# Hiring Task: Building a Real-Time Chat Application using Sockets

Role: Full Stack Developer at Kuvaka Tech

## Objective:

Develop a simple chat server and client that communicate in real-time over the network using sockets. The application should support multiple client connections to the server simultaneously, allowing users to send and receive messages in a shared chatroom environment.

## Requirements:

### 1. Server:

- **Initialization:** The server should start listening on a specified port for incoming connections.
- **Connection Handling:** It should be able to handle multiple client connections concurrently, using either threading or asynchronous I/O.
- **Broadcast Messages:** When a message is received from one client, it should broadcast the message to all connected clients, excluding the sender.

### 2. Client:

- **Connection:** The client should connect to the server using its IP address and port.
- **User Interface:** Implement a simple text-based interface where users can type messages and see messages from others in real-time.
- **Message Sending:** Allow the user to send messages to the server, which will then be broadcasted to other clients.
- **Receiving Messages:** Display messages from other users in real-time.

## Technical Constraints:

- **Language:** You should use Node.js.
- **Concurrency:** For handling multiple clients, use threads (or async/await patterns if supported by your chosen language).

- **Socket API:** Use the standard socket library provided by your chosen language for network communication.
- **No External Libraries:** Do not use any external libraries or frameworks for the chat logic. Standard libraries for threading or asynchronous operations are allowed.

## Deliverables:

### 1. Source Code:

- Server and client application source code with clear comments.
- Organize the code into files/classes/modules as appropriate for your chosen language.

### 2. Documentation:

- A README file that includes:
  - Instructions on how to run the server and client applications.
  - A brief description of the application's architecture and how concurrency is handled.
  - Any assumptions or design choices you made and why.

## Evaluation Criteria:

- **Functionality:** The server and client meet the specified requirements.
- **Concurrency Handling:** Effective use of concurrency or asynchronous patterns to handle multiple clients.
- **Code Quality:** Clarity, readability, and organization of the code.
- **Robustness:** Handling edge cases and potential errors, especially in network communication and concurrency.
- **Documentation:** Completeness and clarity of the documentation and instructions.

## Submission Guidelines:

- Submit the compressed archive through the email hr@kuvaka.io
- attach all the files/documents required.
- clearly mention the subject as "Full Stack Developer Submission - [Your Full Name]".

- Share the deployment link with the email.
- Share the repo link with the email.
- Make sure your submission is accessible for evaluation.

**Code Repository:**

- Create a public repository on a version control platform (GitHub, Bitbucket). Share the link in the submission mail. Include all your source code, organized into files/classes/modules, within this repository.

**Deployment:**

- Share the deployment link with your submission mail.

**README Documentation:**

- Craft a detailed README file within your repository.
- Cover the following aspects:
  - Instructions on how to run the server and client applications.
  - A brief description of the application's architecture and how concurrency is handled.
  - Any assumptions or design choices you made and the reasoning behind them.
  - Guidance on accessing the chat application once deployed.

## Deadline:

Submit your assignment within 72 Hours of receiving this mail message.