2/11/2025

- Started unit tests using the unittest library
- Realized that sockets are streams and hence we need to append message length to the start of each client/server communication
    - Rewrote our client and server parsing code in response to this in order to make the code more stable, also updated in the design document linked in 2/7
- Fixed bugs related to not flushing out the "input" to a server request, which led to a particular request being tried several times in a row
- Started JSON implementation, relevant documentation: https://docs.python.org/3/library/json.html
    - Note that json implementation keeps the convention where the total number of characters in the message is appended to the front of the message (to ensure nothing got lost)
- Tried to write unit tests with both the server and client running but couldn't debut forcibly closed connections errors
    - Instead, wrote unit tests for each of the "data processing" functions within the server (ex. create_account) that manipulate the key data structure
- Realized I was using the wrong port for the unit tests, reverted back to server and client running the unit tests
    - Fixed a bug where creating an account logs you in automatically
- Added helpful print/logging statements in the code and deleted all of the random comments and print statements used during the process

2/10/2025, Session 2

- Pedro resolved issues in fitting server and ui together
- Changed the implementation of some of the wire protocols to be more clear
    - Ex. added messageId, the user the message is from, the timestamp to the read message return value
- Added new functionality that binds a particular socket to a user on login and unbinds them on logout
    - Needed to provide "real-time" updates when a logged in user receives a message

2/10/2025

- Worked on remaining functions in server
- Discussed best ways to search accounts, delete accounts, and delete messages with Pedro

    - Decided to do the actual searching on the client-side, so the server just needs to provide the client with a list of all accounts
    - Decided that because security was not the primary focus delete account/message/etc. would simply pass in username of the desired account via the socket, as opposed to binding a certain username to a certain socket on login or similar
- Finished server functionality on list accounts, send messages, read messages, delete messages, delete accounts according to the specification we wrote up on 2/7
    - Only wire protocol versions for now, json versions will come later
- Tried fitting frontend and server together, ran into some strange issues
    - Infinite requests to connect to server ← Pedro will to try resolve

2/7/2025

- Brainstormed with Pedro about the details of our implementation and wire protocol, the results of which are here: CS2620 Design Exercise 1
    - Will be using Python for this project
- Started implementing the server of the chat app with adapted code from lecture
- Decided to use a "match-case" (switch) statement for processing the connections
- Decided to write helper functions for each connection that accept already pre-processed data so that when we switch to the json format for the second part of the assignment we will only need to change the preprocessing code and not these functions
- Added a "logged in" field to our accounts structure
- Considered adding a lock to our accounts structure but decided not to, because requests are processed one at a time so we should never have two requests being processed simultaneously
    - IF we did multithreading locking would be needed
- Considered writing a helper file for the functions for each connection but since they all need to access the accounts data structure in the main file I decided not to
- Decided to make error codes 1 number for now, wrote comments explaining what each one meant
- Changed the helper functions to return a more generalized format instead of just pure error codes. Now they return [True/False, additional data]
    - The first element in the list True/False indicates if the call to the helper completed successfully without error
    - The second element in the list gives any additional data needed for return
        - For a success, this may be empty (no additional return info needed), a list (of accounts or messages), etc.
        - For a failure/error, this term is a string in the format "[errorcode]: [error message]"

        - By including both an error code and error message, we don't have to change the format of the string between the wire protocol and JSON parts - the json can take the whole string for more descriptiveness, while the wire protocol can just use the error code
- Finished create account, login, and logout in this session.