

Haorui Chen_Task4

2019-03-27

Menu

1. Part 4.1	2
1.1 Development.....	2
1.2 Test	2
1.3 Evaluation	3
1.4 Q&A.....	3
2. Part 4.2	5
2.1 Development.....	5
2.2 Test	5
2.3 Evaluation	6
2.4 Q&A.....	6
3. Part 4.3	8
3.1 Development.....	8
3.2 Test	8
3.3 Evaluation	9
3.4 Q&A.....	9
4. Part 4.4	11
4.1 Development.....	11
4.2 Test	11
4.3 Evaluation	12
4.4 Q&A.....	12
5. Part 4.5	14
5.1 Development.....	14
5.2 Test	16
5.3 Evaluation	17
5.4 Q&A.....	17

1. Part 4.1

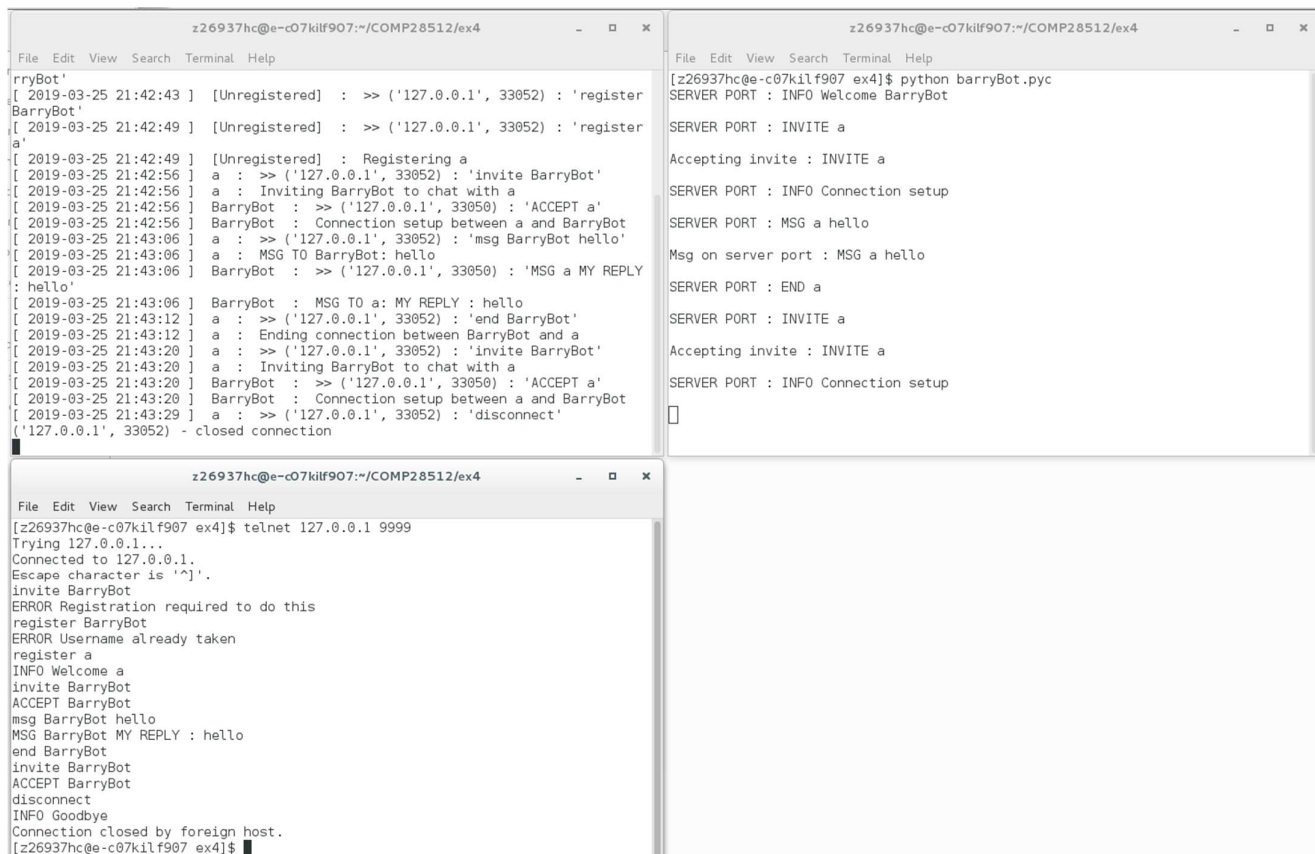
1.1 Development

Before starting on android studio, I first checked that the Telnet client can register with the proxy-server, that its name and IP address are stored in the location-server, that it can send messages to BarryBot and receive his replies and that functionalities of proxy-server and BarryBot provided by sending feasible requests through TELNET client. This can also help me to make clear how communication is achieved between client and server and get a straightforward view of what information I may get in later development in android studio. I opened proxy-server, BarryBot and TELNET server following this procedure and checked commands one-by-one, simulating whole procedure of communication between two clients.

For 'Hello BarryBot', I first figured out a feasible choice for graphics when setting up emulator. I got EXITCODE 139 with default choice 'Automatic', After searching on internet I discovered that I need to use 'Software-GLES 2.0'. I looked for the specific file that controls content that will be manifested on screen and altered it.

1.2 Test

For getting familiar with proxy-server and BarryBot, I tried a random command before registration, register, invite BarryBot, send message to it, end connection, re-invite it and disconnect to simulate a normal procedure of online chatting. The demonstration of this procedure is shown in pic 1-1 below. Reaction of client, proxy-server and BarryBot all meet my expectation.



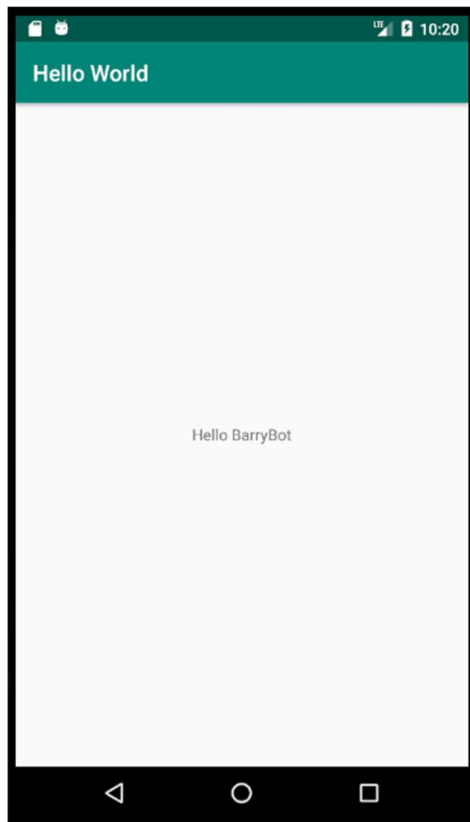
```
z26937hc@e-c07kilf907:~/COMP28512/ex4
File Edit View Search Terminal Help
rryBot'
[ 2019-03-25 21:42:43 ] [Unregistered] : >> ('127.0.0.1', 33052) : 'register
BarryBot'
[ 2019-03-25 21:42:49 ] [Unregistered] : >> ('127.0.0.1', 33052) : 'register
a'
[ 2019-03-25 21:42:49 ] [Unregistered] : Registering a
[ 2019-03-25 21:42:56 ] a : >> ('127.0.0.1', 33052) : 'invite BarryBot'
[ 2019-03-25 21:42:56 ] a : Inviting BarryBot to chat with a
[ 2019-03-25 21:42:56 ] BarryBot : >> ('127.0.0.1', 33050) : 'ACCEPT a'
[ 2019-03-25 21:42:56 ] BarryBot : Connection setup between a and BarryBot
[ 2019-03-25 21:43:06 ] a : >> ('127.0.0.1', 33052) : 'msg BarryBot hello'
[ 2019-03-25 21:43:06 ] a : MSG TO BarryBot: hello
[ 2019-03-25 21:43:06 ] BarryBot : >> ('127.0.0.1', 33050) : 'MSG a MY REPLY
: hello'
[ 2019-03-25 21:43:06 ] BarryBot : MSG TO a: MY REPLY : hello
[ 2019-03-25 21:43:12 ] a : >> ('127.0.0.1', 33052) : 'end BarryBot'
[ 2019-03-25 21:43:12 ] a : Ending connection between BarryBot and a
[ 2019-03-25 21:43:20 ] a : >> ('127.0.0.1', 33052) : 'invite BarryBot'
[ 2019-03-25 21:43:20 ] a : Inviting BarryBot to chat with a
[ 2019-03-25 21:43:20 ] BarryBot : >> ('127.0.0.1', 33050) : 'ACCEPT a'
[ 2019-03-25 21:43:20 ] BarryBot : Connection setup between a and BarryBot
[ 2019-03-25 21:43:29 ] a : >> ('127.0.0.1', 33052) : 'disconnect'
('127.0.0.1', 33052) - closed connection

z26937hc@e-c07kilf907:~/COMP28512/ex4
File Edit View Search Terminal Help
[z26937hc@e-c07kilf907 ex4]$ python barryBot.pyc
SERVER PORT : INFO Welcome BarryBot
SERVER PORT : INVITE a
Accepting invite : INVITE a
SERVER PORT : INFO Connection setup
SERVER PORT : MSG a hello
Msg on server port : MSG a hello
SERVER PORT : END a
SERVER PORT : INVITE a
Accepting invite : INVITE a
SERVER PORT : INFO Connection setup
[]

z26937hc@e-c07kilf907:~/COMP28512/ex4
File Edit View Search Terminal Help
[z26937hc@e-c07kilf907 ex4]$ telnet 127.0.0.1 9999
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^'.
invite BarryBot
ERROR Registration required to do this
register BarryBot
ERROR Username already taken
register a
INFO Welcome a
invite BarryBot
ACCEPT BarryBot
msg BarryBot hello
MSG BarryBot MY REPLY : hello
end BarryBot
invite BarryBot
ACCEPT BarryBot
disconnect
INFO Goodbye
Connection closed by foreign host.
[z26937hc@e-c07kilf907 ex4]$
```

1-1 simulation of online chatting using client and proxy-server

For setting up environment of sample project, the string that needs to be altered is in activity_main.xml, effect of the alter is shown in pic 1-2.



1-2 Result of modifying the 'Hello World' message to 'Hello BarryBot'

1.3 Evaluation

By finishing this part, I understood how client and proxy server communicate and satisfied about its functionality. Also, I got familiar with android studio emulator configurations by setting up its running environment under this sample project.

1.4 Q&A

1. Are you satisfied that the two Python programs 'lab4-server.py' and 'BarryBot.py' will allow you to test a simple messaging system developed in Android?

A: Yes. They are simulation of a real server and another user respectively, the proxy server can achieve transmission of messages from one to another, distinguish sender and receiver of every message and have a pretty low delay; barryBot can be used to check whether message is received correctly, and by sending the same message automatically, we can test if our client can receive message from others correctly. Also, the whole chatting procedure in a messaging system like invitation, acceptance/declination and ending can all be represented by them.

2. If BarryBot were a real user, how would he know when the TELNET client has terminated your connection to him? How could the protocol be improved in this respect?

A: if we deliberately add END command, TELNET will send an END command to server, the server can know we are ending connection from BarryBot from the type of command and takes the charge of informing BarryBot of this; yet if we disconnect or kill directly, BarryBot cannot be informed and error will occur if it tries to send message to you.

3. Why are literal strings discouraged in Android? What is the preferred alternative?

A: Because they are prone to be changed very often, and hard-coding them means that if you want to make some alteration to some/all of them you will have hard-times first locating them then altering them one by one, especially when the code base is huge. Instead, we usually put all literals in string.xml. Also, if the strings tend to be excessively

long, saving them into strings.xml helps scale down file size.

4. What is the purpose of the following files and directories created by Gradle: (a) src/main/java, (b) res/, (c) AndroidManifest.xml?

A: (a) for including the class that set up the main page of our application; (b) for including all the resource files for drawing UI and application logo (on different mobile resolution level); (c) Resides within '···\main' and defines some characteristics of your application including its name, permissions needed, SDK version (version of Android studio), etc.

5. How does the naming of a package affect the file structure of an Android project?

A: It will influence range of variables in classes within package. Also, contains the source codes for your application, including tests.

2. Part 4.2

2.1 Development

In this part, I simply added a line: `System.exit(0);` to `onClick()` function of 'KILL' button and change information of 'SEND' button to be send to proxy-server as "REGISTER myname". Also, I kept proxy-server shut down first to test sample app with LogCat open to observe any error message when clicking 'SEND' button.

```
killButton.setOnClickListener((v) -> {  
    // OnClick actions here  
    // Exit app  
    System.exit( status: 0);  
});  
  
// Get the text area for commands to be transmitted as defined in the Res dir xml code  
register = "REGISTER myname";  
// Get the send button as defined in the Res dir xml code  
Button sendButton = findViewById(R.id.btnSendCmd);
```

2-1 codes added in part 4.2

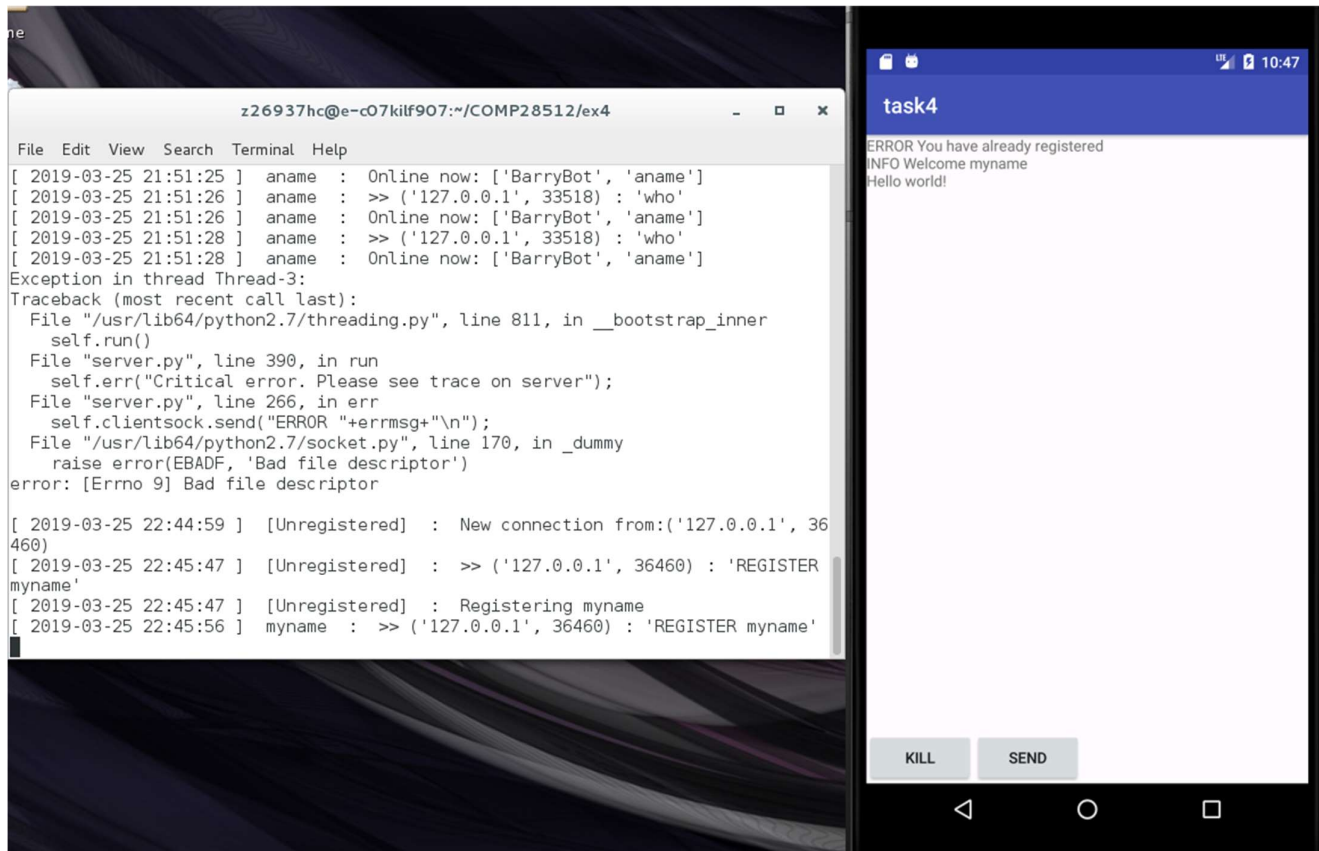
2.2 Test

With proxy-server shut down, clicking 'SEND' that represent 'REGISTER myname' command will get an error, as shown in pic 2-2.

```
03-25 22:49:41.573 4893-4932/com.example.mbassjsp.task4 E/Network and receiver: Socket failed  
failed to connect to /10.0.2.2 (port 9999): connect failed: ECONNREFUSED (Connection refused)  
03-25 22:49:41.573 4893-4932/com.example.mbassjsp.task4 E/Network and receiver: Socket failed  
failed to connect to /10.0.2.2 (port 9999): connect failed: ECONNREFUSED (Connection refused)
```

2-2 error in logcat when clicking 'SEND' without a server running

With proxy server running, clicking 'SEND' will result in corresponding reaction in proxy server and feedback on TextView, clicking again will get an error message, which all meet my expectation. Details are shown in pic 2-3.



2-3 test for clicking 'SEND' button

When clicking 'KILL' button, application can quit successfully, yet an error is given on proxy-server as we are quitting

so abruptly as at this stage I did not end connection and disconnect. Error message is shown in pic 2-4.

```
[ 2019-03-25 22:54:08 ] [Unregistered] : New connection from:('127.0.0.1', 36968)
[ 2019-03-25 22:54:23 ] [Unregistered] : >> ('127.0.0.1', 36968) : 'REGISTER myname'
[ 2019-03-25 22:54:23 ] [Unregistered] : Registering myname
Exception in thread Thread-1:
Traceback (most recent call last):
  File "/usr/lib64/python2.7/threading.py", line 811, in __bootstrap_inner
    self.run()
  File "server.py", line 390, in run
    self.err("Critical error. Please see trace on server");
  File "server.py", line 266, in err
    self.clientsock.send("ERROR "+errmsg+"\n");
  File "/usr/lib64/python2.7/socket.py", line 170, in _dummy
    raise error(EBADF, 'Bad file descriptor')
error: [Errno 9] Bad file descriptor
```

2-4 errors appear when quitting app without disconnect first

2.3 Evaluation

Currently the app is rather primitive, only register and quitting is achieved and error handling is not added, and title of app is not a demo version. Current sample app is far from one that can function normally.

2.4 Q&A

1. Where did you store your fixed text message?

A: I stored it in MainActivity.java by using a String.

2. How did you check that the 'btnSendCmd' and 'btnKill' buttons had the desired effect?

A: For btnSendCmd, I check by context on 'context screen' that displays information from proxy-server. When we click 'send' button, the context is welcome (username), showing it has the desired effect; for btnKill, I check whether I can exit the application and go back to the desktop. This happens and I can know that KILL button works. Also, we can check logcat and see if corresponding message appears at console.

3. Explain how you used the logcat debugging facility and show an example of the output obtained (in your demo and your report).

A: I use logcat prints to monitor the behavior of my code. Since in every branch of code we are provided logcat printouts, I can easily make sure of what part of code have been executed and by comparing it with expected behavior I can know whether it currently have errors or not.

For example, when I click SEND again after I've clicked once, I get

03-20 17:32:59.164 4587-4668/com.example.mbassjsp.task4 I/Transmitter: Sending command: REGISTER .name

03-20 17:37:58.229 4587-4682/com.example.mbassjsp.task4 I/Transmitter: Sending command: REGISTER .name

03-20 17:37:58.261 4587-4660/com.example.mbassjsp.task4 I/Network and receiver: MSG recv : ERROR You have already registered

The first line indicates I have successfully registered (since no extra lines appears), and the second and third line correctly shows that user .name has registered, all meet my expectation.

4. How does the 'runOnUiThread' method deal with inter-thread communications in Android?

It uses TextView component of its parent thread (main activity thread) it gets beforehand and uses a method of the

component as receiver. As this thread resides in current working thread, it can also use messages in this thread and give the message to receiver in another. These threads are stored in a queue, There may be many runnable events waiting in this queue to be run by the main thread. So the UI thread will deal with these one by one in a way that will not cause any conflicts. When the UI thread is ready to deal with an action placed on the queue by a worker-thread, it will do so safely.

5. Why is 'runOnUiThread' needed for the NetworkConnectionsAndReceiver thread but not for the Transmitter thread?

A: because work of 'runOnUiThread' is getting message from server and put to client. This is needed for NetworkConnectionsAndReceiver as part of its work is to keep the receiver channel open. Also, this has nothing to do with sending message to server, which is this only work of Transmitter thread.

3. Part 4.3

3.1 Development

In this stage a text input box is added to enable the user to enter his or her name, the name is used to register user on proxy-server when clicking 'SEND' button. Another button for getting online user list is also implemented, yet it should only return proper list when user is registered. All the codes added in this step are in pic 3-1. At current stage all feedback message will all be put in given Textview box. Layout of current stage is pic 3-3.

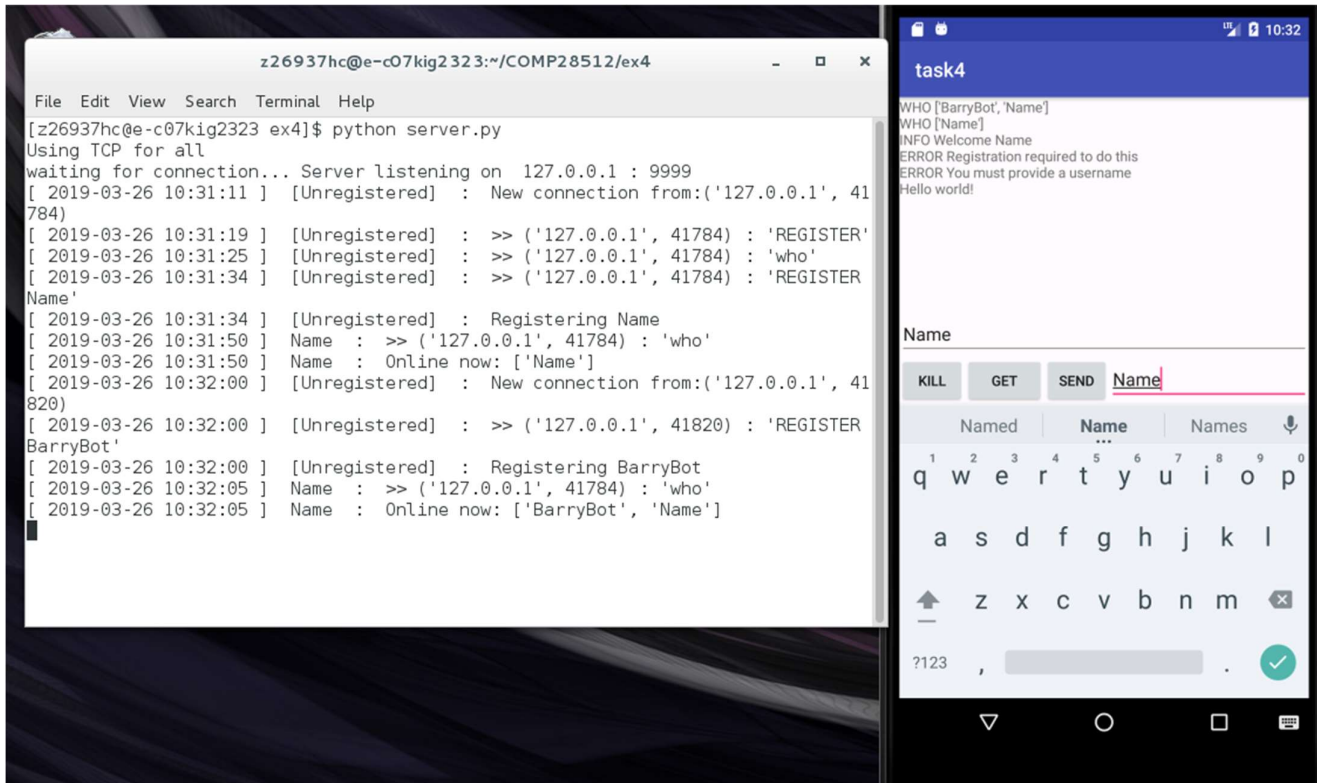
```
// Get the text area for commands to be transmitted as defined in the Res dir xml code
register = "REGISTER ";
// Get the send button as defined in the Res dir xml code
Button sendButton = findViewById(R.id.btnSendCmd);
// Make the kill button receptive to being clicked
// Button click handler
sendButton.setOnClickListener(onClick(v) -> {
    EditText editText = (EditText) findViewById(R.id.editText);
    message = editText.getText().toString();
    // OnClick actions here
    // Instantiate the transmitter passing the output stream and text to it
    if(networkConnectionAndReceiver.getOutputStream() != null) { // Check that output stream has be setup
        Transmitter transmitter = new Transmitter(networkConnectionAndReceiver.getOutputStream(), transmitterString: register+message);
        transmitter.start(); // Run on its own thread
    }
});

// Get the send button as defined in the Res dir xml code
Button getButton = findViewById(R.id.btnGet);
// Make the kill button receptive to being clicked
// Button click handler
getButton.setOnClickListener((v) -> {
    Transmitter transmitter = new Transmitter(networkConnectionAndReceiver.getOutputStream(), transmitterString: "who");
    transmitter.start(); // Run on its own thread
});
```

3-1 code for 'SEND' and 'GET' button

3.2 Test

Clicking without a username already in text input box will give an error on server, reminding user to give a username, clicking 'GET' button will also return error message, mention that the operation can only be done with a user online with the app. With a username given, registration can be successfully done and an online list can be given to user when clicking 'GET' button. The whole procedure of testing, including message on proxy-server and response on app is shown in pic 3-2.



3-2 testing procedure for part 4.3

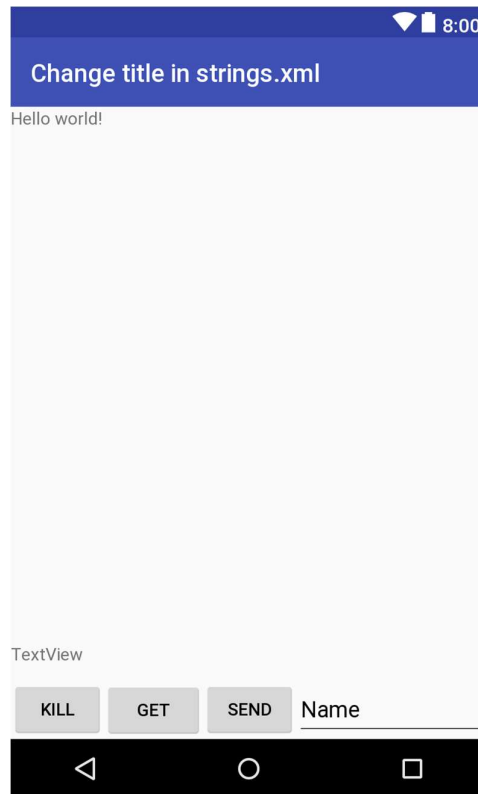
3.3 Evaluation

Minor improvements is implemented in this version, giving more flexibility on choosing username, 'GET' button help us to get status of other users, being a starting point for user interaction, yet communication is still not achieved, meaning that current version is still far from a reasonable simulation of a messaging system.

3.4 Q&A

1. How is your screen layout (however simple) made suitable for the application as developed so far?

A: A screenshot of current layout is provided here. Title of the application is high above the main page, below is enough space for displaying message received from server. At the bottom of application page I give 3 buttons for 3 functions required in this part respectively, and a Plain Text bar for input username is on the right.



3-3 application layout at current stage

2. What happens if the proxy-server is not on-line?

A: When we click SEND, nothing will happen. Since we are not sending anything to the server as it does not exist, we cannot expect to get any feedback. Yet there is an error in LogCat as hint for us:

03-20 22:17:20.143 4563-4633/? E/Network and receiver: Socket failed

failed to connect to /10.0.2.2 (port 9999): connect failed: ECONNREFUSED (Connection refused)

3. What happens if you try to register with a name that is already in use?

A: Proxy server will notice and give corresponding feedback: 'ERROR You have already registered'. runOnUiThread will put this to UI to get us know.

4. Could anything else go wrong with this preliminary version of the app?

A: If I KILL it then go back to application, we can still register with a new user name, which we are supposed to succeed in a normal messenger system.

4. Part 4.4

4.1 Development

In this part, button for inviting, accepting and setting up a text message link are implemented. Code for them are in pic 4-1, 4-2 and 4-3 respectively. I can invite an online user (yet inviting offline user or no user parameter given will have error giving back); can accept invitation from a user (yet accept that of a offline one will give error); and can send message to other online user when a connection between us has been set up. Screenshot of app layout at current stage is pic 4-5.

```
invite = "INVITE ";
// Get the send button as defined in the Res dir xml code
Button inviteButton = findViewById(R.id.btnInvite);
// Make the kill button receptive to being clicked
// Button click handler
inviteButton.setOnClickListener((v) -> {
    EditText editText = (EditText) findViewById(R.id.editText);
    message = editText.getText().toString();
    // OnClick actions here
    // Instantiate the transmitter passing the output stream and text to it
    if(networkConnectionAndReceiver.getOutputStream() != null) { // Check that output stream has be setup
        Transmitter transmitter = new Transmitter(networkConnectionAndReceiver.getOutputStream(), transmitterString: invite+message);
        transmitter.start(); // Run on its own thread
    }
});
```

4-1 code added for implementing 'INVITE' button

```
accept = "ACCEPT ";
// Get the send button as defined in the Res dir xml code
Button acceptButton = findViewById(R.id.btnAccept);
// Make the kill button receptive to being clicked
// Button click handler
acceptButton.setOnClickListener((v) -> {
    EditText editText = (EditText) findViewById(R.id.editText);
    message = editText.getText().toString();
    // OnClick actions here
    // Instantiate the transmitter passing the output stream and text to it
    if(networkConnectionAndReceiver.getOutputStream() != null) { // Check that output stream has be setup
        Transmitter transmitter = new Transmitter(networkConnectionAndReceiver.getOutputStream(), transmitterString: accept+message);
        transmitter.start(); // Run on its own thread
    }
});
```

4-2 code added for implementing 'ACCEPT' button

```
msg = "MSG ";
// Get the send button as defined in the Res dir xml code
Button msgButton = findViewById(R.id.btnMsg);
// Make the kill button receptive to being clicked
// Button click handler
msgButton.setOnClickListener((v) -> {
    EditText editText = (EditText) findViewById(R.id.editText);
    message = editText.getText().toString();
    // OnClick actions here
    // Instantiate the transmitter passing the output stream and text to it
    if(networkConnectionAndReceiver.getOutputStream() != null) { // Check that output stream has be setup
        Transmitter transmitter = new Transmitter(networkConnectionAndReceiver.getOutputStream(), transmitterString: msg+message);
        transmitter.start(); // Run on its own thread
    }
});
```

4-3 code added for implementing 'MSG' button

4.2 Test

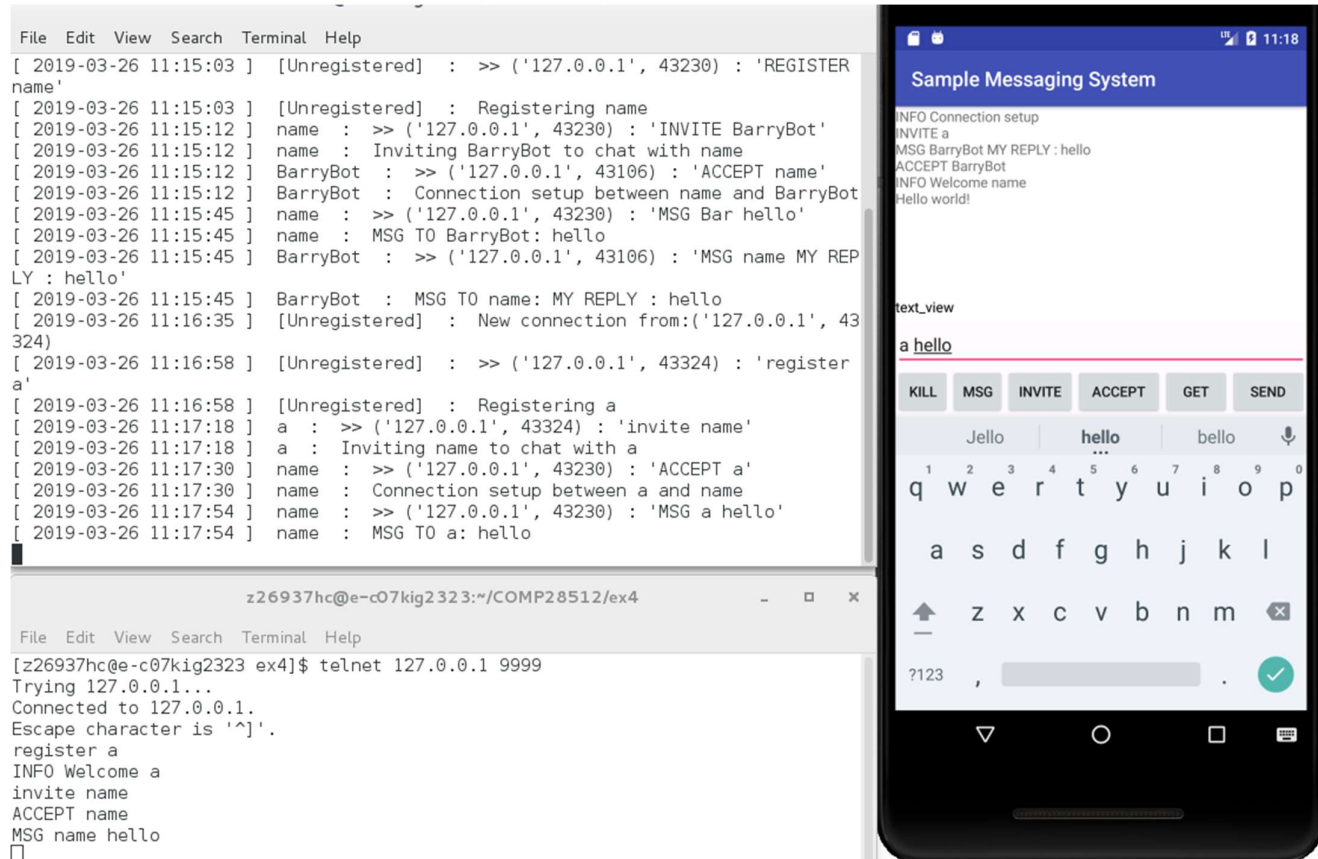
After registration, I invited BarryBot and expect that a connection is set between me and BarryBot, since it can accept invitation automatically. Relevant information shall appear at app message box, proxy-server and BarryBot client, a connection is set up between BarryBot and me means that button 'INVITE' is working properly.

After a connection is set up, I sent message to BarryBot and observed both proxy-server and app client to see if respond from BarryBot can be shown on app client, successfully showing message means that button 'MSG' is working properly.

Then I used telnet client to register another user with name 'a' to proxy-server and try to send an invitation to user

'name' that is controlled by mobile app. Click 'ACCEPT' button to accept the invitation, doing this successfully means that button 'ACCEPT' is working properly.

The whole procedure of testing these three buttons is shown in pic 4-4.



4-4 testing procedure for part 4.4

4.3 Evaluation

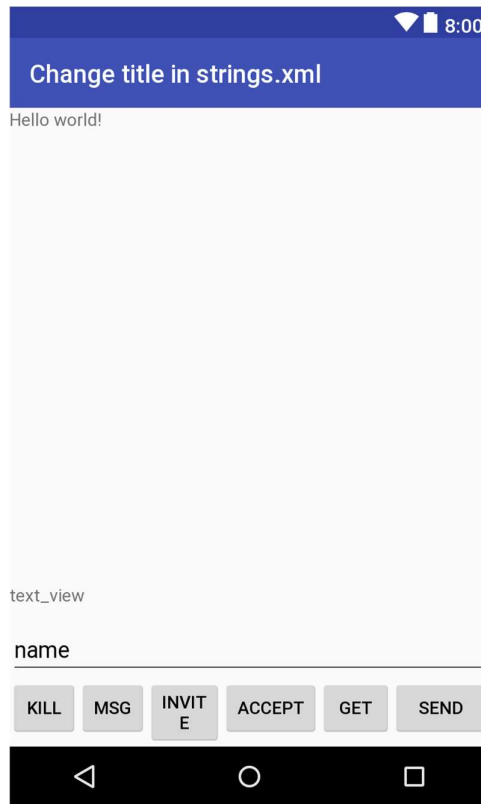
Basically, all possible functions for a chatting procedure on a normal messaging system is implemented by the end of this stage, yet compare to a mature app like Facebook Messenger, this sample app is still rather primitive: User need to register, get online list, invite and accept by clicking button, which shall be done automatically when clicking a user from online list, and online list is expected to be updated real-time. Also, in a mature messaging system, we also don't need to choose a person to send message in chatting page, which is not implemented in this version.

4.4 Q&A

1. What are the main features of your new layout and how do they work?

A: I give an individual line for Plain Text bar and space for displaying message from server is slightly less. As three new functions have been implemented, I put buttons for them in the line for buttons that already existed.

Different commands for different buttons are already provided in onClick function for each of them respectively, yet we still need something from Plain Text bar for parameters of these command, so we type them in and click corresponding button and message in Text View can get us know whether or operation is valid.



4-5 application layout at current stage

2. What are the possible disadvantages of using the proxy-server's MSG command to convey the communications to and from clients, especially if you are thinking about introducing spoken messages and multimedia?

A: we have to explicitly point out receiver of this message, yet for spoken message this gives extra inconvenience by first saying whom the information is sent to; for multimedia it will be difficult to add information of receiver.

5. Part 4.5

5.1 Development

In this step I mainly focused on these improvements:

1. Activate the second TextView bar for online list, the list is parsed into usernames to replace the raw message we get directly from proxy-server. Previous record of the list is cleared when updating.
2. Use a spinner module in UI to choose an online user to MSG/INVITE/ACCEPT, this will help reduce error of typo of target user when using textbox for username input. List in spinner is updated along with the online list above every time 'GET' is clicked.
3. Changed Layout and color of TextView bar on UI page.
4. Disconnect current user from proxy-server first before quitting when clicking 'KILL', so that we can use the same username to register when enter the app next time and get rid of error shown in pic 2-4.

In NetworkConnectionsAndReceiver thread, I added code for a spinner, a flag for choosing the TextView when clicking a button and parsed the raw message into usernames and send them to spinner and the new TextView respectively. Adding an extra view for choosing requires constructor of NetworkConnectionsAndReceiver thread to change correspondingly. Relevant Code are shown in pic 5-1 and 5-2.

```
public boolean flag=false;
private List<String> data_list;
private ArrayAdapter<String> arr_adapter;
private Spinner spinner;
private String online;

//class constructor
public NetworkConnectionAndReceiver(AppCompatActivity parentRef, int viewID, int supportID)
{
    this.parentRef=parentRef; // Get reference to UI
    this.viewID = viewID;
    this.supportID=supportID;
}
```

5-1 adding spinner and flag , changing constructor

```
parentRef.runOnUiThread(() -> {
    // Display message, and old text in the receiving text area
    if(flag==false)
        receiverDisplay.setText(message + "\n" + receiverDisplay.getText());
    message = message.substring(6,message.length()-2);
    Log.v(LOGTAG, msg: "before split"+message);
    if(flag==true){
        online = "Currently Online:\n";
        spinner = parentRef.findViewById(R.id.spinnerOnline);
        Log.v(LOGTAG, msg: "after split");
        String temp[] = message.split( regex: ",", "");
        data_list = Arrays.asList(temp);
        for(String str:data_list)
            Log.v(LOGTAG, str);
        arr_adapter = new ArrayAdapter<String>(parentRef,R.layout.support_simple_spinner_dropdown_item,data_list);
        arr_adapter.setDropDownViewResource(R.layout.support_simple_spinner_dropdown_item);
        spinner.setAdapter(arr_adapter);

        for (String str:data_list){
            online = online.concat(str+'\n');
        }
        receiverDisplay.setText("");
        receiverDisplay.setText(online + "\n" + receiverDisplay.getText());
    }
});
```

5-2 parsing raw message of online list and send it to spinner and TextView

In UI thread, declaration of spinner module is also added (in pic 5-3) and flag for changing TextView in NetworkConnectionsAndReceiver thread is changed in each button clicking. For 'GET' button, new TextView is used and flag is set to true, as shown in pic 5-4; for all other button except 'KILL', I used original TextView and flag is set to false, as shown in pic 5-5.

```
Log.i(LOGTAG, msg: "Starting task4 app"); // Report to Logcat
spinner = findViewById(R.id.spinnerOnline);
TextView buddyReceiver = findViewById(R.id.onLineBuddyView);
buddyReceiver.setMovementMethod(new ScrollingMovementMethod());
```

5-3 spinner declaration in UI thread

```
getButton.setOnClickListener((v) -> {
    networkConnectionAndReceiver.setFlag(true);
    Transmitter transmitter = new Transmitter(networkConnectionAndReceiver.getOutputStream(), transmitterString: "who");
    transmitter.start(); // Run on its own thread
});
```

5-4 flag is set to true when clicking 'GET'

```
if(networkConnectionAndReceiver.getOutputStream() != null) { // Check that output stream has be setup
    networkConnectionAndReceiver.setFlag(false);
    Transmitter transmitter = new Transmitter(networkConnectionAndReceiver.getOutputStream(), transmitterString: register+message);
    transmitter.start(); // Run on its own thread
}
```

5-5 flag is set to false otherwise (except KILL)

For safe quit, I send 'DISCONNECT' to proxy-server first before quit app, since Transmitter thread cannot run immediately, I added a 50ms delay to it before finally quitting the app, as shown in pic 5-6:

```
killButton.setOnClickListener((v) -> {
    // OnClick actions here
    Transmitter transmitter = new Transmitter(networkConnectionAndReceiver.getOutputStream(), transmitterString: "DISCONNECT");
    Log.i(LOGTAG, msg: "disconnect"); // Report to Logcat
    transmitter.start(); // Run on its own thread
    try {
        // thread to sleep for 1000 milliseconds
        Thread.sleep( millis: 50);
    } catch (Exception e) {
        System.out.println(e);
    }
    // Exit app
    System.exit( status: 0);
});
```

5-6 safe quit by sending 'DISCONNECT' before quit

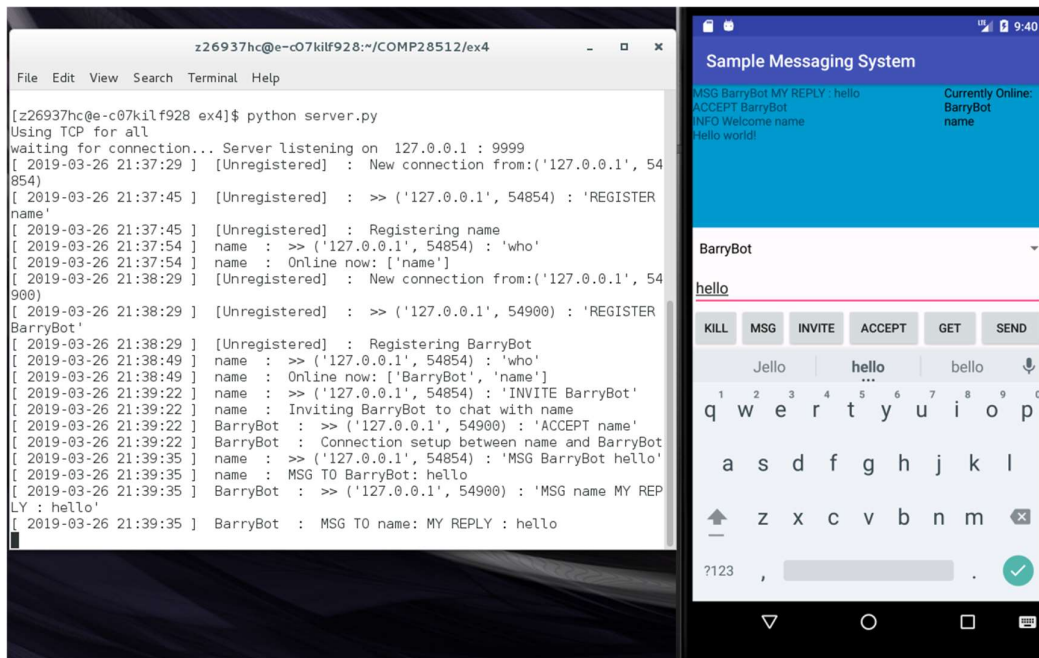
Final layout of sample app is shown in pic 5-7:



5-7 final application layout

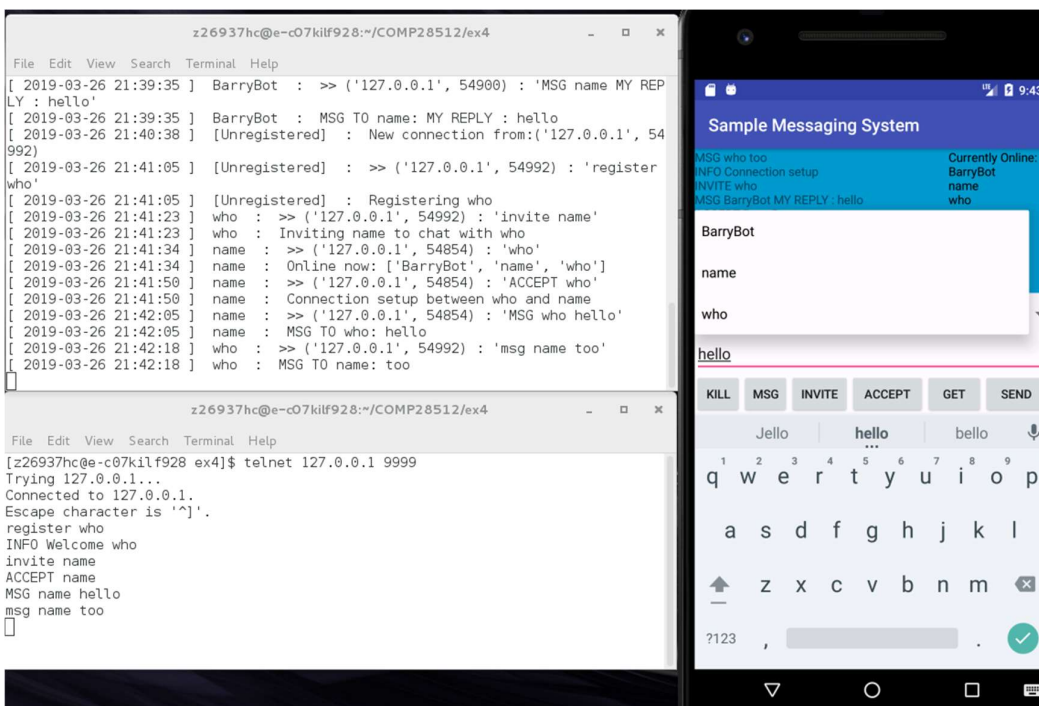
5.2 Test

First I tested if parsing can be done correctly, spinner and 'INVITE' and 'SEND' work well with spinner. After registration and starting BarryBot, I get online list and see if it appears in spinner and TextView. Then I invite Barry with 'BarryBot' selected in spinner and send a message to them. I observe that all the stuffs work properly, testing procedure to current state is shown in pic 5-8.



5-8 test record of procedure above

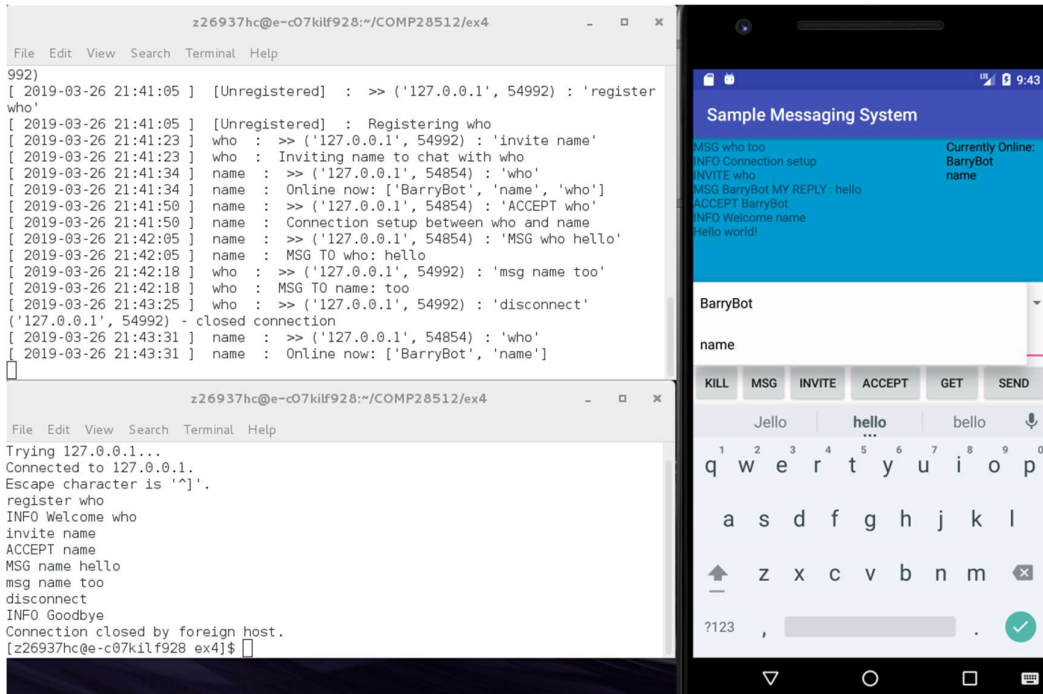
Then I register another user using telnet client and send invitation to user 'name', updating online list at this point shall see newly registered user 'who' appears in both TextView and spinner. Choose 'who' from spinner and click 'ACCEPT' shall set up a link between app and telnet client. A two-way communication test is then done to examine if it works properly. Procedure of test above is recorded in pic 5-9.



5-9 test record of procedure above

Next I disconnect user 'who' and update online user list, we can see it is updated in TextView and spinner, as shown

in pic 5-10. Through out the whole procedure, I observed that list in TextView and spinner will change accordingly with current state every time and is always right, also, layout and color of TextView is changed, meaning that first three improvements is working well.



5-10 updating online user list

Finally, I click 'KILL' and observe message given in proxy-server, no error appears anymore and 'DISCONNECT' is executed before quitting app, and when I enter the app again and use the same username to register, I succeeded, as shown in pic 5-11, meaning that the last improvement is working correctly.

```
[ 2019-03-27 16:47:11 ] [Unregistered] : Registering name
[ 2019-03-27 16:47:12 ] name : >> ('127.0.0.1', 49406) : 'who'
[ 2019-03-27 16:47:12 ] name : Online now: ['name']
[ 2019-03-27 16:47:14 ] name : >> ('127.0.0.1', 49406) : 'DISCONNECT'
('127.0.0.1', 49406) - closed connection
[ 2019-03-27 16:47:21 ] [Unregistered] : New connection from:('127.0.0.1', 49436)
[ 2019-03-27 16:47:24 ] [Unregistered] : >> ('127.0.0.1', 49436) : 'REGISTER name'
[ 2019-03-27 16:47:24 ] [Unregistered] : Registering name
[ 2019-03-27 16:47:25 ] name : >> ('127.0.0.1', 49436) : 'who'
[ 2019-03-27 16:47:25 ] name : Online now: ['name']
[ 2019-03-27 16:47:27 ] name : >> ('127.0.0.1', 49436) : 'DISCONNECT'
('127.0.0.1', 49436) - closed connection
[ 2019-03-27 16:47:41 ] [Unregistered] : New connection from:('127.0.0.1', 49452)
[ 2019-03-27 16:47:44 ] [Unregistered] : >> ('127.0.0.1', 49452) : 'REGISTER name'
[ 2019-03-27 16:47:44 ] [Unregistered] : Registering name
[ 2019-03-27 16:47:45 ] name : >> ('127.0.0.1', 49452) : 'who'
[ 2019-03-27 16:47:45 ] name : Online now: ['name']
[ 2019-03-27 16:47:47 ] name : >> ('127.0.0.1', 49452) : 'DISCONNECT'
('127.0.0.1', 49452) - closed connection
```

5-11 safe quit

5.3 Evaluation

Currently the sample messaging system is more colorful and safer with message given a bit more reasonable. Also, text messages needed for user to input is shorter, making it more user-friendly, yet there it still way to go to make this sample messaging system resemble a real client.

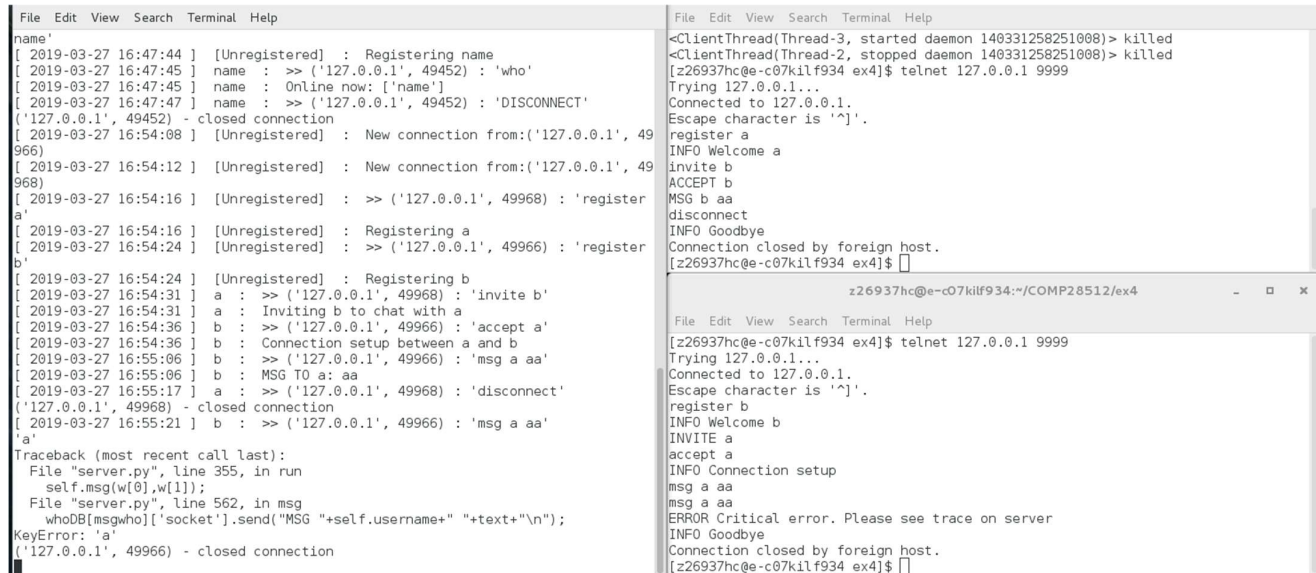
5.4 Q&A

1. What are your improvements?

A: They are written in 5.1.

2. Are there any deficiencies in the protocol as currently implemented by the proxy-server?

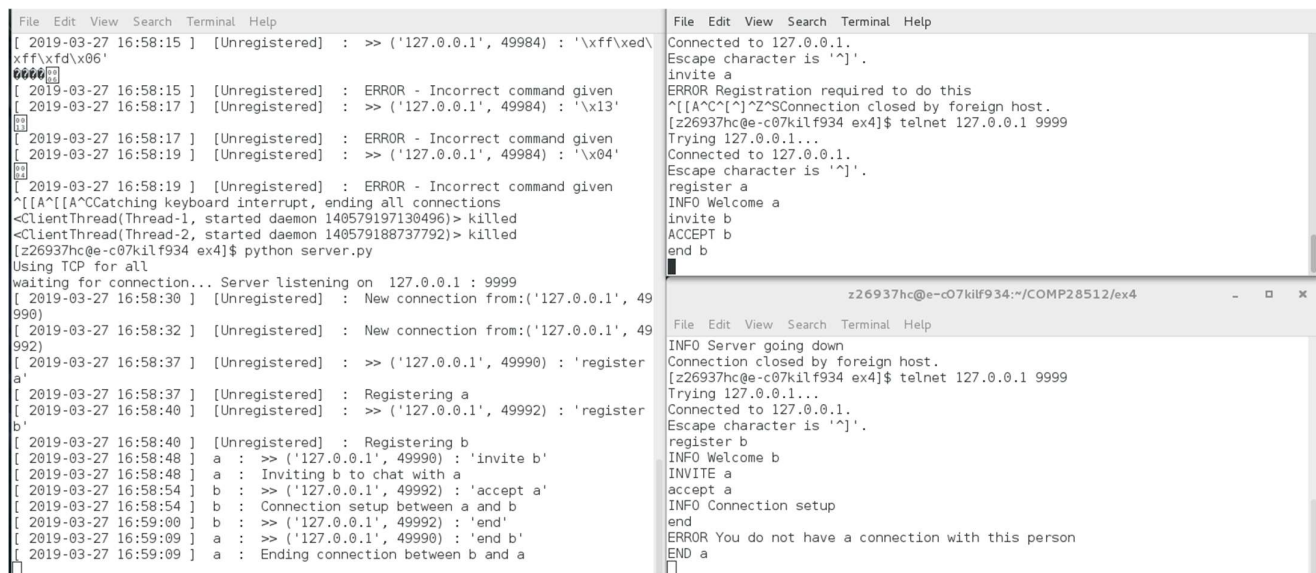
A: 1. The server is still not safe. If a connection is set between 2 users and one user force quit the client(maybe losing network connection), the server cannot handle this and gives an error, as shown in pic 5-12; also, feedback from proxy-server to client is not sufficient, take END command for example, if one user use END to end connection, indeed proxy-server and the other user can know this, yet the user who executed this command doesn't know whether he/she has ended connection successfully, as shown in pic 5-13; also, implementation for msg is not suitable for vocal input or multimedia, as stated in the second question of part 4.4.



```
File Edit View Search Terminal Help
[ 2019-03-27 16:47:44 ] [Unregistered] : Registering name
[ 2019-03-27 16:47:45 ] name : >> ('127.0.0.1', 49452) : 'who'
[ 2019-03-27 16:47:45 ] name : Online now: ['name']
[ 2019-03-27 16:47:47 ] name : >> ('127.0.0.1', 49452) : 'DISCONNECT'
('127.0.0.1', 49452) - closed connection
[ 2019-03-27 16:54:08 ] [Unregistered] : New connection from:('127.0.0.1', 49966)
[ 2019-03-27 16:54:12 ] [Unregistered] : New connection from:('127.0.0.1', 49968)
[ 2019-03-27 16:54:16 ] [Unregistered] : >> ('127.0.0.1', 49968) : 'register a'
[ 2019-03-27 16:54:16 ] [Unregistered] : Registering a
[ 2019-03-27 16:54:24 ] [Unregistered] : >> ('127.0.0.1', 49966) : 'register b'
[ 2019-03-27 16:54:24 ] [Unregistered] : Registering b
[ 2019-03-27 16:54:31 ] a : >> ('127.0.0.1', 49968) : 'invite b'
[ 2019-03-27 16:54:31 ] a : Inviting b to chat with a
[ 2019-03-27 16:54:36 ] b : >> ('127.0.0.1', 49966) : 'accept a'
[ 2019-03-27 16:54:36 ] b : Connection setup between a and b
[ 2019-03-27 16:55:06 ] b : >> ('127.0.0.1', 49966) : 'msg a aa'
[ 2019-03-27 16:55:06 ] b : MSG T0 a: aa
[ 2019-03-27 16:55:17 ] a : >> ('127.0.0.1', 49968) : 'disconnect'
('127.0.0.1', 49968) - closed connection
[ 2019-03-27 16:55:21 ] b : >> ('127.0.0.1', 49966) : 'msg a aa'
'a'
Traceback (most recent call last):
  File "server.py", line 355, in run
    self.msg(w[0],w[1]);
  File "server.py", line 562, in msg
    whoDB[msgwho]['socket'].send('MSG "+self.username+" "+text+"\n");
KeyError: 'a'
('127.0.0.1', 49966) - closed connection

z26937hc@e-c07kilf934:~/COMP28512/ex4
File Edit View Search Terminal Help
<ClientThread(Thread-3, started daemon 140331258251008)> killed
<ClientThread(Thread-2, stopped daemon 140331258251008)> killed
[z26937hc@e-c07kilf934 ex4]$ telnet 127.0.0.1 9999
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^J'.
register a
INFO Welcome a
invite b
ACCEPT b
MSG b aa
disconnect
INFO Goodbye
Connection closed by foreign host.
[z26937hc@e-c07kilf934 ex4]$
```

5-12 sudden disconnection of a user causes error on server



```
File Edit View Search Terminal Help
[ 2019-03-27 16:58:15 ] [Unregistered] : >> ('127.0.0.1', 49984) : '\xff\xfd\x06'
[ 2019-03-27 16:58:15 ] [Unregistered] : ERROR - Incorrect command given
[ 2019-03-27 16:58:17 ] [Unregistered] : >> ('127.0.0.1', 49984) : '\x13'
[ 2019-03-27 16:58:17 ] [Unregistered] : ERROR - Incorrect command given
[ 2019-03-27 16:58:19 ] [Unregistered] : >> ('127.0.0.1', 49984) : '\x04'
[ 2019-03-27 16:58:19 ] [Unregistered] : ERROR - Incorrect command given
^[[A^[[A^Catching keyboard interrupt, ending all connections
<ClientThread(Thread-1, started daemon 140579197130496)> killed
<ClientThread(Thread-2, started daemon 140579188737792)> killed
[z26937hc@e-c07kilf934 ex4]$ python server.py
Using TCP for all
waiting for connection... Server listening on 127.0.0.1 : 9999
[ 2019-03-27 16:58:30 ] [Unregistered] : New connection from:('127.0.0.1', 49990)
[ 2019-03-27 16:58:32 ] [Unregistered] : New connection from:('127.0.0.1', 49992)
[ 2019-03-27 16:58:37 ] [Unregistered] : >> ('127.0.0.1', 49990) : 'register a'
[ 2019-03-27 16:58:37 ] [Unregistered] : Registering a
[ 2019-03-27 16:58:40 ] [Unregistered] : >> ('127.0.0.1', 49992) : 'register b'
[ 2019-03-27 16:58:40 ] [Unregistered] : Registering b
[ 2019-03-27 16:58:48 ] a : >> ('127.0.0.1', 49990) : 'invite b'
[ 2019-03-27 16:58:48 ] a : Inviting b to chat with a
[ 2019-03-27 16:58:54 ] b : >> ('127.0.0.1', 49992) : 'accept a'
[ 2019-03-27 16:58:54 ] b : Connection setup between a and b
[ 2019-03-27 16:59:00 ] b : >> ('127.0.0.1', 49992) : 'end b'
[ 2019-03-27 16:59:09 ] a : >> ('127.0.0.1', 49990) : 'end a'
[ 2019-03-27 16:59:09 ] a : Ending connection between b and a

z26937hc@e-c07kilf934:~/COMP28512/ex4
File Edit View Search Terminal Help
Connected to 127.0.0.1.
Escape character is '^J'.
invite a
ERROR Registration required to do this
^[[A^[[A^Z^SConnection closed by foreign host.
[z26937hc@e-c07kilf934 ex4]$ telnet 127.0.0.1 9999
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^J'.
register a
INFO Welcome a
invite b
ACCEPT b
end b
INFO Server going down
Connection closed by foreign host.
[z26937hc@e-c07kilf934 ex4]$ telnet 127.0.0.1 9999
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^J'.
register b
INFO Welcome b
INVITE a
accept a
INFO Connection setup
end
ERROR You do not have a connection with this person
END a
```

5-13 insufficient response(connection ended by user a)

3. How could the proxy-server be improved?

A: Implement error handling for errors like that demonstrated in the previous question to increase robustness and user friendliness; give messages on whether a command is executed successfully or not to both(maybe all)users, giving client the right to choose whether these messages shall be all shown to user or not; improve implementation of messaging.