# 

**OS PROJECT**

PROJECT NAME:

**MULTI - USER CHAT SYSTEM**

Team 6:

22wh1a6606 – G. Revathi

22wh1a6616 – N.Divya Saahithya

22wh1a6626 – R. Ashritha

22wh1a6636 – B. Hemanya

22wh1a6646 – B. Soumya

22wh1a6656 – A. Gayathri

22wh5a6602 – P. Kavya

**PROBLEM STATEMENT:**

Implement a multi-user chat system using C that consists of a server and multiple clients.

**ABSTRACT:**

It is a simple implementation of simple client server chat system where once clients are connected to server, then they can communicate with each other. The client first needs to connect with the server and can then issue two commands -

1. GET - This command fetches the list of clients that are currently connected to server.

2. SEND (client number) (message) - SEND followed by client number which can be used to send the message to particular to that client number.

Key features of the system include user authentication, secure data transmission, dynamic user management, and the ability to handle concurrent connections. The implementation incorporates socket programming, threading, and robust error handling to ensure a stable and responsive chat environment.

The project addresses the challenges of building a distributed chat system, emphasizing a modular and extensible architecture. Additionally, considerations are given to user experience, ensuring an intuitive interface for clients and effective administration tools for the server.

By developing this multi-user chat system, the project aims to provide a practical and educational resource for individuals interested in network programming, distributed systems, and concurrent communication applications using the C programming language.

**PROGRAM:**

CHAT CLIENT:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define PORT 55555

int main() {

int client\_socket;

struct sockaddr\_in server\_addr;

char message[1024];

// Create socket

if ((client\_socket = socket(AF\_INET, SOCK\_STREAM, 0)) == -1) {

perror("Socket creation failed");

exit(EXIT\_FAILURE);

}

// Configure server address

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_port = htons(PORT);

server\_addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

// Connect to the server

if (connect(client\_socket, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) == -1) {

perror("Connection failed");

exit(EXIT\_FAILURE);

}

// Receive welcome message

recv(client\_socket, message, sizeof(message), 0);

printf("%s\n", message);

// Enter the chat room

while (1) {

printf("Enter a message: ");

fgets(message, sizeof(message), stdin);

// Send the message to the server

send(client\_socket, message, strlen(message), 0);

}

// Close the socket

close(client\_socket);

    return 0;

}

CHAT SERVER:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define PORT 55555

#define MAX\_CLIENTS 10

int clients[MAX\_CLIENTS];

int num\_clients = 0;

void broadcast(int sender, char \*message, int length) {

for (int i = 0; i < num\_clients; i++) {

if (i != sender) {

send(clients[i], message, length, 0);

}

}

}

void handle\_client(int client\_socket, int index) {

char buffer[1024];

while (1) {

int recv\_size = recv(client\_socket, buffer, sizeof(buffer), 0);

if (recv\_size <= 0) {

close(client\_socket);

clients[index] = -1;

break;

}

buffer[recv\_size] = '\0';

broadcast(index, buffer, recv\_size);

}

}

int main() {

int server\_socket, client\_socket, addr\_len;

struct sockaddr\_in server\_addr, client\_addr;

// Create socket

if ((server\_socket = socket(AF\_INET, SOCK\_STREAM, 0)) == -1) {

perror("Socket creation failed");

exit(EXIT\_FAILURE);

}

// Configure server address

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_port = htons(PORT);

server\_addr.sin\_addr.s\_addr = INADDR\_ANY;

// Bind the socket

if (bind(server\_socket, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) == -1) {

perror("Bind failed");

exit(EXIT\_FAILURE);

}

// Listen for incoming connections

if (listen(server\_socket, 5) == -1) {

perror("Listen failed");

exit(EXIT\_FAILURE);

}

printf("Server is listening on port %d...\n", PORT);

while (1) {

// Accept connection from a client

addr\_len = sizeof(client\_addr);

client\_socket = accept(server\_socket, (struct sockaddr \*)&client\_addr, &addr\_len);

if (client\_socket == -1) {

perror("Accept failed");

continue;

}

// Add the client to the array

clients[num\_clients++] = client\_socket;

// Create a new thread to handle the client

int index = num\_clients - 1;

printf("Connection established with client %d\n", index);

// Handle the client in a separate thread

if (fork () == 0) {

handle\_client(client\_socket, index);

exit (0);

}

}

    return 0;

}

**OUTPUT:**

