Communication Networks project: chat app

Kou Jialun

December 22, 2021

1 Introduction

This project applies some concepts taught in the course of Communication Networks to the practical example of a chat room. To this end, an application is made on which people can have conversations. A special focus lies on security: user authentication and message encryption are important parts of the assignment.

For the execution of the project, a combination of Socket.io, mongoosedb JavaScript, CSS and HTML is used. The idea is to create a website with registration/login and chatting functionalities.

2 Architecture

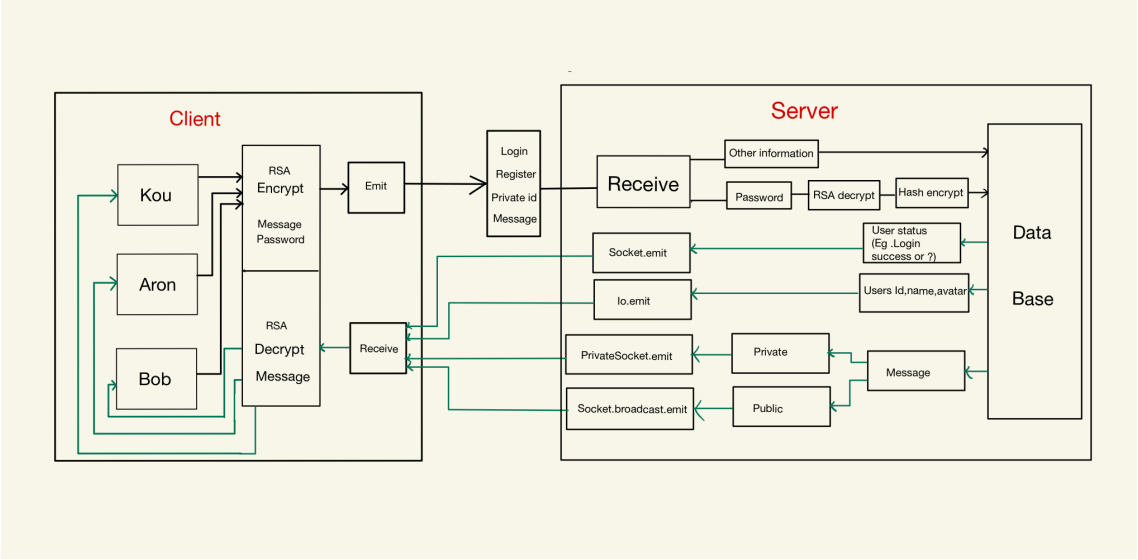

Figure 1: Architecture of code

socket.emit : information transfer object is the client corresponding to the current socket, and each client socket does not affect each other. this port will be used to transfer user status, eg. Whether the login password is correct,or whether the user registration is successful, etc.)

io.sockets.emit : The information transfer object is all clients. the user list will be sent to client through this port.

socket.broadcast.emit : information transfer object is all clients, excluding the client corresponding to the current socket. server will judge the value of Private id , if Private id is empty , the server will confirm this message is not private, then the message will be sent to all of other users through this port.

Privatesocket.emit : if the private id is not empty , server will transfer message to specific user based on Privateid through this port.

2.1 Register and Login

As shown in figure 2, every new user must register his username and password to be able to chat with the other users.

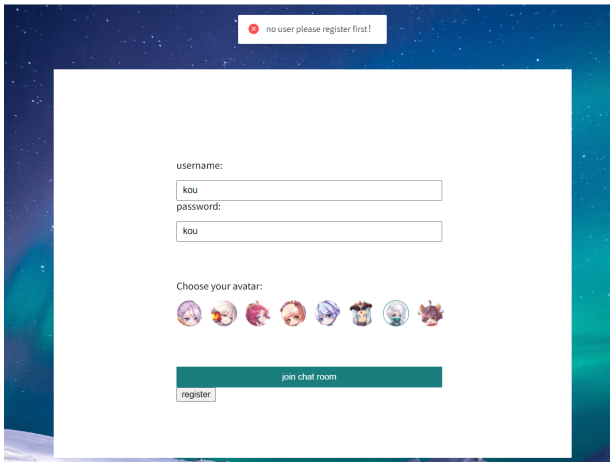

Figure 2: no user

2.1.1 Register

After user submit register information, the client will send the user name and password entered by the user to the server. If the user not exists in the Database, the user registration is successful ,in the same time , the user information will also be stored in database. Otherwise, the server will inform the client that the user already exists.

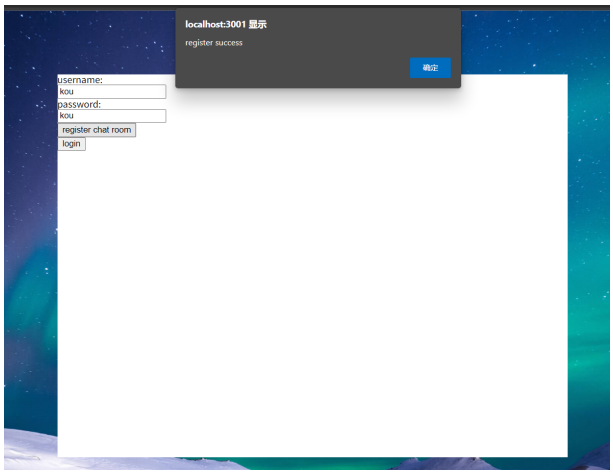

Figure 3: register success

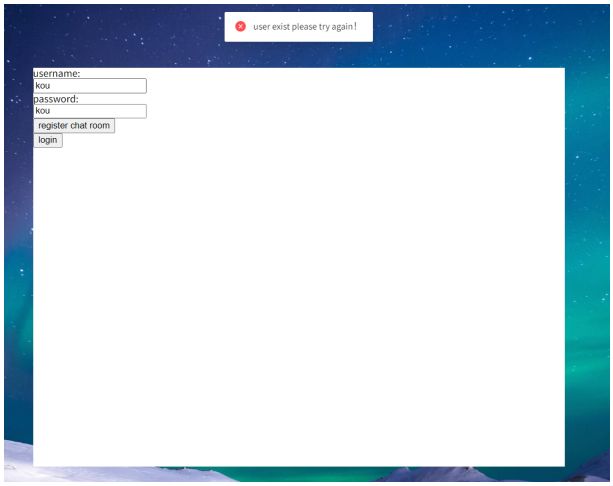

Figure 4: user exist

2.1.2 login

After the user completes the registration, the page will automatically jump to the login interface. After the user submits the login information to the server, the server will first determine whether the user exists in the database, if the user does not exist in the databasethe server will send back user status('please register first') to client (Figure 2). if the user exist ,the server will continue to compare the password submitted by the user with the password stored in the database, and send the comparison result back to the client through the socket.emit port. It is worth noting that if

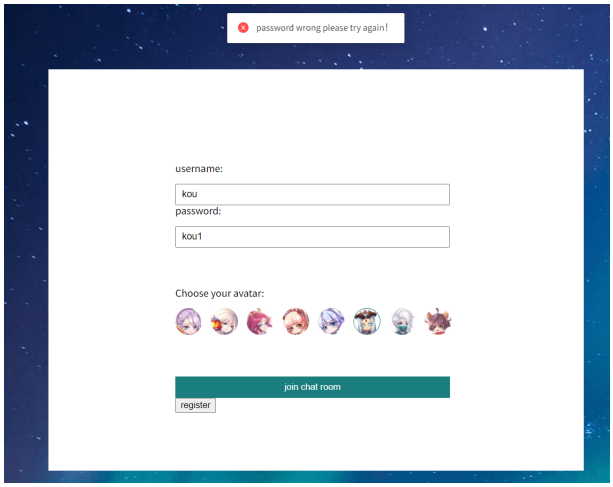

Figure 5: password wrong

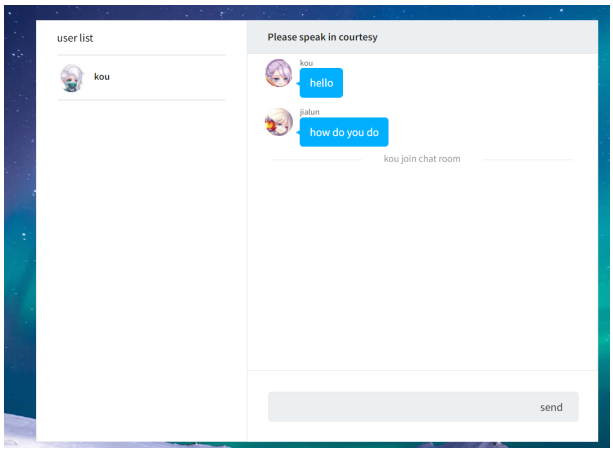

Figure 6: login success

the client sends a get request to the server to switch pages, the url path information will be exposed to the user, which will cause that the user can ignore the login interface and directly enter the chat page by sending a get request. So,in order to avoid that happen , We can put the login and chat pages into one html file, and open the corresponding page according to the user's status sent from server, and close other pages that are not related to this status.As shown in figure 7. In addition ,if user successfully log into chat room, the 'login success' user status and this user's name will be sent to client through io.socket.emit port, then all of the user who are connecting with server can know that a new friends join this chat room.As shown in Figure 6 , the notify message 'kou join chat room 'could be seen by all of users in chat room .

2.2 Chatting

2.2.1 user list

we define an array (users) in server.js file , when another user successfully log in , the new users name , avatar ,and user id will be added into users array , Then , this array as new user list will be sent to client through io.socket.emit . when client receive this new user list, the user list will

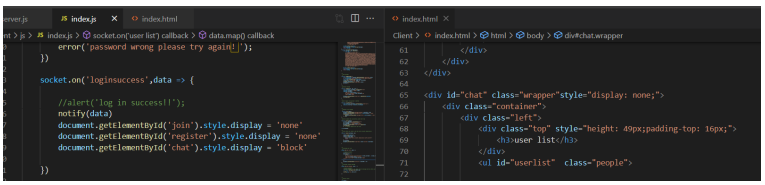

Figure 7: open chat page close login page

be updated and new user's avatar and name will show up at left window. compare figure 6 with figure 8.

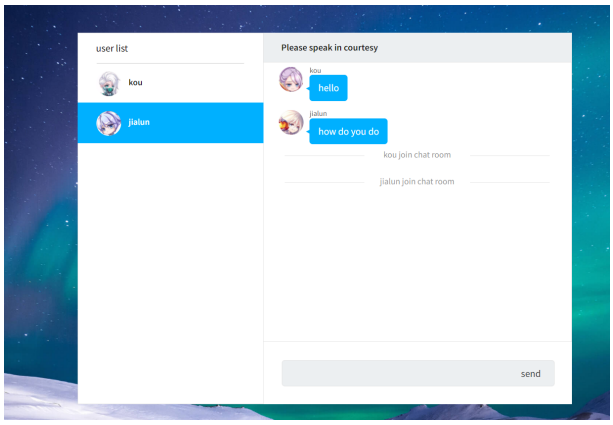

Figure 8: user list update

2.2.2 conversation

As shown in figure 9 , user 'Aron' click user' kou' ' avatar, then 'Aron' get the 'kou' 's username and id , We use kou's id as a private id then send message to server with this private id. at server side, if the received private id is empty , this message will be seen as public message, then the message will be sent back to client through socket.broadcast.emit port and this message could be seen by all of other users, if the private id is not empty , this message will be sent back to specific user according to private id through private.socket.emit port .

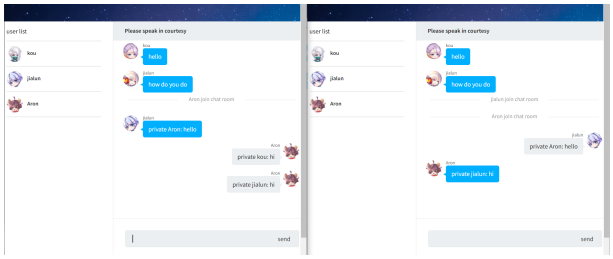

Figure 9: private communication

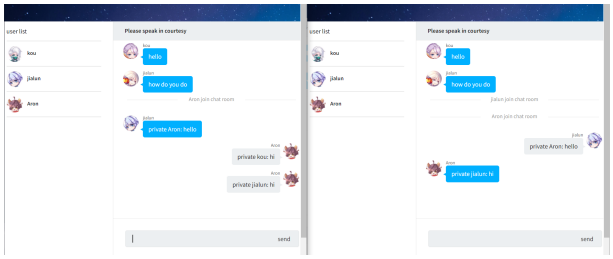

Figure 10: public communication

3 Security

3.0.1 RSA Encryption

RSA works by generating a public and a private key. The public and private keys are generated together and form a key pair.The public key can be used to encrypt any arbitrary piece of data, but cannot decrypt it.The private key can be used to decrypt any piece of data that was encrypted by it's corresponding public key.This means we can give our public key to whoever we want. They can then encrypt any information they want to send us, and the only way to access this information is by using our private key to decrypt it.

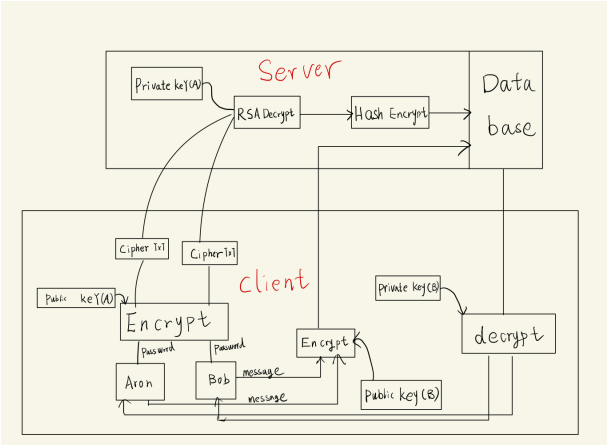

Figure 11: Encryption and decryption

A total of two pairs of keys were produced in this project public key (A) private key (A), public key (B) private key (B), Private key (A) will be stored in server , and the rest of key are stored in client.As shown in figure 11.

3.0.2 password encrypt

If user password is obtained by others in the process of transmission, it is likely to cause great losses to the user ,So, The user password should be first encrypted by using pubulic key(A) on the client side, and then sent to the server. and next the server will decrypt this password by using private key(A) , As shown in figure 12. In order to prevent the server or database from being illegally

```
[nodemon] restarting due to changes...
[nodemon] starting `node server.js`
http://localhost:3001
Encrypt password From client :  AKbm4aLrZpGutyGtsDY/AqeZ0GABYEsrgfGs6FBpRQOBYqPupyTk8rhDlmvvPMMnabtfxg6fQQtq1rHEBwntuCmIPpQqdV016BdGKa8y7+dM7EitSsIqyUQKUOKtq4X1G
7uwsYFqotSS4y99cymextNPqaFX8QmjSxxc1NkOHjs=
Decrypt password  :  kou
```

Figure 12: Encryption and decryption pass word

invaded and causing the leakage of user information, we cannot directly store the password in the database, so we should use hash to encrypt password again, and store the encrypted password into the database (Figure 13) . The advantage of this way is that even if the private key (A) and database information are leaked, the user password stil cannot be decrypted. In addition,When the user logs in, we can hash the login password first, Then compare the login password with the database password.If the encrypted passwords are the same, the user login is successful, otherwise it fails.

3.1 message encrypt

As shown in figure 14, the message will be first encrypted at client side by using public key (B), and then sent to server, the server can not read and store the original message , and then the encrypt message will be sent back to client , and then the message will be decrypt by using private key (B) at client side.

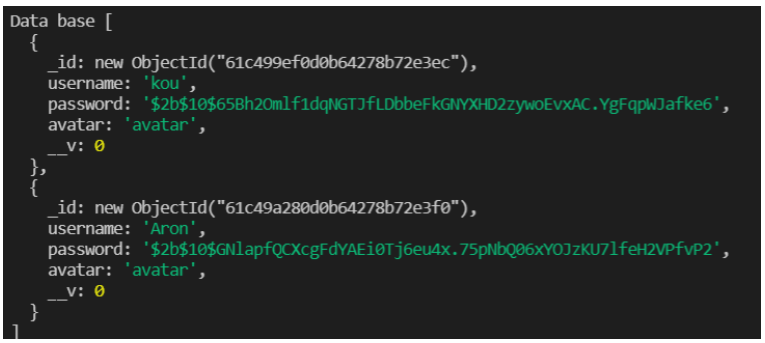

Figure 13: Data Base

Figure 14: Message encryption

4 Challenges

1.The first decision of this project, the choice of programming languages, created the largest amount of challenges and difficulties. With one group member very experienced in these languages, while the rest of the group had little to no experience using them, it was difficult to divide the work in a fair way.

2.When I try to refresh the user list, I need to use the map() function to loop through all user information in the database, but the data returned by the database is an object, and the map function can only traverse the array, So I redefine a users array, and store the username, avatar, and user id into this array, and then send this array to client , finally solved the problem.

3. at beginning, I have no idea about how to realize private communication by addressing username, After searching many ideas on internet ,, i found that Except addressing username , addressing userid is also a good choice to realize private communication , because the socket.id have these properties :The unique identifier of the socket session. Set after the connect event is triggered and updated after the connect event. so I set userid = socket.id ,In addition , i also find this function :io.sockets.sockets.get(privateid), io.sockets contain all of the client or sockets information in its entry, in the same time , the socket.id is the unique identifier for each user (each socket session = each user), so i could use id directly match the user that i want match, then i realize private communication by addressing userid.