Code Design and Implementation

There are 2 files :
1. server.py
2. client.py
The two files are used for running the corresponding applications.

Server.py :

IMP: This has been implemented in Python 3. Thus, requires the data to be encoded(from string) before being sent and decoded(back to string) after being received.

The main function is the starting point of the program. Here an initial socket is being made for the server which listens for client-connections. As soon as a connection is made, a thread is started for the particular server-client socket . Also the function, clientthread is associated with each socket made and so this function runs parallelly with other instances of the function for other sockets.

Now the important function , the heart of the server is the clientthread function. This function takes in argument ‘conn’ which is the socket associated with that instance of the function and this thing is being used all around the function to send, receive messages.

Messages format (For internal representation only, actual messages are gui based)
If the message starts with a ‘0’ , it is a non registration message.
Else the other case has to be a registration message

Non registration message formats:
Single Receiver : @<User> : <Message>
Multiple Receivers : @<User 1> @<User 2> @<User 3>.....@<User n> : <Message>
Multimedia : ^(Any of the above 2 message formats follow here accordingly)
IMP: The ‘^’ symbol at the starting is used to denote the multimedia message.

The parsing of received messages has been done according to the formats to separate the message and sender components and can be fairly understood from the commented code. The functions for processing are standard ones like rstrip(), split(), decode(), encode(),join(),

Multicast_data takes in a socket ‘sock’ from which data was sent, the ‘message’ to be sent, ‘multimedia’ which indicates whether the message is a multimedia file and a list of sockets, ‘list_of_sock’ to which the message has to be multicasted.Actually, the unicast is a special case of this function when the no of sockets in the list of sockets is 1 only and is implemented in this way only and is implemented through this function actually.

Registration message:
All that is done here is : extract the name and password from the received data, and then call the reg_user function to register the new user details in the file ‘database.txt’ which is the local database.

Two main data structures:
USERS_LIST[<name>] = <socket>
SOCK_LIST[<socket>] = <name>

USERS_LIST maps the names of clients to the sockets, and the SOCK_LIST does vice versa. In this way having one of the info allows us to access the other in O(1) time !

The function authenticate() takes in a username , password and verifies if it is in the data base (Local db here, database.txt). Similarly, reg_user takes in a username and password and adds the corresponding entry in the database.txt

The function broadcast_data() takes in a socket and message and broadcasts it to all sockets in the current active connection list apart from itself.

Try and Except blocks have been used to enclose each transaction(sending and receiving) for smooth error handling.(There may still be undiscovered errors not handled by us, but we have tried our best to cover all we could)

The function send_active_list() makes a list of all currently active users, the ‘sock’ argument it takes in is the socket which is about to be closed . Actually this function is called when the ‘sock’ socket is about to be closed, and to update the list of all other active users, this thing makes a current active list to be sent to the rest.

Client Side:

The rundown client application is just an open socket established with the server where most of the message exchanges take place in accordance with the protocol mentioed above. We have added the GUI features using the Tkinter library in python and also used the ttk python module which lets us define some predefined widgets used in the application. Also we have added the features of mutltimedia messaging (MMS’s) using the base64 library used for encoding the different multimedia messages.

We have our main class Application inside which all of our GUI and the different functionalities are implemened. We have used a class since it becomes easier to define the objects of the GUI library Tkinter and the widgets formed are in the normal thinking way of execution such that new frames and objects are created inside a root frame and each frame is enclosed in a function and that there is a sequential execution of the frames. Also styling and formatting is an important functionality that is provided by the library which makes it very easy to change the different objects. The reference can be found easily on googling.

Main Functions:

launch_app() : This has the start frame of the applcation wherein there are two options: one for registering the user and one for logging in.

reg_menu(): This has the registration frame and the registration GUI part of the application. We also take the hostname and the port fields used to open the connection to the server.

reg_user() function, called upon pressing the register button used for opening a tcp connection to the server and performing the basic registration communication in the format required, given above.

client_menu(): Similar to the reg_menu, this function is the maping for the log in page of the user and takes in hostname , port username and password and calls the launch_client() function on pressing login.

launch_client(): It is the main function where all the main communication take place
checks if user is in database by communicating with the backend.
There are four separate frames in this main frame:

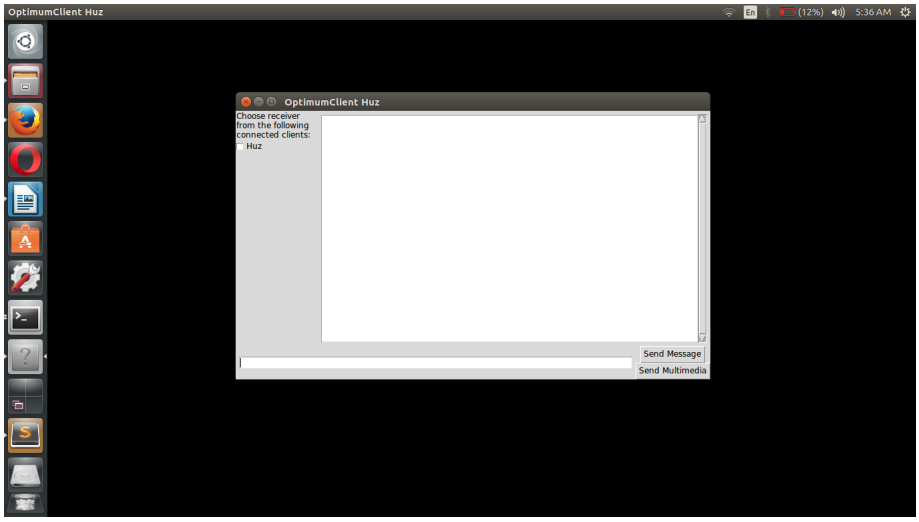
Chat Frame: It contains the chat history of the client in this session. We also have a scroll suboject of this frame which enable you to scroll the widow in case the messages exceed the window size.

Clients Frame: It contains a checkboxes of all the active connections avaiable for the particular user in Checkboxes format.

Entry Frame: Contains the entry box correspoding to the message to be typed.

Button Frame: It contains the two buttons 'send message' and 'send multimedia' which on a press event will send the text in the entry frame or the multimedia file selected by the browse function to the clients selcted in the checkboxes in the Clients frame.

The corresponding styling and placements are done
A new clienchat function thread is created after this to display and update the chats.
On login the user receives the list of active users from the server which it implements in the form of checkboxes such that on selection of one or more than one we send the message or MMS to every one of them using the send and the multimedia_send functions.



Browse(): Uses the tkFileDialog object's function askopenfilename() to create a browsing dialog which on double clicking or pressing select sends the file to the user using the multimedia_send function.

Send(): This is the function that implements the actual sending of messages from the text entered in the chat_entry and sends it to the server with the required encapsulation of data as per our message protocol. All the three types of casting can be performed using this function.

multimedia_send(): Invoked to send a multimedia file to a particular destination. The filename is fetched via the **browse()** method and the contents are read in after opening it in "rb" mode. The contents are encoded in base64 encoding using the **base64.b4encode()** method. Following the protocol, the data to be sent is started with a **'^'** and the list of active users to which it is intended to are appended using **'@'**. This content is now appended by ':' followed by the filepath. The contents

are appended after putting in one more ‘:’ and sent to the server. For display purposes, the chat history is updated with the filepaths only.

ClientChat(): This function is called a number of times by the user and is responsible for updating the frame from time to time, which includes updating the active clients list as well as the chat history block every time a message is received based on the protocol. It destroys the previous frame and updates it and ouputs the new frame after parsing and processing the messages received from the server.

initial_setup() : Returns the list of current active users after the client has registered for the application. Following the protocol, sends a ‘0’ first to the server. Receives the question **“What is your name?”** from the server. An attempt for authentication is made by the client, following which, if there is success, the list of active users is returned. Else the **client_menu()** function is called again.

client_quit() : Procedure to exit the window, Mapped with the ‘X’ button of the window as well.