

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN I



BÁO CÁO BÀI TẬP LỚN

MÔN: LẬP TRÌNH MẠNG

Giảng viên hướng dẫn: **Ths. Nguyễn Hoàng Anh**

Nhóm lớp: Nhóm 14

Nhóm bài tập lớn: Nhóm 06

Sinh viên thực hiện: Nguyễn Đức Hoàng B17DCAT084

Trần Minh Nhật B17DCAT139

Phạm Hải Vũ B17DCAT214

Nguyễn Công Thành B17DCAT167

Hà Nội, tháng 11, 2020



ĐỀ TÀI
GAME AMONGCHU THI ĐẤU ĐỐI KHÁNG
ONLINE

Giáo viên hướng dẫn:

Ths. Nguyễn Hoàng Anh

Lời cảm ơn

Trước tiên với lòng biết ơn sâu sắc nhất, chúng em xin gửi đến quý thầy cô tại Học viện Công nghệ Bưu chính Viễn thông lời cảm ơn chân thành vì đã cùng với tri thức và tâm huyết của mình để truyền đạt vốn kiến thức quý báu cho chúng em trong thời gian học tập tại học viện.

Trong học kỳ này, Học viện đã cho chúng em được học và tiếp cận với các môn học nền tảng rất hữu ích đối với sinh viên, đặc biệt là môn học Lập trình mạng. Bước đầu đi vào thực tế, tìm hiểu sâu hơn về môn học, kiến thức của chúng em còn hạn chế và các kỹ năng còn nhiều thiếu sót. Chúng em xin chân thành cảm ơn thầy – Ths Nguyễn Hoàng Anh đã tận tâm hướng dẫn chúng em thực hiện và hoàn thành môn học này của mình. Thầy đã luôn tạo điều kiện, nhiệt tình và đưa ra những lời khuyên, bài học bổ ích cho lớp chúng em trong suốt thời gian học môn học.

Cuối cùng chúng em xin kính chúc quý thầy cô đang công tác tại Học viện Công nghệ Bưu chính Viễn thông sức khỏe, niềm tin để tiếp tục thực hiện sứ mệnh cao đẹp của mình là truyền đạt kiến thức cho thế hệ nối tiếp.

Chúng em xin chân thành cảm ơn!

Hà Nội, tháng 11 năm 2020

Nhóm 06

Mục lục

LỜI CẢM ƠN.....	2
MỤC LỤC.....	3
DANH MỤC PHÂN CÔNG CÔNG VIỆC	6
DANH MỤC KÝ HIỆU SỬ DỤNG	7
DANH MỤC VIẾT TẮT	8
DANH MỤC HÌNH ẢNH.....	9
DANH MỤC BẢNG BIỂU	10
1. TỔNG QUAN	11
1.1. Giới thiệu trò chơi AmongChu thi đấu đối kháng online	11
1.2. Phân tích yêu cầu ứng dụng/hệ thống	11
1.2.1. Yêu cầu ứng dụng	11
1.2.2. Yêu cầu hệ thống.....	12
1.2.2.1. Server	12
1.2.2.2. Client.....	12
2. PHÂN TÍCH THIẾT KẾ.....	13
2.1. Phân tích thiết kế tổng quan	13
2.1.1. Kiến trúc tổng quan.....	13
2.1.2. Sơ đồ chức năng.....	13
2.1.3. Biểu đồ Usecase tổng quan	14
2.2. Phân tích thiết kế chi tiết.....	15
2.2.1. Đặc tả Usecase	15
2.2.1.1. Đăng nhập	15




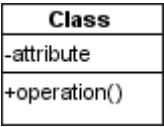
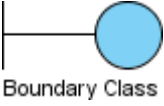


2.2.1.2.	Đăng ký	16
2.2.1.3.	Đăng xuất	17
2.2.1.4.	Gửi thách đấu	17
2.2.1.5.	Phản hồi thách đấu	18
2.2.1.6.	Xem danh sách người dùng	19
2.2.1.7.	Chơi trò chơi	19
2.2.1.8.	Thoát màn chơi	20
2.2.2.	Biểu đồ lớp	21
2.2.3.	Biểu đồ tuần tự	22
2.2.3.1.	Đăng nhập	22
2.2.3.2.	Đăng ký	22
2.2.3.3.	Đăng xuất	23
2.2.3.4.	Gửi thách đấu	24
2.2.3.5.	Phản hồi thách đấu	25
2.2.3.6.	Xem danh sách người dùng	25
2.2.3.7.	Chơi trò chơi	26
2.2.3.8.	Thoát màn chơi	26
2.2.4.	Sơ đồ thực thể quan hệ (ER)	27
3.	KẾT QUẢ ỨNG DỤNG	28
3.1.	Kiến trúc ứng dụng	28
3.1.1.	Giới thiệu Non-blocking IO trong Java (Greenfield, n.d.)	28
3.1.2.	Kiến trúc Server	30
3.1.2.1.	Database	31
3.1.2.2.	ServerQueueRequest	32
3.1.2.3.	ServerTableControl	33
3.1.3.	Kiến trúc Client	34
3.1.3.1.	ResponseHandler	34
3.1.3.2.	LoginControl	34
3.1.3.3.	LobbyControl	34
3.1.3.4.	PikachuController (Phạm Đức Hiên, Hoàng Anh Tuấn, Nguyễn Văn Hân, 2017)	35
3.2.	Cài đặt và triển khai ứng dụng	35
3.2.1.	Cấu trúc thư mục	35
3.2.2.	Triển khai	36
3.2.2.1.	Game Server	36
3.2.2.2.	Website quản trị	37

3.3.	Kết quả thực hiện.....	38
3.4.	Kết luận.....	41
3.4.1.	Tổng kết	41
3.4.2.	Hạn chế cần khắc phục	41
DANH MỤC TÀI LIỆU THAM KHẢO.....		42

Danh mục phân công công việc

STT	Họ và Tên	Phân công công việc	Tỉ lệ hoàn thành
1	Nguyễn Đức Hoàng	Mô tả bài toán Lập trình Server Dựng Database Lắp ráp code của các thành viên khác Deploy trò chơi lên Server Tổng hợp báo cáo	90% 90% 95% 100% 100% 100%
2	Trần Minh Nhật	Tài trợ và cài đặt Server Code và viết báo cáo các phần liên quan đến đăng nhập, đăng ký.	100% 90%
3	Nguyễn Công Thành	Code và viết báo cáo các phần liên quan đến đăng xuất, thách đấu, bảng trạng thái.	90%
4	Phạm Hải Vũ	Code và dựng website quản trị Code và viết báo cáo các phần liên quan đến thoát game, chơi game.	100% 90%

Danh mục ký hiệu sử dụng

KHÁI NIỆM	KÝ HIỆU	Ý NGHĨA
Tác nhân (Actor)		Một người / nhóm người hoặc một thiết bị hoặc hệ thống tác động hoặc thao tác đến chương trình.
Use-case		Một chuỗi các hành động mà hệ thống thực hiện mang lại một kết quả quan sát được đối với actor.
Hệ thống (System)		Biểu hiện phạm vi của hệ thống. Các use-case được đặt trong khung hệ thống.
Lớp (Class)		Là một sự trừu tượng của các đối tượng trong thế giới thực.
Lớp biên (Boundary class)		Nắm giữ sự tương tác giữa phần bên ngoài với phần bên trong của hệ thống (giao diện chương trình).
Lớp điều khiển (Control class)		Thể hiện trình tự xử lý của hệ thống trong một hay nhiều use-case.
Lớp thực thể (Entity class)		Mô hình hóa các thông tin lưu trữ lâu dài trong hệ thống, nó thường độc lập với các đối tượng khác ở xung quanh.

Danh mục viết tắt

STT	Từ ngữ viết tắt	Từ ngữ đầy đủ	Nghĩa tiếng Việt
1	Non-blocking IO	Synchronous Non-blocking Input/Output	

Danh mục hình ảnh

Hình 1: Kiến trúc tổng quan.....	13
Hình 2: Biểu đồ usecase cho tác nhân người chơi.....	14
Hình 3: Biểu đồ usecase cho tác nhân quản trị viên	15
Hình 4: Biểu đồ lớp (Class diagram)	21
Hình 5: Sequence diagram đăng nhập	22
Hình 6: Sequence diagram đăng ký	22
Hình 7: Sequence diagram đăng xuất	23
Hình 8: Sequence diagram gửi thách đấu	24
Hình 9: Sequence diagram phản hồi thách đấu.....	25
Hình 10: Sequence diagram xem danh sách người dùng	25
Hình 11: Sequence diagram chơi trò chơi.....	26
Hình 12: Sequence diagram thoát màn chơi	26
Hình 13: Sơ đồ thực thể quan hệ (ER).....	27
Hình 14: Kiến trúc máy chủ sử dụng Blocking socket	28
Hình 15: Kiến trúc tổng quan máy chủ sử dụng Non-blocking socket.....	29
Hình 16: Cấu trúc máy chủ cơ bản sử dụng Non-blocking socket	29
Hình 17: Kiến trúc Server tổng quan	30
Hình 18: Cấu trúc cơ sở dữ liệu	31
Hình 19: Quá trình lập lịch	32
Hình 20: Cấu trúc ServerTableControl	33
Hình 21: Kiến trúc Client tổng quan.....	34
Hình 22: Mở cổng trên Google Cloud cho trò chơi.....	36
Hình 23: Mở cổng trong Windows Server cho trò chơi	37
Hình 24: Mở cổng trên Google Cloud cho website quản trị.....	37
Hình 25: Mở cổng website quản trị trên Windows Server	38
Hình 26: Giao diện đăng nhập	38
Hình 27: Giao diện sảnh chờ.....	39
Hình 28: Giao diện trò chơi	40
Hình 29: Giao diện website quản trị	40

Danh mục bảng biểu

Bảng 1: Sơ đồ chức năng	13
Bảng 2: Đặc tả usecase đăng nhập	16
Bảng 3: Đặc tả usecase đăng ký.....	17
Bảng 4: Đặc tả usecase đăng xuất.....	17
Bảng 5: Đặc tả usecase gửi thách đấu.....	18
Bảng 6: Đặc tả usecase phản hồi thách đấu	19
Bảng 7: Đặc tả usecase xem danh sách người dùng	19
Bảng 8: Đặc tả usecase chơi trò chơi	20
Bảng 9: Đặc tả usecase thoát màn chơi.....	21

1. Tổng quan

1.1. Giới thiệu trò chơi AmongChu thi đấu đối kháng online

Game Pikachu là một trong những trò chơi kinh điển, mặc dù đã được ra mắt 17 năm nhưng sức hút của trò chơi này vẫn không hề bị giảm nhiệt. Với lối chơi cuốn hút, kịch tính hiện trò chơi Pikachu vẫn là sự lựa chọn hàng đầu khi người chơi muốn xả stress sau những giờ làm việc căng thẳng và mệt mỏi. Trong phiên bản này, nhóm chúng em mong muốn giới thiệu tới thầy và các bạn một phiên bản đối kháng của trò chơi Pikachu cổ điển kết hợp với những hình ảnh từ một tựa trò chơi rất nổi tiếng hiện nay mang tên “Among Us”.

1.2. Phân tích yêu cầu ứng dụng/hệ thống

1.2.1. Yêu cầu ứng dụng

Để tham gia trò chơi AmongChu thi đấu đối kháng online, người chơi sẽ cần sử dụng một tài khoản đăng nhập vào hệ thống. Tài khoản này có thể được tạo thông qua chức năng Đăng kí. Server sau đó sẽ gửi tới người chơi thông tin về những người chơi khác đang online, điểm số hiện tại của họ cũng như của cá nhân người chơi. Người chơi có thể sử dụng tính năng thách đấu để mời một người chơi vào trận đấu với mình, đối phương nhận được lời mời có thể từ chối hoặc chấp nhận lời thách đấu.

Khi đối phương chấp nhận thách đấu, cả hai sẽ cùng được cung cấp một màn chơi AmongChu 10x10 ô là hình ảnh các nhân vật trong tựa trò chơi AmongUs.

Luật chơi:

- Người chơi cần tìm 2 ô có hình ảnh nhân vật AmongUs giống nhau và nói chúng lại để chúng biến mất khỏi màn hình, điều kiện để 2 ô này nối thành công với nhau là chúng cần phải cách nhau không quá 3 đường gấp khúc và trên đường này không chứa vật cản là bất kì một ô nào khác.
- Trò chơi sẽ kết thúc khi một trong hai người chơi hoàn thành việc nối toàn bộ những ô hình nhân vật có trong bàn chơi hoặc khi thời gian kết thúc với kết quả hòa. Thời gian giới hạn cho mỗi bàn chơi là 2 phút. Qua mỗi bàn chơi, người chơi sẽ nhận được những điểm số tương ứng:
 - o Thắng nhận 1 điểm.
 - o Thua nhận 0 điểm.
 - o Hòa mỗi người chơi nhận 0.5 điểm.
- Điều kiện chiến thắng:
 - o Người chơi hoàn thành bàn chơi trước đối thủ.
 - o Đối thủ thoát bàn chơi.
 - o Cả hai cùng không hoàn thành bàn chơi trước 2 phút, người chơi chiến thắng sẽ là người có số điểm cao hơn.

- Kết quả hòa chỉ xảy ra khi cả hai cùng không hoàn thành bàn chơi và cùng có số điểm sau 2 phút.
- Sau khi kết thúc bàn chơi, người dùng sẽ được hỏi có muốn tiếp tục thách đấu với đối phương hay không, nếu cả hai trả lời có, hệ thống sẽ lại xếp cặp và tạo trận đấu giữa hai người chơi.

Kết quả các trận đấu được lưu vào Server. Mỗi người chơi đều có thể vào xem bảng xếp hạng các người chơi trong toàn bộ hệ thống, theo lần lượt các tiêu chí: tổng số điểm (giảm dần), trung bình điểm của các đối thủ đã gặp (giảm dần), trung bình thời gian kết thúc trong các trận thắng (tăng dần).

1.2.2. Yêu cầu hệ thống

1.2.2.1. Server

- Máy chủ có cài đặt MySQL Server, Java JDK, MySQL Connector J
- Thư viện bổ sung cần có: org.apache.commons.collections4
- Kết nối Internet, cần mở 2 port cho việc giao tiếp.

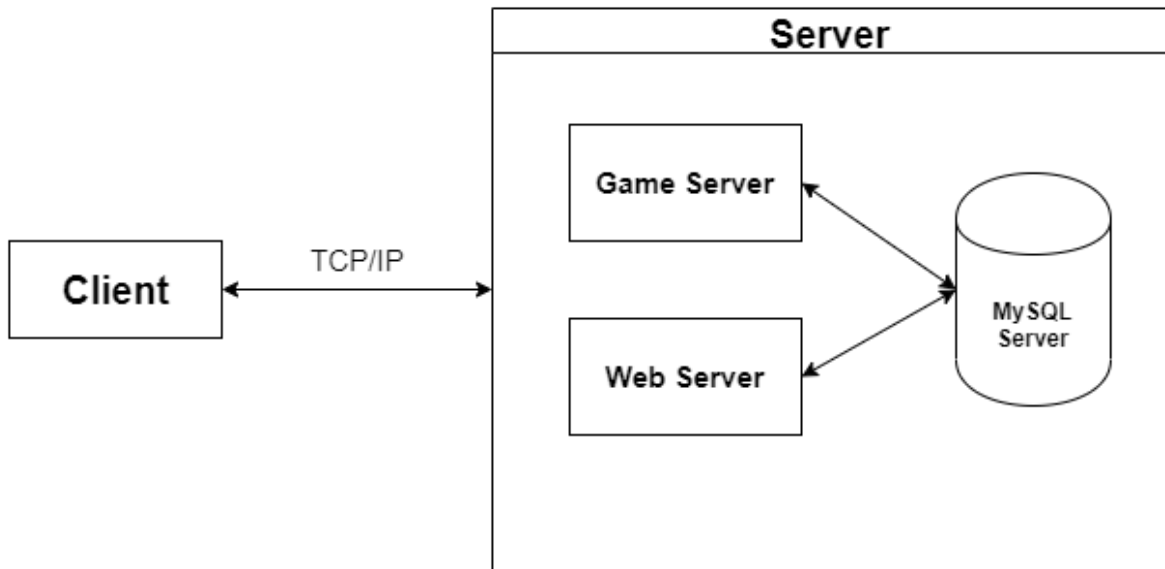
1.2.2.2. Client

- Một thiết bị có cài đặt Java, màn hình tối thiểu 720x460 pixel.
- Kết nối Internet.

2. Phân tích thiết kế

2.1. Phân tích thiết kế tổng quan

2.1.1. Kiến trúc tổng quan



Hình 1: Kiến trúc tổng quan

2.1.2. Sơ đồ chức năng

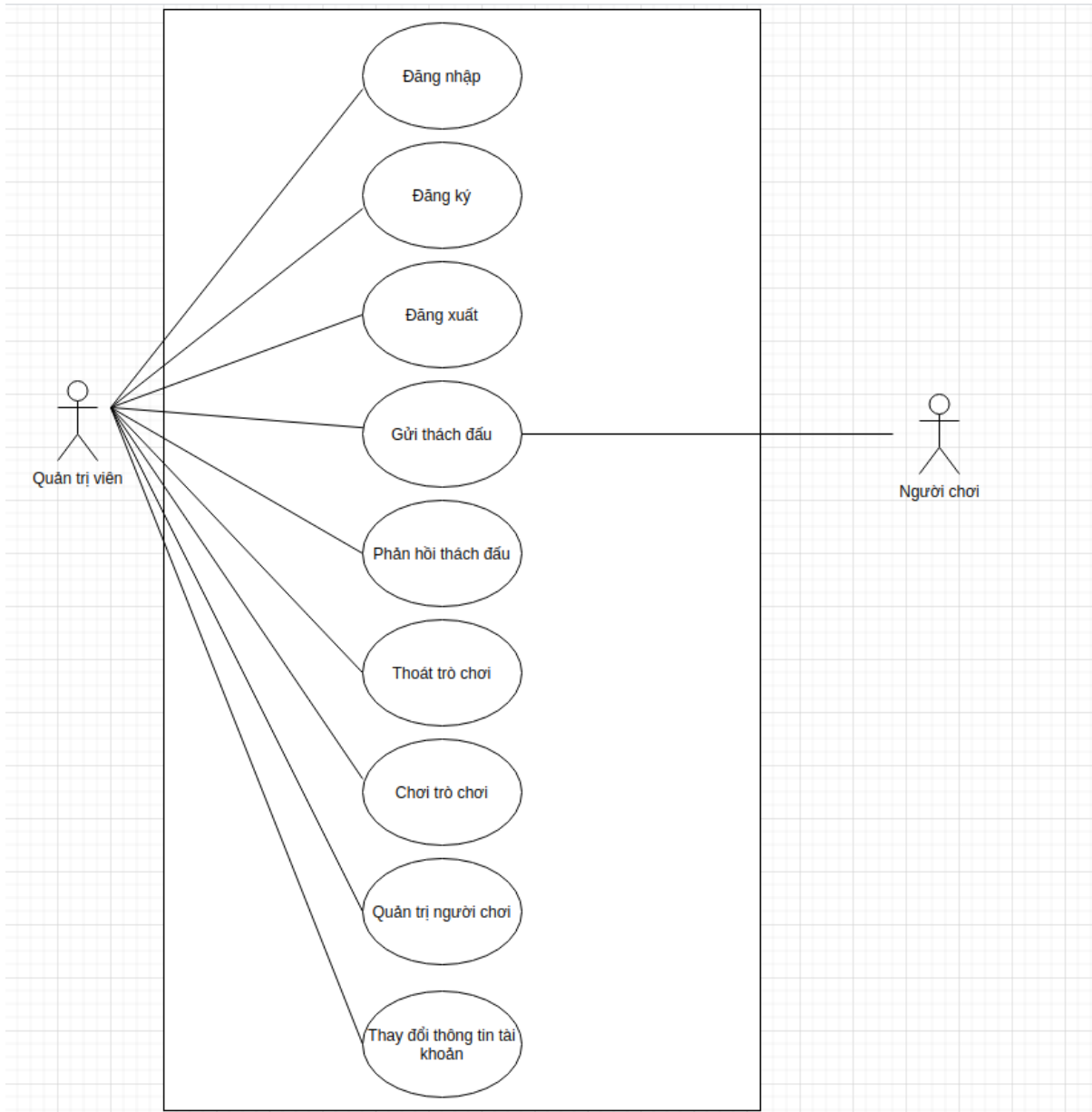
STT	Chức năng	Người chơi	Quản trị hệ thống
1	Đăng nhập	x	x
2	Đăng ký	x	x
3	Đăng xuất	x	x
4	Gửi thách đấu	x	x
5	Phản hồi thách đấu	x	x
6	Thoát trò chơi	x	x
7	Chơi trò chơi	x	x
8	Quản lý người chơi		x
9	Thay đổi thông tin tài khoản		x

Bảng 1: Sơ đồ chức năng

2.1.3. Biểu đồ Usecase tổng quan



Hình 2: Biểu đồ usecase cho tác nhân người chơi



Hình 3: Biểu đồ usecase cho tác nhân quản trị viên

2.2. Phân tích thiết kế chi tiết

2.2.1. Đặc tả Usecase

2.2.1.1. Đăng nhập

Tên Use Case	Login
Đặc tả	Use case cho phép người dùng đăng nhập vào hệ thống.
Actor	Người chơi

Điều kiện kích hoạt	Khi người dùng mở ứng dụng.
Tiền điều kiện	Người dùng phải có tài khoản.
Hậu điều kiện	Người dùng đăng nhập thành công.
Luồng sự kiện chính	<ol style="list-style-type: none"> 1. Hệ thống hiển thị giao diện đăng nhập. 2. Người dùng nhập tài khoản, mật khẩu và bấm nút “Đăng nhập”. 3. Hệ thống kiểm tra thông tin đăng nhập. 4. Thông báo "Đăng nhập thành công" và điều hướng ra màn hình Lobby.
Luồng sự kiện phụ	<p>3a. Sai thông tin đăng nhập: người dùng nhập sai tài khoản hoặc mật khẩu:</p> <ol style="list-style-type: none"> 1. Hiện thị lại giao diện đăng nhập kèm thông báo "Tài khoản hoặc mật khẩu không chính xác". 2. Quay lại bước 2 trong luồng sự kiện chính. <p>3b. Tài khoản đã đăng nhập:</p> <ol style="list-style-type: none"> 1. Hiện thị lại giao diện đăng nhập kèm thông báo "Tài khoản đã đăng nhập ở vị trí khác". 2. Quay lại bước 2 trong luồng sự kiện chính.

Bảng 2: Đặc tả usecase đăng nhập

2.2.1.2. Đăng ký

Tên Use Case	Đăng ký
Đặc tả	Use case cho phép người dùng đăng ký tài khoản.
Actor	Người chơi
Điều kiện kích hoạt	Khi người dùng mở ứng dụng.
Tiền điều kiện	Người dùng đã mở ứng dụng.
Hậu điều kiện	Người dùng đăng ký tài khoản thành công.
Luồng sự kiện chính	<ol style="list-style-type: none"> 1. Hệ thống hiển thị giao diện đăng nhập

	2. Người dùng nhập tài khoản, mật khẩu và bấm nút “Đăng ký”. 3. Hệ thống kiểm tra thông tin tài khoản. 4. Thông báo "Tạo mới tài khoản thành công.
Luồng sự kiện phụ	3a. Tài khoản đã tồn tại: 1. Hiện thị thông báo "Tên tài khoản đã có người sử dụng".

Bảng 3: Đặc tả usecase đăng ký

2.2.1.3. Đăng xuất

Tên Use Case	Đăng xuất
Mô tả	Use case cho phép người dùng đăng xuất khỏi trò chơi
Actor	Người dùng
Điều kiện kích hoạt	Khi Người dùng bấm vào nút “Đăng xuất” trên giao diện của client.
Tiền điều kiện	Người dùng đã đăng nhập vào trò chơi.
Hậu điều kiện	Người dùng đăng xuất thành công.
Luồng sự kiện chính	1. Người dùng bấm vào nút “Đăng xuất” trên giao diện của client. 2. Client đóng kết nối với server và đưa người dùng đến giao diện đăng nhập. 3. Server xóa các request còn tồn tại của người dùng và cập nhật trạng thái offline vào cơ sở dữ liệu.
Luồng sự kiện phụ	

Bảng 4: Đặc tả usecase đăng xuất

2.2.1.4. Gửi thách đấu

Tên Use Case	Gửi thách đấu
Mô tả	Use case cho phép người dùng thách đấu một người dùng khác

Actor	Người dùng
Điều kiện kích hoạt	Khi Người dùng bấm vào nút “Thách đấu” sau khi đã chọn đối tượng thách đấu trên giao diện của client.
Tiền điều kiện	Người dùng đã đăng nhập vào trò chơi.
Hậu điều kiện	Người dùng gửi yêu cầu thách đấu thành công.
Luồng sự kiện chính	<ol style="list-style-type: none"> 1. Người dùng bấm vào nút “Thách đấu” sau khi đã chọn đối tượng thách đấu trên giao diện của client. 2. Client gửi yêu cầu thách đấu đến server. 3. Server tiến hành kiểm tra đối tượng được thách đấu có khả dụng không (Online và hiện tại đang không chơi game) 4. Server gửi yêu cầu thách đấu đến cho đối tượng được thách đấu 5. Client của đối tượng được thách đấu sẽ hiển thị thông báo thách đấu.
Luồng sự kiện phụ	<p>3.1 Server kiểm tra người được thách đấu không khả dụng</p> <p>4.1 Server trả lại thông báo thất bại cho người dùng</p>

Bảng 5: Đặc tả usecase gửi thách đấu

2.2.1.5. Phản hồi thách đấu

Tên Use Case	Phản hồi thách đấu
Mô tả	Use case cho phép người dùng trả lời thách đấu một người dùng khác
Actor	Người dùng
Điều kiện kích hoạt	Khi Người dùng được người dùng khác thách đấu.
Tiền điều kiện	Người dùng đã đăng nhập vào trò chơi và nhận được thông báo thách đấu.
Hậu điều kiện	Người dùng gửi phản hồi thách đấu thành công.
Luồng sự kiện chính	<ol style="list-style-type: none"> 1. Người dùng lựa chọn “Đồng ý” trên cửa sổ thông báo. 2. Client gửi phản hồi về cho server 3. Server sửa trạng thái của người dùng và người thách đấu thành “đang chơi” 4. Server gửi ma trận trò chơi về Client của cả 2 người 5. Client dựa vào ma trận được gửi về để tạo trò chơi.

Luồng sự kiện phụ	

Bảng 6: Đặc tả usecase phản hồi thách đấu

2.2.1.6. Xem danh sách người dùng

Tên Use Case	Xem danh sách người dùng
Mô tả	Use case cho phép người dùng xem danh sách những người dùng đang online
Actor	Người dùng
Điều kiện kích hoạt	Người dùng đã đăng nhập vào trò chơi.
Tiền điều kiện	Người dùng đã đăng nhập vào trò chơi.
Hậu điều kiện	Người dùng xem danh sách người dùng.
Luồng sự kiện chính	<ol style="list-style-type: none"> 1. Người dùng đăng nhập vào trò chơi. 2. Client mở kết nối đến Server 3. Server đưa người dùng vào danh sách các Client cần cập nhật bảng trạng thái 4. Server lấy dữ liệu danh sách người dùng ở cơ sở dữ liệu và gửi về cho các Client trong danh sách
Luồng sự kiện phụ	

Bảng 7: Đặc tả usecase xem danh sách người dùng

2.2.1.7. Chơi trò chơi

Tên Use Case	Chơi game.
Mô tả	Hai người chơi bắt đầu chơi game. Mỗi người sẽ chọn các cặp hình giống nhau trong giao diện màn chơi. Mỗi lần chọn được một cặp giống nhau, điểm của người chơi sẽ tăng lên. Khi số cặp hình đã được chọn bằng 32, màn chơi sẽ kết thúc và người hoàn thành màn chơi trước sẽ được tính kết quả thắng
Actor	Người chơi.
Điều kiện kích hoạt	Khi có 1 người chơi thách đấu 1 người chơi khác thành công và người chơi được thách đấu chấp nhận lời thách đấu thành công
Tiền điều kiện	Người chơi đã đăng nhập vào giao diện sảnh chờ thành công.
Hậu điều kiện	Nghệ sĩ chơi quay lại giao diện sảnh chờ

Luồng Sự kiện chính	<ol style="list-style-type: none"> 1. Client tạo một ma trận 10x10 các hình ảnh ngẫu nhiên và hiển thị ra giao diện màn chơi cho người chơi. 2. Người chơi click chọn từng cặp hình giống nhau trong ma trận. 3. Client xác nhận xem các hình trong 1 cặp được người chơi chọn có giống nhau hay không. 4. Nếu 2 hình trong 1 cặp giống nhau, Client sẽ ẩn 2 hình đó trên giao diện màn chơi đồng thời tăng điểm cho người chơi và tăng số cặp giống nhau đã được chọn. 5. Khi 1 trong 2 người chơi đạt được 32 cặp giống nhau trước, Client sẽ gửi một message thông báo kết quả và người thắng cho Server. 6. Server tiến hành tăng điểm trong hệ thống cho người thắng và gửi lại cho Client một message để hiển thị thông báo kết quả cho người chơi. 7. Cả 2 người chơi sẽ được đưa về giao diện sảnh chờ
Luồng sự kiện phụ	4a. Hai hình trong một cặp không giống nhau: <ol style="list-style-type: none"> 1. Client sẽ hủy chọn hai hình đó.

Bảng 8: Đặc tả usecase chơi trò chơi

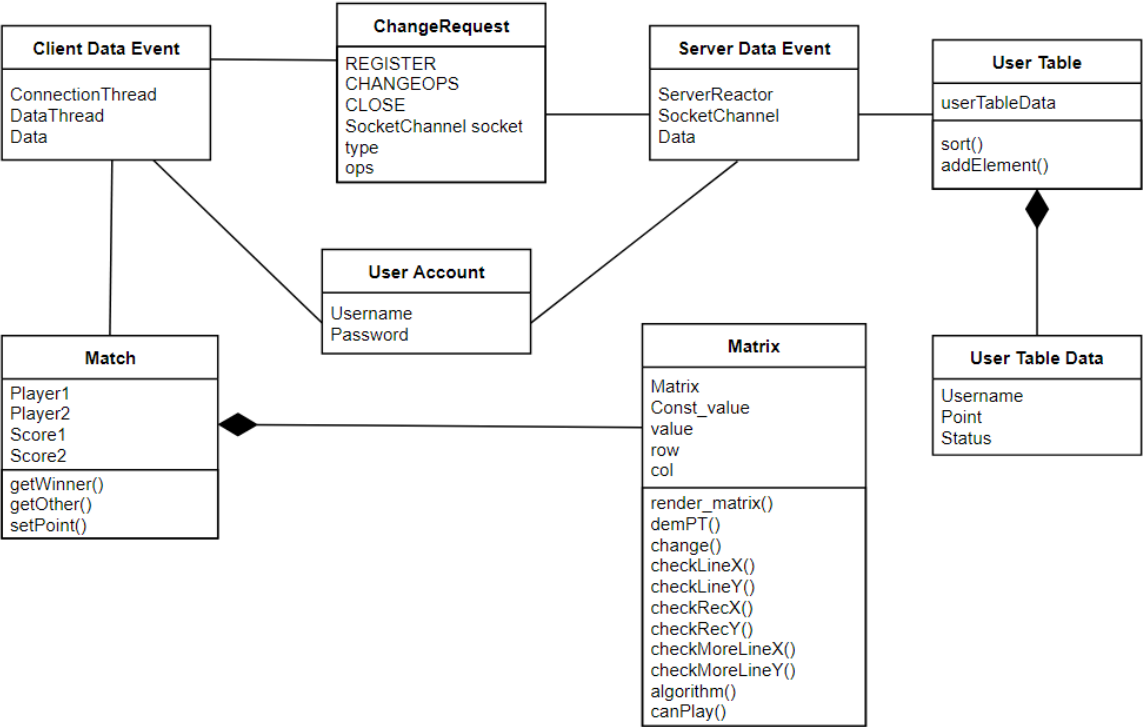
2.2.1.8. Thoát màn chơi

Tên Use Case	Thoát màn chơi.
Mô tả	Người chơi sẽ thoát khỏi màn chơi và quay lại giao diện sảnh chờ. Nếu chỉ có 1 người thoát khỏi màn chơi, người thoát khỏi sẽ bị xử thua, người còn lại sẽ được xử thắng. Nếu cả 2 người chơi cùng thoát khỏi màn chơi thì sẽ được xử hòa.
Actor	Người chơi.
Điều kiện kích hoạt	Khi một trong hai hoặc cả hai người chơi nhấn nút thoát trong cửa sổ màn chơi.
Tiền điều kiện	Người chơi đã đăng nhập và bắt đầu màn chơi thành công.
Hậu điều kiện	Người chơi quay lại giao diện sảnh chờ
Luồng Sự kiện chính	<ol style="list-style-type: none"> 1. Người chơi nhấn nút thoát ở giao diện màn chơi 2. Client sẽ gửi một message thông báo cho Server người chơi yêu cầu thoát. 3. Server xác định xem có bao nhiêu người chơi nhấn nút thoát và gửi 1 message về cho Client thông báo kết quả. 4. Server cộng thêm điểm cho người thắng và đặt lại trạng thái online cho cả 2 người chơi
Luồng sự kiện phụ	4a. Chỉ có một người nhấn nút thoát:

	<ol style="list-style-type: none"> 1. Người thoát sẽ lập tức được đưa trở về giao diện sảnh chờ. 2. Người còn lại sẽ được đưa trở về giao diện sảnh chờ và nhận được thông báo “Đối thủ đã thoát”. <p>4b. Cả 2 người cùng nhấn nút thoát:</p> <ol style="list-style-type: none"> 1. Cả 2 người chơi sẽ cùng được đưa về giao diện sảnh chờ và nhận được thông báo kết quả hòa.
--	---

Bảng 9: Đặc tả usecase thoát màn chơi

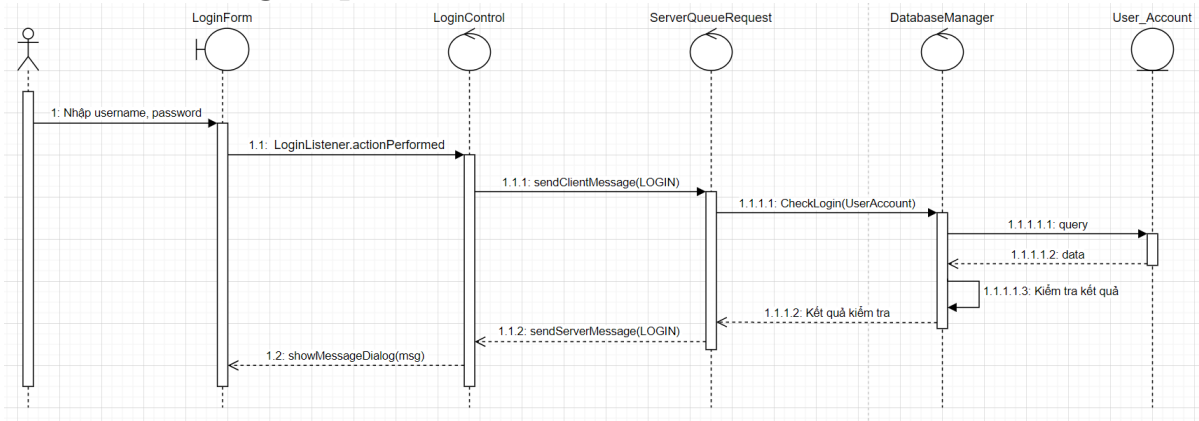
2.2.2. Biểu đồ lớp



Hình 4: Biểu đồ lớp (Class diagram)

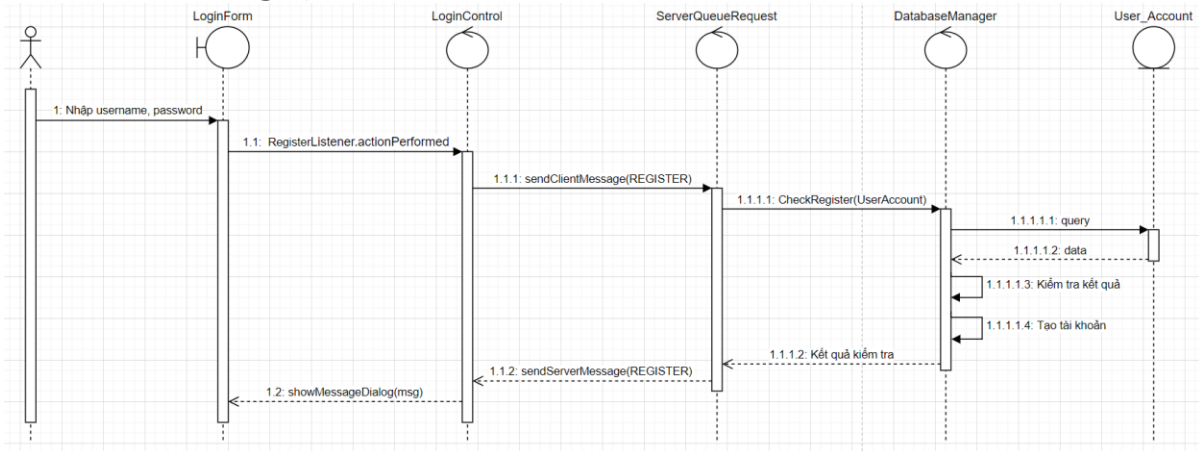
2.2.3. Biểu đồ tuần tự

2.2.3.1. Đăng nhập



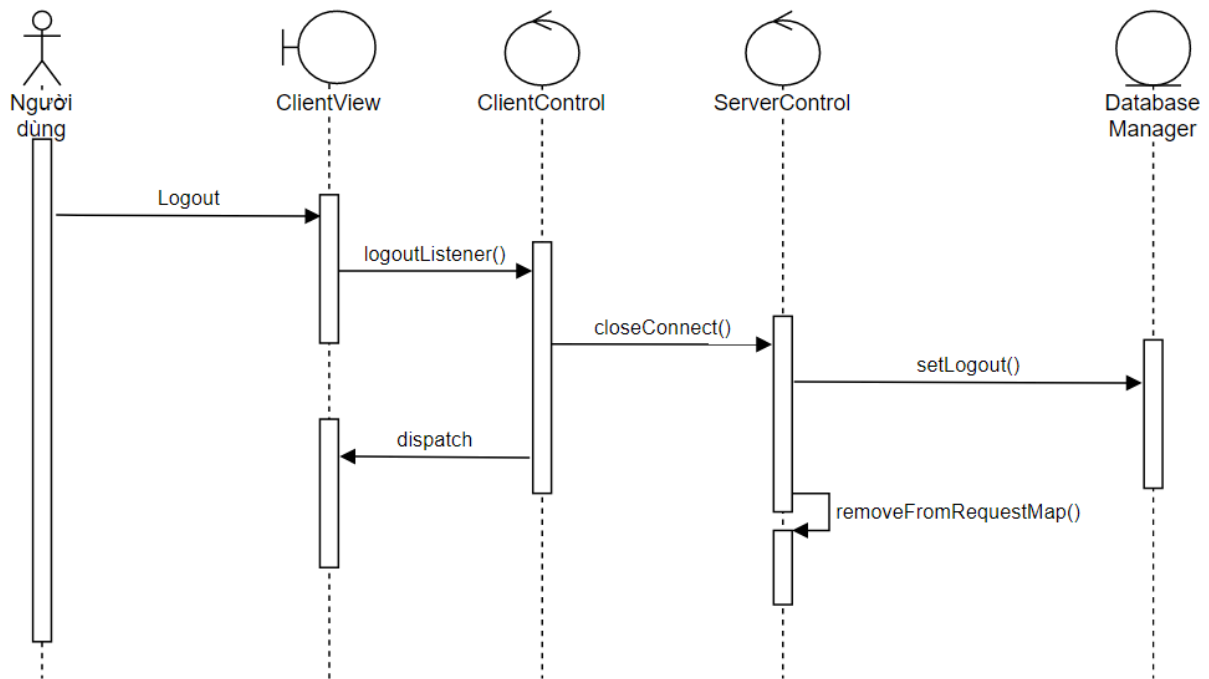
Hình 5: Sequence diagram đăng nhập

2.2.3.2. Đăng ký



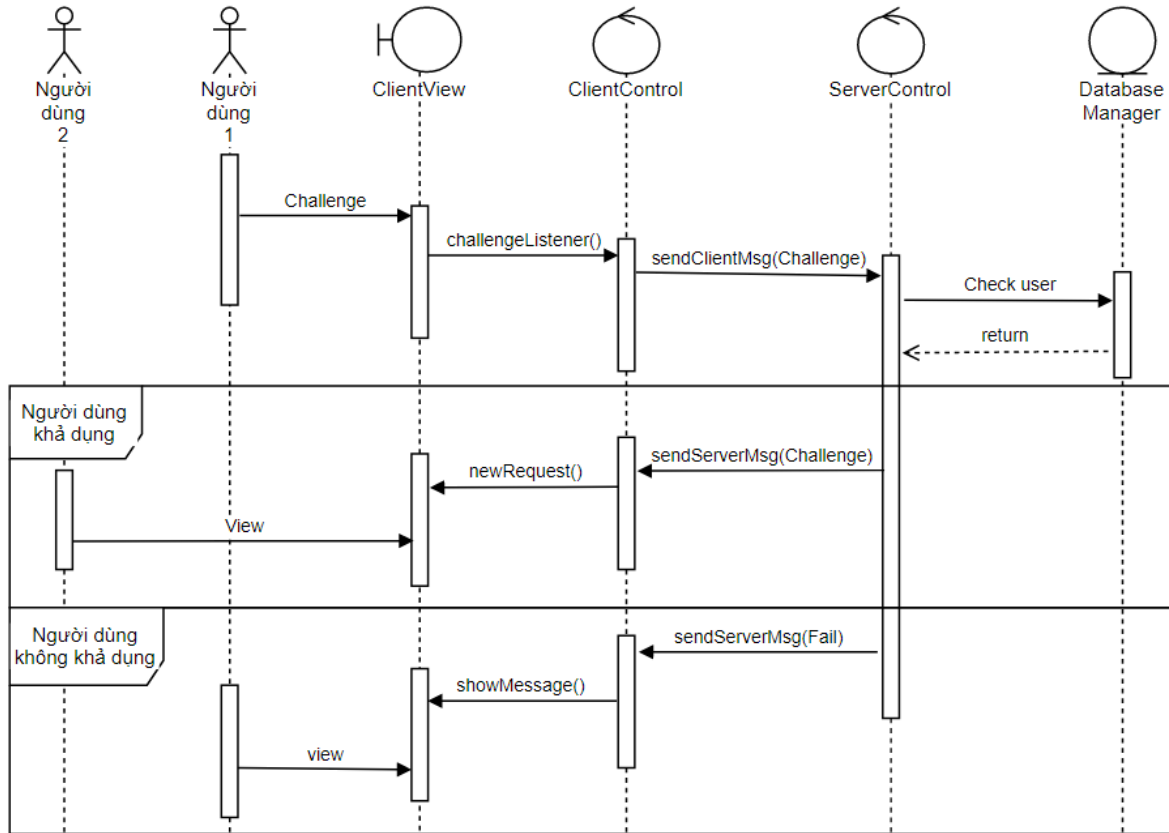
Hình 6: Sequence diagram đăng ký

2.2.3.3. Đăng xuất



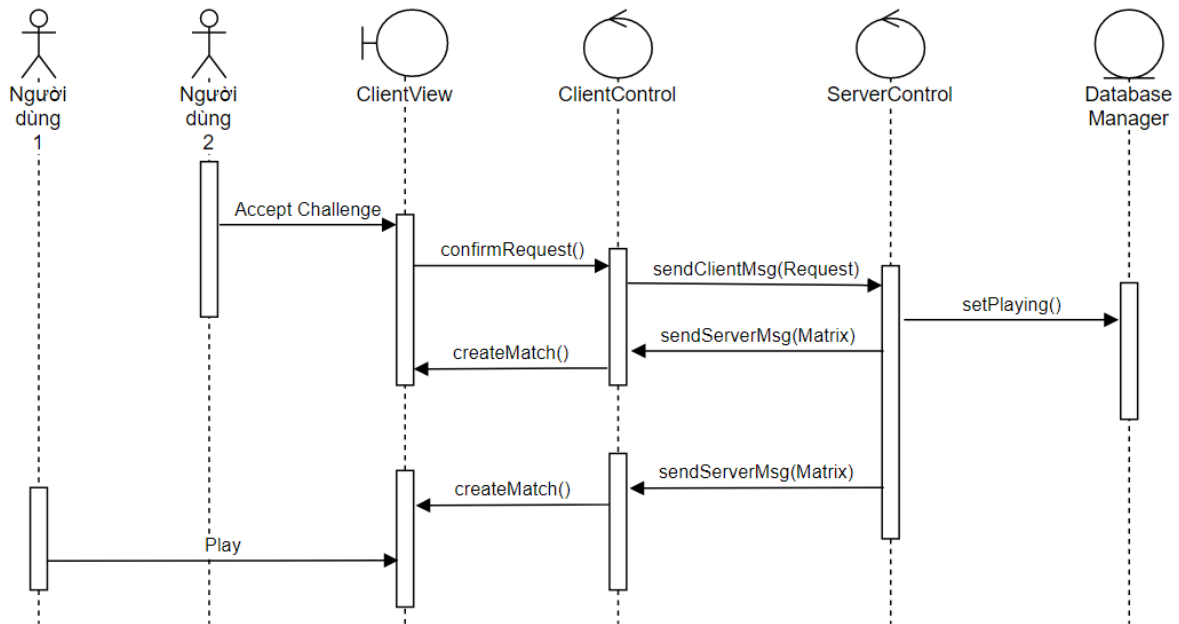
Hình 7: Sequence diagram đăng xuất

2.2.3.4. Gửi thách đấu



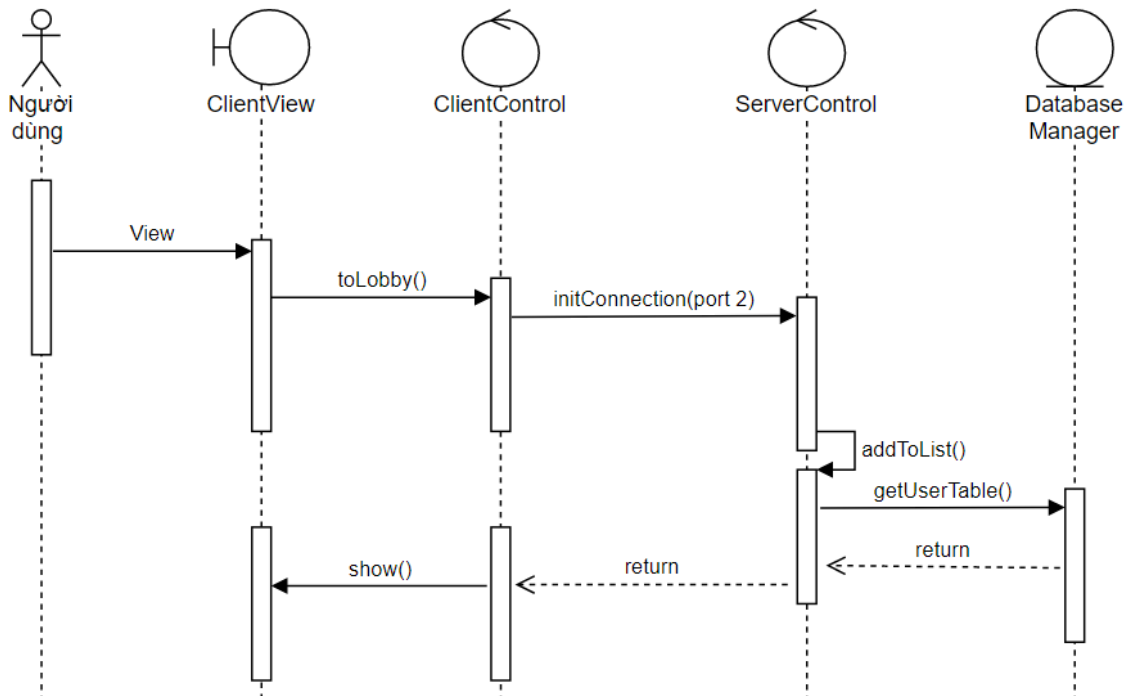
Hình 8: Sequence diagram gửi thách đấu

2.2.3.5. Phản hồi thách đấu



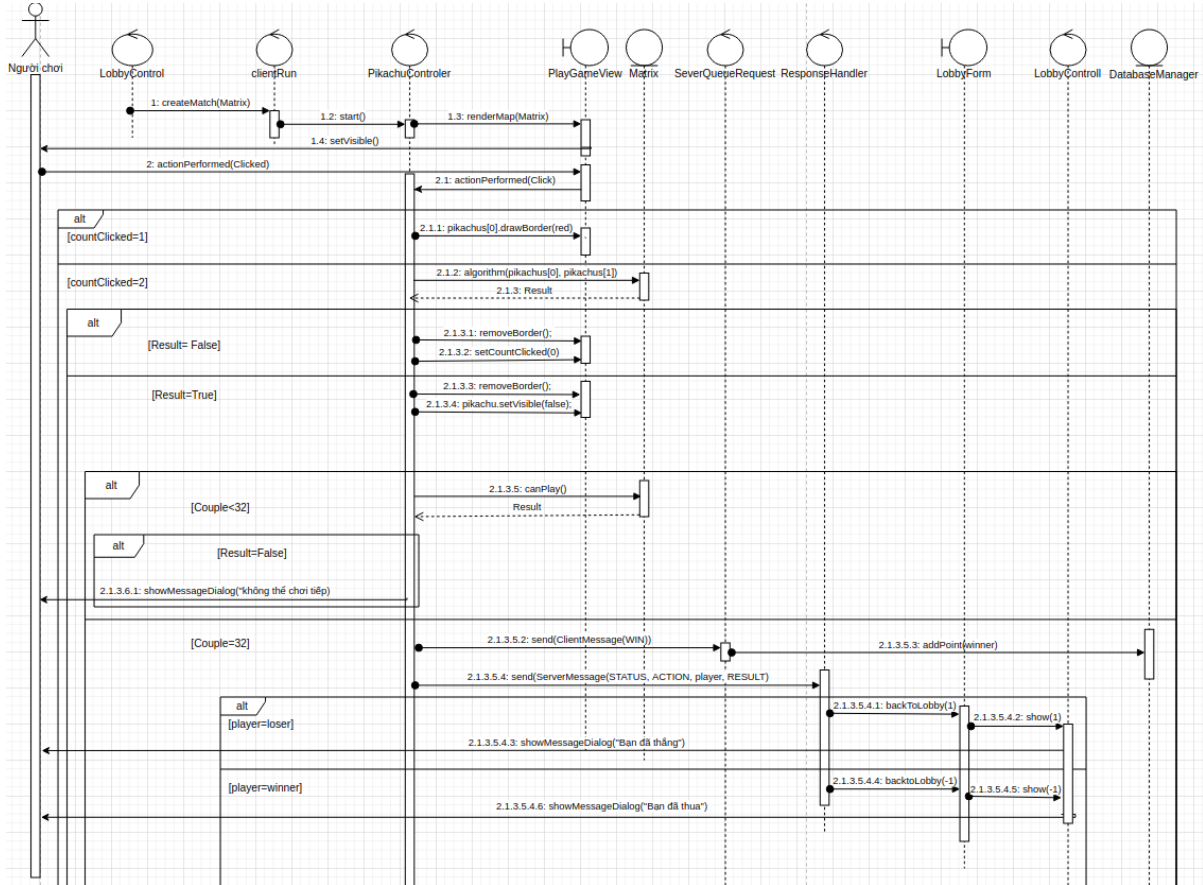
Hình 9: Sequence diagram phản hồi thách đấu

2.2.3.6. Xem danh sách người dùng



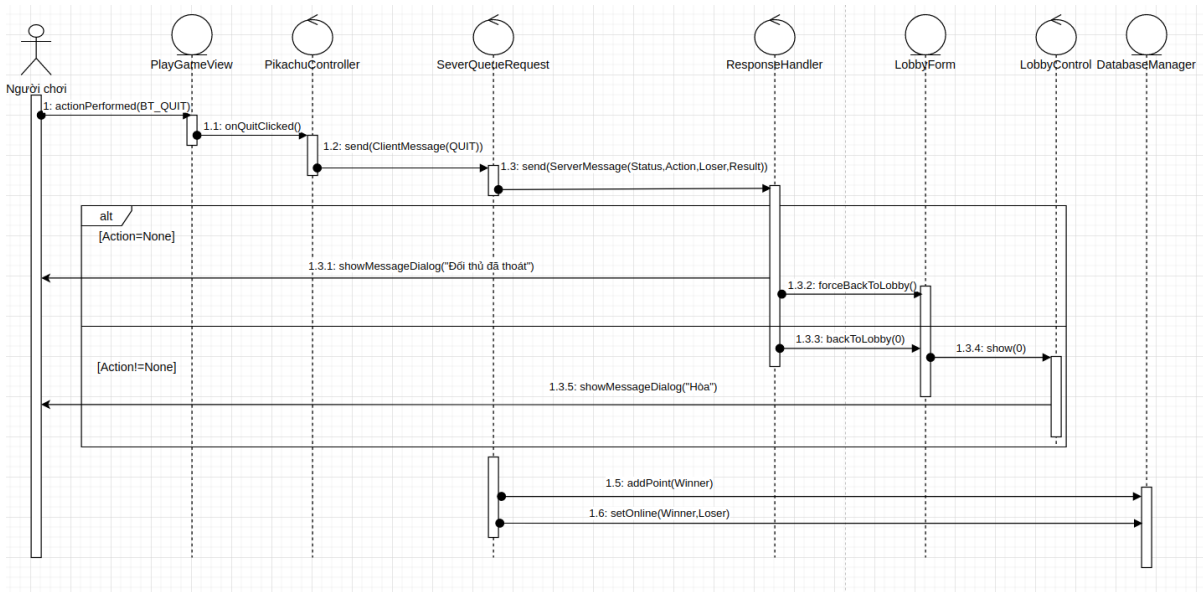
Hình 10: Sequence diagram xem danh sách người dùng

2.2.3.7. Chơi trò chơi



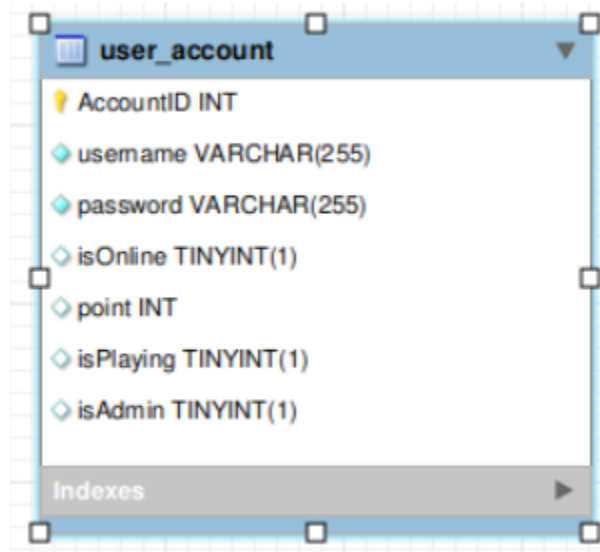
Hình 11: Sequence diagram chơi trò chơi

2.2.3.8. Thoát màn chơi



Hình 12: Sequence diagram thoát màn chơi

2.2.4. Sơ đồ thực thể quan hệ (ER)



Hình 13: Sơ đồ thực thể quan hệ (ER)

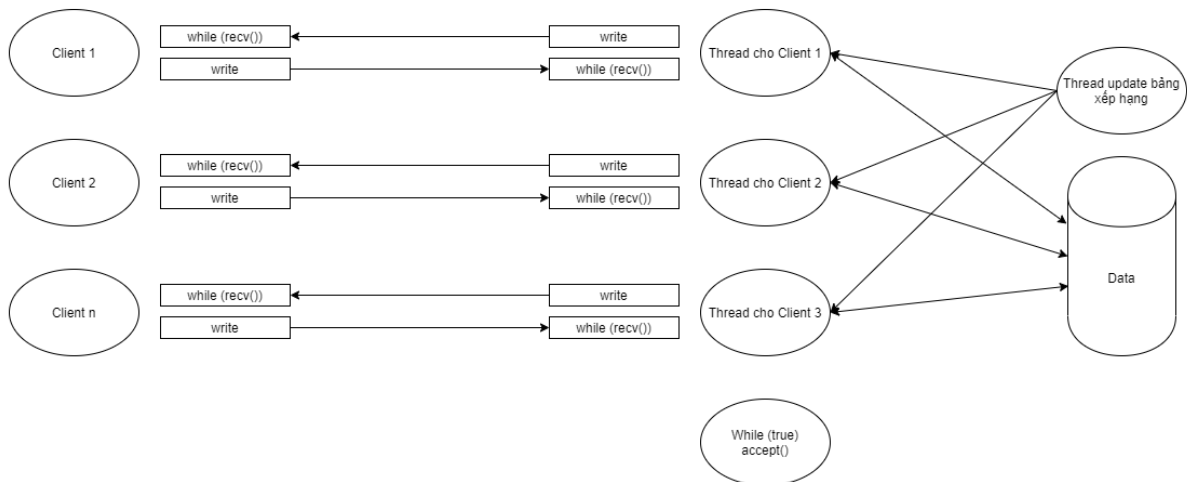
3. Kết quả ứng dụng

3.1. Kiến trúc ứng dụng

3.1.1. Giới thiệu Non-blocking IO trong Java (Greenfield, n.d.)

Trong phần này, chúng em mong muốn giới thiệu với người đọc về một kiến trúc được sử dụng khá rộng rãi trong việc dựng máy chủ hiện nay, đó chính là Synchronous Non-blocking IO. Trong phạm vi bài tập lớn này, nhóm chúng em sẽ sử dụng thư viện java.nio được hỗ trợ trong ngôn ngữ lập trình Java để tạo nên kiến trúc chính trong ứng dụng của mình.

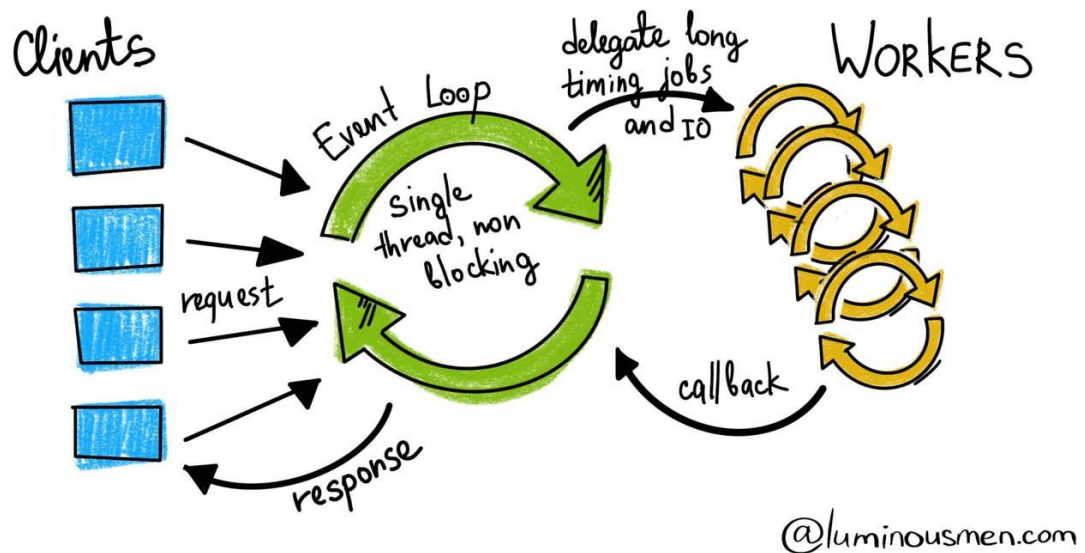
Đầu tiên, chúng em sẽ chỉ ra những hạn chế mà nhóm đã gặp phải trong việc sử dụng Blocking IO socket:



Hình 14: Kiến trúc máy chủ sử dụng Blocking socket

- Với đa số máy chủ thông dụng theo kiến trúc cũ này, thời gian và khả năng xử lý của CPU hầu hết được dành cho việc IO hay đơn giản là không làm gì cả. Với mỗi Client riêng biệt, Server sẽ phải dành ra 2 Thread: Một cho việc xử lý và ghi dữ liệu, một cho việc đọc dữ liệu tương ứng. Điều này gây ra sự lãng phí hiệu năng rất lớn khi mà số lượng Client tăng lên hoặc tăng số cổng/số host mà Server/Client phải kết nối đến.
- Vấn đề về Race condition: Với kiến trúc trên, các Thread của mỗi Client sẽ hoạt động độc lập. Tuy nhiên một số dữ liệu và bộ nhớ vẫn được dùng chung ví dụ như: danh sách Client, bảng xếp hạng,... Điều này đặt ra một bài toán rất lớn về việc xử lý một khối lượng khổng lồ các lỗi Race condition có thể gặp phải hay chính xác hơn là bài toán lập lịch cho Server.

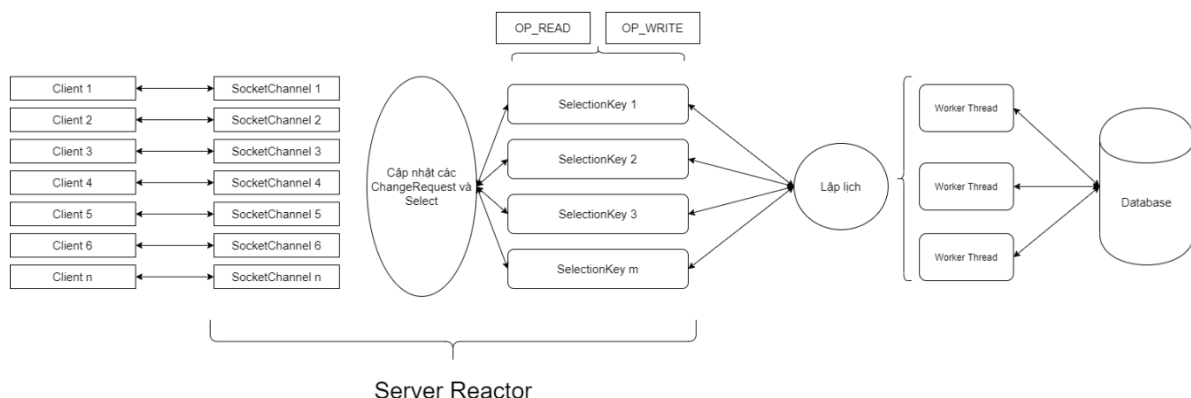
Để khắc phục những điều trên, nhóm chúng em đã sử dụng Non-blocking IO trong Java với kiến trúc như sau: (luminousmen, 2020)



Hình 15: Kiến trúc tổng quan máy chủ sử dụng Non-blocking socket

Đây là một kiến trúc tách biệt hoàn toàn việc IO và xử lý dữ liệu. Nó có thể chia ra 2 phần:

- Server Reactor: Bao gồm 1 thread với chức năng select các request cần được đọc/ghi và xử lý các thao tác tương ứng đối với mỗi Client.
- Server Worker: Có thể bao gồm 1 hoặc nhiều thread. Những thread này sẽ chịu trách nhiệm nhận truy vấn từ Server Reactor, lập lịch, xử lý và trả dữ liệu về cho Server Reactor nếu cần IO với Client.



Hình 16: Cấu trúc máy chủ cơ bản sử dụng Non-blocking socket

Chúng em sẽ chỉ ra những điểm mạnh của cấu trúc này thông qua quá trình hoạt động:

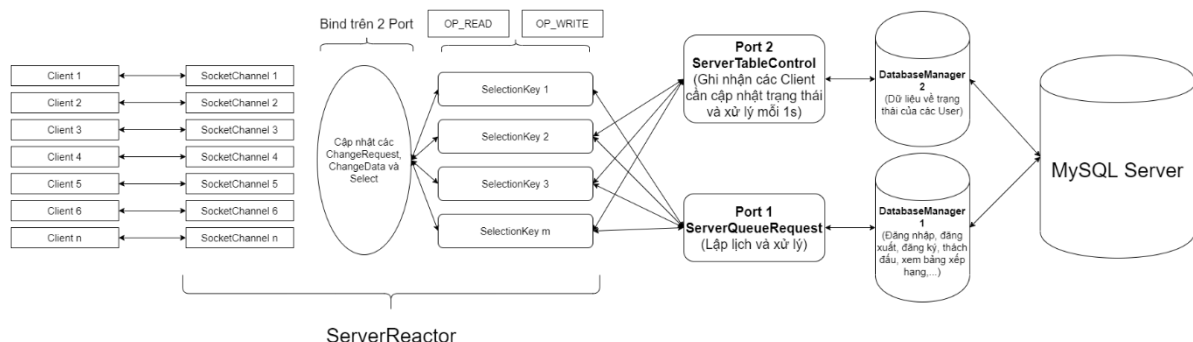
- ServerReactor khi bắt đầu hoạt động sẽ đăng kí các ServerSocketChannel (có tác dụng bind và listen trên các interface) hoặc SocketChannel (Kết nối tới Server khác) với một Selector. Điều này giúp người sử dụng chỉ cần với 1 thread duy nhất vừa có thể đóng vai trò một hay nhiều Server và cùng lúc đó kết nối tới một hoặc nhiều Server khác.
- Selector này có nhiệm vụ chọn ra những ServerSocketChannel hoặc SocketChannel có thể tương tác (ví dụ như read, write, accept, connect) và trả về một List các SelectionKey. Các hoạt động (Operation) tương tác này có thể được dự đoán trước thông qua việc sử dụng:

`SelectionKey.interestOps([OP_READ hoặc OP_WRITE])`

Thông thường trong quá trình hoạt động, một SelectionKey luôn được đặt trong chế độ OP_READ để lắng nghe các truy vấn từ Server/Client. Ngược lại OP_WRITE chỉ được đặt khi Client/Server hiện tại muốn gửi đi một truy vấn tới bất kì một Server/Client nào khác. Điều này giúp cho người sử dụng dễ dàng tùy biến được dữ liệu đọc, ghi tới bất kì một trong những kết nối nào đang được mở. Ngoài ra, việc các truy vấn được xử lý theo một List các SelectionKey sẽ giúp các truy vấn được thực hiện theo một thứ tự nhất định thay vì đồng loạt xử lý.

- Dữ liệu sau khi được tiếp nhận bởi ServerReactor sẽ được lập lịch xử lý và chuyển tới các WorkerThread giúp tránh Race condition trong quá trình vận hành. Trong trường hợp cần trả về dữ liệu ngược lại cho Client/Server, các WorkerThread này sẽ trả về các ChangeRequest và ChangeData về cho ServerReactor. Các request này sẽ được thực thi trước mỗi lần Selector thực hiện select.

3.1.2. Kiến trúc Server



Hình 17: Kiến trúc Server tổng quan

Trương tự kiến trúc bên trên đã trình bày, Server được dựng lên theo kiến trúc Non-blocking IO, trong đó chỉ sử dụng một thread duy nhất cho việc lập lịch và xử lý là ServerQueueRequest và một thread giúp trả lời các yêu cầu về bảng trạng thái mang tên ServerTableControl. Hai Port tương ứng cũng được bind cho mỗi chức năng. Mỗi thread sẽ tạo một handle riêng biệt tới MySQL Server.

3.1.2.1. Database

```
mysql> select * from user_account;
```

AccountID	username	password	isOnline	point	isPlaying	isAdmin
1	hoang	admin	0	1254.5	0	0
2	nhat	admin	0	5449.5	0	0
3	vu	admin	0	3	0	0
4	Thanh	admin	0	0.5	0	0
5	dbgr	admin	0	0	0	0

5 rows in set (0.00 sec)

Hình 18: Cấu trúc cơ sở dữ liệu

- Các cột trong Database bao gồm:

```
CREATE TABLE `user_account` (
  `AccountID` int NOT NULL AUTO_INCREMENT,
  `username` varchar(255) NOT NULL,
  `password` varchar(255) NOT NULL,
  `isOnline` tinyint(1) DEFAULT '0',
  `point` float DEFAULT '0',
  `isPlaying` tinyint(1) DEFAULT '0',
  `isAdmin` tinyint(1) DEFAULT '0',
  PRIMARY KEY (`AccountID`)
)
```

- Trong đó:

- username: Tên tài khoản của người dùng.
- password: Mật khẩu của người dùng.
- isOnline: Được set giá trị True khi User online, ngược lại nhận giá trị False.
- point: Số điểm hiện tại.
- isPlaying: Trạng thái của người chơi khi đã online (Trong trận đấu/Rảnh rỗi).
- isAdmin: Người chơi có phải Admin (dùng cho quản trị trên nền tảng web).

3.1.2.2. ServerQueueRequest

3.1.2.2.1. Lập lịch

Những request tương ứng với Port 1 sẽ được Thread này lập lịch và xử lý và trả về cho ServerReactor.

```
public class EchoWorker implements Runnable {
    private List queue = new LinkedList();

    public void processData(NioServer server, SocketChannel socket, byte[] data, int count) {
        byte[] dataCopy = new byte[count];
        System.arraycopy(data, 0, dataCopy, 0, count);
        synchronized(queue) {
            queue.add(new ServerDataEvent(server, socket, dataCopy));
            queue.notify();
        }
    }

    public void run() {
        ServerDataEvent dataEvent;

        while(true) {
            // Wait for data to become available
            synchronized(queue) {
                while(queue.isEmpty()) {
                    try {
                        queue.wait();
                    } catch (InterruptedException e) {
                    }
                }
            }
            dataEvent = (ServerDataEvent) queue.remove(0);

            // Return to sender
            dataEvent.server.send(dataEvent.socket, dataEvent.data);
        }
    }
}
```

Hình 19: Quá trình lập lịch

Trong hình 6 bên trên, khi ServerReactor nhận được dữ liệu từ phía Client sẽ gọi tới hàm processData(), hàm này sẽ có nhiệm vụ thêm mới request vào queue để lập lịch xử lý. Sau khi thêm mới xong, hàm sẽ notify phần ThreadMain đang chờ đợi dữ liệu xử lý.

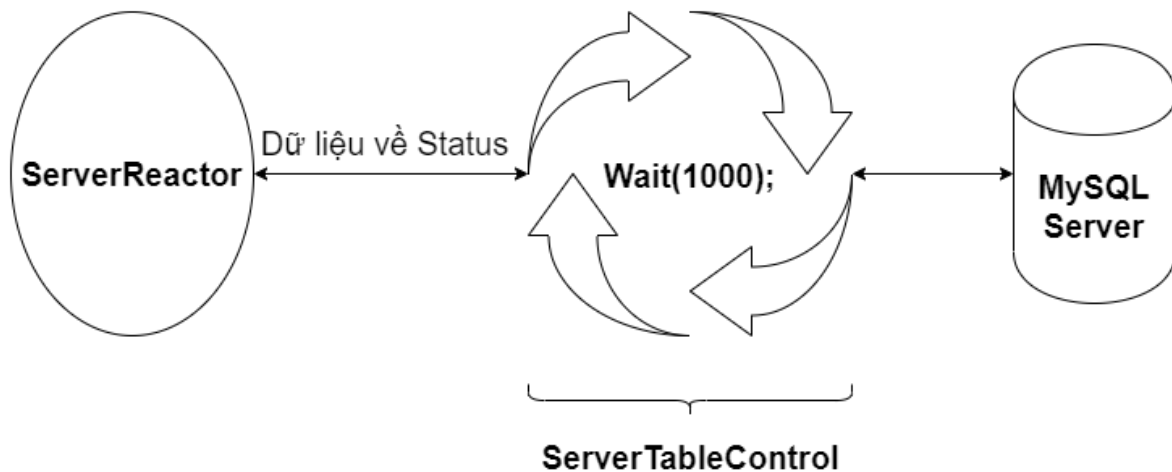
3.1.2.2.2. Xử lý

ServerQueueRequest sẽ thực hiện xử lý các truy vấn sau:

- **Đăng nhập:** ClientMessage.REQUEST.LOGIN
- **Tạo tài khoản:** ClientMessage.REQUEST.REGISTER
- **Đăng xuất:** Được xử lý tương tự như khi Client ngắt kết nối đột ngột.
- **Thách đấu:** ClientMessage.REQUEST.CHALLENGE
- **Chiến thắng:** ClientMessage.REQUEST.WIN (nhận được khi Client hoàn thành bàn chơi trước đối thủ).

- **Thoát bàn chơi:** ClientMessage.REQUEST.QUIT (nhận được khi Client thoát khi đang trong bàn chơi với đối thủ khác).
- **Hết thời gian:** ClientMessage.REQUEST.MATCH (nhận được khi Client và đối thủ cùng không hoàn thành bàn chơi sau 2 phút, cả 2 sẽ phải gửi thông tin điểm số về Server)
- **Khi Client disconnect:** Server sẽ thực thi:
 - o Đặt giá trị isOnline = False và isPlaying = False;
 - o Xóa các lời mời thách đấu (nếu có)
 - o Nếu người chơi đang trong một trận đấu, điểm số sẽ được cộng cho đối thủ cùng với thông báo về việc thoát của người chơi. Đồng thời lưu lại dữ liệu về trận đấu.

3.1.2.3. ServerTableControl

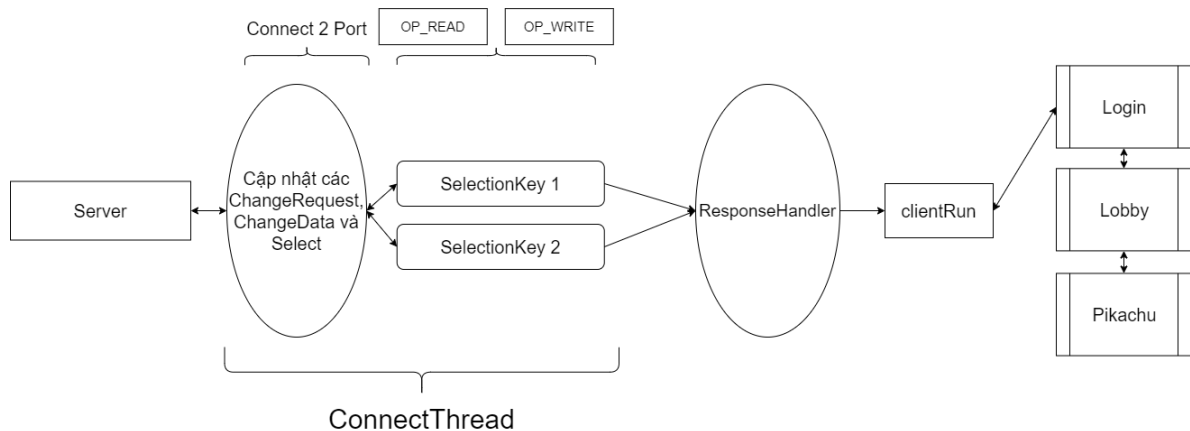


Hình 20: Cấu trúc ServerTableControl

ServerTableControl sẽ có những nhiệm vụ tương ứng với Port 2 như sau:

- Ghi nhận những Client kết nối đến Port 2 vào một List.
- Mỗi 1 giây thực hiện những công việc sau:
 - o Đọc dữ liệu về trạng thái của các User từ MySQL Server.
 - o Tạo các ChangeRequest và ChangeData thông qua ServerReactor gửi tới Client.

3.1.3. Kiến trúc Client



Hình 21: Kiến trúc Client tổng quan

Tương tự kiến trúc của Server, Client được dựng lên theo kiến trúc Non-blocking IO, trong đó chỉ sử dụng một thread duy nhất cho việc lập lịch và xử lý là ResponseHandler, một thread mang tên ConnectThread chịu trách nhiệm nhận và gửi các bản tin tới Server.

3.1.3.1. ResponseHandler

3.1.3.1.1. Lập lịch

- Việc lập lịch của Client hoàn toàn giống Server

3.1.3.1.2. Xử lý

ResponseHandler sẽ xử lý những thông điệp từ Server sau:

- **Kết quả đăng nhập:** ServerMessage.REQUEST.LOGIN
- **Kết quả đăng ký:** ServerMessage.REQUEST.REGISTER
- **Dữ liệu về trạng thái của các User:** ServerMessage.REQUEST.TABLE
- **Lời mời thách đấu:** ServerMessage.REQUEST.CHALLENGE
- **Kết quả lời mời thách đấu:** ServerMessage.REQUEST.CHALLENGE

3.1.3.2. LoginControl

LoginControl thực hiện những công việc sau:

- Khởi tạo LoginForm.
- Khởi tạo kết nối đến Port 1 trên Server.
- Listen các Action liên quan đến đăng nhập, đăng ký từ LoginForm, tạo các ClientMessage tương ứng gửi lên Server và thông báo kết quả.

3.1.3.3. LobbyControl

LoginControl thực hiện những công việc sau:

- Khởi tạo LobbyForm.
- Khởi tạo kết nối tới Port 2 trên Server.

- Cập nhật dữ liệu về trạng thái người dùng, điểm số lên LobbyForm.
- Listen các Action liên quan đến thách đấu, đăng xuất từ LobbyForm, tạo các ClientMessage tương ứng gửi lên Server và thông báo kết quả.

3.1.3.4. PikachuController (Phạm Đức Hiển, Hoàng Anh Tuấn, Nguyễn Văn Hân, 2017)

PikachuController thực hiện những công việc sau:

- Khởi tạo PlayGameView.
- Tính toán thời gian, progress bar cho PlayGameView.
- Listen các Action liên quan đến click nhân vật, thoát trò chơi từ PlayGameView, trò chơi kết thúc, ... và tạo các ClientMessage tương ứng gửi lên Server sau đó thông báo kết quả.

3.2. Cài đặt và triển khai ứng dụng

3.2.1. Cấu trúc thư mục

- Cấu trúc thư mục Client:
 - Controller:
 - clientRun.java
 - ConnectThread.java
 - ResponseHandler.java
 - LoginControl.java
 - LobbyControl.java
 - PikachuController.java
 - Model:
 - ClientDataEvent.java
 - Pikachu.java
 - View:
 - JpanelBackground.java
 - LobbyForm.java
 - LoginForm.java
 - PlayGameView.java
 - RequestForm.java
- Resource (Chứa các tập tin hình ảnh nhân vật AmongUs, icon,...)
- Cấu trúc thư mục Server:
 - Controller:
 - DatabaseManager.java
 - ServerQueueRequest.java
 - ServerReactor.java
 - serverRun.java
 - ServerTableControl.java
 - Model:

- Message:
 - ClientMessage.java
 - ServerMessage.java
 - ChangeRequest.java
 - Match.java
 - Matrix.java
 - ServerDataEvent.java
 - UserAccount.java
 - UserTable.java
 - UserTableData.java
- Utils:
- Utils.java (Chứa thông tin cài đặt về Port 1, Port 2, Hostname,...)

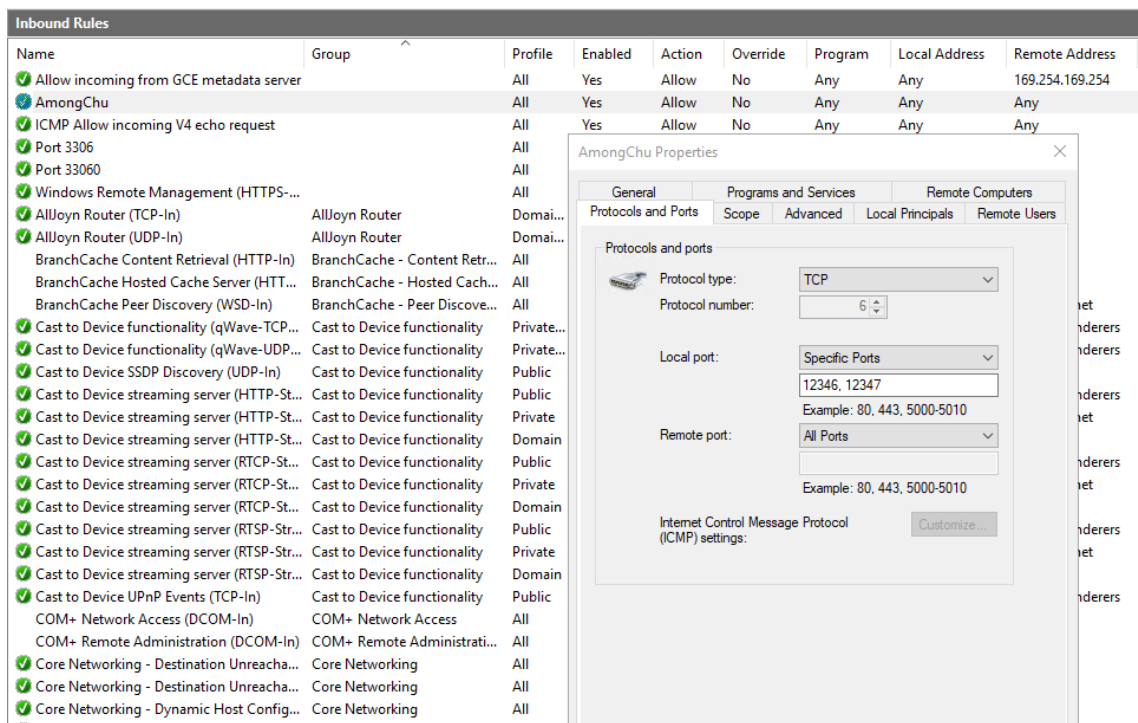
3.2.2. Triển khai

3.2.2.1. Game Server

- Game hiện tại đã được triển khai tại host: 34.92.139.146 với 2 port 12346 và 12347.

Name	Type	Targets	Filters	Protocols / ports	Action	Priority	Network	Logs	Hit count	Last hit
bt1	Egress	Apply to all	IP ranges: 0.0.0.0/0	tcp:12346,12347	Allow	1000	default	On	24	2020-11-28 (10:14:00)
lrm-out	Egress	Apply to all	IP ranges: 0.0.0.0/0	tcp:2375,2376	Allow	1000	default	On	0	No hits
bt1	Ingress	lrm	IP ranges: 0.0.0.0/0	tcp:12346,12347	Allow	1000	default	On	0	No hits
default-allow-http	Ingress	http-server	IP ranges: 0.0.0.0/0	tcp:80	Allow	1000	default	Off	-	-
default-allow-https	Ingress	https-server	IP ranges: 0.0.0.0/0	tcp:443	Allow	1000	default	Off	-	-
lrm	Ingress	Apply to all	IP ranges: 0.0.0.0/0	tcp:12346,12347	Allow	1000	default	On	212	2020-12-02 (07:43:00)
lrm-in	Ingress	Apply to all	IP ranges: 0.0.0.0/0	tcp:2375,2376	Allow	1000	default	On	238	2020-12-02 (08:22:00)

Hình 22: Mở cổng trên Google Cloud cho trò chơi

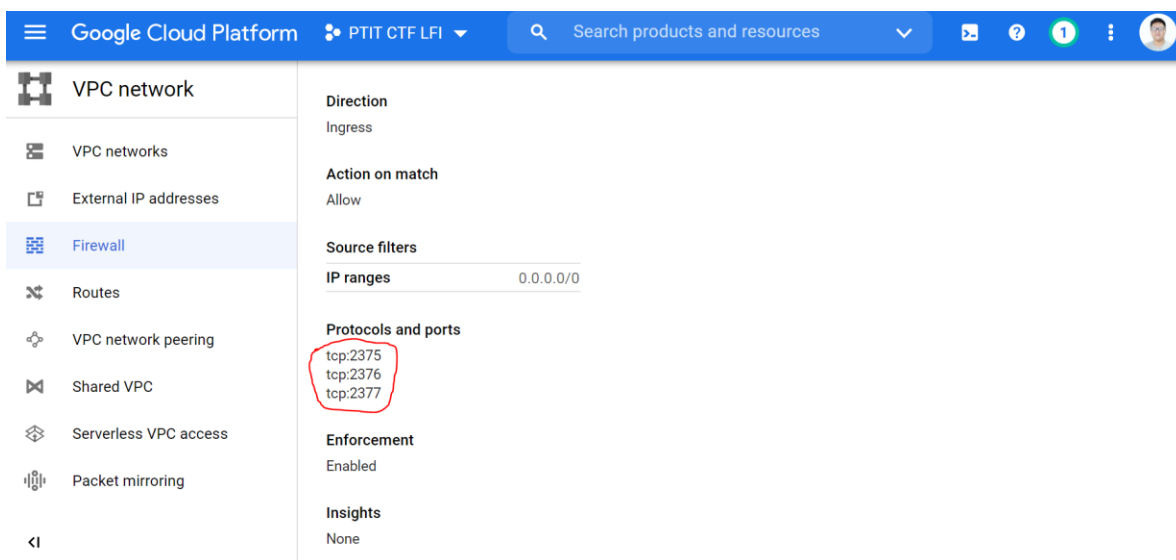


Hình 23: Mở cổng trong Windows Server cho trò chơi

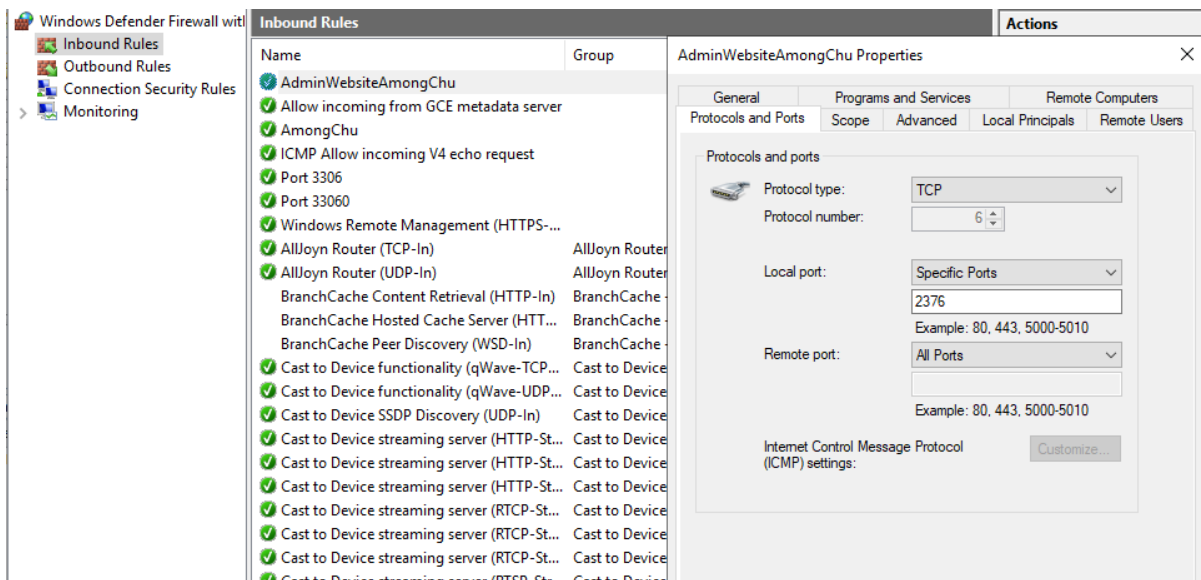
- Toàn bộ source code được public tại: <https://github.com/tconqueror/AmongChu>

3.2.2.2. Website quản trị

- Source code website quản trị được public tại: https://github.com/d3lt4-024/LTM_Web
- URL: <http://amongchu.infosecptit.club:2376/>



Hình 24: Mở cổng trên Google Cloud cho website quản trị

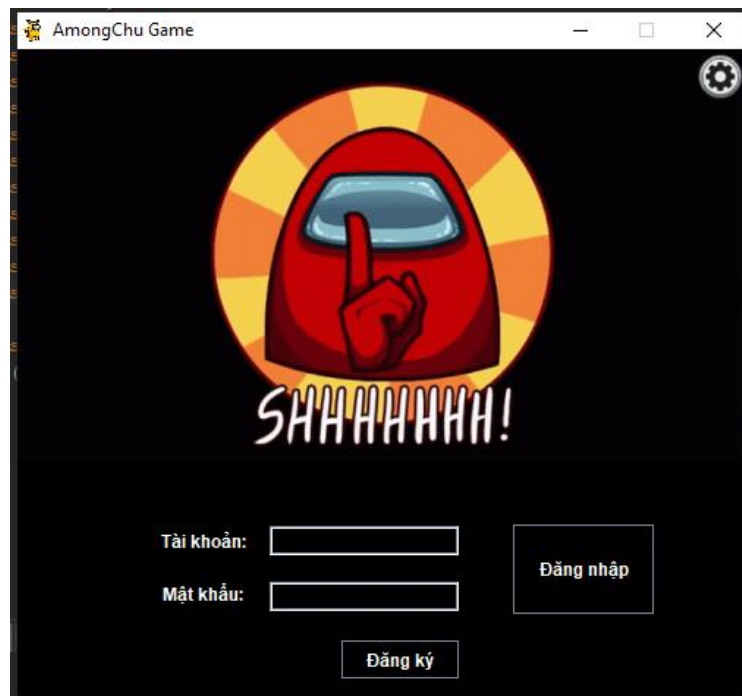


Hình 25: Mở cổng website quản trị trên Windows Server

3.3. Kết quả thực hiện

Danh sách kết quả thực hiện thành công

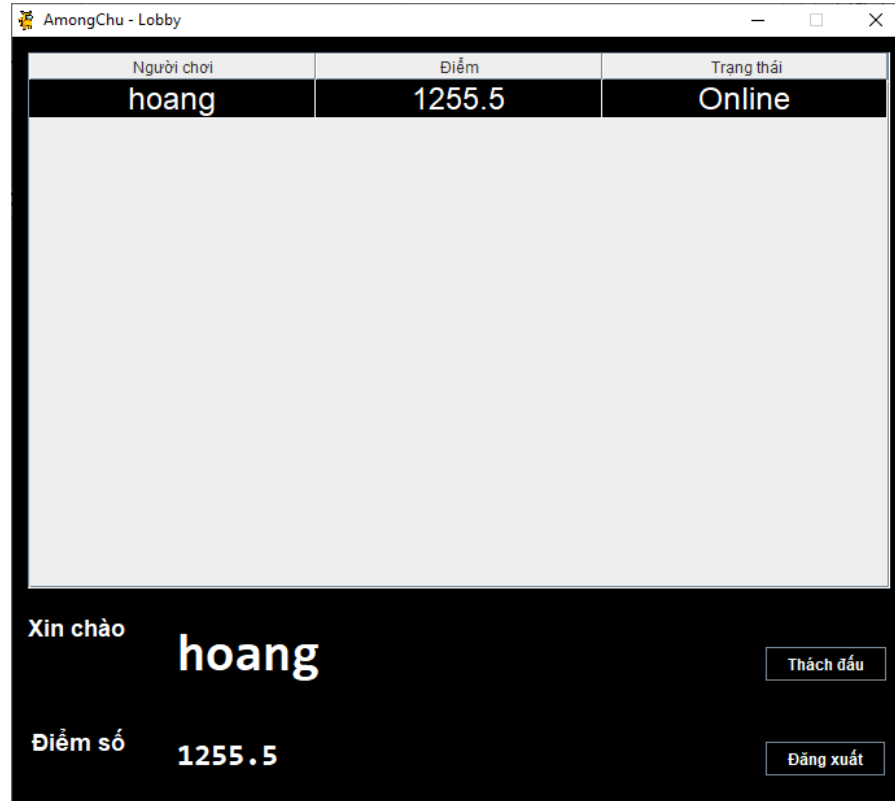
- Đăng nhập:



Hình 26: Giao diện đăng nhập

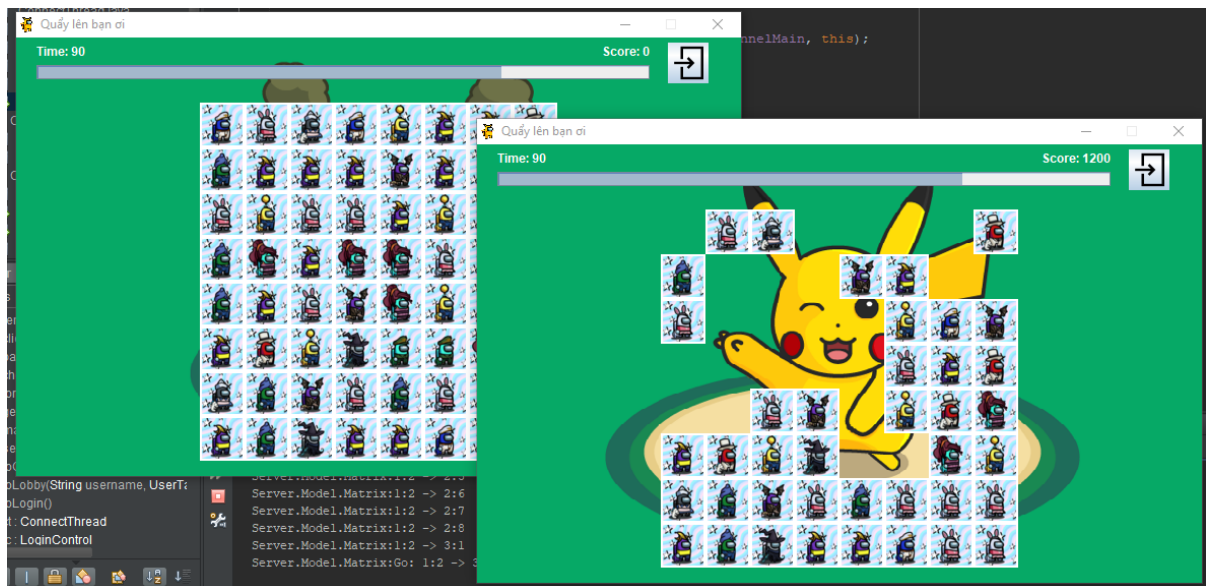
- Đăng nhập thành công.
- Đăng nhập thất bại.
- Đã đăng nhập nơi khác.

- Đăng ký:
 - o Đăng ký thành công
 - o Tài khoản đã tồn tại
- Sảnh chờ



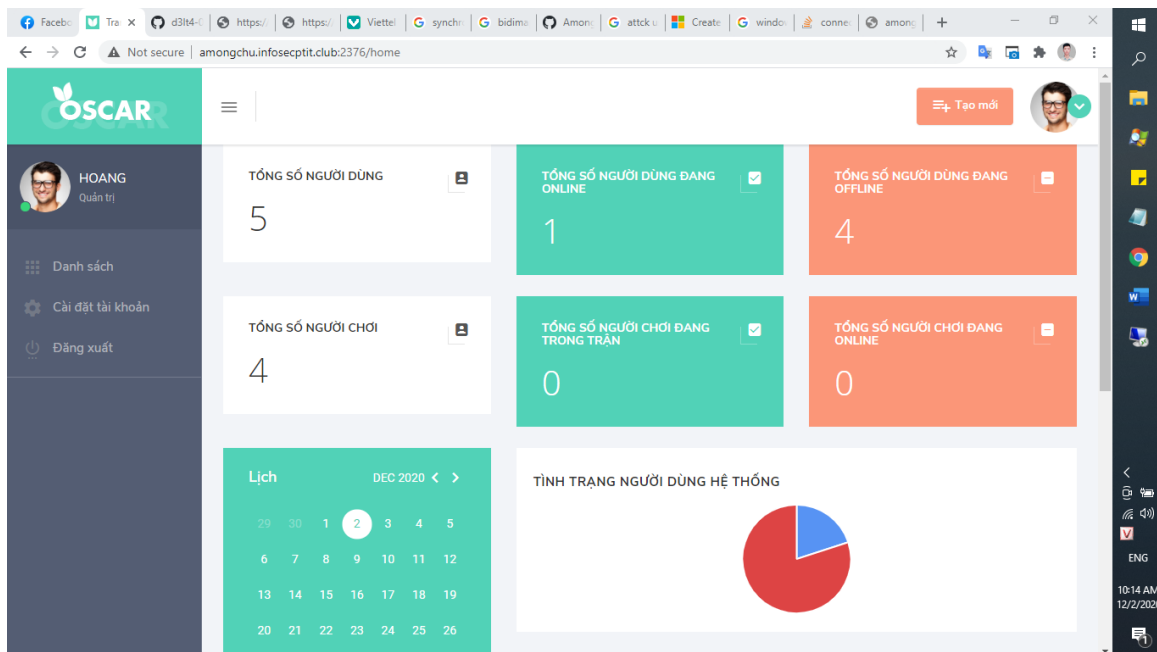
Hình 27: Giao diện sảnh chờ

- Đăng xuất
- Thách đấu: Một người chơi hoàn toàn có thể thách đấu nhiều người một lúc
 - o Người chơi hiện không Online
 - o Người chơi đã trong trận đấu
- Chấp nhận thách đấu: Một người chơi có thể nhận được nhiều lời mời thách đấu trong cùng một thời điểm. Khi người chơi click chấp thuận, các lời mời còn lại sẽ bị xóa khỏi giao diện và người chơi được đưa vào giao diện thi đấu.
- Chơi trò chơi:



Hình 28: Giao diện trò chơi

- Thời gian đếm ngược
- Thoát trò chơi
- Gửi thông báo kết quả trận đấu
- Toàn bộ trường hợp trong trò chơi có thể xảy ra.
- Dấu lại
- Xử lý toàn bộ các Exception xảy ra khi Client mất kết nối.
- Website quản trị:



Hình 29: Giao diện website quản trị

3.4. Kết luận

3.4.1. Tổng kết

Pikachu là một tựa trò chơi rất nổi tiếng và đã được nhóm chúng em làm mới lại theo thể thức thi đấu đối kháng kết hợp với những hình ảnh sinh động từ tựa trò chơi AmongUs với cái tên “AmongChu”

3.4.2. Hạn chế cần khắc phục

- Lỗi SQL Injection.
- Cần kiểm tra dữ liệu Table tại Server thay vì Client.
- Giao diện chưa được đẹp mắt.
- Bảng xếp hạng chưa hoàn thiện.
- Lưu trữ password dạng mã hóa.

Danh mục tài liệu tham khảo

Greenfield, J. (n.d.). Retrieved from The Rox Java NIO Tutorial: <http://rox-xmlrpc.sourceforge.net/niotut/>

luminousmen. (2020, April 26). *Asynchronous programming. Blocking I/O and non-blocking I/O*. Retrieved from <https://luminousmen.com/about/>:
<https://luminousmen.com/post/asynchronous-programming-blocking-and-non-blocking>

Phạm Đức Hiền, Hoàng Anh Tuấn, Nguyễn Văn Hãnh. (2017, Jun 7). *Pika*. Retrieved from Github: <https://github.com/hoangtuanhedspi/Pika>