USER GUIDE - Secure Chat

(Project 1- Group 23 - CPSC 455-02 14041)

Sana Mansoori (CWID: 883033920, sana.mansoori@csu.fullerton.edu)

Jaytee Okonkwo (CWID: 890207855) jokon@csu.fullerton.edu)

I. INTRODUCTION

SecureChat is a real-time chat application built using Flask and WebSockets. It enables multiple users to connect, send messages instantly, and handle connections securely.

II. TECHNOLOGIES USED

- Flask – Handles the web interface and user session management.
- WebSockets – Enables real-time communication between users.
- HTML/CSS/Javascript - Acts as the websocket client.
- Flask-SQLAlchemy – Manages user database and authentication.
- Flask-Bcrypt – Securely hashes user passwords.
- SSL/TLS Encryption – Secures WebSocket communication.
- SQLite – Lightweight database for user authentication.

III. INSTALLATION & SETUP

3.1 PREREQUISITES

- Python 3.x installed
- Web browser (Google Chrome, Firefox, etc.)

3.2 GITHUB REPOSITORY

You can access the GitHub repository https://github.com/mansoorisana/securechat
Or even clone it using below commands:

git clone https://github.com/mansoorisana/securechat.git
cd yourrepo

3.3 INSTALLING DEPENDENCIES

Run the following command in the project directory:
pip install -r requirements.txt

3.4 SETTING UP ENVIRONMENT VARIABLES

Create a .env file in the project root and add:
SECRET_KEY=your-secure-random-key

3.5 GENERATING SSL CERTIFICATE (for WSS Support)

SecureChat requires an SSL certificate to enable encrypted WebSocket (wss://) communication. For current scope we will use a self-signed certificate.
Running the following command will create an SSL certificate and key with default filenames:
openssl req -x509 -newkey rsa:4096 -keyout your_key.pem -out your_cert.pem -days 365 -nodes

If you used different filenames for your certificate and private key, add them to the .env file:
SSL_CERT_PATH=your_custom_cert.pem
SSL_KEY_PATH=your_custom_key.pem

Replace your_custom_cert.pem and your_custom_key.pem with your actual variable names.

IV. RUNNING THE APPLICATION

Start the Flask and WebSocket server with:
python websocket.py

The application will be available at: 🔗 https://localhost:5000/home
The WebSocket server runs at: 🔗 wss://localhost:8765/

V. USING THE CHAT APPLICATION

5.1 ACCESSING THE CHAT ROOM

1. Open https://127.0.0.1:5000/home in your browser.
2. If using a self-signed certificate, add it to the browser as a trusted certificate.
3. Signup with username & password.
4. You will be prompted to login after successful signup & click **Login**
5. You will be redirected to the chat room https://127.0.0.1:5000/chat

5.2 SENDING MESSAGES

- Type a message in the input box and press Enter.
- Messages will appear instantly in the chat room.

5.3 CONNECTION HANDLING

- INITIAL CONNECTION:
    - The server listens on wss://localhost:8765 with SSL.
    - When a client connects, it sends a username as the first message.
    - The server verifies the username against the database before allowing communication.

- DISCONNECTION:
    - Click **Leave Room** to exit.
    - Your session will be cleared.

- RECONNECTION:
    - Server uses a PING/PONG mechanism with ping_interval = 10s and ping_timeout = 5s to detect broken connections.
    - If there is no response within the timeout, the client is considered disconnected.
    - Client attempts reconnection after 3 seconds if unexpected interruption occurs.

5.4 RATE LIMITING

- Users can send up to 5 messages in 10 seconds.
- If a user exceeds this limit:
    - They receive a warning message:
      "Rate limit exceeded! Please wait 10 seconds before sending more messages."
    - Messages sent during the mute period are ignored.