

Jaypee University of Information Technology
Waknaghat, Distt. Solan (H.P.)

Learning Resource Center

CLASS NUM:

BOOK NUM.:

ACCESSION NO.: SP08009 | SP0812009

This book was issued is overdue due on the date stamped below. If the book is kept over due, a fine will be charged as per the library rules.

Due Date	Due Date	Due Date

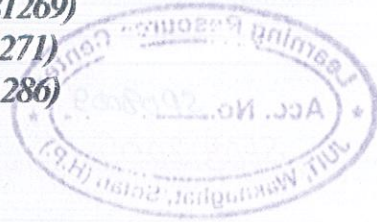
WEBMAIL

SUBMITTED BY:

AKHIL BHANDARI (081269)

HINA SINGHAL (081271)

ABHINAY GARG (081286)



UNDER THE GUIDANCE OF:

Dr. RAVIRASTOGI

(Assistant Professor, JUIT)

Dr. YASHWANT SINGH

(Assistant Professor, JUIT)



MAY-2012



Submitted in partial fulfillment of the Degree of

Bachelor of Technology

DEPARTMENT OF COMPUTER SCIENCE

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,

WAKNAGHAT

TABLE OF CONTENTS

1. Certificate.....I

2. Acknowledgements.....II

3. Summary.....III

4. Problem Statement.....IV

5. Objective and Scope of the Project.....V

6. Methodology.....VI

7. List of Acronyms / Abbreviations.....VII

7. List of Figures.....VIII

8. Chapter 1-Introduction.....1

 1.1 Overview.....1-2

 1.2 History of Email Clients.....2

 1.3 Importance of Email clients.....2

9. Chapter 2-Concept.....3-5

 2.1 Purpose.....4

 2.2Advantages.....4-5

 2.3 Disadvantages.....5

10 Chapter 3-Protocols Used To Support Webmail.....6-9

 3.1 POP3.....6

3.2 SMTP.....	6-7
3.3 IMAP.....	7
3.4 HTTP.....	8
3.5 Mail Processing.....	9
11. Chapter 4-Literature Survey.....	10-15
4.1 Different Email Clients and Their Features.....	12
4.1.1 YahooMail.Com.....	12-13
4.1.2 Gmail.Com.....	13-14
4.1.3 Rediffmail.Com.....	14-15
4.1.4 Indiatimesmail.Com.....	15
12. Chapter 5-Description and Coding.....	16
6.1 Pages.....	16
5.1.1 Login Page.....	16-18
5.1.2 Sign Up Page.....	18-20
5.1.3 Inbox.....	20-21
5.1.4 Compose.....	22-23
5.2 Servlets.....	23-44
5.2.1 Checkuser.....	23-27
5.2.2 Insert Servlet.....	27-30
5.2.3 Readmail Servlet.....	30-33
5.2.4 Send Mail.....	34-36

5.2.5 Message View from Server.....	36-40
5.2.6 Send Attachment	40-43
5.2.7 Database Connection.....	44
13. Future Prospects.....	45
14. Bibliography.....	46-47
15. Web References.....	48

CERTIFICATE

This is to certify that the work titled “**WEBMAIL**” submitted by “**Akhil Bhandari (081269), Hina Singhal(081271), Abhinay Garg(081286)**” in partial fulfillment for the award of degree of **B.Tech** of Jaypee University of Information Technology, Wakanaghat has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor *Rastogi*
Name of Supervisor *Dr. Ravi Rastogi*
Designation *Assistant Prof.*
Date *1-Jun-2012*

ACKNOWLEDGEMENT

We would like to thank Dr. Ravi Rastogi for his kind support at every step of this project. The supervision and support that he gave truly help the progression and smoothness of the project. The co-operation is much indeed appreciated.

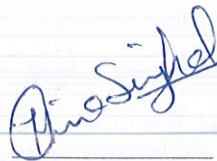
Also, we would like to convey our thanks to Dr. Yashwant Singh for extending his support.

Last but not the least, we express gratitude towards Dr. Nitin (Project coordinator), Brig.(Retd.) Satya Prakash Ghrera (Head of Computer Science Department) and Brig.(Retd.) Balbir Singh (Director of Jaypee University of Information Technology).



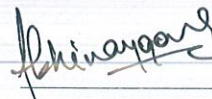
AKHIL BHANDARI

DATE: 1-Jun-12



HINA SINGHAL

DATE: 1-Jun-12



ABHINAY GARG

DATE: 1/6/12

SUMMARY

Electronic mail, E-mail is a method of exchanging digital messages from a computer to one or more recipients. There are two popular way to access email accounts – through webmail (interface) and with email client. With web mail, emails are read or send through a browser and the web mail interface. An email client is a piece of software that a person installs on his machine and uses it to read and send emails from a computer. In this project we have concentrated on implementing webmail.

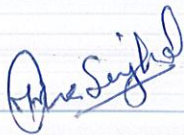
There are three components of an email message - the message envelope, the message header, and the message body. The message header contains control information, sender's email address and one or more recipient addresses, subject header field and a message submission date/time stamp.

The aim of the current system is to connect the citizens through a global network. This will also leads to communication with those once to whom with we have never met. This project will allow us to join more individuals, communities & also to register our personal, professional, social information along with the transfer of the data files & much more.



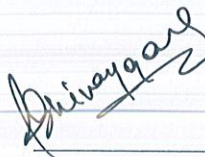
AKHIL BHANDARI

DATE: 1-Jan-12



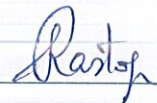
HINA SINGHAL

DATE: 1/5/12



ABHINAY GARG

DATE: 1/6/12



DR. RAVI RASTOGI

SUPERVISOR

DATE: 1/6/12

PROBLEM STATEMENT

The problem statement of the project is to develop an application that is capable of the following functions.

- a) **Register Users:** A user should be able to create an account for the usage of the application and be able to use the various features of that application.
- b) **Store information:** The application should be able to ask information from the users during registration and store them for future use.
- c) **Login the users to their respective accounts:** After registration, the users should be capable of logging into their accounts and use the features of the application
- d) **Protect the user accounts from unauthorized access:** The account of the users should be protected against people whom they do not wish to allow access to their accounts.
- e) **Provide a space to the users to write their emails:** The users should be given a text area where they can compose their messages and edit them before sending them to other users.
- f) **Send Emails:** The users should be able to send their messages to other users of other email clients.
- g) **Attach different files to their mails:** Along with text that has been written, the users should be able to attach documents and files to their emails.
- h) **Receive Emails:** The users should be able to receive messages and emails from other users who send the former emails.
- i) **Allow users to read the received mails:** The users should be able to view the mails that they receive from other users.

OBJECTIVE AND SCOPE OF THE PROJECT

Project Objectives:

The objective of this project is to develop an email client that has all the features that are available in different email clients, but are not available in a single email client.

This project will develop an email client that will be capable of sending and receiving emails along with other features namely password protection, chat with other users, address book etc.

Project Scope:

The scope of an email client is to

- a) SignUp for account access
- b) Facilitate sending of mails
- c) Facilitate receiving of mails
- d) Adding attachments to mails to be sent
- e) View/download attachments from received mails

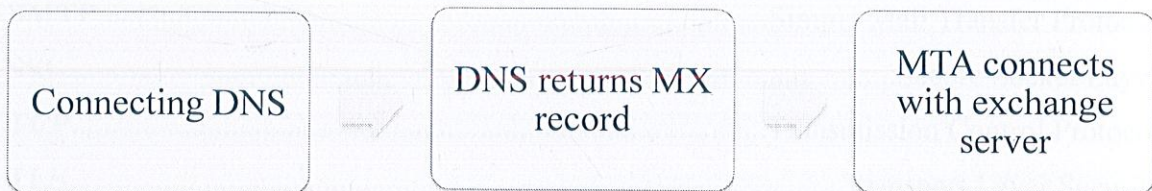
METHODOLOGY

The overall project has been divided into three phases:

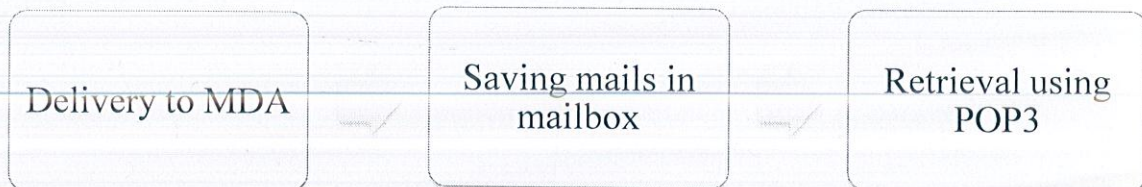
1. Phase one consists of submitting an email is by a mail client MUA to a mail server MSA using SMTP protocol. The MSA delivers the mail to its mail transfer agent MTA.



2. Phase two consists of locating the target host. DNS is used to look up the mail exchanger record (MX record) for the recipient's domain.



3. Exchange server delivers it to a MDA which saves messages in the mailbox. Finally mail is retrieved using POP3.



LIST OF ACRONYMS & ABBREVIATIONS

1. DNS.....Domain Name System
2. HTTP.....Hypertext Transfer Protocol
3. IMAP..... Internet Message Access Protocol
4. IP.....Internet Protocol
5. ISP..... Internet Service Provider
6. MDA.....Mail Delivery Agent
7. MIME..... Multipurpose Internet Mail Extension
8. MSA..... Mail Submission Agent
9. MTA.....Mail Transfer Agent
10. MUA.....Mail User Agent
11. MX RECORD.....Mail Exchanger Record
12. POP.....Post Office Protocol
13. SMTP.....Simple Mail Transfer Protocol
14. SSL.....Secure Socket Layer
15. TCP.....Transmission Control Protocol
16. TLS.....Transport Layer Security

LIST OF FIGURES

Fig 2.1 Basic view of Email processing.....	3
Fig 3.1 Working model with protocols.....	8
Fig 4.1 Snapshot of Yahoo.....	13
Fig 4.2 Snapshot of Gmail.....	14
Fig 4.3 Snapshot of Rediffmail.....	15
Fig 4.5 Snapshot of Indiatimes.....	15

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

There are two popular way to access email accounts – through webmail (interface) and with email client. With web mail, emails are read or send through the browser and the web mail interface. An email client is a piece of software that a person uses to read and send emails from a computer.

The term refers to any system capable of accessing the user's email mailbox, regardless of it being a mail user agent, a relaying server, or a human typing on a terminal. In addition, a web application that provides message management, composition, and reception functions is also considered an email client, but more commonly referred to as webmail.

As email “evolves” and become powerful in more complex, MUA’s are becoming increasingly powerful in the way they interact with email, this gives users increased functionality and flexibility. Popular email clients include Microsoft Outlook, IBM Notes, Mozilla's Thunderbird, and Apple Inc.’s Mail. The term “mail user agent” is less familiar to the average person, but is used in email headers. The headers of the email supply information to the mail servers or computers that handle transferring messages across networks like the Internet. Email Encryption enables to safeguard privacy encrypting the mail sessions, the body of the message, or both.

The popular protocols for retrieving mails include POP3, sending mail is usually done using the SMTP protocol. The Post Office Protocol (POP) allows the client to download messages one at a time and only delete them from the server after they have been successfully saved on local storage.

Client settings require the user's real name and email address for each user's identity. Client settings require the server's name or IP address, and the user name and password for each remote incoming mailbox.

Another important standard supported by most email clients is MIME, which is used to send binary file email attachments. Attachments are files that are not part of the email proper, but are sent with the email.

In a scenario where emails are the most useful, easiest and important form of communication, a mail user agent plays a very important role in easing the task of writing mails and sending it efficiently as it was meant to be sent.

An effective and efficient Webmail Client is one which sends mail without delays and without any trouble receives them to the satisfaction of the user. And with the advent of technology, Mail user agents also facilitate the chat service in which two or more users can instantly message each other without waiting long time for the reply from the other users.

1.2 HISTORY OF EMAIL CLIENTS

In the year 1995, a number of people were trying to enable emails to be accessed on the internet web browser. In the United States, WEBEX was released by Matt Mankins, while Soren Vejrum and Luca Manunza released WWW MAIL and WEBMAIL in Europe. These applications were built in perl scripts and their source codes were fully downloadable. In 1994, Bill Fitler, started working on web based email as a CGI program for Windows NT and demonstrated it publicly in 1995 at Lotusphere.

1.3 IMPORTANCE OF EMAIL CLIENTS

In today's world, where time is of utmost importance an email client proves to be the fastest mode of efficient communication in the world of technology. With the advent of email clients, a user can send mails to any other email client user in any part of the world at the cheap cost of the internet. In the business world, most of the communication takes place through the exchange of emails as they are quick and efficient.

CHAPTER 2

CONCEPT

Email sent from an ISP is handled by that ISP's mail server, which is the equivalent of an electronic post office. The mail comes first to the mail server, is processed, and forwarded towards the destination. Another mail server resides at the destination. It receives all incoming mail and electronically sorts it into mailboxes. The recipient picks up email by using his or her email program to connect to the mail server which request items from the mailbox.

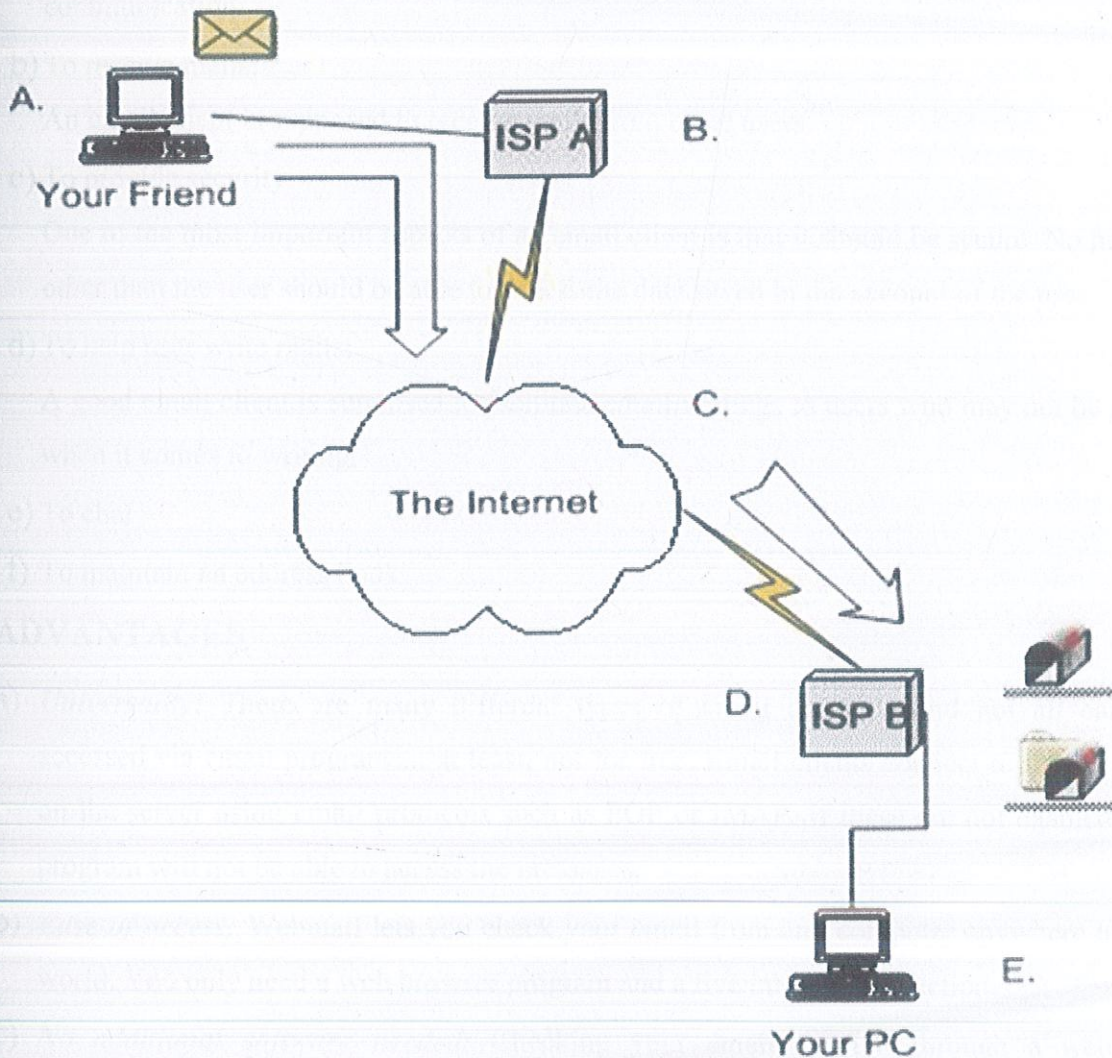


Fig 2.1: Basic view of Email Processing

In a web based email client an email program is built into a website for public use. The server houses many mailbox accounts that users can access through providing a username and password. Once logged into a mailbox, one can write and send email or collect email. Each user maintains a unique email address assigned by the website for his or her exclusive use.

2.1 PURPOSE

The purpose of an email client is

a) To send mails:

The primary task of an email client is to send mails to other users as a medium of communication.

b) To receive mails:

An email client is supposed to receive mails from other users.

c) To provide security:

One of the most important aspects of an email client is that it should be secure. No person other than the user should be able to access the data saved in the account of the user.

d) To help user write mails:

A good email client is supposed to facilitate email writings to users who may not be good when it comes to writing.

e) To chat

f) To maintain an address book

2.2 ADVANTAGES

- a) **Universality:** There are many different types of email accounts and not all can be accessed via email programs... at least, not for free. Email clients connect to the account on the server using email protocols such as POP or IMAP. If these are not enabled, the program will not be able to access the messages.
- b) **Ease of access:** Webmail lets you check your email from any computer anywhere in the world. You only need a web browser program and a live internet connection.
- c) **No additional software needed:** Checking your email account through a webmail interface requires only the web browser program

- d) ***Gentler learning curve:*** It's easier to learn to use email through a webmail interface than a dedicated email client.
- e) ***Customizable and prettier:*** Nowadays, with the launch of themes, webmail interfaces are highly customizable and definitely prettier than email clients.
- f) ***No need to own a computer:*** Webmail accounts can be used from any computer; there is no need to own a computer. This is such a boon for the citizens of underdeveloped countries who can take the advantage of the new communications technology without buying expensive equipments.
- g) ***Updates and addition of new features:*** Webmail interface and features can be modified and tested faster than those in email programs. There is often *no need to download an upgrade and then install it.*
- h) ***Storage space:*** No constrain of storage space for messages and attachments.

2.3 DISADVANTAGES

- a) ***Online Accessibility:*** If one does not have an Internet connection, he will not be able to send or receive email not even saved email.
- b) ***Security and Spam Issues:*** Since web mail is on a central server, a hacker can break in and search thousands or even millions of emails.
- c) Another disadvantage of Web-based email is that all mail sent and received resides on a public mail server, rather than on one's ISP's private mail server.
- d) A Web-based mail server might be less secure than one's ISP's mail server, though no email is truly private unless encrypted.
- e) Since the sole storage location is hosted and controlled by the corporation or institution, the individual does not "have" their email but only has "access" to it and that access is under the sole control of the corporation or institution. This becomes a problem when users lose their email account through hacking or malice and are unable to retrieve the only copies of their stored email.

CHAPTER 3

PROTOCOLS USED TO SUPPORT WEBMAIL

3.1 POP 3

POP3 is an application-layer Internet standard protocol used by local e-mail clients to retrieve e-mail from a remote server over a TCP/IP connection.

This standard protocol is built into most popular e-mail products, such as Eudora and Outlook Express. The POP protocol has been developed through several versions, with version 3 (POP3) being the current standard. It is also built into the Netscape and Microsoft Internet Explorer browsers.

POP3 is designed to delete mail on the server as soon as the user has downloaded it. However, some implementations allow users or an administrator to specify that mail be saved for some period of time. POP can be thought of as a "store-and-forward" service. E-mail clients using POP generally connect, retrieve all messages, store them on the user's PC as new messages, delete them from the server, and then disconnect.

After protocol initiation, Encrypted communication for POP3 is requested, using the STLS command, or by POP3S, which connects to the server using TLS or SSL on well-known TCP port 995.

The original POP3 specification supported only an unencrypted USER/PASS login mechanism. POP3 currently supports several authentication methods to provide varying levels of protection against illegitimate access to a user's e-mail.

3.2 SMTP

Simple Mail Transfer Protocol (SMTP) is an Internet standard for electronic mail (e-mail) transmission across Internet Protocol (IP) networks. SMTP is specified for outgoing mail transport and uses TCP port.

While electronic mail servers and other mail transfer agents use SMTP to send and receive mail messages, user-level client mail applications typically only use SMTP for sending messages to a mail server for relaying.

SMTP is a text-based protocol, in which a mail sender communicates with a mail receiver by issuing command strings and supplying necessary data over a reliable ordered data stream channel, typically a TCP connection.

An SMTP session consists of commands originated by an SMTP client (the initiating agent, sender, or transmitter) and corresponding responses from the SMTP server (the listening agent, or receiver) so that the session is opened, and session parameters are exchanged.

Email is submitted by a mail client, MUA to a mail server, MSA using SMTP. From there, the MSA delivers the mail to its MTA.

3.3 IMAP

Internet message access protocol (IMAP) is one of the two most prevalent Internet standard protocols for e-mail retrieval, the other being the Post Office Protocol (POP). Virtually all modern e-mail clients and mail servers support both protocols as a means of transferring e-mail messages from a server. IMAP is a client/server protocol in which e-mail is received and held for you by your Internet server.

IMAP supports both on-line and off-line modes of operation. E-mail clients using IMAP generally leave messages on the server until the user explicitly deletes them. This and other characteristics of IMAP operation allow multiple clients to manage the same mailbox. Most e-mail clients support IMAP in addition to POP to retrieve messages; however, fewer email services support IMAP. IMAP offers access to the mail storage. Clients may store local copies of the messages

Only if you request to read a specific email message will it be downloaded from the server. You can also create and manipulate folders or mailboxes on the server, delete messages etc. Advantage: As this requires only a small data transfer this works well even over a slow connection such as a modem.

3.4 HTTP

The HTTP protocol is a networking protocol for distributed information systems. HTTP is an application layer protocol designed within the framework of the Internet Protocol Suite.

HTTP functions as a request-response protocol in the client-server computing model. A web browser, for example, may be the client and an application running on a computer hosting a web site may be the server. The client submits an HTTP request message to the server. The server, that performs functions on behalf of the client, returns a response message to the client. The response contains completion status information about the request and may also contain requested content in its message body. Hotmail is a good example of using HTTP as an email protocol.

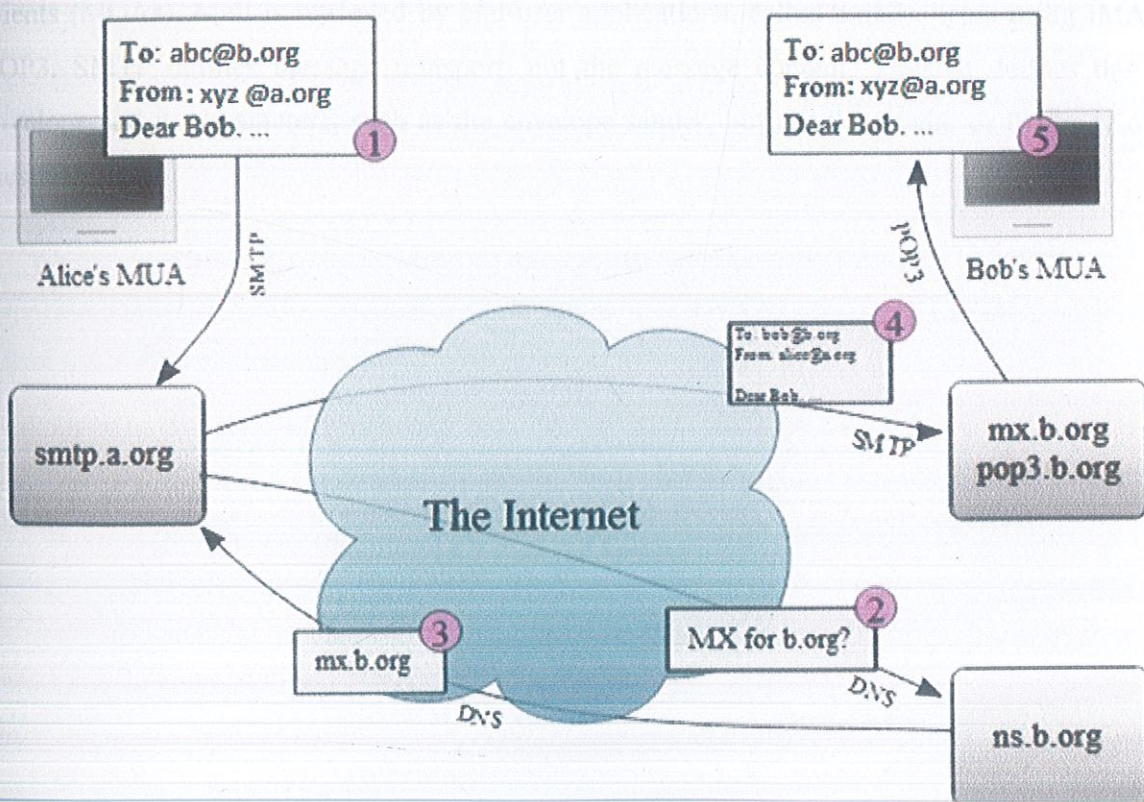


Fig 3.1 Working model with protocols

3.5 MAIL PROCESSING

Email is submitted by a mail client MUA to a mail server MSA using SMTP. The MSA delivers the mail to its mail transfer agent MTA as shown in fig3.1.

The MTA has to locate the target host. It uses the DNS to look up the mail exchanger record (MX record) for the recipient's domain (the part of the address on the right of @). The returned MX record contains the name of the target host. The MTA next connects to the exchange server as an SMTP client.

Once the MX target accepts the incoming message, it hands it to a mail delivery agent MDA for local mail delivery. An MDA is able to save messages in the relevant mailbox format.

Once delivered to the local mail server, the mail is stored for batch retrieval by authenticated mail clients (MUAs). Mail is retrieved by end-user applications, called email clients using IMAP or POP3. SMTP defines message transport, not the message content. Thus, it defines the mail envelope and its parameters, such as the envelope sender, but not the header or the body of the message itself.

CHAPTER 4

LITERATURE SURVEY

An email client, email reader, or more formally MUA, is a computer program used to manage a user's email. The term Webmail (or Web-based e-mail) is used to describe two things. One use of the word is to describe a Webmail client: an email client implemented as a application accessed via a web browser. This article focuses in this use of the term.

The term can refer to any system capable of accessing the user's email mailbox, regardless of it being a mail user agent, a relaying server, or a human typing on a terminal. In addition, a web application that provides message management, composition, and reception functions is sometimes also considered an email client, but more commonly referred to as webmail.

Like most client programs, an email client is only active when a user runs it. The most common arrangement is for a remote MTA server, using a suitable MDA, to add email messages to a client's storage as they arrive. The remote mail storage is referred to as the user's mailbox. The default setting on many UNIX systems is for the mail server to store formatted messages in mailbox, within the user's HOME directory. Of course, users of the system can log-in and run a mail client on the same computer that hosts their mailboxes. In the latter case, the server is not actually remote; it is remote in the most common cases, though.

Emails are stored in the user's mailbox on the remote server until the user's email client requests them to be downloaded to the user's computer, or can otherwise access the user's mailbox on the possibly remote server. The email client can be set up to connect to multiple mailboxes at the same time and to request the download of emails either automatically, such as at pre-set intervals, or the request can be manually initiated by the user.

Webmail has several advantages, including an ability to send and receive email away from the user's normal base using a web browser, thus eliminating the need for an email client.

No matter which type of client you're using, it generally does four things:

- a) Shows you a list of all of the messages in your mailbox by displaying the message headers. The header shows you who sent the mail, the subject of the mail and may also show the time and date of the message and the message size.

- b) Lets you select a message header and read the body of the e-mail message.
- c) Let's you create new messages and send them. You type in the e-mail address of the recipient and the subject for the message, and then type the body of the message.
- d) Lets you add attachments to messages you send and save the attachments from messages you receive.

A major advantage of webmail is that the individual's email is available everywhere there is an internet connection and a browser and the individual does not need a computer with their mail application installed in it. With webmail the users' email is usually backed up with multiple redundancy and corporations and institutions usually provide extremely reliable service as well as excellent spam filtering services.

Some websites are dedicated to providing email services, including Hotmail, Gmail, AOL, and Yahoo; but there are many internet service providers which provide webmail services as part of their internet service package. The main limitations of webmail are that user interactions are subject to the website's operating system and the general inability to download email messages and compose or work on the messages offline, although Gmail does offer Offline Gmail through the installation of Gears. The advantage of webmail provided by a regular mail server is that email remains on the mail server until the user can return to the base computer, when they can be downloaded. Users may be able to choose whether to leave a copy of the email on the server for a backup.

A major disadvantage of webmail is that the hosting corporation or institution retains control over the individual's email as it is performing a storage function in addition to the service function. Since the sole storage location is hosted and controlled by the corporation or institution the individual does not "have" their email but only has "access" to it and that access is under the sole control of the corporation or institution. This becomes a problem when a user loses their email account through hacking or malice and is unable to retrieve the only copies of their stored email. Webmail will also be affected by the speed and quality of the internet connection and this may be a problem for dial-up connection users. A major advantage of webmail is that the individual's email is available everywhere there is an internet connection and a browser and the individual does not need a computer with their mail application installed in it. With webmail the users' email is usually backed up with multiple redundancy and corporations and institutions usually provide extremely reliable service as well as

excellent spam filtering services. Privacy concerns have been raised about webmail as corporations are storing large amounts of personal information.

A good email client has the following features:

- a) Ability to send and receive email quickly.
- b) An address book in which you can store email addresses of your friends, family members and others.
- c) The ability to easily attach other non-text based files using MIME. You can, thus, send pictures, sounds etc. with your email.
- d) Add signatures to outgoing emails.

4.1 DIFFERENT EMAIL CLIENTS AND THEIR FEATURES

No matter what email client you are using, it generally does four things:

- a) Shows you a list of all of the messages in your mailbox by displaying the message headers. The header shows you who sent the mail, the subject of the mail and may also show the time and date of the message and the message size.
- b) Lets you select a message header and read the body of the e-mail message.
- c) Lets you create new messages and send them. You type in the e-mail address of the recipient and the subject for the message, and then type the body of the message.
- d) Lets you add attachments to messages you send and save the attachments from messages you receive.

4.1.1 Yahoo!mail.com

- a) Yahoo offers no storage limits, so you can use your email as personal archive.
- b) You can use the reply bar in your email messages to respond instantly to FACEBOOK and email messages.

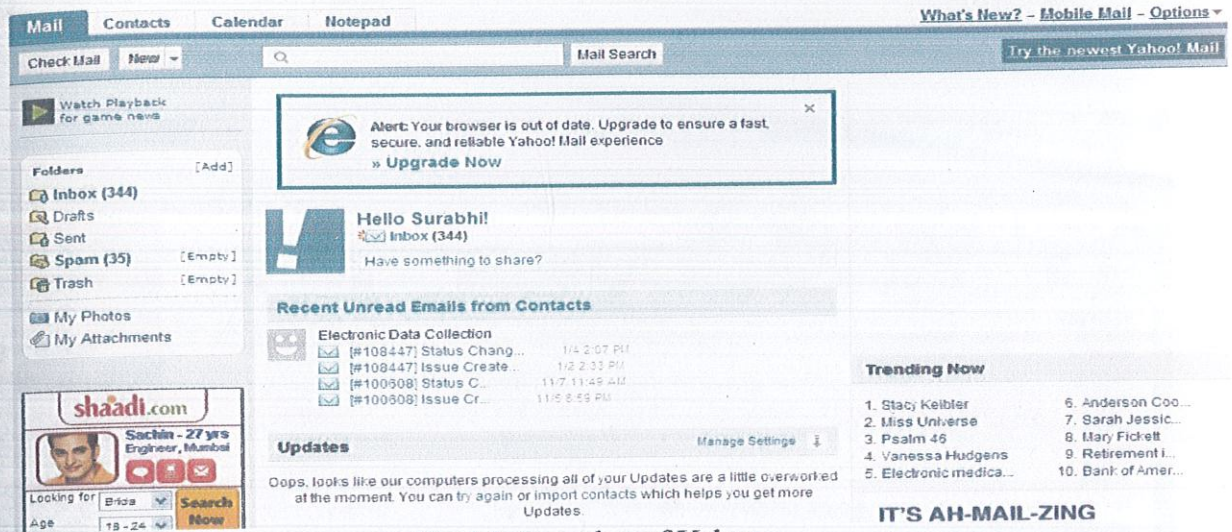


Fig 4.1: Snapshot of Yahoo

4.1.2 Gmail.com

- a) Gmail utilizes Google search technology to organize, find, and retrieve the current messages in your inbox and the messages you have archived.
- b) Gmail gives users 1 GB of storage space for email, so most people will never have to delete another message.
- c) Gmail has also added the ability to import contact email addresses from another email programs.
- d) Gmail added signatures in July, 2004.
- e) Gmail Notifier automatically checks for new messages every two minutes. With it you can see a brief section of text from up to 30 messages and select a sound to indicate you have a new mail.
- f) The Mail Fetcher lets you fetch and download messages from upto 5 another accounts. Mail fetcher will check all of the accounts regularly so that mail from them appears automatically in Gmail.
- g) By creating a contact group, you can quickly send emails to everyone in the group.
- h) It virus scans every attachment you send. Every attachment you receive is scanned twice.

- i) You can chat with a friend or in groups. Send instant messages. Take advantage of mail with video and voice.

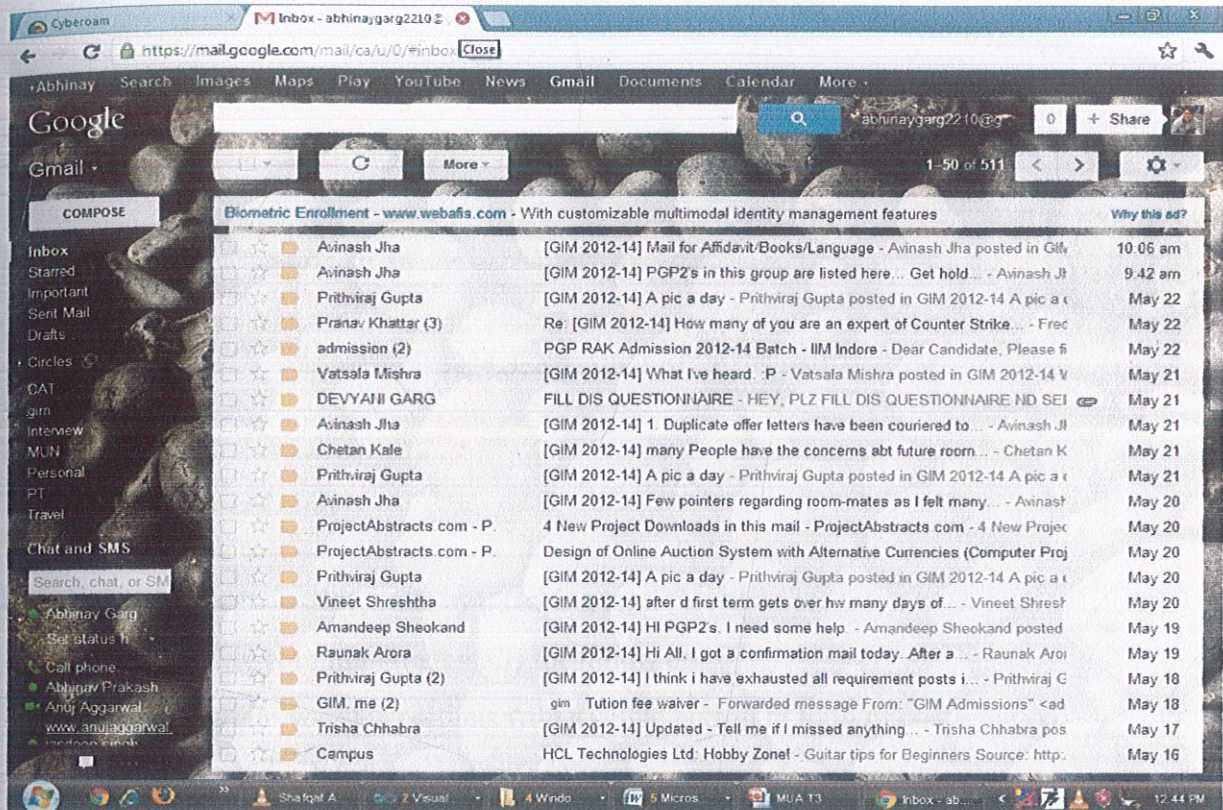


Fig 4.2: Snapshot of Gmail

4.1.3 Rediffmail.com

- The inbuilt virus scanner scans every attachment in your mailbox for known viruses, uncovering any virus threat and cleaning up infected attachments.
- You can import address book from another email client.
- New Technology is being used to control the spam.
- With a single click you can choose to download all attachments you have received.

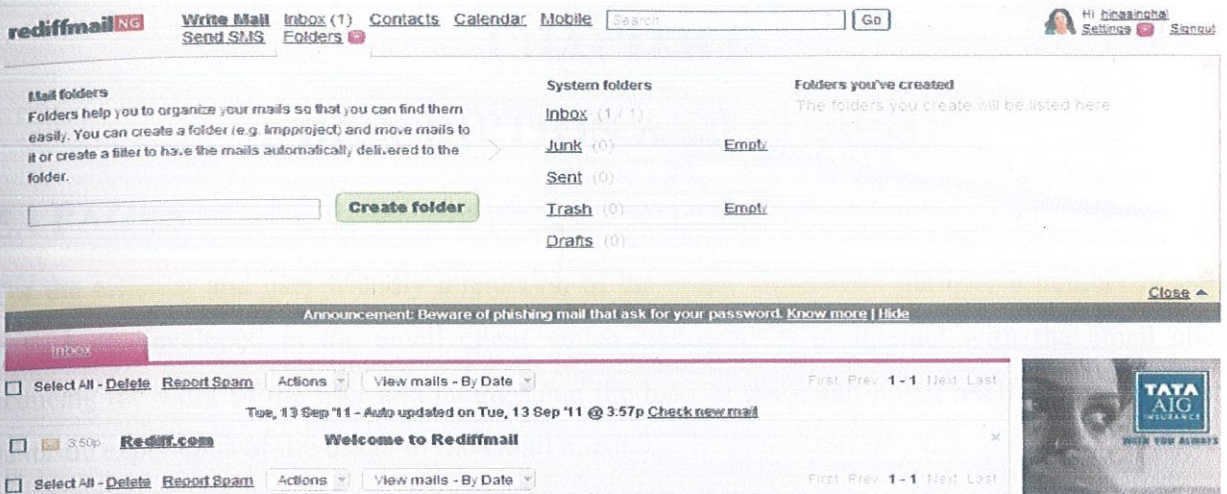


Fig4.3: Snapshot of Rediffmail

4.1.4 Indiatimesmail.com

- a) One can organize his mails by attaching tags.
- b) One can flag your important mails for follow up.
- c) One can group your interactions with a single person in form of conversation.
- d) One can create a briefcase and upload your important documents.

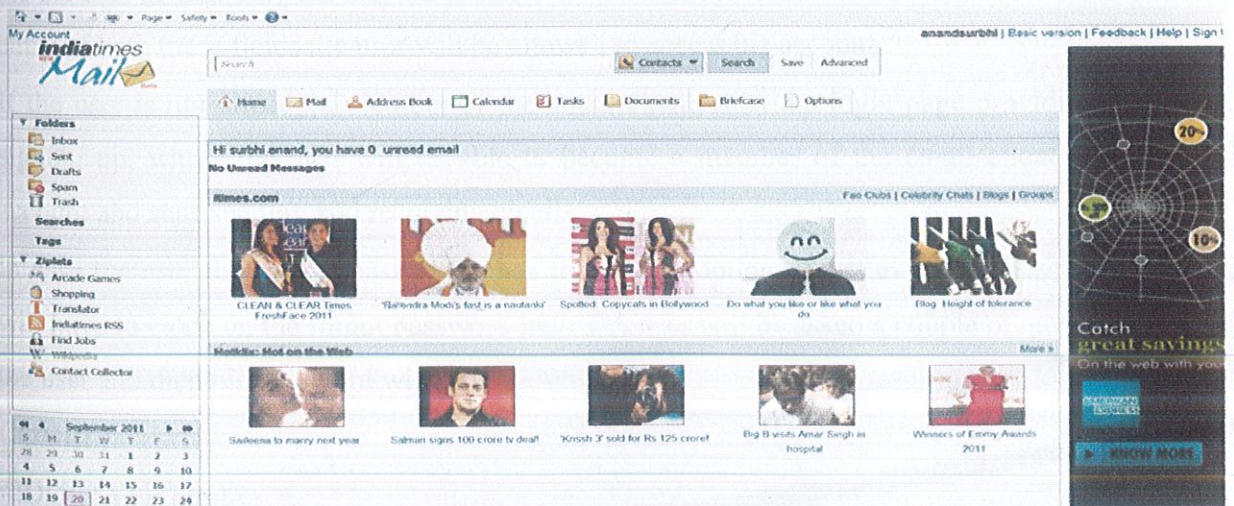


Fig 4.4: Snapshot of Indiatimes

CHAPTER 5

DESCRIPTION AND CODING

5.1 PAGES

For the efficient and user friendly interaction of the email client with the user a number of pages have been developed in the email client which enable users to interact with the email client, reducing the work of the user and taking upon the load of the email client itself to give the user a fantastic experience of the usage of the email client.

These pages have been developed in HTML with Javascript encoded in them. These HTML pages are the front end of the email client that will be visible to the user of the email client who would only be able to see the designs of these pages and with a simple click on the links he will be able to access his emails and send them further.

5.1.1 LOGIN PAGE

This is the first page that will be visible to the user when he enters the URL of the email client in the Web browser. This page consists of a welcome message along with fields asking for the user's username and his password. This username acts as a unique name that has been provided to the user in order to identify the user and enable him to access his emails. The password provides a protection for the user so that only the user can be able to access his mails and no one else. Only on the correct entry of both these fields the user will be allowed to access his account.

If the user is not already a member of the email client, then the login page provides an option of signing up, where the user will be able to become a member of the email client and create his account and start sending and receiving emails.

In case the user forgets his password, then there is an option of retrieving the password. The user will have to click on the forgot password link. The user will be asked a couple of questions to verify the user and then his password will be retrieved.

CODE:

```
<!DOCTYPE html>
```

```
<html>
```



```

<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>JSP Page</title>
</head>
<body background="C:\Documents and Settings\Admin\Desktop\email.jpeg">
  <h1>WELCOME TO THE WEBMAIL CLIENT</h1>
  <form action="checkuser" method="post">
    <table border="0">
      <thead>
        <tr>
          <th colspan="2">Please Login</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>User Name</td>
          <td><input type="text" name="username" value="" /></td>
        </tr>
        <tr>
          <td>Password</td>
          <td><input type="password" name="userpass" value="" /></td>
        </tr>
        <tr>
          <td colspan="2"><button type="submit">Login</button></td>
          <td><a href="SignUp.jsp"> New User? </a>
          <td>
        </tr>
      </tbody>
    </table>
  </form>

```



```
</tr>
</tbody>
</table>
</form>
</body>
</html>
```

5.1.2 SIGN UP PAGE

This page has been created to allow a new user or an existing user to create his or her account on the email client so as to enable him to send and receive emails. To sign up on the email client, the user has to furnish certain information that will be recorded in the database. The user will be asked his desired username, provided it is available, and password. Along with this the user will specify his first name, last name, address, birth date, nationality and phone number of the user. In case the user forgets his password, this information will be helpful in determining the identity of the user.

CODE:

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Sign Up Page</title>
  </head>
  <body>
    <form action="InsertUser" method="post">
      <table border="0">
        <thead>
          <tr>
            <th colspan="2"> Fill up the following fields</th>
```



```
</tr>
</thead>
<tbody>
<tr>
<td>User name </td>
<td><input type="text" name="username" value="" /></td>
</tr>
<tr>
<td>Password </td>
<td><input type="text" name="userpass" value="" /></td>
</tr>
<tr>
<td> First Name </td>
<td><input type="text" name="fname" value="" /></td>
</tr>
<tr>
<td>Last Name</td>
<td><input type="text" name="lname" value="" /></td>
</tr>
<tr>
<td>Address </td>
<td><input type="text" name="address" value="" /></td>
</tr>
<tr>
<td>Birhday</td>
<td><input type="text" name="birthday" value="" /></td>
</tr>
<tr>
<td>Nationality</td>
```



```

        <td><input type="text" name="nationality" value="" /></td>
</tr>
    <tr>
        <td>Phone</td>
        <td><input type="text" name="phone" value="" /></td>
    </tr>
</tbody>
<tr>
    <td>
        <button type="submit">Sign Up</button>
    </td>
</tr>
</table>
</form>
</body>
</html>

```

6.1.3 INBOX

This page displays the messages that have been received by the user. On clicking on different messages the user will be able to read the emails that have been sent to him. Along with this, the user has buttons on this page that will allow him to jump on different sections of the email client.

CODE:

```

<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=windows-1252"/>
    <title>home page</title>
</head>

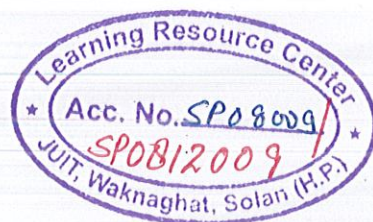
```



```

<body>
<%
String myname = (String)session.getAttribute("username");
    out.println("Welcome "+username+"...");
%>
<table cellspacing="3" cellpadding="2" border="0" width="50%">
  <tr>
    <td>
      <form name="composemail" method="post" action="email.jsp">
        <input type="submit" name="submit" value="Compose Mail"/>
      </form>
    </td>
  </tr>
  <tr>
    <td>
      <form name="inbox" action="popservlet" method="post">
        <input type="submit" name="submit" value="inbox"/>
      </form>
    </td>
  </tr>
  <tr>
    <td>
      <form name="sentmail" method="post" >
        <input type="submit" name="sent" value="sent mail"/>
      </form>
    </td>
  </tr>
</table>
</body>
</html>

```



5.1.4 COMPOSE

On this page, the user will be able to send mails to other users. The user has to specify the email id of the other user he wishes to write to. Along with the email id, the user will have to mention the subject of the email that he is sending. The user will then write the message in the message field. On clicking the send button, the user will be able to send the email. The user will also be able to send emails to other users using the cc and the bcc functions.

CODE:

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <h1>Send Your Email</h1>
    <div align="center">
      <form action="sendmail" method="post">
        <b>To</b><input type="text" name="to" value="">
        <br>
        <b>Subject</b>
          <input type="text" name="subject">
        <b>CC</b>
          <input type="text" name="cc" value="">
        <b>bcc</b>
          <input type="text" name="cc" value="">
        <b>Enter Your Message</b>
        <br/>
        <textarea cols="50" name="msg" rows="20"></textarea>
        <br>
```



```
<b>Send</b>
    <input type="submit" value="Send">
</form>
</div>
</body>
</html>
```

5.2 Servlets

Along with the JSP/HTML pages servlets have been made that work as a back end for the project. These servlets are not visible to the user and are called from JSP pages to execute the queries and commands of the user.

5.2.1 CHECKUSER

This servlet is used to authorize access to the user. This servlet checks the username and password of the user that has been provided on the login page and validates them from the database. If the servlet is able to find the username with the correct password then the servlet will login the user into his account where he can check his mails else the user will be redirected back to the login page.

CODE:

```
package org.servlet;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import javax.servlet.RequestDispatcher;
```



```

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import org.dbconn.DatabaseConnection;

public class CheckUser extends HttpServlet
{

    /**
     * Processes requests for both HTTP
     * <code>GET</code> and
     * <code>POST</code> methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try
        {
            Connection connection=new DatabaseConnection().getConnection();

```



```

        PreparedStatement preparedStatement=connection.prepareStatement("select * from users
where username=? and password=?");
preparedStatement.setString(1, request.getParameter("username"));
preparedStatement.setString(2, request.getParameter("userpass"));
ResultSet resultSet=preparedStatement.executeQuery();
        if(resultSet.next())
        {
            HttpSession hs=request.getSession();
            hs.setAttribute("username", request.getParameter("username"));
            hs.setAttribute("password", request.getParameter("userpass"));
            RequestDispatcher rd=request.getRequestDispatcher("inbox.jsp");
            rd.forward(request, response);
        }
        else
        {
            response.sendRedirect("loginpage.jsp");
        }
    }
    catch(Exception e)
    {

    }
    finally
    {
        out.close();
    }
}

```



```
// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left  
to edit the code.">
```

```
/**
```

```
* Handles the HTTP <code>GET</code> method.
```

```
* @param request servlet request
```

```
* @param response servlet response
```

```
* @throws ServletException if a servlet-specific error occurs
```

```
* @throws IOException if an I/O error occurs
```

```
*/
```

```
@Override
```

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
```

```
    throws ServletException, IOException
```

```
{
```

```
    processRequest(request, response);
```

```
}
```

```
/**
```

```
* Handles the HTTP <code>POST</code> method.
```

```
* @param request servlet request
```

```
* @param response servlet response
```

```
* @throws ServletException if a servlet-specific error occurs
```

```
* @throws IOException if an I/O error occurs
```

```
*/
```

```
@Override
```

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
```

```
    throws ServletException, IOException
```

```
{
```

```
    processRequest(request, response);
```

```
}
```



```

/**
 * Returns a short description of the servlet.
 * @return a String containing servlet description
 */
@Override
public String getServletInfo()
{
    return "Short description";
} // </editor-fold>

```

5.2.2 INSERT SERVLET

This servlet is used to insert a new user to the email client and store his details in the database. When the person clicks on the signup button in the sign up page, this servlet will be called using post method. Once this servlet is executed, if successful, it will take the client on the login page and if unsuccessful, it will come back to the sign up page for the user to sign up again.

CODE:

```

package org.servlet;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;

```



```

import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import org.dbconn.DatabaseConnection;

/**
 * @author Admin
 */
public class InsertUser extends HttpServlet
{

    /**
     * Processes requests for both HTTP
     * <code>GET</code> and
     * <code>POST</code> methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try
        {
            Connection connection=new DatabaseConnection().getConnection();

```



```

        PreparedStatement preparedStatement=connection.prepareStatement("insert into users
values ()");
preparedStatement.setString(1, request.getParameter("username"));
preparedStatement.setString(2, request.getParameter("userpass"));
preparedStatement.setString(3, request.getParameter("fname"));
preparedStatement.setString(4, request.getParameter("lname"));
preparedStatement.setString(5, request.getParameter("address"));
preparedStatement.setString(6, request.getParameter("birthday"));
preparedStatement.setString(7, request.getParameter("nationality"));
preparedStatement.setString(8, request.getParameter("phone"));

int s=preparedStatement.executeUpdate();
if(s > 0)
{
    /*HttpSession hs=request.getSession();
    hs.setAttribute("Username", request.getParameter("username"));
    hs.setAttribute("Password", request.getParameter("userpass"));
    hs.setAttribute("First name", request.getParameter("fname"));
    hs.setAttribute("Last name", request.getParameter("lname"));
    hs.setAttribute("Address", request.getParameter("address"));
    hs.setAttribute("Birthday", request.getParameter("birthday"));
    hs.setAttribute("Nationality", request.getParameter("nation"));
    hs.setAttribute("Phone", request.getParameter("phone"));*/
    RequestDispatcher rd=request.getRequestDispatcher("loginpage.jsp");
    rd.forward(request, response);
}
else
{
    response.sendRedirect("SignUp.jsp");
}

```



```

    }
}
/*
 * TODO output your page here out.println("<html>");
 * out.println("<head>"); out.println("<title>Servlet
 * InsertUser</title>"); out.println("</head>");
 * out.println("<body>"); out.println("<h1>Servlet InsertUser at " +
 * request.getContextPath () + "</h1>"); out.println("</body>");
 * out.println("</html>");
 */

finally {
    out.close();
}
}
}

```

5.2.3 READMAIL SERVLET

This servlet is required to read the mails that have been received by the user on his account. After logging in the email client, the user will be able to see his received emails and the author of those emails. On clicking those emails, the user will be able to see and read his message. At this time this servlet will load the emails that the user wishes to read.

CODE:

```

package org.mail;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.Properties;
import java.util.logging.Level;

```



```
import java.util.logging.Logger;
import javax.mail.Address;
import javax.mail.FetchProfile;
import javax.mail.Folder;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Provider;
import javax.mail.Session;
import javax.mail.Store;
import javax.mail.internet.InternetAddress;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
public class readMailServlet extends HttpServlet
```

```
{
```

```
    /**
```

```
     * Processes requests for both HTTP GET and POST methods.
```

```
     * @param request servlet request
```

```
     * @param response servlet response
```

```
     * @throws ServletException if a servlet-specific error occurs
```

```
     * @throws IOException if an I/O error occurs
```

```
    */
```

```
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
```

```
    throws ServletException, IOException
```

```
{
```

```
    response.setContentType("text/html;charset=UTF-8");
```



```

PrintWriter out = response.getWriter();
try
{

    Properties prop=new Properties();
    prop.put("mail.transport.protocol", "pop");
    Session ss=Session.getInstance(prop);
    Provider p[]=ss.getProviders();
    Store st=ss.getStore(p[4]);
    Try
    {
        st.connect("db2admin",110,"@abc.com","1234");
        Folder rootFolder=st.getDefaultFolder();
        Folder myInbox=rootFolder.getFolder("INBOX");
        myInbox.open(Folder.READ_ONLY);
        Message [] listOfMessage=myInbox.getMessages();
        FetchProfile profile=new FetchProfile();
        profile.add(FetchProfile.Item.ENVELOPE);
        myInbox.fetch(listOfMessage, profile);
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet readMailServlet</title>");
        out.println("</head>");
        out.println("<body>");
        for(int x=0;x<listOfMessage.length;x++)
        {
            Address [] addList=listOfMessage[x].getFrom();
            if(addList!=null)
            {

```



```

        if(addList.length>0)
            out.print("The Sender Is "+((InternetAddress)addList[0]).getAddress());
            out.print("<br>The Subject Of
                Message="+listOfMessage[x].getSubject()+"<br>");
        }
    else
        continue;
}
// out.println("<h1>Servlet readMailServlet at " + request.getContextPath () + "</h1>");
out.println("</body>");
out.println("</html>");
}
catch(MessagingException me)
{ out.print("The Real Message Exception="+me.getMessage()+me.toString()); }
}
catch(Exception e)
{
    out.print("The final Message="+e.getMessage()+e.toString());
    Logger.getLogger(readMailServlet.class.getName()).log(Level.SEVERE, "Message
Exception "+e.getMessage(), e);
}
finally
{
    out.close();
}
}

```


5.2.4 SEND MAIL

This servlet is required to send emails. When the user wishes to send emails to other users, he will specify the receiver of that email, subject and write the email. When the user clicks on the send button, this servlet will read the email and send it to the server from where further on the server will send the email to other users.

CODE:

```
package org.servlet;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.Date;
import java.util.Properties;
import javax.mail.Message;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class sendmail extends HttpServlet {
    /**
     * Processes requests for both HTTP GET and POST methods.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
```



```
* @throws IOException if an I/O error occurs
```

```
*/
```

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)  
throws ServletException, IOException
```

```
{  
    response.setContentType("text/html;charset=UTF-8");  
    PrintWriter out = response.getWriter();  
    try  
    {  
        HttpSession hs=request.getSession(false);  
        String from=(String)hs.getAttribute("username");  
        String to=request.getParameter("to");  
        //String from=request.getParameter("from");  
        String msg=request.getParameter("msg");  
        Properties prop=new Properties();  
        prop.put("mail.transport.protocol", "smtp");  
        prop.put("mail.smtp.host", "localhost");  
        prop.put("mail.smtp.port", "25");  
        Session ss=Session.getInstance(prop);  
        Message m=new MimeMessage(ss);  
        m.setFrom(new InternetAddress(from));  
        m.setRecipients(Message.RecipientType.TO, InternetAddress.parse(to));  
        m.setSentDate(new Date());  
        m.setSubject(request.getParameter("subject"));  
        m.setText(msg);  
        Transport.send(m);  
        out.print("Message was sent successfully");  
    }  
}
```



```

catch(Exception e)
{
    e.printStackTrace();
}
finally
{
    out.close();
}

```

5.2.5 MESSAGE VIEW FROM SERVER

This servlet is required to load the emails from the server using a POP3 client. In order to read the emails from the server this servlet will use the POP3 protocol and load the emails from the server for the users to read.

CODE:

```

package org.servlet;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.Properties;
import javax.mail.Folder;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Provider;
import javax.mail.Session;
import javax.mail.Store;
import javax.mail.URLName;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;

```



```

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class messageViewFromServer extends HttpServlet
{

    /**
     * Processes requests for both HTTP GET and POST methods.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try
        {
            Properties prop=new Properties();
            prop.put("mail.transport.protocol", "pop");
            Session ss=Session.getInstance(prop,null);
            Provider p[]=ss.getProviders();
            //ss.setProvider(p[4]);
            //URLName urlname=new URLName("pop3://messageviewfromserver:110");
            try
            {
                Store st=ss.getStore(p[4]);

```



```

st.connect("localhost",110,"root","a22101989");
Folder rootFolder=st.getDefaultFolder();
Folder inbox=rootFolder.getFolder("INBOX");
inbox.open(Folder.READ_ONLY);
Message [] allmsg=inbox.getMessages();
for(int i=0;i<allmsg.length;i++)
{
    out.print("Id:"+i+" subject "+allmsg[i].getSubject()+"<br>");
}
inbox.close(false);
st.close();
}
catch(MessagingException me)
{
    out.print("The Exception Is "+me.getMessage()+" "+me.toString());
}

/* TODO output your page here
out.println("<html>");
out.println("<head>");
out.println("<title>Servlet messageViewFromServer</title>");
out.println("</head>");
out.println("<body>");
out.println("<h1>Servlet messageViewFromServer at " + request.getContextPath () +
"</h1>");
out.println("</body>");
out.println("</html>");
*/
}

```



```

catch(Exception e)
{
    out.print("The root cause is "+e.getMessage()+" "+e.toString());
}
finally {
    out.close();
}
}

```

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

```

/**
 * Handles the HTTP <code>GET</code> method.
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
    processRequest(request, response);
}

```

```

/**
 * Handles the HTTP <code>POST</code> method.
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs

```



```

    * @throws IOException if an I/O error occurs
    */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        processRequest(request, response);
    }
    /**
     * Returns a short description of the servlet.
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo()
    {
        return "Short description";
    } // </editor-fold>
}

```

5.2.6 SEND ATTACHMENT

This servlet is used to attach files along with the emails that the user wishes to send. For example the user wishes to send an image along with the message that he has written, then the user can use this feature in order to send the emails. An attach file button will be available at the compose mail where the user can specify the location of the file that he wishes to attach.

CODE:

```

import java.io.IOException;
import java.io.PrintWriter;

```



```
import java.util.Date;
import java.util.Properties;
import javax.activation.DataHandler;
import javax.activation.FileDataSource;
import javax.mail.Message;
import javax.mail.Multipart;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeBodyPart;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
public class sendAttachmentServlet extends HttpServlet
{
```

```
/**
```

```
 * Processes requests for both HTTP GET and POST methods.
```

```
 * @param request servlet request
```

```
 * @param response servlet response
```

```
 * @throws ServletException if a servlet-specific error occurs
```

```
 * @throws IOException if an I/O error occurs
```

```
 */
```

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
```

```
throws ServletException, IOException
```



```

{
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try
    {
        Properties prop=new Properties();
        prop.put("mail.transport.protocol", "smtp");
        prop.put("mail.smtp.host", "db2admin");
        prop.put("mail.smtp.port", "25");
        Session ss=Session.getInstance(prop);
        Message msg=new MimeMessage(ss);
        msg.setFrom(new InternetAddress("ss@yahoo.co.in"));
        msg.setSubject("Sending photo as attachment");
        msg.setRecipients(Message.RecipientType.TO, InternetAddress.parse("@db2admin.com"));
        msg.setSentDate(new Date());

        Multipart mailBody=new MimeMultipart();
        MimeBodyPart mainBody=new MimeBodyPart();
        mainBody.setText("Hello I am sending you a photo as an attachment");
        mailBody.addBodyPart(mainBody);

        FileDataSource fds=new FileDataSource("c:\\vodaphone.jpg");
        MimeBodyPart attach=new MimeBodyPart();
        attach.setDataHandler(new DataHandler(fds));
        attach.setFileName(fds.getName());
        mailBody.addBodyPart(attach);
        msg.setContent(mailBody);
        msg.saveChanges();
        Transport.send(msg);
    }
}

```



```

        out.print("Message was send successfully");
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    finally
    {
        out.close();
    }
}

```

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

```

/**
 * Handles the HTTP <code>GET</code> method.
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
    processRequest(request, response);
}

```


5.2.7 DATABASE CONNECTION

This servlet will connect the project to the database using a jdbc connection and using the dbconn package in java. To authenticate the user, we require verifying his username and password which is stored in the database. For the verification of the same, a CHECK USER servlet has been deployed, but that servlet can only be used if the application is able to maintain a connection with the database. This servlet of DATABASE CONNECTION will connect the application to the database, and once connected, the application can not only retrieve data from the database, but also the application will be able enter the data, i.e. on signing up, into the database for further usage.

CODE:

```
package org.dbconn;
import java.sql.Connection;
import java.sql.DriverManager;
public class DatabaseConnection {
    Connection conn;
    public Connection getConnection()
    {
        try
        {
            Class.forName("com.mysql.jdbc.Driver");
            String username="root";
            String password="a22101989";
            String url="jdbc:mysql://localhost:3306/emailclient";
            conn=DriverManager.getConnection(url, username, password);
        }
    }
    catch (Exception e)
    {
    }
    return conn;
}
}
```


CHAPTER 6

FUTURE PROSPECTS

This project can further be modified to take a form of a commercial email client that can be used by commercial purposes such as an email network in an organization with separate email accounts with each of the employee of that organization and by domestic users for its basic functionality of sending and receiving emails along with the other features that have been and will have to be implemented in this application.

For this purpose the project will have to be more efficient in terms of cost, space and time. The technology to be used has to be the latest and proficient technology. A number of other features will also be included in this application to utilize this project on a commercial level.

BIBLIOGRAPHY

1. Partridge, Craig (2008), THE TECHNICAL DEVELOPMENT OF INTERNET EMAIL, *IEEE Annals of the History of Computing*. ISSN 1934-1547
2. Brownlow, Mark (January, 2009), EMAIL AND WEBMAIL STATISTICS, *Email Marketing Reports*.
3. OFFLINE GMAIL (January 27, 2009), Official Gmail Blog.
4. comp.mail.misc Webex Announcement (August 8, 1995), COMP.MAIL.MISC WEBEX ANNOUNCEMENT.
5. InfoWorld (1995), LOTUS CC: MAIL TO GET BETTER SERVER, MOBILE ACCESS, p. 8.
6. Information Week (1995), SURFING THE NET FOR E-MAIL.
7. Business Wire (2000), RECOURSE TECHNOLOGIES APPOINTS VICE PRESIDENT OF ENGINEERING.
8. WWW Mail Client 1.00 ANNOUNCE: WWW MAIL CLIENT 1.00.
9. WEBMAIL - SOURCE CODE RELEASE
10. CV, Dr. Burton Rosenberg CV, BURTON ROSENBERG"
11. Network World (1995), "LOTUS READIES CC: MAIL-WEB HOOKS", pp. 1, 55.
12. PR Newswire (1995), LOTUS ANNOUNCES CC: MAIL FOR THE WORLD WIDE WEB.
13. InfoWorld (1995), CC: MAIL USERS WILL GET E-MAIL THROUGH WEB, p. 12.
14. Network World (1995), MORE FROM LOTUS: X.500 AND THE WEB, p. 10.
15. EMUmail website.
16. SMTP.com SMTP.com
17. Hunt, C (2003). SENDMAIL COOKBOOK. ISBN 0-596-00471-0.
18. Hughes, L (1998). INTERNET E-MAIL PROTOCOLS, STANDARDS AND IMPLEMENTATION. ISBN 0-89006-939-5.

19. Johnson, K (2000). INTERNET EMAIL PROTOCOLS: A DEVELOPER'S GUIDE.. ISBN 0-201-43288-9.
20. Loshin, P (1999). ESSENTIAL EMAIL STANDARDS: RFCS AND PROTOCOLS MADE PRACTICAL. ISBN 0-471-34597-0.
21. Rhoton, J (1999). PROGRAMMER'S GUIDE TO INTERNET MAIL: SMTP, POP, IMAP, AND LDAP. ISBN 1-55558-212-5.
22. Wood, D (1999). PROGRAMMING INTERNET MAIL. ISBN 1-56592-479-7.

WEB REFERENCES

1. **IMAP:** http://en.wikipedia.org/wiki/Internet_Message_Access_Protocol
2. **SMTP:** <http://www.faqs.org/rfcs/rfc821.html>
3. **MTA:** http://en.wikipedia.org/wiki/Message_transfer_agent
4. **DNS:** http://en.wikipedia.org/wiki/Domain_Name_System
5. **Mail listing:** <http://www.gnu.org/software/mailman/index.html>
6. **Courier:** <http://www.courier-mta.org/>