# Client- Server Architecture

In client-server architecture, servers and client interact to provide the services and share resources over network. In chatroom client interacts with central server, which handle the message routing and storage.

We believe this is a potential system architecture for our project for the following reasons. We will also briefly explain how we would implement

1. **Centralized Control and Network Communication:**

Because our project will need to support multiple connection to the same chatroom, it is beneficial to use server-client architecture to separate clients and server, at the same time, effectively pushing real-time updates to clients. The server and clients will be connected to network, either it can be global or local. Server will have an IP address or domain to which client can request to connect. Furthermore:

* *Server Setup*: the server will setup with required resources and functionality that will provide client for specific service. The service can be like viewing the message, creating or deleting the chatroom etc.
* *Listening to Request*: the server will be programmed to hear to client requests in a particular port. To achieve this client will send request to server IP with port address. Clients initiate communication by sending requests to server. These requests typically include particular action or command that describes what client wants the server to do.
* *Request Processing*: on receiving the request from client, the server will generate which will typically include of response data and result of requested action.
* Response Handling: on receiving the response from server client will perform action based on purpose.
* *Statelessness*: client- server communication is usually the statelessness i.e each request from client to server will be independent. This is will be import feature for chatroom as each service will be independent from each other like creating new chatroom, sending message, opening embedded IDE, etc.

1. **Scalability**

Server-client architecture is usually used to handle large number of clients simultaneously. Load balancing and clustering techniques can be applied to ensure flawless communication with large number of clients and server.

1. **Security**

Security measures, such as authentication, authorization, and encryption, are essential components of client-server systems to protect data and ensure that only authorized clients can access the server's resources. This will be an important feature in chatroom to protect users’ information.

# Client- Server Architecture Diagram

In this diagram, there are 2 main components: clients and application server. Client is the front-end, responsible for displays and UI and application server is the backend, comprising different layers that control the functionality, such as database, model, controller, etc.

When a client makes a request of any kind, it will be passed to host server then to application server. The application server will then pass this to a specific server that are responsible for that request. For example, when a new user is registered, it will be passed to user server to handle that request. Unlike MVC architecture, client-server does not represent the inner structure of backend well.