A

Project On

"Distributed Data Management and Recovery System"

Submitted for the partial fulfillment of
B.E. Final Year Information Technology

Submitted By:-
Govind Salvi
Nitin Tater
Don Pandya
Bharat Choudhary

Seminar Guide:-
Asst. Prof. Alok S. Gehlot
(M.B.M Engg. college Jodhpur.)

Department of Computer Science and Engineering
MBM Engineering College, Faculty of Engineering,
Jai Narain Vyas University
Jodhpur (Rajasthan)
Session 2010-11

CERTIFICATE

This is to certify that report entitled "**Distributed data management & recovery system**" submitted to Department of Computer Science and Engineering, MBM Engineering College, JNV University in partial fulfilment of the requirements for the award of the degree of B.E. Final Year IT is the bonafied record of the work done by Govind Salvi , Don Pandya , Nitin Tater , Bharat Choudhary under my supervision and guidance.

Signature of the guide

Asst. Prof. Alok Singh Gehlot

Department of Computer Science and Engineering
MBM Engineering College, Faculty of Engineering,
Jai Narain Vyas University
Jodhpur (Rajasthan)

DECLARATION

We, **Govind salvi, Don Pandya , Nitin Tater, Bharat Choudhary** hereby declare that this project titled "**Distributed data management & recovery system**" is a record of original work done by us under the supervision and guidance of **Mr. Alok Gehlot**. We further certify that this project work has not formed the basis for the award of the Degree/Diploma/Associateship/Fellowship or similar work to any candidate of any university and no part of this report is reproduced as it is from any other source without seeking permission.

Signature of the student(s)

Govind Salvi (07/05766)
Don Pandya (08/06355)
Nitin Tater (07/05880)
Bharat Choudhary (   )

ACKNOWLEDGEMENT

It gives us immense pleasure in presenting our Project report. We would like to take this opportunity to express our deepest gratitude to the people, who has contributed their valuable time for helping me to successfully complete this Project and whose constant guidance and encouragement crown all the efforts with success.

With great pleasure and acknowledgement we extend our deep gratitude to Dr. K. R. Chowdhary, Head, Department of Computer Science and Engineering, MBM Engineering College Jodhpur, for giving us opportunity to accomplish our Project.

It is our profound privilege to express our deep sense of gratitude to **Asst. Prof. Alok Singh Gehlot (MBM Engineering College)** for his precious guidance, constructive encouragement and whole hearted support throughout the training tenure.

We also appreciate the efforts of all the members, the lab in-charges, the administrative staff of **Department of computer Science and Engineering** Jodhpur who have contributed in some form or the other in grooming us.

Govind Salvi (07/05766)

Don Pandya (08/06355)

Nitin Tater (07/05880)

Bharat Choudhary (   )

Abstract

The Project is basically an application of distributed data management and recovery system includes a web server application and FTP Server application. The projects very much relates to a banking enterprise where when an account of a customer is made at a particular branch all of his or her information must be stored at the branch server as well as the central server situated at central branch office.

The distribution of the data is helpful when there is a data lost at any branch. In such situation the recovery can be done by getting the data back from the distributed server.

As banking is an enterprise level organization the projects also includes a web based application server the root level administrator or manager of the bank can also publish the updates of the bank on the server all this information is visible to every employee when he/she visits the server url (uniform resource locator).
Similar to this a user can use the application or data store on the remote computer Through the web server component of the project like wise to store a file on the remote server or get a file from remote server FTP server component of the project helps.

Contents

Introduction

1.) Distributed Database System:-

A distributed database (DDB) is a collection of multiple, logically interrelated databases distributed over a computer network. A distributed database management system (D–DBMS) is the software that manages the DDB and provides an access mechanism that makes this distribution transparent to the users. In a distributed database system, the database is stored on several computers. The computer in a distributed system Communicate with one another through various communication media, such as high-speed network or telephone lines. They do not share main memory or disks. The computer in a distributed system may vary in size and function, ranging from workstations up to mainframe systems.

The computers in a distributed system are referred to by a number of different names, such as sites or nodes, depending on the context in which they are mentioned. We mainly use the term site, to emphasize the physical distribution of these systems.

The main differences between shared-nothing parallel databases and distributed databases are that distributed databases are typically geographically separated, are separately administered, and have a slower interconnection. Another major difference is that, in a distributed database system, we differentiate between local and global transaction. A local transaction is one that accesses data only from sites where the transaction was initiated. A global transaction, on the other hand, is one that either accesses data in a site different from the one at which the transaction was initiated, or accesses data in several different sites. There are several reasons for building distributed database systems, including sharing of data, autonomy, and availability.

Distributed DBMS Promises:-

- Transparent management of distributed,
- Fragmented, and replicated data,
- Improved reliability/availability through,
- Distributed transactions
- Improved performance
- Easier and more economical system expansion

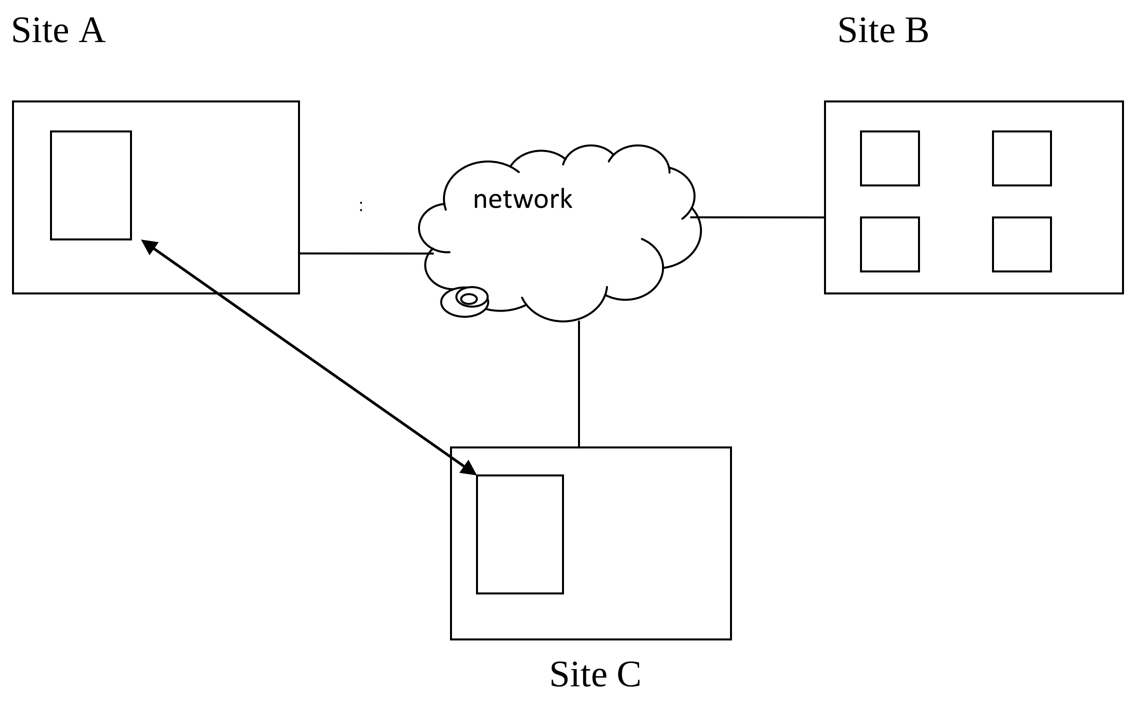


Figure 1.01 A distributed System

Distributed data Storage:

Replication:- The System maintains several identical replicas (copies) of the relation, and stores each replica at a different site. The alternative to replication is to store only one copy of relation.

Fragmentation:- The System partitions the relation into several fragments, and stores each fragment at a different site.

Fragmentation and replication can be combined: A relation can be partition into several fragments and there may be several replicas of each fragment.

i.) Client -Server Architecture:-

Clients and Servers are separate logical entities that work together over a network to accomplish a task. Many systems with very different architectures that are connected together are also called Client/Server.

Client/server computing is the logical extension of modular programming. Modular programming has as its fundamental assumption that separation of a large piece of software into its constituent parts ("modules") creates the possibility for easier development and better maintainability. Client/server computing takes this a step farther by recognizing that those modules need not all be executed within the same memory space. With this architecture, the calling module becomes the "client" (that which requests a service), and the called module becomes the "server" (that which provides the service). The logical extension of this is to have clients and servers running on the appropriate hardware and software platforms for their functions. For example, database management system servers running on platforms specially designed and configured to perform queries, or file servers running on platforms with special elements for managing files. It is this latter perspective that has created the widely-believed myth that client/server has something to do with PCs or Unix machines.

To understand the client server architecture we need to know the following terms.

What is a Client process?

The client is a process (program) that sends a message to a server process (program), requesting that the server perform a task (service). Client programs usually manage the user-interface portion of the application, validate data entered by the user, dispatch requests to server programs, and sometimes execute business logic. The client-based process is the front- end of the application that the user sees and interacts with. The client process contains solution-specific logic and provides the interface between the user and the rest of the application system. The client process also manages the local resources that the user interacts with such as the monitor, keyboard, workstation CPU and peripherals. One of the key elements of a client workstation is the graphical user interface (GUI). Normally a part of operating system i.e. the window manager detects user actions, manages the windows on the display and displays the data in the windows.

What is a Server process?

A server process (program) fulfills the client request by performing the task requested. Server programs generally receive requests from client programs, execute database retrieval and updates, manage data integrity and dispatch responses to client requests. Sometimes server programs execute common or complex business logic. The server-based process "may" run on another machine on the network. This server could be the host operating system or network file server; the server is then provided both file system services and application services. Or in some cases, another desktop machine provides the application services. The server process acts as a software engine that manages shared resources such as databases, printers, communication links, or high poweredprocessors. The server process performs the back-end tasks that are common to similar applications.

Characteristics of Client/Server

- Service
- Shared resources
- Asymmetrical protocols
- Transparency of location
- Mix-and-match
- Message based exchanges
- Encapsulation of services
- Scalability
- Integrity

Client/Server computing is the ultimate "Open platform". It gives the freedom to mix-and-match components of almost any level. Clients and servers are loosely coupled systems that interact through a message-passing mechanism.

Client application program running on the local machine requests a service from another application program – server – running on the remote machine. Commonly server provides service to any client, not a particular client.

Generally, a client application program that requests a service should run only when it is needed. A server program providing service should run all the time, as it does not know when its services will be needed.

The Vision of the Client/Server Computing

Client/server computing is the most effective source for the tools that empower employees with authority and responsibility.

Workstation power, workgroup empowerment, preservation of existing investments, remote network management, and market-driven business are the forces creating the need for client/server computing.

The Client/server computing is an irresistible movement that is reshaping the way computers is being used. Though this computing is very young it is already in full force and is not leaving any area and corner of the computer industry untouched. The Client/Server application development requires hybrid skills that include transaction processing, database design, communication experience, graphical user interface design, and being Internet Savvy. The more advanced applications require knowledge of distributed objects and component infrastructures. Most client/server solutions today are PC LAN implementation that is personalized for the group that uses them. The Client/Server Computing has changed the way in which computers are being used. This has a unique feature of having a strong foothold on the entire spectrum of the computer industry.

For the "PC my World" lot, the Client/Server Computing means - scrapping of every mainframe that cannot fit on the desktop and thus end of the host-centric computing.

For the "Diehard fans of the Mainframes", the Client/Server Computing means unleashing the rebirth of the networked mainframes that will bring every PC in the enterprise block to the fold. For "Men in between", it is a new era of Internet based co-existence and openness in which all can play.

Let us now know what a client is and what is a server?

Client - A client is a single-user workstation that provides presentation services and the appropriate computing, connectivity and the database services and the interfaces relevant to the business need.

Server- A server is one or more multi-user processors with share memory providing computing, connectivity and the database services and the interfaces relevant to the business need.

The Client/Server computing is an environment that satisfies the business need by appropriately allocating the application processing between the client and the server processors.

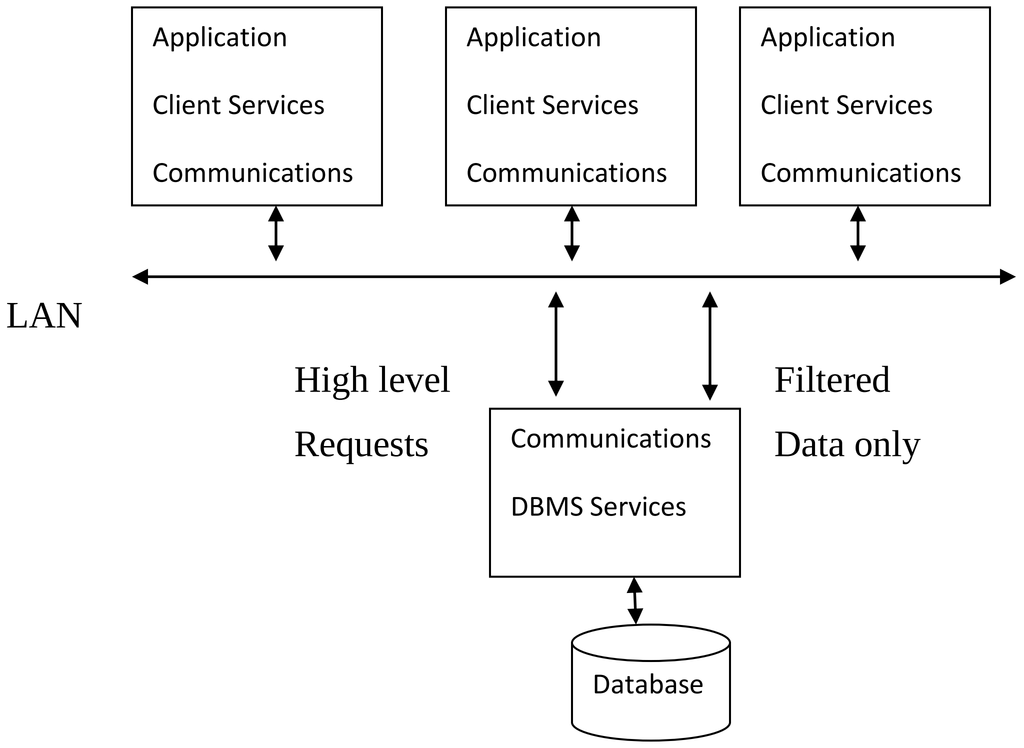


Figure 1.02 Client /Server Architecture

The protocol is the client requests the services from the server; the server processes the request and returns the result to the client. The communication mechanism is a message passing Interprocess communication (IPC) that enables the distributed placement of the client and server processes.

The Client/Server is the generic model and fits what is known in the industry as the “cooperating processing” or “peer-to-peer”. The client/server computing is fundamentally platform independent.

The user of an application wants the functionality (business) it provides; the computing platform provides access to this business functionality. There are no benefits but a considerable amount of risk of exposing the platform to the users.

The changes in the platform and the underlying technology should be transparent to the user, as it is understood that the systems built with transparency to the technology, for the entire user offer the highest probability of solid ongoing return for the technology investment. It is also easily demonstrable that is the developers become aware of the target platform, development will be bound to that platform and they will use special tricks and feature found in that specific platform.

The figure shown below depicts the modern client/server architecture.

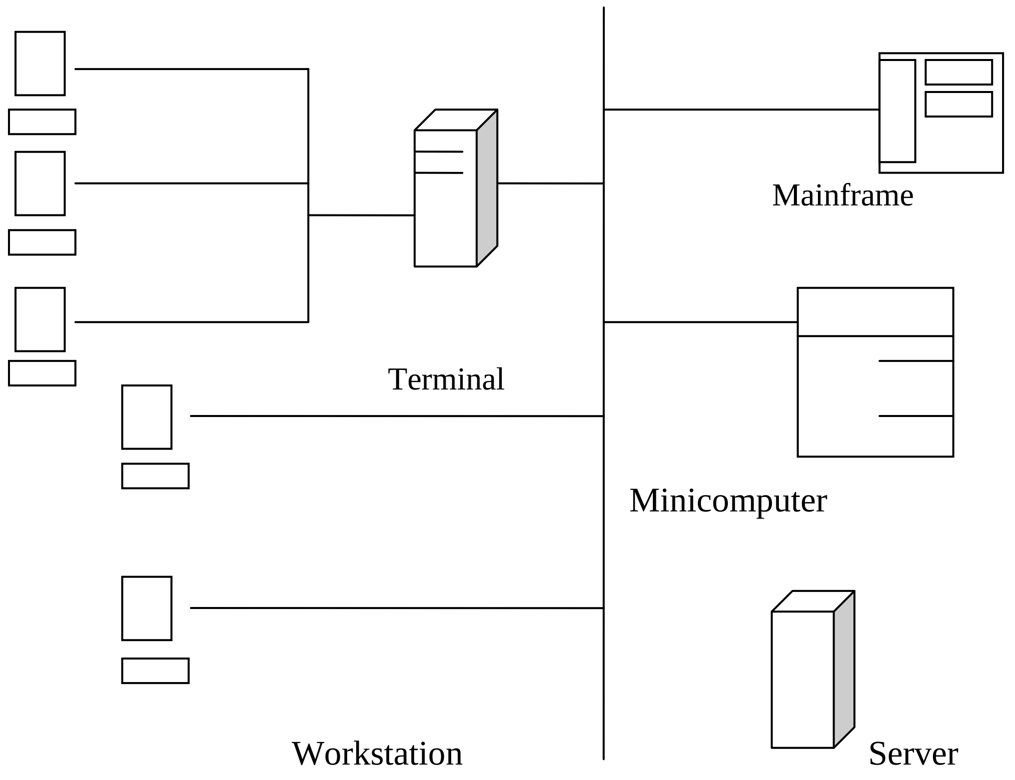


Figure 1.03 Modern Client/Server Architecture

Characteristics of the Client and the Server:-

The clients and the servers are the logical entities that work together over a network to accomplish a task.

The distinguishing characteristics of the Client/Server systems are:

Service: The client/server is primarily a relationship between processes running on separate machines. The server process is a provider of services. The client is a consumer of services. In essence, client/server provides a clean separation of function based on the idea of service.

Shared Resources: A server can service many clients at the same time and regulate their access to shared resources.

Asymmetrical protocols: There is a many-to-one relationship between the clients and the server. Clients always initiate the dialog by requesting a service. Servers are passively awaiting request from the clients. In some cases a client may pass a reference to a call-back object when it invokes a service. This lets the server call back the client. So the client becomes a server.

Transparency of location: The server is a process that can reside on the same machine as the client or on a different machine across a network. Client/Server software usually masks the location of the server from the clients by the redirecting the service calls when needed. A program can be a client, a server, or both.

Mix-and-match: The ideal client/server software is independent of hardware or operating system software platforms. You should be able to mix-and-match client and server platforms.

Message-based exchanges: Clients and servers are loosely coupled systems that interact through a message-passing mechanism. The message is the delivery mechanism for the service request and replies.

Encapsulation of services: The server is a specialist. A message tells a server is requested; it is then up to the server to determine how to get the job done. Servers can be upgraded without affecting the clients as long as the published message interface is not changed.

Scalability: Client/Server systems can be scaled horizontally or vertically. Horizontal scaling means adding or removing client workstations with only a slight performance impact. Vertical scaling means either migrating to a larger and faster server machine or distributing the processing load across multiple servers.

Integrity: The server code and server data is centrally managed, which results in cheaper maintenance and the guarding of shared data integrity. At the same time, the clients remain personal and independent.

Merits and Demerits of the Client Server

1. The Merits of Client/Server Computing

Client/server computing provides the capability to use the most cost-effective user interface, data storage, connectivity, and application services. Frequently, client/server products are deployed within the present organization but are not used effectively.

The client/server model provides the technological means to use previous investments in concert with current technology options. Organizations see opportunities to use technology to provide business solutions. Service and quality competition in the marketplace further increases the need to take advantage of the benefits available from applications built on the client/server model.

Client/server computing in its best implementations moves the data-capture and information-processing functions directly to the knowledgeable worker—that is, the worker with the ability to respond to errors in the data, and the worker with the ability to use the information made available.

i.) Enhanced Data Sharing

Data that is collected as part of the normal business process and maintained on a server is immediately available to all authorized users.

The use of Structured Query Language (SQL) to define and manipulate the data provides support for open access from all client processors and software.

SQL grants all authorized users access to the information through a view that is consistent with their business need. Transparent network services ensure that the same data is available with the same currency to all designated users.

ii.) Integrated Services

In the client/server model, all information that the client is entitled to use is available at the desktop. There is no need to change into terminal mode or log into another processor to access information.

All authorized information and processes are directly available from the desktop interface. The desktop tools—e-mail, spreadsheet, presentation graphics, and word processing—are available and can be used to deal with information provided by application and database servers resident on the network.

Desktop users can use their desktop tools in conjunction with information made available from the corporate systems to produce new and useful information.

iii.) Sharing Resources among Diverse Platforms

The client/server computing model provides opportunities to achieve true open system computing. Applications may be created and implemented without regard to the hardware platforms or the technical characteristics of the software.

Thus, users may obtain client services and transparent access to the services provided by database, communications, and applications servers.

Operating systems software and platform hardware are independent of the application and masked by the development tools used to build the application.

In this approach, business applications are developed to deal with business processes invoked by the existence of a user-created "event."

iv.) Data Interchangeability and Interoperability

SQL is an industry-standard data definition and access language. This standard definition has enabled many vendors to develop production-class database engines to manage data as SQL tables. Almost all the development tools used for client/server development expect to reference a back-end database server accessed through SQL.

Network services provide transparent connectivity between the client and local or remote servers. With some database products, such as Ingres Star, a user or application can define a consolidated view of data that is actually distributed between heterogeneous, multiple platforms.

Systems developers are finally reaching the point at which this heterogeneity will be a feature of all production-class database engine products. Most systems that have been implemented to date use a single target platform for data maintenance. The ability to do high-volume updates at multiple locations and maintain database integrity across all types of errors is just becoming available with production-level quality performance and recovery.

Systems developed today that use SQL are inherently transparent to data storage location and the technology of the data storage platform. The SQL syntax does not specify a location or platform.

This transparency enables tables to be moved to other platforms and locations without affecting the application code. This feature is especially valuable when adopting proven, new technology or if it makes business sense to move data closer to its owner.

v.) Location Independence of Data and Processing

We are moving from the machine-centered computing era of the 1970s and 1980s to a new era in which PC-familiar users demand systems that are user-centered. Previously, a user logged into a mainframe, mini-, or micro application. The syntax of access was unique in each platform. Function keys, error messages, navigation methods, security, performance, and editing were all very visible. Today's users expect a standard "look and feel." Users log into an application from the desktop with no concern for the location or technology of the processors involved.

vi.) Encapsulation:-

In most cases, client-server architecture enables the roles and responsibilities of a computing system to be distributed among several independent computers that are known to each other only through a network. This creates an additional advantage to this architecture: greater ease of maintenance. For example, it is possible to replace, repair, upgrade, or even relocate a server while its clients remain both unaware and unaffected by that change. This independence from change is also referred to as encapsulation.

vii.) Security:-

All the data is stored on the servers, which generally have far greater security controls than most clients. Servers can better control access and resources, to guarantee that only those clients with the appropriate permissions may access and change data. Since data storage is centralized, updates to that data are far easier to administer than what would be possible

viii.) Friendliness

Many mature client-server technologies are already available which were designed to ensure security, 'friendliness' of the user interface, and ease of use. It functions with multiple different clients of different capabilities.

2. The Demerits of Client/Server Computing (Comparison with the P2P)

i.) Traffic congestion on the network has been an issue. As the number of simultaneous client requests to a given server increases, the server can become severely overloaded.

Contrast that to a P2P network, where its bandwidth actually increases as more nodes are added, since the P2P network's overall bandwidth can be roughly computed as the sum of the bandwidths of every node in that network.

ii.) The client-server paradigm lacks the robustness of a good P2P network. Under client-server, should a critical server fail, clients' requests cannot be fulfilled. In P2P networks, resources are usually distributed among many nodes. Even if one or more nodes depart and abandon a downloading file, for example, the remaining nodes should still have the data needed to complete the download.

Web servers:-

The web server software offers access to documents stored on the server. Clients can browse the documents in a web browser. The documents can be for example static Hypertext Markup Language (HTML) files, image files or various script files, such as Common Gateway Interface (CGI), JavaScript or Perl files. The communication between clients and server is based on the Hypertext Transfer Protocol (HTTP). When a document on the web server is requested, an HTTP request is constructed and sent to the server. The HTTP request is encapsulated in a TCP segment. The TCP contains the port address (the destination application on the server, i.e. the web server software) and is in its turn encapsulated into IP packets containing the IP address of the server. On local area networks, the IP packets are put into Medium Access Control (MAC) frames that address the next computer on the route towards the server. Figure shows the decomposition of an HTTP request into TCP segments, IP packets and MAC frames.

An HTTP transaction consists of three steps: TCP connection setup, HTTP layer processing and network processing. The TCP connection setup is performed as a so called three-way handshake, where the client and the server exchange TCP SYN, TCP SYN/ACK and TCP ACK messages. Once the connection has been established, a document request can be issued with an HTTP GET message to the server. The server then replies with a HTTP GET REPLY message. Finally, the TCP connection is closed by sending TCP FIN and TCP ACK messages in both directions. Figure illustrates the TCP communication.

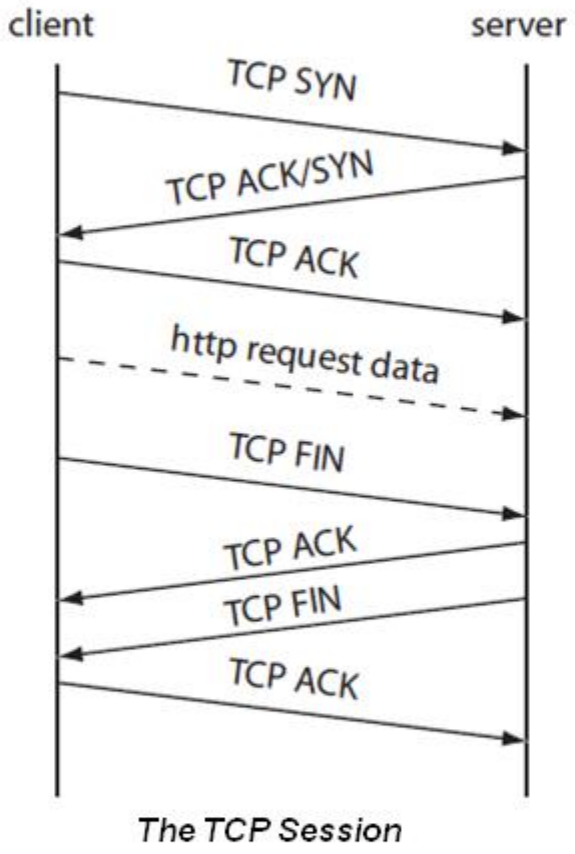


Figure 1.04

There are many web servers on the market today. Four main types can be identified; process-driven, threaded, event-driven and in-kernel web servers. Threaded and process-driven web servers are the most common, with Apache being the most popular currently.

HTTP Request message

A request message from a client to a server includes, within the first line of that message, the method to be applied to the resource, the identifier of the resource, and the protocol version in use.

Request = Request-Line
*(general-header
| request-header
| entity-header)
CRLF
[message-body]

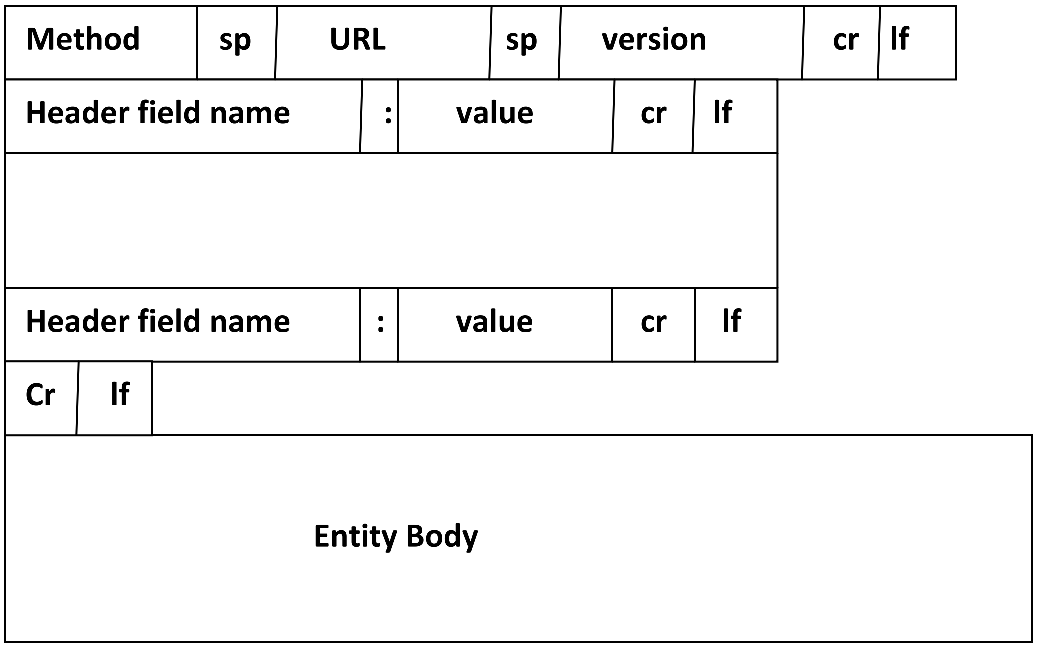
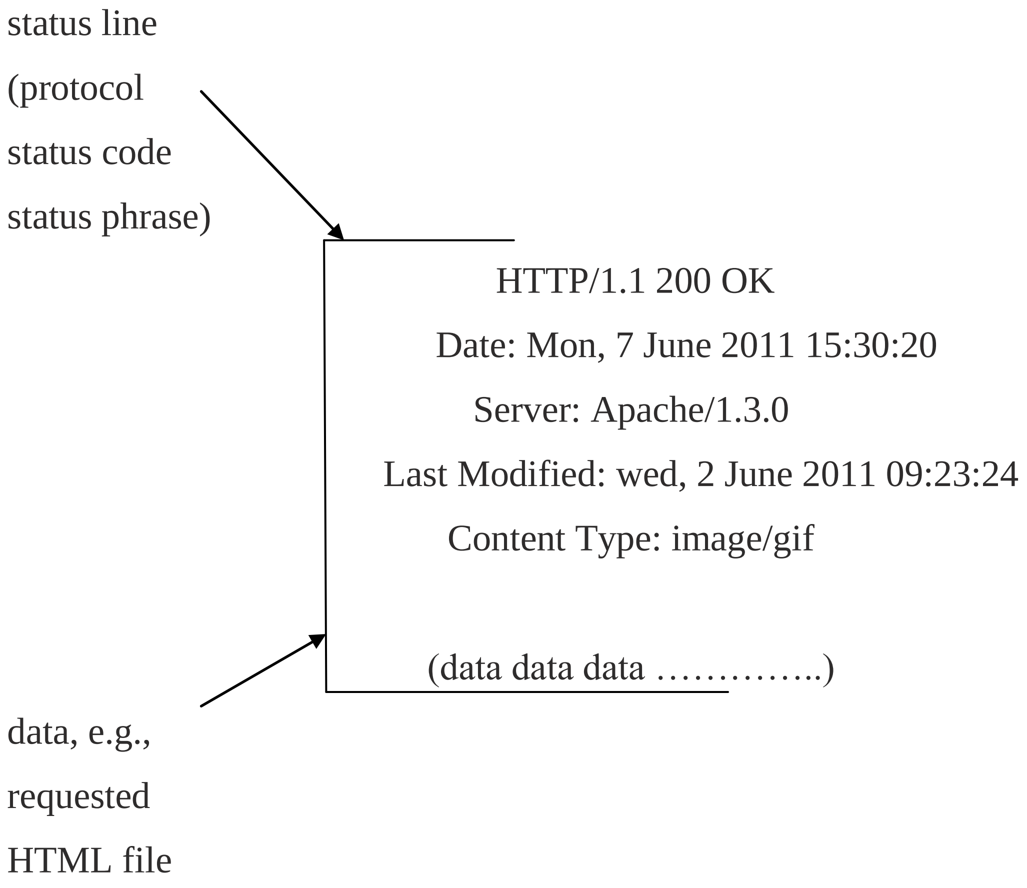


Figure1.05 HTTP Request Format

HTTP Response message

After receiving and interpreting a request message, a server responds with an HTTP response message.


status line (protocol status code status phrase)
data, e.g., requested HTML file

```
HTTP/1.1 200 OK
Date: Mon, 7 June 2011 15:30:20
Server: Apache/1.3.0
Last Modified: wed, 2 June 2011 09:23:24
Content Type: image/gif

(data data data …………..)
```

File Transfer Protocol

FTP is a protocol for moving files from one computer or server to another. These files may be stored in their new location for storage or for viewing/downloading by others on the World Wide Web.

To transfer the file, two connections are used: -

1) the control connection and
2) The data connection.

The control connection is used to send subcommands from the client to the server and receive responses to those commands from the server to the client. The client initiates FTP commands to the FTP server. The data connection is used to transfer the actual files. Both the client and the server interface with the i5/OS file system.

In a typical FTP session, user sits in front of one host (local hast) and wants to transfer files to or from a remote host. In order for the user to access the remote account, the user must provide a user identification and a password. After providing this information, the user can transfer files from local file system to remote file system and vice-versa. As shown in figure below, the user interacts with FTP through an FTP user agent. The user first provide hostname of the remote host , causing the FTP client process in local host to establish a TCP connection with the FTP server process in remote host. The user then provide the user id and password, which get sent over the TCP connection as part of FTP commands. Once the server has authorized the user, the user copies one or more files stored in the local file system into remote file system.

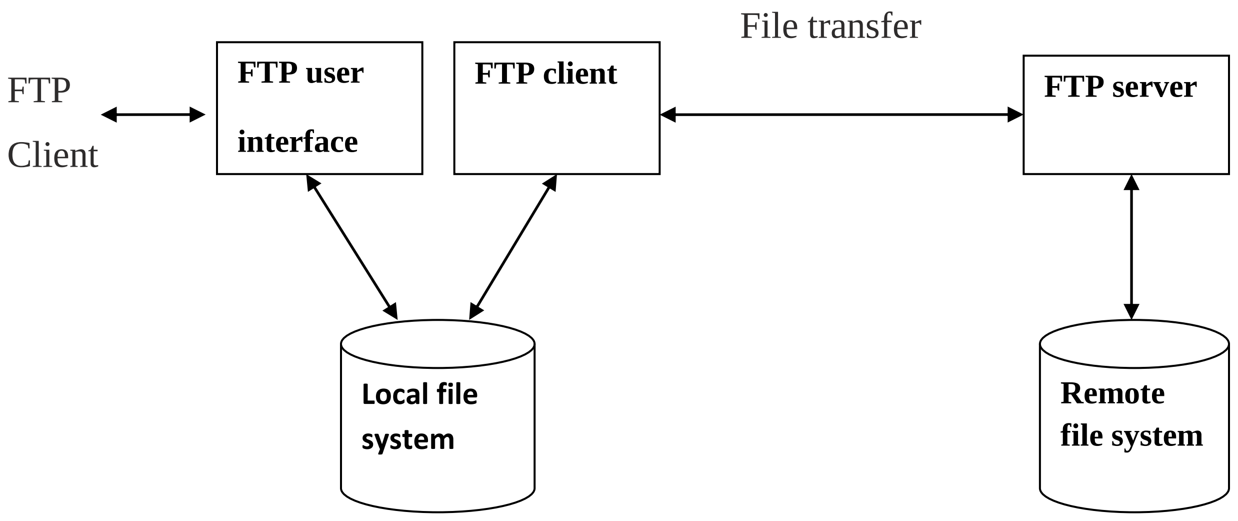


Figure 1.06 FTP moves between local and remote file system

Apache

Introduced in 1995 and based on the popular NCSA http 1.3, Apache is now the most used web server in the world. It is used in more than 67 percent of all web server systems (more than 52 millions in total, July 2004). One of the reasons to its popularity is that it is free to use. Also, since the source code is free, it is possible to modify the web server. Being threaded (threaded or process-driven depending on the

operating system, on Unix, Apache uses processes, while threads are used in Win32 environments) means that Apache maintains a pool of software threads ready to serve incoming requests. Should the number of active threads run out, more can be created? When a request enters the web server, it is assigned one of the free threads that serve it throughout the requests' lifetime. Apache puts a limit on the number of threads that are allowed to run simultaneously. If that number has been reached, requests are rejected.

2.) Software Requirement Specification

2.01.) <u>Introduction:</u>

1.) Purpose: The purpose of this Project is to make you understand how client server architecture works. The project itself consists of a File Transfer Protocol Server, a web server application and normal client server architecture explained through the Banking Enterprise Example. A single client can connect to the multiple servers at a time. Sometimes if a data frailer occurs at a particular server recover is made with the data stored at some other server.

2.) Scope:

- Check the authentication of each user when he/she first Interact with server.
- Create different system users and assign different roles with related permissions.
- Manage all the Information about Authorized use and the level of Authentication.
- Manage the details of registered user.
- Maintain the services provided to the user.

3.) Abbreviations:

a.) About us: Details of contact persons associated with the Administrator.

b.) PHP :

- PHP stands for "personal home page hypertext pre-processor".
- PHP is a "server side scripting language".
- PHP scripts are executed on server.
- PHP is open source software.
- PHP is free to download and use.

c.) HTML: **"**Hypertext Markup Language" is a markup language used to design static web pages.

d.) HTTP: "Hypertext Transfer Protocol" is a transaction oriented client/server protocol between web browser & a Web Server.

e.) HTTPS: **“**Secure Hypertext Transfer Protocol” is a HTTP over SSL (secure socket layer).

e.)TCP/IP: **“**Transmission Control Protocol/Internet Protocol”, the suite of communication protocols used to connect hosts on the Internet. TCP/IP uses several protocols, the two main ones being TCP and IP.

4) References:

- IEEE SRS Format
- Problem Definition (Provided by Mr. Alok Singh Gehlot)

5) Technologies:

-JAVA: Application Architecture
- PHP : Application Architecture
- MYSQL: Database
- MSACCESS: Database
- APACHE: Web-Server

2.02.) <u>Overall Description</u>: Describe the general factors that affect the product and its requirements.

1) Product Perspective:

- The web pages (HTML/PHP) are present to provide the user interface on client side.

-Communication between customer and server is provided through HTTP/HTTPS protocols.

- The Client Software is to provide the user interface on system user client side and for this TCP/IP protocols are used.
- At the server side different servers are running they provide the services like FTP, Web etc.
- On the server side web server is for APACHE and database server is for storing the information.

Application	Client-side	Server-Side
Application architecture	JAVA	JAVA
Web-technology	HTML	PHP
Server	---------	FTP ,Web, Apache
Database	----------	MYSQL,MSACCESS

Table 2.1

2) Software Interface:

Client on Intranet: Client Software, Web Browser, OS (any)
Web Server: Apache, Operating System (any)
Data Base Server: MYSQL, MSACCESS.
Development End: JAVA, PHP, HTML, OS (Windows), Apache Web Server.

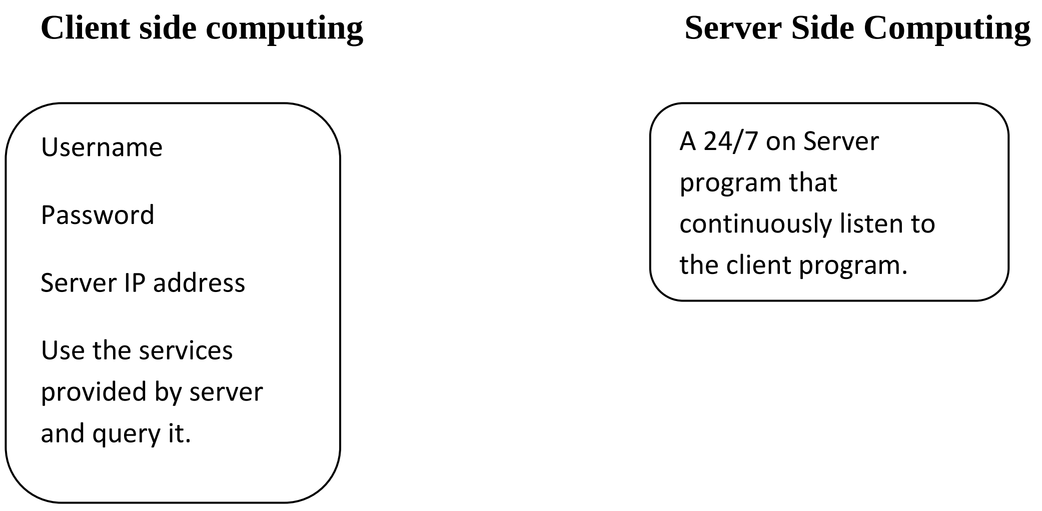
3) Communication Interface:

- Client on Internet will be using HTTP/HTTPS protocol.
- Client on Intranet will be using TCP/IP protocol.

3) User Characteristics:

-Every user should be comfortable of working with computer and net browsing.
-He must have basic knowledge of English too.

On-Line Courses Blue-Print


Client side computing
Server Side Computing
Username
Password
Server IP address
Use the services provided by server and query it.
A 24/7 on Server program that continuously listen to the client program.

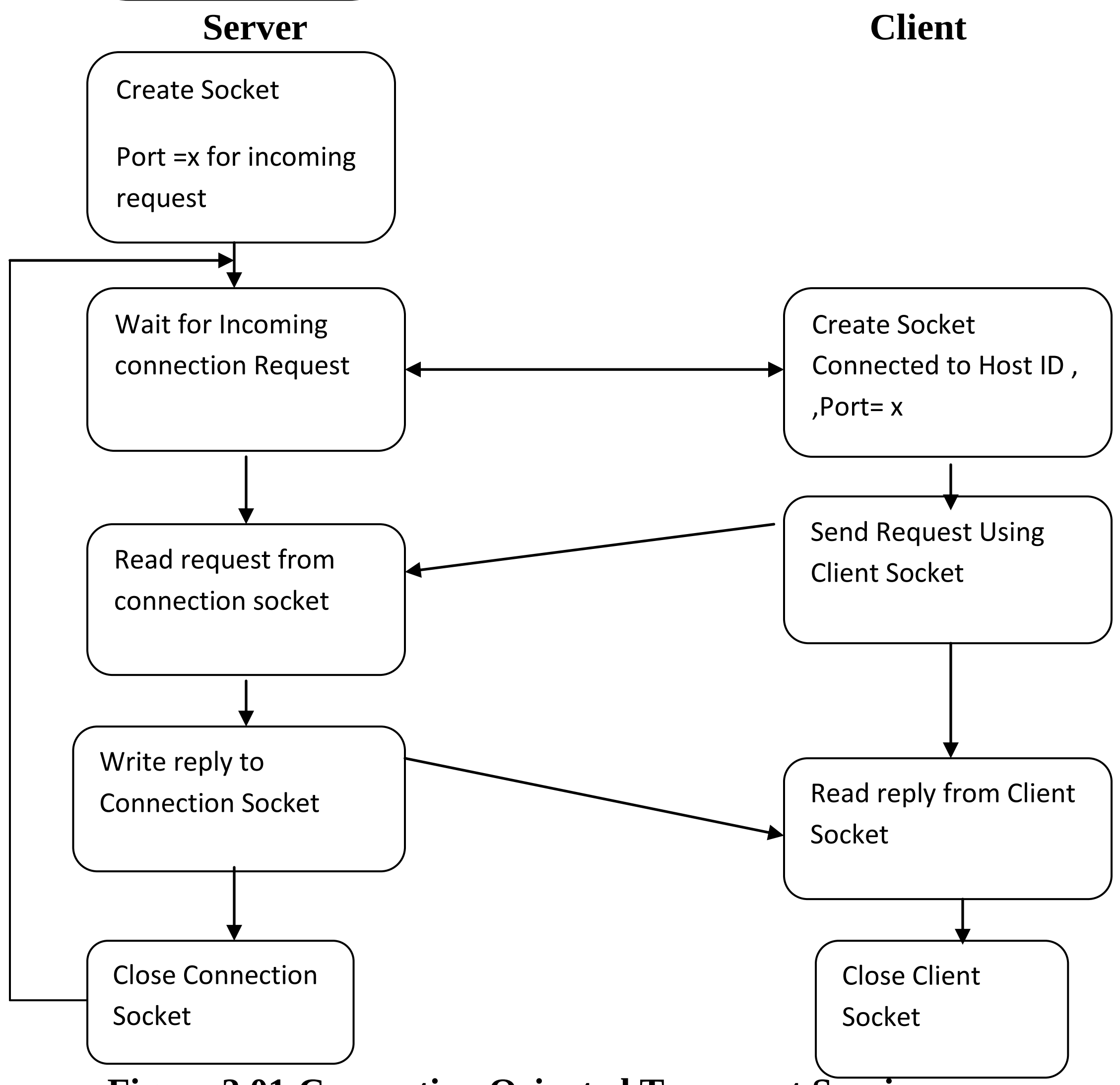

Server
Client
Create Socket
Port =x for incoming request
Wait for Incoming connection Request
Create Socket Connected to Host ID , ,Port= x
Send Request Using Client Socket
Read request from connection socket
Write reply to Connection Socket
Read reply from Client Socket
Close Connection Socket
Close Client Socket

Figure 2.01 Connection Oriented Transport Services

5) Constraints:

- GUI is only in English.
- Login and password is used for identification of Authorized User.
- This system is working for multiple server.
- There is no maintainability of back up so availability will get effected.
- Limited to HTTP/HTTPS.

6) USE CASE DIAGRAM

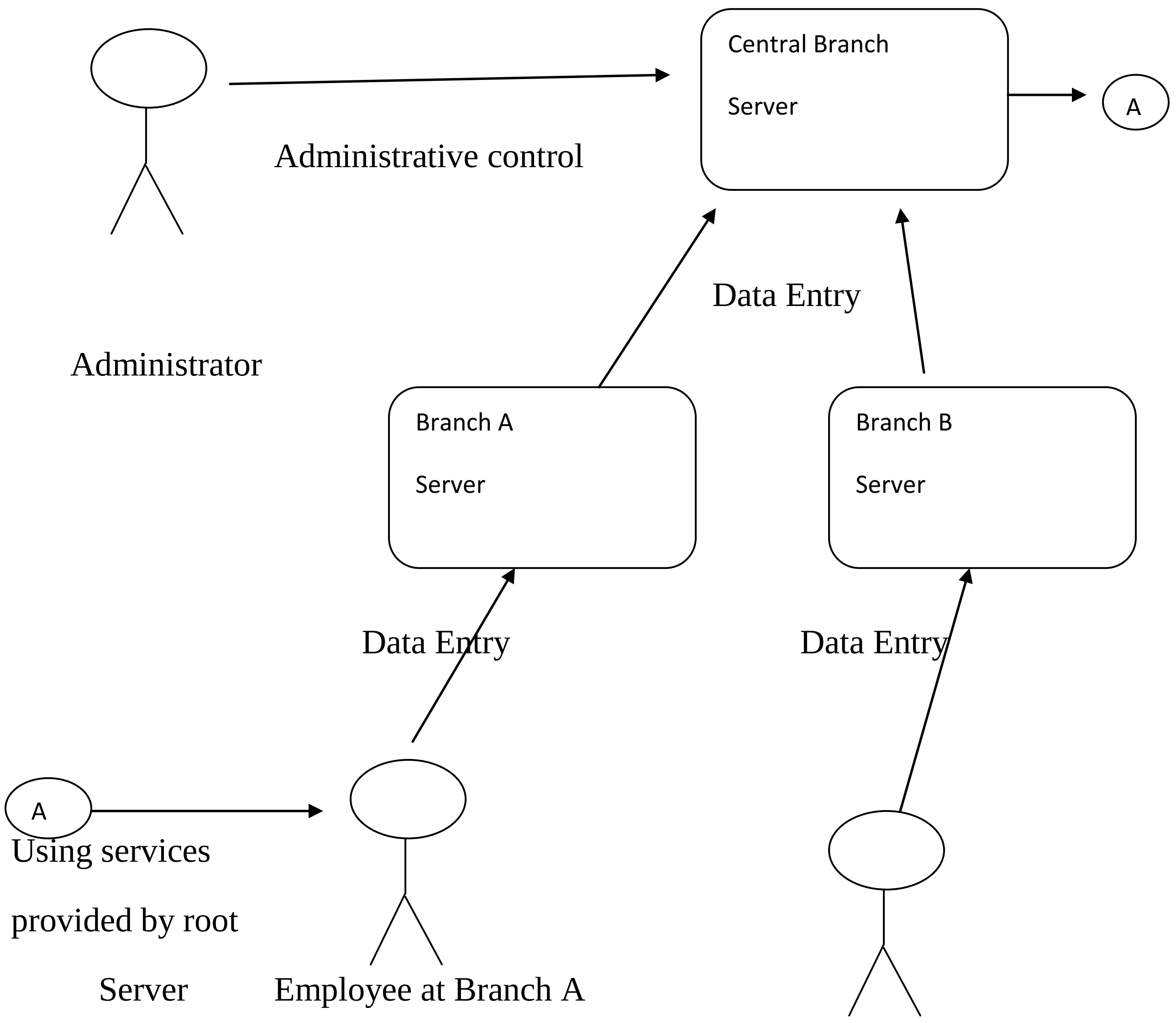


Figure 2.02

This user case shows that data entered by employees of various Branches is stored at the respective branch server as well as at the central server. When data at particular branch is lost recovery is made through the central server. Employees can also use the services provided by the central server e.g. SMTP , FTP etc.

3.) High Level Design

Components: -

In the project the main Components are the Client and Server System. Client sends queries and according to particular query Server takes the particular action. Like when Client wants to know the Balance of a Customer he enters the account number of the Customer Server reads the account number and give the total balance information to client.

The Client Subsystem: -

Client Subsystem includes the client System interconnected by UTP cables the System is multi user System, so that multiple client can be attached to the Server at a time. However the System gets hanged if the load on the server increases drastically to remove the type of problem we need power full Server Systems.

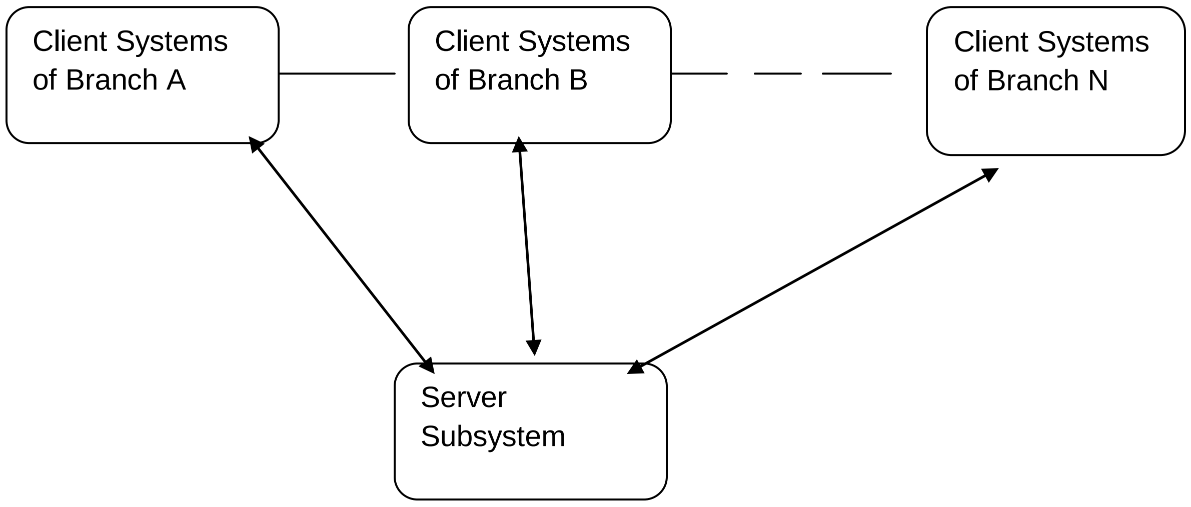

Client Systems of Branch A
Client Systems of Branch B
Client Systems of Branch N
Server Subsystem

Figure 3.01 Client subsystem uses the Services provided by the server

The Server Subsystem: - Server Subsystem includes all Server System attached to the whole System like Central Server, Branch Server etc. The main difference between the Client subsystem and Server Subsystem is Client systems are not

connected directly to each other they can communicate via some server system only on the other hand the Server system are connected directly to each other.
When data are entered by some client at a certain branch, we have to ways to Distribute it either we first store it only at the branch server and then at the end of a particular time (each day) we store the data from each branch to the central Server or we can do it at the same time when we entered it first. The Server Subsystem also provides web services and FTP Services to the Clients.

To provide Web Services to the Client Server Subsystem must use the HTTP protocol (HTTP Request and Response Messages) according to the Format given by (Networking RFC 2616).

The Server Subsystem

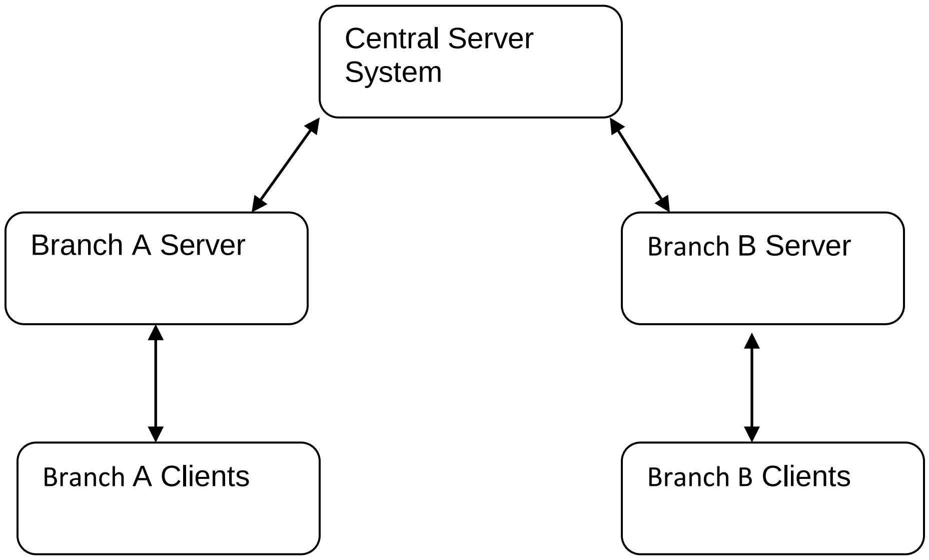


Figure 3.02 Server Subsystem

Interconnection and Interaction between the Components:-

When data at particular branch is lost recovery is made through the central server. Employees can also use the services provided by the central server e.g. SMTP, FTP etc. In the Distributed Environment when Client at a certain branch ask for the records of a particular customer who does not belong to that branch the branch Server rout that query to the Central server the it made the duplicate copy of the record to it self as well for the further concert. Periodic back up should be done at the Branch server as well as the Central server site. When Client system gives queries to the Server system there is a particular transaction control mechanism which handles for multiple queries.

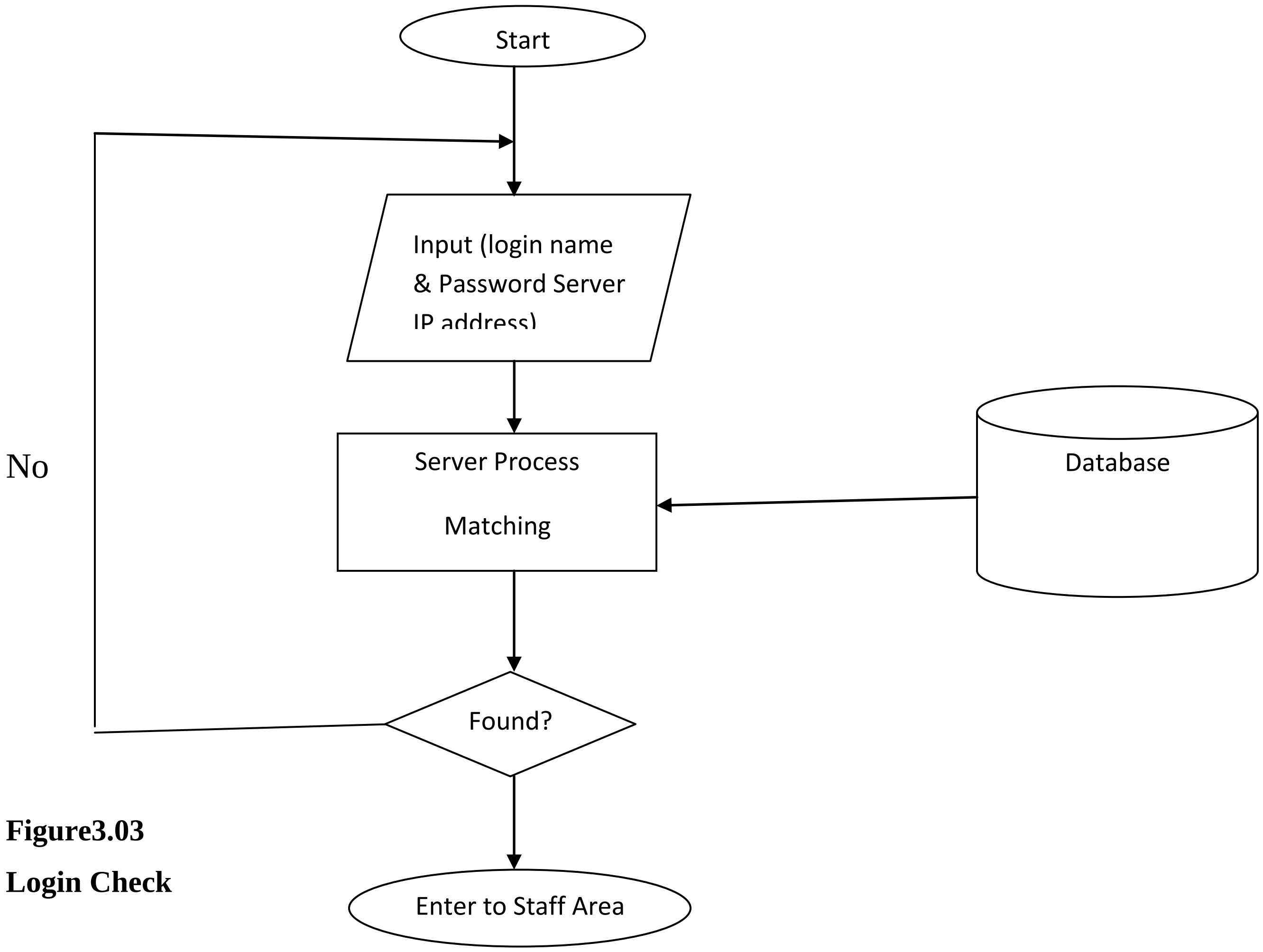


Figure3.03
Login Check

Working of the Web Server Component:-

The web Server attached to the whole application works as like, We have a 24*7 on Server System at the Central Server like (Apache) when Client ask for a particular page the Web Server first finds is it a static web page or a dynamic if static the it directly return it to the asking Client machine or otherwise if dynamic page request it first invoke

the apache server then with the help of it our web server make an equivalent static web page then this static web page is sent to the Client.

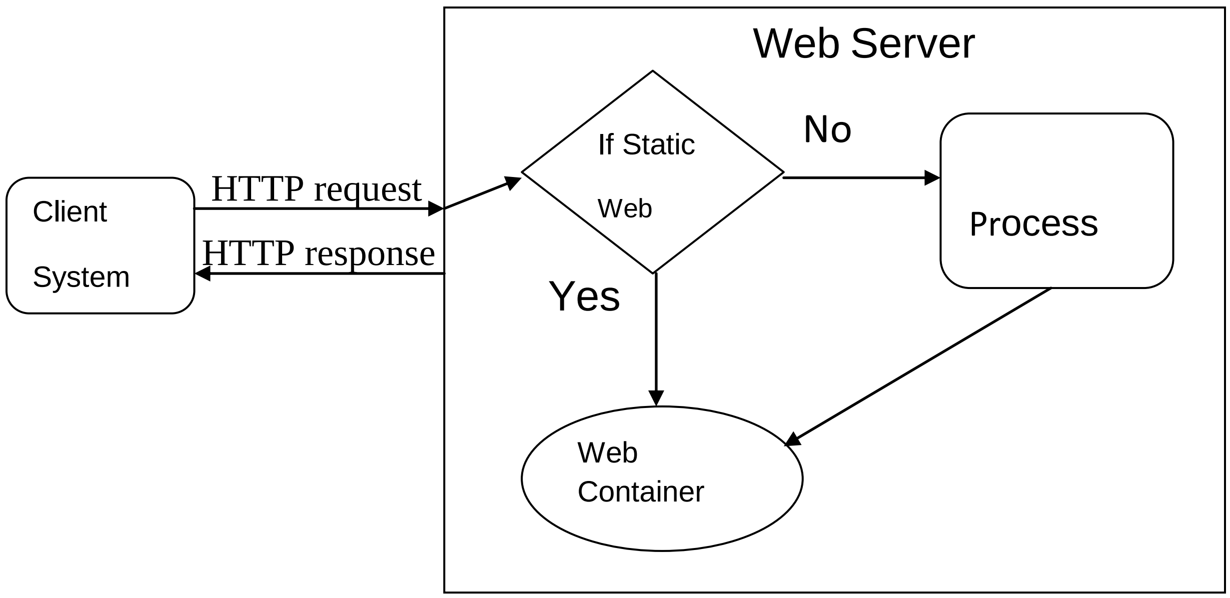


Figure 3.04 Web servers working

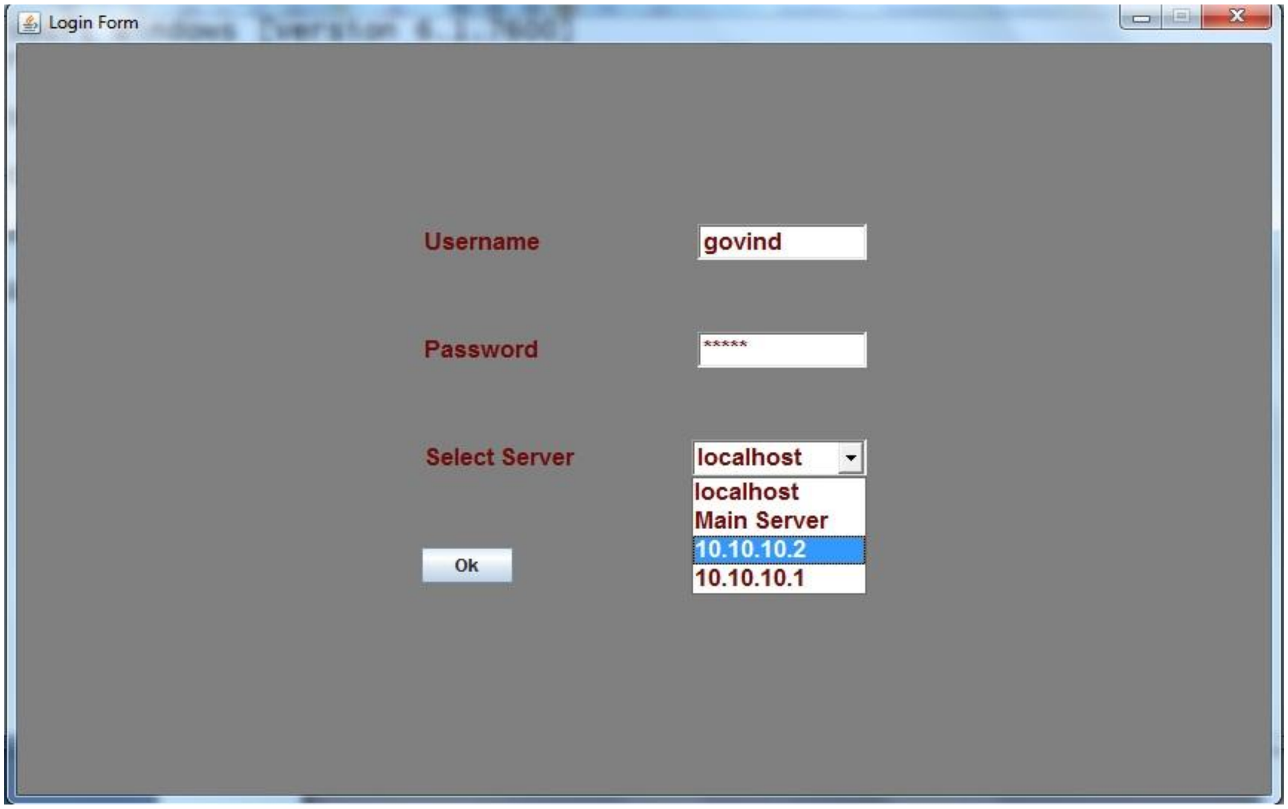
Coding

The goal of the coding activity is to develop correct programs that are also clear and simple. As reading programs is a much more common activity than writing programs, the goal of the coding activity is to produce simple programs that are easy to understand and modify and that are free from errors. Ease of understanding and freedom from defects are the key properties of high quality code.

As we describe we take Banking Enterprise for the explanation of client server architecture.

Here I present some snap shoot ‘s for the project to made you understand how everything going to work.

Client Side Coding



Username govind

Password *****

Select Server localhost

localhost
Main Server
10.10.10.2
10.10.10.1

Figure 4.01 Login form

Here I show login screen that is to be appear in front of a Employee of the bank when it loads the program.

The login screen ask for the username and the password the it ask for the server's IP address to whom client want to connect. When client gives the correct username password and IP address of the client and submitted the form the respective server then checks for the authentication of the client and accordingly redirect him to the home page of the bank or incorrect login.

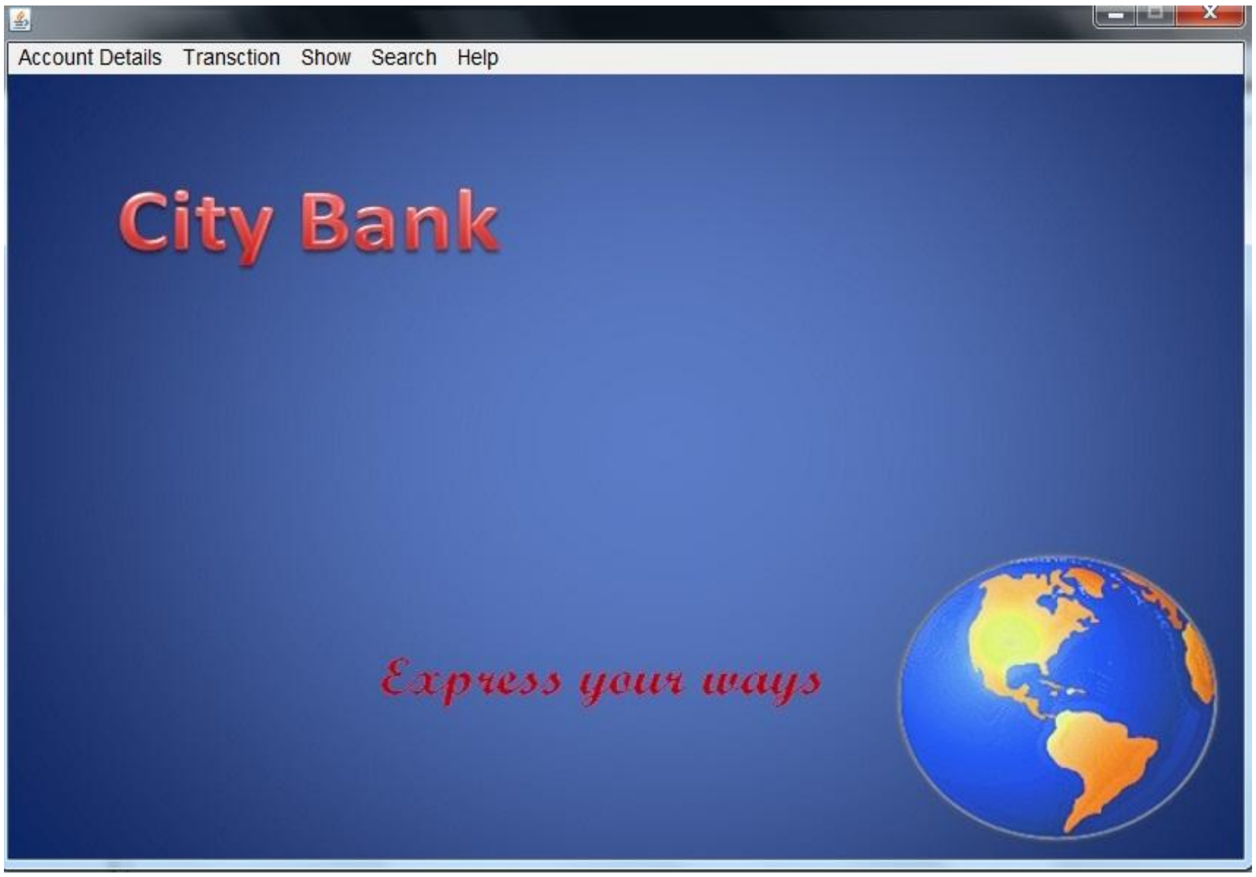
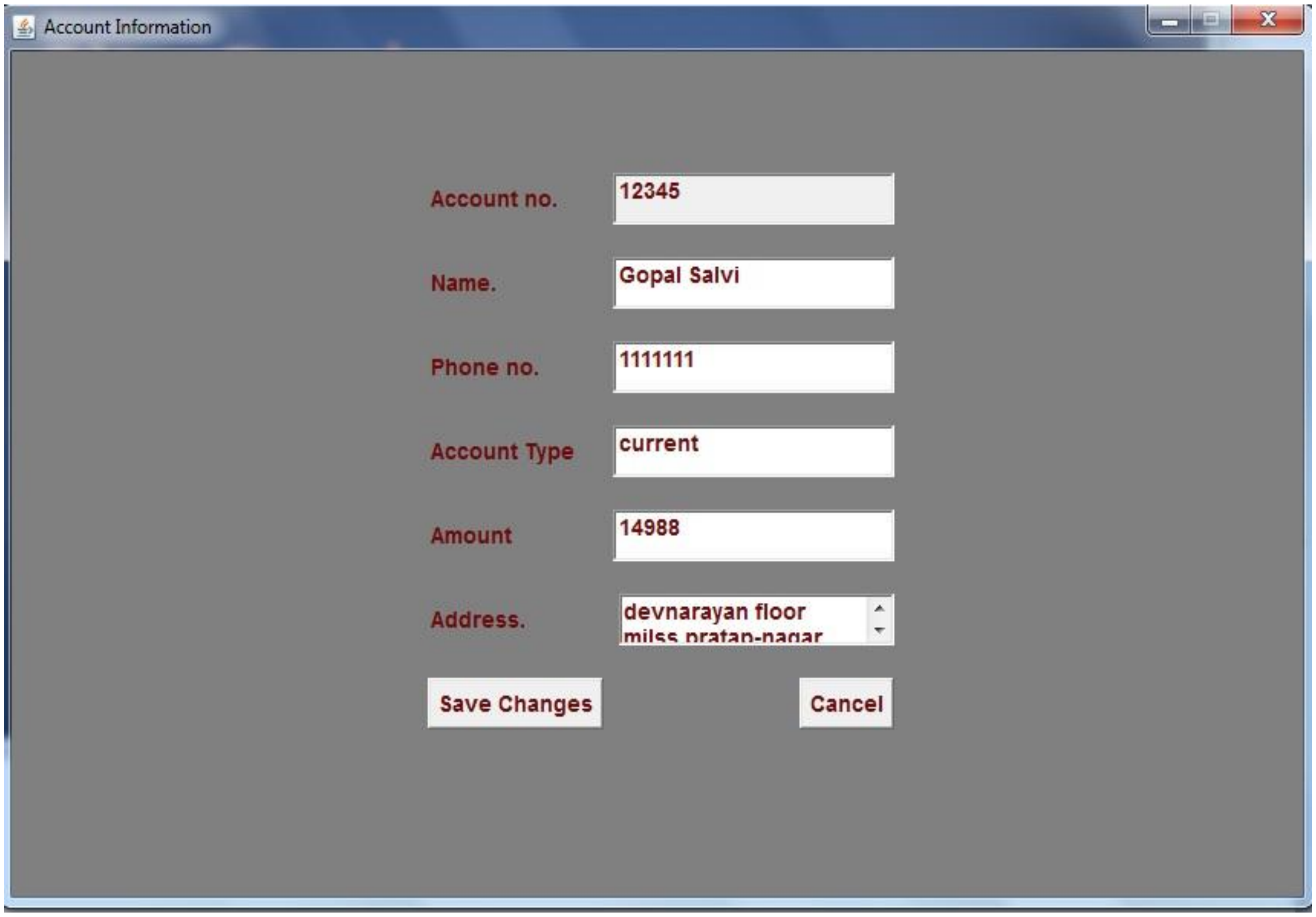


Figure 4.02 Home page of the bank

On the home page we can see various menus like Account Details, Transaction, Show ,Search and help menu. When we click on the particular menu we get the sub menus under the respective categories.

Now we show you the new account page when you fill the corresponding entries to this page and submit whole entries are get submitted to the server to whom your currently connected. As the database is on the server side the database entries are submitted to the server's database and corresponding message is shown to the client.



Account no.	12345
Name.	Gopal Salvi
Phone no.	1111111
Account Type	current
Amount	14988
Address.	devnarayan floor milss pratap-nagar

Figure 4.03 New Account Page

When you want to delete an account you have to submit a authentication to the server like the following snap shoot shows.

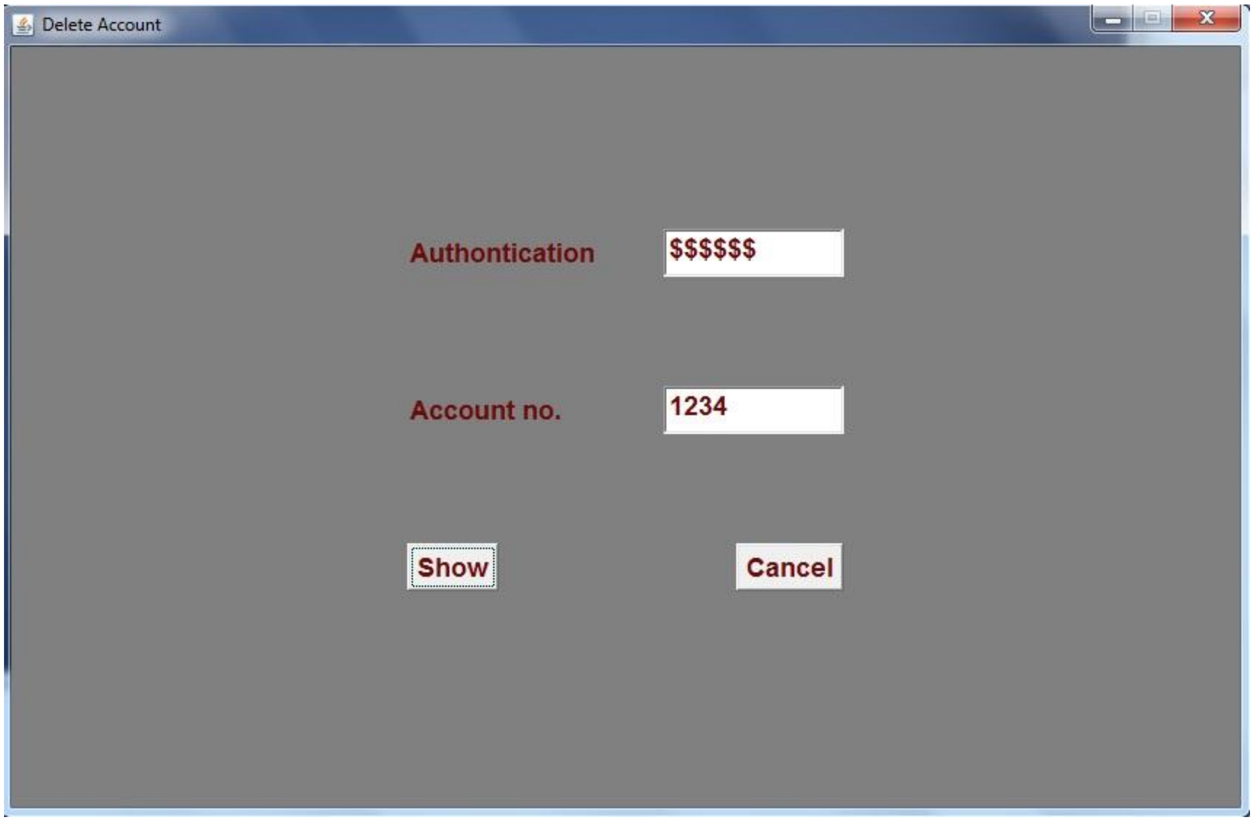


Figure 4.04 authentication check

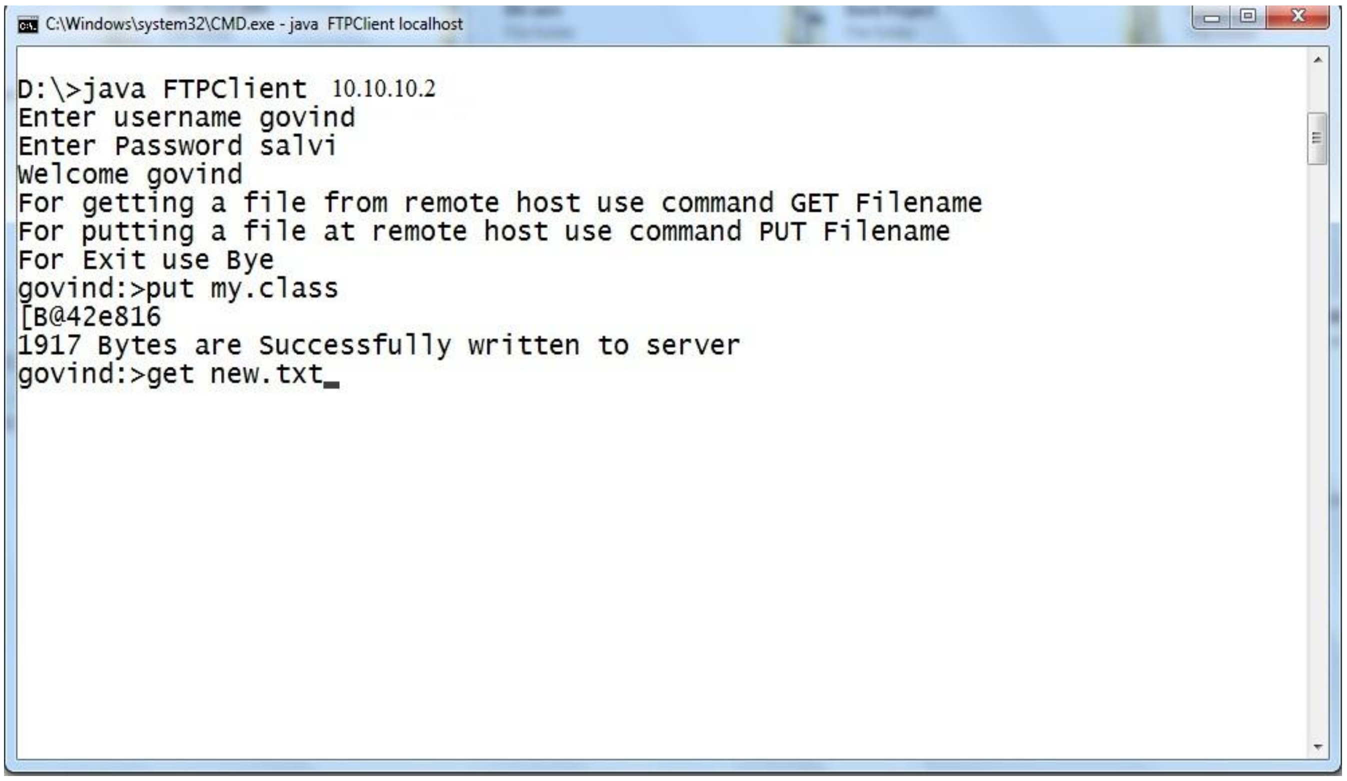
Client side coding for the File Transfer Protocol Server

For connecting to the remote host with FTP client we have to first load the program FTPClient with the IP address of the server the following snap shoot shows how the environment is present in front of the FTP client. When he/she provide the correct username and password now he can use the Following two commands

PUT Filename

GET Filename

Put command put the file to remote host and get command get a file to remote server.



```
D:\>java FTPClient 10.10.10.2
Enter username govind
Enter Password salvi
Welcome govind
For getting a file from remote host use command GET Filename
For putting a file at remote host use command PUT Filename
For Exit use Bye
govind:>put my.class
[B@42e816
1917 Bytes are Successfully written to server
govind:>get new.txt_
```

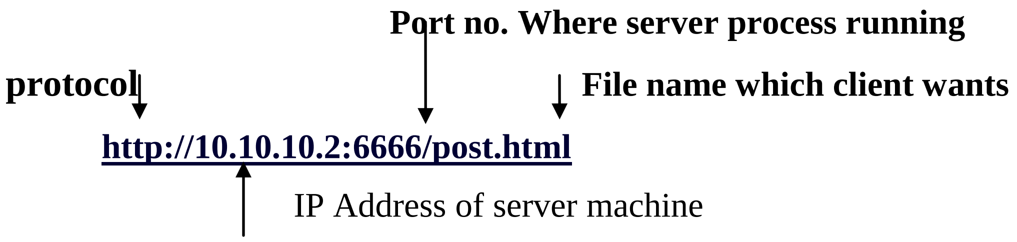
Figure4.05 FTPClient side program

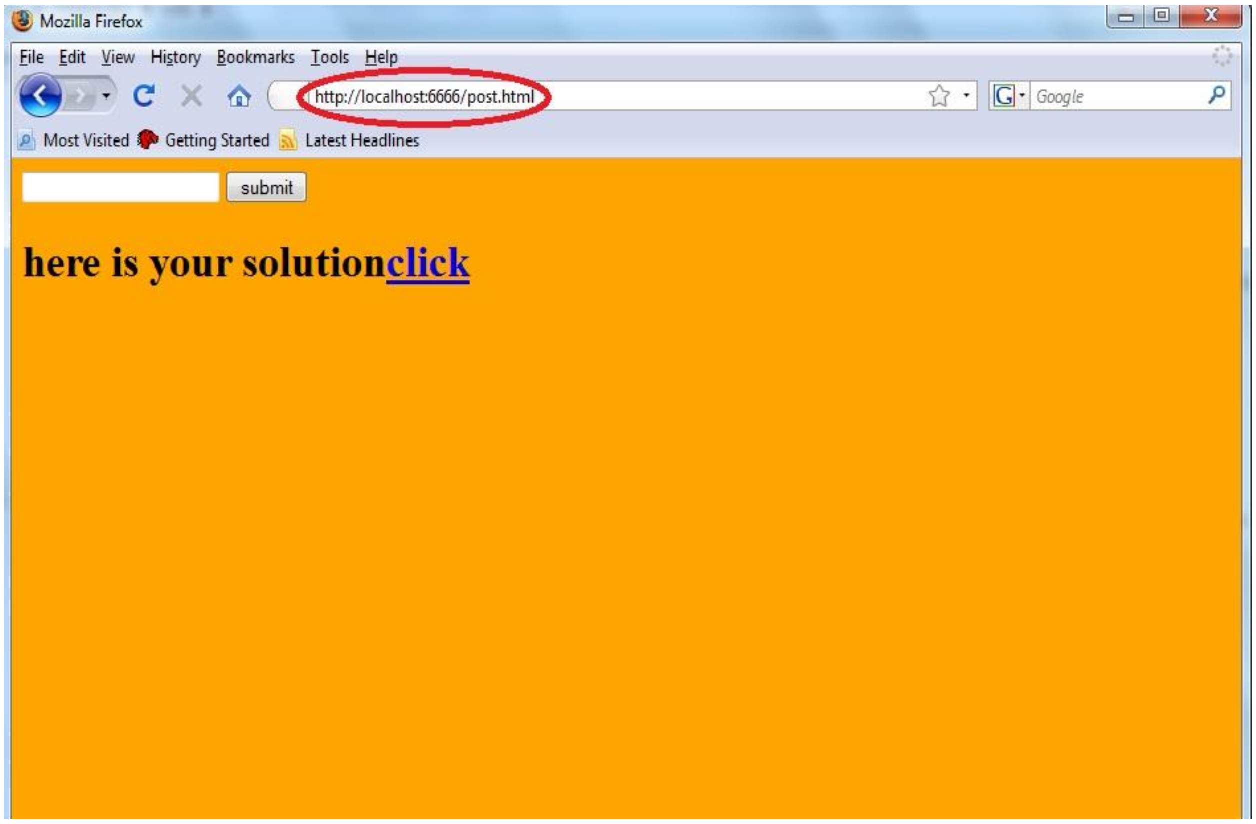
Here we connect to the FTP server running on the machine having IP address 10.10.10.2 then we give the user name and password. The welcome note are shown in front of Client and redirected to his personal commend Interpreter.

Here we put the file named my.class to server and server gives the notification that 1917 Bytes are successfully written to it.

Getting a static page from web server

On some of the server machine our web server program is running 24*7 .when a client wants to use services provided by it .he/she first open up the browser and give the uniform resource locator (url) like wise.


Port no. Where server process running
protocol
File name which client wants
http://10.10.10.2:6666/post.html
IP Address of server machine


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:6666/post.html
Google
Most Visited Getting Started Latest Headlines
submit

here is your solutionclick

Figure4.06

Here on the address bar we can see the url.

Getting a dynamic page from server. To get a dynamic page from server we have to give the file name of the desired file to the browser like wise. when the server find the file in its directory it don’t find it and then throws an Exception Which is handled by apache server and our server became a client for the apache server. Apache server convert the dynamic page to static page and the our server program gives this static page back to the client.

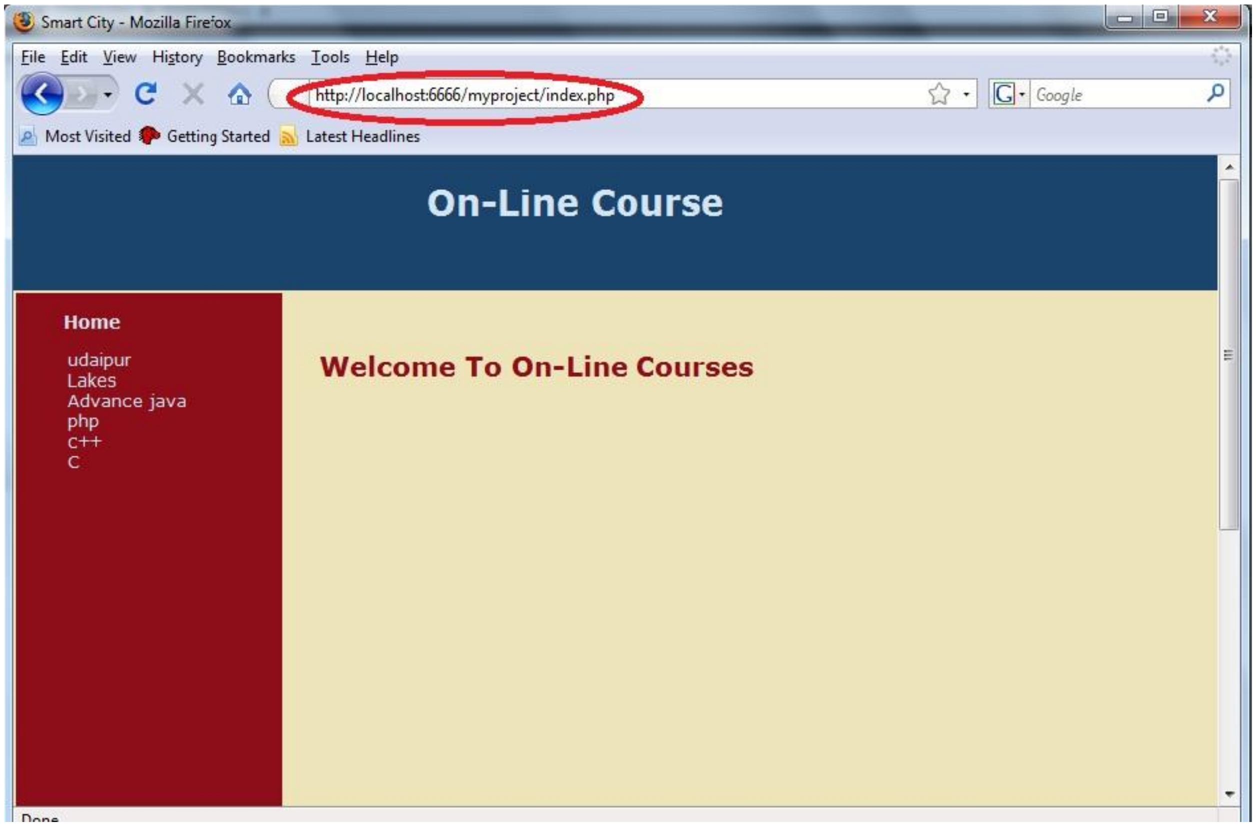

Smart City - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:6666/myproject/index.php
Google
Most Visited Getting Started Latest Headlines
On-Line Course
Home
udaipur
Lakes
Advance java
php
c++
C
Welcome To On-Line Courses
Done

Figure 4.07

Server side computing...

To describe how various servers are working we are presenting source code of some of the critical server files

Source code of web server

```
import server.*;
import java.io.*;
import java.net.*;
import java.util.*;
class server {
public static void main(String[] args) throws Exception {
String s1="";
String filename;
String modi;
ServerSocket ssocket=new ServerSocket(6666);

//BufferedReader infromuser;
//DataOutputStream fromserver;
while(true)
        {
BufferedReader infromuser;
DataOutputStream fromserver;
Socket connectionsocket=ssocket.accept();
infromuser= new BufferedReader(newInputStreamReader
(connectionsocket.getInputStream()));
fromserver=new DataOutputStream(connectionsocket.getOutputStream());
 s1=infromuser.readLine();
StringTokenizer tokeniz=new StringTokenizer(s1);
String getorpost=tokeniz.nextToken();

if(getorpost.equals("GET")||getorpost.equals("POST"))
     {
        filename=tokeniz.nextToken();
        if(filename.startsWith("/")==true)
        {
        filename=filename.substring(1);
        }
        File file;
        int numOfBytes;
```

```
FileInputStream infile;
        try
        {
        file=new File(filename);
                numOfBytes=(int)file.length();
               infile=new FileInputStream(filename);
        }
        catch(Exception e)
        {

        //System.out.print("Exception");
        //try
        //{
          tcpc tcp=new tcpc(s1,infromuser);
          filename="out.html";
         file=new File(filename);
         numOfBytes=(int)file.length();

        infile=newFileInputStream(filename);
        //}
        /*catch(Exception ee)
         {

        //tcpc tcp=new tcpc(s1,infromuser);
         filename="error.html";
          file=new File("error.html");
```

```
                  numOfBytes=(int)file.length();
                infile=new FileInputStream(filename);
                }
                        */
                }
                byte[] fileinBytes=new byte[numOfBytes];
                infile.read(fileinBytes);

                fromserver.writeBytes("HTTP/1.0 200 ok\r\n");
                String type;
                type=mime.getmime(filename);

                fromserver.writeBytes("Content-Type:"+type+"\r\n");
                fromserver.writeBytes("\r\n");

                fromserver.write(fileinBytes,0,numOfBytes);
                  }
                else
                        System.out.println("Bad");connectionsocket.close();
              }
        // connectionsocket.close();

}

}
```

FTP Server code

```
package FTP;
import java.net.*;
import java.io.*;
import java.util.*;
public class FTPServerThread extends Thread
{
        Socket s=null;
        DataOutputStream toclient=null;
        DataInputStream input=null;
        DataOutputStream toclient1=null;
        DataInputStream input1=null;
        public FTPServerThread(Socket socket)
        {
        this.s=socket;
        }
        public void run()
        {
        while(true)
        {
                try
                {
                                //BufferedReader  fromclient=new  BufferedReader  (new
                InputStreamReader(s.getInputStream()));
                toclient=new DataOutputStream(s.getOutputStream());
```

```
input=new DataInputStream(s.getInputStream());
toclient1=new DataOutputStream(s.getOutputStream());
input1=new DataInputStream(s.getInputStream());
String username="",password="";
username=input.readLine();
password=input.readLine();
int xx=FTPLoginCheck.LoginCheck(username,password);
String x=String.valueOf(xx);
toclient.writeBytes(x+"\n");
String first=input.readLine();
String filename=input.readLine();
//System.out.println(filename);
String line;
if(first.equals("get"))
{
        try{
        System.out.println("i am in get");
        FileInputStream fileread=new FileInputStream(filename);
                                File file=new File(filename);
                                System.out.println(filename);
                                int c=(int)file.length();
                                String size=String.valueOf(c);
                                toclient.writeBytes(size+"\n");
                                byte b[]=new byte[c];
                                fileread.read(b);
                                System.out.println(b);
```

```
                                        toclient.write(b,0,c);
                                                       /*

                              while((line=fileread.readLine())!=null)
                              {

              toclient.writeBytes(line+"\n");
          }

              toclient.writeBytes("EOF"+"\n");
                                                            */
              //fileread.close();
              }
                   catch(Exception exp)
                   {}
                        }
                        else if(first.equals("put"))
                        {

                              try
                              {
                                   //System.out.println("i am in put");
                              FileOutputStream fout=new FileOutputStream(filename);
                                             String size=input1.readLine();
                                             int c=Integer.parseInt(size);
```

```
                                                  //System.out.println(filename+c);
                                                  byte b[]=new byte[c];
                                                  //input.read(b,0,c);
                                                   int r=input1.read(b);
                                                   String r1=String.valueOf(r);
                                                  System.out.println("put"+b);
                                                   fout.write(b,0,c);
                                                   toclient1.writeBytes(r1+"\n");

          /*while((c=fromclient.read())!=-1)
          {
          fout.write((char)c);
                    }
     fout.write((char)-1);
          */
          //fout.close();
                    }
                    catch(Exception e)
                    {
                              System.out.println("error");
                    }


                              }
                    //input.close();
                    //toclient.close();
                    }
```

```
                catch(Exception e)
                {}


        }       }
```

Banking server code

```
import work.*;

import java.io.*;

import java.sql.*;

import java.net.*;

public class s

{

        public static void main(String a[]) throws  Exception

        {

        String user,pass,get,send;

        ServerSocket ssocket=new ServerSocket(6565);

                while(true)

                {

                Socket csocket=ssocket.accept();

                BufferedReader fromclient=new BufferedReader(new
InputStreamReader(csocket.getInputStream()));

                DataOutputStream toclient=new
DataOutputStream(csocket.getOutputStream());

                int n=Integer.parseInt(fromclient.readLine());

                switch(n)
```

```
{
case 1:         //login check
        {
        user=fromclient.readLine();
        pass=fromclient.readLine();
        int x=logincheck.check(user,pass);
        send=String.valueOf(x)+"\n";
        toclient.writeBytes(send);
        } break;
case 2:         //delete
        {
        String ac=fromclient.readLine();
        int x=delete.check(ac);
        send=String.valueOf(x)+"\n";
        toclient.writeBytes(send);
        } break;

case 3:                         // new account
        {
        String acno=fromclient.readLine();
        String na=fromclient.readLine();
        String ph=fromclient.readLine();
        String type=fromclient.readLine();
```

```
            String ba=fromclient.readLine();
            String add=fromclient.readLine();
            int x=newac.check(acno,na,ph,type,ba,add);
            send=String.valueOf(x)+"\n";
            toclient.writeBytes(send);
            } break;
    case 4:                 // get account number
            {
            int x=acno.getacno();
            //System.out.print(x);
            send=String.valueOf(x)+"\n";
            toclient.writeBytes(send);
            } break;
    case 5:                       // search account
            {
            String acno=fromclient.readLine();
            searchac ac=new searchac(acno);
            toclient.writeBytes(ac.getac()+"\n");
            toclient.writeBytes(ac.getnam()+"\n");
            toclient.writeBytes(ac.getph()+"\n");
            toclient.writeBytes(ac.gettype()+"\n");
            toclient.writeBytes(ac.getba()+"\n");
            toclient.writeBytes(ac.getadd()+"\n");
            } break;
    case 6:                     //update
            {
            String acno=fromclient.readLine();
            String na=fromclient.readLine();
            String ph=fromclient.readLine();
            String type=fromclient.readLine();
            String ba=fromclient.readLine();
            String add=fromclient.readLine();

            int x=updateac.check(acno,na,ph,type,ba,add);
            send=String.valueOf(x)+"\n";
            //System.out.println(send);
            toclient.writeBytes(send);
            } break;
    case 7:                             // balance inquiry
            {
            String acno=fromclient.readLine();
            searchac ac=new searchac(acno);
            toclient.writeBytes(ac.getac()+"\n");
            toclient.writeBytes(ac.getnam()+"\n");
            toclient.writeBytes(ac.getba()+"\n");
            } break;
```

```
case 10:                        // search name account
        {
        String acno=fromclient.readLine();
        //System.out.print(acno);
        searchac ac=new searchac(acno);
        toclient.writeBytes(ac.getac()+"\n");
        toclient.writeBytes(ac.getnam()+"\n");


        } break;
case 11:                        // Deposit
        {
        String acno=fromclient.readLine();
        String money=fromclient.readLine();
        int x=deposit.check(acno,money);
        send=String.valueOf(x)+"\n";
        toclient.writeBytes(send);


        } break;
case 12:                        // Withdrwal
        {
        String acno=fromclient.readLine();
        String money=fromclient.readLine();
```

```
                    //System.out.print(acno+money);


                    int x=withdrwal.check(acno,money);
                    send=String.valueOf(x)+"\n";
                    toclient.writeBytes(send);


                    } break;
                }
                }
        }
}
```

Enhancement

- Authorization level could be more strict.
- Currently system is working on the basses of double entry. Means first entry is made to the respective branch server then to the central server. By applying the replication technique the limitation can be overcome.

-The Server Subsystem is now working as a private network in future it could be deploy on the public network.

-when we ask for the dynamic page the system redirect the request to apache web server In future we can create such server that could resolve the Dynamic web page by it’s own.

Part II

1.) Bibliography

Books

i.) **Computer Networking by kurose and ross.**

ii.) **Database system concepts by Korth and sudarshan.**

iii.) **Java – The complete reference by Herbert schildt.**