**Documentation**

Version 1.0

26 July, 2011

**Multithreaded Chat Server**

<details redacted>

# 

INDEX

1. **Preface**
2. **Acknowledgement**
3. **SRS**

3.1 Introduction

3.2 Purpose

3.3 Scope

3.4 References

3.5 Document overview

3.6 Overall description

3.7 System environment

3.8 Functional requirements definitions

3.9 Use cases

3.10 Data description

3.11 Graphical user interface

3.12 Networking

3.13 Functional Model

3.14 Functional Requirements

3.15 System requirements

3.16 Technologies for Multithreaded Chat Server

**4.0 Data Dictionary**

**5.0 Analysis of design**

5.1 Flowchart

5.2 Usecase diagram for Server

5.3 Usecase diagram for Client

5.4 Activity diagram

**6.0 Screenshots**

**7.0 Software Testing**

**8.0 System Evolution & Conclusion**

* 1. **PREFACE**

The main purpose of this system is to stay connected to everyone & share your experience with everyone logged onto the system.It shares the interactive motive of a more efficient mechanism in education with optimizing the time equation, & comfort.

This application lets you stay in touch with everyone and you can actively send a message to anyone more efficiently, or can send a mass message, you can ask to a University official without approaching in person.

* 1. **ACKNOWLEDGEMENT**

This project took a lot of effort and work while developing it, but however, it would not have been possible without the support and guidance of our BCA faculty teachers.

I am highly grateful to our Coordinator, Mr. Gaurav Gaharwar and project teacher Kavita Nagpal, for their guidance while making this project.

**3.0 SOFTWARE REQUIREMENT SPECIFICATION**

**3.1 INTRODUCTION**

The user will start the program and is immediately presented with an option to select whether he is to be a host or a client. Once a connection is established, the server connection is initialized. The user is presented with a graphical representation of the user panel with the list of all users and will have the ability to send messages to one another by typing messages in a text box on the screen. The user can chat to multiple users since it is a multithreaded server.

## 3.2 SCOPE

The aim of this project is to develop multiple user chat platform on a single server that implements all of the standard rules of a basic multithreaded chat application, a graphical user interface, and a networking module capable of TCP/IP communication.

The Chat Server is designed to run on a local server and to allow its every client in the respective network to connect on a chat server running at one end & chat with each other based on the interactive teaching & educational purpose to stay connected to everyone at all the times. it will include to registering at the application with a unique username and a password, and basic chat function. The data will be held in an MS Sql server database.

## 3.3 References

[IEEE] The applicable IEEE standards are published in “IEEE Standards Collection,”

2001 edition.

## 3.4 Document overview

The remainder of this document is two chapters, the first providing a full description of the project for the users of this application. It lists all the functions performed by the system. The final chapter concerns details of each of the system functions and actions in full for the software developers’ assistance. These two sections are cross-referenced by topic; to increase understanding by both groups involved.

# 

# **3.5 Overall description**

This chat server encompasses a log of all the users of the application with a swing based layout and the basic chat utilities assigned while all the activities are controlled and monitored from administrator server system. This is a standalone application system where internet connection is necessary to access the system.

## 3.6 System environment

This application will be operated from a remote server where the client side systems will directly link to the main server with a static ip and all the activities will be monitored at server side.

## 3.7 Functional requirements definitions

The user's board will also be updated with moves sent from an opponent across the network.

Functional Requirements of this system is that they connect on a single main server and be in constant communication with each of the same clients which are connected at the same time and talk to each other with an interactive graphical based representation of the system.

The user's panel will also be updated with actions sent from other connected users one the server across the network.

## 

## 3.8 Use cases

Here the application is first launched on the administrator server with a static ip. This henceforth triggers the server launch and a main server is made.

This is a standalone application so it should be installed on every machine. The clients start the same application at client level and the client side system will connect to the main server with a static ip.

Similarly, from the server side the entire log will be kept of who is logged on and not. As per every client logged onto the system, lets you connect with everyone available on the server.

This is a GUI based system layered on Java Swing. As soon as a user hits on a available user’s display name, a window will be launched triggered from the method at client side which evoked the root function at server side system.

The administrator has rights to block a person from communicating to others. The chat histories will be optional.

The very goal of this system is to provide the most basic ground level chat operations/functions which it works it level best to provide the user.

**3.9 Data Description**

## This application will be implemented entirely with the Java programming language. This will provide a complete, cross-platform library for all GUI and networking needs, assuming a Java Virtual Machine is running on any machine using the program.

**3.10 Graphical User Interface**

The GUI Object will be responsible for showing the client side window, available user list panel, chat window, and user menus. This object handles user input and a “chat” window for communication between clients. This module will also be communicating with the user functional objects to access the networking module for any output or input needed.

**3.11 Networking**

The networking module will contain methods handle communications between two instances of the clients. It will have the ability to send and receive data by TCP/IP protocols. It will be multi-threaded to listen for incoming moves or chat messages.

**3.12 Functional Model**

**[Host ]**

Brings up a networking window with text boxes so the user can enter networking parameters (port number, alias).

**[Client ]**

Opens new window with IP parameters (Server IP address, Host port, Server alias) and Connect to Host button.

**[Connect to Host]**

Calls network module to connect to host to initiate new game.

**[Chat Text Box]**

User will enter text here and click Send Chat Message to send a message through the network module.

**[Send Chat Message]**

Sends a string entered in the Chat Text Box across the network.

**[update Status ]**

If a user updates his\her status from “Available” to “Away” then all the users are notified the updated status of the particular user

## 3.13 Functional Requirements

|  |  |
| --- | --- |
| **Use Case Name:** | Multithreaded Chat Server |
| **Priority** | Essential |
| **Trigger** | Main Server Launch |
| **Precondition** | It is connected to a LAN network and the application server is launched. |
| **Basic Path** | 1. The application at administrator end is launched and hence the server is made. 2. The clients on other hosts remotely connect and register their availability on the server and communicate to each other. |
| **Alternate Path** | N/A |
| **Postcondition** | The Server be first launched on the administrator side. |
| **Exception Path** | If there is a connection failure the chat server will crash & exit. |

## 3.14 System Requirements

-Intel Dual Core/Pentium 4

-512 Mb RAM

-Minimum 25 Mb space required.

-Java libraries be installed & Swing

**3.15 Technologies for Multithreaded Chat Server**

* **JAVA**
* **Inline HTML**
* **JAVA SWING**

**4.0 ANALYSIS OF DESIGN**

**[Server Module]**

1. **Name of the Class** : User

**Name of the Entities**: userName, hostName, isOnline, isConference.

1. **Name of the Class** :Stream

**Name of the Entities**: header,username,destination, message, host, user, userlist, data, \_filecontent, filename.

1. **Name of the Class** : Connect

**Name of the Entities**: connected, Con

1. **Name of the Class** : DbUser

**Name of the Entities**: Id, username, passwd, secQ, Answer, tickr.

1. **Name of the Class** : TProtocol

**Name of the Entities**: *null*

1. **Name of the Class** : Server

**Name of the Entities**: server, socket, []data, userSocketList, userObjList, hostname, handler (Handler class), allUserSockets.

**[Client Module]**

1. **Name of the Class** : User

**Name of the Entities**: userName, hostName, isOnline, isConference.

1. **Name of the Class** :Stream

**Name of the Entities**: header,username,destination, message, host, user, userlist, data, \_filecontent, filename.

1. **Name of the Class** : TProtocol

**Name of the Entities**: *null*

1. **Name of the Class** : LoginD

**Name of the Entities**: Id\_username, \_password, \_server, \_port, label1, label2, label3, label4, user, server, port, password, ok, cancel, container, SERVER\_HOST, SERVER\_PORT.

1. **Name of the Class** : UserPanel

**Name of the Entities**: \_userlist, usrtree, treeModel, rootNode, frame, nodeTable, panel, libName, user, dis, dos.

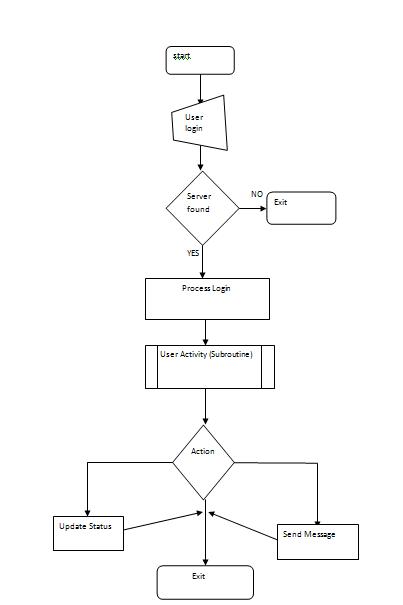
1. **Name of the Class** : ChatWindow

**Name of the Entities**: choose, file, Fsend, about, menubar, frame, thisframe, container, dArea, wArea, send, user, timer, isFocused, dis, dos.

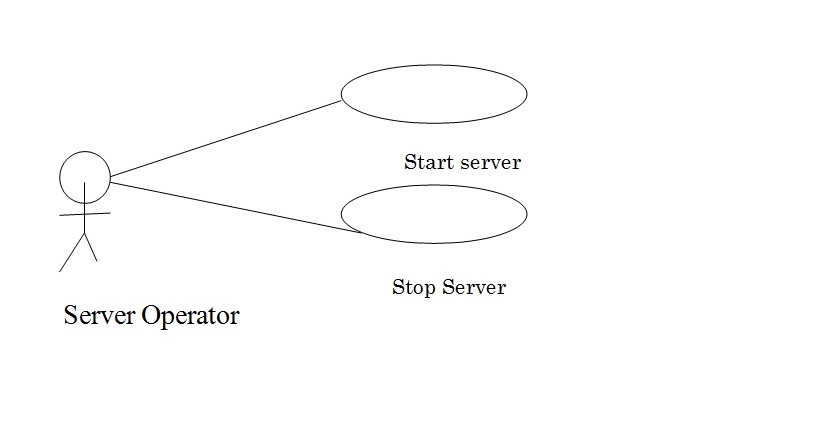
1. **Name of the Class** : Client

**Name of the Entities**: toolkit, screensize, label, menubar, menu, l\_item, e\_item, about, \_username, \_password, \_server, \_address, frame, container, combo, panel, thread, socket, connected, done, user, dis, dos, \_userlist, dialog, frameTable, pr.

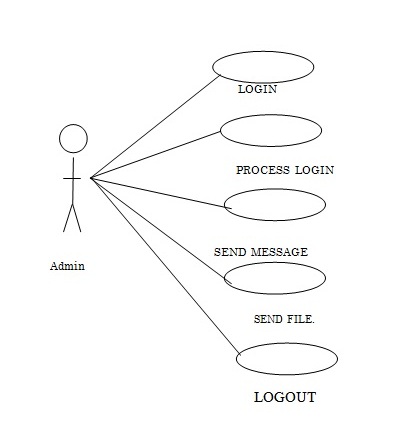
**4.1 Flowchart**

****

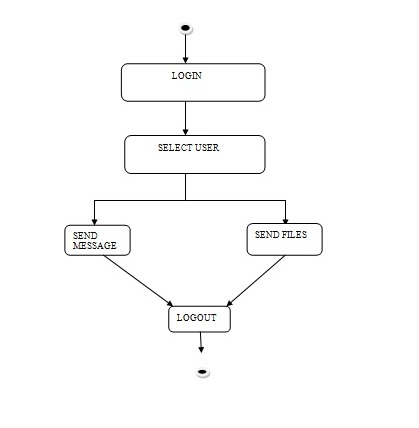
**4.2 Usecase Diagram for Server**

****

**4.3 Usecase Diagram for Client**

****

**4.4 Activity Diagram**

****

**5.0 DATA DICTIONARY**

1. User

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | CONSTRAINTS | DESCRIPTION | MAX. LENGTH | SAMPLE |
| userName | String | - | This field provides the name of the user | - | Max |
| hostname | String | - | This field provides the hostname of the user | - | localhost |
| isOnline | int | - | This field provides the online status of the user | - | 1 |
| isConference | Boolean | - | This field provides the conference status | - | true |

1. Stream

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | CONSTRAINTS | DESCRIPTION | MAX. LENGTH | SAMPLE |
| Header | int | - | This field provides the name of the user | - | Max |
| username | String | - | This field provides the name of the user | - | localhost |
| destination | String | - | This field provides the destination of the target user | - | 192.178.1.1 |
| message | String | - | This field provides the body of the message | - | “…” |
| Host | String | - | hostname | - | localhost |
| user | User | - | User object |  | User object |
| userlist | Vector | - | Stores user obj list |  | <user obj> |

1. TProtocol

– *methods for parsing an object to bytes and vice versa*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | CONSTRAINTS | DESCRIPTION | MAX. LENGTH | SAMPLE |

1. Server

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | CONSTRAINTS | DESCRIPTION | MAX. LENGTH | SAMPLE |
| server | Server | - | This field instanstiates the server object | - | Max |
| socket | Socket | - | This field makes the socket object | - | localhost |
| []data | Byte[] | - | This field provides the destination of the target user | - | 192.178.1.1 |
| userSocketList | Hashmap | - | This makes the list of all the user sockets,for the users logged on | - | Hashmap<Socket> |
| userObjList | Hashmap | - | Stores the user object | - | Hashmap<User> |
| Hostname | String | - | The user hostname |  | localhost |
| handler | Handler | - | Makes the Handler Object |  | <Handler obj> |
| allUserSockets | Vector |  | Makes a list of all user sockets |  | <vector> |

[Client Module]

1. User

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | CONSTRAINTS | DESCRIPTION | MAX. LENGTH | SAMPLE |
| userName | String | - | This field provides the name of the user | - | Max |
| hostname | String | - | This field provides the hostname of the user | - | localhost |
| isOnline | int | - | This field provides the online status of the user | - | 1 |
| isConference | Boolean | - | This field provides the conference status | - | true |

1. Stream

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | CONSTRAINTS | DESCRIPTION | MAX. LENGTH | SAMPLE |
| Header | int | - | This field provides the name of the user | - | Max |
| username | String | - | This field provides the name of the user | - | localhost |
| destination | String | - | This field provides the destination of the target user | - | 192.178.1.1 |
| message | String | - | This field provides the body of the message | - | “…” |
| Host | String | - | hostname | - | localhost |
| user | User | - | User object |  | User object |
| userlist | Vector | - | Stores user obj list |  | <user obj> |

1. TProtocol

– *methods for parsing an object to bytes and vice versa*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | CONSTRAINTS | DESCRIPTION | MAX. LENGTH | SAMPLE |

1. LoginD

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | CONSTRAINTS | DESCRIPTION | MAX. LENGTH | SAMPLE |
| Username | String | - | User name | - | Max |
| \_password | String | - | Password string | - | “…” |
| \_server | String | - | This field provides the server hostname | - | 192.178.1.1 |
| \_port | Int | - | portname | - | 1233 |

1. UserPanel

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | CONSTRAINTS | DESCRIPTION | MAX. LENGTH | SAMPLE |
| \_userlist | Vector | - | Stores user object | - | <User> |
| Usrtree | JTree | - | Generates the list panel of available users | - | - |
| treeModel | DefaultTreeModel | - | Instantiates the respective object | - | - |
| rootNode | DefaultMutableTreeNode | - | Makes the parent node | - | - |
| Frame | Client | - | Instantiates Client object |  | - |
| user | User | - | Makes user object | - | - |
| Panel | UserPanel | - | Makes UserPanel object | - | - |
| DataInputStream | Dis | - | Makes input stream | - | - |
| DataOutputStream | Dos | - | Makes output stream | - | - |

1. ChatWindow

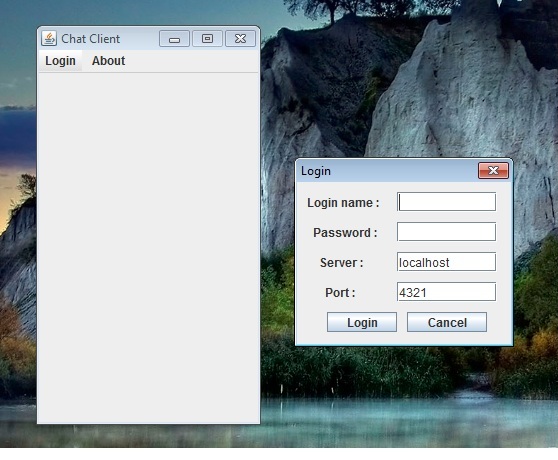
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | CONSTRAINTS | DESCRIPTION | MAX. LENGTH | SAMPLE |
| Choose | JFileChooser | - | To choose file | - | <User> |
| File | JMenuItem | - | File menu item | - | - |
| About | JMenuItem | - | Menuitem about | - | - |
| Menubar | JMenubar | - | Menubar | - | - |
| Frame | Client | - | Client object |  | - |
| Thisframe | ChatWindow | - | ChatWindow object | - | - |
| Container | Container | - | Container | - | - |
| dArea | JEditorPane | - | Display the user message in pane | - | - |
| wArea | JTextArea | - | Writing area for client | - | - |
| Send | JButton |  | Button to send message | - | - |
| User | User |  | User obj | - | - |
| Timer | Timer |  | Timer object to flash window | - | - |
| Dis | DataInputStream |  | Inputstream to read object | - | - |
| Dos | DataOutputStream |  | Outputstream to write object | - | - |

1. Client

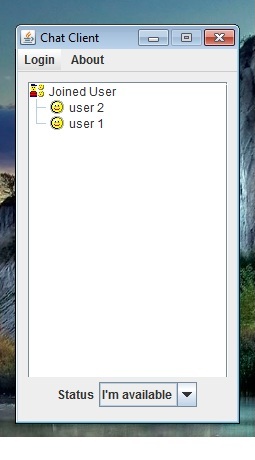
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | CONSTRAINTS | DESCRIPTION | MAX. LENGTH | SAMPLE |
| \_username | String | - | username | - | <User> |
| \_password | String | - | Password | - | - |
| \_server | String | - | Server hostname | - | - |
| \_port | Int | - | Port number of server | - | - |
| panel | UserPanel | - | UserPanel obj to render list |  | - |
| Dialog | ChatWindow | - | ChatWindow object to communicate with other users | - | - |
| Dis | DataInputStream | - | Streams to read /write object | - | - |
| Dos | DataOutputStream | - | Streams to read /write object | - | - |
| \_userlist | Vector | - | Vector to store user object | - | - |
| frameTable | HashMap |  | Store ChatWindow frames for the clients in communication | - | - |
| User | User |  | User obj | - | - |
| Socket | Socket |  | Socket to communicate with server | - | - |
| Frame | Client |  | Client object | - | - |
|  |  |  |  | - | - |

**6.0 Screenshots**

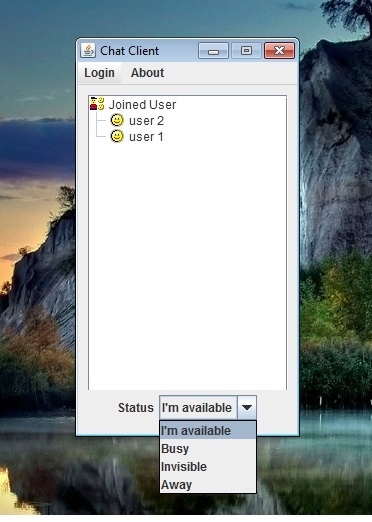
**Login Window**

****

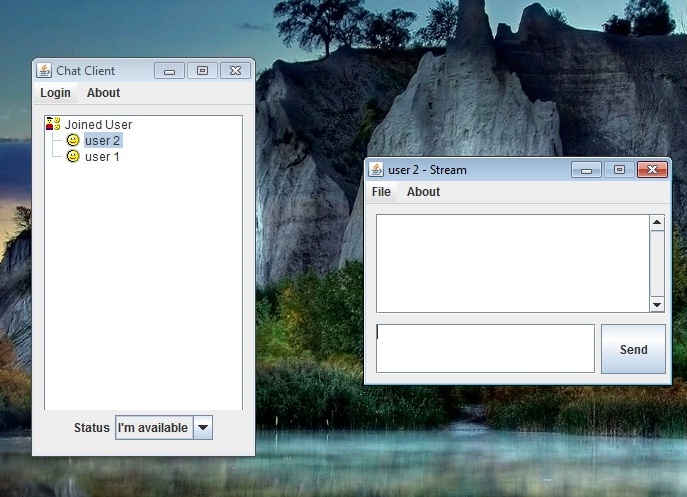
**Main Window**

****

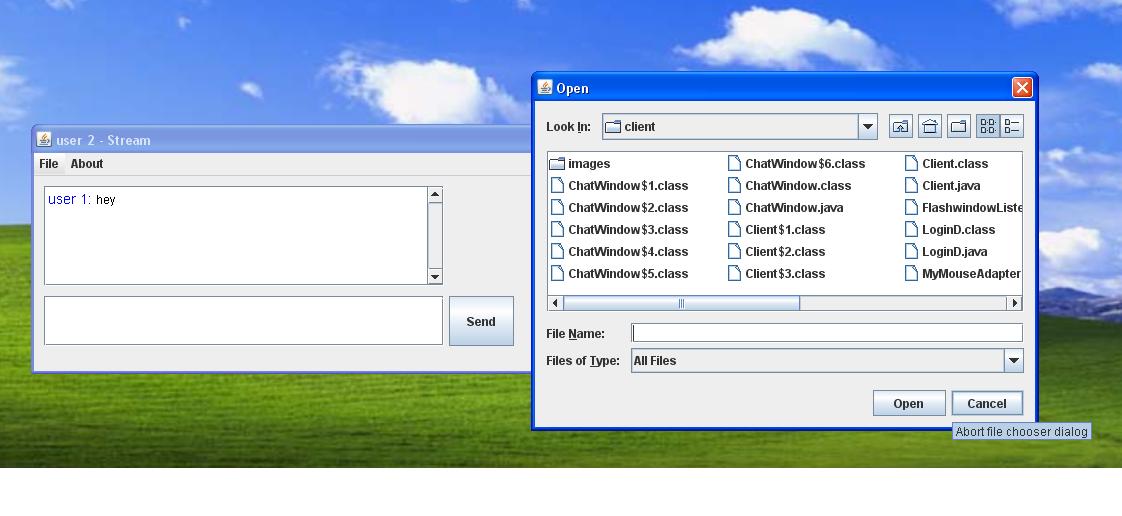
**Status Change Instance**

****

**Chat Window**

****

**Send File**

****

**7.0 Software Testing**

**[UNIT TESTING]**

* Each module after developed is tested individually to test whether there is any further error showing and it is done at very elemental level to test if there are any bugs.

**[INTEGRATION TESTING]**

* Each module is integrated with the other modules and then tested. There are many chances of errors and bugs showing after integration of modules and hence they are cleared out.

**8.0 System Evolution & Conclusion**

**In the future this system** **can be more optimized with more set of operation and functions to use services such as file sharing, voice calling, and block systems, internet surfing, interactive games, possibly more making it a comprehensive chat application suited best for being in constant touch to everyone & share.**

It can also be used in companies for the head user to stay connected to employees working under that person and give feedbacks about the work consistently/ for a emergency mass message or announcement to everyone logged on the server.