**Project Name:** JAVA Chat Application

**Github Link:** https://github.com/projectsforstudents2022/JAVA\_Chat\_Application.git

**Why was this project created?**

With the ability to send and receive messages in real time, chat applications make it simple to communicate with individuals all over the world. Users of real-time chat apps can experience bespoke messaging features that are just like those they would experience in person, providing them with the same engaging and energetic interactions.

**What problem is it solving?**

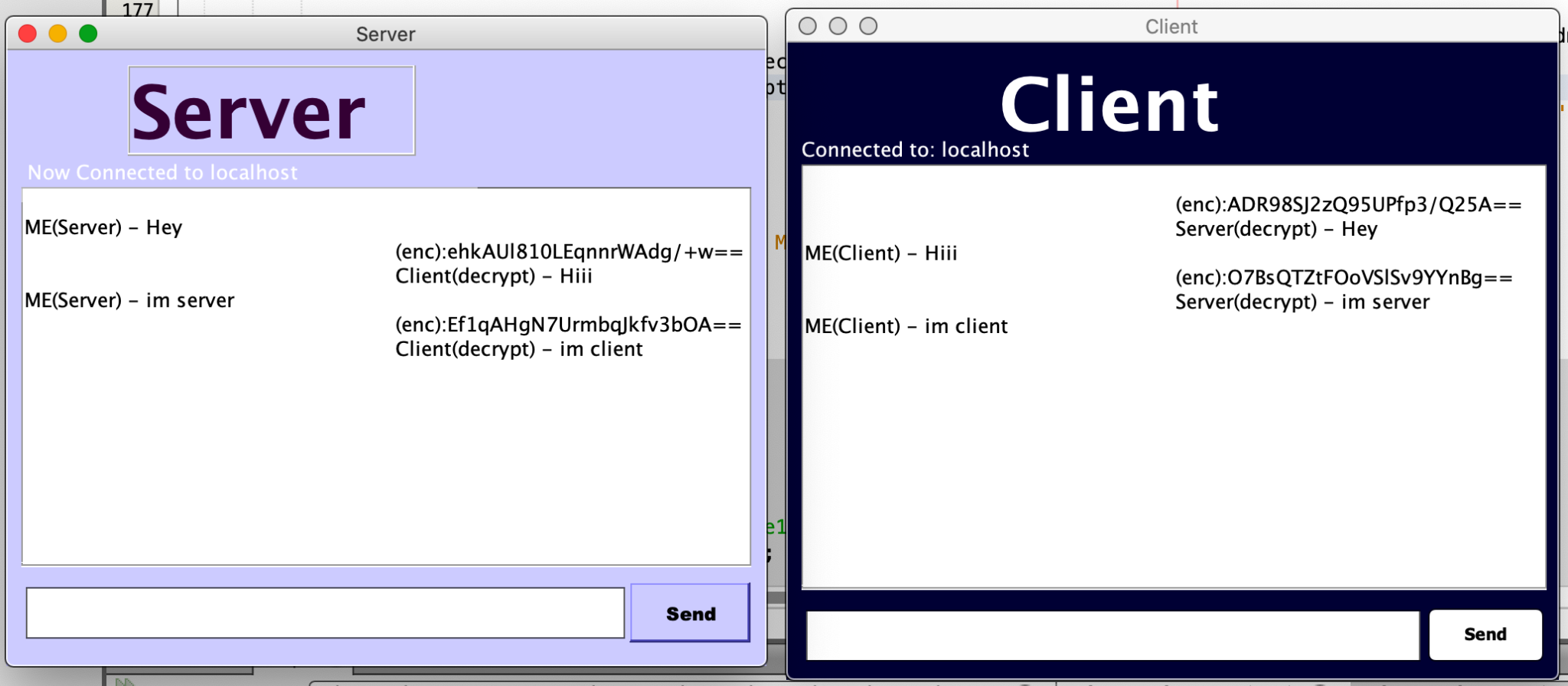
Java chat applications offer client software that runs on users' desktops and server software that runs on any networked system. To begin chatting, our client has to connect to a server that allows for both private and group chats.

**Entire explanation of project**

* **PROPOSED APPROACH**

Since TCP offers dependable delivery, which is essential for the particular application, it is utilized as the transport layer protocol. The lack of a timing guarantee offered by TCP is not particularly significant in the present situation. It uses a singleton class to implement the server. The local inet address and port 2000, which were picked at random and hard-coded, are used by the main thread to open a server socket. Then it waits for customers to establish connections. A User object and a unique thread are both created by the client when it connects to the server. The server keeps a map of User objects tied to clients connected to it that have been hashed against the user name. The word "server" appears in the header of messages that are meant for the server (control messages). To find out who is online and to log off, control messages are sent. In the first scenario, the user-specific thread retrieves all the keys from the hash map and gives the user a string containing a list of all the online users. When exit is requested, the thread closes any open streams and the client-related socket before quitting. Two threads are used to implement the Client, one for incoming messages and the other for outgoing messages. The main thread connects to the server and opens the socket. Then, a print writer and input buffered stream are opened for receiving and sending messages, respectively. Additionally, a buffered stream is created for reading from the console. While another thread handles incoming messages, the main thread handles outgoing messages. The user inputs the message string on the console, followed by the destination user name, to send a message. The message string is then read into the buffered stream reading from the console. The destination user name is added to the message string's beginning by the main thread to construct the formatted message.

* **RESULT**



* **CONCLUSION**

A more efficient and adaptable approach for chatting is provided by the chat programme. It is created using cutting-edge technologies in a way that offers a dependable system. The system's primary benefits are group chat, real-world connectivity, instant messaging, and additional security.