**Group 4 - Communications**
Meeting Minutes
10/3/23 @ 3-4 PM

**Attendees**
Benjamin N.
Aaron N.
Jagjot N.
Kosta N.

- 3:05 meeting start
- 3:12 -3:24 discussed our exact requirements list
    - Fully fleshed out our top 10 requirements.
        - Made sure we included multi-threaded in our server description.
        - We made sure to include read receipts in our requirements.
        - Decided that synchronous and asynchronous chat was the same and had the server manage message syncing once a user comes online.
- 3:24 - 3:35
    - Discussed class candidates. Biggest consideration is that all data that can be kept server side is done so and the client is lightweight.
- 3:35 - 3:51
    - Discussed specific details about how classes are inherent by other classes and how each class satisfies certain requirements. All chats are groups. Direct messages are a two person group. There can be multiple groups with the same list of users with different message histories. Users can be added to groups after initial creation.
- 3:51 - 4:00
    - Schedule has roles divided up amongst the group. Planning to meet again Wednesday (10/4/23) via Discord.

10/10/23 @3-4pm
Attendees
Aaron N.
Jagjot N.
Kosta N.

- 3:03 meeting start
- 3:03 - 3:05
    - Decided on what part of the srs doc needed to be updated.
- 3:05 - 3:32
    - Analyzed UML class diagram. Added association with server and client. Added association with server and user.
    - Realized we needed GUI represented in UML

        - Added GUI to UML with tentative functions and data.
        - Realized user class shouldn't contain password since user object is sent to multiple clients and modified.
    - 3:32 - 4:00
        - Started going over the requirements list. Realized that we were missing key requirements.
        - Added more functional requirements starting with common requirements. Additional requirements were added to the other categories.
        - Added more non-functional requirements.
        - Discussed the next meeting date. Tentatively set for next Tuesday. 10/17

10/17/23 @3-4pm
Attendees
Aaron N.
Jagjot N.
Kosta N.
Benjamin N.

- 3:05 meeting start
    - Connected Kosta through discord
- 3:05 - 3:30
    - Worked on refining SRS document with section 3.2, 3.3
    - Submitted the updated documents to github and assignment
- 3:30 - 4:00
    - Went over the Message object and various candidates for message types such as ChatMessage, RoomListMessage, LoginMessage
    - Started to talk about how we wanted our data to be stored on the server in terms of file storage
    - Went over the server handler and going with handling one thread that can deal with many different uses
    - Discussed how we should have a main message data type and have the individual message types be based off the main message data type.
    - We have CreateRoomMessage, LeaveRoomMessage, AddUserToRoomMessage, RoomListMessage, LoginMessage all are going to inherit the base Message class in a “is-a” relationship.
    - Created the Authenticator data type.
    - Set time for next meeting to be next tuesday to meet 10/24

10/24/23 @3-4pm
Attendees
Aaron N.
Jagjot N.
Kosta N.
Benjamin N.

- 3:03 Meeting start
    - Realized that we needed parameters for our UML diagram that has our methods
    - Servers spins off a handler per client. Figuring out how threaded servers handle clients.
    - Continued discussion on whether authentication happens before the server spawns off new thread or after.
- 3:30 discussion of gui elements
        - Login dialogue appears ontop of main gui window. Main window should be disabled untill authentication is complete
- 3:40 Discussion of method parameters for our UML
- 3:50 authentication flow discussed
    - Client connects, server spins off thread with handle, client authenticates on thread. If successful, communication continues.
- 3:55 started to implement member functions for handler
- 4:03 over

10/31/23 @3-5pm
Attendees
Aaron N.
Jagjot N.
Kosta N.
Benjamin N.

- 3:02 Meeting Start
    - Discussed how message objects will be handled. Rather than having multiple child message object instantiations of the message object for each message type, the instantiations of the message object will have a Boolean array as flags to determine what message it.
- 3:20
    - Decided on how server and client will sync user data.
        - Decided that the way users will be updated on the client side about other users' status will be from the server containing an array of users and when an users status changes,
- 3:36
    - Worked on how data will be stored in files on the server side for data storage.
    - Credential file will contain one to one userID and password
    - Room file will contain a list of room ids indexed by user id.

    - Room files will contain associated room data in a header, a list of associated user ids and a list of serialized msg objects.
    - Maybe: login history file that has a list of users, sign in or out, then timestamp.
- 4:04
    - Decided that messages need input validation to not mess up storage
- 4:06
    - Discussed data synchronization across our server threads and main server.
- 4:15
    - Discussed changing our messages back to specific instantiations of child objects that requires casting.
- 4: 23
    - Began discussion on presentation requirements
    - Went back to figure out the design for logging classes.
    - Discussed the method of keeping user status updated for all users. Server would have a master list that can update user status and send it to the clients when
- 4:49 meeting end