**Software Design Description**

**SysAnatomy: A Reporting Tool for Linux OS**

Version <1.0>

10th March 2016

**Authors:**

Surya Valluri

Sourav Bhowmik

Saranya Radhakrishnan

Contents

**[1.](#_Toc445396925)****[Introduction](#_Toc445396925)** [3](#_Toc445396925)

[1.1 Purpose 3](#_Toc445396926)

[1.2 Scope 3](#_Toc445396927)

[1.3 Audience 3](#_Toc445396928)

[1.4 Document revision history 3](#_Toc445396929)

[**2.** **Design overview** 4](#_Toc445396930)

[2.1 Objective 4](#_Toc445396931)

[2.2 Approach 4](#_Toc445396932)

[2.3 Architecture Overview 4](#_Toc445396933)

[**3.** Topology Design 5](#_Toc445396934)

[**4.** **External Design** 6](#_Toc445396935)

[**5.** **Internal Design** 8](#_Toc445396936)

[5.1 Hardware / Software Requirements 8](#_Toc445396937)

[5.2 Development Environment 8](#_Toc445396938)

[5.3 Proposed Components 8](#_Toc445396939)

[5.3.1 Process Specific Data 8](#_Toc445396940)

[5.3.2 CPU/Memory Specific Data 9](#_Toc445396941)

[5.3.3 Report Generation 9](#_Toc445396942)

[5.4 Error Handling & Recovery 9](#_Toc445396943)

[5.5 Testing 9](#_Toc445396944)

[5.6 Packaging 9](#_Toc445396945)

[5.7 Accessibility 9](#_Toc445396946)

[5.8 Risks and Dependencies 10](#_Toc445396947)

[**6.** User Interface Design 10](#_Toc445396948)

[a. Home Page 10](#_Toc445396949)

[**7.** Database Design 11](#_Toc445396950)

[**8.** Future Scope 13](#_Toc445396951)

[8.1 Storage of User profiles 13](#_Toc445396952)

[8.2 Latest Software Version 13](#_Toc445396953)

[**Appendix A: Acronyms and Abbreviations** 14](#_Toc445396954)

[**Appendix B: Products & Tools** 15](#_Toc445396955)

[**References** 16](#_Toc445396956)

# **Introduction**

## Purpose

The purpose of this design document is to outline the technical details of an online metrics collection tool. Furthermore, it provide an overview for the implementation of online metrics collection tool.

Its main purpose is-

1. Provide the link between functional specification and technical details.
2. Provide detailed description and functionality of various components and their interactions.

## Scope

The system collects metrics related to all applications, processes running in one particular device and stores it in a relational database. The system holds data associated with all active processes in a device. This data is processed, organized and stored in a single relational database, giving access to key performance metrics on a customizable, real-time dashboard.

## Audience

This document is intended for the project sponsor, project development team, technical architect, database designer and quality assurance team.

Project sponsor will critique and guide through the phases of the project and provide important tweaking suggestions and tips.

The project development team will use this document as a reference for development. This will make sure that everyone in the team is on the same page and work efficiently.

Testers are responsible for making sure that the final product conforms to the design document.

## Document revision history

|  |  |
| --- | --- |
| **Major Changes** | **Date** |
| Addition of Risks, packaging, accessibility, testing, error handling and recovery to the document.  Modification of DB tables. | 10th March 2016 |

# **Design overview**

## Objective

Provide a platform that enables the users to better understand the co-relation between the programs running in their devices with the performance of the system.

## Approach

This document is created and will be updated during multiple phases of the project life cycle. The project follows an Agile Methodology for development. Agile method of product life cycle helps to adapt to the unpredictable changes in requirements or otherwise in the product.

This methodology consists of two weekly Scrum meetings to keep track of updates and discussion about achieving goals. Scrum meetings are conducted by scrum master, who may change every two weeks. Scrum master acts as a reporting authority for collecting information from each team member, document them and manage them in bug tracker. Every cycle's output and goals are discussed with the project sponsor, who when approved ensures implementation is being carried out as per the requirements.

## Architecture Overview

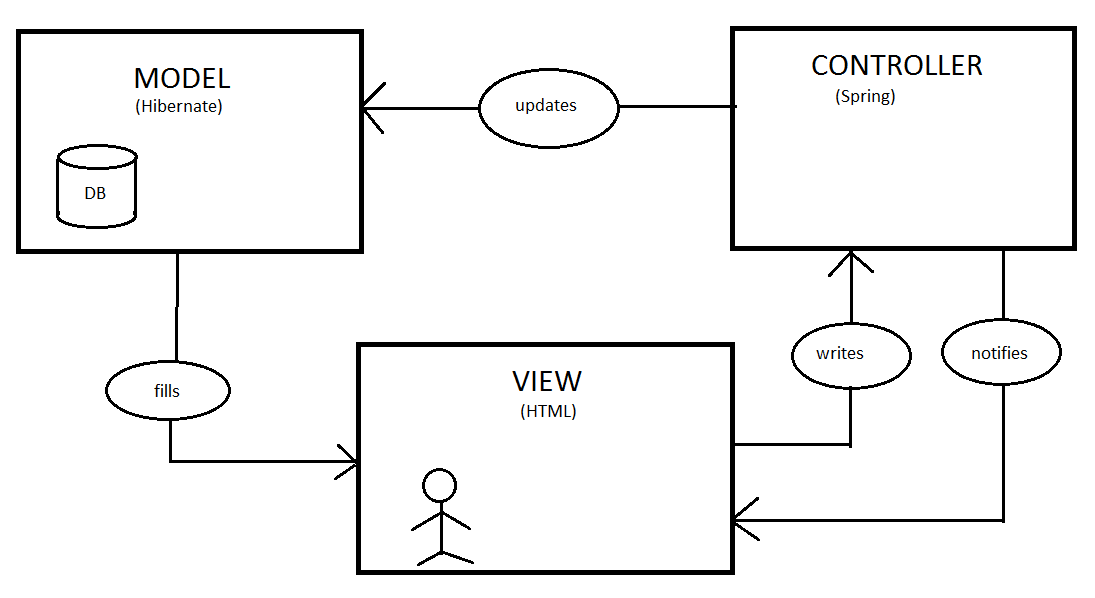


Fig 2.a: An overview of the MVC architecture

The above diagram shows the Model View Controller architecture of the system being developed. The end user generates HTTP requests from a web browser. This request is analyzed and processed by the MVC to produce an appropriate response, which is returned back to the user. The internal functioning is hidden from the end user. The user can only see the view as shown in the diagram above.

The controller works as the manager who directs and regulates all the functionalities. It makes changes to the model and notifies the view about the same. The view updates itself and renders a fresh view.

# Topology Design

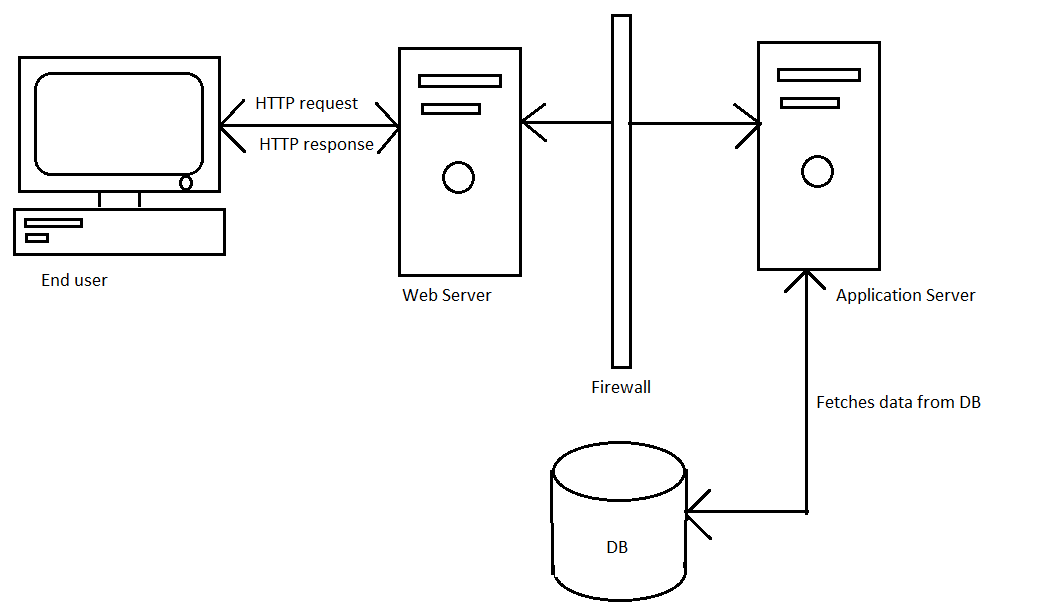


Fig 3.a: A detailed view of the MVC architecture w.r.t SysAnatomy

As shown above, the end user uses his device to connect to our servers via HTTP requests and responses. He interacts with the forms and generates new data that is to be processed and stored in the Database.

The Front-end or UI would be made using HTML, JQuery and JavaScript. The view has the important role of rendering the model data. It also requests updates from the model.

The Controller of the MVC model would make use of JSP to implement the business logic. It would serve functionalities such as defining the application behavior. It would map http requests to model updates. Further, it would be responsible for selecting the view as response for each functionality.

While the Model component of the MVC will be implemented using Spring and Hibernate in Java. It encapsulates application data and application functionality. It also notifies the view of any of the changes. Additionally, it provides interface for state queries.

The data collected from the user’s devices will be stored in SQL Server, a Relational Database. This data will then be processed and analyzed to generate reports and metrics.

# **External Design**

4.1 Use Case Diagrams

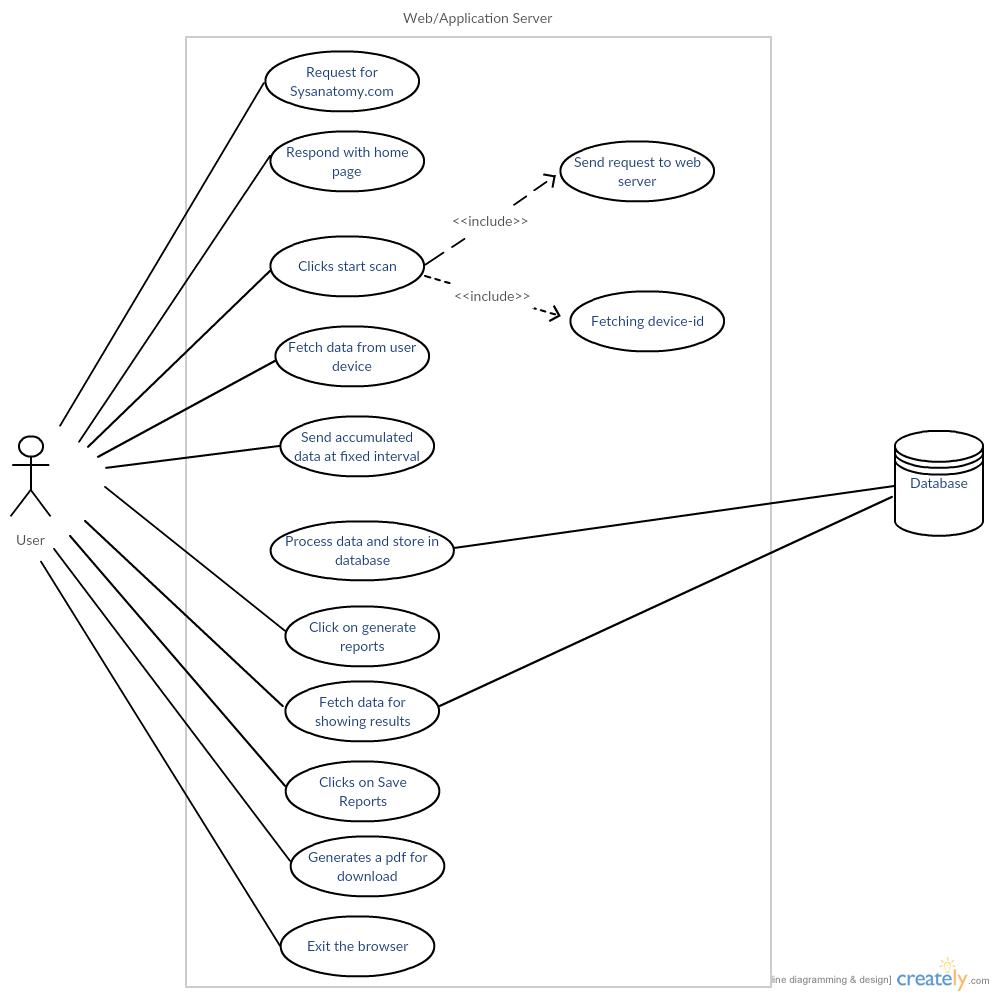


Fig 4.a: Use Case showing the interaction of Web/Application Server with the Database and end user.

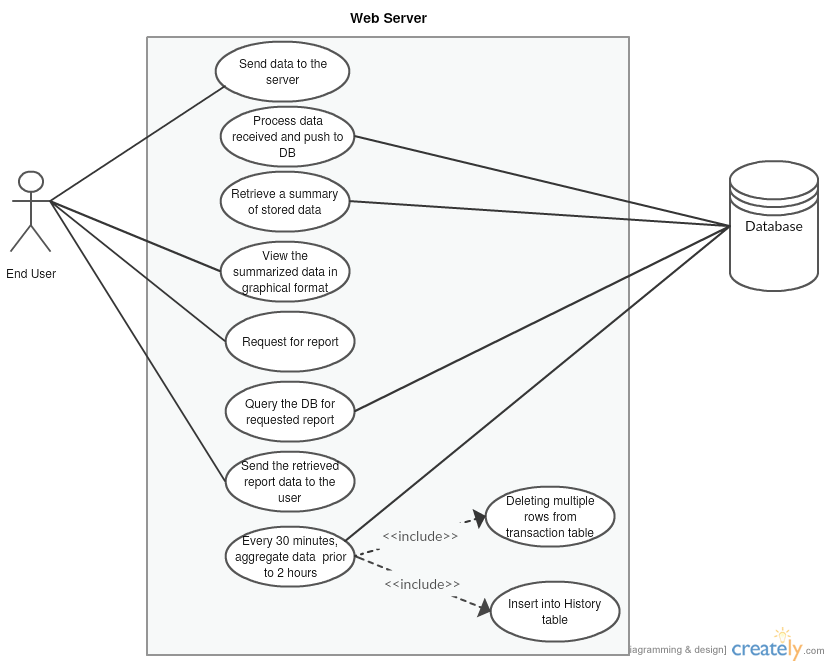


Fig 4.b: Use Case showing the interaction of Web/Application Server with the Database and with/ without end user.

# **Internal Design**

# Hardware / Software Requirements

|  |  |
| --- | --- |
| **CPU** | X86 or higher, any speed. |
| **Operating System** | Linux. |
| **Main Memory** | Minimum: 8MB  Recommended : 16 MB |
| **Display** | Color Required.  Minimum : 256 Colors  Recommended: 16K colors are better. |
| **Disk Space** | As required for any browser. |
| **Web Browser** | IE, Chrome, Mozilla. |

# Development Environment

|  |  |
| --- | --- |
| **Compiler** | Javac |
| **IDE** | NetBeans |
| **Source code repository** | <https://github.com/rsaranya/Project-SysAnatomy> |
| **Defect Tracker** | <https://github.com/rsaranya/Project-SysAnatomy/issues> |
| **Build Process** | Maven |
| **Release Process** | Deploy in a Web/ Application Server like JBoss or Apache Tomcat |

# Proposed Components

## Process Specific Data

This module deals with collection of data related to processes running on the system. This includes the identifier of the process, its parent information, name, memory utilization and the owner. This information gives an insight into each process and their behavior in the system.

Extracting this data, cleaning and transforming it into a generic format helps in converting data into information. This information can be used for generating reports by calculating various metrics. These metrics helps us in understanding the effect the process has in the system or vice versa.

## CPU/Memory Specific Data

This module deals with collecting CPU related data like memory utilization, processing speed, number of processors available for processing, support for hyper threads, logical processors This information gives an insight into how the system is performing.

Extracting patterns and variations in the CPU/memory related data to represent in graphical format like heat maps and graphs for easier understanding.

## Report Generation

All the collected metrics can be used to create customizable reports, which can be printed, saved or exported to csv format.

# Error Handling & Recovery

Apache Log4j 2, an open source logging system will be used to generate logs for the system. The logs can be used to detect failures, if any, during the working of the application. The logs generated will be stored at the server.

# Testing

* + 1. **Test Environment**: Multiple LINUX operating systems.
    2. **Unit Testing**: As and when the modules are developed they will be tested.
    3. **Functional Testing**: Test suites will be defined in this phase using Junit to test the functional correctness of each module.
    4. **UX Testing:** On deployment the system will be tested by peers to get appropriate feedback on their perception.

# Packaging

The finished software product will be packaged using Apache Maven in the default packaging type-war. This will finally be deployed at the web server. The end user can access the program that is sitting on the server by generating HTTP requests.

# Accessibility

The user interface has been given an overall blue theme in order to make sure that people with partial color-blindness (Red-green Dichromacy) have no difficulty in viewing and navigating through the application.

# Risks and Dependencies

Over-ambitious goals or trying to build too much in a limited period of time can be a potential risk towards the successful completion of the project.

The project needs a secure internet connection to run successfully on a browser. It is not a standalone application.

# User Interface Design

## Home Page

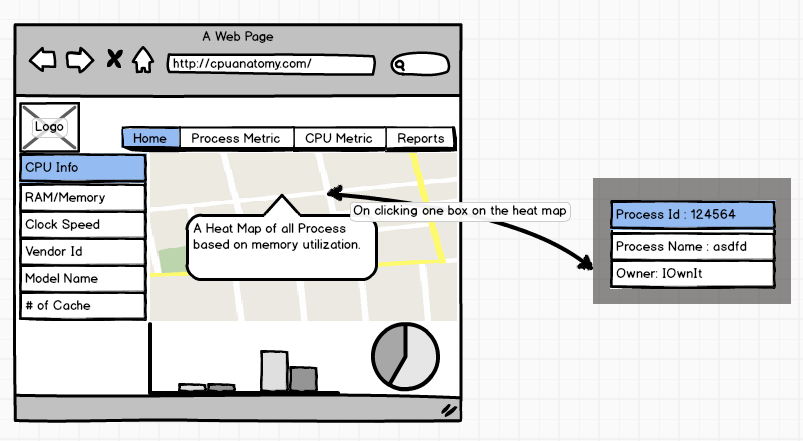


Fig5.1: Main Page containing links to other pages. Contains a snapshot of users device information

# Database Design

***Table Name:*** REQUEST

***Description:*** Contains data to uniquely identify a user.

|  |  |  |
| --- | --- | --- |
| RQ\_SYSTEM\_ID | VARCHAR(50) | unique system identifier |
| RQ\_TOKEN\_ID | INT | auto-generated token id |

***Table Name:*** PROCESS\_INFO\_AGGR

***Description:*** Contains summarized, at fixed intervals, process related data.

|  |  |  |
| --- | --- | --- |
| RQ\_TOKEN\_ID | INT | Token id to track a session |
| PI\_TIME\_STAMP | TIMESTAMP | Date & Time at which the data was collected |
| PI\_PROCESS\_ID | INT | Process id for a particular process |
| PI\_PROCESS\_NAME | VARCHAR(50) | Name of the process |
| PI\_PPID | INT | Parent process id |
| PI\_PROCESS\_OWNER | VARCHAR(50) | Owner of the process |
| PI\_MEM\_USAGE | NUMERIC(6,0) | Aggregated memory usage of the process |
| PI\_RAM | NUMERIC(6,0) | Aggregated CPU utilization by the process |
| PI\_CLOCK\_SPEED | NUMERIC(1,2) | Aggregated clock speeds |
| PI\_THREAD | NUMERIC(6,0) | Aggregated number of active threads |
| PI\_CACHE | NUMERIC(6,0) | Aggregated cache usage for the process |

***Table Name:*** CPU\_INFO\_AGGR

***Description:*** Summarizing CPU related data for a fixed time interval.

|  |  |  |
| --- | --- | --- |
| RQ\_TOKEN\_ID | INT | Token id to track a session |
| CI\_TIME\_STAMP | TIMESTAMP | Date & Time at which the data was collected |
| CI\_IP\_ADD | VARCHAR(50) | IP address of the user |
| CI\_NUM\_PROC | NUMERIC(4,0) | Aggregated number of processors on the user CPU |
| CI\_MEM\_SIZE | NUMERIC(6,0) | Aggregated memory size of the user CPU |
| CI\_RAM | NUMERIC(6,0) | Aggregated  RAM of the user CPU |
| CI\_CLOCK\_SPEED | NUMERIC(1,2) | Aggregated clock speed of the user CPU |
| CI\_THREAD | NUMERIC(6,0) | Aggregated number of threads running on the CPU |
| CI\_CACHE | VARCHAR(50) | Aggregated amount of cache available on CPU |

***Table Name:*** PROCESS\_INFO\_LIVE

***Description:*** Contains dynamic data related to a process.

|  |  |  |
| --- | --- | --- |
| RQ\_TOKEN\_ID | INT | token id to track a session |
| PIL\_TIME\_STAMP | TIMESTAMP | Date & Time at which the data was collected |
| PIL\_PROCESS\_ID | INT | process id for a particular process |
| PIL\_MEM\_USAGE | NUMERIC(6,0) | Memory utilization for the given process at a particular interval |
| PIL\_RAM | NUMERIC(6,0) | CPU utilization for the given process at a particular interval |
| PIL\_CLOCK\_SPEED | NUMERIC(1,2) | Clock speeds for that particular interval |
| PIL\_THREAD | NUMERIC(6,0) | Number of active threads in that time interval |
| PIL\_CACHE | NUMERIC(6,0) | Cache usage for the process at that time interval |

***Table Name:*** CPU\_INFO\_LIVE

***Description:*** Contains dynamic data related to the CPU.

|  |  |  |
| --- | --- | --- |
| CI\_TOKEN\_ID | INT | token id to track a session |
| CIL\_TIME\_STAMP | TIMESTAMP | Date & Time at which the data was collected |
| CIL\_IP\_ADD | VARCHAR(50) | IP address of the user |
| CIL\_NUM\_PROC | INT | Number of processors on the user CPU at that time interval |
| CIL\_MEM\_SIZE | NUMERIC(6,0) | Memory size of the user CPU at that time interval |
| CIL\_RAM | NUMERIC(6,0) | CPU usage at a specific time interval |
| CIL\_CLOCK\_SPEED | NUMERIC(1,2) | Clock speed of the user CPU at that time interval |
| CIL\_THREAD | NUMERIC(6,0) | Number of threads running on the CPU at that time interval |
| CIL\_CACHE | NUMERIC(6,0) | Cache available on CPU at that time interval |

# Future Scope

## Storage of User profiles

This module deals with generating metrics that compare the users registered devices and generate reports using this information. A unique identifier will be used to keep track of user specific information.

Each visitor to the site may choose to create an account and register multiple devices. Useful and detailed comparison of various metrics for all the registered devices will be provided through the application.

## Latest Software Version

Provide with information about latest updates available for various applications installed in the user’s device(s).

# **Appendix A: Acronyms and Abbreviations**

|  |  |
| --- | --- |
| **HTML** | Hypertext Markup Language |
| **MVC** | Model View Controller |
| **UI** | User Interface |
| **HTTP** | Hypertext Transfer Protocol |
| **JSP** | Java Server Pages |
| **UX Testing** | User Experience Testing |

# **Appendix B: Products & Tools**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Software/  Tool | Version | | Source | Description |
| JRE | | 8 | http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html | Runtime Environment to run Java Programs |
| Maven | | 3.3.9 | https://maven.apache.org/ | Build manager for the Java Program |
| Eclipse | | 4.5.1 | https://eclipse.org/ | IDE for developing Java based programs |
| Tortoise git | | 1.8.16.0 | https://tortoisegit.org/ | User Interface for operating Git |
| Git hub | | 10.9 | https://github.com/ | Repository for source code |
| Creatly | | - | https://creately.com/ | Designing software used to create Use Case Diagrams |

# **References**

1. Design Document Template : <http://www.in.gov/fssa/files/QualCheck.pdf>