Main idea: I propose creating a file that can be run on two different terminals, enabling users to exchange messages with each other. To ensure the sending and waiting states of two users, I have devised a plan: the first user to connect to the server (by running the file) will be in the write state, while the second user will initially be in the waiting state.

Technical aspect: To implement this idea, I will use server keys to maintain communication order. There will be three main server keys: server[‘status’], server[‘received’], and server[‘message’]. The server[‘status’] key will hold the current state of the server, which will be one of three possible values: ‘off’, ‘one’, or ‘two’. When the first user runs the file, server[‘status’] will be set to ‘one’, indicating that only one user is currently connected and another connection is required. Once the second user connects, server[‘status’] will be set to ‘two’, indicating that the connection is established. Any new user attempting to connect to the server will be notified that the server is currently in use, and a connection cannot be established.

After successful connections are established, chatting can commence. The first user who connected will be able to type their message, while the second user will receive a prompt that the other user is typing. When the message is received by the server, server[‘received’] will be set to ‘1’, indicating that the second user has a message to receive. The message will be presented to the user, and they will be able to type their response while the first user is notified that the other user is typing. Each user’s state will be stored in a variable called ‘state’ within the file, which can have two values: ‘send’ or ‘wait’. Initially, the first user will have the state ‘send’, while the second user will be in the state ‘wait’. The state will be flipped after sending and receiving a message, enabling the messaging service to meet the specified requirements between two users.

Furthermore, I will ensure that all server errors are caught using ‘try’ and ‘except’ blocks in Python. I will also consider keyboard interrupts and notify users when the server connection is interrupted, providing a reason for the interruption. Additionally, if one user leaves the conversation, the other user will be notified and the connection will be closed.

In conclusion, I have created a safe and reliable messaging service that meets the specified requirements and accounts for all possible errors that may occur.

------------------------------------------------------------------------------------------------------------------------

Main idea: I believe to make a file which can be run in two different terminals and then will let the users to send each other messages. In order to fulfill the requirement of sending and waiting states of two users, I have decided that whichever user connects to the server first(runs the file) will be in the write state and the user who connects second will initially be in the waiting state.

Technical aspect: In order to achieve this idea I need to use server keys in order to keep the communication in order. For this reason I have decided to have three main server keys: server[‘status’], server[‘received’] and server[‘message’]. The server[‘status’] key will hold the current state of the server which will be on of the three possible values: ‘off’, ‘one’ and ‘two’. Whenever the first user runs the file, the server[‘status’] is set to ‘one’ meaning that there is only one user connected so far so we need another connection. As soon as the second user connect the server[‘status’] will be set to ‘two’ indicating that the connection is established. At this point if any new user tries to connect to the server, they will be notified that the server is in use and the connection cannot be stablished. After successfully connecting our users the chatting can start. First user who connected will be able to type their message. Second user will get a prompt that the other user is typing. As soon as the message is received by the server, the server[‘received’] will be set to ‘1’ indicating that the second user has a message to receive. The message will be presented to the user and now they will be able to type their message while the first user will be notified that the other user is typing. I will keep the state of each user in variable state in the file which will be able to have two values: ‘send’ or ‘wait’. So, initially the first user will have state ‘send’ while the second user will be in state ’wait’. State will be flipped after sending and receiving a message. Through this logic I will be able to correctly conduct a messaging service following the provided requirements between two users.

Additionally, I will make sure to catch all the server errors using ‘try’ and ‘except’ blocks in python. I will also take into account the keyboard interrupt and will notify the users when the server connection is interrupted and reason for which it was interrupted. Also, if one user leaves the conversation the other user will be notified and the connection will be closed.

To conclude, I believe that I have succeeded to create a general and safe messaging service between two users provided the requirements and considered all the corner cases for any occurring errors.