JSON Contact Data:

- List of “Contact” objects

Contact

- Name (must be unique, case sensitive)
- Status (online true : offline false)

Contacts (“data”) or <contact data> will be delimited strings as the following:
<username>\0<status>\n<username>\0<status>\n<username>\0<status>

IE:

“user1\0online\nuser2\0offline\nuser3\0online”

Room History (when logging in) will be delimited strings as the folowing:
<room>\0<user>:<message>\n<user>:<message>\n<user>:<message>\0\0<room>\0<user>:<message>\n<user>:<message>

room histories separated by \0 (split this first)

<room history 1>\0<room history 2>\0<room history 3...>

history data is split by \0, separating room id and the chat messages

<roomid>\0<message history>

messages are split by \n (and split by : between user and message, but this parsing isn’t necessary)

<user>:<message>\n<user>:<message>

Therefore, the large and complex example below:

00a1\0user1: Hi there! How are you?\nuser2: I am good, how about you?\nuser3: What about me???\055a2\0user1: this is the beginning of a new chat history\nuser2: yes, yes it is, user 3 isn’t here now\nuser4: But I am here!\0442a\0user1: im alone

Connect to the Chat Server

wss.AddWebSocketService<ClientCommunication>("/chatserver");

Located at /chatserver

Transmission data will always be a Dictionary<string, string> encoded in JSON.
Dictionaries do not include the quotation marks surrounding the data below.

Chat room id’s are just called “room”

API:

Client Sends...	**Server to client...**	**Server to others...**	**Notes**
“request”:”login” “user”:”johndoe” “password”:”abc123”	“request”:”login” “result”:”false”	N/A	username doesn’t exist or password is wrong or already logged in
	“request”:”login” “result”:”true”	“request”:”contactupdate” “data”:<contact data>	correct information

	“data”:<contact data> “roomhistory”:”00a1\0name 1: hi there\nname2: oh hello there\055b2\0user1: im here\nuser2: so am i”		See above the API table for how the room history is delimited.
“request”:”signup” “user”:”johndoe” “password”:”abc123”	“request”:”signup” “result”:”false”	N/A	user name already exists
	“request”:”signup” “result”:”true”	N/A	successfully created new user - user will be logged in No info to others because new user - no friends yet
“request”:”addcontact” “user”:”friendname”	“request”:”addcontact” “result”:”false”	N/A	user does not exist or is already a contact friend
	“request”:”addcontact” “result”:”true” “data”:<contact data>	N/A	we could notify the user that was added that “johndoe” added him as a friend - maybe?
“request”:”removecontact” “user”:”enemyname”	“request”:”removecontact” “result”:”false”	N/A	user wasn’t a friend or doesn’t exist
	“request”:”removecontact” “result”:”true” “data”:<contact data>	N/A	
“request”:”logout”	“request”:”logout” “result”:”true”	“request”:”contactupdate” “data”:<contact data>	New contact data for all other users incase they knew the logged out user - their status needs updated No need to send anything to the logged out user - they are gone
“request”:”leavechat” “room”:”00a1”	“request”:”chatuserleft” “room”:”00a1” “user”:”johndoe” “validusers”:”name1\0name2\0name3\0...\0...”		Remove self from chat session, everyone else can keep talking. Tell everyone else that johndoe left. Note: if johndoe was the 2nd to last person in the chat, the last person should be notified that everybody left, and their chat window should close. Room 00a1 would be disbanded in this case. The valid users is the list of users that can be added to the chat (mutual friends of everybody in the room).
“request”:”startchat” “user”:”myfriend”	“request”:”startchat” “result”:”true” “room”:”00a1” “users”:”johndoe\0myfriend” “validusers”:”name1\0name2\0name3\0...\0...” “history”:””		Create new chat session - opens on both ends. Both users notified of a “startchat” with the other person listed under user. User can be offline. The valid users is the list of users that can be added to the chat (mutual friends of everybody in the room).

			History is always blank (here at least).
	“request”:”startchat” “result”:”false”	N/A	The given user wasn’t a two-way friend, you can’t start a chat with them :P. Or use didn’t exist.
“request”:”sendchat” “room”:”00a1” “message”:”Hello friends!”	“request”:”receivechat” “room”:”00a1” “user”:”johndoe” “message”:”Hello friends!”		Message sent to everyone. Client who sent the message only displays it from the server’s response (incase they get disconnected, sending messages should not show up if you aren’t connected to the server)
“request”:”addtochat” “room”:”00a1” “user”:”friendtwo”	“request”:”addtochat” “room”:”00a1” “user”:”friendtwo” “result”:”false”	N/A	User was not on everybody’s friend list. This usually will not show up because the client should only display users to add that are in the same group as everyone in the room.
	[*to all people in the current chat*] “request”:”chatuserjoined” “room”:”00a1” “user”:”friendtwo” “validusers”:”name1\0name2\0name3\0...\0...” [*to the added user*] “request”:”startchat” “room”:”00a1” “users”:”johndoe\0myfriend\0friendtwo” “validusers”:”name1\0name2\0name3\0...\0...” “history”:”name1: Hi there\nname2: oh hello there”		~~First block is only sent to the sender. This is just a confirmation message. You shouldn’t do anything with it other than that.~~ Second block is sent to all people in the current room. Third block is sent to the added user (telling them to join the room). The valid users is the list of users that can be added to the chat (mutual friends of everybody in the room).
[internal only, do not use in client] *“request”:”connect”*	N/A	N/A	message sent to server (from the server) when the connection is opened but not yet logged in

Client side

```
String message = “login “ + username + “ “ + password;
server.sendData(message.ToJSON());        //maybe?

Void receiveData(String serverResponse) {
        if(serverResponse.startsWith(“correct”)) {
                //logged in correctly
        }
}
```

Server side

```
String receiveData(String clientMessage) {
        if(clientMessage.startsWith(“login”) {
                //check if the login info is correct
                if(correct)
                        Return “correct”;
                Else
                        Return “incorrect”;
        } else ….
}
```

A and B must both know C to add them to the chat
All must know D for C to add them to the chat

A and B must know each other to create a chat

“A: hithere\nB: hello\n”

//Determine the delimiters
Delimiter ‘\n’ - cannot be in a username

Dictionary<string, string> message;
JSON.encode(message);

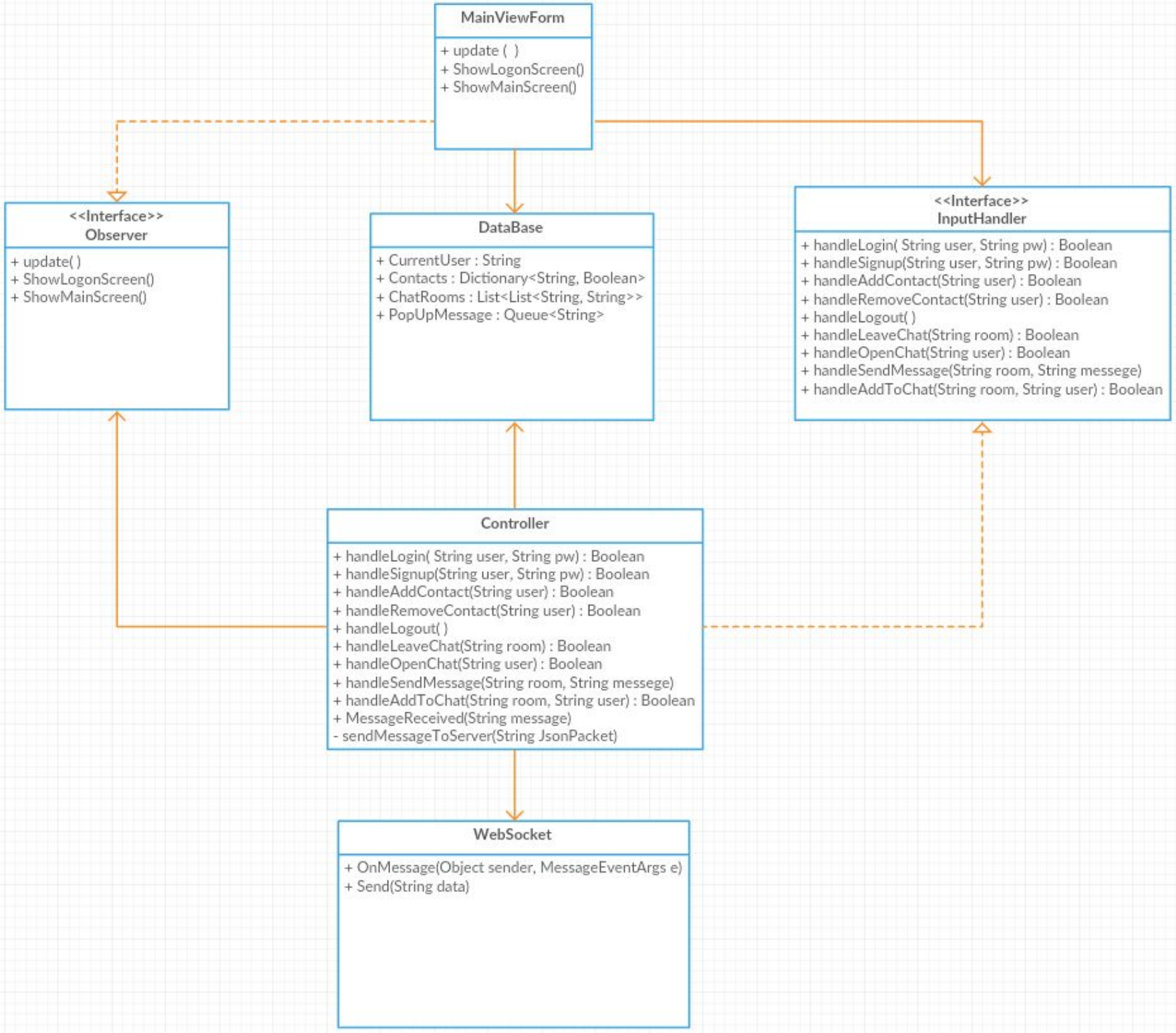
{action:”StartChat”,ChatId:”chatID”,user:”username”}

Dictionary<string, string> message = JSON.decode(e.Data);

Client Use Case Realisations

Add new contact
*User clicks button requesting to add new contact.
*'Add new contact' form is displayed, requesting name of new contact.
*User enters username of contact they wish to add, then clicks 'Ok' button.
*Username is sent to the controller for validation.
*If username is valid, the controller creates a jsop packet containing the username and sends it to the websocket
*Websocket sends the json packet to the server application.
*The server responds with a json packet indicating whether the contact was added.
*The controller unpacks the json packet to identify the result.
*If the packet indicates that the request was successful, the controller updates the contact data in the client's database.
*The controller notifies the client view form that the contact was added so that it can display the information to the user.

Logout
*User clicks the 'logout' button.
*View form tells the controller to send logout request.
*Controller creates json packet to notify server of logout.
*Controller sends packet to Websocket to send to server.
*Controller clears contact data from client database.
*Controller notifies main client view form to close and opens login form.


MainViewForm
+ update ( )
+ ShowLogonScreen()
+ ShowMainScreen()
<<Interface>>
Observer
+ update( )
+ ShowLogonScreen()
+ ShowMainScreen()
DataBase
+ CurrentUser : String
+ Contacts : Dictionary<String, Boolean>
+ ChatRooms : List<List<String, String>>
+ PopUpMessage : Queue<String>
<<Interface>>
InputHandler
+ handleLogin(String user, String pw) : Boolean
+ handleSignup(String user, String pw) : Boolean
+ handleAddContact(String user) : Boolean
+ handleRemoveContact(String user) : Boolean
+ handleLogout( )
+ handleLeaveChat(String room) : Boolean
+ handleOpenChat(String user) : Boolean
+ handleSendMessage(String room, String messege)
+ handleAddToChat(String room, String user) : Boolean
Controller
+ handleLogin(String user, String pw) : Boolean
+ handleSignup(String user, String pw) : Boolean
+ handleAddContact(String user) : Boolean
+ handleRemoveContact(String user) : Boolean
+ handleLogout( )
+ handleLeaveChat(String room) : Boolean
+ handleOpenChat(String user) : Boolean
+ handleSendMessage(String room, String messege)
+ handleAddToChat(String room, String user) : Boolean
+ MessageReceived(String message)
- sendMessageToServer(String JsonPacket)
WebSocket
+ OnMessage(Object sender, MessageEventArgs e)
+ Send(String data)