Part a

Explanation of our program:
We have 3 main classes: Server.py, Client.py and unit_test.py.

Server:

In the server class we are creating the server, which will listen and handle new users that want to connect to the server. We have functions that handle the file transfer (see FTPtoClient).
We've created a TCP header with sequence number, ack number, ack, sf and rwnd.
4 bytes of sequence number, 4 bytes of ack number, 1 byte of ack , 1 byte of sf and 2 bytes of rwnd.
Firstly the function 'toHeader' checks how much bytes in total the client can accept. After it gets the amount, it turning back the received amount of bytes to an integer as a tuple.
Then, to handle the transfer we used congestion control (you can see the diagram and explanation of the congestion control in the attached pdf file).
In this class we have the GUI implementation using tkinter.
We have the handle client function which is detecting if any client want to connect or disconnect, and handle their requests and process them. The server is allowing each client to connect, see list of online users, download a file from the server, send private message and public message and to disconnect if he want to.
Over the TCP, we used UDP too to speed up the transfer process (as requested).

Client:

In the client class we have two similar function to the server class (toHeader, fromHeader).
This class basically describes the client functionality, the received segments from the server and functions that handles the file download (after successfully 3 way handshake).
We have the GUI implementation of the client side using tkinter.
We have functions that helps the client to send message, receive message and download a file from the server (after successfully connection and 3 way handshake).

Unit_test:

First, we setting up a new server. We register 3 new users, and connecting them to the server.
We checking if the connection to the server works and all the functions work properly.
Checking if we can send message, receive message and download a file from the server.
All the tests passed successfully, but if you would run the tests after you've started already a connection it may throw you an error. This happens because the serve is still working and not allowing for new set up.

In the classes above we've used also multiple threading methods to avoid network problems in our system. For example if few users download the same file in the exact time, multi threading will overcome this problem.