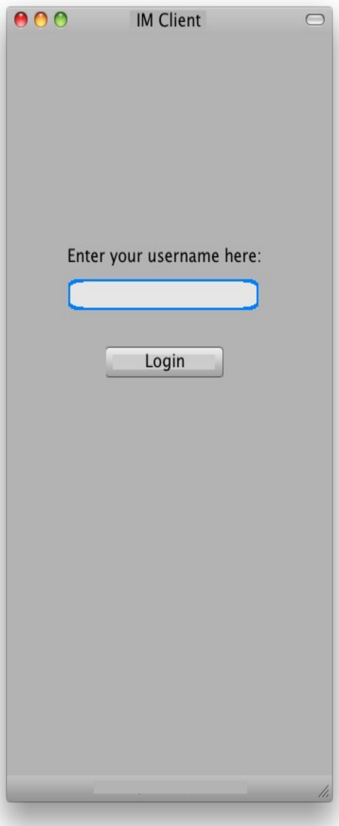
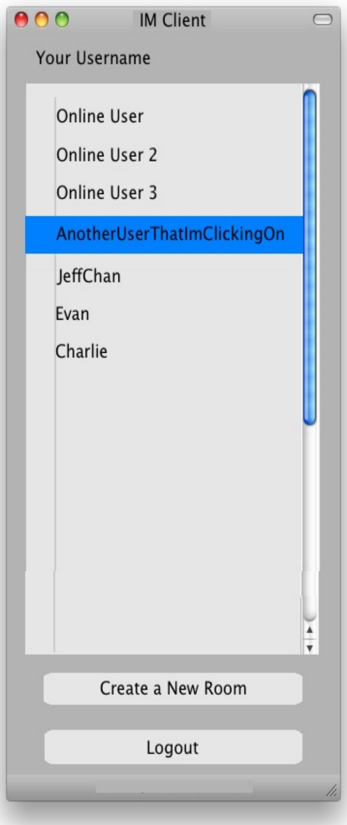
6.005 Project 2
Jeffrey Chan, Charles Liu, Evan Thomas

Client GUI Description

When the program is started, the user is presented with the login window (Figure 1) to type in their username. Hitting the enter key or clicking the login button will connect them to the server.

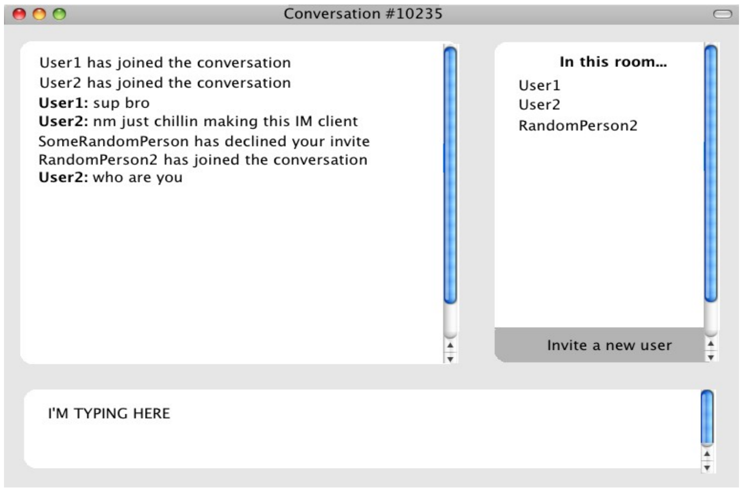




The next screen will be the “buddy list”, a list of all users that are online at the time. There are also two buttons on the bottom. The “Create a New Room” button will create a new chat room with only the current user in it. The “Logout” button will disconnect the user from the server, close all current chat windows, and display only the login window.

Single clicking a user will highlight the user in the buddy list. Double clicking a user will create a new room and invite that user to the chat. If the user already has a chat open in which the only other participant is the person that was double clicked, then that chat window will come into focus.

The conversation window has three main components. A log of the conversation appears in the top left, the people currently in the chat room are in the top right. The bottom text box is where a user types to input a message. Hitting enter will send the message to all users currently in the room.

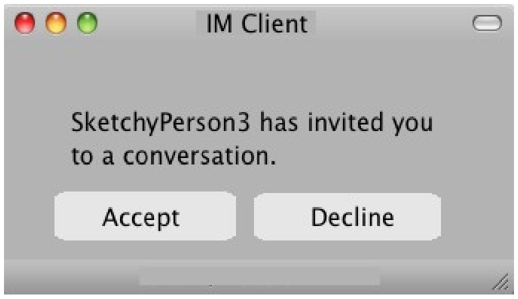


User1 has joined the conversation
User2 has joined the conversation
User1: sup bro
User2: nm just chillin making this IM client
SomeRandomPerson has declined your invite
RandomPerson2 has joined the conversation
User2: who are you

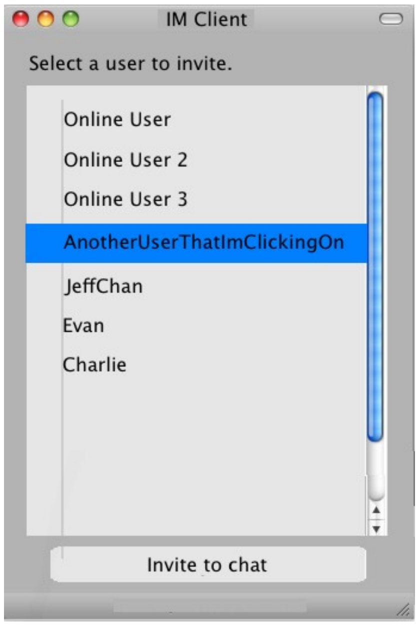
In this room...
User1
User2
RandomPerson2

I'M TYPING HERE

To invite additional people to the room, the user clicks on the "Invite a new user" button, which brings up a window with all online users not in the current room. From there the user can invite other online users to the chat, at which point the invite dialog box will pop up for the invitees, where they can subsequently accept the invitation, and be added to the chat room, or decline the invitation.



SketchyPerson3 has invited you to a conversation.



Select a user to invite.

Online User
Online User 2
Online User 3
AnotherUserThatImClickingOn
JeffChan
Evan
Charlie

6.005 Project 2
Testing Strategy

Jeffrey Chan, Charles Liu, Evan Thomas

1. Automated Testing

NotifyLogin:
-Login as the only user
-Login as with other users present
-Login with the same username as another user.
-Login, Logout, then Login again with the same username
-Cases violating grammar protocol

NotifyCreateRoom:
-Create one room
-Create multiple rooms
-Cases violating grammar protocol

NotifyInvite
-Create room, invite, logout, accept
-Create room, invite, leave room, accept (do we have leave room functionality)
-Create room, invite multiple people, accept accept
-Create room, invite multiple, decline, decline
-Create room, invite multiple, accept, decline
-Create room, Invite to user that is offline
-Create room, Invite from user that signs off immediately after
-Invite to nonexistent room number
-Invite to killed room number
-Invite to room number, killed, accepted
-Create room, Multiple users invite same person to same room
-Cases violating grammar protocol

NotifyAccept/Decline
-Create room, User invites, accept.
-Create room, User invites, decline.
-Create room, accept nonexistent invite.
-Create room, decline nonexistent invite
-Create room, accept to no longer existent room.
-Create room, decline to no longer existent room.
-Create 2 rooms, accept two invites from same user for different rooms
-Create room, accept two invites from two users for same room.
-Create room, accept one, decline one from two users for same room.
-Create room, decline both invites from two users for same room.
-Cases violating grammar protocol

NotifySay
-Create room, say message with no one in room
-Create room, say message with other users in the room
-Get invited to room, accept, say message
-Create room, say message with a new line character in the middle of string.
-Create room, say message with all character classes, tabs, whitespace, \r\n etc in the middle of string
-Create room, say message without \n at end
-Create room, say message with \n in the middle only.
-Create room, say message from different user(not sure if we still have username as part of NotifySay)
-Say message to a room not currently in.
-Say message to a room that is invited but not accepted
-Say message to a declined room.
-Say message same message to room 1: A and B and room 2: A and B and C

NotifyLogout
-Logout, say message to rooms that used to be members of
-Logout, have another user say something
-Logout, have another user invite you to a room
-Logout, accept prexisting invite
-Logout, attempt to create room
-Logout, attempt to invite someone
-Logout, attempt to say accept invites

2. Testing by hand

Login page
-Click login without anything typed in(empty string)
-Click login with illegal characters(symbols, \n, \r, \ etc)
-X out login button

Buddy List
-Click create new room with no users selected, with one user selected, with multiple users selected
-Logout with no rooms open
-Logout with rooms open
-X out buddy list

Room Window
-Press enter without any text
-Type text before and after a new user has joined
-Type text with no users in the room
-Type text with multiple users in the room
-Type text after user is invited, before they decline

-Type text after user is invited, before they accept
-Type long text. Make sure scroll bar works.
-Test the history of recently joined users
-Invite user with no one selected
-Invite user with multiple users selected
-Invite user with one user selected
-Invite user already invited for a room
-Invite user already declined from a room
-Invite user who already has invite for a different room
-X out buddy list without closing room windows
-X out room windows, then get a message from that room.
-Click inviteUser, X out room
-Click inviteUser, X out List.
-Click inviteUser

Accept User
-Two invites to the same room, different users, accept/decline combinations
-Two invites to different rooms, same user, accept/decline combinations
-X out invite.
-get invite, X out buddyList
-X out invite, then get reinvited by same and different users
-X out invite, then get different

Thread Saftey Arguments

ChatServer

The chat server is threadsafe because the only fields that are ever accesed by multiple threads are synchronized hashmaps (which are threadsafe data objects). Also, there are no race conditions because any pair of methods can be called concurrently. This is because almost all methods just have one line which returns or does some operation on a synchronized hashmap, so there are no operations to interleave. The only method that has multiple lines that could be interleaved is createRoom. If a removeRoom call were interleaved between the roomList.put line and the return room line the roomList would not have all the rooms it should, which would violate the spec of the ChatServer. This will not happen though because the removeRoom is only called is by a room object on itself, and it is impossible for a room object to remove itself before it is given to a User (which only happens after createRoom finishes).

User

User objects only have threadsafe fields. Furthermore User objects have methods that can be divided into three basic categories.

1.) Methods that a User object calls on itself (these are methods that the client "calls", ie if the client sends a createRoom message the user object calls its own createRoom method). These methods can never inteleave with each other.
2.) Methods that other Users/Rooms/ChatServer call on this user. These methods can all be called concurrently because the only thing they do is call the sendToUser method, whose entire body is wrapped in a synchronized(out){}, so its as if it is never called concurrently.

So methods of type 1 never interleave with each other, methods of type 2 can interleave with anything, but thats ok because it just sends the user some message and then exists (doesn't affect any fields of this User object).

Room

Room objects are threadsafe because each method mutates completely different fields. The only methods that mutate the same field are join and leave, but these objects will never be called concurrently because if a user object is joining a room it cannot be leaving a room at the same time.