Asynchronous Chat Server and Client

The purpose of the project is to create Server and Client chat, where clients can connect to one server and can communicate with each other in real time. I am going to implement the project using **asyncio** and **websockets** libraries of python. There are two main python files: *server.py* and *client.py*, where you can find the functionalities of server and client respectively.

When the server is connected, it prints a note that server is successfully connected. Afterwards, when clients are connected to the server, they can send messages to each other in real time. In the process, the server systematically prints messages by telling how many connected clients there are or whenever client is connected or disconnected from the server.

One challenge I faced during implementation is that, after connecting the server and clients successfully, I started to send messages, everything worked fine, but without any reason each time after some seconds my server kept disconnecting. After investigating the code and understanding that there is no error in the code, I did research and find out that by default the server gets disconnected after some time because of inactivity of clients. So, I change some parameters to overcome the problem of server disconnection (ping_interval=None, ping_timeout=None).

Another challenge I had was that the clients could not simultaneously type messages and receive, print the messages of another clients. Finally, after understanding from which part of code the problem is coming, I added *aioconsole* library in my code so that the client can both send and receive messages concurrently without blocking the event loop, ensuring the smooth operation of the asynchronous chat application.

Optionally, when clients are getting connected to the server, they have chance to define names for themselves. The server keeps last messages so the clients can see recent chat history. Messages include timestamps by indicating the time that each message was sent by the client. Chat also allows the users to use emojis. Also, if the user wants to send direct message to specific client to have private messages, the client can in front of the private message type 'dm' or 'DM' after it specify the name of the client and finally the private message text, like this example ('DM|dm clientName privateMessage' - 'DM Client2 Hello dear'). In this case other clients will not receive the private message and it will not be printed in the server.