A

Project Report on

# Magnum Peer To Peer System (MPPS)

**SUBMITTED IN PARTIAL FULFILLMENT**
**FOR THE REQUIREMENT OF THE AWARD OF DEGREE**
**IN BACHELOR OF ENGINEERING (I.T)**

## Submitted By:


| | |
|---|---|
| **Kunjan Aggarwal** | **04-IT-526** |
| **Ashish Dhingra** | **04-IT-562** |
| **OmPrakash Yadav** | **04-IT-563** |


## Under the Guidance of:


***Dr. Pankaj Mohindroo*** ***Mr. Parveen Gupta***



Institute of
**Technology and Management**


DEPARTMENT OF COMPUTER SCIENCE & IT
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
SECTOR 23-A, GURGAON
(Affiliated to Maharshi Dayanand University, Rohtak)

MAY 2008

## Table Of Contents

# ***ACKNOWLEDGEMENT***

*We take this opportunity to express our profound sense of gratitude and respect to all those who helped us throughout the duration of the project.*

*This project was a fruitful, innovative and a great learning experience for all of us. Many people along the way mentored us and we wish to express our gratitude to everybody who helped us in all possible way. Their inputs, insights, guidance and technical knowledge made this project a worthwhile experience.*

*We express our gratitude to Dr. Ranjit Biswas, Head of Department, Computer Science and Information Technology for his cooperation and encouragement during the completion of this project.*

*We extend our utmost gratitude to Mr. Pankaj Mahindroo, our Project Guide, who has always stood by our side and appreciated and encouraged us to get into more and more ventures. He enlightened us during various stages of the development of this project and provided us with useful examples, which proved to be of immense help in the successful completion of this project.*

*We are extremely thankful to Mr. Parveen Kumar Gupta, Senior Lecturer, Computer Science and Information Technology Department, for his academic expertise, guidance, helpful suggestions and valuable ideas which encouraged us at each and every stage of our project.*

*We also extend our sincere gratitude to all our teachers who made unforgettable contribution. We thank all the non-teaching staff of our institution that was always ready to help in whatever way they could.*

**Kunjan Aggarwal**

**Ashish Dhingra**

**OmPraksh Yadav**

# ***Abstract***

*The project is a step towards understanding the true power of LAN s. We have LAN s everywhere but still we are just limited to using them for simple printer or file transfer using cumbersome tools like FTP and Telnet.*

*Magnum, as per U.S. Military Dictionary is a gun designed to fire cartridges that are more powerful than its caliber would suggest. This exactly is our aim as we try to put forward a simple LAN networking solution which presents to the world the true potential of LAN s which today is moving towards a slow and very unfortunate death..*

*Many a times one finds people in the same office communicating with each other through telecom, making notes of conversation and each other's extension no. on pads, white boards, forgetting the fact that they are actually connected to each other through LAN and thus they can directly CHAT with each through their PCs, thus removing the need of searching for each other's extension no. just to get a link to talk*

*Also a common phenomenon in institutions is of storing the important files on a limited set of systems and using them as a storehouse to retrieve this data from. Now if this storehouse crashes suddenly the whole office finds itself in a chaotic situation with everyone trying to ask everyone else if an urgently required file is lying with them on their system storage disk.*

*Why need a storage house if you can simply have an interface that just asks for a file name, matches it with names of all the files present on the LAN and gives the result back to you with not only the names but also the addresses where they are present, a short file description, and the last date when the file was modified, thus making it easy for you to decide which file is of most use to you.*

*Not only this, it can also recognize whether same file is present on many systems and hence use them as mirror links thus getting the file to you at many fold greater speeds.*

*This project is an attempt to make life easy for institutions with systems with lesser power as it allows users of one system to have their tasks performed on other idle systems present on the LAN thus, turning the LAN into a super computer in itself.*

*We just hope that you find our efforts useful and this project is not an end but a new beginning to tell software vendors who keep dishing out with websites and web portals that equally and even more important are the local LAN s and there is a huge potential in LAN s which still has not been explored.*

# Introduction

## History

Information Retrieval (IR) is a very old human need; much older than the invention of the computer and its wide-spread use. The concept of retrieving relevant information from a large collection of documents was well known in the ancient world and in various human societies.

With the introduction of computers in our every-day lives, IR was greatly aided and it consequently became an unbreakable and successful part of computing science as well as a separate science, due to its strong mathematical roots and its many applications

The early developers of ARPANET and the Internet allowed themselves to dream about a day when computers around the world would be linked together to share resources in a peer-to-peer (Host to Host where each host is an independent entities or nodes present on network) fashion.

In December 1970, Network Control Program (NCP), the first “host-host” networking protocol and the precursor to TCP/IP was created. The host-host concept—which we now call peer-to-peer—was crucial in developing the Internet. Every computer on the network was an equal: Each computer could access the resources of any other computer on the network while making its own resources available. Communication among hosts was also equal. No computer was seen as a client or component of another and all computers shared more-or-less equal bandwidth access to one another.

Several events have conspired to change the Internet landscape from primarily peer-to-peer to the now more familiar client-server architecture. The Internet has gradually become more commercial, and corporations build firewalls around their information to control access. Millions of people “log on” to the Internet using desktop computers that cannot match the power of the servers that form the backbone of the Internet. And many popular Internet applications and services, including the World Wide Web and FTP, are based on client server architecture.

The real problems are at the server side of the equation. Despite economic downturns, the number of people on the Internet is still increasing. The promises of every computer connected to a broadband connection are not yet a reality, so many of the planned services are too big for the miniscule dialup speeds.

Despite the reduction in costs of computers, Internet servers are still very expensive. The reason is that as the Internet grows, the capacity of the server needs to increase. Because CPU capacity can only double every couple years, servers must use multiple CPUs or clusters of servers to have the capacity to serve hundreds of thousands of users. Because of all this hardware, servers are costly to run and require very expensive software. Because of the large number of users, great leaps of technology have been made in the past few years to cluster Web servers and application servers. Databases have entered the terabyte range as well.

*Therefore, again host to host networking applications have emerged like:*

## Napster

Napster's hybrid P2P network, consisting of a centralized server for performing search functionality, could be modeled as a single rendezvous peer and multiple simple peers, all using TCP as a network transport. The rendezvous peer provides simple peers with the capability to locate an MP3 file download the file from its host peer.

Napster doesn't provide a complete solution for bypassing firewalls, and it is capable of traversing only a single firewall. Napster provides no message-routing capabilities, meaning that simple peers on the network can't act as router peers to enable other peers to perform double firewall traversal.

## Gnutella

In the Gnutella network, each peer acts as a simple peer, a rendezvous peer, and a router peer, using TCP for message transport and HTTP for file transfer. Searches on the network are propagated by a peer to all its known peer neighbors, which then propagate the query to other peers.

Gnutella peers don't provide a full router peer capability, which means that, as with Napster, Gnutella peers are capable of traversing only a single firewall.

The latest movement in the field is that of Sun Microsystems JXTA set of protocols, which are also on the public domain. These protocols aim to help developers build P2P applications or include P2P functionality into their systems.

The project undertaken was thought-of with JXTA in mind and its purpose was to attempt to combine the two aforementioned technologies on a much more solid ground than before. The ***Magnum Peer to Peer System (MPPS)*** provides users with great features like:

- ***Interoperability*** — MPPS is designed to enable peers provisioning P2P services to locate and communicate with one another independent of network addressing and physical protocols.

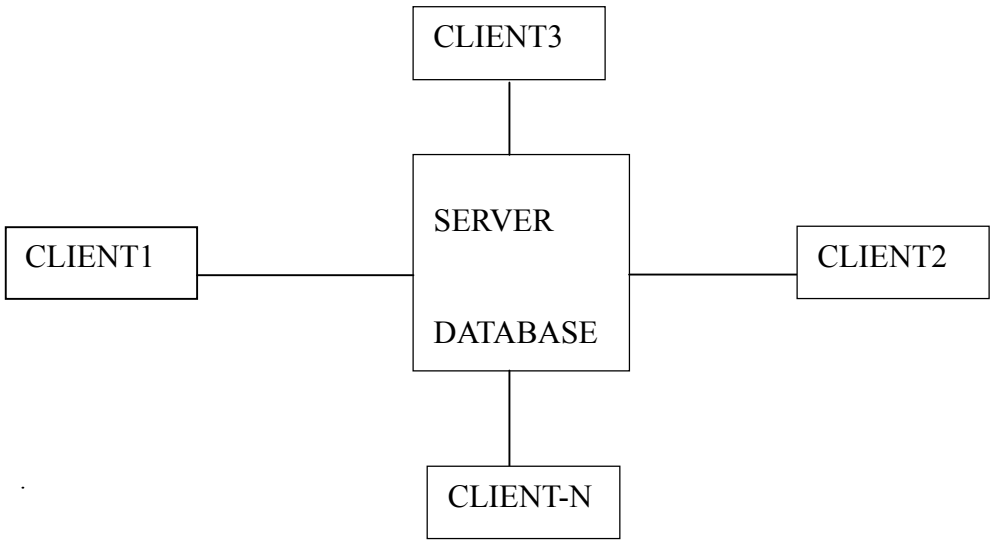
- ***Platform independence*** — MPPS is designed to be independent of programming languages, network transport protocols, and deployment platforms.

- ***Ubiquity*** — MPPS is designed to be accessible by any device with a digital heartbeat, not just PCs or a specific deployment platform.

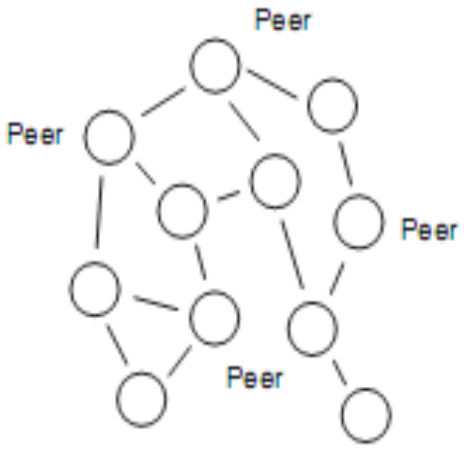
## Peer to peer (P2P):

- P2P is a class of applications that takes advantage of resources storage, cycles, content, human presence available at the edges of the Internet.
- Peer to Peer (P2P) computing is the sharing of resources between computers. Such resources include processing power, knowledge, disk storage and information from distributed databases.
- In the last several years, about half a billion Dollars have been invested in companies developing P2Psystems.

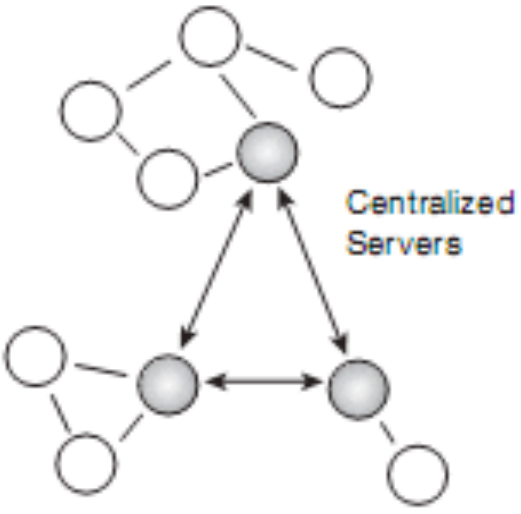
In a typical p2p architecture the network differs hugely from the client-server model one generally finds everywhere.




Client-Server Architecture




Hybrid Peer to Peer Topology




Centralized Peer to Peer Topology

## Scope of Magnum Peer to Peer System (MPPS)

- Managing resources of a single user, which would include maintaining indexes and records about entities like shared resources, download history etc.
- Dynamic computation of upload and download speeds for the resources being shared and retrieved to and from the network.
- Authentication of peer before allowing them access into peer groups.
- Monitoring and live updating of the status of all the peers present in the group
- Allowing users to decide the peers they don't want to have communication with.
- Sorting results of search queries as per user discretion like by size, number of active up loaders etc.

## Objectives

*The major goals of MPPS are*:

- To quickly locate other peers on the network with dynamic discovery across firewalls and NATs.
- To easily share resources with anyone across the network
- To create peer groups for sharing personalized services.
- To monitor peer activities and status remotely
- To provide secure communication between peers on the network

## Performance Characteristics

The MPPS has been built keeping in mind various network topologies being used across the internet. The most essential requirements for optimal usage of our system are:

- Network Connectivity
- 128 MB RAM
- Java Virtual Machine or Java Runtime Environment

Some of the key issues which must be considered before using MPPS to ensure no connectivity errors are:

## Bandwidth

User represents both client and server parts of an application, utilizing as much bandwidth going in as going out. But this is not how most ISPs are configured to work. For example, some cable modem and DSL ISPs have 1.5Mbps inbound and 128Kbps outbound.

It would be better for the performance if inbound speed equaled the outbound speed. Even if there were more wideband ISPs providing symmetric access, the infrastructure of the ISP may still only support asymmetric loads tilted toward inbound and not outbound traffic.

## Copyright infringes

Intellectual property is more of an issue in a P2P network because there is less built-in control. In a server environment, the authentication and authorization of users to access data is simple. The Data, authentication services, and rights management are centralized at the server. P2P Decentralizes content distribution and is thus seen as uncontrolled.

The idea that control is lost is not a certainty. In fact, the only reason that control is lost is because most developers do not implement rights management. Very often, it is easy enough to trust the User. There are several different ways to track documents and who uses them. Not all of these techniques are foolproof, but many have a reasonable cost to benefit ratio.

## Firewalls

Firewalls are used to protect corporate networks from unauthorized network connections, either incoming from the outside network or outgoing from the internal network. Typically firewalls use IP filtering to regulate which protocols may be used to connect from outside the firewall to the internal network or vice versa.

Because a firewall might block incoming connections, a peer outside the firewall will most likely not be capable of connecting directly to a peer inside the firewall. A peer within the network might also be restricted to using only certain protocols (such as HTTP) to connect to locations outside the firewall, further limiting the types of P2P communication possible.

# Introduction to Java

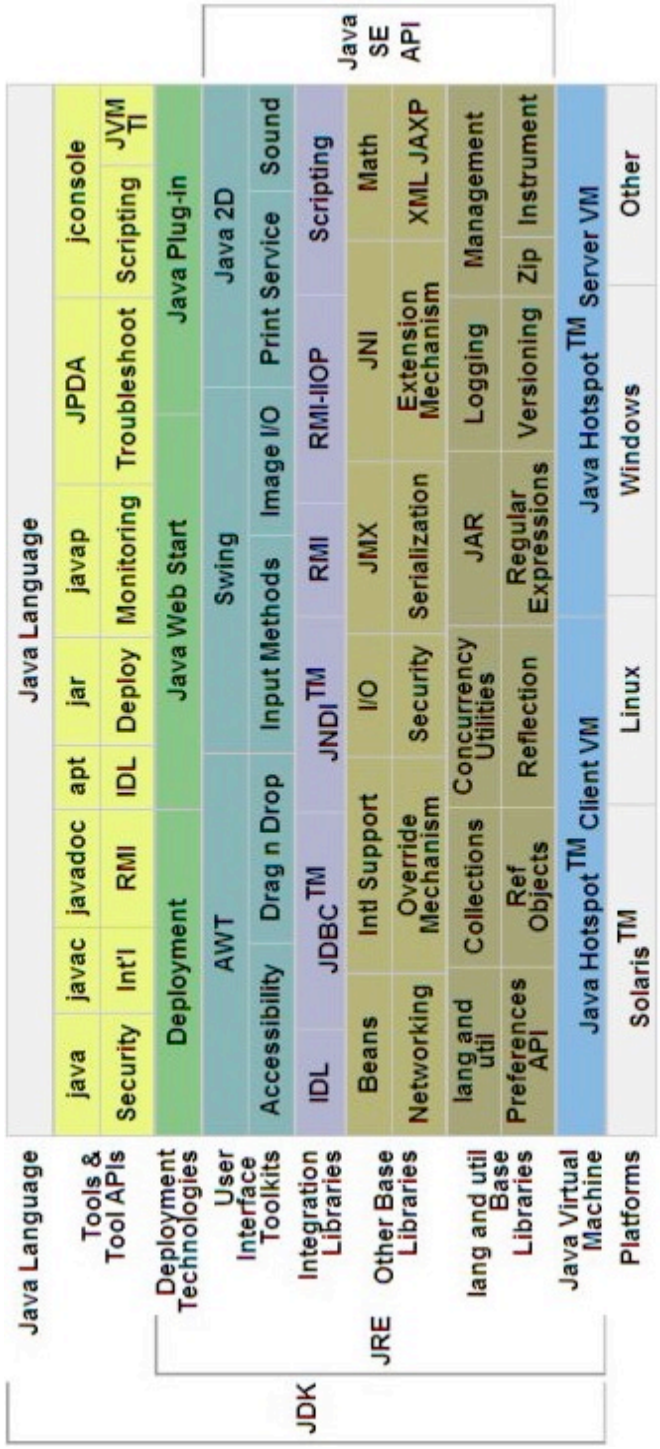
**Java** is a programming language originally developed by Sun Microsystems and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to byte code that can run on any Java virtual machine (JVM) regardless of computer architecture.

The original and reference implementation Java compilers, virtual machines, and class libraries were developed by Sun from 1995. As of May 2007, in compliance with the specifications of the Java Community Process, Sun made available most of their Java technologies as free software under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java and GNU Class path.

## Primary goals

There were five primary goals in the creation of the Java language

1. It should use the object-oriented programming methodology.
2. It should allow the same program to be executed on multiple operating systems.
3. It should contain built-in support for using computer networks.
4. It should be designed to execute code from remote sources securely.
5. It should be easy to use by selecting what were considered the good parts of other object-oriented languages.


Java Language
Java Language
java
javac
javadoc
apt
jar
javap
JPDA
jconsole
Security
Int'l
RMI
IDL
Deploy
Monitoring
Troubleshoot
Scripting
JVM TI
Tools & Tool APIs
Deployment
Java Web Start
Java Plug-in
Deployment Technologies
AWT
Swing
Java 2D
User Interface Toolkits
Accessibility
Drag n Drop
Input Methods
Image I/O
Print Service
Sound
Integration Libraries
IDL
JDBC™
JNDI™
RMI
RMI-IIOP
Scripting
Java SE API
Beans
Intl Support
I/O
JMX
JNI
Math
JDK
JRE
Other Base Libraries
Networking
Override Mechanism
Security
Serialization
Extension Mechanism
XML JAXP
lang and util Base Libraries
lang and util
Collections
Concurrency Utilities
JAR
Logging
Management
Preferences API
Ref Objects
Reflection
Regular Expressions
Versioning
Zip
Instrument
Java Virtual Machine
Java Hotspot™ Client VM
Java Hotspot™ Server VM
Platforms
Solaris™
Linux
Windows
Other

# Java Virtual Machine

A **Java Virtual Machine** (**JVM**) is a set of computer software programs and data structures which use a virtual machine model for the execution of other computer programs and scripts. The model used by a JVM accepts a form of computer intermediate language commonly referred to as Java byte code. This language conceptually represents the instruction set of a stack-oriented, capability architecture.

Java Virtual Machines operate on Java byte code, which is normally (but not necessarily) generated from Java source code; a JVM can also be used to implement programming languages other than Java. For example, Ada source code can be compiled to Java bytecode, which may then be executed by a JVM. JVMs can also be released by other companies besides Sun (the developer of Java) -- JVMs using the "Java" trademark may be developed by other companies as long as they adhere to the JVM specification published by Sun (and related contractual obligations).

The JVM is a crucial component of the Java Platform. Because JVMs are available for many hardware and software platforms, Java can be both middleware and a platform in its own right — hence the expression "write once, run anywhere." The use of the same bytecode for all platforms allows Java to be described as "compile once, run anywhere", as opposed to "write once, compile anywhere", which describes cross-platform compiled languages. The JVM also enables such unique features as Automated Exception Handling which provides 'root-cause' debugging information for every software error (exception) independent of the source code.

The JVM is distributed along with a set of standard class libraries which implement the Java API (Application Programming Interface). The virtual machine and API have to be consistent with each other[*dubious – discuss*] and are therefore bundled together as the *Java Runtime Environment*.

The **Java Development Kit** (**JDK**) is a Sun Microsystems product aimed at Java developers. Since the introduction of Java, it has been by far the most widely used Java SDK. On 17 November 2006, Sun announced that it would be released under the GNU General Public License (GPL), thus making it free software. This happened in large part on 8 May 2007[1] and the source code was contributed to the Open JDK.

## ***JDK contents***

The primary components of the JDK are a selection of programming tools, including:

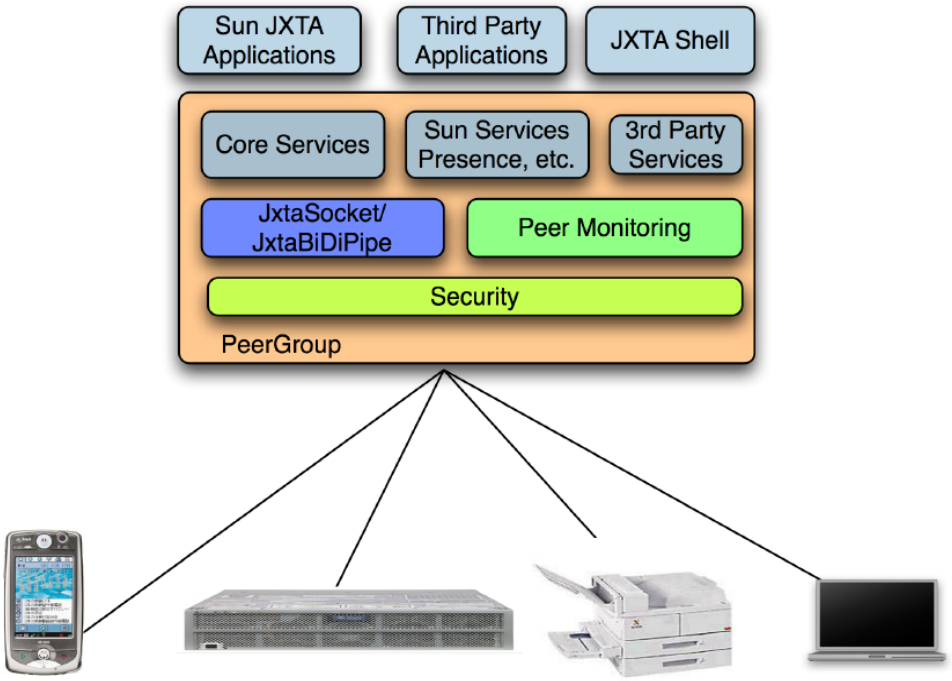
- java – The loader for Java applications. This tool is an interpreter and can interpret the class files generated by the javac compiler. Now a single launcher is used for both development and deployment. The old deployment launcher, jre, is no longer provided with Sun JDK.
- javac – The compiler, which converts source code into Java bytecode
- jar – The archiver, which packages related class libraries into a single JAR file. This tool also helps managing jar files (archived java class files)
- javadoc – The documentation generator, which automatically generates documentation from source code comments
- jdb – The debugger
- javap – The class file disassembler
- appletviewer – This tool can be used to run and debug Java applets without a web browser
- javah – This is the C header and stub generator and used to write native methods
- extcheck – This utility can detect Jar conflicts
- apt – The annotation processing tool
- jhat – (Experimental) Java heap analysis tool
- jstack – (Experimental) This utility prints Java stack traces of Java threads
- jstat – (Experimental) Java Virtual Machine statistics monitoring tool
- jstatd – (Experimental) jstat daemon
- jinfo – (Experimental) This utility gets configuration information from a running Java process or crash dump
- jmap – (Experimental) This utility outputs the memory map for Java and can print shared object memory maps or heap memory details of a given process or core dump
- idlj – The IDL-to-Java compiler. This utility generates Java bindings from a given IDL file.

The JDK also comes with a complete JRE usually called a *private* runtime. It consists of a Java Virtual Machine and all of the class libraries that will be present in the production environment, as well as additional libraries only useful to developers, such as the internationalization libraries and the IDL libraries.

# Introduction to Jxta

## *Overview*

The JXTA software architecture is divided into three layers,




**•JXTA Core**

The JXTA core encapsulates the minimal and essential primitives that are common to P2P networking. It includes building blocks to enable key mechanisms for P2P applications, including discovery, communication transports (including firewall and NAT traversal), the creation of peers and peer groups, and associated security primitives.

**•Services Layer**

The services layer includes network services that may not be absolutely necessary for a P2P network to operate, but are common or desirable in a P2P environment. Examples of network services include searching and indexing, directory, storage systems, file sharing, distributed file systems, resource aggregation and renting, protocol translation, authentication, and PKI (Public Key Infrastructure) services.

**•Applications Layer**

The applications layer includes implementation of integrated applications, such as P2P instant messaging, document and resource sharing, entertainment content management and delivery, P2P E-mail systems, distributed auction systems, and many others.

The boundary between services and applications is not rigid. One customer's application can be viewed as a service to another customer. The entire system is designed to be modular, allowing developers to pick and choose a collection of services and applications that suits their needs.

## What is JXTA™ Technology?

JXTA is an open network computing platform designed for peer-to-peer (P2P) computing by way of providing the basic building blocks and services required to enable anything anywhere application connectivity.
The name “JXTA” is not an acronym. It is short hand for *juxtapose*, as in side by side. It is a recognition that P2P is juxtaposed to client-server or Web-based computing, which is today’s traditional distributed computing model.
JXTA provides a common set of open protocols backed with open source reference implementations for developing peer-to-peer applications. The JXTA protocols standardize the manner in which peers:

•Discover each other

•Self-organize into peer groups

•Advertise and discover network resources

•Communicate with each other

•Monitor each other

The JXTA protocols are designed to be independent of programming languages and transport protocols alike. The protocols can be implemented in the Java programming language, C/C++, .NET, Ruby, and numerous other languages. Furthermore, they can be implemented on top of TCP/IP, HTTP, Bluetooth, and other network transports all the while maintaining global interoperability.

# Software Requirement Specification (SRS)

## 1. Introduction

### 1.1. **Purpose**:

Magnum Peer to Peer System (MPPS) is intended to help the user (individual/organization) in sharing resources and services across the internet in a decentralized manner.

This document is meant to describe in vivid detail the features of MPPS, so as to serve both as a guide to the developers as well as a software validation document for the prospective client.

### 1.2. Scope:

We describe what features are in the scope of the software and what are not in the scope of the software developed.

*In Scope:*

a) Managing and maintaining indexes and records about entities like shared resources download history etc.

b) Dynamic computations of upload and download speeds for the resources being shared and retrieved to and from the network.

c) Authentication of peer before allowing them access into peer groups.

d) Monitoring and live updating of the status of all the peers present in the group.

e) Allowing users to decide the peers they don't want to have communication with.

f) Sorting results of search queries as per user discretion like by size, number of active uploaders etc.

*Out of Scope:*

a) Authenticity of the resource cannot be ensured.

b) External anti-virus soft wares are required to check the files for viruses and other malicious codes.

c) High Communication speeds cannot be guaranteed, as they are dependent on network traffic.

## 1.3. **Definitions, Acronyms, and Abbreviations:**

*Acronyms and Abbreviations:*

a) MPPS : Magnum Peer to Peer System
b) SRS : Software Requirements Specification.
c) P2P : Peer-to-Peer.
d) JXTA : Juxtapose
e) GUI : Graphical User Interface.
f) KBps : Kilo Bytes per second.
g) Kbps : Kilo Bits per second
h) Mbps : Mega Bits per second

*Definitions*:

a) **Peer:** A participating entity in a P2P network. A peer can be a computer process, a computer or another electronic device able of sending and receiving digital signals, or a group of them. Peers offer both client and server functionality to the network.
b) **Seeder:** A peer making a resource available for other peers.
c) **Leecher:** A peer downloading a resource from other peers.
d) **Peer group:** A conceptual group of peers in the network that provide a common service. The actual difference between a peer and a peer group, theoretically, comes down to the intended interaction of the rest of the network with it.
e) **Discovery:** The process by which a peer in the network discovers other peers or resources of interest.
f) **Connections:** The data channels that are established between peers at run-time and by which information is exchanged.
g) **JXTA:** An open source peer-to-peer protocol specification begun by Sun Microsystems in 2001

### 1.3. Overview:

The rest of this SRS is organized as follows:

*Section 2*: It gives an overall description of the software. It tells what level of proficiency is expected of the user, some general constraints while making the software and some assumptions and dependencies that are assumed.
*Section 3:* It gives the specific requirements, which the software is expected to deliver. Functional requirements are given by various use cases. Some performance requirements and design constraints are also given.

*Section 4*: It gives some possible future extensions of the system.

## 2. Overall Description:

### 2.1 **Product Perspective:**

MPPS is aimed towards an individual or organization, which wants to share and avail resources with other users present on a network without having to maintain a centralized database repository. MPPS is a user-friendly, 'quick to
learn' and reliable software for the above purpose.

MPPS is intended to be a stand-alone product and does not depend on the availability of other software. It runs on both UNIX and Windows based platforms with ease.

### 2.2 **Product Functions:**

MPPS should support the following use cases:

| **Class of use case** | **Use cases** | **Description** |
|---|---|---|
| Use case related to Initial setup | Initial setup | Install and configure MPPS |
| Use case related to Search | File search | Search a particular file |
| | Peer search | Search a particular peer |
| | Group search | Search a particular group of peers |
| Use case related to Security | Blocked Users | Select users to be banned |
| | Allowed Users | Select user to be permitted for communication |
| | Data Security | Select folders you want to share |
| Use case related to Connection | Connection start time | Tells the starting time of session |
| | Connection end time | Tells the ending time of session |
| | Total Connection time | Tells the time duration of session |

| | | |
|---|---|---|
| Use case related to Storage | File Name | What is the name of the file |
| | Owner of file | Which user of the system owns that file |
| | Size of file | Indicates the size of the file on disk |
| | Date & Time of creation | When the file was created |
| Use case related to User Interface | Search bar | You can search any pattern |
| | Download bar | To show your download progress |
| | Storage | Where you can store your files |
| | Menus | For a GUI(Graphical User Interface) |
| Use case related to Data Transfer | Leachers | Available peers who have available files |
| | Speed | Data transferred per unit time (KBps) |
| | Time elapsed | Total Time since the transfer started. |
| | Time left | Estimated time remaining for the transfer to complete. |
| | Progress | How much percent of file is downloaded |
| Use case related to Chat | Block User | Block the user with whom you don't want to chat |
| | Search Peer | Search a particular peer to chat |
| | Request Chat | Select the particular peer and start chat |
| Use case related to Download | Progress Bar | Indicates the progress of a selected file |
| | Preview Bar | You can preview an audio or video file |
| Use case related to Upload | Shared Folders | Specify which files which you want to share |
| | Upload Limit | You can specify maximum upload limit |

2.3 **User Characteristics:**

User should know how to operate on a PC (personal Computer)
User should have some knowledge about the internet to take advantage of advance features.

2.4 **Principal Actors:**

The two principal factors are "users "and "a network of systems or internet"

2.5 **General Constraints:**

1. For resource sharing MPPS requires a network connection.
2. MPPS is single user software.
3. Both communication ends must be MPPS clients.

2.6 **Assumptions and Dependencies:**

a) If a firewall is installed on the user system then the user allows MPPS connection permission (required only at the first run).
b) User is assured that the contents shared are their own responsibility i.e. contents are not corrupt and roprietary
c) Java Runtime Environment is running when the MPPS is run.

# 3. Specific Requirements:

3.1 **Functional Requirements:**

We describe the functional requirements by giving various use cases.
*Use case related to installation:*

***Use Case 1**: Installation*
*Primary Actor*: User
*Pre Condition*: Internet connection available.
*Main Scenario:*
1. User initiates MPPS installation program.
2. System asks the user for the home directory in which all the working Files will be created. User is also asked for the initial login and password.
3. User specifies the home directory and login/password.
4. System creates the working files in the specified home directory.

*Working files contain:*
a. Authorization information.
b. List of resources to be shared.
c. Security related files like blocked users etc.
d. Folders for temporary data.

*Alternate Scenario*:
(a). Java Runtime Environment not found.
(b). Installation aborted.

***Use Case 2**: Search Procedure*

*Primary Actor*: User
*Pre Condition*: MPPS is running
*Main Scenario*:

1. User enters the search criteria
2. MPPS will display the results according to search
3. According to user requirements user will select a particular file or pattern.

*Alternative Scenario:*

1. No results found
    - Check network connection.

2. No Appropriate result found
    - Change search criteria

***Use Case 3**: Security*
*Primary Actor*: Security Module
*Pre Condition*: MPPS is running
*Main Scenario*:

1. User can block other peers (deny access to them)
2. A peer or group of peer allowed access
3. Data security is done

***Use Case 4**: Connection*
*Primary Actor*: Internet or Network
*Pre Condition:* MPPS is running
*Main Scenario*:

1. When users have selected a file to download then a connection will be established between user and available peers
2. Depending on available leachers and internet speed that will show about how much time is left for downloading.

*Alternative Scenario:*

1. Connection terminates
    a. Try to check the connection, if not found prompt user.
2. No Progress in Downloading
    a. Check your network connection
    b. It may happen if available peer has gone off line.

***Use Case 5**: Storage*
*Primary Actor*: Storage Module
*Pre Condition*: MPPS is running
*Main Scenario*:

1. User will specify files he wants to share
2. File information like name, size will be stored.

Alternative Scenario:

1. Not enough disk space
   a. Save file anywhere else where enough size is free for file.
   b. Manage disk space by deleting other files(that are not of use).

***Use Case 6**: Interface*
*Primary Actor*: User
*Pre Condition*: MPPS is running
*Main Scenario*:

1. User can add text to search box to search the file or peer or group
2. User can get a status of download from visuals
3. User has freedom to store downloaded file anywhere
4. Menus provide other accessibility option like view, preferences, option etc.

***Use Case 7**: Download*
*Primary Actor*: Download Module
*Pre Condition*: MPPS is running
*Main Scenario*:

1. How many leachers are available and what's the speed of downloading in calculated.

*Alternative Scenario:*
File already Exist

1. MPPS prompt the use if file is already present and going to overwrite
2. During download preview facility is also available

***Use Case 8**: Chat*
*Primary Actor*: User
*Pre Condition*: MPPS is running
*Main Scenario*:

1. Once user has logged in MPPS application. User will search a particular peer by name or he can search all peers by *(asterisk) operator. The result of search will be displayed in table.
2. User can select the particular peer he wants to chat, after selecting the peer just click on chat button.
3. A chat interface will be displayed to user where he can send and receive messages.

*Alternate scenario:*

1. Chat interface is not visible even after clicking on chat button
   - Check the status of selected peer. if it is offline you can't chat
2. Suddenly chat has been stopped

a. Check your Internet Connection
b. Check status of your friend.

***Use Case 9***:*Login*
*Primary Actor*: User
*Pre Condition*: MPPS is installed
*Main Scenario*: To use MPPS, user has to first give his nick name on the network.

*Alternate scenario:*
1. Login interface is not being displayed.
   a. Check your network connection
   b. Check Java Support i.e. JVM (Java Virtual Machine) on your system

***Use Case 10***:*File Upload*
*Primary Actor*: Upload Module
*Pre Condition*: MPPS is running
*Main Scenario*: The files that user wants to share with others will specify in his shared folder. Other peers who need that file will search into share folder of all the peers or a particular peer. Then that user will download that file.

*Alternate scenario:*
*File size is too large*
- *Compress the file by using compressors like (WinZip, Winrar) etc.*

### 3.2 **Performance Requirements:**

(a) Should run on 500 MHz, 64 MB machine.
(b) 90% of the responses should be within 2 sec,

### 3.3 **Development Requirements:**

| **Requirements** | **Purpose** |
|---|---|
| Java Development Kit (1.7.0) | To design MPPS application. |
| Java Runtime Environment (1.7.0) | To run MPPS application. |
| NetBeans (6.0) | To provide the IDE (Integrated Development Envronment). |
| JXTA Library | To provide support to P2P concept. |

### *3.4* ***Compatibility Requirements***

**HARDWARE REQUIREMENTS**

| **Requirement** | **Minimum** | **Recommended** |
|---|---|---|
| Processor | Intel® Pentium® llI | Intel® Pentium® lV |
| RAM | Windows 2000 Professional 128 MB<br>Windows XP Professional 192 MB | Windows 2000 Professional 256 MB<br>Windows XP Professional 256 MB |
| Hard Disk Space | 250 MB on Installation Drive | |
| LAN | 10/100 Mbps | 100 Mbps |

**SOFTWARE REQUIREMENTS**

- Java Run Time Environment
- Java Development Kit(JDK 1.4 or above)
- Windows NT or above

3.5 **Design Constraints:**

1. *Security:* The files in which the information regarding securities and portfolios Should be secured against malicious deformations.
2. *Fault Tolerance*: Data should not become corrupted in case of system crash or Power failure.

3.6 **External Interface Requirements:**

The user screen is split vertically into two panes. The left pane contains the Peer names, groups and file library. The right part displays the information related to resource sharing that is specified on the left pane. Appendix B shows the intended user screen.

### *3.7* ***Support and Training Requirements***

| Requirement | Purpose |
|---|---|
| Knowledge about Internet Explorer | Accessing Web Service. |
| Knowledge about computers | For Operating the Web Service. |
| Knowledge of English Language | For Operation. |

## 4. Future Extensions:

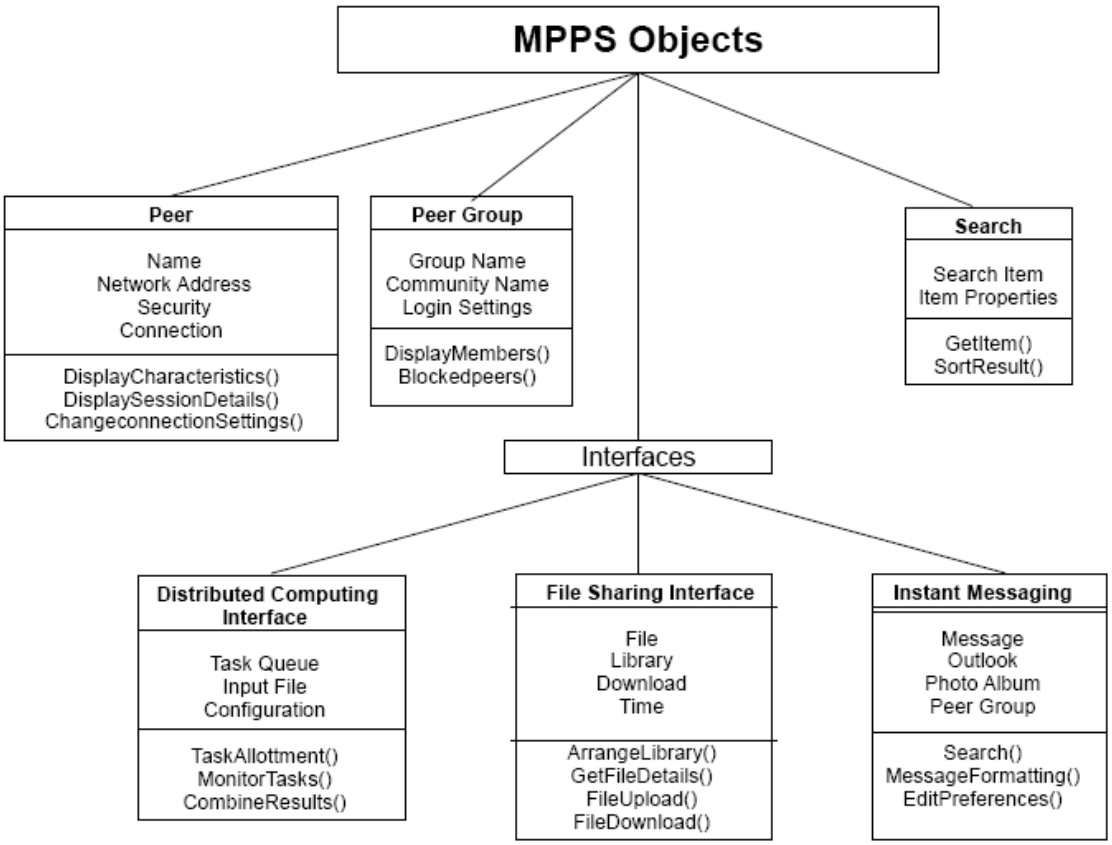
a). MPPS is intended to be a single user software. A possible future extension would be to make it multi user.

b). To add additional functionalities like resource preview and buffered play for multimedia files before actual download.

# System Design
# (Object-Oriented Approach)

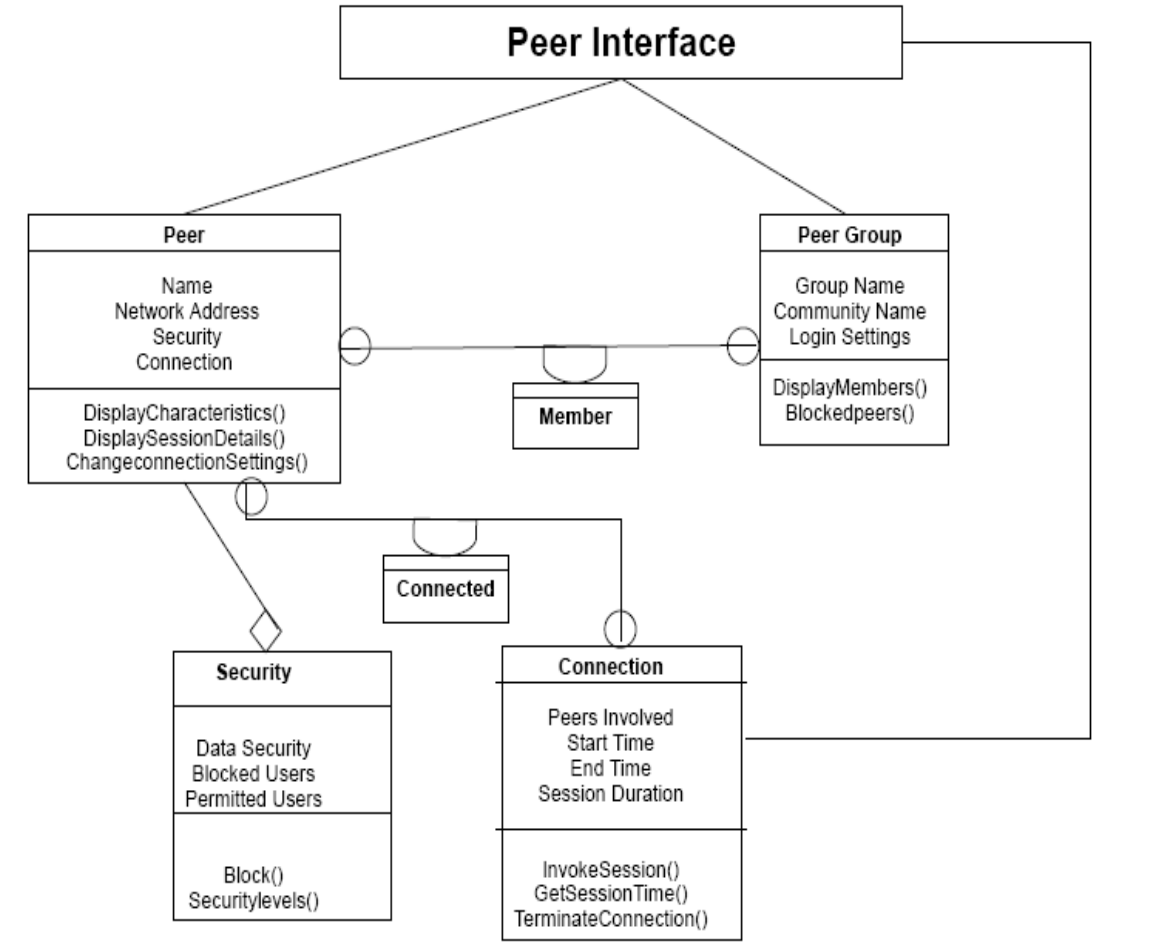
## 1. Object Diagrams

### 1.1 Level 1

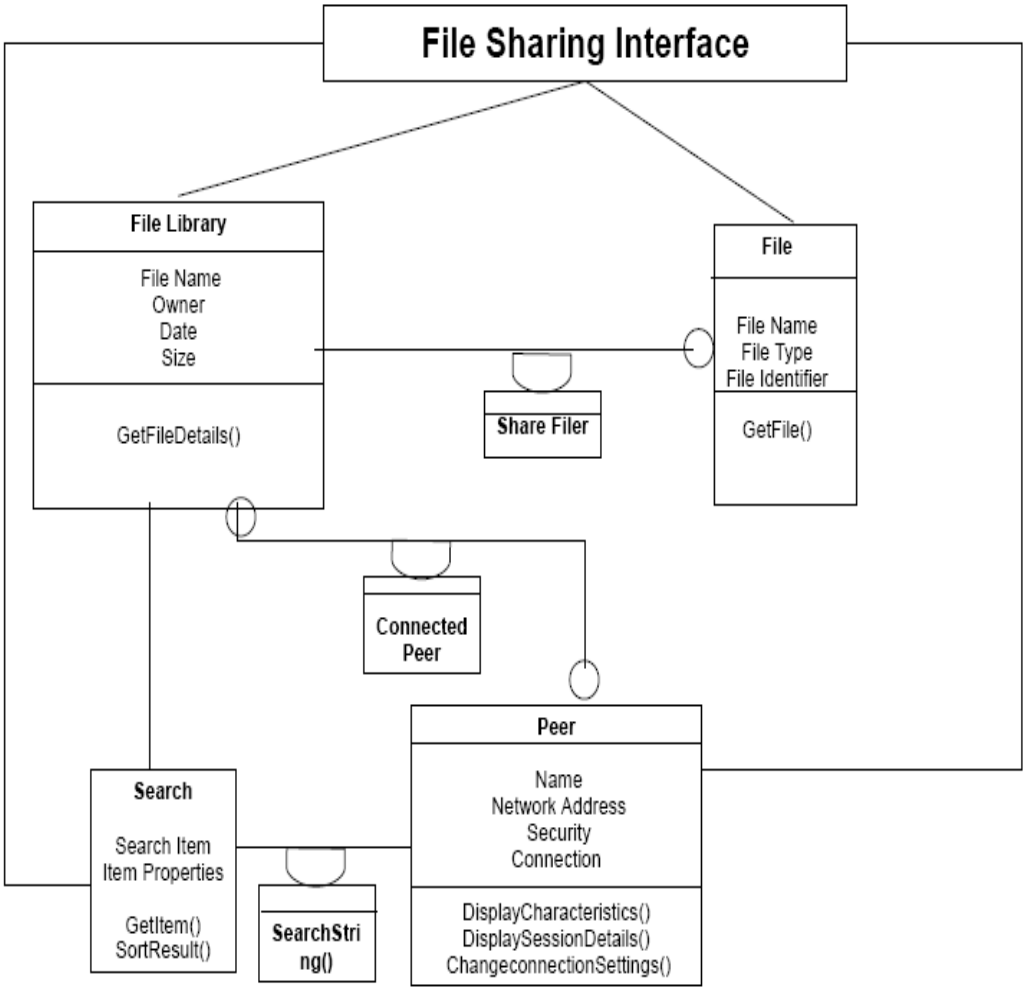



*Complete Magnum Object Model*

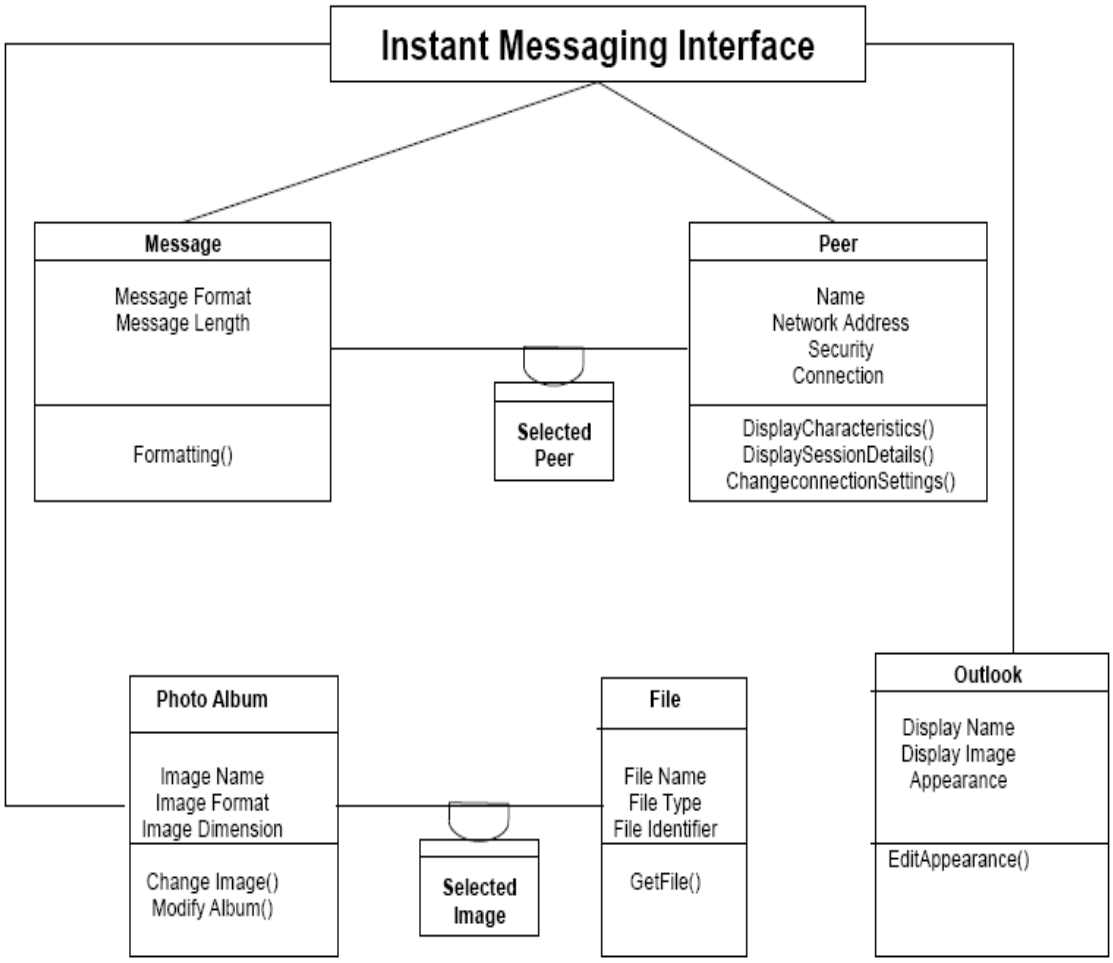
## 1.2 Level 2



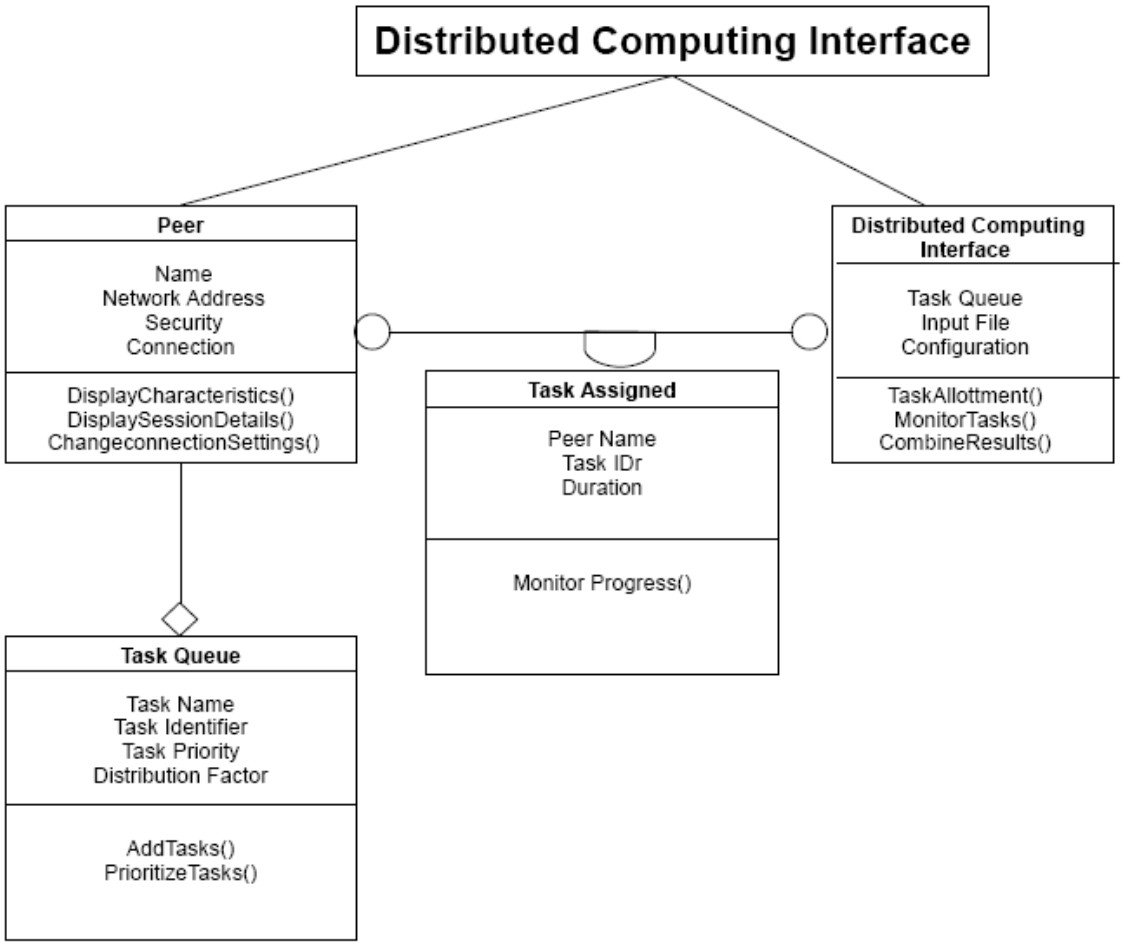

*(i).Object Diagram of Peer Interface*




*(ii). Object Diagram of File Sharing*

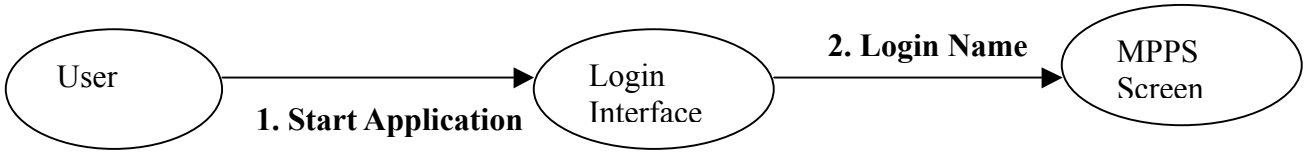



*(iii). Object Diagram of Instant Messaging*

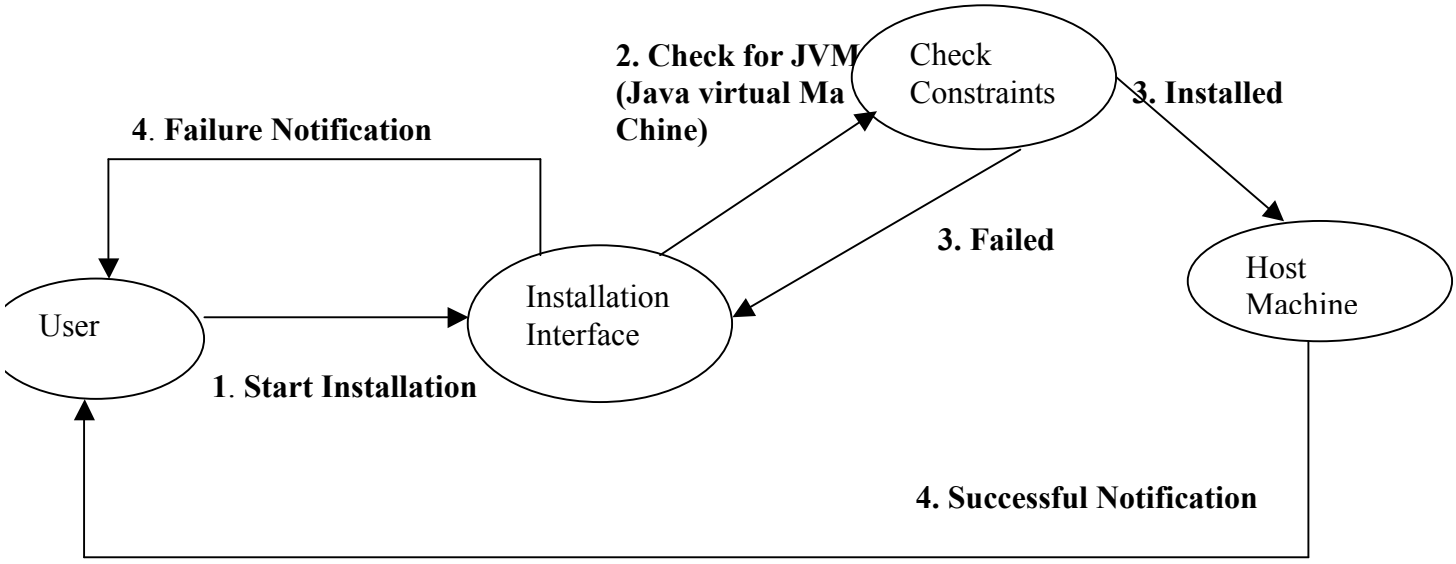



*(iv). Object Diagram of Instant Messaging*

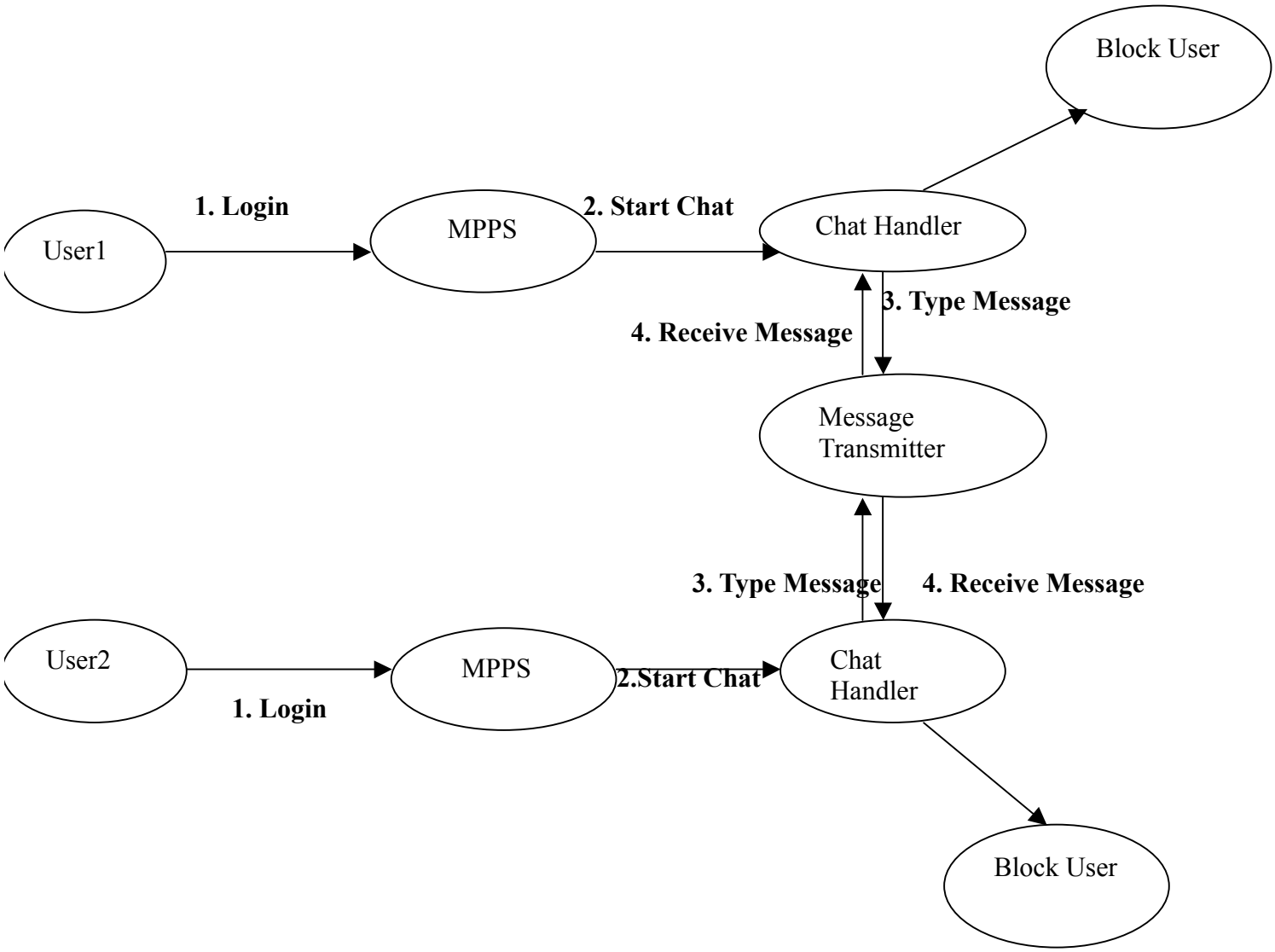
## 2.Analysis Phase



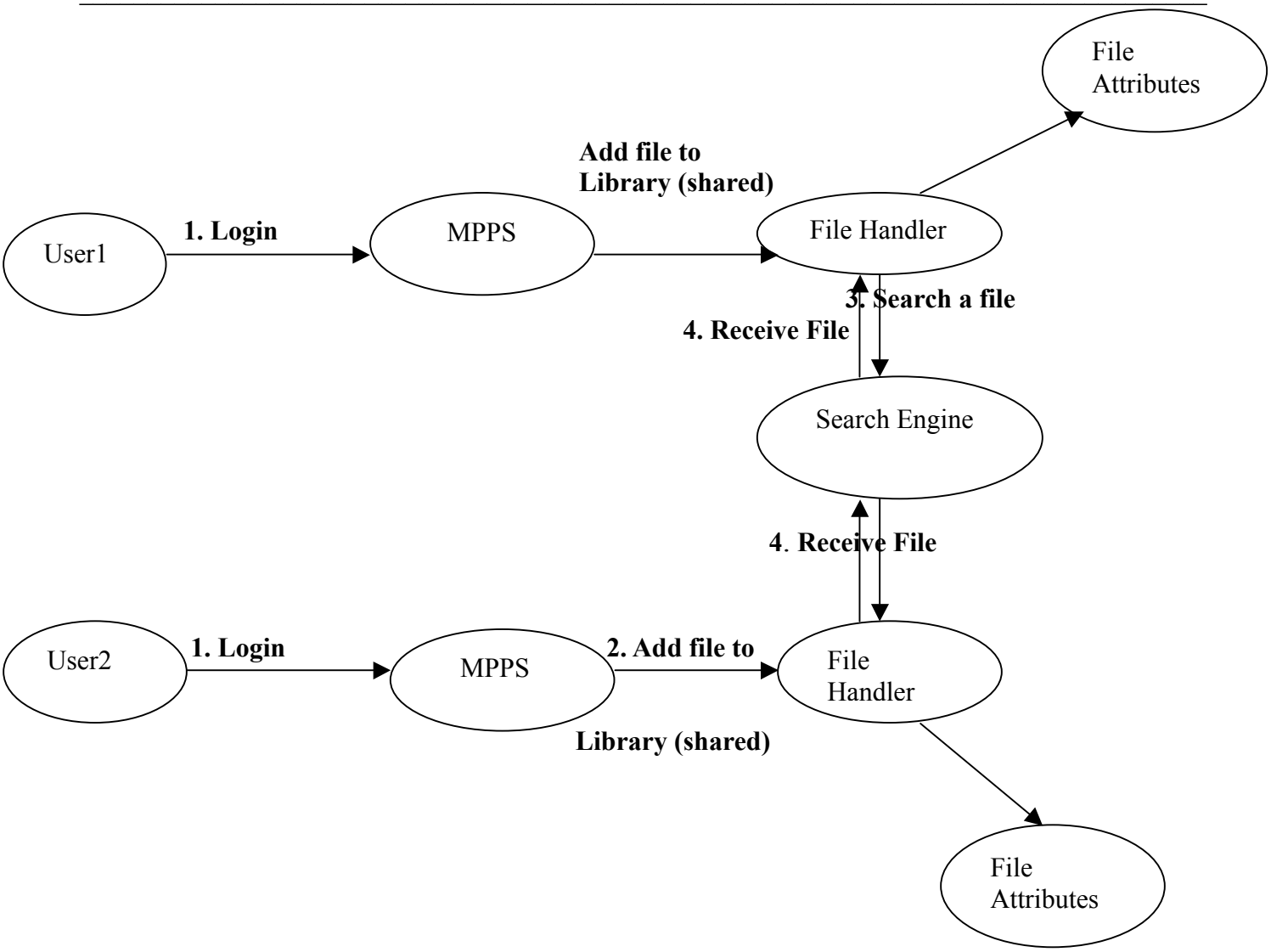

(i).A Collaboration Diagram for the Login use-case



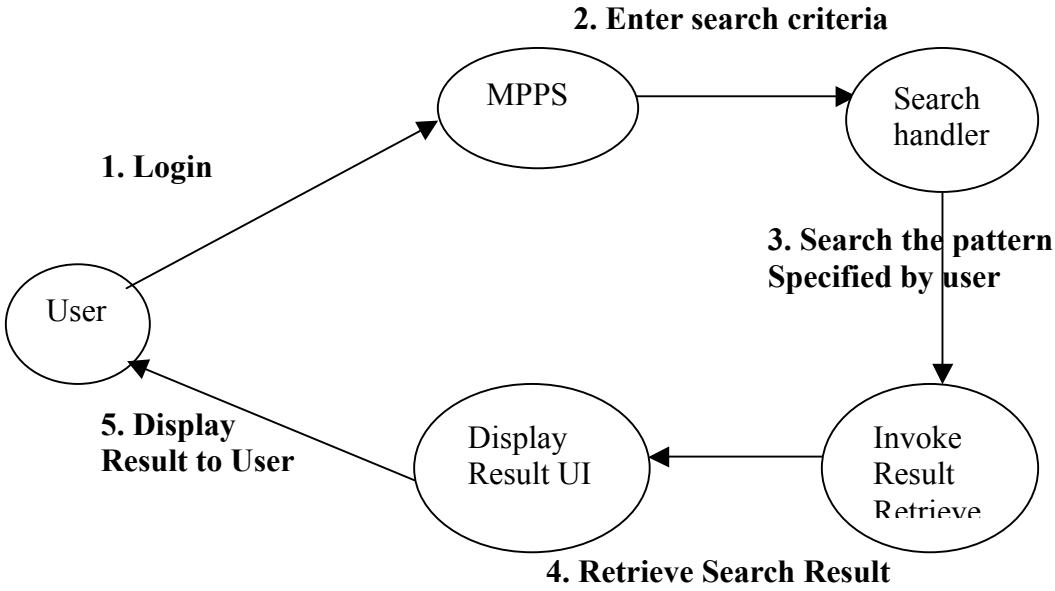

(ii).A Collaboration Diagram for the Initial Setup use-case



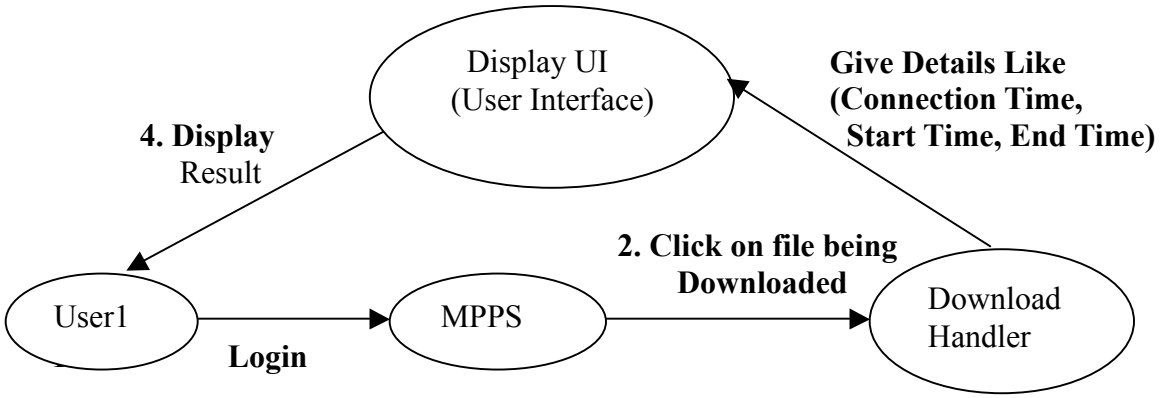

(iii). A Collaborative Diagram for the use case Chat




(iv). A Collaborative Diagram for the use case File Sharing

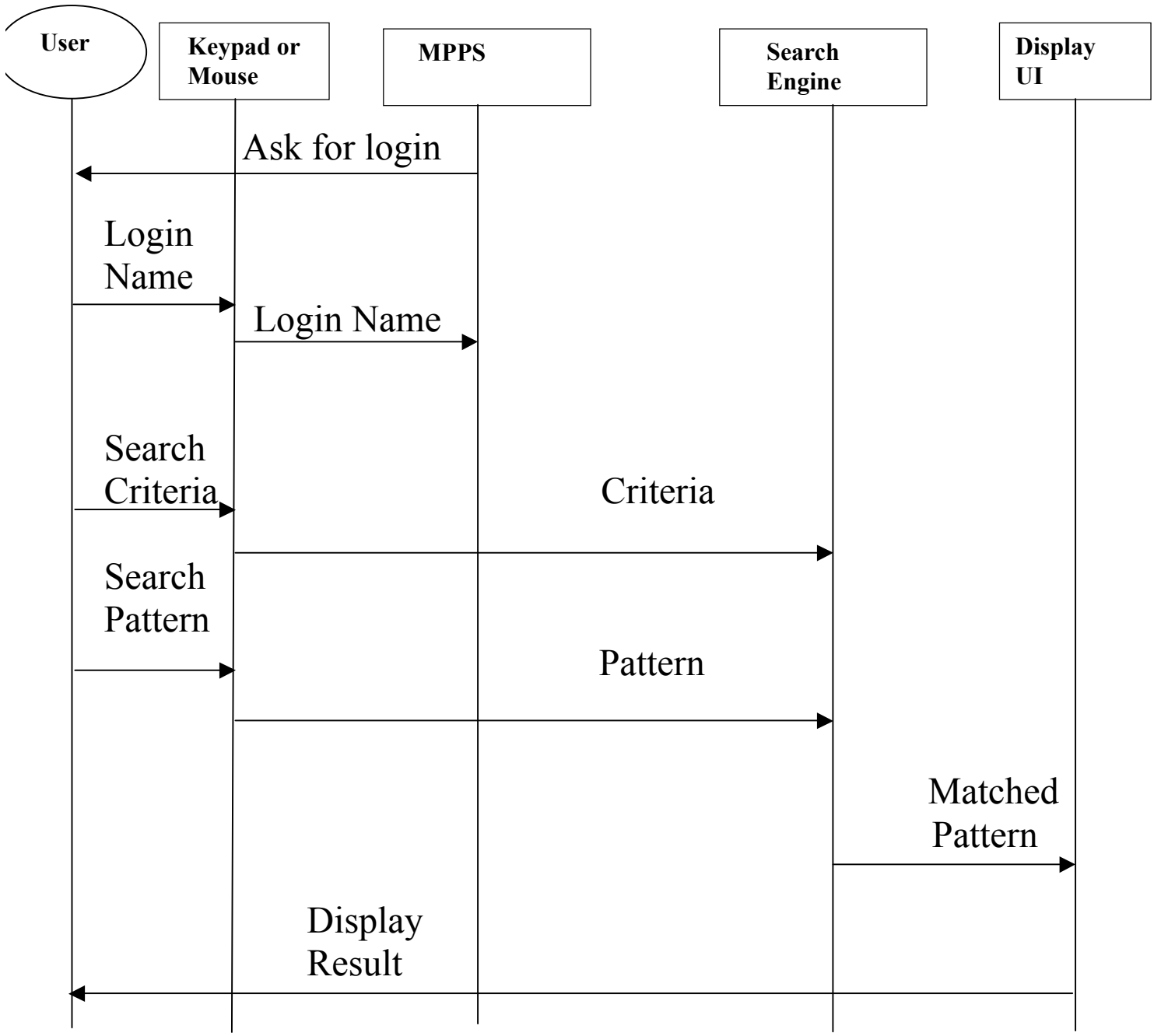



(v). A Collaborative diagram of Use case Search

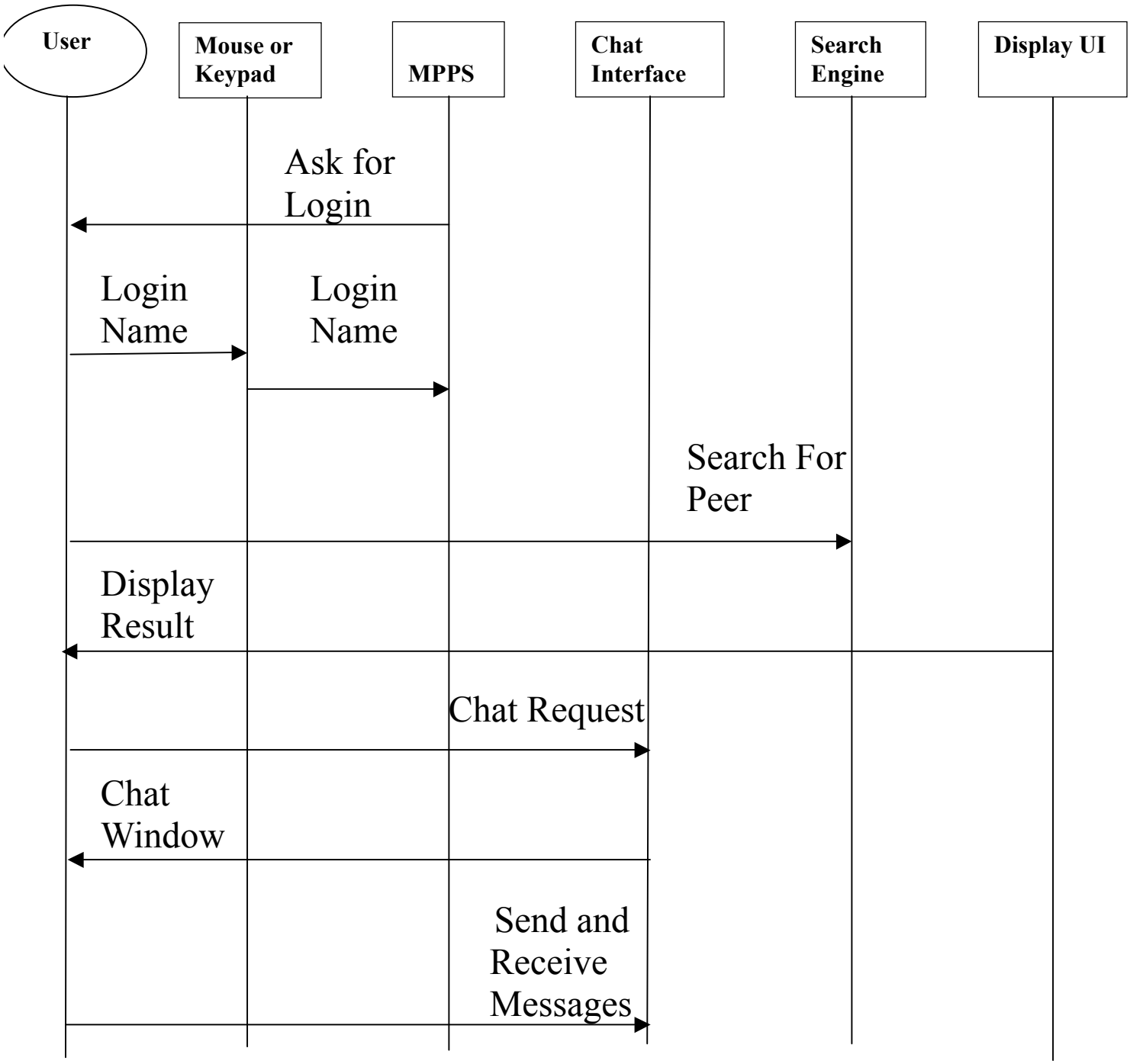



(vi). A Collaborative diagram of Use case Connection

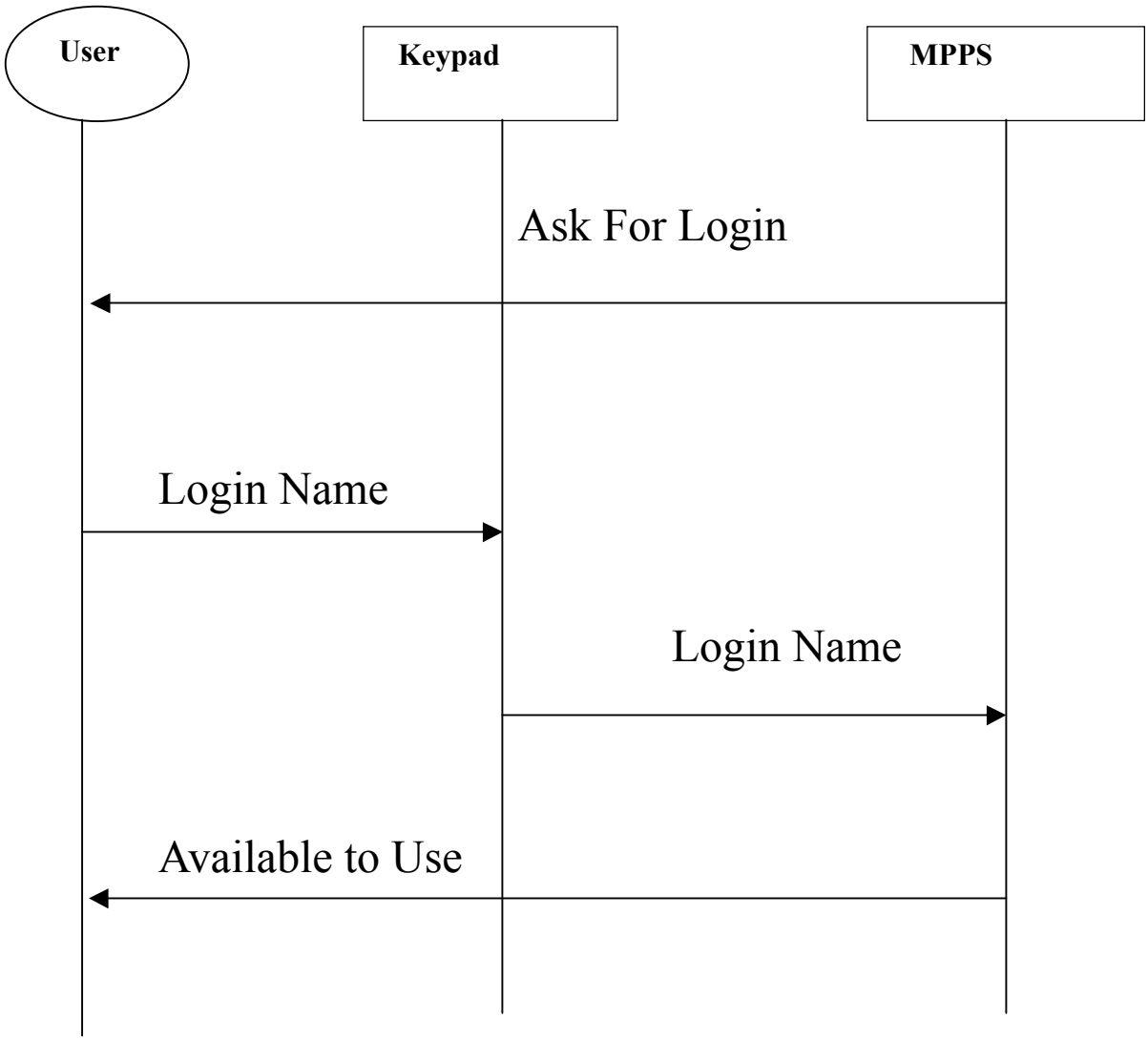
## 3. Design Phase



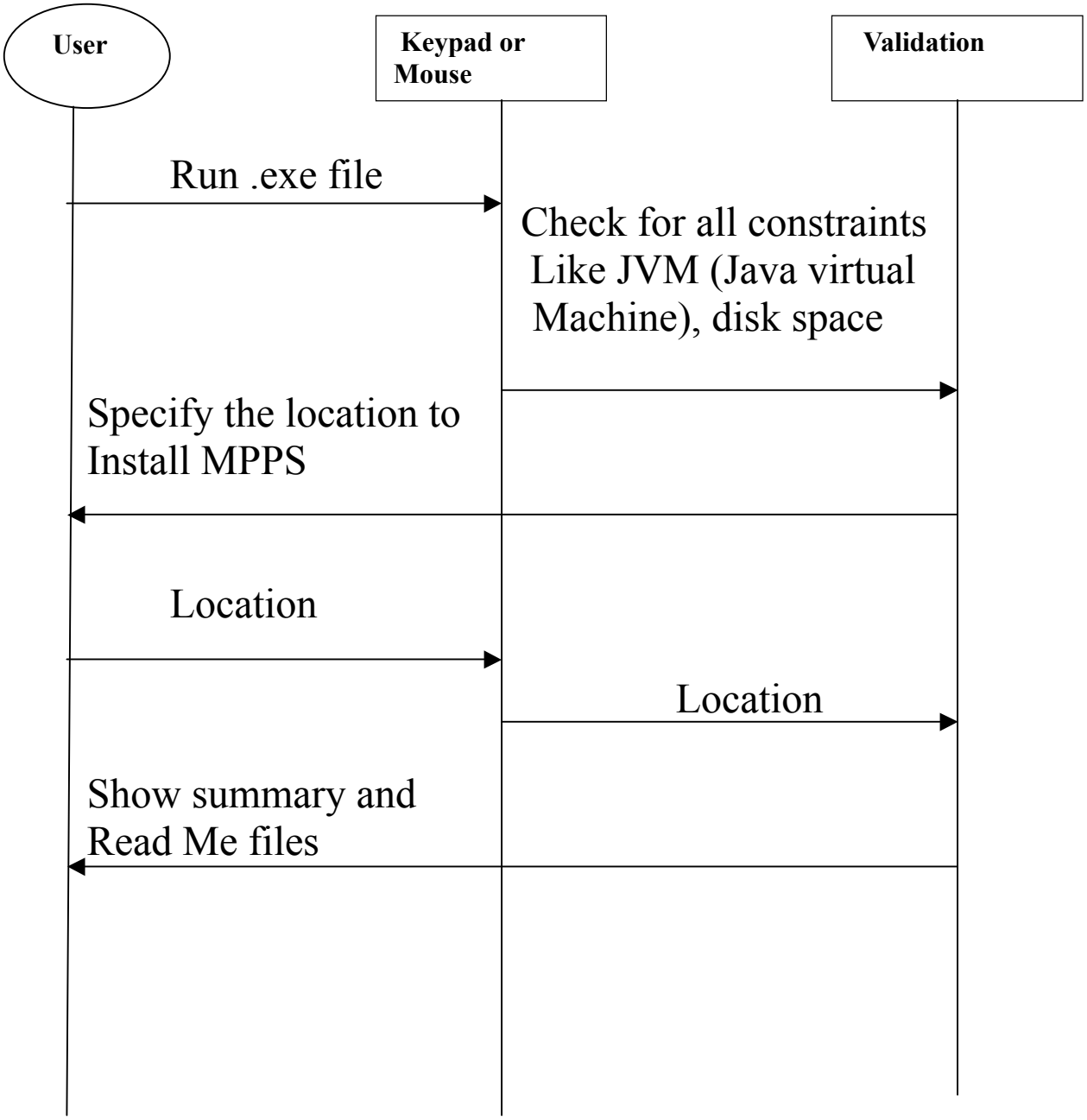

(i). A Sequence Diagram of Search Use case



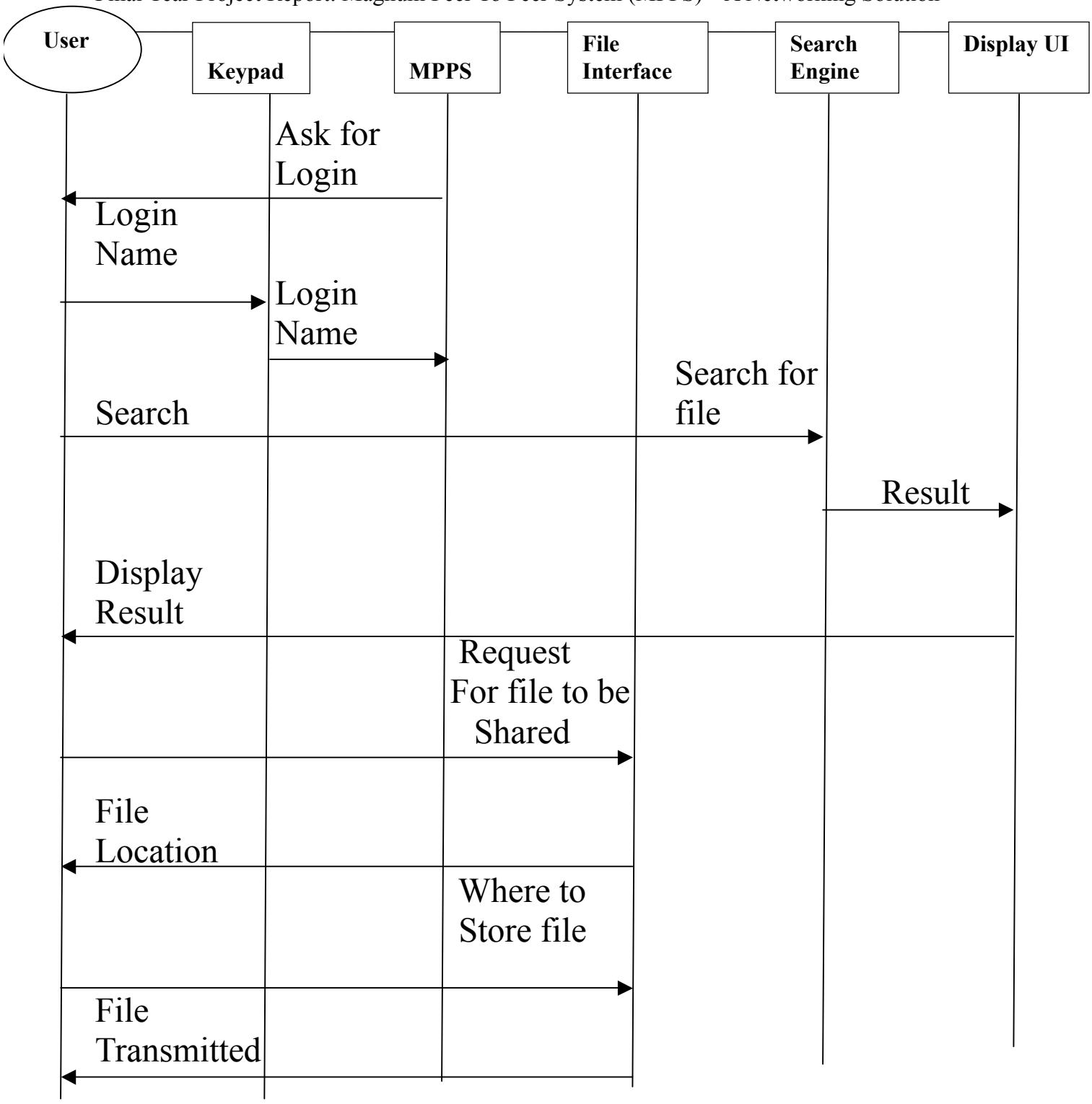

(ii). A Sequence Diagram of Chat use case




(iii). A Sequence Diagram of Login use case




(iv). A Sequence Diagram of Initial Setup use case




(v). A Sequence Diagram of File Sharing use case

# Testing

| Test Case ID | Purpose | Steps | Expected Result |
|---|---|---|---|
| 1 | Installation of MPPS | Click on .exe file to start setup | System should ask the user for the home directory in which all the Working files will be created. Along with that system should check for Java Run Time Environment |
| 2 | Login | Click on MPPS icon to start it. | MPPS will ask for a login name that name would be displayed to other peers on the network |
| 3 | Searching a peer | First peer should have logged onto MPPS system, after that select peer from dropdown box and type the peer name which you want to search | If Search criteria and search pattern both are valid then system should display the desired result to user in a table |
| 4 | Chat with a peer | To chat with a particular peer first you have to search that peer if status of that peer is online only then you can chat with that peer | When user will on chat button, system should check whether peer's status is online and peer is not blocked/if ok then a chat interface should be displayed to user |
| 5 | Uploading a file | To upload a particular file you have to save that file in shared folder, click on add button in library to add a particular file | When user will click on add file button MPPS should add that file to library after checking constraints like file size, whether file already exists or not? |
| 6 | Downloading a file | To download a file type file name in search box, and select file from pop-up menu. If file exists on the network result will be displayed in table. you can select a peer from table and then click on download button to start downloading | MPPS will search file and as user clicks on download button MPPS should check whether firewall is turned off or not. it should be turned off to start downloading/ |

# References

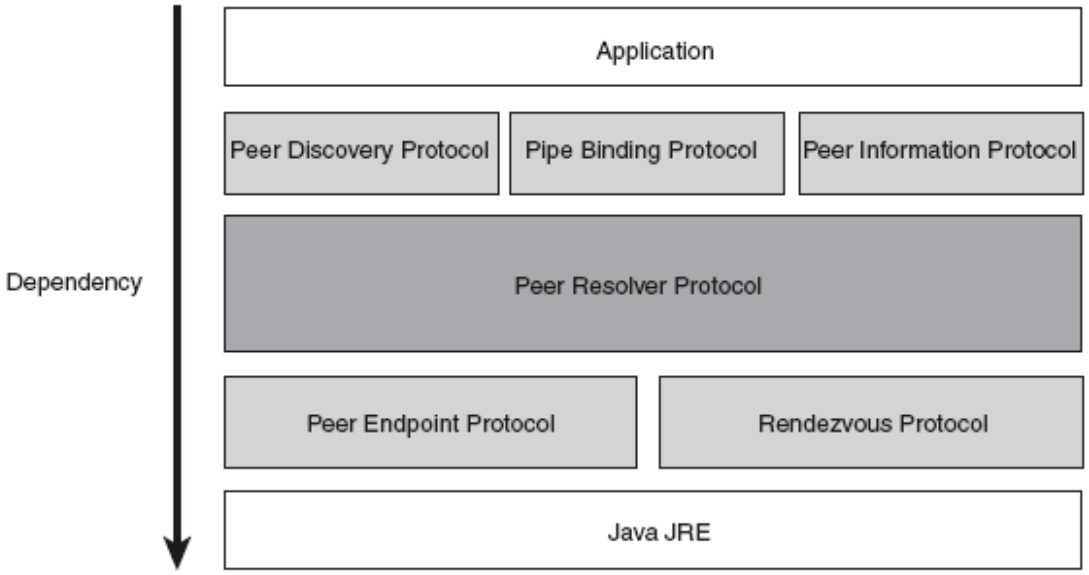

- http://jxta.dev.java.net — home Web page for Project JXTA
- http://jxta-spec.dev.java.net — Project JXTA specification
- Project JXTA: An Open, Innovative Collaboration, Sun Microsystems white paper.
- Project JXTA: A Technology Overview, Li Gong, Sun Microsystems white paper.
- JXSE Programmer's Guide v2.5
- Core Java 2 by Horstmann and Cornell

# Appendix:

## Appendix A: JXTA Specification Protocol Hierarchy:

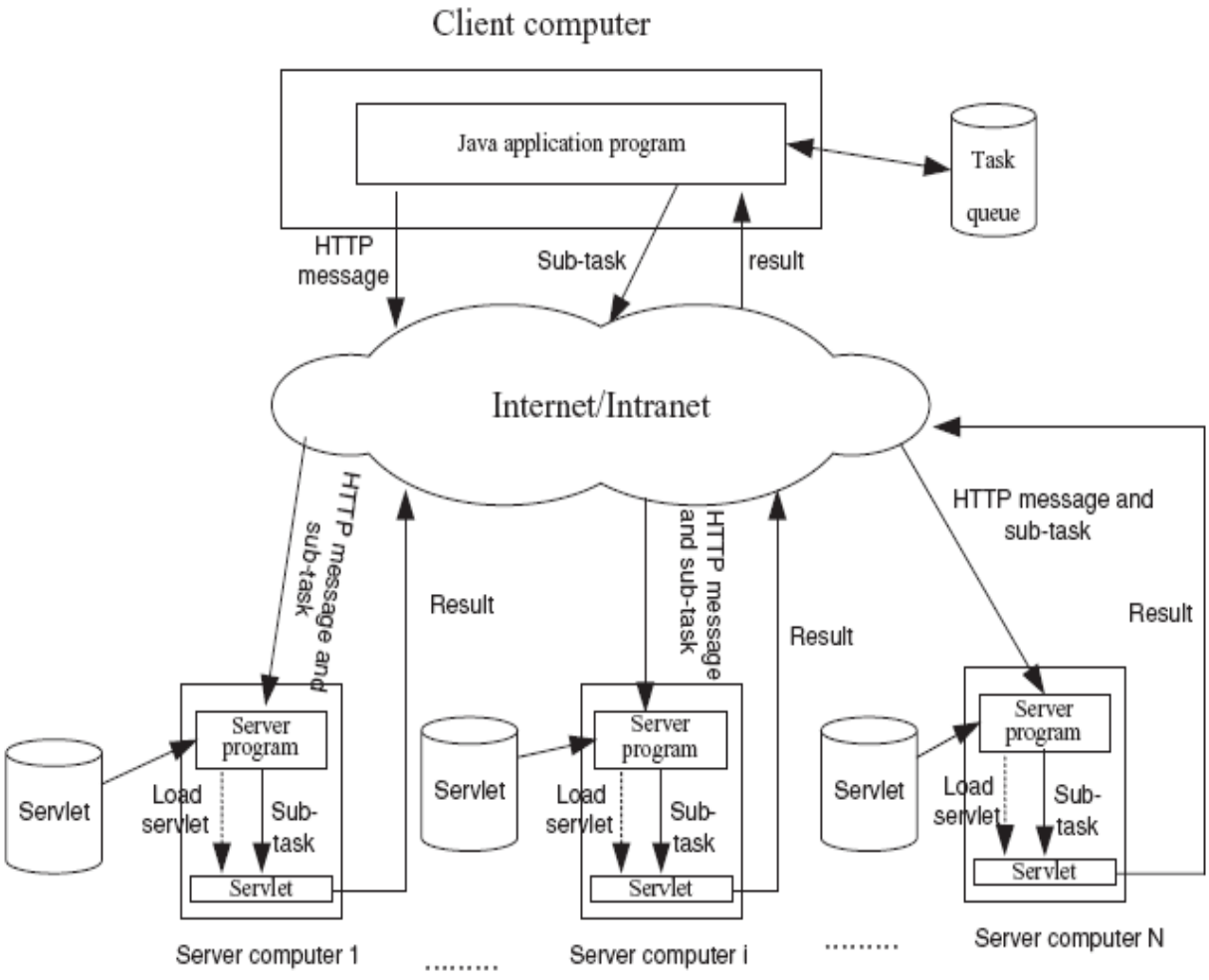
Currently, there are six protocols in the system:

1. **Peer Resolver Protocol (PRP)**—Used to send a query to any number of other peers and to receive a response.
2. **Peer Discovery Protocol (PDP)**—Used to advertise content and discover content.
3. **Peer Information Protocol (PIP)**—Used to obtain peer status information.
4. **Pipe Binding Protocol (PBP)**—Used to create a communication path between peers.
5. **Peer Endpoint Protocol (PEP)**—Used to find a route from one peer to another.
6. **Rendezvous Protocol (RVP)**—Used to propagate messages in the network.

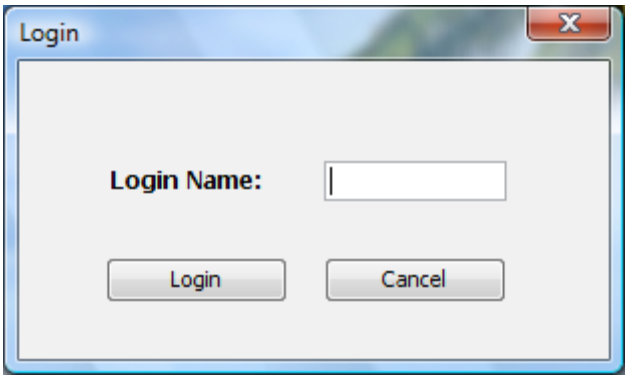


## Appendix B: Overview of Distributed Computing:

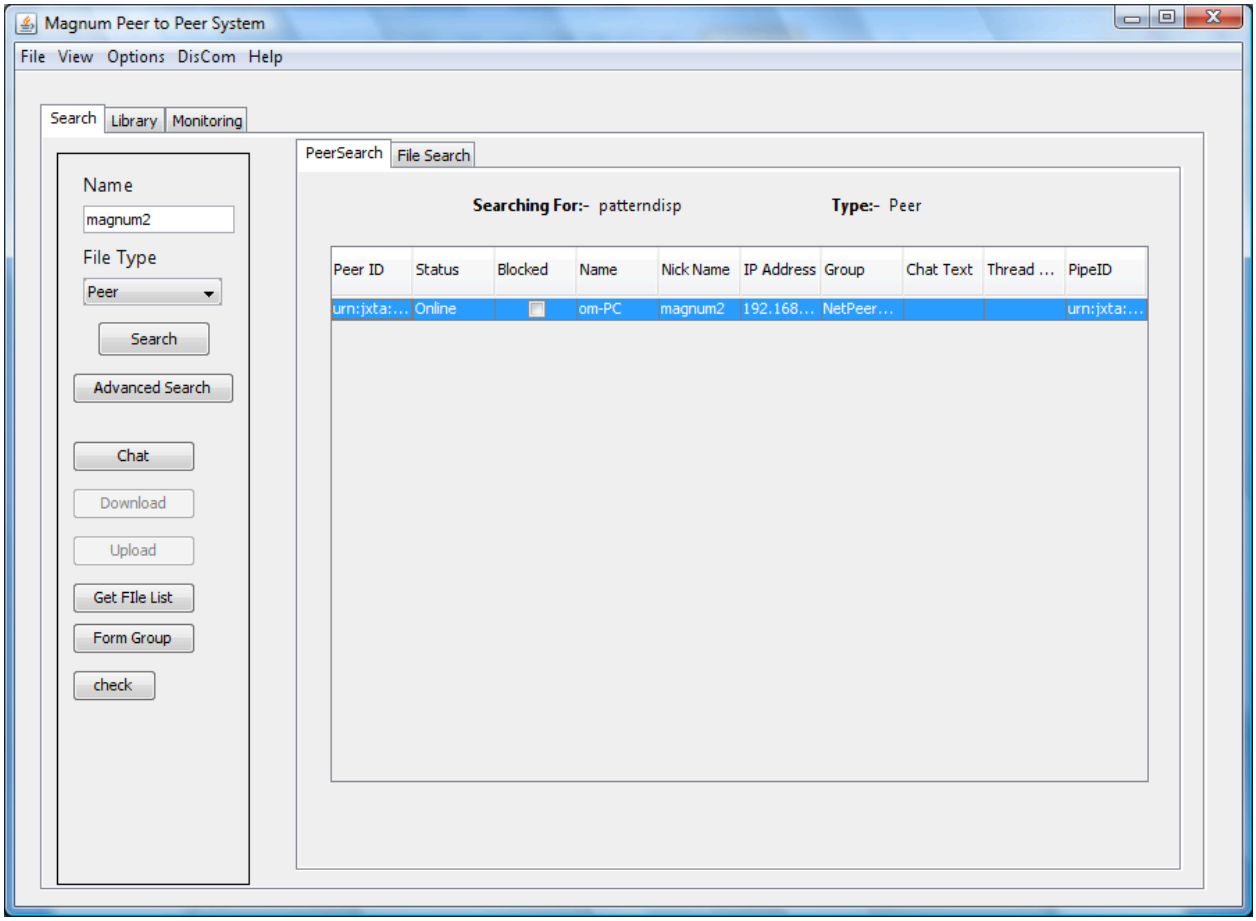
The diagrams below shows the outline of distributed computing environment through servlets.



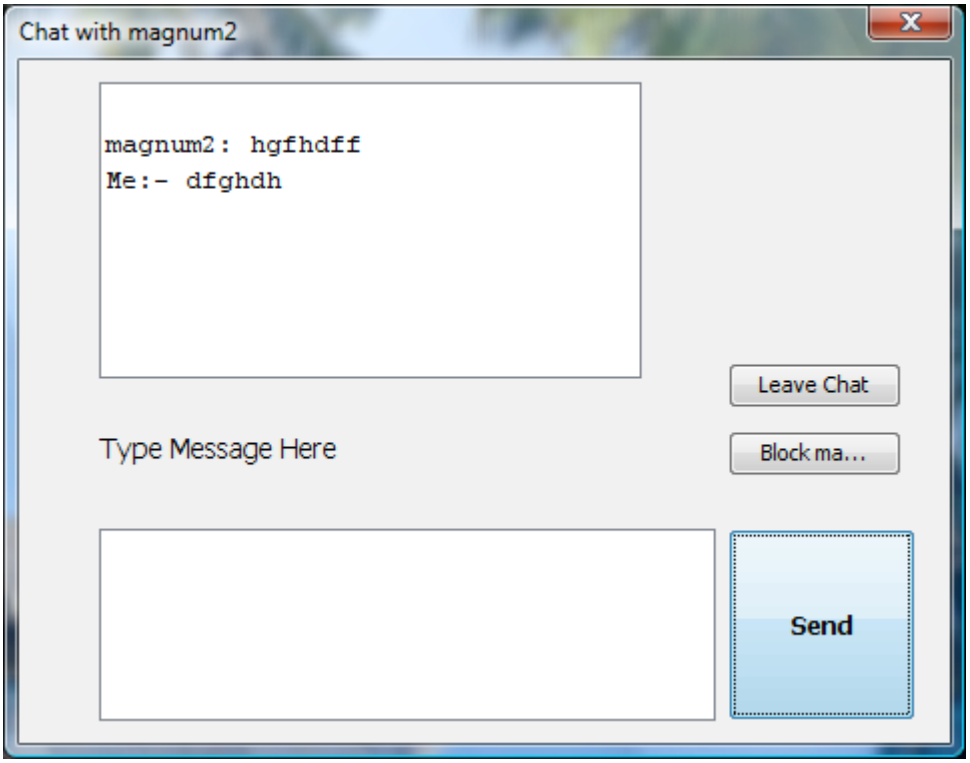
## Appendix C: User Screens:

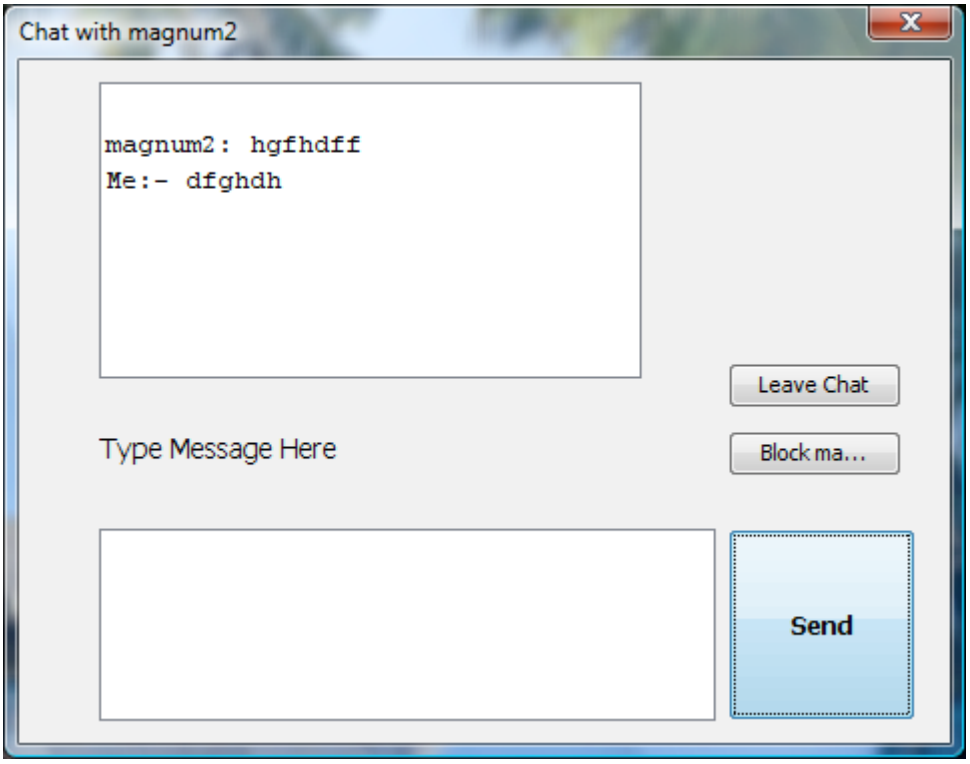



(i). Login Screen

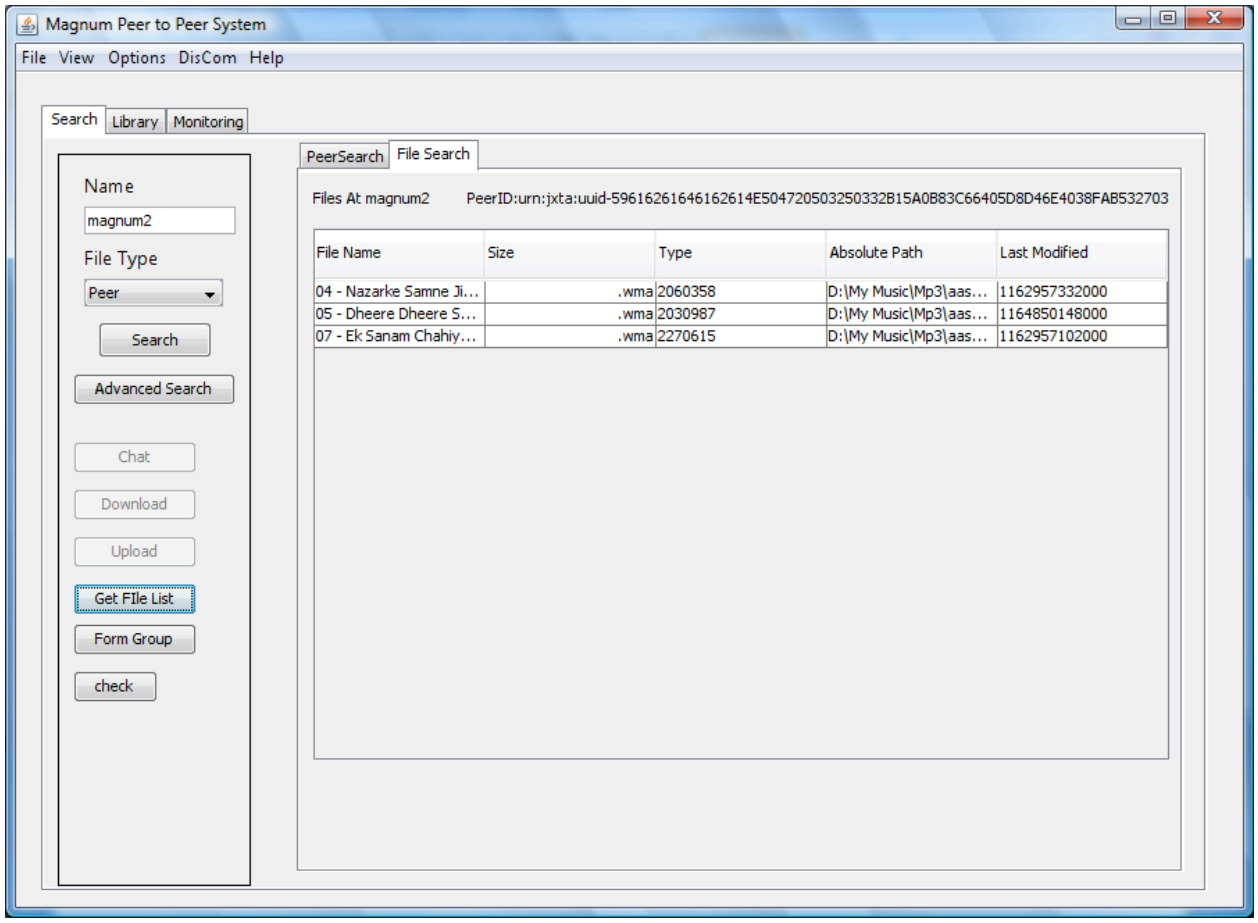



(ii). Peer Search Screen

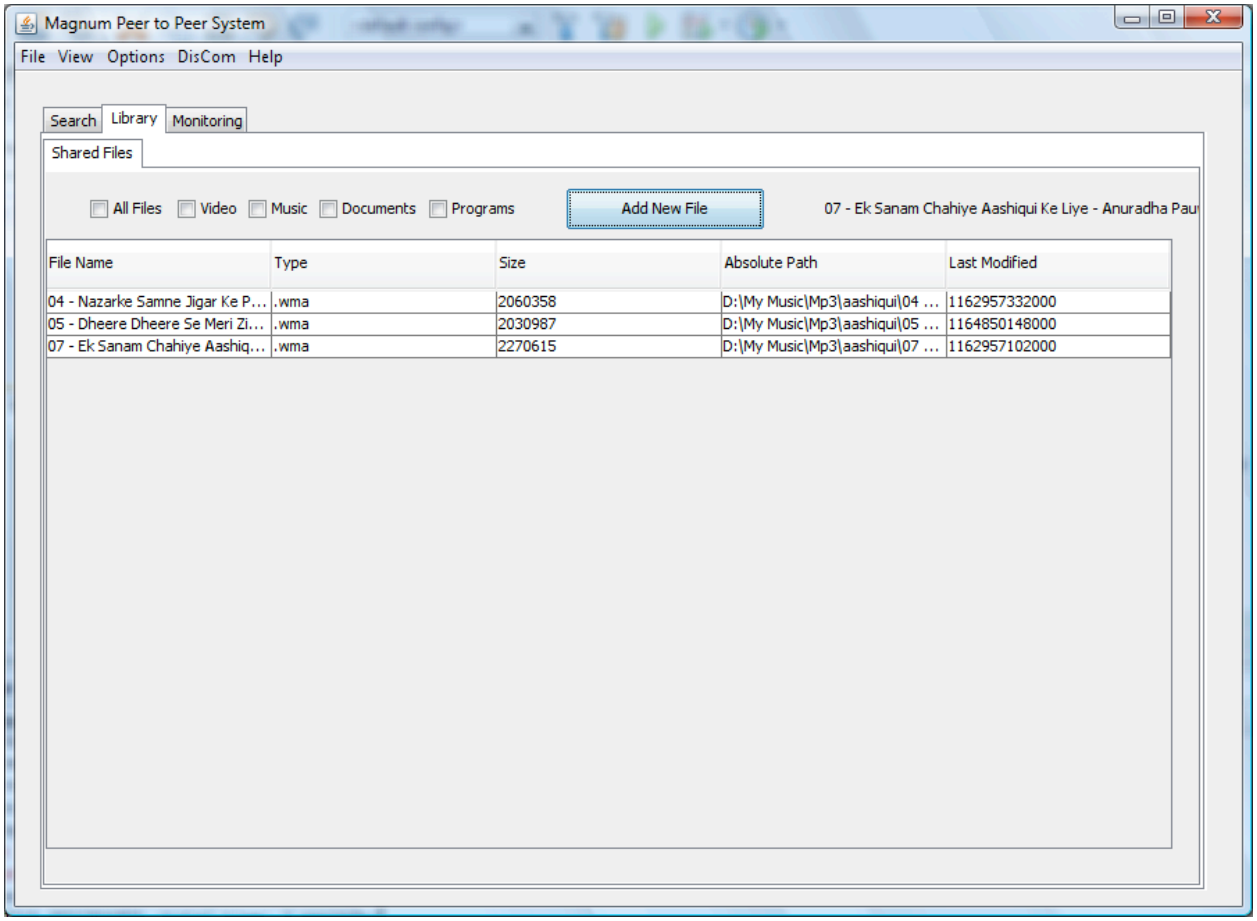



(iii). Chat Dialogs




(iv). File Search Screen




(v). Shared File Library Screen