

# Smart Integration Monitor

## EPI-Use

### User Manual

---



**CRYSTAL FIRE TECHNOLOGIES**

---

Juandré Barnard	11061015
Pieter Le Roux	10454862
Handré Watkins	10115405

## Table of Contents

1. Purpose .....	3
2. Installation Guide .....	4
2.1 Proxy Server module .....	4
2.2 Interface module deployment .....	5
3. Getting Started.....	7
4. Using the monitoring tool .....	7
4.1. Login.....	7
4.2 The Dashboard .....	8
4.2.2 Requests per URI.....	10
4.2.3 Failure Rate .....	11
4.2.4 Overhead Traffic .....	12
4.2.4 Overall System Performance .....	12
4.3 User Defined Rules.....	13
4.4 Live Activity .....	14
5. Maintaining the system .....	15
6. Conclusion.....	15

## 1. Purpose

The purpose of this application is to assist the client to determine the finer details of the network communications on an application server. With proper use, this monitoring tool will help the client to improve the performance of network communication, to reduce failures and faults in the application server and to get an overall estimation of the current network status.

## 2. Installation Guide

This section highlights and guides the setup and deployment of all components needed to run the Smart integration monitor and all related applications. The deployment will be split into two sub-sections: the deployment of the proxy server and the deployment of backend and web services.

### Physical Requirements

The deployment will be done on a Java enterprise enabled server and both packages will be build using maven as WAR files.

### Server deployment

After the prebuild setup and build, the packages can only be included on an application server. No extra deployment operations will be needed.

#### 2.1 Proxy Server module

##### Pre-build setup

The proxy can be deployed on any internal java application server and communicates with the main application in a one to many fashion. Pre-build setup for the proxy includes modification to the configuration file to point to the correct java communication context factory.

i.e. Configuration file

Factory: default factory

Queue: global: Queue

Run glassfish and go to admin console (//localhost:4848, login with detail if changed)

Go to Resources->JMS Resouces->Destination Resources

Click on Destination Resources and “new”

JDNI Name: “Queue”

Physical Destination Name: “jms”

Select Resource Type: “javax.jms.Queue”

Press OK

##### Build

Building the newly configured proxy request maven to be installed on the local computer to run the command.

mvn Proxy.POM

The WAR file will be created after unit test and sanity test have been performed.

## 2.2 Interface module deployment

### PostgreSQL setup

To access the PostgreSQL database the following procedure needs to be followed:

Install PostgreSQL 9.3 and glassfish 4(any server that can handle war files)

- Set your PostgreSQL to localhost with port 5432 username is “postgres” and password is “1”
- You can change this by changing it in the pool glassfish and change the proxy server
- Create a database with pgadmin3 and call it “dB Monitor”
- Run the sequences queries that are provide with the package, opening it with PostgreSQL and run them in there execution way.
- File is called “Sequences-run first.sql”
- Run the create tables queries that are provide with the package, opening it with PostgreSQL and run them in there execution way.
- File is called “Tables.sql”

### Pre – Build Instructions

- Download *postgresql-9.3-1102.jdbc41.jar* library file from the PostgreSQL website
- Place downloaded file in the following URL's:
  - C:\Program Files\glassfish-4.1\glassfish\lib
  - C:\Program Files\glassfish-4.1\glassfish\domains\domain1\lib
  - C:\Program Files\glassfish-4.1\glassfish\domains\domain1\lib\ext
- Start glassfish server v4, go to admin domain, default <http://localhost:4848/>
- Go to Resources->JDBC->JDBC Connection Pool click on "new"
- Give a name for the pool and choose Resource Type: "javax.sql.ConnectionPoolDataSource"
- Choose in Database Driver Vendor: "Postgresql" and click next
- Select all properties and delete them all. Add a url, username and password properties.
- url: "jdbc:postgresql://localhost:5432/dbMonitor"
- username: "postgres"
- password: "1"
- Click finish
- Go to Resources->JDBC->JDBC Resources, click on new
- Call the JNDI name "postgres" (if you want to change it you have to change the persistence.xml value as well)
- Choose the pool name that you just created
- Then say OK
- Go to Applications and click on Deploy
- Choose file to where the war file is and just say ok and leave the other things to default.
- Then select the deployed file and click on activate (if not activated)
- Then your link to the interface will be <http://localhost:8080/smartintegratedsystem> using command line, but you have to set the whole URL line yourself.

### Build

Building the newly configured proxy requires maven as well and to run the command

Mvn build interface.POM

The WAR file will be created after unit test and sanity test have been performed

### 3. Getting Started

After setting up the application, monitoring the network on an application server is easy. The design is very user friendly and self-explanatory, thus the user will find it easy to navigate to the required destination.

Once the application is deployed on the application server, the system will open the index page in the default browser. Alternately the user should open a web browser and navigate to the following URL: <http://localhost:8080/SmartIntegrationMonitor/>

### 4. Using the monitoring tool

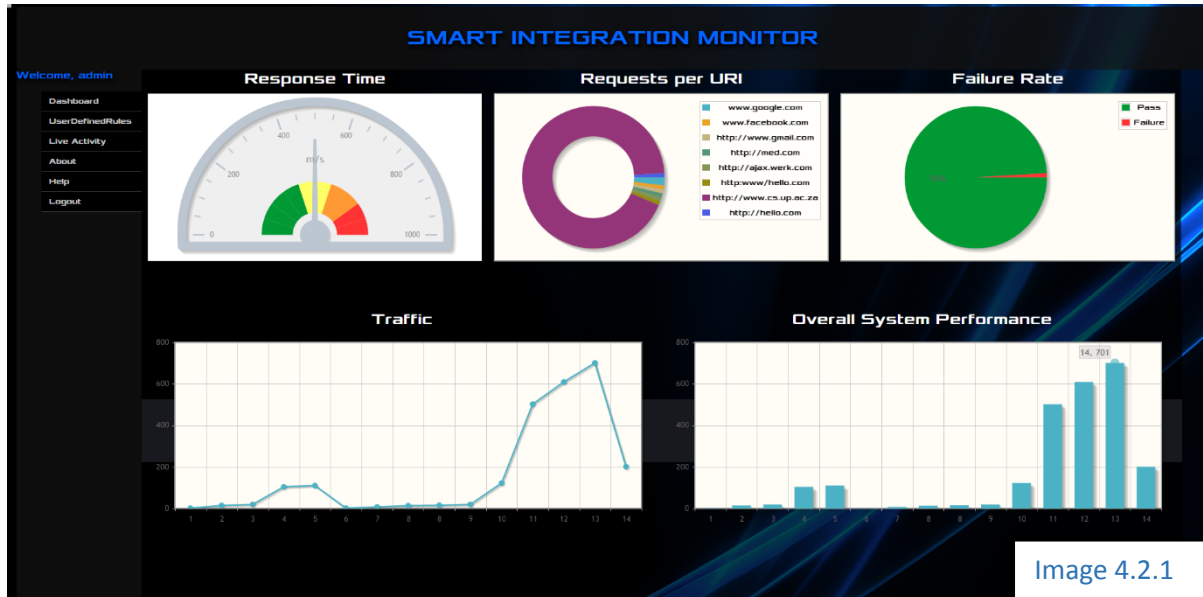
#### 4.1. Login

The first interface of the application (image 4.1) will require the user to login using a valid username and password previously added to the database by the administrator (administrator privileges include the ability to add new users for auditing purposes and not security purposes).



## 4.2 The Dashboard

After logging in with a valid username and password, the application will redirect to the dashboard page (image 4.2.1). This is the main interface of the monitoring tool and will be used to graphically display the data of the network communications on the application server that is being monitored.



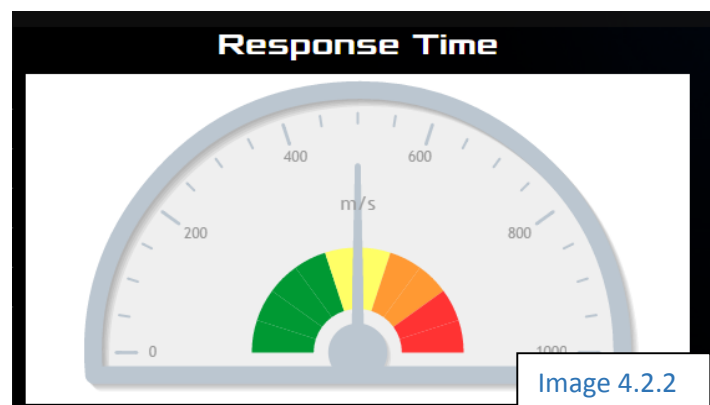
The dashboard page will firstly display a header area to identify the current page of interaction. The next part is a static menu bar to enable efficient navigation through the entire system. This menu will be present on every page of the application server.

The final and main part of the dashboard page is the graphs, charts and gauges. These *widgets* are used to visually represent the data of the current network statistics to easily assess the current network.

The dashboard displays five critical network variables, namely: Response time, Requests per URI, Failure Rate, Overhead Traffic and Overall System Performance. Each of these will be discussed in detail.

### 4.2.1 Response Time

On the dashboard the response time is represented on the dashboard by a gauge (image 4.2.2) that mimics a speedometer of a vehicle. The current response time is displayed by the needle and is measured in milliseconds per response.





Once you click in the Response Time link (the gauge title), it navigates you to the main response time page. Here you will be presented with a line graph displaying the latest 15 response times, as well as a table to get more detailed and accurate response times coupled with the relevant URI and time of response (image 4.2.3).

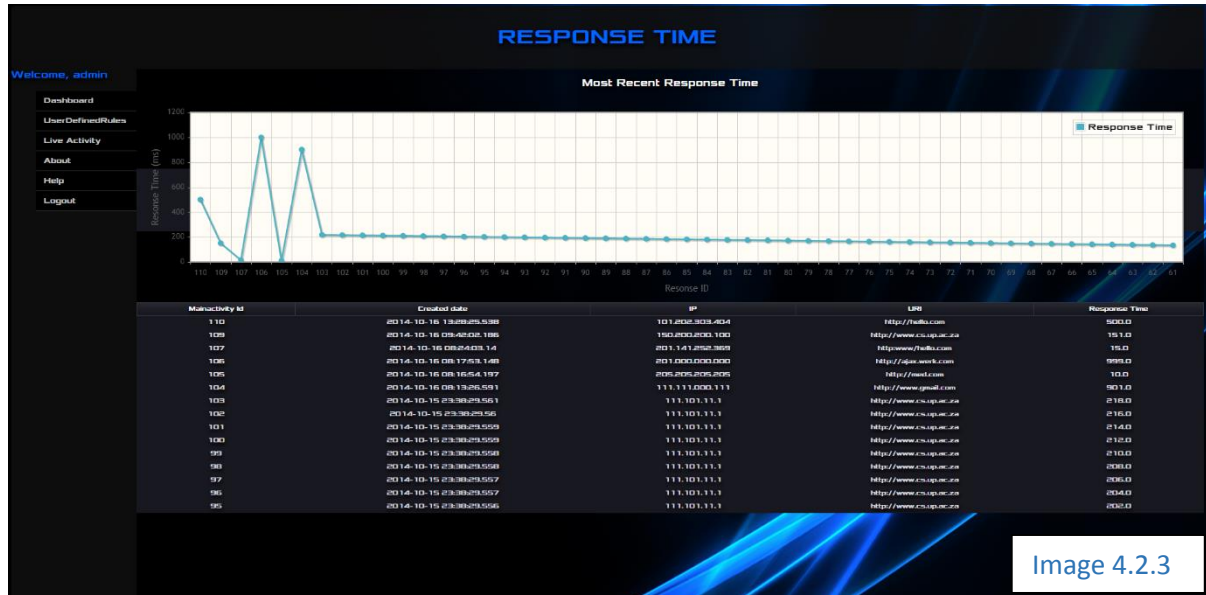


Image 4.2.3

#### 4.2.2 Requests per URI

The second widget shows the average request per URI in the form of a doughnut chart, where each URI is represented by a different color (image 4.2.4).

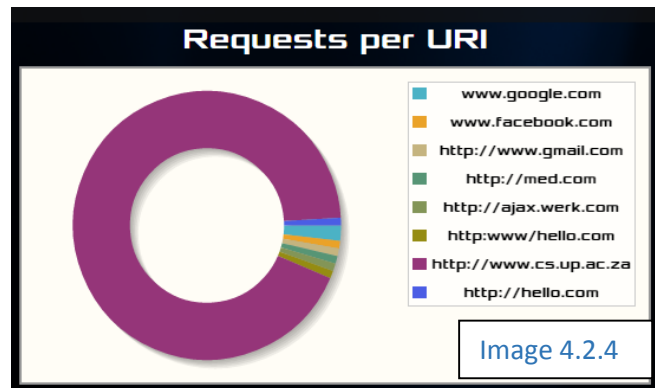


Image 4.2.4

The heading link then redirects you to the main requests per URI page that will once again give you the doughnut chart as well as a data table, listing every different URI and then showing how many requests were made per URI (image 4.2.5).

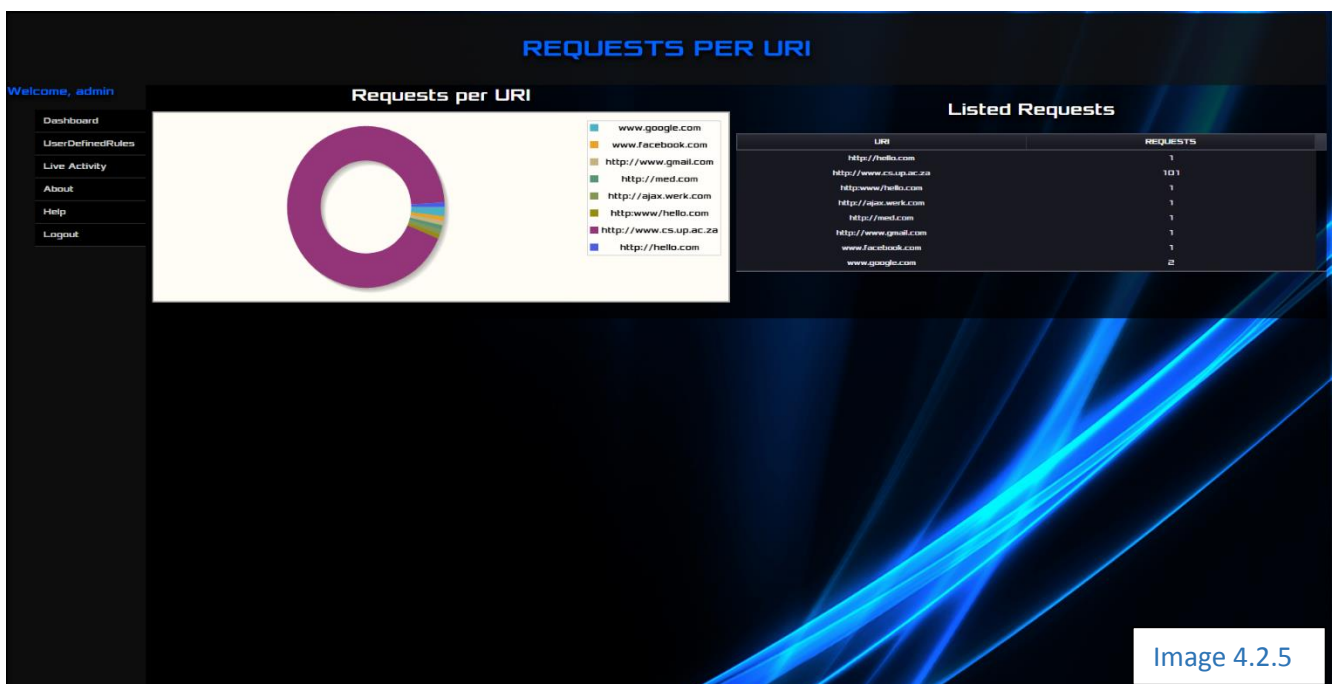
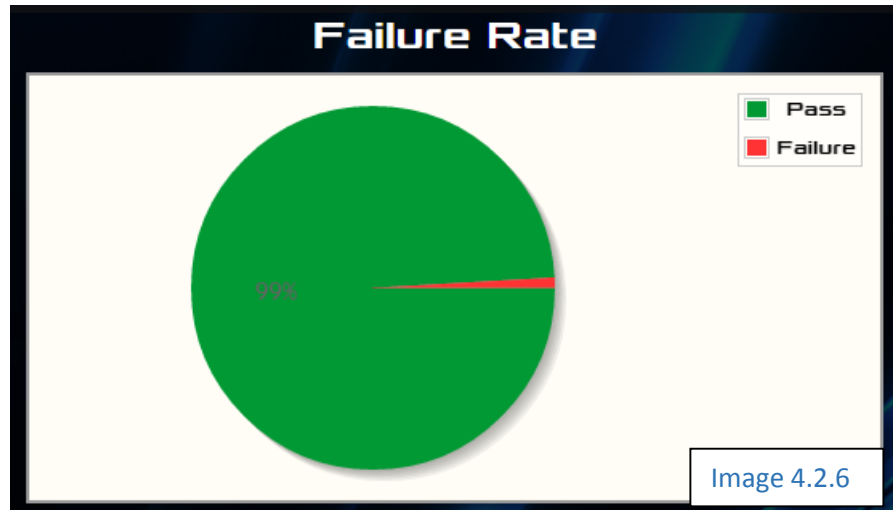


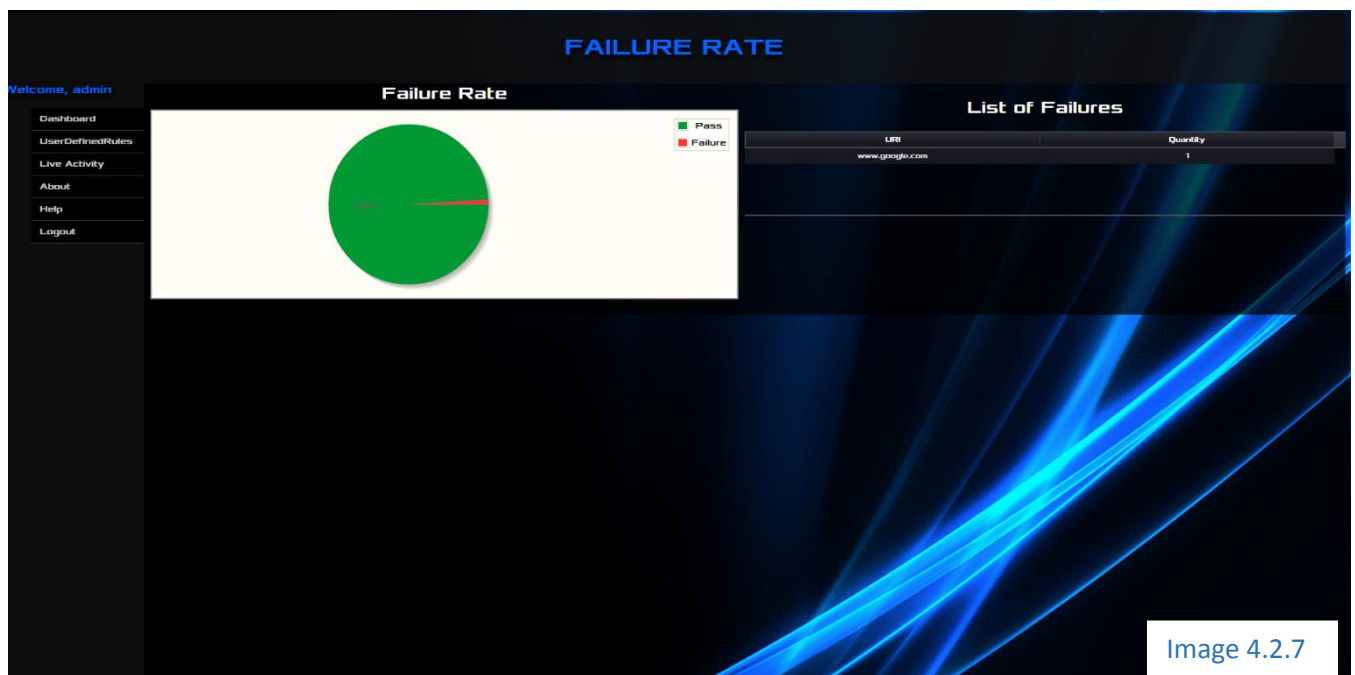
Image 4.2.5

### 4.2.3 Failure Rate

Next we have the Failure Rate widget (image 4.2.6). It is displayed in a pie chart that contains segments, failed calls and successful calls. This shows the amount of failures (red) as a percentage, per the amount of successful calls (green).



The Failure Rate heading will then redirect you to the main failure rate page (image 4.2.7). This page will once again show you the failure rate repressed by a pie chart as well as a list of failed request, coupled with the corresponding URI, in a data table.



#### 4.2.4 Overhead Traffic

The next widget displays the overhead traffic of the network and this gives you an overall idea of how busy the current application server's network communication is (image4.2.8).

The basic idea behind traffic is to display how many requests there are between the consumer and provider.

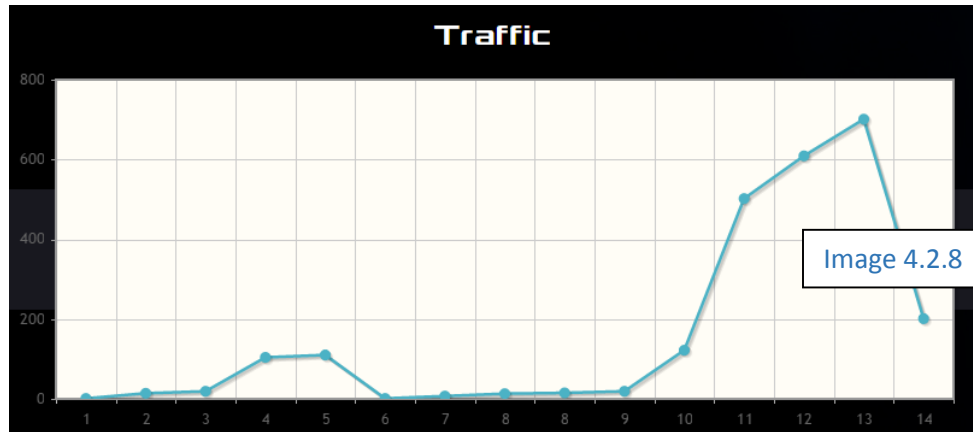


Image 4.2.8

#### 4.2.4 Overall System Performance

The overall system performance shows the user how the network on the application server is performing when weighing up all of its variables (images 4.2.9). It does this by measuring all of the requests made, and then comparing the amount of failed requests. This then gives you a percentage and will be displayed as *system health* that indicates if the network on the application server is not failing (health that is less than 50%) and how good the communication is in the network.

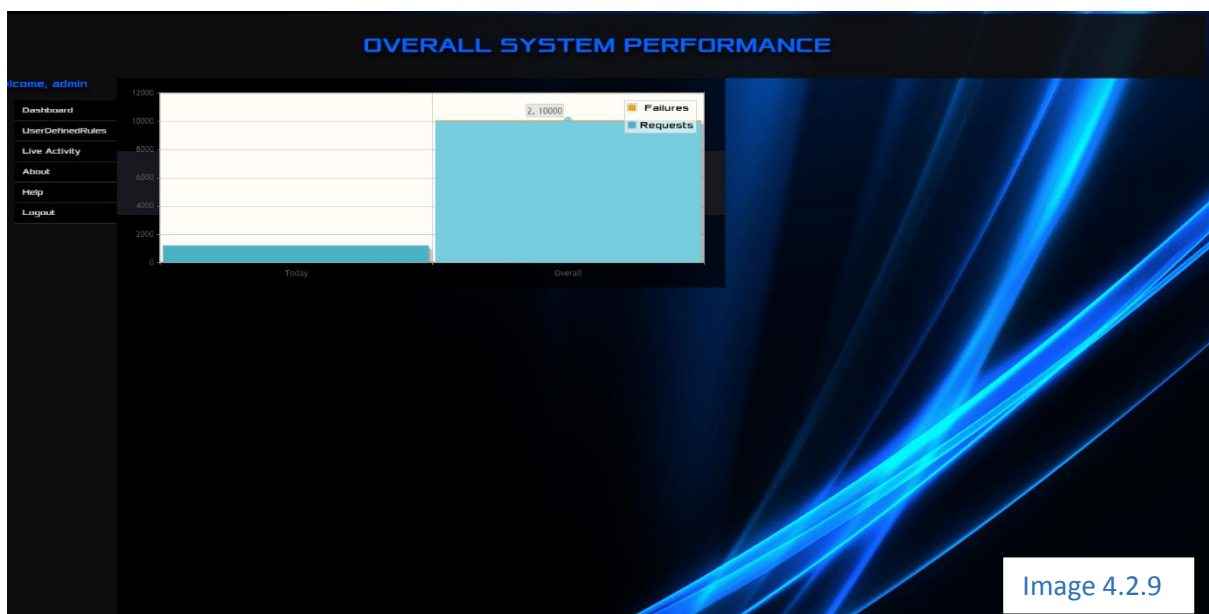
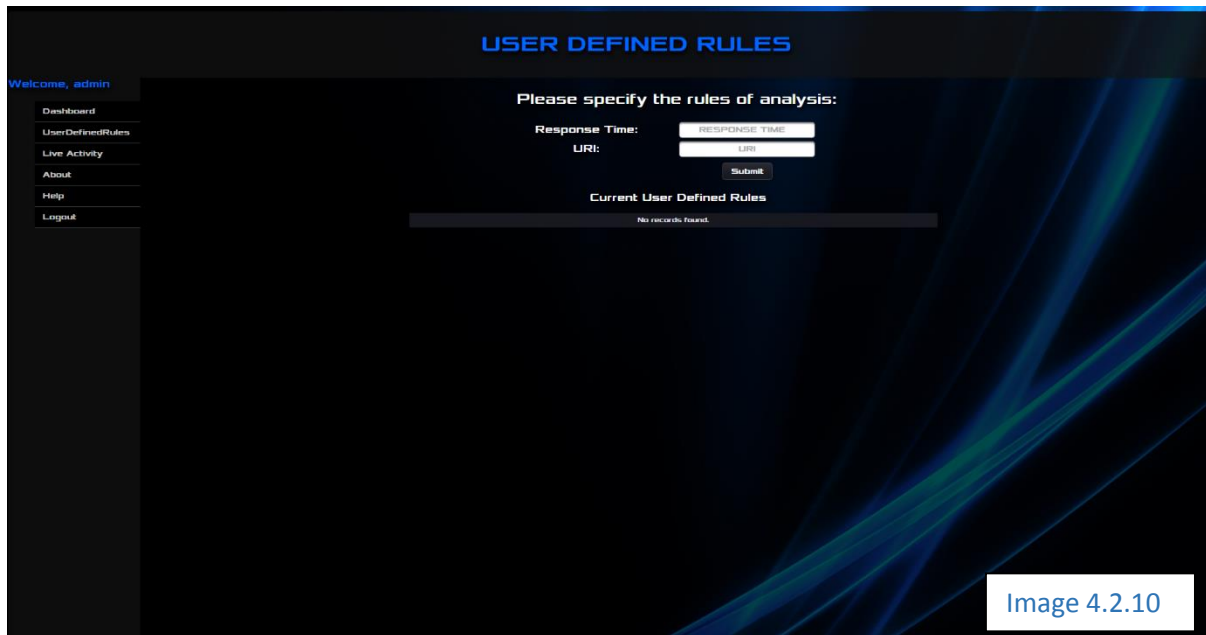


Image 4.2.9

### 4.3 User Defined Rules

The User Defined Rules is the page where the client can specify their own values to have the system function according to their liking (image 4.2.10).

This specific page will allow the user to specify a response time and address it to a certain URI. This will then be the cutoff threshold, that is, the processes that have a lower response time will be identified as successful calls, and those with a higher response time, will be identified as failed calls.



**USER DEFINED RULES**

Welcome, admin

Dashboard  
UserDefinedRules  
Live Activity  
About  
Help  
Logout

Please specify the rules of analysis:

Response Time:

URI:

Current User Defined Rules

No records found.
-------------------

Image 4.2.10

## 4.4 Live Activity

The Live Activity page is the actual essence of the system (image). Here you can see the accurate and live statistics of the current network in the application server.

Firstly you will see the Main activity table that will show you the process ID, data and time of the request, the IP address, the Uniform Identifier and the response time calculated in milliseconds.

After that the distressed activity table will show all of the failed requests of the network along with similar details.

Main Activity Id	Created date	IP	URL	Response Time
110	2014-10-19 13:32:59.538	101.200.200.404	http://hello.com	500.0
109	2014-10-19 09:40:02.186	100.200.200.100	http://www.csug.ac.jp	15.0
107	2014-10-19 08:45:03.14	201.141.200.389	http://www.fedex.com	15.0
106	2014-10-19 08:17:26.146	201.200.200.200	http://open.ubuntu.com	999.0
105	2014-10-19 08:16:54.197	200.200.200.200	http://www.igmp.com	10.0
104	2014-10-19 08:15:26.591	111.111.000.111	http://www.gmail.com	9801.0
103	2014-10-19 21:38:29.561	111.101.11.1	http://www.csug.ac.jp	218.0
102	2014-10-19 21:38:29.565	111.101.11.1	http://www.csug.ac.jp	216.0
101	2014-10-19 21:38:29.559	111.101.11.1	http://www.csug.ac.jp	214.0
100	2014-10-19 21:38:29.559	111.101.11.1	http://www.csug.ac.jp	212.0
99	2014-10-19 21:38:29.558	111.101.11.1	http://www.csug.ac.jp	210.0
98	2014-10-19 21:38:29.558	111.101.11.1	http://www.csug.ac.jp	208.0
97	2014-10-19 21:38:29.557	111.101.11.1	http://www.csug.ac.jp	206.0
96	2014-10-19 21:38:29.557	111.101.11.1	http://www.csug.ac.jp	204.0
95	2014-10-19 21:38:29.556	111.101.11.1	http://www.csug.ac.jp	202.0
94	2014-10-19 21:38:29.556	111.101.11.1	http://www.csug.ac.jp	200.0
93	2014-10-19 21:38:29.555	111.101.11.1	http://www.csug.ac.jp	198.0

Distress Activity Id	Created date	IP	URL	Time Expired	Response Time
1	2014-10-19 17:10:50.090	101.200.200.145	www.google.com	15	2500.00

You will also have an option to download the current data of each table by clicking on the relevant button. The formats that are available is xml and csv.



## 5. Maintaining the system

The way in which the system is designed, it will permanently run and monitor the application server that it is deployed upon, even if that application server is stopped.

Once you need to monitor different application servers or even multiple servers, you simply deploy the proxy on the required server and proceed to the monitoring tool to see the network activity.

All you need to do is to login using your username and password as a new session will be created.

## 6. Conclusion

After implementing this system, you should have a better overall description of how your network is functioning within the application server. This knowledge will help you to increase the performance of the communication process by reducing failures and faults and ensuring effective connections are transporting data uninterruptedly and at an efficient speed.