Secure WebSocket Chat Application

Author: Curtis Quan-Tran
Date: 2/13/2025

Introduction

This is a real-time chat application that enforces **secure communication (WSS only)** by blocking **insecure WebSockets (ws://)** and allowing only **secure WebSockets (wss://)**.

Technology Used

- **Frontend**: JavaScript
- **Backend**: Python
    - Flask
    - Flask-SocketIO
    - Flask-Sessions
    - gevent
    - gevent-websocket
- **Security**:
    - Self-signed SSL for WebSockets using Gevent
    - SHA-256 for credential hashing.
- **Database**: JSON-based user storage

User Guide

Installation Guide

1. Clone the Repository and set up the virtual environment (Linux).

```
git clone https://github.com/curtisqt30/websocket-project.git
cd websocket-project
```

```
python3 -m venv venv
source venv/bin/activate
```

2. Install the required dependencies.

```
pip install -r requirements.txt
```

3. Create a Self-Signed SSL certificate.

```
openssl req -x509 -newkey rsa:2048 -keyout key.pem -out cert.pem -days 365
-nodes
```

Running the Application

1. Run the WebSocket Server from the terminal

```
python3 app.py
```

2. Open your default web browser and enter the URL.

```
https://localhost:5000/
```

- Troubleshoot: Browsers might show an **insecure certificate warning**. However, click **Advanced** and **accept the risk** since it's a **self-signed certificate**.
- Note: Ensure **port 5000** is open if you are using a firewall.

How to Use

1. Register/Login

- **New users:** Click **register**, and enter a username & password.
- **Existing users:** Log in with your **credentials**.

2. Join the Chat

- The chat interface allows users to send messages in real time.
  - Messages have a **limit of 50 characters**.
  - Users can only send **one message per second**.
  - **Inactive users** who haven't sent messages in **20 seconds are alerted** of forceful disconnection.

        - To prevent disconnecting from the chatroom users have to send a message.
        - Otherwise, **10 seconds after the warning**, inactive users are forcefully disconnected and are redirected to the login page.
- Messages are broadcast to all connected users.

3. Logout

- Click **leave chat** to log out and get redirected to the log-in page.
- Alternatively, exit out the tab/browser.

Additional Notes

Key Features

- Rate Limiting
    - 50 Character Limits
- Inactivity Time Limit
    - **Warning** after 20 seconds
    - **Forceful removal** after 30 seconds.
    - One Message per Second
- User Authentication
    - Username/Password Authentication
    - Credentials are **hashed** using SHA-256 and are stored in a JSON file.
- Partially Secure Real-time Communication
    - **Real-time** chat messaging using Flask WebSockets
    - Uses a **Self-Signed SSL Certificate**.
    - **Enforce** HTTPS and **reject** HTTP Connections.
- Web-Based Application
    - **Accessible** via a browser.

Future Improvements

- User Input Validation/Sanitization
    - Implement **validation** to prevent malicious input.
- Improve Activity Monitoring
    - Track mouse movement and key pressing to **improve inactivity detection**.
- Improve Chat Room Support
    - **Limit** the number of users per room.

    - Implement **multiple** chat rooms instead of a single global chat room.
- Improve Session Management

AI Assistance & Contribution Acknowledgement

- Frontend Development
    - AI was used to assist with developing the **UI structure**.
        - style.css, login.html, chat.html, and register.html
- Backend Development
    - **Error handling**, **debugging**, and **logging** mechanisms were made based on AI recommendations.
- Security Considerations
    - Developing **session timeouts** and **character limits** was aided with AI.
- Documentation Writing
    - Some sections in this user guide, such as **formatting, wording, and structuring**, were AI-Assisted.