VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY UNIVERSITY OF
TECHNOLOGY FACULTY OF COMPUTER SCIENCE AND ENGINEERING

COMPUTER NETWORKS (CO3094)

Assignment

NETWORK APPLICATION

Advisor: Nguyen Le Duy Lai

Students: Nguyen Minh Hung – 2052504

Hong Huy Man – 2052593

Dang Tien Dat – 2052436

HO CHI MINH CITY, NOVEMBER 2022

TABLE OF CONTENT

I. Introduction

Nowadays, a chat application **makes it easy to communicate with people anywhere in the world by sending and receiving messages in real-time**. With a chat app, users are able to receive the same engaging and lively interactions through custom messaging features, just as they would in person. Not only that, chat apps can now even transfer files, which have a lot of usage.

Based on that, we implement a chat app application using Client-Sever and Peer-to-Peer (P2P) protocols in this assignment.

II. Requirement Analyst

In this chat application, we implement the following functions and their protocol accordingly

Functional:

- Authentication: Users can register a new account in the server (TCP) and log in to the server (TCP).
- Friendlist: Users can retrieve a friend list from the server (TCP)
- Addfriend: Users can add a new friend by entering a friend's username (TCP).
- Chat: Users can start a chat with a friend who is online (TCP).
- File transfer: While chatting, users can send files to their friends (TCP).

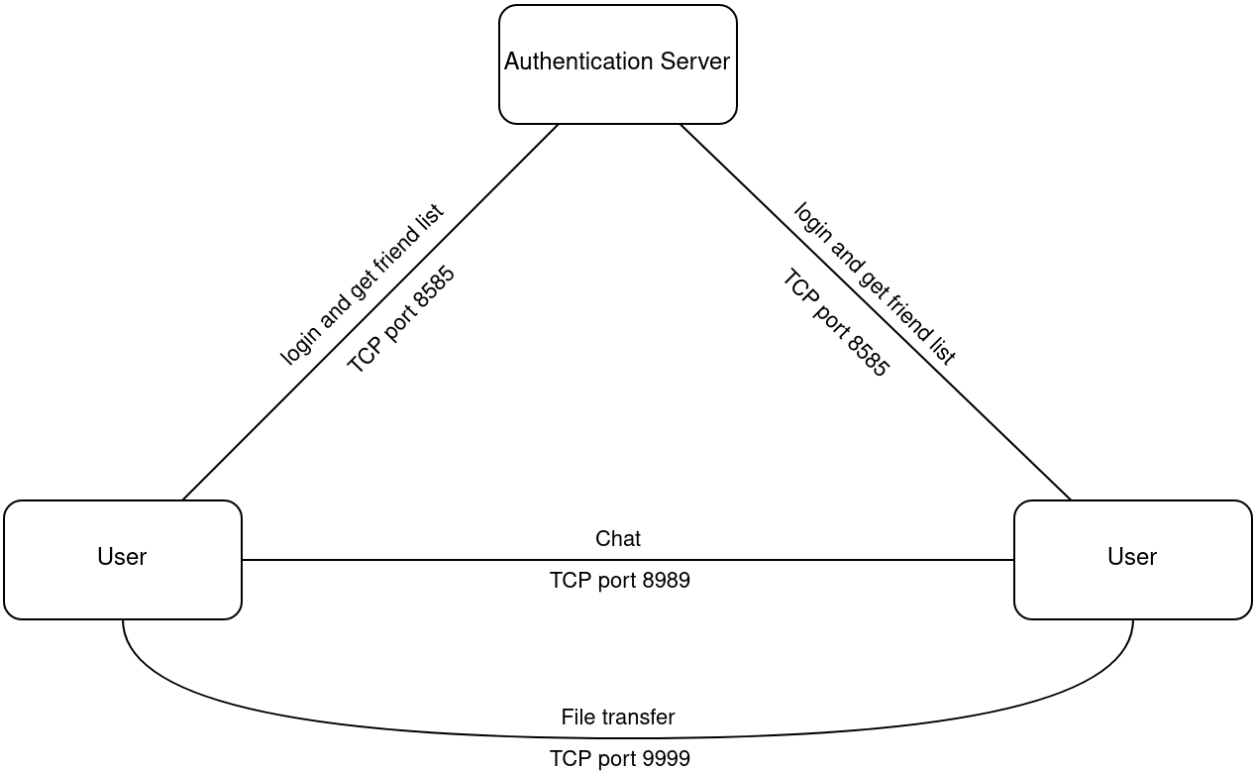
Non-functional:

- Users can chat with multiple people at the same time.
- Users can send files to multiple people at the same time.
- Users can receive files from multiple people at the same time.
- The connection between each user is P2P.
- User data on the server is saved if the server is shut down.

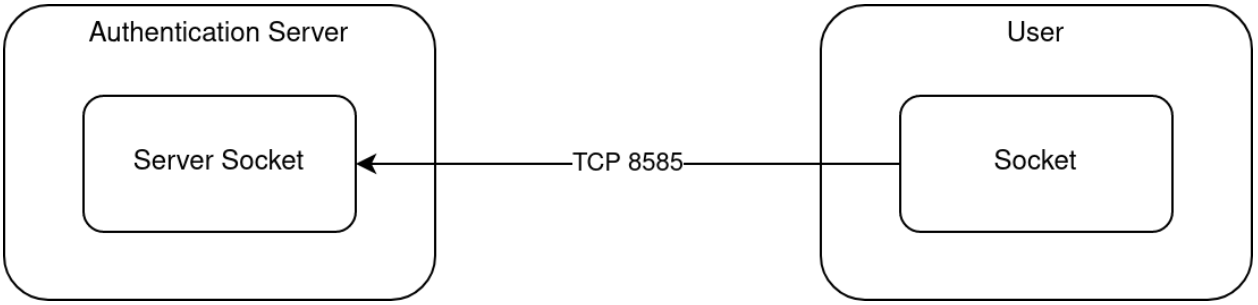
III. Application architecture

In this subsection, we use the below diagrams to show an overview of our application architecture.

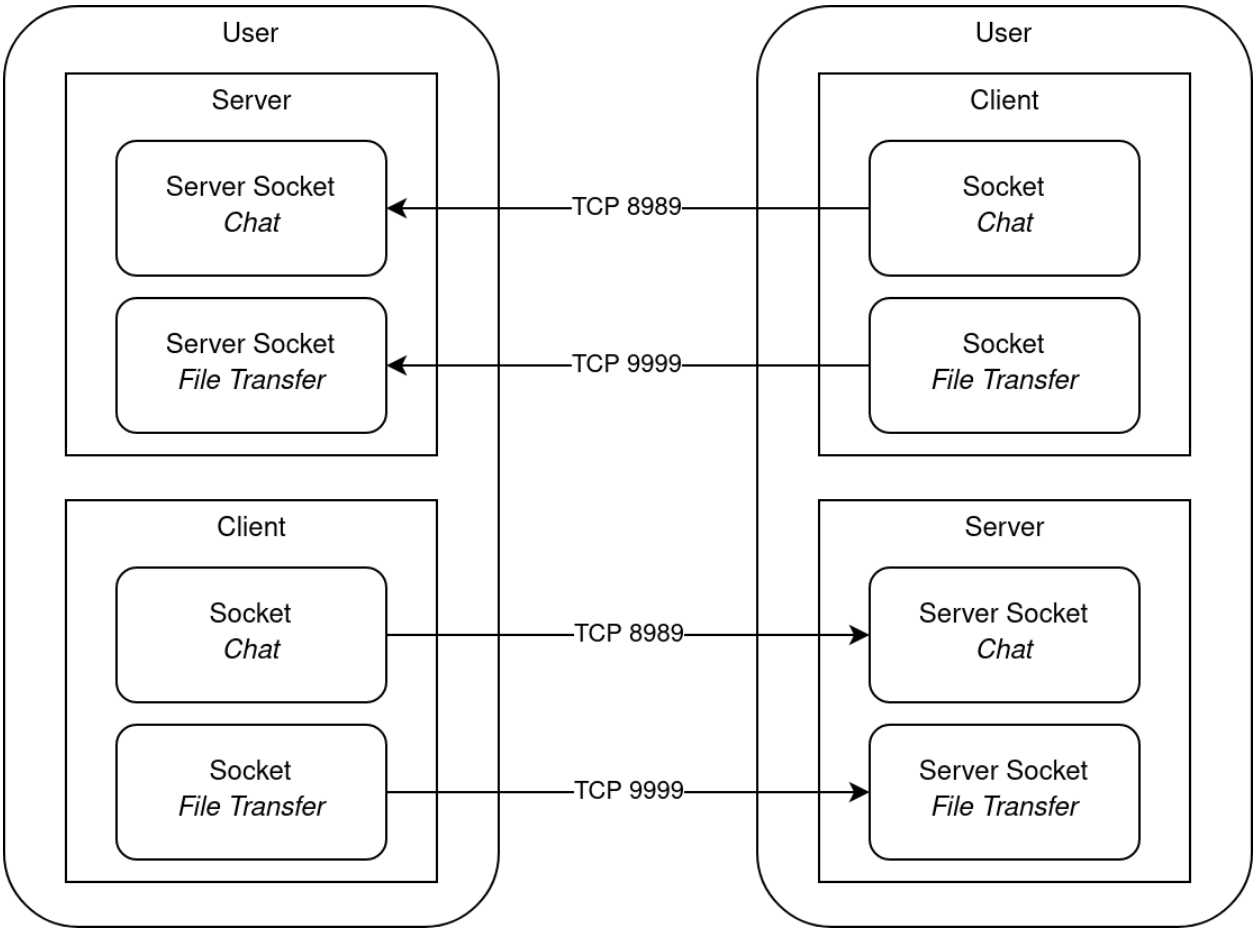
1. System overall architecture



2. Authentication Server - User

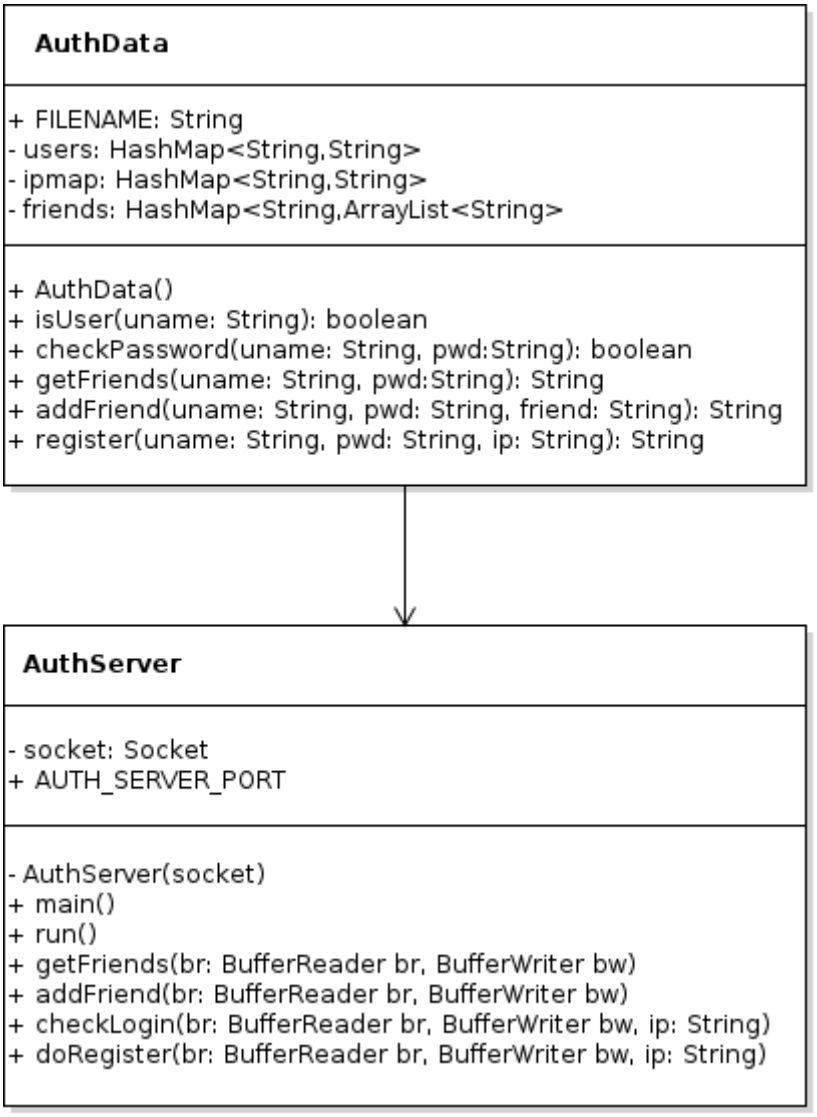


3. User - User

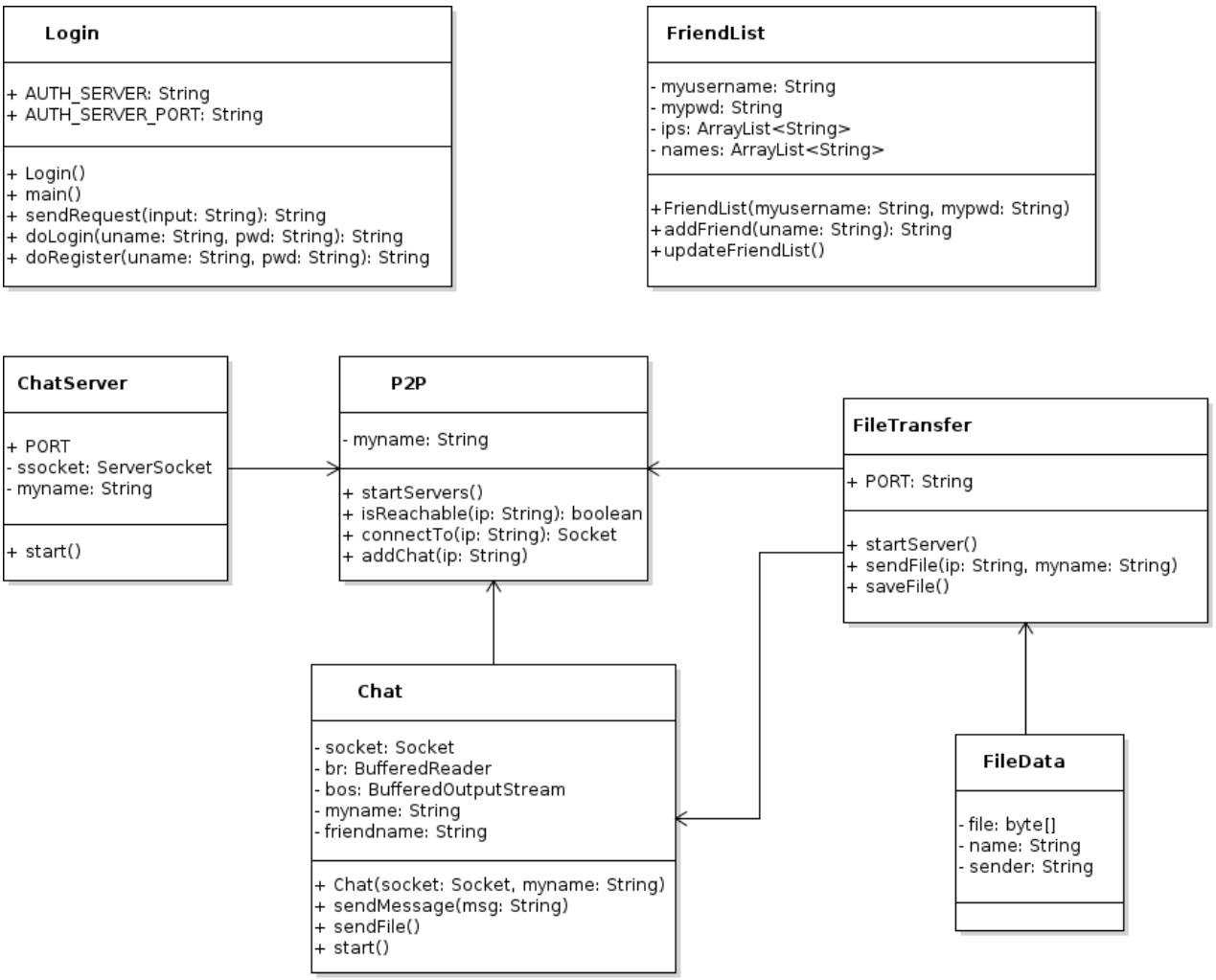

User
Server
Server Socket
Chat
Server Socket
File Transfer
Client
Socket
Chat
Socket
File Transfer
TCP 8989
TCP 9999
User
Client
Socket
Chat
Socket
File Transfer
Server
Server Socket
Chat
Server Socket
File Transfer
TCP 8989
TCP 9999

IV. Class diagram

1. Sever

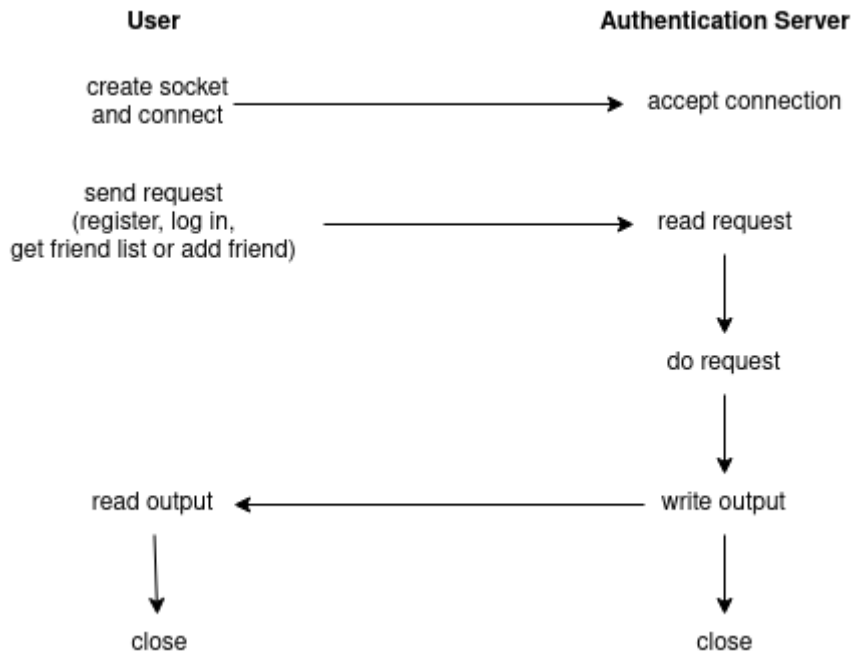


2. User


Login
+ AUTH_SERVER: String
+ AUTH_SERVER_PORT: String
+ Login()
+ main()
+ sendRequest(input: String): String
+ doLogin(uname: String, pwd: String): String
+ doRegister(uname: String, pwd: String): String
FriendList
- myusername: String
- mypwd: String
- ips: ArrayList<String>
- names: ArrayList<String>
+FriendList(myusername: String, mypwd: String)
+addFriend(uname: String): String
+updateFriendList()
ChatServer
+ PORT
- ssocket: ServerSocket
- myname: String
+ start()
P2P
- myname: String
+ startServers()
+ isReachable(ip: String): boolean
+ connectTo(ip: String): Socket
+ addChat(ip: String)
FileTransfer
+ PORT: String
+ startServer()
+ sendFile(ip: String, myname: String)
+ saveFile()
Chat
- socket: Socket
- br: BufferedReader
- bos: BufferedOutputStream
- myname: String
- friendname: String
+ Chat(socket: Socket, myname: String)
+ sendMessage(msg: String)
+ sendFile()
+ start()
FileData
- file: byte[]
- name: String
- sender: String

V. Function description

1. Registration, Login, Get friend list and Add a friend



Register:

+ Request string: “Register\n<username>\n<password>\n”
+ Output string: “DONE”, “Username Already in Use”
+ Description:
    1. Users send a registration request with username and password to the Authentication Server.
    2. The server checks the username against its database
        a. If username already exists, write output “Username Already in Use”
        b. If username can be used, add a new user with the corresponding username and password to the database, then write output “DONE”

Log in:

+ Request string: “Login\n<username>\n<password>\n”
+ Output string: “DONE”, “Wrong Password”, “Username does not exist”
+ Description:
    1. User sends a login request with username and password to the Authentication Server.
    2. The server checks the username and password against its database
        a. If username does not exist, write output “Username does not exist”
        b. If username exists but wrong password, write output “Wrong Password”

c. If username and password match, write output “DONE”. The application then proceeds to get user's friend list.

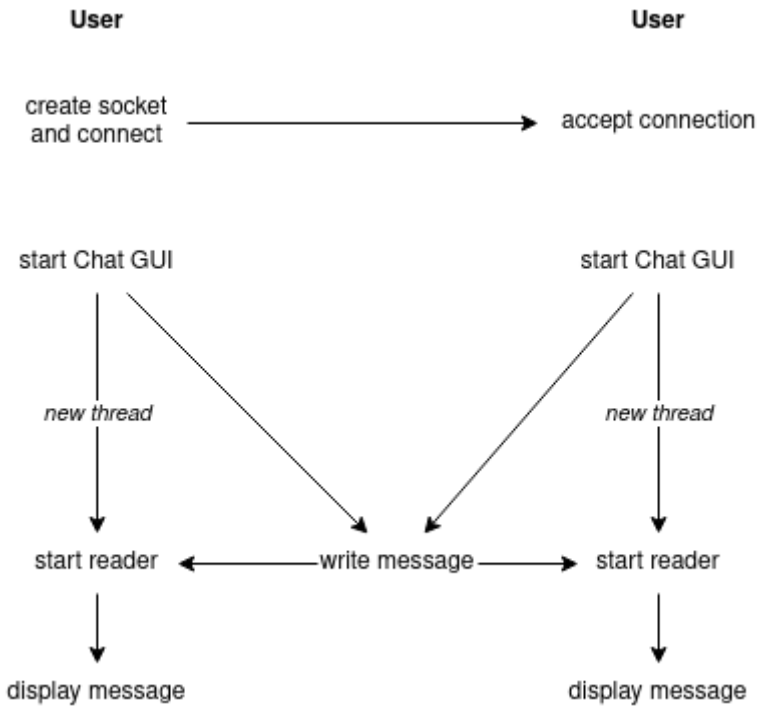
Get friend list:

+ Request string: “Friends\n<username>\n<password>\n”
+ Output string: “<IP #1>\n<username #1>\n
  <IP #2>\n<username #2>\n
  ...
  <IP #n>\n<username #n>\n”
+ Description:
  1. User sends a get friend list request with username and password to the Authentication Server (requires authentication for this request due to security reasons).
  2. Server retrieves friends' names and IPs, and writes output back to the user.

Add a friend:

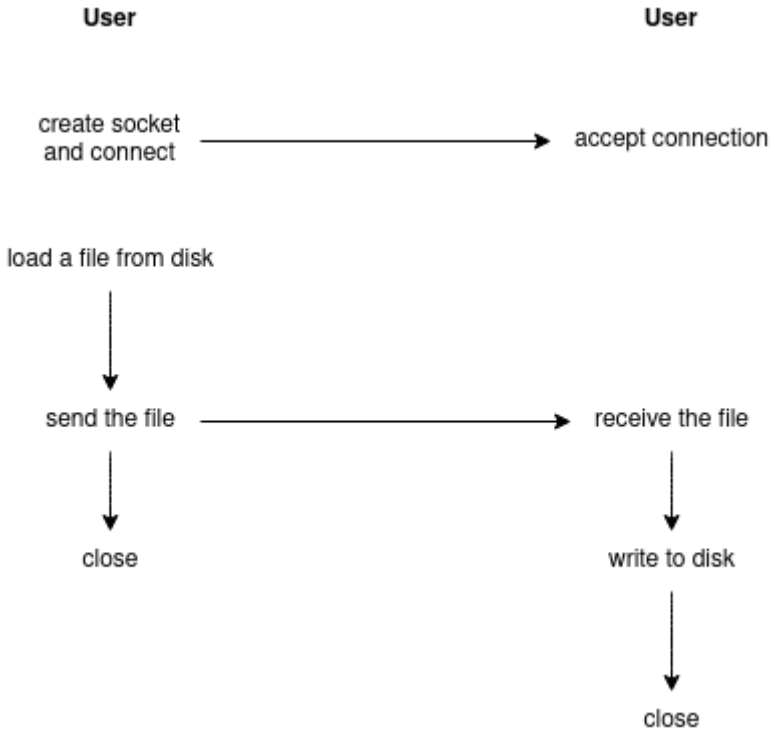
+ Request string: “Add Friend\n<username>\n<password>\n<friendsname>\n”
+ Output string: “DONE”, “Already a Friend”, “Username does not exist”
+ Description:
  1. User sends an add a friend request with username and password to the Authentication Server (also requires authentication for this request due to security reasons).
  2. Server checks friend’s username against its database
     a. If friend’s username does not exist, write output “Username does not exist”
     b. If friend’s username exists but user already added, write output “Already a Friend”
     c. Otherwise, the server adds a new friend for the user in its database and writes output “DONE”. The application then sends a get friend list request to the server to update user’s friend list.

2. Chat



Description: After users click on a name in the friend list, the application will send a connect request to the corresponding friend. When the connection is successfully created, a chat box will appear on each side of the user's screen and both can start chatting. When one side chooses to send the message, the message will be displayed on both sides as well.

3. File transfer



Description: When the sender send a file, a box will open and ask the sender to browse the file the location, and vice versa, the application also ask the receiver to browse the location to save the file when the file is sent successfully.

VI. Validation and evaluation of actual result:

After researching, implementing, and completing our code, we did approximately 20 tests on different laptops and computers (about five on each one) to ensure the application runs smoothly. We did encounter some unexpected bugs and some small UI-UX problems so we spent some more time refining our application and repeat testing.

After fixing all the problems, we can conclusively conclude that:

- All the features of the application work perfectly fine and meet the assignment requirements.
- Even though the GUI is quite simple, it can be used by everyone without much difficulty.
- The application can work on many different operating systems.
- There are no glitches and the latency between chat is negligible.

However, there are still some difficulties that remain:

- The file transfer sends files to the receiver's RAM, so if the size of the file is too large, it will cause lag. We suggest sender should only send files that have size less than 500Mb.
- Difficulty in connecting when using public IP due to ISP security policy.

VII. User manual

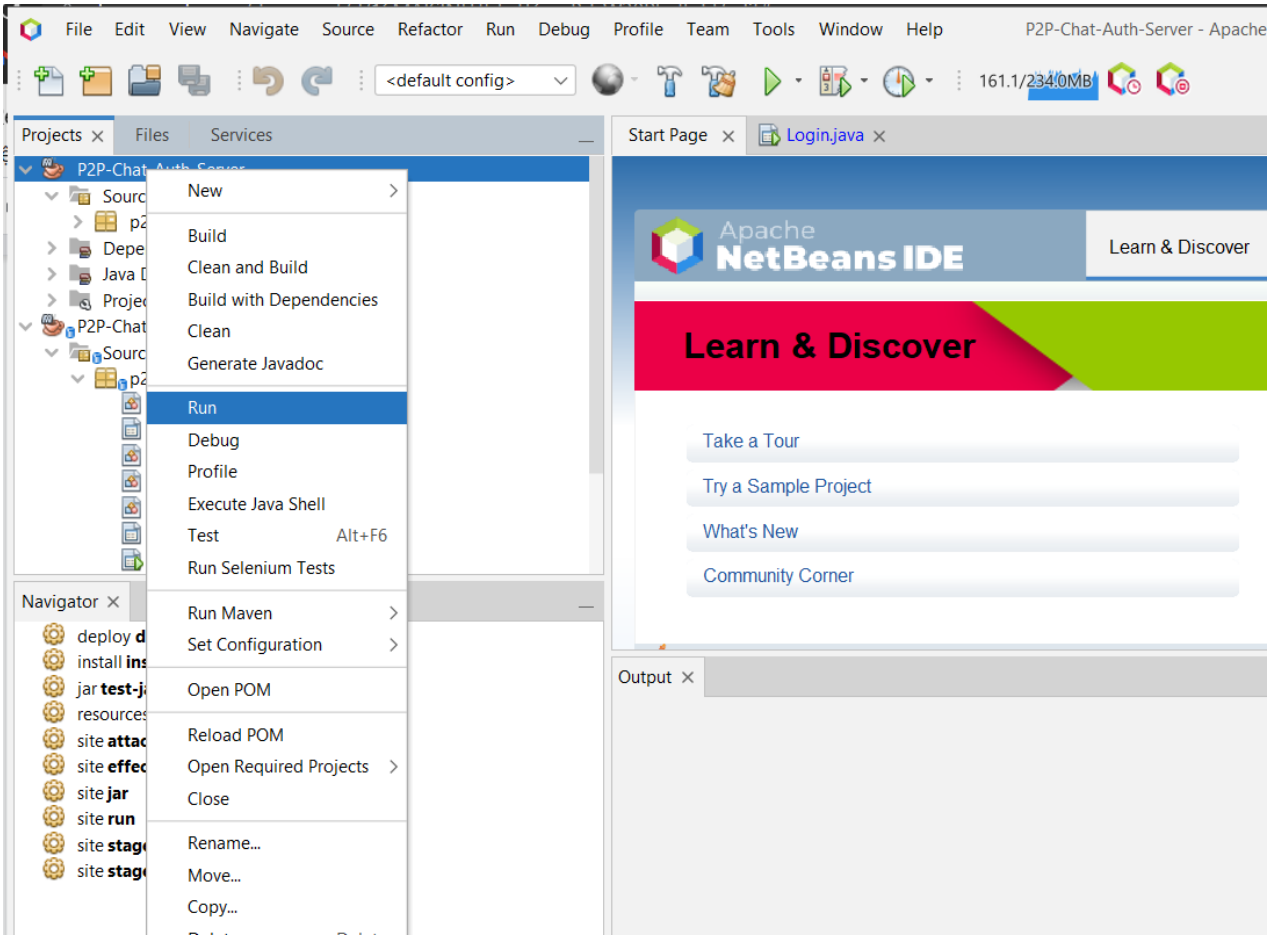
1. Set Up

a. Source Code

To run the source code you will need to have these programs installed:

- Java(TM) SE Runtime Environment 18 or above
- Apache Netbeans IDE 15

After installation, open Apache Netbean, open folder "P2P-Chat-Auth-Server" and "P2P-Chat-Client". Then right-click the folder and select "Run". The "P2P-Chat-Auth-Server" must be run before "P2P-Chat-Client" to initialize the server for the chat application to work.



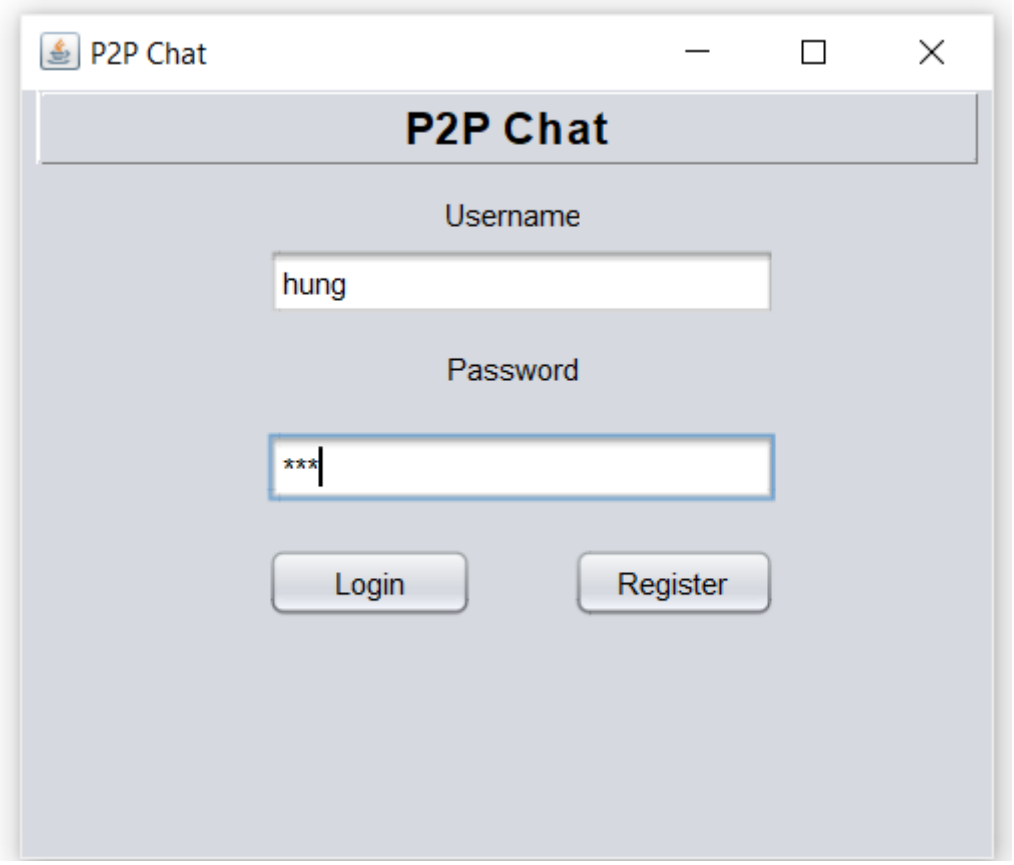
b. Compiled file

Just double-click on the "P2P-Chat-Auth-Server-1.0.jar" file and then do the same with "P2P-Chat-Client-1.0.jar"

2. Run the program

a. Login and Register

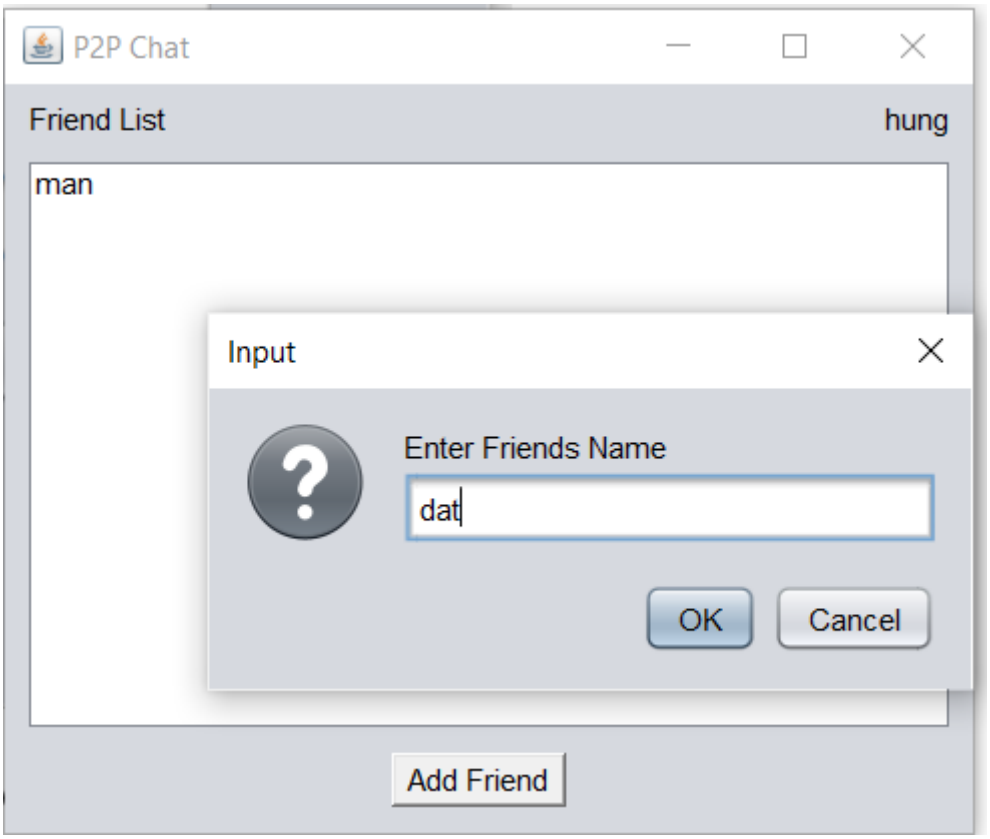
To register an account, type in your Username and Password, then click on Register button.



Doing the same to login into an existing account.

b. Add Friend

To add a new friend, click on “Add Friend” button, type your friend Username and click “OK”



c. Chatting

Just left-clicks (one time) on your friend Username in the friend list and start chatting.

d. File transfer

Click on the below button to send the file: