuning Summary

<This section should include a brief summary of any tuning that was required and performed or any suggested tuning for future versions.>

8.3 Conclusions

<This section should be a brief recap of what criteria were met and what were not. It may also include a graph like the one below to show when performance begins to degrade exponentially, rather than gracefully:



>

8.4 Recommendations

<This section should detail all recommendations, such as whether or not to go live with the system, whether future testing, tuning, upgrades are required, etc.>