OFFLINE MAIL HANDLING

PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the degree of

P-1169

BACHELOR OF ENGINEERING IN INFORMATION TECHNOLOGY

OF THE BHARATHIAR UNIVERSITY, COIMBATORE

Submitted by

V.N.KARTHIKEYAN (0027S0084)

S.SURESH KUMAR (0027S00115)

Under the valuable Guidance of

Mrs.S.Devaki , M.S.,

Assistant Professor, CSE DEPARTMENT

MARCH 2004

Department of Information Technology

Kumaraguru College of Technology

(Affiliated to Bharathiar University)

COIMBATORE – 641 006

KUMARAGURU COLLEGE OF TECHNOLOGY
(Affiliated to Bharathiar University)
COIMBATORE – 641 006, TAMIL NADU, INDIA
Approved by AICTE, New Delhi - Accredited by NBA

Department of Information Technology

CERTIFICATE

This is to certify that the project entitled

"OFFLINE MAIL HANDLING"

has been submitted by

V.N.Karthikeyan (0027S0084)

S.Suresh Kumar (0027S0115)

in partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Information Technology of the Bharathiar University, Coimbatore – 641 046 during the academic year 2003-2004

[illegible]
Dr.S.Thangasamy Ph.D.,
(Head of the Department)

[illegible]
Mrs.S.Devaki M.S.,
(Project Guide)

Submitted for the university examination held on 26/03/04

[illegible] 26/3
Internal Examiner

[illegible]
External Examiner

Declaration

We,

V.N.Karthikeyan 0027S0084

S.Sureshkumar 0027S0115

declare that the project entitled "Offline Mail Handling", done by us and to the best of our knowledge, a similar work has not been submitted earlier to the Bharathiar University or any other institution, for fulfillment of the requirement of the course study.

This project report is submitted on the partial fulfillment of the requirement for all awards of the degree of Bachelor Of Information Technology of Bharathiar University.

[illegible]

Place: Coimbatore. V.N.Karthikeyan

[illegible]

Date : 24.3.04 S.Sureshkumar

ACKNOWLEDGEMENT

We express our profound respect and sincere gratitude to **Dr.K.K.Padmanaban,Ph.D., Principal, Kumaraguru College of Technology,** Coimbatore, for his kind co-operation in allowing us to take up this project work.

We record our sincere thanks to, **Dr.S.Thangasamy, Ph.D.,** the **Head of the Department, Computer Science and Engineering**, **Kumaraguru College of Technology,** Coimbatore, for encouraging us to take up this project and for his constant encouragement and support to complete this project.

We express our sincere thanks to **Mrs.S.Devaki M.S., Assistant Professor**, **Department of Computer Science and Engineering,** the project **Co-ordinator** and our **guide**, without whose motivation and guidance we could not have completed this project successfully.

We are extremely grateful to our Class Advisor **Ms. P. Sudha, B.E.,** for her support throughout the duration of the project.

We also express our thanks to all other faculty members and technical supporting staff of the Department of Computer Science and Engineering and the Department of Information Technology for their support and timely help.

We also take this opportunity to thank our parents and our friends for their constant support and encouragement.

SYNOPSIS

This project entitled "**Offline Mail Handling**" aims to retrieve the e-mails Id-s specified in the application and helps in avoiding the online checking of the mails. In our application the system is made to intimate the user, after the time interval specified by the user to establish the dialup. After the dialup has been established, it will logon to the e-mail accounts of the specified Mail-IDs.

This connection is used to copy the contents of the inbox of the logged mail accounts into the system's local disk and stored in the path specified by the user. If there are any emails that are prepared to send to others, then they will be sent through the established connection before the connection is being disconnected. Whenever the user wants to read the message they can see the emails from the system's local disk without connecting with the internet.

CONTENTS

1.INTRODUCTION

New advantages in the technology have allowed companies to be more flexible when it comes to learning. Now a days Internet becomes the essential thing in our daily life. Every thing become so fast by using internet, but it is costly to use in our country. The increased usage of internet also results in the increased traffic in the network connection. The Logical System aims at the development of a solution to the industry and as well as for the normal public to save the time and avoiding manual working in wasting the time for doing same process frequently.

This software is being developed targeting the creation of an awareness among the people that people do not need to waste their times simply connecting and disconnecting the internet even for checking their mails online. So in this project we are going to create a method for downloading the mails in a short while and save them in the local disk. Whenever the user finds free time they can read their mails offline from the local disk. This saves the cost and the traffic in the network is being reduced to a greater and hence the network can be used in an effective way.

1.1 EXISTING SYSTEM AND ITS LIMITATIONS:

The present system is that the user has to login to their mail-IDs separately and thus they are in need to connect to the internet each and every time they want to check the inbox contents and need to view the mails online. This technique of viewing the mail will result in more consumption of the time in establishing connection each and every time and then checking mail. This will also result in wastage of the cost of connecting with the internet.

The main disadvantage of the existing system is that the cost is to be paid for the internet even during the time of mail composing, as they are typed when the user is being connected with the internet. Even though these are not the major factors that affects the firm's performance directly, they may indirectly produce effects in a large extent.

1.2 PROPOSED SYSTEM AND ITS ADVANTAGES:

In the proposed system, it is made to intimate the user to establish the dialup after every time interval specified in the timer, when the application is being activated .After this the logical connection is made by logging to the POP3 accounts of the specified mail-IDs and copy the content of the inbox into the system's local disk. If the local disk of the system consists of any mails that being prepared offline those mails can also be sent before the connection is de linked. Whenever the user wants to the read the message, the user can see their mails from the local disk, without connecting with the internet.

ADVANTAGES:

1. By this method of retrieving the mail, time of the users are saved a lot and so there is no need for the internet connection when the mails are to be prepared and to be read.

2. This helps in saving the cost of the internet connection. As there is no need for the connection of internet, the cost that is to be paid for the internet connection is saved.

3. The number of users using the internet simultaneously gets reduced and hence each user can have a faster accessing of internet.

2. SYSTEM REQUIREMENT ANALYSIS

2.1 PRODUCT DEFINITION:

The usage of internet in recent day's becomes very vast and hence there results in the traffic in the network. The traffic can be reduced by minimizing the number of users from the simultaneous access of the network. To solve this problem our product helps in reducing the time of using the internet for accessing the mails. This helps in reducing the cost of using the internet as the internet is connected only for a short while when the mail transaction occurs.

2.2 PROJECT PLAN:

To develop this project, we followed the phased life cycle model. The first phase is the analysis phase. This phase is involved in the analysis of the problem statement. After the careful analysis of the problem, the problem the solution for the problem is identified and then the requirements for developing the solution in an implementable form are found. Then we entered the design phase. These two phases are given the highest significance since it allowed us to have a very good knowledge about what we are going to do and how we are going to do. After the design of the product has been completed the implementation phase is entered. For implementation we chose the platform Dot net and choose the language VB.net for our project at this stage. Testing and debugging are then carried out for completing the project.

3.SOFTWARE REQUIREMENT SPECIFICATION

3.1.Introduction:

1.Purpose:

The purpose of the project “Offline Mail Handling”, is to minimize the time of the employees to connect with the internet and read their mails and thus it will help to reduce the amount of money paid by the company for having connection with the internet.

2.Scope:

The use of our product will help the users of internet in an organization to utilize their time in checking their Emails as they not need to connect the internet by himself. This will also reduce the cost of internet connection as the reading of mail with internet connection online is avoided.

3.2. General Description:

1. Product Perspective:

The product developed will have good user interface facility as in the other applications and so it will be very easy to use and understand. It will also ensure good security as the mail content of the other users cannot be accessed by every users as they are protected with the password.

2. Product Functions:

The purpose of our application is to made the local system to made a dial-up connection with the internet and access the Email accounts of the specified mail-IDs. The contents in the inbox of each and every mail-IDs specified, are transferred. If present, the pre composed mails can also be sent before the connection is get de-linked. Whenever the user wants to the read the message, the user can access their mail contents from the local system without connecting with the internet.

3. User Characteristics:

The users of our product need not have very good computer proficiency. Any persons who does not know any of the computer languages can be able to use our product. They can compose the mail at any time and can send and no need to know how and when the mail will reach the destination. It will be sent at the time when the application run.

4. General Constraints:

The users must be aware of their passwords and their mail Ids, to ensure security. Then only they can be able to view their inbox contents from local server and also send the composed mails to the local database.

3.3.Specific Requirements:

1.Functional Requirements:

1.1.Introduction:

The features of our product will be very much useful in the companies in which the employee's time is to be saved a lot. This also facilitates the firm to minimize the cost of using the net connection.

1.2.List of Inputs:

1.User's mail Id.
2.Password.
3.Composed mail and destination Id.
4.POP3 server name
5.SMTP server name

1.3.Information processing requirements:

The interface used to connect the local firm's server with the web server is the normal modem used for internet connection. The user side of the application requires a local network connection with the local server of the organization. The server of the organization needs the database management system to process the transaction of the mails between the local server and the user on the client.

2.Design Constraints:

2.1.Hardware specifications:

Processor	:	Pentium III
RAM	:	64 MB
Hard Disk	:	4 GB
Speed	:	230 MHz
Keyboard	:	Standard 101 keys
Floppy Disk	:	1.44" FDD
Monitor	:	14" Color
Mouse	:	3 Button Logitech
Modem	:	Any high speed modem.

2.2.Software specifications:

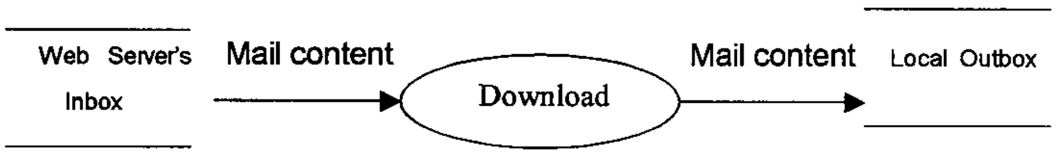
Operating System	:	Windows 2000,XP or ME
Tools Used	:	SQL Server, VB.NET
Framework	:	.NET Framework

2.3.User Interfaces, Screen formats:

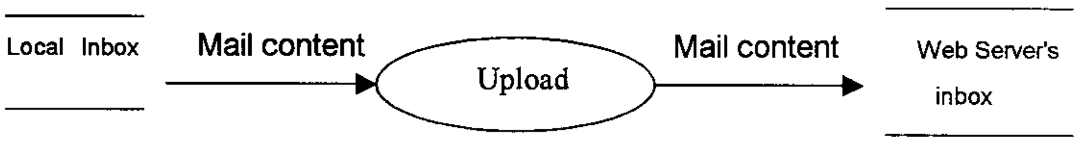
The user interfaces to be designed will be user friendly so that no other professional training is required on the user part. A user interface for entering the mail id and password and also for composing the new mails.

3.3 DATAFLOW DIAGRAM:

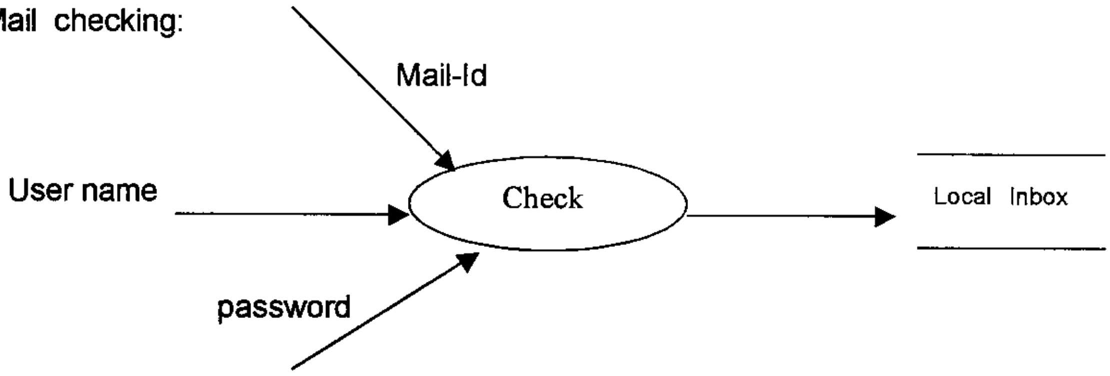
Content Downloading:



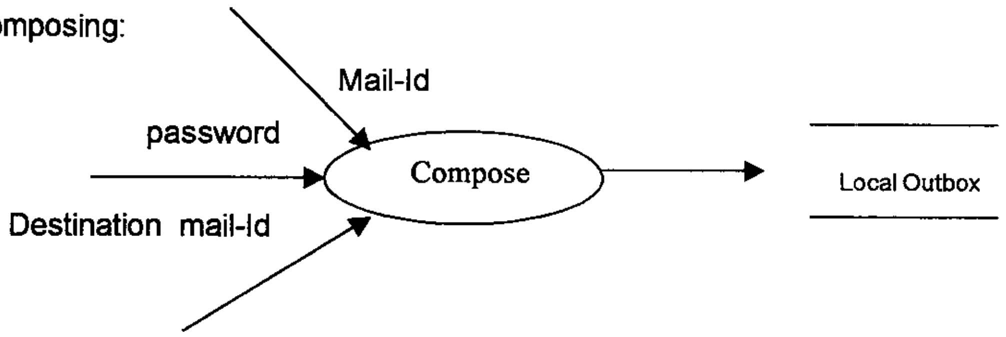
Content Uploading:



Mail checking:



Mail composing:



4. DESIGN DOCUMENTATION

According to Webster, the process of design involves "conceiving and planning out in mind" and "making a drawing, pattern, or sketch of". Generally, any work if done separately as modules will be easier to handle. This project involves the following modules:

4.1 System Design Description:

- **Timer setting and connection establishment:**

  In this module the timer is created and the time after which the application has to be initialized is specified by the user during the initialization of this module. Then the timer will monitoring it for the regular intervals. After the timer reaches the specified interval the dialup connection form will be shown and the user may allow the application to establish the dialup and to continue it or he may not permit, if he wants to quit.

- **Content Downloading:**

  When the timer runs for the specified interval, the web server will be connected. Then the POP3 account for the specified mail-IDs will be checked. If the account name, mail-ID and the corresponding password are matched correctly, then the logical connection will be formed and the contents in the inbox of the account will be fetched using the POP3 protocol. The location in which the retrieved content is to be stored should be specified by the user.

- **Content Uploading:**

  After the mails have been retrieved, this module will send the mails that are present in the local disk to the outbox of the web server with the connection established for the retrieval. This uploading is done using the SMTP protocol. When the uploading process have been completed the control returns to the first module which will disconnect the internet connection.

- **Off-line processing:**

  The processes that are being done offline are the checking of the retrieved, composing the mails that are to be sent by the user and the addition of new accounts to the application.

  1. User account details:

     This is the process of storing the account details in the local data base using SQL Server database agent. The account detail consists of the user name, password, SMTP server name, POP3 server name and the ports used for their connection.

  2. Mail checking:

     Without the internet connection the mails can be checked by retrieving the mails that stored in the local disk. For this the user name and the password should be provided.

  3. Mail composing:

     The mail composing can be done in offline. The composed mail will be sent to the local disk. These mails will be sent to the destination web server during the next firing of the application, i.e., the next time of the timer module.

4.2 Detailed Design Description:

- **Electronic mail:**

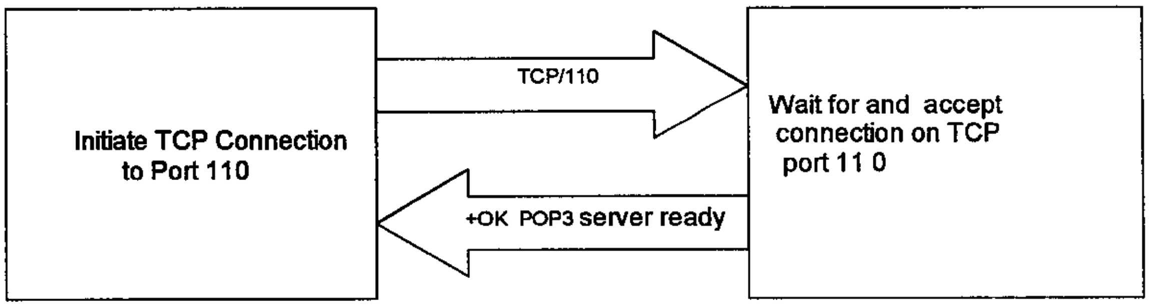
  Electronic mail is a simple and efficient means of transferring information between users on a network-in this case, the Internet. E-mail system provides a means of allowing an originating user to write a message and to address that message for a destination user. This message usually consists of standard text, but with the MIME (Multipurpose Internet Mail Extension or Multimedia E-mail) standard, originating users can attach documents from applications such as Microsoft Word for Windows and Microsoft Excel or even entire applications, sounds and images.

The Internet is connected through the SLIP or PPP connections. When we connect, we use an account that has been assigned to us by an Internet Service Provider (ISP). This account provides SMTP server for sending e-mail and POP server for receiving e-mail.

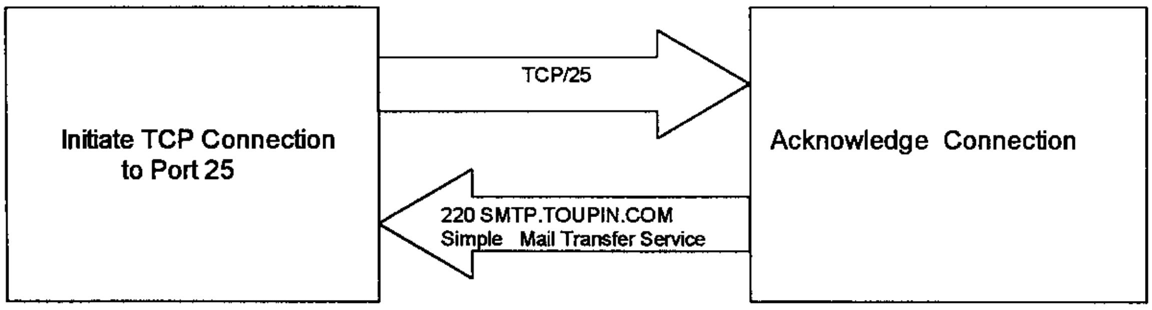
- **Connection establishment:**

The connection with the web server's mailbox is established using the ports available for both the POP3 and SMTP. The SMTP establishes the connection using the port 25. The POP3 establishes the connection through the communication port 110. For establishing the connection both the SMTP and POP3 uses the TCP connection request command to the web server. Then the web sever will response to the connection request, while not, the connection fails.

POP3 connection establishment


Initiate TCP Connection to Port 110
TCP/110
+OK POP3 server ready
Wait for and accept connection on TCP port 11 0

SMTP connection establishment


Initiate TCP Connection to Port 25
TCP/25
Acknowledge Connection
220 SMTP.TOUPIN.COM Simple Mail Transfer Service

- **Content Downloading:**

From connection establishment to it's termination, a POP3 session progresses through various states. They are

1.AUTHENTICATION STATE:

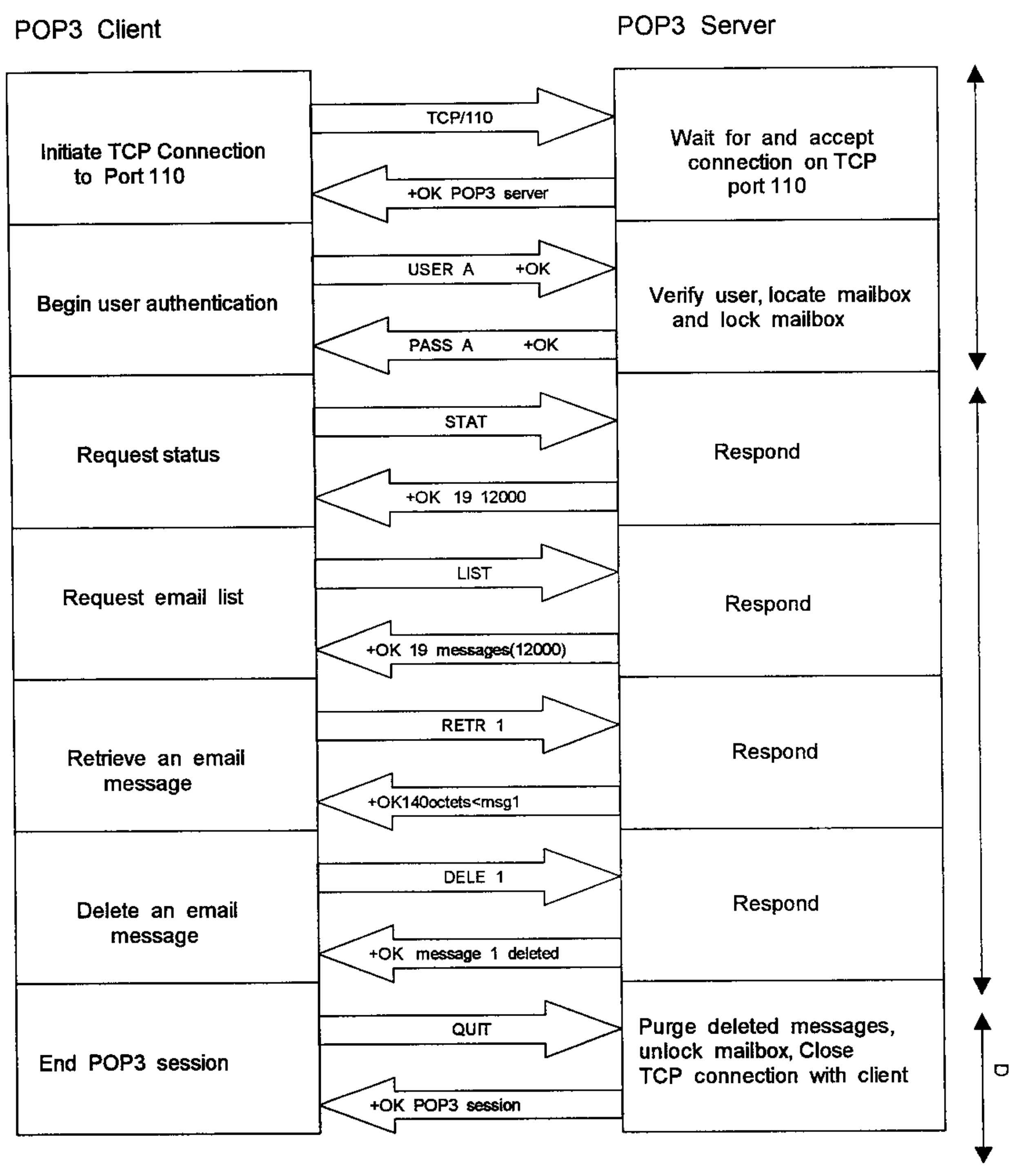
The initial state of the mail downloading is the ***AUTHENTICATION*** state. It involves establishment of the connection and user authentication processes.

2.TRANSACTION STATE:

Immediately after the successful authentication, the POP3 server is in the ***TRANSACTION*** state. In this state the client machine can send commands and receive responses, status information and the email text.

3.UPDATE STATE:

Upon completion of all desired client operations, the client can send a ***QUIT*** command to the server to make the server change states to the ***UPDATE*** states.



- **Content Uploading:**

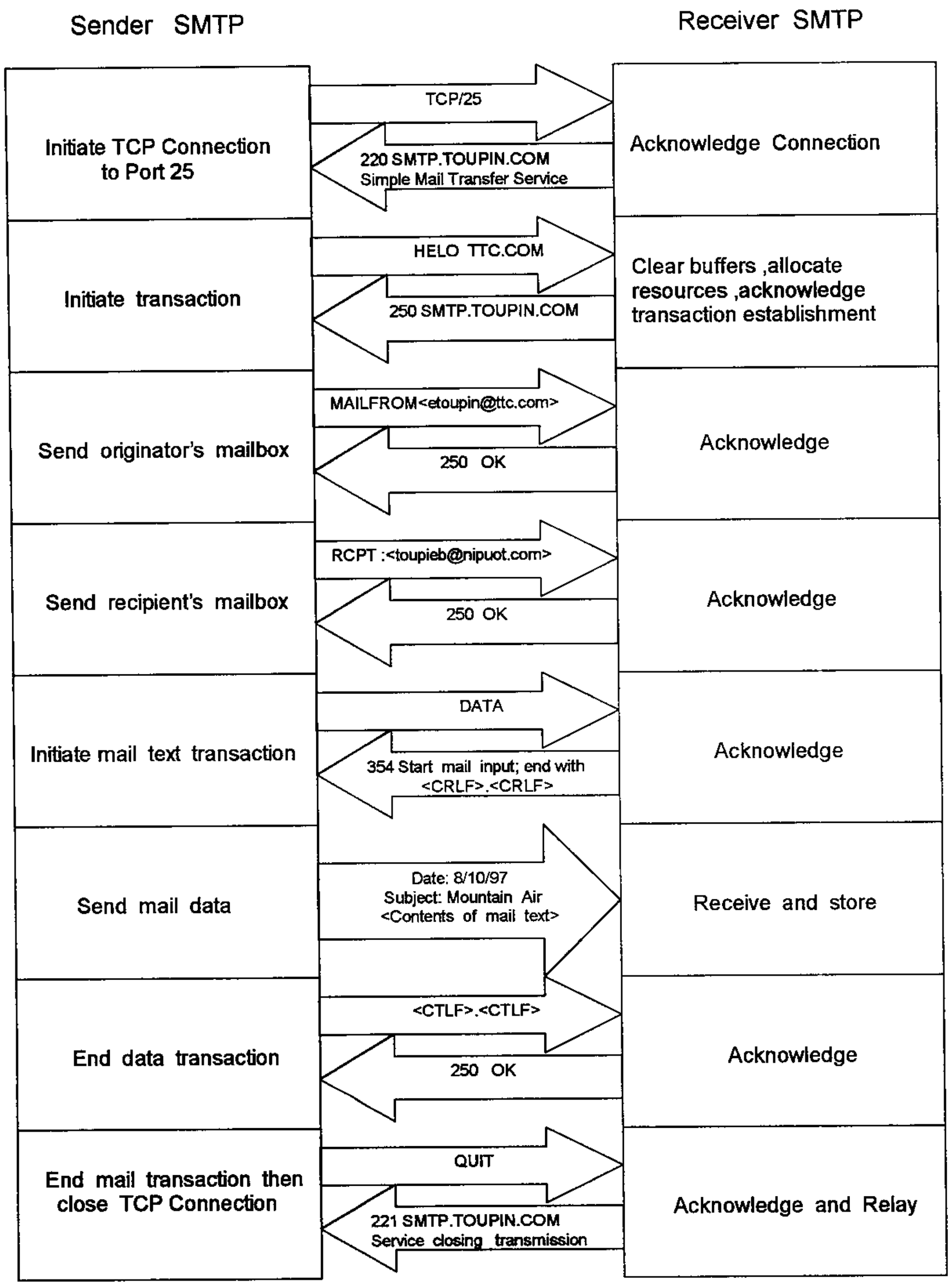
Once the stream-oriented connection to the port 25, over which the communication can occur, is being established, the originator (client) can begin transaction with the SMTP server to which it is connected.

The first set of transaction to occur involve the acknowledgement that the hosts are communicating properly and are available. This is accomplished with the simple ***HELO*** command.

Upon the completion of the entire transaction process and before the termination of the TCP connection, the Sender-SMTP should send the ***QUIT*** command. This command tells the Receiver-SMTP to close the transaction channel and forwards the mail to each of the recipients specified.

UPLOADING COMMANDS:

COMMANDS	**FUNCTIONS**
1.MAIL FROM:<reverse-path><CRLF>	This command of the Sender-SMTP tells the Receiver-SMTP that a new mail transaction is starting and to reset all its state tables and buffer, including any recipients or mail data. The ***<reverse-path>*** contains the reverse source routing list of hosts, showing the path that the e-mail traversed. The first host in this should be the host sending this command; it is usually used for error reporting back to the originator of the e-mail.
2.RCPT TO:<forward-path><CRLF>	This command is used to specify the recipient's e-mail in the ***<forward-path>*** in a standard format.
3.DATA <CRLF>	All the header items and related informations like Date, Subject, To, Cc, From and so on., are submitted as a part of the segment of data using the DATA command.



Sender SMTP
Receiver SMTP
Initiate TCP Connection to Port 25
TCP/25
220 SMTP.TOUPIN.COM Simple Mail Transfer Service
Acknowledge Connection
Initiate transaction
HELO TTC.COM
250 SMTP.TOUPIN.COM
Clear buffers ,allocate resources ,acknowledge transaction establishment
Send originator's mailbox
MAILFROM<etoupin@ttc.com>
250 OK
Acknowledge
Send recipient's mailbox
RCPT :<toupieb@nipuot.com>
250 OK
Acknowledge
Initiate mail text transaction
DATA
354 Start mail input; end with <CRLF>.<CRLF>
Acknowledge
Send mail data
Date: 8/10/97
Subject: Mountain Air
<Contents of mail text>
Receive and store
End data transaction
<CTLF>.<CTLF>
250 OK
Acknowledge
End mail transaction then close TCP Connection
QUIT
221 SMTP.TOUPIN.COM Service closing transmission
Acknowledge and Relay

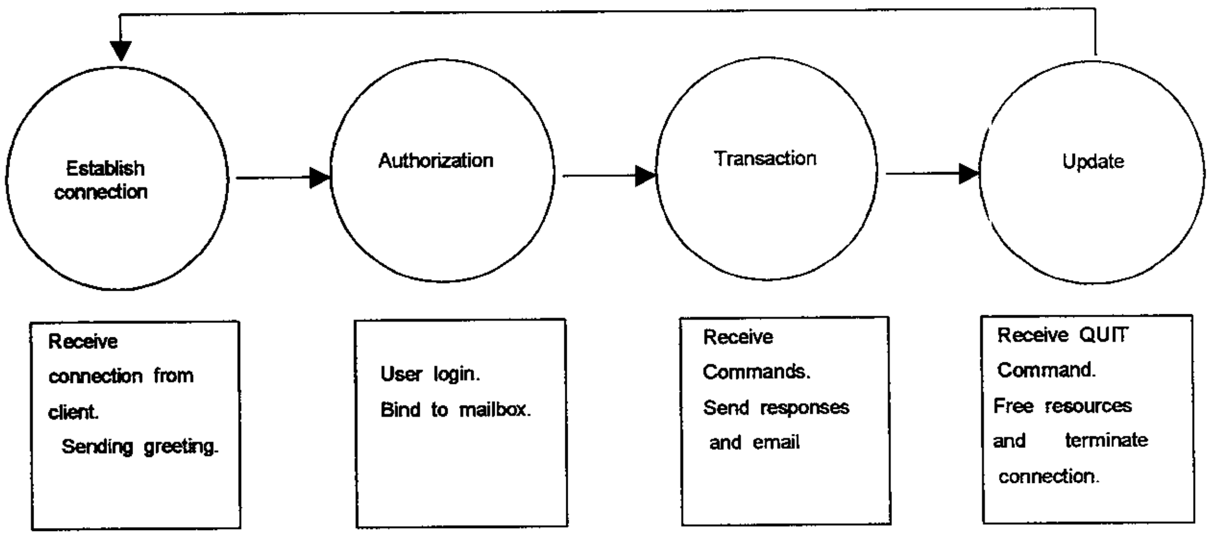
5. DESIGN TOOLS

5.1.POP3:

The purpose of this Post Office Protocol is to help the client machine to query a server's mail box for a mail that was previously delivered by SMTP. POP3 is an important for mail communications, in that it is often difficult to have a large message transport system, such as SMTP on most end-user or client machine Also, it is difficult and expensive to maintain a dial-up connection for a long time, which prohibits the capability to dynamically receive mail from other SMTP server.

To tend to some of these resource and connectivity problems, it is more efficient to manage mail on a central server and provide a means for smaller nodes to retrieve mail from the mail server using mail agents such as Eudora or Pegasus. The server node maintains a full mail transport system such as SMTP, and provides a mailbox service to give these smaller nodes a place to retrieve their email. The primary objective of POP3 is to permit a workstation to dynamically access a mailbox on a server host. This means that a POP3 server is used to allow a workstation to retrieve mail that the server holds for it.

The states and their respective functionalities of the POP3 server are represented in the following figure.


Establish connection
Authorization
Transaction
Update
Receive connection from client. Sending greeting.
User login. Bind to mailbox.
Receive Commands. Send responses and email
Receive QUIT Command. Free resources and terminate connection.

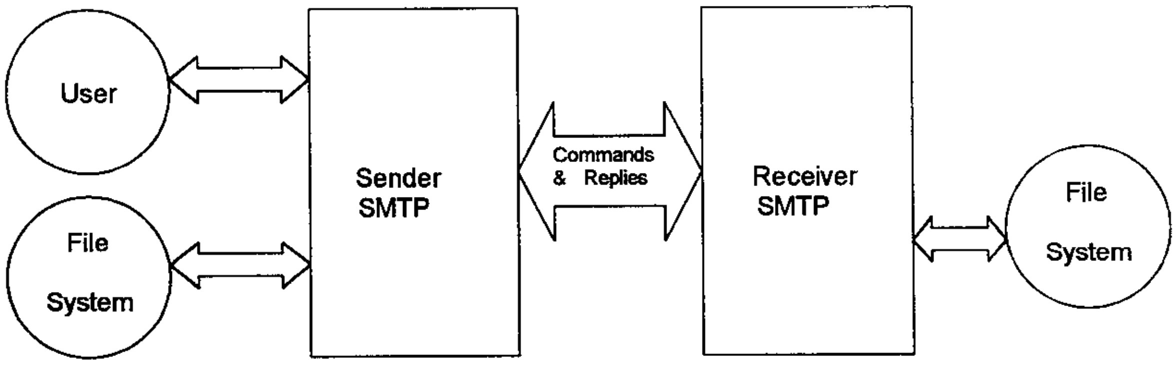
5.2.SMTP:

The Simple Mail Transfer Protocol is the one that is used to transfer the mail from the client to the destination server. The client first authenticates that the hosts are communicating properly by using the acknowledgement from the server as the response to the ***HELO*** command send by the client. Then the transaction takes place with the help of the commands like ***MAIL***, ***RCPT*** and ***DATA***.

After the transaction has been completed, the client sends QUIT command to quit the connection. Sometimes a client has more than one message to transfer to the same server. Instead of quitting, it can send an ***RSET*** request to clear the server's envelope, and then continue with a new ***MAIL***, ***RCPT***, etc.

If the second message has the same contents and the same envelope sender as the first message, the client can save time by merging the recipient lists and sending the message just once.

SMTP MODEL

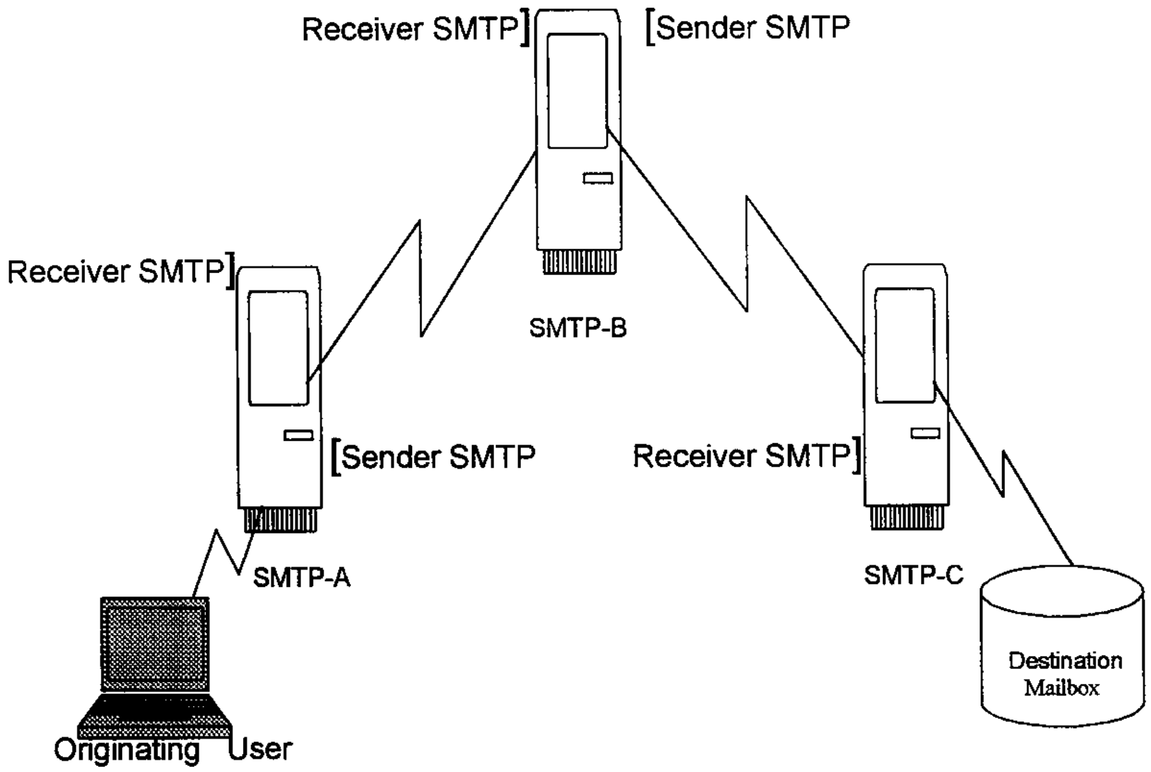


The sender SMTP establishes a two way transmission channel to a Receiver SMTP. The receiver SMTP may be the ultimate destination or an intermediate relay SMTP server.

Relaying E-Mail:

An e-mail message might go through numerous intermediate SMPT servers. The ***forward-path*** from the ***RCPT*** command may be a source route of the form @ SMTP-A.COM,@ SMTP-B.COM, ED @ SMTP-C.COM, where SMTP-A, SMTP-B, SMTP-C are hosts through which the mail should travel before reaching the mailbox for ED on SMTP-C.

The Receiver SMTP transforms the command arguments by moving its own identifier from the ***forward-path*** to the beginning of the ***reverse-path***. The Receiver SMTP then becomes a Sender SMTP, establishes a transmission channel to the next SMTP in the ***forward-path***, and sends it the mail.



Role Reversal:

The first host in the ***reverse-path*** should be the host sending SMTP commands, and the first host in the ***forward-path*** should be the host receiving the SMTP commands. To perform e-mail relaying, the SMTP servers change their role by using the ***TURN*** command.

5.3. DOT NET FRAMEWORK:

The .NET Framework is a new computing platform that simplifies application development in highly distributed environment of the Internet. The .NET Framework is designed to fulfill the following objectives:

- To provide a consistent object-oriented programming environment whether object code is stored and executed locally, or executed remotely.
- To provide a code-execution environment that minimizes software deployment and versioning conflicts.
- To provide a code-execution environment that guarantees safe execution of code, including code created by an unknown or semi-trusted third party.
- To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.
- To make the developer experience consistent across widely varying types of applications, such as Windows-based applications and Web-based applications.
- To build all communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code. The .NET Framework has two main components:

FEATURES OF .NET:

1.The common Language Runtime.
2.NET Framework class library.
3.Common Type System.
4.Metadata.
5. Assemblies
6.Security.

1.THE COMMON LANGUAGE RUNTIME:

- The common language runtime is the foundation of the .NET Framework. The runtime as an agent that manages code at execution time, providing core services such as memory management, thread management, and remoting.
- CLR also enforcing strict type safety and other forms of code accuracy that ensure security and robustness.
- The concept of code management is a fundamental principle of the runtime. The code management contains two types of codes.
    - Managed Code.
    - Unmanaged Code.
- Code that targets the runtime is known as managed code, while code that does not target the runtime is known as unmanaged code.

2. DOT NET FRAMEWORK CLASS LIBRARY:

- The .NET Framework class library is a collection of reusable types that tightly integrate with the common language runtime. The class library is object oriented, providing types from which your own managed code can derive functionality.
- The .NET class library reduces the time associated with learning new features of the .NET Framework.

- The .NET Framework collection classes implement a set of interfaces that is used to develop user's own collection classes.
- The .NET Framework types enable you to accomplish a range of common programming tasks, including tasks such as string management, data collection, database connectivity, and file access.
- The .NET Framework is used to develop the following types of applications and services:
    - ➢ Console applications.
    - ➢ Scripted or hosted applications.
    - ➢ Windows GUI applications (Windows Forms).
    - ➢ ASP.NET applications.
    - ➢ XML Web services.
    - ➢ Windows services.

1.3.COMMON TYPE SYSTEM:

The common type system supports two general categories of types, each of which is further divided into subcategories.

- Value types
    - ➢ Value types directly contain their data, and instances of value types are either allocated on the stack or inline in a structure.
    - ➢ Value types can be built-in (implemented by the runtime), user-defined, or enumerations.
- Reference types
    - ➢ Reference types store a reference to the value's memory address, and are allocated on the heap.
    - ➢ Reference types can be self-describing types, pointer types, or

interface types.

- ➢ The type of a reference type can be determined from values of self-describing types.
- ➢ Self-describing types are further split into arrays and class types.

1.4. METADATA:

- The .NET Framework makes component interoperation even easier by allowing compilers to emit additional declarative information into all modules and assemblies. This information, called metadata, helps components to seamlessly interact.
- Metadata is binary information describing user's program that is stored either in a common language runtime portable executable (PE) file or in memory.
- When user compile the code into a PE file, metadata is inserted into one portion of the file, while the code is converted to Microsoft intermediate language (MSIL) and inserted into another portion of the file.
- Every type and member defined and referenced in a module or assembly is described within metadata.
- When code is executed, the runtime loads metadata into memory and references it to discover information about your code's classes, members, inheritance, and so on.
- Metadata stores the following information:
    - ➢ Description of the assembly.
    - ➢ Identity (name, version, culture, public key).
    - ➢ The types that are exported.
    - ➢ Other assemblies that this assembly depends on.
    - ➢ Security permissions needed to run.
    - ➢ Description of types.

- Name, visibility, base class, and interfaces implemented.
- Members (methods, fields, properties, events, nested types).
- Attributes

1.5. ASSEMBLIES:

- Assemblies are the building blocks of .NET Framework applications; they form the fundamental unit of deployment, version control, reuse, activation scoping, and security permissions.
- An assembly is a collection of types and resources that are built to work together and form a logical unit of functionality.
- An assembly provides the common language runtime with the information it needs to be aware of type implementations.
- It contains code that the common language runtime executes. Microsoft intermediate language (MSIL) code in a portable executable (PE) file will not be executed if it does not have an associated assembly manifest.
- It forms a security boundary. An assembly is the unit at which permissions are requested and granted.
- It forms a type boundary. Every type's identity includes the name of the assembly in which it resides. A type called MyType loaded in the scope of one assembly is not the same as a type called MyType loaded in the scope of another assembly.
- It forms a reference scope boundary. The assembly's manifest contains assembly metadata that is used for resolving types and satisfying resource requests.
- It forms a version boundary. The assembly is the smallest versionable unit in the common language runtime; all types and resources in the same assembly are versioned as a unit.

- It forms a deployment unit. This allows applications to be kept simple and thin when first downloaded.
- It is the unit at which side-by-side execution is supported.
- Assemblies can be static or dynamic.

1.6. SECURITY:

The .NET Framework provides several mechanisms for protecting resources and code from unauthorized code and users:

- ASP.NET Web Application Security provides a way to control access to a site by comparing authenticated credentials (or representations of them) to Microsoft Windows NT file system permissions or to an XML file that lists authorized users, authorized roles, or authorized HTTP verbs.
- Code access security uses the permissions <cpconpermissions.htm> to control the access code has to protected resources and operations. It helps protect computer systems from malicious mobile code and provides a way to allow mobile code to run safely.
- Role-based security provides information needed to make decisions about what a user is allowed to do. These decisions can be based on either the user's identity or role membership or both.

6.SYSTEM TESTING

Levels of Testing:

The details of the software functionality tests are given below. The testing procedure that has been used is as follow:

- ➢ Unit Testing
- ➢ Integration Testing
- ➢ Validation Testing
- ➢ Output Testing

Unit Testing:

Unit testing is carried out to verify and uncover errors within the boundary of the smallest unit or a module. In this testing step, each module was found to be working satisfactory as per the expected output of the module. In the package development, each module is tested separately after it has been completed and checked with valid data. Unit testing exercise specific paths in the modules control structure to ensure complete coverage and maximum error detection.

Integration Testing:

Integration Testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of higher-order tests are conducted. The main objective in this testing process is to take unit-tested modules and build a program structure that has been dictated by design.

The following are the types of integrated testing:

Top down Integration:

This method is an incremental approach to the construction of the program structure. Modules are integrated by moving downward the control hierarchy, beginning with the main program module. The module sub-ordinates to the main program module are incorporated to the structure in either a depth-first or breadth-first manner.

Bottom up Integration:

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to given level is always available and the need for stubs is eliminated. The bottom up integration strategy may be implemented with the following steps:

All the modules which are developed during unit testing are combined together and they are executed and verified whether the linking of the files and the corresponding modules without any error.

Validation Testing:

At the end of integration testing, software is completely assembled as a package, interfacing errors have been uncovered and correction testing begins. Validation is achieved through series of black box tests that demonstrate the conformity that the requirements have been achieved and the documentation prepared is correct and other requirements are also being met. Here all the modules which are checked in the integration testing are assembled and programmed code are tested for any correction.

Output Testing:

Output testing is series of different test whose primary purpose is to fully exercise the computer based system. Although each test has a different purpose, all the work should be verified so that all system element have properly integrated and perform allocated functions.

Output testing is the stage of implantation, which is aimed at ensuring that the system works accurately and efficiently before live operation commences. The input screens, and the output documents were checked and the required modifications are made to suite the program specification. Then using test data prepared, the whole system was tested and found to be a successful one.

Here the whole system is checked by giving sample inputs i.e. the password, SMTP and POP3 server names for the email Ids that has the POP3 accounts and it is checked for the whole function so that the system functions well and the requirements are met.

7.FUTURE ENHANCEMENTS

Our project is so flexible that it can be enhanced so easily in the future. Some more features could also be added to our project. This product is developed for a single user. This can also be implemented in local area networks so that a number of users can make use of the application.

Our product requires a manual dialup connection establishment. This can be automated by making the timer module to establish a dialup by itself which should be installed in the server machine alone. The content uploading and the downloading modules can be implemented in the server machine which is the only machine needed to have Internet connection. The clients can use the existing simple LAN network connection and check their mails offline.

8. CONCLUSION

The complete design and development of the system "Offline Mail Handling" is presented in this tract. The system has user friendly features. It is possible for any novice user to use the system. The programming techniques used in the design of the system provide a scope for further expansion and implementation of any changes, which may occur in the future. The system has been tested for the hosts running on any Operating System.

Since, the requirements of any organization and their standards are changing day by day; the system has been designed in such a way that its scope and boundaries could be expanded in the future with little modifications. As a further enhancement, this system can include the other features specified in the Future Enhancement Section

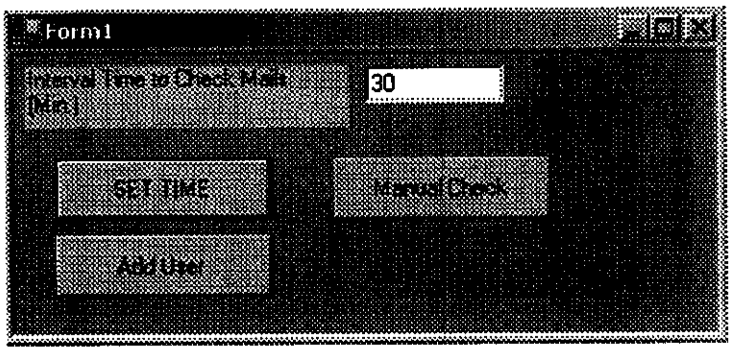
9. REFERNCES

- Programming Internet Controls – Galgotia publications [MarkuPope]
- .Net Franewonk Essentials Shroff publishers & Distributors[O'Reily].
- www.msdn.com
- www.google.com
- Author Name : Daniel F. Savarese (From Web).

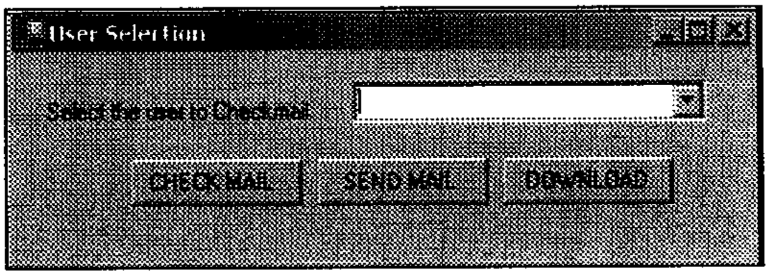
10. ANNEXTURE

10.1 OUTPUT SCREENS

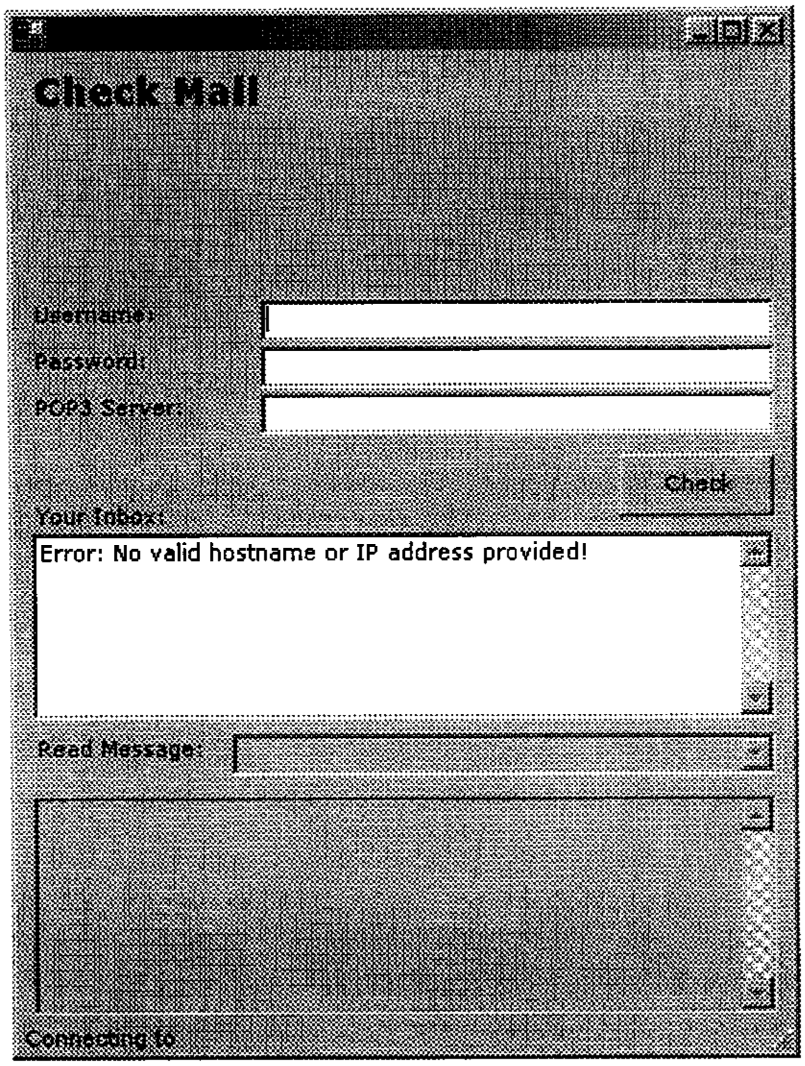
TIMER SETTING FORM:



USER SELECTION FORM:



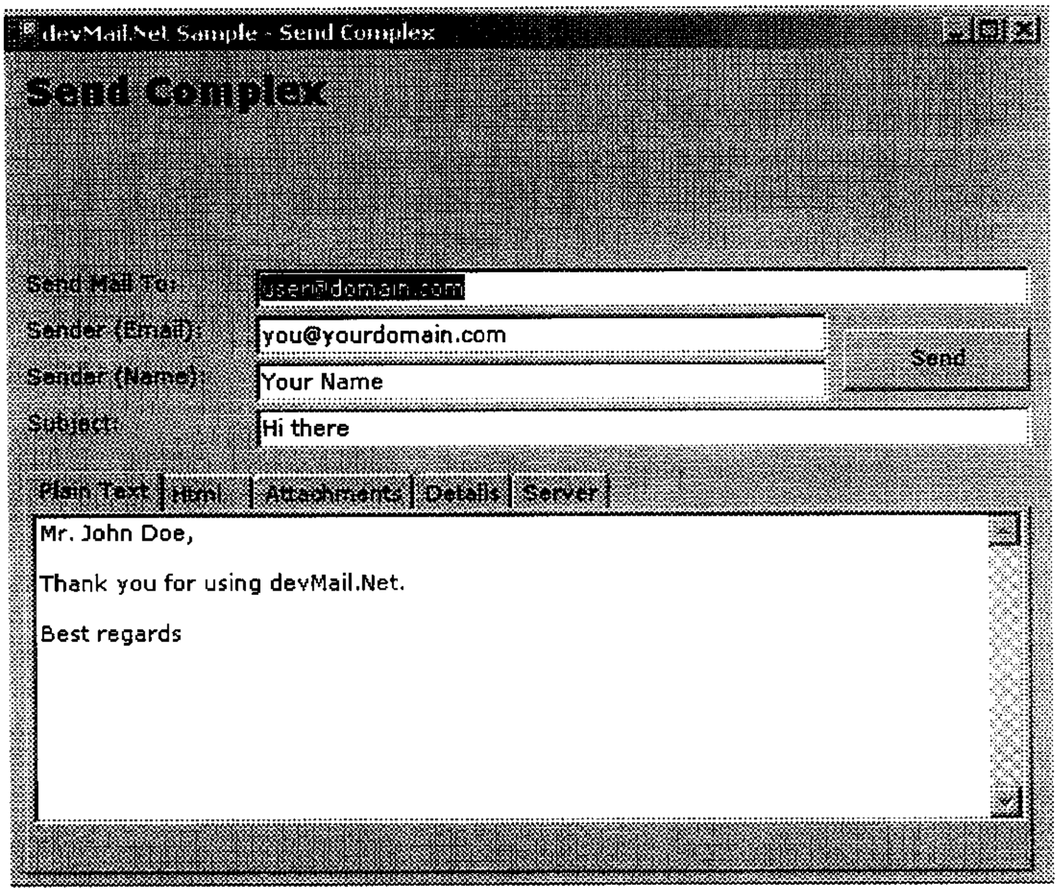
CHECK MAIL FORM:



```
Error: No valid hostname or IP address provided!
```

Connecting to

SEND MAIL FORM:



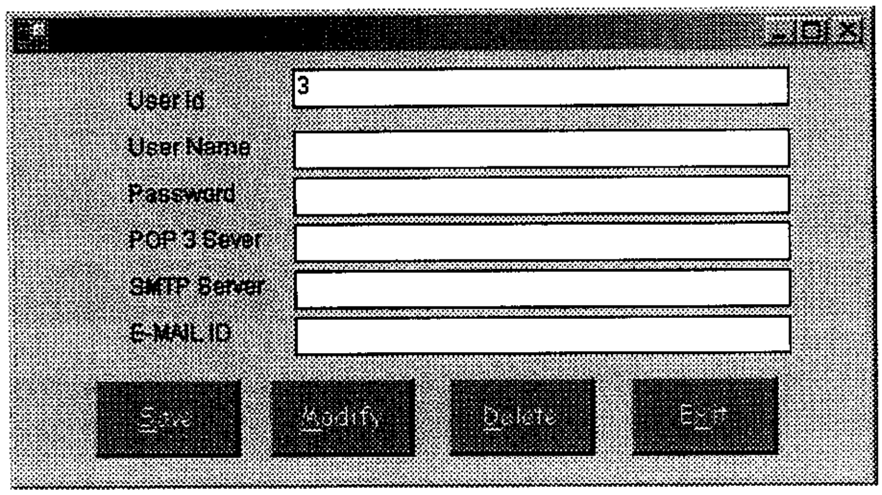
Send Mail To: user@domain.com
Sender (Email): you@yourdomain.com
Sender (Name): Your Name
Subject: Hi there

Mr. John Doe,

Thank you for using devMail.Net.

Best regards

ADD USER FORM:



User Id	3
User Name	
Password	
POP 3 Sever	
SMTP Server	
E-MAIL ID	

10.2 SOURCE CODE:

```
Imports devBiz.Net.Mail
Imports System.IO
Imports System.Windows.Forms
Imports System.Runtime.Serialization
Imports System.Runtime.Serialization.Formatters.Soap


Public Class frmMaindown
   Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

   Public Sub New()
      MyBase.New()

End Sub

   Friend WithEvents Label2 As System.Windows.Forms.Label
   Friend WithEvents Label3 As System.Windows.Forms.Label
   Friend WithEvents Label6 As System.Windows.Forms.Label
   Friend WithEvents txtUserName As System.Windows.Forms.TextBox
   Friend WithEvents Label1 As System.Windows.Forms.Label
   Friend WithEvents txtPassword As System.Windows.Forms.TextBox
   Friend WithEvents txtServer As System.Windows.Forms.TextBox
   Friend WithEvents cmdDownload As System.Windows.Forms.Button
   Friend WithEvents Label4 As System.Windows.Forms.Label
   Friend WithEvents lblFolder As System.Windows.Forms.LinkLabel
   Friend WithEvents optHeadersAsSystem.Windows.Forms.RadioButton
   Friend WithEvents optFull As System.Windows.Forms.RadioButton
   Friend WithEvents sb As System.Windows.Forms.StatusBar

   Friend WithEvents Inbox As New POP3()
   Friend WithEvents pb As System.Windows.Forms.ProgressBar
   Friend WithEvents btnBrowse As System.Windows.Forms.Button

      Me.txtUserName = New System.Windows.Forms.TextBox()
      Me.Label1 = New System.Windows.Forms.Label()
      Me.Label2 = New System.Windows.Forms.Label()
      Me.txtPassword = New System.Windows.Forms.TextBox()
      Me.txtServer = New System.Windows.Forms.TextBox()
      Me.Label3 = New System.Windows.Forms.Label()
      Me.cmdDownload = New System.Windows.Forms.Button()
Me.Label6 = New System.Windows.Forms.Label()
Me.Label4 = New System.Windows.Forms.Label()
Me.optHeaders = New System.Windows.Forms.RadioButton()
Me.optFull = New System.Windows.Forms.RadioButton()
Me.lblFolder = New System.Windows.Forms.LinkLabel()
Me.sb = New System.Windows.Forms.StatusBar()
Me.pb = New System.Windows.Forms.ProgressBar()
Me.btnBrowse = New System.Windows.Forms.Button()
Me.SuspendLayout()

    'txtUserName
'
Me.txtUserName.Location = New System.Drawing.Point(128, 136)
Me.txtUserName.Name = "txtUserName"
Me.txtUserName.Size = New System.Drawing.Size(264, 21)
Me.txtUserName.TabIndex = 1
Me.txtUserName.Text = "usemame"
'
'Label1
'
Me.Label1.Location = New System.Drawing.Point(8, 136)
Me.Label1.Name = "Label1"
Me.Label1.Size = New System.Drawing.Size(120, 16)
Me.Label1.TabIndex = 2
Me.Label1.Text = "Username:"
'
'Label2
'
Me.Label2.Location = New System.Drawing.Point(8, 160)
Me.Label2.Name = "Label2"
Me.Label2.Size = New System.Drawing.Size(120, 16)
Me.Label2.TabIndex = 3
Me.Label2.Text = "Password:"
'
'txtPassword
'
Me.txtPassword.Location = New System.Drawing.Point(128, 160)
Me.txtPassword.Name = "txtPassword"
Me.txtPassword.PasswordChar = Microsoft.VisualBasic.ChrW(42)
Me.txtPassword.Size = New System.Drawing.Size(264, 21)
Me.txtPassword.TabIndex = 4
Me.txtPassword.Text = "password"
'
'txtServer
'
Me.txtServer.Location = New System.Drawing.Point(128, 184)
Me.txtServer.Name = "txtServer"
Me.txtServer.Size = New System.Drawing.Size(264, 21)
Me.txtServer.TabIndex = 6
Me.txtServer.Text = "localhost"
'
'Label3
'
Me.Label3.Location = New System.Drawing.Point(8, 184)
Me.Label3.Name = "Label3"
Me.Label3.Size = New System.Drawing.Size(120, 16)
Me.Label3.TabIndex = 5
Me.Label3.Text = "POP3 Server:"
'
'cmdDownload
'
Me.cmdDownload.Location = New System.Drawing.Point(312, 336)
Me.cmdDownload.Name = "cmdDownload"
Me.cmdDownload.Size = New System.Drawing.Size(80, 32)
Me.cmdDownload.TabIndex = 13
Me.cmdDownload.Text = "Download"
'
'Label6
'
Me.Label6.Location = New System.Drawing.Point(8, 8)
Me.Label6.Name = "Label6"
Me.Label6.Size = New System.Drawing.Size(384, 40)
Me.Label6.TabIndex = 14
Me.Label6.Text = "Download Messages"
'
'Label4
'
Me.Label4.Location = New System.Drawing.Point(8, 272)
Me.Label4.Name = "Label4"
Me.Label4.Size = New System.Drawing.Size(384, 16)
Me.Label4.TabIndex = 18
Me.Label4.Text = "Messages will be downloaded to:"
'
'optHeaders
'
Me.optHeaders.Checked = True
Me.optHeaders.Location = New System.Drawing.Point(128, 216)
Me.optHeaders.Name = "optHeaders"
Me.optHeaders.Size = New System.Drawing.Size(264, 16)
Me.optHeaders.TabIndex = 20
Me.optHeaders.TabStop = True
Me.optHeaders.Text = "Download only message headers"
'
```

```
    'optFull
    '
    Me.optFull.Location = New System.Drawing.Point(128, 240)
    Me.optFull.Name = "optFull"
    Me.optFull.Size = New System.Drawing.Size(264, 16)
    Me.optFull.TabIndex = 21
    Me.optFull.Text = "Download full messages"
    '
    'lblFolder
    '
    Me.lblFolder.Location = New System.Drawing.Point(8, 288)
    Me.lblFolder.Name = "lblFolder"
    Me.lblFolder.Size = New System.Drawing.Size(296, 48)
    Me.lblFolder.TabIndex = 22
    '
    'sb
    '
    Me.sb.Location = New System.Drawing.Point(0, 371)
    Me.sb.Name = "sb"
    Me.sb.Size = New System.Drawing.Size(400, 16)
    Me.sb.TabIndex = 24
    '
    'pb
    '
    Me.pb.Location = New System.Drawing.Point(288, 376)
    Me.pb.Name = "pb"
    Me.pb.Size = New System.Drawing.Size(96, 8)
    Me.pb.TabIndex = 25
    '
    'btnBrowse
    '
    Me.btnBrowse.Location = New System.Drawing.Point(312, 296)
    Me.btnBrowse.Name = "btnBrowse"
    Me.btnBrowse.Size = New System.Drawing.Size(80, 32)
    Me.btnBrowse.TabIndex = 26
    Me.btnBrowse.Text = "Browse..."
    '
    'frmMaindown
    '
    Me.AutoScaleBaseSize = New System.Drawing.Size(6, 14)
    Me.ClientSize = New System.Drawing.Size(400, 387)
    Me.Controls.AddRange(New System.Windows.Forms.Control()
    Me.Text = "devMail.Net Sample - Download Messages"
    Me.ResumeLayout(False)

End Sub
```