

***19CSE301***

***COMPUTER NETWORKS***

**TCP Client Server Chat Application**

***TEAM – 5***

|  |  |  |
| --- | --- | --- |
| S.No | Name | Roll No |
| 1 | Prasad Simhadri | CB.EN.U4CSE22438 |
| 2 | Siddhardh Reddy | CB.EN.U4CSE22439 |
| 3 | Vamsi Krishna | CB.EN.U4CSE22458 |
| 4 | Shanmuka Vardhan | CB.EN.U4CSE22461 |

**­­­­­­­Project Title:**

**TCP Client Server Chat Application**

**Project description**:

The TCP Client-Server Chat Application is a real-time communication system that enables multiple clients to connect to a central server and exchange messages. By using the TCP protocol, the server ensures reliable message delivery while managing all client connections and routing messages between them. This application showcases the fundamental concepts of client-server architecture, network communication, and concurrent processing using multi-threading techniques.

**Scenario 1:** User Joining the Chat

* **Client Initialization:** The user launches the chat client application, initiating the connection process.
* **Nickname Registration:** The user is prompted to enter a unique nickname, which serves as their identifier within the chat.
* **Server Connection Establishment:** The client establishes a connection with the server, transmitting the chosen nickname for authentication and identification purposes.
* **New User Announcement**: The server broadcasts a notification to connected clients, announcing the arrival of the new user and updating the chat roster in real-time.
* **Active Chat Participation:** The user is now able to send and receive messages, engaging fully in the chat conversation.

**Scenario 2:** User sending a message

* **Message Composition:** The user crafts a message in the chat interface, preparing to share their thoughts with the group.
* **Message Forwarding:** The client sends the composed message to the server, which acts as a central hub for message distribution.
* **Message Rendering:** All users receive and display the message in their respective chat windows, creating a seamless and interactive conversation experience.

**Scenario 3:** User leaving the chat

* **Client Termination**: The user closes the chat client or disconnects from the server, initiating the disconnection process.
* **Server Detection**: The server detects the disconnection and updates its active user list, removing the disconnected user from the roster.
* **Disconnection Notification**: The server broadcasts a notification to connected clients, informing them that the user has left the chat and ensuring that everyone is aware of the change in the conversation dynamics.

**Objectives**:

* To design and develop a robust and efficient real-time chat application that facilitates seamless communication.
* To create a scalable server that can concurrently handle multiple client connections, ensuring a responsive and reliable experience.
* To guarantee secure and consistent message delivery by using the reliability features of the TCP protocol

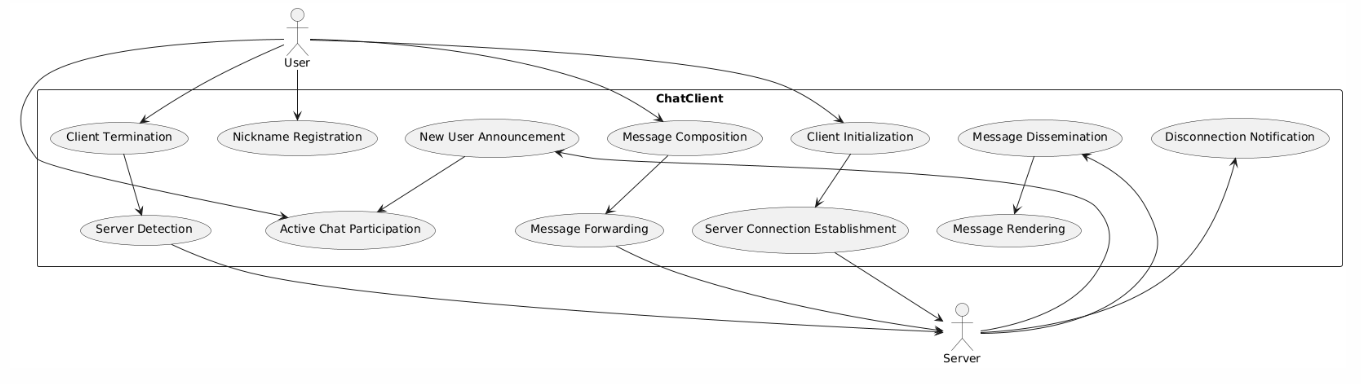
**Technologies required to use**:

* Python (for both server and client implementations)
* TCP/IP Protocol Suite
* Socket Programming
* Multi-threading

**Features:**

* **Instant Messaging**: Enables users to exchange messages in real-time, facilitating instant communication.
* **Scalable Connectivity**: The server supports multiple clients, allowing for group conversations and a dynamic user experience.
* **Personalized Identification**: Each user is assigned a unique nickname, ensuring clear identification and distinction within the chat.
* **Global Messaging:** Broadcasts messages to connected clients, ensuring that everyone is informed and up-to-date.

**Use case Diagram:**



**Communication Protocols:**

**TCP (Transmission Control Protocol):** Ensures reliable, ordered, and error-checked delivery of messages between the server and clients.

**Custom protocol commands:**

1. **NICK:** Command sent by the server to request the client's nickname.
2. **MSG:** Command used by clients to send messages to the server.
3. **JOIN/LEAVE:** Notifications sent by the server when users join or leave the chat.

**Benefits:**

* **Reliable Communication:** TCP ensures messages are delivered reliably and in the correct order.
* **Scalability:** The server can manage multiple clients, making it suitable for group communications.
* **User Identification:** Unique nicknames allow easy identification of users.
* **Cross-Platform Compatibility:** The application can be run on various operating systems without modification.

**Design Requirements:**

|  |  |  |
| --- | --- | --- |
| S.No | Requirement | Description |
| 1 | Concurrent Users | Support for at least 50 simultaneous clients. |
| 2 | Message Delivery Time | Messages should be delivered in under 1 second. |
| 3 | Network Protocol | |  | | --- | |  |  |  | | --- | | Use TCP for reliable and ordered message delivery. | |
| 4 | Error Handling | Detect and manage disconnections and network errors. |
| 5 | Security | Implement basic security measures to prevent unauthorized access. |
| 6 | User Interface | Provide a simple text-based user interface. |
| 7 | Platform Support | |  | | --- | |  |  |  | | --- | | The application should run on Windows, Linux, and MacOS. | |

**Protocol and Connection Specifications:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Protocol/Command** | **Purpose** | **Direction** | **Description** |
| TCP | Connection-oriented communication | Bi-directional | Ensures reliable message delivery between client and server. |
| NICK | Request the client's nickname | Server → Client | Ensures reliable message delivery between client and server. |
| MSG | |  | | --- | |  |  |  | | --- | | Send a message from the client to the server | | Server → Client | Clients send messages to the server, which are then broadcast to all users. |
| JOIN | Notify users of a new user joining | Server → Client | Broadcasts to clients when a new user joins the chat. |
| LEAVE | Notify users of a user leaving | Server → Client | Broadcasts to clients when a user leaves the chat. |

**Work Plan:**

* **Define the project scope, features, and technical requirements:** Outline the objectives, target audience, and technical specifications for the chat application.
* **Design the architecture and protocol specifications:** Develop a client-server model, component diagram, and communication protocols for efficient and secure interaction.
* **Implement the server and client code using socket programming:** Create server sockets for multi-threaded message handling and client interfaces for user interaction and message sending.
* **Test the application, identify, and fix bugs:** Conduct testing, logging errors and fixing issues iteratively.
* **Deploy the application and prepare documentation:** Set up hosting infrastructure, launch the application, and create documentation for future reference.

**Conclusion:**

The goal of the Chat Application project is to develop a reliable and approachable real-time communication platform that facilitates smooth communication between numerous users. An environment for users to connect and communicate will be provided by the program through the use of a client-server architecture and the implementation of essential features such user registration, real-time message exchange, and user presence awareness.