6.005 Project 3: GUI Chat First Deliverable

Group 17: David Wen, Olivia Bishop, Wesley Graybill

May 4, 2009

Team Contract

Goals:

Our goal is to complete the specifications in a timely manner and get an A on the project. If time permits, we may try for the award. We will encounter time obstacles, as it is the end of semester and we will all be very busy.

Meeting Norms:

Over the weekend, we can meet during the day, and during the week, we can meet in the evenings. We all have evening commitments, but around those we are mainly free. We will have meetings in Burton Conner, since we all live there and it is convenient. Before the second deadline, we will meet two or three more times, and for the third deadline we will have more individual work, but meet periodically also. It is okay to eat during meetings. We will rotate recording minutes.

Work Norms:

We anticipate it will take 20 hours a week to make this project successful. Work will be distributed evenly. When we have a design, we will distribute coding assignments. In future meeting minutes, we will record who is designed to what task. If someone does not follow through on a commitment, we will send a strongly worded email to them. We agree that our code should be clear and commented, and we don't expect difference opinions on the quality of work. To deal with different work habits, we will set smaller deadlines to have things done at the same time.

Decision Making:

We will make decisions by discussion, and will for the most part want consensus.

Abstract Design

Definition of Conversation:

We decided to define our conversations as chat rooms rather than one-on-one conversations. All conversations are public, and any user can join a conversation by clicking on it. The list of conversations will be displayed as a list in a pane in the GUI. Users in a conversation see a list of other users in that conversation, but other users only see the number of users in a given conversation. Users may be involved in more than one conversation at a time. Also, the server does not keep track of old messages in the data structure for the conversation, so when a user joins a conversation, they only see messages received from that point on.

Design Decisions:

- We chose public conversations over private ones for simplicity reasons, since public conversations do not require invites or requests. We may add functionality for private conversations later.

- We are displaying a list of all conversations to the user instead of requiring the user to type the conversation name because it is easier for the user and goes along with the idea of keeping things public.

- We decided users not involved in a conversation can not view a list of users in the conversation for simplicity, and to limit the amount of data that must go over the network. This functionality may also be added later.

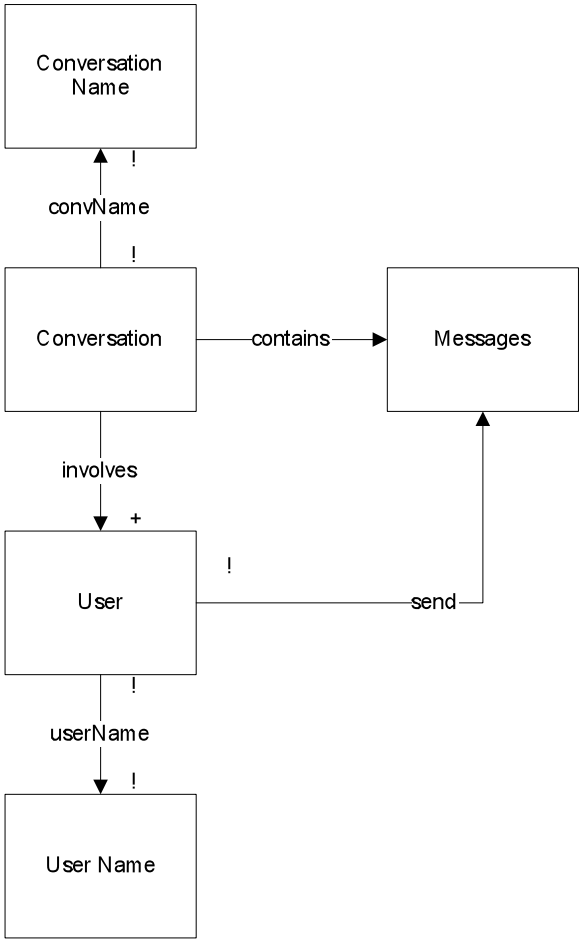
- Users can, however, see the number of users in a given conversation, because that information is useful to the user, especially without the above information.

- Users can be involved in more than one conversation because many people prefer to talk in several groups at once while IMing, and we feel this feature is useful.

- We choose not to keep track of old messages because we feel there is no need to. The server does not need access to them, and if a user does, it will remain in the GUI in the client. Also, this keeps users new to a conversation from viewing the part of the conversation that happened before they arrived. This is a privacy point, that since our system is almost completely public, it is helpful that users do know what they say will only be read by those already involved in the conversation.

- We will have users choose a username at login instead of having specific usernames and requiring passwords. This is for simplicity, and may be added later.

Object Model:



Protocol

We will define our protocol for communication between our IM client and server as follows. Below we list the types of messages that both the client and server will send along with a short description of the situation in which the message is sent.
A message will follow the grammar:

message ::= header : (< field >)*

Here, header will be restricted to any of the capitalized types below, and field refers to the information relevant to a message with that header. The number of fields is specific to the message type as described below.

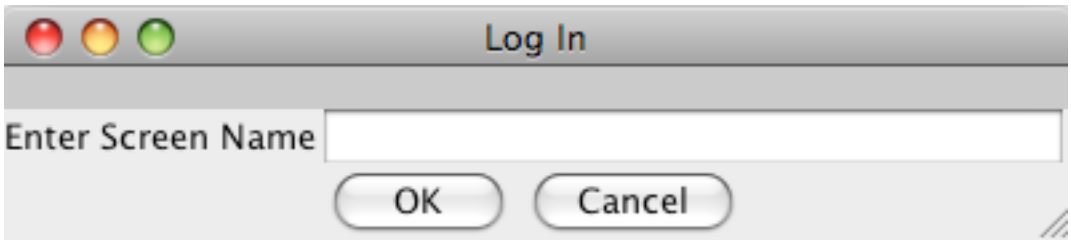
Client Messages

LOGIN: <username> -- at login, the client sends a message with the username request
NEWCONV: <convname> -- sent to server when the client requests to create a new conversation with the given name
JOINCONV: <convname> -- sent to server when the client requests to join the conversation of the given name
LEAVECONV: <convname> -- sent to server when the client leaves a conversation
MESSAGE: <convname> <message> -- sent to server when the client sends a message in the given conversation

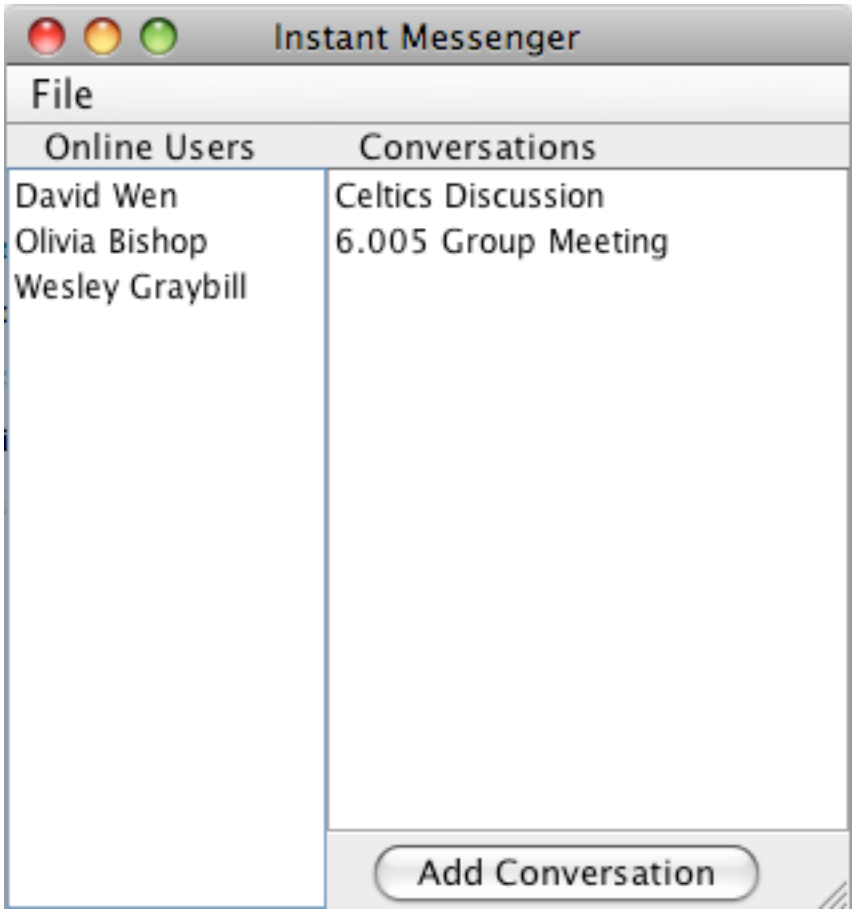
Server Messages

LOGINTAKEN: -- after a LOGIN message is received, if the requested login is already taken, send notification message back to client
LOGINCONFIRM: <list of users> -- after a LOGIN message is received, if login is unique, send notification back to client requesting login; includes a list of users currently logged in so that the client may populate its user list
NEWUSER: <username> -- when a new client logs in, this is sent to all clients notifying them of the new client with the given username
LOGOUT: <username> -- when a new client logs out, this is sent to all clients notifying them of the event
CONVTAKEN: <convname> -- when a client requests to create a new conversation, this message is sent in reply if the given conversation name is already taken
CONVCONFIRM: <convname> -- when a client requests to create a new conversation, this message is sent in reply when the conversation is created
JOINCONVCONFIRM: <convname> <list of users> -- sent to a client when the server has added the client to the given conversation; includes the list of users currently in the conversation so that the client may populate its list of users in the conversation
JOINCONVDENY: <convname> -- sent to a client to notify them that the client cannot join the conversation; this may be used, for example, in the event that a user tries to join a conversation, but in the process the conversation has been deleted
USERJOINEDCONV: <convname> <username> -- sent to all clients in the given conversation to notify them that a client with the given username has joined
USERLEFTCONV: <convname> <username> -- sent to all clients in the given conversation to notify them that a client with the given username has left
MESSAGE: <convname> <username> <message> -- sent to all users involved in the given conversation; includes the username of the user who sent the message and the actual message
CONVADDED: <convname> -- sent to all clients to notify them that a conversation with the given name has been created
CONVDELETED: <convname> -- sent to all clients to notify them that the conversation with the given name has been deleted

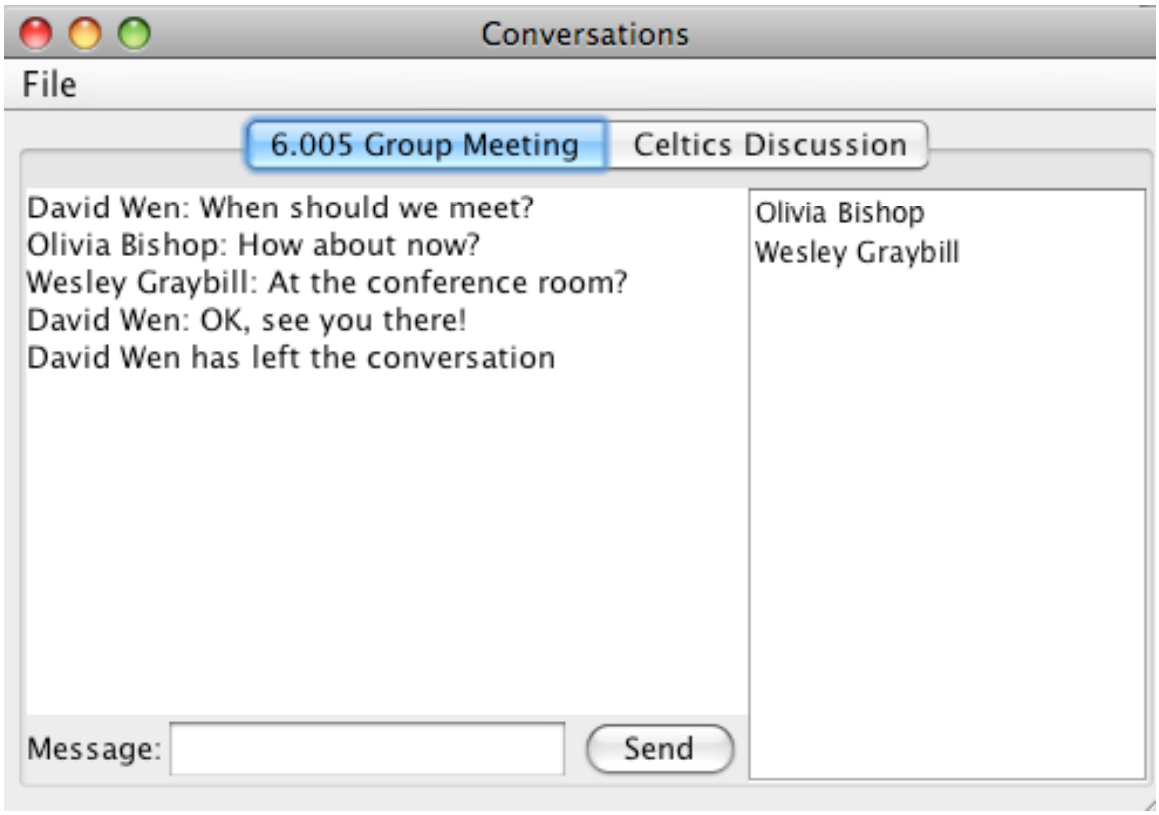
Usability Design



Log In Dialog – The user enters his screen name in the text field and clicks OK to log into the instant messenger. There is a text field on top of the window that notifies the user if the screen name is already taken or if it is an invalid screen name.



Main Window – On the left side of the window, the user sees a list of other online users' screen names. On the right, there is a list of conversations that the user can join by double clicking on the name. If the user wants to add a conversation, the user clicks on the Add Conversation button and selects a name for the new conversation.



Conversations Window – The Conversations Window opens upon joining the first conversation. Every subsequent conversation joined is added to this window as an additional conversation tab. In each conversation tab, there is a text area that contains all the messages sent by users and other important notifications, such as users joining or leaving. Users type messages at the bottom of the text area and click send to add to the conversation. To the right of the text area, there is a list of users who are currently in the conversation. Leaving conversations can be done through the menu bar under File.