Instant Messaging (IM) Platform Project Documentation

Project Overview

The Instant Messaging (IM) Platform is a real-time communication system designed to enable users to exchange messages securely and efficiently. This project incorporates key networking concepts and provides a robust client-server architecture with user authentication and messaging capabilities.

Team Members:

- Olufewa Favour Alonge
- Timilehin Falusi

Key Features

1. **Network Architecture**
    - Central server handles multiple client connections using TCP/IP.
    - Client applications connect to the server for all operations.
    - SQLite database used for secure and scalable user authentication.
2. **User Authentication**
    - Register a user with a unique username and password.
    - Prevent duplicate registrations.
    - Authenticate users during login.
3. **Messaging Features**
    - **Broadcast Messaging**: Send messages to all online users.
    - **Private Messaging**: Send direct messages to specific users.
4. **Online Users**
    - View a list of currently online users.
5. **Disconnection Handling**
    - Users can gracefully disconnect from the server.

Installation and Setup

Requirements

- Python 3.7 or higher
- SQLite (comes pre-installed with Python)

Steps to Set Up

1. Clone the repository or extract the provided project files.
2. Ensure the following files are in the project directory:
    - server_script.py
    - client_script.py
    - users.db (optional; created automatically if not present)
3. Run the server:
4. python server_script.py
5. Run the client:
6. python client_script.py

Repeat this step for multiple clients.

Usage Guide

Commands

- **REGISTER <username> <password>**: Registers a new user with the given username and password.
- **LOGIN <username> <password>**: Logs in with the provided credentials.
- **ONLINE**: Displays a list of currently online users.
- **MESSAGE <message>**: Sends a broadcast message to all online users.
- **PRIVATE <username> <message>**: Sends a private message to the specified user.
- **QUIT**: Disconnects from the server.

Code Structure

Server Script (server_script.py)

- **Database Initialization**:

    - Creates an SQLite database to store user credentials.
- **Command Handling**:
    - Processes commands sent by clients (e.g., REGISTER, LOGIN, MESSAGE).
    - Tracks online users and manages communication.
- **Threading**:
    - Handles multiple client connections concurrently.

Client Script (client_script.py)

- **Server Connection**:
    - Establishes a TCP connection to the server.
- **Command Interface**:
    - Provides an interactive console for users to execute commands.
- **Error Handling**:
    - Manages connection errors and invalid commands.

Testing and Validation

Test Cases

1. **User Registration**:
    - Attempt to register with a unique username.
    - Try registering with an existing username (should fail).
2. **Login**:
    - Login with valid credentials.
    - Attempt login with invalid credentials (should fail).
3. **Messaging**:
    - Send a broadcast message and verify delivery to all users.
    - Send a private message and verify delivery to the specified user.
4. **Online Users**:
    - Verify the list of online users reflects active connections.
5. **Disconnection**:
    - Ensure users are removed from the online list upon disconnection.

Future Improvements

1. **Enhanced Security**:
    - Encrypt passwords in the database.
    - Use SSL/TLS for secure client-server communication.
2. **Scalability**:
    - Use a distributed server architecture for handling more clients.
3. **User Interface**:
    - Develop a graphical user interface (GUI) for easier interaction.
4. **Additional Features**:
    - Add group messaging.
    - Implement user status (e.g., Away, Busy).

Acknowledgments

This project was developed as part of our coursework for EECS 563 (Introduction to Communication Network).