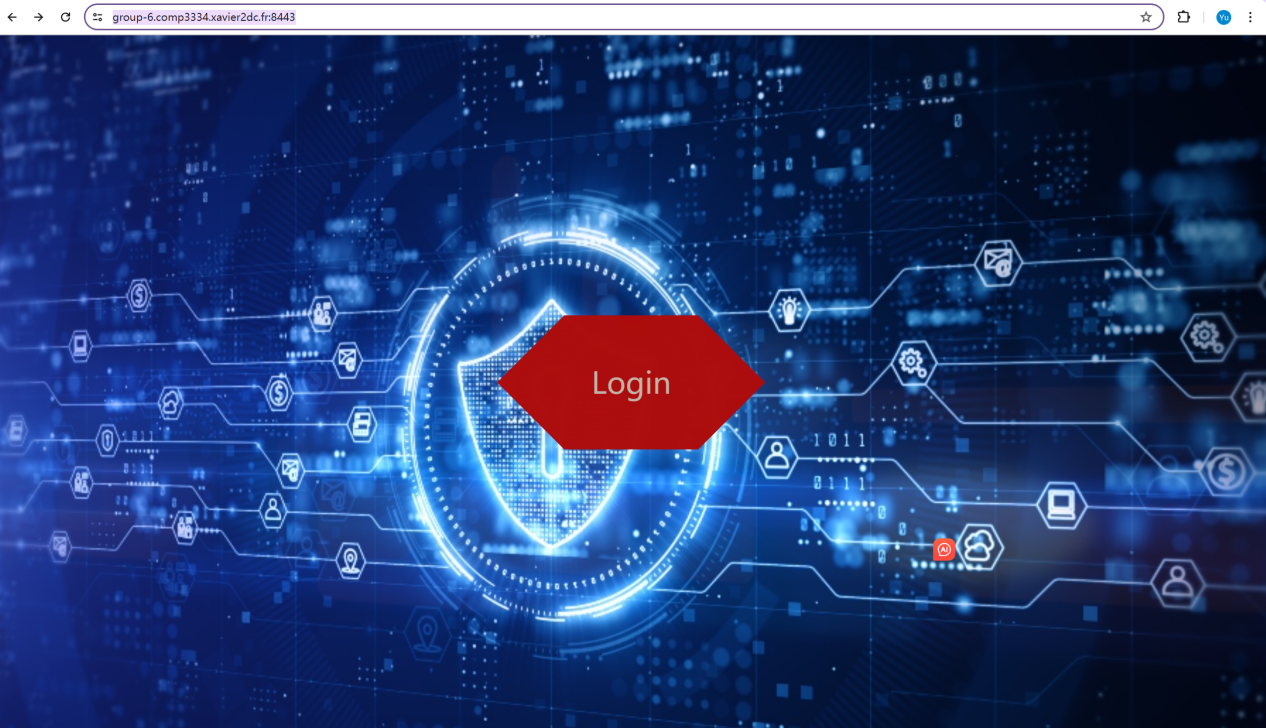
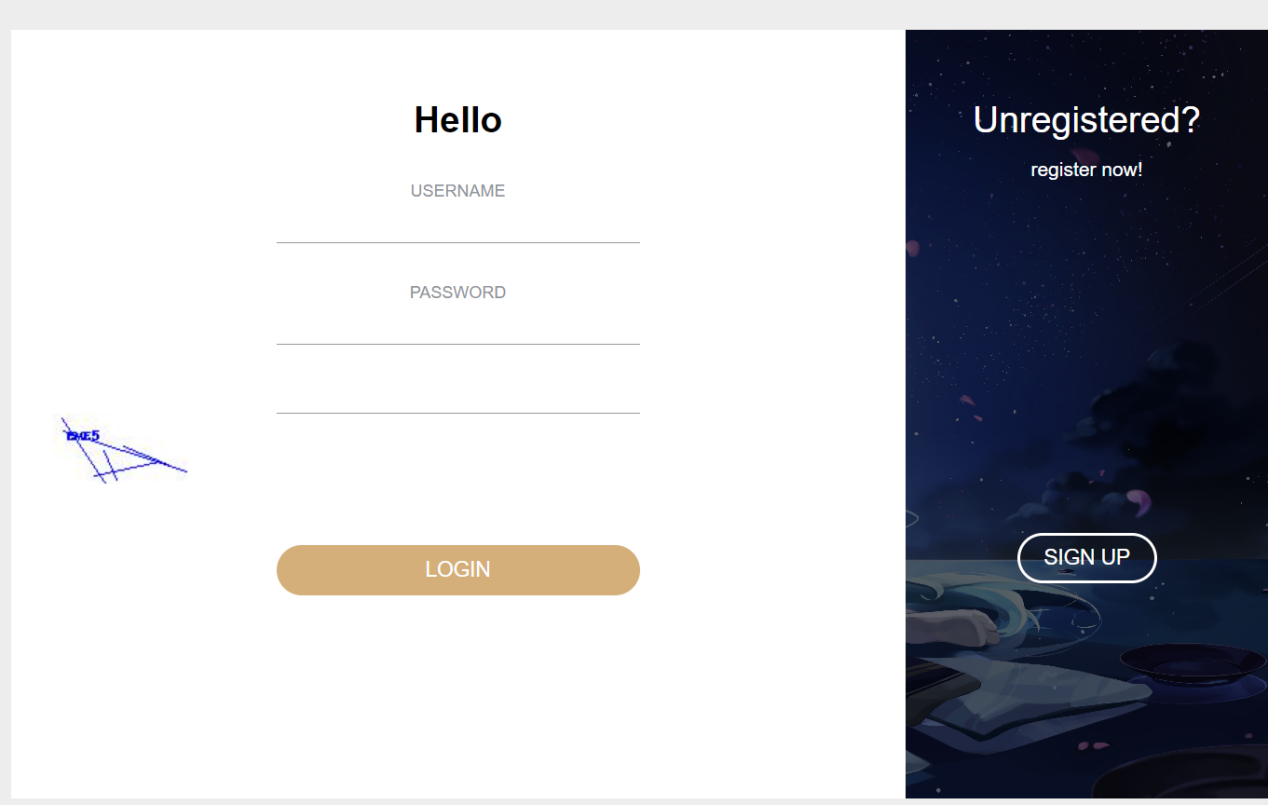
**Authentication**

First, we have the link to access the chat app: https://group-6.comp3334.xavier2dc.fr:8443/

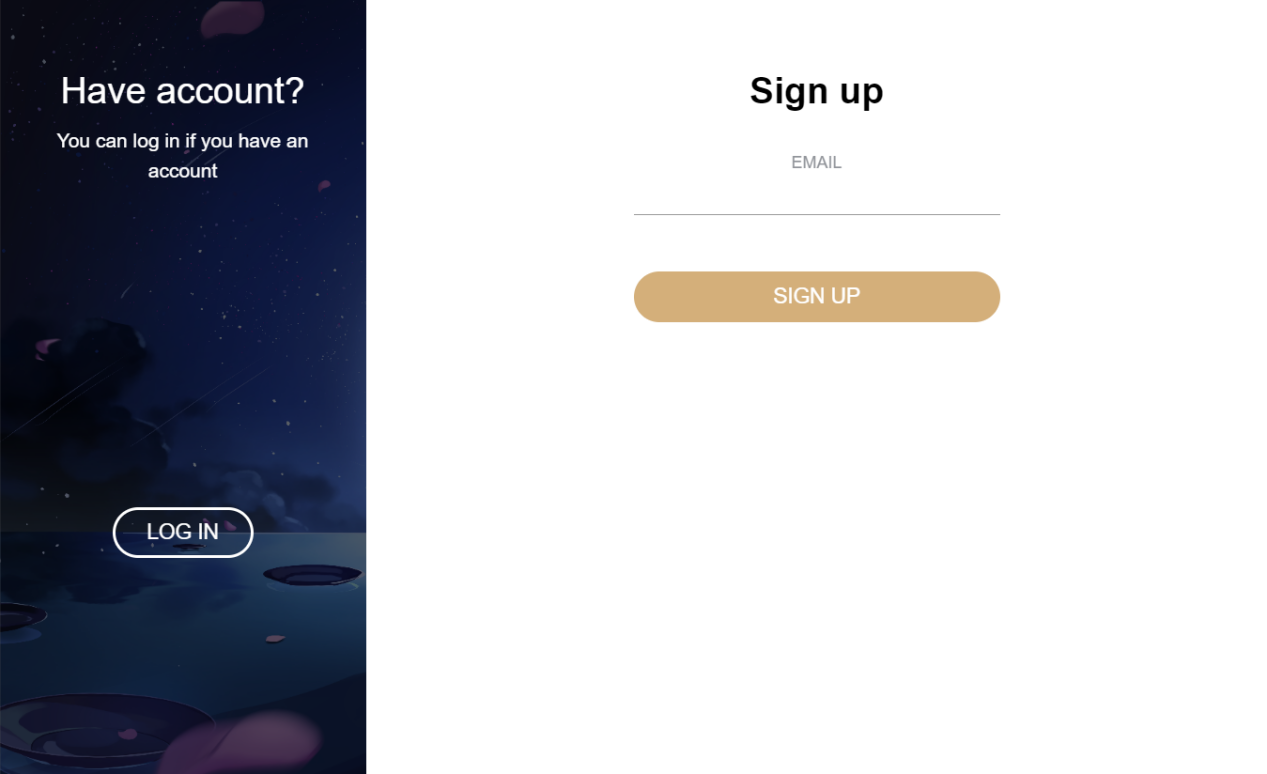
After the customer enters the link, he can see the home page of the chat app is shown below:



The customer can click Login and enter our chat app.

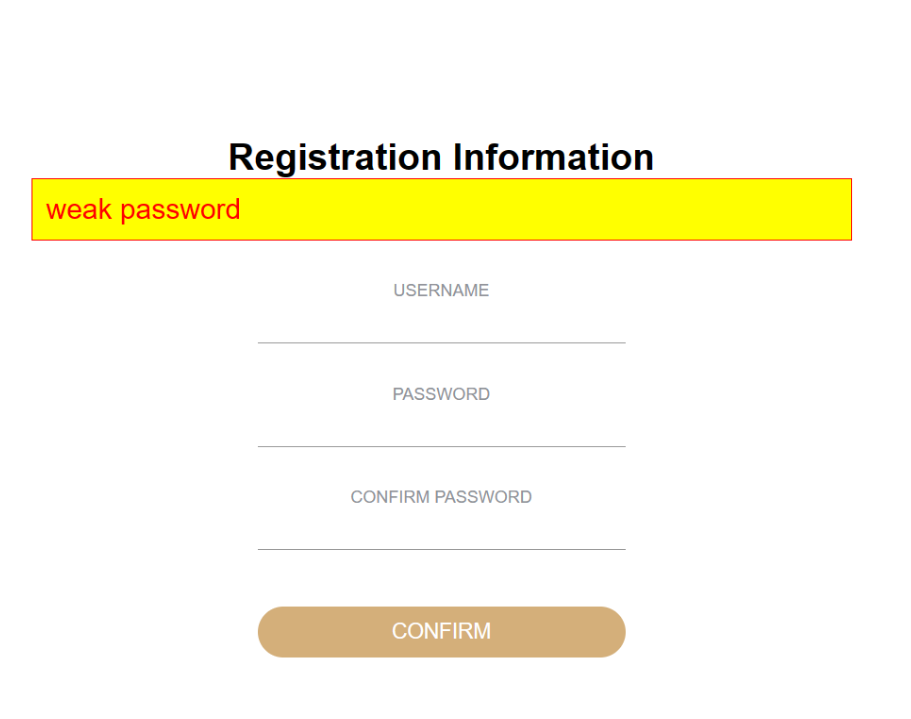


If the customer is a new customer, he may click SIGN UP to create a new account belonging to him.

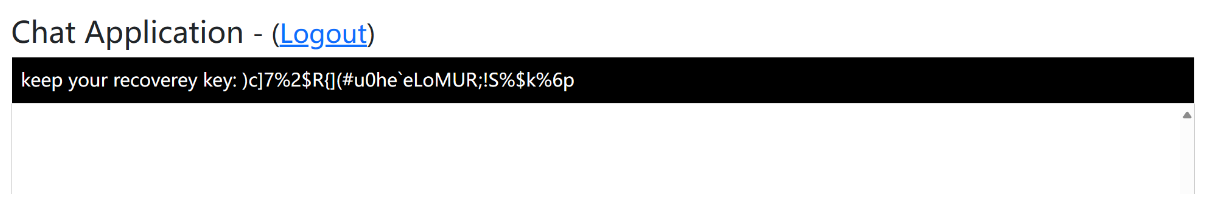


If the old customer clicks incorrectly, he has the opportunity to login by clicking LOG IN on the left.

For the new customer,he should use his email to sign up.For instance, if he has his own email [ksm6ule7@linshiyouxiang.net](mailto:ksm6ule7@linshiyouxiang.net,),he can input it and click SIGN UP.

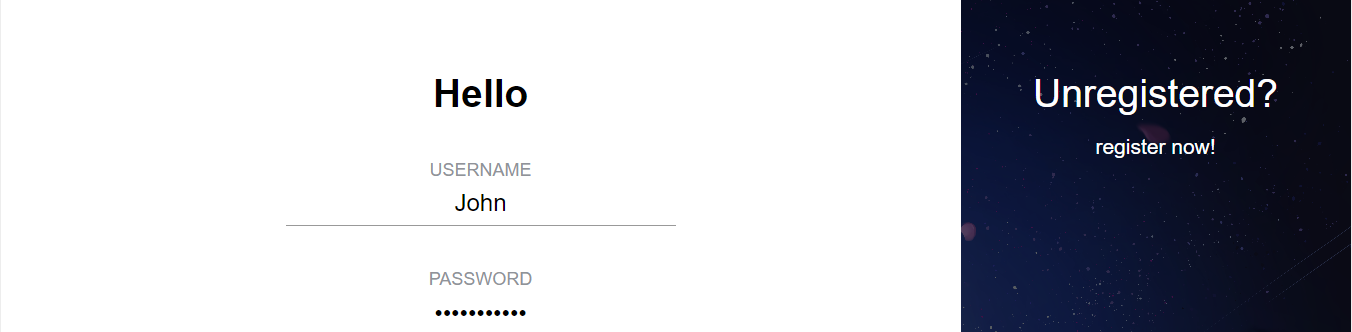


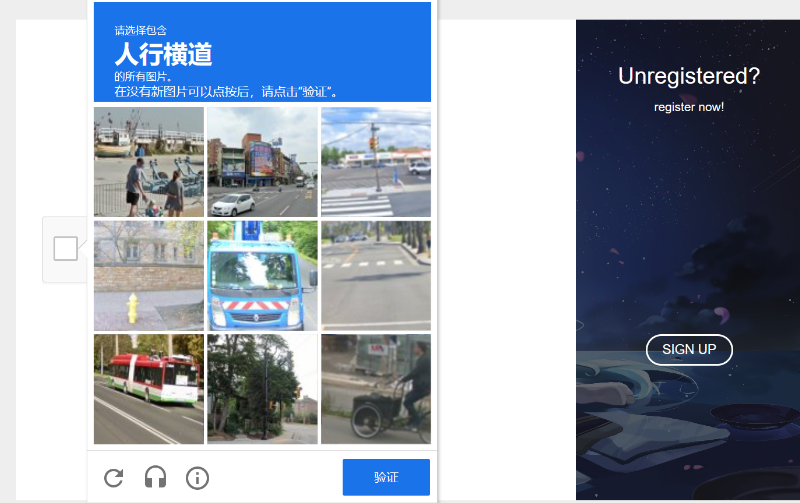
Then he can see the page above. Then the customer can enter his name and set his new password.If the name of the customer is John:However,for password,we have to ensure the password he sets is secure. If he set the password like 2,Q,12345,QWERT,Love520,1314520,the password is quite easy to decrypt.So he will see the sign of weak password.If he set the password secure:Hbc82706697,he can enter after conforming.An error will be reported if the two passwords do not match.



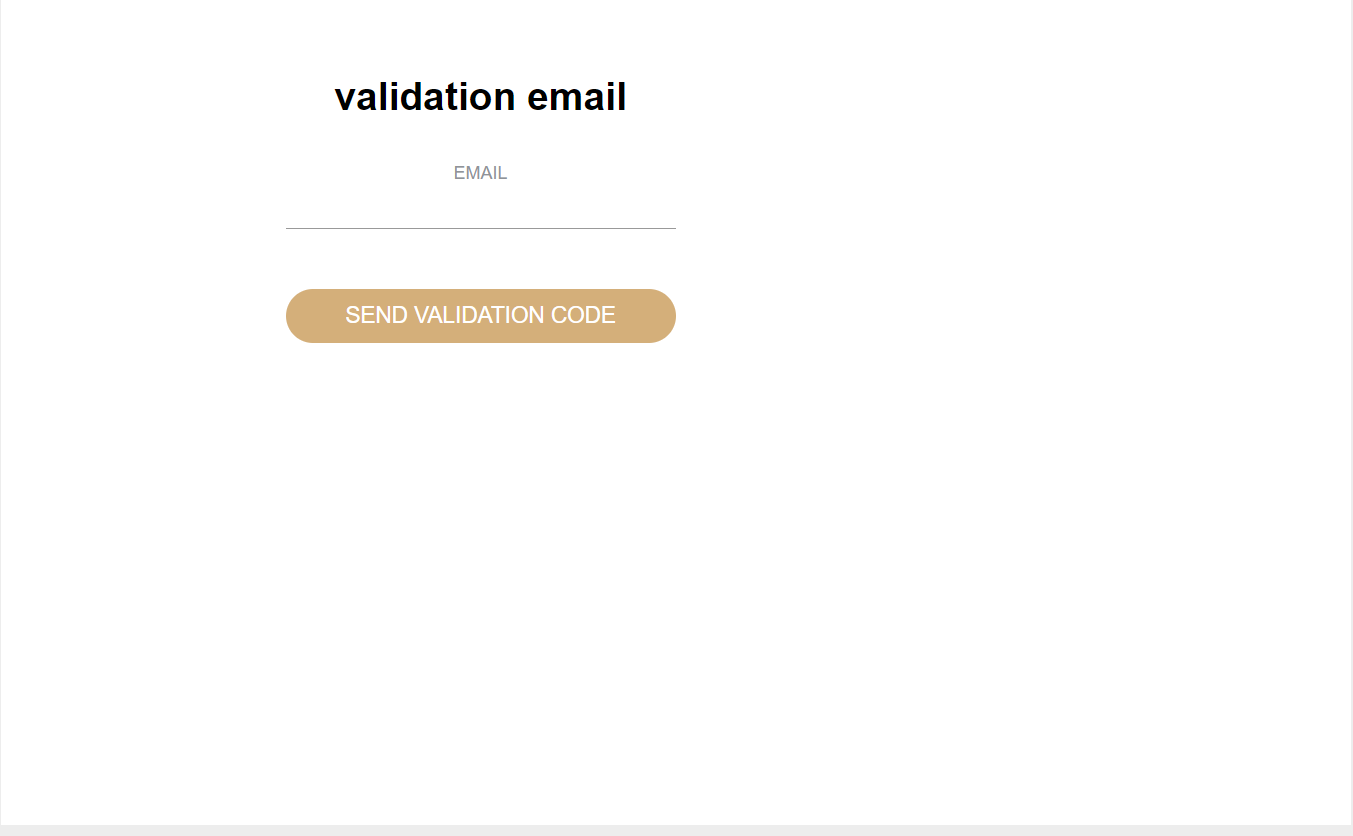
After that,the customer will see the page above. The recovery key can only show once, and it is quite hard to decrypt, human cannot use brute force to decrypt.Due to the fact that the length is larger than 24, also the character is random since there may include the English character,number and so on, it is also protected by secure protocol in storage.

When he wants to leave the app,he can click the Logout.Once the customer wants to chat with others, he can firstly choose a contact,secondly,he can type the message. No worry for the case that information cannot be withdrawn, he can click the Erase Chat to delete. Additionally, if he wants to refresh the recovery key, he can click the Refresh Keys. When he thinks that his chat is appropriate,he can click send.



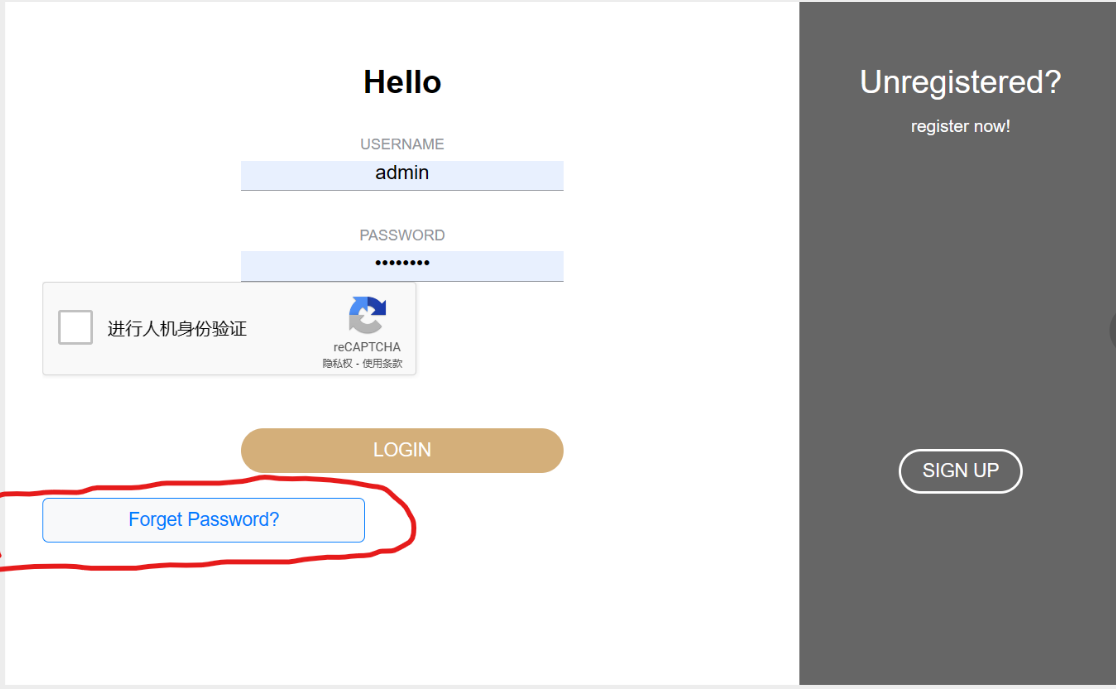


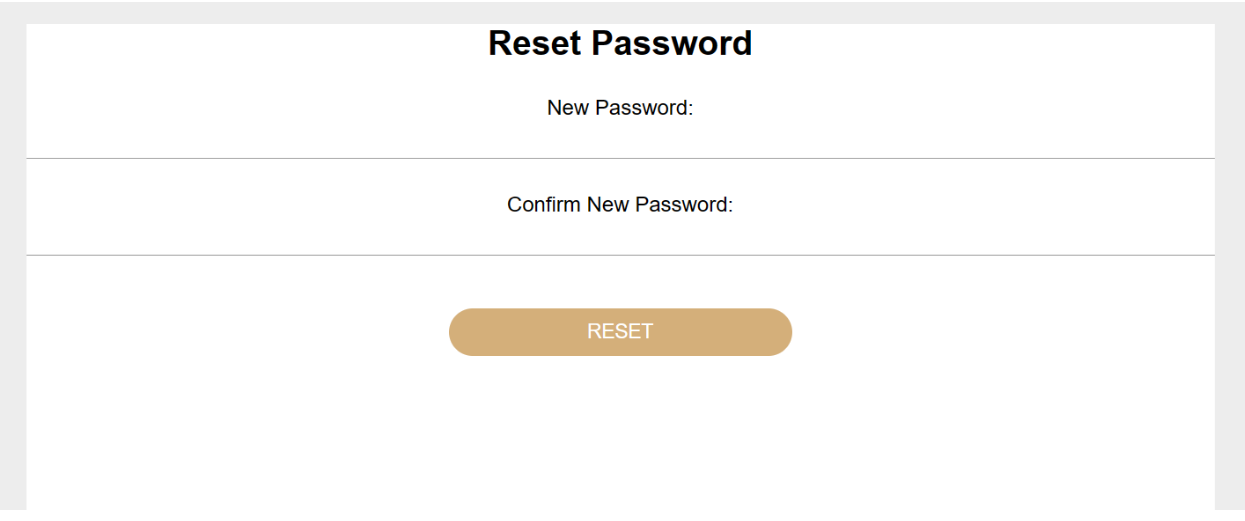
To make sure the user is not a robot, we set up pattern recognition. By the user correctly selected the image, we will pass the user.

If the client has signed up the app, he can turn to LOGIN page and enter his name and password.Then he can use graphic verification to ensure he is not a robot. Then he can send the validation code as the picture following:  


Then he can come to the main page.Same as sign in.

It should be noted that users do not have to forget the previous password and panic.After they click the “Forget Password” button,they can come to the password resetting page. They can reset the password, once the identity authentication is successful and the new password meets the requirements, they can return to the Login page to log in again.





Code explanation:

As in lecture 5,we choose to use SHA256 for hashing content. We use bcrypt.hashpw.to hash SHA256.This is where we create the SHA256 hash object, update the contents of the hash object, and get the encryption result.

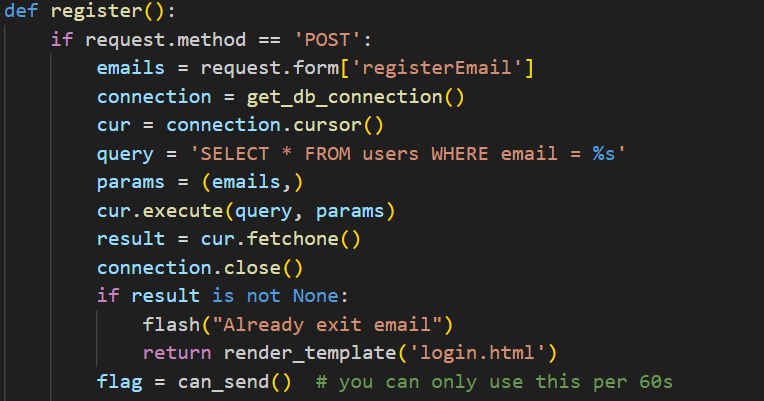
In order to avoid the evil result that the attacker can invade the computer after knowing that the verification code is valid for a long time, our verification code will refresh after 60 seconds.

A verification code of the specified length is then generated.

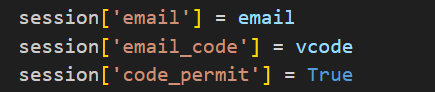
In the capTcha generation part I called my own interface, if you do not need to directly change the body to body = 'body'

In the process of downloading the CAPTcha image, I drew the string, added the interference line and saved the generated Captcha image.

For the issue of recovery key,the length must be larger than 24.



Then check if the email address and verification code is in the session, and then check if it is the POST method.



In order to ensure the uniqueness of email addresses, we have carried out the following Settings.

**E2EE chat**

To accomplish the online chat application based on E2EE, we will divide the task. Temporally, the task is divided into front-end and back-end parts: the back-end performs key generation, derivation, validation and updating, and the front-end securely transmits and stores the passwords and related parameters generated by the back-end. Spatially, the task is divided into client-side and server-side. The client side is responsible for handling user operations, feeding back chat data and storing some necessary encrypted parameters using LocalStorage, while the server side is responsible for acting as a platform and medium for user communication, scheduling messages from the client side as well as storing user account data and chat logs.

Firstly, we introduce the back-end process for generating keys. We utilize the ECDH key exchange protocol, which specifies the use of P-384 conic curves and generates key pairs (containing a public key and a private key) for each of the two users participating in the chat. Then, based on the key pairs of the two users, we still use the ECDH algorithm to generate a Shared Secret, which is generated from the private key of the message receiver and the corresponding public key of the sender, but due to the characteristics of the ECDH algorithm, even if the public and private keys of the two users are different, the generated Shared Secret is the same. The Shared Secret generated is the same. In our function ‘deriveSharedSecret’, we first use the variable ‘importedPublicKey’ to store the public key passed from the server, and then combine it with our own private key to get the Shared Secret.

Once the Shared Secret is obtained, we can derive more cryptographic parameters based on this value, which is common between the two communicating parties. The derivation method we have chosen is HKDF (HMAC-based Key Derivation Function). We derive two types of keys, the first one is the encryption key for encrypting the plaintext and the other one is the mac key for protecting the iv. IV (Initialization Vector) is a parameter in the encryption algorithm used to ensure the randomness of the process. With the protection of these two keys, the security of the transmitted content will be greatly improved.

We start by generating the encryption key; the encryption algorithm we use is AES-GCM (Advanced Encryption Standard - Galois/Counter Mode), and the hash method we use is SHA-256. We use the same configuration for generating the mac key, except that we use Hash for selecting the key material. HMAC method is used for selecting the key material.

After completing the derivation algorithm, we need to generate a pair of encryption key and mac key (denoted as keypair= {encryptionKey, macKey}) for each of the communicating parties. At the time of communication, each communicating party should obtain these codes independently and be able to select the correct key for decryption or message validation upon receiving the relevant json message. In addition, to protect the security, the communication process from user1 to user2 and from user2 to user1 should use different keypair.

The parameters used to generate the keypair are the same shared secret, salt, and a different info, which is a string containing the user ids of the two communicating parties to distinguish the direction of the communication. In the subsequent process, the function should choose to call different keypair according to the specific scenario.

After obtaining all the required encryption parameters, we enter the encryption and decryption process. At the beginning of the encryption function, we need to randomly generate a unique iv for the message and generate a corresponding iv counter starting from 0. The Iv counter will later act as a seed for generating new iv. For each new session initiated by the user, the value of the counter is increased by 1. This ensures that different sessions initiated by the user will have different iv values, which guarantees the security of the communication.

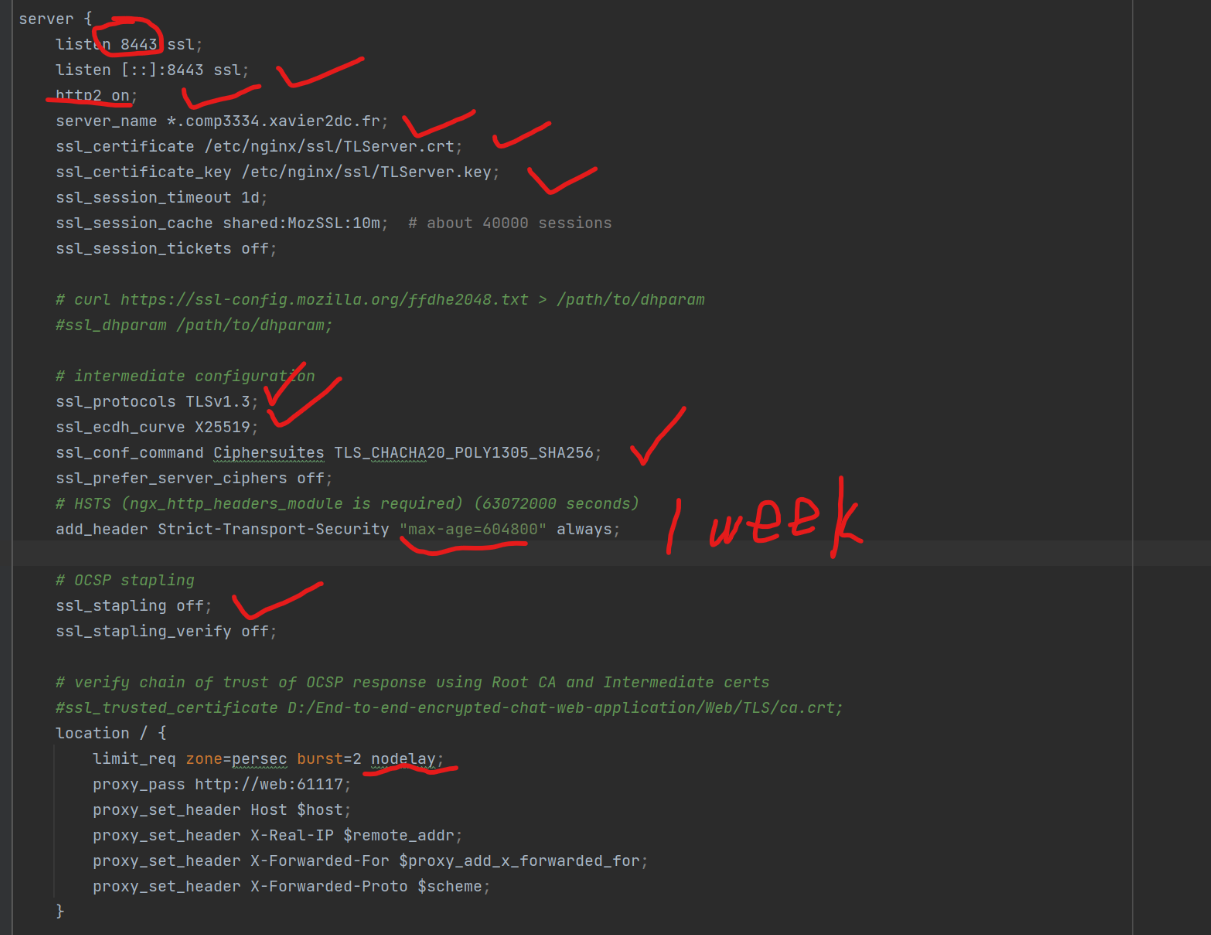
For Web Crypto API, the content processed in the encryption operation should be binary data instead of strings. So first we utilize TextEncoder to encode the message to be encrypted. After obtaining the byte sequence, we combine it with iv, additional data, and encryption Key to encrypt it into a ciphertext using the AES-GCM algorithm. In this case, additional data is similar to info in the key derivation phase, which is a string identifier used to show the two communicating parties and the direction of communication. After obtaining the ciphertext, we also need to encrypt the iv of the messages. The only things we need here are the mac key and the iv value, and the obtained encrypted iv is stored in the variable ‘ivMac’.

Once the encryption is complete, we will reverse the encryption process and decrypt it on the client of the message recipient. In the encryption function, we encrypt the message itself and the message's iv, using the encryption key and the mac key, respectively, and in the decryption function, we decrypt the message with the corresponding encryption key and verify the iv with the corresponding mac key. is passed along with the message to the decryption function (the receiver client), we need to verify the value of iv to prevent it from being tampered with and to rule out replay attacks before decrypting the message for security reasons. Therefore, we first verify the encoded iv by recalculating the mac value using the HMAC algorithm, mac key, and encoded ‘ivmac’.

After completing these pre-decryption verifications, we start decrypting the message. We choose the AES-GCM algorithm (same as for encryption), binaryized iv, additional data and the corresponding encryption key to get the decrypted plaintext. Of course, the plaintext is still a sequence of bytes, we call TextEncoder again at the end to return the plaintext as a string for the receiver to read.

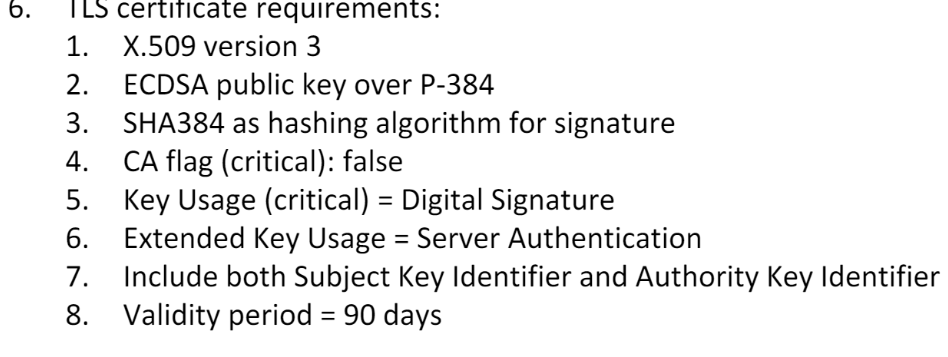
After finishing the implementation of the algorithms in the back-end, let's explain how the front-end part of these algorithms are combined into a complete communication process. First of all, after both parties have successfully logged in and selected each other as the communication object, the clients of both parties will automatically execute ‘generateECDHKeyPair’ to generate the public and private keys. Next, the public keys of both parties will be converted into json messages, passed to the server, and dispatched by the server to the corresponding communication object. After that, both servers generate a shared secret based on their private keys and each other's public key, which in turn generates an encryption key and a mac key. next, the client waits for the user to enter a valid text message on the web page. When the user enters a valid text message and clicks the send button on the page, the message is immediately encrypted on the user's client and the encrypted text is converted into a json message that is sent to the server. The server then sends the json message to the specified user and stores a backup on the server using SQL. When the ciphertext is sent to the recipient's client, the client first performs iv-authentication and eliminates replay attacks, and then finally decrypts the message. The keys used for decryption are stored in local storage. If the user refreshes the interface, the server will automatically forward the ciphertext of the history message to the client, and then the client will find the corresponding key to decrypt the message.

**TLS**

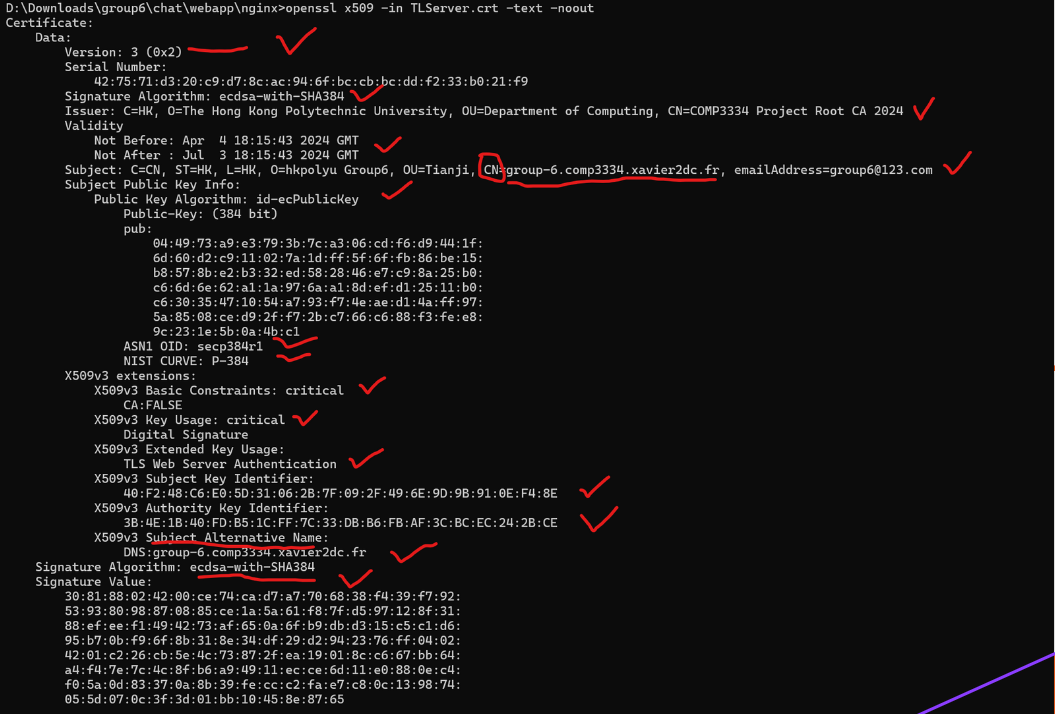


1. We can see from the intermediate configuration,the TLS version is 1.3.
2. ssl\_ecdh\_curve is x25519
3. The implement can be observed from ssl\_conf\_command.
4. From ssl\_stapling off and ssl\_stapling\_verify both of we can see that no OCSP stapling
5. “max-age=604800seconds is the total seconds for one week.

6.



We had implemented the required of 6.



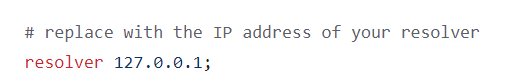
The top line shows the version is 3. The signature algorithm shows that it is ECDSA with SHA284. X509v3 constraint is critical and CA is false.The key usage is critical equals to data signature.The extended key usage equals to TLS web server authentication. Below we can see both the subject key identifier and authority key identifier. From Apr 4 to Jul 3,the validity period is 90 days.

7.



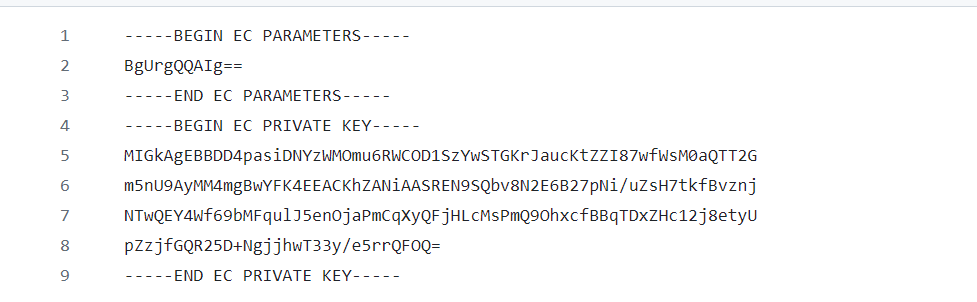
After enter the link , we can successfully,reach the home page of the app.

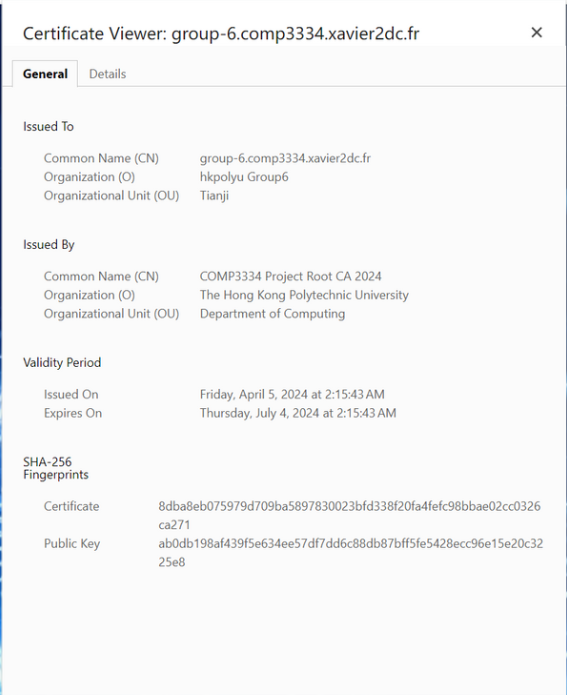
8.



The picture above shows above has fulilled the requirement.The host is ‘127.0.0.1’

1. We can see the domain name and the private name below.





1. We need to install the certificate on the computer and open the certificate.

