



## **High Level Design and Low Level Design**

### **Instant Chatter Application**

## **Index**

### **1.Introduction**

- 1.1 Intendend Audience
- 1.2 Project Purpose
- 1.3 Key Project Objective
- 1.4 Project Scope

### **2. Design Overview**

- 1.1 Design Objective
- 1.2 Design Alternative
- 1.3 User Interface
- 1.4 Validations

### **3. System Architecture**

- 3.1 Client Server Architecture
- 3.2 Sockets
- 3.3 TCP

### **4. Detailed System Design**

- 4.1 Flowchart of the application
- 4.2 User/Client Interface Detailed Data Flow
- 4.3 Flowchart for user Registration

### **5. Tools Report**

- 5.1 Valgrind Report (server side)
- 5.2 Valgrind Report (client Side)
- 5.3 Splint Report

### **6. Testing Reports**

- 6.1 Cunit Testing Report
- 6.2 Integration Testing Reports
  - 6.2.1 User Registration and Validation
  - 6.2.2 Active User List
  - 6.2.3 Message Exchanges
  - 6.2.4 Send and Receive Messages
- 6.3 Ctags Report

### **7. Pseudocode**

- 7.1 Server side

## **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 be 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.

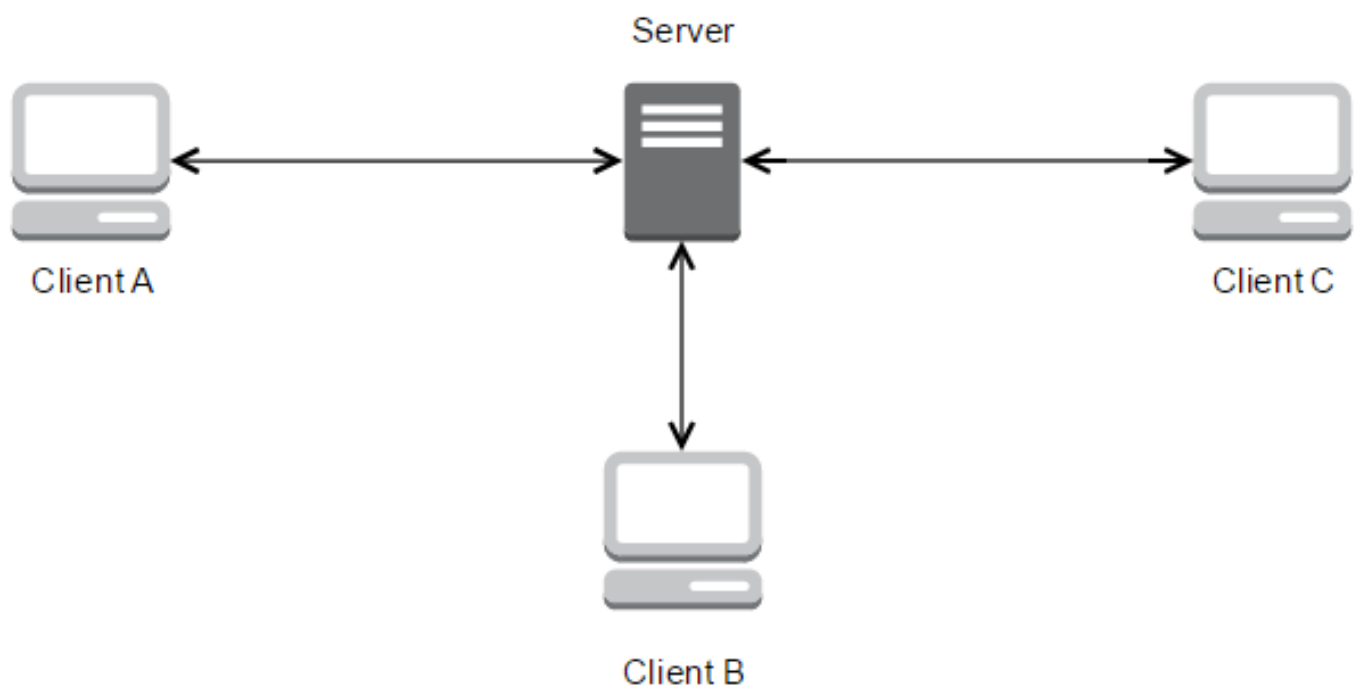


Fig: Typical client server architecture

### 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.

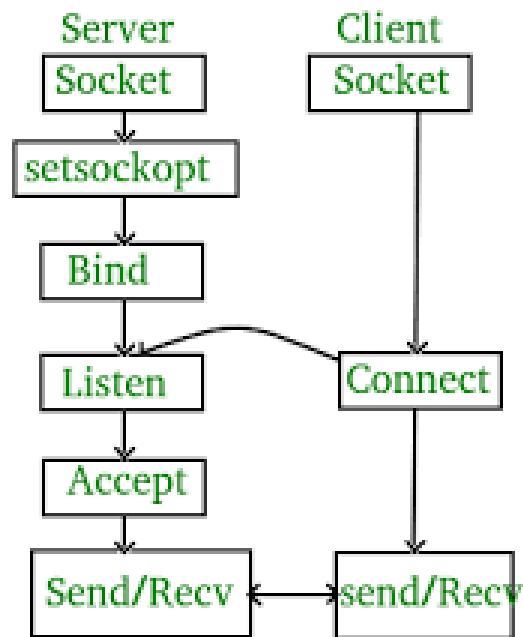


Fig: Server Client Connection through socket

**Sockets** let apps attach to the local network at different **ports**

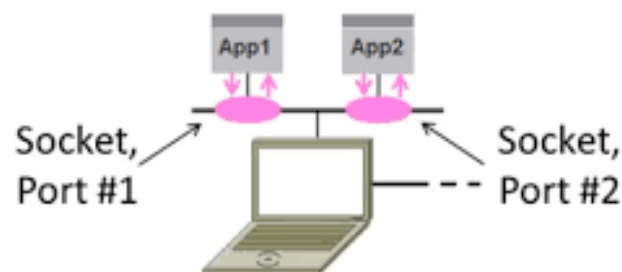


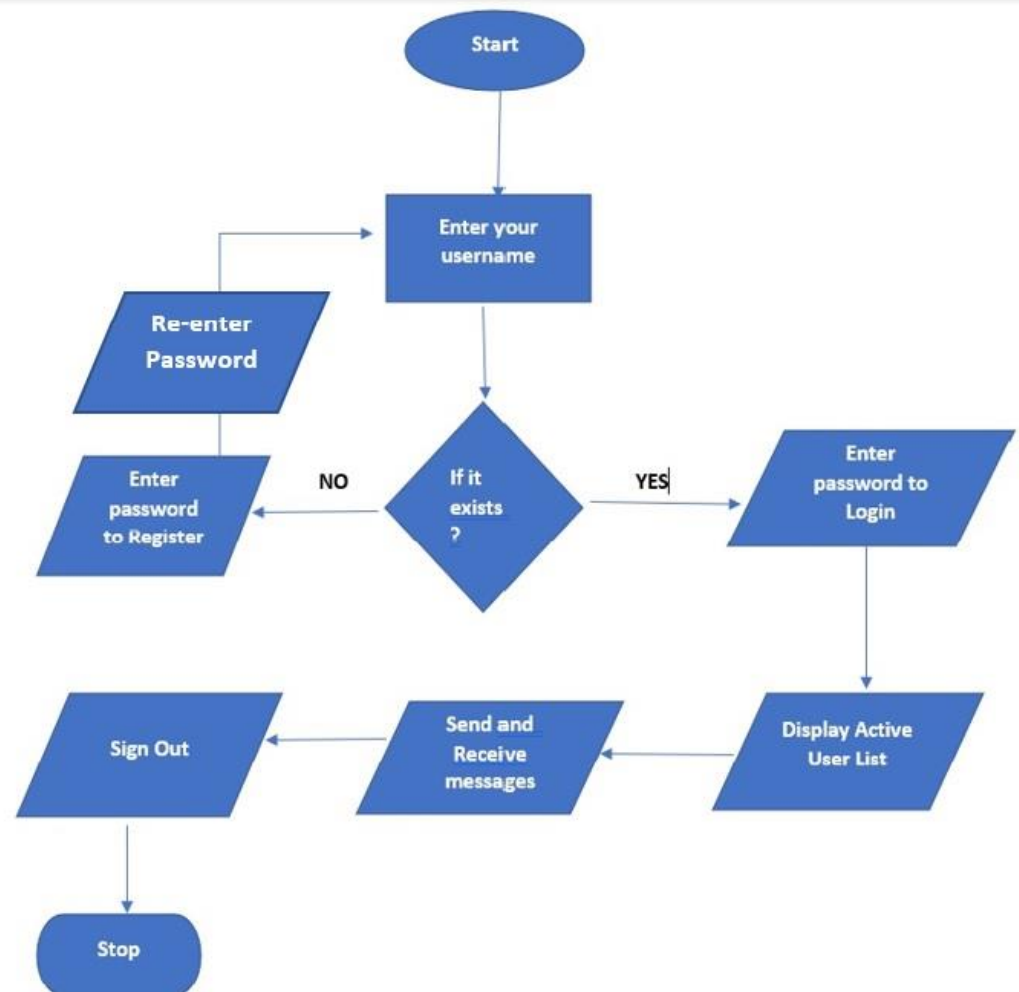
Fig: Image describing the operations of a socket

### 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.

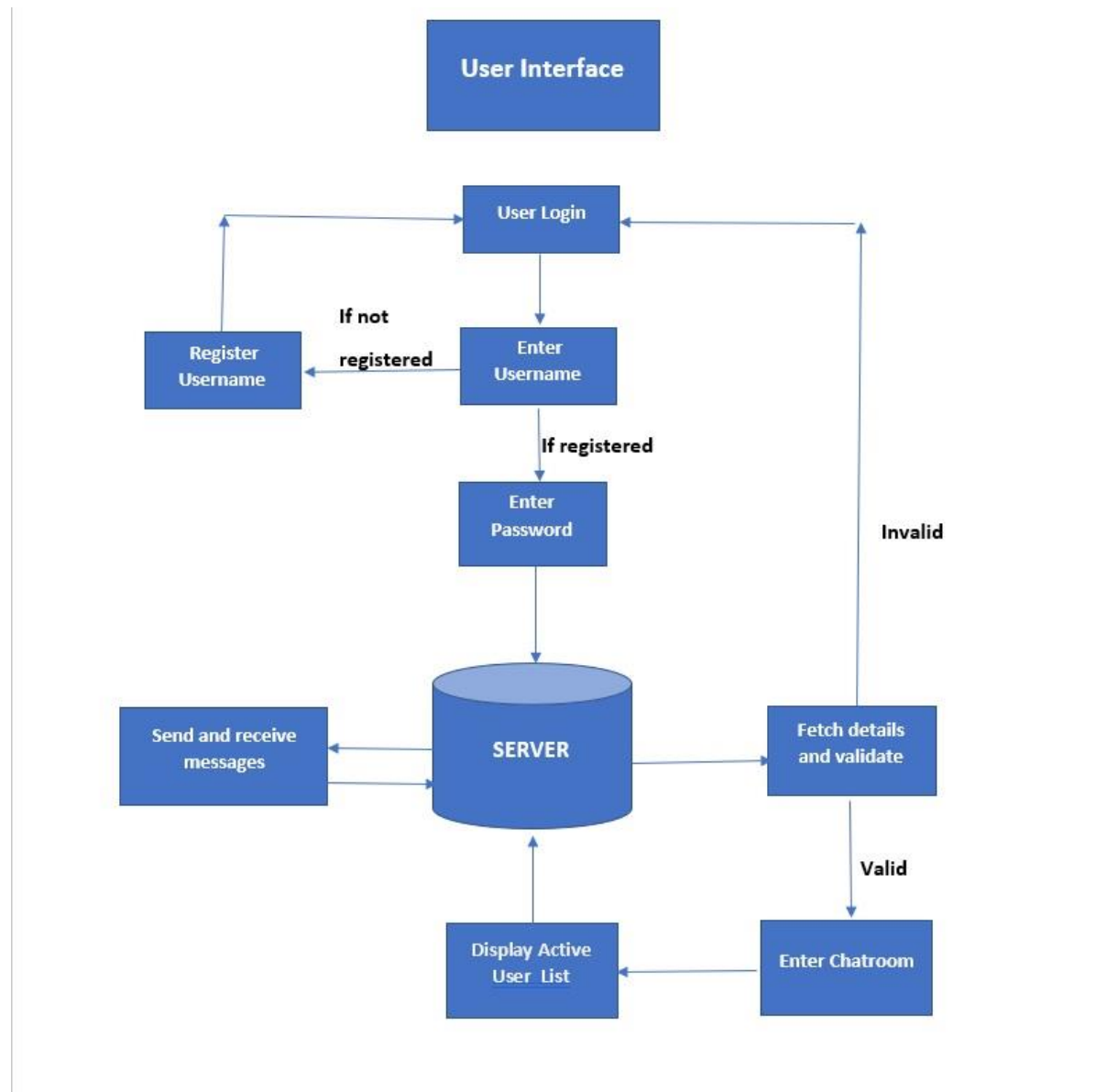
## 4. Detailed System Design

### 4.1 Flowchart Of the application

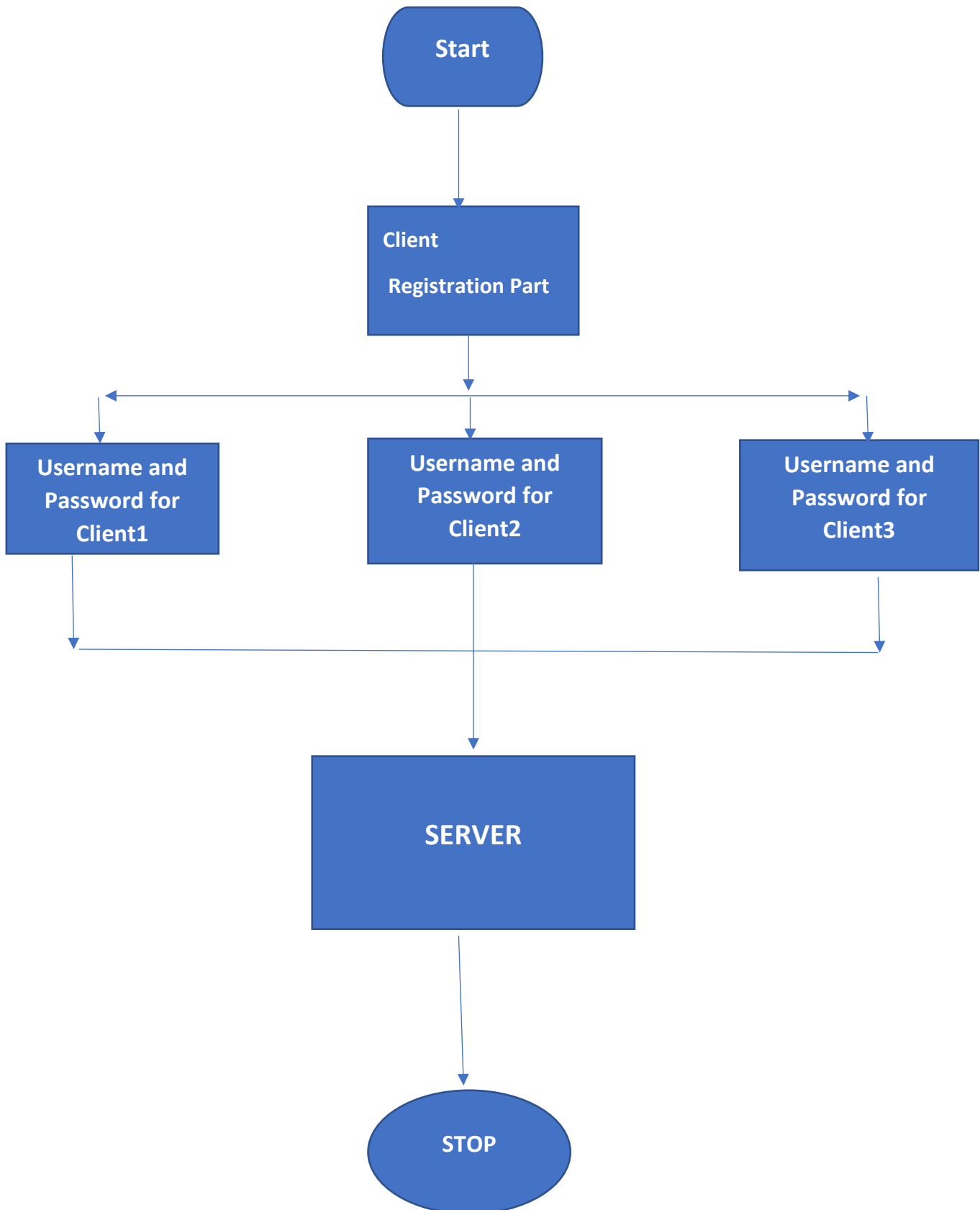




## 4.2 User/Client Interface Detailed Data Flow



### 4.3 Flow Chart for User Registration



## 5. TOOLS REPORT

### 5.1 Valgrind Report (Server File)

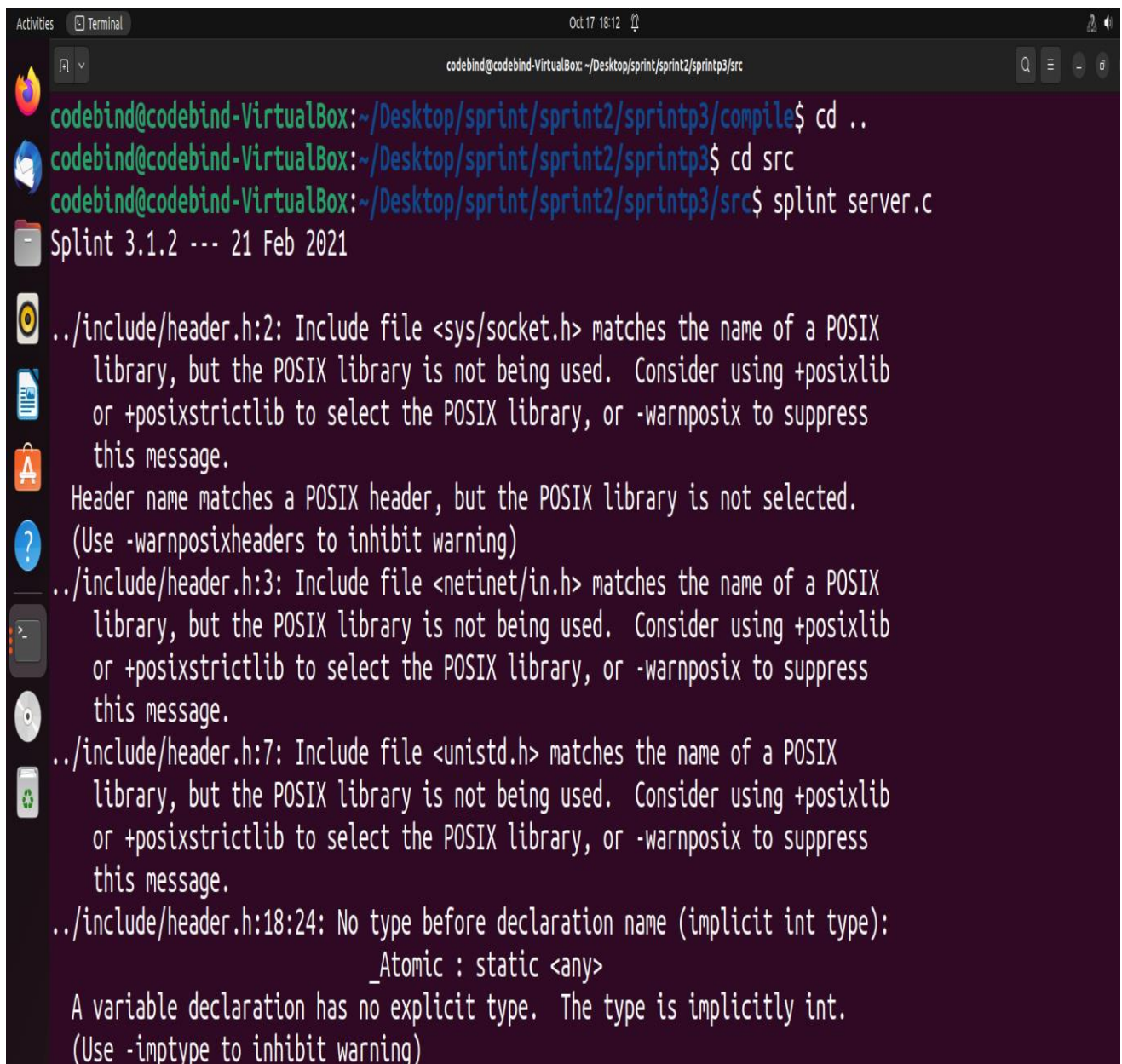
```
codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprint3/compile$ valgrind ../bin/server.exe
==3506== Memcheck, a memory error detector
==3506== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==3506== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==3506== Command: ../bin/server.exe
==3506==
=== WELCOME TO THE CHATROOM ===
^C==3506==
==3506== Process terminating with default action of signal 2 (SIGINT)
==3506==   at 0x49915D7: accept (accept.c:26)
==3506==   by 0x109F51: main (server.c:216)
==3506==
==3506== HEAP SUMMARY:
==3506==   in use at exit: 1,024 bytes in 1 blocks
==3506== total heap usage: 1 allocs, 0 frees, 1,024 bytes allocated
==3506==
==3506== LEAK SUMMARY:
==3506==   definitely lost: 0 bytes in 0 blocks
==3506==   indirectly lost: 0 bytes in 0 blocks
==3506==   possibly lost: 0 bytes in 0 blocks
==3506==   still reachable: 1,024 bytes in 1 blocks
==3506== suppressed: 0 bytes in 0 blocks
==3506== Rerun with --leak-check=full to see details of leaked memory
==3506==
==3506== For lists of detected and suppressed errors, rerun with: -s
==3506== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprint3/compile$
```

### 5.2 Valgrind Report (Client side)

```
codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprint2/compile$ valgrind ../bin/client.exe 8880
==3533== Memcheck, a memory error detector
==3533== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==3533== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==3533== Command: ../bin/client.exe 8880
==3533==
Please enter your name: ^C
Name must be less than 30 and more than 2 characters.
==3533==
==3533== HEAP SUMMARY:
==3533==   in use at exit: 0 bytes in 0 blocks
==3533== total heap usage: 2 allocs, 2 frees, 2,048 bytes allocated
==3533==
==3533== All heap blocks were freed -- no leaks are possible
==3533==
==3533== For lists of detected and suppressed errors, rerun with: -s
==3533== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprint2/compile$
```

## 5.3 Splint Report



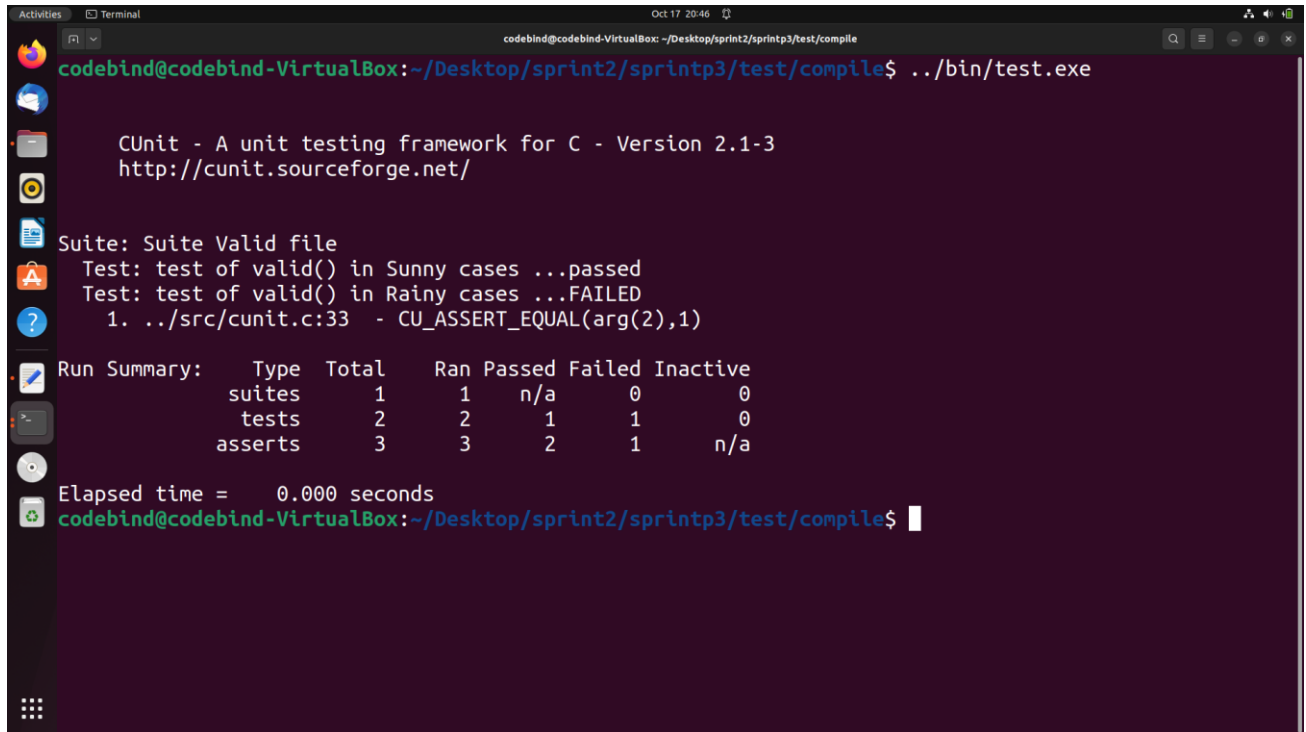
A terminal window titled "Terminal" showing the execution of the Splint static analysis tool. The user is in a directory path `~/Desktop/sprint/sprint2/sprintp3/src`. The command `splint server.c` has been run, and the output shows several warnings related to POSIX library usage and implicit integer types. The terminal window has a dark background with a sidebar on the left containing icons for various applications. The top of the window shows the date and time as "Oct 17 18:12".

```
codebind@codebind-VirtualBox: ~/Desktop/sprint/sprint2/sprintp3/src
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



The screenshot shows a terminal window with the following output:

```
codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprint3/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)

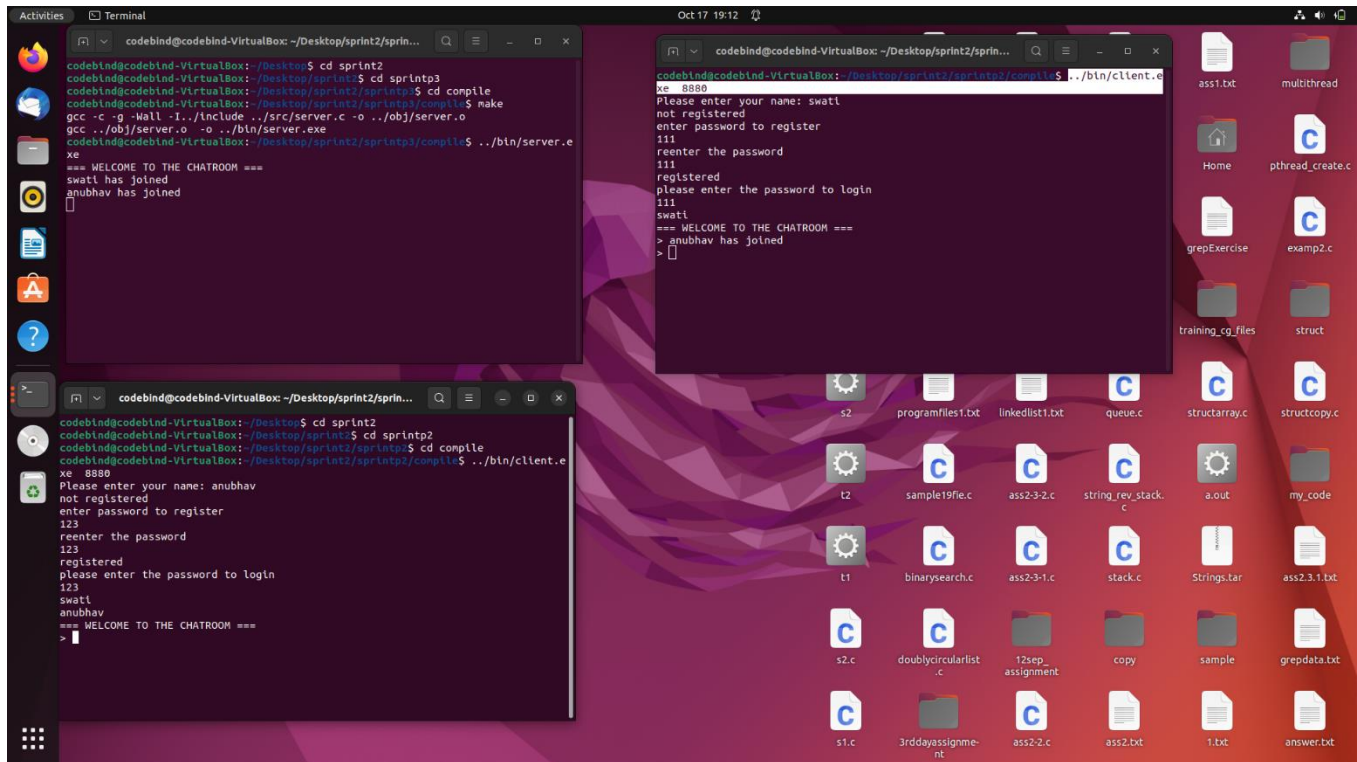
Run Summary:
  Type      Total    Ran  Passed  Failed  Inactive
  suites      1        1    n/a      0         0
  tests       2        2     1        1         0
  asserts     3        3     2        1        n/a

Elapsed time = 0.000 seconds
codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprint3/test/compile$
```

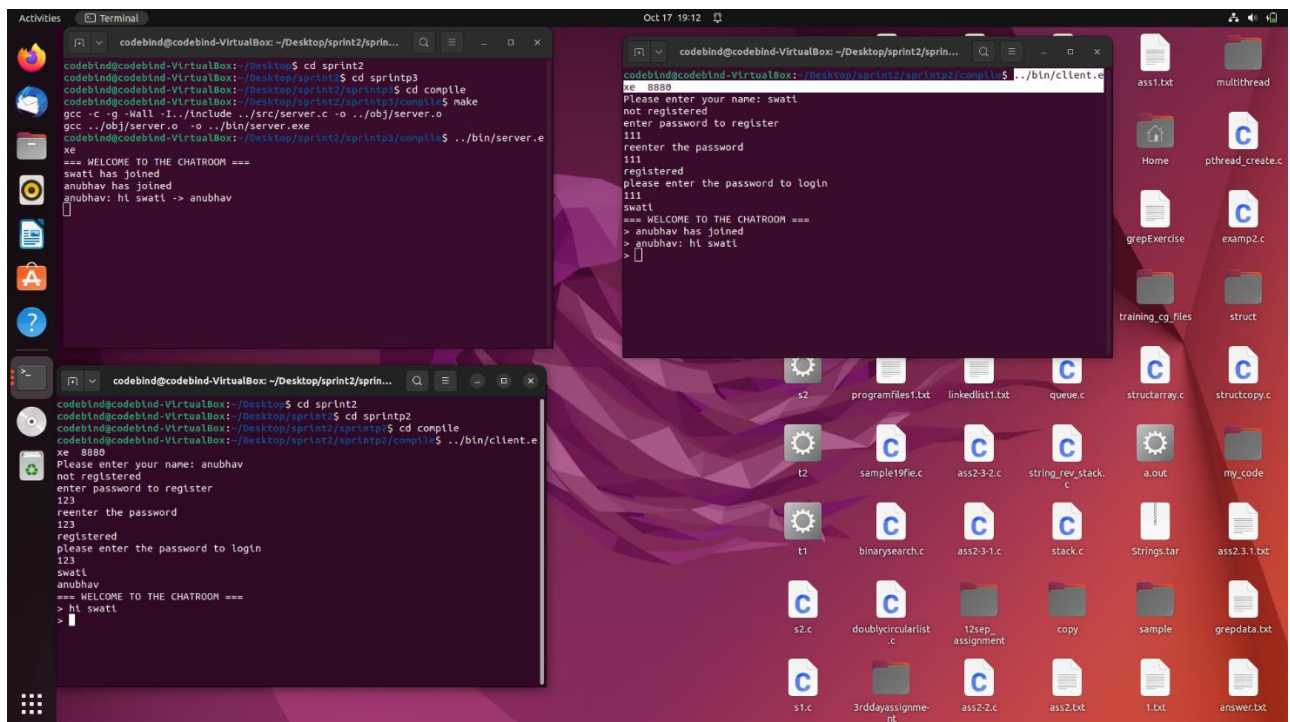
Type	Total	Ran	Passed	Failed	Inactive
suites	1	1	n/a	0	0
tests	2	2	1	1	0
asserts	3	3	2	1	n/a

## 6.2 Integration Testing Reports

### 6.2.1 User Registration and Login



### 6.2.2 Active User List Display





## 6.2.3 Message Exchanges

```
codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprin...
codebind@codebind-VirtualBox: ~/Desktop$ cd sprint2
codebind@codebind-VirtualBox: ~/Desktop/sprint2$ cd sprint3
codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprint3$ cd compile
codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprint3/compile$ make
gcc -c -g -Wall -I../include ./src/server.c -o ../obj/server.o
codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprint3/compile$ ./bin/server.e
xe 8880
=== WELCOME TO THE CHATROOM ===
swati has joined
anubhav has joined
anubhav: hi swati -> anubhav
pavithra has joined
pavithra: hlo swati -> pavithra
[]

codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprin...
codebind@codebind-VirtualBox: ~/Desktop$ cd sprint2
codebind@codebind-VirtualBox: ~/Desktop/sprint2$ cd sprint2
codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprint2$ cd compile
codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprint2/compile$ ./bin/client.e
xe 8880
Please enter your name: anubhav
not registered
enter password to register
123
reenter the password
123
registered
please enter the password to login
123
swati
anubhav
=== WELCOME TO THE CHATROOM ===
> hi swati
> pavithra has joined
> pavithra: hlo swati
> []

codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprin...
codebind@codebind-VirtualBox: ~/Desktop$ cd sprint2
codebind@codebind-VirtualBox: ~/Desktop/sprint2$ cd sprint2
codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprint2$ cd compile
codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprint2/compile$ ./bin/client.e
xe 8880
Please enter your name: pavithra
not registered
enter password to register
345
reenter the password
345
registered
please enter the password to login
345
swati
anubhav
pavithra
=== WELCOME TO THE CHATROOM ===
> hlo swati
> []
```

## 6.2.4 Sending and Receiving Message

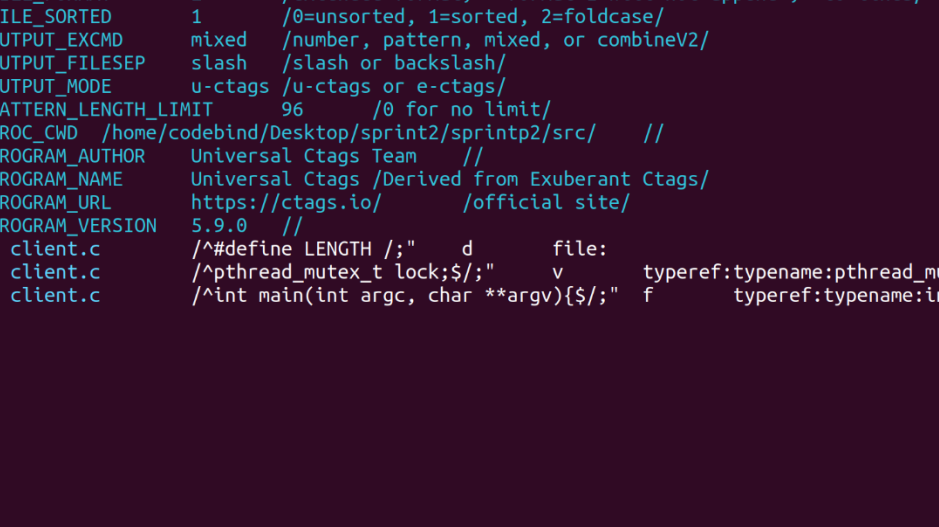
```
codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprin...
codebind@codebind-VirtualBox: ~/Desktop$ cd sprint2
codebind@codebind-VirtualBox: ~/Desktop/sprint2$ cd sprint3
codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprint3$ cd compile
bash: cd: compile: No such file or directory
codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprint3$ cd compile
codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprint3/compile$ make
gcc -c -g -Wall -I../include ./src/server.c -o ../obj/server.o
codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprint3/compile$ ./bin/server.e
xe 8880
=== WELCOME TO THE CHATROOM ===
swati has joined
swati: 111 -> swati
anubhav has joined
anubhav: hello swati -> anubhav
pavithra has joined
pavithra: hi anubhav -> pavithra
[]

codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprin...
codebind@codebind-VirtualBox: ~/Desktop$ cd sprint2
codebind@codebind-VirtualBox: ~/Desktop/sprint2$ cd sprint2
codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprint2$ cd compile
codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprint2/compile$ ./bin/client.e
xe 8880
Please enter your name: anubhav
not registered
enter password to register
111
reenter the password
111
enter valid password
reenter the password
111
registered
please enter the password to login
111
swati
anubhav
=== WELCOME TO THE CHATROOM ===
> hello swati
> pavithra has joined
> pavithra: hi anubhav
> []

codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprin...
gcc ./bin/client.o -o ../bin/client.exe
codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprint2/compile$ ./bin/client.e
Usage: ../bin/client.exe <port>
codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprint2/compile$ ./bin/client.e
xe 8880
Please enter your name: swati
not registered
enter password to register
111
reenter the password
111
registered
please enter the password to login
111
swati
111=== WELCOME TO THE CHATROOM ===
>
> anubhav has joined
> anubhav: hello swati
> pavithra has joined
> pavithra: hi anubhav
> []

codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprin...
codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprint2$ cd compile
codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprint2/compile$ ./bin/client.e
xe 8880
Please enter your name: pavithra
not registered
enter password to register
345
reenter the password
345
registered
please enter the password to login
345
swati
anubhav
pavithra
=== WELCOME TO THE CHATROOM ===
> hi anubhav
> []
```

### 6.3 Ctags Report

[illegible]

```
codebind@codebind-VirtualBox: ~/Desktop/sprint2/sprint2/src
!_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/sprint2/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
```



## **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:-

Connect to server socket

Send And Receive Messages Through server using Sockets

