

---

# Software Requirements Document for CyChat

---

**Author:** Group 05

**Andrew Dorman**

**Henri Bai**

**Matt Herbst**

**Bryan McCoy**

<b>Version</b>	<b>Date</b>	<b>Author</b>	<b>Change</b>
0.1	09/13/07	SM	Initial Document
1.0	9/27/07	AD	SRS-1
2.0	10/15/07	BM	SRS-2

---

# Table of Contents

---

<b>1</b>	<b>Introduction.....</b>	<b>3</b>
1.1	Purpose .....	3
1.2	Scope .....	3
1.3	Definitions, acronymns, abbreviations .....	3
1.4	References .....	4
1.5	Overview .....	4
<b>2</b>	<b>Overall Description.....</b>	<b>5</b>
2.1	Product Perspective .....	5
2.2	Product functions .....	7
2.3	User characteristics.....	36
2.4	Constraints.....	36
2.5	Assumptions and Dependencies .....	36
<b>3</b>	<b>Specific Requirements .....</b>	<b>37</b>
3.1	External Interface Requirements .....	37
3.2	FEATURES .....	37
3.3	Performance requirements .....	46
3.4	Design Constraints.....	46
3.5	Software System Attributes .....	47
3.6	Other Requirements.....	48
<b>4</b>	<b>Appendix.....</b>	<b>49</b>
4.1	Screenshots .....	49
4.2	Screen Flow Diagram .....	51

# 1 Introduction

## 1.1 PURPOSE

The purpose of this document is to describe the expected high-level operation of the CyChat visual chat system. Naturally, the design is subject to change as the project is further developed and refined. The descriptions given are tailored for the customer, and thus they are of a mostly non-technical nature. They are intended to give the customer a general idea of how the program idea was interpreted.

## 1.2 SCOPE

This document will denote the users of the CyChat system and some of their specific abilities. A general overview of the sequence of common interactions among the user, client software, and the server will also be described. A focus on foreground tasks is made, since although there are several background tasks such as avatar management, these are transparent to the user and thus mostly unfit for a customer-focused document.

## 1.3 DEFINITIONS, ACRONYMS, ABBREVIATIONS

Term	Description
Action area	An area inside of a room that has a server-side script associated with it. Instead of moving to that spot when the user clicks inside the action area, the server executes the script. The user has no way to tell if an area is an action area unless the background or images displayed by the server indicate that it is one.
Avatar	The visual appearance of a user. This can be changed by the user and is then made visible to all other users in the same room.
Client	The program used by the user to connect to a server. It has a graphical interface to display information and the room in which the user is.

Room	A specific "channel" in a chat server. A user is in one room at one time on a server. Normally, messages sent go to all people in the same room as the sender (with the exception of whispers and global messages). Rooms are visually represented as a rectangle with a background image downloaded from the server.
Server	The program used by the developer to set up a common area for users to chat in. It manages all of the rooms and clients connected to it. "Server" may also be used as a general term for the collection of rooms and scripts that the server program is composed of.
Closet	The local collection of Avatars available for wearing by a user. Avatars can be imported into the closet. This closet is stored locally on the user's computer.
Image item	An image that can be seen by all users in a room that is not static like the room's background and is not user-controlled like an Avatar. They can be modified by action areas.
Bookmark	A saved configuration of a server that has been previously visited. A bookmark consists of server name, network address, and port.
Wear	To use an avatar. A user is said to be wearing an avatar when it is in use as the user's visual representation.
Dewear	To remove an avatar from use.
World	The collection of rooms hosted on a server

## 1.4 REFERENCES

None

## 1.5 OVERVIEW

None

## 2 Overall Description

### 2.1 PRODUCT PERSPECTIVE

A related product that really stands out is The Palace. The Palace is a visual chat program from about ten years ago that served as the motivation for this project. In The Palace, users are represented as avatars and chat visual through chat bubbles. Users can navigate through different rooms on the server and can interact with others and the environment.

Other chat clients in general can also be considered related and competitive products, as one of the main goals of our project is communication. Other chat clients, such as AIM or MSN Instant Messenger, offer textual chat. CyChat provides the basic textual chat along with addition features to enhance the user experience.

#### 2.1.1 Concept of Operations

CyChat is a program made up of two important components – the server and the client. The server needs to be initially set up by a Developer. The Developer determines the characteristics and behavior of the server. Once a server is set up, Users can connect to the server.

The client is the front end of the application. The client is a means of communicating with the server, and in turn, other clients. The client is where the User logs in to a server, and also where the User controls all aspects of CyChat. Chatting, movements, interactions all are handled by the client, which sends the data to the server then displays the resulting conditions.

#### 2.1.2 Major User Interfaces

A Screen Flow Diagram of the use cases is included in the APPENDIX

##### 2.1.2.1 Example Screenshot and description

Screenshots are included in the APPENDIX

#### 2.1.3 Hardware Interfaces

Keyboard, Monitor, Mouse (standard I/O), modem (network access required)

2.1.4 Software Interfaces

None

2.1.5 Communication Interfaces

Broadband Connection

2.1.6 Memory Constraints

None

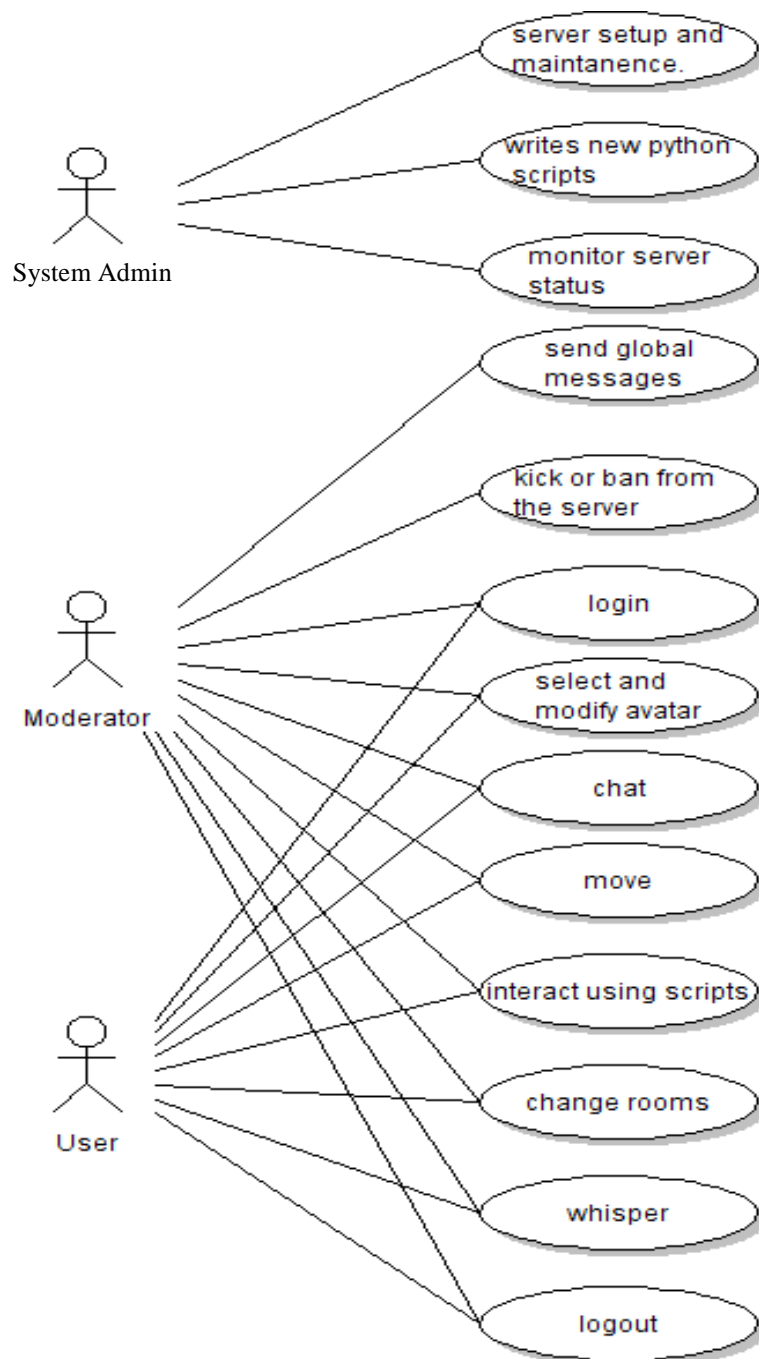
2.1.7 Operations

None

2.1.8 Site Adaptation Requirements

None

## 2.2 PRODUCT FUNCTIONS



The use case basis for CyChat is that there will be 3 actors with different abilities and access rights. The user is the main actor who can use the program but has no control over any other users. The Moderator is a user who has additional privileges that can affect the entire user community. The System Admin is the individual in charge of starting and maintaining the server and is the only one who has access to the source code.

### 2.2.1 Server Setup and Maintenance

The initial startup and continued maintenance of the server running the CyChat program.

Actors: System Admin

Main Flow:

1. Install the program
2. Run the program from the java command line

Alternate flow:

*If any problems are encountered:*

2. a. Restart the server.

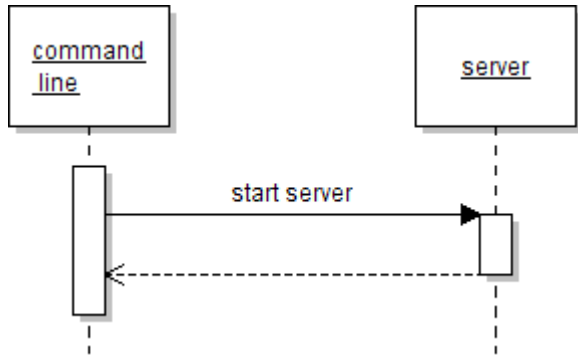
Precondition:

Server is not already running

Post-condition:

Server is running.

### 2.2.1.2 Sequence Diagram for Server Setup and Maintenance.



### 2.2.2 Write New Scripts

Add additional functionality to the program by writing and inserting new scripts for the application. Most will add some form of interactivity to the program such as movement or games.

Actors: System Admin

Main Flow:

1. Write and develop the new script.
2. Add to the server program.
3. Restart the server to have the changes take effect.

Alternate flow: none.

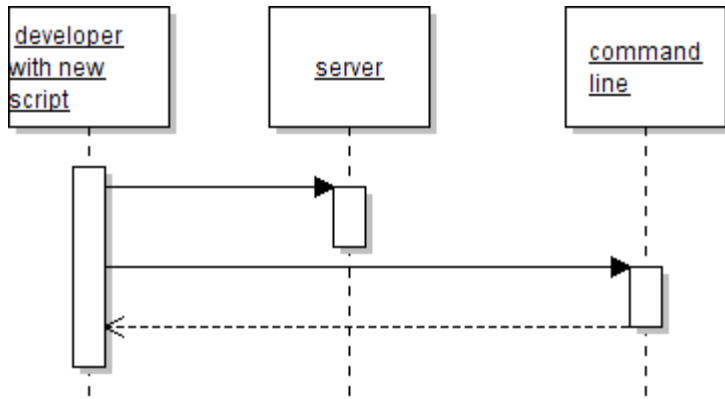
Preconditions:

Server is taken offline.

Post-condition:

Server is running with new scripts.

### 2.2.2.1 Sequence diagram for Adding New Python Scripts.



### 2.2.3 Monitor Server Status

Be able to remotely check up on server to be sure that it is running correctly.

Actors: System Admin

Main Flow:

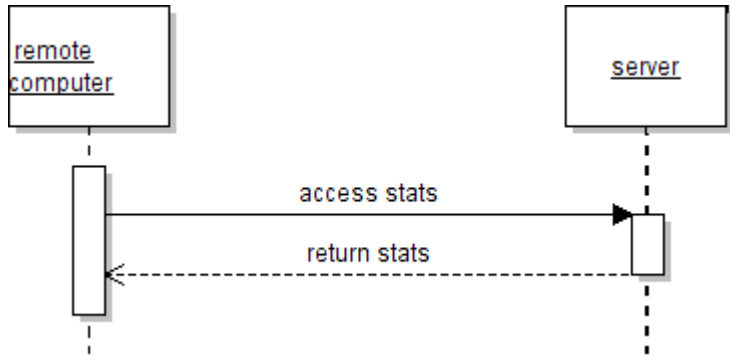
1. Access the server remotely.
2. Check usage and other statistics.

Alternate Flow: none.

Precondition: none.

Post-condition: none.

### 2.2.3.1 Sequence Diagram for Monitor Server Status



#### 2.2.4 **Send global message**

Allows a moderator to send a message that can be seen by all other users who are logged into the server

Actors: Moderator

Main Flow:

1. Moderator presses a hotkey to enable a state that lets the user or moderator to enter text using a keyboard
2. Moderator types a message
3. Moderator clicks a button on the screen, or presses enter (both are equivalent)
4. Message is sent to the server
5. Server receives the message and checks to see what type it
6. Server sends message to everyone who is logged on the server
7. All clients receive the message
8. All clients read the message type, and shows the message on the screen

Alternate Flow: none.

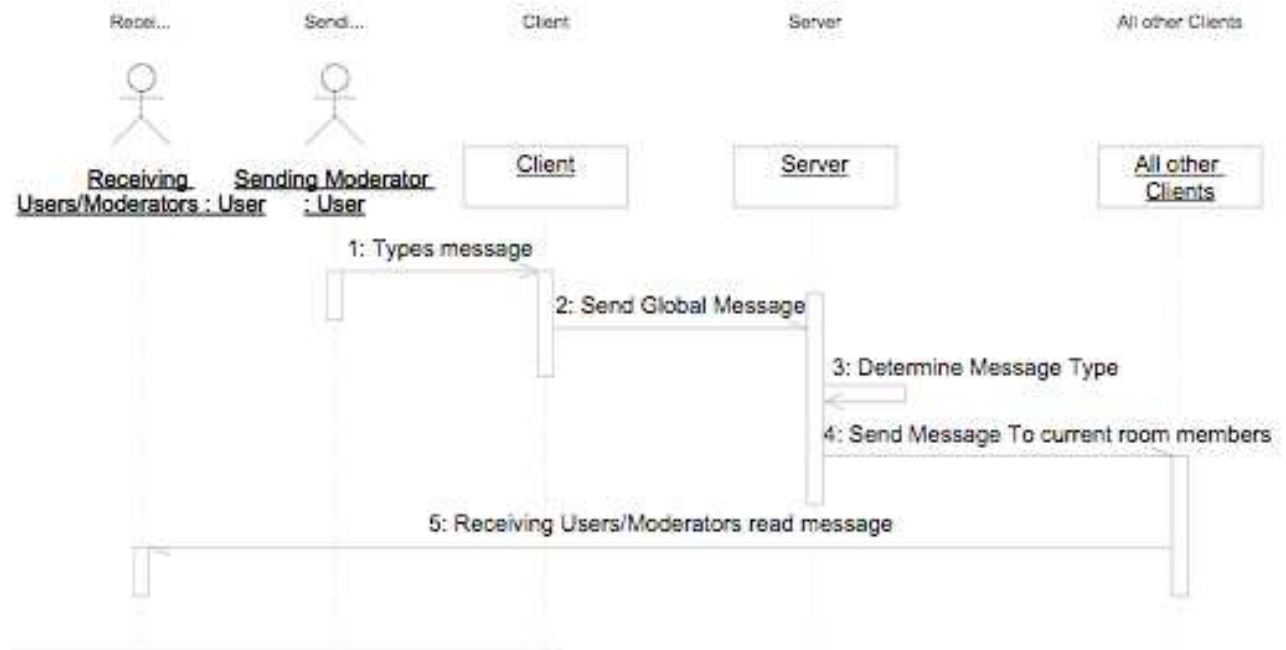
Precondition:

Moderator must be log in and is in a valid room

Post-condition:

Message has been logged by the server

### 2.2.4.1 Sequence Diagram for Send global message



### 2.2.5 Kick or ban from the server

Allows a moderator to kick or ban a certain user from the server.

Actors: Moderator

Main Flow:

1. Moderator sends the command to kick a particular User
2. Moderator Client sends the command message to the server
3. The server parses the message, and checks to see if the User is logged on.
4. The server sends a message to the user to notify the user is being kicked
5. The server forcefully disconnects the user from the server.

Alternate Flow: none.

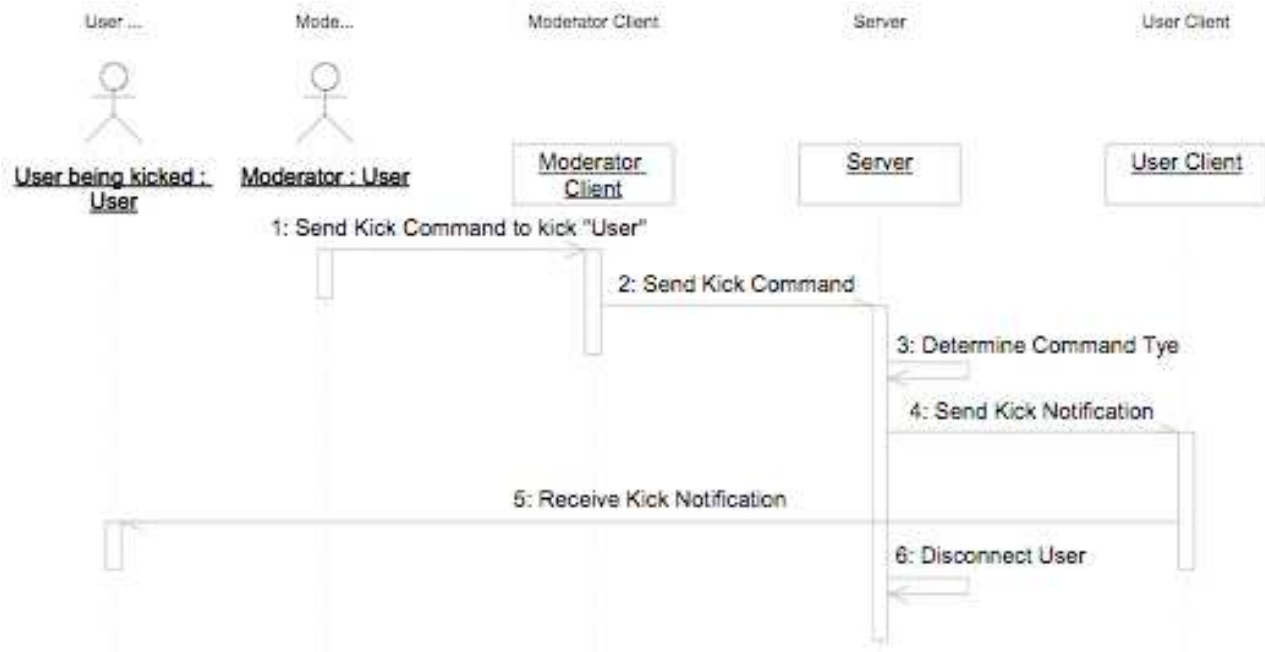
Precondition:

User to be kicked must be logged in. Moderator must be logged in with Moderator privilege.

Post-condition:

User is disconnected from the server. Kick has been logged on the server

### 2.2.5.1 Sequence Diagram for Kick or ban from the server



### 2.2.6 Login

A user logs into server.

Actors: User, Moderator

Main Flow:

1. User selects log in.
2. User selects server settings.
3. User enters desired username.
4. Client connects to specified server.

Alternate Flow:

*For a Moderator to log in, replace step 4 with:*

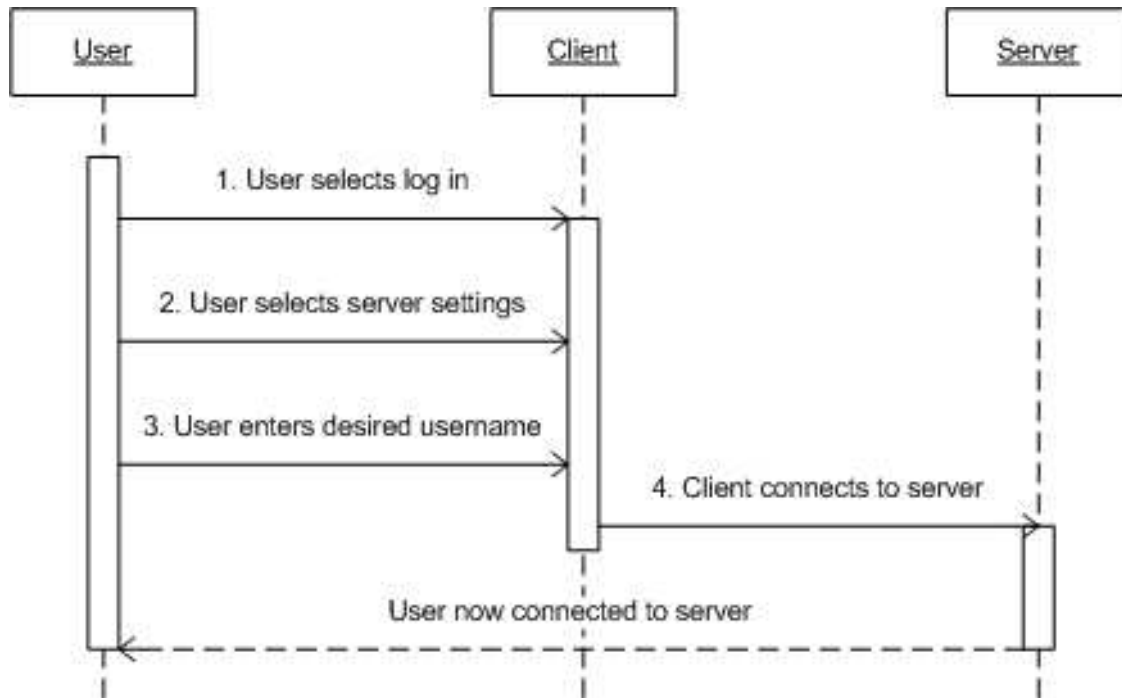
4. a. User selects Moderator log in.
4. b. User enters valid Moderator password
4. c. Client connects to specified server, User has Moderator capabilities.

Precondition: none

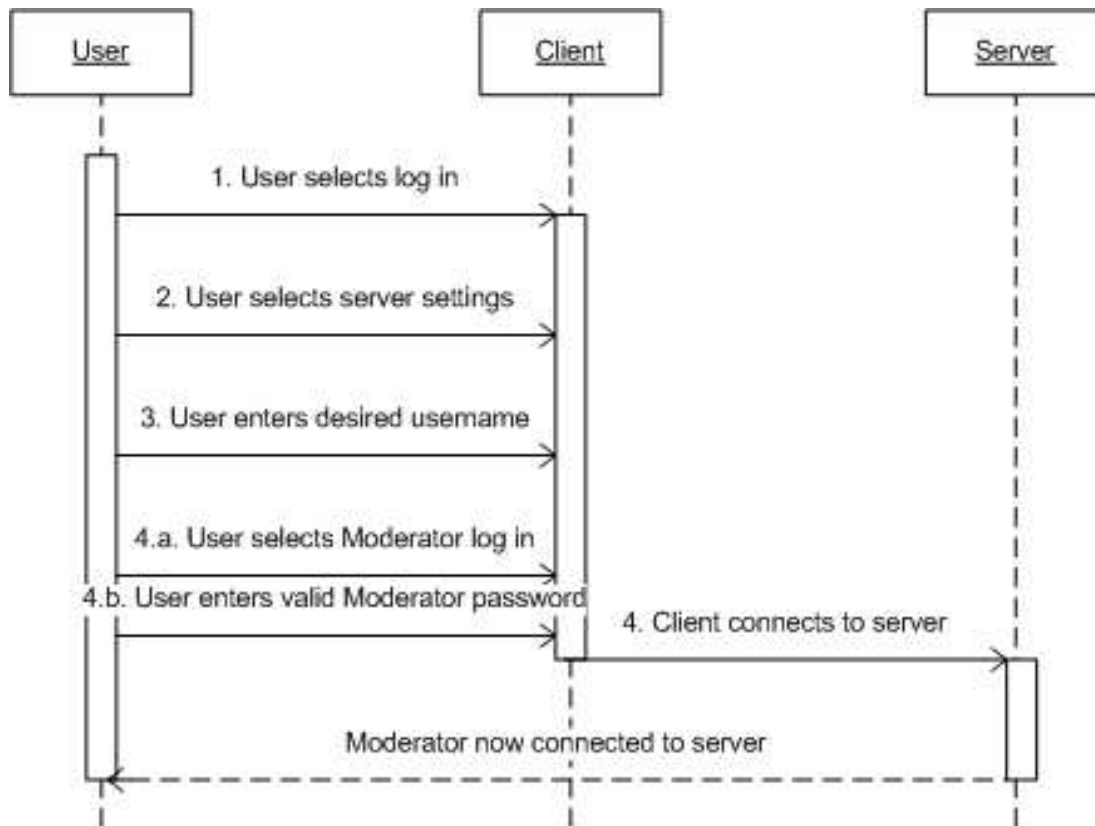
Post-condition:

User is logged in to server.

### 2.2.6.1 Sequence Diagram for Login



### 2.2.6.2 Sequence Diagram for Login (Moderator login)



### 2.2.7 **Select and modify avatar**

Allows a user to select a new avatar, from their repository or import a new image.

Actors: User, Moderator

Main Flow:

1. User selects avatar window.
2. User selects avatar from repository
3. Avatar is uploaded to the server and distributed to the other clients

Alternate Flow:

*User wants to import an avatar. Replace step 2 with:*

2. a. User selects image file from local computer
2. b. New avatar now in repository

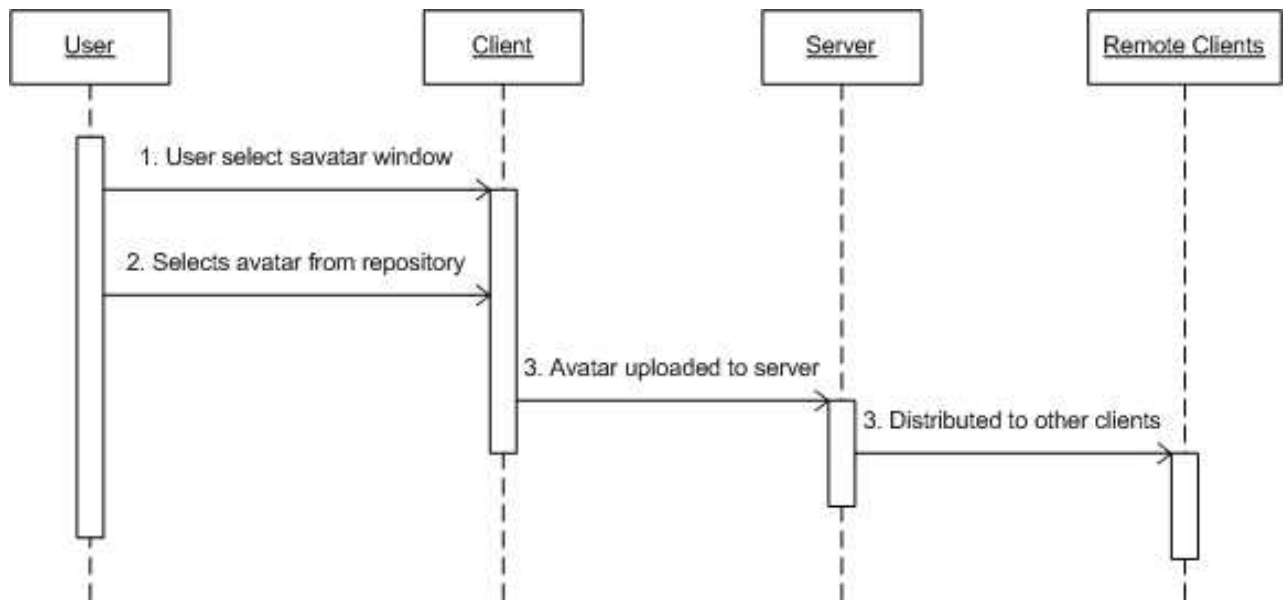
Preconditions:

User must be logged in to server.

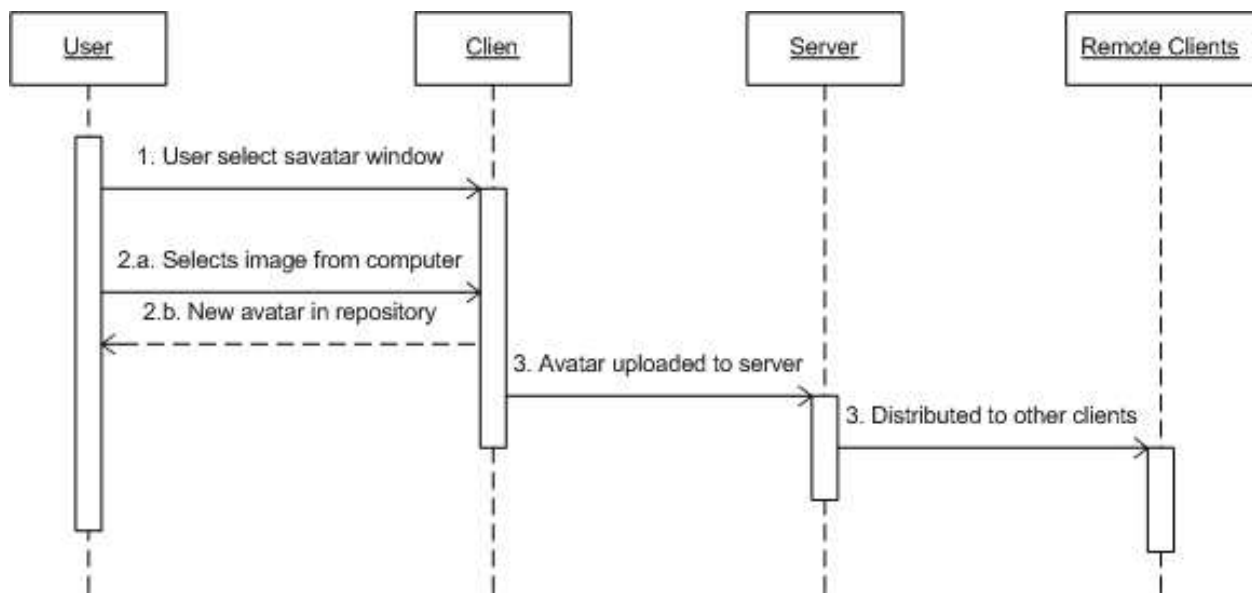
Post-condition:

User's display is updated to show new avatar.

### 2.2.7.1 Sequence Diagram for Select and modify avatar



### 2.2.7.2 Sequence Diagram for Select and modify avatar (import)



### 2.2.8 Chat

Allows a user to send a message that can be seen by other users in the same chat room.

Actors: User, Moderator

Main Flow:

1. User activates a state that lets the user or moderator to enter text using a keyboard
2. User or Moderator types a message
3. User or Moderator clicks a button on the screen, or presses enter (both are equivalent)
4. Message is sent to the server
5. Server receives the message and checks to see what type it
6. Server sends a message to everyone in the chat room the sender is in
7. All clients receive the message
8. All clients read the message type, and shows the message on the screen for the user to see

Alternate Flow: none.

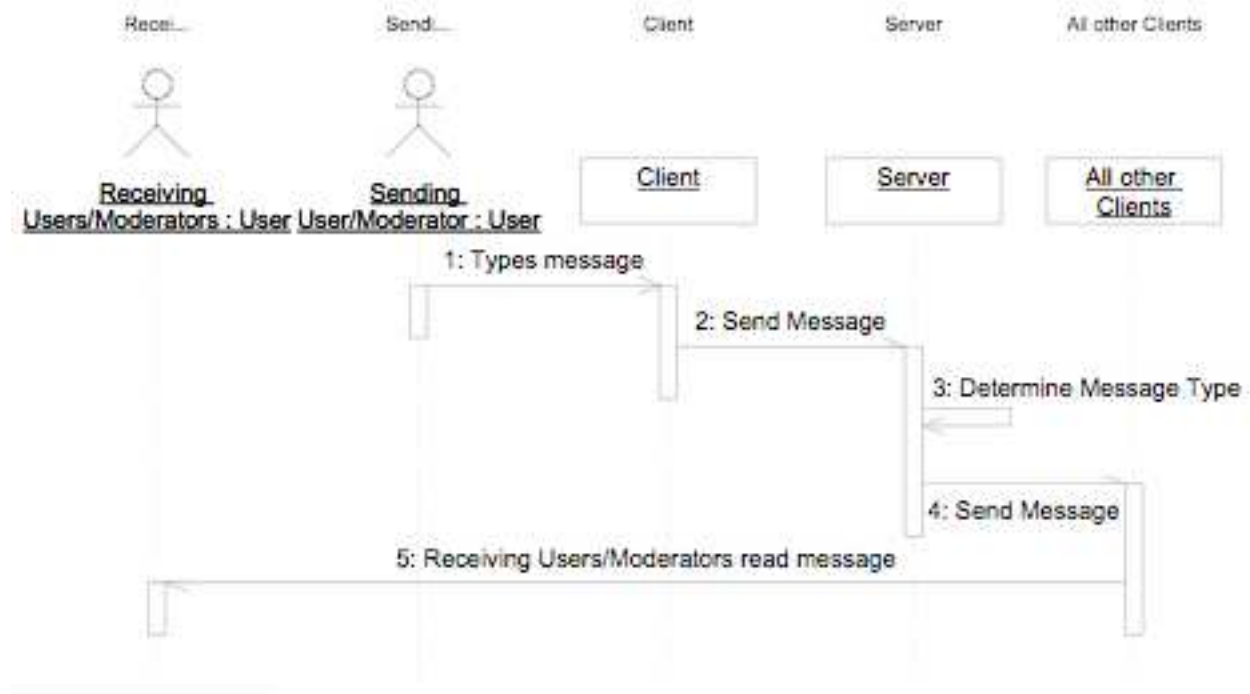
Precondition:

User or Moderator must be log in the server.

Post-condition:

Chat has been logged by the server

### 2.2.8.1 Sequence Diagram for Chat



### 2.2.9 Move

A user moves from one point in a room to another. This movement is reflected on the screens of everyone in the room.

Actors: User, Moderator

Main Flow:

1. User clicks somewhere in the room window that isn't an action area
2. The client sends a movement message to the server
3. The server analyzes the point and determines that the user can move there
4. The server updates the client's location state
5. The server sends the new client location to everyone in the room, including the client
6. The client updates the display to reflect the movement based on the server message

Alternate Flow: none.

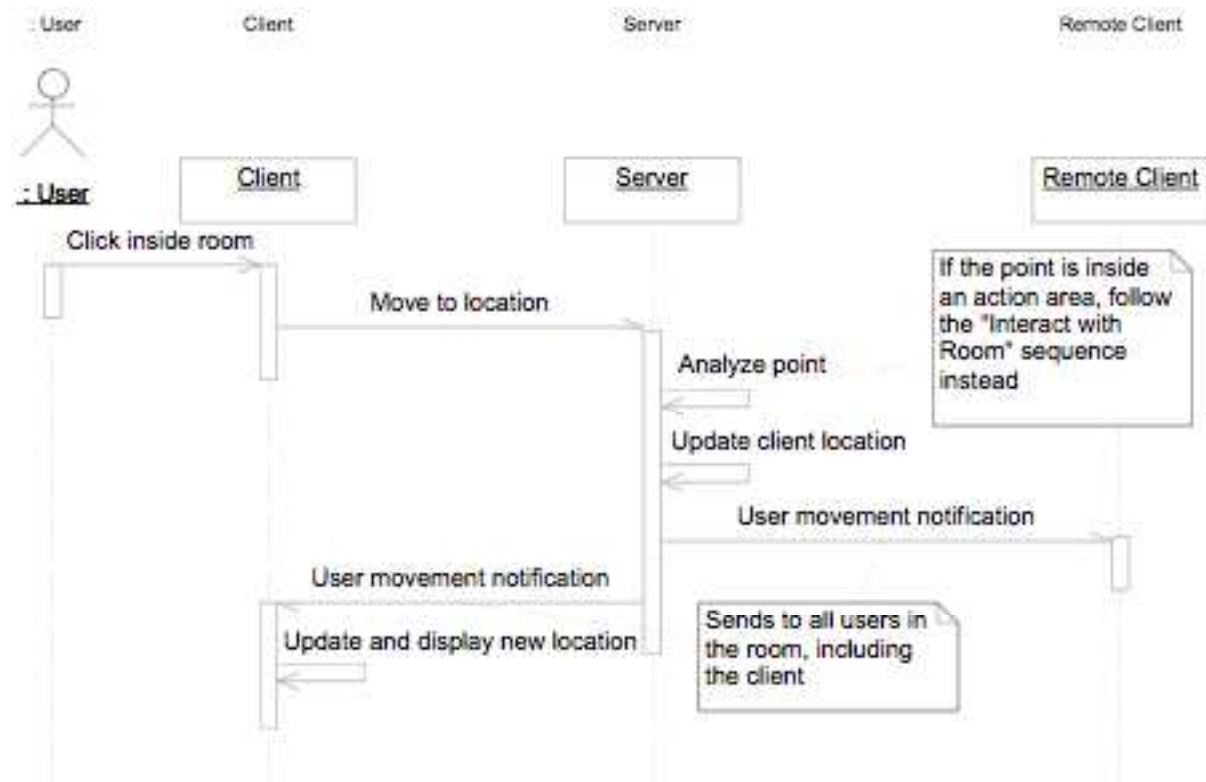
Precondition:

The user is logged onto a server.

Post-condition:

The user is in the new location in the room.

### 2.2.9.1 Sequence Diagram for Move



### 2.2.10 Interact with environment

A user clicks on an action area inside of a room.

Actors: User, Moderator

Main Flow:

1. User clicks somewhere inside an action area in the room window
2. The client sends a movement message to the server
3. The server analyzes the point and determines that it is an action area, and executes the appropriate script
4. The server updates room and client states as requested by the script
5. The server sends the new room and client information to everyone in the room, including the client
6. The client updates the display to reflect the changes

Alternate Flow: none.

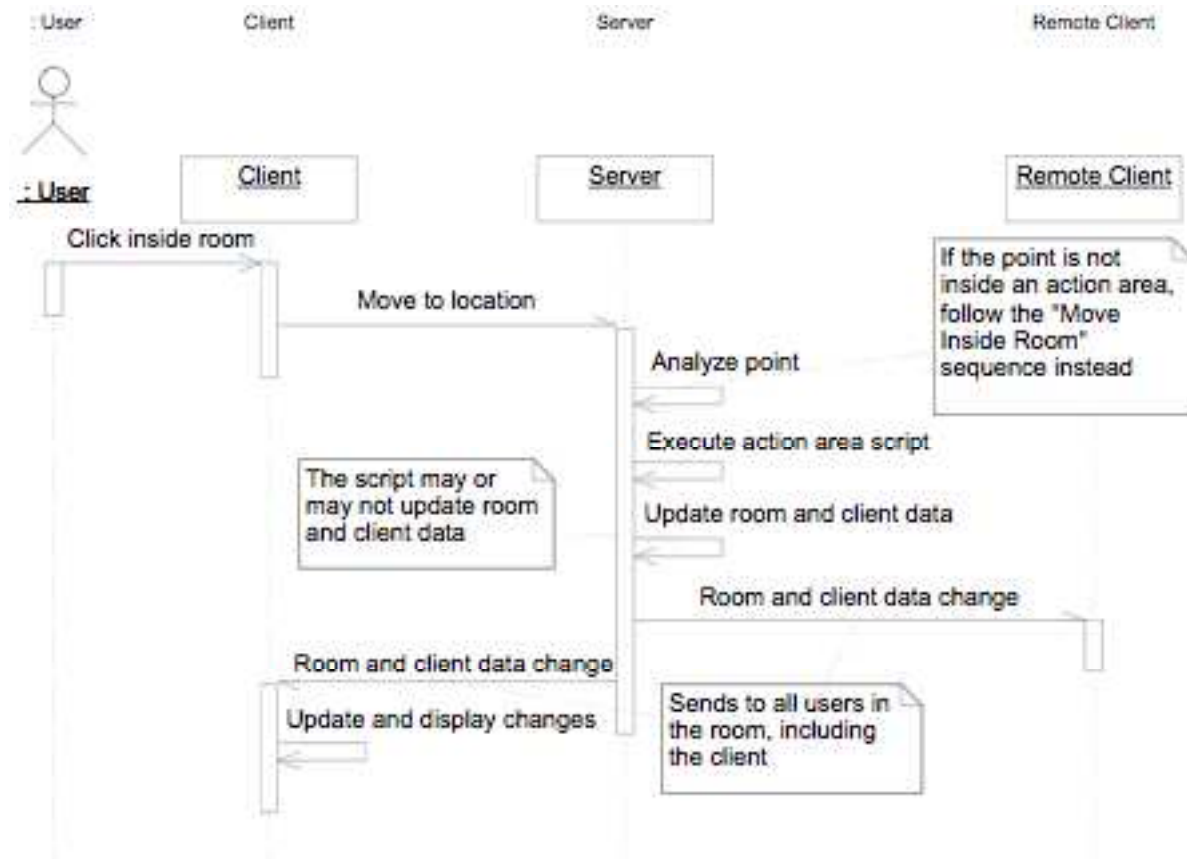
Precondition:

The user is logged onto a server.

Post-condition:

The room and client states are altered correctly and reflected on all clients

### 2.2.10.1 Sequence Diagram for Interact with environment



### 2.2.11 Change rooms

A user moves to another room. All users in the old and new rooms are notified of the change.

Actors: User, Moderator

Main Flow:

1. User requests room list
2. The client requests the room list from the server
3. The server sends the current room list to the client
4. The client displays the room list
5. The user selects a room to go to
6. The client sends the room change request to the server
7. The server removes the user from the old room
8. The server sends user departure message to all clients in the old room
9. The server sends user entering message to all clients in the new room
10. The server adds the user to the new room and sends room information (name, background, users in room) to the client
11. The client updates the display to reflect the new room information

Alternate Flow:

*At step 6, the room is found to be full. Replace steps 7-11 with:*

7. A. The server sends a room full message to the client
7. B. The client notifies the user that the room is full

*Client does not have background image when room info is received. Replace step 11 with:*

- 11.a. The client updates the display to reflect the new room information, replacing the background with a temporary image
- 11.b. The client sends a file request to the server, requesting the background image
- 11.c. The server sends the requested image
- 11.d. The client caches the image and updates the display

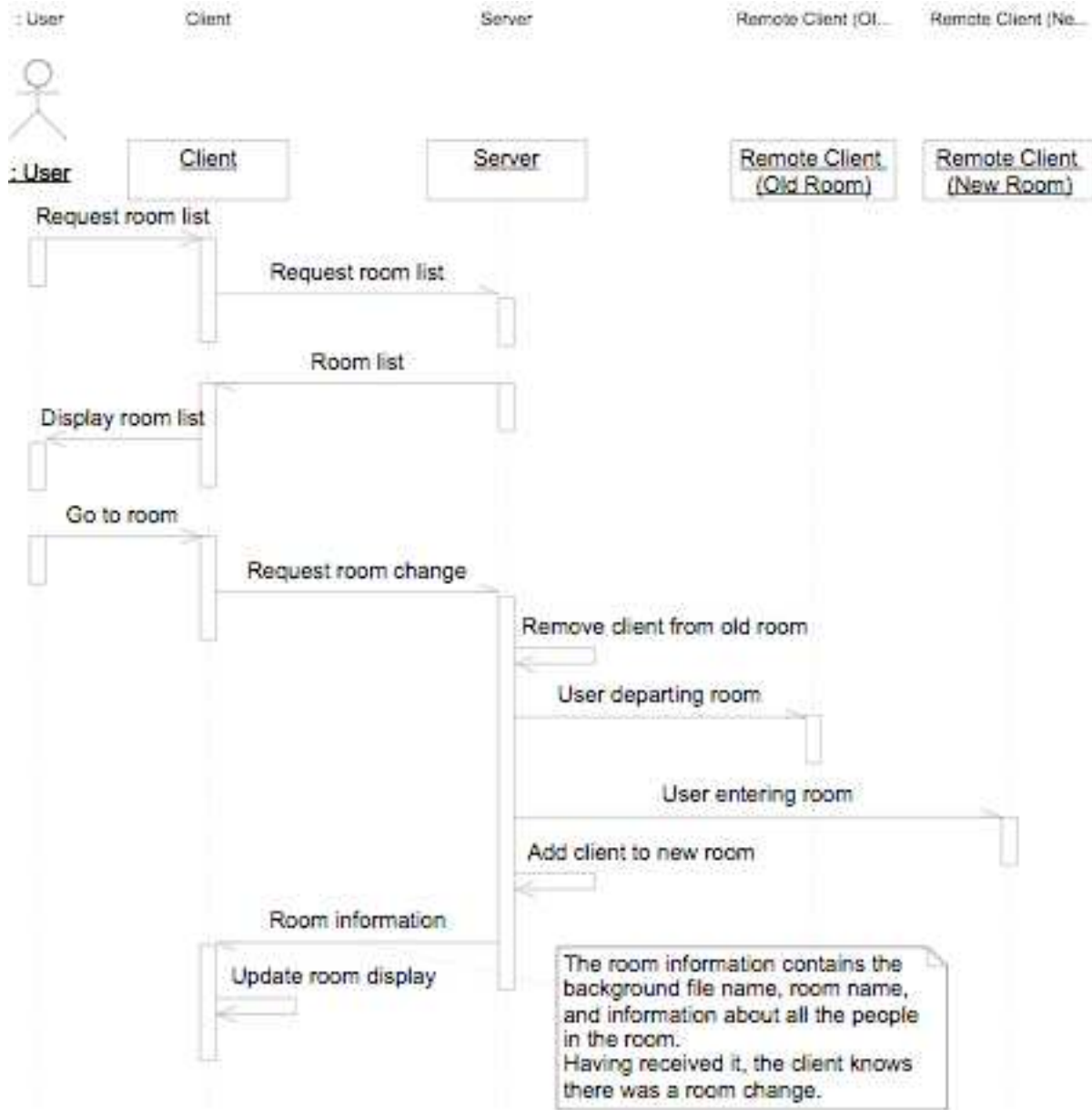
Precondition:

The user is logged onto a server.

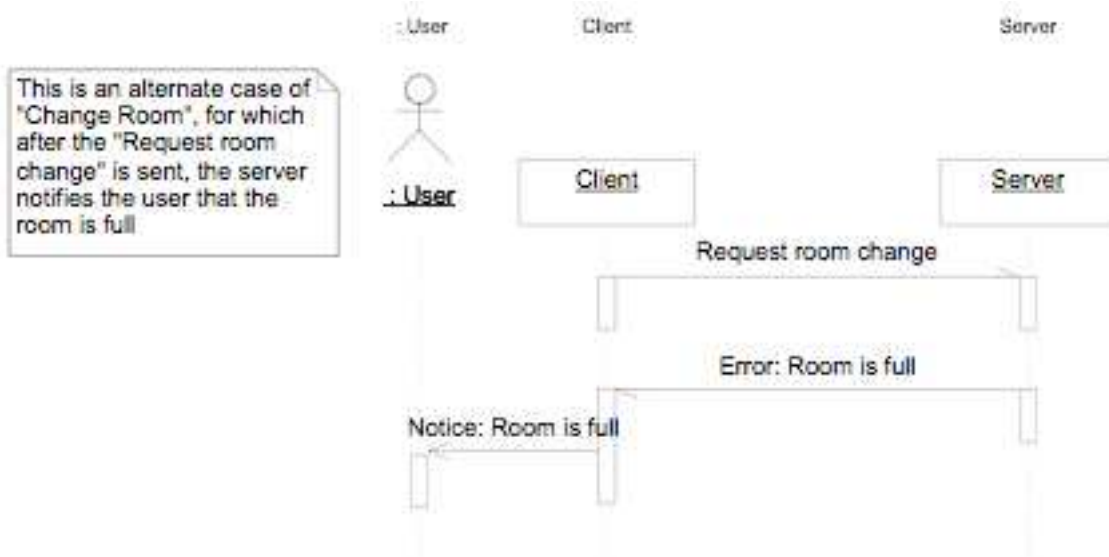
Post-condition:

Client is in the new room.

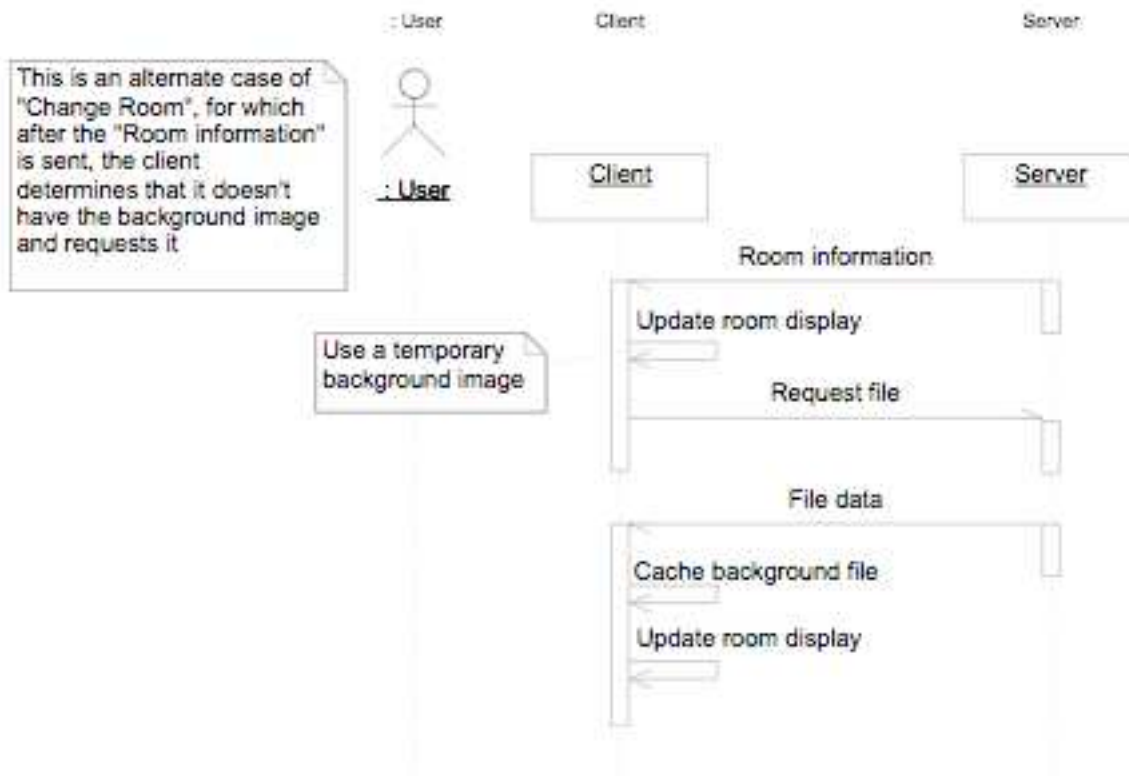
### 2.2.11.1 Sequence Diagram for Change rooms



### 2.2.11.2 Sequence Diagram for Change rooms (room full)



### 2.2.11.3 Sequence Diagram for Change rooms (doesn't have background image)



### 2.2.12 **Whisper**

Allows a user to send a message that can only be seen by a specific user.

Actors: User, Moderator

Main Flow:

1. User activates a state that lets the user or moderator to enter text using a keyboard
2. User or Moderator types a message
3. User or Moderator clicks a button on the screen, or presses enter (both are equivalent)
4. Message is sent to the server
5. Server receives the message and checks to see what type it
6. Server checks to see if the receiver is logged in
7. If the receiver is logged in, send the message to the receiver
8. The receiver receives the message
9. The message is shown on the screen for the receiver only.

Alternate Flow:

*Replace step 7 and all steps afterward in the main flow with the following:*

7. a. If the receiver is not logged in, send an error message to the sender. The message contains the error: 'the receiver is not logged on'
7. b. Sender receives error message and shows the error to the sender's screen.

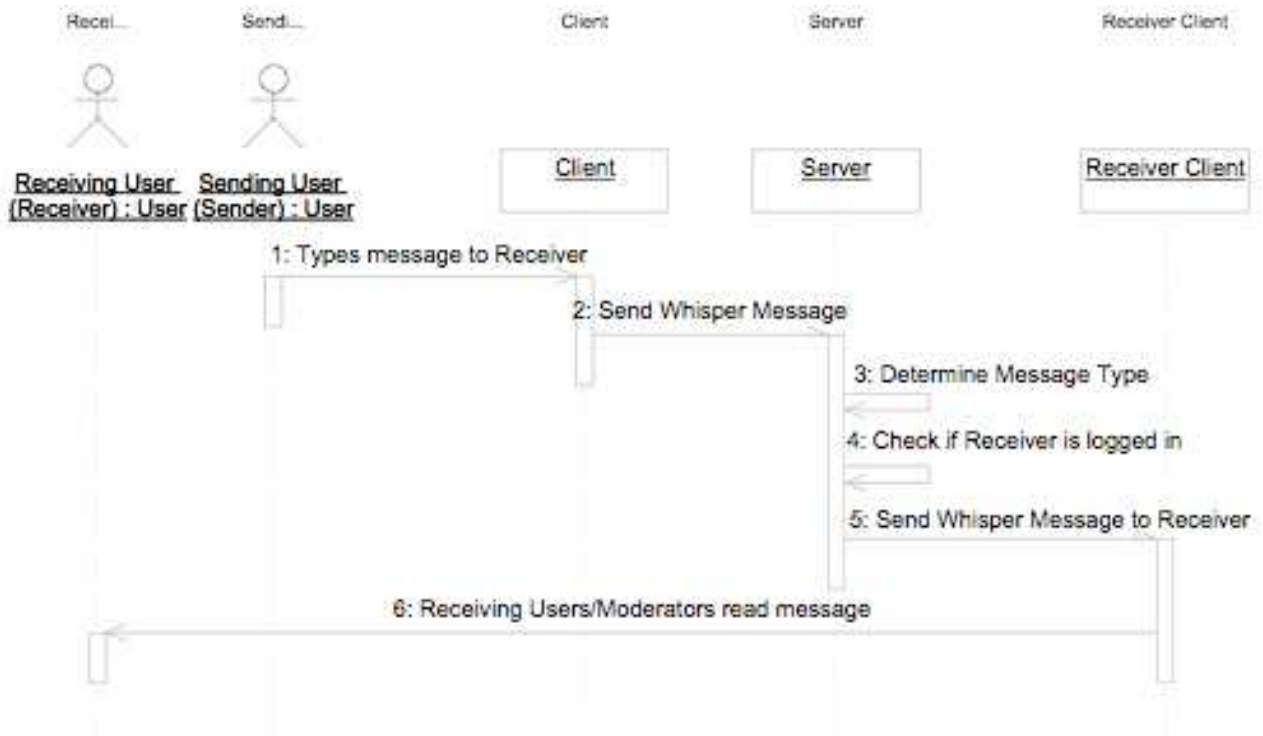
Preconditions:

User or Moderator must be log in and is in a valid room

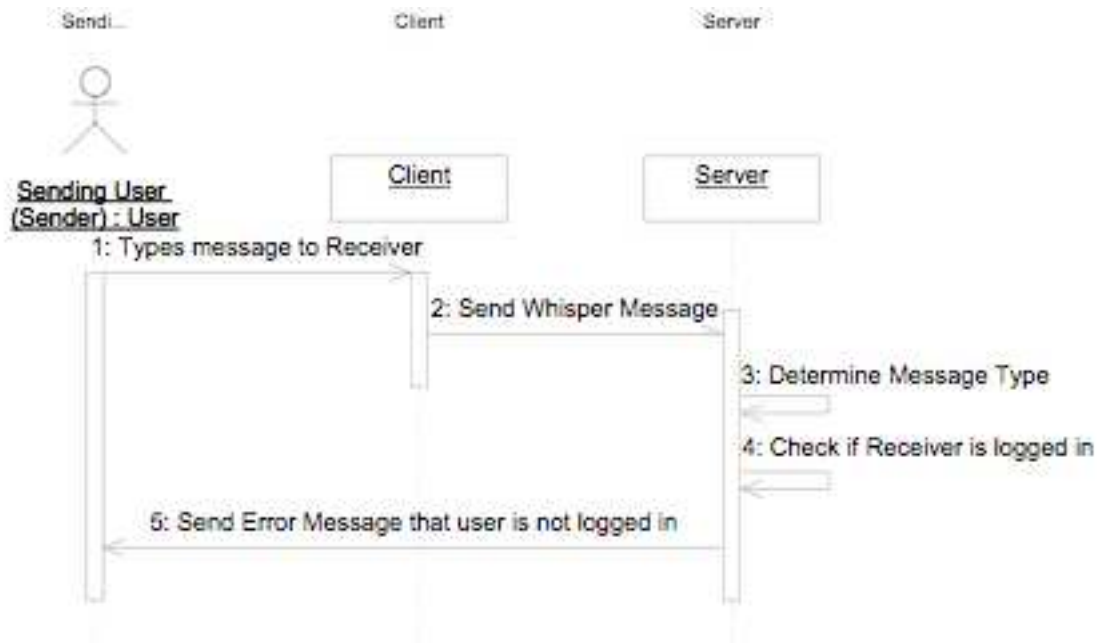
Post Conditions:

Whisper chat is logged on the server

### 2.2.12.1 Sequence Diagram for Whisper



### 2.2.12.2 Sequence Diagram for Whisper (recipient not logged in)



### 2.2.13 Logout

Allows a moderator to kick or ban a certain user from the server.

Actors: User, Moderator

Main Flow:

1. User selects log out.
2. Client disconnects from server.

Alternate Flow: none.

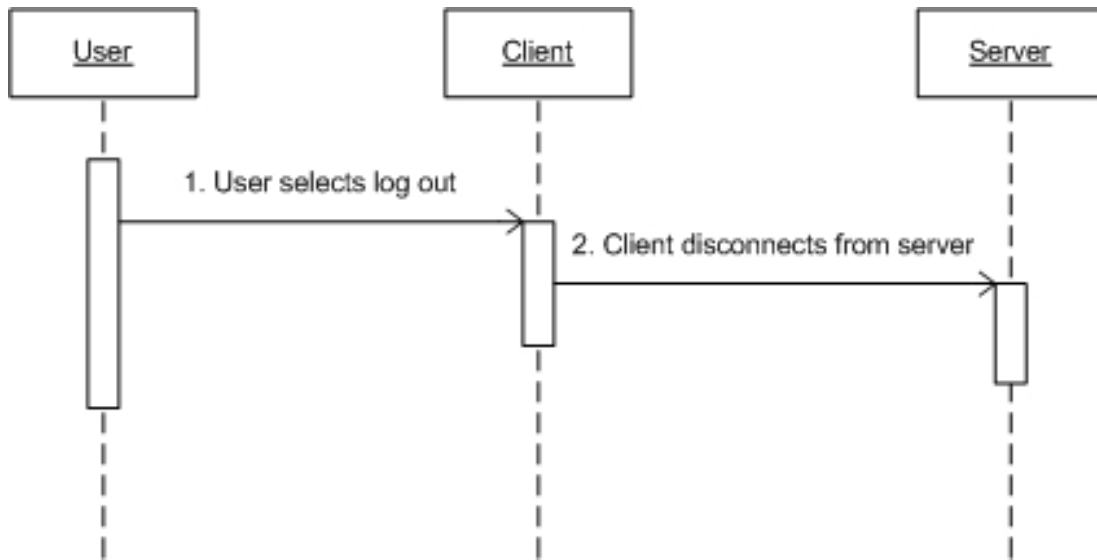
Precondition:

User must be logged into server.

Post-condition:

User is disconnected from the server.

### 2.2.13.1 Sequence Diagram for Logout



## 2.3 USER CHARACTERISTICS

### **System Admin:**

The System Admin is the initial user of the program. He is responsible for setting up the server and keeping it running. The System Admin is also the one who can access the source code and make modifications such as adding functionality through scripts.

### **User:**

The user is the most common actor. He will be able to log on to the server using the client and make use of the various parts of the program but will not be able to have a direct influence on other users. He can chat, change his image, and use scripts.

### **Moderator:**

The moderator has all the same functionality as the user but has other added abilities. He can send global messages to all users. Also if a user is not abiding by the rules of the server, he has the ability to kick them from it.

## 2.4 CONSTRAINTS

Runs on Windows XP, Vista, Ubuntu Linux, and Mac OS X  
- Works for Java 1.4+

## 2.5 ASSUMPTIONS AND DEPENDENCIES

[None]

## **3 Specific Requirements**

### **3.1 EXTERNAL INTERFACE REQUIREMENTS**

#### **3.1.1 User Interfaces**

Keyboard

Mouse

Monitor

#### **3.1.2 Hardware Interfaces**

A Pentium III processor or equivalent

A vesa compatible graphics card

CD Rom Drive

#### **3.1.3 Software Interfaces**

Operating System capable of running a java virtual machine version 1.4

#### **3.1.4 Communications Interfaces**

Broadband Internet Connections

### **3.2 FEATURES**

#### **3.2.1 Connection**

3.2.1.1 Upon initializing the server, the system administrator shall be able to specify the TCP/IP port to open to listen for clients

3.2.1.2 Upon attempting to connect, the client shall be able to specify a user name, a network address, and port to which to attempt to connect.

3.2.1.3 Moderators shall be able to log in to a server with a password to gain moderator access.

- 3.2.1.4 The client and server shall use a one-way encryption scheme for the moderator password on login.
- 3.2.1.5 The server shall support at least 100 concurrent client connections.
- 3.2.1.6 [Optional] The server shall implement a message limit from clients, such that when more than X messages in a Y-second interval are received, the client is kicked from the server and given warning of the limit. The values X and Y shall be defined server-wide by the system administrator. The system administrator shall be able to disable the message limit entirely.
- 3.2.1.7 [Optional] The system administrator shall be able to specify a regular expression that defines a valid name for non-moderator users. With such a regular expression defined, unless the user is a moderator, if a connecting client sends an invalid name the server shall kick the user and notify the user that the given name was invalid.

## 3.2.2 Environment Actions

- 3.2.2.1 The system administrator shall be able to define a set of action areas for each room.
- 3.2.2.2 Action areas shall be invoked only when a user in the room clicks within its borders.
- 3.2.2.3 A user shall not be able to view any of the above information about action areas within a given room.
- 3.2.2.4 [Optional] The system administrator shall be able to define a set of chat actions for each room. (They can do/see the same things as action areas, but can see the contents of the message that activated them instead of the position that was clicked)
- 3.2.2.5 [Optional] Chat actions shall be invoked only when users send a room chat message within the room in which the chat action is contained.

### 3.2.3 Action Area Abilities

3.2.3.1 Action area scripts shall be able to change state variables defined by the system administrator.

3.2.3.2 Action area scripts shall be able to move any user (identified by name) to another room (identified by name).

3.2.3.3 Action area scripts shall be able to move any user to another position within the room.

3.2.3.4 Action area scripts shall be able to manipulate existing image item properties (identified by name) in the same room.

3.2.3.5 Action area scripts shall be able to read the state variables defined by the system administrator.

3.2.3.6 Action area scripts shall be able to read the name of the room to which they belong.

3.2.3.7 Action area scripts shall be able to read the name of the user that activated it and what position in the room was clicked to do so.

3.2.3.8 Action area scripts shall be able to read the names and positions of all users in the room to which they belong at the time of invocation.

3.2.3.9 [Optional] Action area scripts shall be able to generate anonymous chat bubbles at any position in the room.

3.2.3.10 [Optional] The server shall set a timer when a script is activated and terminate it if it runs for more than X seconds, then send a non-fatal server error message to all users in the room informing them that a script timed out. The value X shall be defined server-wide by the system administrator.

### 3.2.4 Moderator abilities

- 3.2.4.1 The server shall ignore the kick command and send an error acknowledgement back to the client if the kick command is initiated by a non-superuser.
- 3.2.4.2 The server shall disconnect the user with the supplied user name when a superuser initiates the kick command.
- 3.2.4.3 If the user name is not found when the kick command is initiated, the server shall ignore the kick command.
- 3.2.4.4 The client shall provide an interface to kick a user from the server.
- 3.2.4.5 [Optional] The server shall log any instance of the kick command being initiated and mark it being as successful or not.
- 3.2.4.6 [Optional] The moderator clients shall log the kick command being initiated.
- 3.2.4.7 [Optional] The user client shall log the successful kick command being initiated.
- 3.2.4.8 The server shall ignore the Send Global Message command and send an error acknowledgement back if the command is initiated by a non-superuser.
- 3.2.4.9 The server shall send a copy of the Global Message to all users on the server.
- 3.2.4.10 The client shall provide an interface to send a global message.
- 3.2.4.11 [Optional] The server shall log any instance of the Send Global Messages.
- 3.2.4.12 [Optional] The moderator clients shall log the send global message command being initiated.
- 3.2.4.13 [Optional] The user clients shall log the successful send global message being initiated.

### 3.2.5 Graphical Interface

- 3.2.5.1 The client shall support multiple interface languages using Unicode. The user shall be able to select an interface language.
- 3.2.5.2 When the client wants to get a list of rooms, the client shall send a request to the server.
- 3.2.5.3 When the server receives a request for a list of rooms, the server shall send the room listing to the client.
- 3.2.5.4 When the client requests the room listing, the client shall display the room listing provided by the server.
- 3.2.5.5 When the client requests the user listing, the client shall display the user listing provided by the server.
- 3.2.5.6 [Optional] The room listing will provide user count per room.
- 3.2.5.7 [Optional] The user listing will provide the current room per user.
- 3.2.5.8 A bookmark shall consist of server name, network address, and port.
- 3.2.5.9 The client shall save bookmarks across all sessions to local disk.
- 3.2.5.10 The client shall provide an interface for adding bookmarks.
- 3.2.5.11 The client shall provide an interface for remove bookmarks.
- 3.2.5.12 The client shall hold a record of previously sent messages in a message pane.
- 3.2.5.13 The client shall log all chat type messages and add it to the message pane.
- 3.2.5.14 When the user presses the up key, the client shall replace the contents of the message box with the most recent message sent by the user.
- 3.2.5.15 The client shall provide an interface to export everything from the chat log from memory onto the disk.

3.2.5.16 [Possible] The user shall be able to traverse through sent messages using the up and down arrow keyboard keys.

### 3.2.6 World

3.2.6.1 The server administrator shall be able to specify the Moderator password for the world.

3.2.6.2 The server administrator shall set the characteristics for each of the rooms in the world.

3.2.6.3 The client shall relay to the server where a user clicks on the screen.

3.2.6.4 The server shall determine if action is to be taken based on the location clicked.

3.2.6.5 The server administrator shall be able to set a specific world occupancy limit.

3.2.6.6 The server shall not allow anymore users to log in to the world when occupancy is full.

3.2.6.7 The server administrator shall be able to set a specific room occupancy limit for each room.

3.2.6.8 The server shall not allow users to enter a room when occupancy is full.

3.2.6.9 A user shall be in exactly one room at any time.

3.2.6.10 A user shall leave one room before entering another.

3.2.6.11 A user shall be able to move directly from any room in the world to any other room in the world.

3.2.6.12 The server shall store information for each user logged in to the world. This information includes username, the user's current avatar, the current room the user is in, the current position in the room, and whether or not the user is a moderator.

### 3.2.7 Chat

- 3.2.7.1 The server and client shall support Unicode in chat messaging.
- 3.2.7.2 Users shall be able to chat publicly to all other users in a room.
- 3.2.7.3 The server shall distribute room chat messages to the clients for all users in a room.
- 3.2.7.4 The client shall display room chat messages from all users in a room, in the form of speak bubbles emanating from the avatar of the user who sent the message.
- 3.2.7.5 [Possible] The user shall be able to set visual characteristics of the speak bubbles for a message, such as color and shape.
- 3.2.7.6 Users shall be able to whisper, or private chat, to any other user logged in to the world, whether in the same room or in different rooms.
- 3.2.7.7 The server shall distribute whispered messages to a specific user's client.
- 3.2.7.8 The client shall display received whispered messages in a distinguishable font.
- 3.2.7.9 If the whispered message is sent from a user in the same room, the client shall display the message in a speak bubble emanating from the avatar of the user who sent the message.
- 3.2.7.10 If the whispered message is sent from a user not in the same room, the client shall display the message in a square text box appearing in the upper left hand corner of the screen. The message shall identify the sending user.
- 3.2.7.11 The server shall distribute global messages to all users logged in to the world.
- 3.2.7.12 The client shall display received global messages in a square text box appearing in the upper right hand corner of the screen.
- 3.2.7.13 The client shall not identify the moderator who sent the global message.

## 3.2.8 Avatars

- 3.2.8.1 The client shall display avatars for each user in the same room.
- 3.2.8.2 Each user shall have a position in a room and their avatar displayed at that position.
- 3.2.8.3 The users own avatars shall be stored locally on their own computer.
- 3.2.8.4 The avatars shall be tracked by the server and assigned a unique identifier.
- 3.2.8.5 The client shall provide multiple preloaded avatars to use.
- 3.2.8.6 If the avatar is new to the server, the client shall upload it.
- 3.2.8.7 The user's avatar shall be distributed to everyone in the room.
- 3.2.8.8 The server shall accept partially transparent avatars.
- 3.2.8.9 The avatars shall be no larger than x bytes and y pixels. X and Y will be determined at a later time by the system admin.
- 3.2.8.10 The server shall accept avatars in .jpg or .png format
- 3.2.8.11 A user shall wear exactly one avatar at a time.
- 3.2.8.12 A user shall be able to un-wear their avatar
- 3.2.8.13 The avatar menu shall include the option to import a new avatar.
- 3.2.8.14 The avatar menu shall include the option to remove an old avatar.
- 3.2.8.15 The avatar menu shall include the option to wear a different avatar.
- 3.2.8.16 The user name shall be displayed below their avatar.
- 3.2.8.17 The client shall keep a collection of all the users' avatars (closet).
- 3.2.8.18 If the user is not wearing an avatar, they shall be displayed as a default avatar.
- 3.2.8.19 The default avatar shall be preloaded and come with the client.

3.2.8.20 When an avatar changes, the change shall be appropriately displayed on all other users in the same room screen.

3.2.8.21 Only 1 closet shall be open on a user's client at a time.

### 3.2.9 Room Appearance

3.2.9.1 Background images shall only be added by the developer.

3.2.9.2 The server shall support the background image in .jpg or .png format

3.2.9.3 Each room shall have only 1 background image.

3.2.9.4 Each background image shall be permanently stored on the server.

3.2.9.5 The background image shall be of a constant dimension for all rooms in the server.

3.2.9.6 The server shall allow multiple image items in each room.

3.2.9.7 Each image item shall have a position, an image, and a unique name determined by the system admin.

3.2.9.8 The server shall support image items in .jpg or .png format.

3.2.9.9 Image items shall be no larger than x bytes and y pixels. X and Y will be determined at a later time by the system admin.

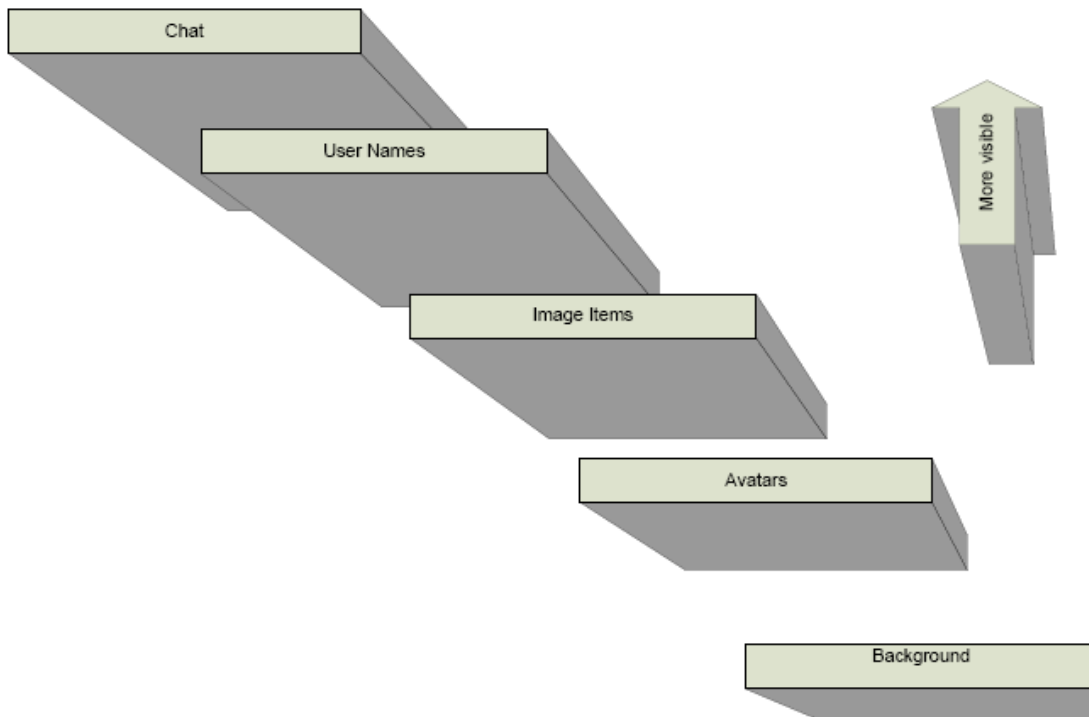
3.2.9.10 Action areas shall be able to change the image and the position of the image.

3.2.9.11 Only the system admin shall be able to add or delete image items.

3.2.9.12 Foreground images that are partially transparent shall be supported.

3.2.9.13 The rooms, avatars, and foreground images shall be displayed in 24 bit color

3.2.9.14 The images in the room shall have a set order of depth according to this diagram.



### 3.3 PERFORMANCE REQUIREMENTS

- 3.3.1 There shall be a time limit for a python script to be run on the server.
- 3.3.2 The client shall communicate efficiently with the server under a limited internet
- 3.3.3 The server shall handle a maximum of 100 concurrent connections efficiently.

### 3.4 DESIGN CONSTRAINTS

Runs on Windows XP, Vista, Ubuntu Linux, and Mac OS X

1 GHz processor, broadband connection, 512 MB memory, 1 GB free disk space

Works for Java 1.4+

Swing GUI for client

Java Sockets

Uses Jython for server-side Python scripting

## **3.5 SOFTWARE SYSTEM ATTRIBUTES**

### **3.5.1 Reliability**

3.5.1.1 The client shall immediately reconnect to the server in the event of a dropped connection.

3.5.1.2 The server shall not have more than 100 concurrent connections

### **3.5.2 Availability**

3.5.2.1 The program shall be freely available under a GNU license.

### **3.5.3 Security**

3.5.3.1 The server shall provide an authentication mechanism to log moderators into the server.

3.5.3.2 All python scripts shall be run on the server.

### **3.5.4 Maintainability**

3.5.4.1 The code shall be properly documented and segmented to allow further maintenance.

### 3.5.5 Portability

3.5.5.1 The software shall be able to run on multiple platforms with minimal dependencies.

3.5.5.2 The complete software client excluding dependencies shall take a minimal size on the storage medium.

## 3.6 OTHER REQUIREMENTS

None

## 4 Appendix

### 4.1 SCREENSHOTS

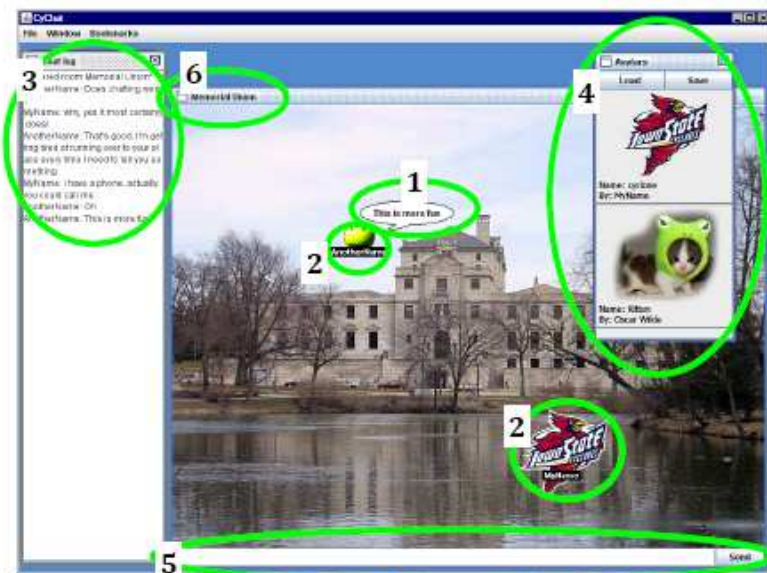
#### Connection screen

1. Main menu (gives connect/disconnect and other options)
2. Server information
3. User name



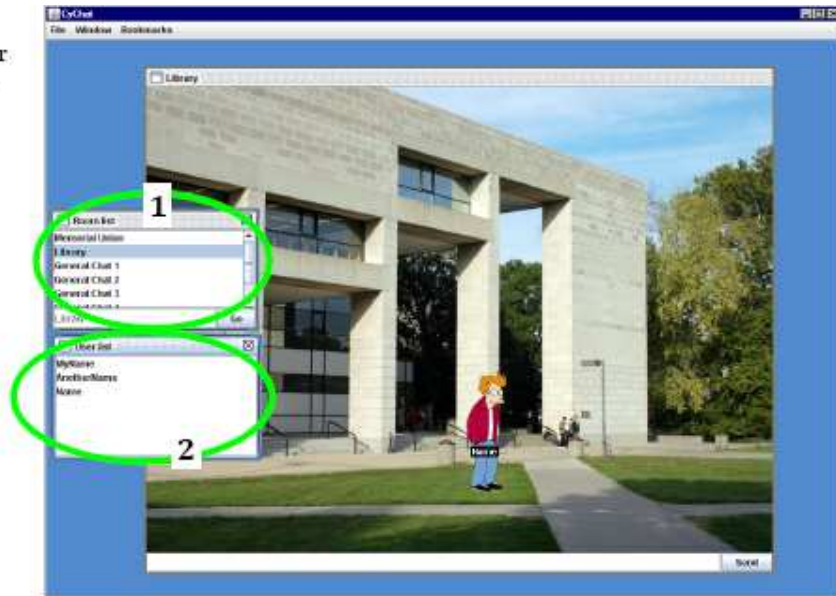
#### Chat demonstration

1. Chat bubble
2. Users (2)
3. Chat log
4. Avatar (appearance) selections
5. Chat box
6. Room name



### Alternate rooms

1. Current rooms on server
2. Current users on server



### Scripting potential

1. Users interact with room by clicking
2. Room generates whatever scripts dictate



