Design Document

Task Process Manager

Authors

S.No.	Name	Email
1	Vivek Vellaiyappan Surulimuthu	Vivek.Surulimuthu1@marist.edu
2	Neha Ersavadla	Neha.Ersavadla1@marist.edu
3	Deeksha Sudini	Deeksha.Sudini1@marist.edu

Revision History

Revision	Dates & Time	Revised By	Current Plan	Comments
1	09/21/2018 & 8pm	All team members	- draft	
2	09/22/2018 & 8pm	All Team Members	- draft the project layout (platform, tech, etc.)	
3	09/27/2018 & 5pm	All team members	- modifications before demo if necessary	
4	10/11/2018 & 9pm	All team members	- Final documentation for ver 2 done	- fixed the tech stack we are about to follow

5				

Table of Contents

Target Audience

The design document gives a brief description on the application to the development team and the testing team to approach towards the process and analyze whether the application meets the proposed requirements of the client.

End User : Developers, Testers, Tech Support, Client

Objective

- To monitor the real time process of a machine that includes but not limited to usage of cpu, memory, network, etc.
- To show informative statistics about the computer's performance.
- To have a real time information on memory leaks, concurrency issues

Understanding how the machine utilizes the resources to which it's been allocated has been a hard task and there are many theories and practices being made to monitor the real time working process of the machine. Real time working process includes how the machine uses the RAM memory, CPU cycles, Hard disk memory, networking data flow and so on.

With a task manager that monitors the machine run-time process utilization and displaying the aforementioned usage holistically in an intuitive manner, the user can easily understand which process is running and how machine utilizes the resources to which its been allocated. Applications of this product will help the user to start/stop/make a process idle, understand the data packets flow happening the network, cpu usage for any type of the application being run.

The task process manager to be developed at the end of this semester will help to achieve aforementioned applications.

MVP

Versions	**Aim**
1	- retrieve real time system level information and reveal it in the UI properly
2	- adding visualization features to have a better data insights
3	- allow user to handle the process attribute (start, idle, kill)
4	- to reveal data sights over process issues like Memory leaks, concurrency, etc.
5	- to suggest automatically how to handle the process (kill or idle) if it crosses a particular threshold which if not handled properly might give issues like memory leak, etc.

Real time statistics aimed to show:

Statistics Type	**Aim**
System memory statistics	total, free, shared
CPU statistics	load averages, user cpu, system cpu
Process level statistics	process arguments, memory consumption, cpu consumption, credential info, state, environment, open file descriptors

File system level statistics	local and remote mounted file systems (NTFS, ext, SMB, NFS, etc), capacity, utilization
Network interface level statistics	all available network interfaces detected and monitored for bytes received/transmitted, packets received/transmitted, collisions, errors, dropped packets

Requirements

As of now, user will be able to view the working functionality of the task manager application by executing the jar file and navigating to the working url, given the user sets up all the required environment.

Kindly refer https://spring.io/guides/gs/serving-web-content/

Specifications

Spec design varies from time to time based on the Spring/MVP versions team is trying to achieve. But the overall tech stack structure is as follows

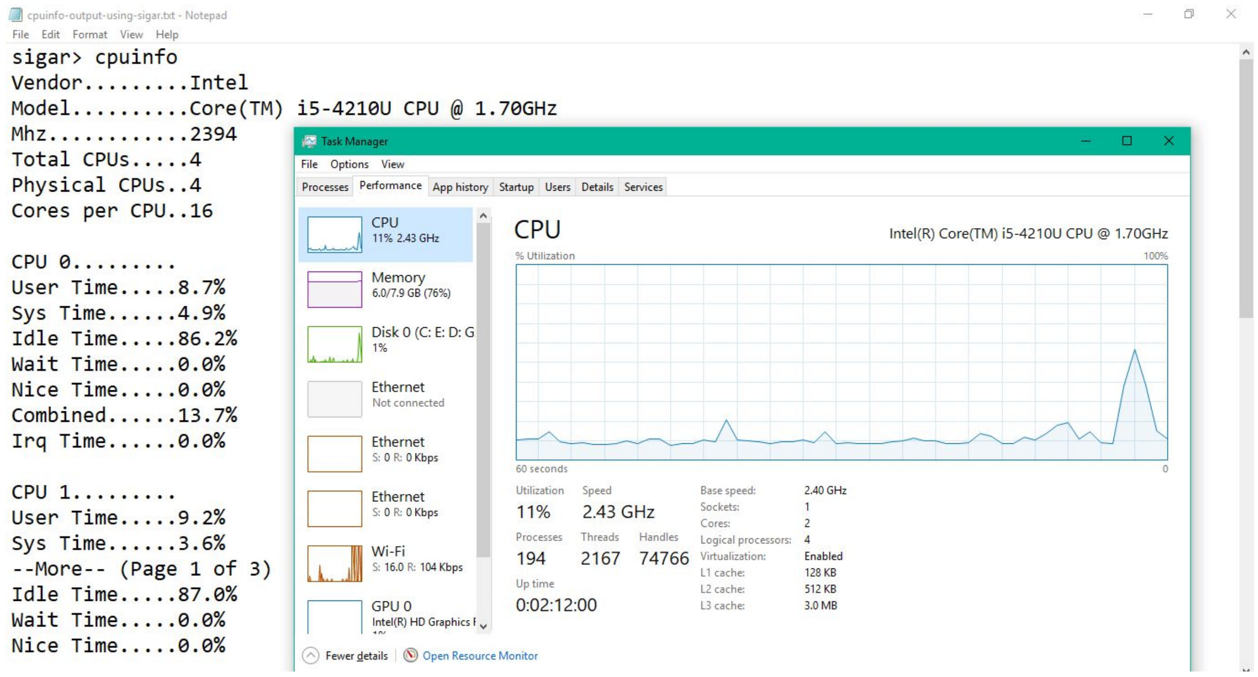
Prototype

CPU related info data being pulled using sigar

```
sigar> cpuinfo
Vendor.........Intel
Model.........Core(TM) i5-4210U CPU @ 1.70GHz
Mhz............2394
Total CPUs.....4
Physical CPUs..4
Cores per CPU..16
```

```
CPU 0.........
User Time.....8.7%
Sys Time......4.9%
Idle Time.....86.2%
Wait Time.....0.0%
Nice Time.....0.0%
Combined......13.7%
Irq Time......0.0%

CPU 1.........
User Time.....9.2%
Sys Time......3.6%
--More-- (Page 1 of 3)
Idle Time.....87.0%
Wait Time.....0.0%
Nice Time.....0.0%
Combined......12.9%
Irq Time......0.0%

CPU 2.........
User Time.....9.6%
Sys Time......3.8%
Idle Time.....86.4%
Wait Time.....0.0%
Nice Time.....0.0%
Combined......13.5%
Irq Time......0.0%

CPU 3.........
User Time.....9.7%
Sys Time......3.8%
Idle Time.....86.4%
--More-- (Page 2 of 3)
Wait Time.....0.0%
Nice Time.....0.0%
Combined......13.5%
Irq Time......0.0%

Totals........
User Time.....9.3%
Sys Time......4.0%
Idle Time.....86.3%
Wait Time.....0.0%
Nice Time.....0.0%
Combined......13.3%
Irq Time......0.2%

sigar>
```

Screenshot


cpuinfo-output-using-sigar.txt - Notepad
File Edit Format View Help
Task Manager
File Options View
Processes Performance App history Startup Users Details Services
Fewer details
Open Resource Monitor

```
sigar> cpuinfo
Vendor.........Intel
Model..........Core(TM) i5-4210U CPU @ 1.70GHz
Mhz............2394
Total CPUs.....4
Physical CPUs..4
Cores per CPU..16

CPU 0.........
User Time.....8.7%
Sys Time......4.9%
Idle Time.....86.2%
Wait Time.....0.0%
Nice Time.....0.0%
Combined......13.7%
Irq Time......0.0%

CPU 1.........
User Time.....9.2%
Sys Time......3.6%
--More-- (Page 1 of 3)
Idle Time.....87.0%
Wait Time.....0.0%
Nice Time.....0.0%
```

CPU 11% 2.43 GHz
Memory 6.0/7.9 GB (76%)
Disk 0 (C: E: D: G 1%
Ethernet Not connected
Ethernet S: 0 R: 0 Kbps
Ethernet S: 0 R: 0 Kbps
Wi-Fi S: 16.0 R: 104 Kbps
GPU 0 Intel(R) HD Graphics F

CPU Intel(R) Core(TM) i5-4210U CPU @ 1.70GHz
% Utilization 100%
60 seconds 0

Utilization	11%
Speed	2.43 GHz
Processes	194
Threads	2167
Handles	74766
Up time	0:02:12:00
Base speed:	2.40 GHz
Sockets:	1
Cores:	2
Logical processors:	4
Virtualization:	Enabled
L1 cache:	128 KB
L2 cache:	512 KB
L3 cache:	3.0 MB

Approach

- Project Management: Agile methodology (Scrum)
- Software Product Development Path: (following MVPs)
    - Gathering system level information by parsing the information to get better data insights and showing the info on the website page.
    - Approaching towards the project in MVC model where the controller servlet fetches the data from the model to the view.
    - Fetching real time data and retrieving the data to the view part.
    - Showing the process timings and graphs related to the memory usage and time.
    - Cleaning the unnecessary usage of resources to improve the performance.

External Design

GUI

- Will be updating the wireframes for the product soon

API

- Sigar API - https://github.com/hyperic/sigar - Sigar is a free software library (under the Apache License) that provides a cross-platform, cross-language programming interface to low-level information on computer hardware and operating system activity. The library provides bindings for many popular computer languages and has been ported to over 25 different operating system/hardware combinations. Sigar stands for System Information Gatherer And Reporter and was originally developed by Doug MacEachern, the author of the popular mod_perl module for the Apache web server.

Use cases

User actions

- Goto the port, say http://127.0.0.1:8000/ after running the application using IntellIj IDE.
- Refer the wireframe above to view how exactly the product works.

System events

- After application being started, the system level information will be pulled via JNI and stored in SQLite3 using JDBC connectivity. The stored information will be processed and then displayed in the UI of the website.

Internal Design

Hardware	**Software**
Minimum requirements ● RAM - 1 GB ● Processor - Intel Core 2 Duo ● HDD - 80GB ● OS - Windows 7	● Java ● JNI ● JDBC ● SQLite3 ● Spring ● HTML, CSS, JavaScript ● D3.js

	● GitHub

Development Standards:

For this project, we decided that we would stick to the Agile development model. This is because of the various advantages that are included in this model considering the group size we are operating.

Since, we are building this project in Spring MVC model Java is going to be our go- to language. For UI/UX purpose we will be using Javascript in the initial stage and some other technologies can also be added based on the requirements on the go.

Software License

This application is going to be a free licensing software. In other words, it is going to be open-source due to the fact that the number of advantages it has over proprietary.

Development Environment:

- IntellIj - IDE for this project.
- GitHub – VC
- Bug tracker - JIRA or GitHub issues
- Build Sys - Maven or Chef-puppet
- CI / CD - Jenkins, Docker
- Tech stack: Java, Spring, JNI, SQLite3, D3.js, Front end languages (HTML, CSS, JS)

Software Flow

Program Initialization (program runtime-environment initialization)

Metrics Collection (collecting system-level information)

Data Aggregation (parsing collected data)

Database Storage (storing parsed valid information in database)

Data Visualization (show information in UI that varies in real time)

Termination

Test environment

We are using JUnit as the testing environment.

JUnit is a Java library to help you perform unit testing. Unit testing is the process of examining a small "unit" of software (usually a single class) to verify that it meets its expectations or specification.

Writing a test case

Test cases will be created accordingly for the type of Testing process being performed. That includes Unit Testing (JUnit), System Testing, Integration Testing, Regression Testing, etc.

System Testing

System testing is performed over here to verify whether the end product meets the specified requirements. Methods such as Black box testing will be used.

Packaging

The product will be delivered in .war file format. Using build tools such as Maven with Spring Boot, end user can run the product successfully provided he/she sets up the environment.

Build System Used: Maven
Maven coordinates

- <groupId>com.triofoxes.project</groupId>
- <artifactId>process-monitor</artifactId>
- <version>0.0.1-SNAPSHOT</version>
- <packaging>war</packaging>

Metadata: kindly refer group ID and version above
Build Information

- Packaging type: war

Resources & Dependencies

- spring-boot-starter-web

- spring-boot-starter-actuator
- spring-boot-devtools
- spring-boot-starter-test

Server used:

- Embedded Tomcat server

Instructions to execute this project:

- https://github.com/vivekVells/TaskProcessManager#instructions-to-run-this-project-in-your-machine

Things planned to improve on

- Have to run the jar file by using a command
- Planning to reduce all these complexity in a different way!

Security

For this application we do not need any admin permissions but the device should allow the software to access system level information.

Globalization

English will be the official language for the documentation or user manual for this application.

Supporting Material

Glossary

- JNI
- JDBC
- Spring MVC
- SQLite3
- JSON
- SIGAR
    - Sigar is a free software library (under the Apache License) that provides a cross-platform, cross-language programming interface to low-level information on computer hardware and operating system activity.

Risks and Dependencies

The risk for now would be that the design document being proposed is in draft stage and we may or may not adopt to some other technology. This is a standalone web application that is not dependent on any other application services.

References:

Websites:

- https://blog.prototypr.io/todo-b6c608656211
- https://www.lifewire.com/task-manager-walkthrough-4029769
- https://source.android.com/reference/tradefed/com/android/tradefed/device/CpuStatsCollector#summary
- https://www.javalobby.org/java/forums/m91813165.html
- https://spring.io/guides/gs/serving-web-content/ - Spring packaging and execution

Books

Research papers