Documentation for the final project

Instant Messaging

Content

User documentation

Program information

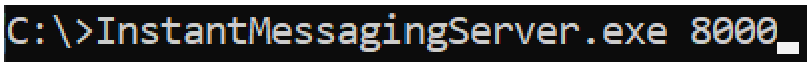
The program functions as an instant messaging application that allows users to send messages to each other. It is split into two parts – server and client.

To create a chatroom, you need to run the server part of the application. It does not have a GUI and functions only through the console. It logs important events such as users joining, leaving, connections failing and also logs all messages (for monitoring purposes, etc.). It sends out announcements for some of the aforementioned events to notify active users of what is happening in the chatroom.

Users only need the client part of the application. It has a functioning GUI. They can connect to the chatroom by inserting the correct information and logging in with a username. They may leave whenever they want and connect to other chatrooms without affecting the servers' functionality.

Interacting with the program

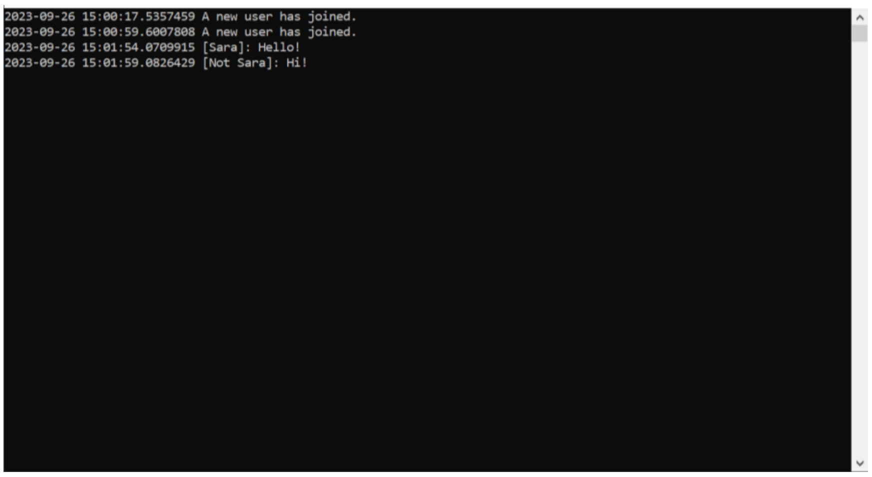
As was mentioned before, to create a chatroom, you need to have a functioning server. Setting it up is quite easy. You can run the server application from your console. One argument is required and that is the port number to which users should connect. It should look something like this:



```
C:\>InstantMessagingServer.exe 8000_
```

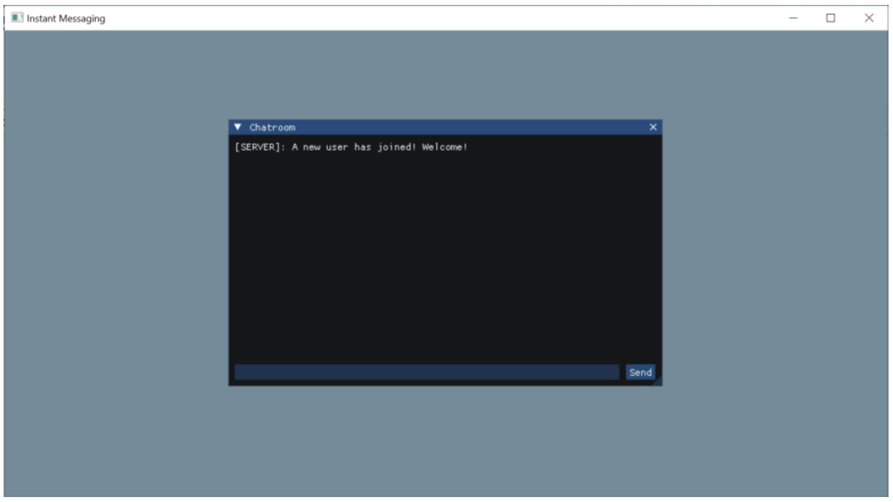
If none or too many arguments are provided, the program will not create a server and instead will show the user an error message.

Once it is successfully set up, users can connect to it and chat. Logs are displayed in the console as well. They consist of the time and date during which the action happened and the event itself. If it is a message instead of an event, it will contain the username and message.



```
2023-09-26 15:00:17.5357459 A new user has joined.
2023-09-26 15:00:59.6007808 A new user has joined.
2023-09-26 15:01:54.0709915 [Sara]: Hello!
2023-09-26 15:01:59.0826429 [Not Sara]: Hi!
```

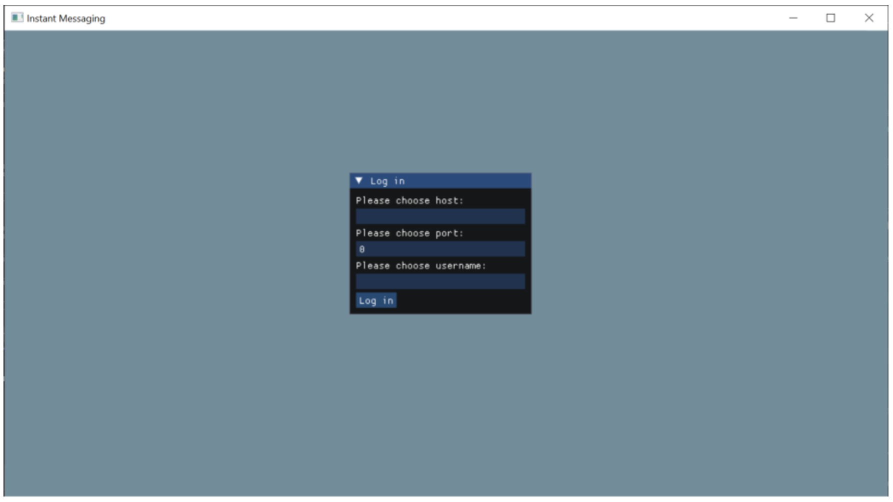
You can't see the announcements it sends out from the server's point of view. You can only see them in the chatroom as a user. They show up like any other message would, with the username being SERVER.



```
[SERVER]: A new user has joined! Welcome!
```

Nothing else is needed on the server's side. As long as it is running, users can join. If the server is closed, all users will be disconnected, and an error message will let them know what happened.

As a user that wants to join a chatroom, you should run the client application. It does not require any arguments, as everything is handled by the GUI. Upon starting the program, you will be met with the log in screen.

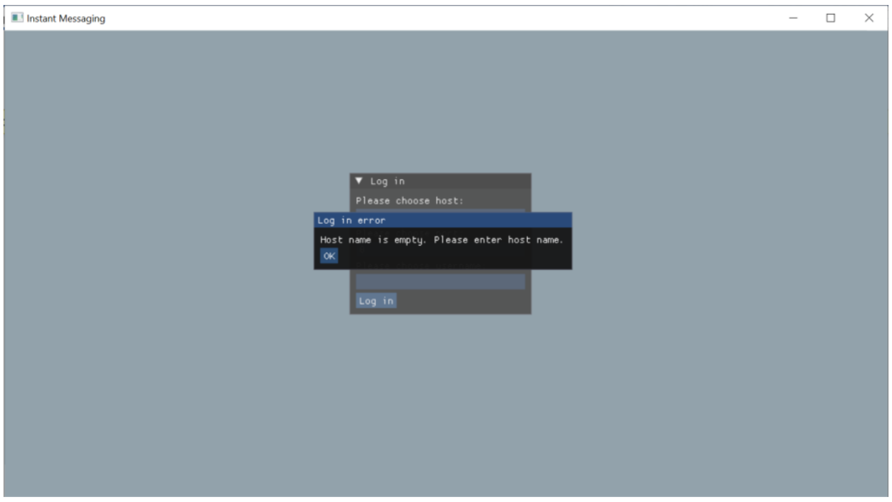


```
Please choose host:

Please choose port:
0
Please choose username:
```

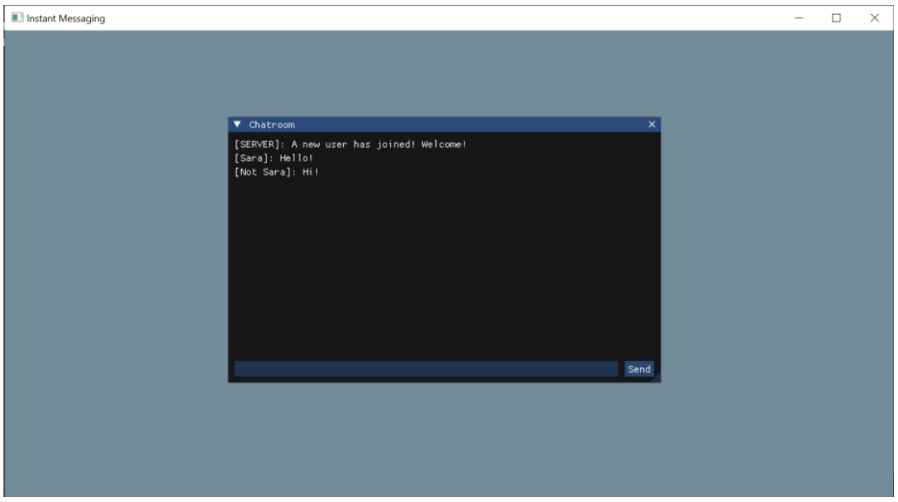
You need to fill out the required information to join a chatroom (host, port, username), otherwise the program won't even attempt connecting. You can try to log in by clicking on the Log in button or pressing enter after writing your username. The username is limited to 20 characters.

If you leave the host or username blank, it will show you a log in error, which contains information about what is missing. If you put a blatantly wrong port number (for example a negative number), it will automatically switch it back to the last plausible option.



```
Host name is empty. Please enter host name.
```

Similarly, you will get an error if the format in which you wrote the host address is wrong. If it is correct, but no such server exists/it is the wrong port, the program will attempt to connect and eventually fail, once again giving you an error. If nothing goes wrong, a chatroom window will appear and the log in window will disappear.



```
[SERVER]: A new user has joined! Welcome!
[Sara]: Hello!
[Not Sara]: Hi!
```

You can then send messages by clicking on the Send button or pressing enter. The maximum number of characters in one message is 1000. As more messages are sent, the chat becomes scrollable. However, only the most recent 100 messages will be displayed. Older messages will be lost and only visible in the server log.

Development documentation

Specification summary

A simple instant messaging application made possible through a server-client setup. It should allow a server to create a chatroom and monitor it as well as closing the chatroom upon closure. Users should be able to connect to an active server and send messages to each other through it. They should be able to disconnect and establish a new connection easily.

Program structure

The program is split into two parts (client and server). The client is then split into several files based on a logical structure.

Both parts include the message header. It contains all that is necessary to transfer a message, namely the username of the sender and the content. It also has functions related to byte orders, as we need to keep endianness in mind while sending messages.

The server files have three classes. One class is the server, which listens for new connections and adds them to the chatroom. The room class functions as the chatroom, and thus has everything directly related to it. It contains adding and removing users as well as the logic for delivering messages (including announcements) and event logging. The last class is the session class, which is equivalent to a user (we only have one session per user). It takes care of all the user functions in regard to the server, such as handling the messages sent through the server.

The client files have two classes. One is the message handler, which contains functions related to messages which we have in our possession, whether that be saving incoming ones, queuing new ones to be sent, or rendering them. The other class is the client class, which handles the client connection as well as the sending and receiving of messages to/from the server.

The errors files have all the error handling mechanisms. That includes having the common errors saved along with the messages that should be displayed when they occur, and all the logic related to displaying the error popups.

The UI files contain classes for each window we have as well as an overall UI class. Each of the classes have a render method, which displays the corresponding UI, and additional functions needed for the right functionality of the window.

The main file is code copied from the ImGui example. It sets up the GUI library, the main window and the backends.

Possible extensions

One of the possible extensions that I thought of was allowing the user to join several chatrooms at the same time. I believe it would be more convenient and comfortable for the user if they had several chatrooms they'd like to be part of, as they'd have to choose just one right now.

A bigger extension would be adding a friend system to the program, where the user could add other users to their friend list. Perhaps they could see whether their friends are online or offline and even add their own status for others to see.

Conclusion

Learning how to work with the Boost library was harder than I expected, as it sometimes threw exceptions that were difficult to find and fix. Looking back at it, perhaps I should've chosen a different library for the networking part of my program. Perhaps it would've made it easier. On the other hand, ImGui was very nice to work with, so making the GUI for the program ended up being my favorite part of the journey.