**ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH**

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA KHOA HỌC – KỸ THUẬT MÁY TÍNH

---------------o0o---------------

****

L**UẬN VĂN TỐT NGHIỆP**

**THIẾT KẾ TRANG WEB THI TRẮC NGHIỆM ONLINE**

**GVHD: ThS.Nguyễn Hữu Hiếu**

**SVTH: Nguyễn Tuấn Anh**

**MSSV: 17333403**

**TP. HỒ CHÍ MINH, NGÀY 04 THÁNG 06 NĂM 2021**

# Chức năng website

## Đăng ký

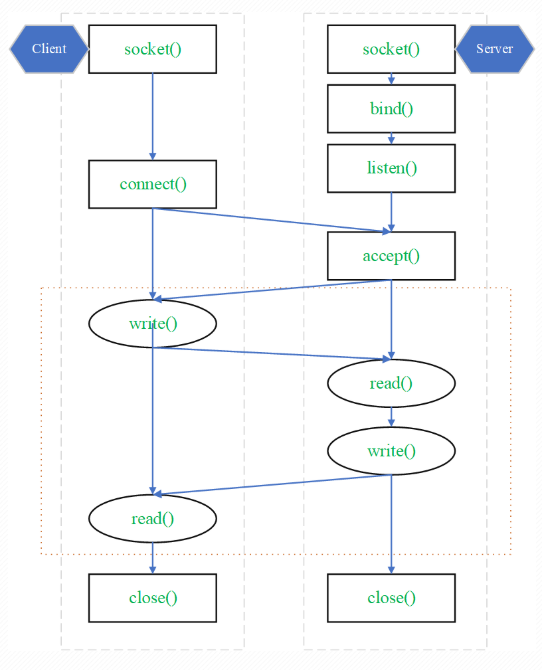
* Thêm giao diện – Hoàn thành
* Thêm thành viên mới – Hoàn thành
* Bắt lỗi và hiển thị lỗi (chưa hoàn thành)
* Tổng kết: Hoàn thành 95%

# LÝ THUYẾT

## Tổng quan

Socket là phương tiện hiệu quả để xây dựng các ứng dụng theo kiến trúc Client-Server. Socket có hai chế độ: có nối kết (**TCP – Transmission Control Protocol**) và không nối kết ( **UDP – User Datagram Protocol**).

### 2.1.1 Mô hình Client-Server sử dụng Socket ở chế độ có nối kết (TCP)

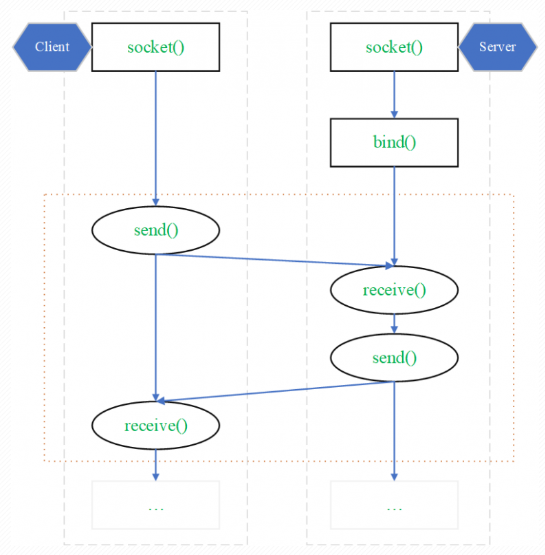


**Figure 1: Mô hình TCP**

Chế độ TCP có thể phân thành 4 giai đoạn như sau:

* **Giai đoạn 1**: Server tạo Socket, gán số hiệu cổng và lắng nghe yêu cầu nối kết. Server sẵn sàng phục vụ Client.socket(): Server yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển.
  + bind(): Server yêu cầu gán số hiệu cổng (port) cho socket.
  + listen(): Server lắng nghe các yêu cầu nối kết từ các client trên cổng đã được gán.
* **Giai đoạn 2**: Client tạo Socket, yêu cầu thiết lập một nối kết với Server.
  + socket(): Client yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển, thông thường hệ thống tự động gán một số hiệu cổng còn rảnh cho socket của Client.
  + connect(): Client gởi yêu cầu nối kết đến server có địa chỉ IP và Port xác định.
  + accept(): Server chấp nhận nối kết của client, khi đó một kênh giao tiếp ảo được hình thành, Client và server có thể trao đổi thông tin với nhau thông qua kênh ảo này.
* **Giai đoạn 3**: Trao đổi thông tin giữa Client và Server.
  + Sau khi chấp nhận yêu cầu nối kết, thông thường server thực hiện lệnh read() và nghẽn cho đến khi có thông điệp yêu cầu (Request Message) từ client gởi đến.
  + Server phân tích và thực thi yêu cầu. Kết quả sẽ được gởi về client bằng lệnh write().
  + Sau khi gởi yêu cầu bằng lệnh write(), client chờ nhận thông điệp kết quả (ReplyMessage) từ server bằng lệnh read().
* **Giai đoạn 4**: Kết thúc phiên làm việc.
  + Các câu lệnh read(), write() có thể được thưc hiện nhiều lần (ký hiệu bằng hình ellipse).
  + Kênh ảo sẽ bị xóa khi Server hoặc Client đóng socket bằng lệnh close().

### 2.1.2 Mô hình Client-Server sử dụng Socket ở chế độ không nối kết (UDP)



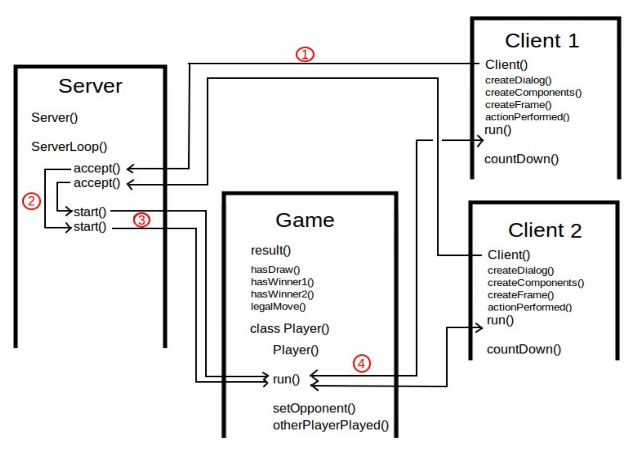
**Figure 2: Mô hình UDP**

Chế độ UDP có thể phân thành 3 giai đoạn như sau:

* Giai đoạn 1: Server tạo Socket – gán số hiệu cổng.
  + socket(): Server yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển.
  + · bind(): Server yêu cầu gán số hiệu cổng cho socket.
* Giai đoạn 2: Client tạo Socket.
  + socket(): Client yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển, thông thường hệ thống tự động gán một số hiệu cổng còn rảnh cho socket của Client.
* Giai đoạn 3: Trao đổi thông tin giữa Client và Server.
  + Sau khi tạo Socket xong, Client và Server có thể trao đổi thông tin qua lại với nhau thông qua hai hàm send() và receive().
  + Đơn vị dữ liệu trao đổi giữa Client và Server là các Datagram Package (Gói tin thư tín).
  + Protocol của ứng dụng phải định nghĩa khuôn dạng và ý nghĩa của các Datagram Package. Mỗi Datagram Package có chứa thông tin về địa chỉ người gởi và người nhận (IP, Port).

# SƠ ĐỒ DIAGRAM

## 3.1 Sơ đồ Class diagram

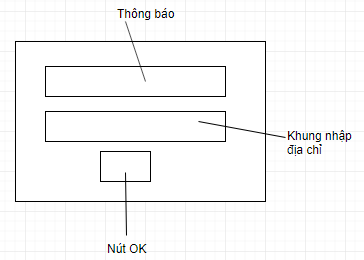
****

**Figure 3: Sơ đồ Class diagram**

* + Liệt kê các yêu cầu đặt ra
  + Ghi cụ thể (có tính định lượng) các **yêu cầu, chi tiết kỹ thuật rõ ràng.**

## 3.2 Sơ đồ giao diện

* Khung đăng nhập

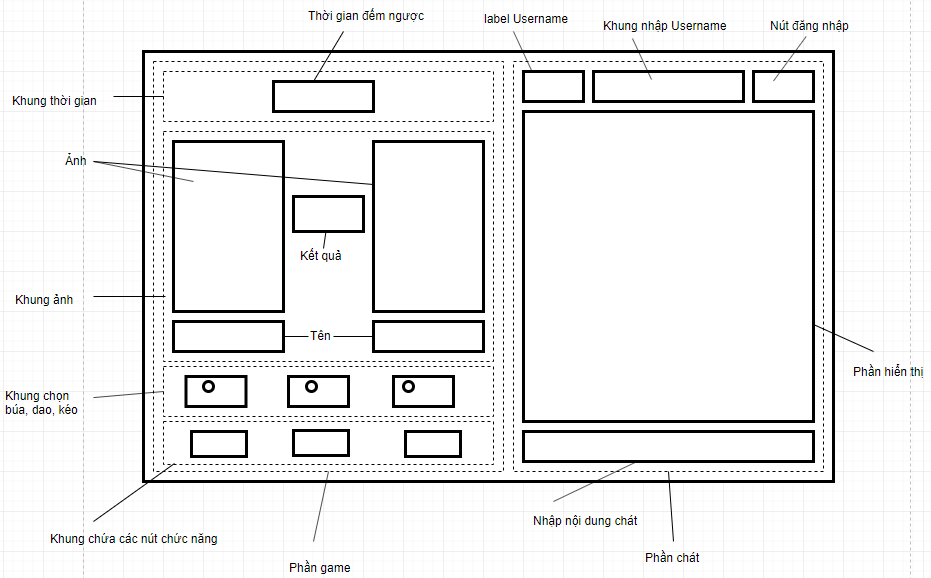


**Figure 4: Sơ đồ đăng nhập**

Ứng dụng phía Server đã được chạy trước đó.

Người chơi khi chạy ứng dụng phía client phải nhập địa chỉ server hợp lệ để kết nối với server.

* Khung màn hình chính



**Figure 5: Sơ đồ màn hình chính**

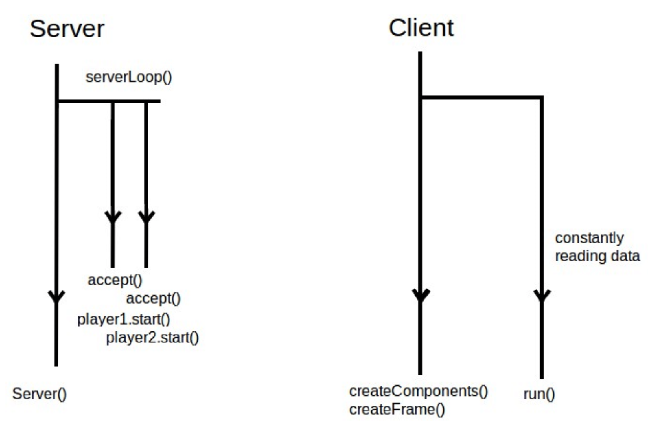
Khung màn hình chính gồm hai phần. Phía bên trái giành cho chức năng của trò chơi. Phía bên phải để thực hiện chức năng đăng nhập, khung chát, hiển thị kết quả.

Ban đầu, người chơi phải đăng ký username hợp lệ (không để trống), phía server sẽ hiển thị thông báo hướng dẫn, phần nhập nội dung chát được mở. Khi người chơi thứ hai hoàn thành thủ tục như trên thì hai người có thể trò chuyện và chơi game.

Để chơi game, người chơi thứ nhất (người đăng nhập trước) sẽ nhấn nút chọn để chọn phần lựa chọn “Búa” – “Dao” –“Kéo”. Phần thời gian đếm ngược sẽ tự động đếm trong vòng 5 giây cho đến khi về 0. Khi thời gian trở về 0, người chơi thứ hai mới được thực hiện lượt chơi của mình. Khi người chơi chọn thì sẽ có ảnh tương ứng hiển thị trên UI. Kết quả được hiển thị tại cả khung chát và phía khung giữa hai ảnh thông qua một label kết quả.

Khi một trong hai người chơi rời khỏi game, server sẽ thông báo cho người còn lại biết để thoát game.

## 3.3 Sơ đồ Thread



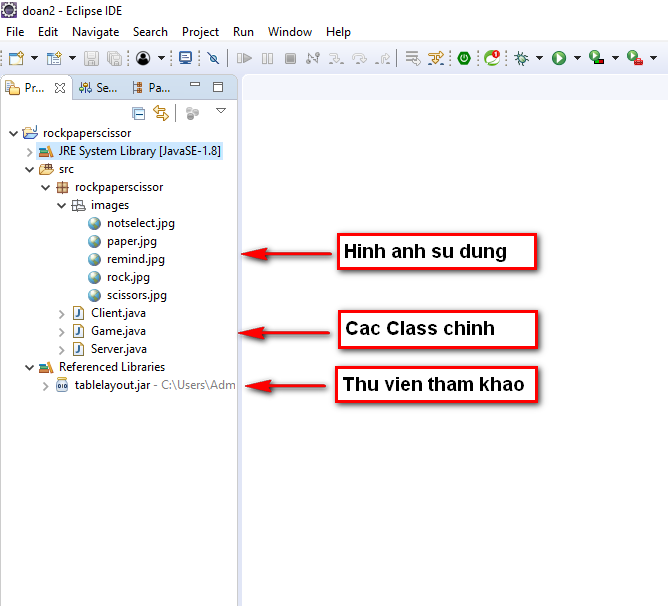
**Figure 6: Sơ đồ thread**

Khi server đã tạo xong socket với cổng kết nối xác định, server sẽ liên tục lắng nghe yêu cầu từ client, ghép cặp hai người chơi với nhau.

Phía client, sau khi được chấp nhận kết nối sẽ khởi tạo một giao diện chính , liên tục trao đổi thông tin với server. Server sẽ dựa vào từ khóa ngay phía trước tin nhắn của phía Client để xác định loại yêu cầu và xử lý cho hợp lệ.

# CẤU TRÚC, THÀNH PHẦN PHẦN MỀM THỰC TẾ

4.1 Cấu trúc phần mềm



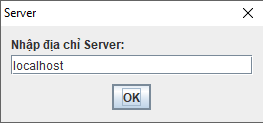
**Figure 7: Sơ đồ project**

Project có cấu trúc khá đơn giản, gồm ba class chính:

* **Server.java**: Khởi tạo socket, tạo kết nối cho từng cặp client
* **Client.java**: Class chính, gửi yêu cầu tới server, khởi tạo game cho hai người chơi, đọc ảnh từ thư mục “images” để hiển thị lên UI, cũng như thực hiện nhiều chức năng khác.
* **Game.java**: Khởi tạo game cho hai người chơi.
* **Thư mục images**: Lưu giữ ảnh sử dụng cho project
* **Thư viện tablelayout**.jar: Thiết kế giao diện theo dạng cấu trúc bảng. Các thành phần của giao diện được xét vị trí theo chỉ số. Thư viện có ưu điểm giúp phần mềm chạy được đa hệ điều hành (các máy Mac không có window form).

# KẾT QUẢ THỰC HIỆN

## 5.1 Client kết nối tới server



**Figure 8: Nhập địa chỉ Server**

## 5.2 Giao diện chính



**Figure 9: Màn hình chính**

## 5.3 Người chơi đăng ký username

* Người chơi thứ nhất



**Figure 10: Chào mừng người chơi 1**

* Người chơi thứ hai

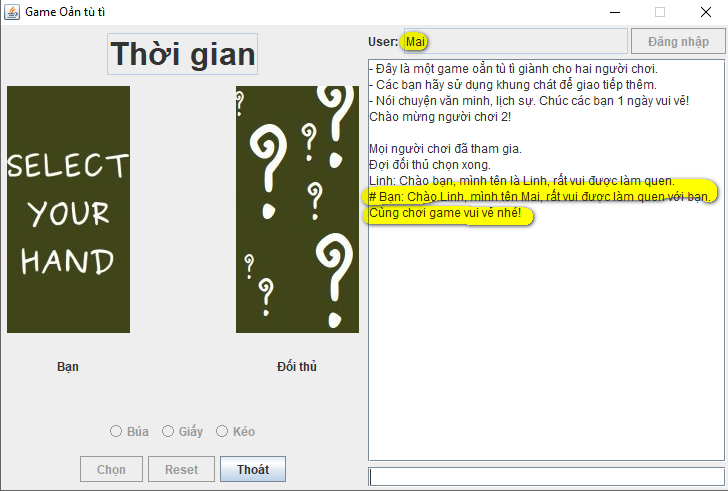


**Figure 11: Chào mừng ngươi chơi 2**

## 5.4 Người chơi trò chuyện



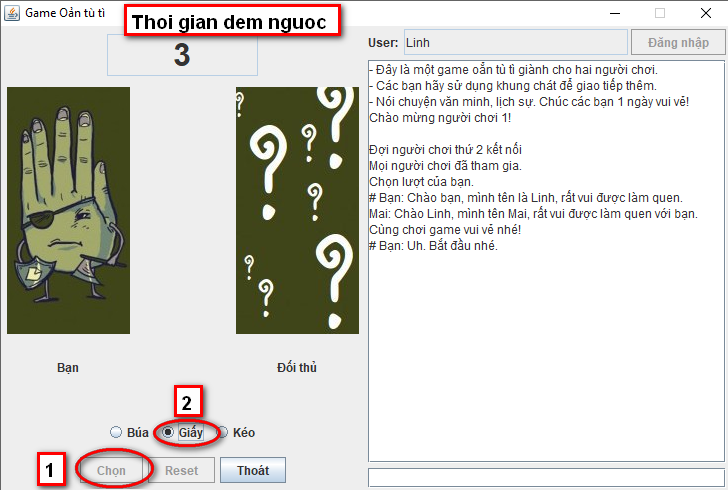
**Figure 12: Người dùng chát 1**



**Figure 13: Người dùng chát 2**

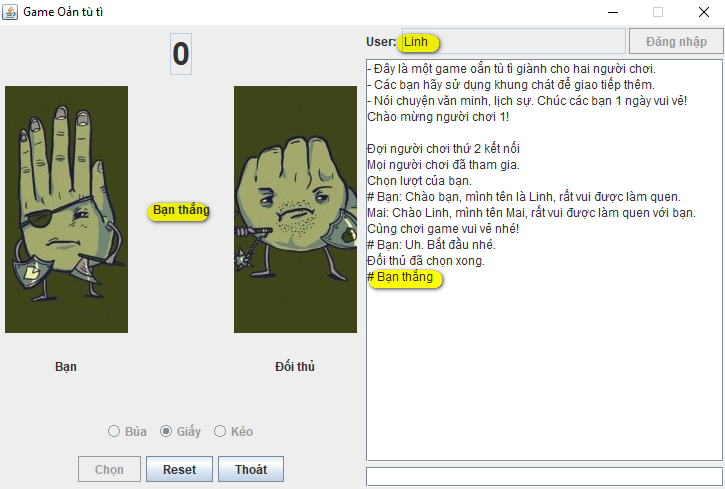
## 5.5 Bắt đầu trò chơi

* Người chơi lựa chọn

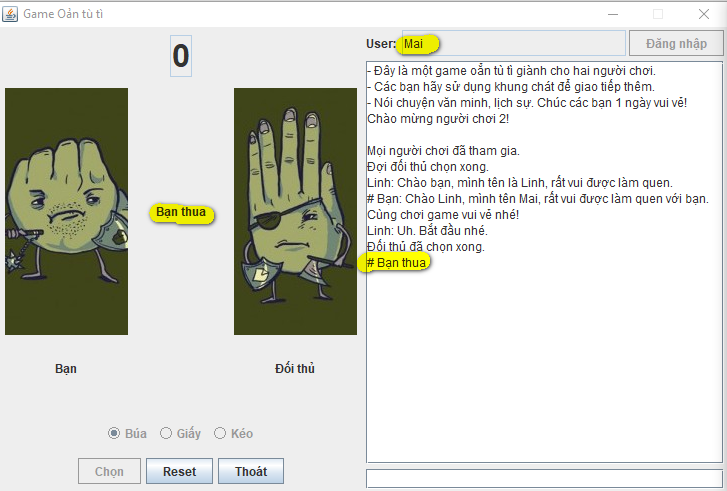


**Figure 14: Bắt đầu game**

* Kết quả thắng-thua

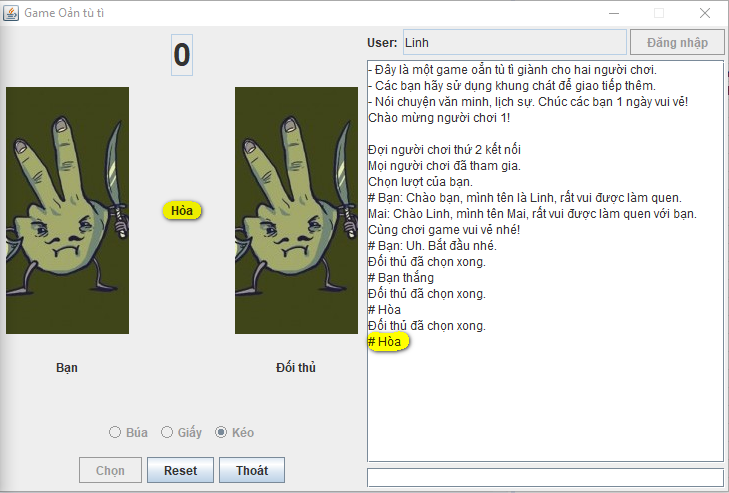


**Figure 15: Kết quả 1**



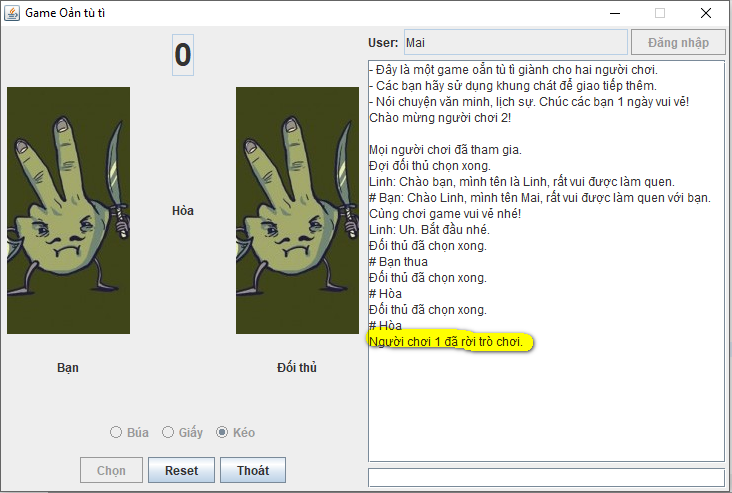
**Figure 16: Kết quả 2**

* Kết quả hòa



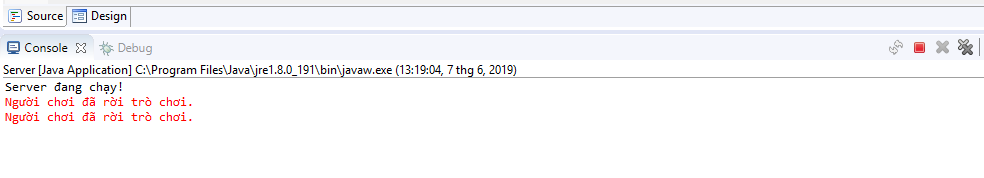
**Figure 17: Kết quả 3**

## Nhận thông báo người chơi còn lại thoát ứng dụng



**Figure 18: Thông báo rời game**

## Server ghi nhận log



**Figure 19: Server lưu log**

# 6 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## 6.1 Kết luận

Phần mềm đã đạt được một số kết quả ban đầu đề ra:

* Cho phép hai người chơi thông qua hai máy khác nhau
* Hai người chơi có thể trò chuyện qua khung chát
* Có hình ảnh minh họa cho phần lựa chọn giúp game sinh động hơn

Tuy nhiên, phần mềm vẫn còn nhiều thiếu xót cần bổ sung sửa chữa như:

* Chưa cho phép người chơi tự lựa chọn đối thủ
* Chưa có thông báo lỗi khi đăng nhập sai, người dùng phải chạy lại ứng dụng.
* Phần layout chính có kích thước cố định.
* Không có phần tỉ số khi hai người chơi
* Phía server chưa có giao diện quản lý
* ...

## 6.2 Hướng phát triển

Phần mềm còn nhiều điểm hạn chế như đã nêu ở phần kết luận. Do đó cần bổ sung thêm để cho ứng dụng thêm hay, hấp dẫn và hoàn thiện hơn nữa.

# 7 TÀI LIỆU THAM KHẢO

1. Xây dựng ứng dụng Client-Server với Socket trong Java.

<https://gpcoder.com/3679-xay-dung-ung-dung-client-server-voi-socket-trong-java/>

1. Networked RockPaperScissors w/ Swing (Java.net)

<http://sharifuddin6.blogspot.com/2014/05/networked-rockpaperscissors.html>

1. Countdown function

<https://docs.oracle.com/javase/7/docs/technotes/guides/lang/Countdown.java>

1. Java Socket Programming Examples

<https://cs.lmu.edu/~ray/notes/javanetexamples/>

1. Create a simple Chat Client/Server Solution

<http://pirate.shu.edu/~wachsmut/Teaching/CSAS2214/Virtual/Lectures/chat-client-server.html>

1. Images

<http://sale.images.woot.com/Rock_Paper_Scissors4xeDetail.png>