

MỤC LỤC

DANH MỤC CÁC BẢNG	3
I. Giới thiệu đề tài	4
1. Ngữ cảnh sử dụng phần mềm	4
2. Mô tả phần mềm	4
3. Đặc điểm, tính năng	4
4. Hướng xây dựng phần mềm.....	4
5. Sơ đồ truyền tin giữa Server và Client.....	6
6. Sơ đồ mô tả chức năng của phần mềm	9
II. Quá trình thực hiện.....	10
1. Thiết kế giao diện	10
2. Thiết kế lớp	14
2.1 Thiết kế lớp cho Server	14
2.1.1 Các lớp được sử dụng cho Server.....	14
2.1.2 Mô tả các phương thức của Server	15
2.2 Thiết kế lớp cho Client.....	22
2.2.1 Các lớp được sử dụng cho Client	22
2.2.2 Mô tả các phương thức của Client.....	24
III. Phân công công việc	32
KẾT LUẬN	33
TÀI LIỆU THAM KHẢO	34

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	13

DANH MỤC CÁC BẢNG

Bảng 2.1 Thiết kế giao diện.....	11
Bảng 2.2 Danh mục các lớp được sử dụng của Server.....	14
Bảng 2.3 Bảng mô tả các phương thức trong lớp Message.....	15
Bảng 2.4 Bảng mô tả các phương thức trong lớp Database	16
Bảng 2.5 Bảng mô tả các phương thức trong lớp ServerThread	17
Bảng 2.6 Bảng mô tả các phương thức trong lớp ServerSocket	18
Bảng 2.7 Bảng mô tả các phương thức trong lớp ServerFrame	21
Bảng 2.8 Danh mục các lớp được sử dụng của Client	22
Bảng 2.9 Bảng mô tả các phương thức trong lớp Message.....	24
Bảng 2.10 Bảng mô tả các phương thức trong lớp Download	24
Bảng 2.11 Bảng mô tả các phương thức trong lớp Upload	25
Bảng 2.12 Bảng mô tả các phương thức trong lớp History.....	26
Bảng 2.13 Bảng mô tả các phương thức trong lớp SocketClient	28
Bảng 3.1 Bảng mô tả phân công công việc	32

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 nghiên cứu và phát triển phần mềm chạy trên nền hệ điều hành này. Phần mềm 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 phần mềm

- ✓ Tên 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 phần mềm: 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 phần mềm

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

- 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 bằng 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_req**
- 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 client 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

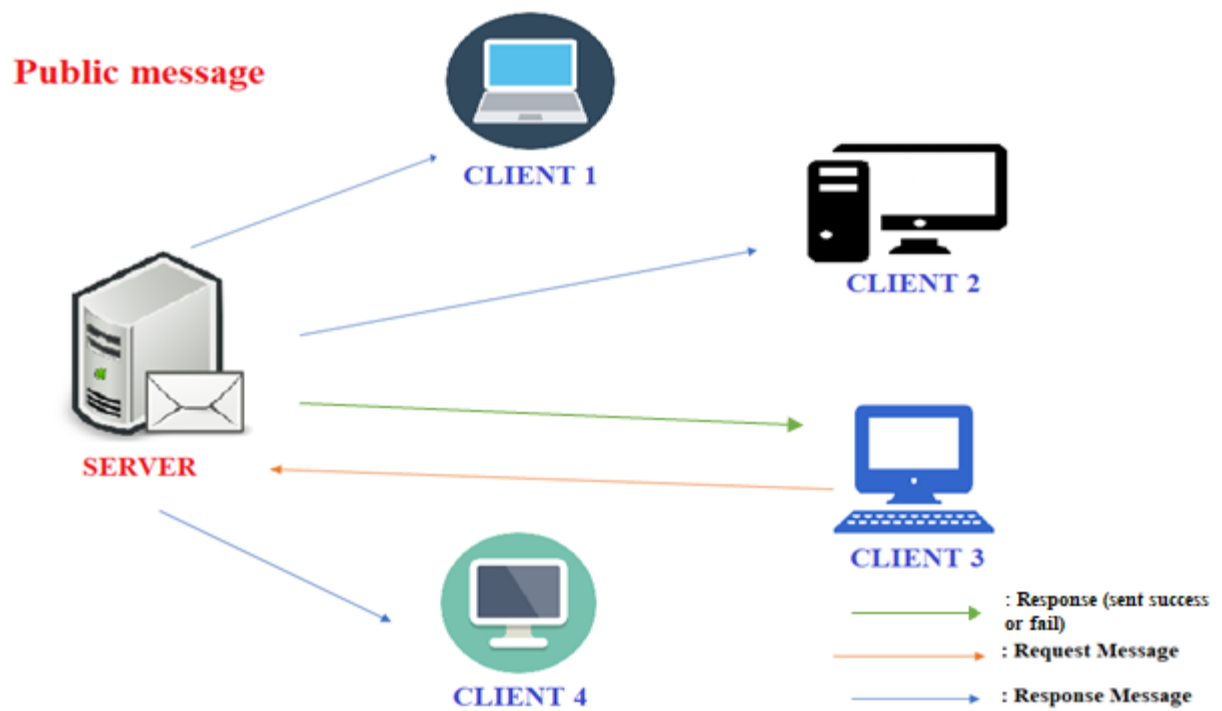
// Kết nối tới cổng mà người nhận gửi về

Upload upl = new Upload(addr, port, ui.file, ui);

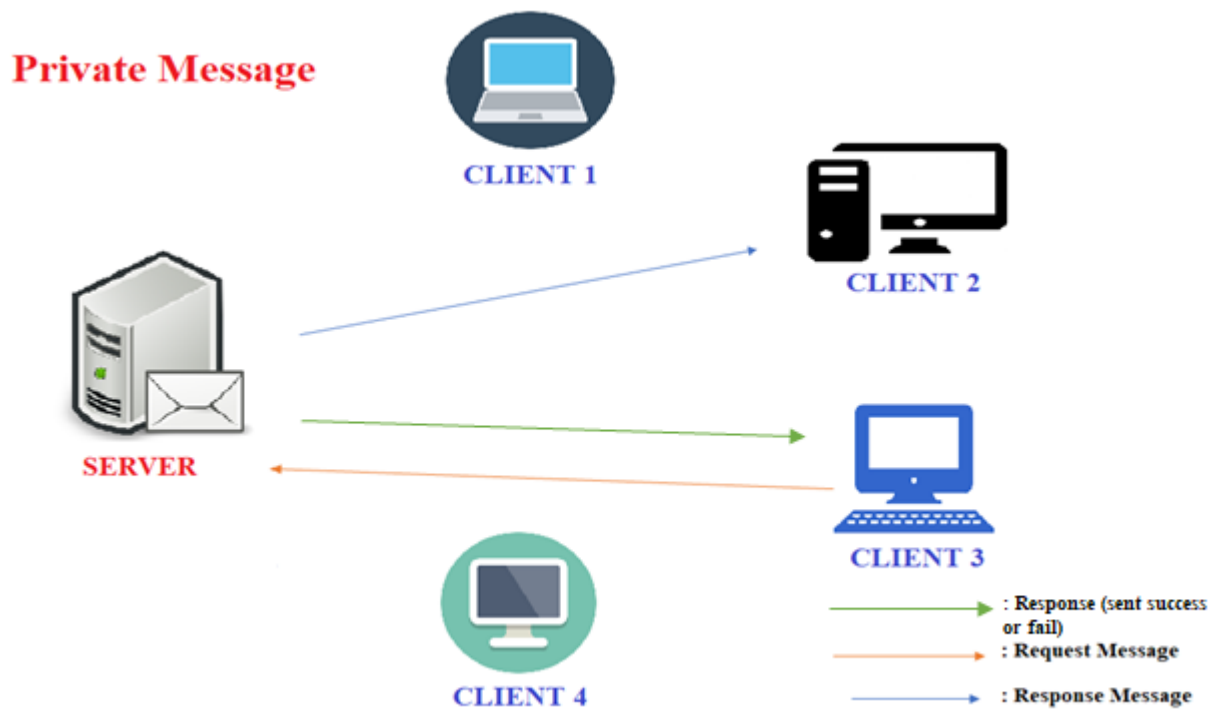
Thread t = new Thread(upl);

t.start();
```

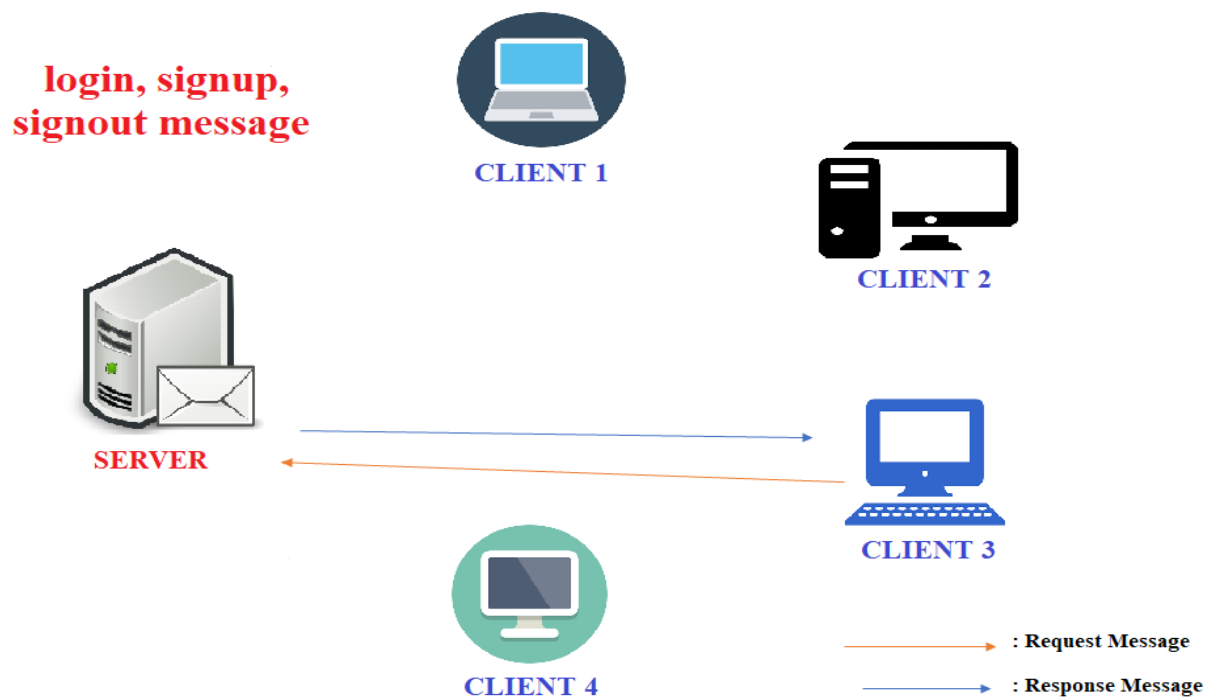
5. Sơ đồ truyền tin giữa Server và Client



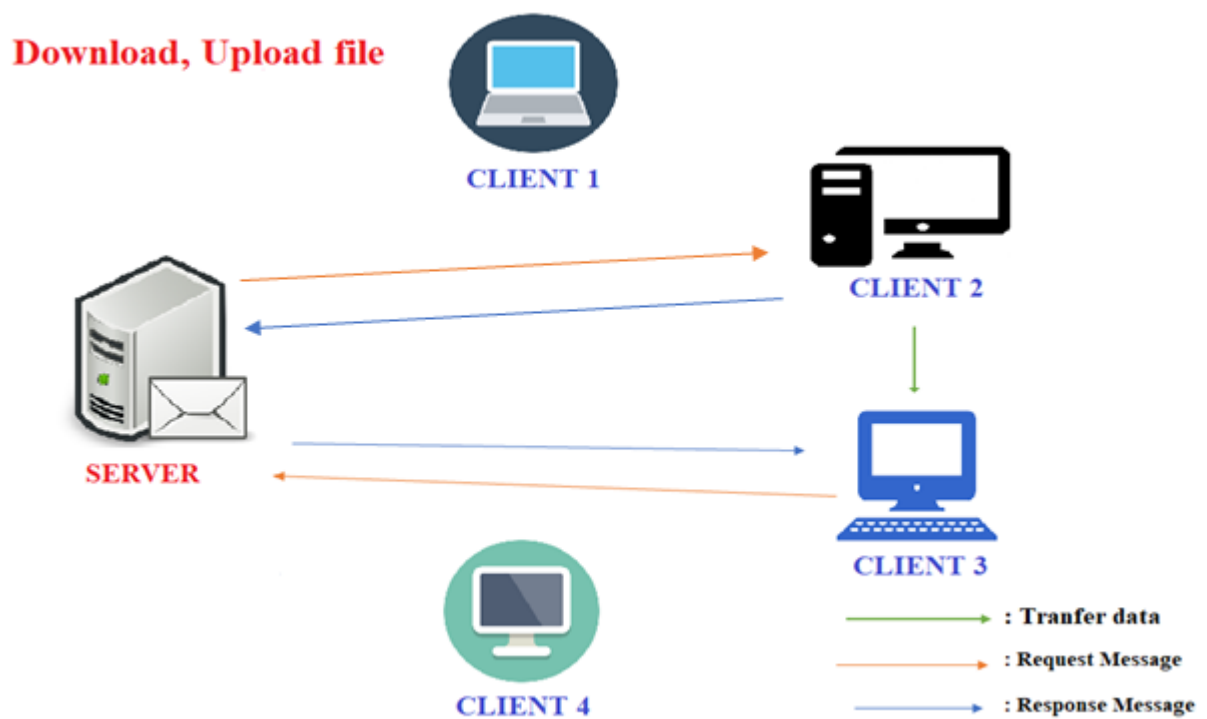
Hình 1.1 Sơ đồ truyền tin Public Message



Hình 1.2 Sơ đồ truyền tin Private Message

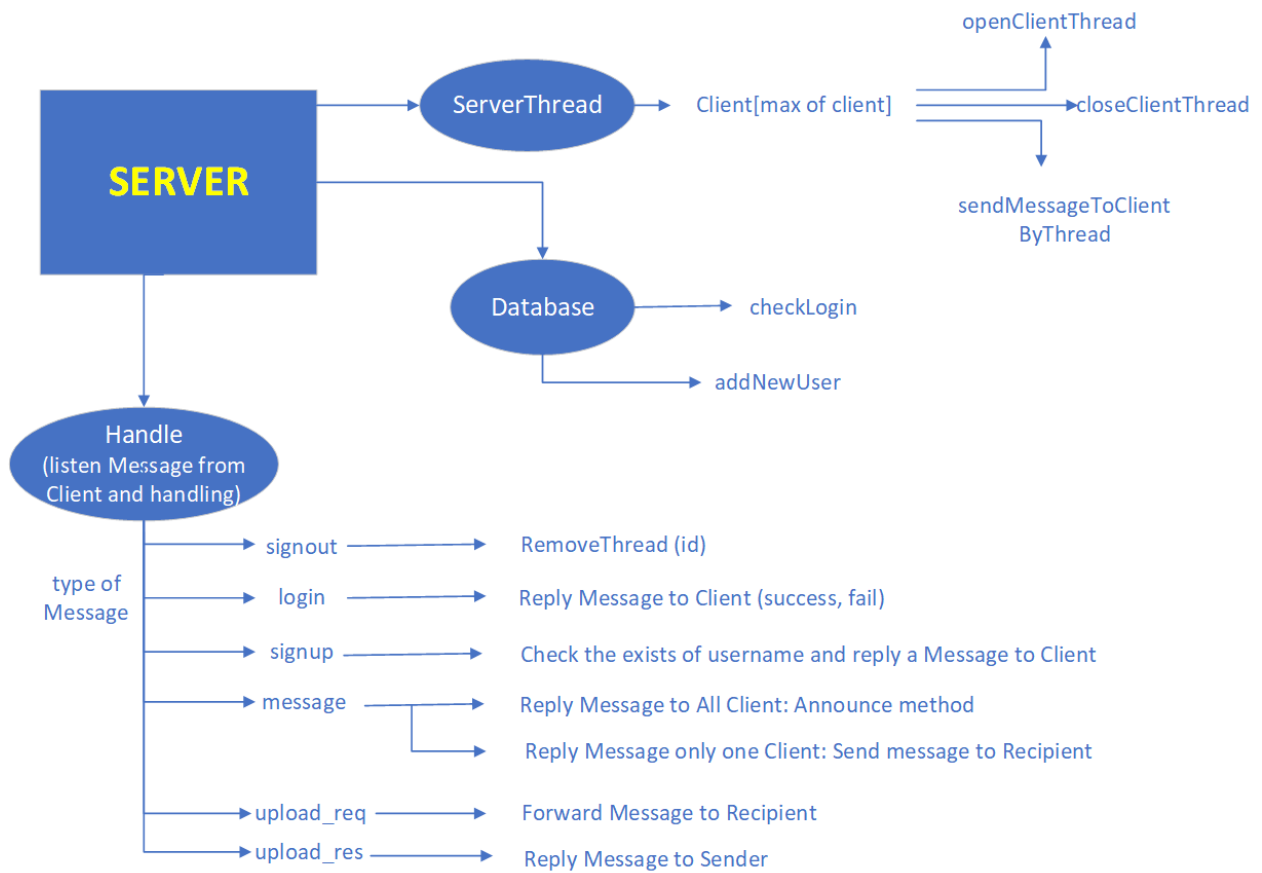


Hình 1.3 Sơ đồ truyền tin connect, login, signup, signout message

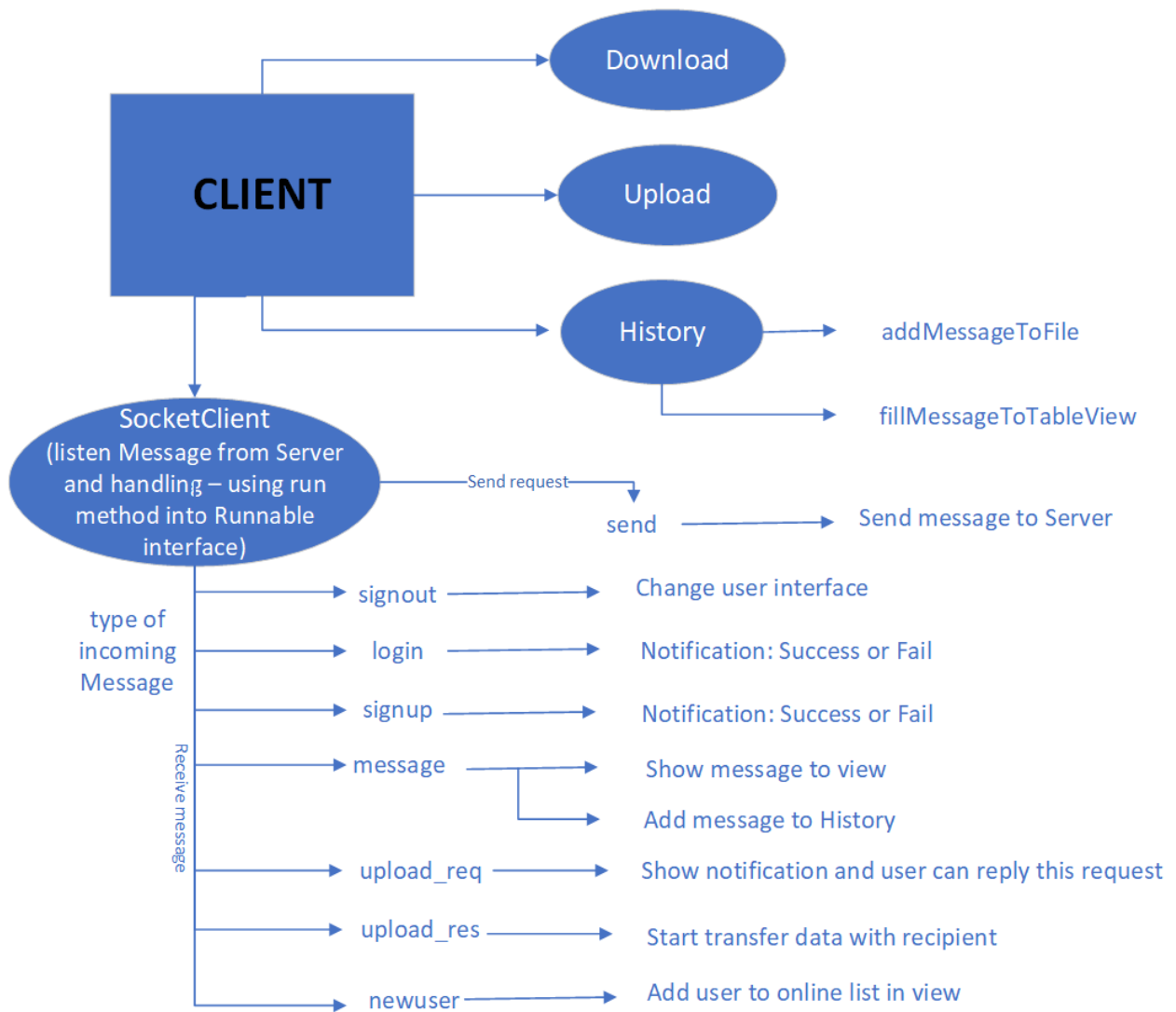


Hình 1.4 Sơ đồ truyền tin download, upload và transfer file

6. Sơ đồ mô tả chức năng của phần mềm



Hình 1.5 Sơ đồ mô tả chức năng của Server

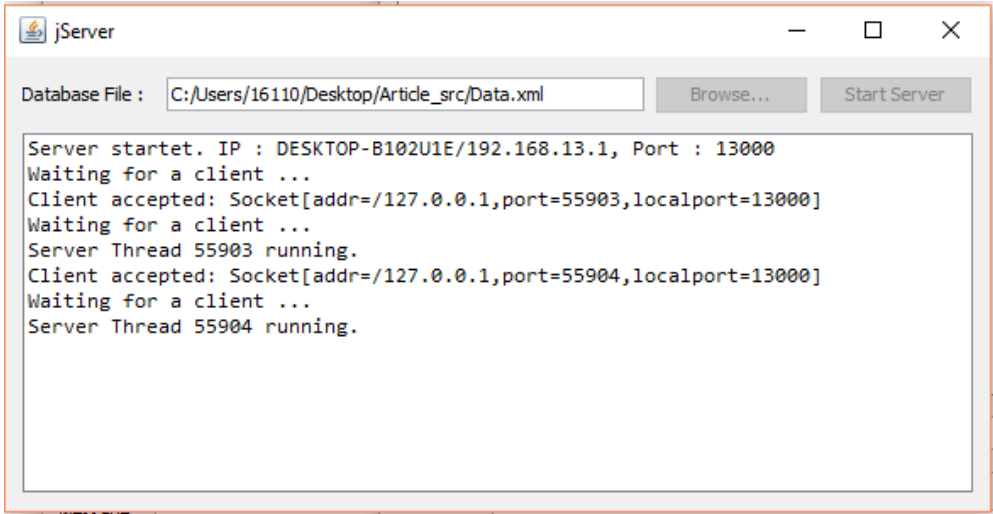


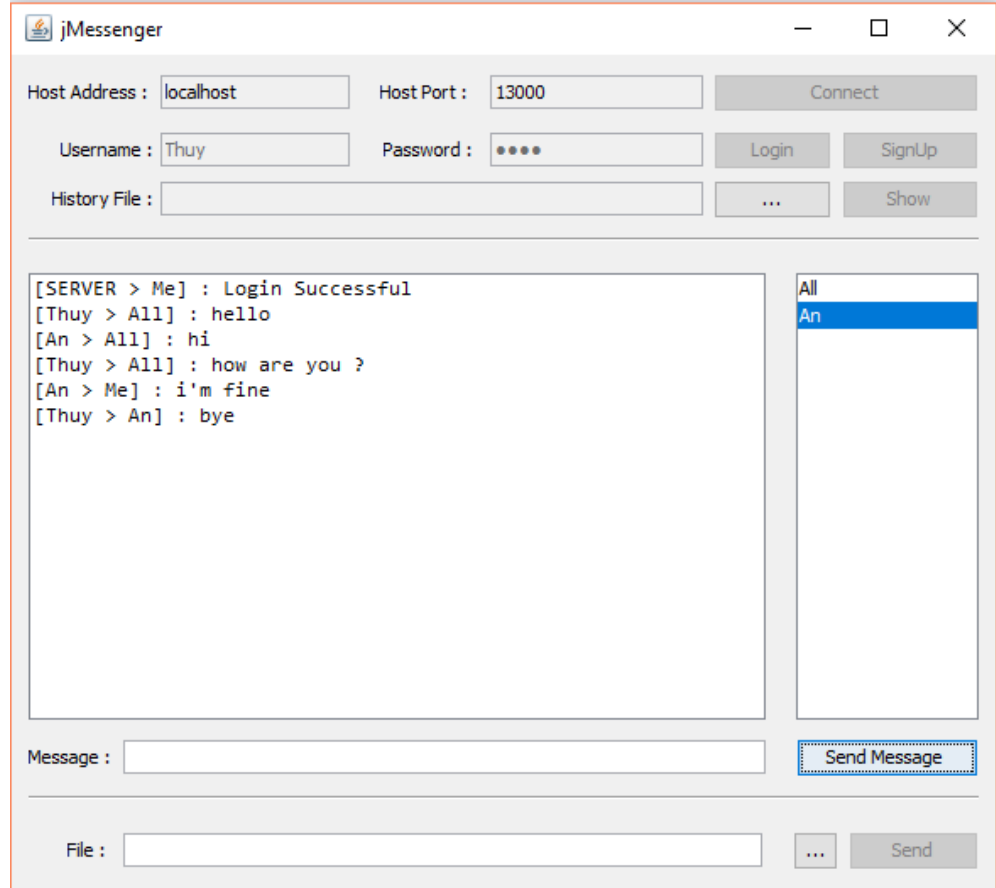
Hình 1.6 Sơ đồ mô tả chức năng của Client

II. Quá trình thực hiện

1. Thiết kế giao diện

Bảng 2.1 Thiết kế giao diện

1	Màn hình/Cửa sổ/Dialog	<p>Màn hình Server</p>  <p align="center">Hình 2.1 Màn hình Server</p>
	Người thiết kế & giải thích ngắn gọn	<p>Ngô Công An</p> <ul style="list-style-type: none"> - Button chọn đường dẫn tới file chứa dữ liệu người dùng (*.xml) bao gồm tài khoản và mật khẩu người dùng - Button chạy Server - Panel thể hiện các trạng thái của chương trình (trạng thái kết nối, client đăng nhập, ...).
	Mục đích chính của màn hình	Tạo giao diện đơn giản cho người quản lý server và thể hiện được các trạng thái của server khi đang chạy.
2	Màn hình/Cửa sổ/Dialog	Màn hình chat Client

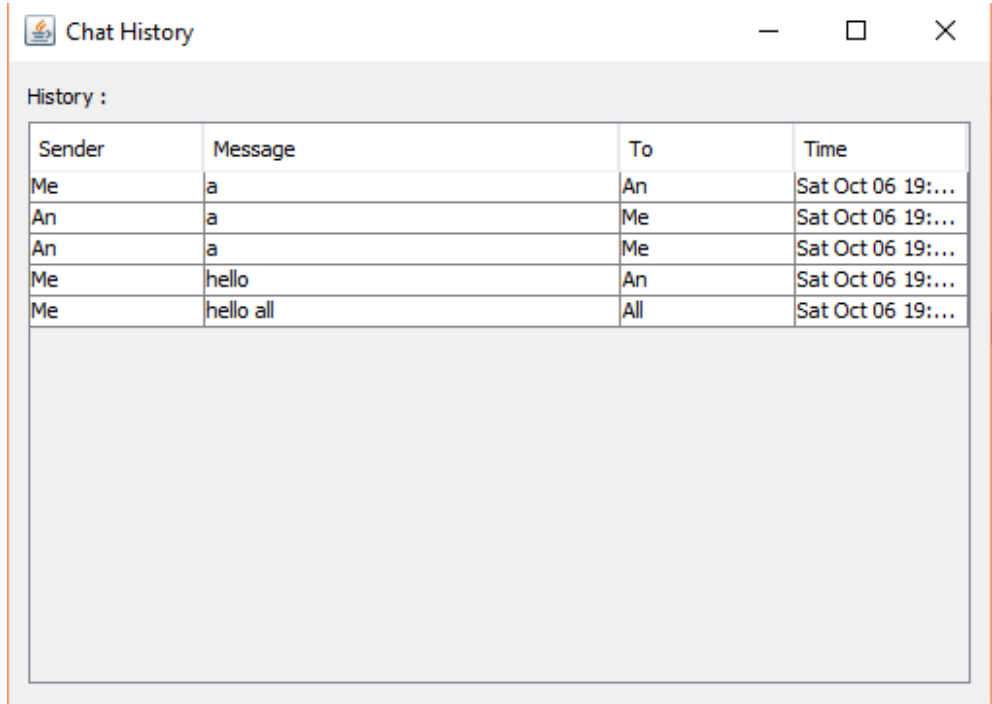


Hình 2.2 Màn hình Client

Người
thiết kế &
giải thích
ngắn gọn

Đào Xuân Thủy

- Cho phép người dùng nhập địa chỉ máy chủ, cổng kết nối và kết nối tới Server.
- Đăng nhập bằng tài khoản cá nhân hoặc đăng ký tài khoản.
- Chọn file lưu lại lịch sử chat.
- Button xem lại lịch sử chat.
- Panel giao diện chat.
- Cửa sổ chọn người nhận (chat với tất cả hoặc một cá nhân).
- Khung nhập tin nhắn để gửi đi.

		- Chọn file để gửi file cho người nhận.
	Mục đích chính của màn hình	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.
3	Màn hình/Cửa sổ/Dialog	<div>Màn hình hiển thị lịch sử chat</div> <div></div> <div>Hình 2.3 Màn hình History</div>
	Người thiết kế & giải thích ngắn gọn	<div>Đào Xuân Thủy</div> <div>- 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.</div>
	Mục đích chính của màn hình	Xem lịch sử chat của client khi ấn vào button “Show” tại cửa sổ chat.

2. Thiết kế lớp

2.1 Thiết kế lớp cho Server

2.1.1 Các lớp được sử dụng cho Server

Người thực hiện: Ngô Công An

Bảng 2.2 Danh mục các lớp được sử dụng của Server

TT	Tên Lớp	Mục đích thiết kế
1	Message	<ul style="list-style-type: none">- Chứa thông tin của một tin nhắn được gửi đi hoặc nhận về bao gồm:<ul style="list-style-type: none">+ Type+ Sender+ Content+ Recipient- 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, ...).
2	Database	Kiểm tra đăng nhập và tạo tài khoản.
3	ServerThread	<ul style="list-style-type: none">- Lấy những thuộc tính như:<ul style="list-style-type: none">+ SocketServer+ socket+ ObjectInputStream+ ObjectOutputStream+ ServerFrame- Tạo thread để xử lý nhận, gửi tin nhắn.

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

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"> - Tên Phương thức: Message - Input: Type, Sender, Content, Recipient - Output: Không có 	Tạo các đối tượng của một tin nhắn (type, sender, content, recipient)	SocketServer.java (46,260)

	Mã giả: Không có vì đơn giản		
--	------------------------------	--	--

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"> - Tên phương thức: Database - Input: filePath - Output: Không có - Mã giả: Không có vì đơn giản 	Tạo ra đối tượng filePath	SocketServer.java (86,106)
2	<ul style="list-style-type: none"> - Tên phương thức: userExit - Input: username - Output: True or false - Mã giả: If (getTagValue (“username”,eElement).equal(username) {return true } return false 	Kiểm tra user đang được tạo có bị trùng với các user đã được tạo trong file lưu dữ liệu người dùng hay kho	Database.java (dòng 51) SocketServer.java (dòng 214)
3	<ul style="list-style-type: none"> - Tên phương thức: CheckLogin - Input: username, password - Output: True or false - Mã giả: If(!userExit(username)) return false Else{return true } 	Kiểm tra user và password lúc đăng nhập có đúng với giá trị trong data	SocketServer.java (dòng 177)
4	<ul style="list-style-type: none"> - Tên phương thức: addUser - Input: username, password - Output: Không có 	Tạo newuser và newpassword sau đó lưu vào file dữ liệu	SocketServer.java (dòng 216)

	- Mã giả: Không có vì đơn giản		
5	<ul style="list-style-type: none"> - Tên phương thức: <code>getTagValue</code> - Input: <code>sTag, eElement</code> - Output: không có - Mã giả: không có vì đơn giản 	Lấy các tài khoản có trong <code>filePath</code> để đăng nhập	<i>Database.java</i> (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	<ul style="list-style-type: none"> - Tên phương thức: <code>send</code> - Input: <code>Msg</code> - Output: không có - Mã giả: không có vì đơn giản 	Gửi đi tin nhắn giữa các client	<code>SocketServer.java</code> (dòng 180, 186, 192, 208, 225, 237, 270)
2	<ul style="list-style-type: none"> - Tên phương thức: <code>GetID</code> - Input: không có - Output: ID - Mã giả: Output ID 	Trả về giá trị ID vừa gửi message	<code>SocketServer.java</code> (dòng 156)
3	<ul style="list-style-type: none"> - Tên phương thức: <code>run</code> (kế thừa từ interface <code>Runnable</code>) - Input: không có - Output: không có - Mã giả: <code>while(true) client msg=(client)streamIn.readObject</code> 	Chạy thread cho hiển thị client đang hoạt động	Vì là hàm kế thừa từ interface <code>Runnable</code> 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"> - Tên phương thức: open - Input: không có - Output: không có - Mã giả: không có vì đơn giản 	Tạo môi trường để các client hoạt động	SocketServer.java (dòng 319)
5	<ul style="list-style-type: none"> - Tên phương thức: close - Input: không có - Output: không có - Mã giả: không có vì đơn giản 	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"> - Tên phương thức: SocketServer - Input: Frame - Output: không có - Mã giả: if(port=server.getLocalPort) {start} else {RestryStart} 	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"> - Tên phương thức: run (kế thừa từ interface Runnable) - Input: không có 	Chạy server và chờ các kết nối từ client	Vì là hàm kế thừa từ interface Runnable nên được tạo và kích

	<ul style="list-style-type: none"> - Output: không có - Mã giả: while(thread!=null) addThread 		hoạt chạy bên trong luồng chính và được chạy song song với luồng chính
3	<ul style="list-style-type: none"> - Tên phương thức: start - Input: không có - Output: không có - Mã giả: if(thread==null) { start } 	Tạo thread cho các client hoạt động	SocketServer.java (dòng 93,113,320)
4	<ul style="list-style-type: none"> - Tên phương thức: stop - Input: không có - Output: không có - Mã giả: if(thread==null) { stop } 	Ngắt thread	SocketServer.java (dòng 53, 307) ServerFrame.java (dòng 112)
5	<ul style="list-style-type: none"> - Tên phương thức: findClient - Input: ID - Output: trả về giá trị ID - Mã giả: không có vì đơn giản 	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"> - Tên phương thức: handle - Input: ID, msg - Output: không có - Mã giả: không có vì đơn giản 	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	SocketServer.java (dòng 47)

		tới tất cả) và xử lý các yêu cầu của client gửi tới như: signup, signout, login, openThread...	
7	<ul style="list-style-type: none"> - Tên phương thức: Announce - Input: type, sender, content - Output: không có - Mã giả: không có vì đơn giản 	Gửi tin nhắn đến tất cả client trong thread	SocketServer.java (dòng 167, 176, 189, 206)
8	<ul style="list-style-type: none"> - Tên phương thức: findUserThread - Input: usr - Output: Vị trí của client trong Thread - Mã giả: If(client[i].username.equal(usr)) thì trả về giá trị của client 	Tìm client trong thread	SocketServer.java (dòng 175, 212, 241, 270)
9	<ul style="list-style-type: none"> - Tên phương thức: remove - Input: ID - Output: không có - Mã giả: không có vì đơn giản 	Xóa client khỏi thread và đóng thread	SocketServer.java (dòng 52, 168)
10	<ul style="list-style-type: none"> - Tên phương thức: addThread - Input: Socket - Output: không có 	Khi có client kết nối sẽ cho kiểm tra và tạo ra thread để client hoạt động	SocketServer.java (dòng 127)

	- Mã giả: if(clientCount<clients.length) thì cho client kết nối. Ngược lại thì không cho kết nối		
--	---	--	--

Bảng 2.7 Bảng mô tả các phương thức trong lớp ServerFrame

TT	Phương thức	Mục đích	Tên file, dòng chứa khai báo
1	- Tên phương thức: ServerFrame - Input: không có - Output: không có - Mã giả: không có vì đơn giản	Tạo giao diện server	ServerFrame.java (dòng 149)
2	- Tên phương thức: initComponents - Input: không có - Output: không có - Mã giả: input (jbutton, jtext, jlabel, jScrollPane)	Tạo button bắt sự kiện như Browse, Start server	ServerFrame.java (dòng 19)
3	- Tên phương thức: jButton1ActionPerformed - Input: evt - Output: không có - Mã giả: server = new SocketServer(this) sau đó cho tất các nút thao tác	Khi kết nối server thành công thì cho ẩn hai button Browse và Start server	ServerFrame.java (dòng 49)

4	<ul style="list-style-type: none"> - 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	<ul style="list-style-type: none"> - Tên phương thức: jButton2ActionPerformed - Input: evt - Output: không có - Mã giả: if(file!=null){filePath=file.getPath} 	Nếu chưa chọn filePath thì ấn nút Start server	ServerFrame.java (dòng 64)

2.2 Thiết kế lớp cho Client

2.2.1 Các lớp được sử dụng cho Client

Người thực hiện: Đào Xuân Thủy

Bảng 2.8 Danh mục các lớp được sử dụng của Client

TT	Tên Lớp	Mục đích thiết kế
1	Message	<ul style="list-style-type: none"> - Chứa những thông tin của một tin nhắn được gửi đi hoặc nhận về, bao gồm: + Type + Sender + Content + Recipient

		<ul style="list-style-type: none"> - 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, ...).
2	Download	<ul style="list-style-type: none"> - Lấy những thuộc tính từ giao diện chat như: <ul style="list-style-type: none"> + ServerSocket + port + Socket + Các thuộc tính khác: địa chỉ lưu file, FileOutputStream, InputStream lấy từ socket. - Thực hiện quá trình tải một tệp từ người gửi về máy.
3	Upload	<ul style="list-style-type: none"> - Lấy những thuộc tính như: <ul style="list-style-type: none"> + Địa chỉ máy chủ + port + File + Đối tượng Giao diện chat dùng để thay đổi giao diện sau khi chạy code. - Thực hiện quá trình tải lên một tệp từ máy cá nhân qua máy người nhận.
4	History	<ul style="list-style-type: none"> - Quản lý lịch sử chat sẽ thực hiện các chức năng: thêm một Message vào file XML dùng để lưu trữ và điền dữ liệu vào giao diện hiển thị lịch sử.
5	SocketClient	<ul style="list-style-type: none"> - Lấy những thuộc tính từ giao diện chat như: <ul style="list-style-type: none"> + Port + Socket

		+ Địa chỉ máy chủ + History + Các thuộc tính khác: FileOutputStream, InputStream lấy từ socket. - 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.
--	--	--

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

TT	Phương thức	Mục đích	Tên file, dòng chứa khai báo
1	- Tên phương thức: toString - Input: không có - Output: string - Mã giả: không có vì đơn giản	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

TT	Phương thức	Mục đích	Tên file, dòng chứa khai báo
1	- Tên phương thức: Download - Input: String, ChatFrame	Khởi tạo một đối tượng Download, lấy về đường dẫn lưu file	SocketClient.java (dòng 132)

	<ul style="list-style-type: none"> - Output: không có - Mã giả: Input: (saveTo, ui) Server= new ServerSocket (port 0); Port = server.getPort; This.ui = ui; 	tải về và các thuộc tính từ Giao diện chat.	
2	<ul style="list-style-type: none"> - Tên phương thức: Run (kế thừa từ interface Runnable) - Input: không có - Output: không có - Mã giả: không có vì đơn giản 	Thực hiện lưu tệp tải từ người gửi 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

TT	Phương thức	Mục đích	Tên file, dòng chứa khai báo
1	<ul style="list-style-type: none"> - Tên phương thức: upload (hàm khởi tạo) - Input: Stringaddr, port, filePath - Output: không có - Mã giả: không có vì đơn giản 	Khởi tạo một đối tượng Upload, lấy về đường dẫn file, giao diện (dùng để thay đổi giao diện), tạo một socket bởi địa chỉ máy	SocketClient.java (dòng 152)

		chủ, và cổng port, Input/Output Stream.	
2	<ul style="list-style-type: none"> - Tên phương thức: run (kế thừa từ interface Runnable) - Input: không có - Output: không có - Mã giả: tạo một buffer kiểu byte [1024]; Đọc dữ liệu từ bufer và dùng OutputStream để ghi; Thông báo upload thành công trên textFiled; Đóng kết nối Input/Output 	Từ máy client tải lên tệp cần chuyển.	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

TT	Phương thức	Mục đích	Tên file, dòng chứa khai báo
1	<ul style="list-style-type: none"> - Tên phương thức: history (hàm khởi tạo) - Input: String filePath - Output: không có - Mã giả: Input: (filePath) gán filePath cho thuộc tính filePath của đối tượng history 	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"> - Tên phương thức: addMessage - Input: String time, message msg - Output: không có - Mã giả: Tạo mới DocumentBuilderFactory, DocumentBuilder, Document (prase.(filePath)) + Tạo Element message (Element message, Element sender, Element content, Element recipient, Element time) + Nối Element message vào Node data + Tạo mới TransformerFactory, Transformer, DOMSource, StreamResult save file *.xml (file lưu lịch sử chat). 	Thêm một Message vào file History.xml để lưu trữ tin nhắn	SocketClient.java (dòng 54, 192)
3	<ul style="list-style-type: none"> - Tên phương thức: FillTable - Input: HistoryFrame - Output: không có - Mã giả: Tạo một DefaultTableModel lấy từ bảng 	Điền dữ liệu vào bảng hiển thị trên giao diện HistoryFrame	HistoryFrame.java (dòng 20)

	<p>hiển thị trên giao diện xem lịch sử.</p> <p>+ Tạo mới DocumentBuilderFactory, DocumentBuilder, Document (prase.(fXmlFile))</p> <p>+ Tạo một NodeList lấy Element có tên message từ Document</p> <p>+ Dừng vòng lặp đọc các Node trong NodeList và thêm thông tin các tin nhắn đó vào DefaultTableModel</p>		
4	<p>- Tên phương thức: getTagValue</p> <p>- Input: String sTag, eElement</p> <p>- Output: String</p> <p>- Mã giả: Tạo NodeList lấy tất cả các Element con theo tên Tag</p> <p>+ Tạo Node lấy giá trị đầu tiên của NodeList</p> <p>+ Trả về giá trị của Node</p>	Lấy giá trị của Tag theo tên Tag	History.java (dòng 75, 76)

Bảng 2.13 Bảng mô tả các phương thức trong lớp SocketClient

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

1	<ul style="list-style-type: none"> - Tên phương thức: SocketClient (hàm khởi tạo) - Input: ChatFrame - Output: không có - Mã giả: input (ui) this.ui = ui; + Lấy địa chỉ máy chủ; port từ ui; + Create socket từ port; + Input/Output Stream từ Socket; + Lấy history từ ui 	<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>	<p>ChatFrame.java (dòng 302)</p>
2	<ul style="list-style-type: none"> - Tên phương thức: run (kế thừa từ interface Runnable) - Input: không có - Output: không có - Mã giả: + if (type = message) print (sender > Me sender > 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. 	<p>Bao gồm các chức năng: Nhận tin nhắn, đăng nhập, kết nối tới server, đăng ký, đăng xuất, upload và download file.</p>	<p>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.</p>

	<ul style="list-style-type: none"> + if (type = signup) { if (content = TRUE) Enable button, print “Signup success” else print “Signup Failed” } + if (type = signout) print “sender > 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 + Xóa tất cả nội dung trên bảng hiển thị tin nhắn + Dừng Thread mà người dùng đang chạy + In ra thông báo: “Exception SocketClient run ()” 		
3	<ul style="list-style-type: none"> - Tên phương thức: send - Input: message 	<p>Chức năng gửi tin nhắn tới một user khác hoặc chat tới</p>	<p>ChatFrame.java (dòng 42, 305,</p>

	<ul style="list-style-type: none"> - Output: không có - Mã giả: In ra nội dung tin nhắn trong màn hình console + 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. 	tất cả mọi người online.	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 phần mềm 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.