# COMPUTER NETWORKS

# MINI PROJECT

# SIM(SECURE INSTANT MESSENGER)

## TEAM MEMBERS

| | |
|---|---|
| DARAM GUNASHEKAR | RA2011026010208 |
| VUPPALA KAMALESH KUMAR | RA2011026010198 |
| HARIKESH | RA2011026010193 |
| G.SUBRAHMANYAM | RA2011026010188 |

## PROJECT SCOPE:

- Our idea regarding mini project is building a chatting website using AWS and we named it as SIM (Secure Instant Messenger). In this project we aim to execute a simple chat server for a LAN where all the clients can send messages.
- Also, we use the concept of multi-threading to create processes for each client. Here, we set up a socket on each end and allow a client to interact with other clients via the server. The socket on the server side associates itself with some hardware port on the server-side.
- Any client that has a socket associated with the same port can communicate with the server socket.

## OBJECTIVE:

- The main objective of the project is to create a secure website for instant messaging (that include computer networking concept, threading) and understanding the functionalities of aws services.

**FUNCTIONALITIES:**

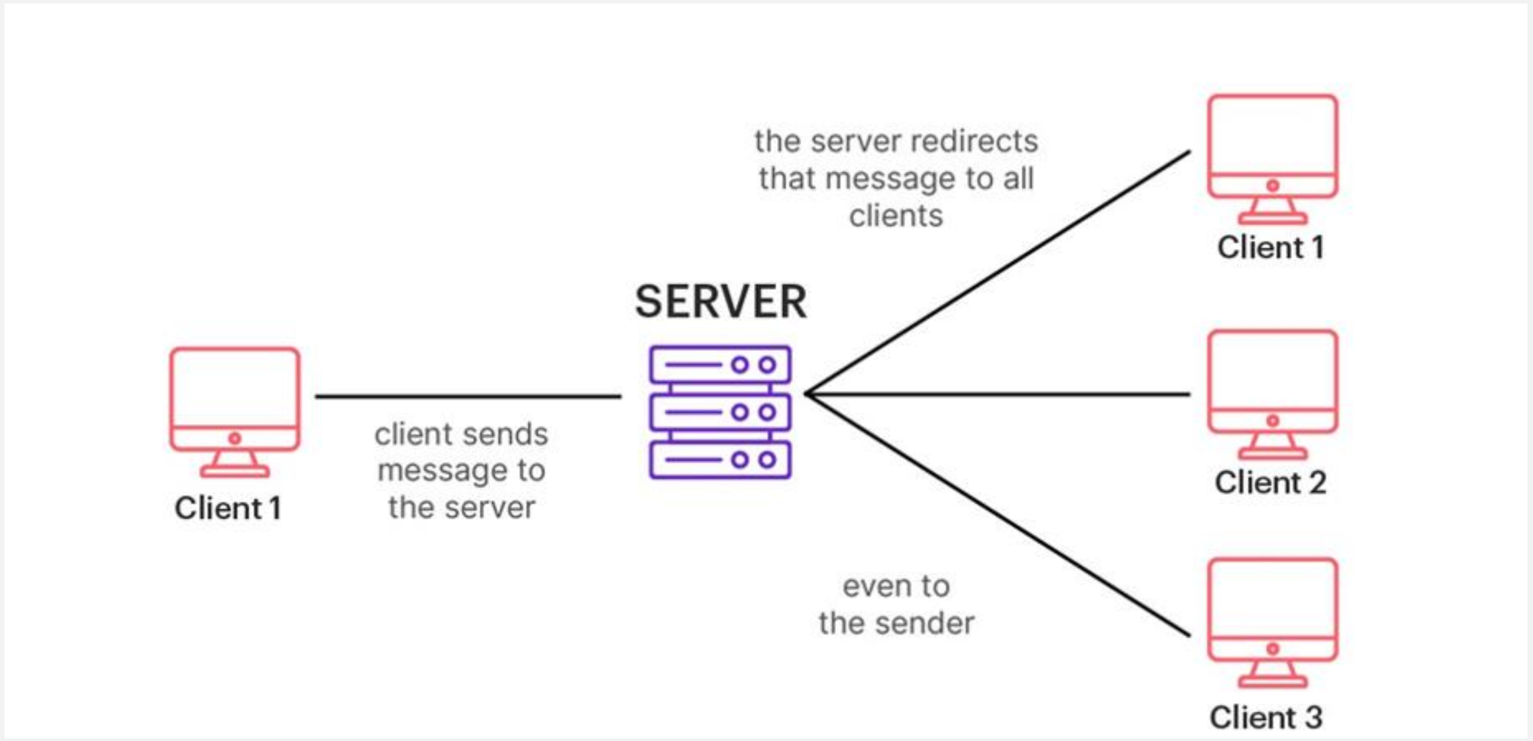
- This project gives users the ability to establish a chat server to communicate with each other over a local area network. Using a laptop as a server node, and the laptop's private IP address as a server's IP address, we can set up a chat server you can use on your local area network. A colleges' LAN which is further divided into three buildings is described in the sample network rubric. Each building has a set of IP addresses that are assigned to all 50 users, and any of the nodes in the set can act as a server, and the remaining nodes can connect to the server.

## REQUIREMENTS:

- Processor: Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz, 1800 Mhz, 4 Core(s
- RAM: 4 GB
- Hard disk capacity: 40 GB
- Network Interface card: 32bit PCI/ISA Ethernet or MODEM
- Operating systems:  Microsoft Windows 10 Home Single Language
- Communication protocol: TCP/IP

## DESIGN:

- The server-side script will try to establish a socket and bind it to an ip deal with and port specific by way of the user. The script will then live open and obtain connection requests and could append respective socket items to a listing to preserve music of energetic connections. **Each time a consumer connects, a separate thread could be created for that user. In every thread, the server awaits a message and sends that message to other customers currently on the chat**. If the server encounters an error while trying to acquire a message from a selected thread, it will go out that thread.
- The customer-side script will virtually try and get admission to the server socket created at the required ip address and port. As soon as it connects, it will continuously take a look at as to whether or not the input comes from the server or from the consumer, and accordingly redirects output. If the enter is from the server, it displays the message on the terminal. If the input is from the consumer, it sends the message that the consumer enters to the server for it to be broadcasted to other users.


the server redirects that message to all clients
Client 1
SERVER
Client 1
client sends message to the server
Client 2
even to the sender
Client 3

## THANK YOU