

User Guide Documentation

SecureChat is a chat system made for remote colleagues of the company SecureTech Solutions. It uses WebSockets and is secure, convenient, and efficient for messaging.

Overview

This guide provides step-by-step instructions on how to use our Websocket chat system, <https://chat-455.web.app/>.

We hosted our system using Google Cloud (back-end) and Firebase (front-end). Files for file sharing are hosted on Google Cloud. Additionally, we have an Uptime Robot that monitors our backend URL. The monitoring page is: <https://stats.uptimerobot.com/AQxwnAV9HS/800462842>. The robot is able to detect our server, but the server does not have any resources to respond to the ping.

Prerequisites

Before starting, ensure you have the following installed on your system:

- <https://chat-455.web.app/> (This is the URL to our chat.)

Features:

1. Chat and Status Indicator:

After logging in, you will see a list of registered users. Offline users are indicated with a black dot. Online users are indicated with a green dot. Choose the online user you want to chat with. Write your message in the text box and press the button to the right of it to send. The user's message should appear for the other user automatically.

The image displays two screenshots of a messaging application interface, illustrating a conversation between two users: user100 and user17.

Screenshot 1 (Top): Chat with user17

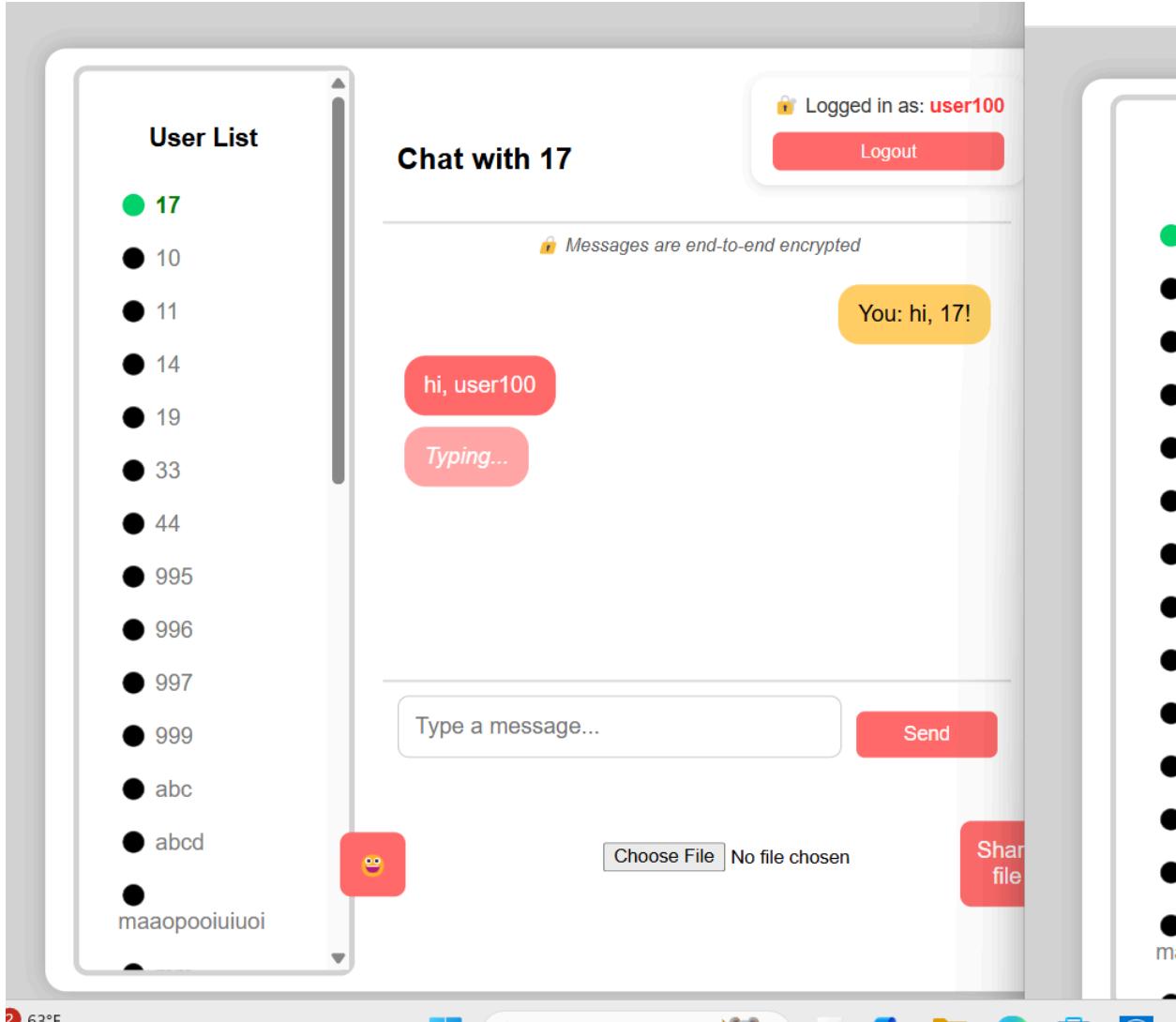
- User List:** Shows a list of users, with user17 highlighted in green.
- Chat with 17:** The title bar indicates "Chat with 17".
- Message History:** A message from user100 says "hi, user100". A response from user17 says "You: hi, 17!".
- Input Field:** A text input field with placeholder "Type a message..." and a "Send" button.
- File Options:** Buttons for "Choose File" (No file chosen), "Share file", and an emoji icon.
- User Info:** Top right shows "Logged in as: user100" and a "Logout" button.

Screenshot 2 (Bottom): Chat with user100

- User List:** Shows a list of users, with user100 highlighted in green.
- Chat with user100:** The title bar indicates "Chat with user100".
- Message History:** A message from user17 says "hi, 17!". A response from user100 says "You: hi, user100".
- Input Field:** A text input field with placeholder "Type a message..." and a "Send" button.
- File Options:** Buttons for "Choose File" (No file chosen), "Share file", and an emoji icon.
- User Info:** Top right shows "Logged in as: 17" and a "Logout" button.

The conversation between the users, user100 and 17.

Additionally, a “typing...” indicator appears when the other user is typing.

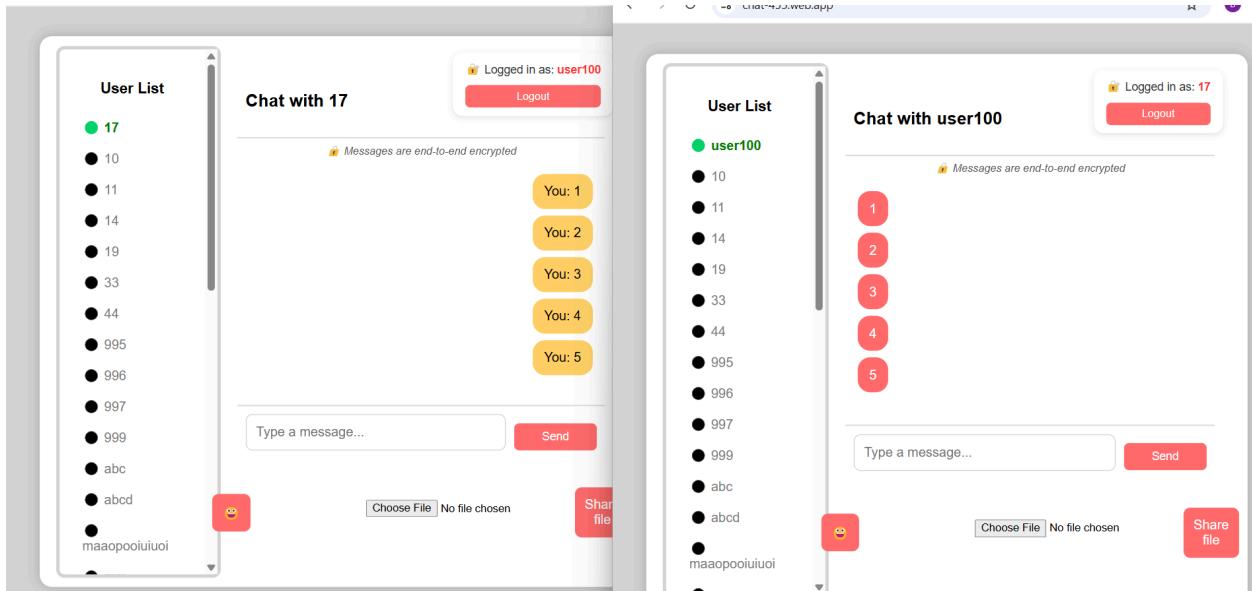


2. Test rate limiting:

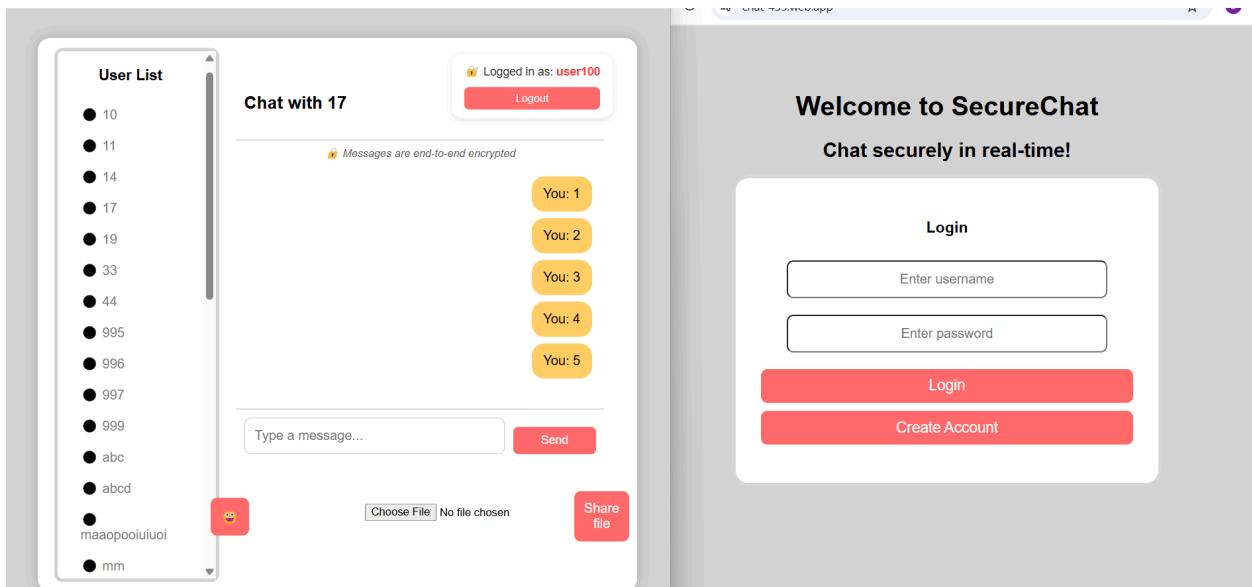
Users are only allowed to send five messages within a twenty-second period. Sending more messages within a twenty-second interval will result in your message not going through to the other user. Look at the other user's window to see if the “spamming” messages came through. Because of rate limiting, they are not expected to reach the other users.

3. Disconnection detection:

Press the logout button for one of the users. Go to the other user that is still connected. You will see on the screen of the still-logged in user that the dot near the disconnected user is not black, meaning that the user is offline and disconnected.



User100 and 17 are connected to each other. 17's status indicator on User100's side shows he is online. User100's status indicator on 17's side shows that User100 is online too.



17 logs out and on User100's side, 17 is automatically marked as disconnected with the black dot.

4. Automatic reconnection after interruption:

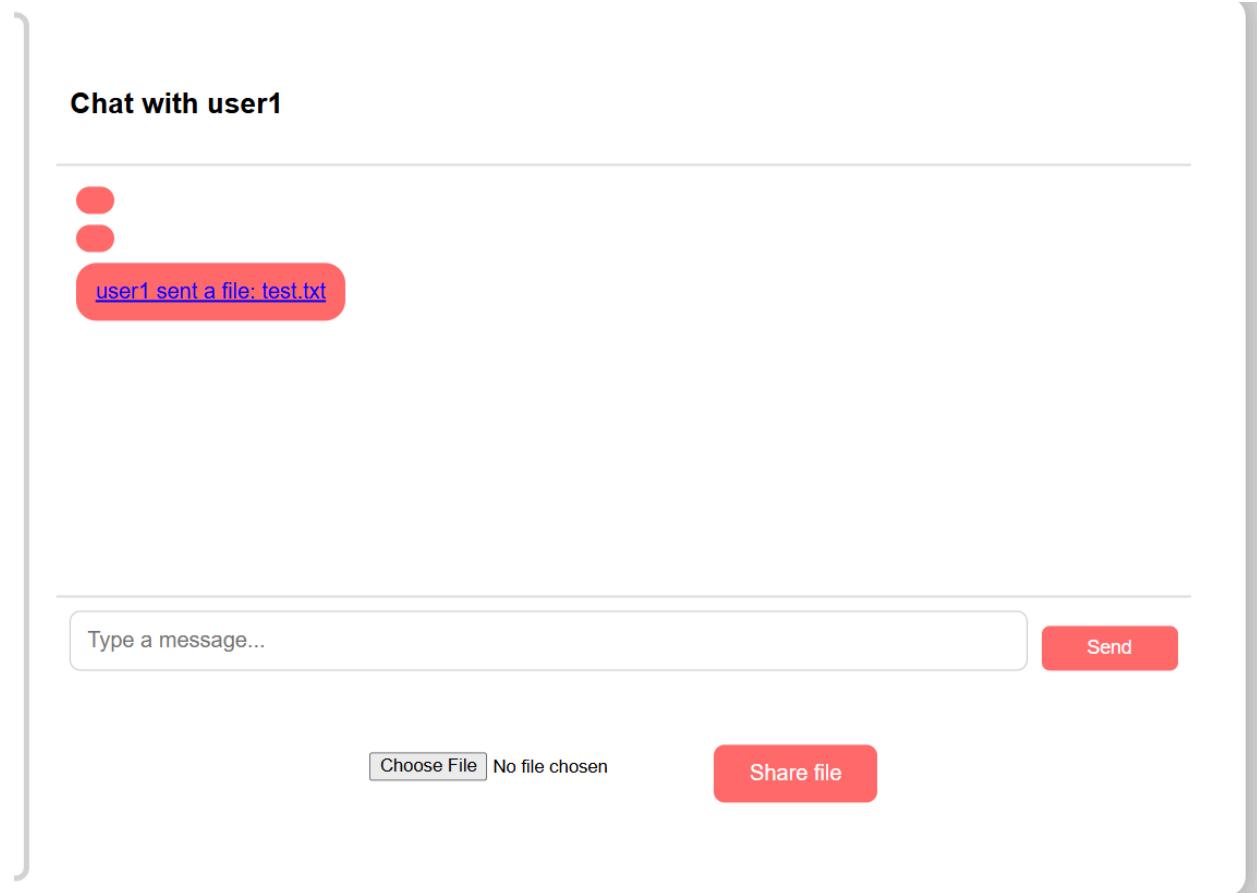
Log one user out to simulate interruption. Then log the user back in. The messages that were exchanged before the interruption occurred automatically appear. You can then send messages by clicking on the user you were chatting with. The other user does not have to do anything to re-connect.

5. File Sharing Functionality:

Our WebSockets system allows file sharing with encryption. After logging in, press “Choose file”. You will then select a file from your computer. Then, press share file. The file will appear on the sender’s and receiver’s chat box with a downloadable link.

The screenshot shows a messaging interface titled "Chat with user2". On the left, there are two red circular profile pictures. On the right, there is a yellow speech bubble containing the text "You sent a file: test.txt". At the bottom, there is a message input field with the placeholder "Type a message..." and a red "Send" button. Below the input field, there are two buttons: "Choose File" with the text "test.txt" next to it, and a red "Share file" button.

The sender sends a txt file. It appears in the sender's chat box.



The receiver receives the downloadable file link.

The above options only work for small files. For larger files, we have hosted five files in Google Cloud that you can use as a public file link. They are:

- Mars: <https://storage.googleapis.com/securechat-filecloud-hosting/mars.jpg>
- Meme: <https://storage.googleapis.com/securechat-filecloud-hosting/meme.jpg>
- Mountains: <https://storage.googleapis.com/securechat-filecloud-hosting/moutains.jpg>
- Squid: <https://storage.googleapis.com/securechat-filecloud-hosting/squid.jpg>
- Statue of Liberty: <https://storage.googleapis.com/securechat-filecloud-hosting/statue.jpg>

All you need to do is copy and paste these file links into the message input field and send.

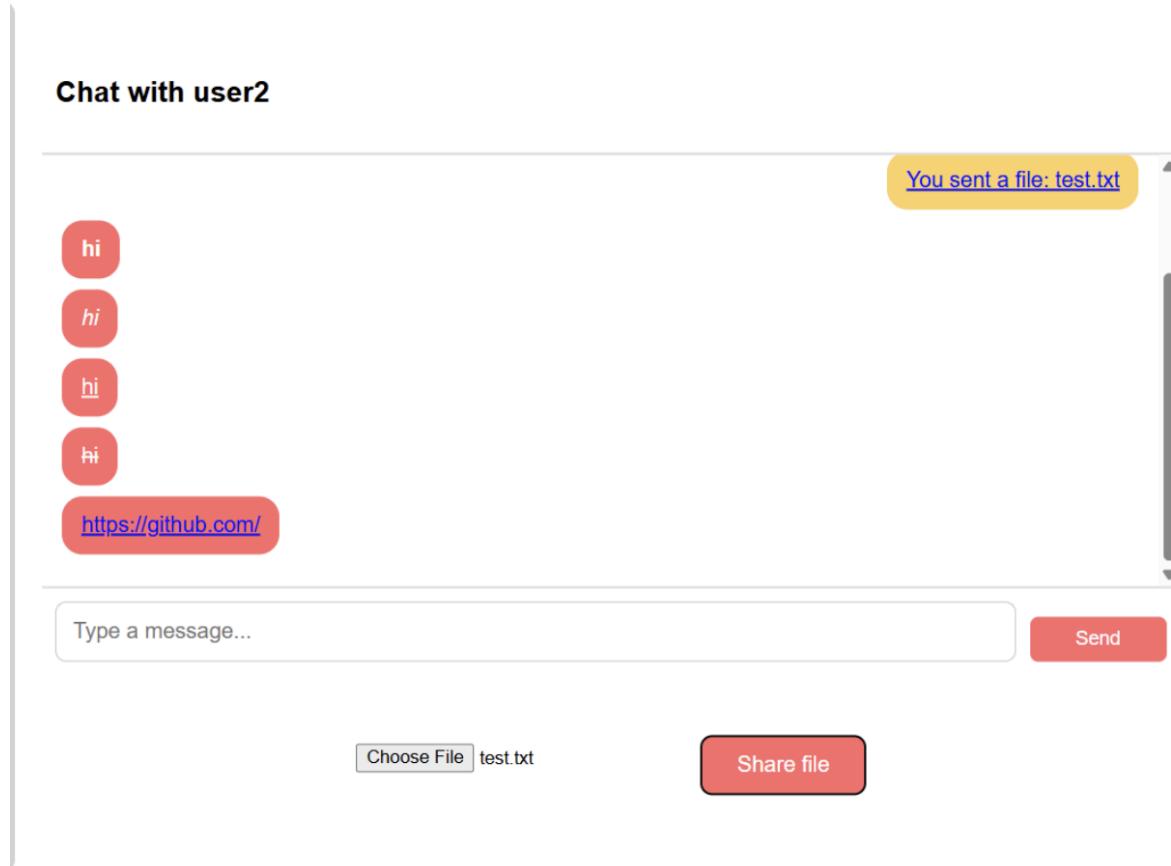
Because we want to block malicious files, we have limited file sharing to files that are maximum 15 MB. The files cannot be script or exe files.

6. Basic Text Formatting

- To send bold text, press Ctrl + b before typing.
- To send italicized text, press Ctrl + i before typing.
- To send underlined text, press Ctrl + u before typing.
- To send strikethrough text, press Ctrl + x before typing.

Brian Wei and Sagarkumar Patel

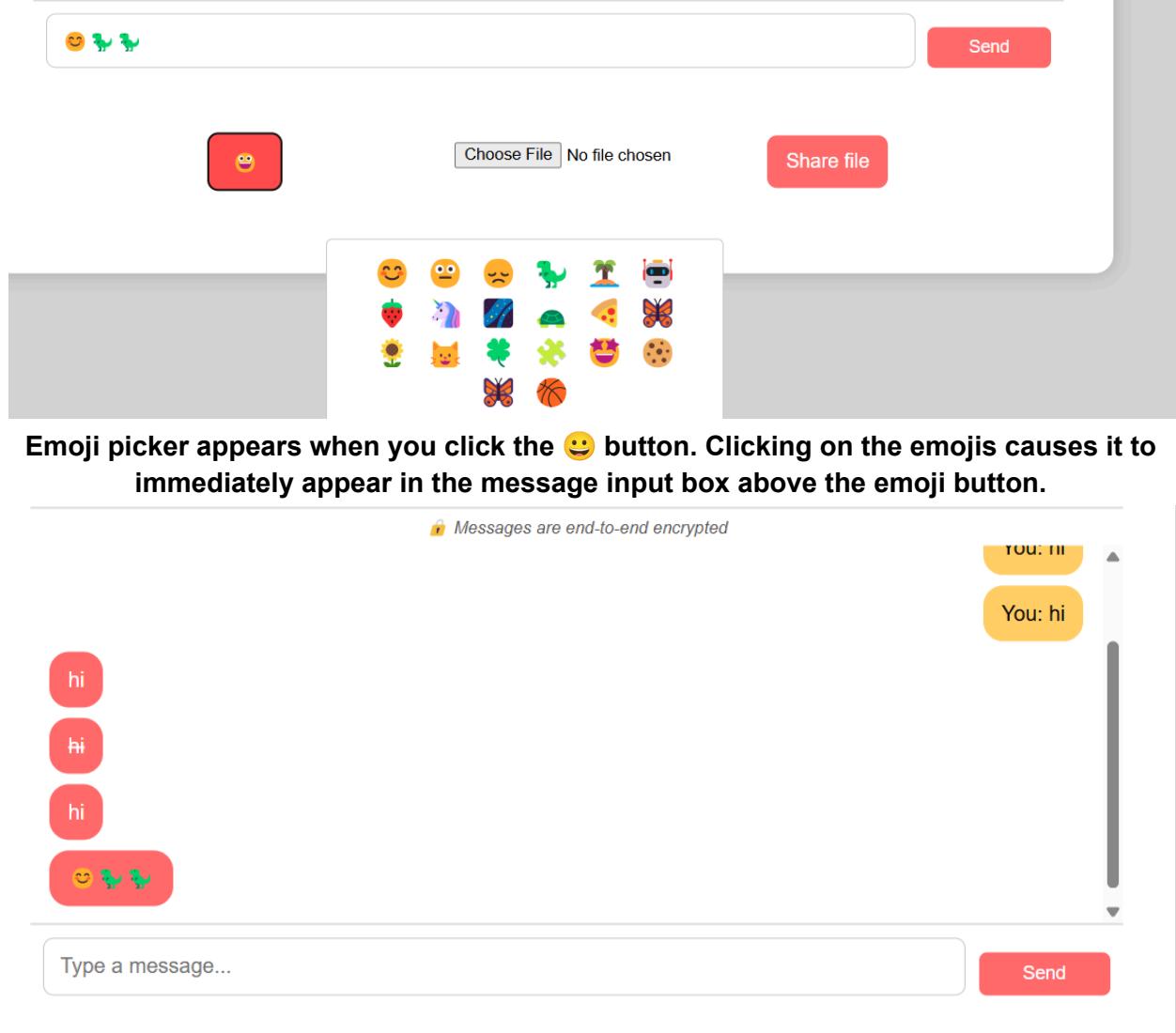
- Sending links will automatically appear as a clickable link on both sides.



**The receiver successfully receives the bold, italicized, underlined, and strikethrough text.
The clickable URL link is also received successfully.**

7. Unicode Emoji Picker

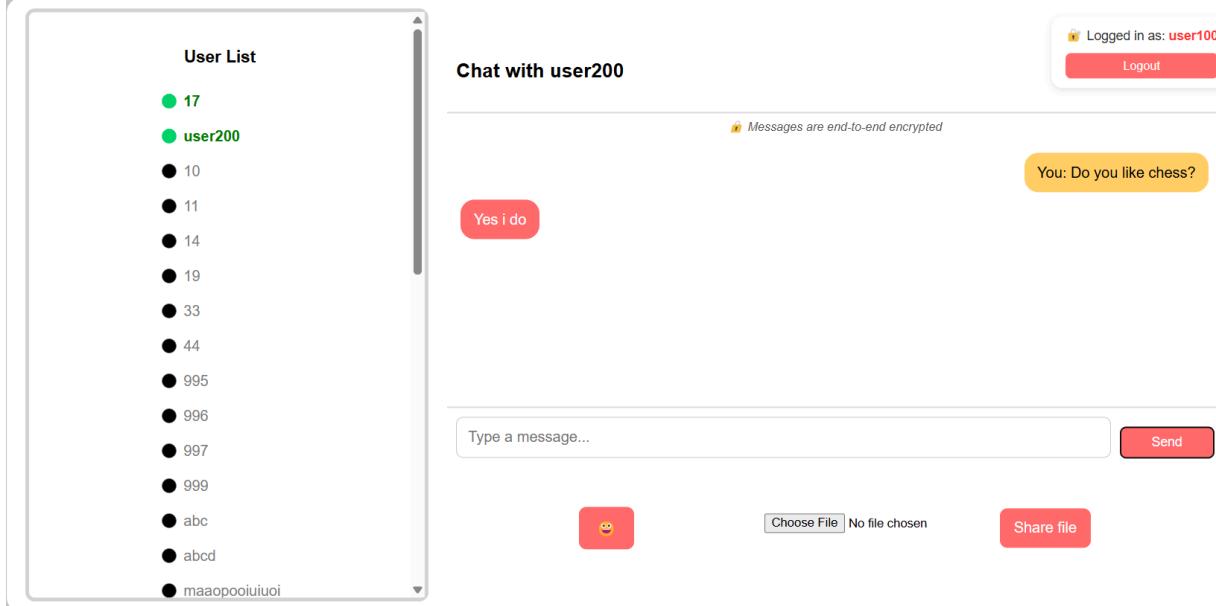
The chat system gives you an option to choose emojis to send. Press the 😊 button. You will see a selection of emojis you can choose from. Clicking on the chosen emoji automatically brings it to the message input field. Press send to send the emoji.



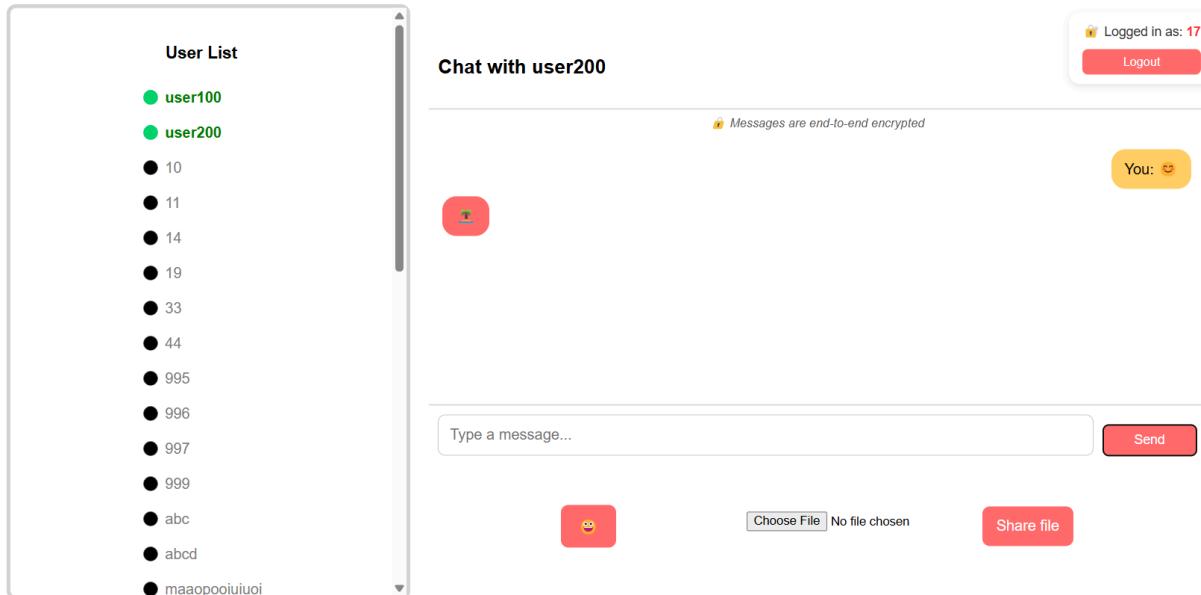
After pressing “Send”, the receiver successfully receives the emojis.

8. Multiple Users

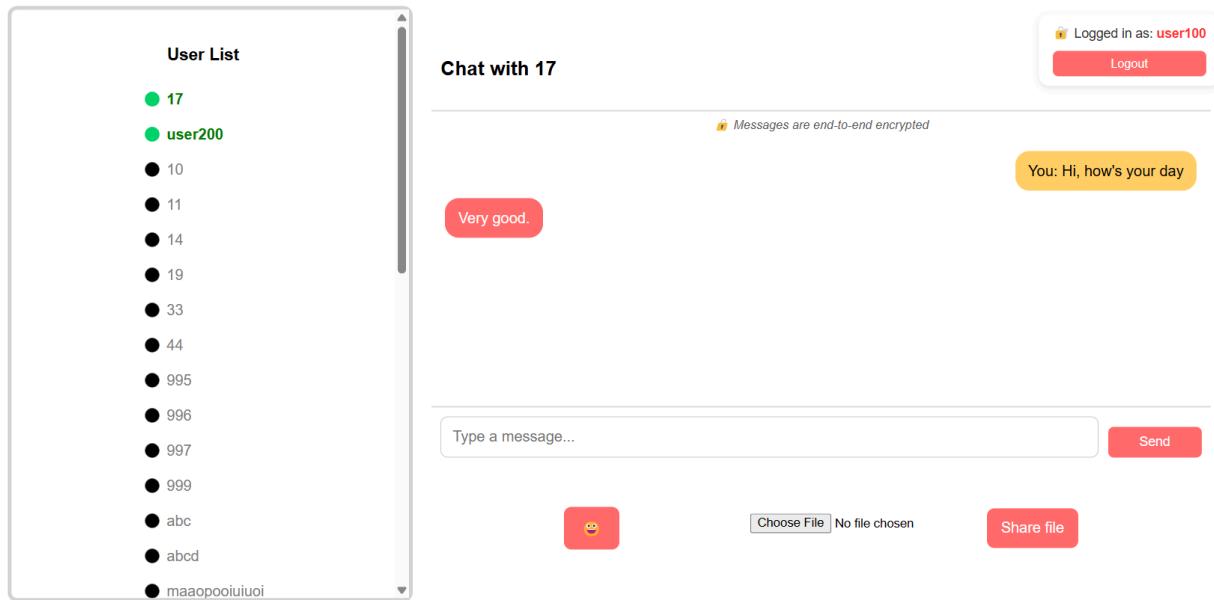
Our chat application includes the ability to have more than two users. You can select the user you want to talk to on the left panel.



The text messages between user200 and user100. The left panel allows you to easily switch between different users.



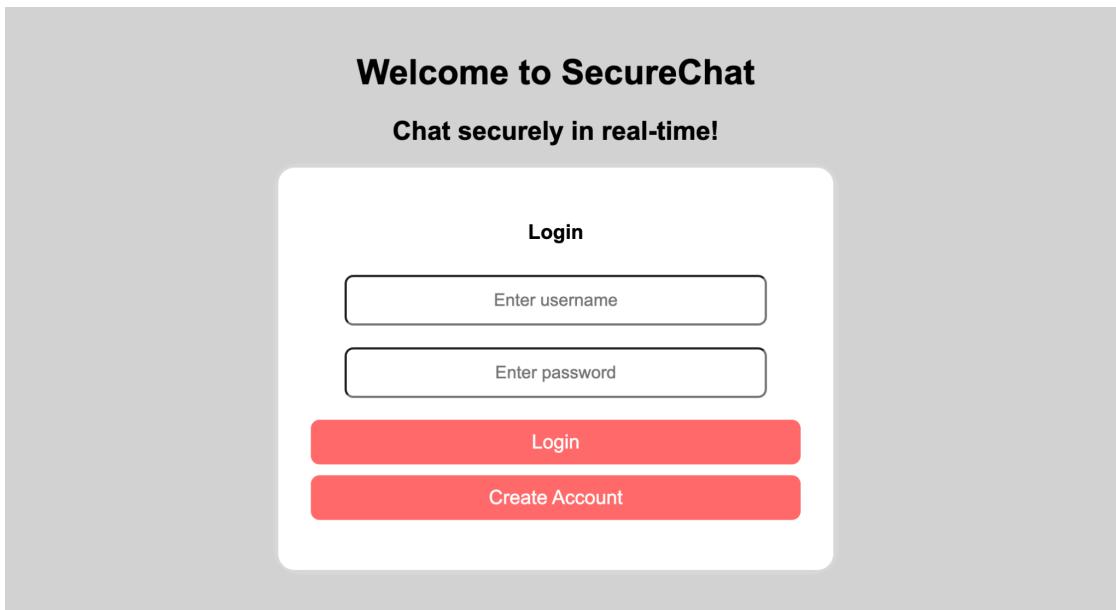
Text messages between user200 and 17.



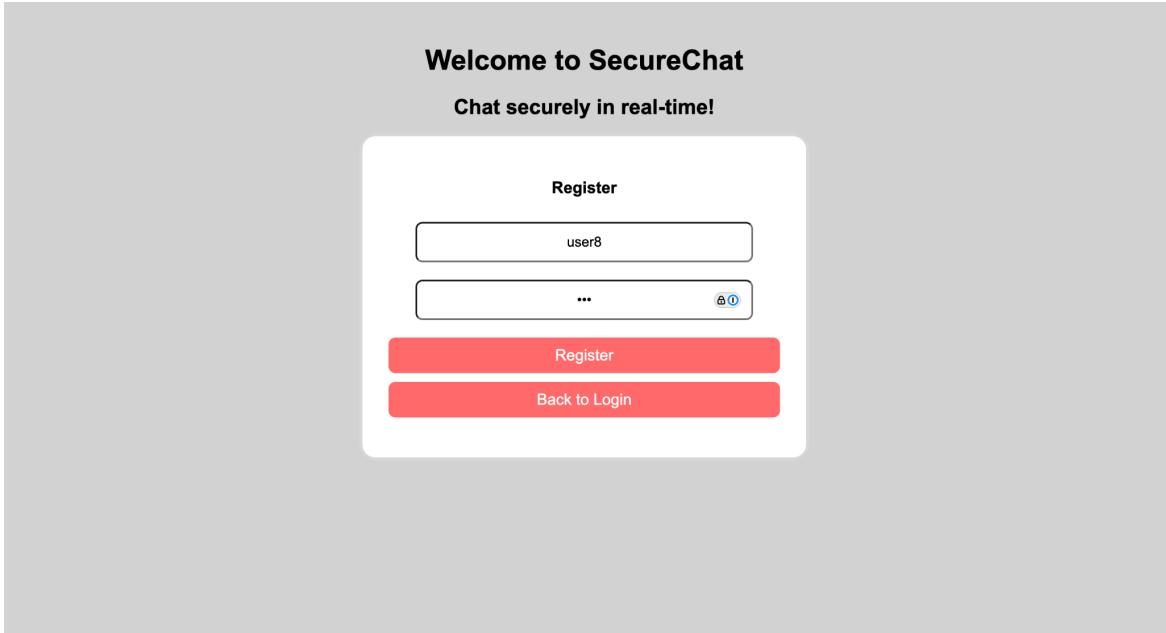
Text messages between 17 and user100.

9. User Registration/Login

Our new update includes a feature that allows user to register their accounts alongside login with us by using our website.



Users can simply go to create an account.



Register their account, ex:user8

A screenshot of the 'SecureChat' user interface after login. On the left is a 'User List' sidebar showing 17 users (17, user200, 10, 11, 14, 19, 33, 44, 995, 996, 997, 999, abc, abcd, maaopooiuuioui) with green dots indicating they are online. The main area shows a 'Chat with 17' window. At the top right, it says 'Logged in as: user100' and has a 'Logout' button. A message 'Very good.' is shown in a red bubble. Below the message is a text input field with placeholder 'Type a message...' and a 'Send' button. At the bottom are file sharing options: a smiley face icon, a 'Choose File' button with 'No file chosen', and a 'Share file' button.

Users will see this screen after login is successful, which also shows all registered users on the left-hand side.

The system also has the ability to secure all passwords. Our website hashes all passwords and stores them as hashes. When users log in, the website compares the hashed passwords on the server side to ensure a seamless and secure login experience.

```

user8 $2b$12$qSdeyviZuwm9jwZ0iIxAiu2hC6VvP/YZRHJ4AV9NXku/qLJpnFO
user6 $2b$12$4fUgBmFw2/kkr7ddSrX98uom/5Q1Q4mRGREZimn6ETeBBdNeyCNUa
abc $2b$12$VHwT8WjhyDspulgEjrmUNefr9/jj5VnhcGmKdeeX3pAFLyjekyya2
abcd $2b$12$RtsEZaZYxAhP.48JEl/P2uAyxPwlPutLC3LsEWiSzD2aKfjN4l66S
user7 $2b$12$9mT8H0PJgrE.g.xFgY0Kg.7PAUhQGtrJAwLa9mx6aDNqGsVxZ8WEC
user9 $2b$12$8daFWbrrMMeUP.I67n5.YOYhrbQ1CKtGBNqsFm4DLiQia5xyZ0yzi
user10 $2b$12$vFLpqhmb.zlIqCbXDT2kYuuWfIKd/00J4fpSTY881Ql8kxBxp/E.C
user11 $2b$12$kE3t10ad80WStOBkfmbgLufLH7kDCq0VK31/CbHv7l0GV3DJsf8fq
user50 $2b$12$gz85Sng8cA5aG18jrkEwauTB/CUcNWs6Hst2708c0UKLLM1h.DYGm
user51 $2b$12$03xix5RW.dxHaqxh0Qs5euL/i79.vXKpDql4a.EaQdQck1eY77uEG
user54 $2b$12$3pnYUVAr86t4iE44B9C5..EWFEsoasn4qV8N9/g4QpiiwX0oPqVMbw
user57 $2b$12$iPMf271FN/09cWS18JKyVeJxH0cpv7XVMWjrA6Tmuy0ue0S/2/Ju6
user60 $2b$12$ZGhdMwi/69vPrkvMyo.r407eID5vUnRjpjYgXAQ6cHVx1f0XbU/8m
17 $2b$12$98fssEtdWCmgbQZHxusZC0JJrYu2aoqTVtFP8kIJ7Dj9oLvQF73fq
14 $2b$12$s16IIREFpbro1rAmtbZfMe7bkcmk0m3i.g1q7mXgHhQ0/kU33/z50
19 $2b$12$R3a7AKwDlW8SyW0pAZ/FSe0WaK5Cew1UxIbxZuwbsmFTI8frZFWa
10 $2b$12$nCyu4bU80IZ080IGm8tzMOR/zIB7L6bs1QSkN7.S425i5FTc4mQhm
11 $2b$12$IkWPJr2AxVLihCnQjQWCL.Tp9pSM4210z/lcHMHNi57teU2j02Vda
33 $2b$12$k0G8P5XNcdEyc2QfoI.xeQkJp3RgKaC5HzyFQnVNS0RdNraw.ht6
44 $2b$12$Uca8HoDxjjW65.zuNxAfP0RRTfWNn6YhUajuE.cC/ya6Mhu6kngIa
mm $2b$12$dppVWkwAmeLqjjY/mClkuxIrE7nwAb4xtStwDiJ23dtiUurlslnW
maaopooiuuo $2b$12$n1LcURQL1Ym9AzCeYY8HJuuhj3AXiejGvKJqvcCPVrttsF5V4jhfwW
999 $2b$12$rUtgBj.zaZ.QpTwIzt55S.KFs4niwo78ulZC1eZD/2xaAvvfzCwJG
997 $2b$12$KXPfk7YHcKPavPLEejXJ1.4H05VMGcJYvQlI2Q4LL1UhxxvZdBNZW
996 $2b$12$76sGS4U08r6/2w9zPy1dputjME5uCkKRwZo4PkTT1dL1FWSnEtKO
995 $2b$12$bJ1bjb6cp0VC0F1CrbBxNuu7EzNGzXKcb0jizKets4WZpHx7xh0eW
user100 $2b$12$uYIU72zuIk2XX5PkvecFYQeRJLhKLZB0hoTZCNfkY6znFB2urfQy
user/ $2b$12$/hJEg9Tun/Da8K34PYL3t0JE2PNn4Cqjp2isb9hA50w6qKagb7rdu
user< $2b$12$v07kbuH3/3F4g0CmEPEbWeTpapT2MFsemVNhyrY1tiaWwrPY1UaMm
user.. $2b$12$89EXK8SNraw.STKbvvP.ZerYJFUboVHB3qzd1W/9sT6K15CwrTtLW
user56 $2b$12$J1mkH83gcCd7DSFDRI2xg.V2rfgE411ydTxvEvVApieAclkIODlq6
user101 $2b$12$p0xyUvmnHz15mzxa7KwSG.Yh/RbfFhkk10p6hxP9pKXNjyo1HOBmG
user102 $2b$12$iRyiRFD3t/zUQHvwUqlh4OurwgJ1k9z00cZ8X902MBbwUVUC9mPPC
user34 $2b$12$eB72ytjj3cVzTNBasE5tLeBQ5/681f48ArFG3kMfuw5Ldhw6BJYw
user1 $2b$12$r2iCzbJLqt9ySuHFQtRB200zLY9NidSq186nZ8raWuaSNy5YAAkIW
user200 $2b$12$PPXfh067M1dCSN70sw6MVuy6V8As7FPRsdSnsLVM6TzMNOQy8Mo60
user1000 $2b$12$T0t6um0X9h0jYxKIsmyNOSCc3sG79Q0ReP0iIYvjo6bqRnHzdQTq
user1001 $2b$12$BhLVPLhTq4eTY1FeW4pGWOZPd1W6w84JQKxKA3ZoNqyP01i2ZSBk.

```

All user passwords are hashed and stored. The usernames are on the left. Hashed passwords are on the right.

This file is stored in Google Cloud.

10. Chat logs

Our application has the ability to log chats into a file. The logged messages are listed in order that they were sent. The timestamps are provided and the encrypted messages are listed as well. For each message, you will see the sender and receiver.

```
[2025-05-03 19:15:38] 17 -> @user100 7d18ZHp+qhgLXjhAUwpoXw==  
[2025-05-03 19:20:19] user100 -> @17 7d18ZHp+qhgLXjhAUwpoXw==  
[2025-05-03 19:20:19] user100 -> @17 7d18ZHp+qhgLXjhAUwpoXw==  
[2025-05-03 19:20:19] user100 -> @17 w0tYJ69keG1X2R1G1r7Y0w==  
[2025-05-03 19:20:19] user100 -> @17 7d18ZHp+qhgLXjhAUwpoXw==  
[2025-05-03 19:20:43] 17 -> @user100 7d18ZHp+qhgLXjhAUwpoXw==  
[2025-05-03 19:26:07] user100 -> @17 TFR2KuWb5pjIjRv9/s6WAg==
```

Chat log (from Google Cloud) shown between user100 and 17. The timestamps and the encrypted message (for security) are shown.

11. Encryption

Our end-to-end encryption uses AES and a key for encryption and decryption. The client encrypts the message with the key. When the message is on the server, it is still encrypted. Once it reaches the other client, the other client decrypts it to plaintext. Our key is kept secret so we made sure that it is not hard coded into the code.

```
[2025-05-03 19:50:58] user1001 -> @user1000 7d18ZHp+qhgLXJhAUwpoXw==  
[2025-05-03 19:51:00] user1000 -> @user1001 pImVSloYavpIzCMfENMfzCzjJdnfdXw0W3yamIju40eMgJKocvDvggdbEZ  
R2qFvyfLCNDMbtzZs+tC9XF63+JPwdoi+jLcJqtY
```

The logging functionality occurs in the server, so it shows that the messages in the server are encrypted. The long encrypted message is a file, which shows that files are encrypted as well.

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. Under the 'WebSockets history' tab, there is a list of captured messages. The first message is highlighted:

#	URL	Direction	Edited	Length	Notes	TLS
14	https://chat-server-965301149573.us-central1.run.app/	→ To server		24		✓
15	https://chat-server-965301149573.us-central1.run.app/	← To client		34		✓
16	https://chat-server-965301149573.us-central1.run.app/	→ To server		24		✓
17	https://chat-server-965301149573.us-central1.run.app/	← To client		34		✓
18	https://chat-server-965301149573.us-central1.run.app/	→ To server		15		✓
19	https://chat-server-965301149573.us-central1.run.app/	→ To server		15		✓
20	https://chat-server-965301149573.us-central1.run.app/	← To client		15		✓
21	https://chat-server-965301149573.us-central1.run.app/	← To client		15		✓
22	https://chat-server-965301149573.us-central1.run.app/	→ To server		15		✓
23	https://chat-server-965301149573.us-central1.run.app/	→ To server		15		✓
24	https://chat-server-965301149573.us-central1.run.app/	← To client		15		✓
25	https://chat-server-965301149573.us-central1.run.app/	← To client		15		✓
26	https://chat-server-965301149573.us-central1.run.app/	→ To server		15		✓

Message

Pretty Raw Hex

```
1 user1001: F4XgPEMFfeTwBBnRBpYrUlw==
```

Burp Suite captures the message, and you can see that it is encrypted.

12. Input Sanitization

Brian Wei and Sagarkumar Patel

The screenshot shows the Burp Suite interface. At the top, there's a table of network traffic with columns for number, URL, status, and count. Below the table is a 'Message' section with tabs for Pretty, Raw (selected), and Hex. A message is typed into the text area: '1 user1000'. On the right side of the interface, there's a large, mostly empty workspace.

79	https://chat-server-965301149573.us-central1.run.app/	→ To server	8
80	https://chat-server-965301149573.us-central1.run.app/	→ To server	1
81	https://chat-server-965301149573.us-central1.run.app/	← To client	233
82	https://chat-server-965301149573.us-central1.run.app/	← To client	50

Message

Pretty Raw Hex

1 user1000

The screenshot shows the 'Send WebSocket Message' dialog. It has a 'Send' button and a dropdown menu set to 'To server'. Below the dropdown is another set of tabs: Pretty, Raw (selected), and Hex. A message is typed into the text area: '<script>alert('XSS')</script>'. The rest of the interface is a large, empty workspace.

Send WebSocket Message

Send To server

Pretty Raw Hex

1 <script>alert('XSS')</script>

In Burp Suite, I modify the username input in the login page and send it. I found no issues.

Send WebSocket Message

Send To client ▾

Pretty Raw Hex

```
1 user1000: Ofv4zvqw0cES0QrddcYoVg==
```


Send WebSocket Message

Send To client ▾

Pretty Raw Hex

```
1 user1000: <script>alert('XSS')</script>
```

Logged in as: **user1001**

Logout

Chat with user1000

(Messages are end-to-end encrypted)

hihihihihihi

You: hihihhiiiiii

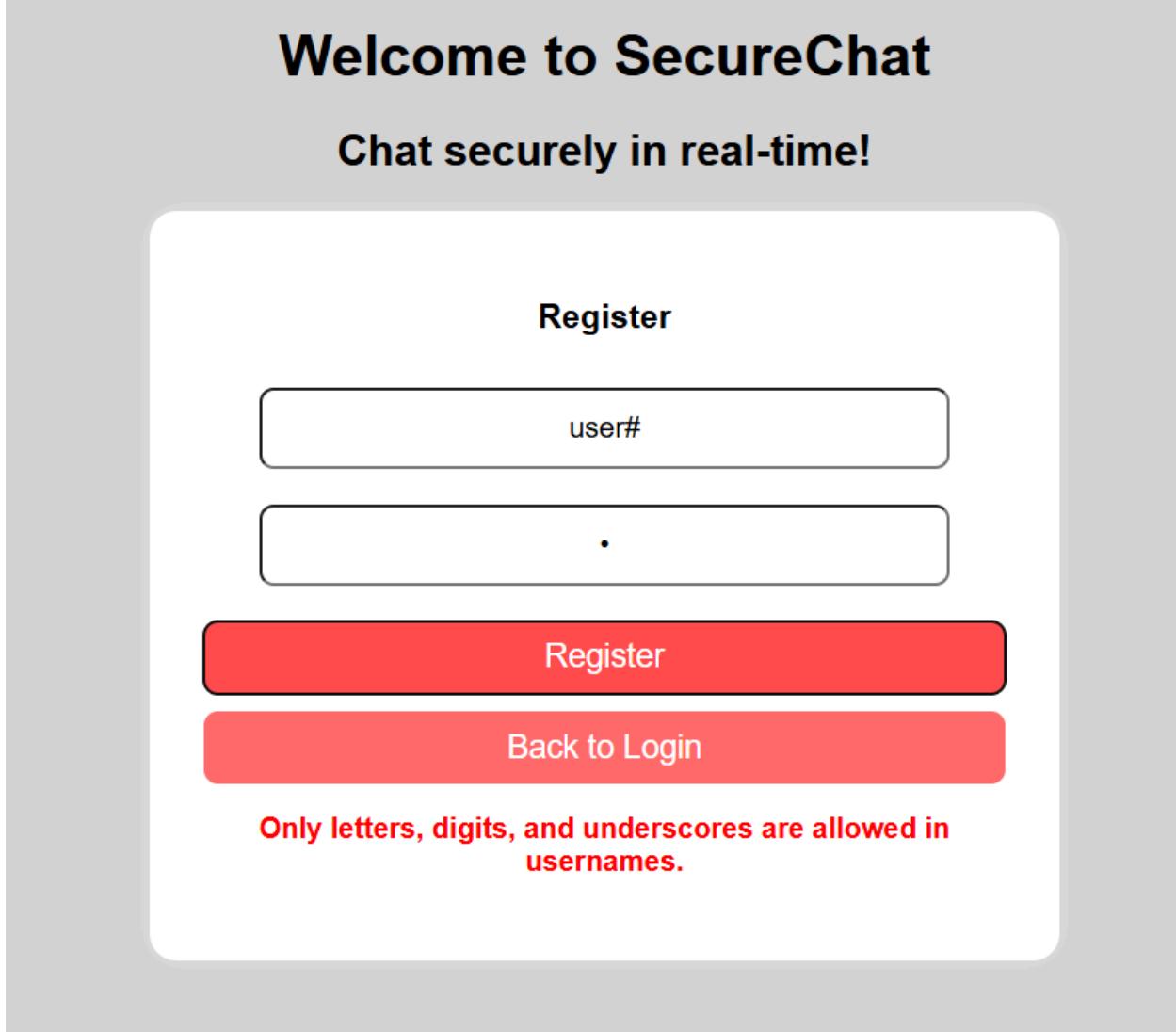
You:

Type a message...

Send

In Burp Suite, I captured a sent message and modified it in the repeater and sent it. Due to sanitization, it doesn't harm our system.

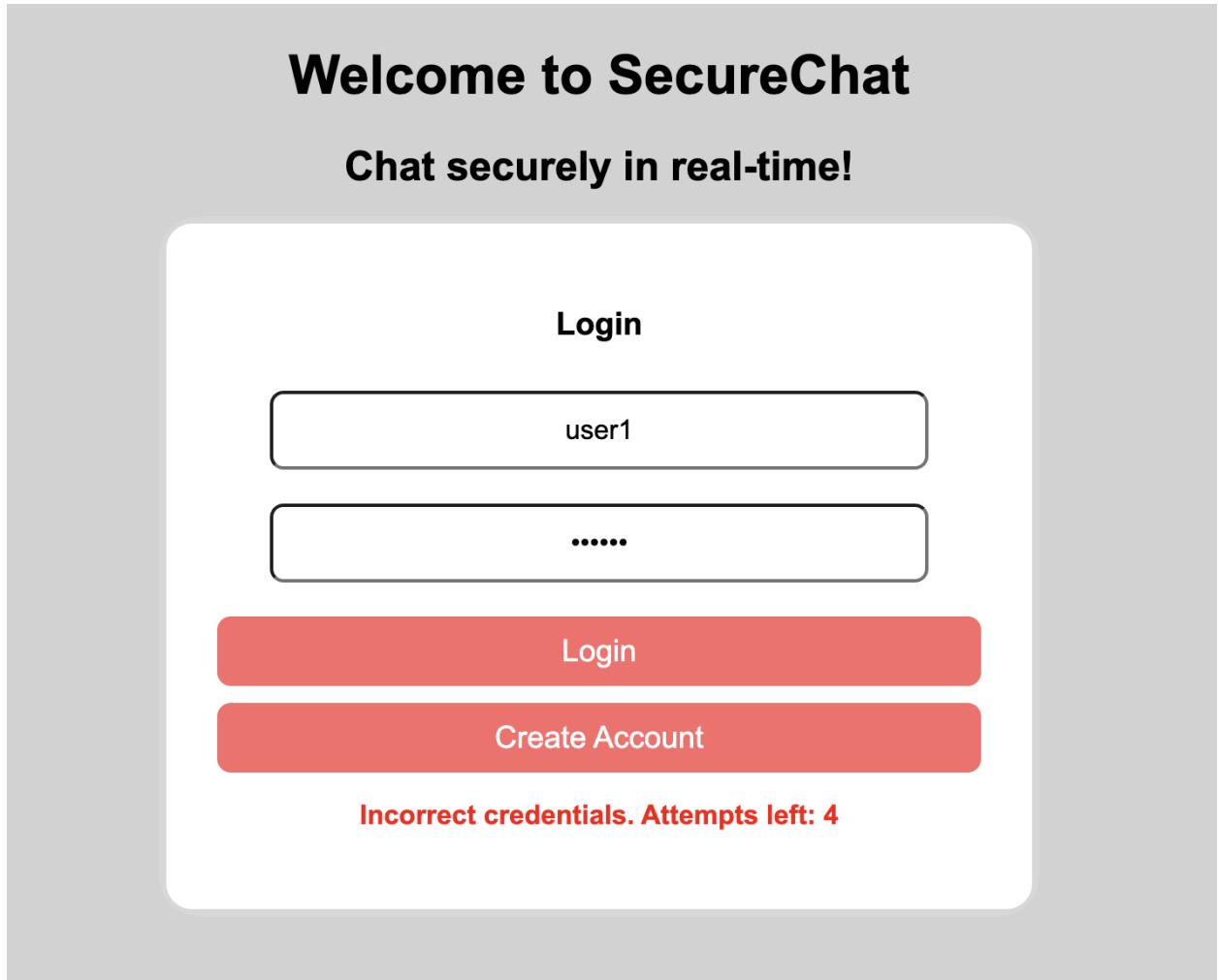
A sanitization policy we have is that usernames can only contain letters, numbers, and underscores.



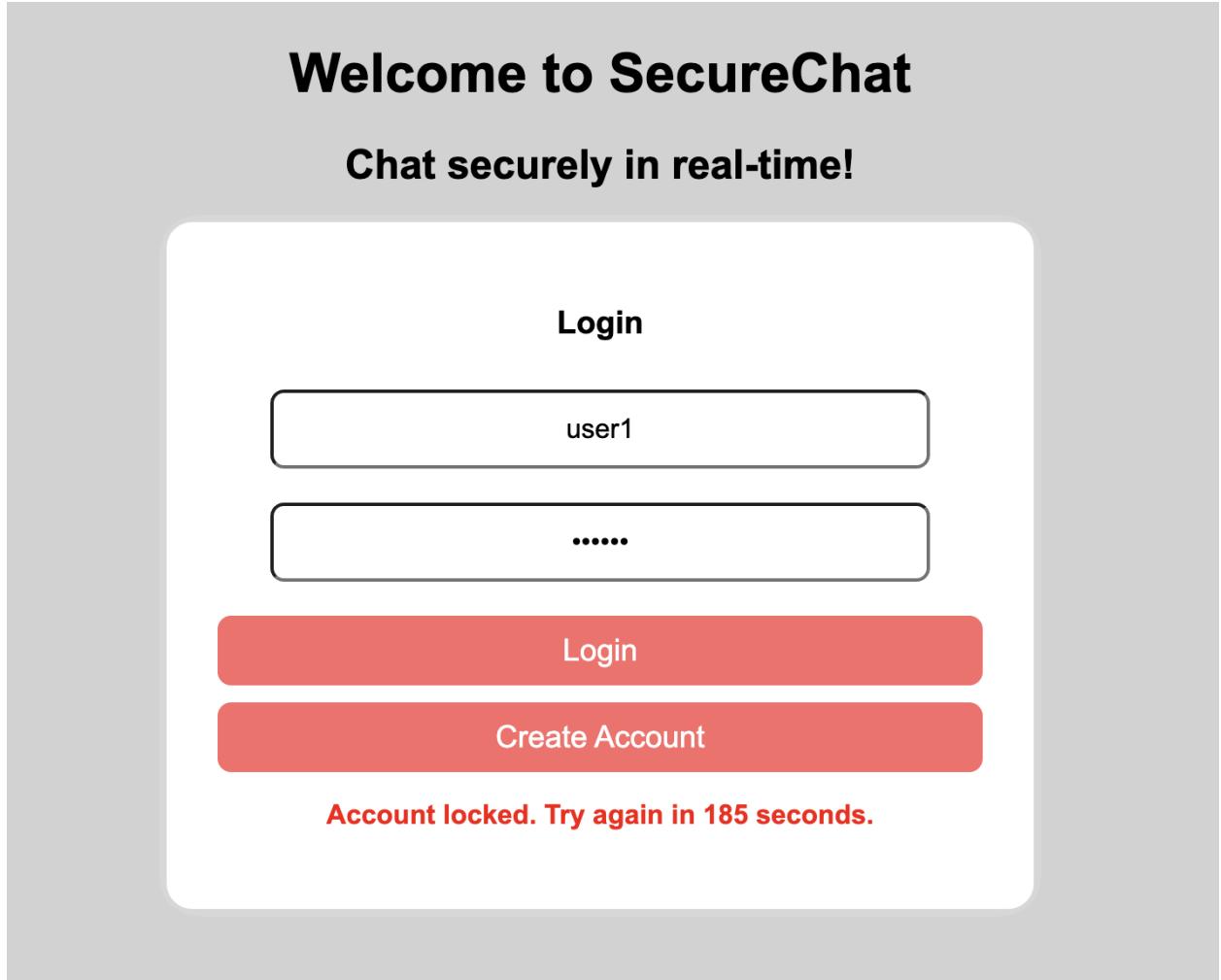
To prevent any kind of injection or XSS, this policy is being enforced. The username in the image cannot be registered or used in the login page.

13. Log-in/Registration attempts limitation.

We implemented a feature to prevent brute-force login attempts. If a user attempts multiple login attempts with the wrong password, a timer counts down after each failed attempt. After a certain number of failed attempts, the user will receive a timeout.

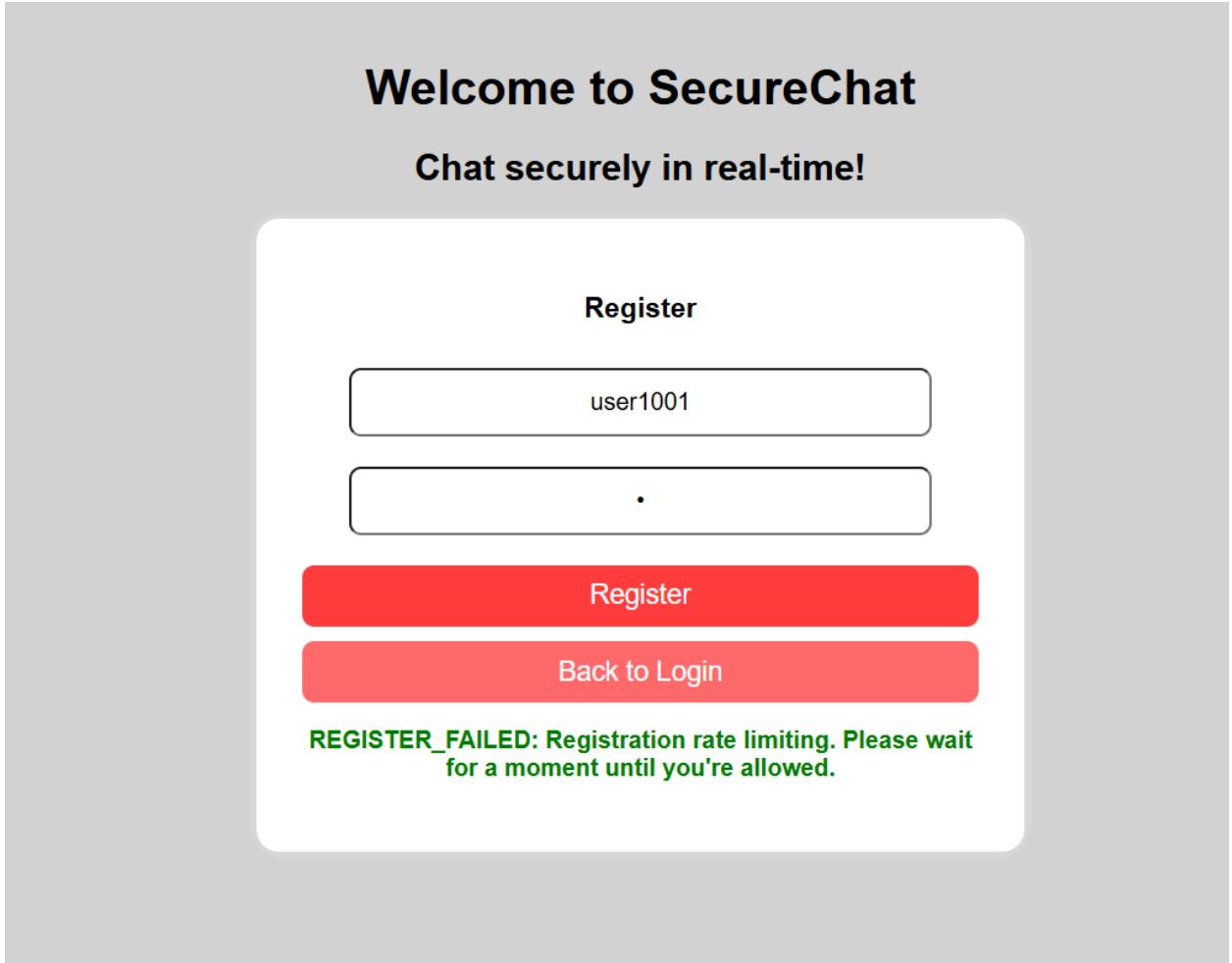


The user gets 5 attempts to log in.



After five failed attempts, the system will put a 5-minute timer on the user's account.

Our registration limit policy usually limits users to register only one account within a 15 second time period. This is meant to ban bot registration attacks.



User1001 cannot be registered in the above example because of our registration rate limiting policy.

14. Cloud infrastructure for front and back end

We used Google Cloud for hosting our back-end, as well as storing our user.txt (text file for user credentials) and user log. Part of file sharing is Iso hosted on Cloud.

Brian Wei and Sagarkumar Patel

Bucket details

securechat-users-bucket

Location	Storage class	Public access	Protection
us-east1 (South Carolina)	Standard	Not public	Soft Delete

Objects Configuration Permissions Protection Lifecycle Observability New Inventory Reports Operations

Folder browser 1K Buckets > securechat-users-bucket

Create folder Upload Transfer data Other services

Filter by name prefix only Filter objects and folders 1 Show Live objects only

Name	Type	Created	Storage class
logs/	Folder	—	—
users.txt	text/plain	May 10, 2025, 3:39:01 PM	Standard

Buckets + Create Refresh

Filter buckets

Name	Created	Location type	Location	Default storage class	Last modified	Public ac
cpsc-455-securechat_cloudbuild	Apr 26, 2025, 4:49:54 PM	Multi-region	us	Standard	Apr 26, 2025, 4:49:54 PM	Subject to review
securechat-filecloud-hosting	May 4, 2025, 12:42:48 PM	Region	us-west2	Standard	May 4, 2025, 12:45:08 PM	⚠️ Public
securechat-users-bucket	Apr 26, 2025, 5:23:27 PM	Region	us-east1	Standard	Apr 26, 2025, 5:23:27 PM	Not public

Our front end is hosted on Firebase:

secure-chat

Spark plan Getting started? Tell Gemini about your project

+ Add app

Build

Hosting

Downloads (7d total) 186KB -47.7%

195K 97.7K 0 May 3 May 4 May 5 May 6 May 7 May 8 May 9

Deployment history

Deployed May 3, 2025 5:23 PM

Deployed by sagarpatel7699@gmail.com

This week Last week

15. Extra Features

These are features that are not required to be implemented, but we decided to add it in for a better user experience.

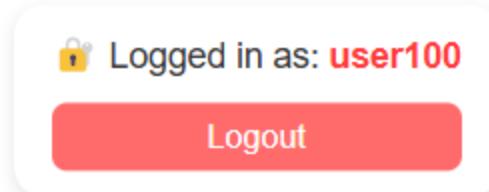
Notification Feature

User List

- **17 (1)**
- 10
- 11
- 14
- 19
- 33
- 44

The user is notified that there is one new message from 17. Clicking on 17 makes the notification disappear.

Logout Feature



At the top right shows your username and a logout button you can use after you are tired chatting.