**Report on results of performance testing: Task10 – Perform stability test**

**Purpose**

Stability testing involves testing a system with a significant load extended over a significant period, to discover how the system behaves under sustained use. For example, in software testing, a system may behave exactly as expected when tested for 1 hour but when the same system is tested for 3 hours, problems such as memory leaks cause the system to fail or behave randomly.

The goal is to discover how the system behaves under sustained use. That is, to ensure that the throughput and/or response times after some long period of sustained activity are as good or better than at the beginning of the test.

It is basically used to check the memory leaks.

**Environment**

Virtual machine emulated with Oracle VM VirtualBox Manager Version 6.0.10:

|  |  |
| --- | --- |
| OS | Windows 10 |
| Base Memory | 6144 MB |
| Processors | 2 |
| Acceleration | VT-x/AMD-V, Nested Paging, PAE/NX, Hyper-V Paravirtualization |
| Attached to | Bridget Adapter |
| Adapter Type | Intel PRO/1000 MT Desktop (82540EM) |
| Promiscuous Mode | Deny |
| MAC Address | 0800272FEB0A |

Host Machine:

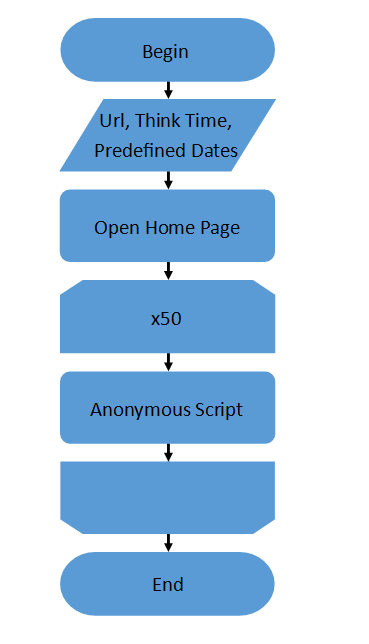
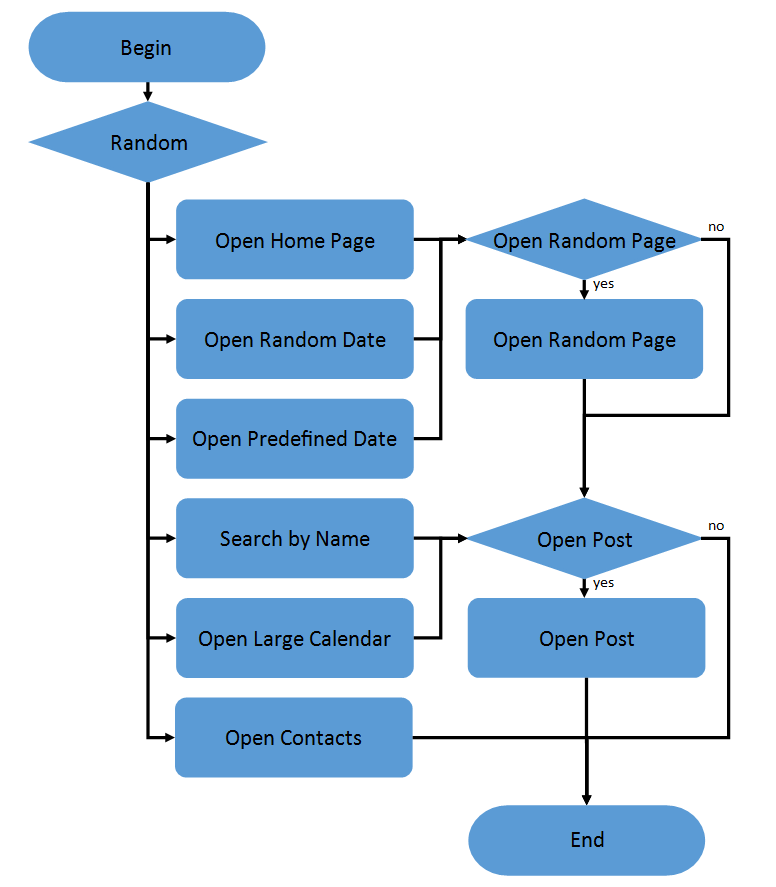
|  |  |
| --- | --- |
| Processor | Intel(R) Core (TM) i7-8700 CPU @ 3.20GHz 3.19 GHz |
| Installed memory(RAM) | 32.0 GM (31.7 GB usable) |
| System Type | 64-bit Operating System, x64-based processor |

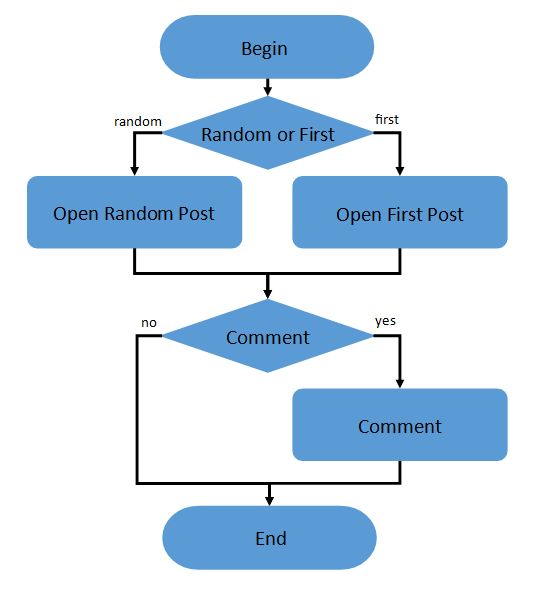
Parameters:

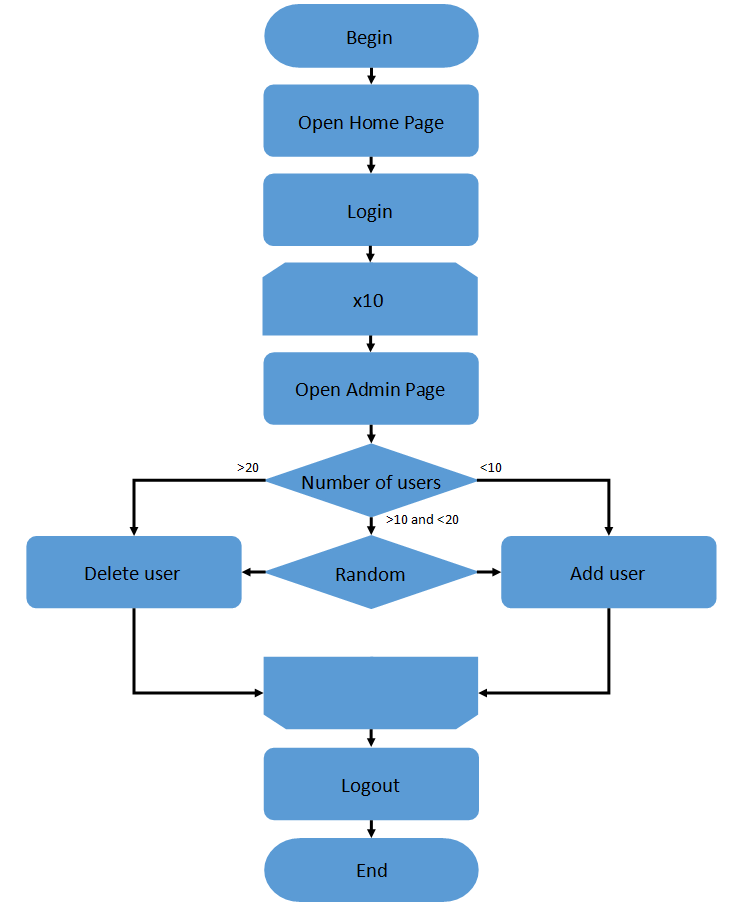
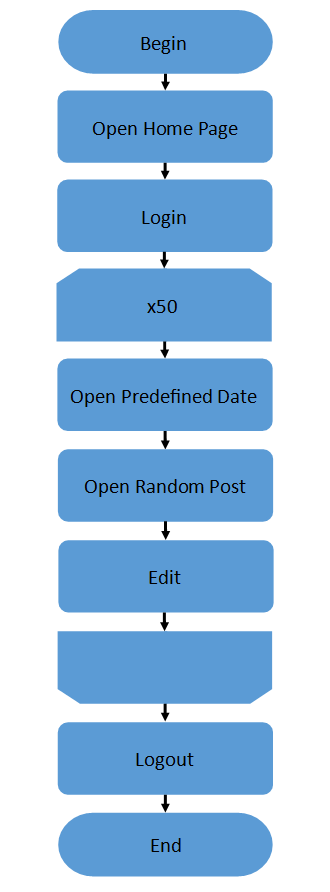
|  |  |
| --- | --- |
| Server Name | 10.66.154.88/blog |
| Protocol | http |
| Timer delay | 3000 ms |
| Timer deviation | 1000 ms |

**Load Model**

**Anonymous script:**

****

**Admin script: Editor script:**

**** 

|  |  |
| --- | --- |
| Number of anonymous users | 16 (40 % from maximum load) |
| Number of editor users | 1 |
| Number of admin users | 1 |
| Ramp-Up-Period | 2.5 minutes |
| Loop Count | Forever |
| Duration | 12 hours |

**Results**

**With the optimal configuration, the system has been able to operate stably for 5 hours. However, there was another CPU jump at the end and the application stopped responding. After identifying the server metrics, I decided to start looking for a problem with the CPU. This was prompted by a significant increase in the number of context switches.**

**Soon it turned out that during the test the number of context toggles increased much more than the 30K recommended for two cores. Without finding the answer to the available metrics, I decided to take a look at the system status after the failure. I found that all the time after the crash, the system handles 10 requests, in 10 threads with 10 connections. In addition, I found that the specific system call that is running all this time is ntdll!RtlUserThreadStart, which as I noticed after reading the forums there are many problems associated with high CPU. I also found some useful tools with the help of which I was able to analyze the process and get stack trace of threads which can't process the request.**

**I couldn't get a superficial look at what caused the problem, but it seems to me that the problem is exactly that the application is running on a virtual machine, which means that the cause of the failure may be the peculiarities of the virtual machine's interaction with cores. Looking at memory usage as well as disk operations, I can assume that the failures are not related to them, because you can't see catastrophic degradation on the charts, and even after I've experimented, I've come to the point where failures occur regardless of the flow scenario and the load used, the load only affects how fast it happens.**

**I haven't spent any more time figuring out the stack trace, so I attach the analysis to the report.**

****

**As for requests, I showed requests that have degraded. This may or may not be due to the increase in the number of comments that are generated in the script.**

