The client and server communicated via synchronous gRPC calls. Client information was persistently stored via files in the UserInfo directory. The file user_names.txt contained the usernames for all users, regardless if they were currently logged in. Each user had three designated files to contain their followers, who they were following, and their timeline.

When the server was initialized, I created a gRPC connection using the port passed in as a command line argument. After making the UserInfo directory, I waited for any incoming client connections. On the server side, I had two separate vectors to store users. The vector all_users_vect contained the usernames of all users that had been created. A second vector session_user_vect stored the usernames of all users that logged in during the current server session. When a new client tried to login, I checked that the user was not currently logged in. If the user was not currently logged in but the account had already been created, then it was not necessary to create the initial user files for them.

Most of the functions in the server involved unpacking the request from the client into individual arguments. Once information such as the username and type of request is known, then the server accessed or altered the file system to gain information about the client making the request. Once the database had been updated or the information had been gathered, a status was returned back to the client. Depending on the type of response received from the server, the client can either display information regarding an error or output the requested information to the console.

The timeline implementation was the most involved part of the project. On the client side, it was necessary to create an additional thread to read from the server while the main thread sent messages to the server. These messages contained the timestamp, post, and username of the user who made the post. Multithreading was not needed on the server side. The server gathered the username of the client using the server context metadata. Then the past 20 messages were gathered from the timeline and sent back to the client. Lastly, the server read posts from the client in a while loop and updated all users' timeline who were following the client. A map was needed to associate a username with the server reader writer. This ensured that messages were being sent to the correct client.