EVALUATING HTTP PERFORMANCE FROM STREAMS

PROJECT SPECIFICATION

- **Team Name:** Nagios
- **Team Members:**
    - Atla Prashant
    - Chilukuri, Megh Phani Dutt
    - Garg, Prafull
    - Grandhi, Veera Venkata Santosh S G
    - Kalidindi, Rajeev Varma
    - Kolli, Samuel Sushanth
    - Madala, Sravya
    - Musinada, Suren
    - Naguru, Sriram Prashanth
    - Peddireddy, Divya
    - Rajana, Poojitha
- **Document Type:** Project Specification
- **Version Number:** Version 1.3
- **Publication Date:** May 20th, 2015

1. INTRODUCTION

In this document, we present a proposal on how to evaluate HTTP performance from streams. From the feedback of CEO, the updated version release is v1.3 where we elucidate on how to develop a tool to monitor the HTTP traffic.

The document is partitioned into various sections. Section I gives an overview of the customer's needs and the urge to develop a new product. Section II suggests a solution addressing the demands of the customer. The limitations of what is included and excluded in this project are explained in Section III. Section IV depicts the work breakdown structure describing the toll gates and milestones. Section V refers to the Configuration management. Section VI states Progress tracking. Section VII states Quality control. Section VIII includes Risk Management and finally, section IX includes System release plan.

Release v1.3 on 2015-05-20

- Updated release

Version history is as follows:

PUBLICATION DATE	VERSION	DESCRIPTION	CHANGES
2015-05-20	v1.3	Updated Version	Updated the following from the feedback of CEO. • Risk management is modified. Refer section 11.
2015-05-14	v1.2	Updated Version	Updated the following from the feedback of CEO. • Version management of source code is included. • Progress tracking is modified. See section 9 • Risk management is changed. See section 11 • Plan and schedule of the tests are modified.

2015-05-05	v1.1	Updated Version	Updated the following from the feedback of CEO. • Changes made to system architecture • RESTful API • Programming is assigned to all the team members. • Version management is explained using Github. • System building and Release management is changed. See section 8. • Systematic process of implementing the project is changed. See section 9. • Risks in the project are mentioned and changed. • Included different types of tests and dates are booked in the Projectlibre plan. • Developer documentation is changed, See section 12.3.3.
2015-04-27	v1.0	Initial Release	

2. GLOSSARY AND ABBREVIATIONS

HTTP: Hypertext Transfer Protocol

It is a protocol at the application level for communication of data between the network elements such as clients and servers.

GUI: Graphical User Interface

An interface which allows the users to communicate with the electronic devices through visual icons. In some cases, it contains audio feedback as well as voice control.

DPMI: Distributed Passive Measurement Infrastructure

This interface is used to read the data stream at various measuring points.

RESTful: Representational State Transfer

An architectural pattern to improve portability, scalability of the system.

3. BACKGROUND

The customer is a data centre company providing rack space and rudimentary networking to clients. They are basically web service providers. The increased demand in live streaming over HTTP generates more traffic towards the servers.

Thus, the clients have expressed their displeasure over the services provided to them. Slow request-response time, poor service quality are a few of such problems. There is an overall reduction in the quality of the internet services provided to the end users.

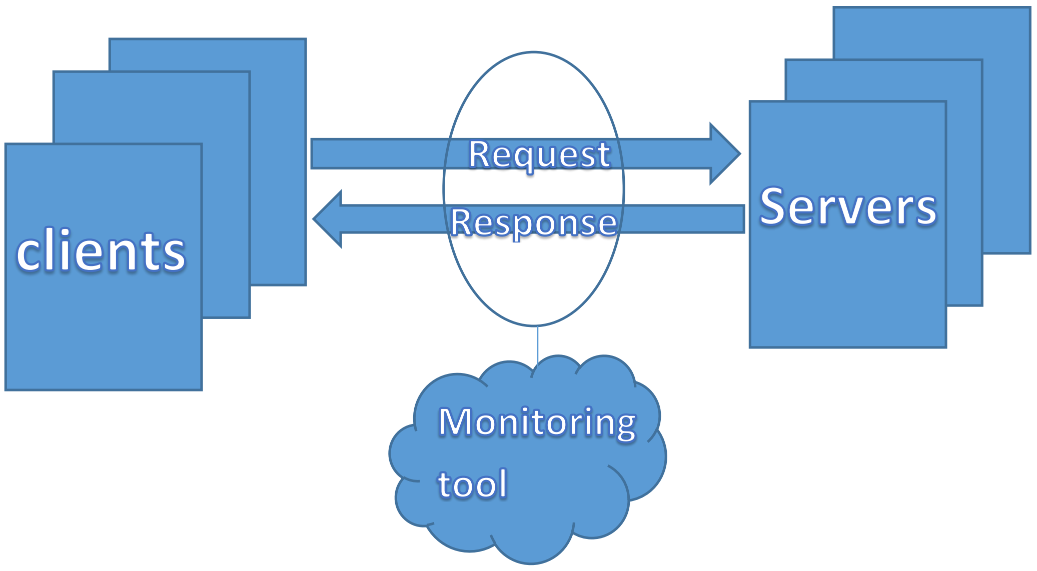
The customer requires a solution to the above issues along with a picturesque way to exhibit these improved services with the help of a tool. Quick responses are expected. This tool thus monitors the HTTP performance of the servers present in the data centre.

4. PROPOSED SOLUTION

The following methods are devised to meet the requirements of the customer.

System architecture

A typical internet consists of multiple clients and servers. The clients are users. The servers are located at the data centre.



In the above figure, an HTTP request is **for** an HTML file, a document or a movie**.** An HTTP response carries the data back (which is requested) or an error (when the resource is not available). In some cases, the response carries a redirection code, when the resource is moved.

We plan to develop a monitoring tool which observes the HTTP request-response time per server, bit rate and loss requests. This tool maintains a historical track record for about four weeks. We use a DPMI to read the data stream. We use time series graphs to represent this data. Statistical inference like standard deviation and confidence analysis can be carried out on this data for further investigation.

The performance of the servers is observed. We calculate the request-response rate, bit rate and loss requests associated with each server. We plot a graph which displays these performance metrics of all the servers in the data centre. We also display these metrics for a cluster of servers for comparison. A cluster of servers is a group of independent servers working together to provide high availability of services to the clients.

We assign threshold levels to the servers. The threshold levels are normal, warning and critical. A notification is sent when threshold limits are exceeded. These threshold limits are configured for each device which we are monitoring.

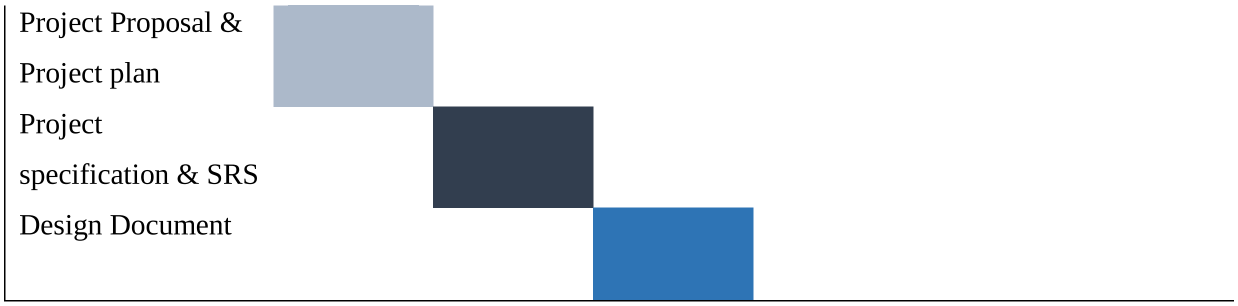
A simple web interface with user authentication will be provided to display the observed statistics in the form of graphs. After successful login, a dashboard is provided which displays the critical problems associated with each server. We can also drill down and zoom-in on to each device individually for detailed information.

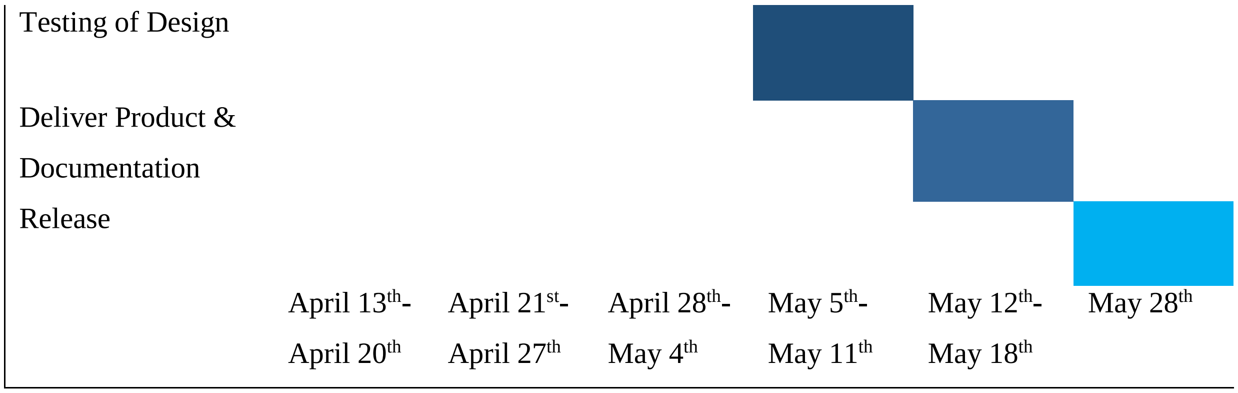
RESTful API is an interface which can be used to export data to a third party and import data from the third party in any agreed-upon format (for example, time series data or any other format). Graphical cross correlation between our data and provided data can be performed for further analysis.

5. LIMITATIONS

The tool is for monitoring the HTTP traffic over multiple data streams. It is applicable for monitoring of services like request-response time. It informs the faults to the observer by issuing proper fault notifications. The fault notifications can be sorted by activating a software that allows us to take proper intervention i.e., send E-mail. This tool cannot capture the entire user information. Only truncated packets are available for observation. Sampling of raw data obtained at various measuring points might be required. Modelling the client behaviour is also an important limitation. Server performance metrics like disk usage and memory utilisation and also network metrics like delay, reliability and error rate are beyond the scope of this project.

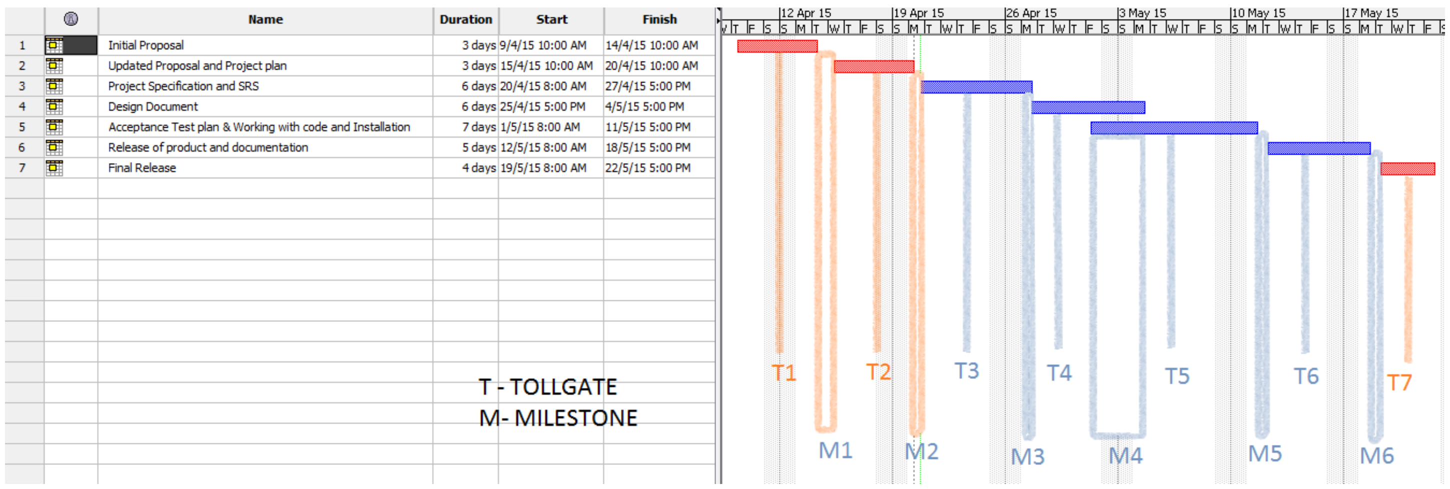
6. TIME PLAN





A Gantt chart showing the proposed time plan for project completion

- 13-04-2015: Project Proposal
- 20-04-2015: Project Plan
- 21-04-2015: Project Specification
- 27-04-2015: Software Requirement Specifications (SRS)
- 04-05-2015: Design Document
- 11-05-2015: Testing of Design, Project Demo
- 18-05-2015: Acceptance of design plan, deliver product, documentation
- 28-05-2015: Release



		Name	Duration	Start	Finish
1		Initial Proposal	3 days	9/4/15 10:00 AM	14/4/15 10:00 AM
2		Updated Proposal and Project plan	3 days	15/4/15 10:00 AM	20/4/15 10:00 AM
3		Project Specification and SRS	6 days	20/4/15 8:00 AM	27/4/15 5:00 PM
4		Design Document	6 days	25/4/15 5:00 PM	4/5/15 5:00 PM
5		Acceptance Test plan & Working with code and Installation	7 days	1/5/15 8:00 AM	11/5/15 5:00 PM
6		Release of product and documentation	5 days	12/5/15 8:00 AM	18/5/15 5:00 PM
7		Final Release	4 days	19/5/15 8:00 AM	22/5/15 5:00 PM

T - TOLLGATE
M- MILESTONE

A work break down structure is designed to meet the deadlines within the allotted time. The various targets to be achieved are classified into milestones (for the development team) and milestones (for the customer/CEO).

The project duration is of six weeks. Each developer is supposed to work for 8hours/business day. This is equivalent to 30days/240 hours per person. The entire team is supposed to work for sixty weeks on a full time basis (as on average, the members per team are 10). This time is equivalent to a one and a half man year.

The below table gives the division of Milestones and Tollgates:

Milestones:

Milestone 1	Project Plan
Milestone 2	Design Document
Milestone 3	Project Documentation

Tollgates:

Tollgate 1	Project Proposal
Tollgate 2	Software Requirement Specification
Tollgate 3	Acceptance of Design Plan
Tollgate 4	Final release of the Product

Table: Milestones and Tollgates

Milestones are primarily concerned with the project team. But, when it comes to project plan, both the CEO and Customer are involved. Similarly, in Tollgates, the acceptance of design plan involves discussion between the team members, Customer and CEO. The final product needs to be accepted both by the customer and CEO.

7. Project Organization

In project organisation the work in the Nagios team is divided to all the team members equally. The assigning of work is shown in the below table.

Work Assigned	Team Members
1. Programming A. Server side B. Client side	 Suren, Phani Dutt, Vera Venkata Santosh S G, Samuel Sushanth, Divya, Poojitha, Praful Atla Prashant, Prashanth, Sravya Madala, Rajeev Varma
2. Documentation	Poojitha, Samuel Sushanth
3. Testing	Prashanth
4. Management	Garg Prafull

8. Configuration Management

- **Version Management :**

  The code developed for every component is assigned a version number along with document. Every time the code is changed, required changes are made in the version number and document. For example, if we have taken project proposal as the type of document the version release of that document is version 1.0 which is the major release. After the comments received by the CEO, required changes are made and the version is changed for a minor release i.e. version 1.1.

  The version management of source code is tracked by the codelines where codelines are the sequence of source code versions. These versions of the source code are tracked using the github and it is similar to the documents. Github helps for tracking of the source code, when different or more Number of team members edit the same file.

  Using the github the version management is updated by using the below commands and the procedure. Firstly, we need to generate a SSH key and it should be uploaded in the git then the git is cloned. After that for uploading the file we use the below commands and by using this github we can also modify the files or the documents and the source code which are uploaded in the git.

1. Git add – Used for adding the changes
2. Git commit –m “commit” – Indicates where the changes are made
3. Git push – used for pushing the documents
4. Git pull – Appear changes with their versions

- **System Building :**

  The total work plan is discussed among the team members and the information is exchanged among them. An expected blueprint is prepared with the initial idea. The blueprint has initial code with expected changes in the code. The initial version is submitted to both CEO and Customer. After the feedback given by them, required changes are made. The same procedure is repeated for all the versions of source code, product and documentation and changes are recorded.

  Here we are going to use the system builder where our own source code or the script is prepared for both back end and front end. By using the build script generation the source code or the script is built.

- **Release management:**

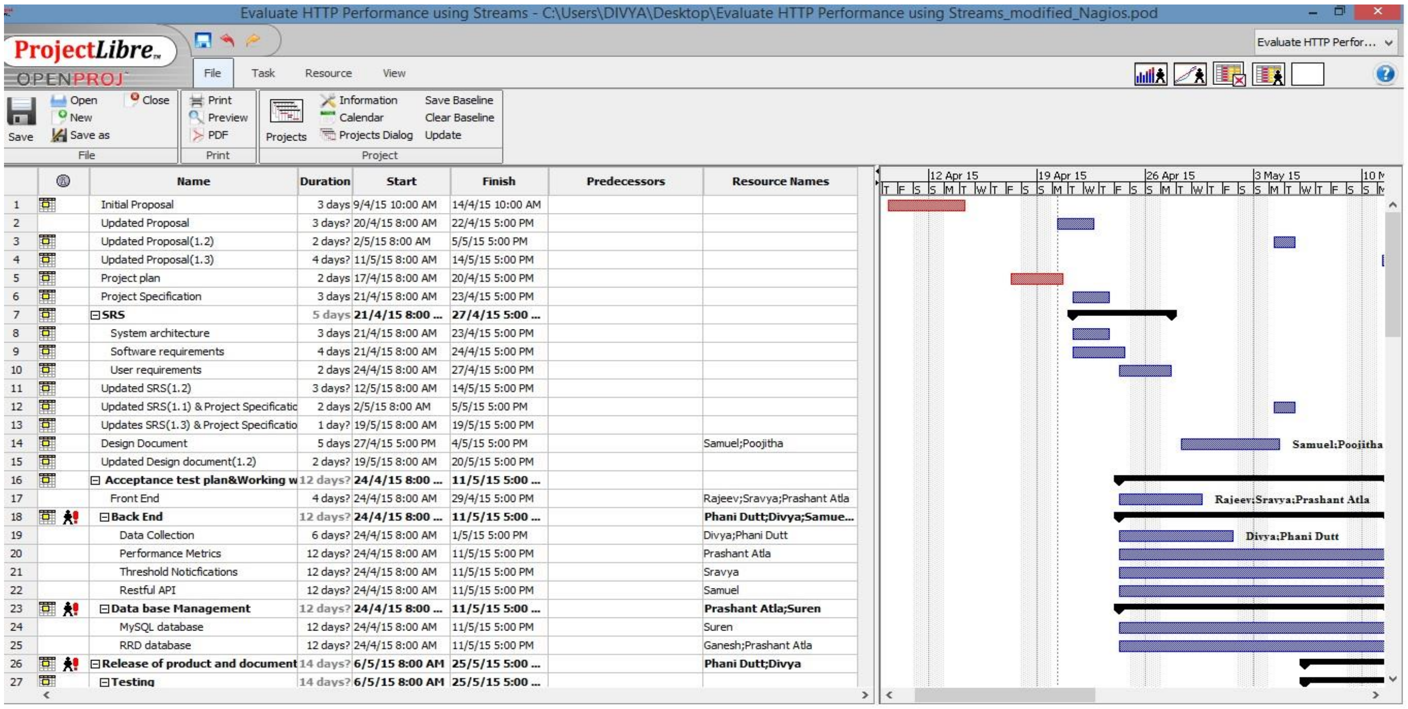
The testing team verifies each and every component of the product thoroughly for bugs. Each time version number is allotted for each component when a bug is rectified. Before release of the product, the tool is tested under several conditions. Then a documentation is prepared with a clear view of installing and setting up the product and a checklist is prepared which is used to verify the release bundle.

Before every release, the product is updated from the feedback of customer. A history of changes made for each update is documented.

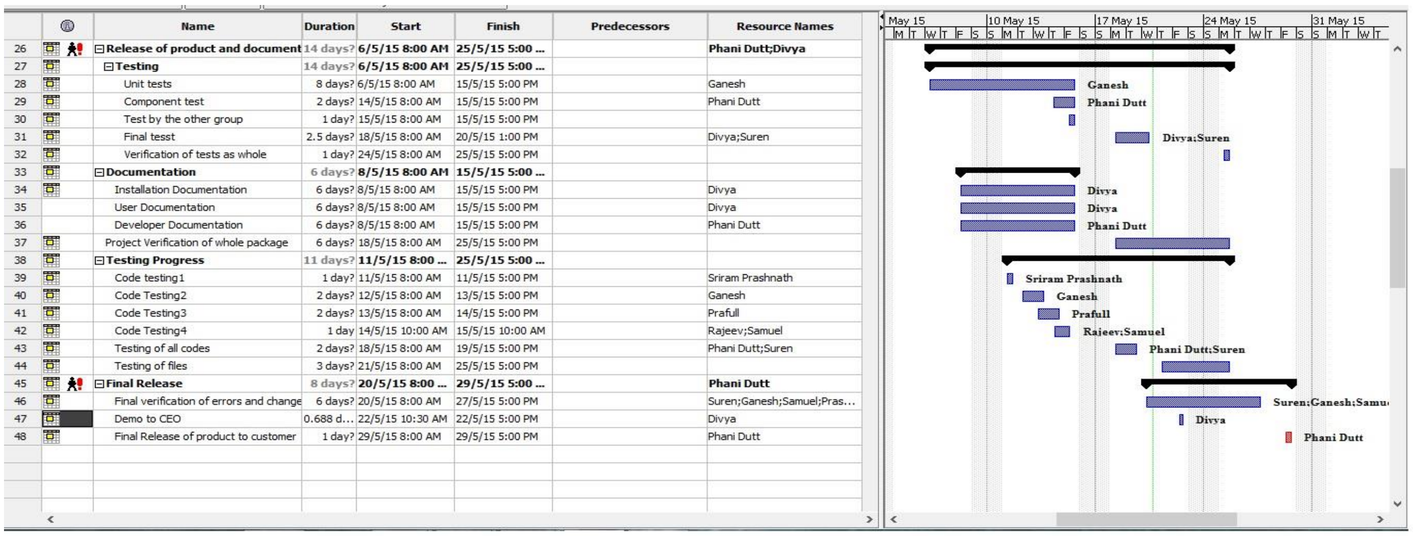
9. Progress Tracking

The tasks are assigned by the manager to the team members. He checks whether the tasks are performed within the required time.

The table above gives how the manager divided the tasks among the team members and assigned deadlines to them. The progress is discussed among the team members. The progress is discussed by keeping the meetings among the team members.



	Name	Duration	Start	Finish	Predecessors	Resource Names
1	Initial Proposal	3 days	9/4/15 10:00 AM	14/4/15 10:00 AM		
2	Updated Proposal	3 days?	20/4/15 8:00 AM	22/4/15 5:00 PM		
3	Updated Proposal(1.2)	2 days?	2/5/15 8:00 AM	5/5/15 5:00 PM		
4	Updated Proposal(1.3)	4 days?	11/5/15 8:00 AM	14/5/15 5:00 PM		
5	Project plan	2 days	17/4/15 8:00 AM	20/4/15 5:00 PM		
6	Project Specification	3 days	21/4/15 8:00 AM	23/4/15 5:00 PM		
7	**⊟SRS**	5 days	**21/4/15 8:00 ...**	**27/4/15 5:00 ...**		
8	System architecture	3 days	21/4/15 8:00 AM	23/4/15 5:00 PM		
9	Software requirements	4 days	21/4/15 8:00 AM	24/4/15 5:00 PM		
10	User requirements	2 days	24/4/15 8:00 AM	27/4/15 5:00 PM		
11	Updated SRS(1.2)	3 days?	12/5/15 8:00 AM	14/5/15 5:00 PM		
12	Updated SRS(1.1) & Project Specificatic	2 days	2/5/15 8:00 AM	5/5/15 5:00 PM		
13	Updates SRS(1.3) & Project Specificatio	1 day?	19/5/15 8:00 AM	19/5/15 5:00 PM		
14	Design Document	5 days	27/4/15 5:00 PM	4/5/15 5:00 PM		Samuel;Poojitha
15	Updated Design document(1.2)	2 days?	19/5/15 8:00 AM	20/5/15 5:00 PM		
16	**⊟ Acceptance test plan&Working w**	12 days?	**24/4/15 8:00 ...**	**11/5/15 5:00 ...**		
17	Front End	4 days?	24/4/15 8:00 AM	29/4/15 5:00 PM		Rajeev;Sravya;Prashant Atla
18	**⊟Back End**	12 days?	**24/4/15 8:00 ...**	**11/5/15 5:00 ...**		**Phani Dutt;Divya;Samue...**
19	Data Collection	6 days?	24/4/15 8:00 AM	1/5/15 5:00 PM		Divya;Phani Dutt
20	Performance Metrics	12 days?	24/4/15 8:00 AM	11/5/15 5:00 PM		Prashant Atla
21	Threshold Noticfications	12 days?	24/4/15 8:00 AM	11/5/15 5:00 PM		Sravya
22	Restful API	12 days?	24/4/15 8:00 AM	11/5/15 5:00 PM		Samuel
23	**⊟Data base Management**	12 days?	**24/4/15 8:00 ...**	**11/5/15 5:00 ...**		**Prashant Atla;Suren**
24	MySQL database	12 days?	24/4/15 8:00 AM	11/5/15 5:00 PM		Suren
25	RRD database	12 days?	24/4/15 8:00 AM	11/5/15 5:00 PM		Ganesh;Prashant Atla
26	**⊟Release of product and document**	14 days?	**6/5/15 8:00 AM**	**25/5/15 5:00 ...**		**Phani Dutt;Divya**
27	**⊟Testing**	14 days?	**6/5/15 8:00 AM**	**25/5/15 5:00 ...**		



	Name	Duration	Start	Finish	Predecessors	Resource Names
26	**⊟Release of product and document**	14 days?	**6/5/15 8:00 AM**	**25/5/15 5:00 ...**		**Phani Dutt;Divya**
27	**⊟Testing**	14 days?	**6/5/15 8:00 AM**	**25/5/15 5:00 ...**		
28	Unit tests	8 days?	6/5/15 8:00 AM	15/5/15 5:00 PM		Ganesh
29	Component test	2 days?	14/5/15 8:00 AM	15/5/15 5:00 PM		Phani Dutt
30	Test by the other group	1 day?	15/5/15 8:00 AM	15/5/15 5:00 PM		
31	Final tesst	2.5 days?	18/5/15 8:00 AM	20/5/15 1:00 PM		Divya;Suren
32	Verification of tests as whole	1 day?	24/5/15 8:00 AM	25/5/15 5:00 PM		
33	**⊟Documentation**	6 days?	**8/5/15 8:00 AM**	**15/5/15 5:00 ...**		
34	Installation Documentation	6 days?	8/5/15 8:00 AM	15/5/15 5:00 PM		Divya
35	User Documentation	6 days?	8/5/15 8:00 AM	15/5/15 5:00 PM		Divya
36	Developer Documentation	6 days?	8/5/15 8:00 AM	15/5/15 5:00 PM		Phani Dutt
37	Project Verification of whole package	6 days?	18/5/15 8:00 AM	25/5/15 5:00 PM		
38	**⊟Testing Progress**	11 days?	**11/5/15 8:00 ...**	**25/5/15 5:00 ...**		
39	Code testing1	1 day?	11/5/15 8:00 AM	11/5/15 5:00 PM		Sriram Prashnath
40	Code Testing2	2 days?	12/5/15 8:00 AM	13/5/15 5:00 PM		Ganesh
41	Code Testing3	2 days?	13/5/15 8:00 AM	14/5/15 5:00 PM		Prafull
42	Code Testing4	1 day	14/5/15 10:00 AM	15/5/15 10:00 AM		Rajeev;Samuel
43	Testing of all codes	2 days?	18/5/15 8:00 AM	19/5/15 5:00 PM		Phani Dutt;Suren
44	Testing of files	3 days?	21/5/15 8:00 AM	25/5/15 5:00 PM		
45	**⊟Final Release**	8 days?	**20/5/15 8:00 ...**	**29/5/15 5:00 ...**		**Phani Dutt**
46	Final verification of errors and change	6 days?	20/5/15 8:00 AM	27/5/15 5:00 PM		Suren;Ganesh;Samuel;Pras...
47	Demo to CEO	0.688 d...	22/5/15 10:30 AM	22/5/15 5:00 PM		Divya
48	Final Release of product to customer	1 day?	29/5/15 8:00 AM	29/5/15 5:00 PM		Phani Dutt

10. Quality Control

The quality control measures the quality of the product. For delivering good quality, the testing person in the team is responsible to check and run tests, such that the product covers all the user requirements. The following are the reviews taken to control the quality of the product.

1. Proposed solution Review
2. Project plan and design Review
3. Requirements Review
4. Code Review
5. Standard Review
6. Validation tests are performed on the tool and feedback is collected from the customer each time.
7. Proper documentation for the code.

We require a specific dates for the above reviews. The start and end dates were mentioned in the Project Libre plan and the entire team requires a meeting were the tester in the team is responsible to test the source code.

11. Risk Management

It deals with the risks faced when we develop the software and is an important job for a project manager, their impact on the components and their effect. So, risk management provides a way to avoid the risks and in certain cases to minimize the risks.

The below are the Risks:

1. Component failure.
2. If the assigned person leaves the task unfinished then it would be difficult for a new person to continue the task. So, it leads to delay in the schedule.
3. Resources availability to the team.
4. If there are some bugs in the developed product then the performance of that system degrades when it is compared to the actual system.
5. If there is a competitor producing the same software then there is a risk of stealing the market by other software.

   The strategies to deal with the above are:

1. Each component must be verified thoroughly and tested several times and script must be verified.
2. Assigning a right person for a particular task.
3. The manager of the team reports to the CEO about the non-availability of resources.
4. The source code and documentation should be perfectly done without bugs.
5. Easy user interface and a multiple accessing environment then the competitor software is developed

12. System release plan

12.1 Testing plan:

The whole product is developed and test is performed. The testing person responsible will test the tool to check whether all the requirements of the tool are correct or not. Every component in the tool is checked for bugs. The tool is again verified to check if it satisfies all requirements of the customer by checking the execution of all the scripts for required inputs. The tests that are included are as follows:

1. Unity Testing: Here the individual units of source code are tested to determine whether to use or not.
2. Integration Testing: The output of unit test is a input for the integration testing. Here the individual codes are combined
3. System testing: The combined code in the integration test is tested for functionality's of the application.
4. Validation testing: The code is tested whether it satisfies the customer's requirements and specifications.
5. Verification testing: The documents are verified along with the code.

   Time Schedule:

   After the tool is developed the user verifies and the tests are again performed by allocating some time to repair the reported bugs.

   - Testing while developing the product: 2015-05-09 to 2015-05-17
   - Testing after product is developed: 2015-05-18 to 2015-05-20

12.2 Packaging plan:

A packaging plan is provided to the user in the ZIP archive which contains both documentation and the software.

12.3 Documentation plan:

12.3.1 Installation documentation:

We would like to provide a PDF installation documentation. It will cover all requirements on software installations and configuration background for components.

We divided the installation documentation into three stages for the time schedule:

1. Documentation antecedent to testing: 2015-05-14
2. Documentation during testing: 2015-05-17
3. Documentation succeeding: 2015-05-20

12.3.2 User documentation:

We provide the user documentation in PDF. It includes the product documentation i.e. the functionality of each tool, relation between different components in the tool. It gives a clear cut representation of the tool for each input and gives the necessary output.

The below is the time schedule.

1. Documentation antecedent to testing: 2015-05-15
2. Documentation during testing: 2015-05-18
3. Documentation succeeding: 2015-05-20

12.3.3 Developer documentation:

A developer documentation helps the developer for further implementations of the tool. This document gives the systematic approach for developer in the API development. The developer documentation includes source code, data format, DB tables and description of entry tables.

The below is the time schedule.

1. Documentation antecedent to testing: 2015-05-16
2. Documentation during testing: 2015-05-19
3. Documentation succeeding: 2015-05-20

13. REFERENCES:

[1] DPMI: Distributed Passive Measurement Infrastructure,
URL: https://github.com/DPMI/libcap_utils
[2] Patrik Arlos, Markus Fiedler, and Arne A.Nilsson, A Distributed Passive Measurement Infrastructure in Passive and Active Measurement Workshop (PAM05), US, 2005