|  |
| --- |
| Application Name-Version |
| Performance Test Plan |
| Click here to enter a date. |

Table of Contents

[Revision History i](#_Toc491792059)

[Introduction 1](#_Toc491792060)

[What is Performance Testing? 1](#_Toc491792061)

[What Types of Testing Will Be Performed? 1](#_Toc491792062)

[Functional Benchmark testing 1](#_Toc491792063)

[Load / Concurrency Testing 2](#_Toc491792064)

[Soak Testing 2](#_Toc491792065)

[Introduction To The Application 3](#_Toc491792066)

[What is Application Name-Version? 3](#_Toc491792067)

[Application Goals For This Testing Event 3](#_Toc491792068)

[Testing Environment Considerations 3](#_Toc491792069)

[Functional Benchmark Testing 4](#_Toc491792070)

[Goals 4](#_Toc491792071)

[Methodology 4](#_Toc491792072)

[Load / Concurrency Testing 5](#_Toc491792073)

[Goals 5](#_Toc491792074)

[Methodology 5](#_Toc491792075)

[Soak Testing 7](#_Toc491792076)

[Goals 7](#_Toc491792077)

[Methodology 7](#_Toc491792078)

[Server Performance Monitoring 8](#_Toc491792079)

[Web and Application Server Monitoring 8](#_Toc491792080)

[Database Server Monitoring 8](#_Toc491792081)

[Evaluating The Results 9](#_Toc491792082)

[Functional, Load and Soak Test Results 9](#_Toc491792083)

[Server Performance Monitoring Testing Results 9](#_Toc491792084)

[Testing Tools 10](#_Toc491792085)

[Feature Setup 11](#_Toc491792086)

[Test Cases To Perform 11](#_Toc491792087)

[Performance Test Case 1 11](#_Toc491792088)

[Performance Test Case 2 12](#_Toc491792089)

[Environment Consideration 13](#_Toc491792090)

[Testing to be Executed in - Choose an item. 13](#_Toc491792091)

[Production Environment - For Comparison – Not Testing 13](#_Toc491792092)

[Ticket Process 14](#_Toc491792093)

[Team Members and Responsibilities 15](#_Toc491792094)

[Performance Testing Schedule 16](#_Toc491792095)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Additions / Modifications** | **Prepared / Revised By** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

This document outlines the Performance Test plan for Application Name-Version and will describe the testing strategies and approach performed by the Mercer Performance Engineers.

## What is Performance Testing?

Performance Testing is the act of placing hardware counters on a system while having virtual users run a prerecorded “script” of real user events. This simulates a *load* on the system. Those counters are then set to monitor and record system performance, such as:

* CPU Usage
* Memory Usage
* Page Load Times

It’s important to note that Performance Testing is not intended to test the end-user experience, but to provide an estimation of how the hardware will perform in a live environment.

## What Types of Testing Will Be Performed?

There are many different types of tests in the Performance Testing portfolio. Each test has its purpose, but not all types of testing are required. To define which tests will be needed, we must first examine the goals.

* To create a baseline with which to compare future releases and upgrades
* To ensure the application system can maintain a certain level of users
* To check for memory leaks

### Functional Benchmark testing

Functional Benchmark Testing is used to determine the speed and effectiveness of the system in question. This type of analysis will typically occur with a single user and then with an expected average number of users. The user/s will be scripted to navigate through the major functionalities of the application without putting the system under a significant load. This event is typically performed to establish a baseline with which to compare future releases, to ensure the new code did not degrade the performance.

This type of benchmarking will occur at three times of the day (local to Performance engineer):

* Morning
* Mid-Day
* Evening

These series of assessments will typically be performed on three separate times. Average of three results will be included in the final Performance Test results documentation.

### Load / Concurrency Testing

Load/Concurrency Testing is designed to analyze how the system responds under an increasing load. This is typically done to ascertain the operating capacity of the system as well as identify any bottlenecks in the software and/or hardware. This portion of testing will typically occur with a base load of users and will increase at predefined intervals, until a maximum number of users are reached.

During this time, performance counters will be monitored and recorded and will be available in the Performance Test results documentation.

### Soak Testing

Soak Testing is the process of running a system at an unusually high load for a prolonged period of time. A typical Soak Testing event would execute predefined user events for a higher than average number of virtual users. Each script would be repeated continuously over the course of a 4 hour/8 hour/12hour/24 hour period depending on the application size and usage. This puts a burden on memory and is used to reveal any memory leaks within the system as well as any issues that might appear with prolonged usage of the application.

During this time, performance counters will be monitored and recorded and will be available in the Performance Test Results documentation.

# Introduction To The Application

This section will outline the basic functionality of the application being tested as well as any major updates. Likewise, goals and environment details are also listed.

## What is Application Name-Version?

Application Name-Version Enter description of the application here.-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Application Name-Version includes the following:

* [List items to be tested, i.e. All new code]
* [List items to be tested, i.e. Import position classes]
* [List items to be tested, i.e. Match multiple jobs]

### Application Goals For This Testing Event

Check the goals for this testing event

|  |  |
| --- | --- |
|  | Baseline the system for future comparison – Functional Benchmark Testing |
|  | Expose any obvious issues under a load – Load Testing |
|  | Expose any memory leaks – Soak Testing |

These goals have to be attainable by executing the types of performance testing listed on pages 1-2.

### Testing Environment Considerations

The Performance Testing environment can differ with the requirements of each test event. For this event, the environment will be:

|  |  |
| --- | --- |
| **Operating System** |  |
| **Browser Simulation** |  |
| **Load Test Tool** |  |
| **Maximum User Load** |  |
| **Environment** |  |
| **Running To Failure** |  |

# Functional Benchmark Testing

Functional Benchmark Testing will monitor the speed and effectiveness of the application. The defined user tests will be performed by a single user as well as an estimated site average number of users.

The primary sections of focus for this round will include:

* [List items, i.e. Site Navigation]
* [List items, i.e. Search Functionality]

## Goals

* Create a baseline performance numbers to compare future releases
* Give a general “feel” for how the site will function under an estimated average load
* Estimate site performance at multiple times throughout the day and week

## Methodology

Below outlines the primary methodology for the Mercer Functional Benchmark Testing. This includes the estimated dates and times for the analysis, the primary Performance Engineer, what will be included in the tests, and more.

|  |  |
| --- | --- |
| **Performance Test Engineer** |  |
| **Performance Test Tool** |  |
| **Number of Virtual Users** |  |
| **Loop Time For Max Users** |  |
| **Local Time Zone** |  |
| **Estimated Start Date** |  |
| **Environment** |  |
| **Operating System** |  |
| **Browser Simulation** |  |

This type of testing incorporates:

* Performing predefined user event scripts in a **single user** scenario
  + This script will only be ran through its entirety one time
* Performing predefined user event scripts (estimated average for this application) scenario
  + This script will loop for a predetermined amount of time
* Perform all benchmarks at three different times of the day, relative to local time
  + Morning
  + Mid-Day
  + Evening
* Perform the above tests, in full, on three separate days
  + The best results of each time of day will be given as the test findings

# Load / Concurrency Testing

Load Testing helps to determine how the system will perform under a give load. Performance test scripts will be created and Virtual users will be assigned based on the user distribution as per business need. User will be ramped up with defined intervals until the max user is reached. User keep performing the actions until the test is stopped or until the full duration of the test is completed.

The primary focus for this round of testing will include following scenarios,

* Site Navigation
* Search Functionality

Representatives from the Server Administrators team and the Database Administrators team are to be present during the testing to monitor servers and react in the event of hardware or server crashes occur or to notify the Performance Engineer in the event that a crash may be imminent. Should this occur the testing may be halted to assess the causes of the crash. At this point, all those involved in testing will decide whether the current testing event is to be continued or postponed to a later date.

## Goals

* Create a baseline of hardware utilization usage with which to measure and compare future releases
  + Measure hardware utilization under different load increments
* Monitor and report on errors for virtual users
* Manually verify that the site is functional under heavy usage

## Methodology

Below outlines the primary methodology for the Mercer Load / Concurrency Testing. This includes the estimated timeline for performing the test, the primary tester, what will be included in the tests, and more.

|  |  |
| --- | --- |
| **Performance Test Engineer** |  |
| **Performance Test Tool** |  |
| **Initial Number of Virtual Users** |  |
| **Maximum Number of Virtual Users** |  |
| **Step Time** |  |
| **Step Amount** |  |
| **Distribution of Test Cases** |  |
| **Local Time Zone** |  |
| **Estimated Start Date** |  |
| **Estimated Start Time** |  |
| **Environment** |  |
| **Operating System** |  |
| **Browser Simulation** |  |
| **User Increments To Measure** |  |
| **Servers To Monitor** |  |
| **Hardware Parameters To Measure** |  |
| **Software Parameters To Monitor** |  |

This type of testing incorporates:

* Performing the predefined user event scripts distributed evenly across an initial load of (Predefined no of users) users
  + Test scripts are repeated until the event is halted or concluded
  + The numbers of users are increased by every seconds\minutes until the (Predefined no of users) users mark has been met or until the event is halted
* Server administrator will monitor resource utilization of all affected servers web, app and db servers
* Front end monitoring from a live connection
  + This is monitored manually and continual verbal reports will be given during the call
* If the testing is in production it will take place during the MGTI maintenance window on a Saturday.
  + MGTI Maintenance Window:
    - Saturdays, 8:00 a.m. – 12:00 p.m. Eastern (unless otherwise noted)

# Soak Testing

Soak testing is used to identify issues that might arise with prolonged use of the system. Issues found here are most generally related to memory leaks. To perform this analysis, a set of users, usually greater than an expected average amount of users will perform the predefined user event test scripts, repeated over the course of 8/12/16/24 hours.

The primary focus for this round of testing will include the following scenarios,

* Site Navigation
* Search Functionality

## Goals

* Verify the application does not show signs of any memory leaks
* Verify any issue that would arise of prolonged usage of the system

## Methodology

Below outlines the primary methodology for the Mercer Soak Testing. This includes the estimated timeline for performing the test, the primary tester, what will be included in the tests, and more.

|  |  |
| --- | --- |
| **Performance Test Engineer** |  |
| **Performance Test Tool** |  |
| **Number of Users** |  |
| **Distribution of Test Cases** |  |
| **Local Time Zone** |  |
| **Estimated Start Date** |  |
| **Estimated Start Time** |  |
| **Estimated End Date** |  |
| **Estimated End Time** |  |
| **Environment** |  |
| **Operating System** |  |
| **Browser Simulation** |  |

This type of testing incorporates:

* Performing the predefined user event scripts distributed evenly across (Predefined no of) virtual users
  + Test scripts are repeated until the event is halted or it concluded in its entirety
  + Test event takes place over a prolonged period (4hour/8hour/12hour/24hour)
* RAM utilization monitoring on all affected servers
* This test will take place immediately following the Load/Concurrency test event

# Server Performance Monitoring

The process of server performance monitoring will track and record the overall health and performance of the servers for the duration of the Load and Soak Testing. CA tickets must be entered prior to the test event to enable the counters. The following information will be monitored and recording during each of the test events:

## Web and Application Server Monitoring

* CPU utilization
* Average physical memory usage
* Additional hardware metric monitoring maybe requested

## Database Server Monitoring

* CPU utilization
* Average physical memory usage
* Additional hardware metric monitoring maybe requested

# Evaluating The Results

This section will give an overview of how the results will be evaluated and disseminated. All results will be compiled in a report by the Performance Engineer and will be shared with all the stake holders.

## Functional, Load and Soak Test Results

Once each testing is completed the results are shared with all the stake holders. If any issue arise responsible parties in the below table will be notified.

* Performance bottlenecks
* Poorly performing web pages, URL’s and/or transactions
* Failures
* Coding problems
* Memory leaks
* Exceeding thresholds

**Responsible parties:**

|  |  |
| --- | --- |
| **Business Product Manager** |  |
| **Technical Project Manager** |  |
| **Project Manager** |  |
| **QA Manager** |  |
| **Release Manager** |  |

## Server Performance Monitoring Testing Results

If the monitoring and recording of server performance show any of the follow (but not limited to), a member of the Performance team will notify the responsible parties to escalate the issues and determine what, if any, action is necessary.

* Unexpected increase in resource usage
* Nearing of a threshold – long term predictability curve
* Limited disk space

**Responsible parties:**

|  |  |
| --- | --- |
| **Business Product Manager** |  |
| **Technical Project Manager** |  |
| **Project Manager** |  |
| **QA Manager** |  |
| **Release Manager** |  |
| **Enterprise Architect** |  |
| **Database Administrator** |  |

# Testing Tools

This section will list the currently available and preferred tools for use in Performance Testing

**Performance Testing Tools**

* + 1. Visual Studio 2015
    2. Visual Studio Online (Cloud based performance Testing)
    3. Performance Center 12.5

Above will be used to script test cases and create virtual users and to execute all automated Performance Tests.

**Server Monitoring Tools**

* + 1. eHealth
    2. Perfmon
    3. vRops
    4. App Dynamics
    5. Splunk

Above are used tomonitor server performance and utilization during the performance testing.

# Feature Setup

This section will outline the features intended to be tested during the test events as well as user creation, data that needs to be setup and test cases to be executed.

## Test Cases To Perform

Once the above is in place, the Performance Engineer will begin recording scripts and creating any necessary data pools to test the functionality of the system.

*Below is an example of test cases that need to be created for our testing. It will vary based on application and testing needs.*

### Performance Test Case 1

This test case will verify the following:

* The “Home” page
* The Left-hand menus
* The “What’s New” section
* The “Health” page

|  |  |
| --- | --- |
| **Test Step** | **Times to Capture** |
| Navigate to the BELONG site: <http://belong2013.mercer.com> | PTC01a |
| Click on the **Manage Your 401 (k)** link in the left-hand **Tools** menu | PTC01b |
| Navigate back to the BELONG site **HOME** page by using the **back** button on the browser |  |
| Click on the **Find a Doctor** link in the left-hand **Tools** menu | PTC01c |
| Navigate back to the BELONG site **HOME** page by using the **back** button on the browser |  |
| Click on the **Vision Benefits** link in the left-hand **Tools** menu | PTC01d |
| Navigate back to the BELONG site **HOME** page by using the **back** button on the browser |  |
| Click on the **Prescription Benefits** link in the left-hand **Tools** menu | PTC01e |
| Navigate back to the BELONG site **HOME** page by using the **back** button on the browser |  |
| Click on **Welcome to Belong 2013!** link in the **What’s New** section (if available) | PTC01f |
| Navigate back to the BELONG site **HOME** page by using the **back** button on the browser |  |
| Click on the **HEALTH** tab | PTC01g |
| Navigate back to the BELONG site **HOME** page by using the **back** button on the browser |  |
| Click on the **Medical Claims Guide** link | PTC01h |
| Click on the **Manage Your 401 (k)** link | PTC01i |
| Click on the **2014 FSA/HAS Guide** link | PTC01j |

### Performance Test Case 2

This test case will verify the following:

* The “What If I” section
* The “Money” page

|  |  |
| --- | --- |
| **Test Step** | **Times to Capture** |
| Navigate to the BELONG site: <http://belong2013.mercer.com> |  |
| Click on the **HOME** tab |  |
| Click on the **Add a Spouse?** link | PTC02a |
| Navigate back to the BELONG site **HOME** page by clicking the **HOME** tab |  |
| Click on the **Have a Baby?** link | PTC02b |
| Navigate back to the BELONG site **HOME** page by clicking the **HOME** tab |  |
| Click on the **Relocate to Europe?** link | PTC02c |
| Navigate back to the BELONG site **HOME** page by clicking the **HOME** tab |  |
| Click on the **Have to Take Care of a Family Member?** link | PTC02d |
| Navigate back to the BELONG site **HOME** page by clicking the **HOME** tab |  |
| Click on the **Need Personal Time Off?** link | PTC02e |
| Click on the **MONEY** tab | PTC02f |

# Environment Consideration

This section will outline the environment expected for this release.

## Testing to be Executed in - Choose an item.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **HARDWARE IN Test Environment** | | | | |  |  |  |
| Site URL to Test Environment: https:// | | | | | | | |
| Server Function\Tier | IP Address | Machine Name | VM or Physical | Shared Yes\No | Memory | # of Processors | Other |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

## Production Environment - For Comparison – Not Testing

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **HARDWARE IN PRODUCTION** | | | | |  |  |  |
| Site URL to Production: https:// | | | | | | | |
| Server Function\Tier | IP Address | Machine Name | VM or Physical | Shared Yes\No | Memory | # of Processors | Other |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

# Ticket Process

The following steps will be followed in the event of finding an issue during the testing process.

In case of issue found during the performance testing are related to serve configuration and hardware infrastructure, most of the time tuning work is done by IT and DBA teams and resetting is done.

If the issue is related to application functionality, respective function QA team will be notified and Functional QA will create a defect in their respective defect tracking tool (TFS, QC etc.) and track it.

# Team Members and Responsibilities

The below list is a high-level overview of team members and their responsibilities. This does not include any additional items discussed in other sections of this document.

|  |  |  |
| --- | --- | --- |
| **Role** | **Responsibility** | **Assigned** |
| IAP Team – Performance Engineer | * Create Performance Test Plan * Provide input on scenarios to be selected for Performance Testing * Run Load and Soak Testing * Research unexpected results * Analyze and report on performance results |  |
| QA Manager | * Facilitate test period with performance team * Provide input on Performance Test Plan * Facilitate weekly status meeting * Review test results |  |
| Solutions Architect | * Provide assistance in resolving issues pertaining to system architecture * Provide information on changes to the architecture and how they may affect the performance of the application * Provide input on selecting Standard Users Events for Performance Testing |  |
| Database Administrator | * Monitor and analyze servers during performance testing * Provide assistance when there is negative findings of monitoring and analysis |  |
| Tech. Ops. | * Monitor servers during performance testing * Provide assistance when there is negative findings of monitoring and analysis |  |
| Business Product Manager | * Provide input on Scenarios to be selected for Performance Testing * Provide assistance when there is negative finding of monitoring and analysis |  |
| Project Manager | * Coordinate with app owners, TPM/Production Managers on test dates * Alert app owners, TPM/Production Managers to any performance degradations |  |

# Performance Testing Schedule

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Tasks** | **Responsible Party** | **Status** |
|  | Have meeting with Application PM, QA Manager and Director of QA to determine testing needs. |  |  |
|  |  |  |  |
|  | Get sample scripts and URL to verify application is compatible with Testing Tool. |  |  |
|  |  |  |  |
|  | Gather all the data needed to create the Test Plan. |  |  |
|  |  |  |  |
|  | Verify Testing Environment has been created and confirm date build to test will be deployed. Will this fit in our schedule? |  |  |
|  |  |  |  |
|  | Obtain Test Scripts from PM\QA Manager. |  |  |
|  |  |  |  |
|  | Verify all hardware to monitor has been discovered by eHealth and can be monitored. |  |  |
|  |  |  |  |
|  | Prepare Test Plan. |  |  |
|  |  |  |  |
|  | Meet with PM and QA Manager to review Test Plan. |  |  |
|  |  |  |  |
|  | Set up data and record performance test scripts. |  |  |
|  |  |  |  |
|  | For Load Testing enter in tickets to have servers monitored. One ticket for Web\App\etc. servers a second ticket for DB to be monitored and to have them enable the counters. |  |  |
|  |  |  |  |
|  | Send out meeting invites to all involved for Saturday load testing. Confirm all necessary parties are to attend. |  |  |
|  |  |  |  |
|  | Get authorization to load test from PMs of all applications involved if these are shared servers. Testing my degrade application performance. |  |  |
|  |  |  |  |
|  | Run Functional Benchmark Testing |  |  |
|  |  |  |  |
|  | Run Load Testing |  |  |
|  |  |  |  |
|  | Run Soak Testing |  |  |
|  |  |  |  |
|  | Gather and analyze test results |  |  |
|  |  |  |  |
|  | Prepare Performance Test Summary |  |  |
|  |  |  |  |
|  | Review results with Dev and QA Managers to sign off on distribution of the report |  |  |
|  |  |  |  |
|  | Send out Performance Test Summary |  |  |
|  |  |  |  |
|  | Go-no-go meeting |  |  |
|  |  |  |  |
|  | Deploy to Production |  |  |