CS314 SimpleChat P2 Requirements & Use Cases

Caleb Tebbe & Zach Kaplan

April 1, 2013

**Requirements**

1. **Problems:** 
   1. Any individual that has the client program can connect to a SimpleChat server and impersonate another user.
   2. Users are unable to send private messages to other users.
   3. There is no channel functionality. Clients all chat within a global chat session.
   4. There is no functionality for a user to have someone else monitor their messages while they are away from the computer (in a meeting, out to lunch, etc.).
2. **Background Information:**

**Note:** SimpleChat is an OCSF chat program that enables multiple users to connect to a host running a SimpleChat server and exchange messages. For a list of currently implemented functionality upon which the following requirements are based, please refer to documentation for Phase 1 and Phase 2 of the SimpleChat program.

Issues Considered:

**Issue 1:** Should a user be able to log in under a duplicate user name?

**Option 1.1:** Allow for users to impersonate other users and hope that they are caught.

*Advantage:* A simpler solution to the issue:

**Option 1.2:** Implement a feature that disables the ability of users to impersonate others.

*Advantage:* Users can be certain of who they are chatting with.

*Decision: Choose option 1.2.*

**Issue 2:**  How are private messages handled?

**Option 2.1:** Create a separate channel through which messages can be sent between two parties.

*Advantage:* Uses channel feature that is required for phase 3

**Option 2.2:** Allow private messages to be sent without the need for opening a chat channel between the two parties

*Advantage:* Does not require users to open a new channel every time they wish to send a private message

*Decision: Choose option.2.2*

**Issue 3:** Allow client to join multiple channels?

*Decision: Yes, but this will require multiple channel commands to be implemented*

**Issue 4:** Will blocking work the same in channels and communication via private message?

*Decision: Yes,**blocking a user will apply to communications using channels as well as private messages.*

**Issue 5:** Can messages be forwarded to blocked, offline, or non-existent users?

*Decision: No, if the user that wishes to forward has blocked the intended recipient they must first unblock the intended recipient before setting up a forward. Similarly, the intended recipient of forwarded messages must be online before a forward can be initiated. Finally, the user must be a valid user within the chat network for the forward to setup properly.*

**Issue 6:** Can messages be forwarded to more than one user?

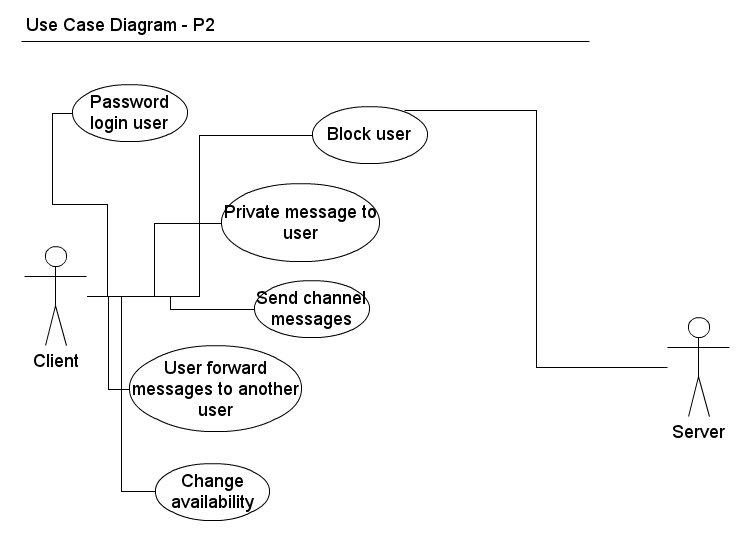
*Decision: No, only one user may be forwarded to at a time.*

1. **Environment and System Models:** These features that make up Phase 3 of SimpleChat are to be added at the same time the availability feature (discussed in the instructions for Project 2) is added. There are no other environmental considerations.
2. **Functional Requirements:**

* **Commands**: The following commands can be issued from the client UI.
  + **#channel <channel> <msg/null>**
    - Allows creation of channels as well as communication using existing channels.
    - “channel” is the channel name
    - “msg” is the message, if any (can be null)
    - If arg2 is null, initiates the creation of a chat channel
    - If arg2 is null, and arg1 is the name of an existing channel, the user becomes a member of that channel.
    - If arg2 is not null a message is sent across that channel.
    - A user can create and join multiple channels.
    - Blocking feature applies to messages sent via channels.
    - Channel messages are distinguishable
  + **#private <user> <msg>**
    - Sends a private message between two users
    - “user” is the recipient of the private message
    - “msg” is the message
    - Blocking functionality applies to private messages
    - Private messages cannot be sent to non-existent users
    - Private messages are distinguishable
  + **#forward** **<user>**
    - Sets up a forward which enables all messages sent from one user to forward to another user of their choice.
    - Forwarded messages are distinguishable.
    - Blocking rules apply to forwarding
    - Forwards cannot be set up with non-existent users
* **Operation**:
  + 2.1: If a channel is created the follow effects occur
    - The channel will exist even if the created logs off
    - A list of valid users for each channel is kept. This is updated as users log on and off
    - All blocking rules apply to channels as well
  + 2.2: If a forward is setup the following effects will follow
    - All messages send to the client that has set up the forward will be forwarded onto the target of the forward.
    - Blocking rules to forwarding. For example: user X cannot forward to user Y if user X has blocked user Y.
    - Blocking can occur after a forward has been established.

1. **Other Requirements:** All previous requirements (Phase 1, Phase 2) still apply. There are no other requirements.

**Use Case Diagram**

****

**Use Case Descriptions**

**Use Case:** Change availability

**Steps:**

*Actor Actions:*

1. Enter an availability command (#available or #notavailable)

*System Response:* Acknowledge change in availability.

**Use Case:** Forward messages to another user

**Steps:**

*Actor Actions:*

1. Enter forward command (#forward) followed by desired user.

*System Response:* Acknowledge creation of forward.

**Use Case:** Send channel messages

**Steps:**

*Actor Actions:*

1. Enter channel command (#channel) followed by desired channel name, and message.

*System Response:* Echo message as normal within the confines of the channel.

**Use Case:** Send private message

**Steps:**

*Actor Actions:*

1. Enter private message command (#private) followed by desired user name, and message

*System Response:* Echo message as normal within the confines of the private message.

**Use Case:** User login

**Steps:**

*Actor Actions:*

1. Upon logging in, user enters a password using the password command

*System Response:* Acknowledge successful creation of password

**Use Case:** Block a user

**Steps:**

*Actor Actions:*

1. Enter block command (#block) followed by desired user name

*System Response:* Acknowledge success of block