``

ZOTCOMM

CONNECTING THE WORLD.

QuickChat

VERSION 0.1.1A

TECHNICAL MANUAL

TEAM MEMBERS
Mehdi Lakhoua
Yuki Hayashi
Ye Myat Kyaw
David Tiao
Rijul Arora

UCI EECS 22L
WINTER 2018
TEAM 17

Table of Contents

Glossary... 2

1. Client Software Architecture Overview

1.1 Main data types and structures.. 3

1.2 Major software components.. 4

1.3 Module interfaces... 5

1.4 Overall program control flow.. 7

2. Server Software Architecture and Overview

2.1 Main data types and structures... 9

2.2 Major software components.. 10

2.3 Major interfaces.. 10

2.4 Overall program control flow.. 11

When Register/username/password……………………………………… 13

When the server receives “ADD/username”... 14

When the server receives “REMOVE/username”....................................... 15

When the server receives “ACCEPT_REQUEST/USERNAME”................. 16

When the server receives “IGNORE_REQUEST?USERNAME”................ 17

When the server receives “REQUEST_CONTACTS/username”................ 18

When the server receives “REQUEST_FRIEND_REQUESTS/username”.19

3. Installation

3.1 System requirements, compatibility.. 20

3.2 Setup and configuration.. 20

3.3 Building, compilation, installation... 20

4. Documentation of packages, modules, interfaces

4.1 Detailed description of data structures... 22

4.2 Detailed description of functions and parameters.. 23

4.3 Detailed description of the communication protocol.. 31

5. Development plan and timeline

5.1 Partitioning of tasks.. 33

5.2 Team Member responsibilities.. 33

Back Matter

Copyright... 33

References.. 34

Index.. 34

Glossary

Username: The user's custom name that is displayed on the chat client which cannot be the same as other client's username

Password: The user's custom password that is used to login the account

Contact: A member of the contact list of a user.

Status: An icon that shows whether a user is offline or online.

Login: The act of logging in to the account

Account: The right to log into the chat application server.

Registration: The action to create an account in the Chat Application.

Contact List: A list that shows all the added contacts of a user.

Friend Request: A request to add a person into the contact list in order to chat with that person.

Server: Provides a service function (centralized resource) to one or more clients.

Client: Computer hardware or software that accesses a service made available by a server.

Internet: A global communication network used to exchange data using communication protocols.

User: A person who operates something, in this case, the chat application.

Window: A graphical control element that consists of a visual area containing buttons, data entry boxes, etc.

Protocol: A method of exchanging data over the network.

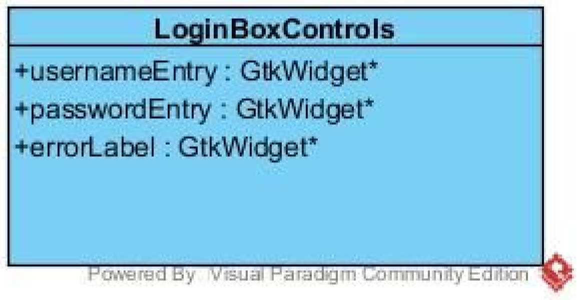
TCP: Transmission Control Protocol is a set of rules that is used to interconnect network devices on the internet.

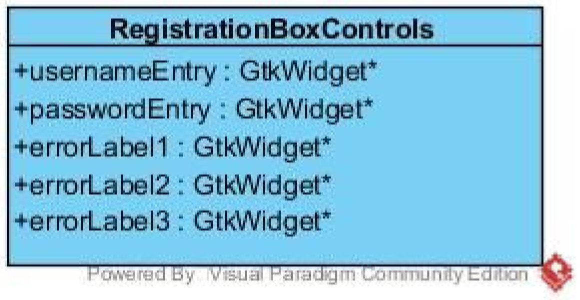
Peer to Peer: P2P is a distributed application architecture that allows peers to communicati with each other.

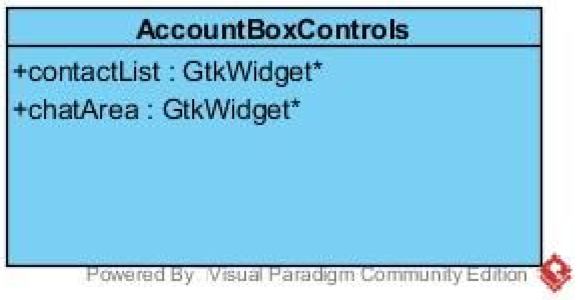
Internet Protocol: A set of rules governing the format of data sent over the network.

1. Client Software Architecture Overview

1.1 Main Data Types and Structure




RegistrationBoxControls
+usernameEntry : GtkWidget*
+passwordEntry : GtkWidget*
+errorLabel1 : GtkWidget*
+errorLabel2 : GtkWidget*
+errorLabel3 : GtkWidget*



These three structs are used to hold any widgets that need to be passed to callback functions when a button is clicked. They allow us to update the client GUI from within the functions.

- When the “Login” button is clicked in the login window, we will pass a pointer to the LoginBoxControls to the callback function that will ask the server to verify the information and allow the login. The function will also be able to update the error label in the login window.

- When the “Confirm” button is clicked in the registration window, we will pass a pointer to the RegistrationBoxControls to the callback function that will ask the server to verify the information and create the account. The function will also be able to update the three error labels in the registration window.

- When a username in the contact list is clicked to open a new chat conversation, a friend is added, or there is a change to the contact list of a user or their status,
  We will pass a pointer to the AccountBoxControls to the appropriate function.

The function will be able to update the chat area with a new conversation, and update the contact list and status of the contacts.

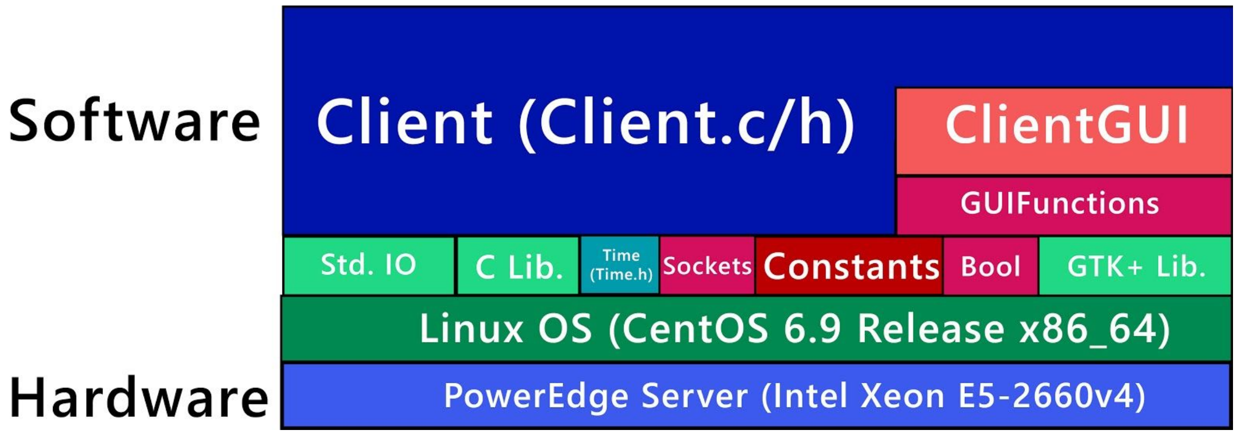
TypeDefs

You may encounter the following typedefs in the source code.

typedef struct LoginBoxControls LBC;
typedef struct RegistrationBoxControls RBC;
typedef struct AccountBoxControls ABC;

1.2 Major Software Components

Module	File	Dependencies
Client	Client.c Client.h	ClientGui.h, Constants.h stdlib.h, stdio.h, Time.h, Sockets.h
ClientGui	ClientGui.c ClientGui.h	GuiFunctions.h GTK library, stdbool.h
GuiFunctions	GuiFunctions.c GuiFunctions.h	GTK library
Constants	Constants.h	



1.3 Module Interfaces

Explanation of the program control flow:

<u>Login and Registration control flow</u>

Beginning of the program

The program will first create login and registration GUI then start GUI main loop. If ‘X’ button which is the exit button is clicked, the program will end. The exit button can be clicked anytime to end the program. However, if the exit button is not clicked, the program will display a login menu containing a login button and a register button.

Login_Button_Clicked

If the login button is clicked from the login window after typing the username and password, the password will be encrypted and the program will send a login request to server and wait for the reply. If the server denies the login, the error label will be updated. If the server allows the login, the program will create an account box and request contacts from the server and wait for the reply. Once the contacts are received by the client, the program will update the contact area of the account box and display it on the window. If the logout button in the account box is clicked, the account box will be destroyed and the login menu will be displayed as well.

Register_Button_Clicked

If the register button is clicked from the main menu instead of the login button, the program will display the registration box containing a cancel button and a confirm button. If the cancel button is clicked from the registration menu, the program will go back to displaying the login screen. If the user proceeds to register an account by typing in username, password and confirm password, and pressing the confirm button, the client server will verify if both passwords match or not. If the passwords do not match, error label 3 will be updated. However, if the passwords do match, the password will be encrypted and the client will send the register request to the server and wait for the reply. If the server notices that the username is invalid or taken, error label 1 will be updated. Also if the password is invalid, error label 2 will be updated. Or if there is no error label being updated, the program will display the registration success and it will lead back to the login menu.

<u>Adding and Removing contact, accepting and ignoring friend requests control flow</u>

Add_Remove_Contact Button_Clicked

If “add/remove a contact” button is clicked on the account window, the program will create “add/remove a contact” popup window.

Add_Contact_Button_Clicked

If an username is typed in and the add button is clicked on a popup window, the program will send the Add Contact request to the server and wait for the reply. If the username is invalid or does not exist or is already added or if the request is already sent or received, an error label will be updated depending on the error. If there is no error, the friend request will be sent and the program will display “friend request has been sent.”

Remove_Contact_Button_Clicked

If an username is typed in and the remove button is clicked on a popup window, the program will send the Remove Contact request to the server and wait for the reply. If the username is invalid or not in the contact list, a corresponding error label will be updated. Or if there is no error, the username will be deleted and the program will display “Username has been removed from contacts.”

After all of the actions, the program will request contacts from the server, wait for the reply and update the contact area of the account window.

Friend_Requests_Button_Clicked

If a friend requests button is clicked, the program will create friend requests popup window and send the Get Friend Requests request to the server.

Ignore_Friend_Request_Button_Clicked

If an ignore button is clicked on the popup window, the client will send Ignore Friend Request request to the server and wait for the reply. If there is no request found, “friend request could not be found” will be displayed. If there is no error, the friend request will be ignored

Accept_Friend_Request_Button_Clicked

If an accept button is clicked on the popup window, the client will send Accept Friend Request request to the server and wait for the reply. If the username is already in the contacts, it will send the Ignore Friend Request request to the server, proceed the same and ignore button is clicked. If there is no error, the friend request will be accepted.

After all these actions, the friend request will be removed from the friend request list and it will request contacts from the server and wait for the reply to update the contact area from the account window.

1.4 Overall program control flow

Registration process control flow

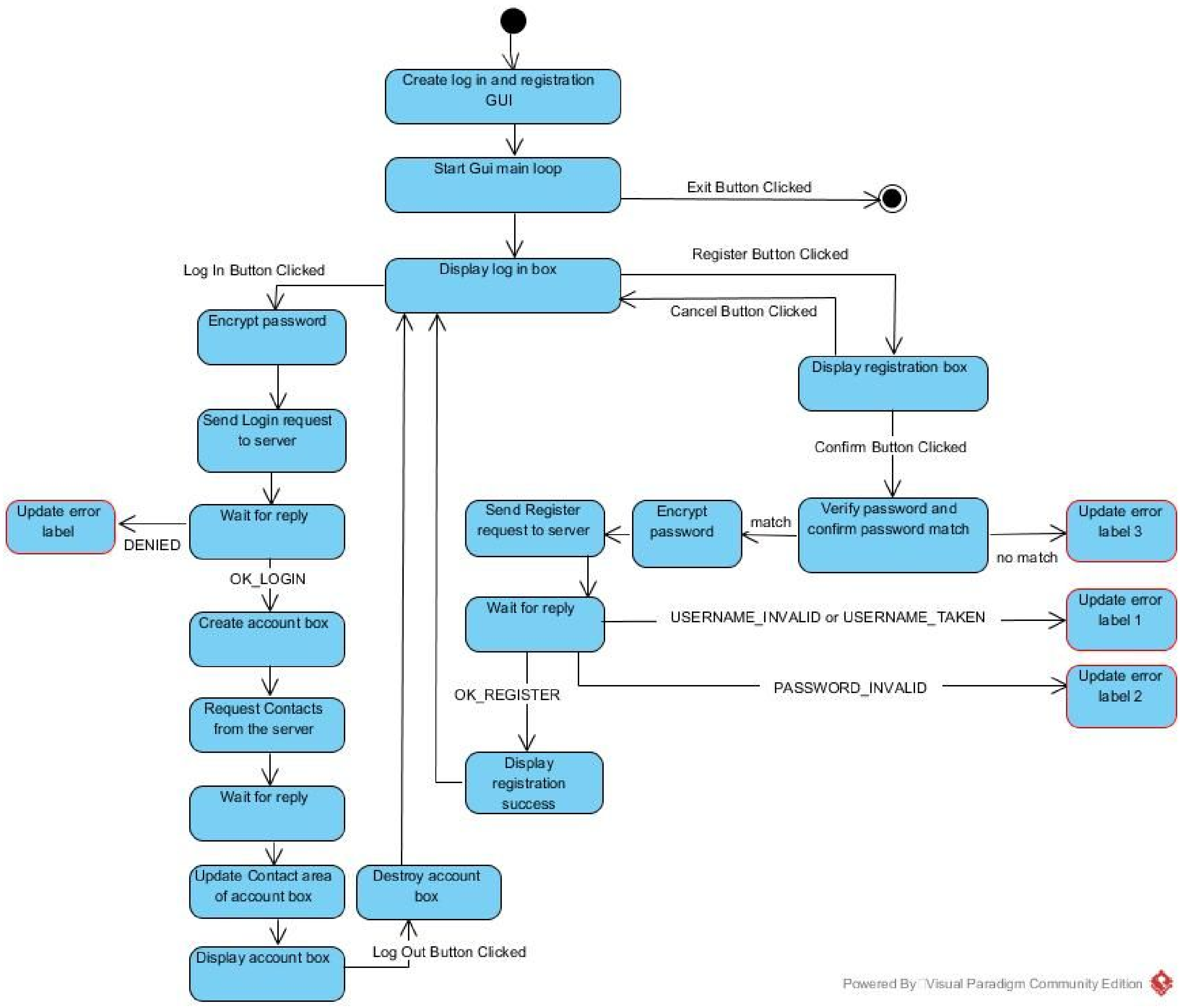
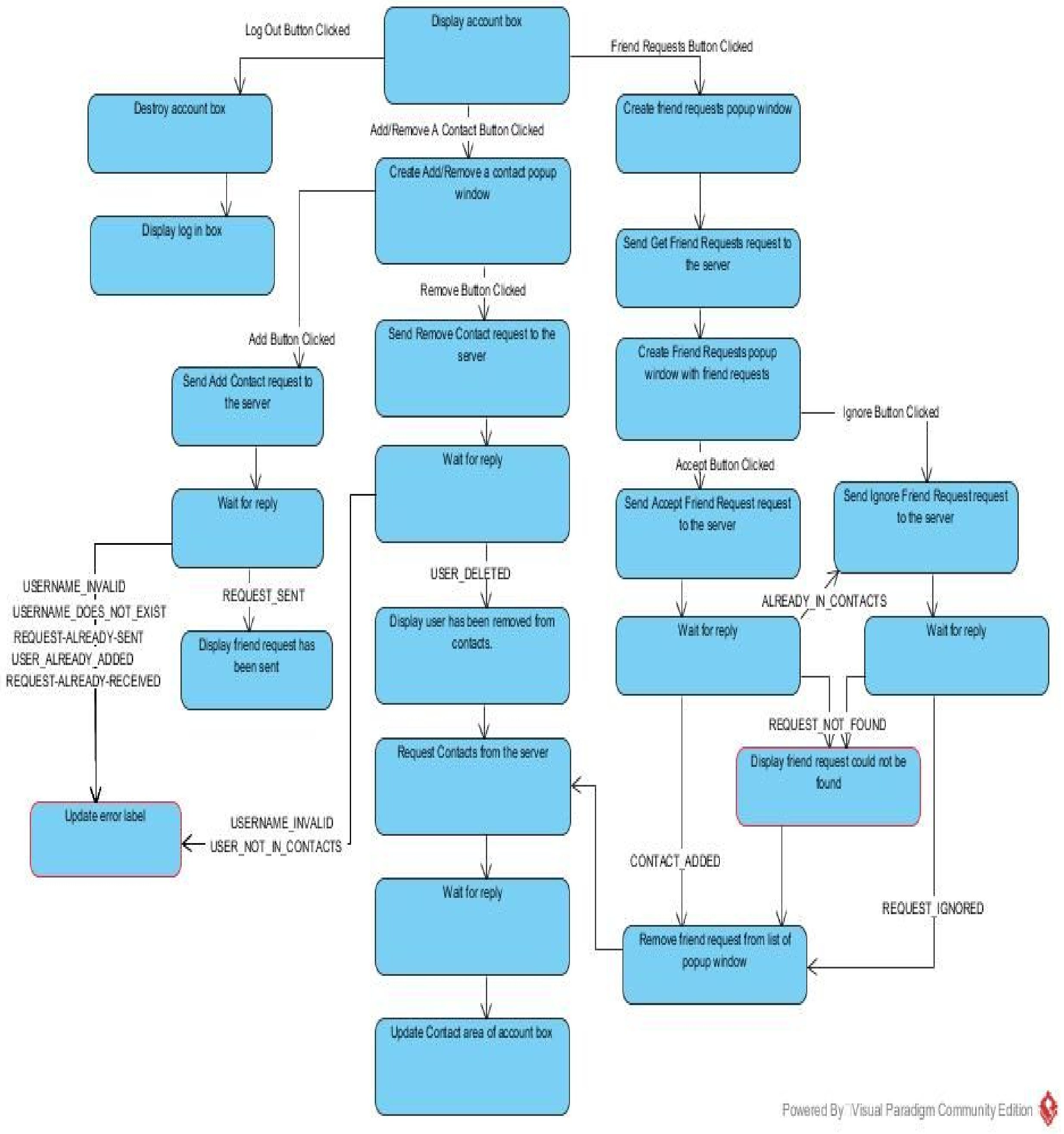


Figure: Registration Control Flow

Adding and Removing contacts, Accepting and Ignoring friend requests control flow


Display account box
Log Out Button Clicked
Friend Requests Button Clicked
Destroy account box
Display log in box
Add/Remove A Contact Button Clicked
Create Add/Remove a contact popup window
Create friend requests popup window
Send Get Friend Requests request to the server
Create Friend Requests popup window with friend requests
Remove Button Clicked
Add Button Clicked
Send Remove Contact request to the server
Send Add Contact request to the server
Wait for reply
Ignore Button Clicked
Accept Button Clicked
Send Accept Friend Request request to the server
Send Ignore Friend Request request to the server
USER_DELETED
USERNAME_INVALID
USERNAME_DOES_NOT_EXIST
REQUEST-ALREADY-SENT
USER_ALREADY_ADDED
REQUEST-ALREADY-RECEIVED
REQUEST_SENT
Display friend request has been sent
Display user has been removed from contacts.
ALREADY_IN_CONTACTS
REQUEST_NOT_FOUND
Display friend request could not be found
Request Contacts from the server
Update error label
USERNAME_INVALID
USER_NOT_IN_CONTACTS
CONTACT_ADDED
REQUEST_IGNORED
Remove friend request from list of popup window
Update Contact area of account box
Powered By Visual Paradigm Community Edition

';**Figure: Adding and Removing contacts, Accepting and Ignoring friend requests control flow**

Chatting with a friend

When the user clicks on the name of a contact, the client will search and load to the chat area any previous conversation if it exits. The notification count will no longer be at the contact list area if unread messages are opened from that contact. Then, the user can type his messages in the chat area and click the send button. Once the send button is clicked, the message will be sent to the contact. When a client receives a message, it will be displayed in the chat area. The user can also chat with other contacts by clicking their names in the contact list, and the chat boxes for other contacts will show up on the tab in the chat area and the client will do the same procedure.

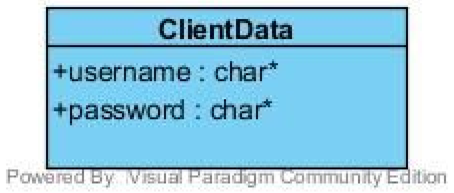
2. Server Software Architecture and Overview

2.1 Main data types and structures

The main data type to be used in the server program is a socket. A socket allows communication between different computers through the use of Unix file descriptors, which are unsigned integers associated with an open file. In QuickChat, the socket will use TCP to guarantee delivery in a network in the same order the data was sent, else the sender will receive an error.

ClientData

Stores the client's username and password in ClientData.



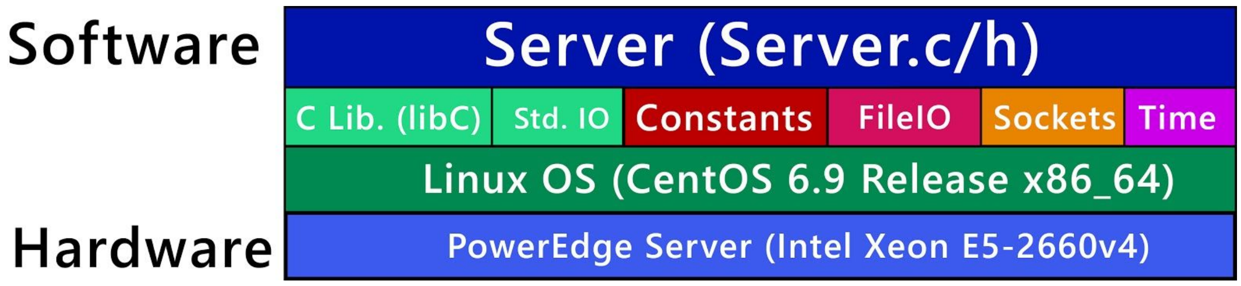
TypeDefs

You may encounter the following typedefs in the source code

typedef struct ClientData CLIENTDATA;

2.2 Major software components

Module	File	Dependencies
Server	Server.c Server.h	FileIO.h, Constants.h
FileIO	FileIO.c FileIO.h	Constants.h
Constants	Constants.h	



2.3 Major interfaces

The client will end various requests to the server. The server will then handle each request accordingly.

“LOGIN/username/password” : Client is asking to login

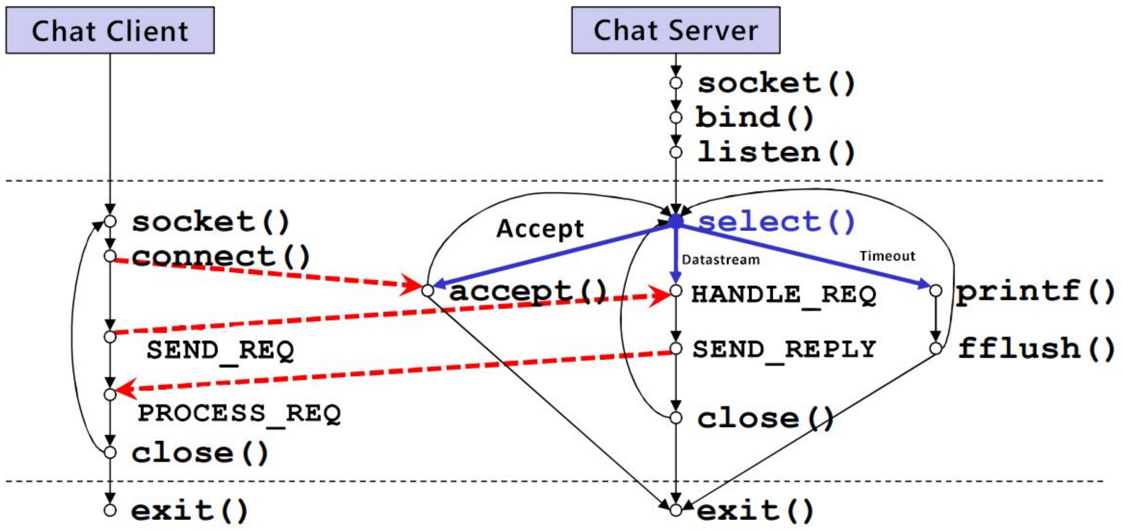
“REGISTER/username/password” : Client is asking to create an account
“ADD/username” : Client is asking to send a friend invite to another user
“REMOVE/username” : Client is asking to remove a friend from his contact list
“ACCEPT_REQUEST/USERNAME” : Client is asking to accept a friend request from another user
“IGNORE_REQUEST/USERNAME” : Client is asking to ignore a friend request from another user
“REQUEST_CONTACTS/username” : Client is asking server to send his contact list
“REQUEST_FRIEND_REQUESTS/username” : Client is asking server to send his friend requests list

The server will handle each request through the ProcessRequest(int DataSocketFD) method. It will use the following functions to verify the request data and perform the operation

void ProcessRequest(int DataSocketFD)
bool IsValidUserName(char* username)
bool IsValidPassword(char* password)
void AddContact(char* client, char* friend)
void RemoveContact(char* client, char* friend)
void AcceptFriendRequest(char* client, char* friend)
void RemoveFriendRequest(char* client, char* friend)
bool IsNotTakenUsername(char* username)
bool IsInContactlist(char* client, char* friend)
bool IsInFriendRequests(char* client, char* friend)

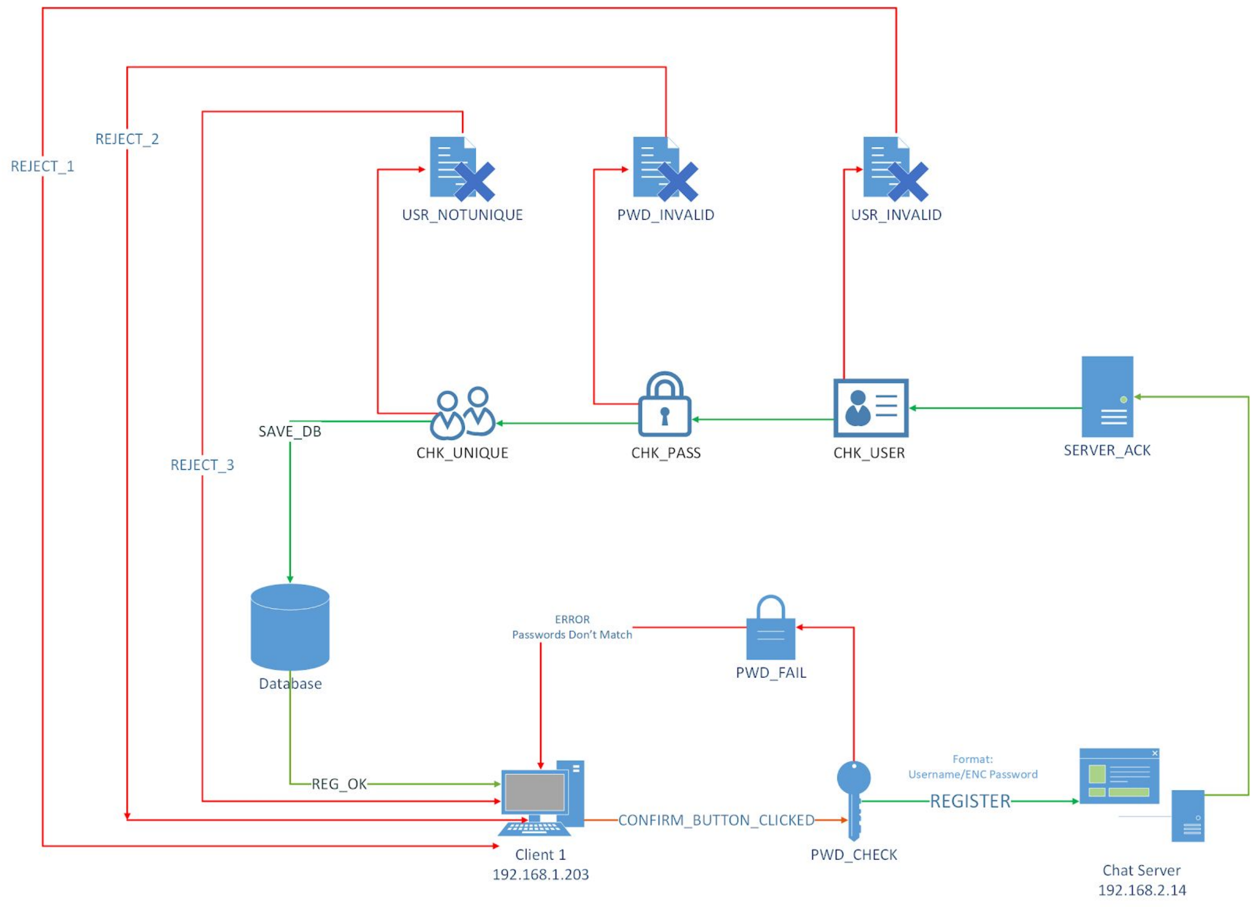
2.4 Overall program control flow

This diagram describes the main server loop and how it interacts with a client.



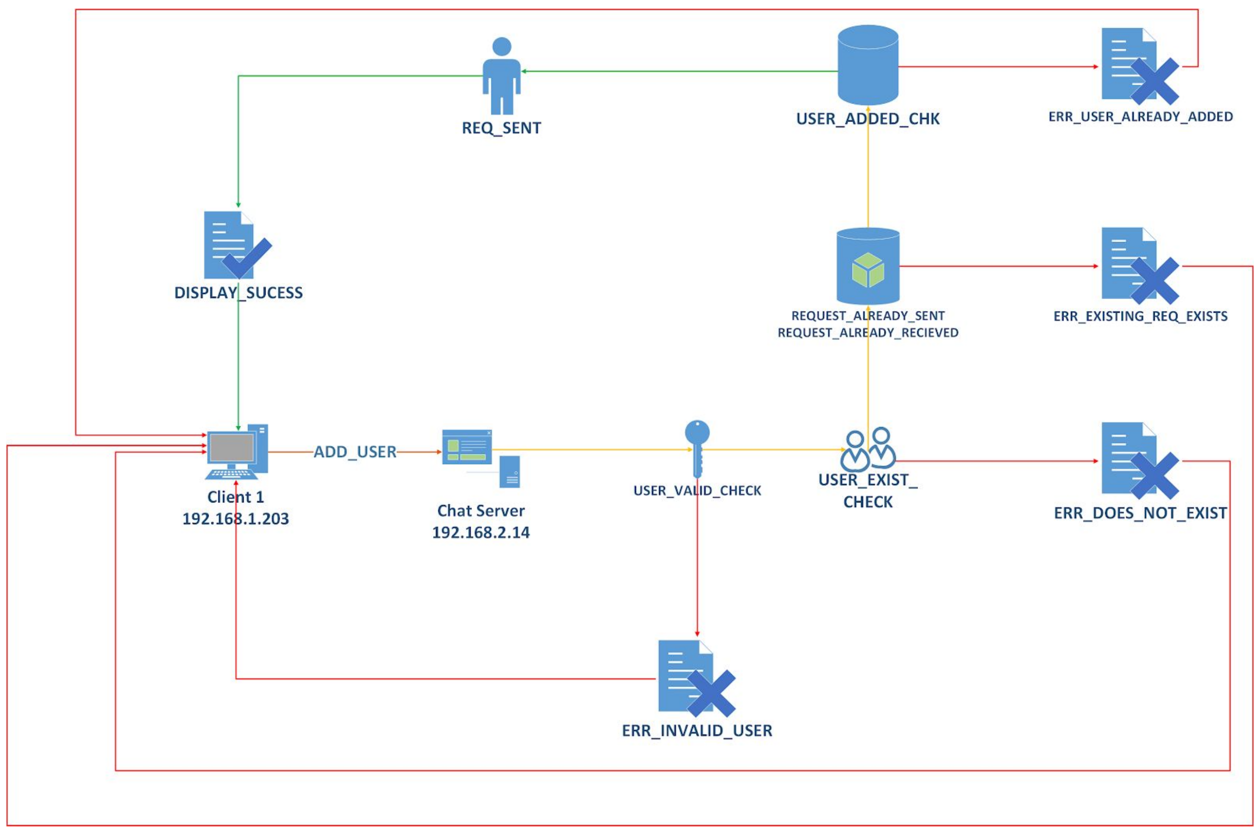
These following diagrams show the operations the server performs in order to handle each request from a client and the response the server sends.

When the server receives “REGISTER/username/password”



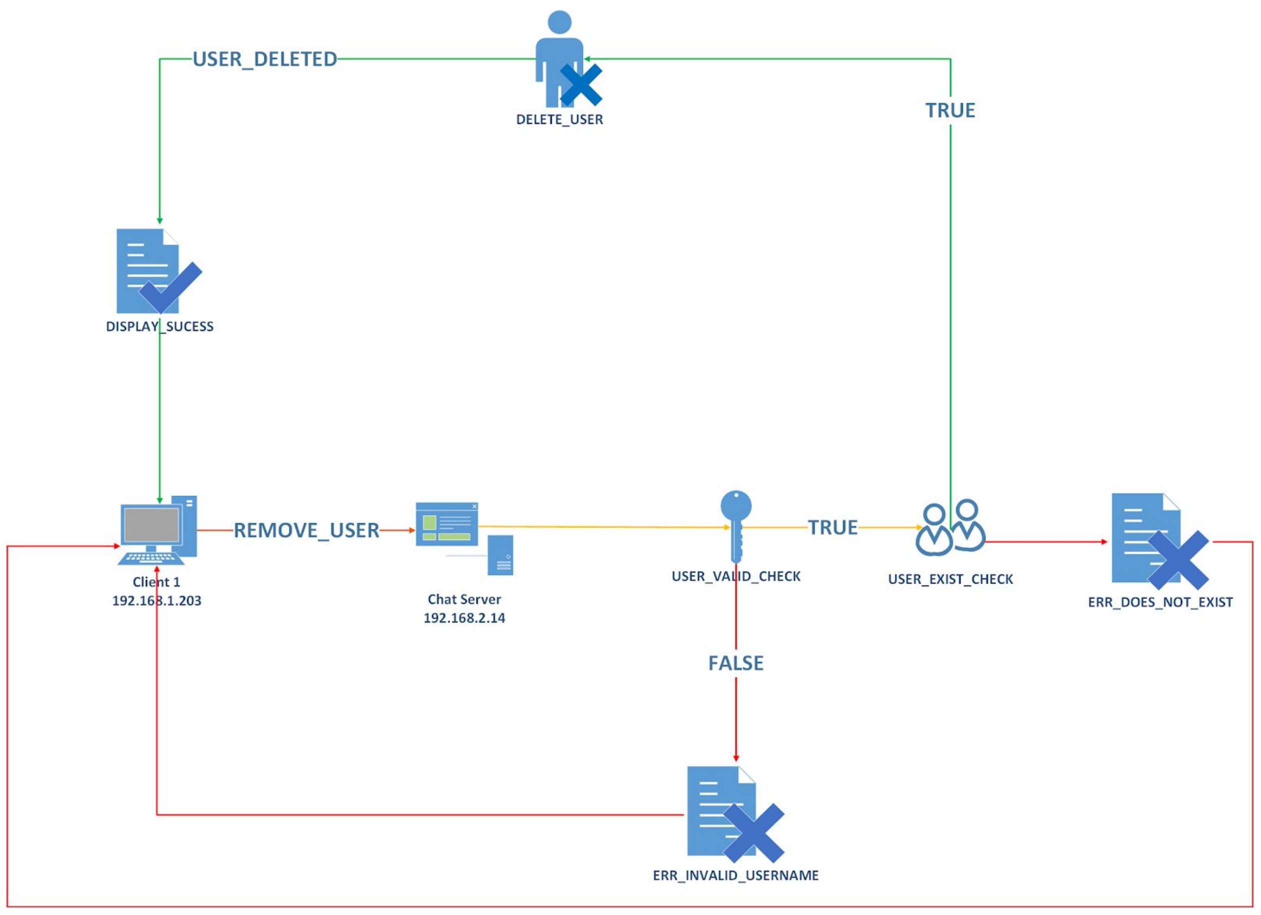
When the server receives a REGISTER/username/password request from a client, it acknowledges and proceeds to check if the username is valid or not. If it is not valid, the server will respond with “USERNAME_INVALID”. Otherwise, the server will check if the password is valid or not. If it is not valid, the server will respond with “PASSWORD_INVALID”. Last but not least, the server checks if the username is not taken. If it is taken, the server will respond with “USERNAME_TAKEN”. Else, the server will create the account and respond with “OK_REGISTER” to signify the registration is a success.

When the server receives “ADD/username”



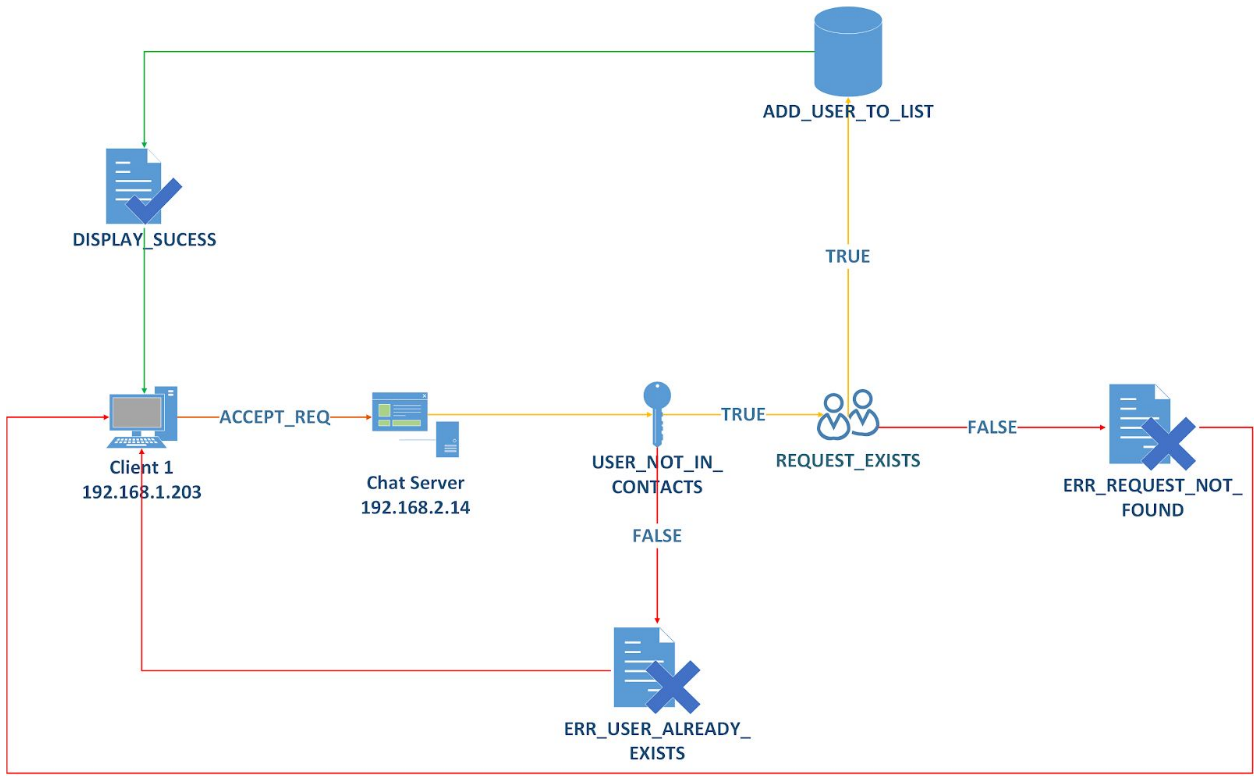
When the server receives an “ADD/username” request from the client, it first checks if the username is valid or not. If the username is invalid, the server will respond with “USERNAME_INVALID”. Otherwise, the server checks if the username exists or not. If it doesn’t exit, the server responds with “USERNAME_DOES_NOT_EXIST”. Then, the server checks if the username is already in the contact list. If it is, the server will respond with “USERNAME_ALREADY_ADDED”. Next, the server checks if client is already in the other client’s friend request list. If it exists, the server will respond with “REQUEST_ALREADY_SENT”. Last but not least, the server checks if the username is already in the client’s friend request list. If it is in the friend request list, the server will respond with “REQUEST_ALREADY_RECEIVED”. Else, the server will add the friend request to the other client’s friend request list and respond with “REQUEST_SENT” to signify adding a friend is a success.

When the server receives “REMOVE/username”



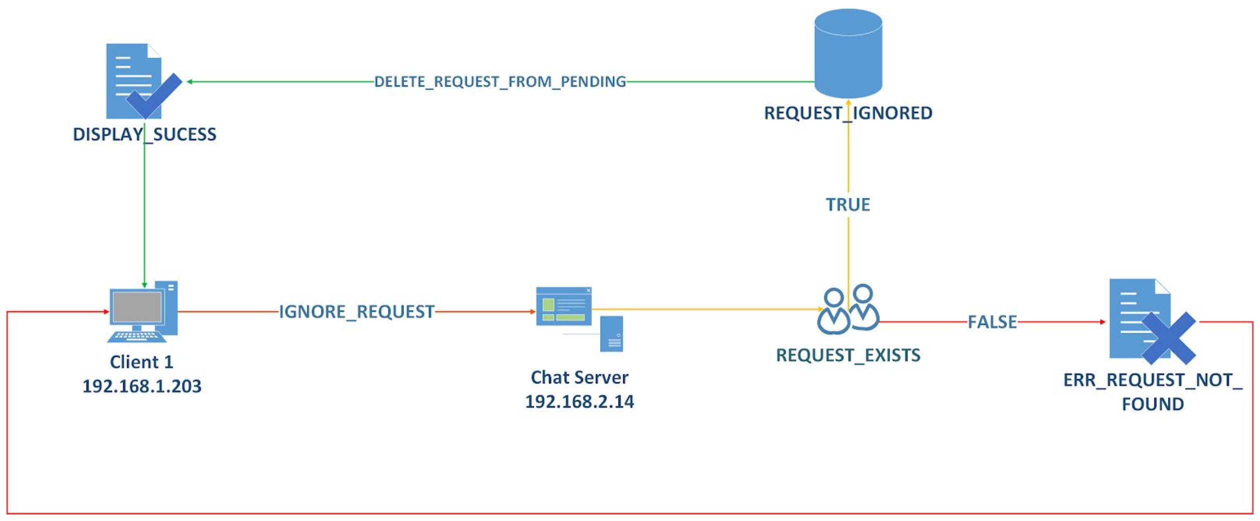
When the server receives REMOVE/username request from the client, it will check if the username is valid. If it is invalid, the server will respond with ‘USERNAME_INVALID” to the client. The server will also checks if the username is in contact list. If it is not in the contact list, the server will respond with ‘USERNAME_NOT_IN_CONTACTS” to the client. If the server checks all of the conditions above and there is no error, the server will delete the contact from the contact list and respond with “USER_DELETED” to the client to signify removing a friend is a success.

When the server receives "ACCEPT_REQUEST/USERNAME"



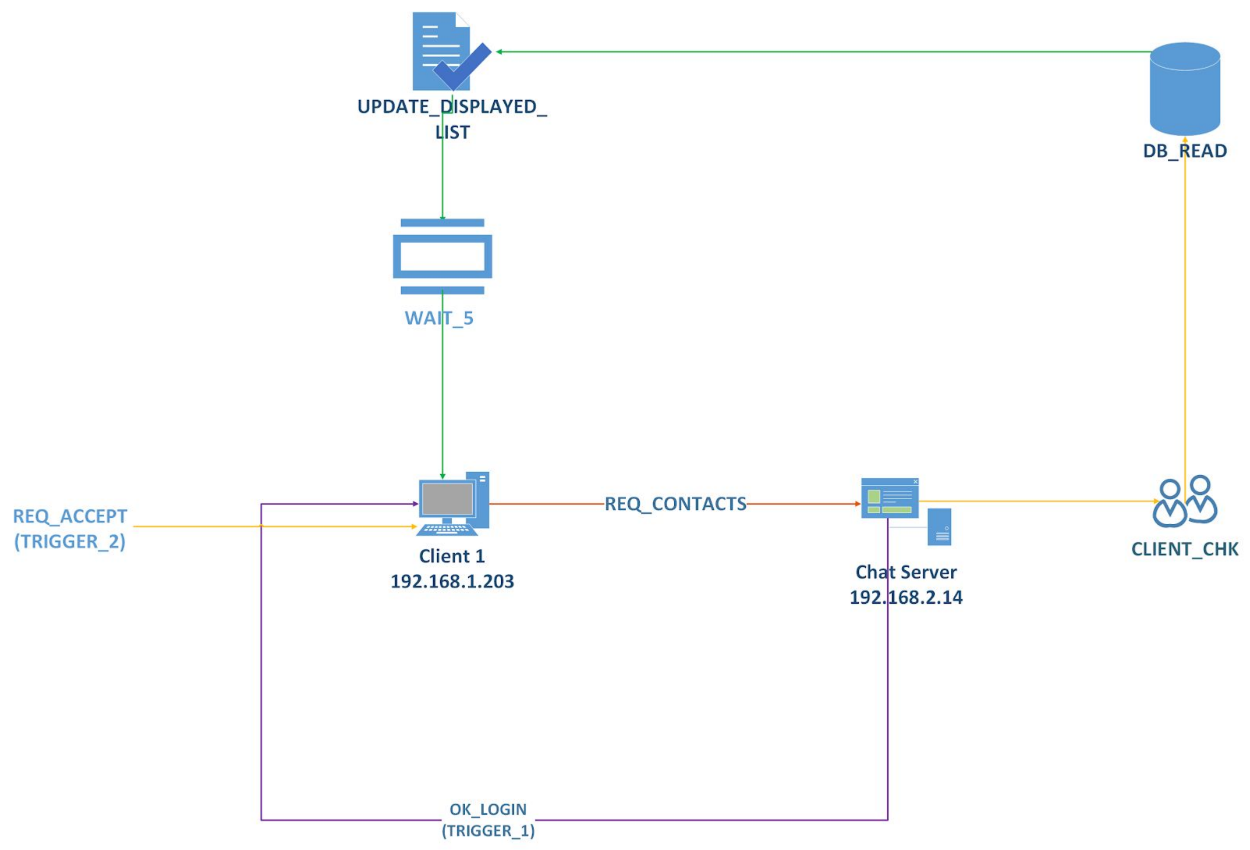
When the server receives "ACCEPT_REQUEST/USERNAME" request, it checks if the username is not in the contact list. If it is in the contact list, the server cannot accept the friend request and will respond with "ALREADY_IN_CONTACT" to the client. Second, the server checks if the username is in friend requests of the client. If it is not in friend requests, the server will respond with "REQUEST_NOT_FOUND" to the client. Else, the server will add the friend to the contact list, delete the friend request from the friend request list and respond with "REQUEST_ACCEPTED" to signify accepting a friend request is a success.

When the server receives “IGNORE_REQUEST/USERNAME”



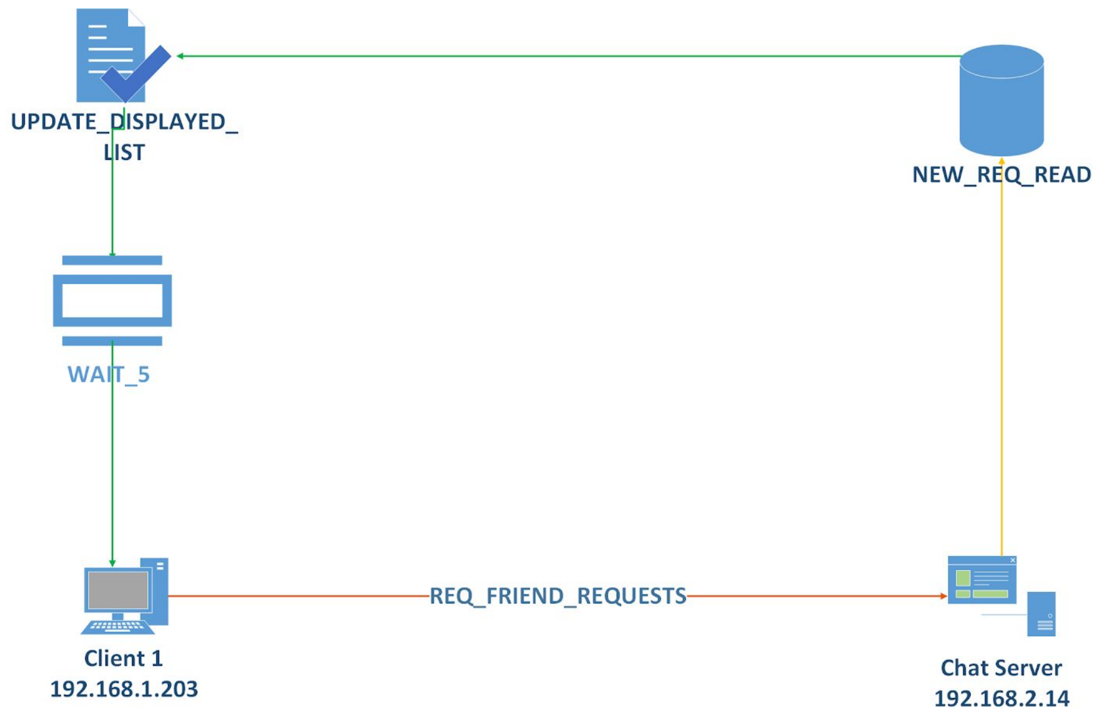
When the server receives “IGNORE_REQUEST/USERNAME” request, it checks if the username is in friend requests of the client. If it is not in friend requests, the server responds with “REQUEST_NOT_FOUND” to the client. If there is no error, the server will remove the friend request from the friend request list and respond with “REQUEST_IGNORED” to signify ignoring a friend request is a success.

When the server receives “REQUEST_CONTACTS/username”



When the server receives “REQUEST_CONTACTS/username” request, the server checks the username and reads the contact list of the client. Then, the server will respond by sending the contacts back to the client in the form “contact1/contact2/contact3/…”

When the server receives “REQUEST_FRIEND_REQUESTS/username”



When the server receives “REQUEST_FRIEND_REQUESTS/username”, it checks the username and reads the all friend requests of the client. Then, the server will respond by sending the friend requests back to the client in the form “friend_request1/ friend_request2/ friend_request3/...”

3. Installation

3.1 System Requirements, compatibility

Hardware : CPU with x86 ISA support
OS:	Red Hat Enterprise Linux x64
	x11 Forwarding (For Possible Updates)

3.2 Setup and Configuration

To install this program:

1. Type gtar xvzf QuickChat V1.0_src.tar.gz to extract the files.
   The source code will be in the src folder.
2. Type   make clean   to empty the bin folder.
3. Make to compile and make test, make test to run

3.3 Building, compilation, installation

To build the program, first compile the source code into object files using "-c" option for gcc to create the object files for each module, e.g.

```
% gcc -c FileIO.c -o FileIO.o -ansi -std=c99 -Wall
```

Make sure to use a c99 compiler.

We will provide a Makefile with the following targets:
-all: the target to generate the executable program

-clean: the target to clean all the intermediate files, e.g. object files, game logs
Therefore, you can build the program by typing “make clean” and “make all”

Server:

To start the server, the user will enter the command

```
./ChatServer
```

The server will display some initialization messages, and begin listening to client connection requests and other commands.

Client:

To run the client, the user will enter the command

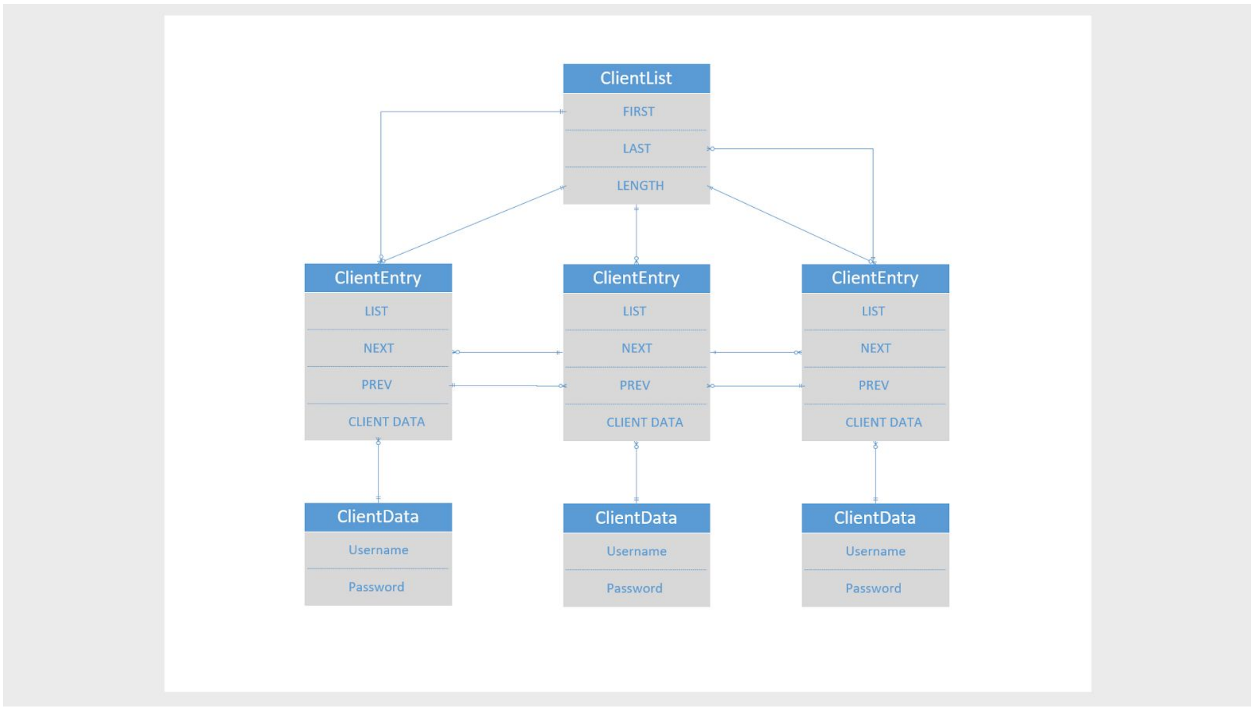
```
./Chat
```

The GUI window will open shortly if there are no issues with X11 graphics forwarding.

4. Documentation of packages, modules, interfaces

4.1 Detailed description of data structures

To save the client information in the server, we store them in the double linked list system. The list has the following structure:



```
Typedef struct ClientEntry{
        CLIST* LIST;
        CLIENTDATA* NEXT;
        CLIENTDATA* PREV;
        CLIENTDATA* ClientData;
        }CENTRY;

Typedef struct ClientList{
        CENTRY* FIRST;
        CENTRY* LAST;
        int Length;
}CLIST;

Typedef struct ClientData{
        char* username;
        char* password;
}CLIENTDATA
```

4.2 Detailed description of functions and parameters

Server.h

void FatalError(const char* ErrMsg)

Prints diagnostics errors and aborts the program

- ErrMsg	a descriptive error message that will be printed

int MakeServerSocket(uint16_t PortNo)

Initializes the server by calling socket(), bind() and listen() functions

- PortNo	the port number that the server and client use

void TimeOutFunction(void)

Prints that Server has nothing to do when the server is in time out.

void ProcessRequest(int DataSocketFD)

Processes a request that was sent from a client

- DataSocketFD	socket for a new client

void ServerMainLoop()

makes a simple server main loop

ClientList.h

CENTRY* CreateClientEntry (CLIENTDATA* ClientData)

Creates a client entry using the given ClientData and returns a pointer to the entry.

- ClientData	a pointer to the ClientData

void DeleteClientEntry(CENTRY* entry)

Deletes a client entry passed from the parameter and frees the memory that is used.

- entry	a pointer to the entry

CLIST* CreateClientList()

Creates a list of all client contacts and returns a pointer to the list return NULL when there is no contact.

void DeleteClientList(CLIST* list)

Deletes the list of all client contacts and frees the memory that was allocated.

- list	a pointer to the list

void AppendClient(CLIST* list, ClientData* ClientData)

Creates an entry for the Client_Data and adds it to the end of the list.

- list	a pointer to the list
- ClientData	a pointer to the ClientData

FileIO.h

char Deconstruct(char* str, const char delimiter)**

Used for splitting a char* into its sub char*'s based on the given delimiter

- str	a pointer to the char array
- delimiter	delimiting character

void FreeArray(char array)**

deallocates the memory previously allocated by a call to malloc

- array	a pointer to the array to free

void AddUser(char* username, char* password)

adds an username and password to a directory

- username	a pointer to the username you try to add
- password	a pointer to the password you try to add

void AddContact(char* client, char* friend)

Adds a friend by adding friend request to the other user's friend request list.

- client	a pointer to the client's username
- friend	a pointer to the friend's username to be added

void AddFriendRequest(char* client, char *friend)

Adds a friend request to the client's friend request.

- client	a pointer to the client's username
- friend	a pointer to the friend's who adds friend request to the client

bool IsInContactList(char* client, char* friend);

returns true if the friend is in the contact list of the client

- client	a pointer to the client's username
- friend	a pointer to the friend's username to be verified

bool IsInFriendRequests(char* client, char* friend)

returns true if the friend is in the friend requests list of the client

- client	a pointer to the client's username
- friend	a pointer to the friend's username to be verified

bool IsNotTakenUsername(char* username)

returns true if the username is taken already

- username	a pointer to the username to check whether it is taken

bool IsValidUsername(char* username)

returns true if the username is valid

- username	a pointer to the username to check whether it is a valid username

bool IsValidPassword(char* password)

returns true if the password is valid

- username	a pointer to the username to check whether it is a

		valid password

void RemoveContact(char* client, char* friend)

Removes a friend from the contact list of the client

-	client	a pointer to the client’s username
-	friend	a pointer to the friend’s username to be removed

void AcceptFriendRequest(char* client, char* friend)

Adds a contact to the contact list of the client and remove that friend request from the friend requests list
of the client.

-	client	a pointer to the client’s username
-	friend	a pointer to the friend’s username to be accepted

void RemoveFriendRequest(char* client, char* friend)

Removes the friend request from the friend request list of the client

-	client	a pointer to the client’s username
-	friend	a pointer to the friend who made the request

Client.h

void FatalError(const char* ErrMsg)

prints diagnostics errors and abort

-	ErrMsg	a pointer to the descriptive error message that will be printed

void UpdateContactListAndStatus(char* client)

updates the contact list and status of each contact

-	client	a pointer to the client’s username

char* Talk2Server(const char *Message, char *RecvBuf)

communicates with the server

-	Msg	a pointer to the request for the server to perform the operation
-	RecvBuf	a pointer to a message buffer for receiving a message

bool PasswordsMatch(char* p1, char* p2)

returns true if the password and the confirmed password match

-	p1	a pointer to the password that a user types in the password entry
-	p2	a pointer to the password that a user types in the confirm password entry

void Encrypt_Password (char* password)

Encrypts a password inside the client server before sending to the server

-	password	a pointer to the password that will be encrypted

void Request_Friend_Requests(char* client, char* friend)

Requests the friend requests of the user from the server

-	client	a pointer to the client’s username
-	friend	a pointer to the friend that is in the friend request list

void Request_Contact_List(char* client, char* friend)

Requests the contact list of the client from the server

-	client	a pointer to the client’s username
-	friend	a pointer to the friend that is in the contact list

ClientGui.c

static void Log_In_Button_Clicked(GtkWidget* widget, gpointer data)

Callback function for when the login button is clicked in the login box. Sends a Login request to the server.

-	widget	a widget that takes action to log in an account
-	data	a pointer to LoginBoxControls needed for logging in

static void Confirm_Button_Clicked(GtkWidget* widget, gpointer data)

Callback function for when the confirm button is clicked in the registration box. Sends a Confirm request to the server

-	widget	a widget that takes action to confirm the username, password, and confirm-password.
-	data	a pointer to the data needed to perform the operation

static void Add_Button_Clicked(GtkWidget* widget, gpointer data)

Callback function for when an add button is clicked in the Add/Remove a friend popup window. Sends an Add Contact request to the server.

-	widget	a widget that takes action to open “add a contact”
-	data	a pointer to the data needed to perform the operation

static void Remove_Button_Clicked(GtkWidget* widget, gpointer data)

Callback function for when an remove button is clicked in the add/remove a friend popup window. Sends a Remove Contact request to the server.

-	widget	a widget that takes action to open “remove a contact”

- data a pointer to the data needed to perform the operation

static void Accept_Friend_Request_Button_Clicked(GtkWidget* widget, gpointer data)

Callback function for when an accept button is clicked in friend requests popup window. Sends an accept request to the server.

- widget	a widget that takes action to accept an invitation
- data	a pointer to the data needed to perform the operation

static void Ignore_Friend_Request_Button_Clicked(GtkWidget* widget, gpointer data)

Callback function for when an ignore button is clicked in the friend requests popup window. Sends an Ignore request to the server.

- widget	a widget that takes action to ignore an invitation
- data	a pointer to the data needed to perform the operation

static void Friend_Requests_Button_Clicked(GtkWidget* widget, gpointer data)

Create a dialog box which contains a background picture, a friend's request, accept button and ignore button .

- widget	a widget that takes action to send friend request
- data	a pointer to the data needed to perform the operation

static void Shut_Down_Button_Clicked(GtkWidget* widget, gpointer data)

Callback function for when a shutdown button is clicked. Sends an shutdown request to the server

- widget	a widget that takes action to send shutdown request
- data	a pointer to the data needed to perform the operation

static void Cancel_Button_Clicked(GtkWidget* widget, gpointer data)

Callback function for when a cancel button is clicked in the registration box. Send a cancel request to the server

- widget	a widget that takes action to send cancel request
- data	a pointer to the data needed to perform the operation

static void Register_Button_Clicked(GtkWidget* widget, gpointer data)

Callback function for when the register button is clicked in the login box. Sends a Register request to the server.

- widget	a widget that takes action to register an account
- data	a pointer to the data needed to perform the operation

static void Exit_Button_Clicked(GtkWidget* widget, gpointer data)

Callback function for when the exit button is clicked in the login box. Sends an exit request to the server

- widget	a widget that takes action to exit an login menu

- data	a pointer to the data needed to perform the operation

static void Log_Out_Button_Clicked(GtkWidget* widget, gpointer data)

Callback function for when the logout button is clicked in the logout box. Send an logout request to the server

- widget	a widget that takes action to log out the account
- data	a pointer to the data needed to perform the operation

static void Add_Remove_Button_Clicked(GtkWidget* widget, gpointer data)

Creates a dialog box which contains a background picture, “Type a friend’s name” entry, add button, and remove button.

- widget	a widget that takes action to add or remove a contact
- data	a pointer to the data needed to perform the operation

static void Contacts_Button_Clicked(GtkWidget* widget, gpointer data)

Callback function for asking the contacts information to the server

- widget	a widget that takes action to get contacts
- data	a pointer to the data needed to perform the operation

static void Register_Confirm_Button_Clicked(GtkWidget* widget, gpointer data)

Callback function for when the register confirm button is clicked in the registration box. Sends a Register Confirm request to the server.

- widget	a widget that takes action to confirm the username and password and confirm password
- data	a pointer to RegisterBoxControls needed for registering an account

static void Register_Cancel_Button(GtkWidget* widget, gpointer data)

Callback function for when the register cancel button is clicked in the registration box. Sends a Register Cancel request to the server.

- widget	a widget that takes action to cancel the registration and go back to the login page
- data	a pointer to the data needed to perform the operation

ClientGui.h

GTKWidget* Create_Registration_Box()

creates the window for the registration area which contains a background picture, QuickChat Logo, username label, username entry, password label, password entry, confirm password label, confirm password entry, confirm button, and cancel button

GTKWidget* Create_LogIn_Box()

creates the window for the login area which contains a background picture, QuickChat Logo, welcome label, username label, username entry, password label, password entry, login button, registration label and registration button

GTKWidget* Create_Account_Box()

creates window for the account area which contains a friend list, status, chat log, message entry, send button, add/remove a contact button, friend requests button, logout button, tabs, and scroll bars.

void SaveConversation(char* client, char* friend)

Saves the conversation between the client and his friend in a text file

- client	a pointer to the client's username
- friend	a pointer to the friend that you would like to save conversation

void RetrieveConversation(char* client, char* friend)

Retrieve the conversation between the client and his friend from the text file

- client	a pointer to the client's username
- friend	a pointer to the friend that you would like to retrieve the conversation

char* SendRequest2Server(const char* msg, char* RecvBuf)

Sends a request to the server to perform the operation

- Msg	a pointer to the request for the server to perform the operation
- RecvBuf	a pointer to a message buffer for receiving a message

void Remove_All_Children (GtkContainer* container)

Helper method that removes all widgets from a container

- container	the container that you would like to delete

void Set_New_Container_Content (GtkContainer* container, GtkWidget* widget)

Helper method that removes all widgets from the container and adds a new widget

- container	the container that you would like to delete
- widget	the widget that you would like to add in the new container

void Quick_Info_Message (GtkWidget* widget, const char* title, const char* message)

prints the informational message

- widget	the widget that you would like to show the message
- title	a pointer to the title you would like to show in the window

-	message	a pointer to the request for the server to perform the operation

void Quick_Error_Message (GtkWidget* widget, const char* title, const char* message)

prints the error message

-	widget	the widget that you would like to show the message
-	title	a pointer to the title you would like to show in the window
-	message	a pointer to the request for the server to perform the operation

int Quick_Yes_No_Message (GtkWidget* widget, const char* title, const char* message)

prints the yes or no question message

-	widget	the widget that you would like to show the message
-	title	a pointer to the title you would like to show in the window
-	message	a pointer to the request for the server to perform the operation
-		

4.3 Detailed description of the communication protocol

A. Explanation of communication related function calls

Communication between client and server goes through the following phases
Server side:

1. initializing the server and waits for a client to log in.
   void initializeServer(uint16_t PortNo)

2. Server Main loop
   void ServerMainLoop(int ServSocketFD, ClientHandler HandleClient, TimeoutHandler
   Handle timeout, int Timeout)
   The server will wait for a client request, processes the request then responds.
   void ProcessRequest(const char*ErrMsg)

If there are any errors, FatalError(const char* ErrMsg) will print diagnostic error and abort the program.

Client side:

1. Client connects to the server via sockets
2. Client waits for user action from the GUI
3. Client sends a request to the server

SendRequest2Server(const char* msg, char* RecvBuf), sends the message to the server.

4. Client waits for a response from the server
5. Client updates GUI for the user

B. Explanation of communication protocol flags/settings

Requests to the server will all be char* string like messages and will start with a keyword. The keywords and server responses are listed in section 2.4.

“LOGIN/username/password” : Client is asking to login
“REGISTER/username/password/” : Client is asking to create an account
“ADD/username” : Client is asking to send a friend invite to another user
“REMOVE/username” : Client is asking to remove a friend from his contact list
“ACCEPT_REQUEST/USERNAME” : Client is asking to accept a friend request from another user
“IGNORE_REQUEST/USERNAME” : Client is asking to ignore a friend request from another user
“REQUEST_CONTACTS/username” : Client is asking server to send his contact list
“REQUEST_FRIEND_REQUESTS/username” : Client is asking server to send his friend requests list

5. Development plan and timeline

5.1 Partitioning of tasks

Server - Rijul Arora, Mehdi Lakhoua, David Tiao
Client- Rijul Arora, Mehdi Lakhoua, David Tao
ClientGUI- Yuki Hayashi, Ye Myat Kyaw, Mehdi Lakhoua
GuiFunctions- Mehdi Lakhoua, Ye Myat Kyaw, Yuki Hayashi
Client List - Yuki Hayashi
FileIO - David Tiao, Rijul Arora

5.2 Team Member responsibilities

Team members:

Yuki Hayashi, Rijul Arora, Mehdi Lakhoua, Ye Myat Kyaw, David Tiao

Server: intitializeServer, ServerMainLoop - Rijul Arora, Mehdi Lakhoua, David Tiao
Server: FatalError, ProcessRequest - Mehdi Lakhoua
Server: IsValidUsername, IsValidPassword - Rijul Arora
FileIO: AddContact, RemoveContact, AcceptFriendRequest, RemoveFriendRequest - David Tiao
FileIO: IsNotTaken, IsInContactList, IsInFriendRequests - Rijul Arora
Client: UpdateContactList, Request_Friend_Requests, Request_Contact_List - David Tiao
Client: PasswordsMatch, Encrypt_Password, FatalError - Rijul Arora
ClientGUI: Add_Contact_Buton_Clicked, Remove_Contact_Button_Clicked, Friend_Requests_Button_Clicked , Accept_Friend_Request_Button_Clicked, Ignore_Friend_Request_Button_Clicked - Yuki Hayashi
ClientGUI: Login_Button_Clicked, Register_Button_Clicked, Register_Confirm_Button_Clicked, Register_Cancel_Button, Friend_Requests_Button_Clicked - Ye Myat Kyaw
ClientGUI: Create_LogIn_Box, Create_Registration_Box, Create_Account_Box, SaveConversation, RetrieveConversation, SendRequest2Server - Yuki Hayashi
GUIFunctions: Remove_All_Children, Set_New_Container_Content - Mehdi Lakhoua
ClientList: CreateClientEntry, DeleteClientEntry, CreateClientList, DeleteClient, DeleteClientList, AppendClient - Yuki Hayashi

Back Matter

Copyright

References

All diagrams and images were made by the team members using visual paradigm, visio and photoshop

Index

3
account
add
callback
chat
client
confirm
contact list
conversation
GUI
error
function
label
login
pointer
server
status
struct
widget

4
chat area
components
contact
dependency
file
hardware
module
software
source code
typedef
update

5
Add Contact request
account box
add button

add/remove a contact button
add/remove a contact popup
cancel button
confirm button
confirm password
contact area
encrypt
error
error label
exit button
login
login button
login menu
login request
login screen
login window
main loop
main menu
module interface
password
popup
popup window
program
register button
register request
registration
registration box
registration menu
username

6

Accept Friend Request request
accept button
contact list
friend request
friend request button
friend request list
friend request popup window
Get Friend Requests request
Ignore Friend Request
ignore button
Remove Contact request
remove button

8

chat box
contact list area
message
notification number box
send button

10

“ACCEPT_REQUEST/USERNAME”
“ADD/username”
“IGNORE_REQUEST/USERNAME”
“LOGIN/username/password”
“REGISTER/username/password”
“REMOVE/username”
“REQUEST_CONTACTS/username”
“REQUEST_FRIEND_REQUESTS/username”

12

OK_REGISTER
PASSWORD_INVALID
USERNAME_INVALID
USERNAME_TAKEN

14

REQUEST_ALREADY_RECEIVED
REQUEST_ALREADY_SENT
REQUEST_SENT
USERNAME_ALREADY_ADDED
USERNAME_DOES_NOT_EXIST

15

REQUEST_ACCEPTED
REQUEST_NOT_FOUND
USERNAME_NOT_IN_CONTACTS

USER_DELETED

16

ALREADY_IN_CONTACT
REQUEST_IGNORED

18

Installation
bin
compatibility
compile
configuration
extract
file
folder
setup
system requirement

19

command
compiler
executable
initialization
target