

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**  
**TRƯỜNG ĐẠI HỌC ĐẠI NAM**



**BÀI TẬP LỚN**

**TÊN HỌC PHẦN: LẬP TRÌNH MẠNG**

**ĐỀ TÀI: PHÁT TRIỂN ỨNG DỤNG CHAT**

**Giáo viên hướng dẫn: GV.Nguyễn Quang Hưng**

**Sinh viên thực hiện:**

Stt	Mã sv	Họ và tên	Lớp
1	1571020193	Trần Quý Nhất	CNTT-1502

**Hà Nội, năm 2024**

## LỜI NÓI ĐẦU

Lập trình mạng không chỉ là một công cụ mạnh mẽ để kết nối các thiết bị với nhau mà còn mở ra một thế giới của các ứng dụng và dịch vụ truyền thông trực tuyến. Trong thời đại mà việc giao tiếp và trao đổi thông tin trở nên ngày càng phổ biến, việc phát triển các ứng dụng chat đã trở thành một trong những ưu tiên hàng đầu của nhiều nhà phát triển và doanh nghiệp. Ứng dụng chat không chỉ đơn giản là một công cụ để trò chuyện mà còn là cầu nối giữa con người, giúp họ kết nối và giao tiếp một cách dễ dàng và nhanh chóng. Từ các ứng dụng chat cá nhân đến các nền tảng trò chuyện trong các doanh nghiệp và cả trong lĩnh vực y tế và giáo dục, việc áp dụng lập trình mạng để phát triển các ứng dụng chat mang lại nhiều lợi ích và tiện ích không ngờ. Trong báo cáo này, chúng ta sẽ tìm hiểu về quy trình phát triển ứng dụng chat từ các khái niệm cơ bản như client-server, giao thức truyền thông, đến việc sử dụng các công nghệ và ngôn ngữ lập trình phổ biến như Socket.IO, Firebase, hay Java để xây dựng các tính năng và chức năng cần thiết cho một ứng dụng chat hoàn chỉnh. Báo cáo này không chỉ tập trung vào việc mô tả cách thức hoạt động của các ứng dụng chat mà còn sẽ đi sâu vào các vấn đề thực tiễn như bảo mật, quản lý người dùng, và hiệu suất hệ thống. Chúng ta cũng sẽ thảo luận về những thách thức và cơ hội khi phát triển các ứng dụng chat trong môi trường kinh doanh và xã hội ngày nay. Mục tiêu của báo cáo này là cung cấp một cái nhìn tổng quan và chi tiết về quá trình phát triển ứng dụng chat, giúp bạn hiểu rõ hơn về lập trình mạng và áp dụng kiến thức này vào việc xây dựng các ứng dụng truyền thông trực tuyến hiệu quả và linh hoạt.

## **LỜI CẢM ƠN**

Em xin gửi lời cảm ơn chân thành và sâu sắc nhất đến giảng viên hướng dẫn của chúng em trong môn lập trình mạng. Những kiến thức và sự hướng dẫn mà giảng viên đã chia sẻ không chỉ là nguồn cảm hứng mà còn là động lực lớn giúp chúng em tiến xa hơn trong hành trình học tập và nghiên cứu của mình. Qua từng buổi học, giảng viên đã truyền đạt những khái niệm cơ bản và phức tạp trong lĩnh vực lập trình mạng một cách rõ ràng và dễ hiểu. Sự kiên nhẫn và tận tâm của giảng viên đã giúp chúng em vượt qua những thách thức, hiểu biết sâu hơn về cách hoạt động của mạng và áp dụng kiến thức vào thực tiễn. Bằng sự hiểu biết sâu sắc và kinh nghiệm phong phú trong ngành, giảng viên không chỉ là người hướng dẫn mà còn là người cố vấn đáng tin cậy, luôn sẵn lòng chia sẻ và giúp đỡ khi chúng tôi gặp khó khăn. Em biết ơn vô cùng vì sự tận tâm và sự dày công của giảng viên, vì những giờ giảng dành cho chúng em, cũng như vì sự hỗ trợ và động viên không ngừng. Sự ảnh hưởng của giảng viên không chỉ dừng lại trong lớp học mà còn là nguồn động viên lớn lao để chúng em không ngừng phấn đấu và hoàn thiện bản thân. Xin chân thành cảm ơn giảng viên đã là người đồng hành đáng kính trọng trong hành trình học tập và phát triển của chúng em. Sự đóng góp và sự cống hiến của người đã là điểm sáng, là nguồn động viên không giới hạn không chỉ cho chúng em mà còn cho cả cộng đồng học thuật và ngành công nghiệp

## NỘI DUNG

<b>CHƯƠNG 1. MÔ TẢ VỀ JAVA SWING</b> .....	5
1. Java Swing là gì? .....	5
2. Đặc điểm của Java Swing .....	5
3. Ứng dụng của Java Swing .....	6
<b>CHƯƠNG 2: TÌM HIỂU VỀ SOCKET</b> .....	7
1. Lập Trình Socket Cơ Bản Với TCP/IP Trong Java .....	7
2. Tổng quan về Socket .....	7
3. Lập trình TCP Socket với Java .....	9
<b>CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG CHAT</b> .....	11
1. Đăng nhập .....	11
2. Đăng xuất .....	13
3. Đổi mật khẩu .....	14
4. Gửi tin nhắn .....	15
5. Gửi ảnh .....	16
6. Gửi file .....	18
7. Xóa tin nhắn .....	19
<b>KẾT LUẬN</b> .....	20
1. Ưu điểm .....	20
2. Nhược điểm .....	20
3. Hướng phát triển.....	20
<b>DANH MỤC TÀI LIỆU THAM KHẢO</b> .....	21

## CHƯƠNG 1. MÔ TẢ VỀ JAVA SWING

### 1. Java Swing là gì?

Java Swing là cách gọi rút gọn khi người ta nhắc đến Swing của Java Foundation (JFC). Nó là bộ công cụ GUI mà Sun Microsystems phát triển để xây dựng các ứng dụng tối ưu dùng cho window (bao gồm các thành phần như nút, thanh cuộn,...).

Swing được xây dựng trên AWT API và hoàn toàn được viết bằng Java. Tuy nhiên, nó lại khác với AWT ở chỗ bộ công cụ này thuộc loại nền tảng độc lập, bao gồm các thành phần nhẹ và phức tạp hơn AWT.

Các gói javax.swing bao gồm các lớp cho Java Swing API như JMenu, JButton, JTextField, JRadioButton, JColorChooser,...

Việc xây dựng ứng dụng sẽ trở nên dễ dàng hơn với Java Swing vì chúng ta có các bộ công cụ GUI giúp đỡ công việc.

Swing được chính thức phát hành vào tháng 3/1998. Nó đi kèm với thư viện Swing 1.0 với hơn 250 lớp, 80 giao tiếp.

Hiện nay con số này đã được tăng lên, ở phiên bản Swing 1.4 có 451 lớp và 85 giao tiếp.

Các bản phát hành Java 2 (SDK 1.2 và các phiên bản mới hơn) đều bao gồm Swing trong runtime environment.

### 2. Đặc điểm của Java Swing

Các đặc điểm chính của Java Swing:

**Độc lập nền tảng (Platform Independent):** Swing được xây dựng trên nền tảng Java, do đó, các ứng dụng Swing có thể chạy trên bất kỳ hệ điều hành nào có cài đặt Java Virtual Machine (JVM).

**Lightweight Components:** Các thành phần của Swing không phụ thuộc trực tiếp vào hệ điều hành để hiển thị, mà chúng được vẽ hoàn toàn bằng Java. Điều này cho phép Swing có khả năng tùy biến giao diện cao và tránh được sự khác biệt giữa các hệ điều hành.

MVC Architecture: Swing sử dụng kiến trúc Model-View-Controller (MVC), tách biệt phần logic dữ liệu (Model), phần hiển thị (View), và phần điều khiển (Controller). Điều này giúp cho việc bảo trì và mở rộng ứng dụng dễ dàng hơn.

Pluggable Look and Feel: Swing cho phép thay đổi giao diện (look) và cảm nhận (feel) của các ứng dụng mà không cần thay đổi mã nguồn. Người dùng có thể lựa chọn từ các giao diện mặc định của Swing hoặc tự tạo các giao diện tùy biến.

Các thành phần chính của Swing

- JFrame: Đây là cửa sổ chính của ứng dụng Swing, tương đương với cửa sổ ứng dụng trong các hệ điều hành.
- JPanel: Là một vùng chứa các thành phần khác, có thể được lồng nhau để tạo ra các bố cục phức tạp.
- JButton: Nút bấm cho phép người dùng thực hiện một hành động khi bấm vào.
- JLabel: Nhãn để hiển thị văn bản hoặc hình ảnh.
- JTextField: Trường nhập liệu văn bản một dòng.
- JTextArea: Trường nhập liệu văn bản nhiều dòng.
- JList: Danh sách cho phép người dùng chọn một hoặc nhiều mục.
- JTable: Bảng hiển thị dữ liệu dưới dạng hàng và cột.

### 3. Ứng dụng của Java Swing

Java Swing được dùng để hỗ trợ tạo giao diện đồ họa người dùng (với Java).

Bộ công cụ này cung cấp các bộ điều khiển nâng cao như thanh trượt, colorpicker, Tree, TabbedPane và bảng điều khiển,..

Swing có những đặc điểm:

- Độc lập với thiết bị
- Có thể tùy chỉnh, mở rộng
- Khá nhẹ, có thể cấu hình

Ngoài ra bạn cũng có thể tùy chỉnh các điều khiển xoay một cách dễ dàng mà không ảnh hưởng đến các thành phần khác.

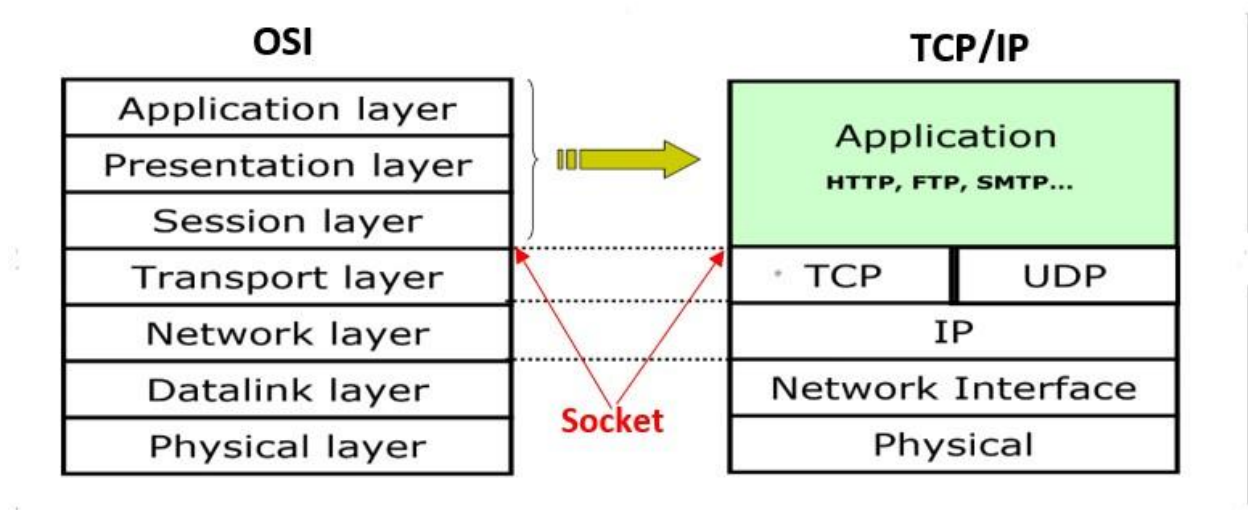
## CHƯƠNG 2: TÌM HIỂU VỀ SOCKET

### 1. Lập Trình Socket Cơ Bản Với TCP/IP Trong Java

Socket được biết đến là cánh cửa giao tiếp giữa hai tiến trình ứng dụng. Socket hỗ trợ nhiều ngôn ngữ (C, java, perl, python,...) cũng như nhiều platform (\*nix, Windows,...). Socket được viết bằng các ngôn ngữ khác nhau và chạy trên bất kỳ platform nào đều có thể giao tiếp với nhau. Trong bài viết này mình gửi đến các bạn khái niệm cơ bản về Socket cũng như cách lập trình Socket với các loại giao thức.

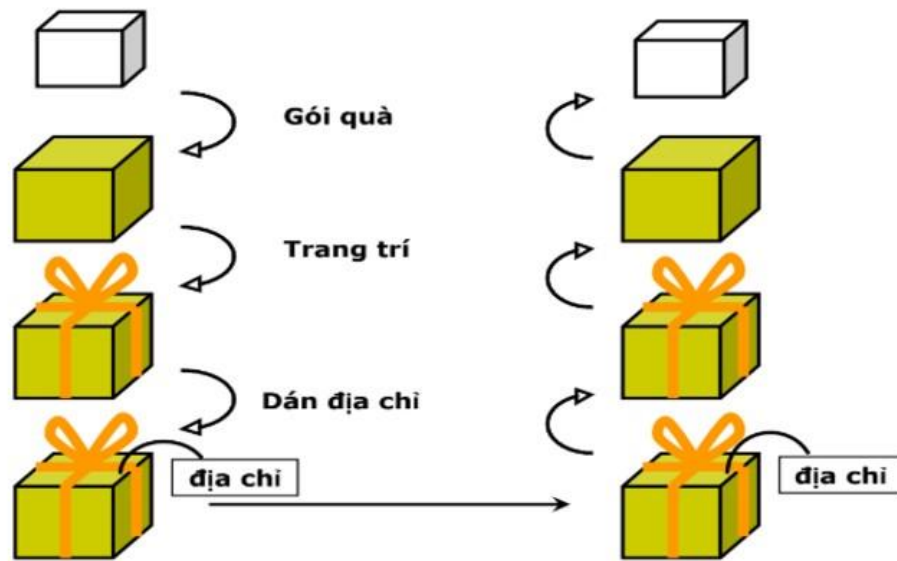
### 2. Tổng quan về Socket

Trong hệ thống mạng máy tính tồn tại những mô hình tham chiếu có kiến trúc phân tầng (OSI, TCP/IP...) nhằm hỗ trợ chức năng trao đổi thông tin giữa các ứng dụng ở nhiều máy tính khác nhau.



Ảnh 2.1: Tổng quan về Socket

Dữ liệu bên gửi sẽ được đóng gói (Encapsulation) từ tầng trên đến tầng cuối là tầng vật lý (Physical Layer), sau đó nhờ tầng vật lý này chuyển dữ liệu đến tầng vật lý máy bên nhận, bên nhận tiến hành giải mã (decapsulation) gói dữ kiện từ tầng dưới lên tầng trên cùng, là tầng ứng dụng (application layer).



**Ảnh 2.2: Tổng quan về Socket**

Ở đây, Socket chính là cửa giao tiếp giữa tầng ứng dụng và tầng giao vận (Transport layer). Nói cách khác, Socket là giao diện do ứng dụng tạo ra trên máy trạm, quản lý bởi hệ điều hành qua đó các ứng dụng có thể gửi/nhận thông điệp đến/từ các ứng dụng khác. Ở đó, Socket sẽ được ràng buộc với một mã số cổng (Port Number) để giúp tầng giao vận định danh được ứng dụng nhận/gửi thông điệp.

Các bạn có thể thấy ở hình ảnh trên, tầng giao vận có 2 phương thức là TCP (Transmission Control Protocol) và UDP (User Datagram Protocol), như vậy socket cơ bản là có 2 loại: Stream Socket sử dụng TCP truyền dòng bytes và Datagram Socket sử dụng UDP truyền gói tin. Với ngôn ngữ lập trình Java, chúng ta được cung cấp 3 loại khác nhau của sockets:

**Stream Socket (TCP) :** Tạo luồng dữ liệu hai chiều, đáng tin cậy, có trình tự và không trùng lặp, dữ liệu chỉ được gửi/nhận khi có đã có liên kết. Dùng với Socket Class của java.

**Datagram Socket (UDP):** Có thể nhận dữ liệu không theo trình tự, trùng lặp. Dùng với DatagramSocket Class.

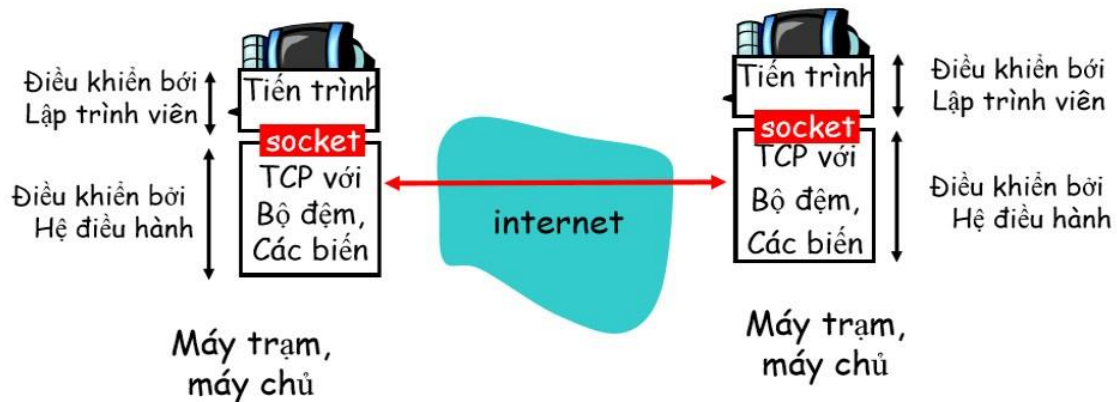
**Multicast Socket :** cho phép dữ liệu được gửi đến nhiều bên nhận một lúc. Dùng với DatagramSocket Class.



Socket được hỗ trợ trên nhiều ngôn ngữ như C, Java, Pearl, Python,... Sau đây là một ví dụ lập trình socket với Java.

### 3. Lập trình TCP Socket với Java

Trong bài viết này mình sẽ nói về lập trình Socket sử dụng TCP.



**Ảnh 2.3: Tổng quan về Socket**

Đúng như tính chất của TCP chúng ta cần có liên kết 2 chiều trước khi server và client có thể trao đổi thông điệp với nhau.

Ban đầu, phía server tạo Socket được ràng buộc với một cổng (port number) để chờ nhận yêu cầu từ phía client.

Tiếp đến phía client yêu cầu server bằng cách tạo một Socket TCP trên máy kèm với địa chỉ IP và port number của tiến trình tương ứng trên máy server. Khi client tạo Socket, client TCP tạo liên kết với server TCP và chờ chấp nhận kết nối từ server.

TCP cung cấp dịch vụ truyền dòng tin cậy và có thứ tự giữa client và server, giữa máy chủ và máy nhận chỉ có 1 địa chỉ IP duy nhất. Thêm vào đó, mỗi thông điệp truyền đi đều có xác nhận trả về.

Sau đây là một ví dụ ứng dụng đơn giản về lập trình TCP Socket với Java.

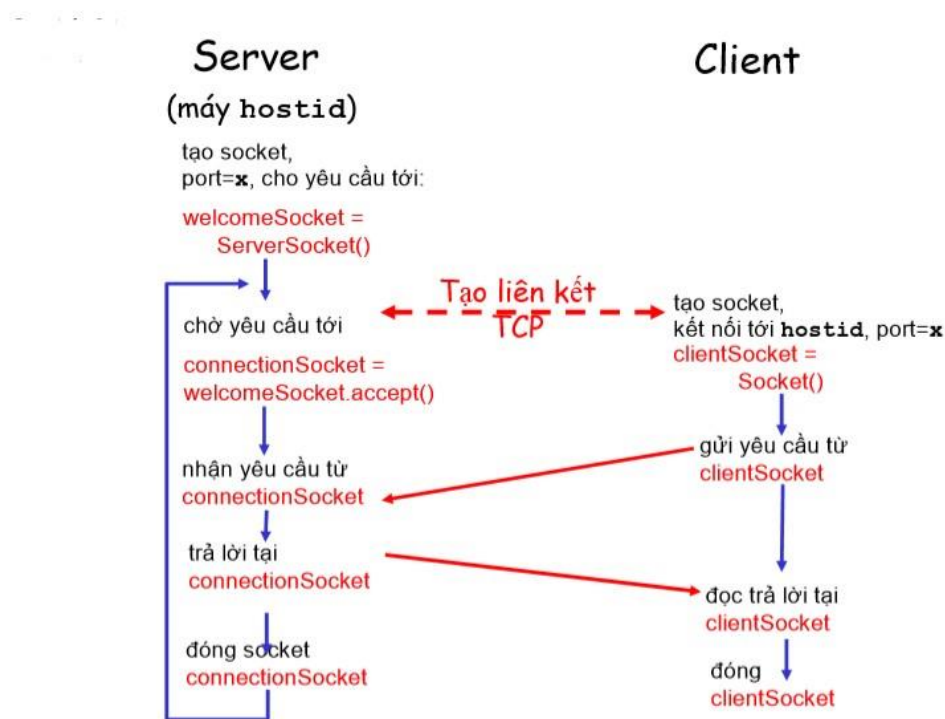
Miêu tả ứng dụng:

Client đọc dòng văn bản nhập từ bàn phím người dùng , gửi tới server qua Socket

Server đọc các dòng văn bản gửi từ Socket

Server sẽ chuyển lại dòng văn bản kèm theo “Server accepted” tới phía client qua Socket

Client đọc dòng văn bản từ socket và in ra dòng văn bản nhận được từ server



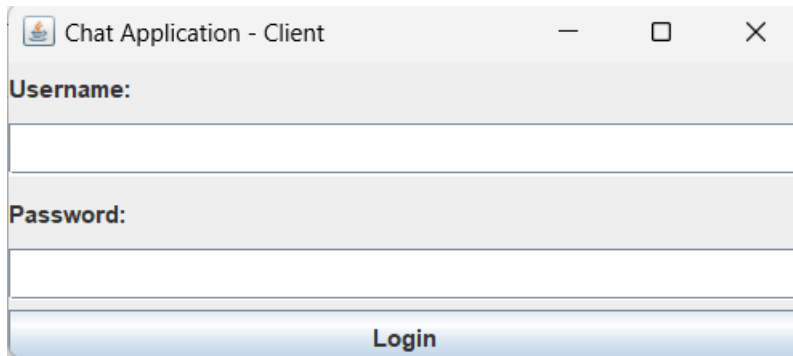
**Ảnh 2.4: Tổng quan về Socket**

Chúng ta có thể thấy rằng mỗi phía server và client đều có 2 luồng dữ liệu, một luồng ra Socket để gửi thông điệp và một luồng vào từ Socket để nhận thông điệp, như vậy với mỗi bên mình có hai biến input và output (inFromServer, outToServer và inFromClient, outToClient).

## CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG CHAT

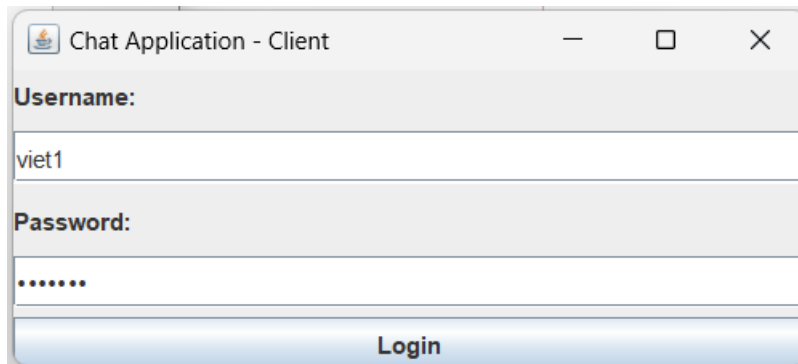
### 1. Đăng nhập

- Người dùng có thể đăng nhập vào ứng dụng bằng cách nhập tên người dùng và mật khẩu của họ



Ảnh 3.1: Giao diện đăng nhập của Client

- Hệ thống sẽ xác thực thông tin và cung cấp quyền truy cập vào ứng dụng nếu thông tin hợp lệ
  - *Hợp lệ*

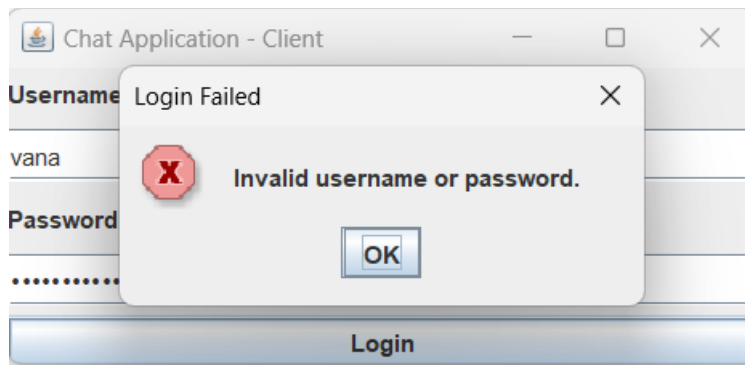


Ảnh 3.2: Đăng nhập thành công



Ảnh 3.3: Giao diện ứng dụng chat

- **Không hợp lệ**



Ảnh 3.4: Sai thông tin đăng nhập

• Code

```

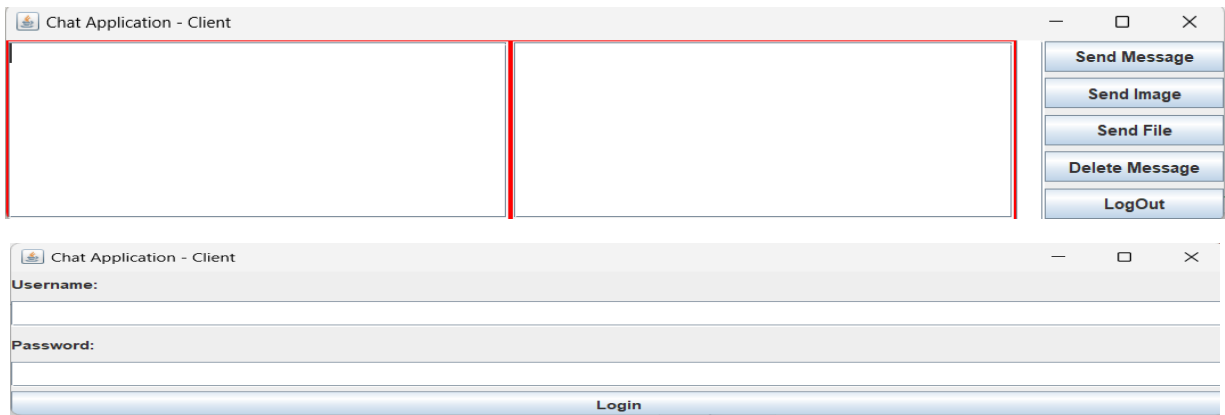
JPanel authPanel = new JPanel(new GridLayout(5, 2, 5, 5));
JLabel usernameLabel = new JLabel("Username:");
authPanel.add(usernameLabel);
JTextField usernameField = new JTextField();
authPanel.add(usernameField);
JLabel passwordLabel = new JLabel("Password:");
authPanel.add(passwordLabel);
JPasswordField passwordField = new JPasswordField();
authPanel.add(passwordField);
JButton loginButton = new JButton("Login");
loginButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String username = usernameField.getText();
        String password = new String(passwordField.getPassword());
        // Thực hiện xác thực thông tin đăng nhập ở đây
        if ((username.equals("viet2") && password.equals("2102032")) || (username.equals("viet1") && password.equals("2102031"))) {
            isLoggedIn = true;
            panel.remove(authPanel);
            panel.add(createMainPanel(), BorderLayout.CENTER);
            revalidate();
            repaint();
            connectToServer();
        } else {
            JOptionPane.showMessageDialog(Client1GUI.this, "Invalid username or password.", "Login Failed", JOptionPane.ERROR_MESSAGE);
        }
    }
});

```

Ảnh 3.5: Phần trình bày

## 2. Đăng xuất

- Người dùng có thể đăng xuất khỏi tài khoản của họ
- Sau khi đăng xuất, người dùng sẽ không thể truy cập vào các chức năng của ứng dụng cho đến khi được đăng nhập lại



Ảnh 3.6: Giao diện

- Code

```

]       JButton logOutButton = new JButton("LogOut");
]       logOutButton.addActionListener(new ActionListener() {
-           public void actionPerformed(ActionEvent e) {
-               logOut();
-           }
-       });
-       buttonPanel.add(logOutButton);
-       inputPanel.add(buttonPanel, BorderLayout.EAST);
-
-       mainPanel.add(inputPanel, BorderLayout.CENTER);
-
-       return mainPanel;
-   }

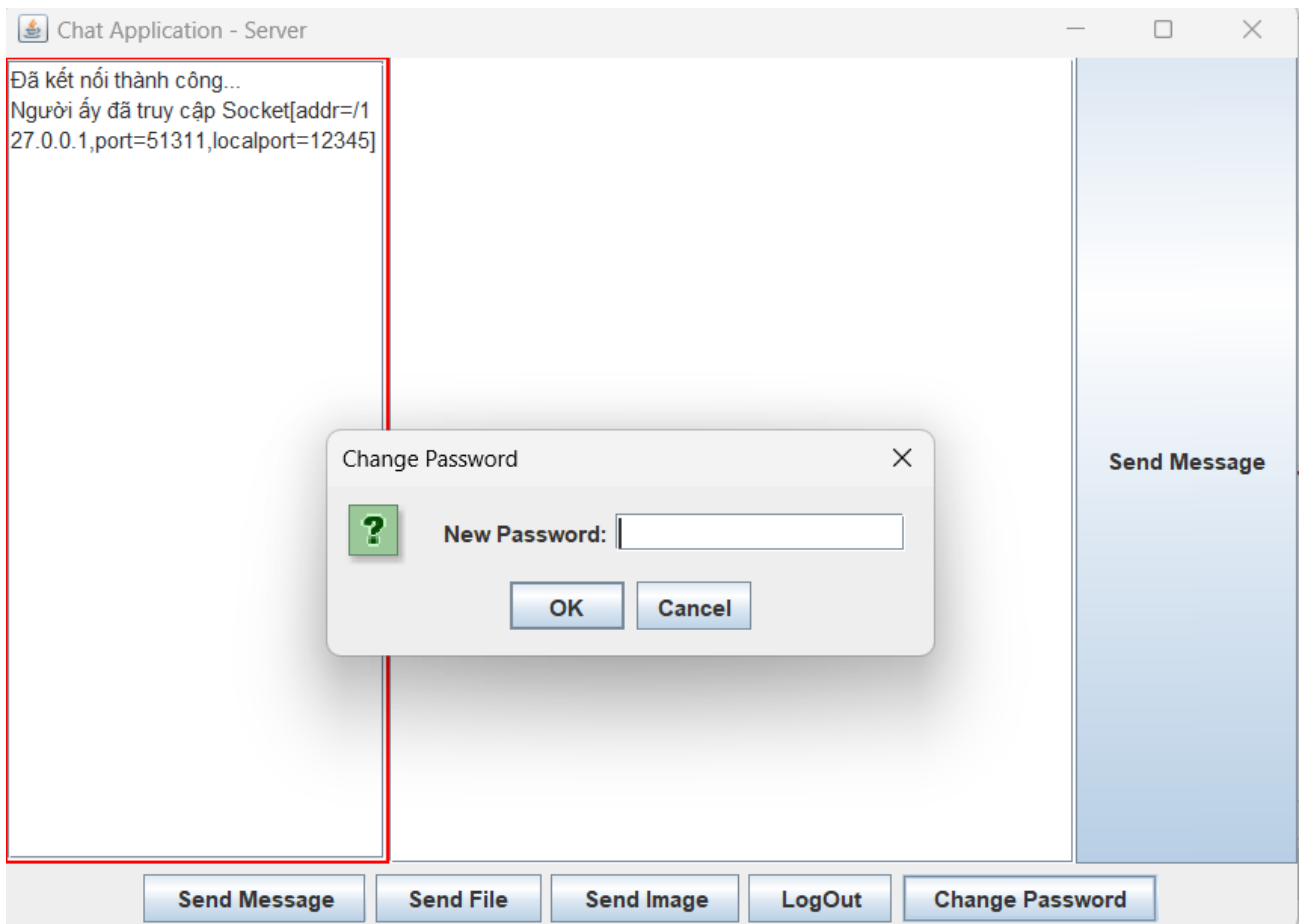
]   private void logOut() {
-       isLoggedIn = false;
-       getContentPane().removeAll();
-       getContentPane().add(createAuthPanel(), BorderLayout.NORTH);
-       revalidate();
-       repaint();
-   }

```

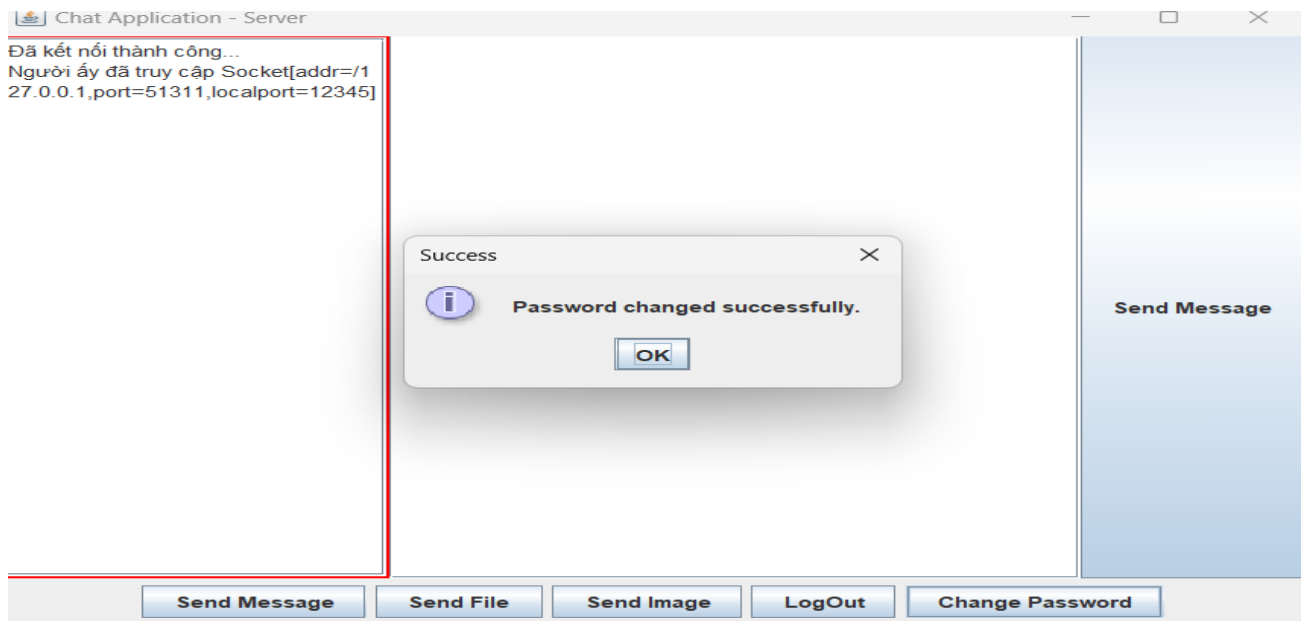
Ảnh 3.7: Phần trình bày

### 3. Đổi mật khẩu

- Người dùng có thể thay đổi mật khẩu của mình bằng cách cung cấp mật khẩu hiện tại và nhập mật khẩu mới
- Hệ thống sẽ xác thực mật khẩu hiện tại và cho phép người dùng cập nhật mật khẩu mới.



Ảnh 3.8: Giao diện đổi mật khẩu



**Ảnh 3.9: Thành công**

- Code

```

,

JPanel panel = new JPanel();
JTextField newPasswordField = new JTextField(15);
panel.add(new JLabel("New Password:"));
panel.add(newPasswordField);

int result = JOptionPane.showConfirmDialog(null, panel, "Change Password", JOptionPane.OK_CANCEL_OPTION);
if (result == JOptionPane.OK_OPTION) {
    String newPassword = newPasswordField.getText();
    // Thay đổi mật khẩu ở đây
    JOptionPane.showMessageDialog(this, "Password changed successfully.", "Success", JOptionPane.INFORMATION_MESSAGE);
}
}

```

**Ảnh 3.10: Phần trình bày**

#### 4. Gửi tin nhắn

- Người dùng có thể gửi tin nhắn văn bản cho các liên hệ trong danh sách bạn bè hoặc nhóm chat
- Tin nhắn được gửi sẽ hiển thị trong cuộc trò chuyện và được nhận ngay lập tức bởi người nhận



**Ảnh 3.11: Giao diện**

- Code

```

1  JButton sendMessage = new JButton("Send Message");
2
3  JButton sendButton = new JButton("Send Message");
4  sendButton.addActionListener(new ActionListener() {
5      public void actionPerformed(ActionEvent e) {
6          sendMessage();
7      }
8  });
9  buttonPanel.add(sendButton);

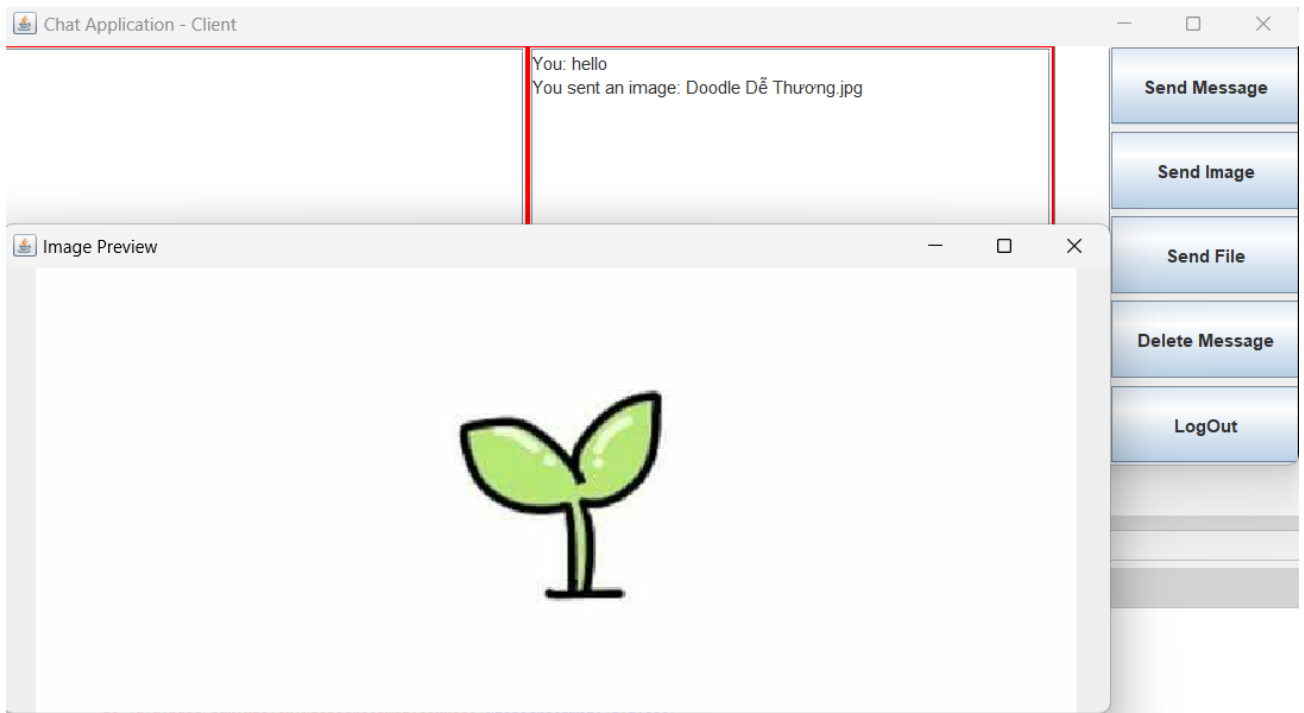
```

**Ảnh 3.12: Phần trình bày**

## 5. Gửi ảnh

- Người dùng có thể gửi hình ảnh từ thư viện của điện thoại chụp ảnh mới để gửi
- Hình ảnh được gửi sẽ được hiển thị trong cuộc trò chuyện và có thể được xem bởi tất cả các thành viên





**Ảnh 3.13: Giao diện**

- Code

```
private void sendImage() {
    if (!isLoggedIn) {
        JOptionPane.showMessageDialog(this, "Please login first.", "Not Logged In", JOptionPane.WARNING_MESSAGE);
        return;
    }
    JFileChooser fileChooser = new JFileChooser();
    int returnValue = fileChooser.showOpenDialog(this);
    if (returnValue == JFileChooser.APPROVE_OPTION) {
        File selectedFile = fileChooser.getSelectedFile();
        chatArea.append("You sent an image: " + selectedFile.getName() + "\n");

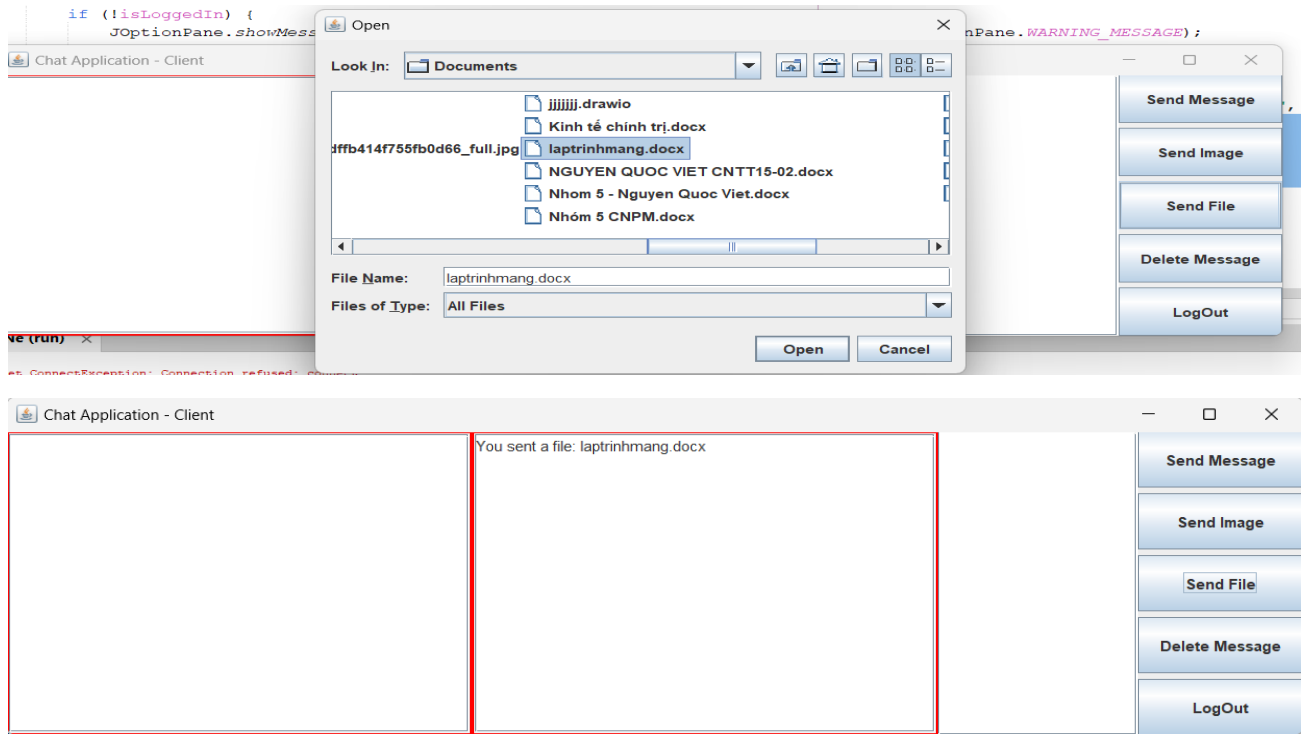
        try {
            // Hiển thị trước ảnh
            BufferedImage img = ImageIO.read(selectedFile);
            JFrame frame = new JFrame("Image Preview");
            JLabel label = new JLabel(new ImageIcon(img));
            frame.getContentPane().add(label, BorderLayout.CENTER);
            frame.pack();
            frame.setLocationRelativeTo(null);
            frame.setVisible(true);

            ObjectOutputStream objectOutputStream = new ObjectOutputStream(socket.getOutputStream());
            objectOutputStream.writeObject(selectedFile);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

**Ảnh 3.14: Phần trình bày**

## 6. Gửi file

- Người dùng có thể gửi tệp tin từ bộ nhớ của thiết bị
- Tệp tin được gửi sẽ có thể được tải xuống và xem bởi người khác



Ảnh 3.15: Giao diện

- Code

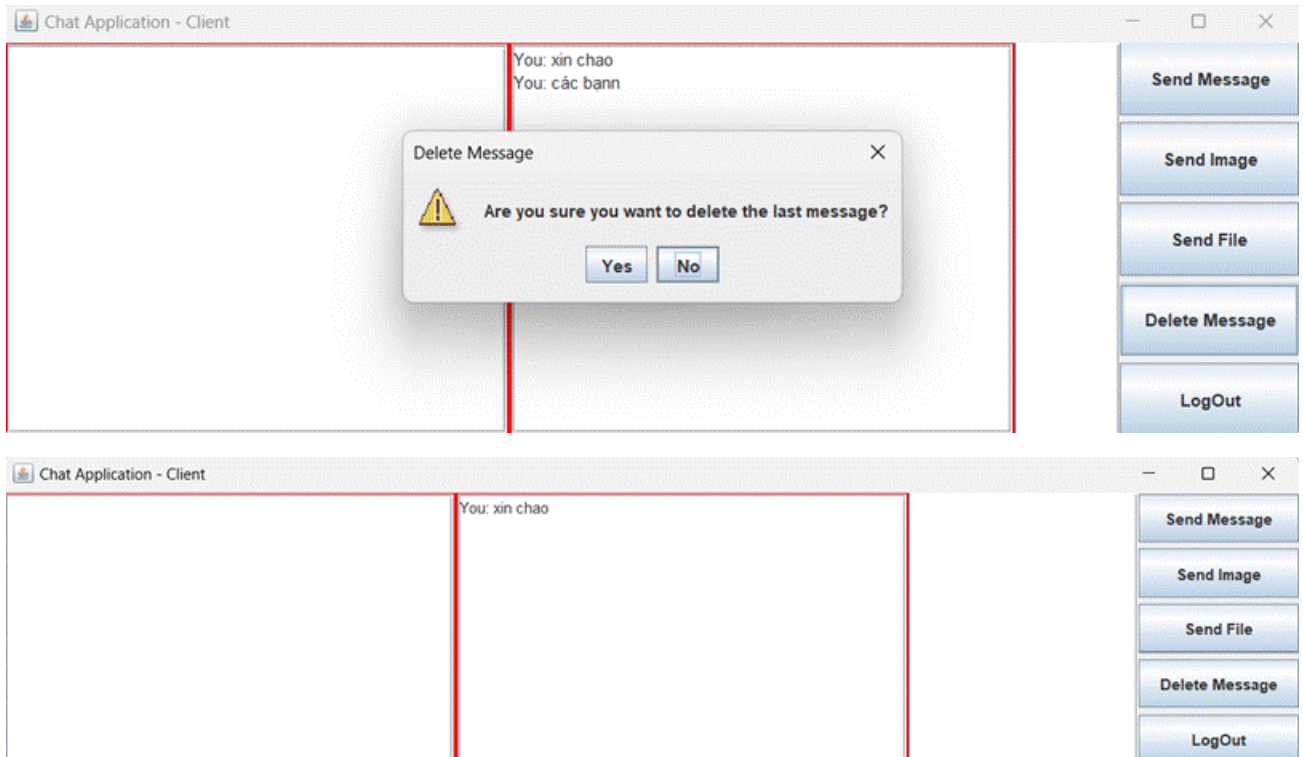
```
private void sendFile() {
    if (!isLoggedIn) {
        JOptionPane.showMessageDialog(this, "Please login first.", "Not Logged In", JOptionPane.WARNING_MESSAGE);
        return;
    }
    JFileChooser fileChooser = new JFileChooser();
    int returnValue = fileChooser.showOpenDialog(this);
    if (returnValue == JFileChooser.APPROVE_OPTION) {
        File selectedFile = fileChooser.getSelectedFile();
        chatArea.append("You sent a file: " + selectedFile.getName() + "\n");
        out.println("FILE:" + selectedFile.getName());

        try {
            FileInputStream fileInputStream = new FileInputStream(selectedFile);
            byte[] buffer = new byte[1024];
            int bytesRead;
            while ((bytesRead = fileInputStream.read(buffer)) != -1) {
                socket.getOutputStream().write(buffer, 0, bytesRead);
            }
            fileInputStream.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Ảnh 3.16: Phần trình bày

## 7. Xoá tin nhắn

- Người dùng có thể xoá tin nhắn đã gửi
- Sau khi xoá, tin nhắn không còn xuất hiện trong cuộc trò chuyện của cả người gửi lẫn người nhận



Ảnh 3.17: Giao diện

- Code

```
private void deleteMessage() {
    if (!isLoggedIn) {
        JOptionPane.showMessageDialog(this, "Please login first.", "Not Logged In", JOptionPane.WARNING_MESSAGE);
        return;
    }
    String[] options = { "Yes", "No" };
    int choice = JOptionPane.showOptionDialog(this, "Are you sure you want to delete the last message?", "Delete Message",
        JOptionPane.YES_NO_OPTION, JOptionPane.WARNING_MESSAGE, null, options, options[1]);
    if (choice == JOptionPane.YES_OPTION) {
        String text = chatArea.getText();
        int lastIndex = text.lastIndexOf("\nYou:");
        if (lastIndex >= 0) {
            int startIndex = text.substring(0, lastIndex).lastIndexOf("\nYou:");
            chatArea.setText(text.substring(0, startIndex));
        }
    }
}
```

Ảnh 3.18: Phần trình bày

## KẾT LUẬN

### 1. Ưu điểm

- Có thể kết nối mọi người để trò chuyện
- Gửi tin nhắn đa phương tiện

### 2. Nhược điểm

- Tính bảo mật chưa cao
- Giao diện người dùng chưa thân thiện
- Vẫn còn sơ sài và chưa hoàn thiện ứng dụng

### 3. Hướng phát triển

- Phát triển giao diện thân thiện với người dùng hơn
- Tăng cường tính bảo mật thông tin cho người dùng
- Có thể sử dụng được đa nền tảng

**DANH MỤC TÀI LIỆU THAM KHẢO**

- [1]. Nguyễn Hồng Sơn (2007), *Giáo trình hệ thống Mạng máy tính CCNA* (Semester 1), NXB Lao động xã hội.
- [2]. Phạm Quốc Hùng (2017), *Đề cương bài giảng Mạng máy tính*, Đại học SPKT Hưng Yên.
- [3]. James F. Kurose and Keith W. Ross (2013), *Computer Networking: A top-down approach sixth Edition*, Pearson Education.