Phase 1: UML Design

We need to design the system's structure using **UML class diagrams** that focus on the following key areas:

1. **User**: Represents a chat user with attributes like `username`, `status`, and the rooms they're in.
2. **Chat Room**: Represents a chat room with a list of users.
3. **Message**: Different types of messages exchanged in the chat (public, private, room-based).
4. **Server**: The core entity that manages users, rooms, and messaging.
5. **Client**: Represents the user's interface and handles communication with the server.

Key Entities for UML

- **User**: `username`, `status (ACTIVE/AWAY/BUSY)`, `rooms (list of rooms)`, `message queue`
- **Room**: `roomName`, `users`
- **Message**: `type (TEXT, PUBLIC_TEXT, etc.)`, `content`, `sender`, `receiver`
- **Server**: `users (list of all connected users)`, `rooms (list of all created rooms)`, methods for handling user login, messaging, and room management
- **Client**: Methods for sending and receiving messages, connecting to server

---

Phase 2: UML Class Diagram

I'll sketch out a UML class diagram to represent these entities. Here's a **high-level overview** of the relationships:

```
+--------------------+            +--------------------+
|       User        |    |                Message       |
+--------------------+            +--------------------+
| - username: String |             | - type: String     |
| - status: String   |             | - content: String  |
| - rooms: List<Room>|            | - sender: String   |
+--------------------+            | - receiver: String |
                                  +--------------------+

+--------------------+      +--------------------+
|      Server       |<--------|     Client       |
+--------------------+      +--------------------+
| - users: List<User>|       | - connect()       |
| - rooms: List<Room>|       | - sendMessage()    |
| + identifyUser()   |       | - receiveMessage() |
| + sendText()       |       | - disconnect()     |
+--------------------+      +--------------------+

+--------------------+
|      Room          |
+--------------------+
| - roomName: String |
| - users: List<User>|
+--------------------+
```

Next Step: Confirmation

Once the UML is confirmed, we'll move to **Phase 3: C# Implementation**. The implementation will involve:

1. **Server-Side**:
    - Handling user identification, messaging, rooms, and statuses.
    - Managing connections, sending responses, and processing chat messages.
2. **Client-Side**:
    - Connecting to the server, sending commands (identify, text, status change, etc.), and receiving updates.

Let me know if you need changes to the **UML design**, and I'll get the code started!

```
ChatApp/
│
├── Server/
│   └── ChatServer.cs
├── Client/
│   └── ChatClient.cs
└── Models/
    ├── User.cs
    ├── Room.cs
    └── Message.cs
```