High Level Design and Low Level Design

Instant Chatter Application

Index

1. Introductions

The Instant Chatter Application is a system in which the user can register and afterwards login into the system using the valid username and password. After successful login attempt , the user will able to view rest of the users that are also logged in at that particular time. Once the user is registered , their data i.e username and password is stored on the server in a structured format.

1.1 Intended Audience

The target audience set for this project is can be identified as an office management or any other organization who are willing to share their data and messages among their registered users.

1.2 Project Purpose

The Instant Chatter Application is a project that helps us in understanding the concept functions, data structure and system programming architecture. It can create a record of the data of the users that have successfully registered into the system. Their data can be efficiently used to validate them when they will try to login again in the future.

1.3 Key Project Objectives

a) Allow the user to register on the system

b) Allow the user to login after registration

c) Validating the credentials of already registered users.

d) Display the list of currently active users in the server.

e) Allow users to send and receive messages among themselves.

f) Sign out the user

1.4 Project Scope

The project aims to create and develop a server to client chat application. While registration it ask for the username and password from the user along with other valuable information like name etc. It stores all this data into the server for

validation part later on. The messages can very efficiently be transferred among the users into this system. This function also provide the function of searching a chat done earlier. Chat backup is done on the server and user can very easily search the chat done by them.

2. Design Overview

2.1 Design Objectives

- **Instant Chatter application has the following modules on the user/client side.**

Name of the Module	New User
Handled by	
Description	The user gets himself registered into the system

Name of the Module	Registered user Login
Handled by	
Description	User logs in after registration

Name of the Module	Failed to validate
Handled by	
Description	User is unable to provide correct credentials

Name of the Module	Sending and Receiving messages
Handled by	
Description	Users are able to send and receive message

Name of the Module	Search Chat
Handled by	
Description	Users are able to search their existing chat.

Name of the Module	Sign out
Handled by	
Description	Users are able to sign out after giving specific commands.

- **Instant chatter application has the following function running on the server side .**

Name of the Module	User Registration
Handled by	
Description	User credentials are store into the server in defined data types

Name of the Module	User Validation
Handled by	
Description	User is validated and logged in based on the correctness of his provide credentials

Name of the Module	Active User List
Handled by	
Description	List of users currently logged into the server is displayed

Name of the Module	Message exchanges
Handled by	
Description	Receive message from the sender and broadcast it to other connected clients

2.2 Design Alternatives

We have used linked list and struct data types to store and validate the credentials of the users. Dynamic memory allocation is carried out. The data is send-over the client server network through the sockets system call function.

2.3 User Interface paradigm

The instant chatter system gives the users an option to log in and send messages to the list of users that are currently online on the server.

2.4 Validations

- Clients should strictly give the port number into the command line argument.
- Server should be established before running the client side files.
- Name and passwords should be 32 and 5 characters long respectively.

3. System Architecture

An application architecture dictates how an application is structured over various end systems. The application developer designs the application architecture. The predominant architectural paradigms used in modern network applications are the client-server architecture and the peer-to-peer (P2P) architecture.

3.1 Client Server Architecture

A client-server architecture consists of an always-on host, called the server, which responds to requests from many other hosts, called clients. The clients communicate through the server. The server usually has a static IP address, because it is more convenient for the clients. If the IP address constantly changed, it would be difficult to connect to the server. In recent years, the client/server architecture has become commonplace as the mechanism that brings the centralized corporate databases to desktop PCs on a network. In a client/server environment, one or more servers manage the centralized database and clients gain access to the data through the server.

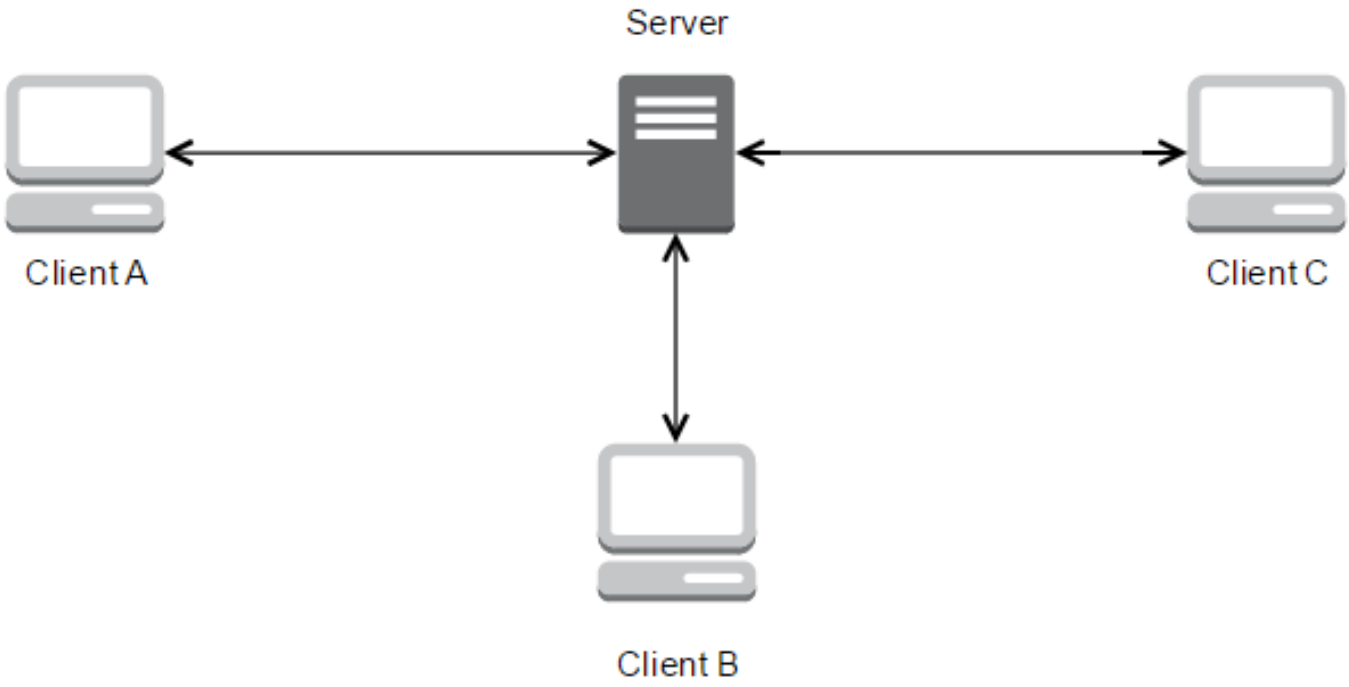


Fig: Typical client server architecture

3.2 Sockets

Sockets are software interfaces through which a process sends messages into, and receives messages from a network. A process is a program that is running within a host. A networked application consists of pairs of processes that communicate by exchanging messages across a network. Messages sent by one process must go through the network to get to another process. The processes can be thought of as houses, and the sockets as their doors.

A socket functions as an interface between the application layer and the transport layer. It is an API between the application and the network.

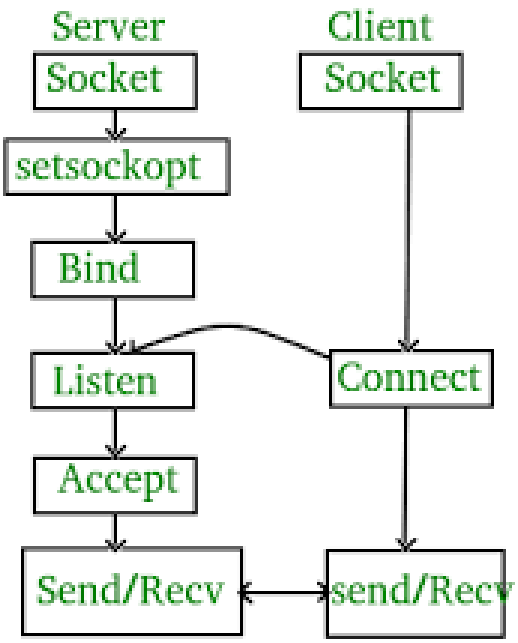


Fig: Server Client Connection through socket

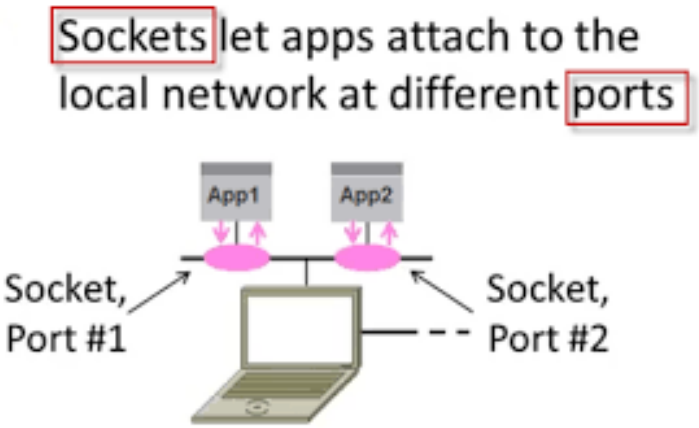


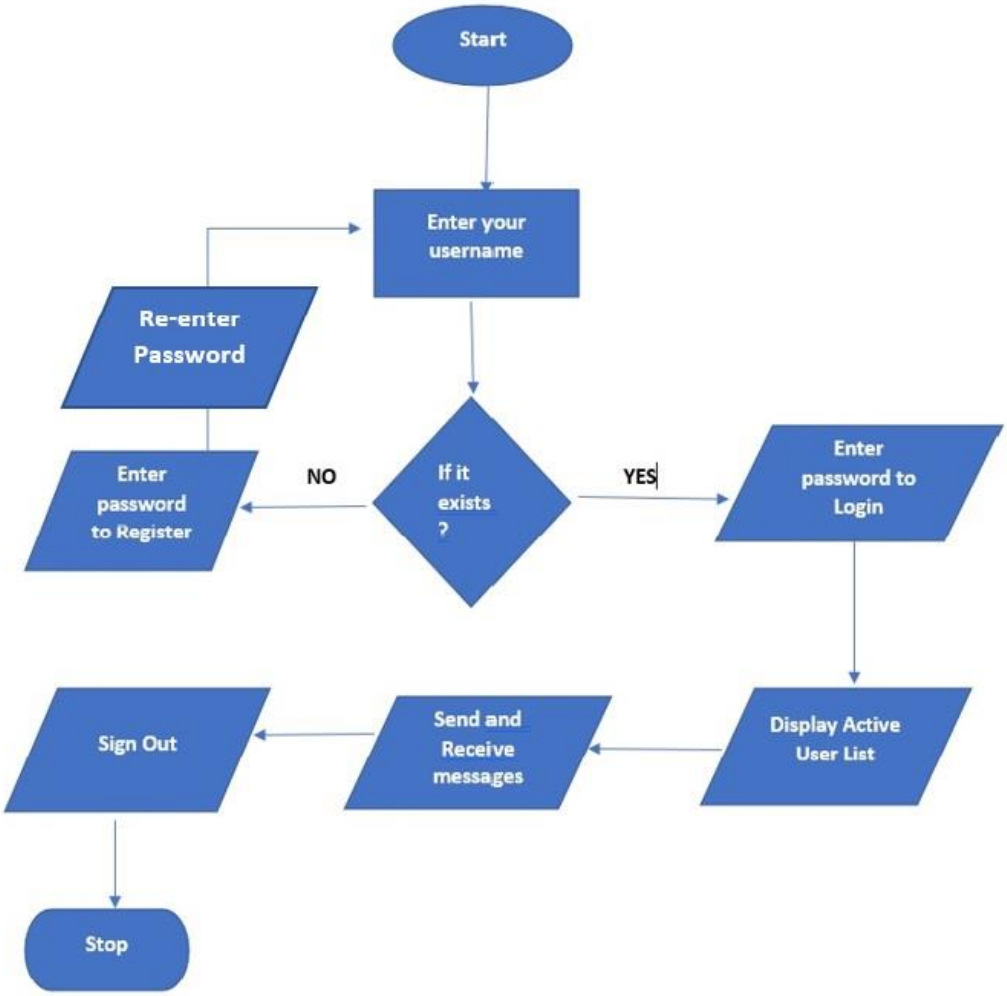
Fig: Image describing the operations of a socket

3.3 TCP

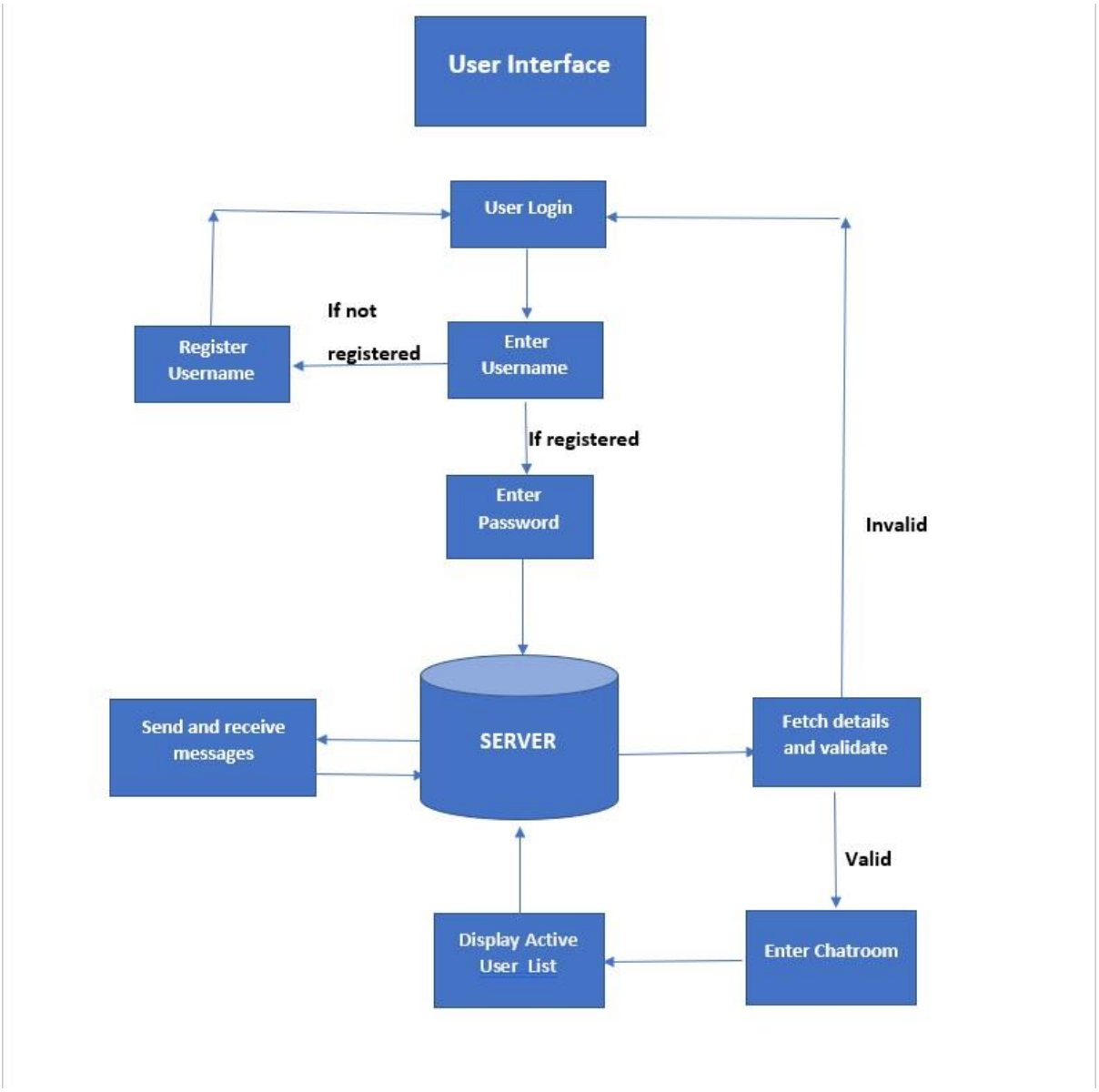
TCP (Transmission Control Protocol) provides reliable data transfer. Data is delivered in the correct order and with no duplicates. TCP includes a congestion-control mechanism, which throttles a sending process when the network between the sender and receiver is congested. TCP requires nontrivial connection state tracking at both ends of the connection. The recipient must acknowledge received data. The sender must resend any unacknowledged data. TCP requires a larger header than UDP because of the additional services it provides.

4. Detailed System Design

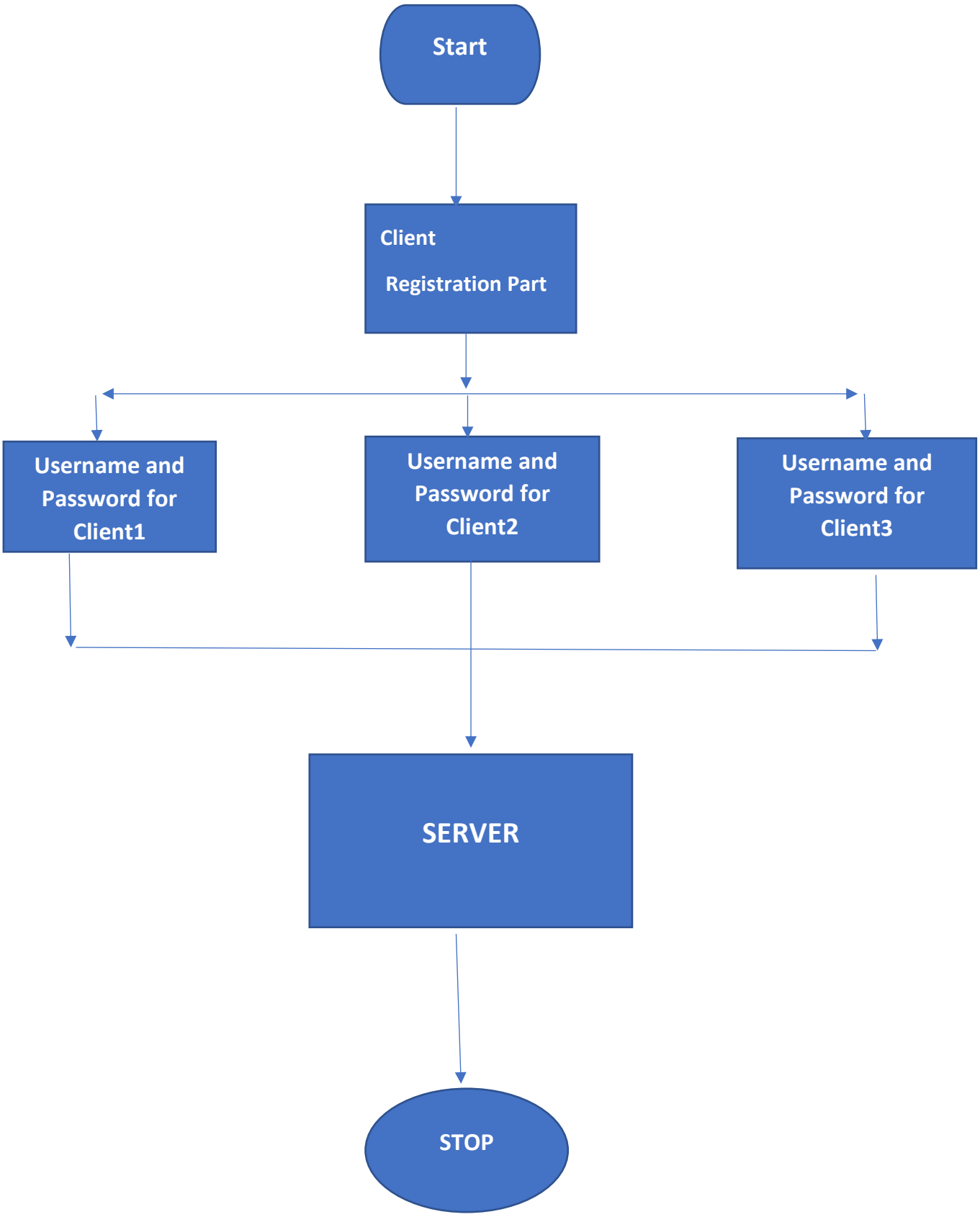
4.1 Flowchart Of the application



4.2 User/Client Interface Detailed Data Flow

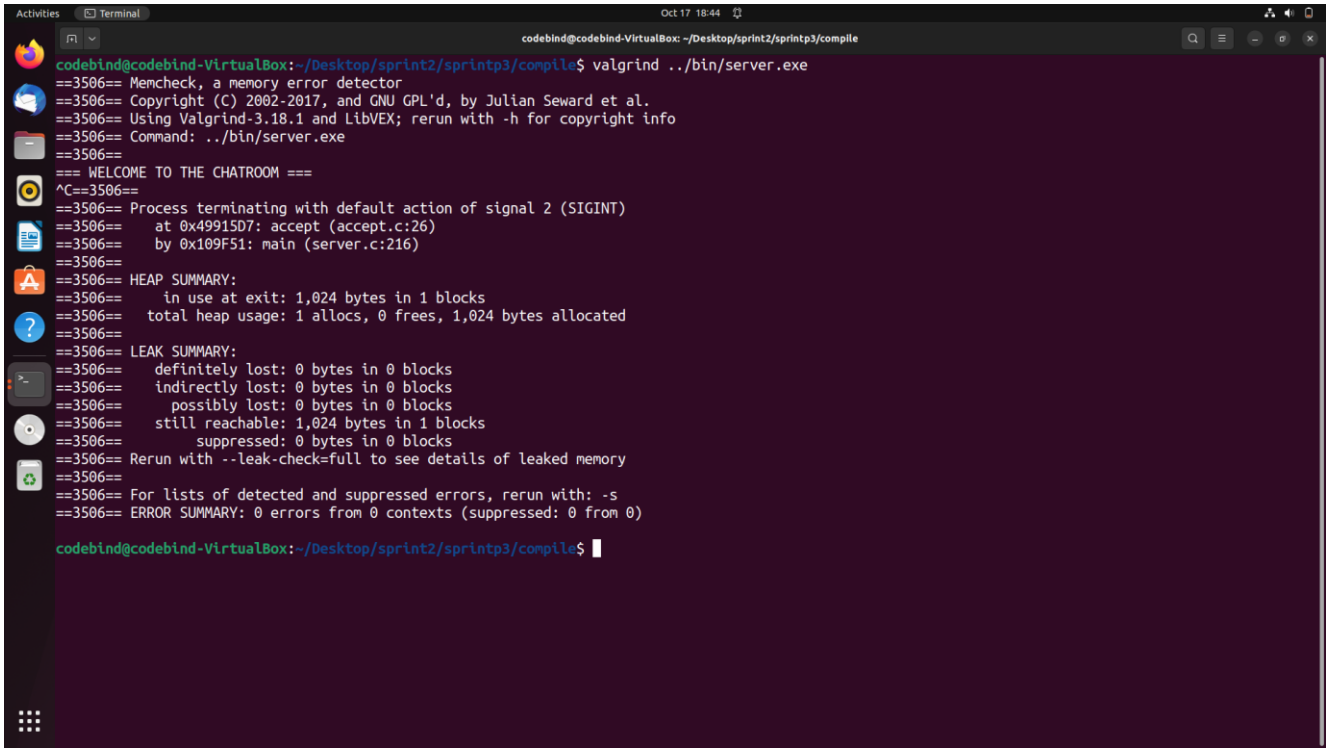


4.3 Flow Chart for User Registration

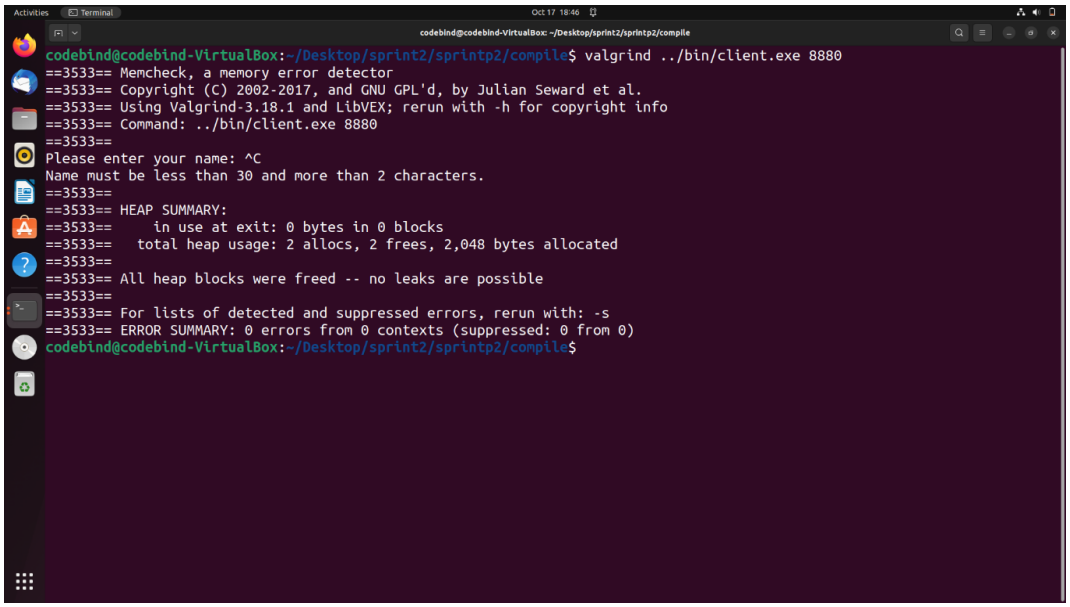


5. TOOLS REPORT

5.1 Valgrind Report (Server File)



5.2 Valgrind Report (Client side)

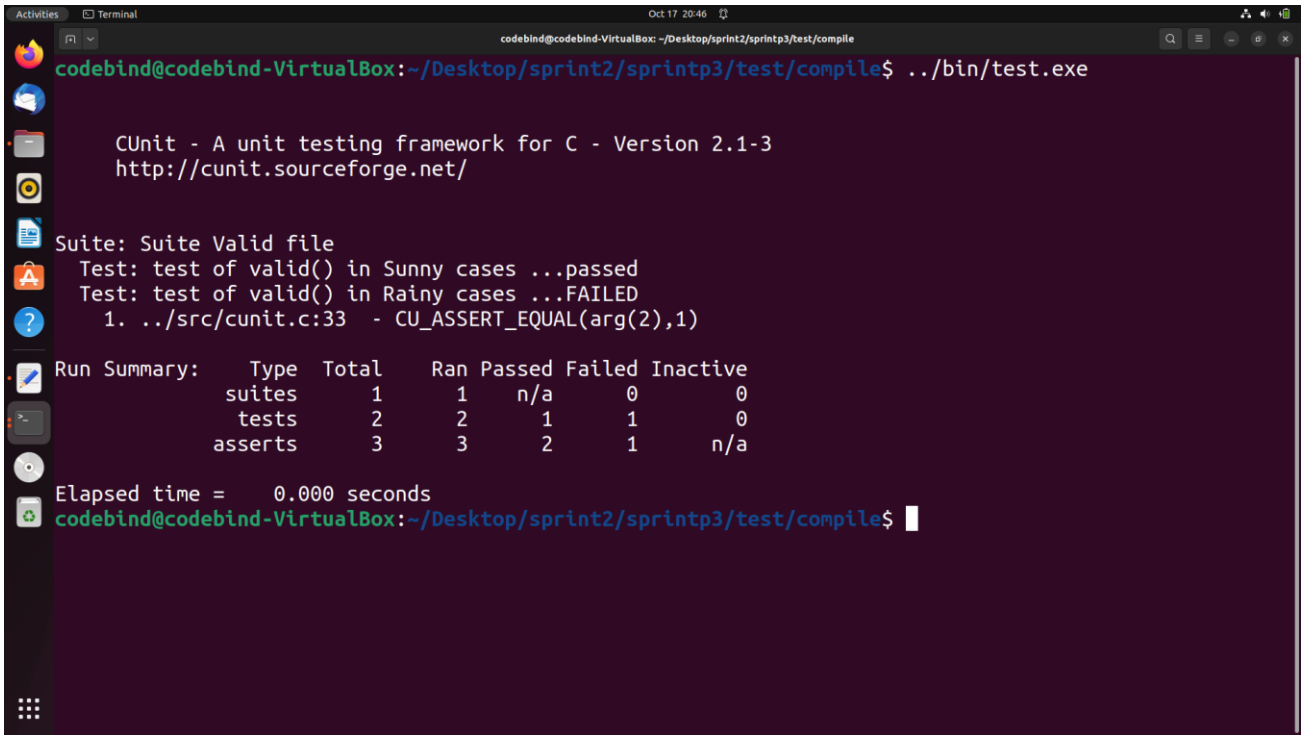


5.3 Splint Report

```
codebind@codebind-VirtualBox:~/Desktop/sprint/sprint2/sprintp3/compile$ cd ..
codebind@codebind-VirtualBox:~/Desktop/sprint/sprint2/sprintp3$ cd src
codebind@codebind-VirtualBox:~/Desktop/sprint/sprint2/sprintp3/src$ splint server.c
Splint 3.1.2 --- 21 Feb 2021

../include/header.h:2: Include file <sys/socket.h> matches the name of a POSIX
    library, but the POSIX library is not being used.  Consider using +posixlib
    or +posixstrictlib to select the POSIX library, or -warnposix to suppress
    this message.
  Header name matches a POSIX header, but the POSIX library is not selected.
  (Use -warnposixheaders to inhibit warning)
../include/header.h:3: Include file <netinet/in.h> matches the name of a POSIX
    library, but the POSIX library is not being used.  Consider using +posixlib
    or +posixstrictlib to select the POSIX library, or -warnposix to suppress
    this message.
../include/header.h:7: Include file <unistd.h> matches the name of a POSIX
    library, but the POSIX library is not being used.  Consider using +posixlib
    or +posixstrictlib to select the POSIX library, or -warnposix to suppress
    this message.
../include/header.h:18:24: No type before declaration name (implicit int type):
                            _Atomic : static <any>
  A variable declaration has no explicit type.  The type is implicitly int.
  (Use -imptype to inhibit warning)
```

6. Testing Reports

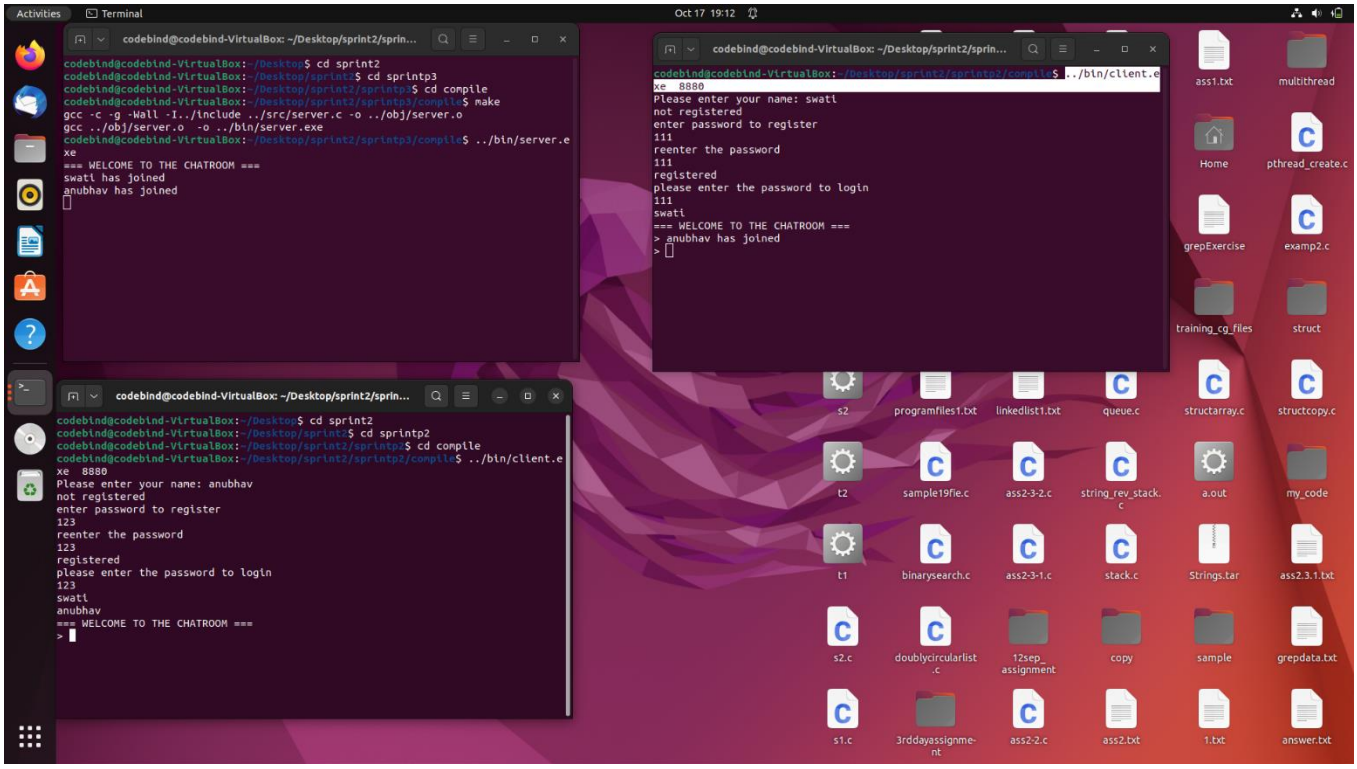
6.1 CUnit Testing Report



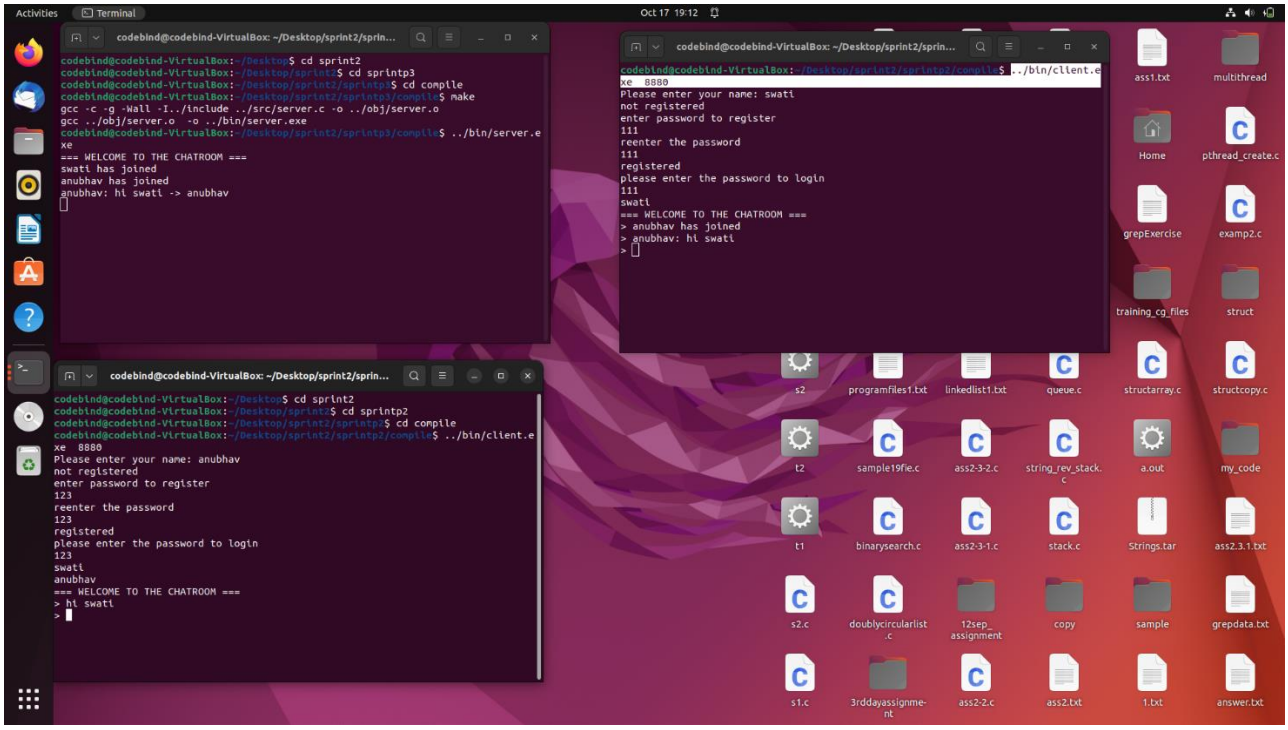
```
codebind@codebind-VirtualBox:~/Desktop/sprint2/sprintp3/test/compile$ ../bin/test.exe


     CUnit - A unit testing framework for C - Version 2.1-3
     http://cunit.sourceforge.net/


Suite: Suite Valid file
  Test: test of valid() in Sunny cases ...passed
  Test: test of valid() in Rainy cases ...FAILED
    1. ../src/cunit.c:33  - CU_ASSERT_EQUAL(arg(2),1)

Run Summary:    Type  Total    Ran Passed Failed Inactive
              suites      1      1    n/a      0        0
               tests      2      2      1      1        0
             asserts      3      3      2      1      n/a

Elapsed time =    0.000 seconds
codebind@codebind-VirtualBox:~/Desktop/sprint2/sprintp3/test/compile$
```

6.2 Integration Testing Reports

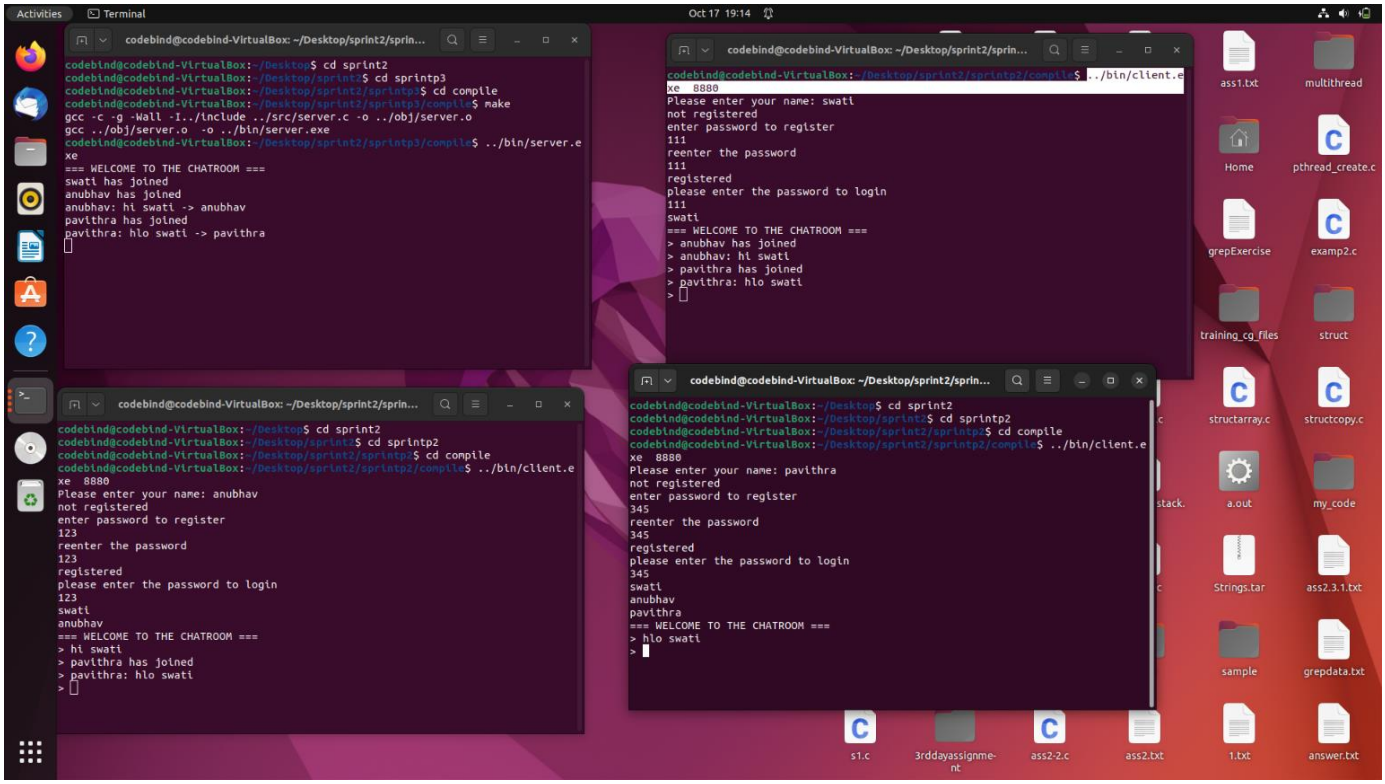
6.2.1 User Registration and Login



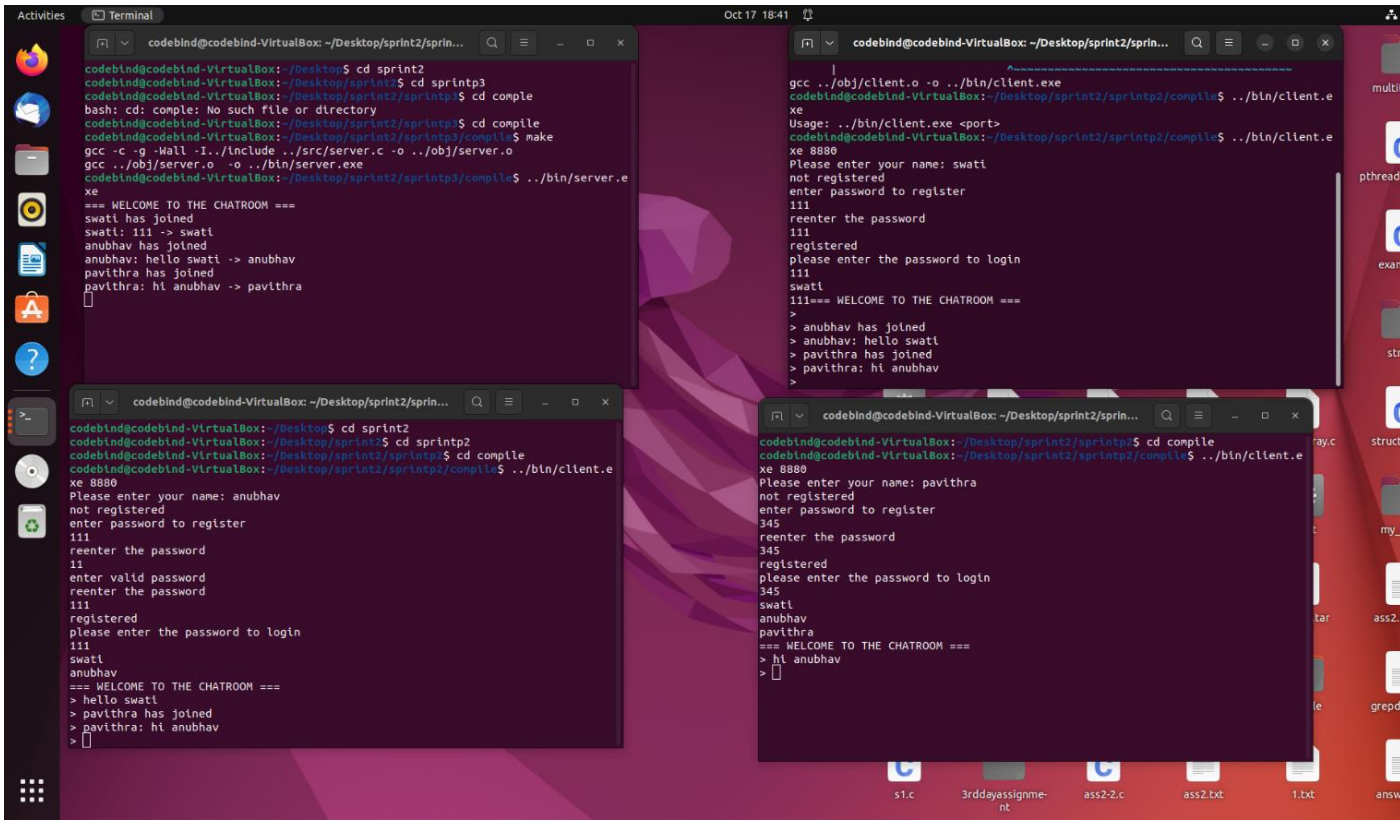
6.2.2 Active User List Display



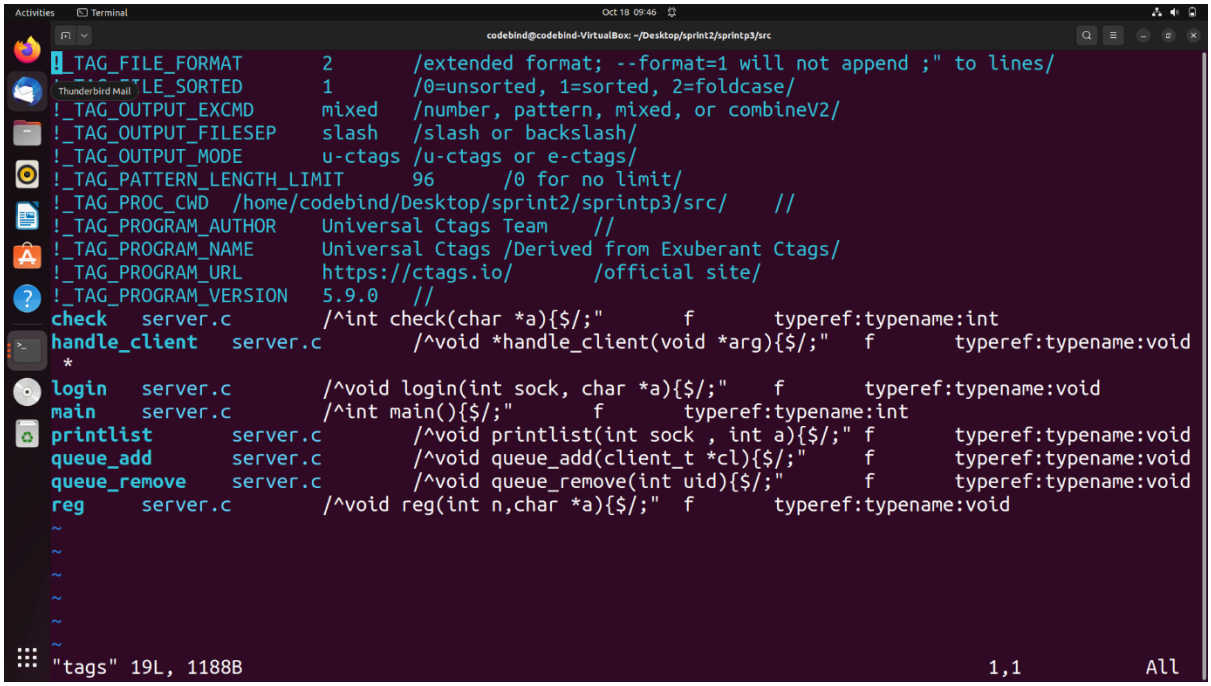
6.2.3 Message Exchanges



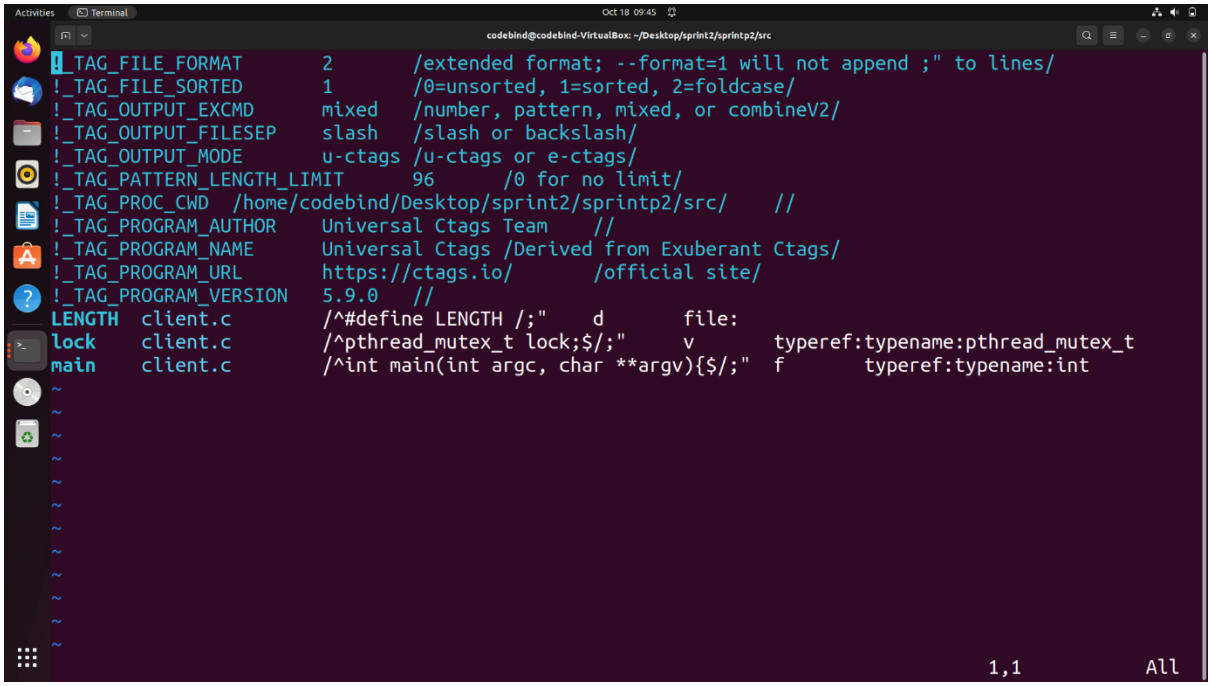
6.2.4 Sending and Receiving Message



6.3 Ctags Report



```
!_TAG_FILE_FORMAT       2       /extended format; --format=1 will not append ;" to lines/
!_TAG_FILE_SORTED       1       /0=unsorted, 1=sorted, 2=foldcase/
!_TAG_OUTPUT_EXCMD      mixed   /number, pattern, mixed, or combineV2/
!_TAG_OUTPUT_FILESEP    slash   /slash or backslash/
!_TAG_OUTPUT_MODE       u-ctags /u-ctags or e-ctags/
!_TAG_PATTERN_LENGTH_LIMIT      96      /0 for no limit/
!_TAG_PROC_CWD  /home/codebind/Desktop/sprint2/sprintp3/src/    //
!_TAG_PROGRAM_AUTHOR    Universal Ctags Team    //
!_TAG_PROGRAM_NAME      Universal Ctags /Derived from Exuberant Ctags/
!_TAG_PROGRAM_URL       https://ctags.io/       /official site/
!_TAG_PROGRAM_VERSION   5.9.0   //
check   server.c        /^int check(char *a){$/;"       f       typeref:typename:int
handle_client   server.c        /^void *handle_client(void *arg){$/;"   f       typeref:typename:void
 *
login   server.c        /^void login(int sock, char *a){$/;"    f       typeref:typename:void
main    server.c        /^int main(){$/;"       f       typeref:typename:int
printlist       server.c        /^void printlist(int sock , int a){$/;" f       typeref:typename:void
queue_add       server.c        /^void queue_add(client_t *cl){$/;"     f       typeref:typename:void
queue_remove    server.c        /^void queue_remove(int uid){$/;"       f       typeref:typename:void
reg     server.c        /^void reg(int n,char *a){$/;"  f       typeref:typename:void
~
~
~
~
~
~
"tags" 19L, 1188B                                                       1,1           All
```



```
!_TAG_FILE_FORMAT       2       /extended format; --format=1 will not append ;" to lines/
!_TAG_FILE_SORTED       1       /0=unsorted, 1=sorted, 2=foldcase/
!_TAG_OUTPUT_EXCMD      mixed   /number, pattern, mixed, or combineV2/
!_TAG_OUTPUT_FILESEP    slash   /slash or backslash/
!_TAG_OUTPUT_MODE       u-ctags /u-ctags or e-ctags/
!_TAG_PATTERN_LENGTH_LIMIT      96      /0 for no limit/
!_TAG_PROC_CWD  /home/codebind/Desktop/sprint2/sprintp2/src/    //
!_TAG_PROGRAM_AUTHOR    Universal Ctags Team    //
!_TAG_PROGRAM_NAME      Universal Ctags /Derived from Exuberant Ctags/
!_TAG_PROGRAM_URL       https://ctags.io/       /official site/
!_TAG_PROGRAM_VERSION   5.9.0   //
LENGTH  client.c        /^#define LENGTH /;"    d       file:
lock    client.c        /^pthread_mutex_t lock;$/;"     v       typeref:typename:pthread_mutex_t
main    client.c        /^int main(int argc, char **argv){$/;"  f       typeref:typename:int
~
~
~
~
~
~
~
~
~
~
~
                                                                        1,1           All
```

7. Pseudocode

7.1 Server side

Main()

Tasks :-

Initialize , create and bind the server

Listen for socket calls from clients side

Output :-

“Welcome to chat room”

“Active User List “

Execution :-

Receive Name from client and check if same name previously exists.

Create a structured list for the credentials

Validate the username and from passwords from the clients during login

Send Successful login or Failure Login messages Accordingly

Allow Message exchanges between Users through connected sockets.

7.2 Clients Side

Main()

Output :-

“Enter your username”

“Enter your password “ if username exists

“successful login “ or “Failure Login “ as received from the server.

“Active user List”

Input :-

1. Username

2. Password

3. Messages to be sent

Tasks:-

Connnect to server socket

Send And Receive Messages Through server using Sockets