Table of Contents

List of Figures

Chapter 1: Project Overview

1.1 Introduction

Today, electronic messaging is part of the communication backbone and the core of information dissemination within companies of all sizes. Everyone uses e-mail, from the mailroom to the laboratories to the CEO's office. And if you are in charge of an organization's mail server, you will be notified when it stops working.

Every activity going on inside the mail server is captured by the system and stored in the form of log files. These activities range from users logging in and out, sending messages, and failed login attempts to critical mail delivery failures and mail server services stopping or not starting properly. The log files that store all this information grow rapidly. If some problem appears in the system, then the first thing every system administrator does is view the corresponding log file. Since log files are plain texts and keep on increasing rapidly it is very difficult for the administrators to understand and pin point the problem. Furthermore, there is no facility for viewing log messages based on date and time of log message generation.

The solution to the problem of storage of log files is the setup of a centralized log server that stores all log files generated by the mail server. The mail server is configured to send log messages to a single remote log server, which acts as a common log server. The log server is then configured to send the log messages to a local database.

There is a big trend in the systems and services management. As WWW technologies are rapidly evolving, those technologies such as Java and Java Server Pages are applied to many computing areas. Systems and services management is one of such areas that WWW technologies can be adopted to increase the efficiency and usability of management systems.

Web-based management has a lot of benefits. It is hardware independent. Web technologies are general enough to be applied in any hardware platforms. It is cheap and ubiquitous solution in most computing areas. Users access information using Web browsers. No additional investment in needed. Using Web browser is trivial and very easy to learn.

Thus, the log messages stored in the database will be extracted by the Java Web Application and presented in the web browser to the user for ease of use and interpretation. In this report, we present the design and implementation of a centralized mail log management system.

1.2 Problem Statement

The project addresses following issues:

- Log files generated by servers are text based and difficult to interpret.
- These files are generated in huge amount containing thousands of lines of messages each day even on small scale servers.
- The size required for log files storage can grow rapidly which necessitates the use of centralized log server.
- The default setting stores the log files in the same computer system where the mail server resides. This becomes a problem because the log files take up incremental storage space as the time passes.
- The security and monitoring of mission critical servers become important issues if log files are kept in the same server that generates the log files.
- The log messages in the same file are difficult to determine based on their severity.
- There are several services running in the particular system so simply viewing /var/log/messages becomes tedious while searching for particular service log like mail log.
- When the system itself gets crashed all information is lost which points toward the requirement of centralized log server to view that system log.

1.3 Objectives

The main objectives behind the project are as follows:

- To create a full-fledged mail server and centralized log management system.
- To handle the difficulty faced by system administrators in viewing log files and taking appropriate course of actions.
- To create a centralized log server that stores all the log information generated by servers.
- To determine the severity of mail log messages at a single view which make it easier to operate.
- To store all mail log information into MYSQL database.
- To provide an easy to use interface for dealing with different logs generated by different servers.
- To create a web application based on Java Enterprise Edition that contains options for viewing specific log files based on message severity, date, and time with rich user friendly interface.

1.4 Project Features

This project handles the issues of mail log files management by setting up mail servers and a centralized log server. Through the use of relational database management system and programming, the log files will be carefully analyzed and presented to the user in a web based application that is easy to use and understand. The main purpose of the system is to make every administrators day to day job easy when it comes to viewing and understanding log files.

The major features of the project are:

i. A functioning mail server environment

ii. A centralized log server for storing log files generated by mail server

iii. A remote logging facility for transferring log files from mail server to centralized log server

iv. A relational database management system for storing log file messages

v. A web based user interface developed using Java Enterprise Edition technology for viewing log files efficiently that contains options for viewing specific log file messages based on message severity, log file generated date and time

1.5 Feasibility Study

The proposed system is feasible in following categories:

Economical

The system is economically feasible for the development and deployment since it uses existing computers that behave as servers. The operating system and software tools being used are easily available and free of cost i.e. CentOS (for operating system), rpm packages (for servers), Eclipse and NetBeans (for IDE), and Java Development Kit (for programming).

Technical

Technically the system is feasible as the systems requirement is to use existing hardware and software equipment. For the system requirement, tools that are used are open source and free of cost. For the hardware requirement, the system requirement is not so high and the hardware facility available is more that sufficient for the system to run smoothly.

Operational

The system is feasible on the operational level because the formulated system is easy in terms of installation and use. The developed system can be used by anyone with basic computer knowledge. By following the simple instructions anyone can use it for viewing day to day logs in an easy and efficient way. The centralized log management system operates smoothly facilitating in making the administrative tasks of the system administrator easy without any complication.

1.6 System Requirement

The requirements for the proposed system are:

- **Hardware Requirement**

  To create a test server environment we need two desktop computers that support Linux Operating System. One of them is used for mail server and the other is used as centralized log server that receives and stores logs generated by mail server.

- **Software Requirement**

  The different types of software required are:

  i. CentOS Linux Operating System

  ii. Mail server software – postfix, dovecot RPM packages

  iii. Web server software – httpd RPM package

  iv. Mail User Agent – squirrel mail RPM package

  v. Log server software – rsyslog, rsyslog-mysql RPM package

  vi. Database server software – mysql RPM package

  vii. Programming software – Java Development Kit

  viii. Programming platform – NetBeans and Eclipse IDEs

  ix. Virtual Machine Software – VMWare Workstation

  x. Java Enterprise Edition Web Application Framework – Java Server Pages and Spring Framework 2.0

Chapter 2: Literature Review

The aim of this project is to create a web based log management system for presenting information to system administrators for easy interpretation. To do this, it is important to first understand the various log files in which the operating system records important events.

The default system setting uses **syslog** facility which is a standard for computer data logging. It separates the software that generates messages from the system that stores them and the software that reports and analyzes them. Syslog can be used for computer system management and security auditing as well as generalized informational, analysis, and debugging messages. It is supported by a wide variety of devices (like printers and routers) and receivers across multiple platforms. Because of this, syslog can be used to integrate log data from many different types of systems into a central repository. Configuration allows directing messages to various local devices (console), files (/var/log) or remote syslog daemons.

Syslog is installed by default as the primary means for computer data logging. The format of its configuration file and the operation of each line in the file were studied by consulting the man pages.

In this project syslog was not used for computer data logging. Instead, **rsyslog** was used to capture logging records and send the log files to remote log server. Both facilities use almost similar configuration settings. The advantage with rsyslog is that it can be configured to send log data to database for efficient storage and future retrievals. Section 3.4 discusses the configuration settings and working of rsyslog service.

There have been a number of efforts which have approached the issue of managing log files. Logwatch facility flags messages that might reflect a problem with the system and forwards them each day in an e-mail message to the system's root user. When CentOS is installed, the logwatch package is installed and configured to watch log files and report suspicious activities to the system administrator.

A report on **Design and Implementation of a Web-based Internet/Intranet Mail Server Management System** by **Jae-Young Kim and James Won-Ki Hong**[5] discusses the design and implementation of Java-based GUI system that enables human users to manage mail server systems from anywhere with friendly, easy-to-use Web browser interfaces. The system architecture is also general enough so that it can be easily extended to manage any other Internet/Intranet application services.

As part of this project, the efforts in a number of areas for managing and viewing log files using web based interface were investigated. The two major log analyzers that we took reference from are as follows:

- **Adiscon LogAnalyzer**

It is a web interface to syslog and other network event data. It provides easy browsing, analysis of real time network events and reporting services. Reports help to keep an eye on network activity. It consolidates syslog and other event data providing an easy to read sheet.

- **Graylog 2 Open source Log Management**

It is an open source log management tool that enables to unleash the power that lays inside the logs. It can be used to run analytics, alerting, monitoring and powerful searches over the whole log base. Quick filter searches let us find and see what errors are produced. The web interface uses Ruby on Rails and the server is written in Java.

This project builds upon the in-depth understanding of log files, various logging daemons and services and above mentioned web based software tools for log management.

Chapter 3: System Development

3.1 Project Management

Good project management is the key to success of any important project. Well managed projects rarely fail. Unmanaged projects, even with competent people and technologies, have never met the schedule and budget constraints. Project management does not assume that all will go well. Problems of various natures nearly always arise during the life time of the project. This fact should be taken into consideration in all management activities including proposal writing, project planning & scheduling and risk management.

During this project there are different task which are performed within certain period of time.

The project was completed with following phases:

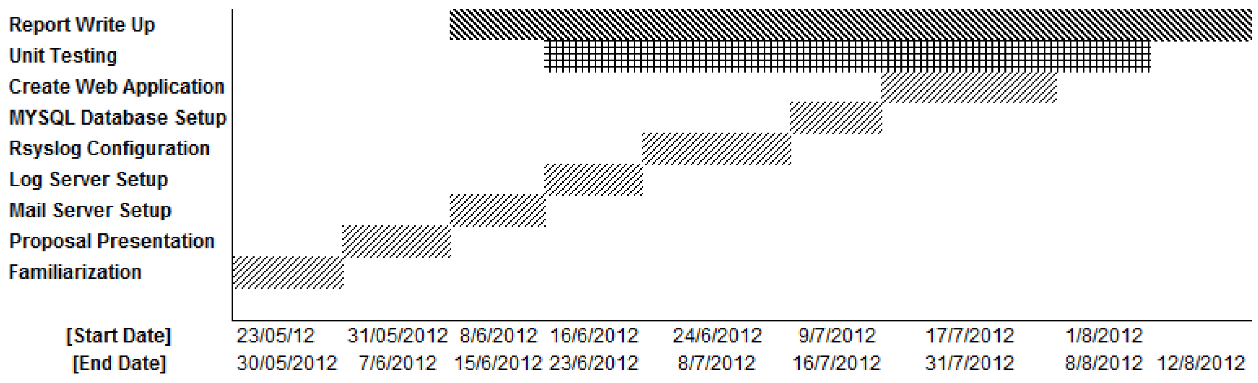


Figure 1: Gantt Chart of Project Schedule

3.2 System Analysis

The current system uses syslog facility to log various types of messages generated by different mail services, such as postfix and dovecot. The main three types of log files are based on message severity. They are informational, warning, and error log files. When any of these events occur, the corresponding service generates log messages which are captured by syslog service and stored in **/var/log** directory. The syslog facility is installed by default and stores all the messages in the same machine that generates the log messages. The diagram below illustrates the current system with default setting.

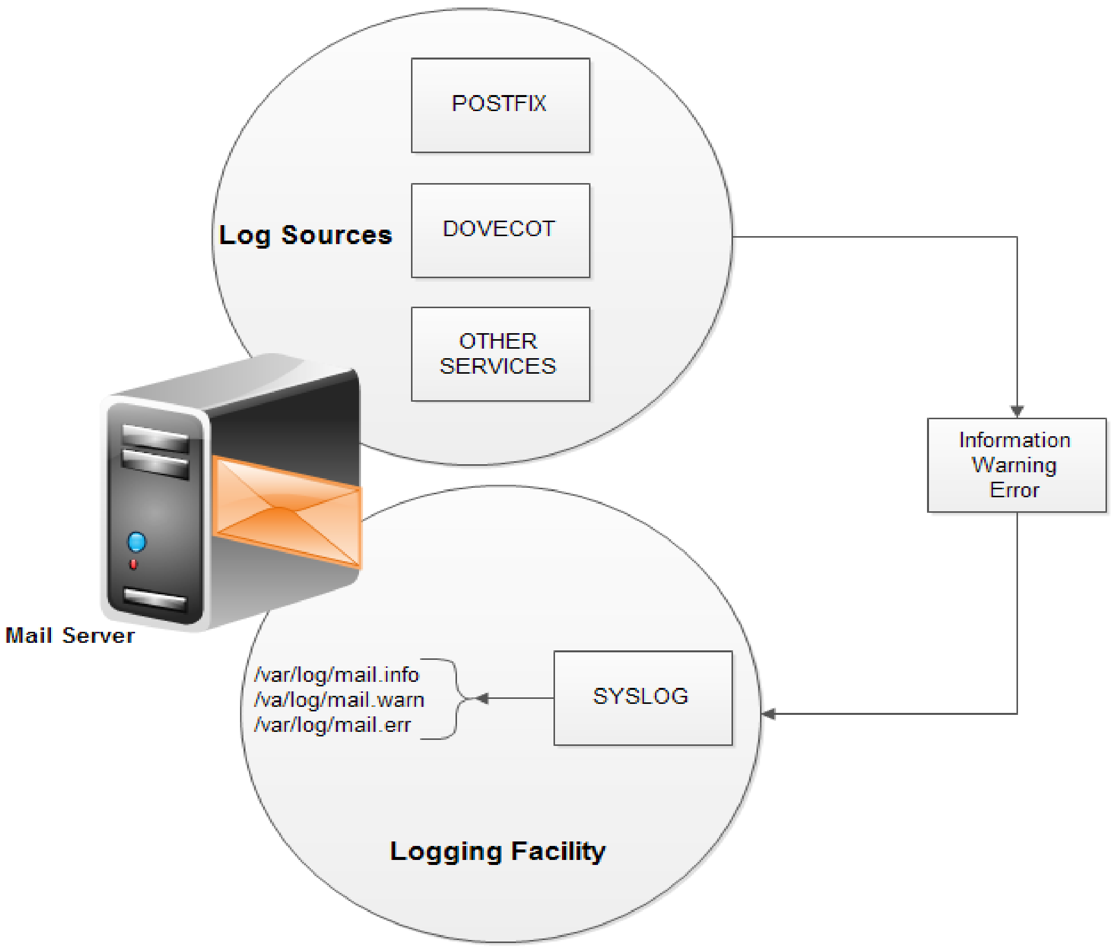
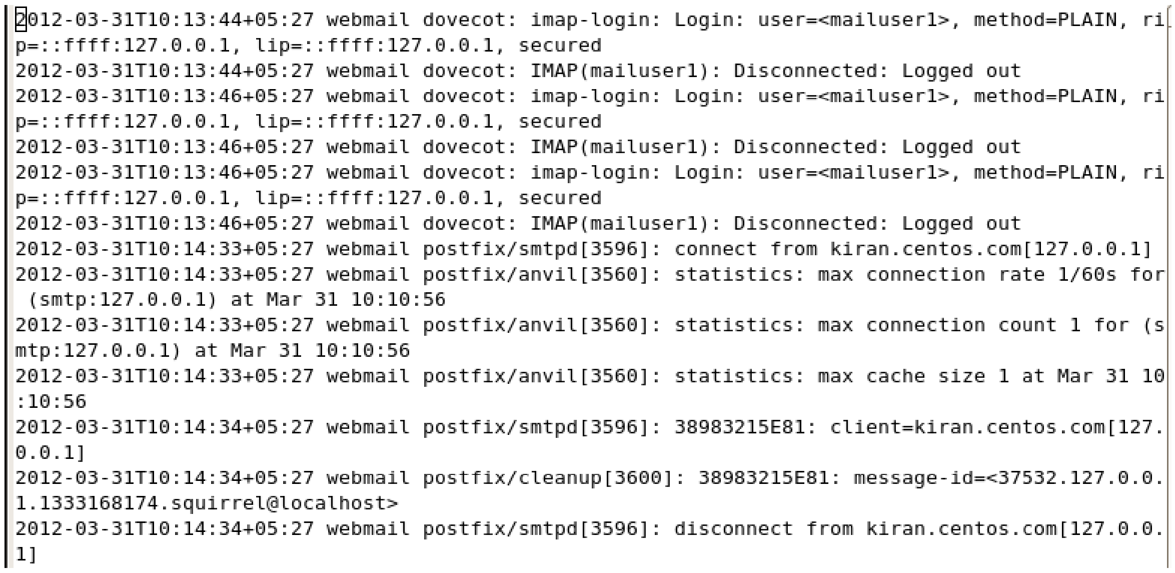


Figure 2: Existing System

Because of the many programs and services that record information to the messages log file, it is important that we understand the format of this file. Each line in the file is a single message recorded by some program or service. Here is a snippet of an actual message log file:



```
2012-03-31T10:13:44+05:27 webmail dovecot: imap-login: Login: user=<mailuser1>, method=PLAIN, ri
p=::ffff:127.0.0.1, lip=::ffff:127.0.0.1, secured
2012-03-31T10:13:44+05:27 webmail dovecot: IMAP(mailuser1): Disconnected: Logged out
2012-03-31T10:13:46+05:27 webmail dovecot: imap-login: Login: user=<mailuser1>, method=PLAIN, ri
p=::ffff:127.0.0.1, lip=::ffff:127.0.0.1, secured
2012-03-31T10:13:46+05:27 webmail dovecot: IMAP(mailuser1): Disconnected: Logged out
2012-03-31T10:13:46+05:27 webmail dovecot: imap-login: Login: user=<mailuser1>, method=PLAIN, ri
p=::ffff:127.0.0.1, lip=::ffff:127.0.0.1, secured
2012-03-31T10:13:46+05:27 webmail dovecot: IMAP(mailuser1): Disconnected: Logged out
2012-03-31T10:14:33+05:27 webmail postfix/smtpd[3596]: connect from kiran.centos.com[127.0.0.1]
2012-03-31T10:14:33+05:27 webmail postfix/anvil[3560]: statistics: max connection rate 1/60s for
 (smtp:127.0.0.1) at Mar 31 10:10:56
2012-03-31T10:14:33+05:27 webmail postfix/anvil[3560]: statistics: max connection count 1 for (s
mtp:127.0.0.1) at Mar 31 10:10:56
2012-03-31T10:14:33+05:27 webmail postfix/anvil[3560]: statistics: max cache size 1 at Mar 31 10
:10:56
2012-03-31T10:14:34+05:27 webmail postfix/smtpd[3596]: 38983215E81: client=kiran.centos.com[127.
0.0.1]
2012-03-31T10:14:34+05:27 webmail postfix/cleanup[3600]: 38983215E81: message-id=<37532.127.0.0.
1.1333168174.squirrel@localhost>
2012-03-31T10:14:34+05:27 webmail postfix/smtpd[3596]: disconnect from kiran.centos.com[127.0.0.
1]
```

Figure 3: Sample log file

Above generated log file is completely textual and does not assist user in making the task of analyzing the log file easy. As time passes the log file grows in size and locating required information becomes very difficult. Users cannot filter the content based on date and time to view specific files.

Since the same machine stores log files and runs mail service, the storage runs out gradually with increasing log messages. Furthermore, if the system is broken into, the cracker may delete or modify the log files because the log files are stored on the same computer. The proposed approach handles above issues and problems.

3.3 System Design

The proposed approach consists of two parts. The first part separates the mail services and location of log files storage to solve above mentioned issues. In this approach, remote logging facility, **rsyslog**, is used to transfer log files generated by the mail server with ip address 192.168.101.28 to the log server with ip address 192.168.101.191 as shown in the diagram below.

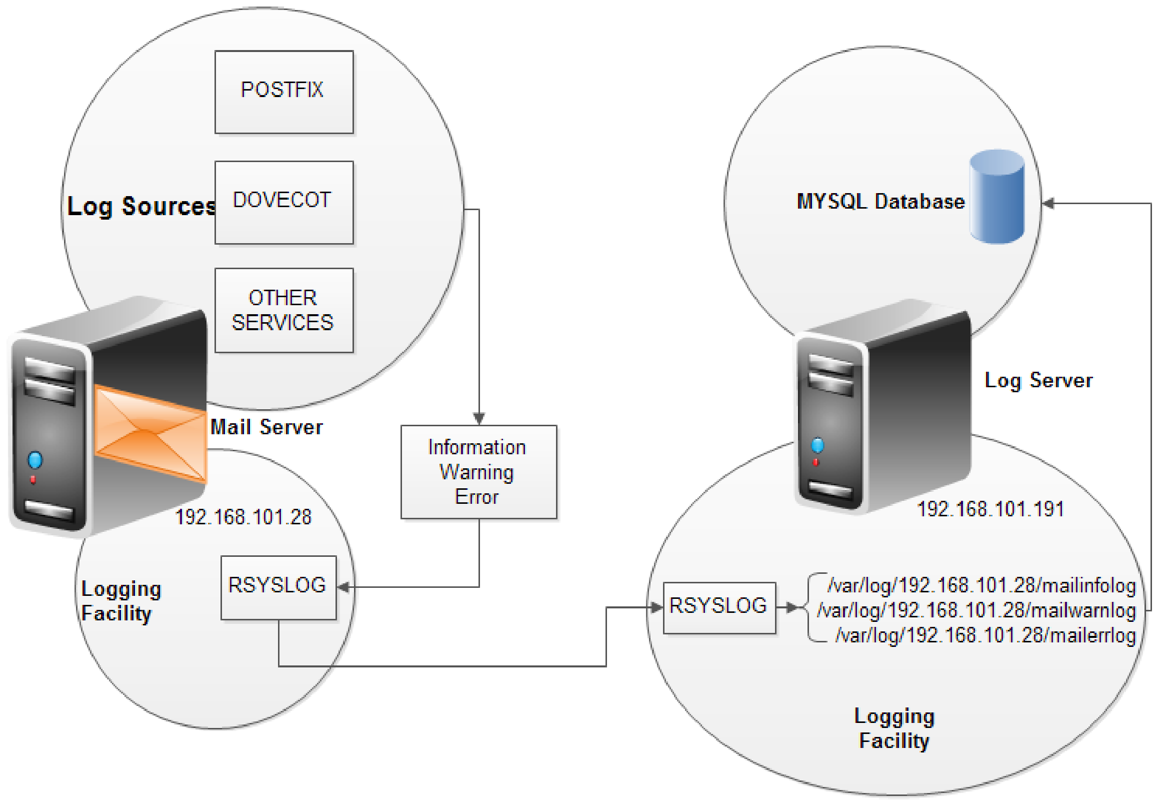


Figure 4: Proposed System

The log files obtained by the log server are stored in the directory **/var/log/192.168.101.28/**. After that a software tool **rsyslog-msyql** sends all the log files to the mysql database.

The second part of the project is the creation of a web based user interface developed using Java Enterprise Edition technology for viewing log files efficiently that contains options for viewing specific log file messages based on message severity, and log file generated date and time. The Java web application uses the content of mysql database to display log file messages.

3.4 System Implementation

The proposed design was implemented as shown in the following diagram.

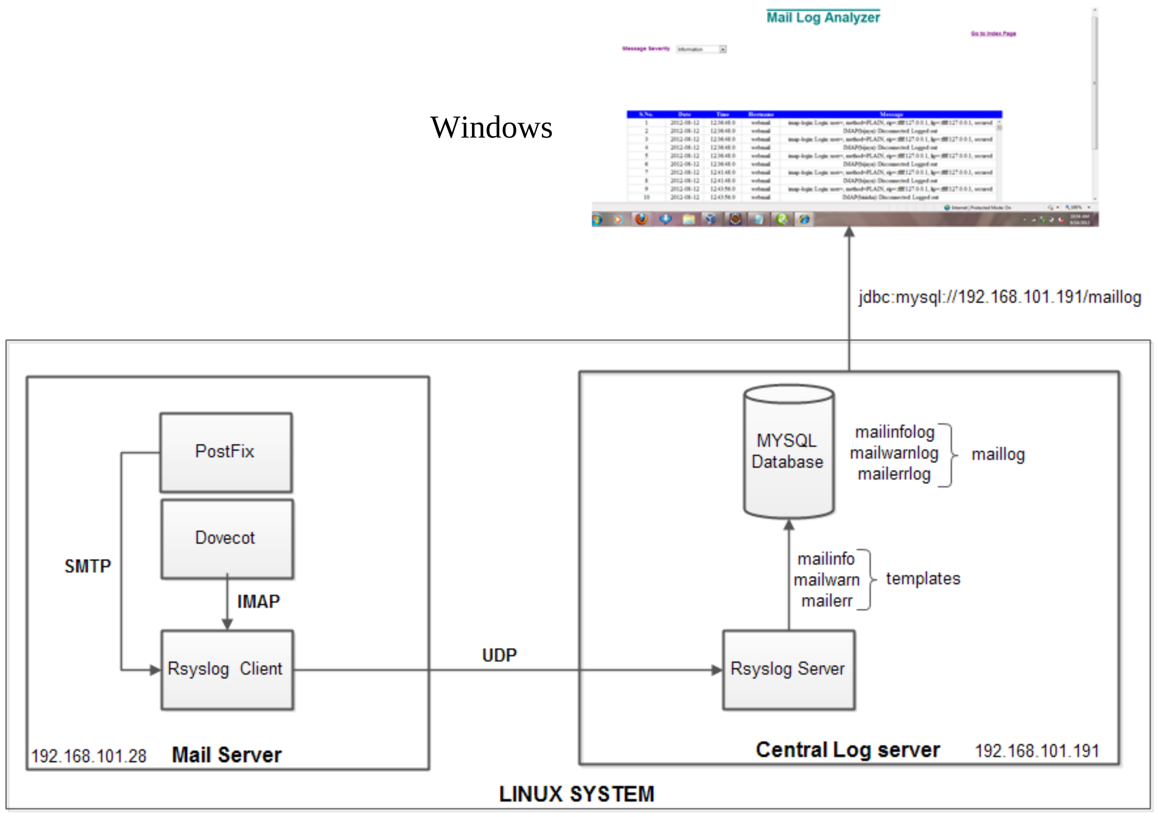


Figure 5: Implementation Architecture

During the system implementation, we configured rsyslog client in the mail server to capture mail log events from various services using SMTP and IMAP protocols. The mail server sends log files to the log server using UDP protocol. The rsyslog server running in the log server transfers mail log files into mysql database. The database named "**maillog**" stores the log files into three tables **mailinfolog, mailwarnlog,** and **mailerrlog**. The Java EE web application running in windows accesses the mysql database and displays the web interface containing the log files.

The implementation is performed in following steps. The configuration files and codes are located in the Appendix section at the end of this report for detailed viewing.

3.4.1 Configuration of mail server and web server

Linux offers a number of alternative methods for handling incoming and outgoing e-mail. CentOS includes Sendmail and Postfix for this purpose. In this project Postfix is used.

i. Postfix

Postfix is a free and open-source Mail Transfer Agent (MTA) that routes and delivers electronic mail. It is considered as a fast, easier-to-administer, and secure alternative to the widely-used sendmail MTA.

Postfix includes numerous individual programs to process mail, managed by a supervisor daemon called **master.** The file **/etc/postfix/master.cf** configures how the subsidiary process should be run. The file **/etc/postfix/main.cf** provides configuration details for setting up the mail server. The **master** daemon does not handle mail directly, but manages the other daemons that do.

Some of these daemons are:

- Smtpd – It listens on port 25 for incoming messages and submits them to the incoming queue.

- Pickup – It moves messages sent by the local postfix server from the maildrop queue to the incoming queue

- Nqmgr – It passes messages from the incoming queue to various processes for transmission, relaying, or local delivery.

The software package used for Postfix Configuration is postfix-2.3.3.-2.1.el5_2.i386.rpm

The configuration file **/etc/postfix/main.cf** defines various parameters for mail server operation. For the postfix configuration purpose, we assign the 'myhostname' parameter as 'webmail.freesource.com'. The 'myhostname' parameter specifies the internet hostname of this mail system.

```
myhostname = webmail.freesource.com
```

The ‘mydomain’ parameter is assigned as ‘freesource.com’. This parameter represents the local internet domain name.

```
mydomain = freesource.com
```

For sending mail, the postfix is configured with the parameter ‘myorigin’ as the internet hostname of the mail system i.e. ‘webmail.freesource.com’

```
myorigin = webmail.freesource.com
```

For receiving mail, all the possible domains are assigned with ‘mydestination’ parameter.

```
inet_interface = ALL

mydestination = webmail.freesource.com,
localhost.freesource.com, localhost, freesource.com
```

If the mail needs to be rejected from local unknown users, set the ‘unknown_local_recipient_reject_code’ parameter as value 550. This is the default value.

```
unknown_local_recipient_reject_code = 550
```

The mail spool directory parameter specifies the directory where UNIX-style mailboxes are kept. The default setting depends on the system type.

```
mail_spool_directory = /var/spool/mail
```

The home_mailbox parameter specifies the optional pathname of a mailbox file relative to a user’s home directory.

```
home_mailbox = Maildir/
```

ii. Dovecot

Dovecot is an open source IMAP and POP3 server for Linux/UNIX-like systems, written primarily with security in mind. It intends to be a lightweight, fast and easy to set up open source mail server. It is simple to set up and requires no special administration and uses very little memory.

Dovecot is self-healing. It tries to fix most of the problems it notices by itself. The problems are logged so the administrator can later try to figure out what caused them.

It includes an IMAP and IMAPS server, as well as POP3 and POP3S server. IMAPS and POP3S are secure servers using SSL to encrypt authentication and data. It listens on all IPv4 and IPv6 interfaces by default.

The software package used for Dovecot configuration is dovecot-1.0.7-7.el5.i386.rpm.

For dovecot configuration, we specify the protocols used for mail delivery in **/etc/dovecot.conf file**

```
protocols = imap
```

Above line configures the mail server to listen on imap port for mail delivery between users.

iii. Squirrelmail

SquirrelMail is an open-source project that provides web-based email application and an IMAP proxy server. It is a standards-based webmail package written in PHP Licensed under the GNU General Public License. SquirrelMail products are free software and currently available in over 50 languages. This webmail is included in the repositories of many major GNU/Linux distributions.

The software package used for Squirrelmail is Squirrelmail-1.4.8-5.el5.centos.10.noarch.rpm

iv. Apache HTTP Server Configuration

The Apache HTTP Server, commonly referred to as Apache is a web server that typically runs on a Unix-like operating system. It is developed and maintained by an open community of developers under the banner of Apache Software Foundation.

Apache supports password authentication and digital certificate authentication. It features configurable error messages, DBMS-based authentication databases, and supported by several graphical user interfaces (GUIs).

The software package used for HTTP server is httpd-2.2.3-43.el5.centos.i386.rpm.

It is required for squirrel mail web interface. The configuration file **/etc/httpd/conf/httpd.conf** contains the settings to allow squirrel mail to run on the web browser. The settings are shown below.

```
Alias /squirrelmail /usr/local/squirrelmail/www
<Directory /usr/local/squirrelmail/www>
  Options None
  AllowOverride None
  DirectoryIndex index.php
  Order Allow,Deny
  Allow from all
</Directory>
<Directory /usr/local/squirrelmail/www/*>
  Deny from all
</Directory>
<Directory /usr/local/squirrelmail/www/images>
  Allow from all
</Directory>
<Directory /usr/local/squirrelmail/www/plugins>
  Allow from all
</Directory>
<Directory /usr/local/squirrelmail/www/src>
  Allow from all
</Directory>
<Directory /usr/local/squirrelmail/www/templates>
  Allow from all
```

```
</Directory>
<Directory /usr/local/squirrelmail/www/themes>
  Allow from all
</Directory>
<Directory /usr/local/squirrelmail/www/contrib>
  Order Deny,Allow
  Deny from All
  Allow from 127
  Allow from 10
  Allow from 192
</Directory>
<Directory /usr/local/squirrelmail/www/doc>
  Order Deny,Allow
  Deny from All
  Allow from 127
  Allow from 10
  Allow from 192
```

3.4.2 Configuration of Rsyslog service in the mail server

Every Linux distribution has some kind of logging mechanism that records all the system activities. The logs files are stored under the /var/log directory according to the services we have run in the computer. These logs are very critical for system administrators for troubleshooting purpose.

The following are the three common methods to log a message:

- Logging on the same server: Messages get written into the local hard drive/local database
- Logging on a remote server: Many systems forward their logs over the network to a central log server. On the central log server, the messages from various systems are written to the local hard drive/database.
- Relay logging: Branch 'A' and Branch 'B' logs the messages on 2 different servers. These server in-turn logs the message to the 'Head Office'.

In a larger environment it's critical to centralize log-files. Third- party tools write to a local log-file but don't have possibility to use syslog. Having a centralized logging is a prerequisite if we want to have our logs intact. As a solution we can use **remote syslog.** Rsyslog is the default logging program on several Linux distributions including Debian and Red Hat based systems. Apart from implementing the syslog protocol, rsyslog adds additional features such as content-based filtering.

Rsyslog is open source software used on LINUX and Unix-like computer systems for forwarding log messages in an IP network. It implements the basic syslog protocol, extends it with content-based filtering, rich filtering capabilities, flexible configuration options and adds important features such as using TCP for transport. When the events recorded in plain files is virtually impossible for queries. Because of this, we can configure rsyslog to write events in MySQL DB and a web interface for queries with filters to facilitate viewing of the logs without have to access the console for such task.

Rsyslog has following features:

- Native support for writing to MySQL databases.
- Native support for sending mail messages.
- Support for (plain) tcp based syslog - much better reliability.
- Support for sending and receiving compressed syslog messages.
- Ability to monitor text files and convert their contents into syslog messages (one per line).
- Ability to configure backup syslog/database servers - if the primary fails, control is switched to a prioritized list of backups.
- Ability to generate file names and directories (log targets) dynamically, based on many different properties
- Control of log output format, including ability to present channel and priority as visible log data.
- Good timestamp format control; at a minimum, ISO 8601/RFC 3339 second-resolution UTC zone.
- Ability to reformat message contents and work with substrings.
- Support for log files larger than 2 GB.
- Support for file size limitation and automatic rollover command execution.
- Support for running multiple rsyslogd instances on a single machine.
- Support for TLS-protected syslog.
- Ability to filter on any part of the message, not just facility and severity.
- Ability to use regular expressions in filters.
- Support for discarding messages based on filters.
- Ability to execute shell scripts on received messages.
- Control of whether the local hostname or the hostname of the origin of the data is shown as the hostname in the output.

- Ability to preserve the original hostname in NAT environments and relay chains.
- Ability to limit the allowed network senders.
- Very experimental and volatile support for syslog-protocol compliant messages.
- World's first implementation of syslog-transport-tls.
- The sysklogd's klogd functionality is implemented as the imklog input plug-in. So rsyslog is a full replacement for the sysklogd package.
- Support for IPv6.
- Ability to control repeated line reduction ("last message repeated n times") on a per selector-line basis.
- Supports sub-configuration files, which can be automatically read from directories. Includes are specified in the main configuration file.
- Supports multiple actions per selector/filter condition.
- MySQL and Postgres SQL functionality as a dynamically loadable plug-in.
- Modular design for inputs and outputs - easily extensible via custom plugins.
- An easy-to-write to plugin interface.
- Ability to send SNMP trap messages.
- Ability to filter out messages based on sequence of arrival.

We need to install rsyslog and use it to replace the CentOS's default sysklogd server. The required packages are:

- rsyslog
- rsyslog-mysql
- mysql-server
- php-mysql

- php-gd
- httpd
- mod_ssl

The procedure is:

```
# yum install rsyslog
# service syslog stop
# chkconfig syslog off
# chkconfig rsyslog on
# service rsyslog start
```

Before understanding how to setup the central logging sever, it is good to understand the configuration structure of rsyslog.

Rsyslog is configured via the rsyslog.conf file, typically found in /etc/ directory. By default, rsyslogd reads the file **/etc/rsyslog.conf**. Rsyslog configuration file is structured in the following manner:

i. Modules

ii. Configuration Directives

iii. Rule line

iv. Templates

i. Modules

Rsyslog has a modular design. This enables functionality to be dynamically loaded from modules, which may also be written by any third party. Rsyslog itself offers all non-core functionality as modules. Each module provides configuration directives, also a modules configuration directive (and functionality) is only available if it has been loaded (using $ModLoad). Consequently, there are a growing number of modules.

The modules are categorized as:

a. Input Modules

Input modules are used to gather messages from various sources. They interface to message generators.

- imfile - input module for text files
- imrelp - RELP input module
- imudp - udp syslog message input
- imtcp - input plugin for tcp syslog
- imptcp - input plugin for plain tcp syslog (no TLS but faster)
- imgssapi - input plugin for plain tcp and GSS-enabled syslog
- immark - support for mark messages
- imklog - kernel logging
- imuxsock - unix sockets, including the system log socket
- imsolaris - input for the Sun Solaris system log source
- im3195 - accepts syslog messages via RFC 3195
- impstats - provides periodic statistics of rsyslog internal counters

b. Output Modules

Output modules process messages. With them, message formats can be transformed and messages be transmitted to various different targets.

- omfile - file output module
- omfwd - syslog forwarding output module
- ompipe - named pipe output module
- omusrmsg - user message output module
- omsnmp - SNMP trap output module
- omtdout - stdout output module (mainly a test tool)

- omrelp - RELP output module
- omruleset - forward message to another ruleset
- omgssapi - output module for GSS-enabled syslog
- ommysql - output module for MySQL
- ompgsql - output module for PostgreSQL
- omlibdbi - generic database output module (Firebird/Interbase, MS SQL, Sybase, SQLLite, Ingres, Oracle, mSQL)
- ommail - permits rsyslog to alert folks by mail if something important happens
- omprog - permits sending messages to a program for custom processing
- omoracle - output module for Oracle (native OCI interface)
- omudpspoof - output module sending UDP syslog messages with a spoofed address
- omuxsock - output module Unix domain sockets
- omhdfs - output module for Hadoop's HDFS file system

c. Parser Modules

Parser modules are used to parse message content, once the message has been received. They can be used to process custom message formats or invalidly formatted messages.

The current modules are currently provided as part of rsyslog:

- pmrfc5424[builtin] - rsyslog.rfc5424 - parses RFC5424-formatted messages (the new syslog standard)
- pmrfc3164[builtin] - rsyslog.rfc3164 - the traditional/legacy syslog parser

- pmrfc3164sd - rsyslog.rfc3164sd - a contributed module supporting RFC5424 structured data inside RFC3164 messages (not supported by the rsyslog team)
- pmlastmsg - rsyslog.lastmsg - a parser module that handles the typically malformed "last messages repeated n times" messages emitted by some syslogds.

A simplified workflow is as follows:

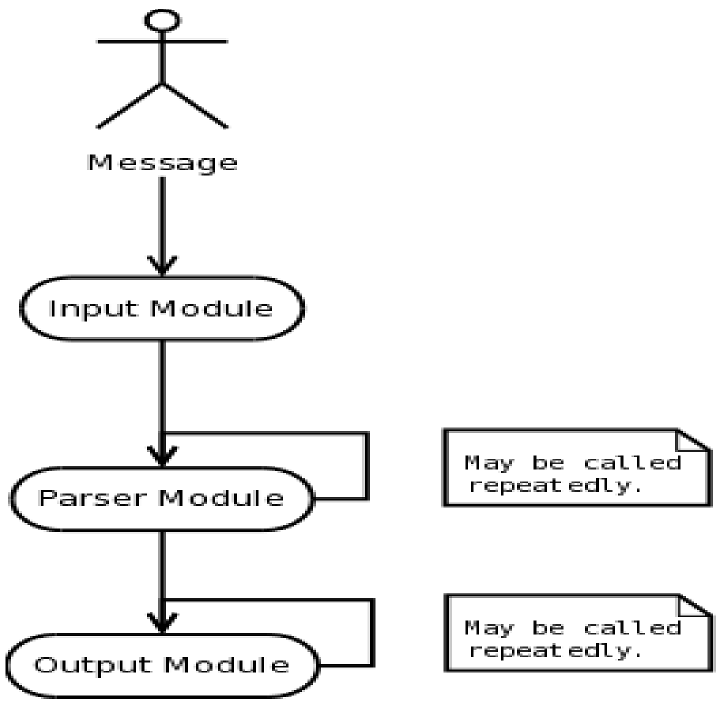
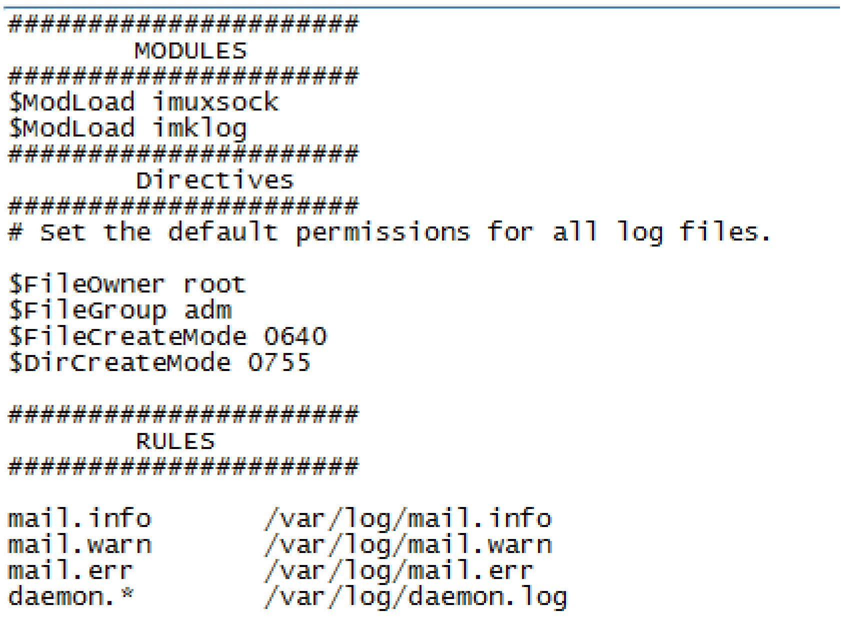


Figure 6: Workflow between modules

ii. Configuration Directives

All configuration directives need to be specified on a line by their own and must start with a dollar-sign. Those starting with the word "Action" modify the next action and should be specified in front of it. A sample configuration is shown below.



```
######################
        MODULES
######################
$ModLoad imuxsock
$ModLoad imklog
######################
        Directives
######################
# Set the default permissions for all log files.

$FileOwner root
$FileGroup adm
$FileCreateMode 0640
$DirCreateMode 0755

######################
        RULES
######################

mail.info       /var/log/mail.info
mail.warn       /var/log/mail.warn
mail.err        /var/log/mail.err
daemon.*        /var/log/daemon.log
```

iii. Rule Line

Every rule line consists of two fields, a ‘selector field’ and an ‘action field’. The selector field is divided into two, ‘facilities & priorities’. Action specifies what action must be taken for the matched rule. The following tables show the list of facilities and priorities with their respective number code.

FACILITIES		
Numerical Code	Keyword	Facility
0	kern	Kernel
1	User	Regular user processes
2	Mail	Mail System
3	daemon	System daemons
4	Auth	Security(authentication and authorization)related commands
5	syslog	Syslog internal messages
6	Lpr	Line printers system
7	news	NNTP subsystem
8	uucp	UUCP subsystem
10	authpriv	Private authorization messages
16-23	Local0-7	Site specific use

Table 1: Facilities for rsyslog

PRIORITIES		
Numerical Code	Keyword	Facility
0	emerg	Emergency: System is unusable
1	alert	Alert: action must be taken immediately
2	crit	Critical: critical conditions
3	err	Error: error conditions
4	warn	Warning: warning conditions
5	notice	Notice: normal but significant conditions
6	info	Informational: informational messages
7	debug	Debug : debug level messages

Table 2: Priorities for facilities

iv. Templates

Templates are a key feature of rsyslog. They allow to specify any format a user might want. They are also used for dynamic file name generation. Every output in rsyslog uses templates - this holds true for files, user messages and so on. The database writer expects its template to be a proper SQL statement - so this is highly customizable too. Templates compatible with the stock syslogd formats are hardcoded into rsyslogd. So if no template is specified, we use one of these hardcoded templates.

There are actually two different types of template:

- string based
- string-generator module based

String-generator module based templates have been introduced in 5.5.6. They permit a string generator, actually a C "program" that generates a format. Rsyslog already contains highly optimized string generators and these are called without any need to configure anything. String-generator module based templates are not the route to take. Usually, we use string based templates instead.

Template options are case-insensitive. Currently defined are:

Sql - format the string suitable for a SQL statement in MySQL format. This will replace single quotes ("'") and the backslash character by their backslash-escaped counterpart ("\'" and "\\") inside each field. Please note that in MySQL configuration, the NO_BACKSLASH_ESCAPES mode must be turned off for this format to work.

Stdsql - format the string suitable for a SQL statement that is to be sent to a standards-compliant sql server. This will replace single quotes ("'") by two single quotes ("''") inside each field. You must use stdsql together with MySQL if in MySQL configuration the NO_BACKSLASH_ESCAPES is turned on.

Either the sql or stdsql option must be specified when a template is used for writing to a database, otherwise injection might occur.

A String-based Template Sample is shown below.

```
$template StdSQLFormat,"insert into SystemEvents (Message, Facility, FromHost, Priority, DeviceReportedTime, ReceivedAt, InfoUnitID, SysLogTag) values ('%msg%', %syslogfacility%, '%HOSTNAME%', %syslogpriority%, '%timereported:::date-mysql%', '%timegenerated:::date-mysql%', %iut%, '%syslogtag%')",SQL
```

The configuration basically works in 4 parts. First, we load all the modules (imtcp, imudp, imuxsock, imklog). Then we specify the templates for creating files. Then we create the rulesets which we can use for the different receivers. And last we set the listeners.

The rulesets are somewhat interesting to look at. The ruleset "local" will be set as the default ruleset. It means that it will be used by any listener if it is not specified otherwise. Further, this ruleset uses the default log paths various facilities and severities.

The ruleset "remote" on the other hand takes care of the local logging and forwarding of all log messages that are received either via UDP or TCP. First, all the messages will be stored in a local file. The filename will be generated with the help of the template at the beginning of our configuration (in our example a rather complex folder structure will be used). After logging into the file, all the messages will be forwarded to another syslog server via TCP.

In the last part of the configuration we set the syslog listeners. We first bind the listener to the ruleset "remote" then we give it the directive to run the listener with the port to use. We use port number 514 for UDP connection which is fixed and similarly assign port number for TCP connection like 10514.

The mail server is configured to send log files to log server as follows.

i. At first install rsyslog package.

   ```
   #yum install rsyslog
   ```

ii. Add the following line (server ip, port) in the existing rsyslog configuration file.

   ```
   *.* @serveripaddress: 514     (Enables UDP forwarding)
   ```

```
*.* @@serveripaddress: 10514
```

(Enables TCP forwarding, You can use any one protocol)

```
mail.err @192.168.101.191:514
mail.info @192.168.101.191:514
mail.warn @192.168.101.191:514
```

iii. Now restart the rsyslog service

```
#service rsyslog restart
```

3.4.3 Configuration of rsyslog service in the log server

The log server is configured to receive log files from the mail server and store files on the /var/log/192.168.101.28/ directory as well as in the mysql database.

It involves following procedure:

i. At first install following packages

```
#yum install rsyslog rsyslog-mysql
```

ii. Now edit the rsyslog configuration file making the required changes.

```
#vi /etc/rsyslog.conf
# Add your Client server IP or IP Range
$AllowedSender UDP,192.168.101.28
# provides UDP syslog reception. For TCP, load imtcp.
$ModLoad imudp
# For UDP, InputServerRun 514
$UDPServerRun 514
```

This one is the template to generate the log filename dynamically, depending on the client's IP address.

```
$template FILE1,"/var/log/%fromhost-ip%/mailinfo.log"
```

```
$template FILE2,"/var/log/%fromhost-ip%/mailwarn.log"

$template FILE3,"/var/log/%fromhost-ip%/mailerr.log"
```

Here, we have generated 3 different templates for mail.info , mail.warn and mail.err that we obtained from client’s computer. After generating templates, we need to insert each template into mysql database to obtain the log information in tabular format.

$template mailinfo,"insert into mailinfolog (Facility, NTSeverity, FromHost, Priority, ReceivedAt, Message) values (%syslogfacility%, %syslogseverity%, '%HOSTNAME%',%syslogpriority%, '%timegenerated:::date-mysql%', '%msg%')",SQL

$template mailwarn,"insert into mailwarnlog (Facility, NTSeverity, FromHost, Priority, ReceivedAt, Message) values (%syslogfacility%, %syslogseverity%, '%HOSTNAME%',%syslogpriority%, '%timegenerated:::date-mysql%', '%msg%')",SQL

$template mailerr,"insert into mailerrlog (Facility, NTSeverity, FromHost, Priority, ReceivedAt, Message) values (%syslogfacility%, %syslogseverity%, '%HOSTNAME%',%syslogpriority%,'%timegenerated:::date-mysql%',   '%msg%')",SQL

Here, the template is an actual SQL statement. Note the "SQL" option: it tells the template processor that the template is used for SQL processing, thus quote characters are quoted to prevent security issues. You can not assign a template without "SQL" to a MySQL output action.

# Log all messages to the dynamically formed file. Now each clients log (192.168.1.2, 192.168.1.3,etc...), will be under a separate directory which is formed by the template FILENAME.

```
mail.info ?FILE1

mail.warn ?FILE2

mail.err ?FILE3
```

3.4.4 Configuration of mysql database in the log server

MySQL is a Relational Database Management System (RDBMS) that runs as a server providing multi-user access to a number of databases.

To install mysql run the command:

```
#yum install mysql-server mysql php-mysql
```

To create a user and database in MYSQL run the command:

```
# mysql –u root -p
<enter_mysql_root_password>
CREATE DATABASE <DB_NAME>;
GRANT ALL PRIVILEGES ON <DB_NAME>.* TO 'my_user'@'localhost' IDENTIFIED BY 'my_password' WITH GRANT OPTION;
```

For a new mysql user with full privileges on all mysql databases you would use:

```
GRANT ALL PRIVILEGES ON *.* TO 'my_user'@'localhost' IDENTIFIED BY 'my_password' WITH GRANT OPTION;
```

Gathering information message is important on Data Center, in some situations we want to store all entries of log files on another server. If a server crashes or gets hacked, it will be able to trace through log files from that machine. This can be accomplished by using centralized log server that receive messages from another hosts. A syslog facility can receive messages from Unix/Linux hosts but also network devices and windows hosts.

In many cases, syslog data is simply written to text files. This approach has some advantages; most notably it is very fast and efficient. However, data stored in text files is not readily accessible for real-time viewing and analysis. To do that, the messages need to be in a database. There are various ways to store syslog messages in a database. For example, some have the syslogd write text files which are later feed via a separate script into the database. Others have written scripts taking the data (via a pipe) from a non-database-aware syslogd and store them as they appear. Some others use database-aware syslogds and make them write the data directly to the database.

We have used rsyslogd on the server. It has intrinsic support for talking to MySQL databases. For obvious reasons, we also need an instance of MySQL running. MySQL support in rsyslog is integrated via a loadable plug-in module. To use the database functionality, MySQL must be enabled in the configuration file before the first database table action is used. This is done by placing the

```
$ModLoad ommysql
```

It is the output module for MYSQL. At the top, we have these module loaded which we need. It handles the MySQL capabilities of rsyslog.

Next, we need to tell rsyslogd to write data to the database. As we use the default schema, we can define template for this or we can use the hardcoded one (rsyslogd handles the proper template linking). So all we need to do is add a simple selector line to /etc/rsyslog.conf:

Syntax:

The following sample writes all syslog messages to the database "syslog_db" on mysqlsever.example.com. The server is being accessed under the account of "user" with password "pwd".

```
*.* action (type="ommysql" server="mysqlserver.example.com"  db="syslog_db"
uid="user" pwd="pwd")
```

In many cases, MySQL will run on the local machine. In this case, we can simply use "127.0.0.1" for database-server. This can be especially advisable, if we do not need to expose MySQL to any process outside of the local machine. In this case, we can simply bind it to 127.0.0.1, which provides a quite secure setup. Of course, also supports remote MySQL instances. In that case, use the remote server name (e.g. mysql.example.com) or IP-address. The database-name by default is "syslog". If we modified the default, we can use our name here. Database-userid and -password are the credentials used to connect to the database. As they are stored in clear text in rsyslog.conf, that user should have only the least possible privileges. It is sufficient to grant it INSERT privileges to the SystemEvents table only.

In /etc/rsyslog.conf file of server we have included following lines:

```
mail.info  :ommysql:127.0.0.1,maillog,bijaya,password;mailinfo

mail.warn  :ommysql:127.0.0.1,maillog,bijaya,password;mailwarn

mail.err  :ommysql:127.0.0.1,maillog,bijaya,password;mailer
```

Here, we have written mail.info, mail.warn, mail.err messages to the database "maillog" on localhost (127.0.0.1). The server is being accessed under the account name "bijaya" with password "password".

Rsyslogd writes syslog messages directly to the database. This implies that the database must be available at the time of message arrival. If the database is offline, no space is left or something else goes wrong - rsyslogd cannot write the database record. If rsyslogd is unable to store a message, it performs one retry. This is helpful if the database server was restarted. In this case, the previous connection was broken but a reconnect immediately succeeds. However, if the database is down for an extended period of time, an immediate retry does not help.

At first we need to create the MySQL Database as follows:

```
CREATE DATABASE Syslog;

USE Syslog;
```

We created a database named “maillog”. The default table named SystemEvents contain following entities.

```
CREATE TABLE SystemEvents

(

ID int unsigned not null auto_increment primary key,

CustomerID bigint,

ReceivedAt datetime NULL,

DeviceReportedTime datetime NULL,
Facility smallint NULL,
Priority smallint NULL,
FromHost varchar(60) NULL,
Message text,
NTSeverity int NULL,
Importance int NULL,
EventSource varchar(60),
EventUser varchar(60) NULL,
EventCategory int NULL,
EventID int NULL,
EventBinaryData text NULL,
MaxAvailable int NULL,
CurrUsage int NULL,
MinUsage int NULL,
MaxUsage int NULL,
InfoUnitID int NULL ,
SysLogTag varchar(60),
EventLogType varchar(60),
GenericFileName VarChar(60),
SystemID int NULL,
Checksum int NULL
)
```

We can create tables to store the log file data in the databases modifying the SystemEvents table according to our necessity. Here, we have created 3 different tables with required entities to store mail log information.

```
CREATE TABLE mailinfolog

(ID int unsigned not null auto_increment primary key,Facility smallint
NULL,NTSeverity int NULL,FromHost varchar(60) NULL,Priority smallint
NULL,datetime NULL,ReceivedAt datetime NULL,Message text );

CREATE TABLE mailwarnlog

(ID int unsigned not null auto_increment primary key,Facility smallint
NULL,NTSeverity int NULL,FromHost varchar(60) NULL,Priority smallint
NULL,datetime NULL,ReceivedAt datetime NULL,Message text );

CREATE TABLE mailerrlog

(ID int unsigned not null auto_increment primary key,Facility smallint
NULL,NTSeverity int NULL,FromHost varchar(60) NULL,Priority smallint
NULL,datetime NULL,ReceivedAt datetime NULL,Message text );
```

Above configuration settings complete our first part of the project in which the mail server sends log files to the log server and the log server stores the log files in the database.

3.4.5 Creation of web application using Java Server Pages

After the log files are stored in the mysql database, a Java Web Application is created and used to retrieve the data from the database using Eclipse IDE. It involves following steps:

i. Open the Eclipse IDE and click File menu and select other.

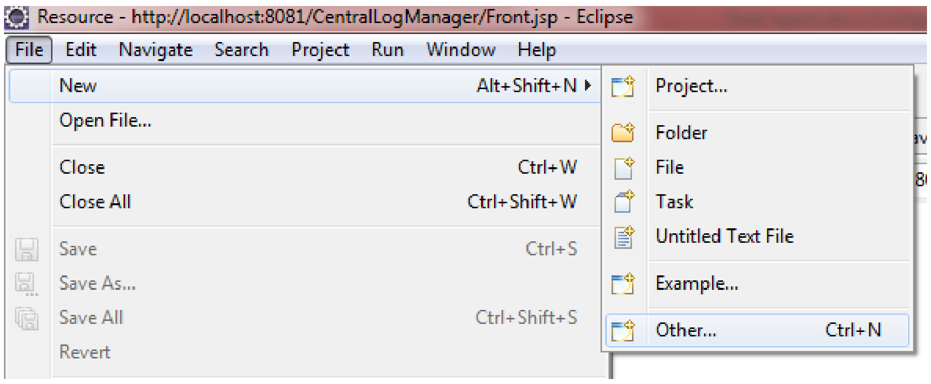


Figure 7: Create new project

ii. In the select a wizard dialog Box, choose dynamic web project as shown in the figure below and click Next.

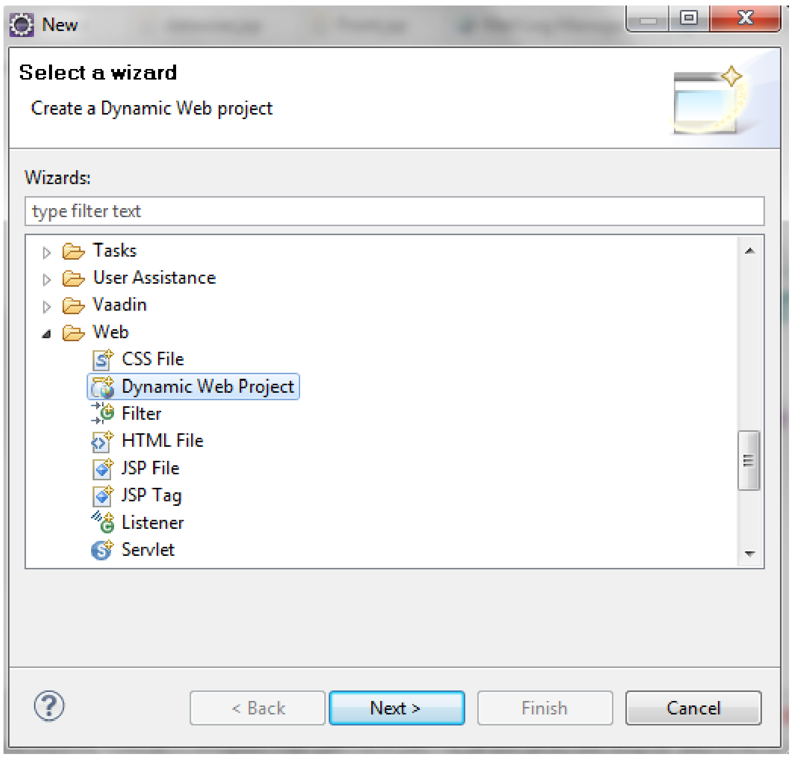
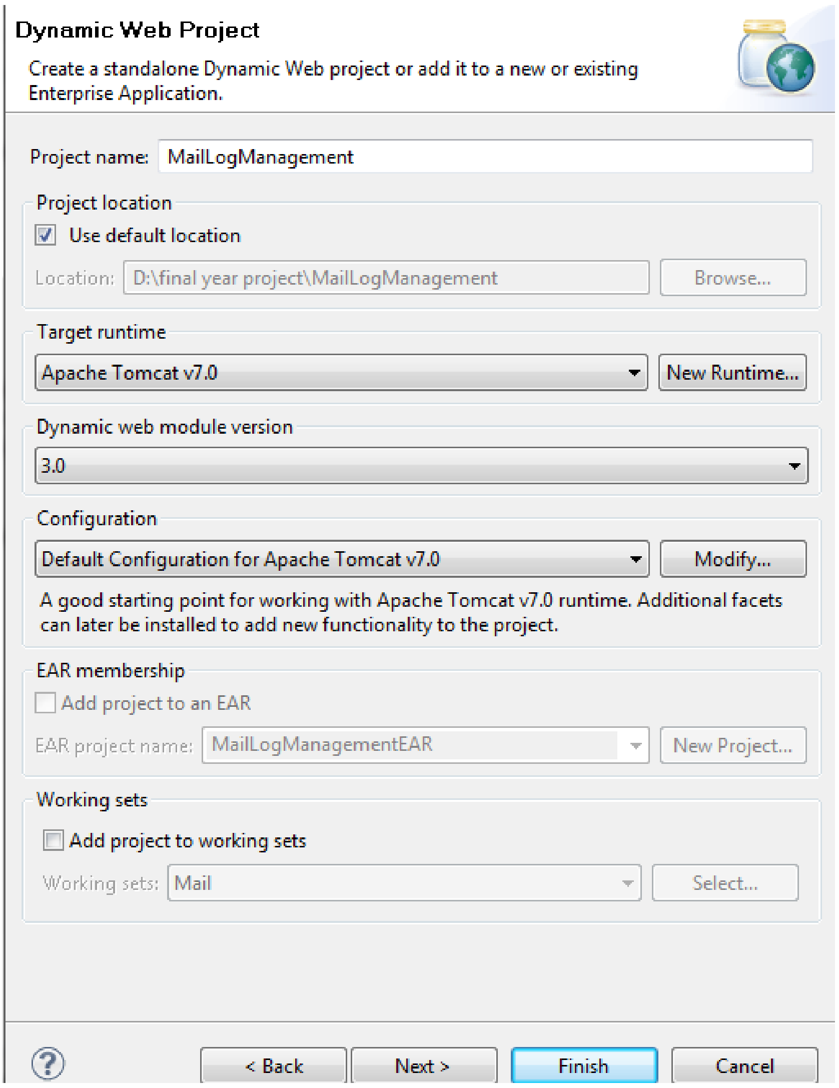


Figure 8: Create new web application project

iii. In the Dynamic web project dialog box, enter the project name, and select the target runtime environment and click Finish.



Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name: MailLogManagement

Project location

☑ Use default location

Location: D:\final year project\MailLogManagement

Target runtime: Apache Tomcat v7.0

Dynamic web module version: 3.0

Configuration: Default Configuration for Apache Tomcat v7.0

A good starting point for working with Apache Tomcat v7.0 runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership

☐ Add project to an EAR

EAR project name: MailLogManagementEAR

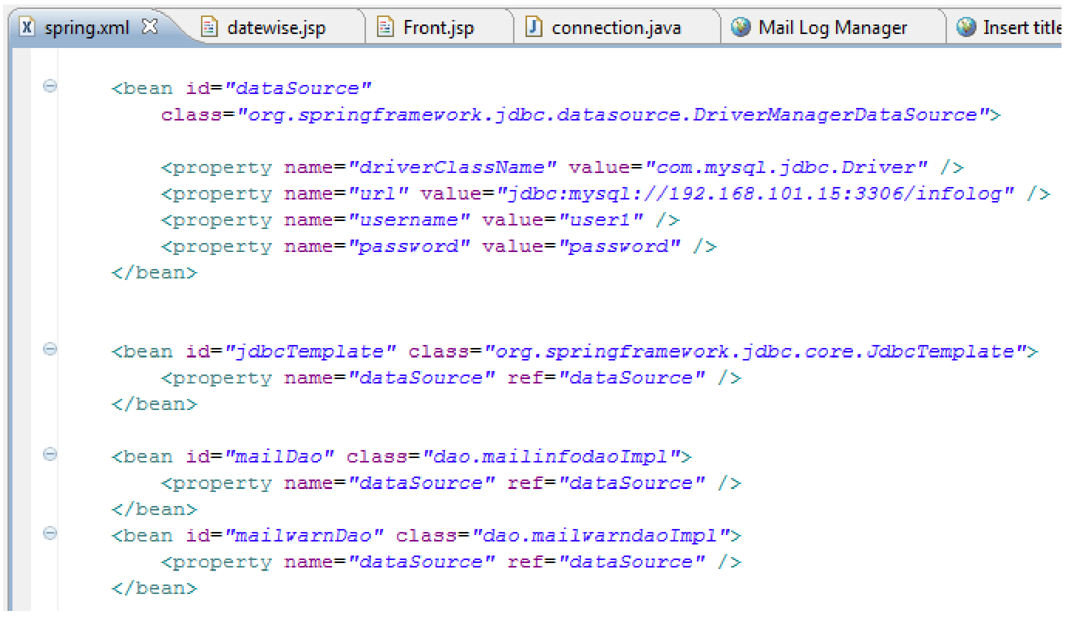
Working sets

☐ Add project to working sets

Working sets: Mail

Figure 9: Providing name and Runtime Environment of project

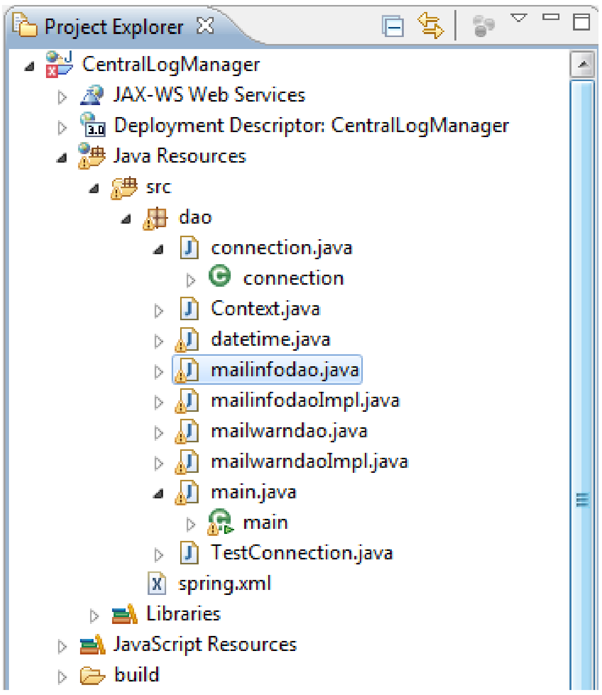
iv. Now connect to the mysql database in linux from windows Eclipse IDE through spring framework using following xml file.



```xml
<bean id="dataSource"
    class="org.springframework.jdbc.datasource.DriverManagerDataSource">

    <property name="driverClassName" value="com.mysql.jdbc.Driver" />
    <property name="url" value="jdbc:mysql://192.168.101.15:3306/infolog" />
    <property name="username" value="user1" />
    <property name="password" value="password" />
</bean>


<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
    <property name="dataSource" ref="dataSource" />
</bean>

<bean id="mailDao" class="dao.mailinfodaoImpl">
    <property name="dataSource" ref="dataSource" />
</bean>
<bean id="mailwarnDao" class="dao.mailwarndaoImpl">
    <property name="dataSource" ref="dataSource" />
</bean>
```

Figure 10: spring.xml file for database connection

v. After the database is connected, next step is to create a Data Access Object package in the Java Resources folder in which we create different java classes that provide interface and implement those interfaces. Each implementation class should be referenced in the spring.xml file. The following screenshot shows different java files created.

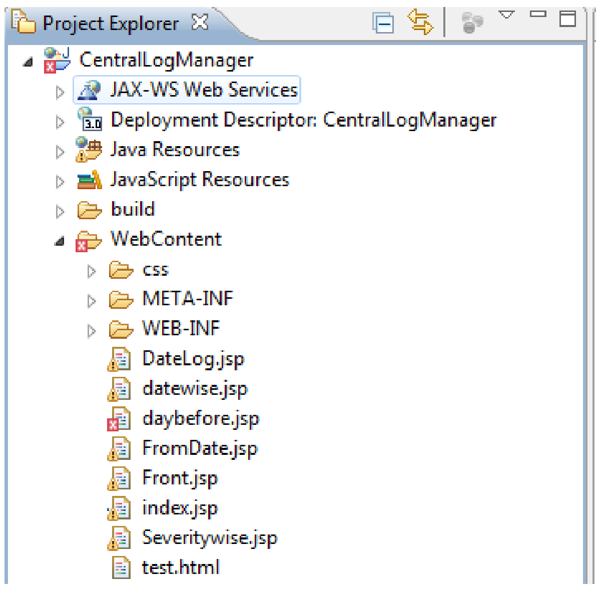


Project Explorer

```
CentralLogManager
  JAX-WS Web Services
  Deployment Descriptor: CentralLogManager
  Java Resources
    src
      dao
        connection.java
          connection
        Context.java
        datetime.java
        mailinfodao.java
        mailinfodaoImpl.java
        mailwarndao.java
        mailwarndaoImpl.java
        main.java
          main
        TestConnection.java
      spring.xml
    Libraries
  JavaScript Resources
  build
```

Figure 11: Files in dao package

vi. Now final step is to create java server pages inside the Web Content folder as shown below.


Project Explorer

CentralLogManager
- JAX-WS Web Services
- Deployment Descriptor: CentralLogManager
- Java Resources
- JavaScript Resources
- build
- WebContent
  - css
  - META-INF
  - WEB-INF
  - DateLog.jsp
  - datewise.jsp
  - daybefore.jsp
  - FromDate.jsp
  - Front.jsp
  - index.jsp
  - Severitywise.jsp
  - test.html

Figure 12: Files in WebContent folder

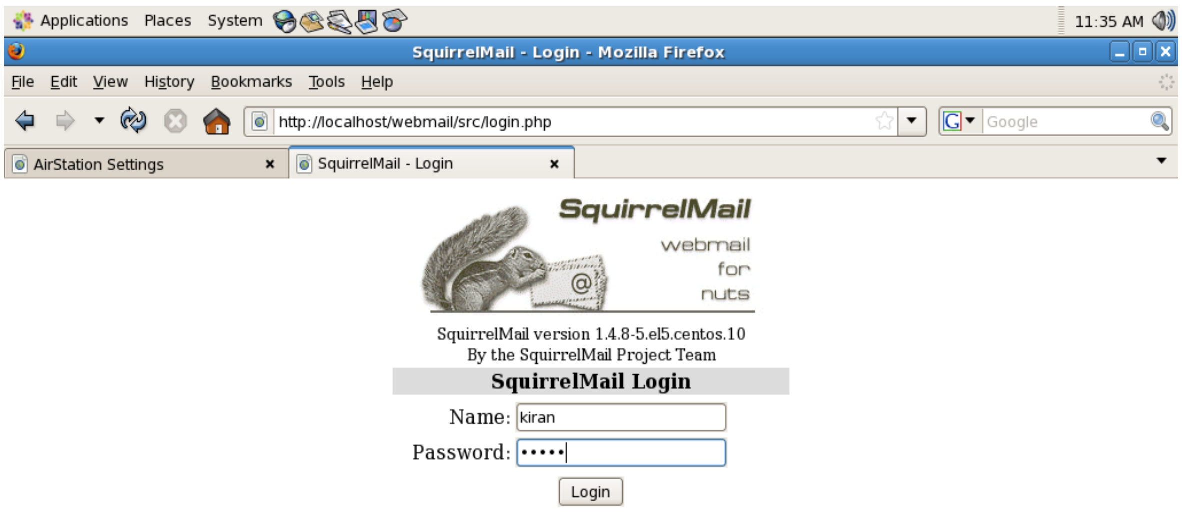
Above configuration provides the basis for programming the web application as per requirement. The detailed content of java and jsp files are included in the Appendix 2 and 3.

3.5 System Testing

Testing strategy in our project involved various aspects which are as follows:

3.5.1 Testing the operability of Web server

Web server is required in our system to run the squirrel mail web interface. The mail interface is shown below which shows the successful configuration of web server:


Applications Places System
11:35 AM
SquirrelMail - Login - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/webmail/src/login.php
Google
AirStation Settings
SquirrelMail - Login
SquirrelMail
webmail for nuts
Login

SquirrelMail version 1.4.8-5.el5.centos.10
By the SquirrelMail Project Team

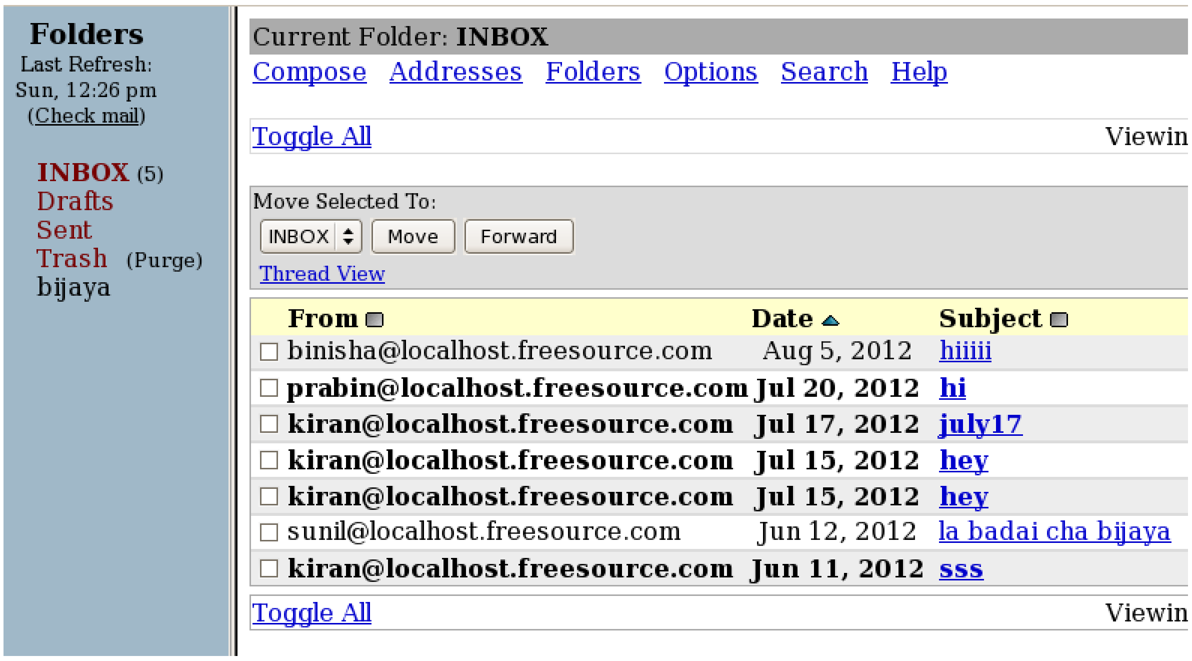
SquirrelMail Login

Name: kiran
Password: •••••

Figure 13: Squirellmail login interface

3.5.2 Testing the operability of Mail server

To test the successful operation of mail servers various user accounts were created. The following screen shots show the successful mail receipt.

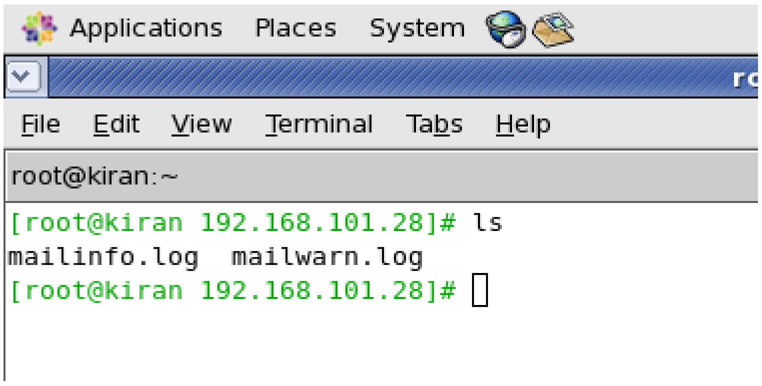


	From	Date	Subject
☐	binisha@localhost.freesource.com	Aug 5, 2012	hiiiii
☐	**prabin@localhost.freesource.com**	**Jul 20, 2012**	**hi**
☐	**kiran@localhost.freesource.com**	**Jul 17, 2012**	**july17**
☐	**kiran@localhost.freesource.com**	**Jul 15, 2012**	**hey**
☐	**kiran@localhost.freesource.com**	**Jul 15, 2012**	**hey**
☐	sunil@localhost.freesource.com	Jun 12, 2012	la badai cha bijaya
☐	**kiran@localhost.freesource.com**	**Jun 11, 2012**	**sss**

Figure 14: Mail received by bijaya@freesource.com from various user accounts

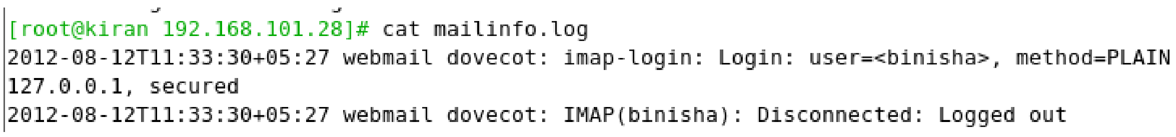
3.5.3 Testing the operability of rsyslog in log server

The rsyslog service in the mail server sends the log files to the log server in the /var/log/192.168.101.28/ directory.



```
[root@kiran 192.168.101.28]# ls
mailinfo.log  mailwarn.log
[root@kiran 192.168.101.28]#
```

Figure 15: Files received from mail server

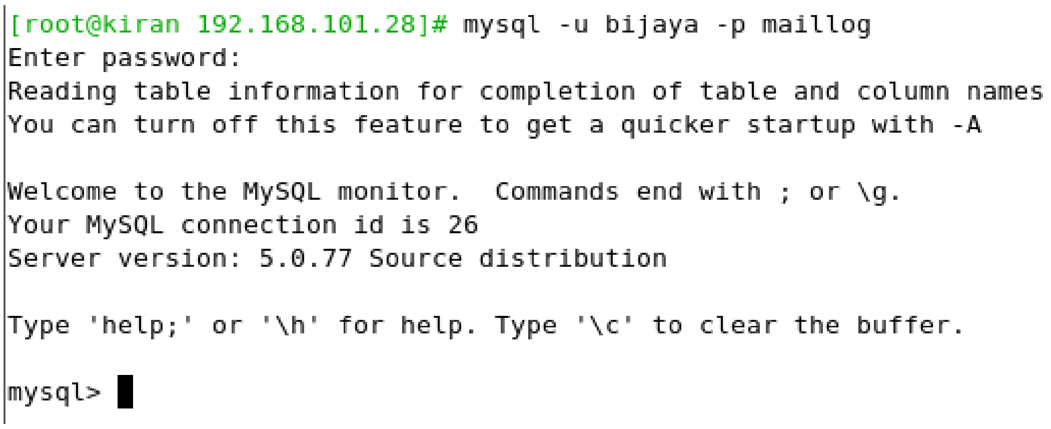


```
[root@kiran 192.168.101.28]# cat mailinfo.log
2012-08-12T11:33:30+05:27 webmail dovecot: imap-login: Login: user=<binisha>, method=PLAIN
127.0.0.1, secured
2012-08-12T11:33:30+05:27 webmail dovecot: IMAP(binisha): Disconnected: Logged out
```

Figure 16: Contents of mailinfo.log file

3.5.4 Testing the operability of rsyslog with MYSQL database

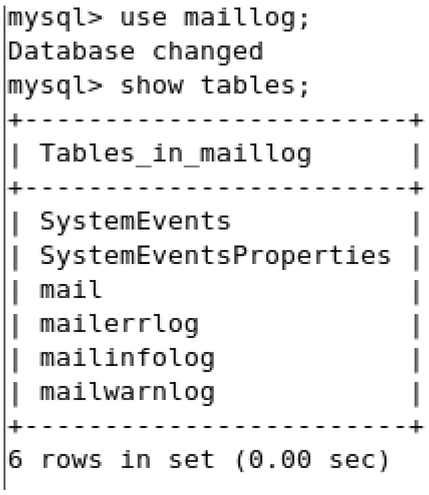
The rsyslog service connects with mysql service using package rsyslog-mysql. The successful connection results in the log file contents being stored in the mysql database. The screenshot below shows the mysql database login process:



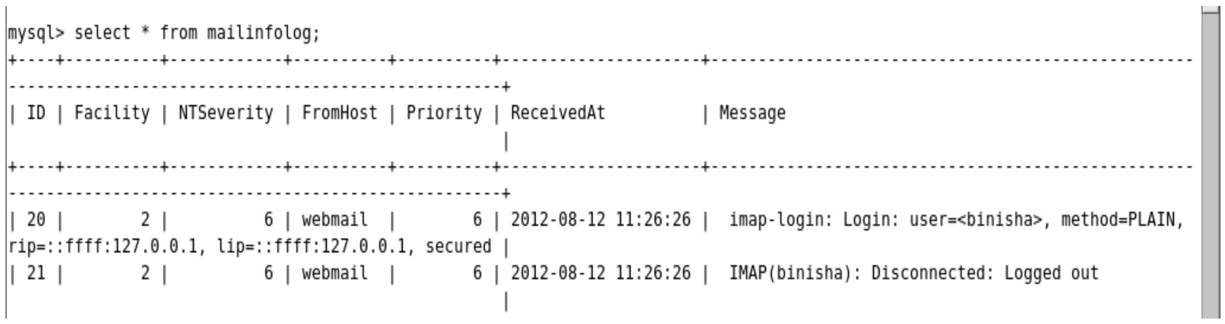
```
[root@kiran 192.168.101.28]# mysql -u bijaya -p maillog
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 26
Server version: 5.0.77 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> 
```

Figure 17: Database login process

The following screen shots show the content of the database:

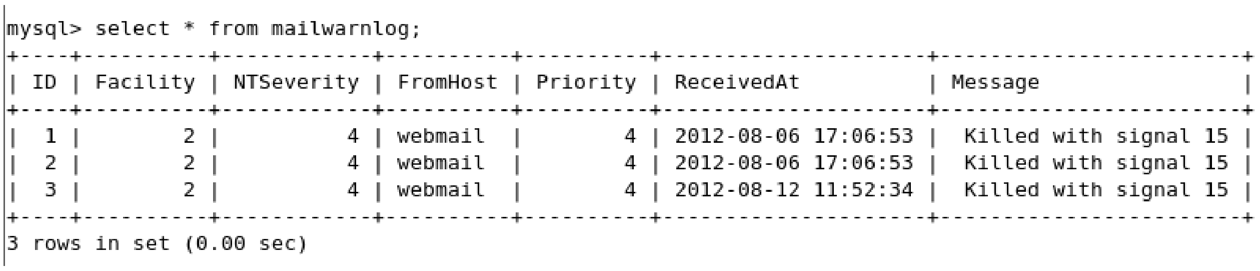


```
mysql> use maillog;
Database changed
mysql> show tables;
+------------------------+
| Tables_in_maillog      |
+------------------------+
| SystemEvents           |
| SystemEventsProperties |
| mail                   |
| mailerrlog             |
| mailinfolog            |
| mailwarnlog            |
+------------------------+
6 rows in set (0.00 sec)
```



```
mysql> select * from mailinfolog;
+----+---------+-----------+---------+---------+-------------------+-----------------------------------------------
-------------------------------------------------+
| ID | Facility | NTSeverity | FromHost | Priority | ReceivedAt          | Message
                                                 |
+----+---------+-----------+---------+---------+-------------------+-----------------------------------------------
-------------------------------------------------+
| 20 |        2 |          6 | webmail  |        6 | 2012-08-12 11:26:26 |  imap-login: Login: user=<binisha>, method=PLAIN,
rip=::ffff:127.0.0.1, lip=::ffff:127.0.0.1, secured |
| 21 |        2 |          6 | webmail  |        6 | 2012-08-12 11:26:26 |  IMAP(binisha): Disconnected: Logged out
                                                 |
```

Figure 18: Contents of mailinfolog table



```
mysql> select * from mailwarnlog;
+----+----------+------------+----------+----------+---------------------+------------------------+
| ID | Facility | NTSeverity | FromHost | Priority | ReceivedAt          | Message                |
+----+----------+------------+----------+----------+---------------------+------------------------+
|  1 |        2 |          4 | webmail  |        4 | 2012-08-06 17:06:53 |  Killed with signal 15 |
|  2 |        2 |          4 | webmail  |        4 | 2012-08-06 17:06:53 |  Killed with signal 15 |
|  3 |        2 |          4 | webmail  |        4 | 2012-08-12 11:52:34 |  Killed with signal 15 |
+----+----------+------------+----------+----------+---------------------+------------------------+
3 rows in set (0.00 sec)
```

Figure 19: Contents of maiwarnlog table

3.5.5 Testing operability of web application

The web application is run and it automatically starts the browser to show the following interface.

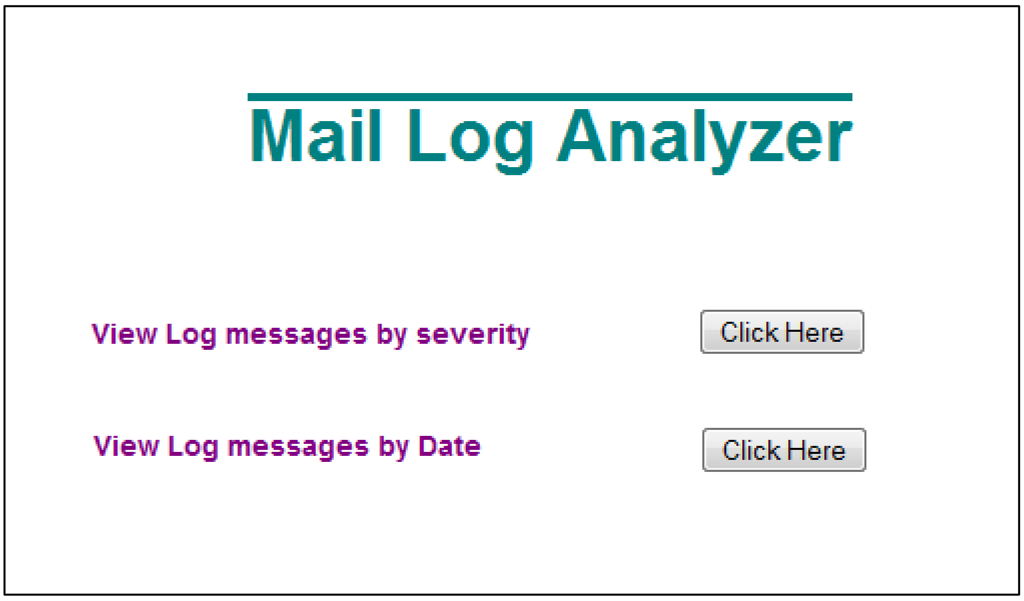
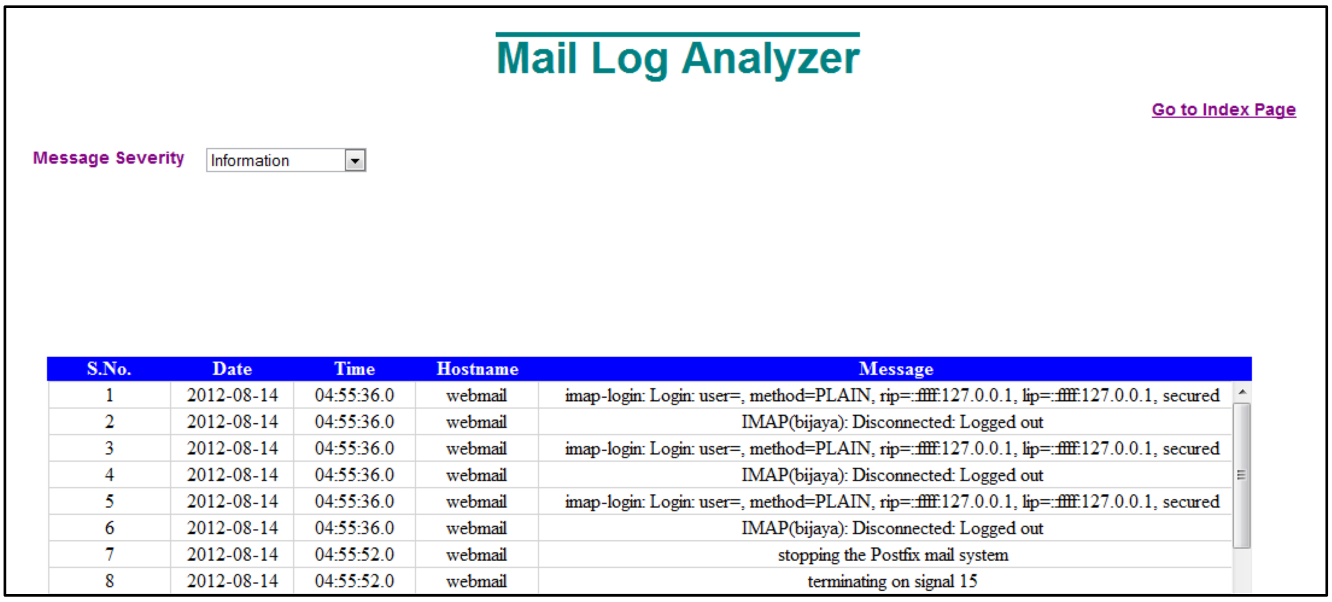


Figure 20: Browser displaying the Front Page

As shown in the snapshot above, we categorized mail log messages according to the severity and date. When we click on the severity button, next interface opens where we can select between information, warning, and error messages. The snapshot for information messages is shown below:



Mail Log Analyzer

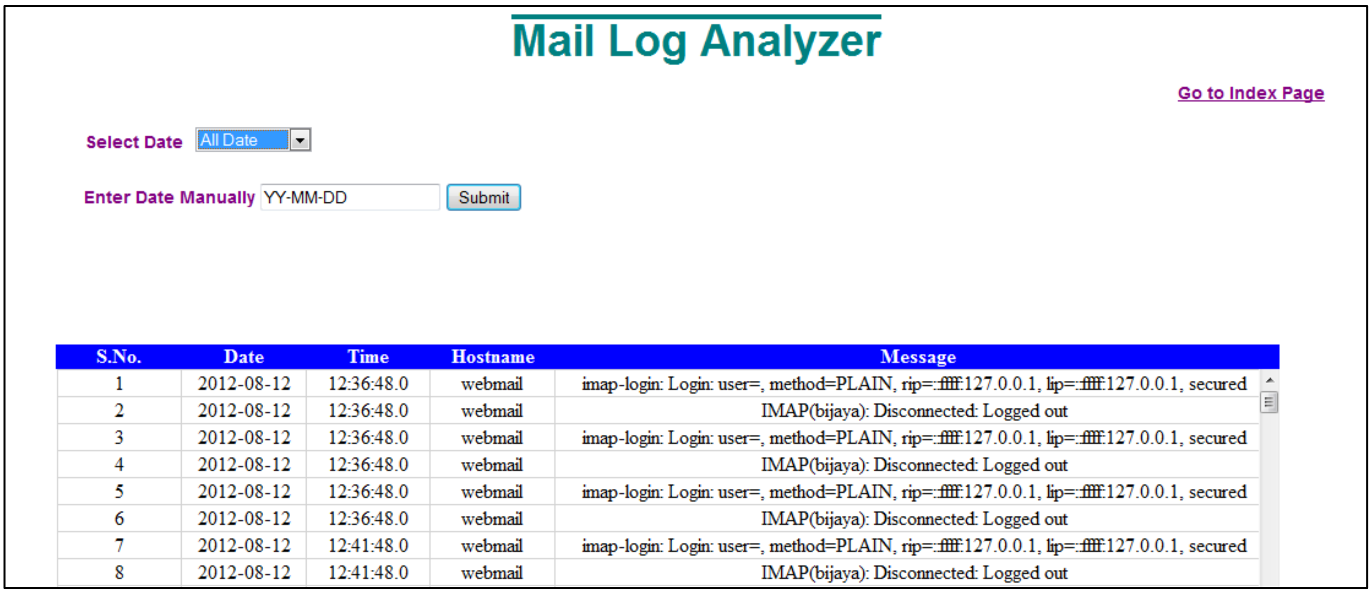
Go to Index Page

Message Severity: Information

S.No.	Date	Time	Hostname	Message
1	2012-08-14	04:55:36.0	webmail	imap-login: Login: user=, method=PLAIN, rip=::ffff:127.0.0.1, lip=::ffff:127.0.0.1, secured
2	2012-08-14	04:55:36.0	webmail	IMAP(bijaya): Disconnected: Logged out
3	2012-08-14	04:55:36.0	webmail	imap-login: Login: user=, method=PLAIN, rip=::ffff:127.0.0.1, lip=::ffff:127.0.0.1, secured
4	2012-08-14	04:55:36.0	webmail	IMAP(bijaya): Disconnected: Logged out
5	2012-08-14	04:55:36.0	webmail	imap-login: Login: user=, method=PLAIN, rip=::ffff:127.0.0.1, lip=::ffff:127.0.0.1, secured
6	2012-08-14	04:55:36.0	webmail	IMAP(bijaya): Disconnected: Logged out
7	2012-08-14	04:55:52.0	webmail	stopping the Postfix mail system
8	2012-08-14	04:55:52.0	webmail	terminating on signal 15

Figure 21: Log Message By Severity

We have also displayed log information by Date where we can select even the severity for particular date. We can also manually enter the date and view the corresponding log messages. The snapshot for log messages by all date is shown below:



Mail Log Analyzer

Go to Index Page

Select Date: All Date

Enter Date Manually: YY-MM-DD Submit

S.No.	Date	Time	Hostname	Message
1	2012-08-12	12:36:48.0	webmail	imap-login: Login: user=, method=PLAIN, rip=::ffff:127.0.0.1, lip=::ffff:127.0.0.1, secured
2	2012-08-12	12:36:48.0	webmail	IMAP(bijaya): Disconnected: Logged out
3	2012-08-12	12:36:48.0	webmail	imap-login: Login: user=, method=PLAIN, rip=::ffff:127.0.0.1, lip=::ffff:127.0.0.1, secured
4	2012-08-12	12:36:48.0	webmail	IMAP(bijaya): Disconnected: Logged out
5	2012-08-12	12:36:48.0	webmail	imap-login: Login: user=, method=PLAIN, rip=::ffff:127.0.0.1, lip=::ffff:127.0.0.1, secured
6	2012-08-12	12:36:48.0	webmail	IMAP(bijaya): Disconnected: Logged out
7	2012-08-12	12:41:48.0	webmail	imap-login: Login: user=, method=PLAIN, rip=::ffff:127.0.0.1, lip=::ffff:127.0.0.1, secured
8	2012-08-12	12:41:48.0	webmail	IMAP(bijaya): Disconnected: Logged out

Figure 22: Log messages by all date

We can also manually enter the date and view the log messages for selected message severity as shown in the two snapshots below:

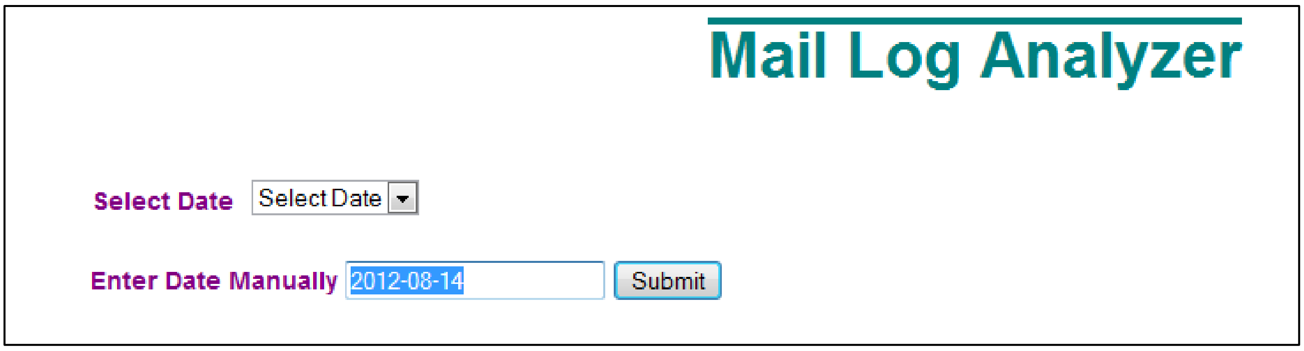
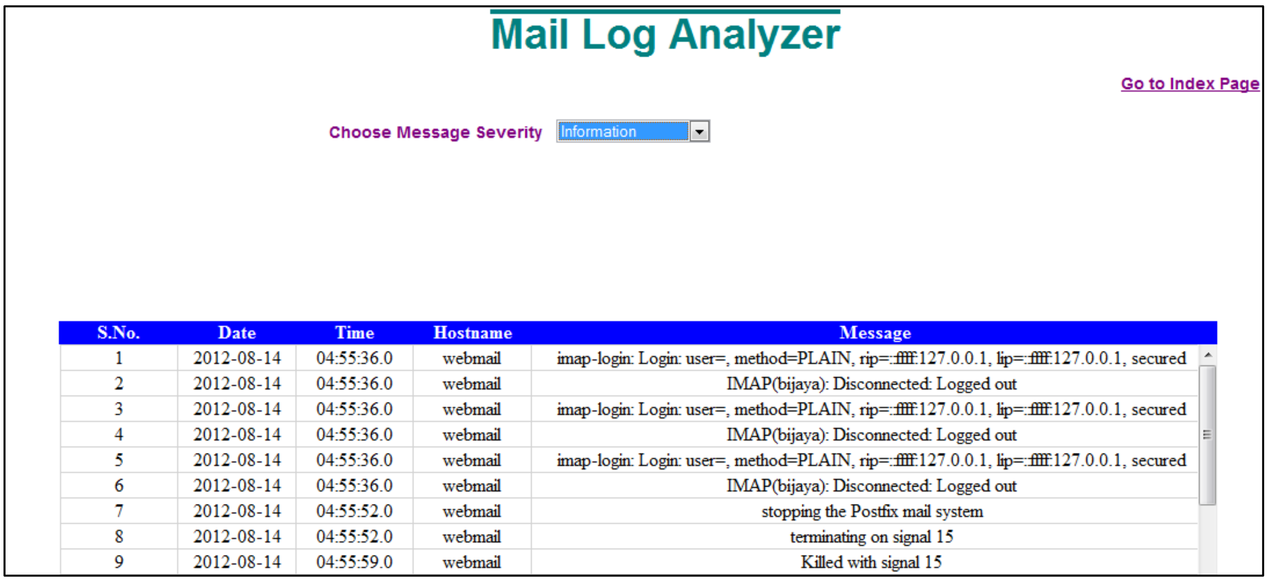


Figure 23: View log messages by entering date manually

In the above snapshot we have entered date 2012-08-12 for which the displayed log messages for message severity information is shown below:



S.No.	Date	Time	Hostname	Message
1	2012-08-14	04:55:36.0	webmail	imap-login: Login: user=, method=PLAIN, rip=::ffff:127.0.0.1, lip=::ffff:127.0.0.1, secured
2	2012-08-14	04:55:36.0	webmail	IMAP(bijaya): Disconnected: Logged out
3	2012-08-14	04:55:36.0	webmail	imap-login: Login: user=, method=PLAIN, rip=::ffff:127.0.0.1, lip=::ffff:127.0.0.1, secured
4	2012-08-14	04:55:36.0	webmail	IMAP(bijaya): Disconnected: Logged out
5	2012-08-14	04:55:36.0	webmail	imap-login: Login: user=, method=PLAIN, rip=::ffff:127.0.0.1, lip=::ffff:127.0.0.1, secured
6	2012-08-14	04:55:36.0	webmail	IMAP(bijaya): Disconnected: Logged out
7	2012-08-14	04:55:52.0	webmail	stopping the Postfix mail system
8	2012-08-14	04:55:52.0	webmail	terminating on signal 15
9	2012-08-14	04:55:59.0	webmail	Killed with signal 15

Figure 24: View log messages for date entered and chosen message severity

Chapter 4: Epilogue

4.1 Results and Analysis

The project derives following results:

- A successful mail server environment was created with the latest state-of-the-art mail server software.
- A centralized log server was created that stored all the log files, thus, freeing up the mail server from storing log files in it.
- The files created were for information, warning and error messages generated by mail server.
- The log storage in log server made the system network more secure and robust.
- The database user was created by granting all privileges that could connect from the windows system.
- The log file data was stored efficiently in the database for any kind of modification and retrieval. The tables were created in maillog database specifying the required parameters.
- A web based log file viewing interface was created using Java Enterprise Edition in Eclipse IDE. The database connectivity was provided by spring framework. As an application server Apache Tomcat version 7.0 was used. The web application made viewing and surveying log files very easy and manageable.

The analysis on project derives both positive and negative aspects. The positive aspects are:

- The package rsyslog-mysql was very useful as it connected the rsyslog logging facility and mysql database and sent all the log file data to database exactly as they were generated.
- The Java Enterprise Edition technology was very useful for developing web application in a short period of time but provided ample resources for deploying the application.

The hindering aspects in the project that are:

- The mail user agent Squirrelmail is easy to use but does not have rich graphical user interface.
- We used postfix as MTA in our mail server instead of sendmail because postfix is fast, easier-to-administer, and secure alternative to the widely-used sendmail MTA.
- The default logging facility syslog does not provide features such as message filtering. To create the proposed system, rsyslog was used that provided all the features not present in the syslog service.
- Some log file messages, such as login by user, are repeated. This redundancy requires more storage. Thus, log server must have enough storage space.
- If sendmail is not removed from the system, it conflicts with the postfix MTA and generates unnecessary log messages.
- While developing web application, it was very difficult to create the graphical user interface without the drag-and-drop features. Each component was added to the web page manually by coding in the java server pages. It would have been very easy and fast for developing application, if we had used visual frameworks with drag-and-drop features for adding components to the web page.

4.3 Problems Faced

The main problems that we faced during our project period can be enlisted as below:

- During project familiarization, finding the required tools and understanding the prerequisites was time consuming.
- The dependencies between different packages while installing the software created compatibility issues.
- The repetition of same log message from the rsyslog server created problems for database storage.
- The log messages generated from local host created inconsistent log file database.
- The database connectivity from linux to windows created issues related to retrieving log files from database.

4.4 Limitations

There are actually many things that can be done with centralized log management system but our system has limitations as listed below:

- We have only focused on mail log messages to identify the problems and security aspects of our mail server.
- We have only one client system at present which acts as mail and web server.
- We have ignored local log messages (kernel log, boot log) and only used remote logging feature of rsyslog for our system.
- The “repeated message reduction on” feature present in rsyslog configuration file for stopping redundant log messages didn’t work.
- We have used squirrelmail which performs well but is very lean and simple. Here, user administration is very dependent on the server.
- There's no such thing as adding a SquirrelMail user; what you need to do is to add users to the IMAP server.

4.5 Future Enhancements

To overcome our limitations and make our system better we can add following future enhancement:

- We can extend the number of client systems in order to view log messages from different servers.
- The number of services could be extended and even local log information can be stored in database.
- We can use syslog-ng over rsyslog because syslog-ng is more scalable and powerful. Nowadays, many organizations prefer syslog-ng to develop centralized log management system.
- The log management system can be enhanced to store and display information for messages based on other severity areas like critical, alert, debug, etc.
- The log messages could be structured more to make them more understandable.
- The system could be added with advance features such as user login counting and off-time user login detection

4.6 Conclusion

The centralized log management developed has made it easier for event analysis. Moreover, centralized logging is better when fatal device failure occurs. During such cases we can still access to the logging information. We have used rsyslog for remote logging which is an enhanced multi-threaded syslogd with focus on security and reliability. The prototype developed gave both time and resource savings, but the observations made it clear that centralization by itself is not very helpful. Centralization of log files together with some kind of visualization is proven to be very helpful; as it will reduce the time spent searching for chains of events.

The prototype covers centralization, some correlation, and visualization. The observations obtained during the testing lead to the conclusion that the system made the log files easily surveyable. The web interface to view the log file made it easily accessible and readable. The argument chain claim that quicker response time leads to increased security. This is a valid assumption, because if security analysts can monitor and respond to events in an almost real-time manner, we are moving towards being proactive instead of reactive.

The motivation was to demonstrate the centralized mail log management system using the web interface so that we can maintain accurate log of all events (mainly mail services) happening in the system in order to allow for later analysis.

References

[1] RH253: Red Hat Network Services and Security Administration, 2007 Red Hat, Inc.

[2] Christopher Negus, Linux Bible, Linux 2010 Edition

[3] Red Hat Enterprise Linux 6.0 Release Notes, 2010 Red Hat, Inc.

[4] RH033: Red Hat Enterprise Linux Essentials, 2008 Red Hat, Inc.

[5] Jae-Young Kim and James Won-Ki Hong, "Design and Implementation of a Web-Based Internet/Intranet Mail Server Management System", 2010

[6] Dovecot Documentation, http://wiki1.dovecot.org/ (Accessed: 10th June, 2012)

[7] Postfix Documentation, http://www.postfix.org/BASIC_CONFIGURATION_README.html (Accessed: 10th June, 2012)

[7] Rsyslog-Documentation, http://www.rsyslog.com/doc/manual.html (Accessed: 8th July, 2012)

Appendix 1

Configuration Files

i. Postfix – main.cf

```
mail_owner = postfix

myhostname = webmail.freesource.com

mydomain = freesource.com

myorigin = webmail.freesource.com

home_mailbox = Maildir/

mail_spool_directory = /var/spool/mail
```

ii. Dovecot – dovecot.conf

```
Protocols = imap, pop, pop3
```

iii. mail server – rsyslog.conf

```
$RepeatedMsgReduction on ()       # log single message for same msgs

# Log all kernel messages to the console.

# Logging much else clutters up the screen.

#kern.*                                        /dev/console



# Log anything (except mail) of level info or higher.

# Don't log private authentication messages!

#*.info;mail.none;authpriv.none;cron.none               /var/log/messages

# The authpriv file has restricted access.

authpriv.*                                      /var/log/secure

# Log all the mail messages in one place.

#mail.err                                                  /var/log/maillog.err

#mail.warn                                                /var/log/maillog.wrn
```

```
#mail.info                                          /var/log/maillog.info
mail.err @192.168.101.191:514
mail.info @192.168.101.191:514
mail.warn @192.168.101.191:514
#mail.* @192.168.101.191:514
#*.* @@192.168.101.191:514
#mail.info :omrelp:192.168.101.191:514
# Log cron stuff
cron.*                              /var/log/cron
# Everybody gets emergency messages
#*.emerg                              *
# Save news errors of level crit and higher in a special file.
uucp,news.crit                          /var/log/spooler
# Save boot messages also to boot.log
local7.*                            /var/log/boot.log
```

iv. log server – rsyslog.conf

```
$ModLoad ommysql
$RepeatedMsgReduction on ()        # log single message for same msgs
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                             /dev/console
# Add your Client server IP or IP Range
$AllowedSender UDP,192.168.101.28
# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
```

```
#*.info;mail.none;authpriv.none;cron.none              /var/log/messages

# The authpriv file has restricted access.

#authpriv.*                                  /var/log/secure

# Log all the mail messages in one place.

#mail.*@192.168.101.28                         /var/log/maillog

# Log cron stuff

cron.*                                      /var/log/cron

# Everybody gets emergency messages

*.emerg                                      *

# Save news errors of level crit and higher in a special file.

uucp,news.crit                               /var/log/spooler

# Save boot messages also to boot.log

local7.*                                    /var/log/boot.log

# provides support for local system logging

#$ModLoad imuxsock

# provides kernel logging support (previously done by rklogd)

#$ModLoad imklog

# provides UDP syslog reception. For TCP, load imtcp.

$ModLoad imudp

# For UDP, InputServerRun 514

$UDPServerRun 514

# This one is the template to generate the log filename dynamically, depending on
    the client's IP address.

$template FILE1,"/var/log/%fromhost-ip%/mailinfo.log"

$template FILE2,"/var/log/%fromhost-ip%/mailwarn.log"
```

```
$template FILE3,"/var/log/%fromhost-ip%/mailerr.log"

#$template FILE4,"/var/log/%fromhost-ip%/mail.log"

#$template StdSQLFormat,"insert into SystemEvents (Message, Facility,
    FromHost, Priority, DeviceReportedTime, ReceivedAt, InfoUnitID,
    SysLogTag) values ('%msg%', %syslogfacility%, '%HOSTNAME%',
    %syslogpriority%, '%timereported:::date-mysql%', '%timegenerated:::date-
    mysql%', %iut%, '%syslogtag%')",SQL

$template mailinfo,"insert into mailinfolog (Facility, NTSeverity, FromHost,
    Priority, DeviceReportedTime, ReceivedAt, Message ) values
    (%syslogfacility%, %syslogseverity%, '%HOSTNAME%', %syslogpriority%,
    '%timereported:::date-mysql%', '%timegenerated:::date-mysql%', '%msg%'
    )",SQL

$template mailwarn,"insert into mailwarnlog (Facility, NTSeverity, FromHost,
    Priority, DeviceReportedTime, ReceivedAt, Message ) values
    (%syslogfacility%, %syslogseverity%, '%HOSTNAME%', %syslogpriority%,
    '%timereported:::date-mysql%', '%timegenerated:::date-mysql%', '%msg%'
    )",SQL

$template mailerr,"insert into mailerrlog (Facility, NTSeverity, FromHost,
    Priority, DeviceReportedTime, ReceivedAt, Message ) values
    (%syslogfacility%, %syslogseverity%, '%HOSTNAME%', %syslogpriority%,
    '%timereported:::date-mysql%', '%timegenerated:::date-mysql%', '%msg%'
    )",SQL

#$template mylog,"insert into
    mail(ReceivedAt,Facility,NTSeverity,FromHost,Message) values
    (%syslogfacility%,%syslogseverity%,'%HOSTNAME%','%msg%')",SQL

# Log all messages to the dynamically formed file. Now each clients log
    (192.168.1.2, 192.168.1.3,etc...), will be under a separate directory which is
    formed by the template FILENAME.

mail.info ?FILE1
mail.warn ?FILE2

mail.err ?FILE3

#mail.* ?FILE4

#mail.info :ommysql:localhost,maillog,bijaya,password;StdSQLFormat

mail.info  :ommysql:127.0.0.1,maillog,bijaya,password;mailinfo

mail.warn  :ommysql:127.0.0.1,maillog,bijaya,password;mailwarn

mail.err  :ommysql:127.0.0.1,maillog,bijaya,password;mailer
```

Appendix 2

JSP files

index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-
1" pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-
8859-1">
<title>Insert title here</title>
</head>
<body>
<span id="title"
style="color: teal; font-family: Arial, Helvetica, sans-serif;
font-size: 36px; font-style: normal; font-weight: bold; left:
477px; top: 121px; position: absolute; text-decoration:
overline">MailLog Analyzer</span>
<table border="1" style="border: 1px solid;">

<span id="severity"
style="color: purple; font-family: Arial, Helvetica, sans-serif;
font-size: 14px; font-style: normal; font-weight: bold; left:
447px; top: 209px; position: absolute; width: 254px; height:
27px;">View Log messages by severity</span>

<span id="view_by_date"
style="color: purple; font-family: Arial, Helvetica, sans-serif;
font-size: 14px; font-style: normal; font-weight: bold; left:
449px; top: 254px; position: absolute; width: 254px; height:
27px;">View Log messages by Date</span>
<span id="button"
style="color: purple; font-family: Arial, Helvetica, sans-serif;
font-size: 14px; font-style: normal; font-weight: bold; left:
```

```
705px; top: 204px; position: absolute; width: 100px; height:
27px;">
<form action="Severitywise.jsp" method="post" name="form1">
<input name="submit" type="submit" id="submit" value="Click Here"
/> </form>
</span>
<span id="button"
style="color: purple; font-family: Arial, Helvetica, sans-serif;
font-size: 14px; font-style: normal; font-weight: bold; left:
707px; top: 253px; position: absolute; width: 100px; height:
27px;">
<form action="FromDate.jsp" method="post" name="form1">
<input name="submit" type="submit" id="submit" value="Click Here"
/>
</form>

</span>
</table>
</body>
</html>
```

SeverityWise.jsp

```
<%@page import="java.util.Map"%>
<%@page import="java.util.List"%>
<%@page import="java.text.SimpleDateFormat"%>
<%@page import="java.util.Calendar"%>
<%@page import="dao.datetime"%>
<%@page import="dao.mailwarndao"%>

<%@page import="java.util.Map"%>
<%@page import="java.util.List"%>

<%@page import="dao.Context"%>
<%@page import="dao.mailinfodao"%>

<%@page import="dao.connection"%>
```

```
<%
mailinfodao
mailInfo=(mailinfodao)Context.get().getBean("mailDao");
mailwarndao
mailwarn=(mailwarndao)Context.get().getBean("mailwarnDao");

%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-
1" pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-
8859-1">
<title>Mail Log Manager</title>
<link href="css/stylesheet.css" rel="stylesheet" type="text/css"/>
<script type="text/javascript">

function display(obj,id1,id2,id3) {
txt = obj.options[obj.selectedIndex].value;
document.getElementById(id1).style.display = 'none';
document.getElementById(id2).style.display = 'none';
if ( txt.match(id1) ) {
document.getElementById(id1).style.display = 'block';
}
if ( txt.match(id2) ) {
document.getElementById(id2).style.display = 'block';
}
}

function displ(obj,id1,id2) {
      txt = obj.options[obj.selectedIndex].value;
      document.getElementById(id1).style.display = 'none';
      document.getElementById(id2).style.display = 'none';
      if ( txt.match(id1) ) {
      document.getElementById(id1).style.display = 'block';
      }
      if ( txt.match(id2) ) {
      document.getElementById(id2).style.display = 'block';
      }
      }

</script>
</head>
<body>
<form id="form1" name="form1" method="post" action="datewise.jsp">
      <div id="wrap" align="center">

            <div id="top">

<span id="title" style="color: teal; font-family: Arial,
Helvetica, sans-serif; font-size: 36px; font-style: normal; font-
weight: bold; left: 480px; top: 24px; position: absolute; text-
decoration: overline">MailLog Analyzer</span>

<span id="severity" style="color: purple; font-family: Arial,
Helvetica, sans-serif; font-size: 14px; font-weight: bold; left:
96px; top: 120px; position: absolute">Message Severity</span>

<span id="form1:dropDownSeverity" style="left: 240px; top: 120px;
position: absolute">
<select name="dropDownBox"
onchange="display(this,'information','warning','error');">

<option> Select option here</option>
<option value="information">Information</option>
<option value="warning">Warning</option>
<option value="error">Error</option>
</select>
</span>
<span id="link" style="color: purple; font-family: Arial,
Helvetica, sans-serif; font-size: 14px; font-style: normal; font-
weight: bold; left: 1003px; top: 80px; position: absolute; width:
163px; height: 20px;"><a href="Front.jsp">Go to Index
Page</a></span>
```

```
      </div>

            <div id="bottom">
                  <div style="margin: 0; padding: 0; border-
collapse: collapse; width: 1000px; height: 300px; overflow:
hidden;" border: 1px solid black;>
                        <table
                              style="margin: 0; padding: 0;
border-collapse: collapse; color: White; width: 999px; height:
20px; text-align: left; background-color: Blue;">
<colgroup>
<col width="100px" />
<col width="100px" />
<col width="100px" />
<col width="100px" />
<col width="580px" />
<col width="16px" />
</colgroup>
<tbody>
<tr style="margin: 0; padding: 0; border-collapse: collapse;">
<th style="margin: 0; padding: 0; border-collapse:
collapse;">S.No.</th>
<th style="margin: 0; padding: 0; border-collapse:
collapse;">Date</th>
<th style="margin: 0; padding: 0; border-collapse:
collapse;">Time</th>
<th style="margin: 0; padding: 0; border-collapse:
collapse;">Hostname</th>
<th style="margin: 0; padding: 0; border-collapse:
collapse;">Message</th>
</tr>
</tbody>
</table>
      <div style="margin: 0; padding: 0; border-collapse:
collapse; width: 998px; height: 300px; overflow: auto;">
      <table style="margin: 0; padding: 0; border-collapse:
collapse; width: 995px;">

<colgroup>
```

```
<col width="100px" />
<col width="100px" />
<col width="100px" />
<col width="100px" />
<col width="580px" />
</colgroup>
<tbody id="information" style="margin: 0; padding: 0; border-
collapse: collapse; display: none">
<%
int i=0;
List<Map> lists=mailInfo.getAll();
for(Map m:lists){
      String message=m.get("Message").toString();
      String hostname=m.get("FromHost").toString();
      String date=m.get("ReceivedAt").toString();
      String tokens [] = date.split(" ");
      String d = tokens[0];
      String time = tokens[1];

      i++;
%>

<tr style="margin: 0; padding: 0; border-collapse: collapse;">
<td style="border: 1px solid lightgrey;"><%=i %></td>
<td style="border: 1px solid lightgrey;"><%=d%></td>
<td style="border: 1px solid lightgrey;"><%=time%></td>
<td style="border: 1px solid lightgrey;"><%=hostname %></td>
<td style="border: 1px solid lightgrey;"><%=message %></td>

</tr>

                                    <%
}

%>




</tbody>
```

```
<tbody id="warning" style="margin: 0; padding: 0; border-collapse:
collapse; display: none;">

<%
int j=0;
List<Map> listss=mailwarn.getAll();
for(Map m:listss){
      String messages=m.get("Message").toString();
      String host=m.get("FromHost").toString();
      String dat=m.get("ReceivedAt").toString();
      String token [] = dat.split(" ");
      String da = token[0];
      String timee = token[1];

      j++;
%>

<tr style="margin: 0; padding: 0; border-collapse: collapse;">
<td style="border: 1px solid lightgrey;"><%=j %></td>
<td style="border: 1px solid lightgrey;"><%=da%></td>
<td style="border: 1px solid lightgrey;"><%=timee%></td>
<td style="border: 1px solid lightgrey;"><%=host %></td>
<td style="border: 1px solid lightgrey;"><%=messages %></td>

</tr>

<%
}

%>

</tbody>
</table>
                        </div>
                  </div>
```

```
            </div>

      </div>
</form>
</body>
</html>
```

Datewise.jsp

```
<%@page import="java.util.Map"%>
<%@page import="java.util.List"%>
<%@page import="dao.mailwarndao"%>
<%@page import="dao.Context"%>
<%@page import="dao.mailinfodao"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-
1" pageEncoding="ISO-8859-1"%>

 <%
Mailinfodao
mailInfo=(mailinfodao)Context.get().getBean("mailDao");
mailwarndao
mailwarn=(mailwarndao)Context.get().getBean("mailwarnDao");
String input = request.getParameter("date");

 %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-
8859-1">
<title>Insert title here</title>
<link href="css/stylesheet.css" rel="stylesheet" type="text/css"
/>
<script type="text/javascript">

function display(obj,id1,id2,id3) {
txt = obj.options[obj.selectedIndex].value;
document.getElementById(id1).style.display = 'none';
document.getElementById(id2).style.display = 'none';
if ( txt.match(id1) ) {
document.getElementById(id1).style.display = 'block';
}
if ( txt.match(id2) ) {
document.getElementById(id2).style.display = 'block';
}
}

function displ(obj,id1,id2) {
      txt = obj.options[obj.selectedIndex].value;
      document.getElementById(id1).style.display = 'none';
      document.getElementById(id2).style.display = 'none';
      if ( txt.match(id1) ) {
      document.getElementById(id1).style.display = 'block';
      }
      if ( txt.match(id2) ) {
      document.getElementById(id2).style.display = 'block';
      }
      }


</script>
</head>
<body>
<div id="wrap" align="center">

            <div id="top">

<span id="title" style="color: teal; font-family: Arial,
Helvetica, sans-serif; font-size: 36px; font-style: normal; font-
weight: bold; left: 480px; top: 24px; position: absolute; text-
decoration: overline">MailLog Analyzer</span>

<span id="severity" style="color: purple; font-family: Arial,
Helvetica, sans-serif; font-size: 14px; font-weight: bold; left:
341px; top: 122px; position: absolute">Choose Message
Severity</span> <span id="form1:dropDownSeverity" style="left:
537px; top: 119px; position: absolute">

<select name="dropDownBox"
onchange="display(this,'information','warning','error');">

<option> Select option here</option>
<option value="information">Information</option>
```

```
<option value="warning">Warning</option>
<option value="error">Error</option>

</select> </span>

<span id="link" style="color: purple; font-family: Arial,
Helvetica, sans-serif; font-size: 14px; font-style: normal; font-
weight: bold; left: 1003px; top: 80px; position: absolute; width:
163px; height: 20px;"><a href="Front.jsp">Go to Index
Page</a></span>

</div>
      <div id="bottom">
                  <div style="margin: 0; padding: 0; border-
collapse: collapse; width: 1000px; height: 300px; overflow:
hidden;">
      <table style="margin: 0; padding: 0; border-collapse:
collapse; color: White; width: 999px; height: 20px; text-align:
left; background-color: Blue;">

<colgroup>
      <col width="100px" />
      <col width="100px" />
      <col width="100px" />
      <col width="100px" />
      <col width="580px" />
      <col width="16px" />
</colgroup>

<tbody>

<tr style="margin: 0; padding: 0; border-collapse: collapse;">
<th style="margin: 0; padding: 0; border-collapse:
collapse;">S.No.</th>
<th style="margin: 0; padding: 0; border-collapse:
collapse;">Date</th>
<th style="margin: 0; padding: 0; border-collapse:
collapse;">Time</th>
```

```
<th style="margin: 0; padding: 0; border-collapse:
collapse;">Hostname</th>
<th style="margin: 0; padding: 0; border-collapse:
collapse;">Message</th>
      </tr>
      </tbody>
      </table>

<div style="margin: 0; padding: 0; border-collapse: collapse;
width: 998px; height: 300px; overflow: auto;">
<table style="margin: 0; padding: 0; border-collapse: collapse;
width: 995px;">

<colgroup>
      <col width="100px" />
      <col width="100px" />
      <col width="100px" />
      <col width="100px" />
      <col width="580px" />
</colgroup>

<tbody id="information" style="margin: 0; padding: 0; border-
collapse: collapse; display: none">

<%
int i=0;
List<Map> lists=mailInfo.getAll();
for(Map m:lists){
      String message=m.get("Message").toString();
      String hostname=m.get("FromHost").toString();
      String date=m.get("ReceivedAt").toString();
      String tokens [] = date.split(" ");
      String d = tokens[0];
      String time = tokens[1];

      i++;

      if (d.equals(input)){
%>
```

```
<tr style="margin: 0; padding: 0; border-collapse: collapse;">
<td style="border: 1px solid lightgrey;"><%=i %></td>
<td style="border: 1px solid lightgrey;"><%=d%></td>
<td style="border: 1px solid lightgrey;"><%=time%></td>
<td style="border: 1px solid lightgrey;"><%=hostname %></td>
<td style="border: 1px solid lightgrey;"><%=message %></td>
</tr>

<%
}
}

%>




</tbody>
      <tbody id="warning" style="margin: 0; padding: 0; border-
collapse: collapse; display: none;">

<%
int j=0;
List<Map> listss=mailwarn.getAll();
for(Map m:listss){
      String messages=m.get("Message").toString();
      String host=m.get("FromHost").toString();
      String dat=m.get("ReceivedAt").toString();
      String token [] = dat.split(" ");
      String da = token[0];
      String timee = token[1];

      j++;
      if (da.equals(input)){
%>

<tr style="margin: 0; padding: 0; border-collapse: collapse;">
```

```
<td style="border: 1px solid lightgrey;"><%=j %></td>
<td style="border: 1px solid lightgrey;"><%=da%></td>
<td style="border: 1px solid lightgrey;"><%=timee%></td>
<td style="border: 1px solid lightgrey;"><%=host %></td>
<td style="border: 1px solid lightgrey;"><%=messages %></td>

</tr>

<%
}
}

%>

</tbody>

</table>
                        </div>
                  </div>



            </div>


      </div>
</body>
</html>
```

Appendix 3

Java Files

mailinfodao.java

```
package dao;

import java.util.List;

public interface mailinfodao {

      public List getAll();
      public List getByDate(String date);
      public List getByTime(String time);
}
```

mailinfodaoImpl.java

```
package dao;

import java.util.List;

import org.springframework.jdbc.core.JdbcTemplate;



public class mailinfodaoImpl extends JdbcTemplate implements mailinfodao {

      @Override
      public List getAll() {
            // TODO Auto-generated method stub
            String sql = "SELECT * FROM mailinfolog order by ID ASC";
            return this.queryForList(sql);
      }
}

      @Override
      public List getByDate(String date) {
            // TODO Auto-generated method stub
      String sql = "SELECT * FROM mailinfolog WHERE ReceivedAt = ?";
            return this.queryForList(sql, new Object[] { date });
      }
```

Datetime.java

```
package dao;

import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.List;
import java.util.Map;

public class datetime {

        //static mailinfodao
mailInfo=(mailinfodao)Context.get().getBean("mailDao");
         public static final String DATE_FORMAT_NOW = "yyyy-MM-dd
HH:mm:ss";

         public static String DateTime()
         {  Calendar cal = Calendar.getInstance();
            SimpleDateFormat sdf = new
SimpleDateFormat(DATE_FORMAT_NOW);
            return sdf.format(cal.getTime());
         }

}
```