

# MỤC LỤC

DANH MỤC CÁC HÌNH .....	2
DANH MỤC CÁC BẢNG .....	3
I. Giới thiệu đề tài .....	4
1. Ngữ cảnh sử dụng phần mềm .....	4
2. Mô tả ứng dụng .....	4
3. Đặc điểm, tính năng .....	4
4. Hướng xây dựng ứng dụng .....	4
5. Sơ đồ truyền tin giữa Server và Client.....	6
6. Sơ đồ mô tả chức năng của ứng dụng .....	9
II. Quá trình thực hiện.....	10
1. Thiết kế giao diện .....	10
2. Thiết kế lớp .....	13
2.1 Thiết kế lớp cho Server .....	13
2.1.1 Các lớp được sử dụng cho Server.....	13
2.1.2 Mô tả các phương thức của Server .....	14
2.2 Thiết kế lớp cho Client.....	21
2.2.1 Các lớp được sử dụng cho Client .....	21
2.2.2 Mô tả các phương thức của các lớp của Client .....	23
III. Phân công công việc .....	311
KẾT LUẬN .....	32
TÀI LIỆU THAM KHẢO .....	33

# DANH MỤC CÁC HÌNH

Hình 1.1 Sơ đồ truyền tin Public Message .....	7
Hình 1.2 Sơ đồ truyền tin Private Message .....	7
Hình 1.3 Sơ đồ truyền tin connect, login, signup, signout message.....	8
Hình 1.4 Sơ đồ truyền tin download, upload và transfer file .....	8
Hình 1.5 Sơ đồ mô tả chức năng của Server .....	9
Hình 1.6 Sơ đồ mô tả chức năng của Client.....	10
Hình 2.1 Màn hình Server .....	11
Hình 2.2 Màn hình Client.....	12
Hình 2.3 Màn hình History.....	12

## DANH MỤC CÁC BẢNG

Bảng 2.1 Thiết kế giao diện.....	10
Bảng 2.2 Danh mục các lớp được sử dụng trong chương trình Server .....	13
Bảng 2.3 Bảng mô tả các phương thức trong lớp Message.....	14
Bảng 2.4 Bảng mô tả các phương thức trong lớp Database .....	15
Bảng 2.5 Bảng mô tả các phương thức trong lớp ServerThread .....	16
Bảng 2.6 Bảng mô tả các phương thức trong lớp ServerSocket .....	17
Bảng 2.7 Bảng mô tả các phương thức trong lớp ServerFrame .....	20
Bảng 2.8 Danh mục các lớp được sử dụng trong chương trình cho Client.....	22
Bảng 2.9 Bảng mô tả các phương thức trong lớp Message cho Client .....	23
Bảng 2.10 Bảng mô tả các phương thức trong lớp Download cho Client.....	24
Bảng 2.11 Bảng mô tả các phương thức trong lớp Upload cho Client .....	25
Bảng 2.12 Bảng mô tả các phương thức trong lớp History cho Client .....	26
Bảng 2.13 Bảng mô tả các phương thức trong lớp SocketClient cho Client .....	28
Bảng 3.1 Bảng mô tả phân công công việc .....	311

# NỘI DUNG

## I. Giới thiệu đề tài

### *1. Ngữ cảnh sử dụng phần mềm*

Dựa vào sự phát triển và phổ biến của hệ điều hành Windows hiện nay, chúng em đã phát triển phần mềm chạy trên nền hệ điều hành này. Phần mềm giúp người dùng có thể kết nối tới một server máy chủ do người dùng tùy chỉnh, từ đó tạo ra một môi trường giúp người dùng có thể trò chuyện riêng tư, gửi tin nhắn tới tất cả mọi người hay trao đổi tài liệu với nhau.

### *2. Mô tả ứng dụng*

- ✓ Tên ứng dụng: Phần mềm TCP Chat
- ✓ Ngôn ngữ lập trình: Java
- ✓ Thư viện được sử dụng cho giao diện: Swing
- ✓ Môi trường lập trình: Eclipse
- ✓ Hướng lưu trữ dữ liệu: File định dạng \*.xml
- ✓ Môi trường thực thi ứng dụng: Hệ điều hành Windows

### *3. Đặc điểm, tính năng*

- Xử lý nhiều người dùng một lúc qua Thread
- Hỗ trợ cho cả tin nhắn công cộng (gửi cho tất cả người dùng) và riêng tư (gửi cho người được chỉ định)
- Đăng ký, đăng nhập
- Hỗ trợ truyền tệp tin
- Xem lại lịch sử chat

### *4. Hướng xây dựng ứng dụng*

- ✓ Cấu trúc Message (cấu trúc đối tượng dùng để giao tiếp giữa Server và Client):

- type: Server và Client có thể hiểu được yêu cầu của tin nhắn như login, message, newuser, ...
- sender: username của người gửi.
- content: Nội dung thực tế của tin nhắn.
- recipient: username của người nhận.
- ✓ Phần mềm được chia làm hai phần:

#### Server:

Có hai lớp chính trong **Server** để xử lý các kết nối và thông điệp. Khi khởi động, **SocketServer** chạy trong một môi trường riêng biệt. Công việc **SocketServer** là chờ kết nối mới từ client và cho mỗi kết nối bắt đầu một luồng mới **ServerThread**. Sau khi kết nối được thiết lập, **ServerThread** sẽ lắng nghe bất kỳ tin nhắn nào và chuyển nó đến **SocketServer** để xử lý. Ngoài ra, nó sẽ chuyển tiếp tin nhắn từ những người dùng khác đến người dùng được kết nối.

```
- // ServerThread đọc các thông điệp gửi tới và chuyển nó cho SocketServer
-
- Message msg = (Message) streamIn.readObject();
- server.handle(ID, msg);
- .....
-
- // SocketServer xử lý các chức năng dựa trên thuộc tính type mà Message gửi tới
-
- public synchronized void handle(int ID, Message msg){
-     if(msg.type.equals("login")){
-         ....
-     }
-     else if(msg.type.equals("message")){
-         if(msg.recipient.equals("All")){ Announce("message", msg.sender,
- msg.content); }
-         else{
-             // Tìm Luồng (Thread) của người nhận và chuyển tiếp cho họ.
-         }
-     }
-     .....
- }
```

#### Client:

Đầu tiên Client kết nối với Server, được chỉ định bởi địa chỉ IP và số cổng của nó.

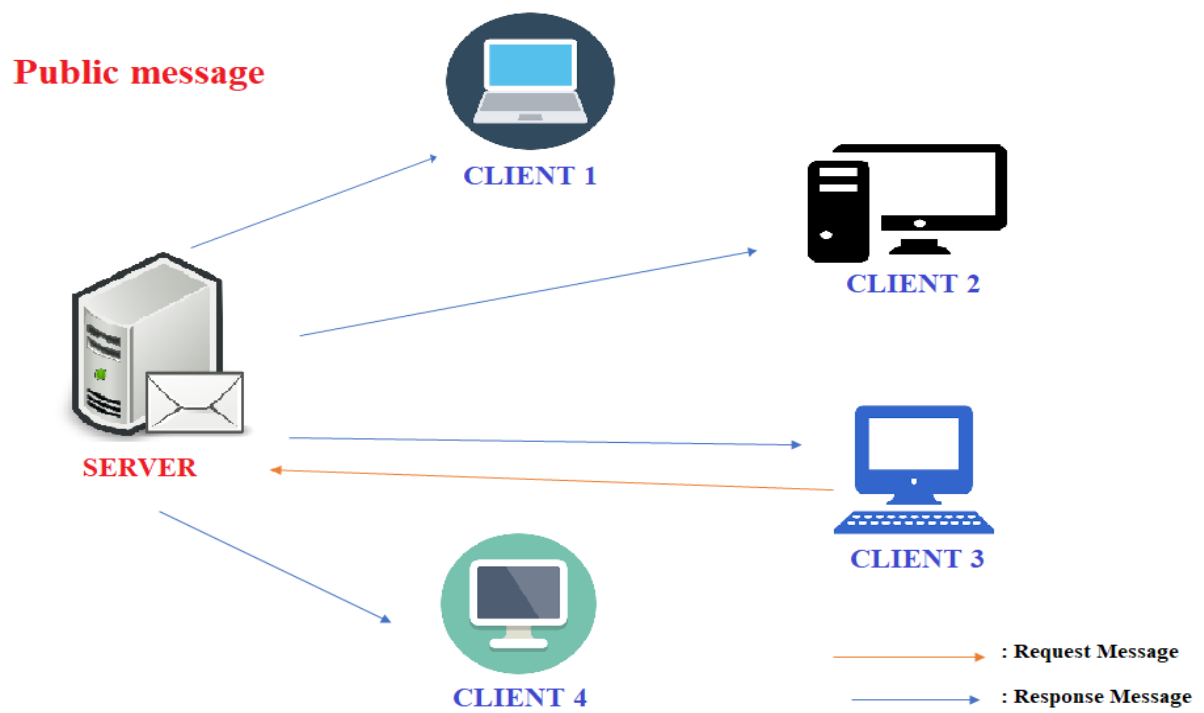
Khi người dùng muốn gửi tệp, trước tiên yêu cầu của họ được gửi qua Message có type = **upload\_req**. Người nhận sau đó thực hiện những việc sau:

- Phía người nhận nhận được một Message kiểu **upload\_res**
- Nếu yêu cầu được chấp nhận thì người nhận sẽ mở một cổng mới
- Người nhận sẽ gửi lại người gửi địa chỉ IP và số cổng
- Người gửi, khi nhận được trả lời đồng ý nhận tệp sẽ kết nối với Socket này và bắt đầu tải lên tệp

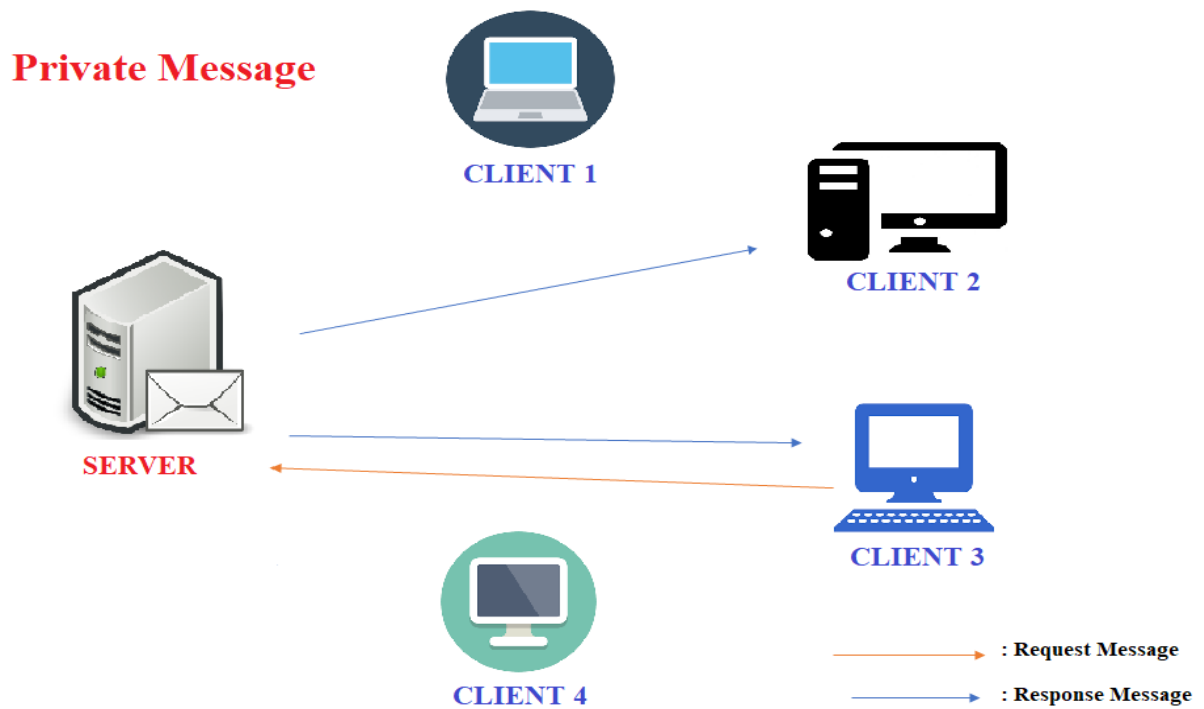
Một lợi thế của phương pháp này là khách hàng có thể trò chuyện và chuyển các tập tin cùng một lúc. Không giống như tin nhắn, các tập tin không đi qua Server.

```
// Ở phía người nhận, mở một Luồng (thread) mới để tải tệp về
Download dwn = new Download(...);
Thread t = new Thread(dwn);
t.start();
send(new Message("upload_res", ui.username, dwn.port, msg.sender));
// Hồi đáp lại cho người gửi địa chỉ IP và cổng kết nối (Port)
.....
// Ở phía người gửi, mở một Luồng mới để tải tệp lên
// Kết nối tới cổng mà người nhận gửi về
Upload upl = new Upload(addr, port, ui.file, ui);
Thread t = new Thread(upl);
t.start();
```

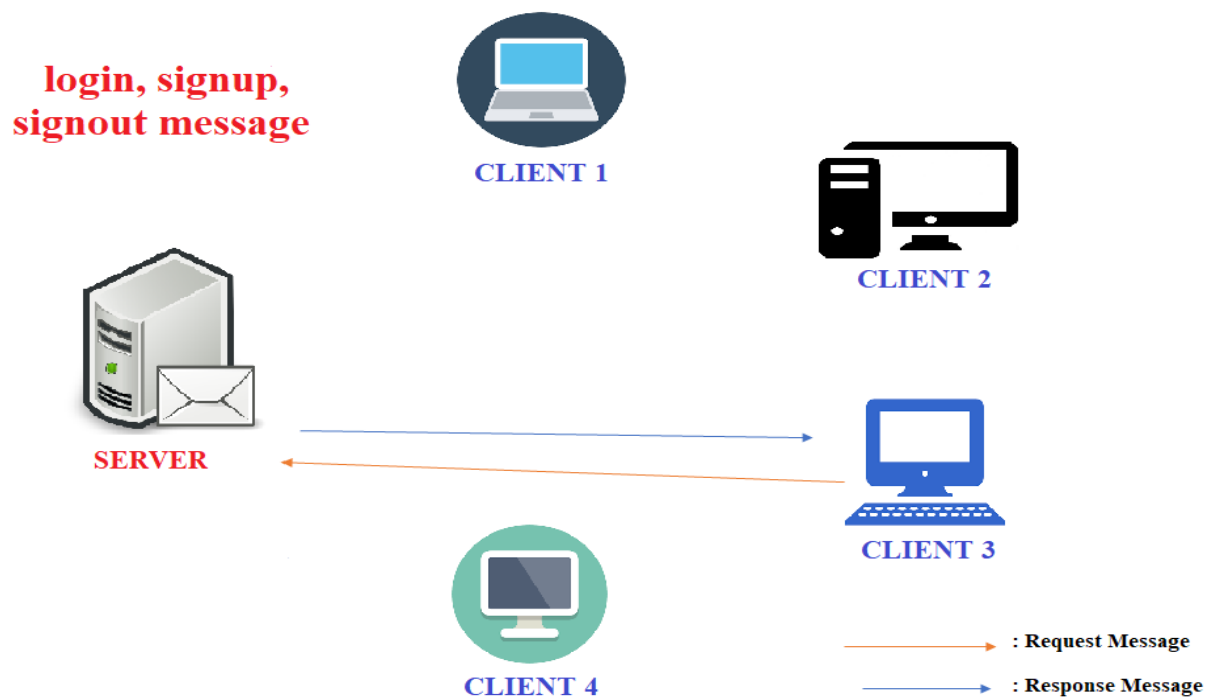
## ***5. Sơ đồ truyền tin giữa Server và Client***



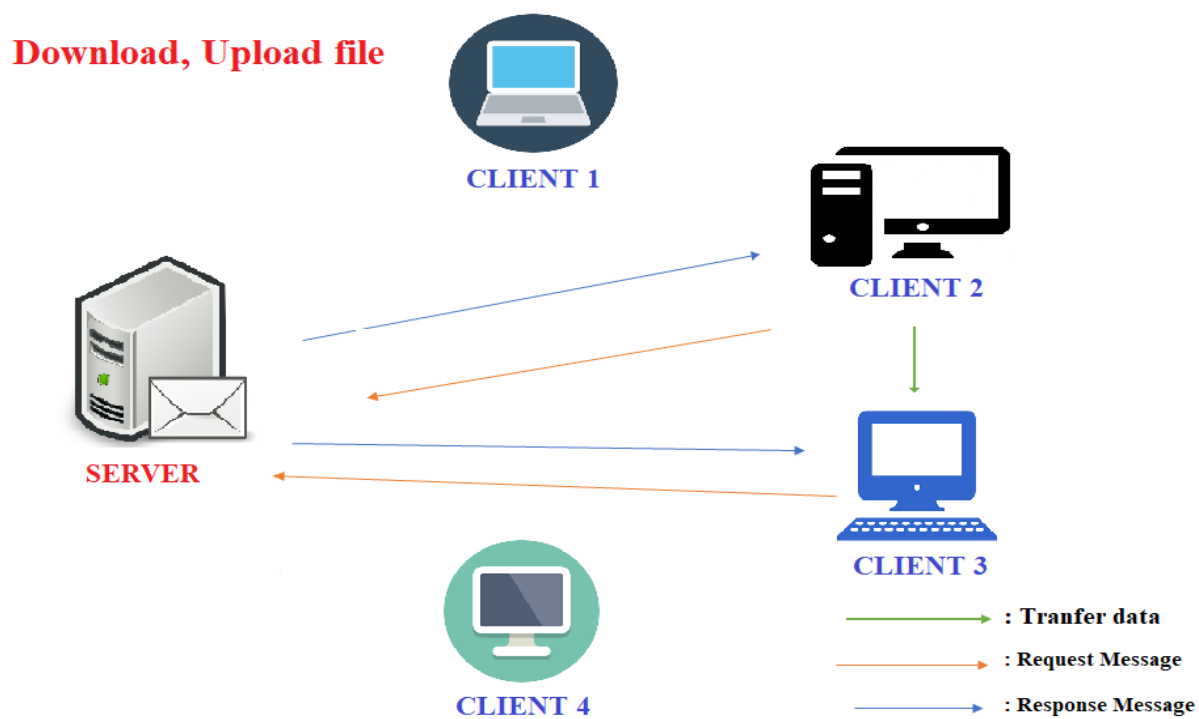
Hình 1.1 Sơ đồ truyền tin Public Message



Hình 1.2 Sơ đồ truyền tin Private Message



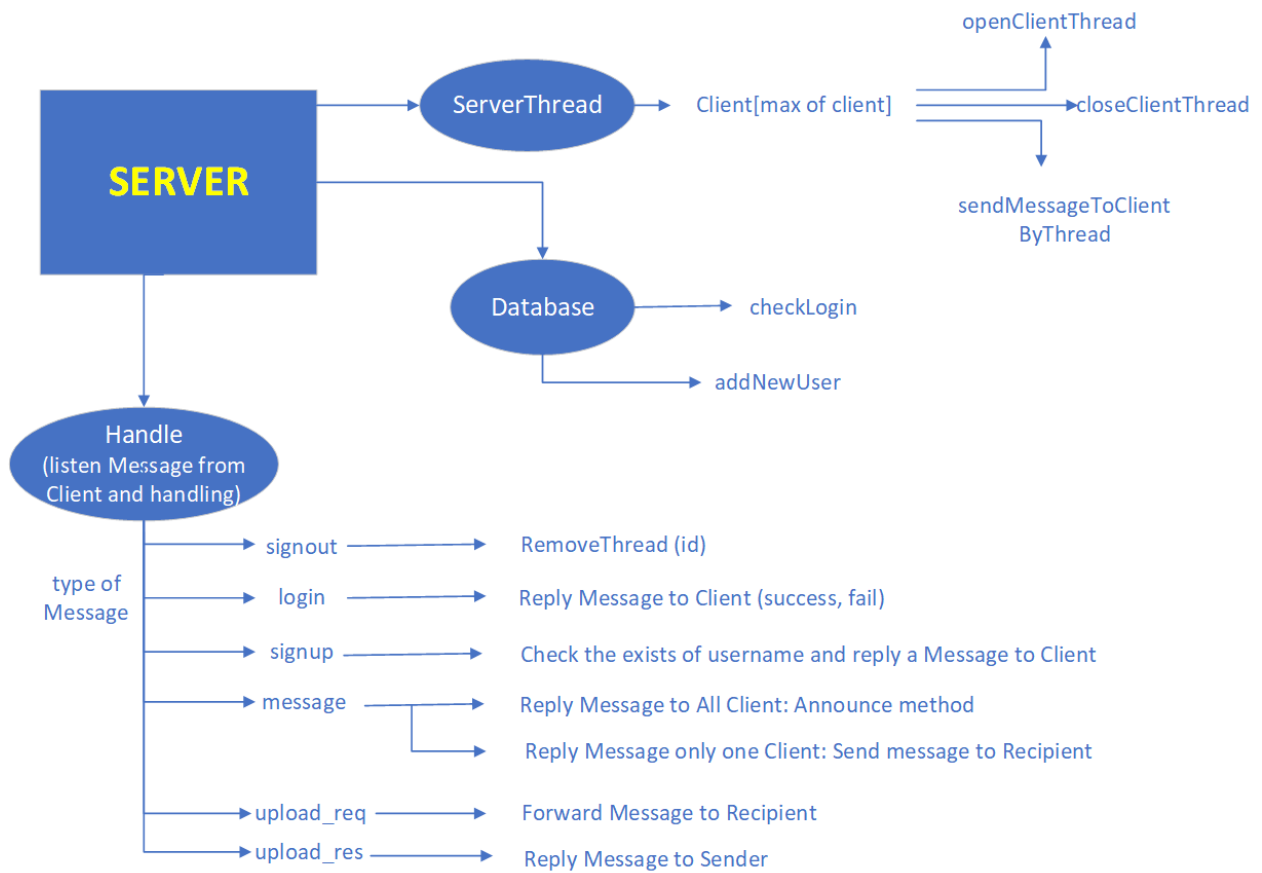
Hình 1.3 Sơ đồ truyền tin connect, login, signup, signout message



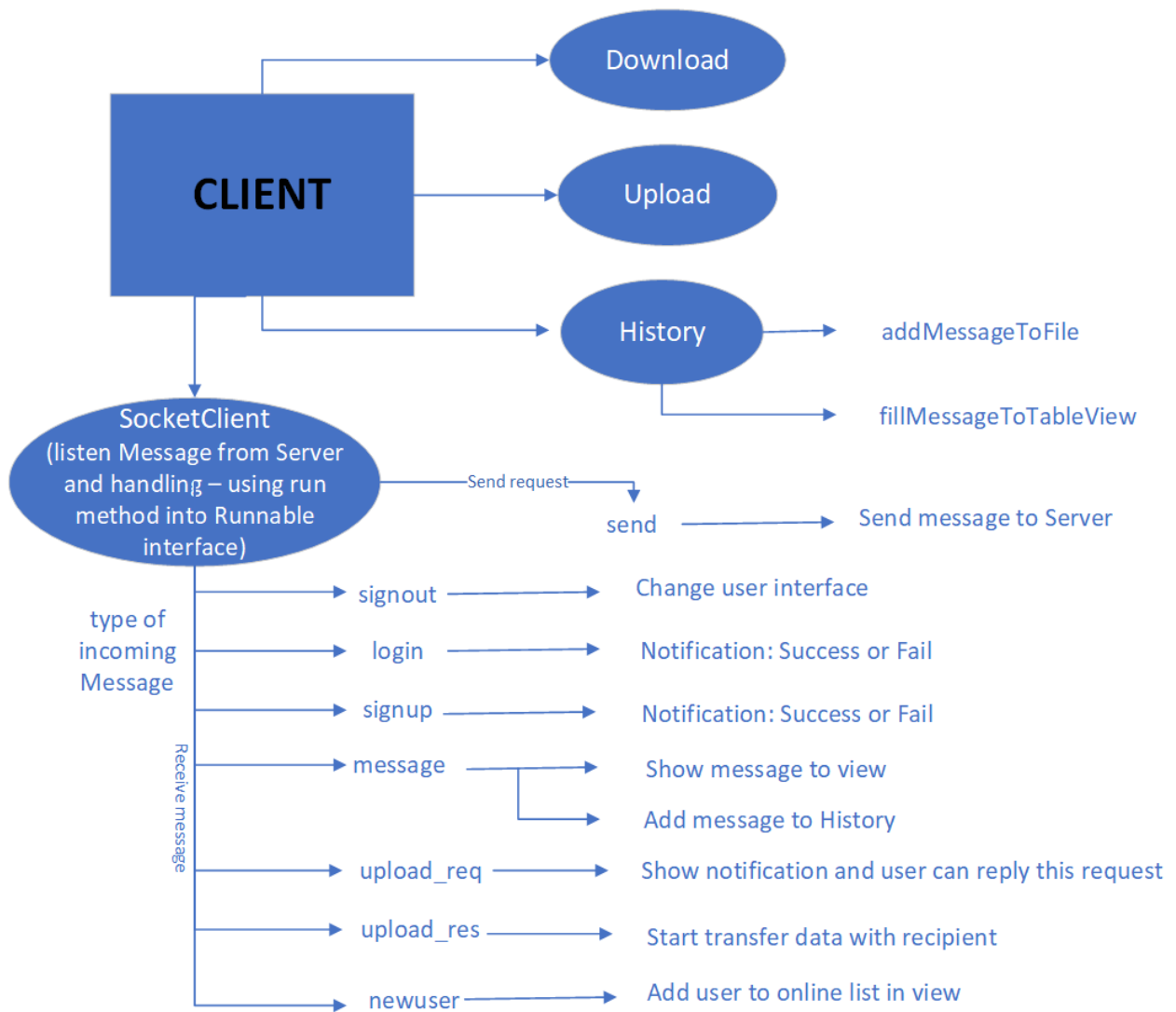
Hình 1.4 Sơ đồ truyền tin download, upload và transfer file



## 6. Sơ đồ mô tả chức năng của ứng dụng



Hình 1.5 Sơ đồ mô tả chức năng của Server



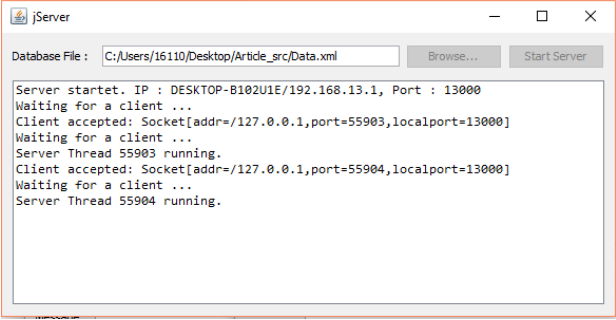
Hình 1.6 Sơ đồ mô tả chức năng của Client

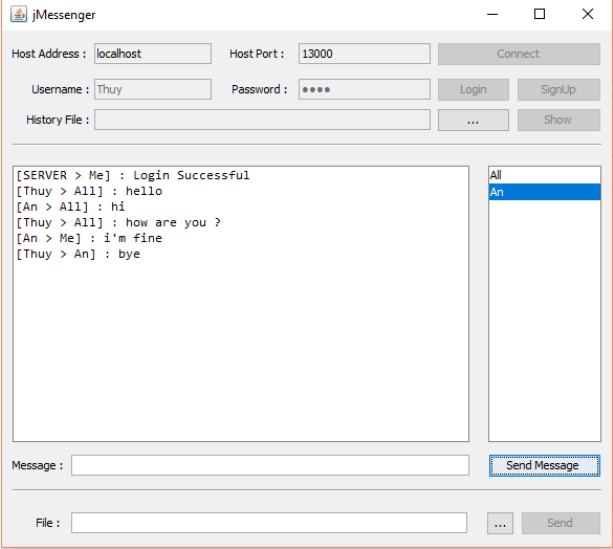
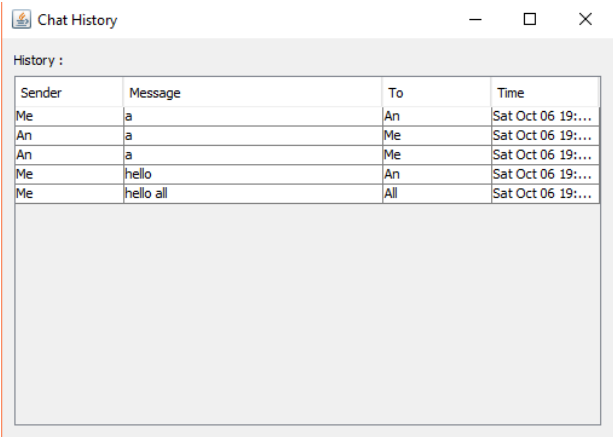
## II. Quá trình thực hiện

### 1. Thiết kế giao diện

Bảng 2.1 Thiết kế giao diện

TT	Màn hình/Cửa sổ/Dialog	Người thiết kế & giải thích ngắn gọn	Mục đích chính của màn hình

1	<p>Màn hình cho Server</p>  <p><i>Hình 2.1 Màn hình Server</i></p>	<p>Ngô Công An</p> <ul style="list-style-type: none"> <li>- Button chọn đường dẫn tới file chứa dữ liệu người dùng (*.xml) bao gồm tài khoản và mật khẩu người dùng</li> <li>- Button chạy Server</li> <li>- Panel thể hiện các trạng thái của chương trình (trạng thái kết nối, client đăng nhập, ...).</li> </ul>	<p>Tạo giao diện đơn giản cho người quản lý server và thể hiện được các trạng thái của server khi đang chạy.</p>
2	<p>Màn hình chat cho Client</p>	<p>Đào Xuân Thủy</p> <ul style="list-style-type: none"> <li>- Cho phép người dùng nhập địa chỉ máy chủ, cổng kết nối và có thể kết nối tới Server.</li> <li>- Đăng nhập bằng tài khoản cá nhân hoặc đăng ký tài khoản.</li> </ul>	<p>Tạo một giao diện chat đơn giản, dễ sử dụng, gồm đầy đủ các tính năng cần thiết và dễ dàng mở rộng chương trình.</p>

	 <p style="text-align: center;"><i>Hình 2.2 Màn hình Client</i></p>	<ul style="list-style-type: none"> <li>- Chọn file dùng để lưu lại lịch sử chat của mình.</li> <li>- Button xem lại lịch sử chat.</li> <li>- Panel giao diện chat.</li> <li>- Cửa sổ chọn người nhận (chat với tất cả hoặc một cá nhân).</li> <li>- Khung nhập tin nhắn để gửi đi.</li> <li>- Chọn file để gửi file cho người nhận.</li> </ul>	
3	<p>Màn hình hiển thị lịch sử chat</p>  <p style="text-align: center;"><i>Hình 2.3 Màn hình History</i></p>	<p>Đào Xuân Thủy</p> <ul style="list-style-type: none"> <li>- Tạo một khung hiển thị nội dung của lịch sử tin nhắn gồm: Người gửi, người nhận, nội dung tin nhắn và thời gian gửi tin.</li> </ul>	<p>Xem lịch sử chat của khách hàng khi ấn vào button “Show” tại cửa sổ chat.</p>

## 2. Thiết kế lớp

### 2.1 Thiết kế lớp cho Server

#### 2.1.1 Các lớp được sử dụng cho Server

Bảng 2.2 Danh mục các lớp được sử dụng trong chương trình Server

Người thực hiện: Ngô Công An

TT	Tên Lớp	Mục đích thiết kế
1	Message	<ul style="list-style-type: none"><li>- Chứa thông tin cần thiết của một tin nhắn được gửi đi hoặc nhận về bao gồm:<ul style="list-style-type: none"><li>+ Type</li><li>+ Sender</li><li>+ Content</li><li>+ Recipient</li></ul></li><li>- Truyền yêu cầu kiểu Message từ giao diện vào đối tượng SocketClient để thực thi chức năng cho chương trình (đăng nhập, đăng ký, upload, ...).</li></ul>
2	Database	<ul style="list-style-type: none"><li>- Lấy thuộc tính filePath.</li><li>- Mục đích của đối tượng là kiểm tra đăng nhập và tạo tài khoản.</li></ul>
3	ServerThread	<ul style="list-style-type: none"><li>- Lấy những thuộc tính như:<ul style="list-style-type: none"><li>+ SocketServer</li><li>+ socket</li><li>+ ObjectInputStream</li></ul></li></ul>

		<ul style="list-style-type: none"> <li>+ ObjectOutputStream</li> <li>+ ServerFrame</li> <li>- Mục đích là tạo Thread để xử lí nhận, gửi tin nhắn.</li> </ul>
4	SocketServer	<ul style="list-style-type: none"> <li>- Lấy những thuộc tính như:</li> <li>+ ServerThread</li> <li>+ Thread</li> <li>+ ServerFrame</li> <li>+ Database</li> <li>- Mục đích là nơi tạo kết nối cho các client, tạo và đóng thread, nhận và gửi các tin nhắn của các client</li> </ul>
5	ServerFrame	<ul style="list-style-type: none"> <li>- Lấy những thuộc tính như:</li> <li>+ SocketServer</li> <li>+ Thread</li> <li>+ filePath</li> <li>+ JfileChoose</li> <li>- Mục đích là xây dựng giao diện server (jTextFiled, jButton, jTextArea, jScrollPane, jLabel).</li> </ul>

### 2.1.2 Mô tả các phương thức của Server

Người thực hiện: Ngô Công An

Bảng 2.3 Bảng mô tả các phương thức trong lớp Message

TT	Phương thức	Mục đích	Tên file, dòng chứa khai báo
----	-------------	----------	------------------------------

1	<ul style="list-style-type: none"> <li>- Tên Phương thức: Message</li> <li>- Input: Type, Sender, Content, Recipient</li> <li>- Output: Không có</li> <li>Mã giả: Không có vì đơn giản</li> </ul>	Tạo các đối tượng của một tin nhắn (type, sender, content, recipient)	SocketServer.java (46,260)
---	---	---	----------------------------

Bảng 2.4 Bảng mô tả các phương thức trong lớp Database

TT	Phương thức	Mục đích	Tên file, dòng chứa khai báo
1	<ul style="list-style-type: none"> <li>- Tên phương thức: Database</li> <li>- Input: filePath</li> <li>- Output: Không có</li> <li>- Mã giả: Không có vì đơn giản</li> </ul>	Tạo ra đối tượng filePath	SocketServer.java (86,106)
2	<ul style="list-style-type: none"> <li>- Tên phương thức: userExit</li> <li>- Input: username</li> <li>- Output: True or false</li> <li>- Mã giả: If (getTagValue ("username",eElement).equal(username) {return true } return false</li> </ul>	Kiểm tra user đang được tạo có bị trùng với các user đã được tạo trong filePath	Database.java (dòng 51) SocketServer.java (dòng 214)
3	<ul style="list-style-type: none"> <li>- Tên phương thức: CheckLogin</li> <li>- Input: username, password</li> <li>- Output: True or false</li> </ul>	Kiểm tra user và password lúc đăng nhập có	SocketServer.java (dòng 177)

	- Mã giả: If(!userExit(username)) return false Else{return true }	đúng với giá trị trong data	
4	- Tên phương thức: addUser - Input: username, password - Output: Không có - Mã giả: Không có vì đơn giản	Tạo newuser và newpassword sau đó lưu vào filePath	SocketServer.java (dòng 216)
5	- Tên phương thức: getTagValue - Input: sTag, eElement - Output: không có - Mã giả: không có vì đơn giản	Lấy các tài khoản có trong filePath để đăng nhập	Database.java (36, 66)

Bảng 2.5 Bảng mô tả các phương thức trong lớp ServerThread

TT	Phương thức	Mục đích	Tên file, dòng chứa khai báo
1	- Tên phương thức: send - Input: Msg - Output: không có - Mã giả: không có vì đơn giản	Gửi đi tin nhắn giữa các client	SocketServer.java (dòng 180, 186, 192, 208, 225, 237, 270)
2	- Tên phương thức: GetID - Input: không có	Trả về giá trị ID vừa gửi message	SocketServer.java (dòng 156)



	<ul style="list-style-type: none"> <li>- Output: ID</li> <li>- Mã giả: Output ID</li> </ul>		
3	<ul style="list-style-type: none"> <li>- Tên phương thức: run (kế thừa từ interface Runnable)</li> <li>- Input: không có</li> <li>- Output: không có</li> <li>- Mã giả: while(true) client msg=(client)streamIn.readObject</li> </ul>	Chạy thread cho hiển thị client đang hoạt động	Vì là hàm kế thừa từ interface Runnable nên được tạo và kích hoạt chạy bên trong luồng chính và được chạy song song với luồng chính
4	<ul style="list-style-type: none"> <li>- Tên phương thức: open</li> <li>- Input: không có</li> <li>- Output: không có</li> <li>- Mã giả: không có vì đơn giản</li> </ul>	Tạo môi trường để các client hoạt động	SocketServer.java (dòng 319)
5	<ul style="list-style-type: none"> <li>- Tên phương thức: close</li> <li>- Input: không có</li> <li>- Output: không có</li> <li>- Mã giả: không có vì đơn giản</li> </ul>	Đóng thread làm việc khi client ngưng hoạt động	SocketServer.java (dòng 301)

Bảng 2.6 Bảng mô tả các phương thức trong lớp ServerSocket

TT	Phương thức	Mục đích	Tên file, dòng chứa khai báo
----	-------------	----------	------------------------------

1	<ul style="list-style-type: none"> <li>- Tên phương thức: SocketServer</li> <li>- Input: Frame</li> <li>- Output: không có</li> <li>- Mã giả: <pre>if(port=server.getLocalPort) {start} else {RestryStart}</pre> </li> </ul>	Tạo server socket, tạo số thread có thể hoạt động	ServerFrame.java (dòng 101, dòng 112)  SocketServer.java (dòng 90, dòng 110)
2	<ul style="list-style-type: none"> <li>- Tên phương thức: run (kế thừa từ interface Runnable)</li> <li>- Input: không có</li> <li>- Output: không có</li> <li>- Mã giả: while(thread!=null) addThread</li> </ul>	Chạy server và chờ các kết nối từ client	Vì là hàm kế thừa từ interface Runnable nên được tạo và kích hoạt chạy bên trong luồng chính và được chạy song song với luồng chính
3	<ul style="list-style-type: none"> <li>- Tên phương thức: start</li> <li>- Input: không có</li> <li>- Output: không có</li> <li>- Mã giả: if(thread==null) {start}</li> </ul>	Tạo thread cho các client hoạt động	SocketServer.java (dòng 93,113,320)
4	<ul style="list-style-type: none"> <li>- Tên phương thức: stop</li> <li>- Input: không có</li> <li>- Output: không có</li> <li>- Mã giả: if(thread==null) {stop}</li> </ul>	Ngắt thread	SocketServer.java (dòng 53, 307)  ServerFrame.java (dòng 112)

5	<ul style="list-style-type: none"> <li>- Tên phương thức: findClient</li> <li>- Input: ID</li> <li>- Output: trả về giá trị ID</li> <li>- Mã giả: không có vì đơn giản</li> </ul>	Tìm các client và kiểm tra xem client có trong dữ liệu	Socketserver.java (dòng 179, 186, 191, 203, 208, 217, 225, 237)
6	<ul style="list-style-type: none"> <li>- Tên phương thức: handle</li> <li>- Input: ID, msg</li> <li>- Output: không có</li> <li>- Mã giả: không có vì đơn giản</li> </ul>	Kiểm tra khi client kết nối nếu đúng tạo thread cho client hoạt động, nhận và gửi đi các tin nhắn của client trong thread (client tới client hoặc client tới tất cả)	SocketServer.java (dòng 47)
7	<ul style="list-style-type: none"> <li>- Tên phương thức: Announce</li> <li>- Input: type, sender, content</li> <li>- Output: không có</li> <li>- Mã giả: không có vì đơn giản</li> </ul>	Gửi tin nhắn đến tất cả client trong thread	SocketServer.java (dòng 167, 176, 189, 206)
8	<ul style="list-style-type: none"> <li>- Tên phương thức: findUserThread</li> <li>- Input: usr</li> <li>- Output: Vị trí của client trong Thread</li> <li>- Mã giả: If(client[i].username.equal(usr)) thì trả về giá trị của client</li> </ul>	Tìm client trong thread	SocketServer.java (dòng 175, 212, 241, 270)

9	<ul style="list-style-type: none"> <li>- Tên phương thức: remove</li> <li>- Input: ID</li> <li>- Output: không có</li> <li>- Mã giả: không có vì đơn giản</li> </ul>	Xóa client khỏi thread và đóng thread	SocketServer.java (dòng 52, 168)
10	<ul style="list-style-type: none"> <li>- Tên phương thức: addThread</li> <li>- Input: Socket</li> <li>- Output: không có</li> <li>- Mã giả: if(clientCount&lt;clients.length) thì cho client kết nối. Ngược lại thì không cho kết nối</li> </ul>	Khi có client kết nối sẽ cho kiểm tra và tạo ra thread để client hoạt động	SocketServer.java (dòng 127)

Bảng 2.7 Bảng mô tả các phương thức trong lớp ServerFrame

TT	Phương thức	Mục đích	Tên file, dòng chứa khai báo
1	<ul style="list-style-type: none"> <li>- Tên phương thức: ServerFrame</li> <li>- Input: không có</li> <li>- Output: không có</li> <li>- Mã giả: không có vì đơn giản</li> </ul>	Tạo giao diện server	ServerFrame.java (dòng 149)
2	<ul style="list-style-type: none"> <li>- Tên phương thức: initComponents</li> <li>- Input: không có</li> <li>- Output: không có</li> </ul>	Tạo button bắt sự kiện như Browse, Start server	ServerFrame.java (dòng 19)

	- Mã giả: input (jbutton, jtext, jlabel, jScrollPane)		
3	- Tên phương thức: jButton1ActionPerformed - Input: evt - Output: không có - Mã giả: server = new SocketServer(this) sau đó cho tắt các nút thao tác	Khi kết nối server thành công thì cho ẩn hai button Browse và Start server	ServerFrame.java (dòng 49)
4	- Tên phương thức: RetryStart - Input: port - Output: không có - Mã giả: if(server!=null) {stop}, else {server=new SocketServer(this,port)}	Nếu chưa có server thì start server mới	SocketServer.java (dòng 98, 132)
5	- Tên phương thức: jButton2ActionPerformed - Input: evt - Output: không có - Mã giả: if(file!=null){filePath=file.getPath}	Nếu chưa chọn filePath thì ẩn nút Start server	ServerFrame.java (dòng 64)

## 2.2 Thiết kế lớp cho Client

### 2.2.1 Các lớp được sử dụng cho Client

Bảng 2.8 Danh mục các lớp được sử dụng trong chương trình cho Client

Người thực hiện: Đào Xuân Thủy

TT	Tên Lớp	Mục đích thiết kế
1	Message	<ul style="list-style-type: none"> <li>- Chứa những thông tin cần thiết của một tin nhắn được gửi đi hoặc nhận về, bao gồm:</li> <li>+ Type</li> <li>+ Sender</li> <li>+ Content</li> <li>+ Recipient</li> <li>- Truyền yêu cầu kiểu Message từ Giao diện vào đối tượng SocketClient để thực thi chức năng cho chương trình (đăng nhập, đăng ký, upload, ... ).</li> </ul>
2	Download	<ul style="list-style-type: none"> <li>- Lấy những thuộc tính từ giao diện chat như:</li> <li>+ ServerSocket</li> <li>+ port</li> <li>+ Socket</li> <li>+ Các thuộc tính khác: địa chỉ lưu file, FileOutputStream, InputStream lấy từ socket.</li> <li>- Mục đích của đối tượng dùng để thực hiện quá trình tải một tệp từ giao diện chat về máy.</li> </ul>
3	Upload	<ul style="list-style-type: none"> <li>- Lấy những thuộc tính như:</li> <li>+ Địa chỉ máy chủ</li> <li>+ port</li> </ul>

		<ul style="list-style-type: none"> <li>+ File</li> <li>+ Đối tượng Giao diện chat dùng để thay đổi giao diện sau khi chạy code.</li> <li>- Mục đích của đối tượng dùng để thực hiện quá trình tải lên một tệp từ máy cá nhân lên server.</li> </ul>
4	History	<ul style="list-style-type: none"> <li>- Quản lý lịch sử chat sẽ thực hiện các chức năng: thêm một Message vào file XML dùng để lưu trữ và diễn dữ liệu vào giao diện hiển thị lịch sử.</li> </ul>
5	SocketClient	<ul style="list-style-type: none"> <li>- Lấy những thuộc tính từ giao diện chat như: <ul style="list-style-type: none"> <li>+ Port</li> <li>+ Socket</li> <li>+ Địa chỉ máy chủ</li> <li>+ History</li> <li>+ Các thuộc tính khác: FileOutputStream, InputStream lấy từ socket.</li> </ul> </li> <li>- Quản lý các chức năng của giao diện chat: gửi tin nhắn, nhận tin nhắn, đăng nhập, kết nối tới server, đăng ký, đăng xuất, download/upload file.</li> </ul>

### 2.2.2 Mô tả các phương thức của Client

Người thực hiện: Đào Xuân Thủy

Bảng 2.9 Bảng mô tả các phương thức trong lớp Message cho Client

TT	Phương thức	Mục đích	Tên file, dòng chứa khai báo
----	-------------	----------	------------------------------

1	<ul style="list-style-type: none"> <li>- Tên phương thức: toString</li> <li>- Input: không có</li> <li>- Output: string</li> <li>- Mã giả: không có vì đơn giản</li> </ul>	Trả về một chuỗi kiểu String chứa các thông tin của một tin nhắn, bao gồm: người gửi, người nhận, nội dung và thời gian gửi/nhận.	SocketClient.java (dòng 41, 187)
---	--	---	----------------------------------

Bảng 2.10 Bảng mô tả các phương thức trong lớp Download cho Client

TT	Phương thức	Mục đích	Tên file, dòng chứa khai báo
1	<ul style="list-style-type: none"> <li>- Tên phương thức: Download</li> <li>- Input: String, ChatFrame</li> <li>- Output: không có</li> <li>- Mã giả: Input: (saveTo, ui)</li> </ul> <pre>Server= new ServerSocket (port 0);</pre> <pre>Port = server.getPort;</pre> <pre>This.ui = ui;</pre>	Khởi tạo một đối tượng Download, lấy về đường dẫn lưu file tải về và các thuộc tính từ Giao diện chat.	SocketClient.java (dòng 132)
2	<ul style="list-style-type: none"> <li>- Tên phương thức: Run (kế thừa từ interface Runnable)</li> <li>- Input: không có</li> <li>- Output: không có</li> <li>- Mã giả: không có vì đơn giản</li> </ul>	Thực hiện lưu tệp tải từ màn hình chat về máy.	Vì là hàm kế thừa từ interface Runnable nên được tạo và kích hoạt chạy bên trong luồng chính và được chạy



			song song với luồng chính
--	--	--	------------------------------

Bảng 2.11 Bảng mô tả các phương thức trong lớp Upload cho Client

TT	Phương thức	Mục đích	Tên file, dòng chứa khai báo
1	<ul style="list-style-type: none"> <li>- Tên phương thức: upload (hàm khởi tạo)</li> <li>- Input: Stringaddr, port, filePath</li> <li>- Output: không có</li> <li>- Mã giả: không có vì đơn giản</li> </ul>	Khởi tạo một đối tượng Upload, lấy về File, giao diện (dùng để thay đổi giao diện), tạo một socket bởi địa chỉ máy chủ, và cổng port, Input/Output Stream.	SocketClient.java (dòng 152)
2	<ul style="list-style-type: none"> <li>- Tên phương thức: run (kế thừa từ interface Runnable)</li> <li>- Input: không có</li> <li>- Output: không có</li> <li>- Mã giả: tạo một buffer kiểu byte [1024];</li> </ul> <p>Đọc dữ liệu từ bufer và dùng OutputStream để ghi; Thông báo upload thành công trên textFiled; Đóng kết nối Input/Output</p>	Thực hiện tải lên server tệp từ máy khách hàng.	Vì là hàm kế thừa từ interface Runnable nên được tạo và kích hoạt chạy bên trong luồng chính và được chạy song song với luồng chính.

Bảng 2.12 Bảng mô tả các phương thức trong lớp History cho Client

TT	Phương thức	Mục đích	Tên file, dòng chứa khai báo
1	<ul style="list-style-type: none"> <li>- Tên phương thức: history (hàm khởi tạo)</li> <li>- Input: String filePath</li> <li>- Output: không có</li> <li>- Mã giả: Input: (filePath) gán filePath cho thuộc tính filePath của đối tượng history</li> </ul>	Khởi tạo một đối tượng History, lấy về địa chỉ lưu file History.xml (lưu trữ lịch sử chat)	ChatFrame.java (dòng 50, 387)
2	<ul style="list-style-type: none"> <li>- Tên phương thức: addMessage</li> <li>- Input: String time, message msg</li> <li>- Output: không có</li> <li>- Mã giả: Tạo mới DocumentBuilderFactory, DocumentBuilder, Document (prase(filePath))</li> <li>+ Tạo Element message (Element message, Element sender, Element content, Element recipient, Element time)</li> </ul>	Thêm một Message vào file History.xml để lưu trữ tin nhắn	SocketClient.java (dòng 54, 192)

	<ul style="list-style-type: none"> <li>+ Nối Element message vào Node data</li> <li>+ Tạo mới TransformerFactory, Transformer, DOMSource, StreamResult save file *.xml (file lưu lịch sử chat).</li> </ul>		
3	<ul style="list-style-type: none"> <li>- Tên phương thức: FillTable</li> <li>- Input: HistoryFrame</li> <li>- Output: không có</li> <li>- Mã giả: Tạo một DefaultTableModel lấy từ bảng hiển thị trên giao diện xem lịch sử.</li> <li>+ Tạo mới DocumentBuilderFactory, DocumentBuilder, Document (prase.( fXmlFile))</li> <li>+ Tạo một NodeList lấy Element có tên message từ Document</li> <li>+ Dùng vòng lặp đọc các Node trong NodeList và thêm thông tin các tin nhắn đó vào DefaultTableModel</li> </ul>	Điền dữ liệu vào bảng hiển thị trên giao diện HistoryFrame	HistoryFrame.java (dòng 20)
4	<ul style="list-style-type: none"> <li>- Tên phương thức: getTagValue</li> <li>- Input: String sTag, eElement</li> </ul>	Lấy giá trị của Tag theo tên Tag	History.java (dòng 75, 76)

	<ul style="list-style-type: none"> <li>- Output: String</li> <li>- Mã giả: Tạo NodeList lấy tất cả các Element con theo tên Tag</li> <li>+ Tạo Node lấy giá trị đầu tiên của NodeList</li> <li>+ Trả về giá trị của Node</li> </ul>		
--	---	--	--

Bảng 2.13 Bảng mô tả các phương thức trong lớp SocketClient cho Client

TT	Phương thức	Mục đích	Tên file, dòng chứa khai báo
1	<ul style="list-style-type: none"> <li>- Tên phương thức: SocketClient (hàm khởi tạo)</li> <li>- Input: ChatFrame</li> <li>- Output: không có</li> <li>- Mã giả: input (ui) this.ui = ui;</li> <li>+ Lấy địa chỉ máy chủ; port từ ui;</li> <li>+ Create socket từ port;</li> <li>+ Input/Output Stream từ Socket;</li> <li>+ Lấy history từ ui</li> </ul>	<p>Khởi tạo một đối tượng SocketClient, lấy về giao diện của chương trình (địa chỉ máy chủ, cổng kết nối, History) đồng thời từ đó tạo ra Socket và ObjectInputStream, ObjectOutputStream</p>	ChatFrame.java (dòng 302)
2	<ul style="list-style-type: none"> <li>- Tên phương thức: run (kế thừa từ interface Runnable)</li> <li>- Input: không có</li> <li>- Output: không có</li> </ul>	<p>Bao gồm các chức năng: Nhận tin nhắn, đăng nhập, kết nối tới server,</p>	Vì là hàm kế thừa từ interface Runnable nên

<p>- Mã giả:</p> <pre> + if (type = message) print (sender &gt; Me    sender &gt; recipient), add History.  + if (type = login) { if (content = TRUE) Enable button và textfield,“Login successfull”; else “Login failed” }  + if (type = newuser) if(username=username) else add new user.  + if (type = signup) { if (content = TRUE) Enable button, print “Signup success” else print “Signup Failed”}  + if (type = signout) print “sender &gt; Me: Bye” và stop thread  + if (type = upload_req) nếu nhận tạo mới thuộc tính Download và send (“upload_res”, “username”, “Download.port”, sender), nếu không send ("upload_res", ui.username, "NO",sender)  + if (type = upload_res) { if (content!= NO) new Upload và Thread, start thread else print “Người nhận không đồng ý nhận file” }  - catch:  + Enable = false các button và textfield </pre>	<p>đăng ký, đăng xuất, upload và download file.</p>	<p>được tạo và kích hoạt chạy bên trong luồng chính và được chạy song song với luồng chính.</p>
--	---	---

	<ul style="list-style-type: none"> <li>+ Xóa tất cả nội dung trên bảng hiển thị tin nhắn</li> <li>+ Dừng Thread mà người dùng đang chạy</li> <li>+ In ra thông báo: “Exception SocketClient run ()”</li> </ul>		
3	<ul style="list-style-type: none"> <li>- Tên phương thức: send</li> <li>- Input: message</li> <li>- Output: không có</li> <li>- Mã giả: In ra nội dung tin nhắn trong màn hình console</li> <li>+ Nếu tin nhắn có loại “message” và nội dung khác “.bye” thì thêm tin nhắn vào History và thêm nội dung tin nhắn vào bảng hiển thị của người dùng.</li> </ul>	Chức năng gửi tin nhắn tới một user khác hoặc chat tới tất cả mọi người online.	ChatFrame.java (dòng 42, 305, 319, 329, 339, 367)

### III. Phân công công việc

Bảng 3.1 Bảng mô tả phân công công việc

Sinh viên thực hiện	Phần trăm đóng góp	Mô tả khái quát mảng công việc SV thực hiện trong đồ án
Đào Xuân Thủy	50%	Thiết kế và viết chương trình cho Client
Ngô Công An	50%	Thiết kế và viết chương trình cho Server

# KẾT LUẬN

Như vậy, đồ án hiện tại đã hoàn thành được được 80% mục tiêu đề ra.

Ưu điểm:

- Tạo ra được ứng dụng thiết thực, thân thiện, có thể tạo một hoặc nhiều server với địa chỉ IP riêng dùng để chat bằng nhiều máy với nhau.

Khuyết điểm:

- Chỉ thực hiện chat được với những người đang online trên Server.
- Chưa thực hiện được chứng năng tạo nhóm chat và chat nhóm, chỉ chat riêng với 1 người hoặc gửi đi tất cả mọi người.

Khó khăn:

- Chưa có hiểu biết về SocketServer, cách thức truyền tin lên server và server phản hồi lại. Từ đó bắt đầu tìm hiểu về công nghệ này và đưa ra hướng đi cho đồ án.
- Cách thức đọc/ghi file \*.xml để lưu trữ dữ liệu. Sau đó đã tham khảo trên internet và dùng cho đồ án của mình.

Hướng đi của đồ án:

- Phát triển chức năng tạo nhóm chat riêng, chat nhóm.
- Phát triển cơ sở dữ liệu riêng dùng để quản lý người dùng, lưu trữ dữ liệu chat.
- Phát triển chứng năng người dùng có thể chat tới người nhận ngay cả khi người nhận đang offline.
- Tiếp tục phát triển chức năng gửi/nhận file để có thể tải được file có dung lượng lớn hơn.



## TÀI LIỆU THAM KHẢO

1. Lê Chí Huy, *Hướng dẫn lập trình Java Socket*, Website học lập trình trực tuyến, <https://o7planning.org/vi/10393/huong-dan-lap-trinh-java-socket>, 09/05/2016.
2. Nilesh Jadav, *How To Make A Chat Application Using Sockets In Java*, Website C# Corner, <https://www.c-sharpcorner.com/article/how-to-make-a-chat-application-using-sockets-in-java/>, 17/04/2017.
3. Nguyễn Khánh, *Đọc tập tin XML sử dụng DOM*, Website hướng dẫn java trực tuyến, <https://huongdanjava.com/vi/doc-tap-tin-xml-su-dung-dom.html>, 01/07/2016.