

THOT Patrol

Bianca Corine Doronila
Cesar Alvarenga
Joshua Miller
Michael Reza Honar
Michael Barbosa
Tun Myat

In affiliation with: The Henry Samueli School of Engineering at The University of California Irvine



THOT Patrol Software
Version 1.0

Released 5th March 2018

Table of Contents

	<u>Page</u>
Glossary	3
1 Instant Messaging	4
1.1 Usage Scenario	4
1.1.1 GUI Prototype Sketch	4
1.2 Goals	8
1.3 Features	8
1.3.1 Basic Features	8
1.3.2 Full GUI	8
1.3.3 Advanced Features	8
2 Installation	9
2.1 System Requirements	9
2.2 Setup and Configuration	9
2.3 Uninstalling	9
3 Program Functions and Features	9
3.1 Client and Server Communication	9
3.2 Logging in and Registering Algorithm	10
3.3 Photo filters	13
3.4 Friends List	13
4 Help and Error Messages	14
5 Copyright	15
6 References	15
7 Index	16

Glossary

account	-	basic information, such as name, username, passwords, and date of birth that are unique to every user.
available	-	user is currently active or online
client	-	people who joined a server
friends	-	people you trust and enjoy being around
friends list	-	contact list of people the user trusts and enjoys being around
GUI	-	a “graphical user interface” that allows users to navigate Remarq
host	-	person who created a chat
idle	-	user is currently away but online on the user application
IM	-	acronym for “instant messaging” -- a message sent via the internet that appears on the recipient’s screen as soon as it is transmitted
invisible	-	user is currently online, but appearing offline
offline	-	user is currently not using the user application
password	-	a unique code that is unique that allows access to your account
register	-	create a new user account
status	-	status to the provider: active, idle, offline
settings	-	area of Remarq where user goes to change account information and preferences; aka “EDIT” button

0 Final Release Notes

Please read the README/INSTALL in parent directory for BEtA release special notes.

As explained in the readme, there are two applets that encapsulate the alpha release of this program. The first one is a GUI. The second is a client-2-client applet shown in 3.3. To use the client-2-client applet, please connect two clients to the server and start typing msgs. There is minimal prompt for “ServerTest” and “ClientTest” (the client-2-client applets). Just start typing and press enter, and the two clients and server will display the messages.

1 Instant Messaging

1.1 Usage Scenario

Messaging has been reimagined with Remarq. From direct messages to group convos and photo sharing, Remarq has you covered. This new messaging experience will provide you a great new way to connect with your friends, although if you have no friends: then this application is not for you. With Remarq, a user can register for their own account and be able to connect with their friends who are also registered users. Furthermore, we have the feature that lets our users change their availability status and also see their friends’ status (online, idle, or offline).

1.1.1 GUI Prototype Sketch

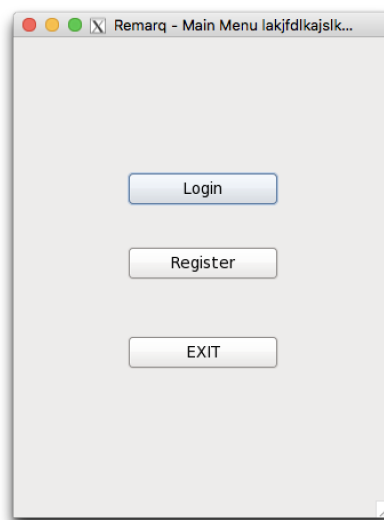


Figure 1.1 shows the first window of the Remarq GUI: allows user to choose between logging in or registering for a new account on our Remarq application

- User can register an account, and a new window will open
- User can log into the app, and a new window will open

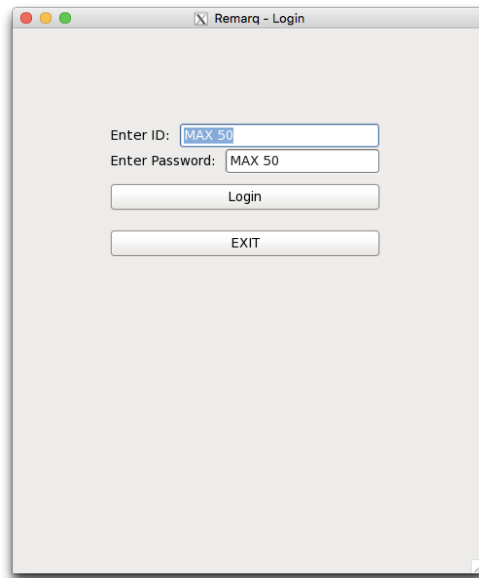


Figure 1.2 shows one of the two potential second window of the Remarq GUI: allows user to enter their user id and password, if they already have an account.

- User inputs their unique user id and password
- EXIT (Cancel) button to go back to the previous Remarq window (see Figure 1.1)
- Login button to access the user's account

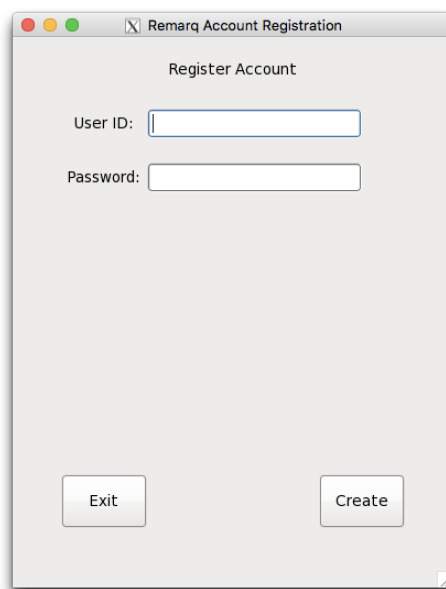


Figure 1.3 shows the second of the two potential second window of the Remarq GUI: allows user to register or create their own account.

- User inputs their desired user id and password

- User has to confirm chosen password to make sure they match,
- User enters their first and last name and date of birth for their profile

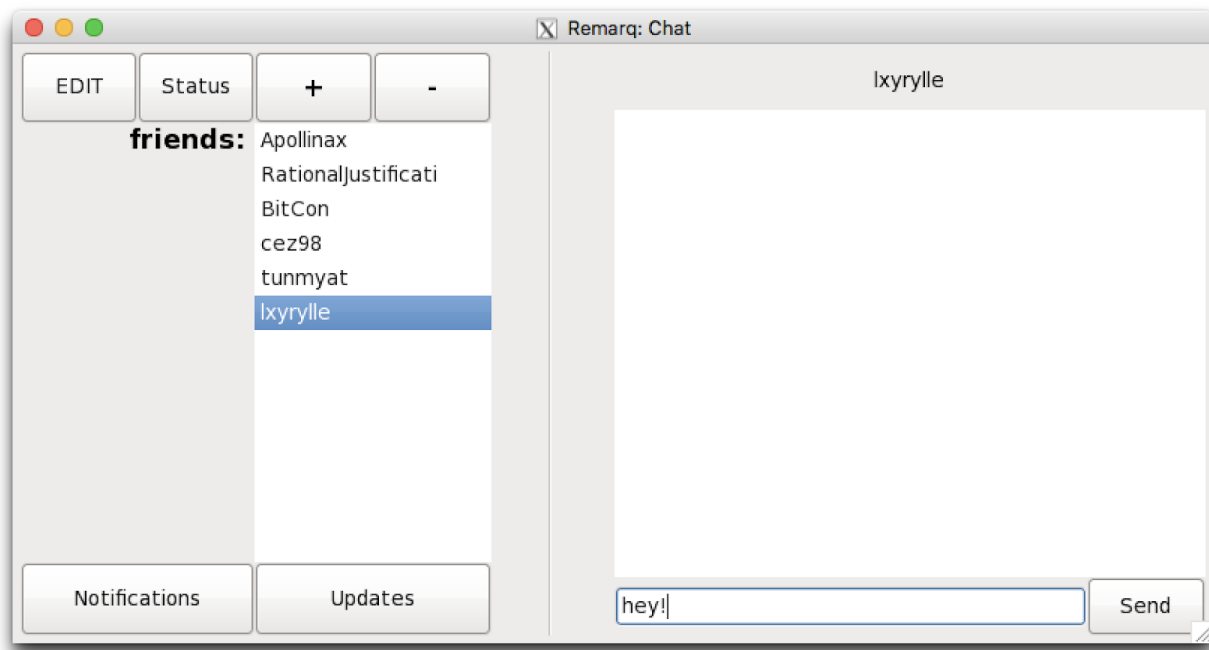


Figure 1.4 shows the third window of the Remarq GUI: shows the user's friends list, a button to edit their profile, and add or delete friends, and the chat window.

- User see their added friends
- User can edit their profile: update first and last name, DoB, through the EDIT button
- User can set their status to available (online), idle, invisible (appear offline) through the Status button
- User can add or delete contacts to their friends list (+ or -) button
 - Dialog Box will pop up to enter the username
 - Delete button automatically enters the username of the highlighted friend
- User will have the same window for each friend, for simplicity's sake
- Whichever username of the friend is highlighted is what shows on the chat bar above, on the right side of the window.

DISCLAIMER: The following buttons and functions are not implemented in our final release: Edit, Delete, Notifications, Updates, and Send* (*Send only displays text on window, but does not send a message.)

1.2 Goals

Our goal is to ensure the best and most satisfying messaging experience to everyone and their friends by providing the innovative instant messaging app Remarq.

In our alpha release, we have two different programs:

1. The first is a client-2-client chat applet, where a required 2 clients are needed to connect to a server (Applets are called ClientTest and ServerTest). This Applet will let the server handle individual messages by each client
2. The second applet is a login/register demo, where with LoginDemoServer and LoginDemoClient running, we can send a Login/Register/Quit protocol over a socket using read() and write().
3. Our alpha has a ASCII GUI. GUI will be available for beta and everything will be perfected for our final release.

1.3 Features

note: features will vary for alpha/beta release

1.3.1 Basic Features

- User can register an account
- User can log into the app
- User can add/delete contacts to their friends' list
- User can leave and join a group chat at any time
- User can see the status of users in their friends list
- User can accept or reject friend requests
- User can chat with other users on their friends list
- User can manage or edit their profile
- Remarq will highlight all unread messages in chat windows
- Remarq will keep track of conversation logs
 - Exports to a .txt file

1.3.2 Full GUI

- Menu where the user can either:
 - Sign In
 - Create an Account
- Update their Status
- Friends (or Contact) List

-
- Add or Delete Friends
 - Accept or Reject Friend Invitations
 - Chat Window
 - *See 1.1.1 for the full GUI prototype.*

1.3.3 Advanced Features

- Notifications on Chat Window
 - “(user) is typing...”
 - “(user) is online/offline/idle”
 - “Message Seen” and “Message Delivered”
- Remarq will be able to retrieve chat history

2 Installation

2.1 System Requirements

- OS: Linux Access on either Windows 7 (or higher) or MacOS
- Processor: 2.0 GHz Dual Core
- Memory: 2GB RAM
- Network: Broadband Internet Connection
- Storage: 500 MB
- Terminal shell

2.2 Setup and Configuration

- Open the command prompt (Terminal, MobaXterm, PuTTY)
- Login to SSH server
- Type: `tar -xvzf Remarq_v1.0_src.tar.gz`
- Find the directory called “bin/”
 - type command “`cd bin/`”
- We must execute a client and server together, server first. There are two different Applets.
 - For LoginDemo, we have
 - `./LoginDemoServer <port>`
 - `./LoginDemoClient <server> <port>`
 - For C2C chat, we have
 - `./ServerTest <port>`
 - `./ClientTest <server><port>`

2.3 Uninstalling

- Open the command prompt
- Type “`rm -r /bin`” in the parent directory of the tar.gz

3 Program Functions and Features

3.1 Client and Server Communication via Client/ServerTest bin

A user will either be able to create a port (host) where they can chat with their friend(s) or join a conversation (client) through a port.

```
[mhonar@bondi v3]$ ./LoginDemoServer 5555
```

Figure 3.1 shows the server is created with port number 8888 and it is waiting for a client to connect

```
[mhonar@bondi v3]$ ./LoginDemoClient bondi 5555
-----
Welcome! Please choose an option.
1: Create an Account
2: Log In
0: Exit
Please choose an option: 
```

Figure 3.2 shows a client is connected to server with port number 8888

3.2 Logging In and Registering Algorithm

Every user will have to log in, or register an account in order to use our program. Every account will be unique, and will require a password to ensure the safety and privacy of every user. Unique means that no two accounts will have the same username, this will cause confusion for both user and us the developers (usernames will be provided on first come first serve basis). The password will be encrypted for user safety. There will also be second time requesting the same password for verification to make sure there is no typo. User typing password will be replaced with asterix. If you forget your password... too bad, just make a new account.

```
Welcome! Please choose an option.
1: Create an Account
2: Log In
0: Exit
Please choose an option: 1
Please enter a desired username (max 20 char.): tunmyat
Please enter desired password (max 20 char.): *****
Please verify the password again(max 20 char.): *****
Sending register protocol to server
cl statement is *success*
Server Response: 'Register Success!'
```

Figure 3.3 shows a user is registered and it is successful because of no conflict username

```
Welcome! Please choose an option.  
1: Create an Account  
2: Log In  
0: Exit  
Please choose an option: 1  
Please enter a desired username (max 20 char.): tunmyat  
Please enter desired password (max 20 char.): *****  
Please verify the password again(max 20 char.): *****  
Sending register protocol to server  
c1 statement is *success*  
Server Response: 'Register Failed! Duplicate Username'
```

Figure 3.4 shows a user tries to register with conflict username so registration failed

```
Welcome! Please choose an option.  
1: Create an Account  
2: Log In  
0: Exit  
Please choose an option: 1  
Please enter a desired username (max 20 char.): Joshua  
Please enter desired password (max 20 char.): *****  
Please verify the password again(max 20 char.): *****  
Password do not match.  
Please enter desired password (max 20 char.): *****  
Please verify the password again(max 20 char.): *****  
Sending register protocol to server  
c1 statement is *success*  
Server Response: 'Register Success!'
```

Figure 3.5 shows a user tries to register with mismatched password when confirming, so it request the password again until verification success

```
Welcome! Please choose an option.  
1: Create an Account  
2: Log In  
0: Exit  
Please choose an option: 2  
Enter your username: raza  
Enter your password: *****  
Sending login protocol to server  
c4 statement is *success*  
Server Response: Login Failed!  
Bad Login. Please retry.
```

Figure 3.6 shows a user tries login with no existing username so login failed

```
Welcome! Please choose an option.  
1: Create an Account  
2: Log In  
0: Exit  
Please choose an option: 2  
Enter your username: tunmyat  
Enter your password: *****  
Sending login protocol to server  
c4 statement is *success*  
Server Response: Login Failed!  
Bad Login. Please retry.
```

Figure 3.7 shows a user tries login with wrong password so login failed

```
Welcome! Please choose an option.  
1: Create an Account  
2: Log In  
0: Exit  
Please choose an option: 2  
Enter your username: tunmyat  
Enter your password: *****  
Sending login protocol to server  
c4 statement is *success*  
Server Response: Login Success!
```

Figure 3.7 shows a successful login with correct username and password

```
Please enter the message: hello
I got your message
Please enter the message: how are you there?
I got your message
Please enter the message: glad to talk to you again
I got your message
Please enter the message: █
```

Figure 3.8 shows a successful messaging to server from client

```
Login msg from client: login data sent
Login successful.
read buffer for protocol is MESSAGE
In switch protocol state message
The message is:
tunmyat: hello

read buffer for protocol is MESSAGE
In switch protocol state message
The message is:
tunmyat: how are you there?

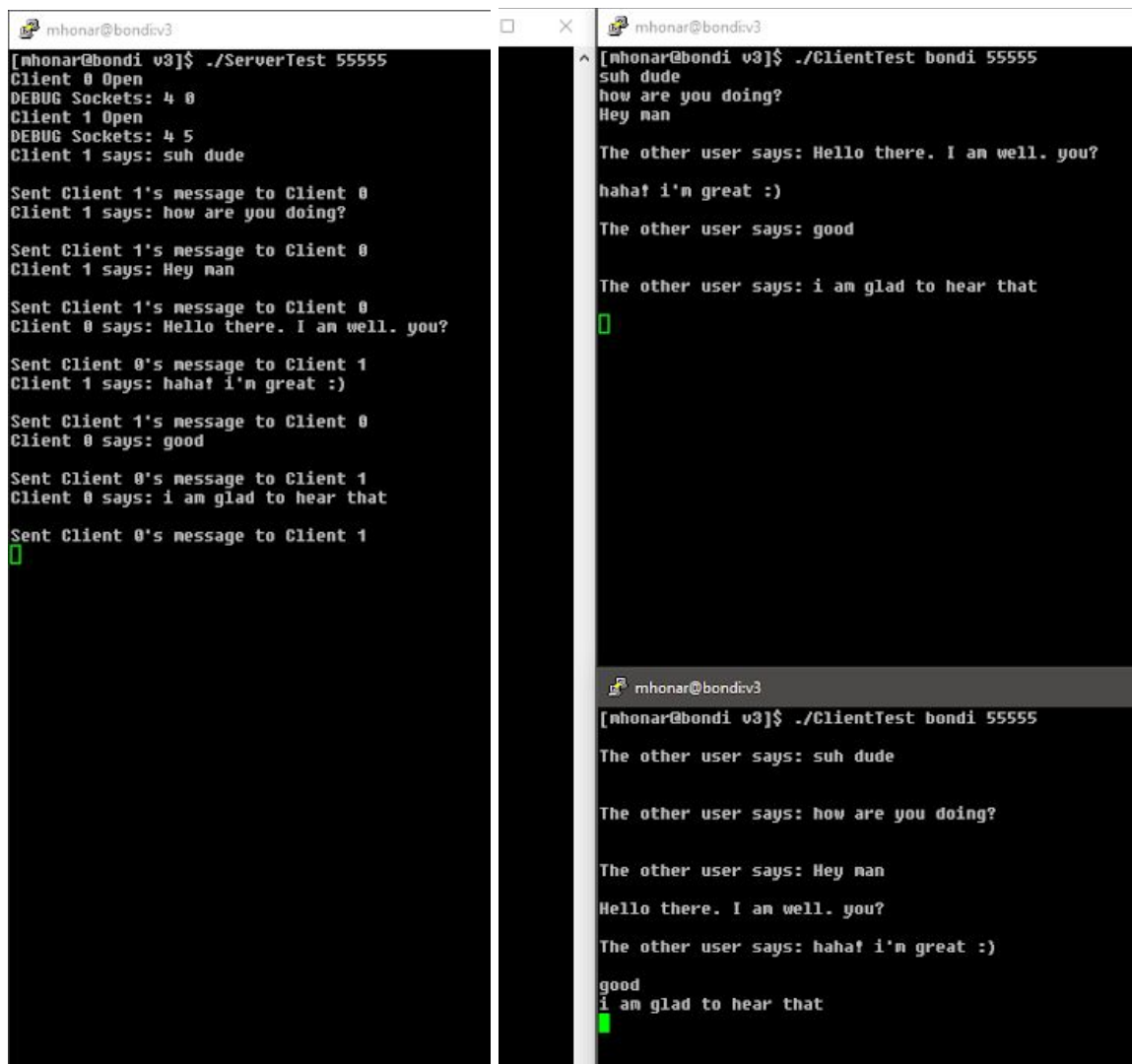
read buffer for protocol is MESSAGE
In switch protocol state message
The message is:
tunmyat: glad to talk to you again
```

Figure 3.9 shows the server received messages successfully from client

3.3 Client-2-Client Communication

As submission towards our alpha, we have written a barebones client-server application where a server waits for two clients to connect, then relays messages back and forth between clients. The server has a max load of 2 clients, and can differentiate both clients, and their messages. This is breakthrough technologies discovered at THOT PATROL's labs towards the progress of Remarq.

Below are screenshots of the 2 clients and the server running:



The image displays three terminal windows. The leftmost window shows the server's output, which includes messages like 'Client 0 Open', 'DEBUG Sockets: 4 0', 'Client 1 Open', 'DEBUG Sockets: 4 5', and a series of relayed messages between Client 0 and Client 1. The middle window is a standard Ubuntu window title bar. The rightmost window shows two separate client sessions. The top session, titled 'ClientTest bondi 55555', shows a user inputting 'suh dude', 'how are you doing?', and 'Hey nan', followed by the server relaying these messages as 'The other user says: Hello there. I am well. you?', 'hahah i'm great :)', and 'The other user says: good'. The bottom session shows the user inputting 'The other user says: i am glad to hear that' and receiving a green cursor. The bottom-most window shows another client session where the user inputs 'suh dude', 'how are you doing?', 'Hey nan', 'Hello there. I am well. you?', 'The other user says: hahah i'm great :)', 'good', and 'i am glad to hear that', with a green cursor at the end.

```
mhonar@bondi v3]$ ./ServerTest 55555
Client 0 Open
DEBUG Sockets: 4 0
Client 1 Open
DEBUG Sockets: 4 5
Client 1 says: suh dude

Sent Client 1's message to Client 0
Client 1 says: how are you doing?

Sent Client 1's message to Client 0
Client 1 says: Hey nan

Sent Client 1's message to Client 0
Client 0 says: Hello there. I am well. you?

Sent Client 0's message to Client 1
Client 1 says: hahah i'm great :)

Sent Client 1's message to Client 0
Client 0 says: good

Sent Client 0's message to Client 1
Client 0 says: i am glad to hear that

Sent Client 0's message to Client 1
[ ]

mhonar@bondi v3]$ ./ClientTest bondi 55555
suh dude
how are you doing?
Hey nan

The other user says: Hello there. I am well. you?
hahah i'm great :)
The other user says: good

The other user says: i am glad to hear that
[ ]

mhonar@bondi v3]$ ./ClientTest bondi 55555
The other user says: suh dude

The other user says: how are you doing?

The other user says: Hey nan
Hello there. I am well. you?
The other user says: hahah i'm great :)
good
i am glad to hear that
[ ]
```

The server is pictured on the left, client 1 and 2 are on the right. As we see, the server can handle two different clients. This and our login functions are our main breakthrough in our alpha.

3.4 Photo Filters

Our program will allow users to add basic filters to images (black and white, polarized, etc.) they may want to share to the people they are chatting with. A notification will appear to asking if the user would like to apply the filter.

DISCLAIMER: For this release, the feature has not been implemented yet.

3.5 Friends List

Every user will be allowed to send and receive, accept or decline friend requests, which has not been fully implemented yet. Your friends list will be organized according to their status if they are currently online, idle, or offline.

DISCLAIMER: For this release, the feature has not been implemented yet

4 Help and Error Messages

- “Username successfully registered. You can start using Remarq.”
 - This message appears when the user registers a new account.
- “This username already exist. Please retype a new username”
 - This error appears when the user tries to register a username that is already registered in the server.
- “This username is not valid. Please retype a new username”
 - This error appears when the user tries to register a username that is not valid such as: it includes space, or special characters.
- “Invalid username or it doesn’t exist. Please retype the username”
 - This error appears when the user tries to sign in with a username that is not registered in the server.
- “Wrong Password! Please retype the password”
 - This error appears when the user inputs a password that is not assigned to specific user name.
- “Forgot your password?” link
 - System dialog will prompt the user to register for a new account.
- “Rejected”
 - Someone doesn’t want to be your friend
- “Friend Request accepted”
 - You’ve made a new friend

5 Copyright

This installation or use of this game is not guaranteed to be bug free, use at your own risk. No refunds, and any user of this application cannot sue our company. We are not liable for any harms done to your machine. All passwords are encrypted and not saved onto our servers for safety purposes. All rights reserved. Copyright.

6 References

Terminology and definitions in C. Doemer, Rainer. Lecture. UCI. 2017-2018

7 Index

<u>Item</u>	<u>Page</u>
Algorithm	10
Chat History	9
Copyright	12
Client	10
Filters	11
Forgot Password	10
Installation	10
Glossary	3
Graphical User Interface (GUI)	4
Host	10
Idle	7
Messaging	4
Notifications	9
References	12
Status Bar	11
Unique	10
Uninstall	10