**Instant Chatters**

**Low Level Design**

Introduction

**1 Purpose:**

The purpose of this project is to implement a socket programming-based chat application that will allow creating a multiple client to chat in private conversations.

This project is to create a chat application with the server and clients to enable the clients to chat with many other clients. This project can play an important role in organization field where employees can be connected through internet.

**1.1 Intended Audience:**

There is no such specific audience, it could be a student or employee or an organization also.

**1.2 Acronyms/Abbreviations:**

| CLIENT | USER |
| --- | --- |
| SERVER | TO ESTABLISH CONNECTION BETWEEN  CLIENTS |
| TCP | TRANSMISSION CONTROL PROTOCOL |

**1.3 Project Purpose:**

The purpose of this project is to implement a socket programming-based chat application that will allow creating a multiple client to chat in private conversations.

**1.4 Key Project Objectives:**

1. Allow user registration

2. Allow users to login

3. Validates username and password

4. Message exchange

5. Sending and receiving messages

6.Active User list & Left users List

7.Allow user to sign out from application

**1.5 Project Scope and Limitation:**

New users can be able to register and register users can be able to login And message exchange between two clients takes place.

**1.5.1 In Scope:**

It provides a general architecture for chat applications, and anyone or organization can use it as the basis for providing instant messaging capabilities. The application is written in an procedure-oriented language called C. The application is divided into two parts, server and client, the server in this application will remain on the local computer. Clients do not need to install any software on their machine. Only network access is needed for communicating with each other.

**1.6 Functional Overview:**

Following header files are included in the program:

a.#include <stdio.h> //INCLUDES INPUT OUTPUT RELATED FUNCTIONS

b.#include <sys/socket.h> //INCLUDES MACROS AND PREDEFINED TYPES SOCKET VARIABLES

c.#include <netinet/in.h>

d.#include <string.h> //INCLUDES ALL FUNCTIONS OF STRING HANDLING

e.#include<arpa/inet.h>//DEFINITIONS FOR INTERNET ADDRESSES

f.#include<stdlib.h> //INCLUDES FUNCTIONS INVOLVING MEMORY ALLOCATIONS<UTILITY FUNCTIONS FOR TYPE CONVERSIONS

The following functions are included in the program:

a. void registration();

Registration() function takes the input from the user such as name, user ID, email, Phone number and password.

And it passes the data given by the user and store it as file in server.

b. void login( );

It asks the users to enter the user id and password for validation.

If the user given data matches the data which is stored in the file in server, the user login is successful.

And it checks if the data given by user is present in the file which has been stored in the server or not. If not by means of one data match and other not, then the users login will fail

.

And the data doesn't find in the file stored in server, then it shows user not found.

c. void compare\_strings(int a,int b);

This function is used as handler for ctrl + c with the help of signal.

With this function, if the user types “bye”, the connection between the clients terminates and the data is stored.

d. void take\_input(char ch[50]);

This function will take the input from the user for registration and login

e. char generate\_username(char email[50],char username[50]);

This function helps the users generate his/ her username with the help of email they have entered.

f. void recv\_message(int client\_des);

Void recv\_message(int client\_des) is used to display the messages to the users from other users.

It gives the details of the sender, and it shows the mesages.

g. void send\_message(int client\_des);

Void send\_message(int client\_des) is used to send the messages to the users from other users.

It gives the details of the receiver, and it shows the messages.

This function ends the connection between the clients.

2 Design Overview:

Instant Chatters comprises of the following modules:

| Name of the Module | Chat between clients |
| --- | --- |
| Handled by | Manohar K |
| Description | Sending messages- from client  Receive message - from another user  Sending message- from user(server side)  Receive message from user(server side) and send it to the specified client. |

| Name of the Module | User registration and login |
| --- | --- |
| Handled by | Jahnavi K |
| Description | User registration: new user details store in file.  Login: search the user given input in user details file and allow login if details match.  User validation : whether the user exists in database or not .Compare the entered username and password with registered username and password  Failed Login: It will display as failed login when user enter the invalid password.The system displays the error message "invalid credentials" and it will route back to login option.  Signout : When the user wants to come out of the application it will display as Bye Bye and terminate. |

| Name of the Module | Server Side Read / Send Messages |
| --- | --- |
| Handled by | Anurag M |
| Description | User1 and User2 will start chatting the server will read the messages from both user1 and user 2 and shift the buffer between both the users and exchange of message takes place. |

**2.1 Design Objectives:**

Instant chatting between clients, there are two objectives namely primary and secondary.

Primary:

Establishing connection between clients with the help of server using TCP protocol. If a client is a new user for instant chatter, he should be able to register and login and if a client is an existing user, he is able to login with a valid username and password.

Secondary:

The clients who are logged in, can be able to chat with the other users .

**2.2 Design Alternative:**

**2.2.1 User Interface Paradigms:**

The instant chatter gives access to users to chat with another user who logged in. The details of the user are stored as a single file. User is given an interface to register a new user, interface for login and exit as well. After login the user gets the interface to start the chat and logout.

**2.2.2 Error Detection / Exceptional Handling:**

New users should register before login or else it displays the no user found. Registered users have to login with valid credentials. otherwise, they will get invalid username or password. After login, the user is able to start a chat with only active users.

**2.2.3 Performance:**

The system will work on the client terminal. The performance depends on the hardware component of the user’s system.

**2.2.4 Maintenance:**

Very little maintenance should be required for this setup. An initial configuration

will be the only system required interaction after system is put together. The

only other user maintenance would be any changes to settings after setup, and any specified special cases where user settings or history need to be changed.

Physical maintenance on the system’s parts may be required, and would result

in temporary loss of data or Internet. Upgrades of hardware and software should have little effect on this project but may result in downtime.

**3 Environment Description:**

**3.1 Time Zone Support:** IST- Kolkata

**3.2** **Language Support:** English

**3.3 User Desktop Requirements:**

a. 64-bit processor, 1 GHz or faster

b. At least 2 GB free hard drive space

c. At least 1 GB RAM

**3.4 Server-Side Requirements:**

a. 64-bit processor, 1 GHz or faster

b. At least 1 GB free hard drive space

c. At least 1GB RAM

**3.4.1 Deployment Considerations:**

a. Easy setup: no session storage daemon, use tmpfs and memory caching to enhance performance.

b. Local storage is used

c. No network latency to consider

d. To scale buys a bigger CPU, more memory, larger hard drive, or additional hardware

**3.4.2 Application Server Disk Space:**

a. No such disk space is required as the program is fully functional on online IDE(s) as well. The Local Operating System is required and one text file to store the records of processes.

**3.4.3 Database Server Disk Space:**

a. No such disk space is required as the program is fully functional on online IDE(s) as well. The Local Operating System is required and one text file to store the records of processes.

**3.4.4 Integration Requirements:**

1. Language: System C

2. Tools: splint, Valgrind, Makefile ,git ,Strace

3. Complier: gcc

4. Linux Environment

**3.4.5 Jobs:**

We can establish connections between clients who are connected to the server. and allow the users to exchange messages .

**3.4.6 Network:** End to End

**3.5 Configuration:**

3.5.1Operating System: Linux environment

**4. Data Structures**:

A structure is a key word that create user defined data type in C/C++. A structure creates a data type that can be used to group items of possibly different types into a single type.Here we used structures to store the Login and Registration details of the user.‘struct’ keyword is used to create a structure

struct user //structure tag

{

char fullName[50]; //memebers or fields of structure.

char email[50];

char password[50];

char username[50];

char phone[50];

};