

README

Last Updated: 2023.5.4

If you are the course TA and need to grade this project, please skip my nonsense in the introduction and jump to "Getting Started".

Table of Contents

1. [README](#)
2. [Introduction](#)
3. [Assessment / Function List](#)
4. [Tutorials](#)
 - a. [Getting Started](#)
 - b. [Multi-Client Chat](#)
 - c. [Other Functions](#)
 - i. [Sending Pictures and Files](#)
 - ii. [Message History](#)
 - iii. [Actions Menu](#)
5. [Techniques Details Explained and Reflections](#)

Introduction

This is a Java Project for CUHK-SZ Year-1 Course CSC1004. It takes 60% of all assessment scheme. For more information, please visit [Course Website](#) and [Project Requirements](#).

This project was finished in April 2023, if you see this project after 2023, all code here cannot speak for my coding skills!

This Project implements a Java Chatroom that can support multi-user chatting. However, this is just a toy example and poor imitation of a real chatroom and can only run on one's own laptop and currently don't have a cloud server. So it cannot serve any real-world purpose.

All code are open-sourced with as-much-as-possible-user-friendly comments. Most of the code are self-explanatory.

All code were written in Java. The GUI was written in Java AWT and Java Swing. The knowledge involved also includes Java Socket Programming, JDBC (Java + mySQL), IOStream, multi-thread Programming and so on. Solid Java SE knowledge is also expected in many details like Collection.

The most tricky part in my mind was the IOStream because there're too many of them and I also had to combine them with Socket Programming and Network Knowledge. Some day I spent six hours on the messaging function but finally in vain... It took me around 50-60 hours to finish (including reviewing Java Socket Programming knowledge) this sixty-percent-of-one-unit project.

For more detailed explanations about the code, you may jump to the last part of this README.

If you are a future student in CUHKSZ learning CSC1004 and encounter problems about this project, feel free to contact me through school email.

ID: 122090180

Points gotten in this project: ? / 100

Assessment / Function List

See [here](#).

Common Grading (20 pts)

- ✓ Code Documention 5 pts
- ✓ Video 5 pts
- ✓ Tutorial for Running 5 pts
- ✓ Can run 5 tps

Multi-Client Chat

1. Multi-Client Chat (20pts)
 - a. (8pts) A server that could support multiple clients to communicate with each other
 - b. (8pts) Could generate multiple clients to work together.

- c. (4pts) The clients could send and receive messages simultaneously.
(The basic function of a group chat app works well.)

2. Login System (20pts)

- a. (5pts) Having a database to record user information.
- b. (4pts) The database works well when the login system visits the database for information.
- c. (4pts) A login system that could receive the inputs (e.g., usernames, passwords).
- d. (4pts) The login system could verify the combination of Username and Password.
- e. (3pts) The login system runs smoothly. We could start to chat after logging into the system.

3. Java GUI (20pts)

- a. (10pts) A GUI for the client (5pts) and A GUI for the login System(5pts).
- b. (4pts) Buttons(1pts) and the buttons work well (3pts).
- c. (4pts) Text fields (1pts) and they work well(3pts).
- d. (2pts) The designs are user-friendly.

4. Advanced Features (20pts)

- a. Registration System (5 pts)
- b. Emoji (5 pts)
- c. Sending Pictures (5 pts)
- d. Message History (5 pts)
- e. Animation (<5 pts) I've implemented the login & Registration animation, but failed in Chatroom window... If partial points would be given I'll be appreciative because the first two already took much much time. But I think I've accomplished the first four, which should be enough.
- f. Password Recovery (no point hereafter). You could input your Username, Email , Old Password and Security Questions you set when signed up to recover your forgotten password!
- g. Verification Code. Clients are required to input a verification code to prevent someone from using brutal password force attack.
- h. Remember the password. If you ticked the checkbox and login, you can avoid inputting your password again and again on your computer.

- i. Look up and Update your profile. After login, you could feel free to update your personal information, which will be recorded by the database, too.

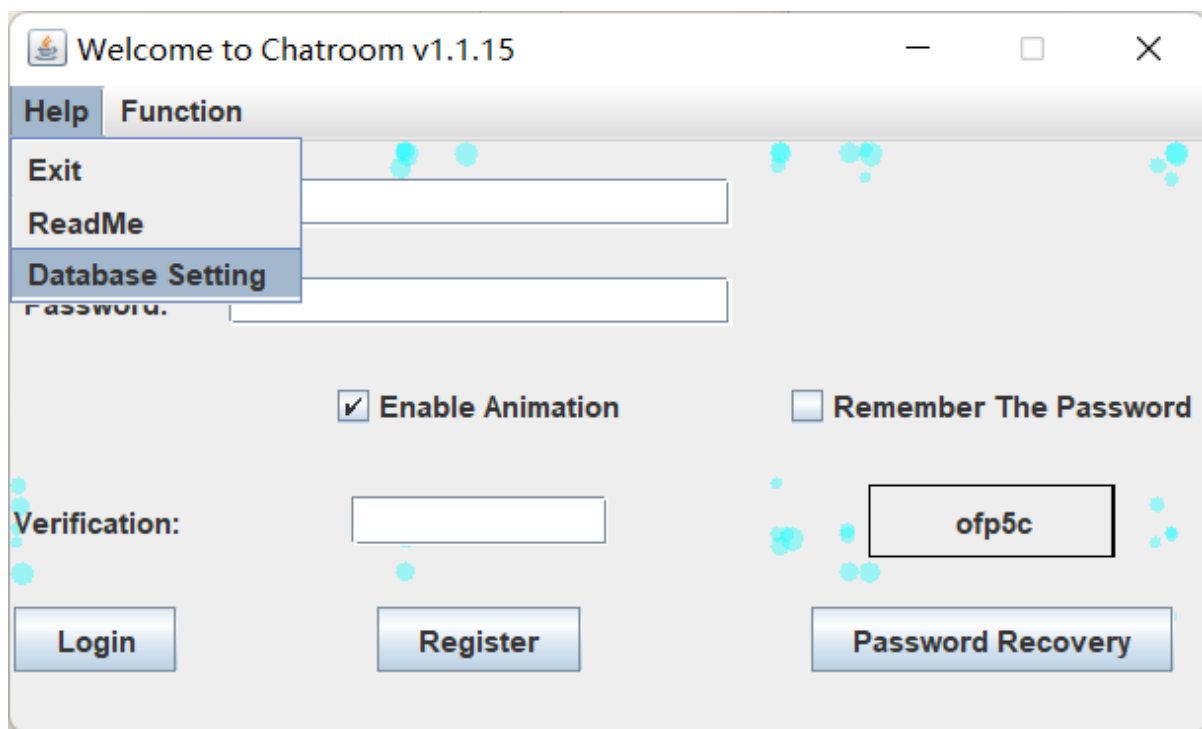
Getting Started

Update 23/4/19: The video tutorial is uploaded and you can see it in the resources folder.

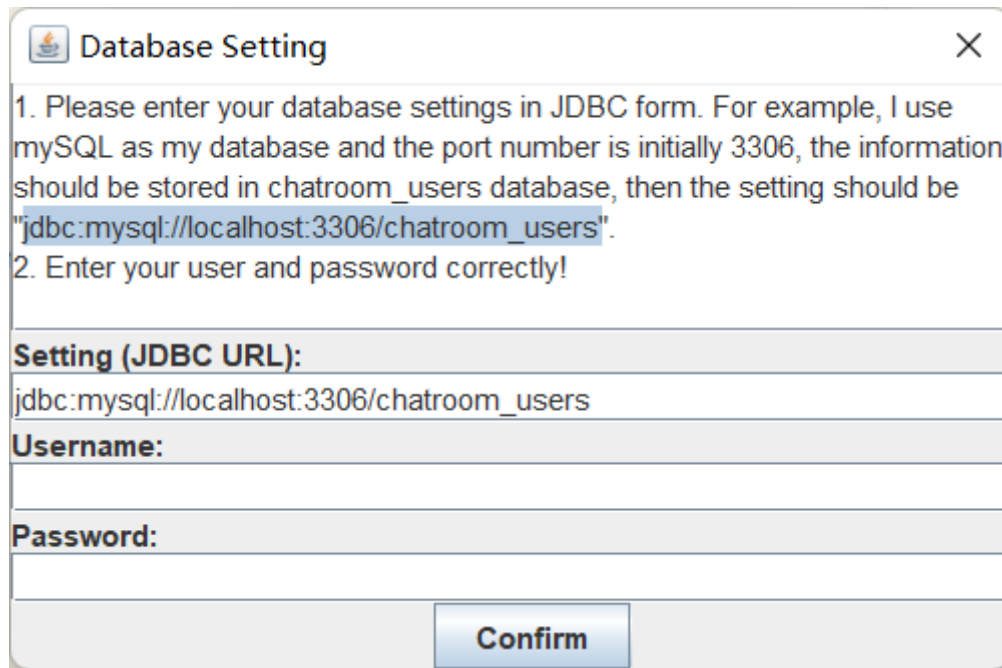
Update 23/4/19: The Bi-weekly report which takes 10% of the assessment was also uploaded for anyone's reference. I actually don't follow the schedule I made at the first week. Plans always change, don't they?

Let's see how the project is implemented.

If you're trying to run the project on your own computer, you should set up your database first!



First, click on Database Setting. You should **copy the setting** for now because the application doesn't support other databases.



Database Setting [X]

1. Please enter your database settings in JDBC form. For example, I use mySQL as my database and the port number is initially 3306, the information should be stored in chatroom_users database, then the setting should be "jdbc:mysql://localhost:3306/chatroom_users".

2. Enter your user and password correctly!

Setting (JDBC URL):
jdbc:mysql://localhost:3306/chatroom_users

Username:

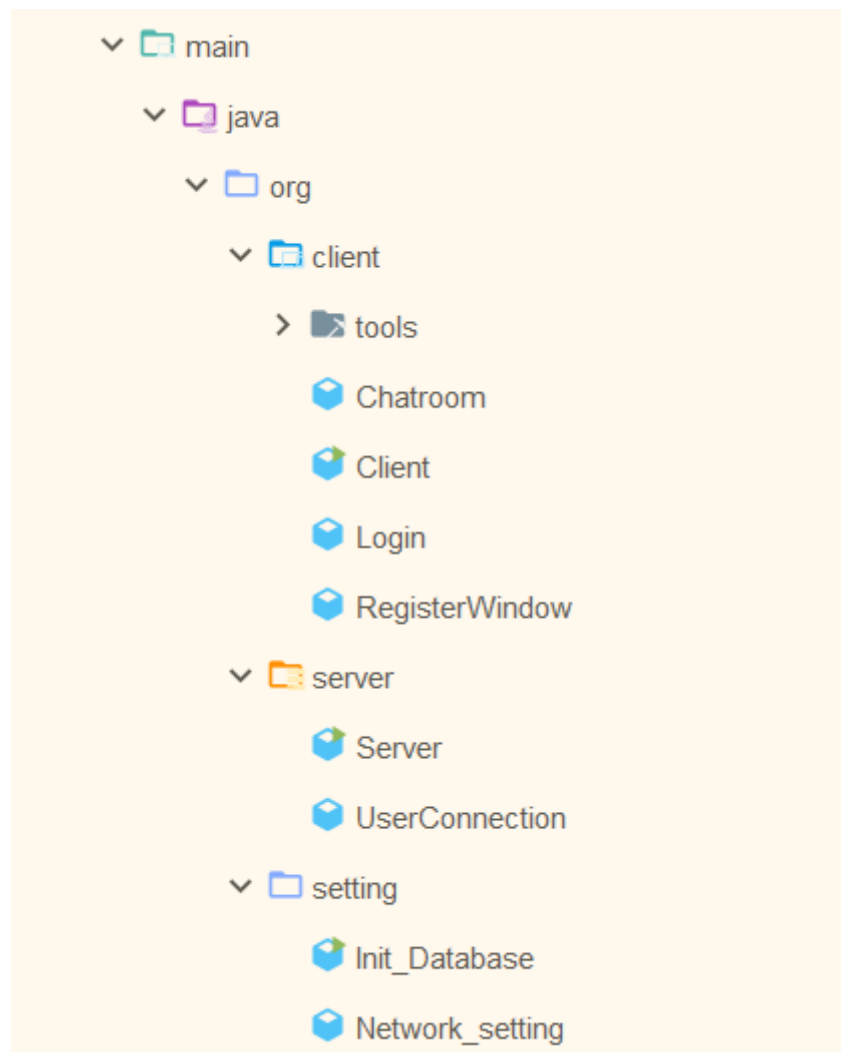
Password:

Confirm

The URL should be modified only when the port number is not by default 3306. Then input your mySQL username and password.

~~Make sure you **don't** have a database called "chatroom_users" now!~~

Update 23/5/4: The Init_Database process is updated and now it's easier to use. It will automatically drop the database "chatroom_users" such that it's easier to test the code. Meanwhile, the bug that failing the connection when you test on your own computer is fixed.

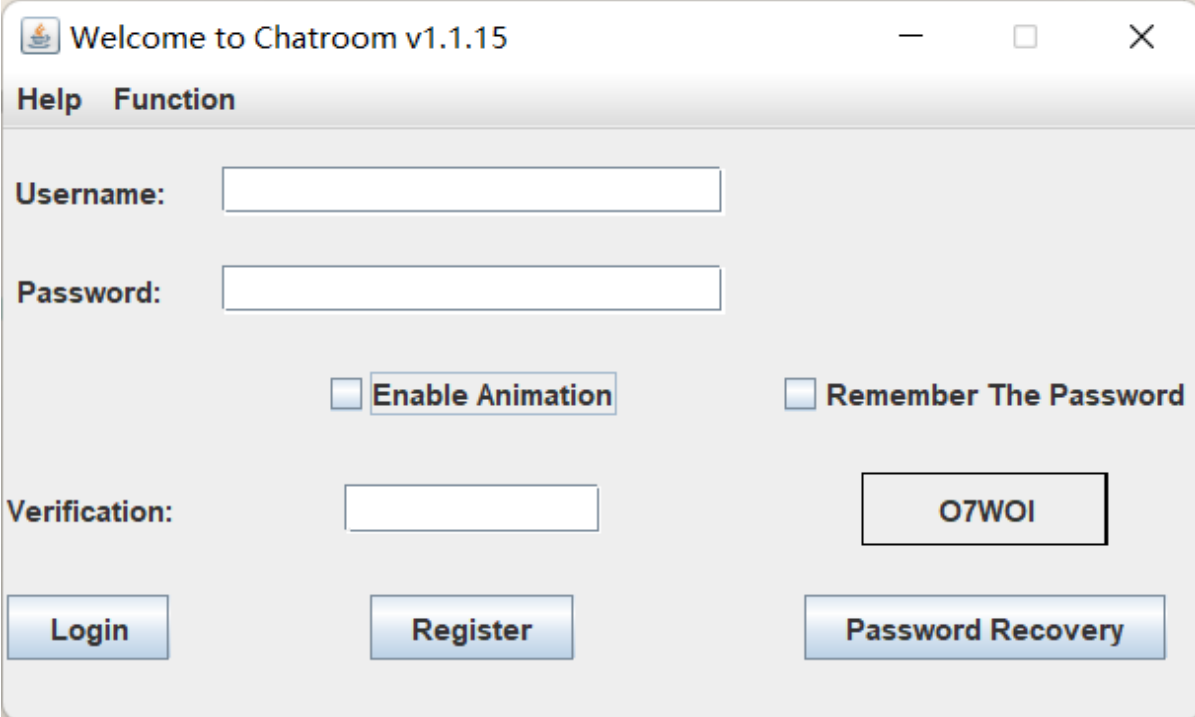


Look at the Project Structure, the main code are divided into three parts. You should know click on setting/Init_Database, and run the main method directly. The database and tables needed are prepared automatically.

The other two runnable programs are "client/Client" and "server/Server". Please run them.

The server doesn't have a GUI and logs will be outputted in the system console. You don't need to understand them in order to use this application.

The client side should look like this.



The image shows a window titled "Welcome to Chatroom v1.1.15" with standard window controls (minimize, maximize, close). Below the title bar is a menu bar with "Help" and "Function". The main area contains the following elements:

- Username:** A text input field.
- Password:** A text input field.
- ☐ **Enable Animation**
- ☐ **Remember The Password**
- Verification:** A text input field.
- A button labeled **O7WOI**.
- Three buttons at the bottom: **Login**, **Register**, and **Password Recovery**.

There's a stupid animation here I used to fill the assessment scheme in the past. You can turn it off with the "Enable Animation".

The GUI is self-explanatory.

Register for new

Notice:
1. The length of username/password should be at least 6 and at most 20.
2. Username/password should only contain numbers, letters and '@#\$%^&_'.
3. The items with '*' are not required.
4. If you prefer not to say your age, fill out -1.
5. Question and answer are used to recover your password. If you are forgetful, please fill out the answer carefully and do remember your email, too!

username

password

confirm passwd

age(-1~120)

Sex

☐ Male

☐ Female

☐ Other

☒ Prefer not to say

E-Mail

Question1*

Answer1*

Question2*

Answer2*

Country*

City*

Introduction*

CLEAR ALL

REGISTER!

The Register Window also has the stupid animation, which I'm too lazy to set a checkbox to close.

Please follow the instructions to fill out the form in order to sign up. After registering successfully, your data will be stored in your database so that you can directly login after this time.

Note that if you want to recovery your password one day, you should remember your username, e-mail carefully! Additionally, you should fill in **at least one** question to secure your account. The '*' items are not required, though. Here's a sample of Successful Signing Up. If you click and nothing happens, you may have too many windows on your desktop, you can search for the sign-up information dialog.

Register for new

Notice:
1. The length of username/password should be at least 6 and at most 20.
2. Username/password should only contain numbers, letters and '@#\$%^&_'.
3. The items with '*' are not required.
4. If you prefer not to say your age, fill out -1.
5. Question and answer are used to recover your password. If you are forgetful, please fill out the answer carefully and do remember your email, too!

username

CSC1004

password

.....

confirm passwd

.....

age(-1~120)

15

Sex

☐ Male ☐ Female ☐ Other ☒ Prefer not to say

E-Mail

CSC1004@cuhk.edu.cn

Question1*

The name of your university.

Answer1*

CUHK, Shenzhen.

Question2*

The school you are in.

Answer2*

School of Data Science.

Country*

China

City*

Shenzhen

Introduction*

CLEAR ALL

REGISTER!

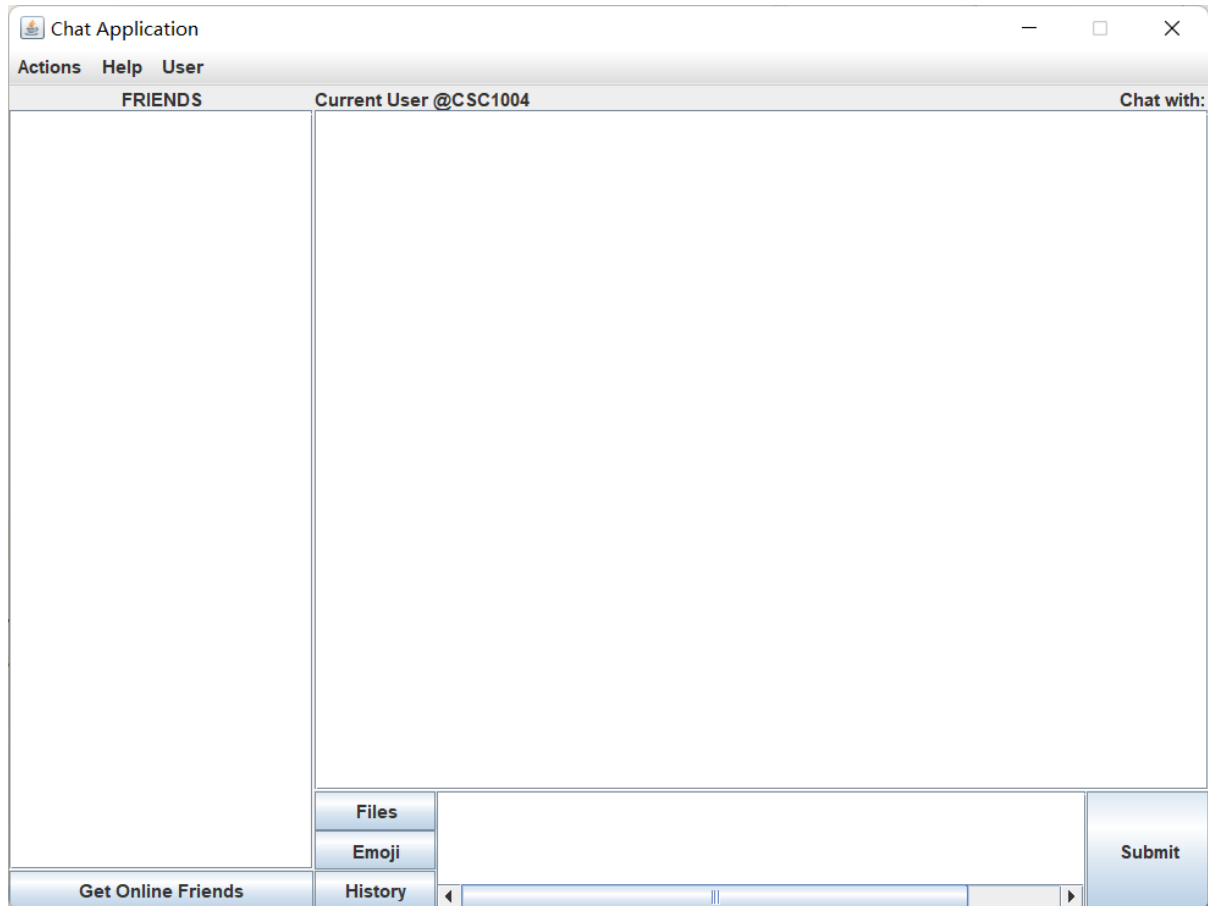
Success!!!

Great!

Then you should enter your username and password, you can also tick the "Remember The Password" to avoid entering them next time.

If you stumble over the Verification code for "I11" issue, you can click right on it to change one. The code is not case-sensitive.

Now you should enter this GUI!



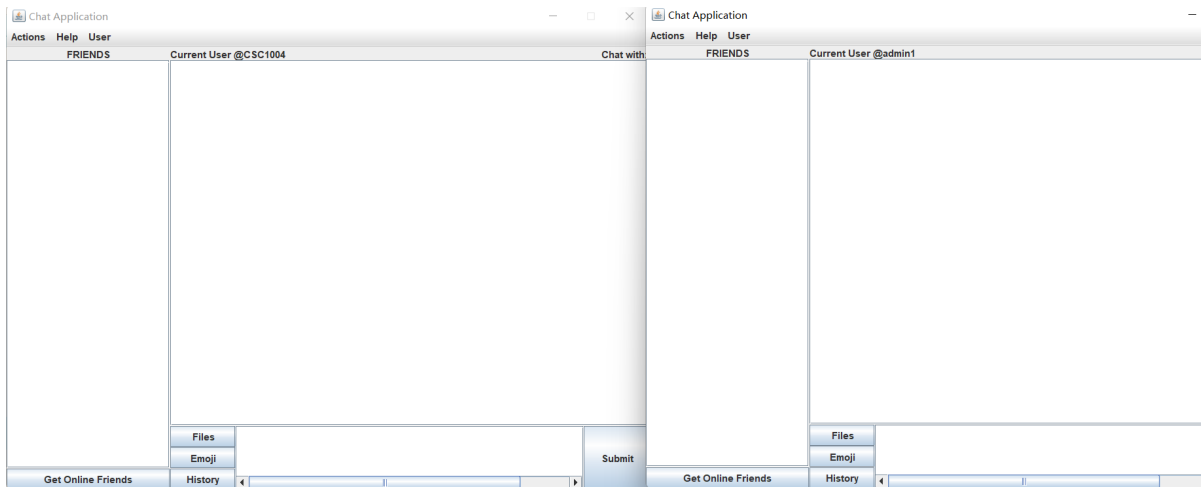
Multi-Client Chat

To generate multiple users to simulate multi-client chat, you can click on "New User Login" to arouse another login window, you can login with another user. **Don't use** your current user! This will cause confusion when chatting (you cannot get yourself while Getting Online Friends).

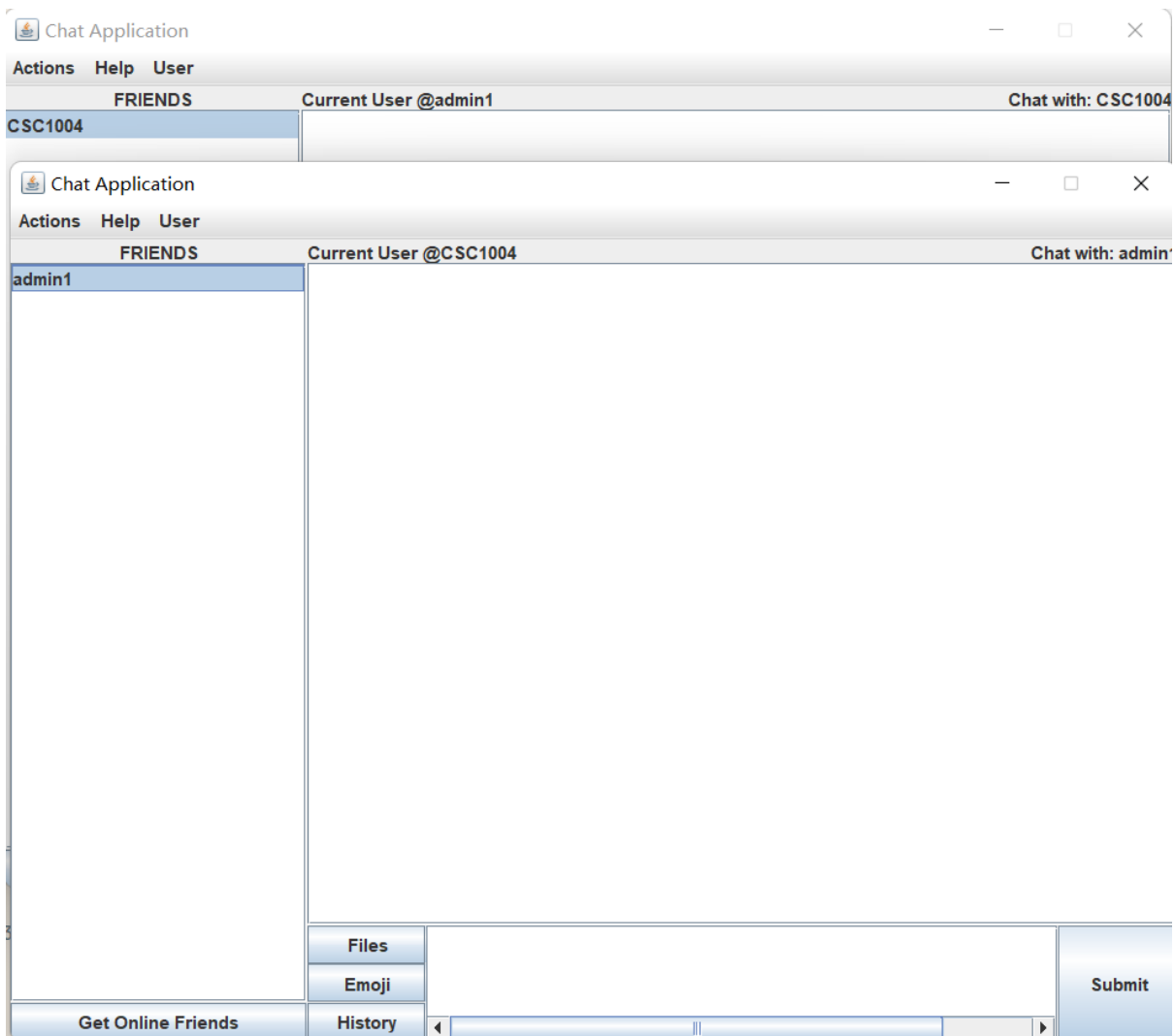
update 23.4.19: The confusion here is solved in the newest version. If you have logged in, you'll be banned from logging with the same user!

The login window may be hidden behind your Chatroom window.

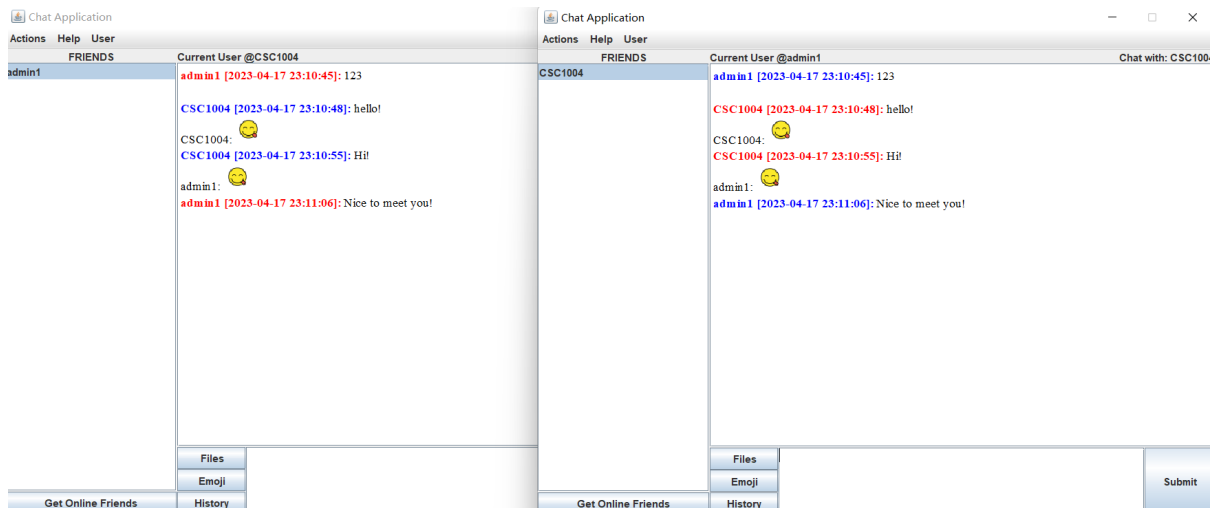
Now you should have two users online!



Click on both sides' "Get online Friends", and you should have each other on the list. You should click to update the list whenever you are about to say something to your friend, because they may already be offline!



Click on the friend you want to chat with, and then you should see both your name and your friend's name on the top corners.



Now you should send what you want with your friends! If you type some text messages, use Ctrl/Command + Enter should also work!

Other Functions

Sending Pictures and Files

Use the "Files" button to choose an image.

The files function will do for all kinds of files. Folders are not supported currently. You can compress it into a zip/rar file, though.

If you send a picture (image files, jpg/jpeg/gif/bmp/tif/tiff), both you and the receiver can see the pictures on the screen and the receiver will get a reminder asking him/her whether he/she wants the file to be saved. Since Files, including Images **will not be recorded by the history** function, you should save the file if necessary, you cannot save the image after this chance, either!

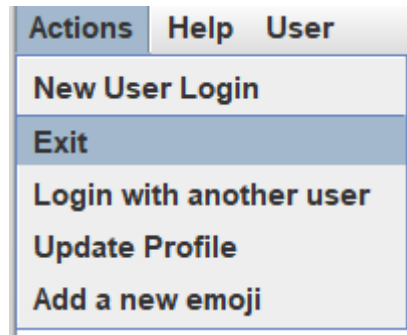
Currently if you save a file that has identical name with another file, it WILL cover the other one!

Message History

Your conversation with another user will be faithfully recorded even after you exit because they are stored in the database.

Note that emoji and files will not be recorded. The former is not important (I think) and the latter one is technically hard to maintain in databases. I'll try to improve this in the future if possible.

Actions Menu



"New User Login" will arouse another login window except the current one as I've mentioned before. Meanwhile "Login with another user" will exit the current user and arouse the new login window.

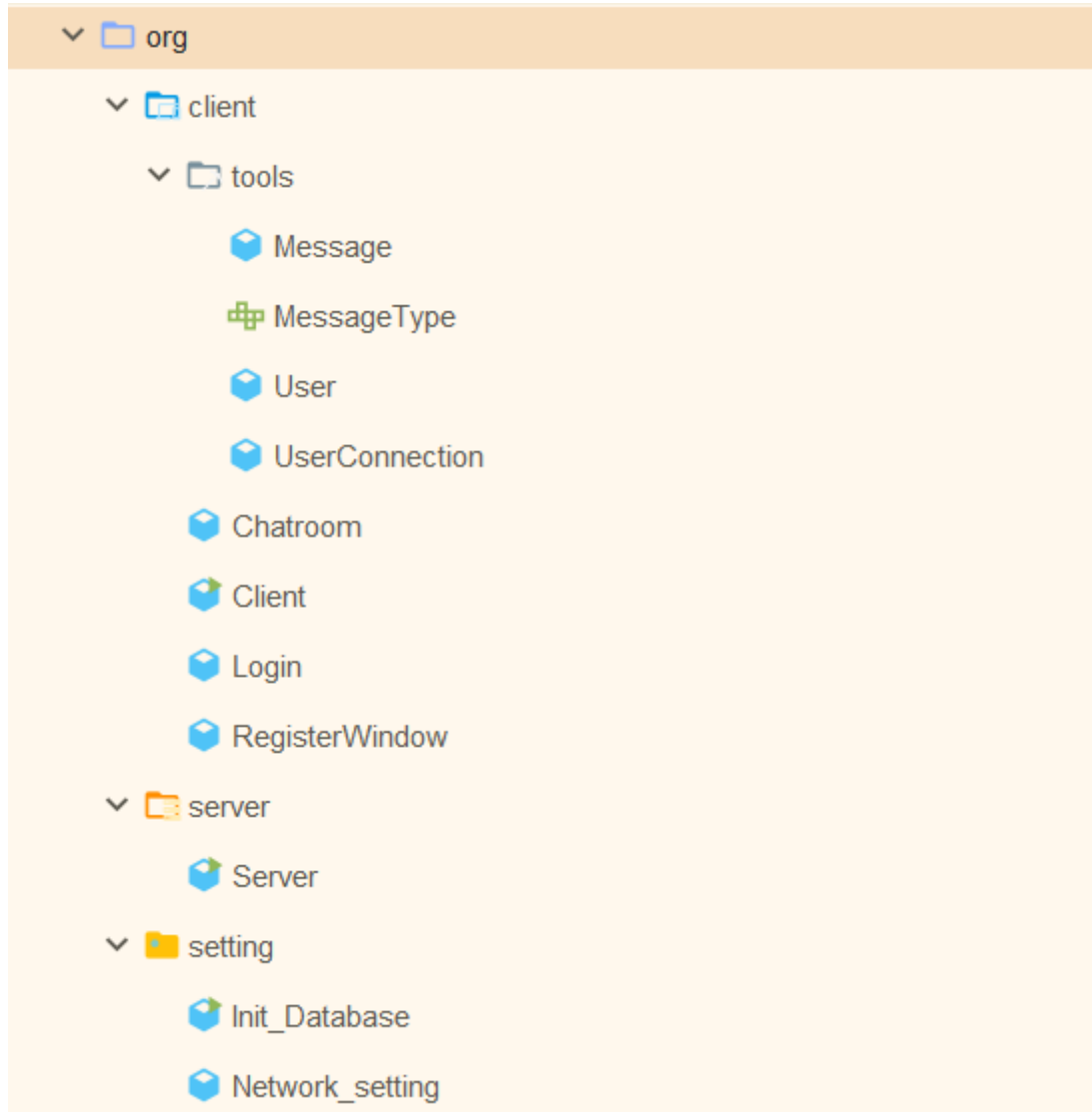
"Update Profile" item can update your information easily. This, for sure, will be recorded by the database.

"Add a new emoji" allows you to add whatever you like in the emoji set. Of course only images are allowed. The emojis are in reality stored in resources/emoji folder, the GUI doesn't support deleting currently. If you want, go to that folder.

Techniques Details Explained and Reflections

Here I'll briefly introduce how my code work in this project.

Let's First look at the source code structure:



The two classes in setting package are introduced in the using tutorial, they are designed to make sure you can run this project on any computer that's equipped with a mySQL. No other purpose served.

The tools are several classes and interface I use in the application.

The Message and MessageType are the most important. They are a protocol that client and server side agree on while sending messages. The Message Class is serializable and can be transmitted using IO stream writeObject() while carrying anything we want to tell the other side. The MessageType points out which kind of message it is. It's actually used to facilitate the coding, in essence they are just Strings. Please don't ask me why I don't use the Enum.

At first I actually provided a rather narrow interface, but I had to add in things like FileExtension, fileName, userList stuff which should not appear in a "Message" class to be honest. Here I take a shortcut to avoid using inheritance & polymorphic, and that's not the best practice in OOP programming. Maybe I'll update this if possible in the future.

The User class is self-explainable. It stores the information of each user.

The UserConnection is an underlying structure to store both the User and the network settings. The class right now is not the initial design, either, for I can only access to the outputStream by passing it wrapped in object due to some unknown error (at least to me now) I encountered while trying to connect the user by IP and port. This might be improved in the future, either.

This class is used in both client and server side and should be synchronized any time possible. The logic is simple, it is at the first place maintained in the server side. When a user logs in, store the userConnection in a list and remove it from the list when the user exits the application. The client can apply for it by clicking on the "Get Online Friends" button by sending the message of type "GET_ONLINE_FRIEND". Then the list fetched by the message responded by the server with type "RET_ONLINE_FRIEND" is used to create the friend list implemented by the friendListModel in Chatroom.class. I won't explain the details of Java GUI(Swing) in this file because that's outdated techniques.

Currently the list should be manually fetched, but this can be improved in the future to be real-time.

The entrance of the application is Server.java and Client.java.

The basic idea of server is simple: maintain the online user list, transfer the messages. The latter is almost the most tricky part. I adopt a threadPool to support real-time chatting between several users. The server should be in charge of listening to incoming messages and dealing with them wisely. The Client side in a similar way. Once you solve the issues with Multi-thread and IO Stream, the other part of the project is actually a matter of time.

I printed many messages in the console in both sides for the convenience of debugging, and I left them on purpose because they are convenient to demonstrate the problems in the programs if there should be future improvement.

The detailed explanation of how the GUI is created and how each kind of messages are dealt with is already in the code comments in an I-think-it-is-user-friendly way. Hope you enjoy reading them.

Other functions are self-explanatory in my opinion. If you have any issues/questions/suggestions about this readme file/my code/my application, feel free to contact me!