Task 3

Java-based Client-Server Chat Application using **Sockets and Multithreading**. This is ideal for your **internship deliverable** under *CODTECH*.

✅ Project Overview

- **Server** listens for client connections and broadcasts messages.
- **Clients** can send and receive messages in **real time**.
- **Multithreaded** server handles **multiple users simultaneously**.

📁 Project Files

1. `ChatServer.java` – Server-side code
2. `ChatClient.java` – Client-side code

🛠️ 1. ChatServer.java (Multi-user Chat Server)

```
import java.io.*;
import java.net.*;
import java.util.*;

public class ChatServer {

    private static Set<ClientHandler> clientHandlers = new
HashSet<>();

    public static void main(String[] args) {
        try (ServerSocket serverSocket = new ServerSocket(1234)) {
            System.out.println("Server started. Waiting for
clients...");

            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("New user connected: " +
clientSocket);
```

```
                ClientHandler handler = new
ClientHandler(clientSocket);
                clientHandlers.add(handler);
                new Thread(handler).start();
            }
        } catch (IOException e) {
            System.out.println("Server error: " + e.getMessage());
        }
    }

    // Broadcast to all clients
    public static void broadcast(String message, ClientHandler sender)
{
        for (ClientHandler client : clientHandlers) {
            if (client != sender) {
                client.sendMessage(message);
            }
        }
    }

    // Remove client on disconnect
    public static void removeClient(ClientHandler client) {
        clientHandlers.remove(client);
    }
}

// Handles each client
class ClientHandler implements Runnable {
    private Socket socket;
    private PrintWriter out;
    private BufferedReader in;
    private String name;

    public ClientHandler(Socket socket) {
        this.socket = socket;
    }

    public void run() {
        try {
            in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            out = new PrintWriter(socket.getOutputStream(), true);

            out.println("Enter your name:");
            name = in.readLine();
            ChatServer.broadcast(name + " joined the chat.", this);

            String message;
            while ((message = in.readLine()) != null) {
                ChatServer.broadcast(name + ": " + message, this);
            }
```

```
        } catch (IOException e) {
            System.out.println("User disconnected.");
        } finally {
            try {
                socket.close();
            } catch (IOException e) {
                //
            }
            ChatServer.removeClient(this);
            ChatServer.broadcast(name + " left the chat.", this);
        }
    }

    public void sendMessage(String message) {
        out.println(message);
    }
}
```

🖥️ 2. ChatClient.java (Client-Side Chat)

```
import java.io.*;
import java.net.*;

public class ChatClient {

    public static void main(String[] args) {
        try (Socket socket = new Socket("localhost", 1234)) {

            BufferedReader input = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            PrintWriter output = new
PrintWriter(socket.getOutputStream(), true);

            BufferedReader keyboard = new BufferedReader(new
InputStreamReader(System.in));

            // Thread to read server messages
            new Thread(() -> {
                String msgFromServer;
                try {
                    while ((msgFromServer = input.readLine()) != null)
{
                        System.out.println(msgFromServer);
                    }
                } catch (IOException e) {
                    System.out.println("Connection closed.");
                }
            }).start();
```

```
            // Main thread to send messages
            String msgToServer;
            while ((msgToServer = keyboard.readLine()) != null) {
                output.println(msgToServer);
            }

        } catch (IOException e) {
            System.out.println("Unable to connect to server.");
        }
    }
}
```

▶ How to Run

1. **Compile both files**:

   ```
   javac ChatServer.java ChatClient.java
   ```

2. **Start the server**:

   ```
   java ChatServer
   ```

3. **Start multiple clients** (in separate terminals or machines):

   ```
   java ChatClient
   ```

Sample Output (Console)

Server:

```
Server started. Waiting for clients...
New user connected: Socket[addr=/127.0.0.1,port=55555,...]
```

Client 1:

```
Enter your name:
Alice
Alice joined the chat.
Bob: Hello Alice!
```

Client 2:

```
Enter your name:
Bob
Bob joined the chat.
```

Alice: Hi Bob!