HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA ĐÀO TẠO SAU ĐẠI HỌC

BÁO CÁO

CÁC HỆ THỐNG PHÂN TÁN

ĐỀ TÀI: HỆ THỐNG CHAT NGANG HÀNG (P2P)

Giảng viên hướng dẫn : TS. Kim Ngọc Bách

Lớp : M25CQHT01-B

Nhóm 11

Thạch Tuấn Cường : B25CHHT007

Nguyễn Mạnh Quân : B25CHHT048

Vũ Anh Tuấn : B25CHHT059

Hà Nội – 2025

Mục lục

I. Phân tích yêu cầu.

1. Giới thiệu bài toán

Trong những năm gần đây, các ứng dụng giao tiếp thời gian thực như chat, gọi điện và truyền dữ liệu trực tuyến đã trở thành một phần không thể thiếu trong đời sống số. Phần lớn các hệ thống này được xây dựng theo mô hình Client–Server truyền thống, trong đó mọi dữ liệu đều phải đi qua máy chủ trung tâm. Mô hình này tuy dễ triển khai nhưng bộc lộ nhiều hạn chế về khả năng mở rộng, độ trễ và chi phí vận hành khi số lượng người dùng tăng cao. Xuất phát từ thực tế đó, nhóm lựa chọn nghiên cứu và xây dựng một ứng dụng chat theo mô hình Hybrid Peer-to-Peer nhằm tận dụng ưu điểm của hệ thống phân tán.

2. Mục tiêu hệ thống

Mục tiêu của đề tài là xây dựng một ứng dụng chat thời gian thực hoạt động theo mô hình Hybrid Peer-to-Peer, trong đó server trung tâm chỉ đảm nhiệm các chức năng điều phối như xác thực người dùng, quản lý quan hệ xã hội và hỗ trợ signaling WebRTC. Dữ liệu chat và file được truyền trực tiếp giữa các peer, giúp giảm tải cho server, tăng hiệu năng và đảm bảo khả năng mở rộng.

3. Đối tượng sử dụng trong hệ thống

Đối tượng sử dụng của hệ thống bao gồm:
- Người dùng cuối: sử dụng ứng dụng để đăng ký tài khoản, đăng nhập, kết bạn và trao đổi tin nhắn.
- Server trung tâm: thực hiện xác thực, điều phối kết nối và signaling.
- Peer: các client trực tiếp thiết lập kết nối P2P để trao đổi dữ liệu.

II. Thiết kế phần mềm

1. Tổng quan kiến trúc hệ thống

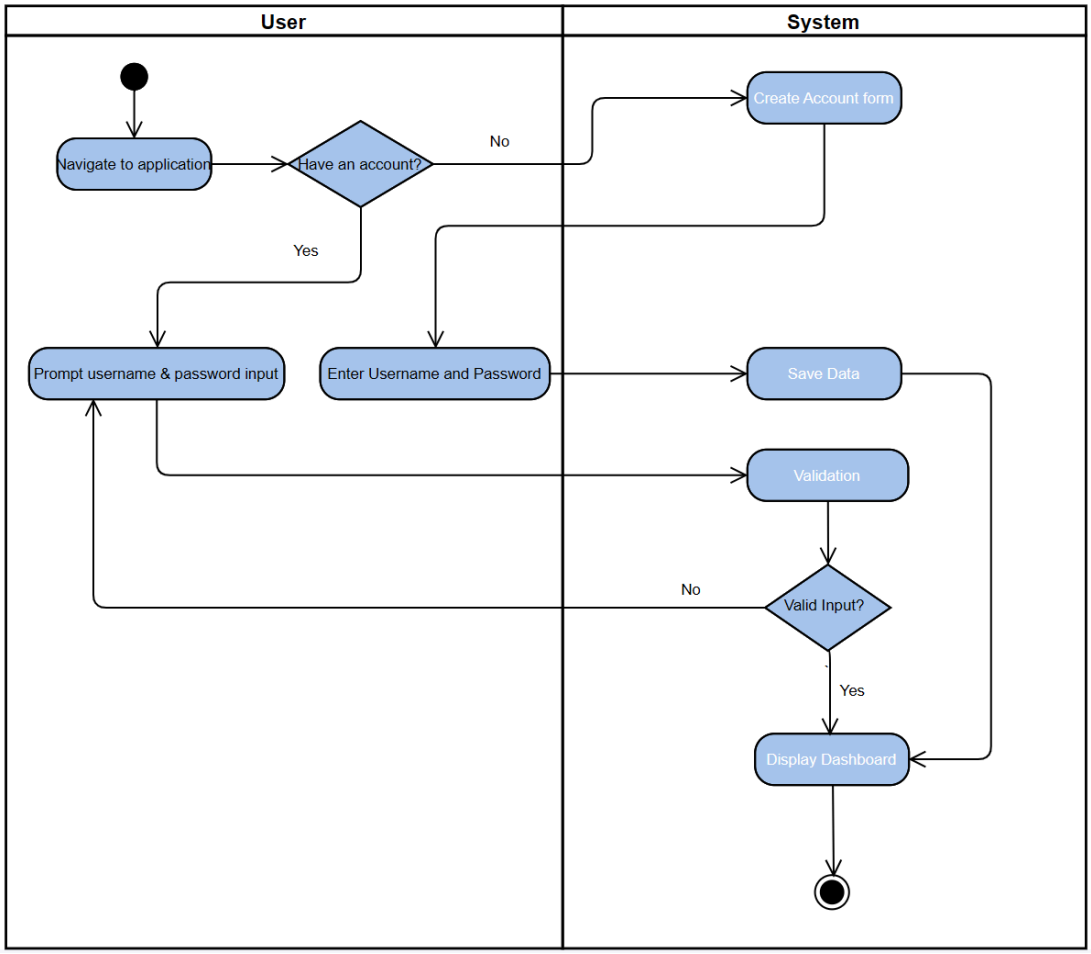
Hệ thống hoạt động theo mô hình **Hybrid P2P**:

1. **Server (Backend + DB):** Đóng vai trò là *Directory Server*. Chịu trách nhiệm xác thực người dùng, lưu trữ danh sách bạn bè, cấu trúc nhóm và hỗ trợ kết nối ban đầu (Signaling).
2. **Client (Frontend):** Đóng vai trò là các *Peers*. Sau khi được Server hỗ trợ kết nối, các Client sẽ thiết lập đường truyền trực tiếp (WebRTC DataChannel) để trao đổi tin nhắn và file mà không đi qua Server.

2. Yêu cầu chức năng

A. Quản lý người dùng và xác thực

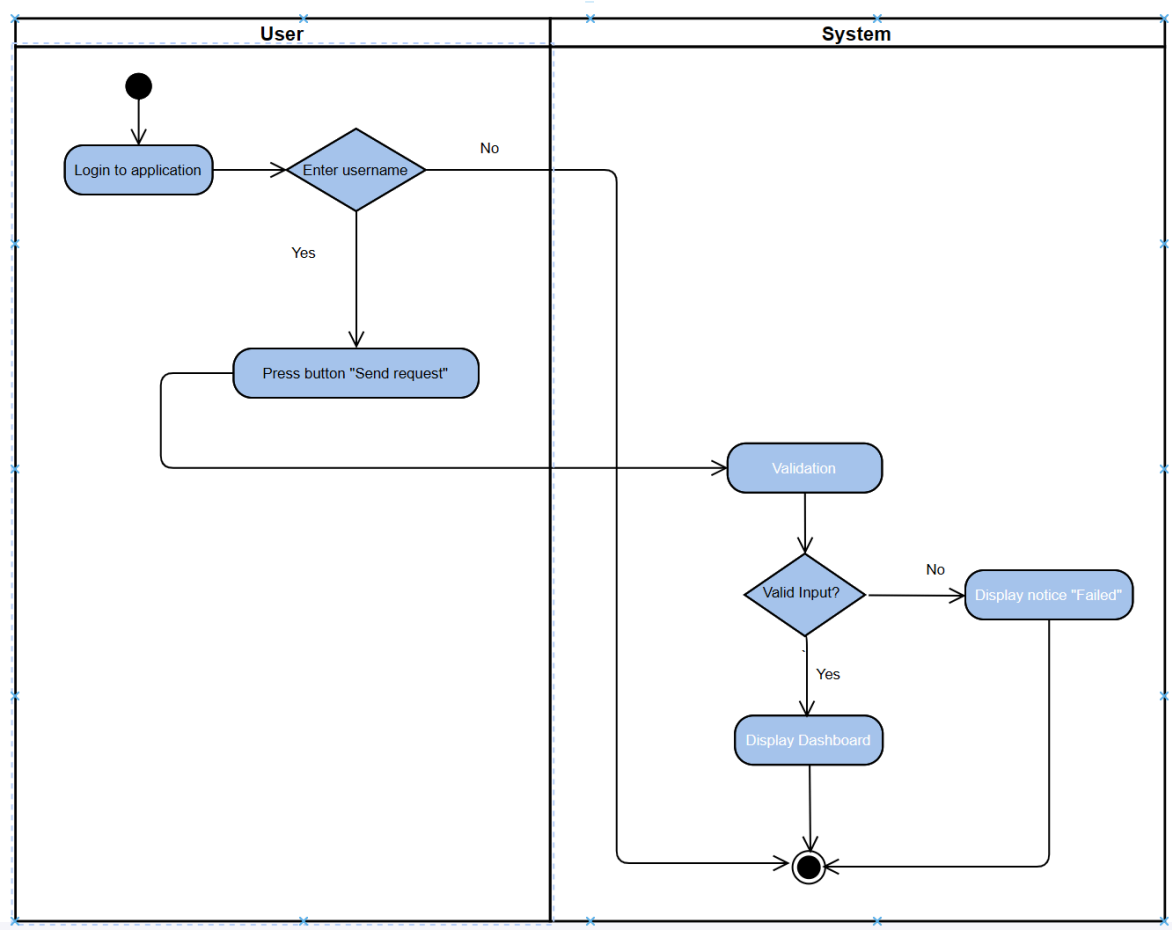
- **Đăng ký/Đăng nhập:** Người dùng đăng ký tài khoản mới với Username/Password, mật khẩu được mã hoá bằng thuật toán bcrypt và đăng nhập để nhận JWT (JSON Web Token) dùng cho việc xác thực API và WebSocket.



- **Quản lý trạng thái:** Hệ thống phải hiển thị trạng thái Online/Offline của bạn bè theo thời gian thực.
- **Rate Limiting:** Hệ thống có cơ chế giới hạn tốc độ request (Rate Limit) để chống spam/DDoS (triển khai token bucket algorithm).

B. Quản lý mối quan hệ (Social Graph)

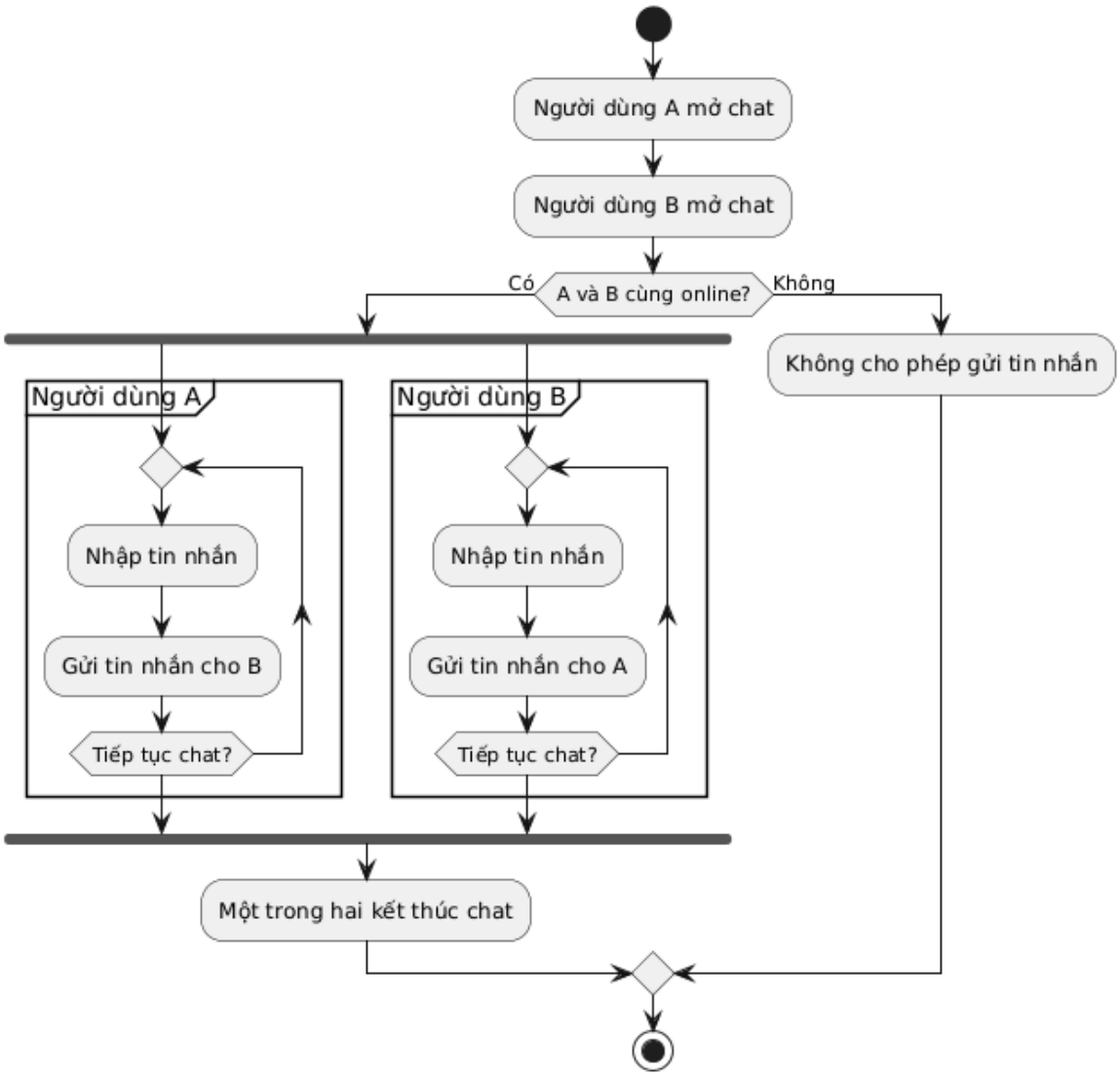
- **Tìm kiếm người dùng và kết bạn:** Cho phép tìm kiếm người dùng khác theo username trong hệ thống để kết bạn. Server lưu thông tin



- **Danh sách bạn bè:** Hiển thị danh sách bạn bè hiện có.
- **Quản lý nhóm:**
    - Tạo nhóm chat mới (Người tạo nhóm tự động là trưởng nhóm).
    - Mời bạn bè vào nhóm (Trưởng nhóm / thành viên).
    - Rời nhóm.

C. Tính năng Chat P2P (WebRTC DataChannel)

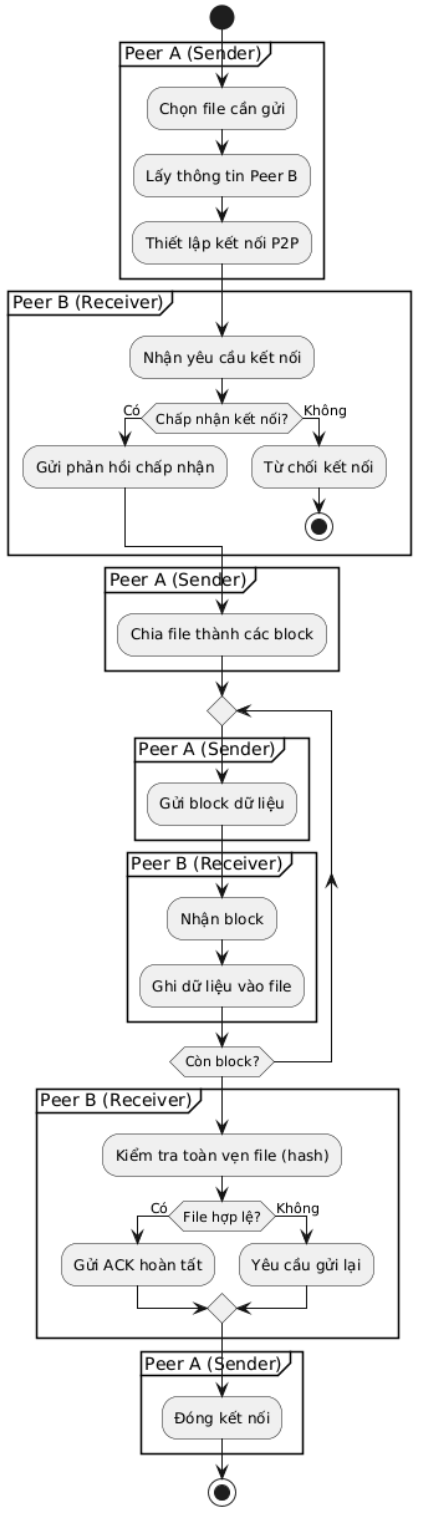
- **Chat 1-1 (Direct Message):** Gửi tin nhắn văn bản trực tiếp giữa hai người dùng thông qua WebRTC.
- **Chat Nhóm (Group Mesh):** Hỗ trợ chat nhóm theo mô hình **Full Mesh** (mỗi thành viên trong nhóm kết nối trực tiếp với tất cả thành viên còn lại).



D. Truyền tải File P2P (File Transfer)

- Hỗ trợ gửi file trực tiếp giữa các client thông qua kết nối P2P sử dụng WebRTC DataChannel. Để đảm bảo trải nghiệm người dùng và độ ổn định của trình duyệt, hệ thống định hướng hỗ trợ các file có kích thước vừa và nhỏ (ví dụ dưới 25MB).
- **Chunking (Chia nhỏ file):** File được chia thành các chunk nhỏ (16KB) để gửi qua DataChannel.
- **Cơ chế gửi:** Bao gồm các bước gửi Metadata (tên, size, checksum) -> Gửi các Chunk -> Client nhận lắp ráp lại file và check toàn vẹn.

Activity Diagram - Chức năng truyền file P2P



3. Yêu cầu phi chức năng

- **Tính nhất quán (Consistency):** Đảm bảo thứ tự tin nhắn đúng đắn ngay cả khi đồng hồ vật lý của các máy trạm không đồng bộ (Sử dụng **Lamport Clock**).
- **Tính sẵn sàng (Availability):** Server trung tâm chỉ phục vụ metadata; nếu server quá tải, các kết nối P2P hiện tại vẫn hoạt động bình thường.
- **Độ trễ thấp:** Tin nhắn đi trực tiếp giữa các peer, giảm độ trễ so với mô hình Client-Server truyền thống.

4. Công nghệ và thư viện sử dụng

Backend

- **Ngôn ngữ:** Go (Golang) 1.24.
- **Framework:** Gin (REST API).
- **WebSocket:** Gorilla WebSocket.
- **Database:** MySQL.
- **Security:** bcrypt cho mật khẩu, JWT cho xác thực phiên.

Frontend

- **Framework:** Vue 3 (Composition API).
- **State Management:** Pinia.
- **Build Tool:** Vite.
- **UI Library:** Ant Design Vue.
- **Core Logic:** WebRTC API (RTCPeerConnection, RTCDataChannel).

5. Thiết kế cơ sở dữ liệu

Cơ sở dữ liệu (MySQL) chỉ lưu trữ metadata, không lưu nội dung chat.

Tên bảng	Chức năng chính	Các trường quan trọng

users	Lưu trữ thông tin định danh.	user_id (UUID), username, password_hash (bcrypt).
friend_requests	Quản lý hàng đợi lời mời kết bạn.	from_user_id, to_user_id, status (pending/accepted).
friends	Lưu trữ đồ thị xã hội (Social Graph).	user_id, friend_user_id (Lưu 2 chiều để truy vấn nhanh).
groups	Lưu trữ thông tin nhóm chat.	group_id, name, owner_user_id.
group_members	Lưu trữ thành viên trong nhóm.	group_id, user_id, role.
presence	Lưu trạng thái hoạt động (Volatile data).	user_id, status (online/offline), last_seen.

6. Thiết kế giao thức giao tiếp

A. Signaling (Tín hiệu hóa qua WebSocket)

Để hai máy trạm nằm sau NAT/Firewall có thể tìm thấy nhau, hệ thống sử dụng giao thức Signaling qua WebSocket:

- **Protocol:** JSON over WebSocket.
- **Message Types:**
    - signal.offer / signal.answer: Trao đổi thông tin SDP (Session Description Protocol).
    - signal.ice: Trao đổi các ứng viên mạng (ICE Candidates).

- presence.update: Thông báo trạng thái người dùng.

B. Data Transport (Vận chuyển dữ liệu qua WebRTC)

- **Công nghệ:** WebRTC DataChannel (SCTP protocol).

Cấu trúc tin nhắn P2P:

```
{
    "type": "chat.message",
    "msgId": "UUID",
    "text": "Nội dung chat",
    "lamport": 12,        // Đồng hồ logic
    "clientTimestamp": 1713500000
}
```

7. Giải thuật phân tán ứng dụng

A. Đồng bộ thứ tự tin nhắn (Logical Clock)

Do mạng P2P có độ trễ bất định và đồng hồ máy tính không đồng bộ, hệ thống sử dụng thuật toán **Lamport Timestamps**:

1. Mỗi Client duy trì một biến đếm L.
2. Khi gửi tin nhắn: L = L + 1, gắn L vào tin nhắn.
3. Khi nhận tin nhắn có timestamp L_msg: Cập nhật L = max(L, L_msg) + 1.

Sắp xếp: Tin nhắn hiển thị được sắp xếp dựa trên (Lamport Clock, Client Timestamp, Message ID).

B. Heartbeat & Failure Detection

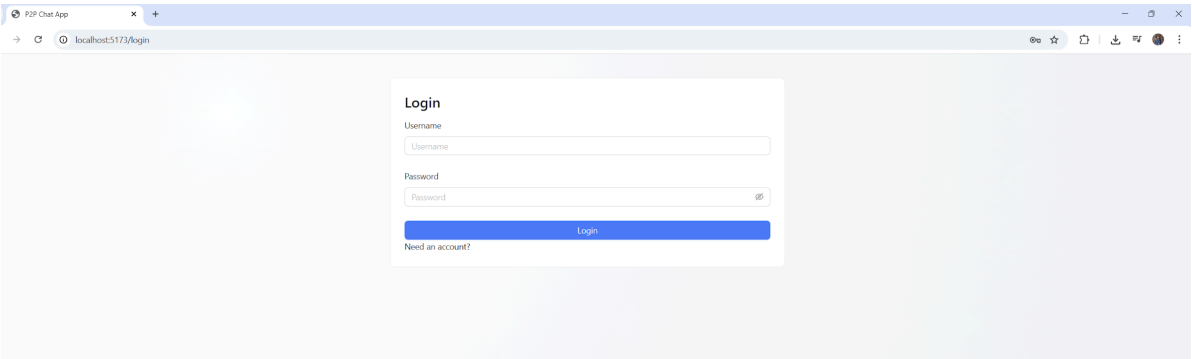
Để phát hiện một peer bị mất kết nối trong mạng ngang hàng:

- Client gửi gói tin chat.ping định kỳ (mỗi 3 giây).
- Nếu không nhận được chat.pong trong khoảng thời gian ngưỡng (9 giây), Client sẽ tự động đóng kết nối và thông báo trạng thái "Disconnected".

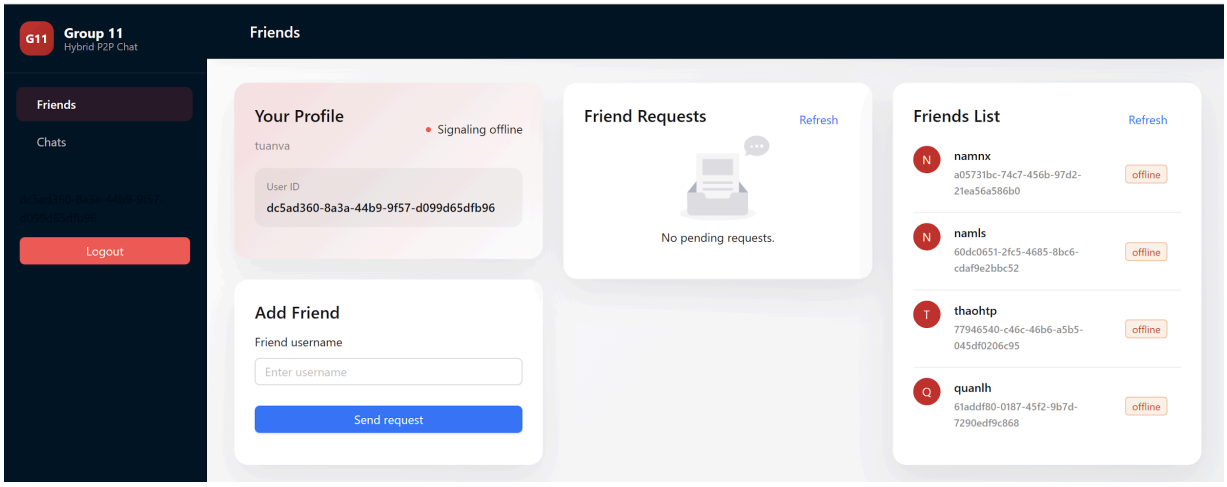
8. Kiến trúc chat nhóm

- **Topology:** Full Mesh (Lưới đầy đủ).
- **Cơ chế:**
  - Nhóm có N thành viên.
  - Mỗi thành viên duy trì N-1 kết nối P2P tới các thành viên còn lại.
  - Khi gửi tin nhắn, Client lặp qua danh sách các kết nối (Peers) và gửi bản sao tin nhắn tới từng kết nối.
  - **Ưu điểm:** Không cần server trung chuyển, giảm tải server.
  - **Nhược điểm:** Tốn băng thông upload của client khi nhóm quá lớn (giới hạn số lượng thành viên nhỏ).

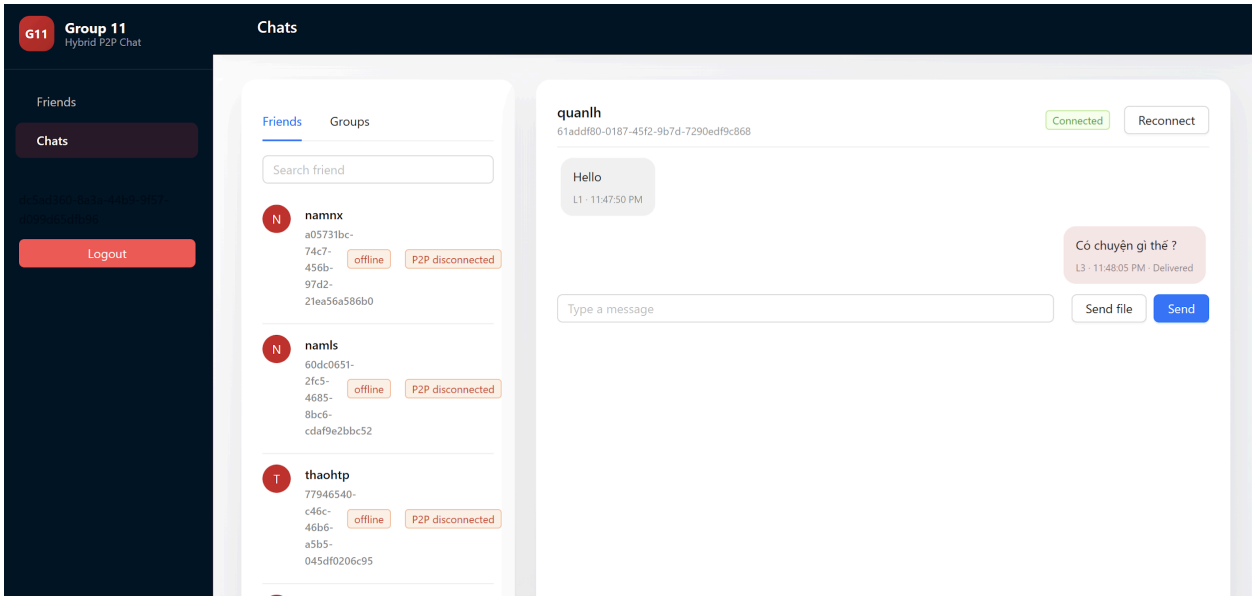
9. Giao diện ứng dụng demo



Giao diện đăng ký/ đăng nhập của hệ thống



Giao diện khi đã login vào hệ thống



Giao diện cửa sổ chat

III. Kết luận

Hệ thống xây dựng thành công một ứng dụng chat phân tán lai ghép. Việc tách biệt lớp điều khiển (Control Plane - Server) và lớp dữ liệu (Data Plane - P2P) giúp hệ thống đảm bảo tính riêng tư dữ liệu người dùng, đồng thời tối ưu hóa tài nguyên máy chủ. Các thuật toán phân tán như Lamport Clock và Full Mesh topology đã giải quyết tốt các vấn đề về đồng bộ và kết nối trong môi trường mạng ngang hàng.