# Advantest Assignment

# Document Information

|  |  |
| --- | --- |
| Author(S) | Shashwat Anand |
| Version | 1.0 |
| Date | 10th March 2019 |

# 1. Business Summary

## 

## 1.1 Objective

Designing a tester tool which can read test suites and execute them on test system depending on required OS, devices and report the status of execution.

## 1.2 Design Abstract

As part of this requirement, following things has to be implemented: -

1. Retrieve the stored test suite files from shared drive (for assignment I am reading from folder)
2. Ideally tester tool which read and figure out the required OS and devices need for execution of the test suite (for POC I kept the name of files as TSuite\_mac\_D1\_D2\_19, where test suite is “TSuite19”, required OS is “mac”, D1 and D2 are devices which is needed to execute the test suite and 19 is the execution time of it.
3. When user passes the test suite file name as argument then tester tool finds the its required OS, needed devices.
4. Then tester tool finds the test system which is having same OS and devices and currently not in use. Ideally this information can be stored in some data base and retrieved (for POC I didn’t created data base schema or entities due to time constraints, I just tried to model the test system data with POJO and keep it in in-memory data structure.
5. Once tester tool finds the test system the it should start the execution of test suite and return to process other test suite which is passed as program arguments to it.
6. Each test suite execution should report its execution status on completion to the user who has started it. To retrieve email is ideally we should read the user name of from system variables and query LDAP (active directory) or company database to find user email and send the report to him/her (here in POC is retrieved the user name used hard code email id to send the report)

# 2. System

# 2.1 System Configuration

Below is my system configuration.

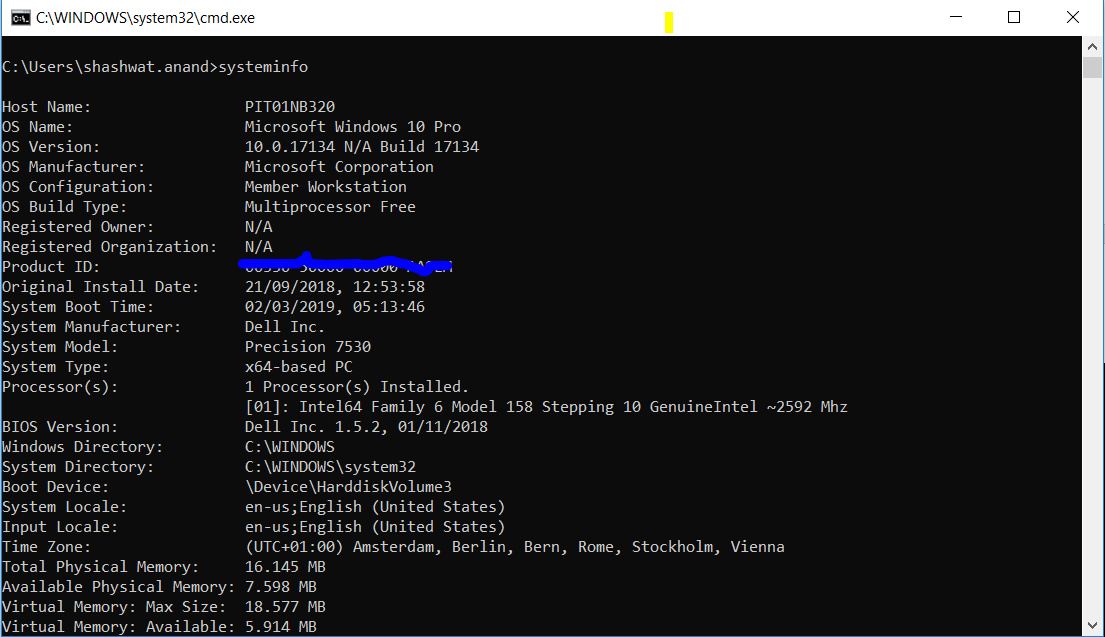


Figure : System Information

# 2.2 Design

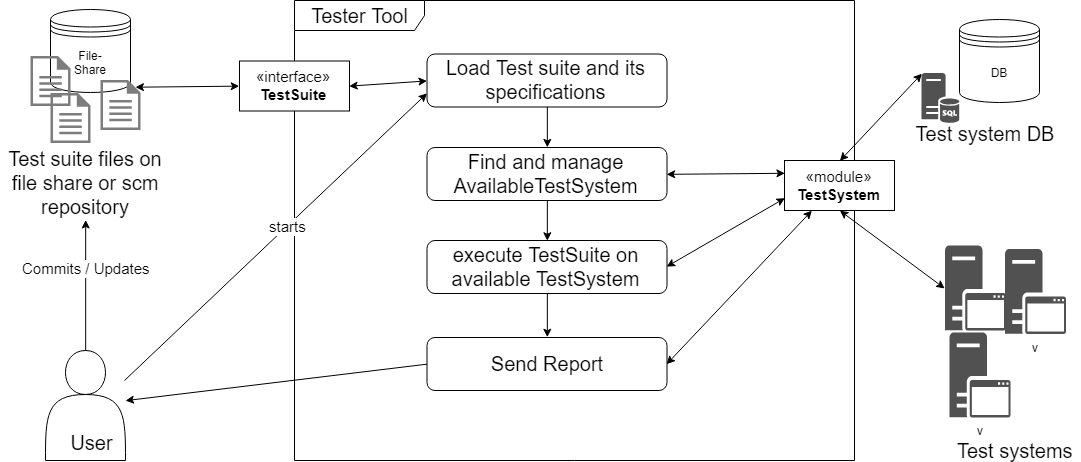


Figure : Architecture Diagram of system

As shown in figure, different modules of the system

* Tester tool
* Test System Database
* Test suite file share or repository

Once user interact with tester tool and gives command to execute the test suite then system should check if test suite name provided by user exists in repository if yes then load it and find the required OS and device needed.

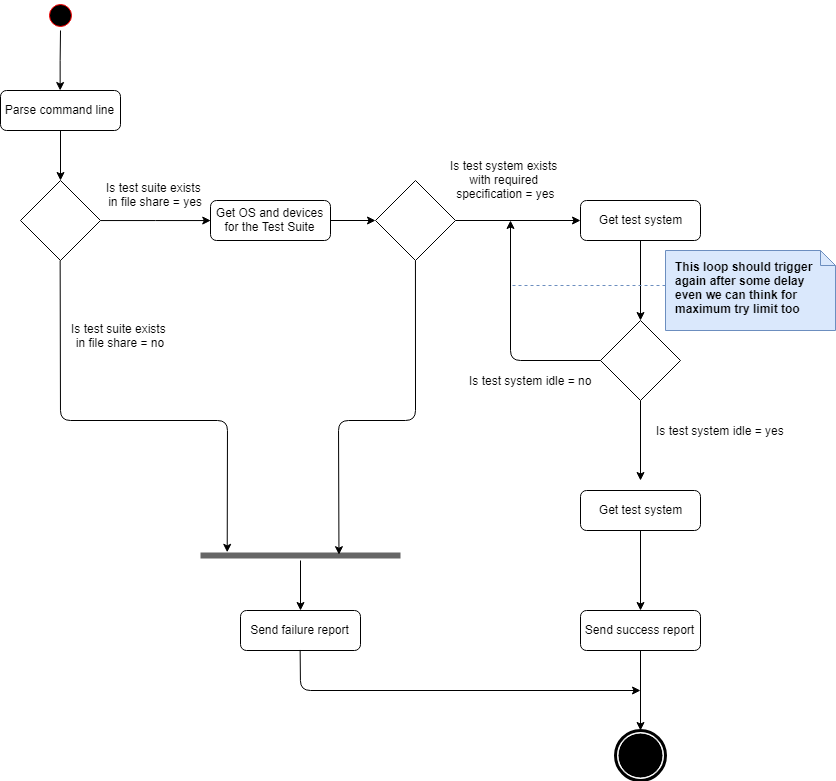


Figure : Process Diagram

With this information tester tool query the database for test system with available system exists of not. Once it found the list of list of test system with needed configuration then it checks the which test system is idle so test suite can be executed on the found test system.

Once test suite execution starts then tester tool should start processing other test suite.

Each test suite execution should report the status of it at the end to user who has initiated the command.

This is explained in above in figure 3 in abstract business process diagram of whole system.

# 2.3 Database design for Test System

Test System table

|  |  |  |
| --- | --- | --- |
| Test System ID | Name | Supported OS |
|  |  |  |

Device table

|  |  |
| --- | --- |
| Device ID | Device Name |
|  |  |

Test System Device relation table

|  |  |  |
| --- | --- | --- |
| Device Table ID | Test System ID | Device ID |
|  |  |  |

APIs needed from test system database CURD operation if Tester tool user can do those operations

* createTestSystem
* updateTestSystem
* deleteTestSystem
* createDevice
* deleteDevice
* updateDevice
* createRelationBetweenTestSystemAndDevice
* removeRelationBetweenTestSystemAndDevice

Below APIs for query operations of Tester tool

* findAllTestSystem
* findTestSystemByName
* findAllDevices
* findDeviceByName
* findDevicesByTestSystem
* findAvaiableTestSystemByOSDevices(OS, DeviceList)

Above API is designed only for minimalistic table. Test System and Device table can many more attributes.

# 2.4 Report Module

Below is sequence diagram for reporting module (although ideally I want to retrieve email and other details of user from LDAP by for POC I used hardcode email)

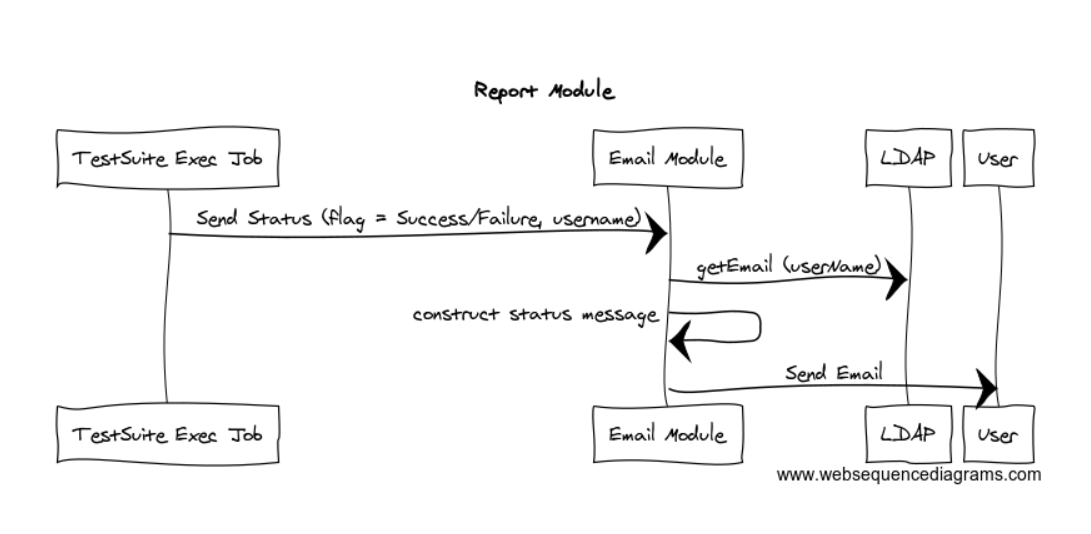


Figure : Report Module Sequence Diagram

# 3. Testing of Tester Tool

# 3.1 Unit Test

Below is list for few test cases

1. Test case to check if test system database is reachable.
2. Test case to check if file share or repository is reachable where test suite is stored.
3. If test system database is managed by the tester tool or (use any mocking framework):
   1. Is creation of test system in data base is successful.
   2. Is deletion of test system in data base is successful.
   3. Is updation of test system in data base is successful.
   4. Similar test cases for devices
4. Test case if tester tool is able to get identify the OS and devices needed by test suite

# 3.2 Integration Test

Below is list for few test cases

Test case to check for all test suite configuration, at least one test system exists where it can be executed. This test should also run on every time when there is change in test suites.

# 4. Improvement

1. Tester tool should support both GUI and command line option for different types of users.
2. Testing of tool should be automated.
3. Adding support and testing for different platforms.
4. Using reactive programing (RxJava) for non-blocking asynchronous code. Choosing the database is also having the impact as current JDFC (which is used internally by Spring Data or Hibernate) is blocking. Currently NoSQL database apis are non-blocking.
5. Report module also should provide some interface (like web site) where can show the status of test suite execution apart from sending email for initiating users.
6. LDAP module should be added for getting the user data for reporting.

8. Execution

With all my day today applications, I found the following results. When I loaded the application with 300 MB text file with 129 processes running on system (Fig 8.1). Following are the observations.

* System was using 5.75 GB of RAM (Fig 8.2)
* After loading the 300 MB file RAM utilization went up by 700 MB (approx.) (Fig 8.3).
* It took 5 seconds to load the file “Time take to load file : 4820” ms

To summarize if we want to load 1 GB file then we may require 2.5 GB RAM for loading the file with you day to day application.

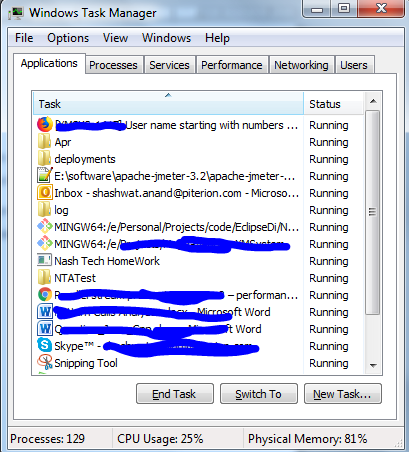


Fig 8. 1

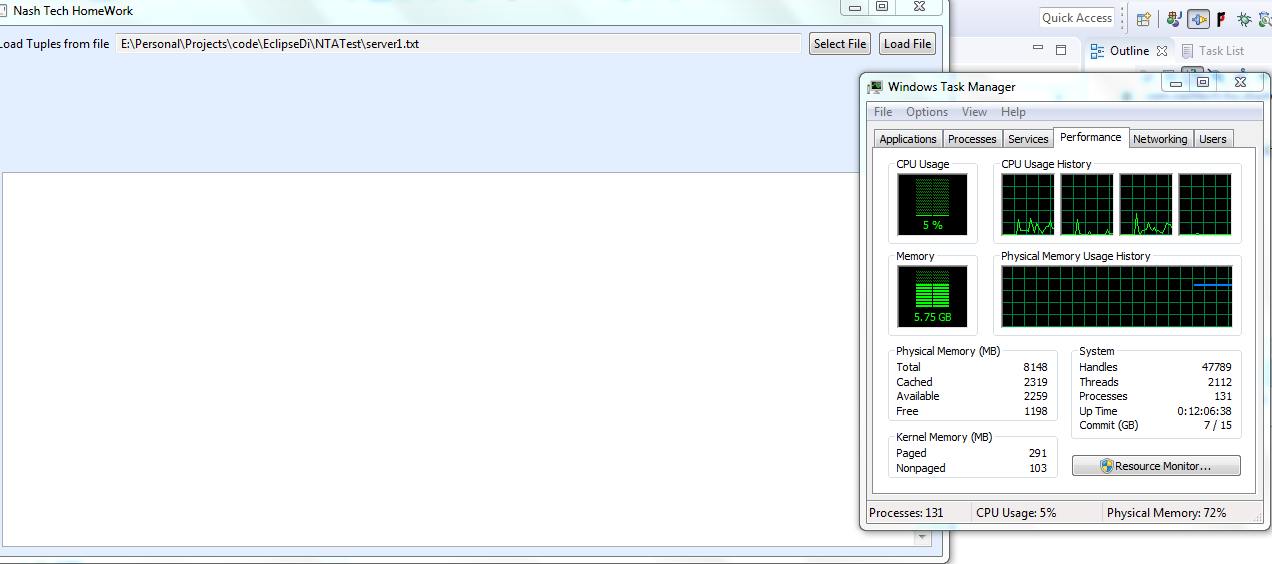


Fig 8. 2

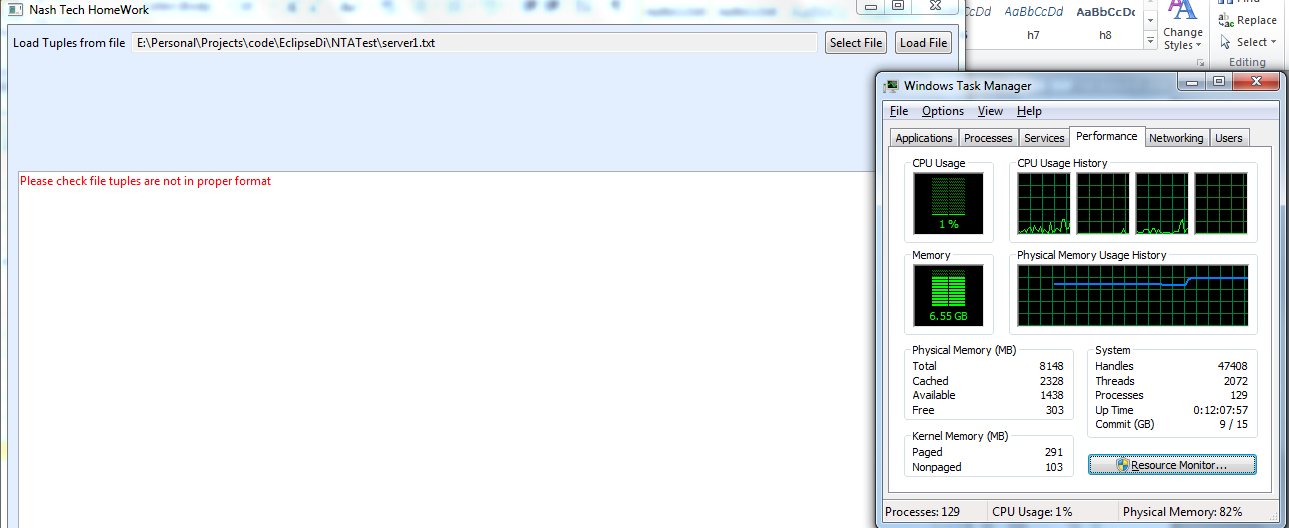


Fig 8. 3