

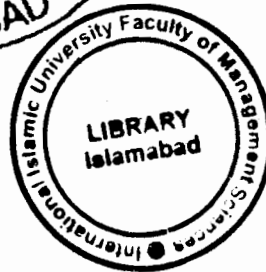
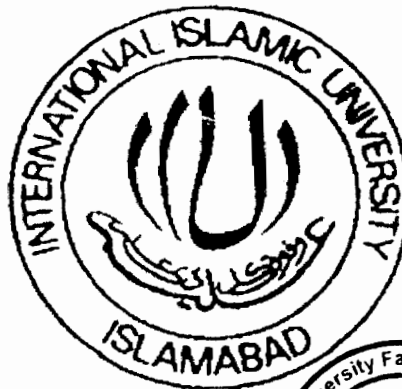


Acc. No. (PMS) T-672

Internet Services Management Console

T00672

DATA ENTERED



Developed by

Muhammad Imran

Supervised by

Ch. Waqas-ur-Rehman

**Department of Computer Science
International Islamic University, Islamabad
(2001)**

DATA ENTERED

MA/MSC
005.4
MUI

1-Graphical user interfaces (computer systems) - software.

2.5
7
2.2.11



**DEPARTMENT OF COMPUTER SCIENCE
INTERNATIONAL ISLAMIC UNIVERSITY, ISLAMABAD**

FINAL APPROVAL

Ass. No. (FMS)

F672

It is certified that we have examined the project and read the accompanied report submitted by *Mr. Muhammad Imran*. It is our judgment that this report is of sufficient standard to warrant its acceptance by International Islamic University, Islamabad for the Master Degree in Computer Science.

COMMITTEE

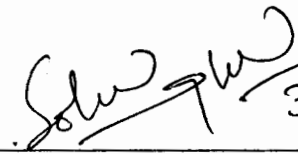
External Examiner

Dr. Nazir Ahmed Sangi
Chairman,
Department of Computer Science,
Allama Iqbal Open University,
Islamabad, Pakistan



Internal Examiners

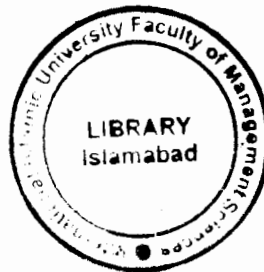
Mr. Suhail Iqbal Ayubi
Lecturer,
Department of Computer Science,
International Islamic University,
Islamabad, Pakistan


_____ 3/1/2002

Supervisor

Mr. Waqas-ur-Rehman
Assistant Professor,
Department of Computer Science,
International Islamic University,
Islamabad, Pakistan





Dated: 3rd January, 2002

A REPORT SUBMITTED TO
INTERNATIONAL ISLAMIC UNIVERSITY, ISLAMABAD
AS
A PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE AWARD OF
MASTER DEGREE IN COMPUTER SCIENCE

Dedicated
to
My Parents

whom prayers and attentions made me able to
reach at this extent

Acknowledgements

All praise to Almighty Allah, the most merciful and compassionate, without His help and blessing, I was unable to complete this project.

This project could not have come about without the help, encouragement and guidance of the following persons: -

My respected project supervisor and teacher **Ch. Waqas-ur-Rehman**. He was available to me whenever I consulted him. He always reserved some time for me though being very busy, whenever I needed any help of any nature from him. Without his guidance and help, I could never be able to develop such software.

My respected supervisors (external) **Mr. Suhail Iqbal Ayubi** and **Mr. Mata-ur-Rehman**. Their sincere professional guidance, encouragement and appreciation remained with me throughout the way. Without their precious guidance and help, I could never be able to develop such software.

My course mates and fellows especially **Farhan Ahmed** (POL), I am really grateful for their cooperation. I always remember their cooperation and support.

And finally my parents, brothers and sisters who provide me every suitable support both morally and financially.

Muhammad Imran

Declaration

I, Muhammad Imran hereby declare that this software neither as a whole nor as a part thereof has been copied out from any source. It is further declared that I have developed this software and the accompanied report entirely on the basis of my personal efforts made under the sincere guidance of my teacher and project supervisor **Ch. Waqas-ur-Rehman** and external supervisor **Mr. Suhail Iqbal Ayubi**. No portion of this project work, both the software and the report has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning.

Muhammad Imran

Project in Brief

Project Title	:	Internet Services Management Console
Objective	:	Web-based Graphical User Interface (GUI) for the configuration of Apache Web Server, DNS Server and SendMail.
Undertaken by	:	Muhammad Imran
Supervised by	:	Ch. Waqas-ur-Rehman
Date Started	:	20 th March 2001
Date Completed	:	8 th September 2001
Tools Used	:	Java Servlets, Turbo C, HTML, JavaScript, MySQL
Operating System	:	Red Hat Linux 6.2
Target System	:	Any operating system with JVM support
System Used	:	Intel Pentium III, 733 MHz

Abstract

Internet Services Management Console provides web-based Graphical User Interface for the process of configuration of Apache Web Server, DNS Server and SendMail. It is basically administrative software, which provides services for administrators to configure their servers. Originally the configuration of Apache, DNS and SendMail is based on the configuration in the directives of the text based configuration files. The Administrator can configure the global server, the configuration of IP-Based virtual domains and also the email configuration remotely anywhere over the Internet. This software facilitates the administrator to input only valid values for each and every directive and the configuration is made automatically in the right way and the relevant server restarted after the configuration to successfully complete the process. Administrator can also host a virtual domain over the server and configure mail users to map any specific domain.

Users can also interact with the software and may put the request of hosting their domain on that specific server from anywhere over the Internet. The software also provides troubleshooting facility for users in all common problems. The main feature of this software is that it is platform independent. It can be used on every platform that supports Java Virtual Machine (JVM). Java is used to provide platform independence. MySQL is used as backend database transactions, which is also a platform database engine.

Table of Contents

Chapter No.	Contents	Page No.
Chapter 1:	INTRODUCTION	1
	1.1 Introduction	1
	1.2 Existing System	2
	1.2.1 Apache Web Server Configuration	2
	1.2.2 DNS Server Configuration	4
	1.2.3 SendMail Configuration	7
	1.2.4 Creating System Users	9
	1.3 Proposed System	11
	1.4 Need and Purpose	12
	1.5 Features of the Product	13
Chapter 2:	TOOLS AND TECHNOLOGIES	15
	2.1 Java™ Servlets	15
	2.1.1 Advantages of Servlets over “Traditional” CGI	16
	2.1.2 Servlet Interface and Life Cycle	17
	2.1.2.1 Request and Response Objects	19
	2.1.2.2 Persistent and Shared Data	19
	2.1.2.3 Session Tracking	20
	2.1.2.4 ServletContext Attributes	21
	2.1.2.5 Request Attributes and Resources	21
	2.1.2.6 Multithreading	22
	2.2 Turbo C	22
	2.3 HTML	23
	2.3.1 HTML and SGML	24
	2.4 JavaScript	25
	2.5 MySQL	29
	2.5.1 The Main Features of MySQL	30
Chapter 3:	PROBLEM ANALYSIS	31
	3.1 Unified Modeling Language (UML)	31
	3.1.1 Primary Artifacts of UML	31
	3.2 Use Cases	32
	3.3 Use Case Diagram	34
	3.3.1 Package “Main Apache Configuration”	35
	3.3.2 Package “Global Apache Configuration”	36
	3.3.3 Package “Host New Domain”	37
	3.3.4 Package “User Requests”	38

Chapter No.	Contents	Page No.
Chapter 4:	SYSTEM DESIGN AND IMPLEMENTATION	39
	4.1 System Design	39
	4.1.1 Logical Design	39
	4.1.2 Physical Design	39
	4.2 Design Objectives	40
	4.2.1 Simplicity	40
	4.2.2 Verifiability	40
	4.2.3 Completeness	40
	4.3 Object-Oriented Design	40
	4.3.1 Steps of Design	41
	4.3.1.1 Defining an Architecture	41
	4.3.1.2 Plan Executable Release	41
	4.4 Identification of Classes	42
	4.5 Class Diagram for ISMC	44
	4.6 Object Modeling of Classes	45
	4.6.1 FileIOServlet	45
	4.6.2 SendMailConfigServlet	46
	4.6.3 UserRequests	47
	4.6.4 RegServlet	47
	4.6.5 AuthenticationServlet	48
	4.6.6 PassServlet	48
	4.6.7 ViewRequests	49
	4.7 Database Design	50
Chapter 5:	TESTING	51
	5.1 Why we do Testing	51
	5.2 Inside the Testing Process	51
	5.3 General Types of Errors	51
	5.4 Types of Testing	51
	5.4.1 Unit Testing	52
	5.4.2 Integration Testing	52
	5.4.3 System Testing	52
	5.4.4 Regression Testing	52
	5.5 ISMC Testing	52
Chapter 6:	USERS' MANUAL	54
	6.1 Features of the Software	54
	6.2 System Requirements	55
	6.3 How to use the Software	55
	6.3.1 Home Page	56
	6.3.2 Administrator Login	57
	6.3.3 Global Apache Configuration	58
	6.3.4 Global Apache Directives Configuration	59
	6.3.5 Main Apache Configuration	60

Chapter No.	Contents	Page No.
	6.3.6 Main Apache Directives Configuration	61
	6.3.7 Host New Domain	62
	6.3.8 Create Email Account	63
	6.3.9 View Domain Requests	64
	6.3.10 View Email Requests	65
	6.3.11 Sign In New User	66
	6.3.12 Forgot Password	67
	6.3.13 Change Password	68
	6.3.14 User Login	69
	6.3.15 Domain Hosting Request	70
	6.3.16 Email Accounts Creation Request	71
	6.3.17 View Domain Statistics	72
	6.3.18 Files and Folders	73
	6.3.19 Troubleshooting	74
	6.3.20 Uploading Problem	75
	6.3.21 Configuring Mail Utilities	76
	6.3.22 Frequently Asked Questions	77
	6.3.23 Terms and Conditions	78
	6.3.24 About us	79

BIBLIOGRAPHY

Chapter # 1

Introduction

Introduction

1.1 Introduction

Linux is a full-featured operating system with all the bells and whistles that are expected from a modern operating system. Linux is a UNIX clone and therefore support all the UNIX features. Linux is best known for it's features like portability among a wide range of hardware, security as it can protect sensitive data from prying eyes, interoperability in ability for different types of computers on a network to communicate with one another and such other features.

The Linux is an open source operating system. Its source code is freely available. Anyone can change the source code to make the program better suited to his or her own needs. Red Hat Corporation is a software company that publishes the Red Hat distribution for the Linux. A number of Linux flavors are easily available in the market, but the Red Hat Linux is most popular among them due to some additional features like simplified installation, Red Hat package manager, Sun Microsystems' StarOffice deluxe, Modern interfaces and Kudzu etc. A part of beauty of Linux is that it runs on a wide range of computers, from small inexpensive desktops on up to huge multi-processor servers. As a server Linux can be set up for use as a small intranet server or a server hosting multiple sites and hundreds of clients. All the software we need to set up is Web Server, DNS server, email server, or a news server.

Apache web server is the most popular web server in the world to serve the web, as more than 50% of the Web Servers in the world use Apache. The Apache web server is built in Red Hat Linux 6.2 and later.

Domain Name Servers contain the information that is necessary to translate back and forth between numeric network IP Addresses that we can see over the net. Symbolic addresses are comparatively easy to be remembered by the users as numeric IP Addresses. All these conversions and mapping is done using a DNS server. Red Hat Linux 6.2 has its own built in DNS server, which is very well known for its performance and stability. It is well known that virtual web sites are more economical rather than setting up dedicated physical web servers for each and every domain over the Internet, both for ISPs and users.

Instead of having dedicated server systems, the existence of virtual web sites is a very good entry for a professional Internet presence.

In order to look really professional, organizations and also individuals want their email addresses to reflect their virtual web domain name addresses. So this need of supporting virtual email servers is solved by the SendMail server, built in Red Hat Linux.

1.2 Existing System

The complete Linux system is based on traditional file system. The entire configuration is based on text files that contain their own pre-formatted syntax. Linux now comes up with a Graphical User Interface. But the text based configuration files are used for increased efficiency and performance.

1.2.1 Apache Web Server Configuration

The Apache web server does not come up with a graphical user interface for administrators. It provides a powerful file-based configuration. It comes up with three plain text files that we can use to configure Apache to our liking. These three configuration files include

access.conf - controls access to our web server's resources.

httpd.conf - the primary configuration file. Tells Apache how to run.

srm.conf -specifies which resources we want to offer to our site.

These three are the simple text based configuration files. In the existing system, the administrator has to configure the directives specified in these files. The administrator has to clearly remember the functionality and valid values for each and every directive and also the details of the configuration of these directives. There are a number of directives, the administrator has to set up for the server getting up and running. The administrator has to keep track of all these directives and set up all of them properly. The configuration of Apache web server is categorized in to two major categories. i-e- *Global Server Configuration* and *Virtual Host Context Configuration*.

The Global configuration affects the overall operation of Apache, such as the number of concurrent requests it can handle or where it can find its configuration files. Using global

Apache configuration directives, an administrator can control the behavior of the web server. These directives are global *Server Config Context* directives, means these directives may appear in the primary server configuration files anywhere outside any container. These directives are fundamental in nature and generally apply to both the primary server (*Server Config Context*) and the virtual servers (*Virtual Hosts Context*). Following is a list of common Apache global configuration directives.

- AccessConfig
- BindAddress
- KeepAlive
- KeepAliveTimeOut
- LockFile
- Listen
- MinSpareServers
- MaxSpareServers
- MaxClients
- MaxKeepAliveRequests
- MaxRequestsPerChild
- PIDFile
- ResourceConfig
- ScoreBoardFile
- ServerRoot
- ServerType
- StartServers
- TimeOut

Then there are directives that set up the values used by the main server, which responds to any requests that aren't handled by a **<VirtualHost>** definition. These values also provide defaults for any **<VirtualHost>** containers we may define later in the file. All of these directives may appear inside **<VirtualHost>** containers, in which case these default settings will be overridden for the virtual host being defined.

- AccessFileName
- CustomLog
- DefaultType
- DirectoryIndex
- DocumentRoot
- ErrorLog
- Group
- HeaderName
- HostNameLookUps
- LogLevel
- Port
- ReadmeName
- ServerAdmin
- ServerName
- ServerSignatures
- TypesConfig
- User
- UseCanonicalNames

The following directives are used to control Apache's resources, and are traditionally stored in the *srm.conf* file.

- DocumentRoot
- DirectoryIndex
- IndexIgnore

1.2.2 DNS Server Configuration

The DNS name server configuration is used to serve virtual sites over the web, rather than having a separate physical web server for each and every domain over the Internet. The DNS configuration in Linux based DNS Server is traditionally also based on configuration in text-based configuration files. The information stored in DNS Servers is more than just a

hostname and an IP address. There are several types of records that an administrator has to set up that are received by a remote computer from a DNS server. Here are some commonly used DNS records.

- **Address Records:** An Address record is the IP Address of a particular hostname. This record is used to relate a symbolic address to a numeric IP Address.
- **Name Server (NS):** The name server records contain name servers for a particular domain.
- **Source of Authority (SOA):** The SOA information returned for a hostname contains the authoritative nameserver for the host and all machines in the same subdomain as the name. Contact information for the domain is also returned.
- **Canonical Name (CNAME):** A CNAME record is an alias to an existing domain name. e-g- *www.yourdomain.com* may have an alias *ftp.yourdomain.com* to point to same server and domain.
- **Mail Exchanger (MX):** The MX records enable to have mail that is sent to a particular hostname automatically to a different server machine.

The configuration required to set up a new virtual host over a web server, assuming that we have a real time IP Domain address 192.168.1 and a real time registered domain name *www.mydomain.com* is: -

- First of all, the administrator has to create a zone for each particular domain in the primary DNS configuration file *named.conf*. The zone creation involves the following configuration in the *named.conf* file.

```
zone "1.168.199.in-addr.arpa" {  
    type master;  
    file "192.168.1";  
};
```

This zone is created for each IP domain Class, not for each domain. For each individual domain, the zone creation has the following configuration syntax.

```

zone mydomain.com IN {
    type master;
    file mydomain.com;
};

```

Note here that both the zone types contain a directive **file**. This directive refers to the *zone files*. The zone files are the files that contain the details about the domains, like TTL, Refresh time, Mail Exchange server for the domain etc. The structure of a zone file is discussed below.

- The next step to configure a virtual domain is to create zone files, one for each IP domain class and one for each virtual domain. A sample file contents for a IP domain class file look like

```

@      IN      SOA      localhost. root.localhost. (
        1997022700 ; Serial
        28800      ; Refresh
        14400      ; Retry
        3600000    ; Expire
        86400 )    ; Minimum
IN     NS      localhost.
@      IN      NS      mydomain.com
1      IN      PTR     192.168.1.1

```

The contents of a zone file for each domain are as follows

```

mydomain IN SOA mydomain.com. root.mydomain.com. (
        10018          ; serial
        43200         ; refresh
        3600          ; retry
        3600000       ; expire
        2592000       ; minimum
        )
IN     NS      ns.mydomain.com.

```

- When the zone and the zone file for a domain name is created, the administrator has to manually configure **<VirtualHost>** directive for each and every domain in the main apache server configuration. A sample configuration is as follows

```
<VirtualHost 192.168.1.1>
  ServerName www.mydomain.com
  ServerAdmin webmaster@mydomain.com
  DocumentRoot /www/virtual/www.mydomain.com
  ErrorLog logs/www.mydomain.com-error_log
  TransferLog logs/www.mydomain.com-transfer_log
</VirtualHost>
```

- Now when all the configuration is made, the IP Address allotted to a domain is needed to *BIND* to a physical Ethernet interface, through which the server is supposed to listen to the requests for that specific domain. This is done by manually issuing the following command (*for IP Address 192.168.1.1 and Ethernet interface eth0*) on the Linux command prompt.

```
ifconfig eth0:1 192.168.1.1
```

- The configuration for a virtual domain is complete. Now the administrator has to restart the Apache and DNS server daemons to successfully complete the configuration process and the changes to take place.

1.2.3 SendMail Configuration

The SendMail configuration is also traditionally based on the text-based configuration files. There are two different types of email servers. One is *SMTP* (responsible for making sure that hundreds of thousands of email messages that are sent daily make their way around the web and end end up in a specific mailbox), the second is *POP3* (used to

move email from the server that holds user's mailbox onto their computer). The major configuration files involved in setting up a SendMail server are: -

- **virtusertable**
- **virtusertable.db**
- **sendmail.cf**
- **aliases**
- **zone files for each domain and IP class.**

The process of creating a email user pointing to a specific domain name (e-g- *you@yourdomain.com*) is some thing more complex. Because it not only involves the configuration in the configuration files, but also creating a unique real time system user on the server, the domain is hosted.

Following are the steps involved in setting up a new virtual email user.

- First of all, the domain is configured to have email accounts reflecting its name. This is done by adding **Mail Exchanger (MX)** entry. This MX entry is added in the zone file for each domain as specified in the zone creation in DNS Server. In the zone file following text lines are added at the end of the zone file.

```
IN      MX      5      mailserver1.  
IN      MX      10     mailserver2.
```

The first MX entry points to the primary mail exchanger (*mailserver1 here*) and the second MX entry points to the secondary mail exchanger (*mailserver2 here*) in case the primary server is unavailable.

- When the MX entries are inserted in the zone files, after successfully restarting the DNS daemon, the specified domain is now configured to set up email accounts, but yet no email accounts are created. Now administrator has

to configure the email account reference in the *virtusertable* file. A sample *virtusertable* entry for a email account (*info@mydomain.com*) is like: -

```
info@mydomain.com      info123
```

- ❑ After *virtusertable* entry is added, the *virtusertable* file is mapped to *virtusertable.db* file by issuing the following command

```
makemap hash virtusertable < virtusertable.db
```

- ❑ The SendMail and DNS daemons are now restarted for changes to successfully take place.
- ❑ To divert the mails of a particular mailbox to any other mail account, the *aliases* file is configured. Let's suppose, we want to divert the mails of *info@mydomain.com* to another mail box *abc123@yahoo.com*, the following is the proper syntax for the *aliases* configuration: -

```
info123                abc123@yahoo.com
```

(note that *info123* is the real time system user, explained later)

1.2.4 Creating System User

To create an email account, a real time system user is needed to be created to receive the mails of that specific email account. In the above-mentioned entry, *info123* points out the real time system user that maps to the virtual email address and contains the emails for that account. Creation of a real time involves the configuration in the following files (assuming *info123* is the user name to be created).

- ***/etc/passwd***
- ***/etc/shadow***
- ***/etc/group***
- **Create a home directory for the user in the */home* directory.**

- First of all, a home directory for each system user is created with name same as user name (normally in the */home* directory) by issuing the following command on the Linux command prompt.

```
mkdir /home/info123
```

- The user name is registered to */etc/group* file to be allocated a unique User ID. The general syntax is as follows.

```
info123:x:502
```

where

<code>info123</code>	→	username
<code>x</code>	→	shows user's password is encrypted
<code>502</code>	→	unique user ID (usually greater than 500 for human users)

- When the home directory is created, the user name entry is added in the */etc/passwd* file in the following syntax.

```
info123:x:501:502:./home/info123:/bin/bash
```

where

<code>info123</code>	→	username
<code>x</code>	→	user's password is encrypted
<code>501</code>	→	specifies the Group ID of the user
<code>502</code>	→	specifies the unique User ID of the user
<code>/home/info123</code>	→	user's home directory
<code>/bin/bash</code>	→	default boot shell

- The */etc/passwd* file only maps to the user name of the system user. The password for each user name is resided in the */etc/shadow* file in the encrypted form, encrypted using MD5 algorithm. A sample password entry is like: -

```
info123:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:d1:d2:d3:d4:d5:
```

where

<code>info123</code>	→	username
<code>xxxxx</code>	→	34 characters encrypted password
<code>d1</code>	→	unique password ID
<code>d2</code>	→	days to keep this account alive
<code>d3</code>	→	password must change after this no of days
<code>d4</code>	→	warn this no of days before expiry of account
<code>d5</code>	→	account expires if not accessed this no of days

1.3 Proposed System

We can clearly examine that the configuration of Apache, DNS ad SendMail servers is not that simple and it needs a very good expertise of configuration of these servers. The proposed system provides a web-based Graphical User Interface for all these configuration for almost all the configuration directives of Apache web server, the process of virtual domain hosting over a web server and also the complete configuration process for configuring virtual email accounts to reflect their virtual domains.

The proposed system involves the web-based configuration from any remote terminal over the web. The administrator can configure the server by just surfing the site. In the server side, the software handles HTTP requests made by different users and performs the appropriate actions. If a request of setting up any Apache server directive is made, the software configures the configuration files as requested and restarts the Apache server daemon automatically.

Similarly if any request is made to host a new site, the software first of all checks for the domain name, if already hosted. If the domain name is not already hosted over this server, it checks whether the request for this domain name has already been placed. If the domain name is not found to be hosted, the domain name is validated. Next step is the validation of IP Address allotted to that specific virtual domain. First of all this IP Address is

confirmed not to be allotted to any other domain name over the web. If IP Address is valid, the Apache server and the DNS Server is configured to host that domain, including the zone file creation and the IP Address is BINDed to Ethernet interface.

The same automated process for the email configuration is adopted. First of all, the requested domain name is validated, whether or not that specific domain is hosted on the same server. If domain name is validated, the user ID is confirmed not to do exist already for that domain. If user ID is also validated, the SendMail server is configured and restarted automatically. For each email account, a separate system user is also created, following the mechanism that first of all the user name same as requested as email ID is searched. If the user with same name already exists in that server, a random user name starting with the requested email ID is selected and a system user is created in the */etc/group* and */etc/passwd* file and the user home directory is created. The requested password for the user is encrypted through MD5 algorithm to produce a 34 character long encrypted password string. This encrypted password is then configured in the */etc/shadow* file.

1.4 Need and Purpose

The configuration process using the traditional text files system is hard to do. It requires from the administrator to have a compact expertise and thorough knowledge for the configuration of each and every directive of each server. The need to develop such software is based on the following facts.

- The administrator has to keep track of all the directives and their possible valid values and must also know about which value has what effect on the server performance.
- Using the text files configuration system, the administrator is supposed to have a deep knowledge of each and every directive and the details of these directive that what directive does what.
- Even if administrator is well aware of all the configuration and has all these directives configuration on his fingertips, there is a big chance of making any error in typing the valid value for a directive.
- Normally the configuration files are so big and lengthy in size, that it is again a trouble for administrator to locate the directive and change its value.

- There are some directives, that changing the value of one directive requires changes in some other directive(s). This is also a difficulty for the administrator to keep track of these directives.
- Any invalid value may lead to failure of getting the server up.
- When the changes are made to the server configuration files, the server daemons are needed to be restarted manually.
- When there are a big number of virtual domains on the server, it is not easy to keep track of which domain is already not hosted and also which IP Address is free for use now.
- The use of duplicate IP Address may cause the existing domain not work properly. e-g- if *192.168.1.1* is already allotted to *mydomain.com* and we try to allocate this same IP Address to any other domain *abc123.com*, it will result in getting *mydomain.com*, down.
- There is also a problem in choosing a system user name for creating virtual email accounts. Duplicate user names may cause the server not work properly.

1.5 Features of the Product

In order to overcome all these problems, the proposed system provides a user-friendly interface for this configuration process. The software overcomes these problems by facilitating the users with the following features

- User-friendly Graphical User Interface (GUI)
- Remote administration is made possible for administrators
- Only valid values are input by the user for the configuration of each and every directive.
- A brief description is given against each directive explaining the purpose and functionality of each directive, valid values for that specific directive and the effect of each valid value of the performance and efficiency of the server.
- There is a less chance of making an error in the configuration. Only valid values are allowed and selection based choices are introduced before user, wherever possible.

- All the valid input user requests are again validated on the server side, so that the chance of making a mistake can be minimized. It is also effective in validating the domain names and IP Addresses for domain registration and hosting.
- System users for virtual email accounts are created on strict validation policy. Only valid system user name is assigned, password is encrypted using MD5 algorithm for more secure system and users are created with desired access rights.
- General users can host their new domain through web.
- Users can manage files and folders for their hosted domain through web.
- Users can create virtual email accounts reflecting their domain named from any remote site over the Internet.

Beyond all these features, it is the fact that the Internet is emerging more and more in times and has become a need of our daily routines. The web-based graphical user interface is the best solution for the configuration, as the administrator may get full control over its server from anywhere throughout the world. Also users can host their new domain, manage their domain, create email accounts reflecting their domains from anywhere over the web.

Chapter # 2

Tools and Technologies

Tools and Technologies

Selection of tools and applications has an important role in the software development. The selection of tools to be used for development of application depends upon the problem and the working environment of the application. The selection of best-suited tools is also important because of various facilities provided by different languages. It is to be examined and decided that which tool provides best solution for our problem domain and environment. In the development of *Internet Services Management Console*, following tools are used: -

- Sun Microsystems' JDK 1.1.3
- Turbo C
- HTML
- JavaScript
- MySQL

JAVA Servlets, being the power of web, have been the major implementation tool for the development of *Internet Services Management Console*.

2.1 Java™ Servlets

Java™ Servlet technology provides web developers with a simple, consistent mechanism for extending the functionality of a web server and for accessing existing business systems. A servlet can almost be thought of as an applet that runs on the server side without its own users' end interface. Java servlets have made many web applications possible. Servlets are the Java platform technology of choice for extending and enhancing web servers. Servlets provide a component-based, platform-independent method for building web-based applications, without the performance limitations of CGI programs. Unlike proprietary server extension mechanisms (such as the Netscape Server API or Apache modules), servlets are server and platform-independent. This leaves us free to select a "best of breed" strategy for our servers, platforms, and tools. Servlets have access to the entire family of Java APIs, including the JDBC™ API to access enterprise databases. Servlets can

also access a library of HTTP-specific calls and receive all the benefits of the mature Java language, including portability, performance, reusability, and crash protection. Today, servlets are a popular choice for building interactive web applications. Third-party servlet containers are available for Apache Web Server, iPlanet Web Server, Microsoft IIS. Servlet containers can also be integrated with web-enabled application servers, such as BEA WebLogic Application Server, IBM WebSphere, iPlanet Application Server, and others. JSP technology is an extension of the servlet technology created to support authoring of HTML and XML pages. It makes it easier to combine fixed or static template data with dynamic content.

Servlets are Java technology's answer to CGI programming. They are programs that run on a Web server and build Web pages. Building Web pages on the fly is useful (and commonly done) for a number of reasons: -

1. **The Web page is based on data submitted by the user.** For example the results pages from search engines are generated this way, and programs that process orders for the Electronic Commerce sites do this as well.
2. **The data changes frequently.** For example, a weather-report or news headlines page might build the page dynamically, perhaps returning a previously built page if it is still up to date.
3. **The Web page uses information from corporate databases or other such sources.** For example, we would use this for making a Web page at an on-line store that lists current prices and number of items in stock.

2.1.1 Advantage of Servlets Over "Traditional" CGI

Java servlets are more efficient, easier to use, more powerful, more portable, and cheaper than traditional CGI and than many alternative CGI-like technologies.

1. **Efficient.** With traditional CGI, a new process is started for each HTTP request. If the CGI program does a relatively fast operation, the overhead of starting the process can dominate the execution time. With servlets, the Java Virtual Machine (JVM) stays up, and each request is handled by a lightweight Java thread, not a heavyweight operating system process.

2. **Convenient.** Besides the convenience of being able to use a familiar language, servlets have an extensive infrastructure for automatically parsing and decoding HTML form data, reading and setting HTTP headers, handling cookies, tracking sessions, and many other such utilities.
3. **Powerful.** Java servlets let us easily do several things that are difficult or impossible with regular CGI. For one thing, servlets can talk directly to the Web server (regular CGI programs can't). This simplifies operations that need to look up images and other data stored in standard places. Servlets can also share data among each other, making useful things like database connection pools easy to implement. They can also maintain information from request to request, simplifying things like session tracking and caching of previous computations.
4. **Portable.** Servlets are written in Java and follow a well-standardized API. Consequently, servlets written for, let say I-Planet Enterprise Server can run virtually unchanged on Apache, Microsoft IIS, or WebStar. Servlets are supported directly or via a plug in on almost every major Web server.
5. **Inexpensive.** There are a number of free or very inexpensive Web servers available that are good for "personal" use or low-volume Web sites. However, with the major exception of Apache, which is free, most commercial-quality Web servers are relatively expensive. Nevertheless, once we have a Web server, no matter the cost of that server, adding servlet support to it (if it doesn't come pre-configured to support servlets) is generally free or cheap.

2.1.2 Servlet Interface and Life Cycle

A servlet is a Java class that implements the Servlet interface. This interface has three methods that define the servlet's life cycle: -

1. **public void init (ServletConfig config) throws ServletException** This method is called once when the servlet is loaded into the servlet engine, before the servlet is asked to process its first request.
2. **public void service (ServletRequest request, ServletResponse response) throws ServletException, IOException.** This method is called to process a request. It can be called zero,

one or many times until the servlet is unloaded. Multiple threads (one per request) can execute this method in parallel so it must be thread safe.

3. **public void destroy()** This method is called once just before the servlet is unloaded and taken out of service. The `init()` method has a `ServletConfig` attribute. The servlet can read its initialization arguments through the `ServletConfig` object, how the initialization arguments are set is servlet engine dependent but they are usually defined in a configuration file. A typical example of an initialization argument is a database identifier. A servlet can read this argument from the `ServletConfig` at initialization and then use it later to open a connection to the database during processing of a request.

```
private String databaseURL;
public void init(ServletConfig config) throws
ServletException
{
    super.init(config);
    databaseURL=config.getInitParameter(database);
}
```

The Servlet API is structured to make servlets that use a different protocol than HTTP possible. The `javax.servlet` package contains interfaces and classes intended to be protocol independent and the `javax.servlet.http` package contains HTTP specific interfaces and classes. Since this is just an introduction to servlets, this distinction has been ignored here and HTTP servlets are basically focused. Class `HttpServlet` is part of the JSDK and implements the `Servlet` interface plus a number of convenience methods. We define our class like this: -

```
import javax.servlet.*;
import javax.servlet.http.*;
public class ReqInfoServlet extends HttpServlet
{
    ...
    ...
    ...
}
```


An important set of methods in `HttpServlet` is the ones that specialize the `service()` method in the `Servlet` interface. The implementation of `service()` in `HttpServlet` looks at the type of request it's asked to handle (GET, POST, HEAD, etc.) and calls a specific method for each type. This way the servlet developer is relieved from handling the details about obscure requests like HEAD, TRACE and OPTIONS and can focus on taking care of the more common request types, i.e. GET and POST.

2.1.2.1 Request and Response Objects

The `doGet()` or `doPost()` method has two interesting parameters: `HttpServletRequest` and `HttpServletResponse`. These two objects give us full access to all information about the request and let us control the output sent to the client as the response to the request. With CGI we read *environment variables* and *stdin* to get information about the request, but the names of the environment variables may vary between implementations and some are not provided by all Web servers. The `HttpServletRequest` object provides the same information as the CGI environment variables, plus more, in a standardized way. It also provides methods for extracting HTTP parameters from the query string or the request body depending on the type of request (GET or POST). As a servlet developer we access parameters in the same way for both types of requests. Other methods give us access to all request headers and help us parse date and cookie headers. Instead of writing the response to *stdout*, as we do with CGI, we get an `OutputStream` or a `PrintWriter` from the `HttpServletResponse`. The `OutputStream` is intended for binary data, such as a GIF or JPEG image, and the `PrintWriter` for text output. We can also set all response headers and the status code, without having to rely on special Web server CGI configurations such as Non Parsed Headers (NPH). This makes our servlet easier to install.

2.1.2.2 Persistent and Shared Data

One of the more interesting features of the Servlet API is the support for persistent data. Since a servlet stays loaded between requests, and all servlets are loaded in the same process, it's easy to remember information from one request to another and to let different servlets share data. The Servlet API contains a number of mechanisms to support this

directly. We'll look at some of them in detail below. Another powerful mechanism is to use a singleton object to handle shared resources.

2.1.2.3 Session Tracking

The `HttpSession` class was introduced in the 2.0 version of the Servlet API. Instances of this class can hold information for one user session between requests. We start a new session by requesting an `HttpSession` object from the `HttpServletRequest` in our `doGet()` or `doPost()` method as follows: -

```
HttpSession session = request.getSession(true);
```

This method takes a boolean argument. `true` means a new session shall be started if none exist, while `false` only returns an existing session. The `HttpSession` object is unique for one user session. The Servlet API supports two ways to associate multiple requests with a session: *cookies* and *URL rewriting*. If cookies are used, a cookie with a unique session ID is sent to the client when the session is established. The client then includes the cookie in all subsequent requests so the servlet engine can figure out which session the request is associated with. URL rewriting is intended for clients that don't support cookies or when the user has disabled cookies. With URL rewriting the session ID is encoded in the URLs our servlet sends to the client. When the user clicks on an encoded URL, the session ID is sent to the server where it can be extracted and the request associated with the correct session as above. To use URL rewriting we must make sure all URLs that we send to the client are encoded with the `encodeURL()` or `encodeRedirectURL()` methods in `HttpServletResponse`. An `HttpSession` can store any type of object. A typical example is a database connection allowing multiple requests to be part of the same database transaction, or information about purchased products in a shopping cart application so the user can add items to the cart while browsing through the site. To save an object in an `HttpSession` we use the `putValue()` method as: -

```
Connection conn;  
conn = driver.getConnection(databaseURL, user, password);  
session.putValue("myappl.connection", conn);
```

In another servlet, or the same servlet processing another request, we can get the object with the `getValue()` method:

```
HttpSession session = request.getSession(true);
Connection con
con =(Connection) session.getValue("myappl.connection");
    if (con != null) {
        // Continue the database transaction
```

We can explicitly terminate (invalidate) a session with the `invalidate()` method or let it be timed-out by the servlet engine. The session times out if no request associated with the session is received within a specified interval. Most servlet engines allow us to specify the length of the interval through a configuration option. In the 2.1 version of the Servlet API there's also a `setMaxInactiveInterval()`. So we can adjust the interval to meet the needs of each individual application.

2.1.2.4 ServletContext Attributes

All servlets belong to one servlet context. In implementations of the 1.0 and 2.0 versions of the Servlet API all servlets on one host belongs to the same context, but with the 2.1 version of the API the context becomes more powerful and can be seen as the humble beginnings of an Application concept. Future versions of the API will make this even more pronounced. Many servlet engines implementing the Servlet 2.1 API let us group a set of servlets into one context and support more than one context on the same host. The `ServletContext` in the 2.1 API is responsible for the state of its servlets and knows about resources and attributes available to the servlets in the context.

2.1.2.5 Request Attributes and Resources

The 2.1 version of the Servlet API adds two more mechanisms for sharing data between servlets: request attributes and resources. The `getAttribute()`, `getAttributeNames()` and `setAttribute()` methods where added to the `HttpServletRequest` class (or to be picky, to the `ServletRequest` superclass). They are primarily intended to be used in concert with the `RequestDispatcher`, an

object that can be used to forward a request from one servlet to another and to include the output from one servlet in the output from the main servlet. The `getResource()` and `getResourceAsStream()` in the `ServletContext` class gives us access to external resources, such as an application configuration file. The `ServletContext` methods, however, can provide access to resources that are not necessarily files. A resource can be stored in a database, available through an LDAP server, anything the servlet engine vendor decides to support. The servlet engine provides a context configuration option where we specify the root for the resource base, be it a directory path, an HTTP URL, a JDBC URL, etc.

2.1.2.6 Multithreading

It is another very good feature of Servlet API that the concurrent requests for a servlet are handled by separate threads executing the corresponding request processing method (e.g. `doGet()` or `doPost()`). It's therefore important that these methods are thread safe. The easiest way to guarantee that the code is thread safe is to avoid instance variables altogether and instead pass all information needed by a method as arguments.

2.2 Turbo C

In today's world of computer programming, there are many high-level languages to choose from, such as C, Pascal, and Java etc. These are all excellent languages suited for most programming tasks. Even so, there are several reasons why many computer professionals feel that C is at the top of the list:

- C is a powerful and flexible language. What we can accomplish with C is limited only by our imagination. The language itself places no constraints on us. C is used for projects as diverse as operating systems, word processors, graphics, spreadsheets, and even compilers for other languages.
- C is a popular language preferred by professional programmers. As a result, a wide variety of C compilers and helpful accessories are available.
- C is a portable language. *Portable* means that a C program written for one computer system (an IBM PC, for example) can be compiled and run on another system (a

DEC VAX system, perhaps) with little or no modification. Portability is enhanced by the ANSI standard for C, the set of rules for C compilers.

- C is a language of few words, containing only a handful of terms, called *keywords*, which serve as the base on which the language's functionality is built. We might think that a language with more keywords (sometimes called *reserved words*) would be more powerful. This isn't true. As we program with C, we find that it can be programmed to do any task.
- C is modular. C code can (and should) be written in routines called *functions*. These functions can be reused in other applications or programs. By passing pieces of information to the functions, you can create useful, reusable code.

As these features show, C is an excellent choice for any task from the simplest to very complex problems. In most of the cases when any hardware level access is required for any task, the best choice for us is programming C.

2.3 HTML

HyperText Markup Language (HTML) is used for creating hypertext on the Web. Conceived as a semantic markup language to mark the logical structure of a document, HTML gives users a way to identify the structural parts of a document. Learning HTML involves finding out what tags are used to mark the parts of a document and how these tags are used in creating an HTML document.

HTML is a system for making up documents with informational tags that indicate how text in the documents should be presented and how the documents are linked together. Hypertext Links are quite powerful. Within the HTML markup scheme lies the power to create interactive, cross-platform, multimedia, and client-server applications. The string of adjectives is not just hype; such schemes do exist. One, called the World Wide Web (WWW), lives on the Internet, providing organization to a wide variety of resources on computers located around the globe. The web also known as WWW or W3, plays a large part in the continuing development of HTML and the web will play a large part in the way we write and structure HTML documents. The World Wide Web represents the largest possible audience for our work.

HTML is not a programming language and an HTML document is not a computer program. It's a lot simpler than that. A computer program is a series of procedures and instructions applied, typically, to external data. An HTML document, however, is the data. The HTML language specifies the grammar and syntax of markup tags that, when inserted into the data, tell browsers (Computer programs that read HTML documents such as Microsoft Internet Explorer, Netscape Navigator etc) how to represent the document. Technically, HTML is defined as a Standard Generalized Markup Language (SGML), Document Type Definition (DTD). An HTML document is said to be an instance of a SGML document.

DHTML is a new and emerging technology that evolved to meet the increasing demand for eye-catching and mind-catching web sites.

DHTML combines HTML with Cascading Style Sheets (CSSs) and scripting languages. HTML specifies a web page's elements like table, frame, paragraph, bulleted list etc. Cascading style sheets can be used to determine an element's size, color, position and a number of other features. Scripting Languages (JavaScript and VBScript) can be used to manipulate the web page's elements so that styles assigned to them can change in response to user's input and demand.

HTML was not originally intended to be a page-layout language; instead, it was to be a language used to mark the structural parts of document-parts such as paragraphs, lists, headings, block quotations, and others. Based on the identification of these document parts, the programs that render HTML documents (Web browsers) display the HTML in a readable form. This organization allows for a separation of a document's structural specification in the HTML code from its formatted appearance in an HTML browser. In practice, there now are many language constructs we can use in HTML to control the appearance of a document.

2.3.1 HTML and SGML

The separation of document specification from document formatting relates to HTML's relationship to SGML. HTML is defined using SGML—an international standard (*ISO 8879:1986, Information Processing-Text and Office Systems (SGML)*) for text information processing. SGML itself is a *metalanguage* (a language to define languages). The goal of SGML is to help format information on-line for efficient electronic distribution,

search, and retrieval in a way that is independent of the appearance details of the document. A document marked according to SGML has no indications of the representation of a document. Only when a presentation program merges the SGML document with style information is the physical layout and appearance of a document apparent.

The data of a document consists of the contents of a document, whether it is text or multimedia, and any information about the information itself (such as administrative or technical information about the document that would not be rendered in its final form). The tags in an SGML document identify the structure: the headings, subheadings, paragraphs, lists, and other components. Finally, the format of a document is its final appearance, after the merging of data, structure, and specifications for how the formatting should be done. Note how all these parts are separable; document data can be created without the author worrying about the structure, structure can be added without worrying about its formatting, and formatting specifications can be created to follow a "house style" or particularities for an organization. And, because all these parts are independent, if the house style of an organization changes, the developers just need to change the specification for the style information instead of all the data or structure of the documents.

The reality of HTML doesn't live up to the ideal of an open system of information dissemination and display. The proliferation of proprietary HTML elements that only certain brands of browsers recognize combined with the wide variety of ways that different browsers on different computer platforms render text and graphics, has made a reliance on browser-independent rendering of content a pragmatic impossibility. Even following only "strict HTML" (without proprietary extensions), developers usually find that incompatibilities in graphics or an unsatisfactory rendering of certain effects occurs.

2.4 JavaScript

JavaScript is a compact, object-based scripting language for developing client and server Internet applications. Netscape Navigator interprets JavaScript statements embedded in an HTML page, and LiveWire enables you to create server-based applications similar to Common Gateway Interface (CGI) programs.

JavaScript is a lightweight interpreted programming language with rudimentary object-oriented capabilities. The general-purpose core of the language has been embedded in

Netscape Navigator and other web browsers and embellished for web programming with the addition of objects that represent the web browser window and its contents. This "client-side" version of JavaScript allows "executable content" to be included in web pages--it means that a web page need no longer be static HTML, but can include dynamic programs that interact with the user, control the browser, and dynamically create HTML content.

Syntactically, the core JavaScript language resembles C, C++ and Java, with programming constructs such as the `if` statement, the `while` loop, and the `&&` operator. The similarity ends with this syntactic resemblance, however. JavaScript is an untyped language, which means that variables do not have to have a type specified. Objects in JavaScript are more like Perl's associative array than they are like structures in C or objects in C++ or Java. Also, JavaScript is a purely interpreted language, unlike C and C++, which are compiled, and unlike Java, which is compiled to byte-code before being interpreted.

One of the most common misconceptions about JavaScript is that it is a "simplified version" of Java, the programming language from Sun Microsystems. Other than an incomplete syntactic resemblance and the fact that both Java and JavaScript can deliver "executable content" over networks, the two languages are entirely unrelated. The similarity of names is purely a marketing ploy (the language was originally called LiveScript, and its name was changed to JavaScript at the last minute).

JavaScript and Java do, however, make a good team. The two languages have disjoint sets of capabilities. JavaScript can control browser behavior and content but cannot draw graphics or perform networking. Java has no control over the browser as a whole, but can do graphics, networking, and multithreading. In Navigator version 3.0, JavaScript can communicate with the Java interpreter built into the browser and can work with and control any Java applets in a web page. This means that in this version of Navigator, JavaScript really can "script" Java

JavaScript is a relatively general-purpose programming language, and, as such, we can write programs in it to perform arbitrary computations. We can write simple scripts. In the context of the Web and web browsers, however, these aren't particularly interesting applications of the language. The real power of JavaScript lies in the browser and document-based objects that the language supports. To give an idea of JavaScript's potential, the

following subsections list and explain the important capabilities of JavaScript and of the objects it supports.

- **Control Document Appearance and Content**

The JavaScript Document object, through its `write()` method, allows us to write arbitrary HTML into a document as the document is being parsed by the browser. For example, this allows us to always include today's date in a document, or to display different text on different platforms, or even perhaps extra text to appear only on those browsers that support JavaScript.

- **Control the Browser**

Several JavaScript objects allow control over the behavior of the browser. The Window object supports methods to pop up dialog boxes to display simple messages to the user and to get simple input from the user. This object also defines a method to create and open (and close) entirely new browser windows, which can have any specified size and can have any combination of user controls. This allows us, for example, to open up multiple windows to give the user multiple views of your web site. New browser windows are also useful for temporary display of generated HTML, and, when created without the menu bar and other user controls, can serve as dialog boxes for more complex messages or user input.

- **Interact with Document Content**

The JavaScript Document objects, and the objects it contains, allow programs to read, and sometimes interact with, portions of the document. It is not possible to read the actual text itself (although this will probably be possible in a future release of JavaScript) but, for example, it is possible to obtain a list of all hypertext links in a document. It is even possible to use JavaScript to obtain an array of all images and Java applets embedded in a document. While HTML forms have traditionally be used only with CGI scripts, JavaScript is much more practical in some circumstances. Calculator programs like those described above are easy to implement with JavaScript, but would be impractical with CGI, because the server would have to be contacted to perform a computation every time the user entered a value or clicked on a button.

- **Interact with the User**

An important feature of JavaScript is the ability to define "event handlers"--arbitrary pieces of code to be executed when a particular event occurs. Usually, the user initiates these events, when (for example) a user moves the mouse over a hypertext link or enters a value in a form or clicks the Submit button in a form. This event-handling capability is a crucial one, because programming with graphical interfaces, such as HTML forms, inherently requires an event-driven model. JavaScript can trigger any kind of action in response to user events.

- **Read and Write Client State with Cookies**

"Cookies" are Netscape's term for small amounts of state data stored permanently or temporarily by the client. They are transmitted back and forth to and from the server and allow a web page or web site to "remember" things about the client--for example, that the user has previously visited the site, or that they have already registered and obtained a password, or that they've expressed preferences about colors or layouts of web pages. Cookies help you provide the state information that is missing from the stateless HTTP protocol of the Web.

- **Interact with Applets**

In Navigator 3.0 and later and all other web browsers, JavaScript can interact with Java applets that are running in the browser. This important feature is part of Netscape's "LiveConnect", a communication layer that allows Java applets, Netscape plug-ins, and JavaScript code talk with and control one another. Using LiveConnect, JavaScript code can read and write properties of, and invoke methods of, Java applets and plug-ins. This capability is tremendously powerful, and truly allows JavaScript to "script" Java.

- **Manipulate Embedded Images**

JavaScript can change the images displayed by an tag. This allows sophisticated effects, such as having an image change when the mouse passes over it or when the user clicks on a button elsewhere in the browser.

2.5 MySQL

MySQL, the most popular Open Source SQL database, is provided by MySQL AB. MySQL AB is a commercial company that builds its business providing services around the MySQL database.

- **MySQL** is a database management system.

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, we need a database management system such as MySQL. Since computers are very good at handling large amounts of data, database management plays a central role in computing, as stand-alone utilities, or as parts of other applications. Also MySQL possesses all the features and capabilities of a relational database management system. A relational database stores data in separate tables rather than putting all the data in one big storeroom. This adds speed and flexibility. Defined relations making it possible to combine data from several tables on request link the tables. The SQL part of MySQL stands for "Structured Query Language" - the most common standardized language used to access databases.

- **MySQL** is Open Source Software.

Open source means that it is possible for anyone to use and modify. Anybody can download MySQL from the Internet and use it without paying anything. Anybody so inclined can study the source code and change it to fit their needs.

- Why use **MySQL**?

MySQL is very fast, reliable, and easy to use. If that is what we are looking for, we should give it a try. MySQL also has a very practical set of features developed in very close cooperation with our users. MySQL was originally developed to handle very large databases much faster than existing solutions and has been successfully used in highly demanding production environments for several years. Though under constant development, MySQL today offers a rich and very useful set of functions. The connectivity, speed, and security make MySQL highly suited for accessing databases on the Internet.

2.5.1 The main features of MySQL

- Platform independent. Works on many different platforms.
- Fully multi-threaded using kernel threads. That means it can easily use multiple CPUs if available
- Fixed-length and variable-length records.
- In-memory hash tables which are used as temporary tables. Handles large databases. **MySQL** has support of databases that contain more than 5,000,000,000 records and more than 60,000 tables.
- All columns have default values. We can use `INSERT` to insert a subset of a table's columns; those columns that are not explicitly given values are set to their default values.
- Written in C and C++. Tested with a broad range of different compilers.
- A very fast thread-based memory allocation system.
- No memory leaks. Tested with a commercial memory leakage detector ().
- We can mix tables from different databases in the same query (as of Version 3.22).

Chapter # 3

Problem Analysis

Problem Analysis

Use of Object-Oriented Technique for problem analysis gives the objects' types (classes) and functionality of the objects (functions and procedures)

3.1 Unified Modeling Language (UML)

The Unified Modeling Language (UML) is a language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of the best engineering practices that have proven successful in the modeling of large and complex systems.

3.1.1 Primary Artifacts of the UML

What are the primary artifacts of the UML? This can be answered from two different perspectives: the UML definition itself and how it is used to produce project artifacts.

- **UML-defining Artifacts**

To aid the understanding of the artifacts that constitutes the Unified Modeling Language itself, this document consists of chapters describing UML Semantics, UML Notation Guide, and UML Standard Profiles.

- **Development Project Artifacts**

The choice of what models and diagrams one creates has a profound influence upon how a problem is attacked and how a corresponding solution is shaped. Abstraction, the focus on relevant details while ignoring others, is a key to learning and communicating. Because of this:

- Every complex system is best approached through a small set of nearly independent views of a model. No single view is sufficient.
- Every model may be expressed at different levels of fidelity.
- The best models are connected to reality.

In terms of the views of a model, the UML defines the following graphical diagrams:

- Use Case Diagram
- Class Diagram
- Behavior Diagrams
- Statechart Diagram
- Activity Diagram
- Interaction Diagrams
- Sequence Diagram
- Collaboration Diagram
- Implementation Diagrams
- Component Diagram
- Deployment Diagram

Although other names are sometimes given to these diagrams, this list constitutes the Canonical diagram names. These diagrams provide multiple perspectives of the system under analysis or development. The underlying model integrates these perspectives so that a self-consistent system can be analyzed and built. These diagrams, along with supporting documentation, are the primary artifacts that a modeler sees, although the UML and supporting tools will provide for a number of derivative views.

3.2 Use Cases

The use case construct is used to define the behavior of a system or other semantic entity without revealing the entity's internal structure. Each use case specifies a sequence of actions, including variants, which the entity can perform interacting with actors of the entity. In the metamodel Use Case is a subclass of Classifier, specifying the sequences of actions performed by an instance of the Use Case. The actions include changes of the state and communications with the environment of the Use Case. The sequences can be described using many different techniques, like Operation and Methods, Activity Graphs, and State Machines. There may be Associations between Use Cases and the Actors of the Use Cases. Such an Association states that an instance of the Use Case and a user playing one of the roles of the Actor communicate. Use Cases may be related to other Use Cases by Extend.

Include, and Generalization relationships. An Include relationship means that a Use Case includes the behavior described in another Use Case, while an Extend relationship implies that a Use Case may extend the behavior described in another Use Case, ruled by a condition. Generalization between Use Cases means that the child is a more specific form of the parent. The child inherits all Features and Associations of the parent, and may add new Features and Associations. The realization of a Use Case may be specified by a set of Collaborations (i.e., the Collaborations define how Instances in the system interact to perform the sequences of the Use Case).

In the UML standards, the Use Cases package is a sub package of the Behavioral Elements package. It specifies the concepts used for definition of the functionality of an entity like a system. The package uses constructs defined in the Foundation package of UML as well as in the Common Behavior package. The elements in the Use Cases package are primarily used to define the behavior of an entity, like a system or a subsystem, without specifying its internal structure. The key elements in this package are Use Case and Actor. Instances of use cases and instances of actors interact when the services of the entity are used. How a use case is realized in terms of cooperating objects, defined by classes inside the entity, can be specified with a Collaboration. A use case of an entity may be refined to a set of use cases of the elements contained in the entity. How these subordinate use cases interact can also be expressed in a Collaboration. The specification of the functionality of the system itself is usually expressed in a separate use-case model (i.e., a Model stereotyped «useCaseModel»). The use cases and actors in the use-case model are equivalent to those of the top-level package.

The use case diagrams representing the System and Actors' interaction of this project according to UML standards are given in the following pages.

3.3 Use Case Diagram

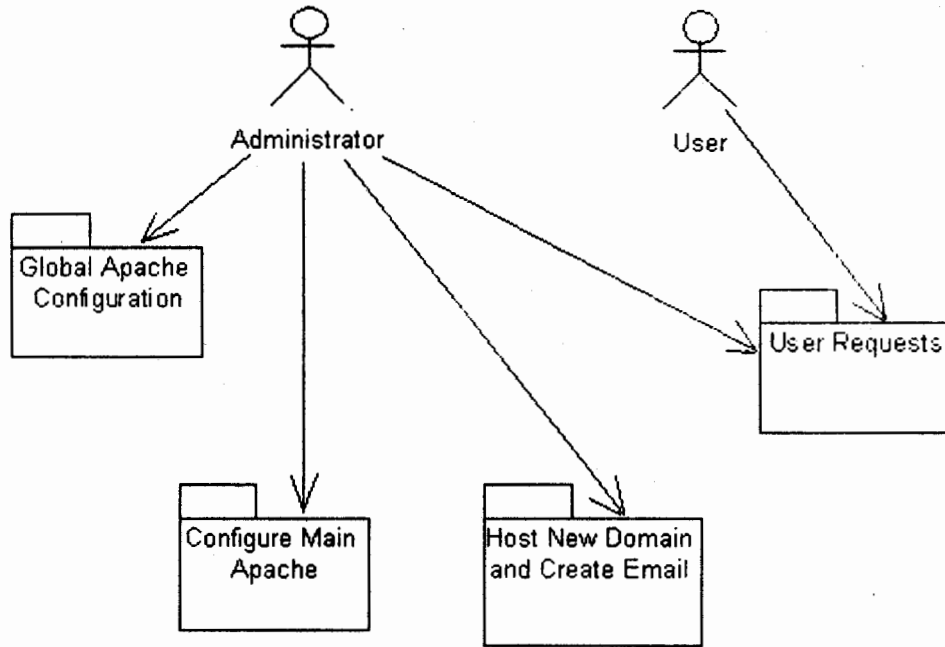


Fig 3.1

3.3.1 Package 'Main Apache Configuration'

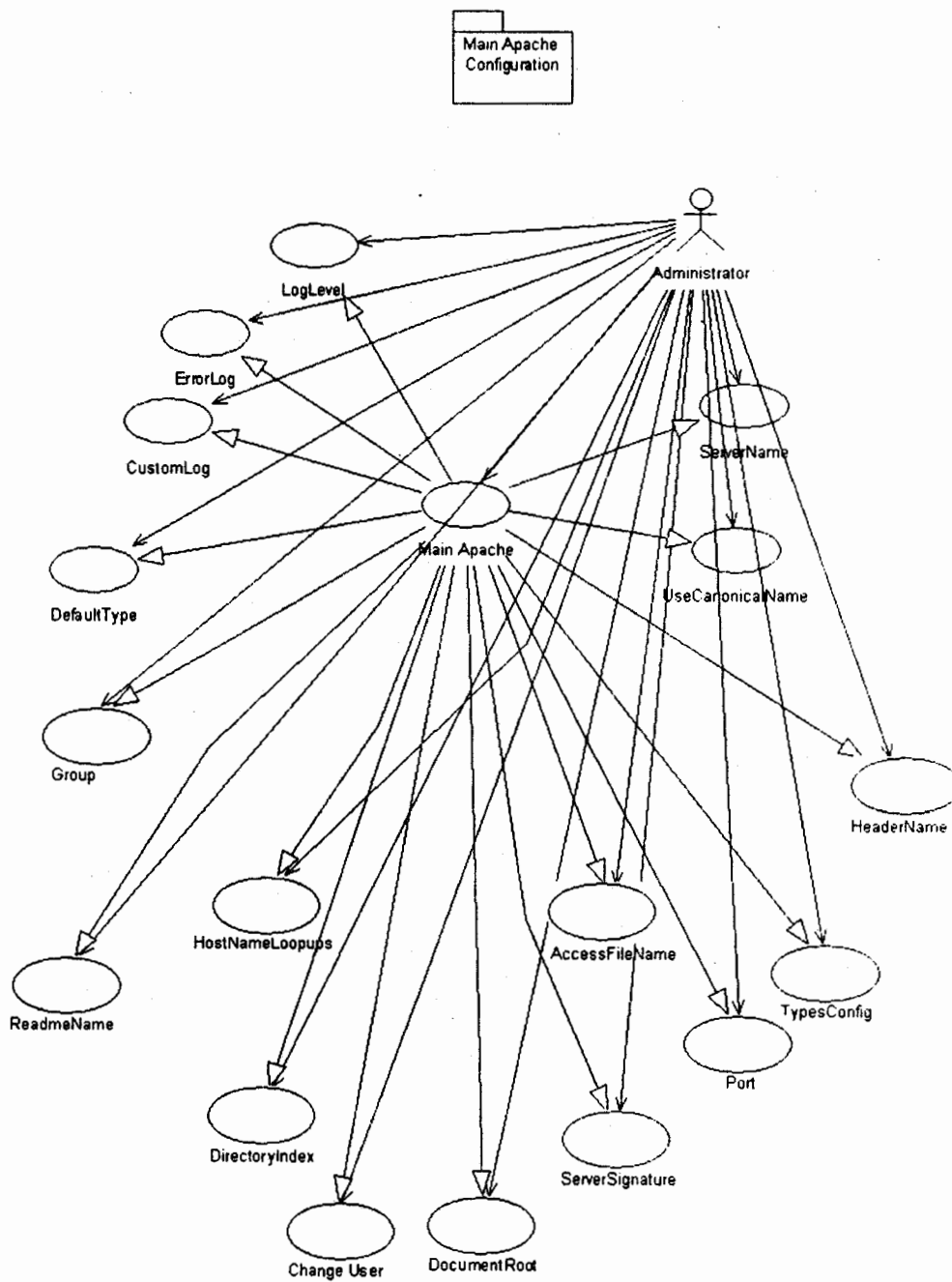


Fig 3.2

3.3.2 Package 'Global Apache Configuration'

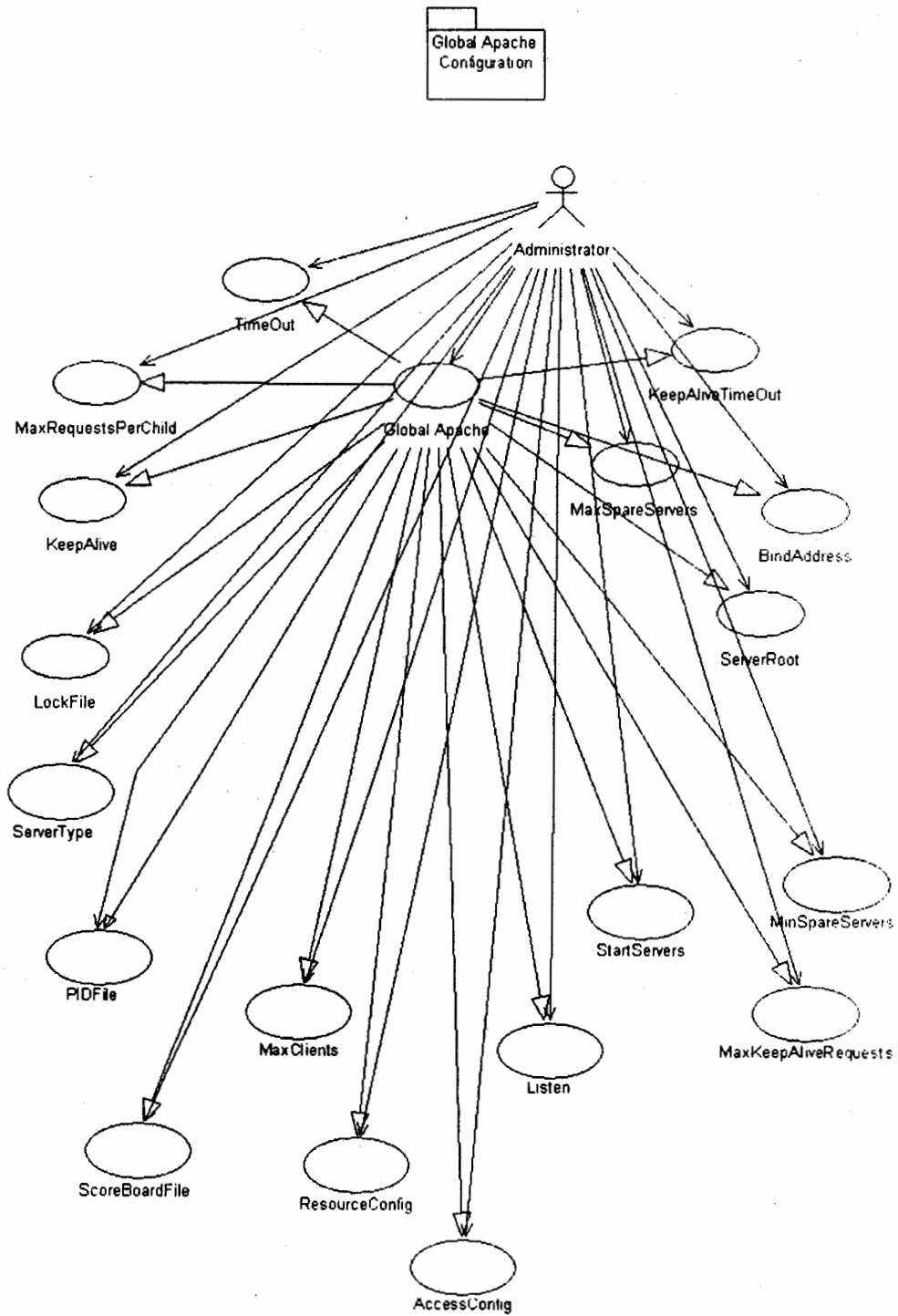


Fig 3.3

3.3.3 Package 'Host New Domain and Create Email Accounts'

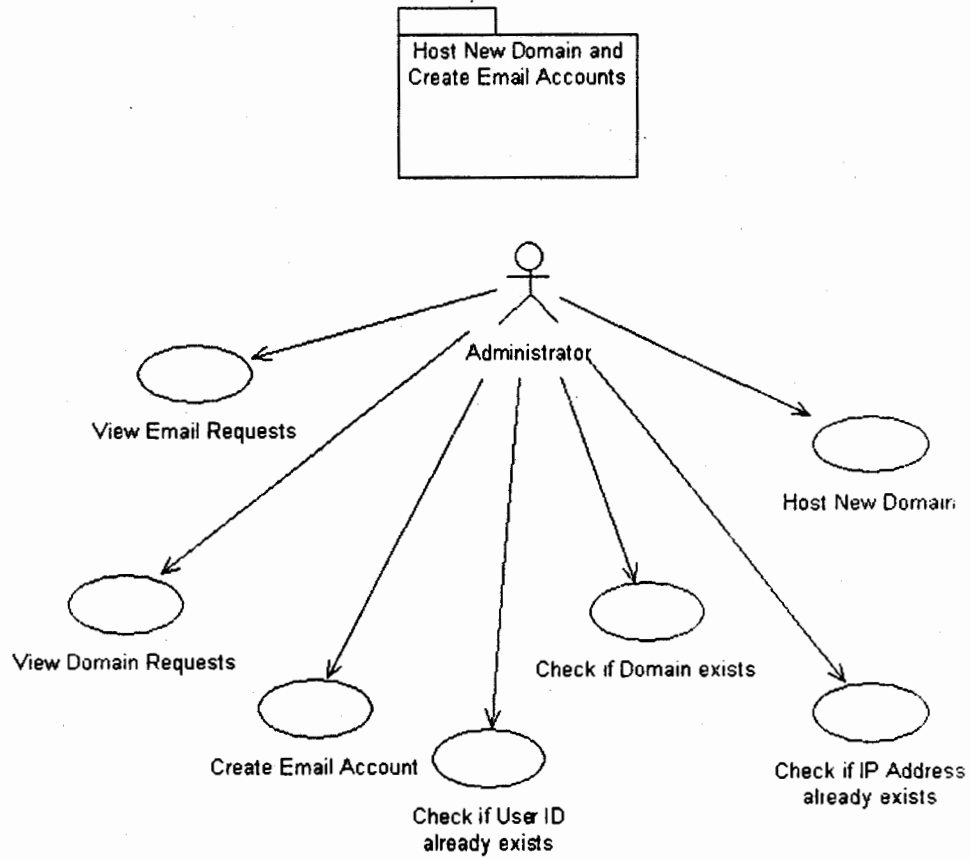


Fig 3.4

3.3.4 Package 'User Requests'

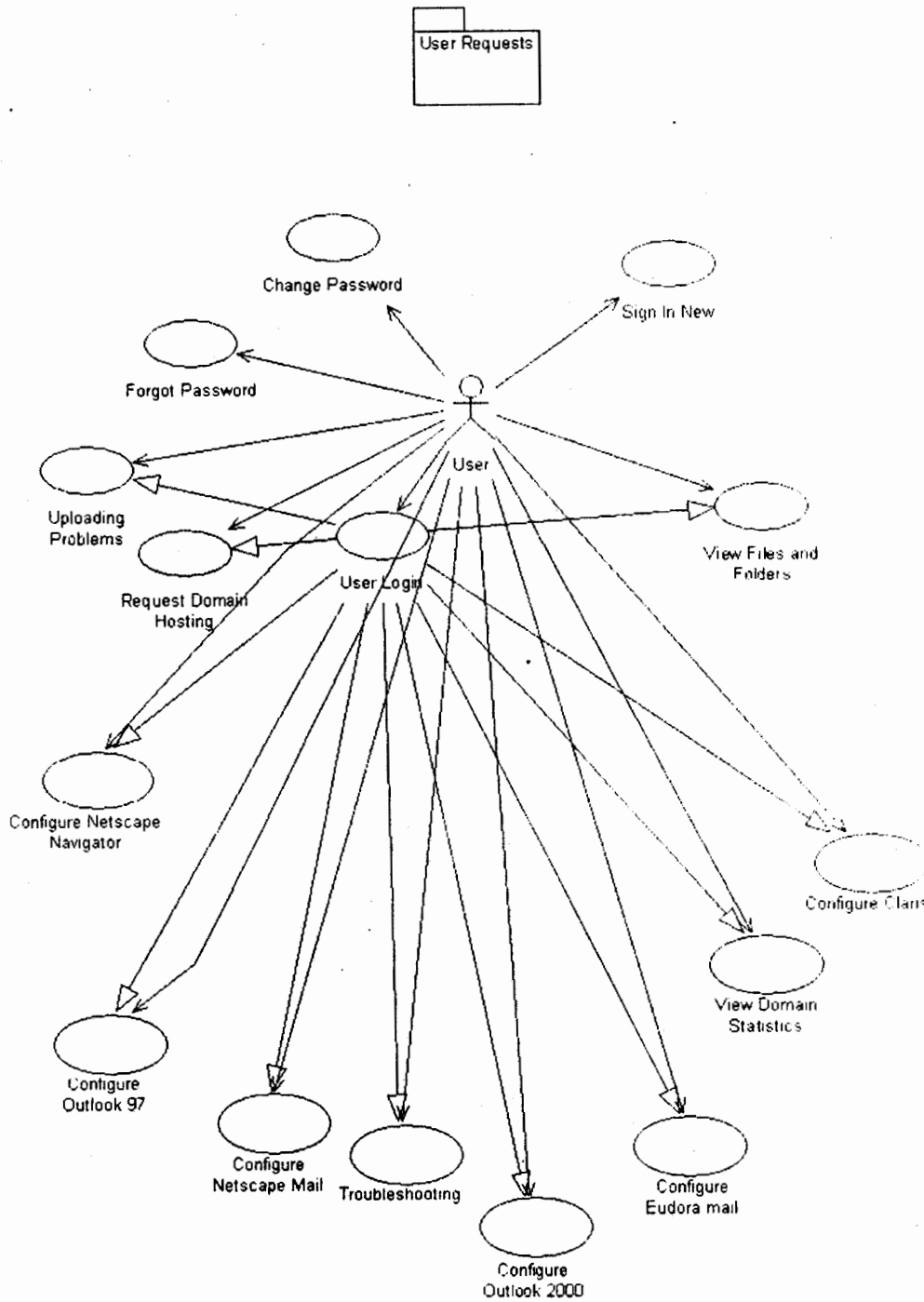


Fig 3.5

Chapter # 4

**System Design
and
Implementation**

System Design and Implementation

4.1 System Design

The design phase begins with the availability of the requirement specification documentation. While the requirement analysis is in the problem domain, the design phase is the first step towards moving from the problem domain to the solution domain.

In the system design phase, all the processes of the proposed system are discussed. This means to identify that what are the different processes, how they will be implemented and how they will interact with each other.

The database structure is also defined in the system design phase. The primary key for each table is mentioned and the relations between the tables are defined. The design of a system built precisely according to it completely satisfies the requirements specified in the requirement specification documentation.

System design is a solution to a problem. Design demands the translation of the requirements gathered during the analysis. The system design is normally based on the following two levels: -

4.1.1 Logical Design

Logical design produces a specification of the new system, which meets the system objectives. The outcome of the logical design is a blueprint of the system. Following are the components of logical design.

- Outputs (query results, displays etc.)
- Inputs (forms etc.)
- Code designing
- Procedures (structure of procedures to collect, transform and output data etc.)

4.1.2 Physical Design

It takes the logical design's blueprint and assembles the program specifications i.e. file, database and user's interface for the selected target hardware and software. These

programs and files fulfill the logical design requirements with some constraints and compromise.

4.2 Design Objectives

An ideal design has the following objectives.

4.2.1 Simplicity

The system design should be simple to understand. This helps to simplify the task of development of the system. Also the maintenance will be easy and the maintenance cost of the system can be reduced.

4.2.2 Verifiability

The verifiability of the design means that the correctness of the design can easily be proven to be according to requirement specifications.

4.2.3 Completeness

The system design is said to be complete if all the components, modules and detailed aspects of the design are specified. All the modules, interfaces and data relationship should be specified.

4.3 Object-Oriented Design

System design is the process of expanding what was learnt during domain analysis into a working implementation. Design is a series of decisions on what is the cost-effective implementation that will carry out the system charter and lead to reuse among many systems.

The primary design goals of a system using object-oriented approach are as follows:

- Provide users with a ready-to-use, expressive visual modeling language to develop and exchange meaningful models.
- Furnish extensibility and specialization mechanisms to extend the core concepts.

- Support specifications that are independent of particular programming languages and development processes.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of the object tools market.
- Support higher-level development concepts such as components, collaborations, frameworks and patterns.
- Integrate best practices.

4.3.1 Steps of Design

Following are the design steps in this case study: -

- Define an overall Architecture
- Plan Executable Release

4.3.1.1 Defining an Architecture

Architecture includes making strategic decisions on services as a GUI, storage, communication of software etc. Architecture design also includes selection of major service software to carry out the major implementation work. This software includes: -

- Red Hat Linux Operating System
- Web based Graphical User Interface using HTML
- Platform independent server and client side implementation

4.3.1.2 Plan Executable Release

An executable release plan includes: -

- The goal of the excusable release
- The classes that will be implemented in this executable release
- The use cases to be used
- Required output simulation

4.4 Identification of Classes

Up till now we took a look on the use cases. Question arises what is the specific purpose, for which these use cases are found out and determined. The answer to this question is that in Object-Oriented techniques, one has to find out the categories, classes and their instances. Also with the help of the use cases, one can directly specify the functions and procedure for the class instances. The goal of defining classes is to identify major abstractions in the problem domain. Domain related classes are logical or conceptual. They are independent of any given implementation of the system. Classes can be found out by the nouns used in the problem definition, but it is not necessary that each noun represent a class.

A *class* is the descriptor for a set of objects with similar structure, behavior, and relationships. The model is concerned with describing the intention of the class, that is, the rules that define it. The run-time execution provides its extension, that is, its instances. UML provides notation for declaring classes and specifying their properties, as well as using classes in various ways. Some modeling elements that are similar in form to classes (such as interfaces, signals, or utilities) are notated using keywords on class symbols; some of these are separate metamodel classes and some are stereotypes of Class. Classes are declared in class diagrams and used in most other diagrams. UML provides a graphical notation for declaring and using classes, as well as a textual notation for referencing classes within the descriptions of other model elements.

Classes names should be meaningful that represent completely the properties and type of the class. Following are the major classes identified for the development of this project.

- **FileIOServlet:** Deals with all the file handling involved in the configuration process of Apache Server and DNS Server, based on user requests.
- **SendMailConfigServlet:** Deals with the configuration of SendMail and the processes involved in the creation of System User.
- **UserRequests:** Deals with the placement of the general users' requests of hosting their domains and creating their personal email accounts and respond to users as per requirements.

- **RegServlet:** Deals with the account creation and registration of a new user and handle the database transactions.
- **AuthenticationServlet:** Deals with the user authentication and authorization of different classes of users, and fulfill users' requests regarding security and authentication.
- **PassServlet:** This servlet class deals with the retrieving user's forgot password and also change any existing password.
- **ViewRequests:** This class implements methods of viewing the domain hosting and email account creation requests put by users and also the methods of hosting a new domain and creating email accounts.

The major classes identification along with their operations and behaviors and the interaction among the classes is shown graphically in the following class diagram.

4.5 Class Diagram for ISMC

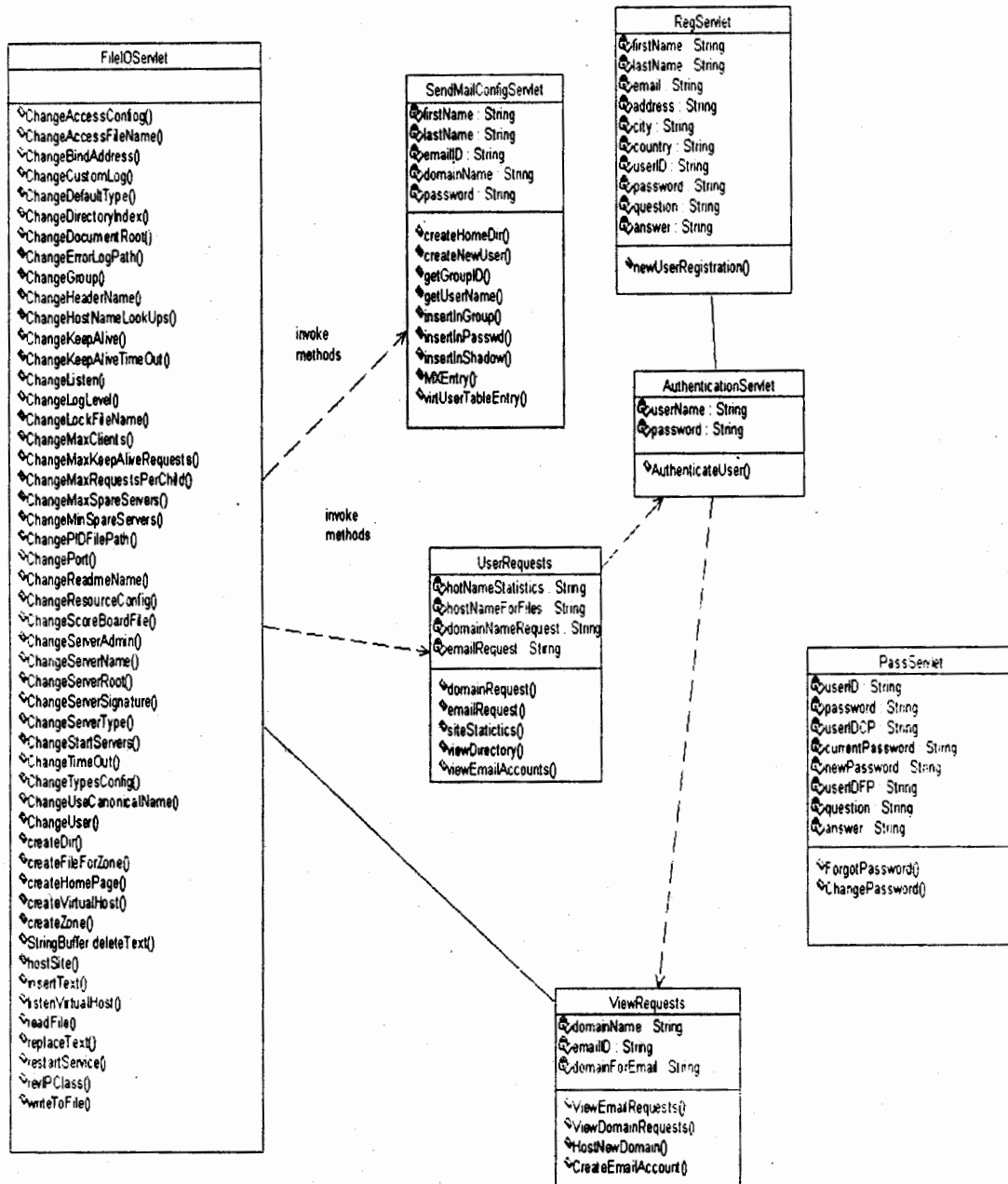


Fig 4.1

4.6 Object Modeling for Classes

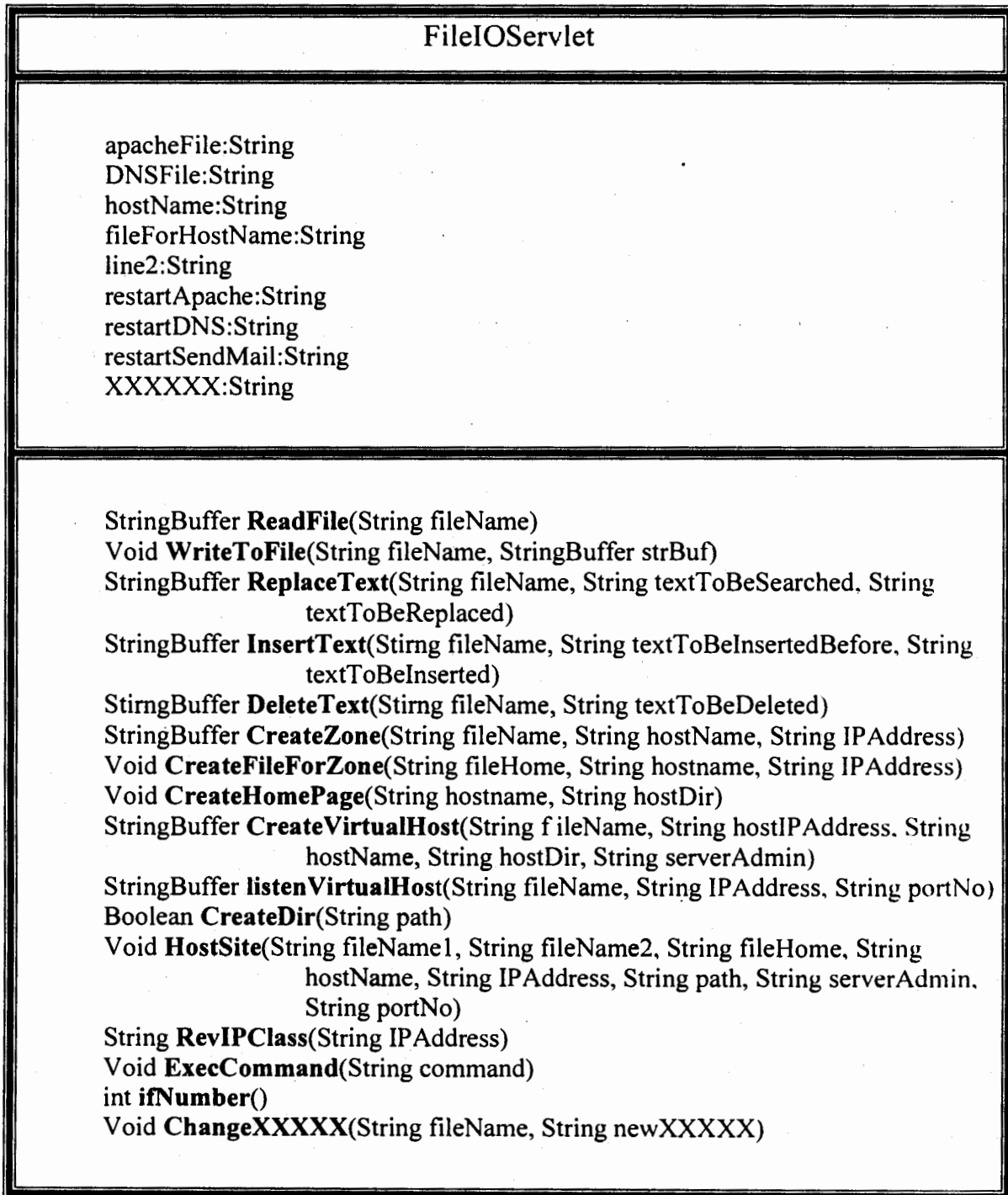


Table 4.1

(Note: In the above class diagram, method `changeXXXXX(String filename, String newXXXXX)` is specified to change any given directive XXXXX. This method is implemented for all possible values of XXXXX)

SendMailConfigServlet
<p>groupFile:String passwdFile:String shadowFile:String apacheFile:String virtFile:String mailServer:String homeDir:String</p>
<p>Int GetGroupID(String groupFile Void InsertInPasswd(String groupFile, String passwdFile, String userName, String homeDir, String name) Void InsertInShadow(String shadowFile, String userName, String password) VOID InsertInGroup(String groupFile, String userName) Boolean CreateHomeDir(String userName) Void CreateNewUser(String groupFile, String passwdFile, String shadowFile, String userName, String password, String homeDir, String name) String GetUserName(String userName) Void DelChar(String filename, Char character) Void MXEntry(String hostName, String mailServer) Void VirtUserTableEntry(String groupfile, String passwdFile, String shadowFile, String password, String homeDir, String name, String virtFile, String apacheFile, String userName, String mailID, String domainName)</p>

Table 4.2

UserRequests
hostNameStatistics:String hostNameForFiles:String domainRequest:String emailRequest:String requestFile:String hostNameForEmail:String page:String
Void SiteStatistics (String hostNameStatistics) Void ViewDirectory (String hostNameForFiles) Void HostRequest (String domainRequest) Void EmailAccountRequest (String emailRequest) Void ViewEmailAccounts (String hostNameForEmail)

Table 4.3

RegServlet
URL:String siteName:String firstName:String lastName:String email:String address:String city:String country:String userID:String password:String question:String answer:String
Void newUserRegistration (String userID, String siteName, String firstName, String lastName, String email, String address,String city, String country, String password, String question, String answer)

Table 4.4

AuthenticationServlet
username:String password:String
Boolean AuthenticateUser (String userName, String password)

Table 4.5

PassServlet
userID:String password:string userIDCP:String currentPassword:String newPassword:String userIDFP:String question:String answer:String
Boolean ChangePassword (String userIDCP, String currentPassword, String newPassword) String ForgotPassword (String userIDFP, String question, String answer)

Table 4.6

ViewRequests
domainName:String emailID:String domainForEmail:String
Void ViewEmailRequests (String emailReqFile) Void ViewDomainRequests (String domainReqFiles) Boolean HostNewDomain (String domainName) Boolean CreateEmailAccount (String emailID, String domainForEmail)

Table 4.7

4.7 Database Design

The availability of inexpensive massive direct access has caused a tremendous amount of research and development activity in the area of Database systems. A database is an integrated collection of data. A database system involves the data itself, the hardware on which the data resides, the software (called the Database Management System DBMS) that controls the storage and retrieval of data and the users themselves. To be able to successfully design and maintain database the following steps should be taken.

- Identification of data entities.
- Identification of attributes of each data entity.
- Identify the relationship between the data entities.

A database is developed for the users registration and authentication services of ISMC. The structure and design of this database is given as under.

Table RegUser:

```

RegUser (
    userID      varchar(20)  PRIMARY KEY,
    siteName    varchar(25)  NOT NULL,
    firstName   varchar(20),
    lastName    varchar(20),
    phone       number(15),
    email       varchar(25),
    address     varchar(30),
    city        varchar(20)
    country     varchar(20)
    userID      varchar(30)
)

```

Table UserAuth:

```

UserAuth (
    userID      varchar(20)  PRIMARY KEY,
    password    varchar(20)  NOT NULL,
    question    varchar(30),
    answer      varchar(30)
)

```

Chapter # 5

Testing

Testing

5.1 Why we do Testing

We do the testing process to identify the maximum number of errors in code with a minimum amount of efforts. Finding an error is thus considered a success rather than failure. On finding an error efforts are made to correct it.

5.2 Inside the Testing Process

A test consists of a number of test cases, where different aspects of the part under test are checked. Each test case tells what to do, what data to use, and what results to expect. When conducting the test, the results including deviations from the planned test case are noted in a test protocol. Normally a deviation indicates an error in the system (although sometimes the test case is wrong and system is right). An error is noted and described in a test report for removal or directly removed by programmer who developed that part.

5.3 General type of Errors

Errors can be of the following types.

- Functional (e.g., a function is missing or incorrect)
- Nonfunctional (e.g., performance is slow)
- Logical (e.g., user interface details are not considered logical)

5.4 Type of Testing

There are number of different types of testing. But the testing types discussed below are normally taken under consideration.

- Unit Testing
- Integration Testing
- System Testing
- Regression Testing

5.4.1 Unit Testing

A unit test is one of a component or a set of components, often done by the developer of the component.

5.4.2 Integration Testing

An integration testing is one of packages that are put together where interface of packages and their collaborations are validated. People having good knowledge of the architecture of the system do this testing.

5.4.3 System Testing

System testing is a functional testing of the entire system and viewed by end user. It is done on the basis of the design. The system test verifies that the system delivers the specified functionality to the end user.

5.4.4 Regression Testing

It is basically a technique to handle changes in the system. A regression test is run after changes have been made to the system, it is actually a series of testes run on the entire system to determine whether any other functionality has been incorrectly affected by the changes. Continuous regression tests will unveil such problem.

5.5 ISMC Testing

Testing process for the Internet Services Management Console (ISMC) started as the different program units were completed.

This project is logically divided in three parts

1. Apache Server Configuration
2. DNS Configuration
3. SendMail Configuration

So all the test phases are primarily implemented on each module, separately and collectively. Initially Unit testing was performed on every program unit of the client side. In client side program units, syntax errors were removed. All the scripting errors were

detached, and the validation checks were tested and corrected entirely. For semantic errors every program unit was tested with the help of test data.

Different client side program units were combined in the form of web pages, and the required functionality of the pages, and their links were also tested. During the integration of all the pages, syntax and semantic errors were checked and removed on the client side. This process was done for the entire client side.

After these tests of the client side, the testing of the server side program units started. The unit tests were applied on every individual module of the server side. The syntax and semantic errors were removed. After this all the modules of the server sides are combined. As the integration completed, again the syntax and semantic errors were checked and removed.

After completion of individual testing of all the modules, the modules were integrated and testing phases were applied then errors were checked and removed. In the next step the system test was applied and all the errors were checked and removed. When all these errors were detected and removed, the final test applied to the system to check whether these changes are effecting the remaining part of the program unit or not. The errors occurred were removed. After these checking all the functionality was checked and found correct.

Chapter # 6

Users' Manual

Users' Manual

6.1 Features of the Software

Internet Services Management Console basically focuses on administration and facilitates the server Administrators running Apache web server, DNS server and SendMail on their server, with a user-friendly interface for the automated process of these servers configuration. It is not only meant for administrators. But general users may also use and interact with it and it best suits the needs and requirements of a general user, including their domain hosting, email accounts creation, troubleshooting etc.

The software itself has the following features to facilitate the administrator and the general users: -

- User Friendly Interface.
- Fast Loading.
- Push Buttons for various activities.
- Hyperlinks to explore the entire site in one touch.
- Can be used through any web browser on any platform.
- Total application is platform independent including the database and the server side implementation.
- Administrator can configure the server remotely.
- Administrator can host a domain on their server from anywhere over the web.
- Administrator can configure email accounts from anywhere over the web.
- Users can put a request to host their domain. Only valid domain names are accepted.
- Users can view the statistics of their already registered domains.
- Users can check files and folders in the home directory of their already hosted domains.
- Users can put a request to configure email accounts reflecting their home page.
- Troubleshooting is done by step-by-step guidance for almost all common problems faced by users.
- Users have an FTP access to upload their files to be served by the server.

6.2 System Requirements

It is well known that Java is a platform independent programming language. The Java Servlet API 2.1 is even web server independent extension of Java. Being this project developed mainly in Java, it is totally platform independent. The server side is developed for Linux platform, but it can work equally well on any platform that supports Java Virtual Machine (JVM). MySQL is used as backend database. MySQL database management system is also platform independent. The use of MySQL makes it totally independent of the platform. The front end is developed using HTML. Here again it is platform independent user interface and can be accessed through any web browser running on any platform.

The purpose of using Java, HTML and MySQL is to develop a platform independent application. The developed system is platform independent and it can run on different operating systems without any change or with slight changes.

The server side implementation needs the following resources

- At least 200 MHz processor
- 2 MB of free hard disk space
- 32 MB RAM
- At least 1 Ethernet interface card

The client side can run on any system with a web browser. (Microsoft Internet Explorer 4.0 or later and Netscape Navigator 4.7 or later fully support the Java servlet API and are recommended)

6.3 How to Use the Software

Being having a web-based interface, this application can be run on any system running a web browser. Loading this software is quite easy and just needs typing the domain name of the server, where it is running, in the address bar of the web browser.

Different activities that Administrator and users can perform using this software are explained as following: -

6.3.1 Home Page

When the site is browsed in any web browser, the following Home page is displayed

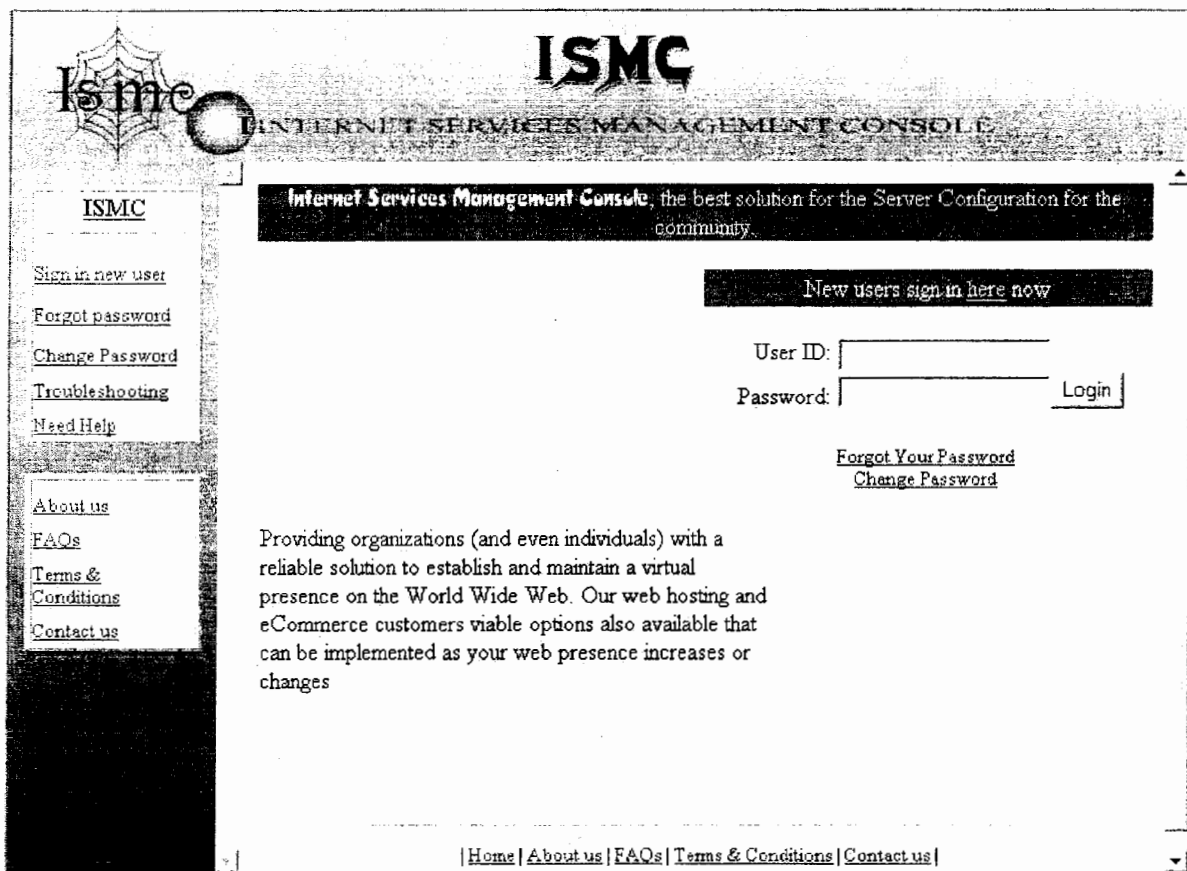


Fig 6.1

- Administrator and registered users can login by providing their login information in the boxes on right side.
- New users can sign in to get registered.
- Users, who forgot their password, can go to *Forgot Password* page.
- Registered users can go to *Change Password* page to change their password.
- Users may get required help through *Troubleshooting* page link.
- Users may view *About ISMC* page, *Terms and Conditions* and *FAQs* page.
- Users may contact the owner or the administrator of the site through *Contact us* link.

6.3.2 Administrator Login

When the Administrator provides its login information and these information are authenticated by the server to be Administrator, the following server generated page is opened.

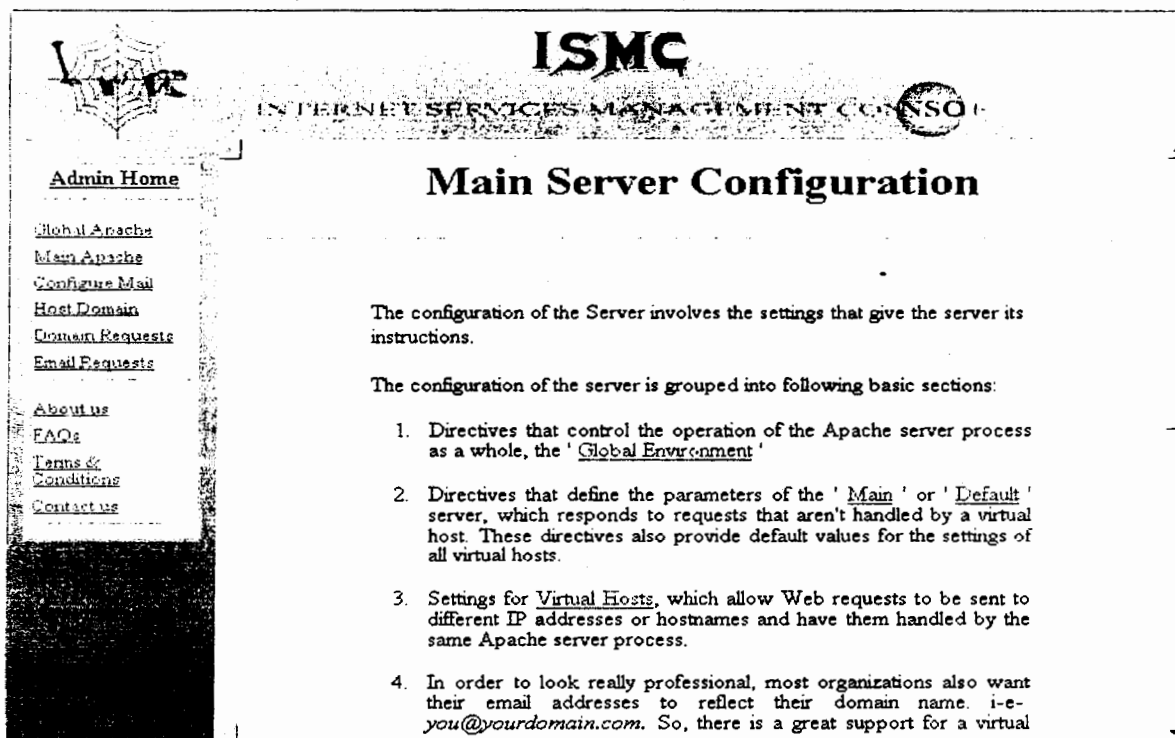


Fig 6.2

- Administrator can click on *Global Apache* link to view the *Global Apache Configuration* page.
- Administrator can click on *Main Apache* link to view the *Main Apache Configuration* page.
- Administrator can click on *Configure Mail* link to go to *Create Email Accounts* page.
- Administrator can click on *Host Domain* link to go to *New Domain Hosting* page.
- Administrator can click on *Domain Requests* link to view the domain hosting requests placed by users.
- Administrator can click on *Email Requests* link to view the Email accounts creation requests placed by users.

6.3.3 Global Apache Configuration

When the Administrator clicks on the *Global Apache* link, following page is opened before user.

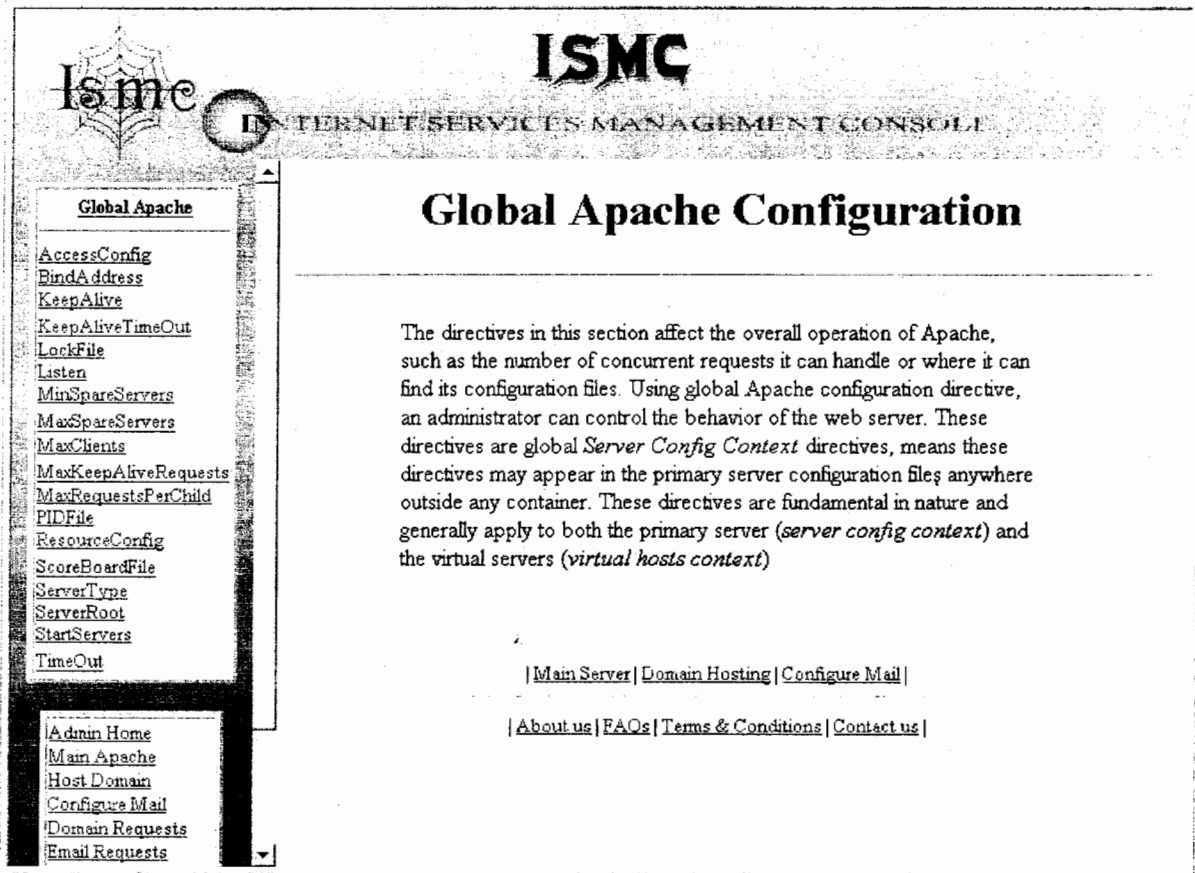


Fig 6.3

All the directives of the *Global Apache Configuration* are displayed in the left frame as hyperlinks. When any of the directives is clicked, the appropriate page for the configuration of that specific directive is displayed in the main frame. Administrator can also click on *Main Server*, *Domain Hosting* and *Configure Mail* links to perform Main Apache Configuration, add new Domain and create new Email users respectively.

6.3.4 Global Apache Directives Configuration

When any of the directives in the left frame is clicked, it opens a page in the main frame for its configuration. That page contains input boxes, radio buttons etc. for each directive as appropriate. Here is a sample page when the *Listen* directive link is clicked.

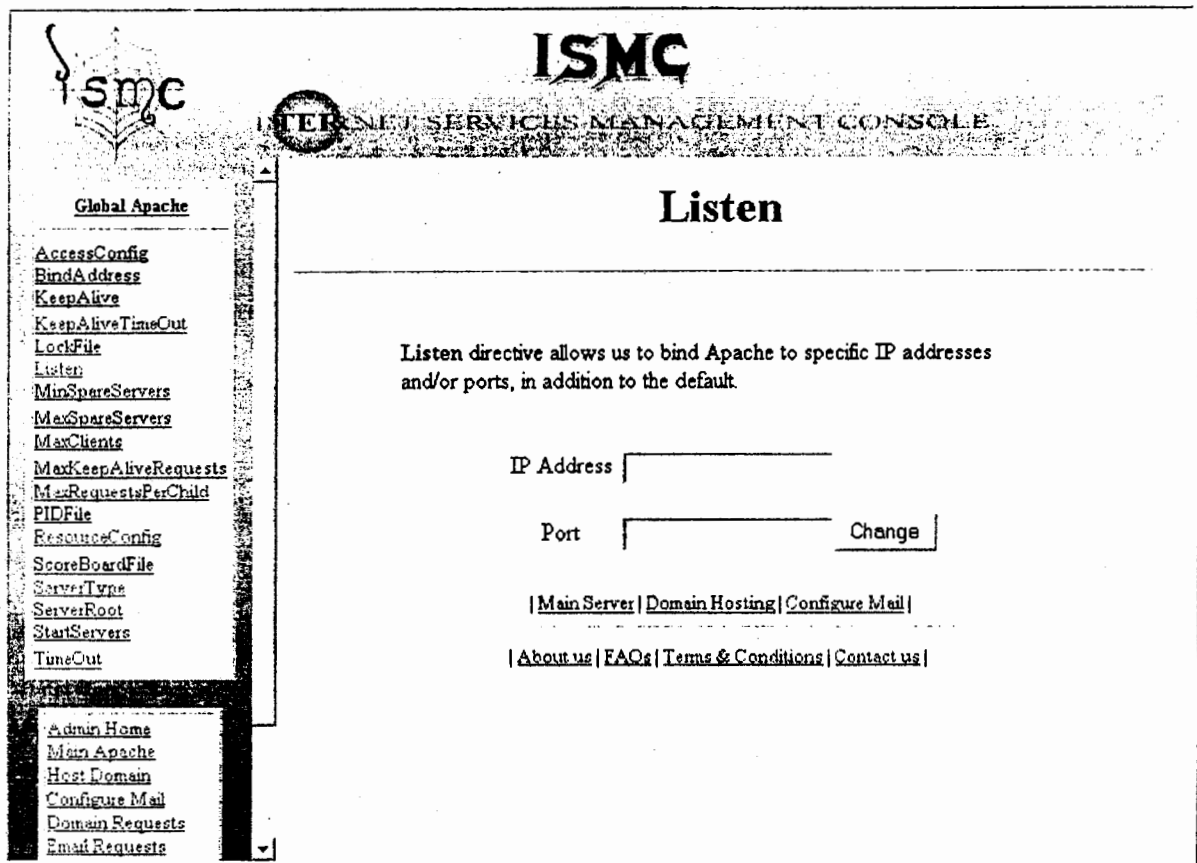


Fig 6.4

Administrator can input valid values and click *Change* button to perform the specified configuration. Same mechanism is adopted for all the directives and each directive link opens its configuration page in the main frame. Administrator can also click on *Main Server*, *Domain Hosting* and *Configure Mail* links to perform Main Apache Configuration, add new Domain and create new Email users respectively.

6.3.5 Main Apache Configuration

When Administrator clicks on *Main Apache* link from Administrator login page, the following *Main Apache Configuration* page is displayed.

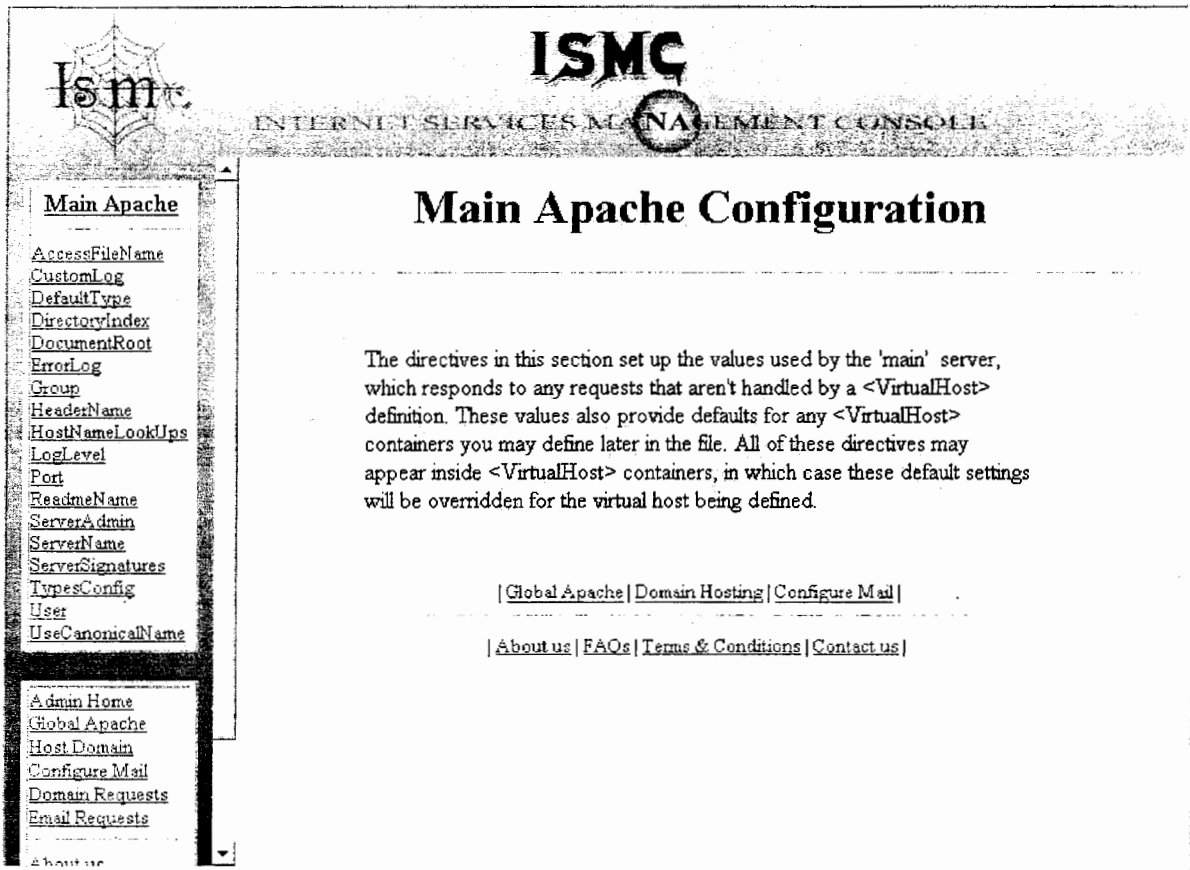


Fig 6.5

Same as in *Global Apache Configuration* page, all the directives of the *Main Apache Configuration* are displayed in the left frame as hyperlinks. When any of the directives is clicked, the appropriate page for the configuration of that specific directive is displayed in the main frame. Administrator can also click on *Global Apache*, *Domain Hosting* and *Configure Mail* links to perform Global Apache Configuration, add new Domain and create new Email users respectively.

6.3.6 Main Apache Directives Configuration

When any of the directives in the left frame is clicked, it opens a page in the main frame for its configuration. That page contains input boxes, radio buttons etc. for each directive as appropriate. Here is a sample page when the *Group* directive link is clicked.

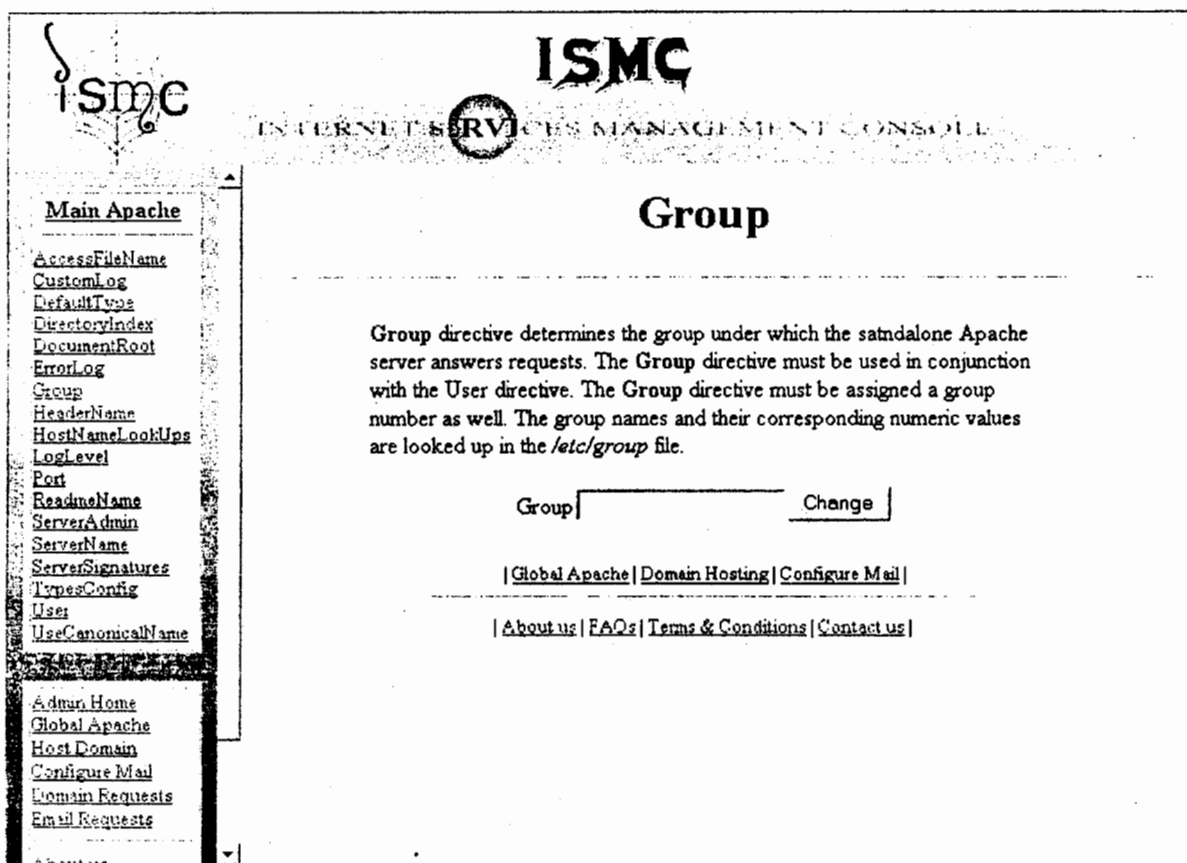


Fig 6.6

Administrator can input valid values and click *Change* button to perform the specified configuration. Same mechanism is adopted for all the directives and each directive link opens its configuration page in the main frame. Administrator can explore the entire site from any page, as links to all the required pages are provided.

6.3.7 Host New Domain

Only Administrator is allowed in this section. When the *Host Domain* link is clicked, it displays a page where Administrator can input valid values for a new domain hosting. On clicking *Host Site* Button, the request is sent to server and the domain is hosted.

ISMC
INTERNET SERVICES MANAGEMENT CONSOLE

Host New Domain

It is possible for a single physical host to have multiple IP addresses for a single network connection. This is commonly called virtual hosting since the machine is acting as several *'virtual'* devices, although it is only one. The *Hosting of a Virtual Site* involves the configuration of *DNS Server* and *Apache Web Server*.

Host Name

IP Address

Port

ServerAdmin

| [Global Apache](#) | [Main Server](#) | [Configure Mail](#) |

| [About us](#) | [FAQs](#) | [Terms & Conditions](#) | [Contact us](#) |

Fig 6.7

Administrator can input only valid values. The entries are again validated on server. If the domain name is not already configured and the IP Address is not already in use, the requested domain is configured. Like all other administrative pages, Administrator can browse the entire site from this page. Hyperlinks to all the main pages are provided. Administrator can click on *Global Apache*, *Main Server* and *Configure Mail* links to perform Global Apache Configuration, Main Apache Configuration and create new Email users respectively.

6.3.8 Create Email Account

This section is also for Administrator only. When the *Configure Mail* link is clicked, a page is opened which inputs required values for creating a new Email Account on the server.

ISM
INTERNET SERVICES MANAGEMENT CONSOLE

Create Email Account

Instead of having a dedicated server system on the internet, an organization (or even an individual) can have a virtual site on an ISP. The virtual web site is really the entree for a professional Internet presence. In order to look really professional, most organizations also want their email addresses to reflect their domain name. i-e- *you@yourdomain.com*. So, there is a great support for a virtual Internet domain with both a virtual web site and virtual email server.

First Name:

Last Name:

Domain Name:

Email User ID:

Password

Confirm Password

| [Global Apache](#) | [Main Server](#) | [Domain Hosting](#) |

| [Home](#) | [About us](#) | [FAQs](#) | [Terms & Conditions](#) | [Contact us](#) |

Fig 6.8

Only valid values are accepted and clicking *Create* button sends the request to the server and after validation, a new email user is created. Links to all other modules are also included in this page, so that Administrator doesn't need to do a lot and explore the desired page by just clicking a link.

6.3.9 View Domain Requests

When the *Domain Requests* link is clicked from the Administrator's home page, a page like following is opened which contains the list of domain names, requested to be hosted by the users.



Fig 6.9

When a domain from the requests list is configured and hosted properly, the domain requests list is automatically updated and that domain name is removed from the requests list.

6.3.10 View Email Requests

When the *Email Requests* link is clicked from the Administrator's home page, a page like following is opened which contains the list of email addresses, requested to be created by the users.

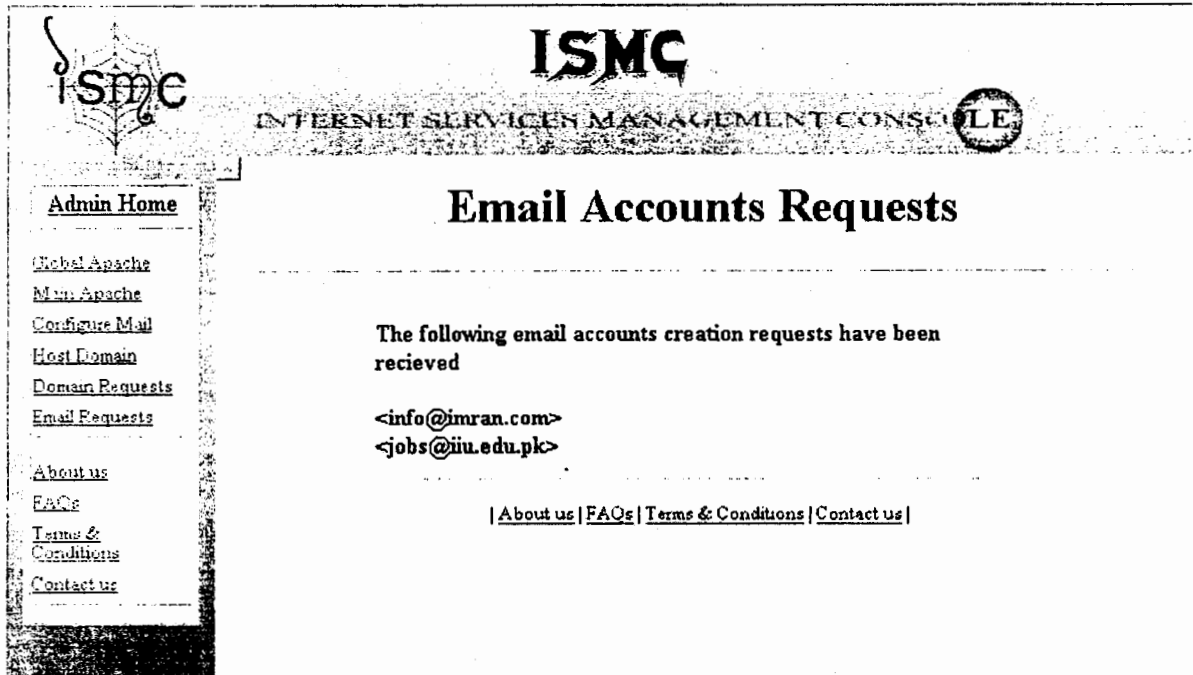


Fig 6.10

When an email account from the requests list is created, the requests list is updated automatically and that email address is removed from the requests list.

6.3.11 Sign in New User

When the *Sign in New User* link is clicked from the home page, a page is opened which input required information to register a new user.

The screenshot shows the 'Sign In New User' page of the Internet Services Management Console (ISMC). The page layout includes a navigation sidebar on the left, a header with the ISMC logo and title, and a central registration form. The form contains two sections: one for personal information and another for login credentials. A 'Register Now' button is located below the second section. At the bottom of the page, there is a footer with navigation links.

ISMC	
Sign in new user	
Forgot password	
Change Password	
Troubleshooting	
Need Help	
About us	
FAQs	
Terms & Conditions	
Contact us	

ISMC
INTERNET SERVICES MANAGEMENT CONSOLE

Sign In New User

Welcome to Internet Services Management Console. If you are a new user and still not registered with ISMC, please fill in the following registration form completely. (The fields with * are required)

First Name:	*	<input type="text"/>
Last Name:	*	<input type="text"/>
Email Address:	*	<input type="text"/>
Mailing Address:		<input type="text"/>
City:		<input type="text"/>
Country:		<input type="text" value="Pakistan"/>

User ID:	*	<input type="text"/>
Password:	*	<input type="text"/>
Confirm Password:	*	<input type="text"/>
Secret Question:	*	<input type="text"/>
Answer to Secret Question:	*	<input type="text"/>

| [Home](#) | [About us](#) | [FAQs](#) | [Terms & Conditions](#) | [Contact us](#) |

Fig 6.11

User can fill in the registration form. The request is sent to server on filling out the form completely and clicking the *Register Now* button. After server side validation, the user is registered and a unique User ID is assigned to it and displayed before user.

6.3.12 Forgot Password

When the *Forgot Password* link is clicked, it opens a new page in the main frame which inputs required information from the user to retrieve the forgot password.

ISMC
INTERNET SERVICES MANAGEMENT CONSOLE

Forgot Password

If you have your personal ISMC User ID and forgot your password, fill in the following form and get your password

Please fill in carefully, as it is the only way to retrieve your password

User ID: *	<input type="text"/>
Secret Question: *	<input type="text"/>
Answer: *	<input type="text"/>

| [About us](#) | [FAQs](#) | [Terms & Conditions](#) | [Contact us](#) |

Fig 6.12

Submit Now button sends the request to server and on validation of the provided information, the forgot password is overwritten by a new password and the new password is displayed before user.

6.3.13 Change Password

When the *Change Password* link is clicked, it opens a new page in the main frame which inputs required information from the user to change password.

ISMC
INTERNET SERVICES MANAGEMENT CONSOLE

Change Password

If you have your personal ISMC User ID and want to change your password, you may change your password here

User ID:	*	<input type="text"/>
Current Password:	*	<input type="text"/>
New Password:	*	<input type="text"/>
Confirm Password:	*	<input type="text"/>

| [About us](#) | [FAQs](#) | [Terms & Conditions](#) | [Contact us](#) |

Fig 6.13

Clicking *Change Password* button sends the request to server and on validation of the provided information, the requested password is changed.

6.3.14 User Login

When any registered user logs in to the system, the following server generated page is opened to facilitate users to perform different tasks, as desired.

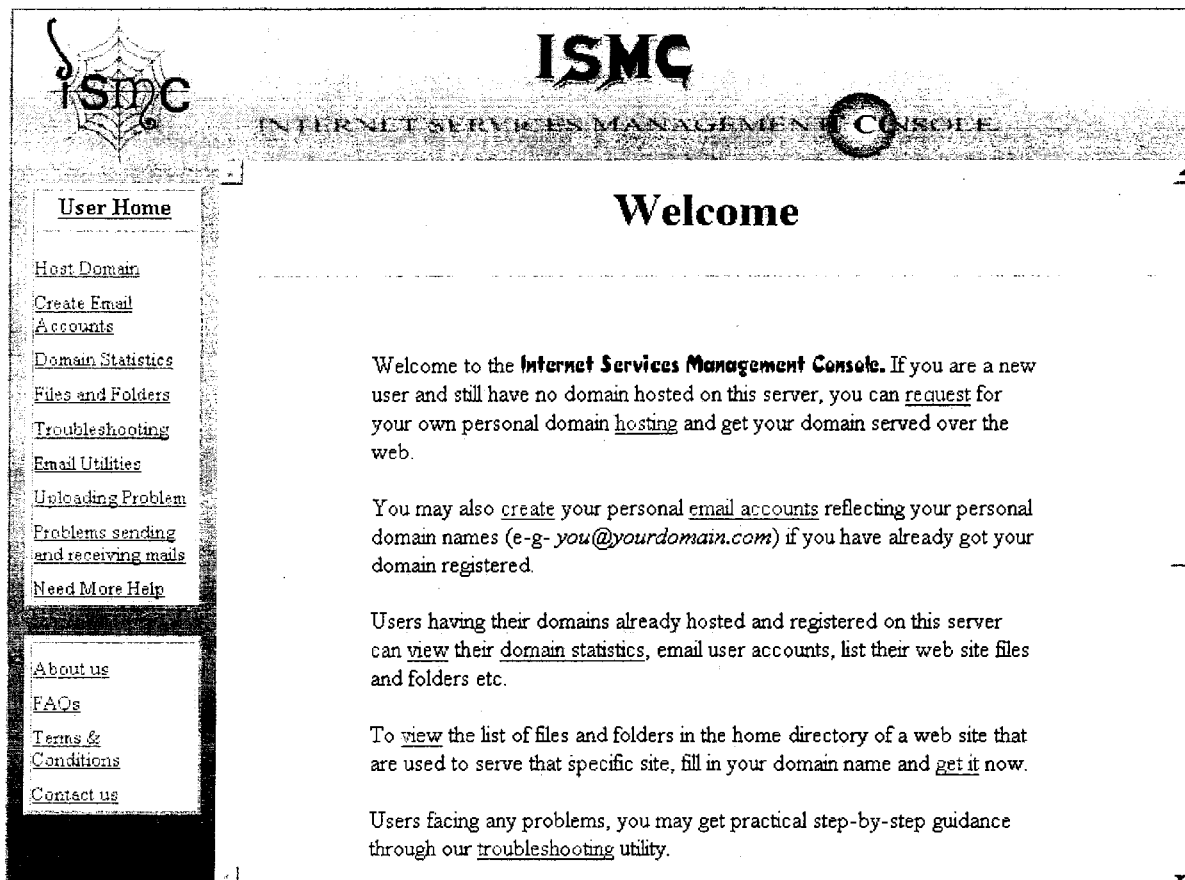


Fig 6.14

All the links are placed in the left frame. User just needs to click a link and the desired page is displayed in the main frame.

6.3.15 Domain Hosting Request

When the *Host Domain* link is clicked, the following page is opened, which inputs user's request to host its domain.

The screenshot shows the 'Host Your Domain' page in the Internet Services Management Console (ISMC). The page has a header with the ISMC logo and the text 'INTERNET SERVICES MANAGEMENT CONSOLE'. A navigation menu on the left includes links for 'User Home', 'Host Domain', 'Create Email Accounts', 'Domain Statistics', 'Files and Folders', 'Troubleshooting', 'Email Utilities', 'Uploading Problem', 'Problems sending and receiving mails', and 'Need More Help'. The main content area is titled 'Host Your Domain' and contains a welcome message: 'Welcome to the Internet Services Management Console. If you are a new user and still have no domain hosted on this server, you may place a request to register and host your site here'. Below the message is a form with a 'Domain Name:' label, an input field, and a 'Submit Now' button. At the bottom of the page, there are links for 'About us', 'FAQs', 'Terms & Conditions', and 'Contact us'.

Fig 6.15

This request is sent to the server. On the server side, this domain name is validated not to be already hosted and neither the same domain name is already requested to be hosted.

6.3.16 Email Accounts Creation Request

When the *Create Email Accounts* link is clicked, the following page is opened, which inputs user's request to create email accounts.

The screenshot shows the 'Create Email Account' page in the Internet Services Management Console (ISMC). The page has a header with the ISMC logo and the text 'INTERNET SERVICES MANAGEMENT CONSOLE'. On the left, there is a navigation menu with the following links: [User Home](#), [Host Domain](#), [Create Email Accounts](#), [Domain Statistics](#), [Files and Folders](#), [Troubleshooting](#), [Email Utilities](#), [Uploading Problem](#), [Problems sending and receiving mails](#), and [Need More Help](#). Below this menu is a box containing links for [About us](#), [FAQs](#), [Terms & Conditions](#), and [Contact us](#). The main content area is titled 'Create Email Account' and contains a welcome message: 'Welcome to the Internet Services Management Console. If you want to create a virtual email account reflecting your own domain, hosted on same server, you can put a request to configure an email account.' Below the message is an 'Email:' input field followed by a 'Submit Now' button. At the bottom of the main content area, there are links for [About us](#), [FAQs](#), [Terms & Conditions](#), and [Contact us](#).

Fig 6.16

When the *Submit Now* button is pressed, this request is sent to the server. On the server side, this domain name is validated to be a hosted domain over the server, the email account not do already exist and neither the same email account request is already placed.

6.3.17 View Domain Statistics

When the Domain Statistics link is clicked, the following page is displayed before user. This page can input user's domain name and send the domain name to the server.

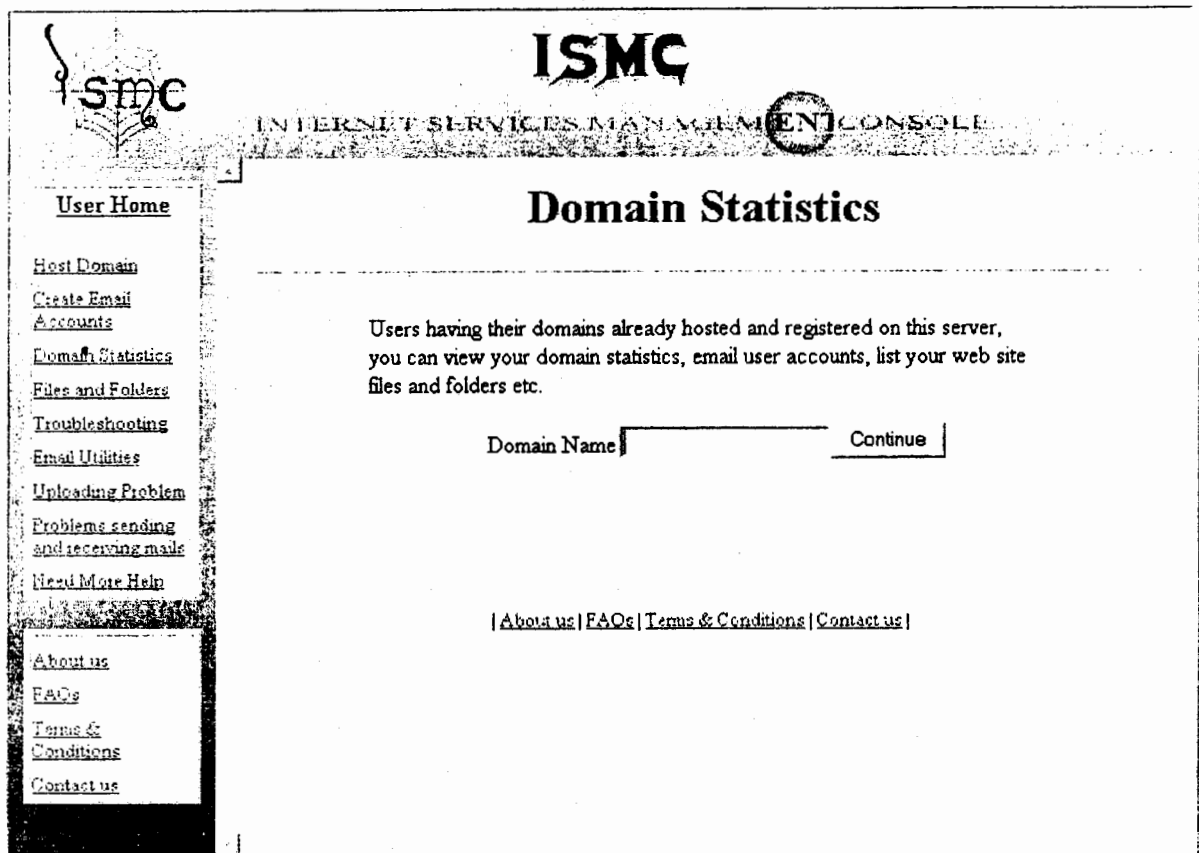


Fig 6.17

Only valid domain names are input. When the *Continue* button is pressed, the domain name is sent to the server. On the server side, the request is again validated to restrict users to access any domain statistics other than its own. If the requested is valid, the server generates a page that contains the statistics of the domain over the server.

6.3.18 Files and Folders

When the *Files and Folders* link is clicked, the following page is displayed before user, which inputs a valid domain name and sends the request to the server.

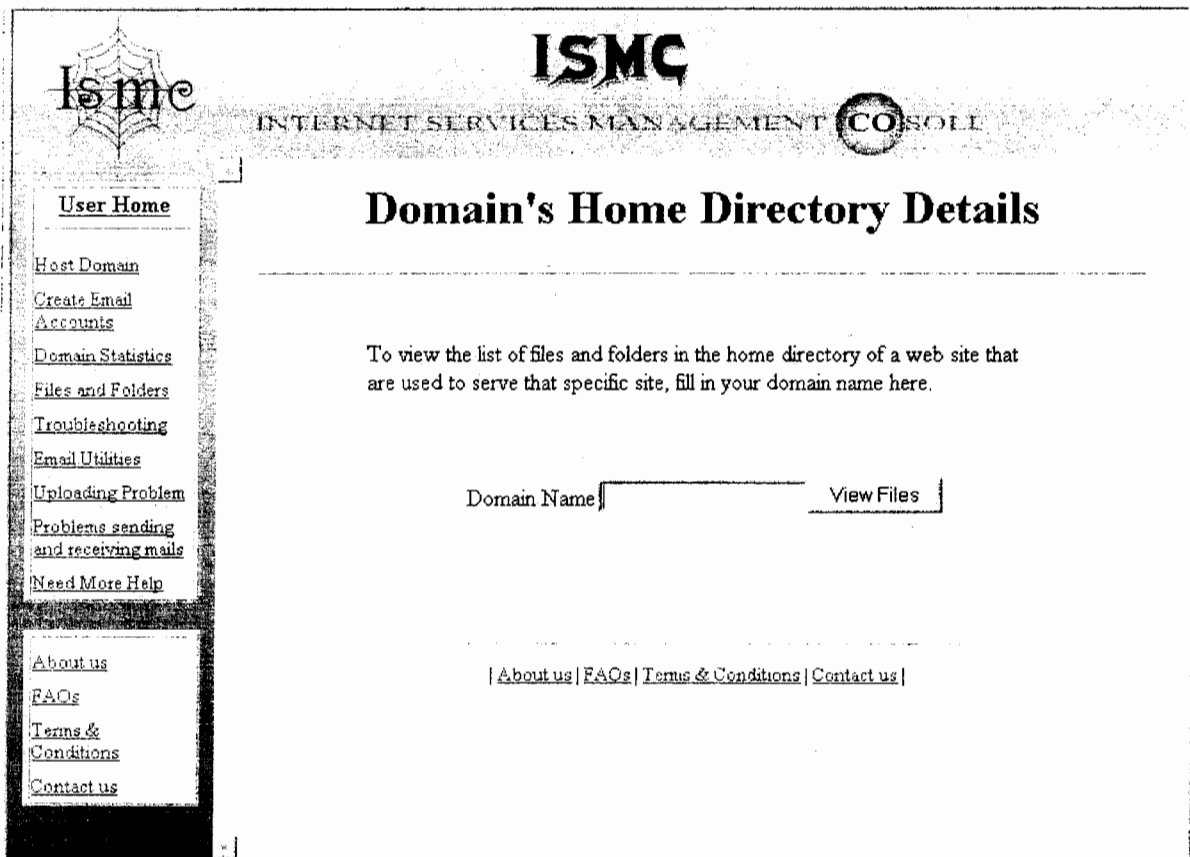


Fig 6.18

Only valid domain names are input. When the *View Files* button is pressed, the domain name is sent to the server. On the server side, the request is again validated to restrict users to access any domain database other than its own. If the requested is valid, the server generates a page that contains the list of files and folders in the home directory of the domain over the server.

6.3.19 Troubleshooting

When the troubleshooting link is clicked, the following page is displayed before user. This page handles with all the troubleshooting of the problems commonly faced by the users.

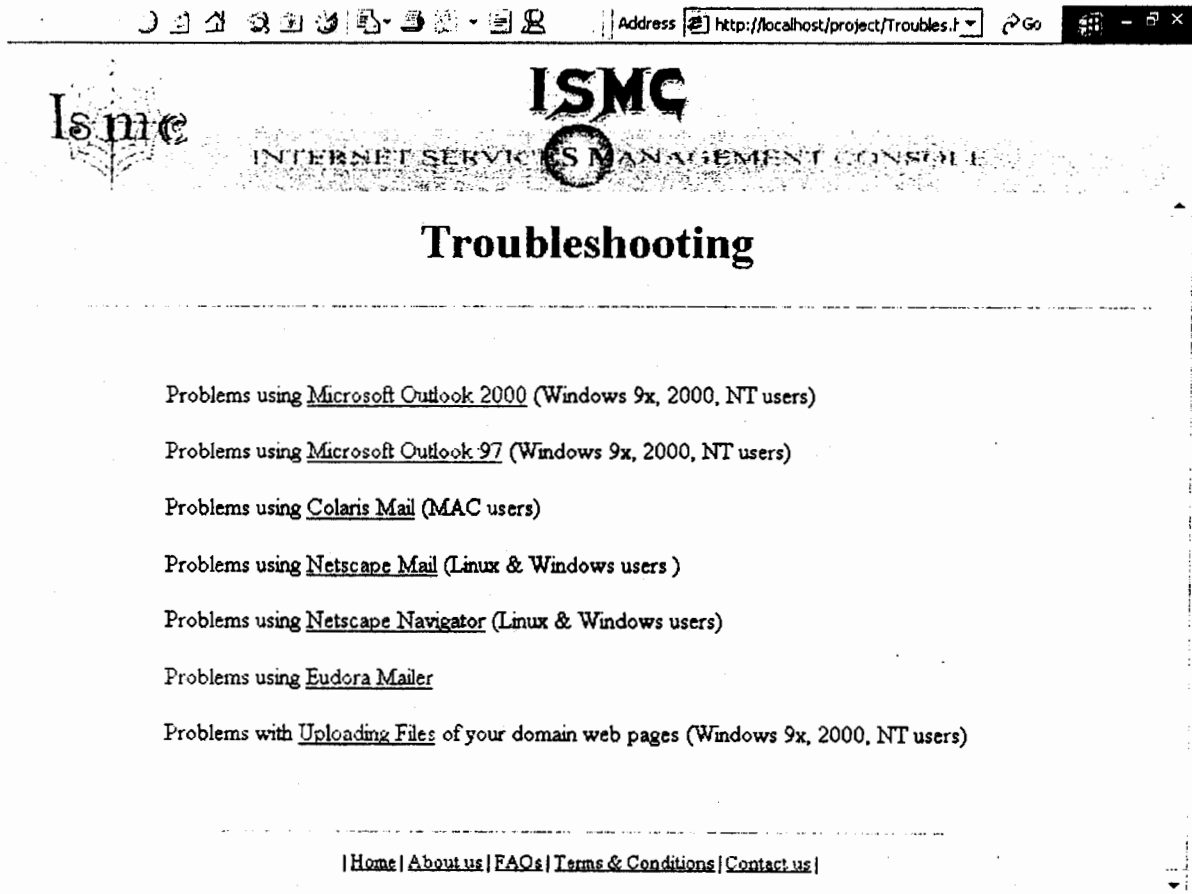


Fig 6.19

This page contains links to different pages. All of them deal with a specific troubleshooting and guide the users with step-by-step guidance.

6.3.20 Uploading Problems

When a user needs any help regarding uploading its files to the server, a step-by-step guide is available. Clicking the *Uploading Files* link opens a page like this.

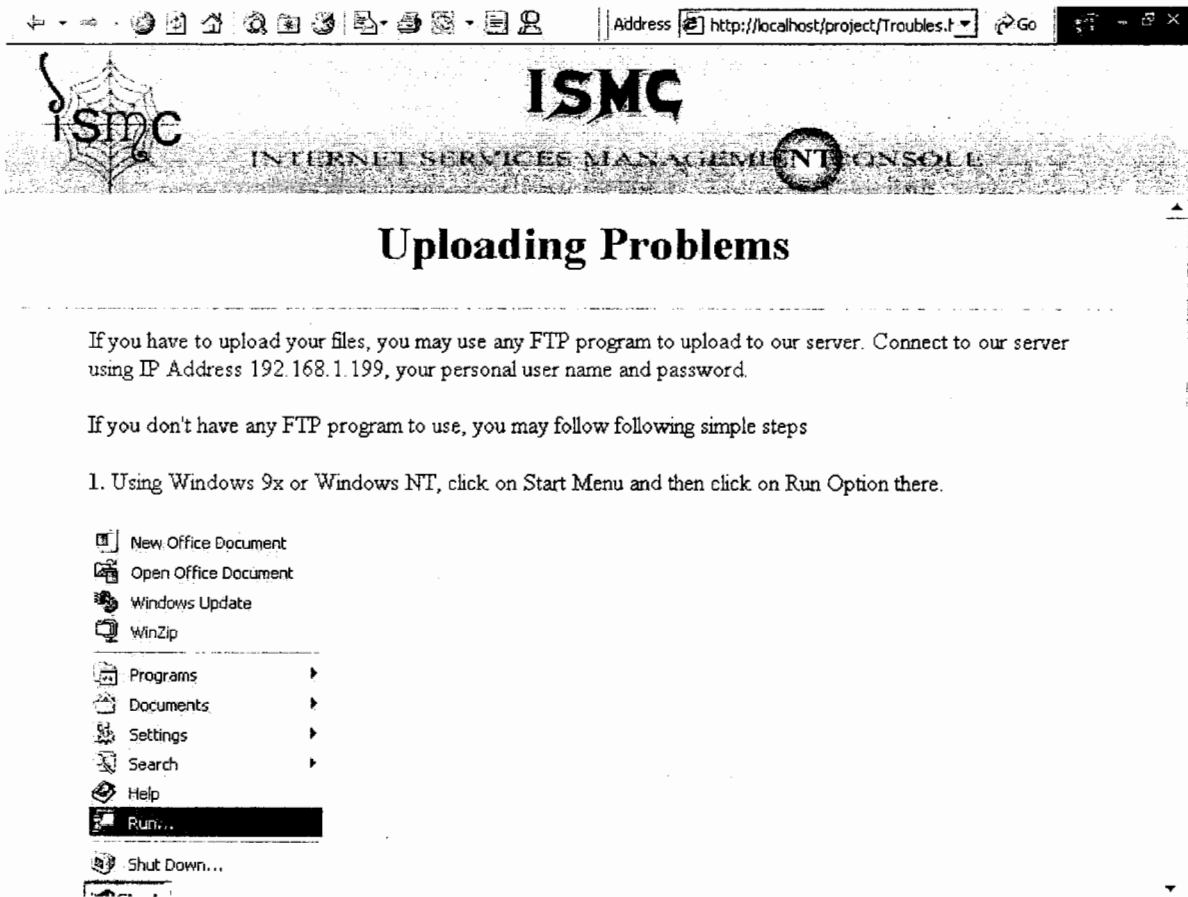


Fig 6.20

6.3.21 Configuring Mail Utilities

When there is any help required, on clicking the specific link, some help page is opened. Here is a sample page displayed, when the *Microsoft Outlook 97* link is clicked.



1. Click on the **Tools** menu and select **Services**.

NOTE: In the Services tab, if you do not see the Internet Mail service installed, you will have to install it. If you do see it, select the item and click on the Properties button.

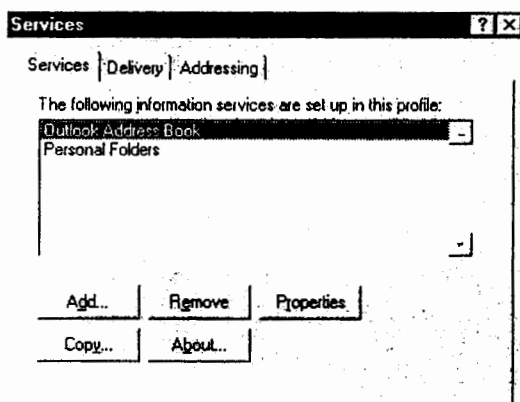
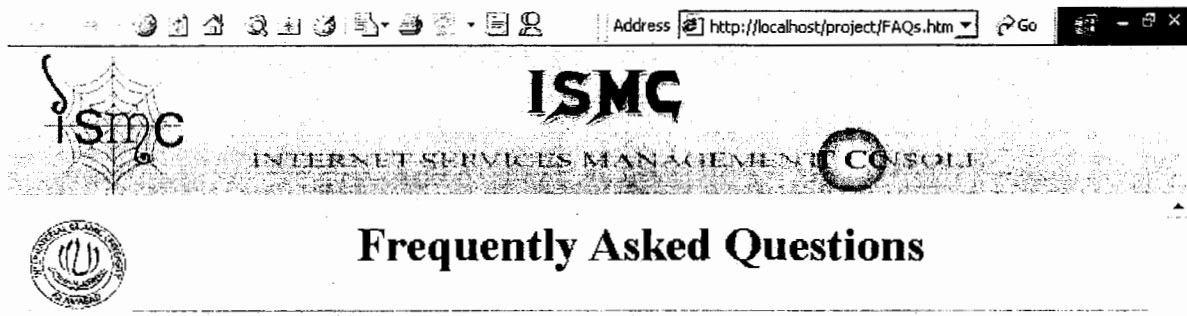


Fig 6.21

6.3.22 Frequently Asked Questions

Following page is opened in new browser window, when the *FAQs* link is clicked



Internet Services Management Console

The following is a list of frequently asked questions (FAQs) about the domain name registration process and the new competitive registration environment. It is expected that this list will be updated frequently, so please check back often.

- **What is a domain name?**

Domain names are the familiar, easy to remember names for computers on the Internet (such as internic.net). They correspond to a series of numbers (called Internet Protocol numbers) that serve as routing addresses on the Internet. Domain names are used generally as a convenient way of locating information and reaching others on the Internet.

- **What does it mean to "register" a domain name?**

The Internet domain name system (DNS) consists of a directory, organized hierarchically, of all the domain names and their corresponding computers registered to particular companies and persons using the Internet. When you register a domain name, it will be associated with the computer on the Internet you designate during the period the registration is in effect.

Fig 6.22

6.3.23 Terms & Conditions

When the *Terms & Conditions* link is clicked, a page is opened in new browser window, which describes the terms and conditions to use ISMC in detail.

Address <http://localhost/project/Terms.htm> Go

ISMC
INTERNET SERVICES MANAGEMENT CONSOLE

Terms & Conditions

Internet Services Management Console

Your utilization of *ISMC*'s services constitutes your express agreement to abide by this Acceptable Use Policy (the "Rules, as the same may be changed from time-to-time by *ISMC*, at its sole and absolute discretion. These Rules are designed to enhance the quality of our Web Hosting services and to protect our customers, and the Internet community as a whole, from illegal, irresponsible, or disruptive Internet activities. These Rules apply to all clients of *ISMC*. We hope and expect that common sense and good judgment will guide all of our customers' use of our *ISMC* Services. Even with such Rules in effect, minors should be supervised when using the Internet.

Service Contract:
ISMC Virtual Hosting Service is subject to the following terms and conditions:

1. The specific fees and surcharges vary depending on the particular plan ordered. *ISMC* may change and institute new fees and surcharges at any time upon thirty(30) days prior notice via e-mail or snail mail. Either party may terminate the service without cause or prior notice.
2. The client agrees to comply with the Acceptable Use Policy set forth above and to refrain from any of the unacceptable uses set forth therein. *ISMC* reserves an absolute right to determine for itself

Fig 6.23

6.3.24 About Us

Clicking *About us* link opens the following page in a new browser window.

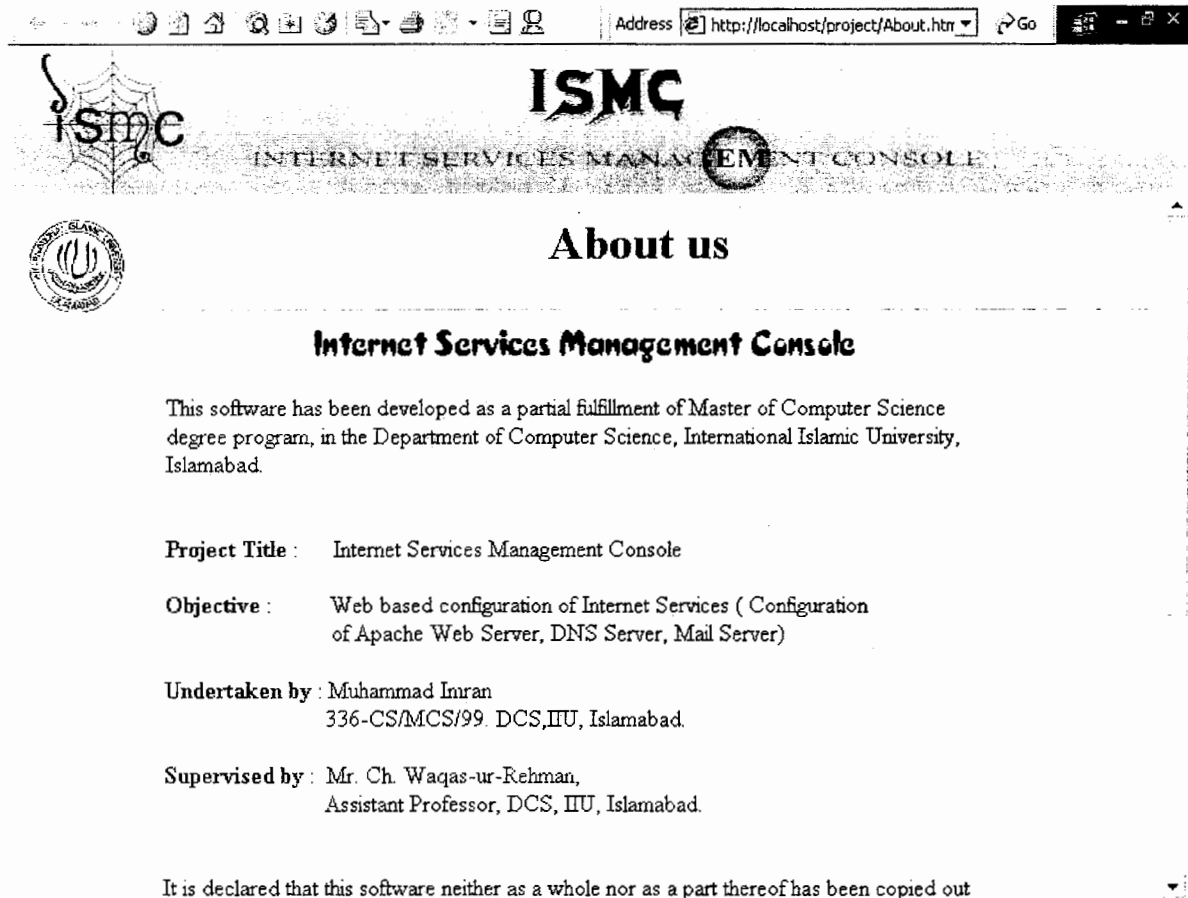


Fig 6.24

BIBLIOGRAPHY

Books

1. Rob Pooley, Perdita Stevens, "**Using UML**", 1st edition, Addison-Wasley, 2000
2. Alan Simpson, John Ray, "**Using RedHat Linux**", 2nd edition, QUE, 1999
3. Kurt Well, "**Linux Programming by Example**", 1st edition, QUE, 2000
4. Muhammad J Kabir, "**Apache Server Bible**", 3rd edition, IDG Books, 1996
5. Khalid A. Mughal, "**Programmer's guide to Java Certification**", 1st edition, Addison-Wasley, 2000
6. Jason Hunter, William Crawford, "**Java Servlet Programming**", 3rd edition, O'Reilly, 2000
7. Shiren & Shiren, "**Advanced JavaScript Programming**", 2nd edition, bpb, 1999
8. Dietel & Dietel, "**C++ How to Program**", 2nd edition, PRENTICE HALL, 1996

Internet Web Sites

<http://www.apach.org>
<http://www.apacheweek.com>
<http://www.linux.org>
<http://www.redhat.com/howto>
<http://www.sendmail.org>
<http://www.virtualhosting.com>
<http://www.informit.com>
<http://www.java.sun.com>
<http://www.coolservlets.com>