A Multi-user Chat Application

CS440 Computer Networks, Spring 2024-2025

start presentation

Authors:

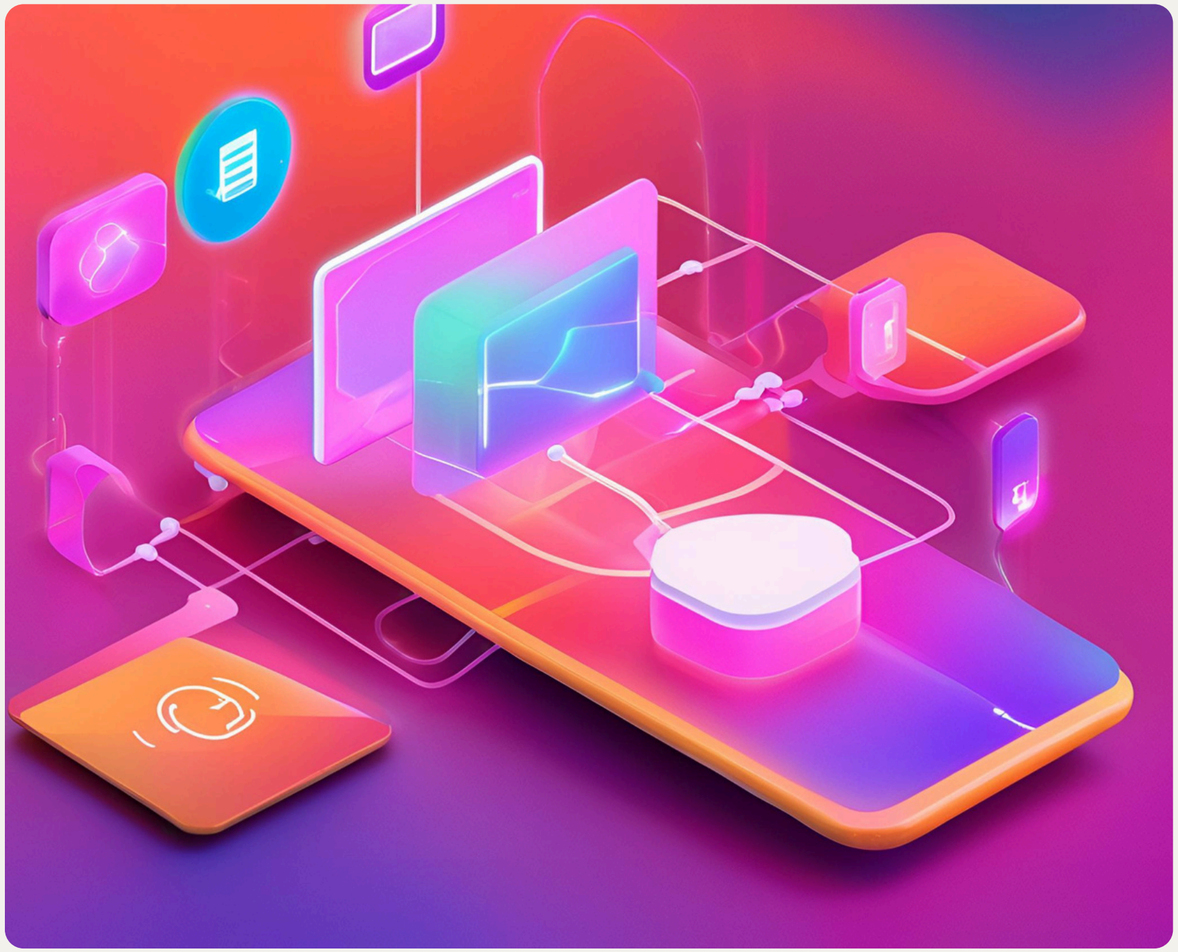
- Nguyễn Hoài Duy (2202087)
- Huỳnh Văn Đông (2202086)
- Trần Phương Nam (2202085)
- Nguyễn Thanh Tuấn (2202084)
- Võ Hữu Nhân (2202083)

Introduction

Real-time chat app using Python

Key features:

- Login / Register
- General, private, and group messaging
- Real-time communication
- Built with Streamlit, Socket Programming, Threading



Technologies Used

- **Streamlit: UI interface**
- **Socket Programming: Client-server communication**
- **Threading: Receive messages in background**
- **Queue: Store & manage incoming messages**
- **Datetime, JSON: Timestamping & message formatting**
- **Streamlit AutoRefresh: UI updates every 2s**

System Architecture

- **Clients <-> Server with Socket**
- **MongoDB for user/group storage**
- **Threads per client**
- **Message queue for async communication**

Session State Management (Client)

Streamlit session_state variables:

- `username`, `client_socket`, `messages`, `connected`
- `auth_status`, `message_queue`, `receive_thread`, `last_refresh`

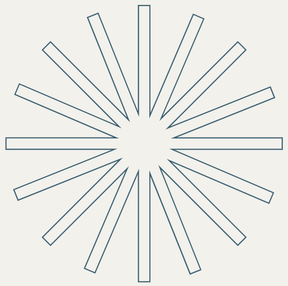
Ensures persistent UI state during chat

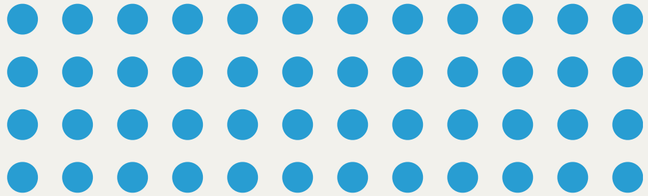
Session State Management (Client)

Ensures persistent UI state during chat

```
 Initialize session state
f 'username' not in st.session_state:
   st.session_state.username = None
f 'client_socket' not in st.session_state:
   st.session_state.client_socket = None
f 'messages' not in st.session_state:
   st.session_state.messages = []
f 'connected' not in st.session_state:
   st.session_state.connected = False
f 'auth_status' not in st.session_state:
   st.session_state.auth_status = None
f 'message_queue' not in st.session_state:
   st.session_state.message_queue = queue.Queue()
f 'last_refresh' not in st.session_state:
   st.session_state.last_refresh = time.time()
f 'receive_thread' not in st.session_state:
   st.session_state.receive_th ↓ d = None
```

Server Connection & Authentication





- Connect to localhost:5555 using socket
- Authentication flow:
  - Choose Login/Register → Enter credentials
  - Server validates → responds with result
- Starts message receiving thread after success

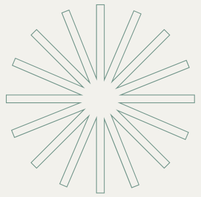
Message Types & Sending

- **General Chat: Sent to all**
- **Private Message: /msg <user> <msg>**
- **Group Message:**
  - **/create_group <name>**
  - **/join_group <name>**
  - **/group_msg <group> <msg>**

Receiving Messages

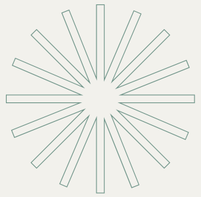
- Background thread continuously calls recv()
- Ignores control keywords (e.g., USERNAME)
- Adds actual messages to message_queue

User Interface (UI)



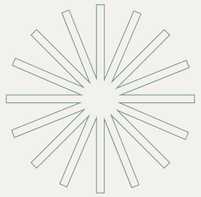
- **Sections:**
- **Authentication form (login/register)**
- **Chat display: styled per message type**
- **Message form with type selection**

Real-time Updates



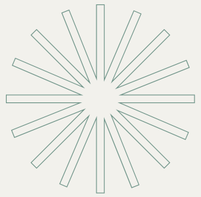
- threading.Thread keeps UI responsive
- st_autorefresh(interval=2000) reloads UI

Error Handling



- **Handles:**
- **Connection timeouts**
- **Wrong credentials**
- **Invalid commands**
- **Disconnected sockets**

Conclusion



- A full-featured real-time chat platform
- Combines networking, UI, multithreading
- Extendable to support:
- File sharing, online presence, emojis, etc.

Demo (Optional)

- **Screenshot series or live demo:**
    1. **Login/Register**
    2. **General chat**
    3. **Private & Group message interaction**

Q&A

Invite audience to ask questions


Thank
Youu