Group names: Prince Moyo(MYXPRI011)

Kholofelo Zondi(ZNDKHO003)

Tokelo Makoloane(MKLTOK002)

1. Introduction

Our chat application is designed to facilitate instant messaging through both TCP and UDP connections. The application includes features for user authentication, privacy settings, real-time communication, server interaction, peer-to-peer communication, and broadcast messaging.

2. Features and Functionality

User Registration:

Ensuring a secure environment is paramount in our chat application. To achieve this, users are required to go through a robust registration process where they create unique credentials. This not only protects user privacy but also establishes a foundation for a trustworthy communication platform.

Privacy Settings:

Recognizing the importance of user control over their information, our chat application incorporates comprehensive privacy settings. Users have the autonomy to customize visibility permissions, determining who can access them and send them messages. This feature adds a layer of personalization, empowering users to tailor their chat experience based on their comfort levels and privacy preferences.

Real-time Communication:

The heart of our application lies in its ability to facilitate real-time communication. Leveraging UDP for real-time media streams, users experience near-instantaneous transmission of messages. This ensures a seamless and dynamic conversation environment, mimicking face-to-face interactions as closely as possible. Real-time communication enhances the user experience, making our chat application a responsive and engaging platform.

Server Interaction:

Clients seamlessly interact with a central server to enhance the functionality of our chat application. This interaction allows users to query the server for dynamic lists of available clients, facilitating easy discovery of potential conversation partners. Additionally, clients obtain essential communication parameters from the server, ensuring a standardized and efficient setup for peer-to-peer connections. The server acts as a central hub, orchestrating the initial steps that lead to successful communication between clients.

Peer-to-Peer Communication:

For reliable and direct communication between users, our application employs TCP for signalling and initializing peer-to-peer connections. This approach ensures the establishment of robust and stable connections, enabling efficient data transfer. By utilizing TCP, we prioritize reliability, making certain that communication channels remain open and responsive throughout the entire interaction. Peer-to-peer communication is the backbone of our system, enabling direct and secure exchanges between users.

Broadcast Messaging:

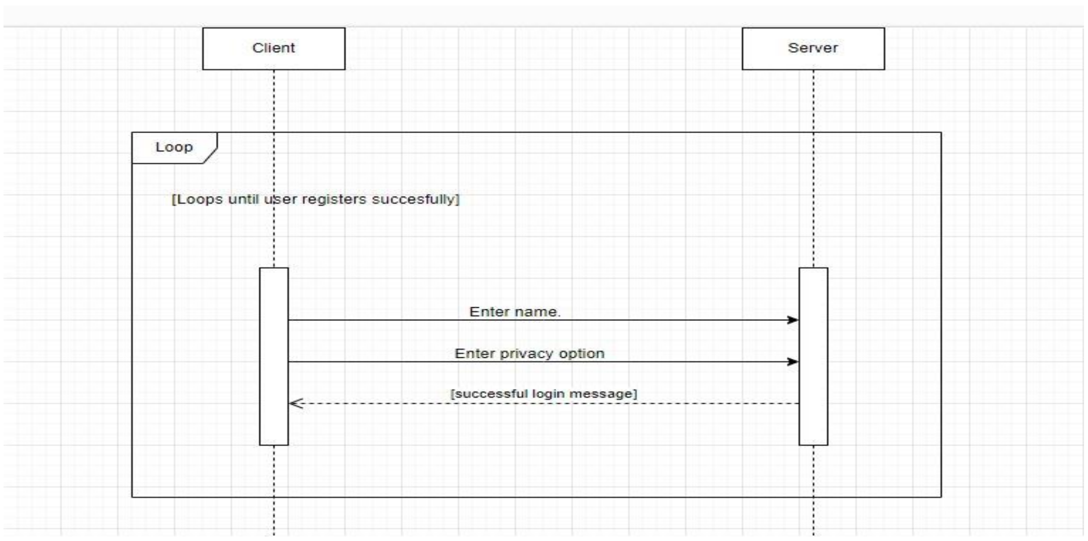
To foster efficient group communication, our chat application introduces a broadcast messaging feature. Users can send messages to all available and public clients simultaneously, promoting collaboration and collective engagement. Whether it's coordinating group activities or sharing important announcements, the broadcast messaging feature enhances the versatility of our application, making it a powerful tool for both one-on-one and group interactions.

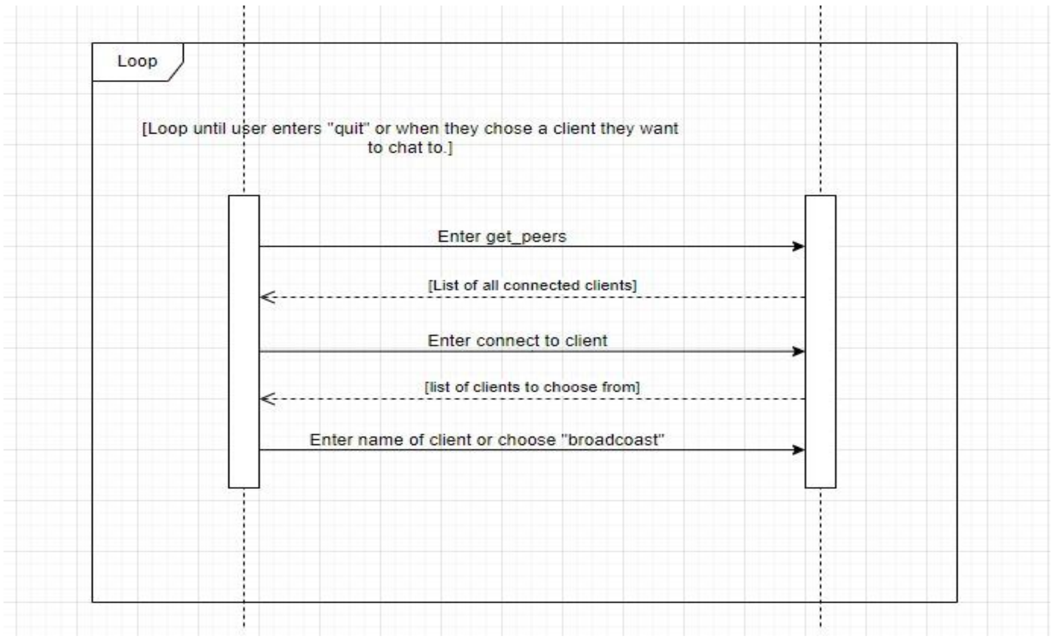
3. Client Implementation

The client-side implementation of our chat application is designed for simplicity and user-friendliness. Upon logging in, users provide their name and privacy preference, initiating a straightforward registration process. The "get_peers" method allows users to quickly retrieve a list of online clients, streamlining the process of finding conversation partners. The "connect to client" method facilitates one-on-one chats, presenting users with a list of available clients for easy selection. Real-time communication is established using UDP connections during these conversations. When sending messages, users specify the recipient's name for targeted communication, and for broadcasting, the "Broadcast" command efficiently reaches those users who have opted for public visibility. The client-side implementation prioritizes an intuitive user experience, making it easy for individuals to register, connect with others, and engage in seamless conversations.

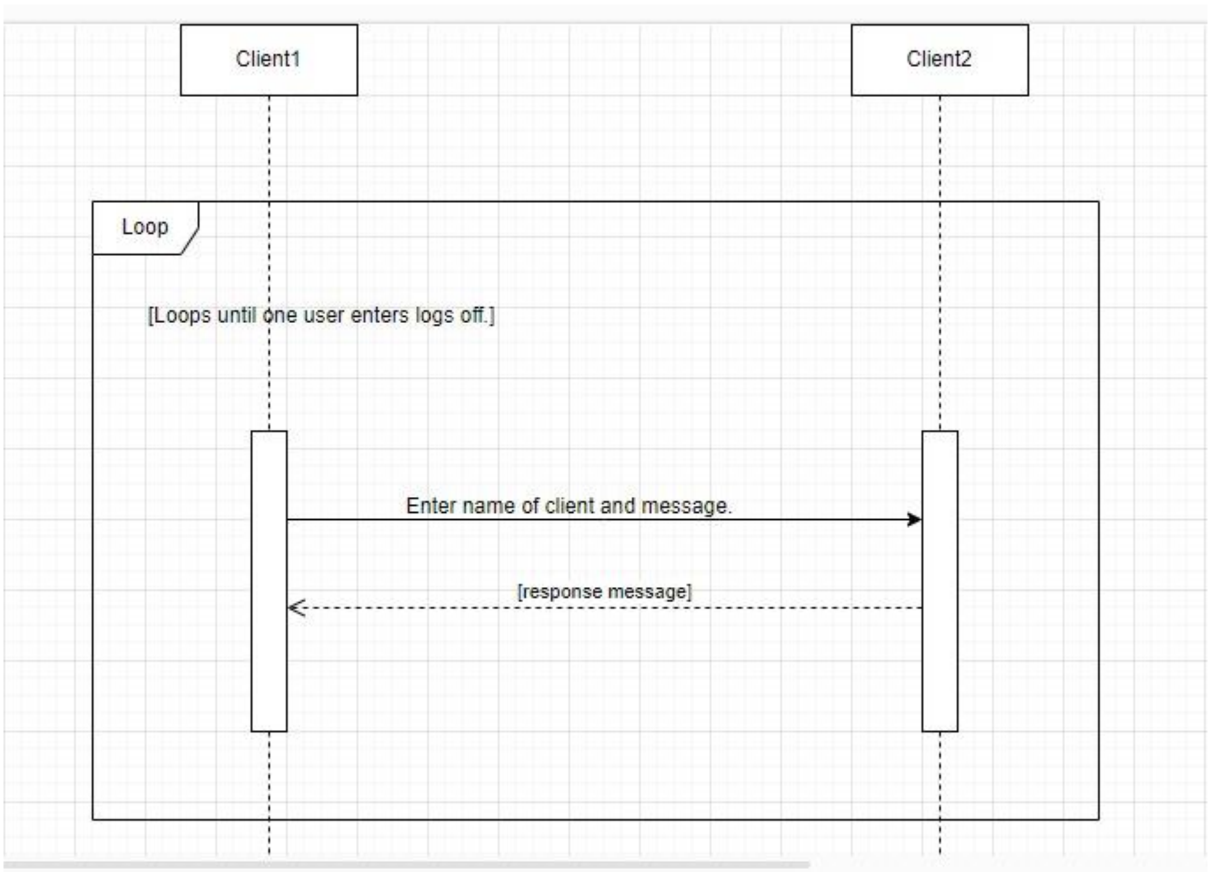
4. Sequence diagrams and protocol specifications

Sequence diagram for client and server interaction:


Client
Server
Loop
[Loops until user registers succesfully]
Enter name.
Enter privacy option
[successful login message]

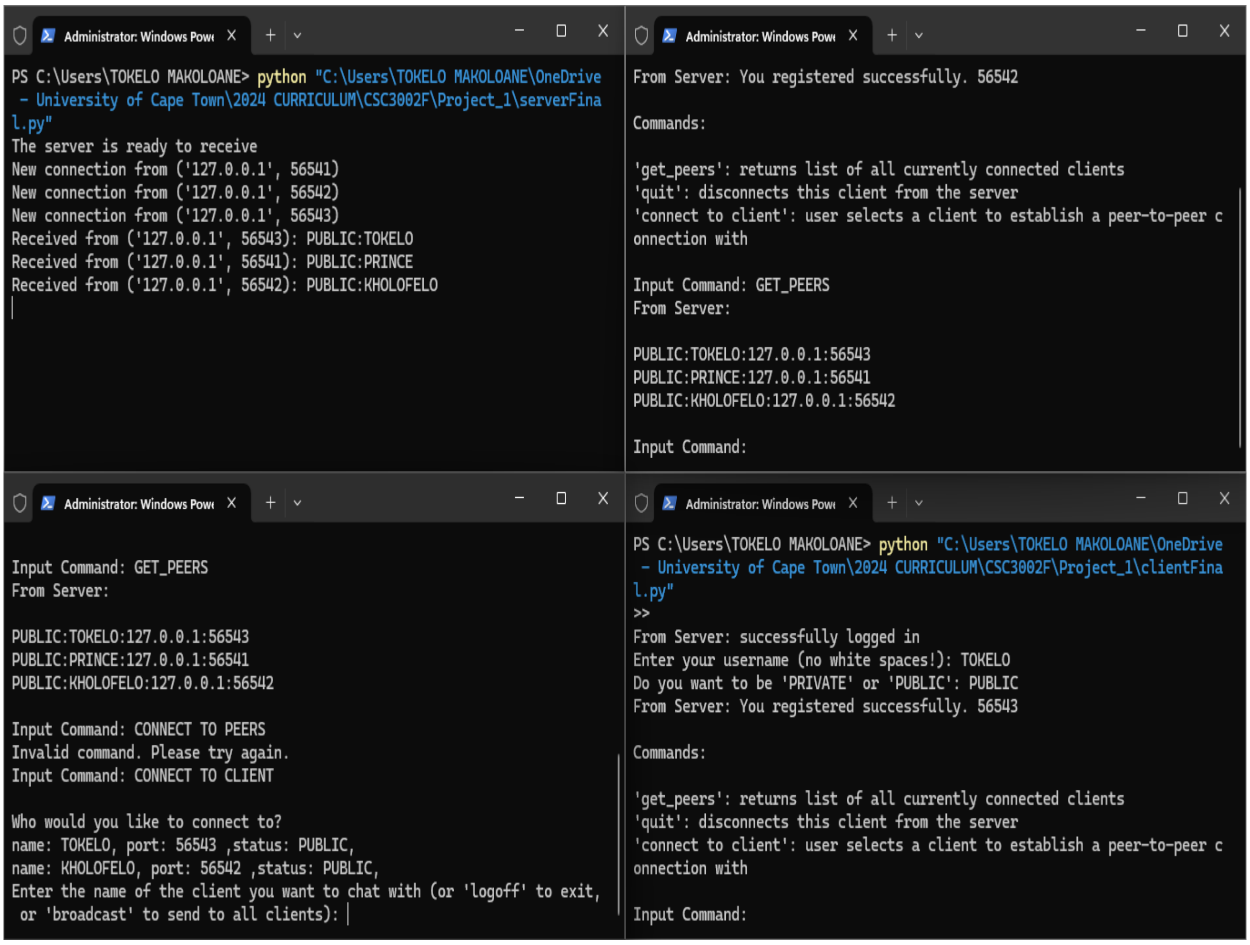

Loop
[Loop until user enters "quit" or when they chose a client they want to chat to.]
Enter get_peers
[List of all connected clients]
Enter connect to client
[list of clients to choose from]
Enter name of client or choose "broadcoast"

Sequence diagram for client-to-client interaction:


Client1
Client2
Loop
[Loops until one user enters logs off.]
Enter name of client and message.
[response message]

-**Protocol specification**: The chat application employs a well-defined protocol to ensure effective communication between clients and the server. Messages exchanged follow a structured format consisting of a header and body. The header includes essential details such as message type, command, recipient information, and sequence data, providing a standardized framework for both clients and the server to interpret messages accurately. The protocol is designed to be dynamic, with clear communication rules for different states, such as 'available(Visible to the server.),' 'connected(connected to server or another client.),' or 'away(Exited or logged off.).' To illustrate these rules, sequence diagrams have been developed, providing a visual representation of the message flows and interactions at various stages of communication. This protocol specification establishes a robust foundation for the chat application, ensuring structured, reliable, and secure communication between clients and the server.

5. Screenshots of application and its features:



```
PS C:\Users\TOKELO MAKOLOANE> python "C:\Users\TOKELO MAKOLOANE\OneDrive
 - University of Cape Town\2024 CURRICULUM\CSC3002F\Project_1\serverFina
l.py"
The server is ready to receive
New connection from ('127.0.0.1', 56541)
New connection from ('127.0.0.1', 56542)
New connection from ('127.0.0.1', 56543)
Received from ('127.0.0.1', 56543): PUBLIC:TOKELO
Received from ('127.0.0.1', 56541): PUBLIC:PRINCE
Received from ('127.0.0.1', 56542): PUBLIC:KHOLOFELO
```

```
From Server: You registered successfully. 56542

Commands:

'get_peers': returns list of all currently connected clients
'quit': disconnects this client from the server
'connect to client': user selects a client to establish a peer-to-peer c
onnection with

Input Command: GET_PEERS
From Server:

PUBLIC:TOKELO:127.0.0.1:56543
PUBLIC:PRINCE:127.0.0.1:56541
PUBLIC:KHOLOFELO:127.0.0.1:56542

Input Command:
```

```
Input Command: GET_PEERS
From Server:

PUBLIC:TOKELO:127.0.0.1:56543
PUBLIC:PRINCE:127.0.0.1:56541
PUBLIC:KHOLOFELO:127.0.0.1:56542

Input Command: CONNECT TO PEERS
Invalid command. Please try again.
Input Command: CONNECT TO CLIENT

Who would you like to connect to?
name: TOKELO, port: 56543 ,status: PUBLIC,
name: KHOLOFELO, port: 56542 ,status: PUBLIC,
Enter the name of the client you want to chat with (or 'logoff' to exit,
 or 'broadcast' to send to all clients):
```

```
PS C:\Users\TOKELO MAKOLOANE> python "C:\Users\TOKELO MAKOLOANE\OneDrive
 - University of Cape Town\2024 CURRICULUM\CSC3002F\Project_1\clientFina
l.py"
>>
From Server: successfully logged in
Enter your username (no white spaces!): TOKELO
Do you want to be 'PRIVATE' or 'PUBLIC': PUBLIC
From Server: You registered successfully. 56543

Commands:

'get_peers': returns list of all currently connected clients
'quit': disconnects this client from the server
'connect to client': user selects a client to establish a peer-to-peer c
onnection with

Input Command:
```

6. Conclusion:

Our chat application successfully integrates key features, providing a secure and efficient platform for instant messaging. The use of both TCP and UDP connections ensures reliability and real-time communication. Challenges faced during implementation were effectively addressed, resulting in a robust chat application.