CP 252, Final Project
Gina MacIntyre (Client)
Chris Logsdon (Server)
Zack Wheeler (Database/Documentation)

Protocol

When the client sends messages to the server (or vice-versa), a code will be attached to the beginning of the message. The following outline shows what each code represents, the value(s) expected with each code, and what will be returned.

Client sent to Server

- CS_FAILURE
  - Generic response. Not incredibly important.
  - Value: None.
  - Server returns: None.
- CS_SUCCESS
  - Generic response. Not incredibly important.
  - Value: None.
  - Server returns: None.
- CS_KEEPALIVE
  - For 15 seconds, the server checks every tick for this message from the client.
    - If a keep-alive is received
      - Server restarts its timer.
      - Server keepalive thread sleeps for 3 seconds.
      - Repeat from the top.
    - If a keep-alive is <u>not</u> received from the client after 15 seconds of listening, the user is disconnected.
  - Value: None
  - Server returns:
    - SC_KEEPALIVE
      - Server is acknowledging the keep-alive request.
- CS_USERCONNECT
  - A request from client to connect to the server.
  - Value: None
  - Server returns:
    - SC_USERCONNECT_FAIL
      - User was not connected to server
    - SC_USERCONNECT_SUCCESS
      - User was successfully connected to server
- CS_USERREGISTER
  - A request from client to create a new user account.
  - Value 1: Username (16 bytes)

    - Value 2: Password (16 bytes)
    - Server returns:
        - SC_USERREGISTER_FAIL
            - User account was not created.
        - SC_USERREGISTER_SUCCESS
            - User account was created successfully.
- CS_USERLOGIN
    - Client request to log into server with a registered user account. This request is not valid if the client is not first connected to the server.
    - Value 1: Username (16 bytes)
    - Value 2: Password (16 bytes)
    - Server returns:
        - SC_USERLOGIN_FAIL
            - User was not logged into server
        - SC_USERLOGIN_SUCCESS
            - User was successfully logged into the server.
- CS_USERLOGOUT
    - Client request to log out of the server. This is meant for immediate log out recognition with the server. The client may still log out and/or disconnect without waiting for a SC_USERLOGOUT_SUCCESS message. If this happens, the user connection will eventually time out on the server.
    - Value: None
    - Server returns:
        - SC_USERLOGOUT_FAIL
            - User was not logged out.
        - SC_USERLOGOUT_SUCCESS
            - User was successfully logged out.
- CS_ROOMCREATE
    - Client request to create a new private chat room.
    - Value 1: Chat room name (16 bytes)
    - Server returns:
        - SC_ROOMCREATE_FAIL
            - Room was not created.
        - SC_ROOMCREATE_SUCCESS
            - Room was created successfully.
- CS_ADDMEMBER
    - Client request to give a specific user access to a specific room.
    - Value 1: Chat room name (16 bytes)
    - Value 2: Number of usernames to follow
    - Value 3+: Username to add (16 bytes)
    - Server returns:
        - SC_ADDMEMBER_FAIL
            - Users were not added as a members of the room.

        - SC_ADDMEMBER_SUCCESS
            - Users were successfully added as members of the room.
- CS_REMOVEMEMBER
    - Client request to revoke access of a specific user to a specific room.
    - Value 1: Chat room name (16 bytes)
    - Value 2: Number of usernames to follow
    - Value 3+: Username to remove (16 bytes)
    - Server returns:
        - SC_REMOVEMEMBER_FAIL
            - Users' access was not removed as a members of the room.
        - SC_REMOVEMAMBER_SUCCESS
            - Users were successfully removed as members of the room.
- CS_USERTXT
    - Client text message meant to be sent out to all users in the room.
    - Value 1:
        - 0x00: Message meant for the public room
        - 0x01: Message meant for a private room
    - Value 2: The text message
    - Server returns:
        - SC_FAILURE
            - An error occurred.
        - SC_SUCCESS
            - No errors occurred.
- CS_TOKENREQUEST
    - Client request to be given the talk token for the room they are in. This message is ignored if the user is not in a private chat room.
    - Value: None
    - Server returns:
        - SC_TOKENREQUEST_FAIL
            - The user was not give the talk token.
        - SC_TOKENREQUEST_SUCCESS
            - The user now has the talk token for the private room they are in.
- CS_TOKENHOLD
    - Client request to continue holding the talk token. This message is ignored by the server if the user did not already have the token, or if the user is not in a private room.
    - Value: None
    - Server returns:
        - SC_TOKENHOLD_FAIL
            - Failed to maintain user's possession of the token. The token is now revoked from the user.
        - SC_TOKENHOLD_SUCCESS
            - The user continues to hold the talk token for the private room they

are in.

- CS_TOKENRELEASE
    - Client telling the server that it is releasing control of the talk token.
    - Value: None
    - Server returns: None.
- CS_USERLISTREQUEST
    - Client request for usernames.
    - Value 1:
        - 0x00: All users in database
        - 0x01: Users of public room
        - 0x02: Users of private room
        - 0x03: All users in the database that are not members of the private room the requester is currently in. Nothing is returned if the requesting user is not the owner of the room.
        - 0x04: All users in the database that are members of the private room the requester is currently in. Nothing is returned if the requesting user is not the owner of the room.
    - Server returns:
        - SC_USERLIST
            - List of users
- CS_AUDIOSETUP
    - A packet containing information on the audio data they are about to begin sending. This message will be relayed to all users in the same room as the user sender.
    - Value: TBD
    - Server returns:
        - SC_AUDIOSETUP_FAIL
            - The server was unable to receive and/or disperse the audio setup packet.
        - SC_AUDIOSETUP_SUCCESS
            - The audio setup packet was successfully received and dispersed to users in the room.
- CS_AUDIOCHUNK
    - A packet containing audio data from the client's recorded microphone input. This message will be dispersed to all users who are in the same room of the user sender.
    - Value: TBD
    - Server returns:
        - SC_FAILURE
            - An error occurred.
        - SC_SUCCESS
            - No errors occurred.
- CS_USERJOINREQUEST
    - Client request to join a specific room.

    - Value 1: Chatroom name
    - Server returns:
        - SC_USERJOINREQUEST_FAIL
            - The user did not join the room.
        - SC_USERJOINREQUEST_SUCCESS
            - The user successfully joined the room.
- CS_USERLEAVEREQUEST
    - Client request to leave the private room they are in. This message is ignored if the user is not in a private room.
    - Value: None
    - Server returns:
        - SC_USERLEAVEREQUEST_FAIL
            - The user did not leave the room.
        - SC_USERLEAVEREQUEST_SUCCESS
            - The user successfully left the room.

Server sent to Client

The following messages are ones that are not responses to client messages. To see the rest of the SC_* messages, look at the “Server returns” section of each message thec lient sends to the server (above).

- SC_KEEPALIVE
    - When the server receives a keep alive message from a client, it sends this message back as an acknowledgement. For 15 seconds, the client checks every tick for this message.
        - If the client receives it
            - Client restarts its timer.
            - Client sleeps for 3 seconds.
            - Repeat from the top.
        - If a keep-alive is <u>not</u> received from the server after 15 seconds of listening, consider the server to be disconnected.
    - Value: None
    - Client returns: None
- SC_USERJOIN
    - When a user joins a private room, the server sends this message out to all users in the room, to let them know that the user has joined the room.
    - Value1: Username
    - Client returns: None
- SC_USERLEAVE
    - When a user leaves a private room, the server sends this message out to all users in the room, to let them know that the user has left the room.
    - Value 1: Username

  - Client returns: None.
- SC_USERTXT
  - When the server receives a CS_USERTXT message, it then attaches the username of the sender, then a colon and a space to the beginning of the text message. It then sends this message out to all clients in the room specified in the CS_USERTXT message.
  - Value 1: Text message
  - Client returns: None
- SC_USERLIST
  - This message is a response to a CS_USERLIST message. It is the number of usernames being sent over, and then the usernames
  - Value 1: Number of usernames being sent
  - Value 2+: The usernames
  - Client returns: None
- SC_AUDIOBEGIN
  - If the server grants a token request to a client, it will then send this audio begin message to every client that is logged into the private room that the token requester is in. This lets those clients know that streaming audio is eminent.
  - Value: None
  - Client returns: None
- SC_AUDIOEND
  - When a client releases the talk token, the server then sends this audio end message out to every client that is logged into the private room that the token requester is in. This lets those clients know that streaming audio is ceasing.
  - Value: None
  - Client returns: None
- SC_TOKENREQUEST_SUCCESS
  - When the server is able to give the talk token to a client, this message is sent in return. It contains a list of all the IPs the client should send audio to.
  - Value 1: The number of IP strings being sent.
  - Value 2+: The IP strings.
- SC_ADDMEMBER_REQUEST
  - When the server grants a request to add members to the room, this message is returned, and it includes the number of users that were successfully added to the room.
  - Value 1: The number of users successfully added to the room.
- SC_REMOVEMEMBER_REQUEST
  - When the server grants a request to remove members from the room, this message is returned, and it includes the number of users that were successfully removed from the room.
  - Value 1: The number of users successfully removed from the room.

Server

Chris Logsdon

Interface Overview:

When the server application is run, a dialog window will pop up immediately, prompting the user to set up the server IP Address and port. This window can be closed and re-accessed through the Server -> Manage Connection menu. However, the controls in the rest of the application will not be enabled until the server is started.

Once the server is set up and started, there will be two tabs for the user to navigate. One will be a "Users" tab, where the user can view information about all users that have registered an account. The other tab will be a "Chat Rooms" tab, where the user can view information about all registered chat rooms.

Users Tab

In the "Users" tab, the user will see a list of the usernames of all client users who have registered an account. Their names will appear in a listbox whether or not they are currently logged in. If a client user is logged in, their name will appear in black. If the client user is *not* logged in, their name will appear in gray. If the user clicks on a client's name, they can view, in the same tab, their username, current IP Address (if they're connected), chat rooms they have access to, and which rooms they are currently in. The chat room list will be displayed in a listbox. If the client user is currently in a chat room, that room's name will appear in black. If they are not currently in a room, the name will appear in gray. All of this updates as events happen.

Chat Rooms Tab

In the "Chat Rooms" tab, the user can see a list of all chat rooms that exist in the database. When the user clicks on a room in the list, they can view details about the room in the same tab. Details they can view will be the room name, the owner user, and a list of allowed/connected users. In the list of allowed/connected users, a user's name will appear in gray if they are allowed in the room but are not connected. Their name will appear in black if they are currently in the chat room. All of this updates as events happen.

GUI:

- Server Setup dialog window
    - IP Address and port number to host on (text boxes)
    - Start Server and Stop Server buttons
    - Get IP button (detects computer's IP Address, or localhost if it's unable)
- Tab View
    - Users tab
        - Listbox of usernames + IP Address
            - Example: clogsdon589 (127.0.0.1)
        - Click on a user in the listbox
            - Information about the user is displayed in a group box

- Username and IP Address (labels)
- Chat rooms the user has access to (list box).
  - If the user is in a chat room, that room’s name will be shown in black. If they are not in the room, the name will be shown in gray.

- Chat Rooms tab
  - Listbox of chat room names
  - Click on a chat room in the list box
    - Room name
    - User who owns the room
    - Listbox of allowed members
      - Members connected appear in black
      - Members not connected appear in grey

Test Cases

User connects with correct username and password

[ ] User sees client with the public chat room, a list of users online, a list of rooms they can join, and some options.

[ ] The user can then join a room that they have access to.

[ ] Once they do, other users in the room will see they have joined, and the user will be presented with a the private room window.

User Creates a Private Room

[ ] User is presented with a box to name their room.

[ ] If failed, it will let the user know with a message box.

[ ] If succeeded, it will let the user know with a message box.

User Adds Members to a private room

[ ] User enters their private room.

[ ] User enters the add members menu through the file menu.

[ ] User uses checkboxes to choose which members they wants to add.

[ ] If not owner, nothing will show up.

[ ] If failed, it will let the user know with a message box.

[ ] If succeeded, it will let the user know with a message box.

Users Removes Members to a private room

[ ] User enters their private room.

[ ] User enters the remove members menu through the file menu.

[ ] User uses checkboxes to choose which members they wants to remove.

[ ] If not owner, nothing will show up.

[ ] If failed, it will let the user know with a message box.

[ ] If succeeded, it will let the user know with a message box.

User Logs Out

[ ] User navigates to logout in the file menu/

[ ] Program quits.

[ ] User can also just close the public room window to log out.

User wants to audio chat

- [ ] User enters private room.
- [ ] user tries to get the token using the key assigned
  - [ ] If someone else already has the token, they will not be able to obtain the token.
  - [ ] Once user has the token, they will be able to voice chat until the token is released.

If a user loses connection to the server

- [ ] A message box appears after 15 seconds notifying the user that they have lost connection.
- [ ] The program closes once the message box is closed.

Database

Overview

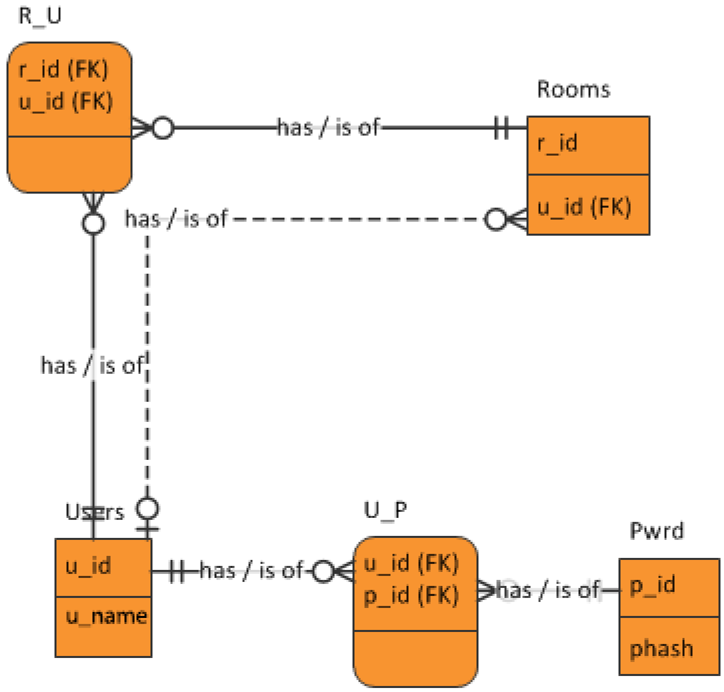
The database will comprised of two parts. The actual MySQL database, and the C# interfacing class for it. C# class will contain functions for accessing and manipulating data in the SQL database.

Such as:

Functions	parameters	returns
CreateUser	string user, pword	returns true if success, false if failed
CheckPassword	string user, pword	returns 1 is success, 0 if failed, -1 if error
CheckUserExists	string user	returns 1 is success, 0 if failed, -1 if error
GetUserList		returns a list of users in db
CreateRoom	string roomName, string owner	returns the new chatroom
CreateRoom	string roomName, string owner, string[] Users	returns the new chatroom with a list of users inside
HasAccessToRoom	string chatroom, string user	returns 1 is access, 0 if none, -1 if error
RoomAddUser	string chatroom, string user	returns 1 is success, 0 if failed, -1 if error
RoomRemoveUser	string chatroom, string user	returns 1 is success, 0 if failed, -1 if error
GetChatroomList		returns a list of chatrooms
GetUserChatroomAccessList	string userName	returns a list of chatrooms a user has access too
GetChatroomsOwned	string owner	returns a list of chatrooms a user owns
GetOwnerOfChatroom	string roomName	returns the owner of a room

The server will be the only one to talk directly to this C# class. The client will send requests to the server when the database is needed, and the server will send the data to the client.

Structure



R_U: the list of rooms users have access to

Rooms: the list of rooms and their owners.
Users: list of users
U_P: junction table for users and passwords
Pwrd: list of passwords

Client

Gina MacIntyre

Overview:

When user launches the application the user will be at a LogIn page. They will connect to the server and then they will enter a username and password and login to the server. Once they are logged in they will see a list of rooms that are available to enter. Once they enter into a room there will be a list of other users in the room. They will be able to chat via text or voice. One person can talk using voice at a time. When there is no one talking the user can press and hold a user defined key to take the token and talk. The user will keep the token until they let go of the key. All of this will be updated in a live fashion.

Structure:

- LogIn Page
  - User name Field
  - Password Field
  - Connect button
  - Reset Button
- Connected - List of Chat rooms
  - List Box of Chat rooms
    - with number of users currently in each room
    - with indication if room is public or private
      - if room is private it will require a password to enter
  - List Box of users
    - with location if in a room or not in a room
  - Create new chat room
    - any user has authority to create new chat room
    - chat room name and public/private status required
      - if private, password required
  - Option to request private chat with any user currently connected
  - Rich text Box for chat in the public room
  - File Menu for creating new rooms, and manually updating rooms/users
  - Window menu for customization of your client
  - Log out Button
- Chat room Page
  - Rich Text Box for all text input by users
    - read-only text box

Rich text Box for user to send text to room
List box for list of users and token to show who is talking
    if no icon is shown for who is talking anyone can talk
    when someone is talking a speaker icon appears next to their name
Option to request private chat with any user currently in room
File Menu to manage users access to room (only works if the user is the owner)
File Menu also allows editing of the push to talk button