Metrics Collector Software Design Document

Version <1.1>

Murali Krishna
Vikram Patil
Vikas Nandanam

10-March-2016

Revision History:

Date	Description	Revision	Editor
02/25/2016	Created Draft	0	Authors
03/10/2016	Created Full Doc	1	Authors

Audience:

This document is a catalog for the testers, developers and customer who can refer the document and can gain in depth knowledge of the project and its working. This is not a finalized document it can be enhanced after implementing of the total project.

Table of Contents

1 Introduction

The Software Design Document is a document to provide documentation which will be used to aid in software development by providing the details for how the software should be built. Narrative and graphical documentation of the software design for the project are within the Software Design Document, including use case models, sequence diagrams, collaboration models, object behavior models, and other supporting requirement information.

1.1Purpose

The purpose of the Software Design Document is to provide a description of the design of a system fully enough to allow for software development to proceed with an understanding of what is to be built and how it is expected to be built. The Software Design Document provides information necessary to provide description of the details for the software and system to be built.

1.2 Scope

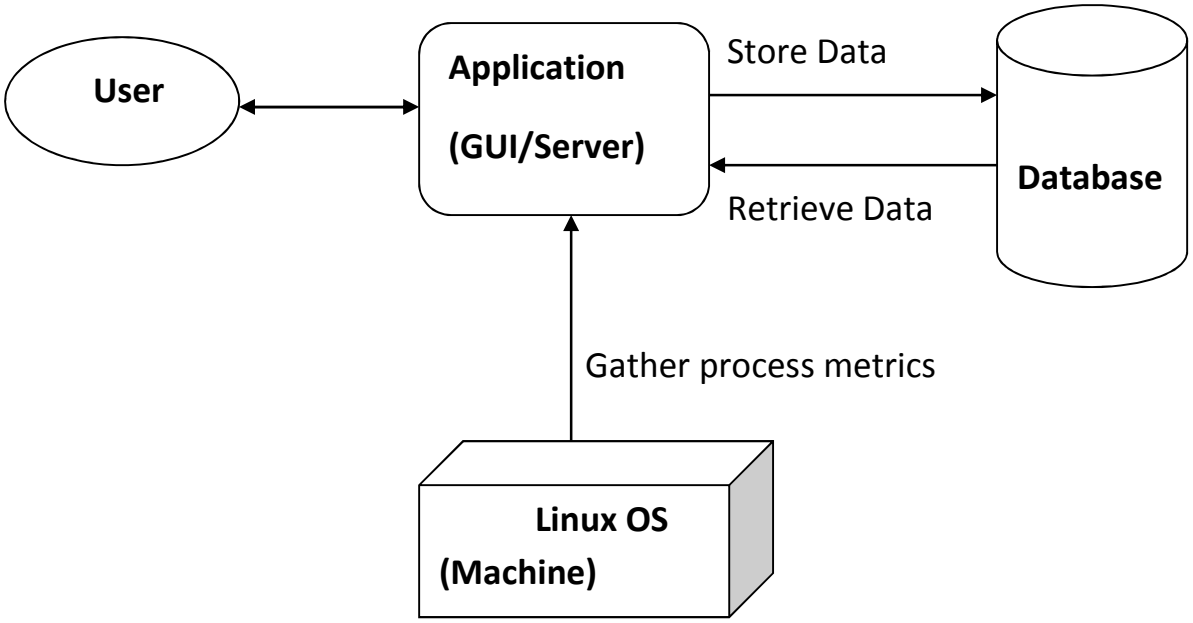
Metrics collector is an application which gathers process metrics from various sources such as running processes on the machine, elapsed time, memory used by the process etc. This application will also collect information about the CPU utilization by the various processes. Application will have a GUI to represent the collected information to the user. Software will be provided with a relational database which will store the metrics data and generate customized reports for the user.

1.3Objective

The goal of this project is to develop user friendly application for gathering and representing process metrics. The software will be an interface between the OS and user. Process metrics data must be stored to analyze the CPU and memory usage of the processes. This software must monitor process metrics in under distinct sections like

CPU, Memory, Disk, Network etc. Metrics collector software has to give CPU and memory usage history to the user.

1.4 Approach



Above figure gives a high end view of the system architecture and various actors involved in it. User interacts with the Metrics Collector application through GUI. Application gathers process metrics information from the operating system to show it to the user and store in the database. As per user's requirement, application queries the database and retrieves data in a report format. Reports are the provided to the user through GUI.

1.5 Pre-requisites

a) Linux OS (Any of Redhat, CentOS)
b) JRE for JavaFX (application program and GUI)
c) MySQL database (relational database)
d) IDE (NetBeans 8.0.2)

2 External Design

2.1 Introduction

The Design Overview is section to introduce and give a brief overview of the design. The System Architecture is a way to give the overall view of a system and to place it into context with external systems. This allows for the reader and user of the document to orient them to the design and see a summary before proceeding into the details of the design.

2.2 Sequence Diagrams

Diagram 1

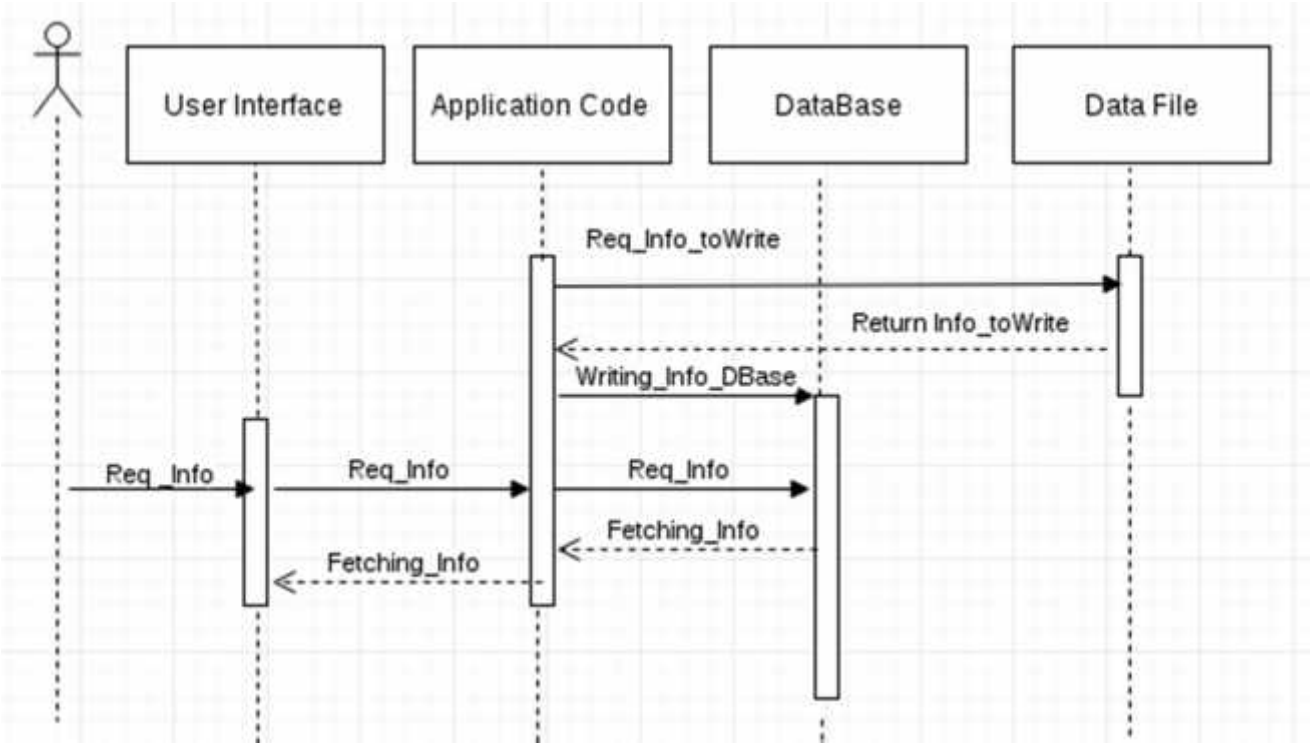


Figure 2.2.1 Sequence Diagram for Metric collector (req_info running in loop)

The above figure illustrates how the sequences of operations are carried for gathering the metrics information. The application code always gathers metrics information from the data files. This gathered information is stored in database by the application code. When a

user requests for information using the user interface the data stored in database is retrieved and shown to user using the GUI.

Diagram 2

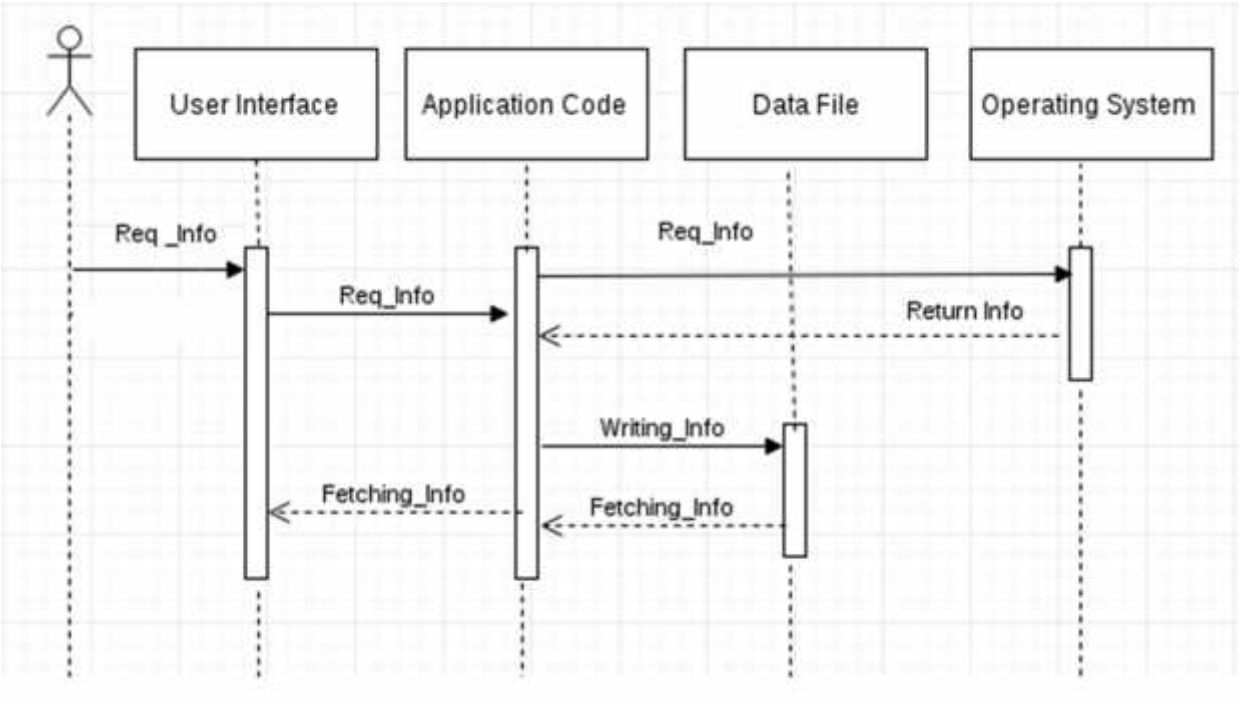


Figure 2.2.2 Sequence Diagram for Metric collector

This figure shows how the operations take place when the application is started. When application is started, the user requests for the metrics information. From the application code the required metrics information is fetched from operating system and displayed to user using user interface.

2.3 System Interfaces

2.3.1 User Interface

The user interface for the system will allow the user to easily generated documents, search for documents, and modify documents. The user should be presented with all main functions on the first user interface page to allow for the user to select the function to use without the need to navigate inward to find it. This desktop application GUI will be developed in Java with JavaFX platform.

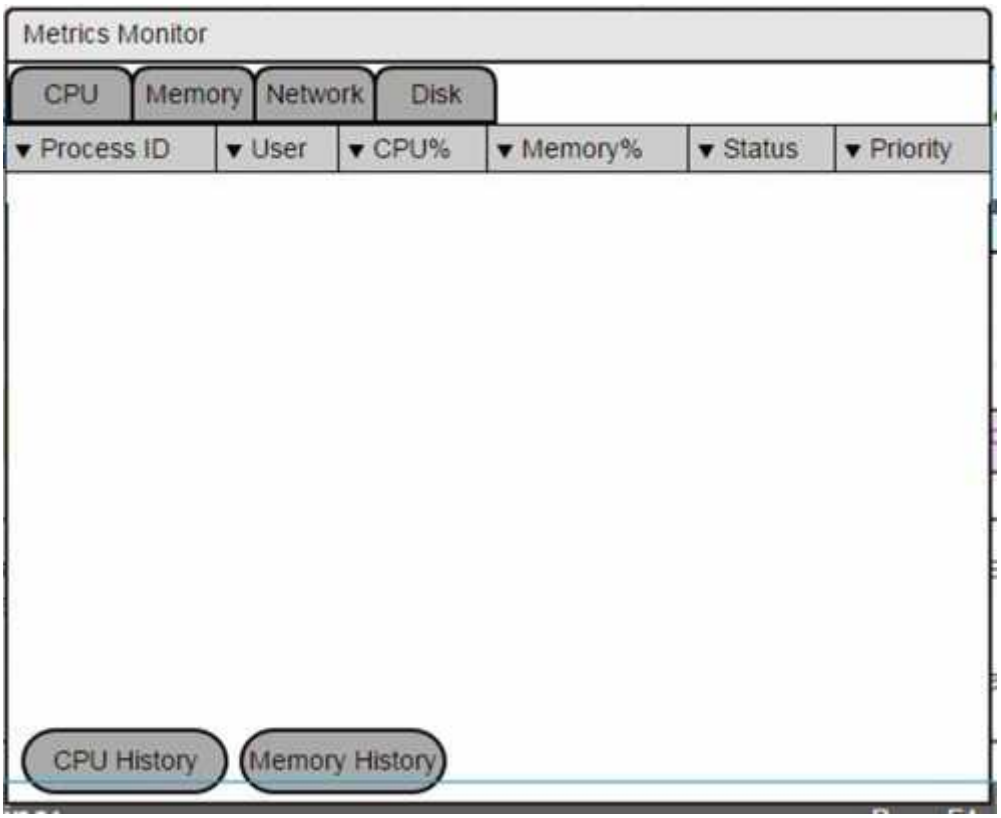


Figure 2.3.1 GUI mockup

As shown in figure 2.3.1 the metrics collector will have four tabs for four distinct sections of metrics data i.e CPU, Memory, Network and Disk. In the body of the window, data is represented in a grid format. At the bottom, there are two buttons to request history reports.

2.3.2 Software Interface

The software will need to interface with a database management system to pull data from it and push data updates to it. The connection will be a standard database connection using JDBC.

3 Internal Design

3.1 Development Standards

The development standard gives the information about the design approach to be used for implementing the application. The development process the project follows an iterative waterfall model. There are advantages with this approach as we can get back to any phase of the development life cycle and modify the design as per the requirements. As the requirements are clear and there is no scope for the change in requirements in future, iterative waterfall model is good enough.

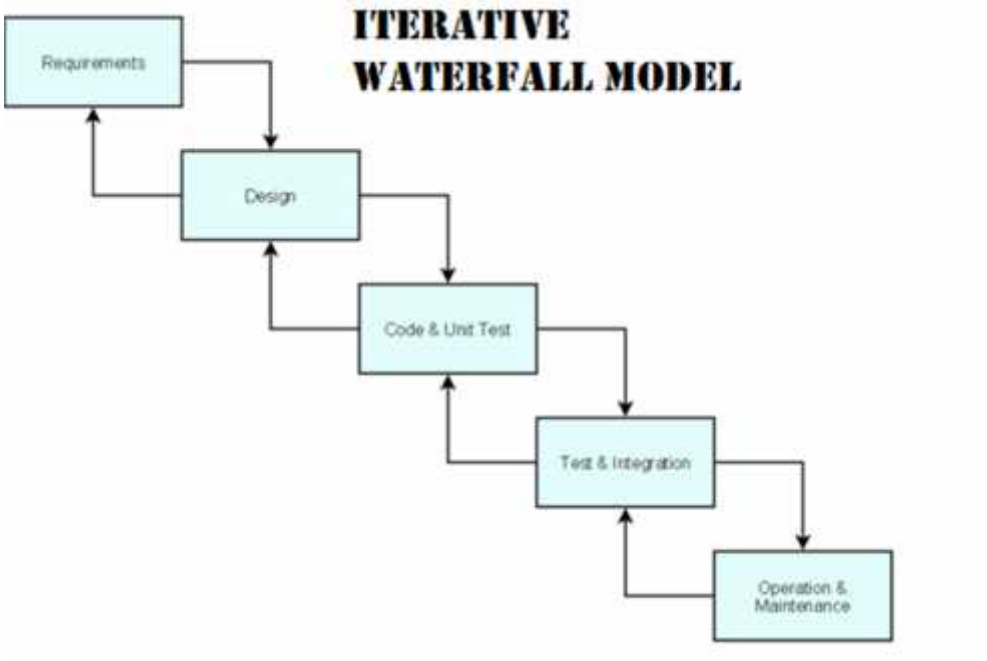


Figure 3.1 Iterative Waterfall model

3.2 Programming Language

1) **UI –** Java with JavaFX framework

   User interface must be lucrative and should be user friendly. JavaFX has replaced Java swing in past few years. JavaFX is better than Java swing in look and feel when it comes to desktop application UI.

2) **Application code- Java and JNI**

   Java for the basic application code and JNI invokes the native code which is important to interact with operating system.

3) **Database- MySQL**

   As the data that the software collects is completely structured, a SQL database is suitable.

4) **Query language- SQL**

   Software provides history reports based on customized report queries.

3.3 Hardware Resources

1) **Physical Machine-** Intel core i5
2) **Operating System-** CentOS 7(Host or Guest, Any)

3.4 Development Environment

1) **IDE – Netbeans**

   Netbeans comes with JavaFX and JUnit support.

2) **Source code repository-** Github

   Github link: https://github.com/Nandanam/Project.git

4 Activity Flow

4.1 Activity Diagram

Activity diagram is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system.

So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow control by using different elements like fork, join etc.

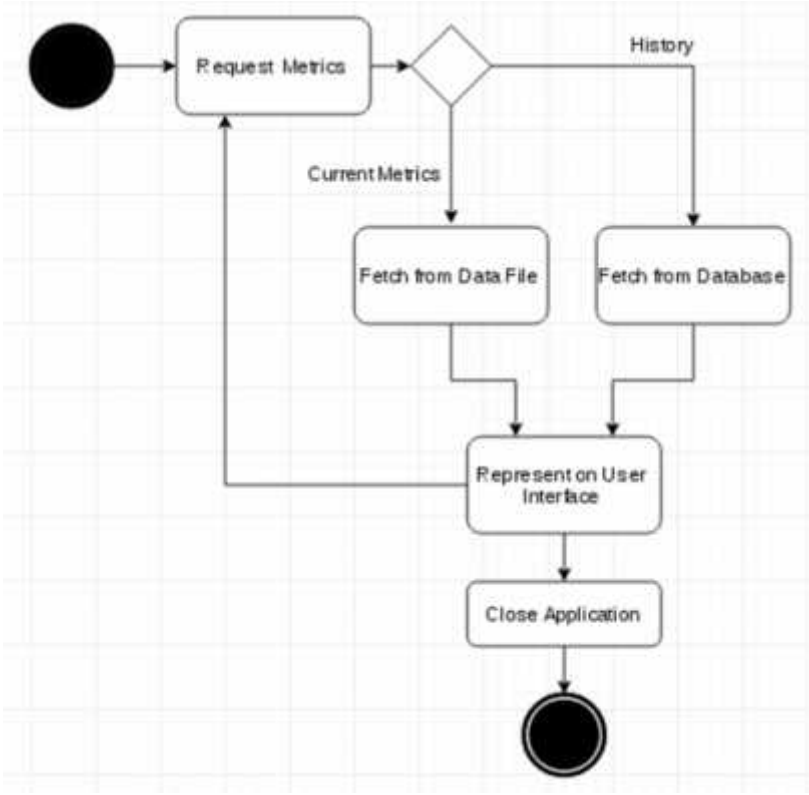


Figure 4.1 Metric Collector Activity Diagram(Fetching Activity)

When the user requests for the information using UI, it can be history or either the current metrics. So, depending on the request of user the metrics are fetched. If current metrics is chosen the metrics are fetched from the data files. If the history of the metrics is chosen, the metrics stored in database is fetched and presented to the user using

user interface. The history of the metrics stored in database which updates database every 10 minutes, database will only hold one hours data.

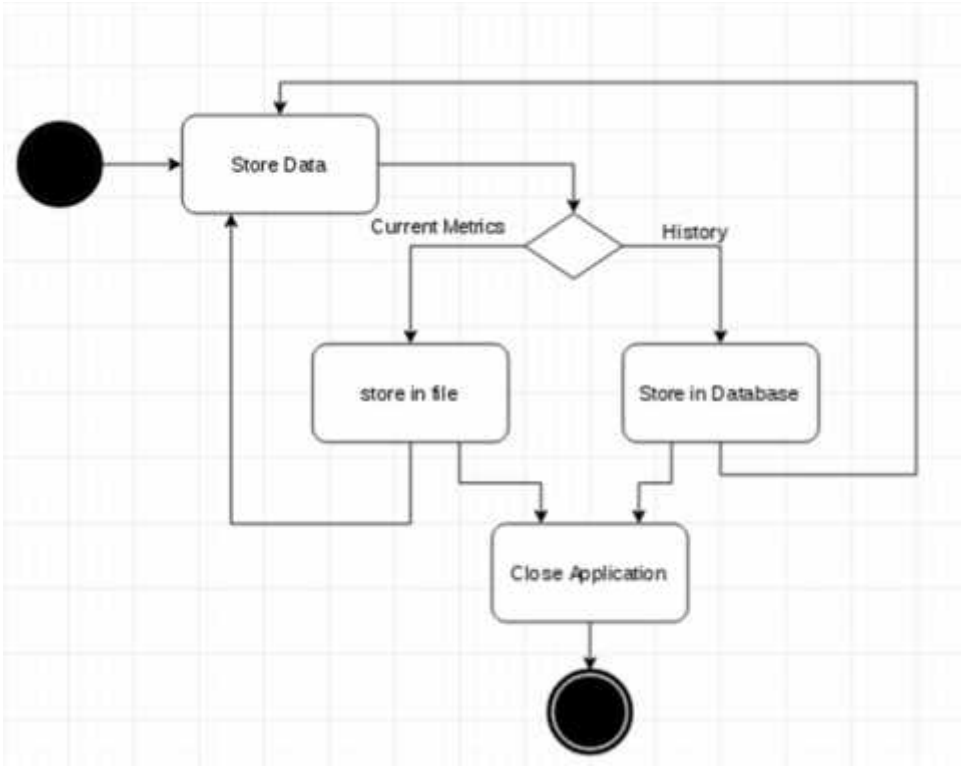


Figure 4.2 Metric Collector Activity Diagram(Storing data Activity)

5 Data Model Design

5.1 Database Tables

Table name: PROCESS_INFO

P_id	Process_name	User	CPU %	Memory %	Status	Priority
Int(5)	Varchar(50)	Varchar(20)	Float(5)	Float(5)	Char(5)	Varchar(10)

The table PROCESS_INFO is responsible for storing the overall information of the process. Following are the attributes

i) **P_id:** P_id stores the current process id. It is unique for every process but may change later on for the same process.

ii) **Process_name:** Stores names of the process.

iii) **User:** Gives the computer user or root.

iv) **CPU %:** This stores information regarding the total usage in percentage for the process.

v) **Memory % :** This stores memory regarding the total usage in percentage for the process.

vi) **Status :** Stores information of process state in a single character.

vii) **Priority:** Stores priority of the process.

Table name : MEMORY_INFO

Total	Used	Free	Shared	Cached
Float(10)	Float(10)	Float(10)	Float(10)	Float(10)

*****All the values shown in this table can be represented in Kb, Mb or GB

The table MEMORY_INFO is responsible for storing the overall memory information. Following are the attributes

I) **Total:** Stores the total amount of RAM the system is having.

II) **Used:** Stores the amount of memory used by the process.

III) **Free:** Stores the amount of memory which is free.

IV) **Shared:** Stores the amount of memory shared among process.

V) **Cached:** Stores the amount of memory cached.

Table name: DISK_INFO

File_system	Size	Used	Available	Use %	Mounted_on
Varchar(20)	Float(5)	Float(5)	Float(5)	Float(5)	Varchar(20)

The table DISK_INFO is responsible for storing the overall disk drive information. Following are the attributes

i) **File_system:** Stores the file system type.

ii) **Size:** Stores size of the file system.

iii) **Used:** Stores the used size of file system.

iv) **Available:** Stores available size of file system.

v) **Use%:** Stores the amount of used disk in percentage.

vi) **Mounted_on:** Stores path of file system in the disk.

Table name: NETWORK_INFO

P_id	User	Program_Description	Sent	Received
Int(5)	Varchar(20)	Varchar(20)	Float(5)	Float(5)

The table NETWORK_INFO is responsible for storing the overall network information. Following are the attributes

i) **P_id:** P_id stores the current process id. It is unique for every process but may change later on for the same process.

ii) **User:** Gives the computer user or root.

iii) **Program_Description:** Stores process description.

iv) **Sent:** Gives sent data over network in kb/sec.

v) **Received:** Gives received data over network in kb/sec.

6 Test Environment

6.1 Test Framework- JUnit

JUnit has become a standard for testing in Java programming language and is supported by almost all IDE's e.g Netbeans, Eclipse etc. So, you work in any standardized IDE environment, you would find the support of JUnit in it. If you are using an IDE for developing Java software, you can create and manage test cases with the help of JUnit wizards which would further enhance your productivity. You can execute a test case or a group of test cases by clicking on some options in the IDE and check the reports instantly using different color combination.

6.2 Code Coverage Tool – TikiOne JaCoCo

The JaCoCoverage Plugin is a Netbeans plugin that enhances the existing NetBeans functionality with new code coverage features.

The plugin works as a transparent additional service that colors all java files according to the unit tests coverage information. With code coverage enabled user continues to work with his/her project in the usual way but can easily view the test coverage of the project classes. The code coverage plugin will update the code coverage data and refresh editors markup every time a unit test (or any selected Ant target) is executed for the project.

7 Deployment

7.1 Packaging Tool- Self Contained Application Packaging

Native packaging was first introduced as a part of the JavaFX 2.2 SDK enabling you to package an application as a native bundle and then installing and running the application without any external dependencies on a system JRE or JavaFX SDK. JavaFX packaging tools provide built-in support for several formats of self-contained application packages. The basic package is simply a single folder on your hard drive that includes all application resources as well as Java Runtime. It can be redistributed as is, or you can build an installable package (for example, EXE or DMG format).

8 Glossary

i) **SDK** – Software Development Kit
ii) **EXE** - Executable file
iii) **IDE** – Integrated Development Environment (Netbeans)
iv) **SQL** – Structured Query Language
v) **GUI** – Graphical User Interface
vi) **JNI** – Java Native Interface
vii) **JDBC** – Java Database Connectivity
viii) **JRE** – Java Runtime Environment
ix) **OS** – Operating System

9 References

http://www.tecmint.com/command-line-tools-to-monitor-linux-performance/
http://www.tecmint.com/12-top-command-examples-in-linux/
https://blogs.oracle.com/pranav/entry/how_to_find_out_cpu_utilizatio
http://www.c-sharpcorner.com/uploadfile/fd0172/how-to-create-celsius-to-fahrenheit-converter-app-and-showin/