SySTemantics Design Document

By

Dixita Sharegar

Bhargav Uppalapati

Siva Chintapalli

**Sponsor**

Anthony Giorgio

Poughkeepsie, NY

February 2016

# **Document Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Author | Date | Status & Description |
| 0.1 | All | 02/25/16 | First Draft |
| 0.2 | Dixita | 03/06/16 | Document history section added |
| 0.3 | Siva Chintapalli | 03/07/16 | Architecture, Design, Use case diagrams added |
| 0.4 | Bhargav | 03/08/16 | Internal Design, other necessary data added |
| 1.0 | All | 03/10/16 | Final Draft |

# **Audience**

This document is intended for designers, developers & engineers who want to modify or extend the existing implementation of the metrics collector application. It is also intended for customers who want a detailed description of the system.

# **Objective**

A Metrics collector is a console based application which runs on the Linux-based operating system. The application incorporates *CPU* stats, memory stats, Network stats. It helps users of the system to manage the CPU throughput, and view network and memory stats allocated to the applications. Keeping track of all the system matrices without a proper application is hard. It is also very difficult to remember the commands to view all the system statistics. The statistics are logged into a database frequently which can be viewed by the user to keep track of system performance.

# **Approach**

The main goal of the project is to give the accurate and reliable information of the System Statistics.

· Goal 1: Implementing the database.

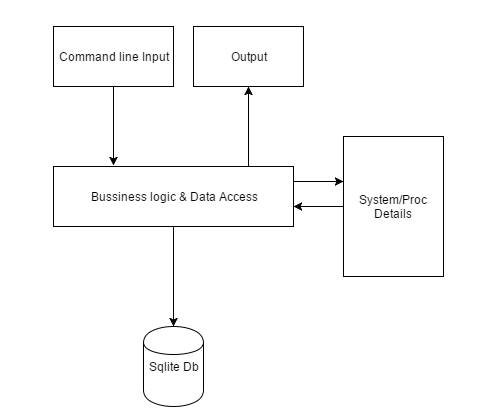
· Goal 2: Designing the User Interface.

· Goal 3: Implementation (Connection to the database and logical Operations)

· Goal 4: Displaying the records/information in a report format.

· Goal 5: Testing and maintenance of the project.

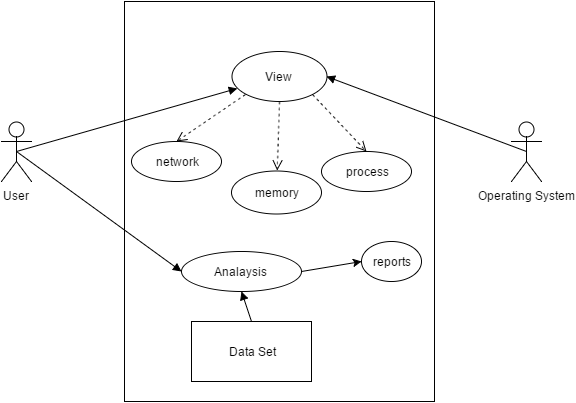
**ARCHITECTURE**



# **External Design**

## Command Line Interface (CLI)

# **Use Case Diagram**

 Application

1. User requests the application to view the running tasks/process in the system.
2. The operating system will control the application to provide the required information.
3. User can view network/memory/process statistics.
4. User can view and print the reports based on the time constraint.

# **Internal Design**

## Development Standards

* The project involves usage of agile methodology.
* Feedback of end-user is taken at every phase and developed accordingly.
* *Java*, *C* and *SQL* languages are used in a development of SySTematics.
* *Ncurses*- interfaces for the editor.
* For graphs *Gnu plot*, *Graphite, Darkstat*.
* SySTematics will be open-source software which will be available to the public.

## Hardware Resources

* Physical machines : Desktop
* Operating systems : Linux
* Application Type : Console Based Application

## 

## Development environment:

* Compilers : GCC,JVM
* IDE : NetBeans
* Source code repository : GitHub
* Build process : Maven
* Database : SQLite
* Instance : Single

# **Software flow**

The installer installs all software packages required for our application.

Metrics will be collected using different Linux commands.

Memory, CPU and network metrics will be aggregated and displayed in common units.

The collected metrics will then be stored in the SQLite database.

Data is fetched from the database based on what the user is viewing. End User is not aware of the processes above this block.

All related metrics will be displayed to the user based on his screen selection.

The application is terminated followed by data collection and storage in database.

## Error Handling & Recovery

The application logs every step or function call, so that it becomes easy for the developer to duplicate the issue and solve it. If a part of the application is not working properly, it will fail gracefully and an error message will be displayed on screen.

## Testability

* Test Environment will be same as the customer’s environment i.e. Linux OS, Standalone machine.
* Test Scenarios include testing with heavy CPU processes, memory consuming processes, I/O processes, etc.
* Regression Testing will be performed after every module to ensure all earlier functionalities are working.
* Test cases will be written to record all results. Function testing and system testing will be performed after completion of every module.

## Packaging

An installer will be used to install and uninstall the application.

**Data**:

This application integrates SQLite database:

* Which dynamically stores CPU stats, memory stats, Network stats.
* SQLite database is used as it supports many programming languages including java.
* The data update frequency is 5 seconds.

**Scope**:

Outcome of the project:

* The objective of this application is to provide the user a clear picture of system performance.
* Any user can run this system and access the data from the database.
* History as old as one week is maintained in the database and can be viewed/retrieved at any time.
* The database maintains the current and past information.
* Network stats, CPU stats, memory stats are represented by visual indication.
* Users can print the logs of the system statistics but cannot change the data.

# **Security**

There is no requirement for the software to run as root, but can be run as root authority if needed.

* Any user with the installation files can install and use the software. If needed, admin rights can be given.

# **Accessibility**

* The basic requirement, amongst other things, that there is sufficient contrast between text and background color.
* Best practice is to avoid the usage of red/green colors. As 1 in 12 people have red/green /yellow color disability.

# **Globalization**

* The UI buttons, display messages, console outputs are in English.
* SysTemantics supports only English language.
* Documentation is in English.

# **Supporting Material**

## Glossary:

* **CPU**: Central Processing Unit.
* **CLI**: (Command Line Interface): is a user interface to a computer's operating system or an application in which the user responds to a visual prompt by typing in a command on a specified line, receives a response back from the system, and then enters another command, and so forth.
* **Use Case Diagram**: A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different [use cases](https://en.wikipedia.org/wiki/Use_case) in which the user is involved.
* **C**: is a general-purpose, imperative computer programming language, supporting structured programming, lexical variable scope and recursion.
* **Java**: Java is a general purpose, high-level programming language.
* **SQL**: is a special-purpose programming language designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system.
* **Ncurses:** (new curses) is a programming library providing an API that allows the programmer to write text-based user interfaces in a terminal-independent manner. It is a toolkit for developing "GUI-like" application software that runs under a terminal emulator.
* **Gnuplot:** is a [command-line](https://en.wikipedia.org/wiki/Command-line) program that can generate two- and three-dimensional [plots](https://en.wikipedia.org/wiki/Plot_%28graphics%29) of [functions](https://en.wikipedia.org/wiki/Function_%28mathematics%29), [data](https://en.wikipedia.org/wiki/Data), and data fits.
* **Graphite:** is a [free open source software](https://en.wikipedia.org/wiki/Free_open_source_software) (FOSS) tool for monitoring and graphing the performance of computer systems.
* **A Java virtual machine** (**JVM**): is an abstract computing machine that enables a computer to run a Java program.
* **NetBeans:** is a [software development](https://en.wikipedia.org/wiki/Software_development) platform written in [Java](https://en.wikipedia.org/wiki/Java_%28programming_language%29). The NetBeans [Platform](https://en.wikipedia.org/wiki/Platform_%28computing%29) allows applications to be developed from a set of modular [software components](https://en.wikipedia.org/wiki/Software_component) called modules.
* **Maven:** is a [build automation](https://en.wikipedia.org/wiki/Build_automation) tool used primarily for [Java](https://en.wikipedia.org/wiki/Java_%28programming_language%29) projects.
* **SQLite:** is a [relational database management system](https://en.wikipedia.org/wiki/Relational_database_management_system) contained in a [C](https://en.wikipedia.org/wiki/C_%28programming_language%29) programming [library](https://en.wikipedia.org/wiki/Library_%28computer_science%29). In contrast to many other database management systems, SQLite is not a [client–server](https://en.wikipedia.org/wiki/Client%E2%80%93server) database engine. Rather, it is embedded into the end program.

**PROJECT BUDGET:**

* No Project Expenses.

# **Risks and Dependencies**

## Risks:

* Manual Testing is performed throughout the project (Automation testing is not performed).
* It is designed as a desktop application.

## Constraints:

* Lack of mobile and web application design implementation.
* Time.

# **Assumptions:**

* All the team members are dedicating their time on completing their tasks.
* The design document may be modified based on the project requirements.