Project 2: GUICHAT

Design Milestone 1

Chau Vu, Jan Rodriguez, Gabriel Frattallone

Conversation design:

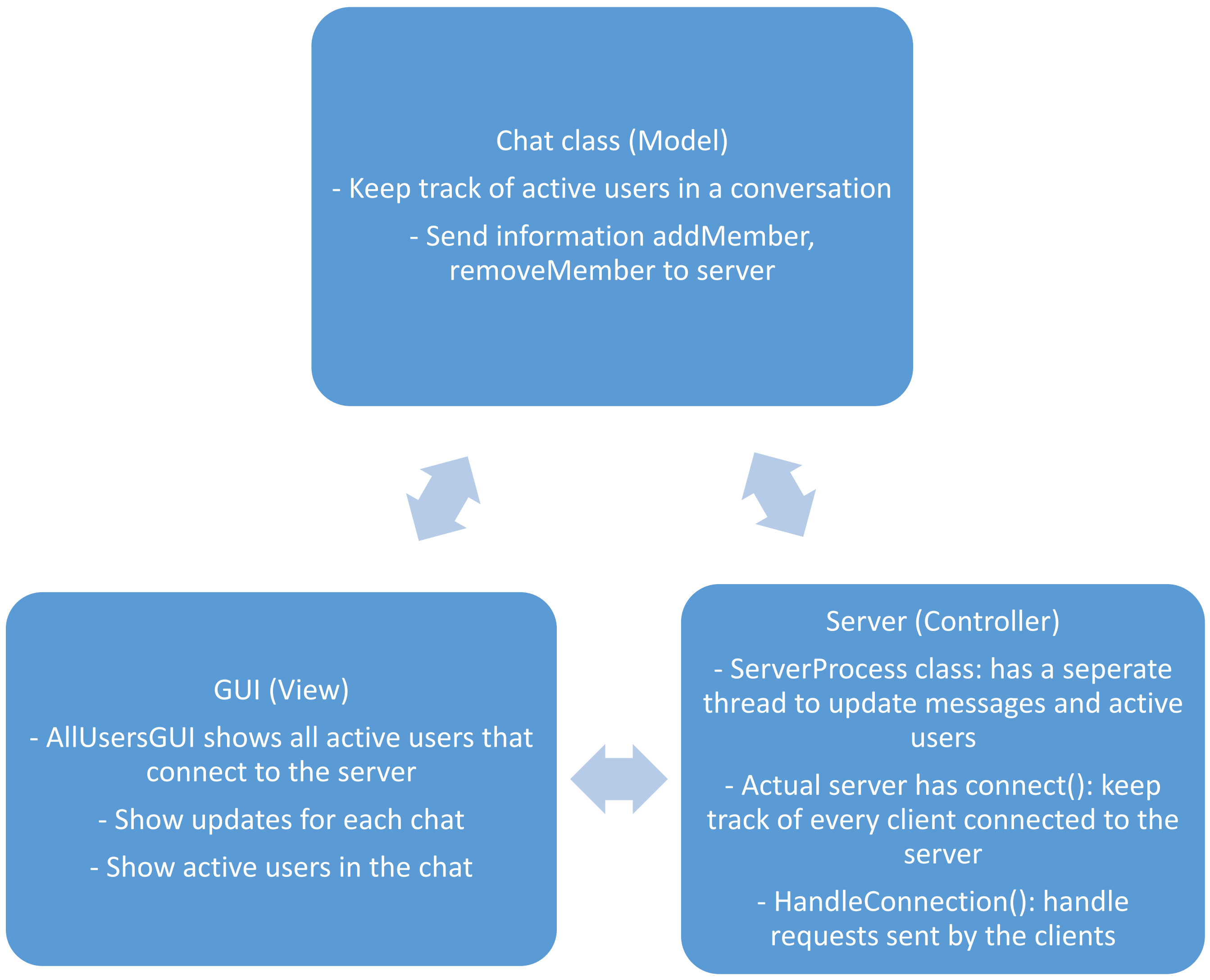
Login process

- Once users connect to the server, they select a username
- Server checks if the username is valid and not being used
- AllUsersGUI will pop up where users can see all active users and create conversations.

Defining a conversation

- Click on a Start Conversation button on AllUsersGUI: a new conversation appears where you can type in friends' names and invite them to the conversation. You have to be invited, cannot join a conversation otherwise.
- When get invited, you must go to the conversation that is popped up, if you dont want to be in the conversation, just close the window, and leave as if you are finishing ones.
- The conversation window will show a textbox that shows messages, a textbox that inputs your messages, an invite buttons and textbox field where you can type in usernames of you friends to invite more people to the conversation. Also there is a text field that includes names of users who are currently in the conversation window.
- The conversation window is disconnected when no user is using it, i.e all users left the conversation.

Model-View-Controller


Chat class (Model)
- Keep track of active users in a conversation
- Send information addMember, removeMember to server
GUI (View)
- AllUsersGUI shows all active users that connect to the server
- Show updates for each chat
- Show active users in the chat
Server (Controller)
- ServerProcess class: has a seperate thread to update messages and active users
- Actual server has connect(): keep track of every client connected to the server
- HandleConnection(): handle requests sent by the clients

Server Class:

-Manages client connections
-Keeps track of connected clients and active conversations
-Methods:
-connect() /**listens for new clients and generates a thread with their username . Ask for username, ensure the username is not being used**/
-handleConnection() /**manages client connections and client activity**/

ServerProcess Class:

- Implements Runnable
- Has Blocking Queue with pending operations
- Sends messages, disconnects users, and invites users to a conversation
- Methods:
    - Run() /** listen to the blockingqueue **/
    - updateChat(Chat, User, Action) /**applies action initiated by user to designated chat**/
    - addPublicChat(), newPublicChat() /** add a new public chat **/
    - addPrivateChat(), newPrivateChat() /** add a new private chat **/
    - join() /** let a user be added to a chat **/
    - disconnect() /** client disconnects from the server **/
    - leaveConversation() /** client disconnects from the chat **/
    - invite() /** client invites new users to the chat **/
    - sendMessage() /** clients communicate **/
    - getpublicChats() /** return list of public chats **/
    - getCreator() /** get username of whom created a public chat **/
    - notMember() /** check if one is a member of a chat **/

Client Class:

-Implements Runnable
-Stores username and socket
-Methods:
- run() /**calls server method handleConnection()**/
- newInvite() /** invite other users for chat **/
- newMessage() /** post message in chat GUI **/
- newChat() /** open conversation GUI for chat **/
- removeChat()
- updateChatMembers() /** call conversation GUI to update current users **/
- getUsername()
- setUsers() /** update users displayed in AllUsersGUI window of client **/
- setChatRooms() /** update list of public chat rooms in AllUsersGUI windows **/
- getChatMap()
- getChatsUpdate() /** call AllUsersGUI window to update list of public chats **/

- getProcessor()
- getSocket()

Chat Class:

-Keeps track of active members
-Has unique ID
-Methods:

- addMember()
- removeMember()
- getMembers()
- getID() /** each chat is assigned a unique id **/

Conversation Class:

-GUI representation for conversation
-Listeners:
  -Send Message
  -Invite User
  -Disconnect

AllUsersGUI Class:

-GUI representation for connected users
-Listeners:

- Start Conversation
- Join public Conversations
- Disconnect

Client-server protocol

```
MESSAGE         :== ( INVITE | INITIALIZE | POST | DISCONNECT_SERV | DISCONNECT_CONV )
INVITE          :== "invite" SPACE USERNAME
INITIALIZE      :== "new”
POST            :== "post” SPACE TEXT
DISCONNECT_SERV :== "disconnect"
DISCONNECT_CONV :== "leave"
USERNAME        :== TEXT
SPACE           :== " "
TEXT            :== “.”+
```

Server to Client Protocol

MESSAGE   :== (INITIALIZE | POST | DISCONNECT)

INITIALIZE   :== ("newPublic” (SPACE USERNAME)* SPACE USERNAME | "newPrivate” (SPACE USERNAME)* SPACE USERNAME)

POST   :== "posted” SPACE USERNAME SPACE TEXT

DISCONNECT   :== "disconnected" SPACE USERNAME

USERNAME   :== TEXT

SPACE   :== " "

TEXT   :== “.”+