

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN



**PBL4: DỰ ÁN HỆ ĐIỀU HÀNH &
MẠNG MÁY TÍNH**

Đề tài: Website Cờ Vua

SINH VIÊN THỰC HIỆN:

Lê Phước Duy

LỚP: 20TCLC-DT3 NHÓM: 21.10C

Huỳnh Văn Lộc

LỚP: 20TCLC-DT3 NHÓM: 21.10C

GIẢNG VIÊN HƯỚNG DẪN: ThS. Nguyễn Công Danh

Đà Nẵng, 12/2023

MỤC LỤC

DANH MỤC HÌNH VẼ.....	4
DANH MỤC BẢNG.....	4
CÁC THUẬT NGỮ SỬ DỤNG	7
LỜI NÓI ĐẦU	8
QUY TRÌNH TRIỂN KHAI.....	9
1. KẾ HOẠCH THỰC HIỆN	9
2. BẢNG PHÂN CÔNG NHIỆM VỤ.....	9
3. QUẢN LÝ DỰ ÁN.....	10
3.1. QUÁ TRÌNH QUẢN LÝ DỰ ÁN.....	10
3.2. QUY TRÌNH CODE	10
3.3. CÔNG CỤ VÀ MÔI TRƯỜNG PHÁT TRIỂN DỰ ÁN.....	11
GIỚI THIỆU ĐỀ TÀI	11
1. MỤC ĐÍCH ĐỀ TÀI	11
2. PHẠM VI ĐỀ TÀI	12
CHƯƠNG 1: CƠ SỞ LÝ THUYẾT	12
2.1. MÔ HÌNH CLIENT - SERVER.....	12
2.2. GIAO THỨC TCP (TRANSMISSION CONTROL PROTOCOL)	13
1.2.1. KHÁI NIỆM CƠ BẢN VỀ TCP	13
1.2.2. TRUYỀN DỮ LIỆU.....	13
1.2.3. ĐẶC ĐIỂM DỮ LIỆU	13
1.2.4. ỨNG DỤNG VÀ ƯU ĐIỂM.....	14
2.3. MÔ HÌNH WEBSOCKET	14
2.3.1. KHÁI NIỆM CƠ BẢN VỀ WEBSOCKET	14
2.3.2. ƯU ĐIỂM VÀ ỨNG DỤNG.....	14
2.4. KIẾN TRÚC MVC TRONG SPRING BOOT	15
CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG	16
2.1. USECASE TỔNG QUAN.....	16
2.2. CƠ SỞ DỮ LIỆU.....	17
2.3. BIỂU DỒ LỚP.....	18
2.4.1. PHÍA CLIENT.....	18
2.4.2. PHÍA SERVER.....	18
2.4. BIỂU ĐỒ TUẦN TỰ (SEQUENCE DIAGRAM).....	19
2.5.1. CHỨC NĂNG ĐĂNG KÝ	19
2.5.2. CHỨC NĂNG ĐĂNG NHẬP	19
2.5.3. CHỨC NĂNG CHƠI CỜ	20

2.5. YÊU CẦU PHI CHỨC NĂNG	20
2.5.1. YÊU CẦU VỀ TỔ CHỨC, TIẾN TRÌNH PHÁT TRIỂN	20
2.5.2. YÊU CẦU VỀ ĐỘ TIN CẬY	20
2.5.3. YÊU CẦU VỀ SẢN PHẨM.....	21
2.5.4. KHẢ NĂNG TÁI SỬ DỤNG	21
CHƯƠNG 3: TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ	21
3.1. TRIỂN KHAI MÔ HÌNH	21
3.2.1. CÁCH THỨC HOẠT ĐỘNG	21
3.2.2. ĐỊNH DANH CLIENT KHI KẾT NỐI ĐẾN SERVER.....	23
3.2.3. XỬ LÝ THỜI GIAN	24
3.2.4. XỬ LÝ PHÂN TRANG	25
3.2.5. XỬ LÝ LOGIC CỜ VUA	25
3.2. KẾT QUẢ.....	27
3.2.1. CHỨC NĂNG ĐĂNG NHẬP, ĐĂNG KÍ.....	27
3.2.2. CHỨC NĂNG XEM THÔNG TIN CÁ NHÂN.....	29
3.2.3. CHỨC NĂNG NHẮN TIN TỔNG, NHẮN TIN RIÊNG	31
3.2.4. CHỨC NĂNG CHƠI CỜ.....	32
3.2.5. CHỨC NĂNG PHÍA SERVER.....	35
KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	36
1. KẾT LUẬN	36
2. HƯỚNG PHÁT TRIỂN	36
TÀI LIỆU THAM KHẢO.....	37
PHỤ LỤC A.....	38
PHỤ LỤC B.....	39

DANH MỤC HÌNH VẼ

Hình 1: Mô hình Client – Server.....	12
Hình 2: Mô hình xử lý phân tán.	13
Hình 3: Minh họa giao thức TCP.....	13
Hình 4: Minh họa giao thức WebSocket.....	14
Hình 5: Kiến trúc Spring Boot.	15
Hình 6: Sơ đồ Usecase tổng quan.	16
Hình 7: Cơ sở dữ liệu.	17
Hình 8: Biểu đồ lớp phía Client.	18
Hình 9: Biểu đồ lớp phía Server.....	18
Hình 10: Biểu đồ tuần tự chức năng đăng ký.....	19
Hình 11: Biểu đồ tuần tự chức năng đăng nhập.....	19
Hình 12: Biểu đồ tuần tự chức năng chơi cờ.	20
Hình 13: Mô hình hoạt động dự án.	21
Hình 14: Thêm phụ thuộc websocket.....	22
Hình 15: Cấu hình Endpoint và STOMP phía server.....	22
Hình 16: Nơi xử lý nhận và phát tin nhắn phía server	22
Hình 17: Import thư viện và cấu hình các giá trị STOMP phía client	22
Hình 18: Đăng ký và lắng nghe phía client.....	23
Hình 19: Định danh client trong WebSocket.	23
Hình 20: Cấu hình các thông số định danh.	24
Hình 21: Thực hiện quản lý thời gian phía server.....	24
Hình 22: Hàm sắp xếp user theo elo.	25
Hình 23: Hàm trả về danh sách các user.	25
Hình 24: Ví dụ chức năng của hàm isValidMove().	26
Hình 25: Ví dụ trường hợp hòa cờ.	26
Hình 26: Giao diện chức năng đăng ký.....	27
Hình 27: Giao diện chức năng đăng nhập.....	28
Hình 28: Giao diện xem thông tin cá nhân và bảng xếp hạng.	29
Hình 29: Giao diện xem thông tin bạn bè.	30

Hình 30: Giao diện chức năng nhắn tin tổng.	31
Hình 31: Giao diện chức năng nhắn tin riêng.	31
Hình 32: Giao diện chức năng chơi với bạn.	32
Hình 33: Giao diện chức năng chơi thể giới.	33
Hình 34: Giao diện khi trong trận.	33
Hình 35: Giao diện server.	35
Hình 36: Quản lí mã nguồn	38

DANH MỤC BẢNG

Bảng 1: Quá trình triển khai.....	9
Bảng 2: Quá trình phân công nhiệm vụ.	10
Bảng 3: Công cụ và môi trường phát triển.....	11
Bảng 4: Mô tả các bảng trong cơ sở dữ liệu.	17
Bảng 5: Mô tả giao diện đăng ký.	27
Bảng 6: Mô tả giao diện đăng nhập.	28
Bảng 7: Mô tả giao diện xem thông tin cá nhân.	29
Bảng 8: Mô tả giao diện xem thông tin bạn bè.	30
Bảng 9: Mô tả giao diện nhắn tin tổng.....	31
Bảng 5.2.6: Mô tả giao diện nhắn tin riêng.....	32
Bảng 10: Mô tả chức năng chơi với bạn.	32
Bảng 11: Mô tả chức năng chơi thể giới.	33
Bảng 12: Mô tả giao diện trong trận đấu.	34
Bảng 13: Mô tả giao diện server.	35

CÁC THUẬT NGỮ SỬ DỤNG

Thuật ngữ	Diễn giải
Piece	Quân cờ
Rook	Quân xe
Knight	Quân mã (quân ngựa)
Bishop	Quân tượng
Queen	Quân hậu
King	Quân vua
Pawn	Quân tốt (quân chuột)
Chiếu vua	Trong một ván cờ vua, chiếu vua xảy ra khi quân vua của một bên đang bị đe dọa bắt mất trong nước kế tiếp.
Interceptor	Được sử dụng để kiểm soát truy cập vào các tài nguyên, quản lý phiên và ghi nhật ký.

LỜI NÓI ĐẦU

Trong thời đại 4.0, ngành Công Nghệ Thông Tin (CNTT) ngày càng chiếm vị thế quan trọng đối với sự phát triển đất nước nói riêng và của cả nhân loại nói chung. Ngày nay, những thành tựu của ngành CNTT được ứng dụng rộng rãi trên nhiều lĩnh vực (từ giải trí, y tế, giáo dục, nghệ thuật, sản xuất, ...). Điều đó đòi hỏi các lập trình viên phải không ngừng cải tiến công nghệ, xây dựng và tối ưu hóa thuật toán nhằm thiết kế thêm nhiều tính năng có thể ứng dụng vào đời sống.

Từ đó, nhóm em đã quyết định chọn đề tài “**Website Cờ vua**” làm đề tài cho **PBL 4**. Đề tài này sẽ áp dụng các kiến thức đã được học trong các môn học liên quan đến phát triển phần mềm, như Công nghệ phần mềm, Phân tích và thiết kế hướng đối tượng, Lập trình Java, Lập trình mạng. Qua đó mang lại website chơi game công nghệ, hiện đại hơn tối ưu hóa hơn so với mô hình chơi cờ vua truyền thống.

Trong quá trình thực hiện, dù có cố gắng nhưng chúng em không tránh khỏi những sai sót. Chúng em kính mong quý thầy cô sẽ thông cảm và tận tình chỉ dẫn, góp ý để đề tài của chúng em ngày càng hoàn thiện hơn.

Chúng em cũng xin gửi lời cảm ơn đến quý thầy cô đã và đang giảng dạy vì đã giúp chúng em có các kiến thức nền tảng trong ngành học.

Đặc biệt, chúng em muốn gửi lời cảm ơn chân thành đến ThS. Nguyễn Công Danh (Giảng viên trường ĐHBK Đà Nẵng) đã tận tình hướng dẫn đồ án môn học này cho chúng em.

Nhóm sinh viên thực hiện đề tài
1. Lê Phước Duy
2. Huỳnh Văn Lộc

QUY TRÌNH TRIỂN KHAI

1. Kế hoạch thực hiện

Thời gian	Nội dung công việc	Sản phẩm
08/09/2023 – 22/09/2023	Phân tích, chọn đề tài. Xác định các ngôn ngữ, mô hình sử dụng.	
23/09/2023 – 14/10/2023	Học, làm chủ các ngôn ngữ sử dụng.	
16/10/2023 – 20/10/2023	Thiết kế Database.	db_pbl4.sql
21/10/2023 – 27/10/2023	Thiết kế giao diện phía client.	Client UI
28/10/2023 – 31/10/2023	Tổng kết lần 1, giải quyết các vấn đề phát sinh. Thông nhất quy tắc code, các công cụ sử dụng.	
01/11/2023 – 08/11/2023	Thiết kế phía server	Các endpoint server
09/11/2023 – 16/11/2023	Xử lý gửi nhận trao đổi thông tin 2 phía	Tương tác gửi nhận thông được cả 2 phía
17/11/2023 - 14/12/2023	Các chức năng còn lại.	Chế độ chơi với bạn, chế độ chơi thể giới, chat tổng, chat riêng
15/12/2023 – 24/12/2023	Kiểm thử và viết báo cáo.	BaoCao.docx

Bảng 1: Quá trình triển khai.

2. Bảng phân công nhiệm vụ

Họ tên	Nhiệm vụ
1. Lê Phước Duy	<ul style="list-style-type: none">- Thiết kế cơ sở dữ liệu.- Thiết kế UI/UX.- Nghiên cứu, tìm hiểu tài liệu về TypeScript.

	<ul style="list-style-type: none">- Thiết kế phía Client- Hỗ trợ phía Server.- Test hệ thống, chỉnh sửa sai sót.- Viết báo cáo.
2. Huỳnh Văn Lộc	<ul style="list-style-type: none">- Thiết kế cơ sở dữ liệu.- Nghiên cứu, tìm hiểu tài liệu về WebSocket.- Test hệ thống, chỉnh sửa sai sót.- Thiết kế phía Server.- Hỗ trợ phía Client.- Viết báo cáo.

Bảng 2: Quá trình phân công nhiệm vụ.

3. Quản lý dự án

3.1. Quá trình quản lý dự án

- Các thành viên thực công việc đã giao trên, trao đổi trực tiếp qua hoặc thông qua Google Meet, MS Team.

- Sau khi hoàn thành từng module, chức năng, các thành viên sẽ họp để xem một cách kĩ càng, đưa ra các giải pháp cải tiến, nếu khả thi và các thành viên trong nhóm đồng ý thì giải pháp sẽ được thực hiện.

3.2. Quy trình code

3.2.1. Đối với class, function

- Khai báo tên phải có mục đích rõ ràng, mô tả được công việc của nó thực hiện. Tên function không được bắt đầu bằng số, bắt buộc bằng chữ cái theo cú pháp camelCase.

Ví dụ: createRoom(), login(), register(),...

- Tên các class được đặt theo cú pháp PascalCase.

Ví dụ: LoginController(), RoomService(),...

3.2.2. Đối với biến

- Tên biến phải mô tả rõ ràng nội dung nó sẽ đảm nhận.

- Tên biến được đặt theo cú pháp camelCase.

Ví dụ: timeStart, timeEnd, mode,...

3.2.3. Một số quy tắc khác

- Nếu một hàm có nhiều cấp lồng nhau, mỗi cấp nên xuống dòng.
- Các đoạn code bằng cấp nên ở cùng một cột với nhau, dòng xuống hàng nên bắt đầu cùng cấp với dòng phía trên.
- Hạn chế dùng comment để giải thích code hoặc chú thích những sự thật hiển nhiên.
- Comment cảnh báo hậu quả, làm rõ ý nghĩa của code.
- Sau khi code xong một module, chức năng, các thành viên trong nhóm sẽ push source code lên Github, phải ghi rõ nội dung hoàn thành khi push.

3.3. Công cụ và môi trường phát triển dự án

Hoạt động	Công cụ	Phiên bản
Server (Java)	IntelliJ IDEA Ultimate	2023.2.1
Client (TypeScript, CSS, HTML)	Visual Studio Code	Visual Studio Code 1.78.1
Database	MySQL Workbench	
Môi trường test	Microsoft Edge	Microsoft Edge 112.0
Công cụ thiết kế các sơ đồ	Draw.io	
Quản lý source code	Github	

Bảng 3: Công cụ và môi trường phát triển.

GIỚI THIỆU ĐỀ TÀI

1. Mục đích đề tài

Trong thời đại phát triển ngày càng nhanh chóng, với sự gia tăng đáng kể của nhu cầu giải trí, trò chơi trí tuệ ngày càng trở thành một phần quan trọng của cuộc sống. Cờ vua, với tính chất chiến thuật và trí tuệ, không chỉ là một trò chơi truyền thống mà còn là một ứng cử viên xuất sắc trong thế giới giải trí hiện đại.

Đề tài của chúng em hướng đến việc tạo ra một nền tảng chơi cờ vua đơn giản, linh hoạt và đáp ứng đầy đủ nhu cầu của cộng đồng người chơi cờ vua. Chúng em tin rằng sản phẩm của mình sẽ mang lại nhiều lợi ích và tạo ra một môi trường chơi cờ vua thú vị và chuyên nghiệp.

2. Phạm vi đề tài

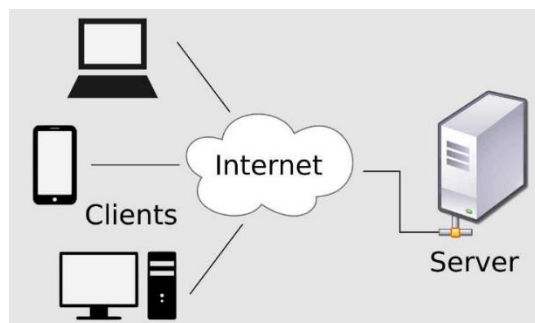
Đề tài sau khi hoàn thiện phải đáp ứng các yêu cầu sau:

- Có giao diện thân thiện người dùng.
- Triển khai cơ sở dữ liệu trên cùng máy chủ.
- Người dùng có thể truy cập được được trò chơi khi chung mạng LAN với máy chủ.
- Dành cho người chơi:
 - Đăng nhập, đăng ký.
 - Xem thông tin cá nhân, bạn bè, thêm mới bạn.
 - Chơi cờ với bạn, chơi thể giới.
 - Trò chuyện trong trận, trò chuyện tổng.
- Dành cho quản lý:
 - Xem số người chơi đang online.
 - Xem thông tin phòng đang diễn ra.

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

1.1. Mô hình Client - Server

Mô hình Client - Server là mô hình giúp các máy tính giao tiếp truyền tải dữ liệu cho nhau qua mạng. Trong mô hình này, Client gửi yêu cầu đến Server, Server xử lý yêu cầu và gửi lại phản hồi.

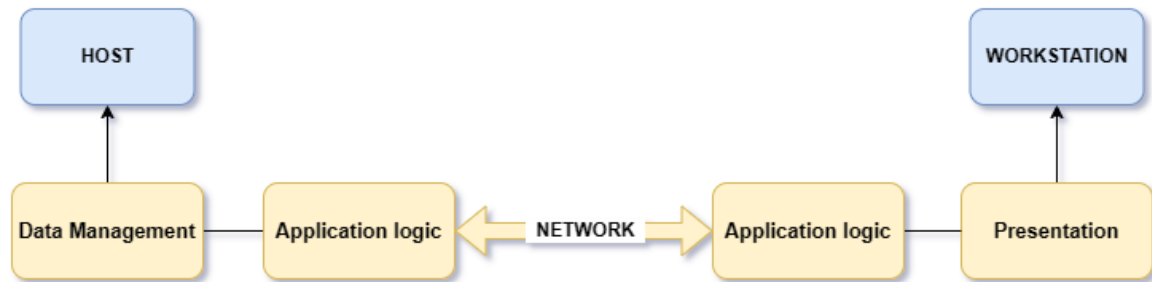


Hình 1: Mô hình Client – Server.

Một mô hình xử lý Client-Server phải có 3 thành phần cơ bản:

- Frontend Client: chạy trên trạm làm việc (Workstation), mà ở đó người sử dụng giao tiếp với ứng dụng để yêu cầu cung cấp dịch vụ.
- Backend Server: chạy trên máy chủ (Host) tiếp nhận thông tin và cung cấp dịch vụ được yêu cầu, cũng như phản hồi truy vấn.
- Mạng máy tính: Có chức năng truyền tải thông tin.

Trong đồ án này, sử dụng mô hình xử lý phân tán. Mô hình này chia sẻ năng lực tính toán trên máy chủ cho máy trạm. Máy trạm ngoài chức năng quản lý giao diện còn được phân bổ chức năng tính logic của cơ sở dữ liệu.

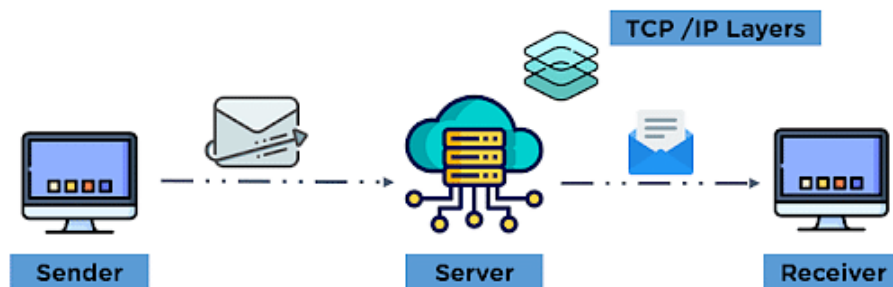


Hình 2: Mô hình xử lý phân tán.

1.2. Giao thức TCP (Transmission Control Protocol)

1.2.1. Khái niệm cơ bản về TCP

TCP là một trong hai giao thức chính trong bộ giao thức TCP/IP. Nó đảm bảo việc truyền thông tin giữa các ứng dụng trên mạng bằng cách tạo ra các kết nối ảo và đảm bảo rằng dữ liệu được gửi đi một cách tin cậy. TCP đảm bảo rằng dữ liệu được chia thành các gói nhỏ và gửi đi theo đúng thứ tự, đồng thời cũng đảm bảo rằng các gói dữ liệu được nhận đúng và không bị mất trong quá trình truyền.



Hình 3: Minh họa giao thức TCP.

1.2.2. Truyền dữ liệu

Phía gửi chia dữ liệu thành các gói tin nhỏ hơn và gán các số thứ tự vào mỗi gói tin để đảm bảo thứ tự chính xác khi nhận dữ liệu.

Với mỗi gói tin nhận được, yêu cầu bên nhận phải gửi phản hồi thông qua một gói xác nhận (ACK). Sau khi hết thời gian chờ hoặc không nhận được xác nhận, nguồn gửi sẽ gửi lại gói tin bị mất hoặc bị hoãn. Nhờ vậy, các vấn đề về lặp gói tin, truyền lại các gói dữ liệu bị hỏng hoặc mất và sai thứ tự gói tin đều được giải quyết.

1.2.3. Đặc điểm dữ liệu

Segmentation (phân đoạn): Dữ liệu truyền qua TCP được chia thành các phân đoạn (segments) để truyền đi qua mạng. Mỗi phân đoạn bao gồm dữ liệu và header chứa thông tin điều khiển.

Đảm bảo độ tin cậy: TCP sử dụng cơ chế ACK (Acknowledge) để xác nhận việc nhận dữ liệu từ phía máy nhận và có thể gửi lại các gói tin bị mất.

1.2.4. Ứng dụng và Ưu điểm

TCP được sử dụng rộng rãi trong các ứng dụng yêu cầu độ tin cậy cao như truyền file, trình duyệt web, email, và các ứng dụng yêu cầu kết nối ổn định.

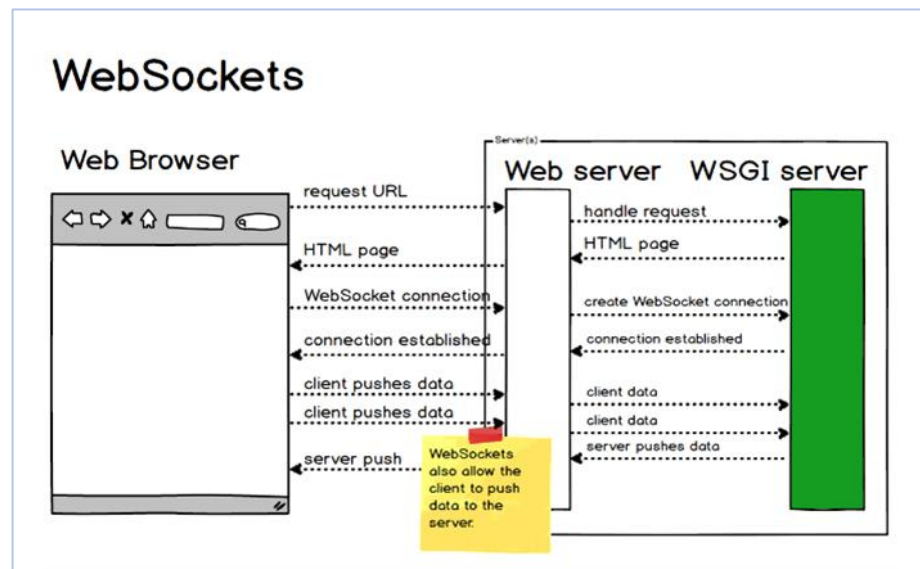
Độ tin cậy và khả năng đảm bảo việc truyền tải dữ liệu một cách an toàn và chính xác là những ưu điểm chính của TCP.

TCP là một trong những giao thức quan trọng trong mạng máy tính, cung cấp cơ chế để truyền tải dữ liệu một cách tin cậy và hiệu quả trên mạng Internet.

1.3. Mô hình WebSocket

1.3.1. Khái niệm cơ bản về WebSocket

WebSocket là một giao thức truyền tải dữ liệu hai chiều (full-duplex), cho phép truyền tải dữ liệu trong thời gian thực giữa trình duyệt web và máy chủ. WebSocket cho phép một kết nối duy trì giữa máy khách và máy chủ, vì vậy các thông tin có thể được gửi đi và nhận lại một cách hiệu quả và nhanh chóng mà không cần phải thiết lập kết nối mới mỗi khi truyền tải thông tin.



Hình 4: Minh họa giao thức WebSocket.

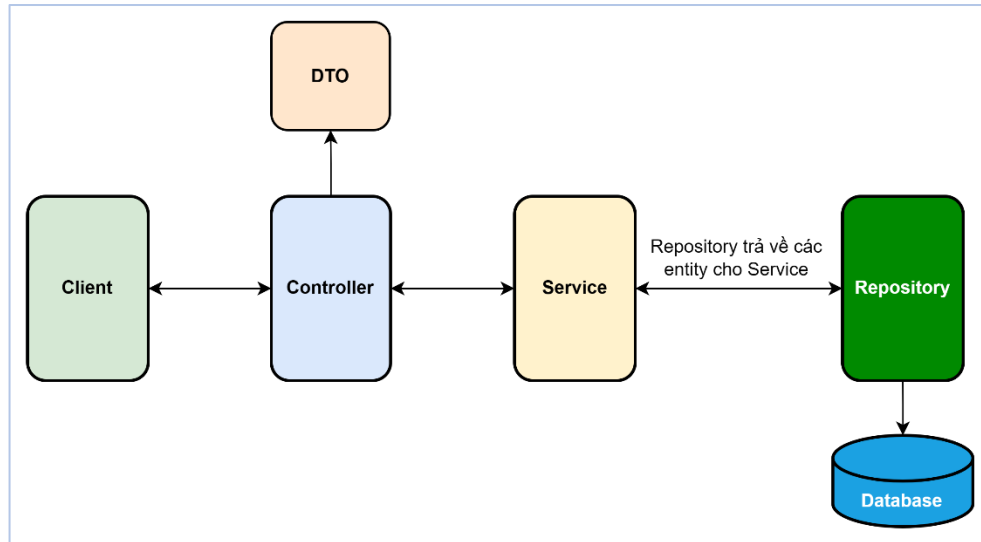
1.3.2. Ưu điểm và Ứng dụng

Truyền tải dữ liệu thời gian thực: WebSocket cung cấp kết nối liên tục, phù hợp cho việc truyền tải dữ liệu thời gian thực như trò chơi trực tuyến, ứng dụng trò chuyện, và thông tin thị trường tài chính.

Hiệu quả và tiết kiệm tài nguyên: So với việc sử dụng HTTP truyền thống, WebSocket giúp truyền tải dữ liệu nhanh chóng và tiết kiệm tài nguyên máy chủ và máy khách.

1.4. Kiến trúc MVC trong Spring Boot

Kiến trúc MVC trong Spring Boot được xây dựng dựa trên tư tưởng "độc lập" kết hợp với các nguyên lý thiết kế hướng đối tượng. Độc lập ở đây chỉ việc các layer phục vụ các mục đích nhất định, khi muốn thực hiện một công việc ngoài phạm vi thì sẽ đưa công việc xuống các tầng thấp hơn.

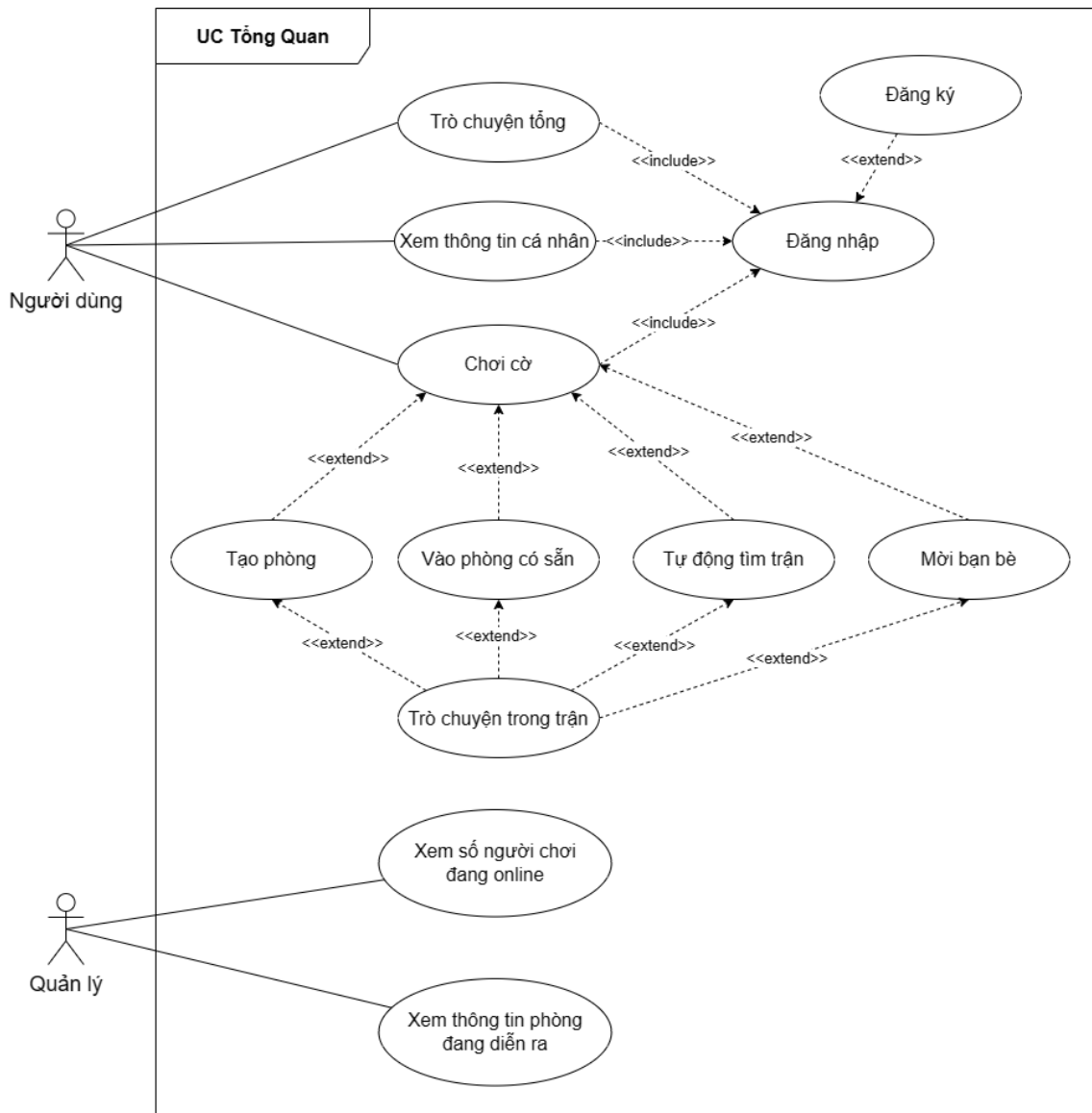


Hình 5: Kiến trúc Spring Boot.

- Tầng Controller: là tầng giao tiếp với bên ngoài, xử lý các yêu cầu và trả dữ liệu cho người dùng.
- Tầng Service: thực hiện các logic nghiệp vụ và xử lý các yêu cầu từ tầng Controller.
- Tầng Repository: chịu trách nhiệm giao tiếp với cơ sở dữ liệu, thực hiện các thao tác truy vấn và cung cấp dữ liệu mà tầng Service yêu cầu.
- Entity: chứa các thực thể tương ứng với các table trong cơ sở dữ liệu.
- DTO: là các class đóng gói data để chuyển giữa Client - Server hoặc giữa các Service. Mục đích tạo ra DTO là để giảm bớt lượng thông tin không cần thiết phải chuyển đi, và cũng tăng cường độ bảo mật.

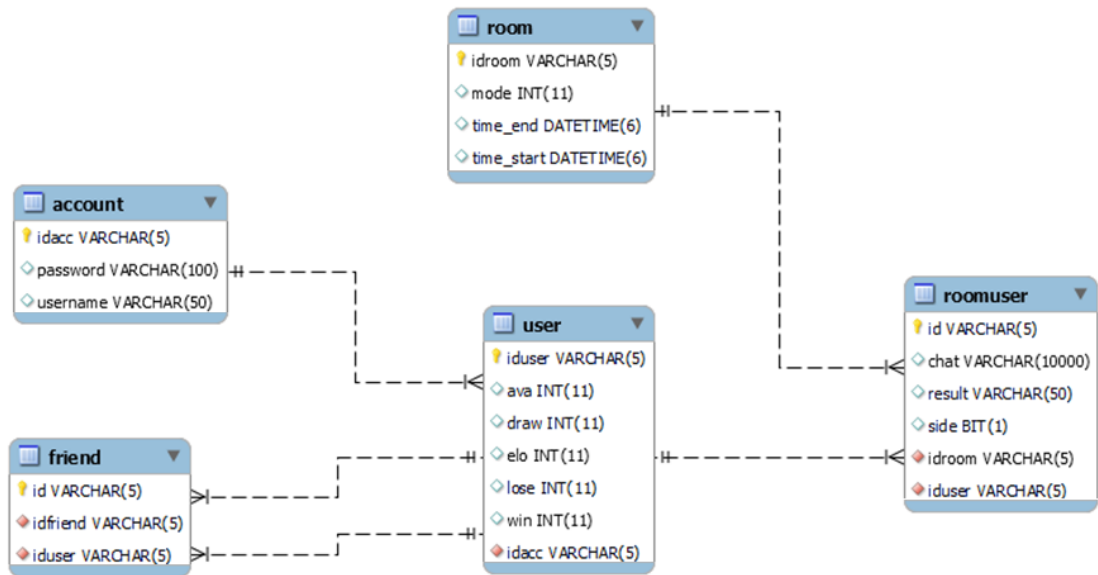
CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

2.1. Usecase tổng quan



Hình 6: Sơ đồ Usecase tổng quan.

2.2. Cơ sở dữ liệu



Hình 7: Cơ sở dữ liệu.

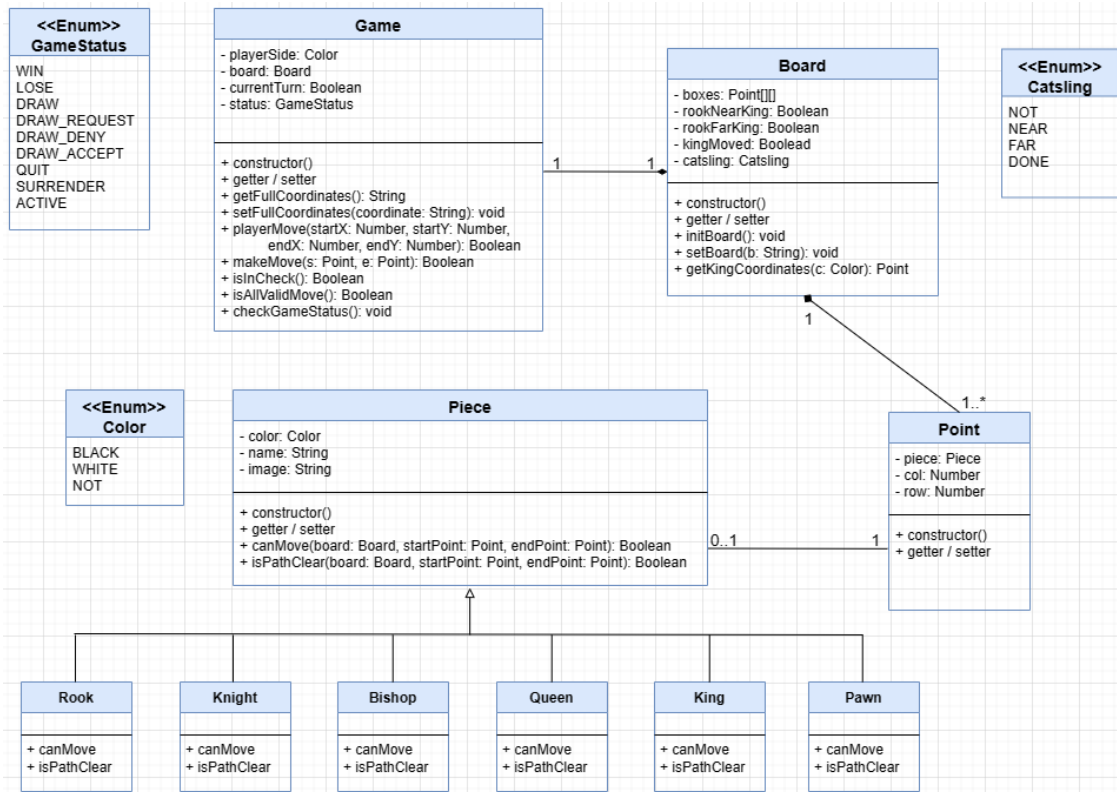
Danh sách các bảng trong cơ sở dữ liệu:

Số thứ tự	Tên bảng	Mô tả
1	account	Lưu thông tin tài khoản.
2	user	Lưu dữ liệu của người dùng.
3	friend	Lưu danh sách bạn bè của người dùng
4	room	Lưu thông tin phòng.
5	roomuser	Liên kết phòng và người dùng.

Bảng 4: Mô tả các bảng trong cơ sở dữ liệu.

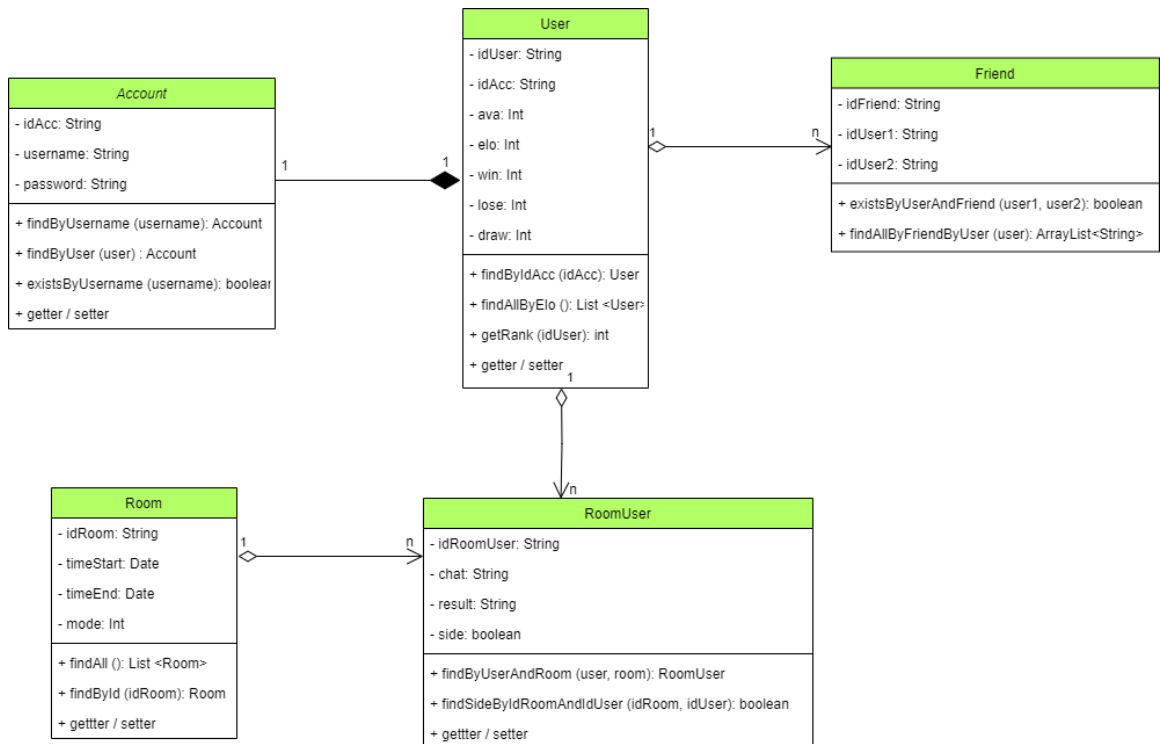
2.3. Biểu đồ lớp

2.4.1. Phía Client



Hình 8: Biểu đồ lớp phía Client.

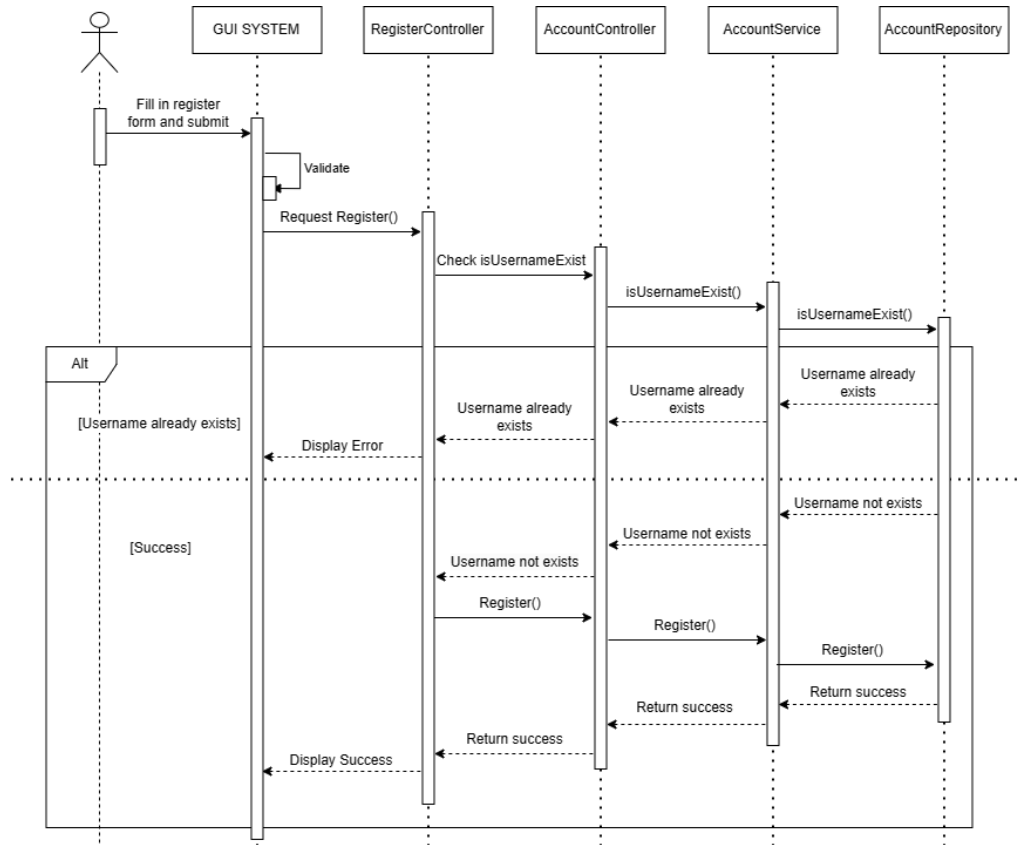
2.4.2. Phía Server



Hình 9: Biểu đồ lớp phía Server.

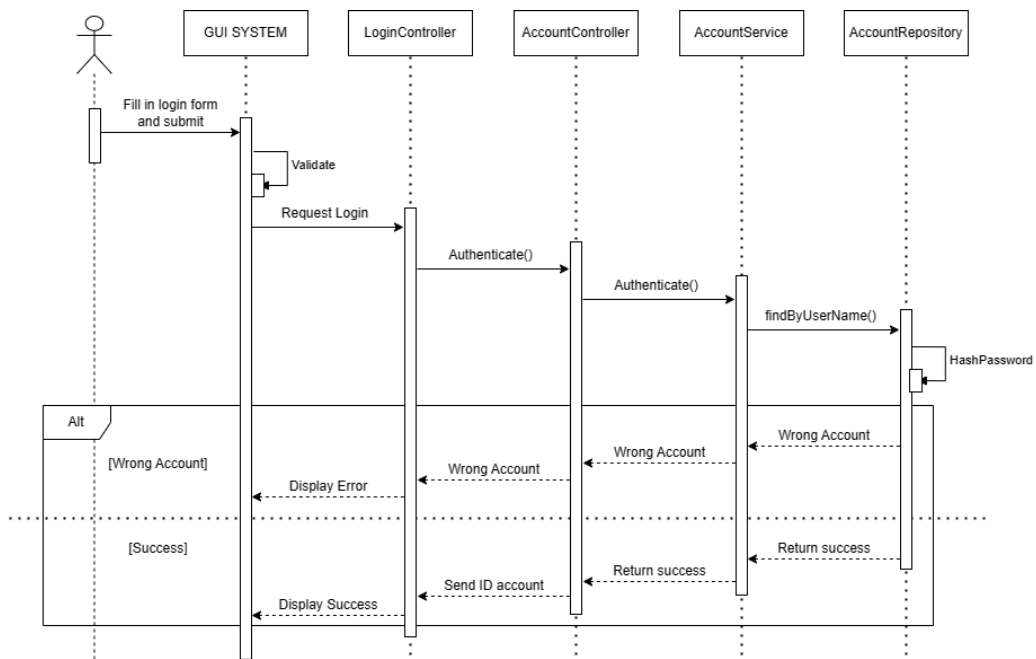
2.4. Biểu đồ tuần tự (sequence diagram)

2.4.1. Chức năng đăng ký



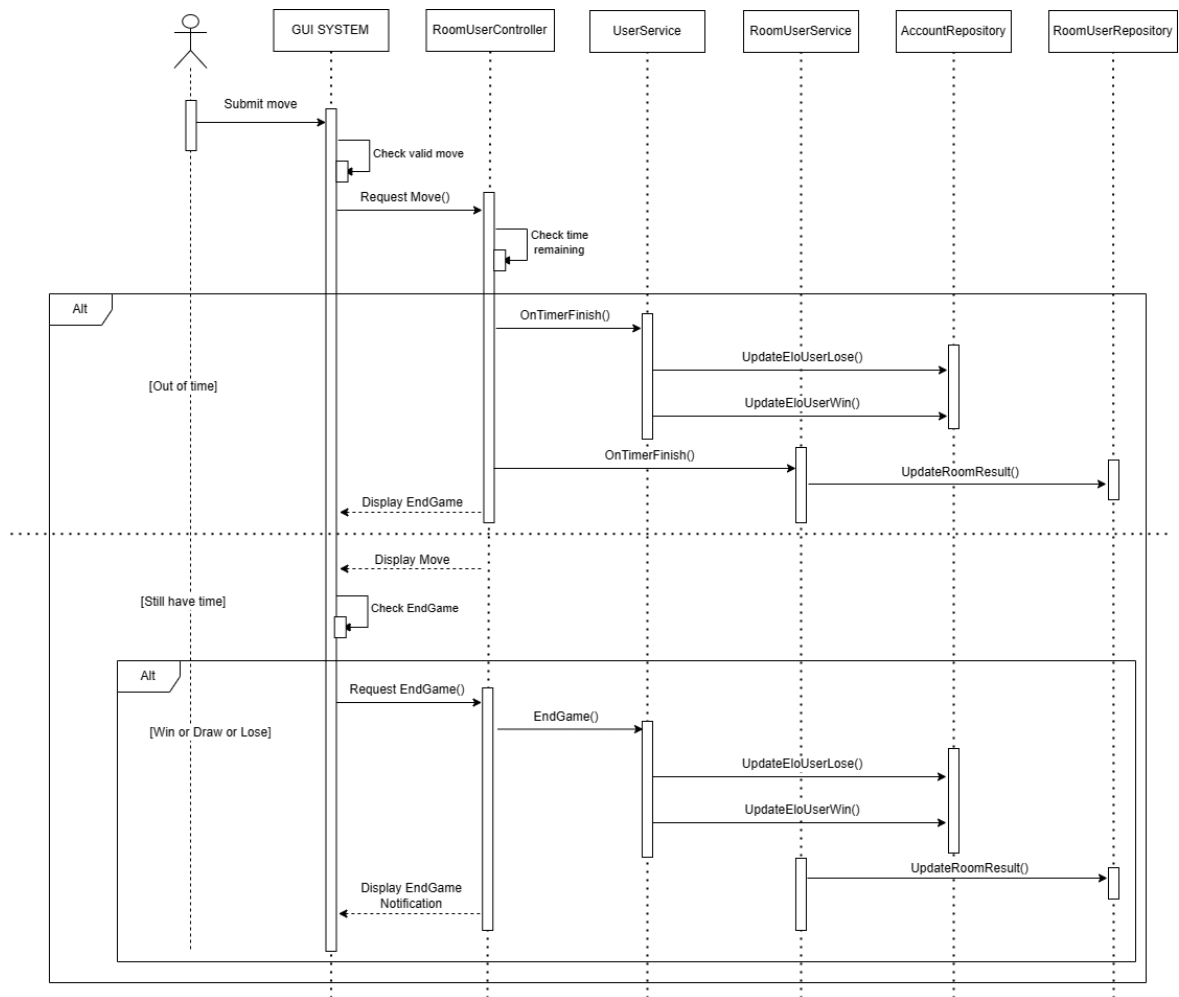
Hình 10: Biểu đồ tuần tự chức năng đăng ký.

2.4.2. Chức năng đăng nhập



Hình 11: Biểu đồ tuần tự chức năng đăng nhập.

2.4.3. Chức năng chơi cờ



Hình 12: Biểu đồ tuần tự chức năng chơi cờ.

2.5. YÊU CẦU PHI CHỨC NĂNG

2.5.1. Yêu cầu về tổ chức, tiến trình phát triển

- Hệ thống được viết bằng ngôn ngữ Java, triển khai theo giao thức WebSocket, sử dụng Spring Boot.
- Phía Client được thiết kế bằng HTML, CSS, Typescript.
- Hệ thống sử dụng hệ quản trị Cơ sở dữ liệu MySQL.
- Dữ liệu trên cơ sở dữ liệu phải được đồng bộ liên tục, hợp lý, tránh thất thoát và dư thừa dữ liệu, gây ra những ảnh hưởng không tốt lên các chức năng của trang web.

2.5.2. Yêu cầu về độ tin cậy

- Hệ thống phải đảm bảo mật khẩu khách hàng, mật khẩu của người dùng được mã hóa thông qua hash password hỗ trợ của framework.

2.5.3. Yêu cầu về sản phẩm

- Giao diện người dùng của hệ thống phải dễ sử dụng và thân thiện với người dùng. Các chức năng và tính năng phải được đơn giản hóa và dễ hiểu để người dùng có thể tương tác với hệ thống một cách thuận tiện và nhanh chóng.
- Hệ thống cung cấp các hướng dẫn sử dụng rõ ràng và hỗ trợ người dùng khi gặp vấn đề.
- Màu sắc sử dụng phải hài hòa, hợp lý, không gây khó chịu cho người dùng.
- Giao diện ngôn ngữ Tiếng Việt.

2.5.4. Khả năng tái sử dụng

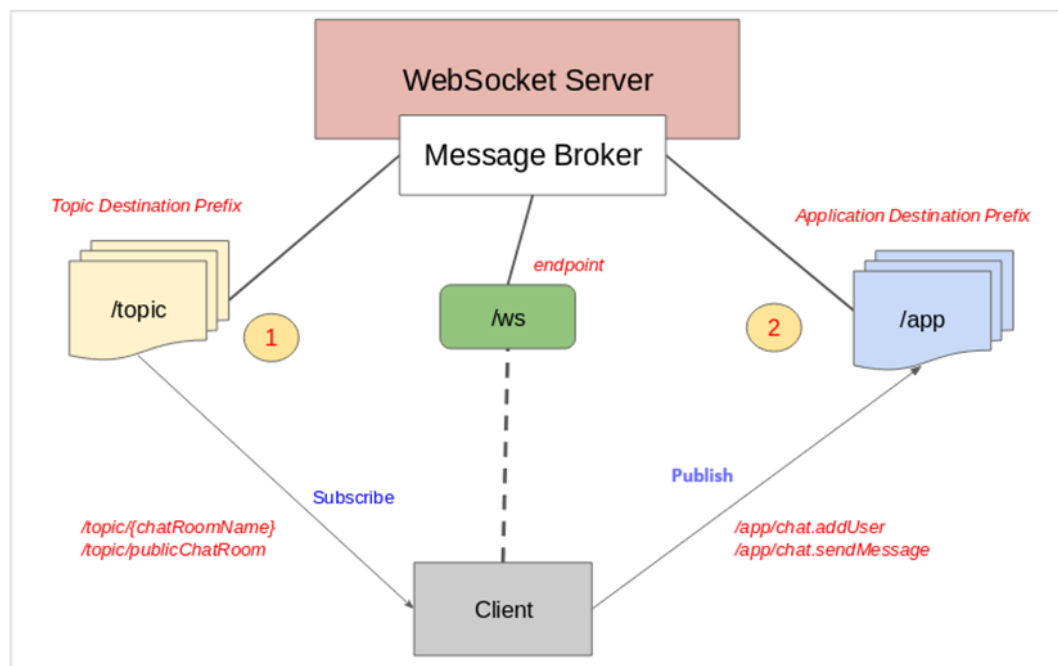
- Tổ chức các folder, source code một cách hợp lý, dễ hiểu. Phân chia thành các lớp riêng biệt, giảm thiểu sự phụ thuộc lẫn nhau.
- Đảm bảo khả năng phát triển, mở rộng trở thành một phần mềm ngày càng chuẩn hóa, sát với thực tế.

CHƯƠNG 3: TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ

2.6. Triển khai mô hình

3.2.1. Cách thức hoạt động

Dự án sẽ hoạt động dựa trên giao thức WebSocket và STOMP, sử dụng Java cho phần máy chủ và TypeScript cho phần máy khách, được hỗ trợ bởi các thư viện cần thiết.



Hình 13: Mô hình hoạt động dự án.

Bước 1: Thêm phụ thuộc (dependency) WebSocket vào dự án

```
implementation 'org.springframework.boot:spring-boot-starter-websocket'
```

Hình 14: Thêm phụ thuộc websocket.

Bước 2: Cấu hình các endpoint WebSocket và STOMP phía server

```
@EnableWebSocketMessageBroker
public class WebSocketConfig implements WebSocketMessageBrokerConfigurer {

    1 usage 1 phuocduy *
    @Override
    public void configureMessageBroker(MessageBrokerRegistry config) {
        config.enableSimpleBroker(...destinationPrefixes: "/topic", "/queue");
        config.setApplicationDestinationPrefixes("/app"); // Xử lý các tin nhắn đến /app
    }

    1 usage 1 phuocduy *1
    @Override
    public void registerStompEndpoints(StompEndpointRegistry registry) {
        registry.addEndpoint(...paths: "/ws").setHandshakeHandler(new UserHandshakeHandler()).setAllowedOriginPatterns("*").withSockJS();
    }

    1 usage 1 loochuynhh
    @Override
    public void configureClientInboundChannel(ChannelRegistration registration) {
        registration.interceptors(new UserInterceptor());
    }
}
```

Hình 15: Cấu hình Endpoint và STOMP phía server.

Bước 3: Triển khai các controller xử lý các yêu cầu của người dùng – nơi phát và nhận tin nhắn của người dùng đã đăng ký.

```
@PostMapping("/publicChat")
@SendTo("/topic/publicChat")
public publicChat publicChat(publicChat message) {
    if(userService.getUserById(message.getIdDUserSend())!=null){
        message.setUserSendName(userService.getUsernameByUserID(message.getIdDUserSend()));
        message.setAva(userService.getUserById(message.getIdDUserSend()).getAva());
        return message;
    }else{
        return null;
    }
}
```

Hình 16: Nơi xử lý nhận và phát tin nhắn phía server

Bước 4: Thiết lập socket ở client để kết nối tới server và cấu hình các giá trị mặc định cho mỗi client.

```
const socket = new SockJS('http://' + window.location.hostname + ':8888/ws');
export const stompClient = new Client({
    websocketFactory: () => socket,
    connectHeaders: {
        userID: localStorage.getItem('userID')!
    },
    debug: (msg) => console.log(msg),
    reconnectDelay: 1000,
    heartbeatIncoming: 1000,
    heartbeatOutgoing: 1000,
});
```

Hình 17: Import thư viện và cấu hình các giá trị STOMP phía client

Bước 5: Thực hiện đăng ký (subscribe) và gửi thông điệp (publish) tới các endpoint đã cài đặt ở phía server để tiến hành trao đổi thông tin.

```
stompClient.publish({
  destination: '/app/publicChat',
  headers: {},
  body: JSON.stringify({ idUserSend: localStorage.getItem('userID'), chat: inputValue, ava: null }),
});
stompClient.subscribe('/topic/publicChat', (message) => {
  const body = JSON.parse(message.body);
  ChatContentFrom(body.userSendName, body.ava, body.chat, true);
});
```

Hình 18: Đăng ký và lắng nghe phía client

3.2.2. Định danh client khi kết nối đến server thông qua WebSocket

- Cấu hình WebSocket và thêm một interceptor để thực hiện định danh cho mỗi kết nối đến server.

```
1 usage  ↳ loochuynhh
@Override
public void configureClientInboundChannel(ChannelRegistration registration) {
    registration.interceptors(new UserInterceptor());
}
}
```

Hình 19: Định danh client trong WebSocket.

- Tạo class PrincipalCustomer implement interface Principal đại diện cho mỗi client kết nối, với 2 thuộc tính userID (dùng để danh client) và status (dùng để chỉ trạng thái client).
- Tạo class UserInterceptor implement interface ChannelInterceptor. Class này sẽ override phương thức preSend và có một danh sách chứa các PrincipalCustomer đang kết nối đến server.
- Phương thức preSend sẽ giúp server nhận diện client qua userID. Việc sử dụng userID (mỗi client chỉ có 1 userID duy nhất) thay vì UUID sẽ giúp server nhận biết nếu client đăng nhập trên nhiều thiết bị.

```

private static Map<String, PricipalCustomer> userMap = new HashMap<>();
@Override
public Message<> preSend(Message<> message, MessageChannel channel) {
    StompHeaderAccessor accessor = MessageHeaderAccessor.getAccessor(message, StompHeaderAccessor.class);

    if (StompCommand.CONNECT.equals(accessor.getCommand())) {
        Object raw = message.getHeaders().get(SimpMessageHeaderAccessor.NATIVE_HEADERS);

        if (raw instanceof Map) {
            Object userID = ((Map) raw).get("userID");

            if (userID instanceof ArrayList) {
                String name = ((ArrayList<String>) userID).get(0).toString();
                PricipalCustomer pricipalCustomer = userMap.get(name);
                //userID null trước khi vào chức năng login
                if ("null".equals(name)){
                    final String randomId = UUID.randomUUID().toString();
                    pricipalCustomer = new PricipalCustomer(randomId, "ONLINE");
                    changeOnline("INCREASE", randomId);
                    userMap.put(randomId, pricipalCustomer);
                }else{
                    //Có userID, lần đầu đăng nhập
                    if (pricipalCustomer == null) {
                        pricipalCustomer = new PricipalCustomer(name, "ONLINE");
                        changeOnline("INCREASE", name);
                        userMap.put(name, pricipalCustomer);
                    }
                }
                accessor.setUser(pricipalCustomer);
            }
        }
    }
    return message;
}

```

Hình 20: Cấu hình các thông số định danh.

3.2.3. Xử lý thời gian

Để tăng tính bảo mật, việc đếm và quản lý thời gian sẽ được thực hiện ở Server thông qua thư viện hỗ trợ là CountdownTimerListener, được chạy trên một luồng riêng, độc lập bằng ScheduledExecutorService. Mỗi lần đếm ngược, thông tin sẽ được gửi về cho Client để hiển thị. Trong trường hợp thời gian kết thúc, một thông báo về việc hết giờ sẽ được gửi và bộ đếm sẽ được loại bỏ khỏi vùng nhớ.

⇒ Bộ đếm thời gian thực, tránh trường hợp xung đột thời gian và gian lận

```

1 usage  ▲ loochuynhh
public void startCountdown() {
    executorService.scheduleAtFixedRate(() -> {
        if (shouldDecreaseValue && countdownValue >= 0) {
            listener.countdown(idUserSend, idUserReceive, countdownValue);
            countdownValue--;
        } else if (countdownValue >= 0) {
            System.out.println("Stop countdown for " + idRoomUser + ": " + countdownValue);
        } else {
            stopCountdown();
            if (listener != null) {
                listener.onTimerFinish(idRoomUser, idRoomUserReceive, idRoom, idUserSend, idUserReceive);
            }
        }
    }, initialDelay: 0, period: 1, TimeUnit.SECONDS);
}

```

Hình 21: Thực hiện quản lý thời gian phía server

3.2.4. Xử lý phân trang trong chức năng xem bảng xếp hạng

- Trong quá trình sử dụng chức năng xem bảng xếp hạng dựa trên điểm số của người chơi trên toàn máy chủ, việc gửi toàn bộ dữ liệu mỗi khi client yêu cầu có thể gây quá tải cho hệ thống. Do đó, kỹ thuật phân trang được áp dụng để giảm tải áp lực cho hệ thống.
- Tạo interface UserRepository extends JpaRepository, trong đó có hàm sắp xếp user theo elo.

```
1 usage  phuocduy
@Query("SELECT u FROM User u ORDER BY u.elo DESC, u.idUser DESC")
Page<User> findAllByOrderByEloDesc(Pageable pageable);
```

Hình 22: Hàm sắp xếp user theo elo.

- Tại UserService, tạo một hàm chuyển đổi Page thành List, trả về dữ liệu của trang tương ứng do client yêu cầu.

```
public List<String> getTopUsers(int pageIndex, int pageSize) {
    pageIndex--; // pageIndex đếm từ 0 nên phải giảm
    Page<User> userPage = userRepository.findAllByOrderByEloDesc(PageRequest.of(pageIndex, pageSize));

    // Chuyển đổi Page<User> thành List<String>
    List<String> userIds = userPage.getContent().stream().map(User::getIdUser).collect(Collectors.toList());

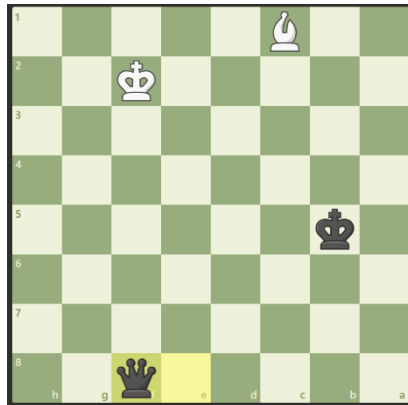
    return userIds;
}
```

Hình 23: Hàm trả về danh sách các user.

3.2.5. Xử lý logic cờ vua

- Lớp cơ sở **Piece.ts** là lớp trừu tượng, trong đó có các phương thức với quyền truy cập public:
 - **Constructor**: được định nghĩa dùng để khởi tạo quân cờ.
 - **canMove()**: là hàm thuần ảo yêu cầu các lớp dẫn xuất phải định nghĩa lại, mục đích của hàm là xây dựng quy tắc di chuyển của quân cờ.
 - **isPathClear()**: là hàm thuần ảo yêu cầu các lớp dẫn xuất phải định nghĩa lại, mục đích của hàm là kiểm tra có quân cờ nào nằm giữa đường đi từ vị trí xuất phát đến vị trí đích.
- Lớp dẫn xuất: **Rook.ts**, **Knight.ts**, **Bishop.ts**, **Queen.ts**, **King.ts**, **Pawn.ts**
- Tạo class game đại diện cho ván cờ. Class game sẽ có 2 phương thức chính:
 - **isInCheck()**: kiểm tra xem ván cờ có đang bị chiếu hay không. Trong quá trình kiểm tra, chỉ cần xem xét các hướng chiếu cơ bản như hàng ngang, hàng dọc, hàng chéo của quân vua và 8 vị trí có thể chiếu vua của quân mã. Điều này giúp giảm độ phức tạp của vòng lặp, tối ưu hóa hiệu suất của chức năng.

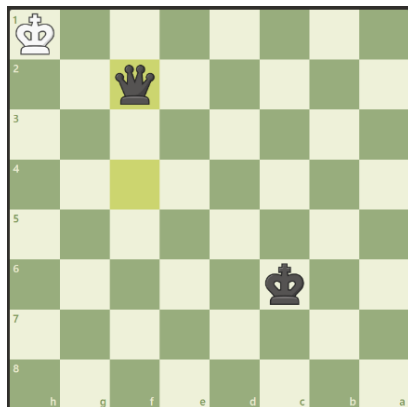
- *isValidMove()*: kiểm tra còn nước đi hợp lệ nào không. Cách thức hoạt động của hàm là lựa chọn một quân cờ bất kỳ thuộc phe của người chơi hiện tại và sau đó sử dụng thuật toán vét cạn để tìm ra nước đi hợp lệ giúp quân vua thoát khỏi thế bị chiếu.



Quân tượng đi đến ô f4 là một nước đi hợp lệ giúp vua trắng thoát khỏi thế bị chiếu.

Hình 24: Ví dụ chức năng của hàm *isValidMove()*.

- Sau mỗi nước đi của đối thủ, phương thức *isInCheck()* và *isValidMove()* sẽ được kích hoạt để kiểm tra tình trạng của ván cờ:
 - Nếu đang bị chiếu vua và không còn nước đi hợp lệ, phe người chơi hiện tại bị xử thua.
 - Nếu không bị chiếu vua và nhưng không còn nước đi hợp lệ, trận đấu kết thúc với tình trạng hòa.
 - Các trường hợp còn lại, ván cờ sẽ tiếp tục diễn ra với lượt đi của người chơi tiếp theo.

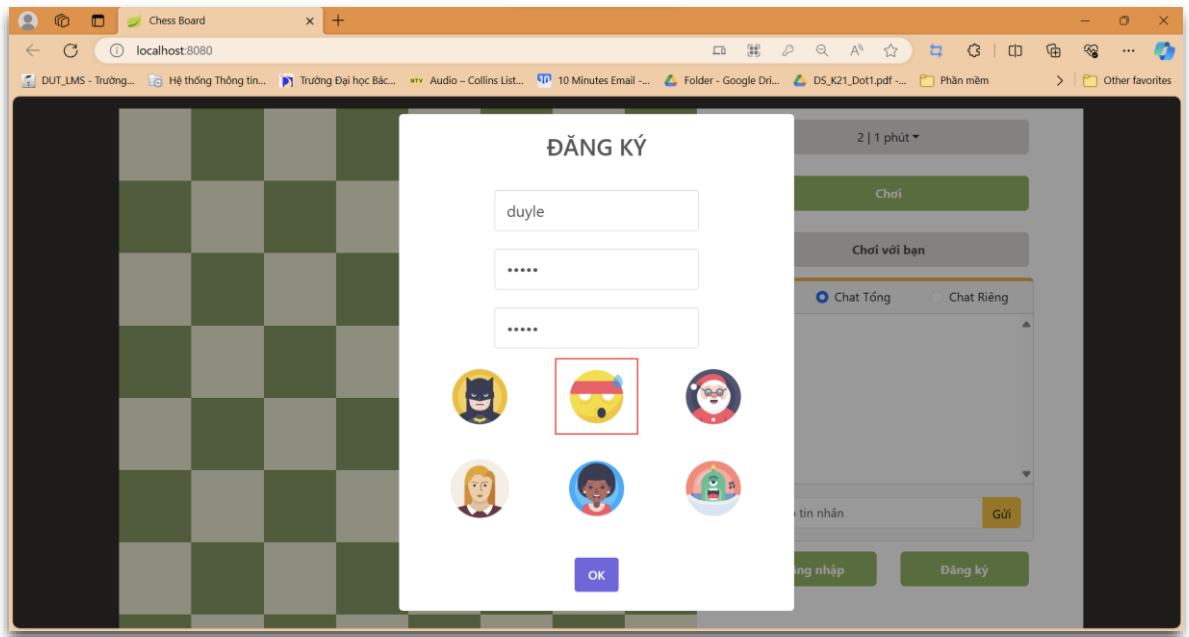


Quân trắng không bị chiếu, tuy nhiên không tồn tại nước đi hợp lệ nên ván cờ này được xử hòa.

Hình 25: Ví dụ trường hợp hòa cờ.

3.2. Kết quả

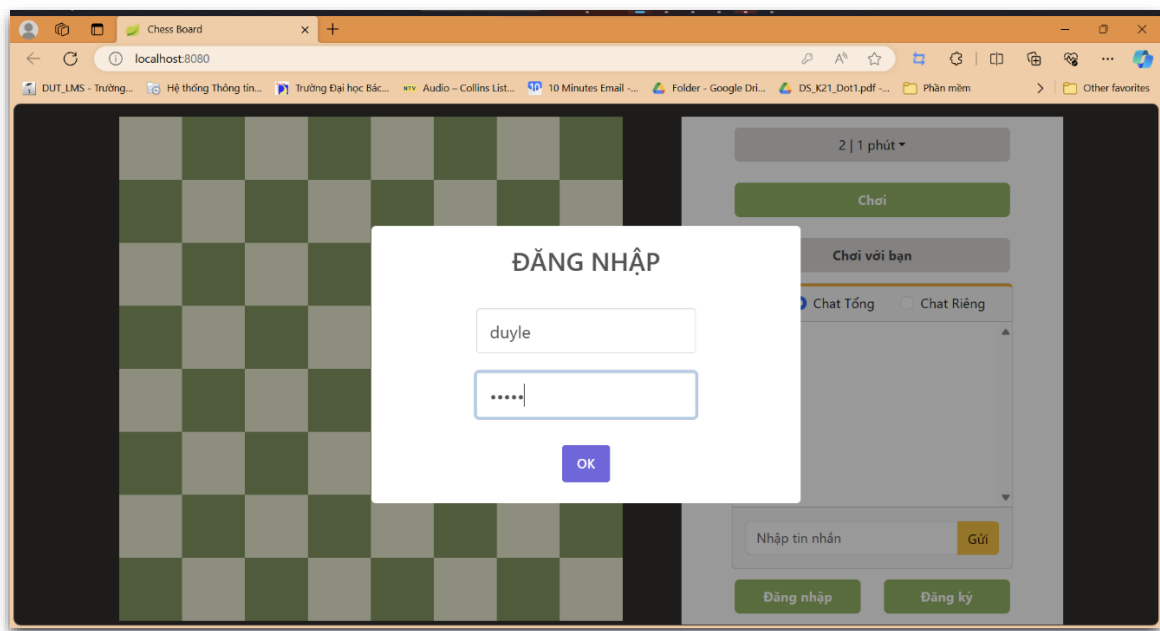
3.2.1. Chức năng đăng nhập, đăng kí



Hình 26: Giao diện chức năng đăng ký.

Màn hình	Giao diện đăng ký.	
Mô tả	Sử dụng cho chức năng đăng ký.	
Nội dung màn hình		
Mục	Kiểu	Mô tả
Tên tài khoản	Text	Tên tài khoản đăng ký (nếu tài khoản đã tồn tại, thông báo lỗi).
Mật khẩu	Password	Mật khẩu tài khoản đăng ký (tối thiểu 5 kí tự).
Xác nhận mật khẩu	Password	Nhập lại mật khẩu.
Avatar	RadioButton	Ảnh đại diện tài khoản.
Ok	Button	Người dùng đăng ký mới tài khoản.

Bảng 5: Mô tả giao diện đăng ký.

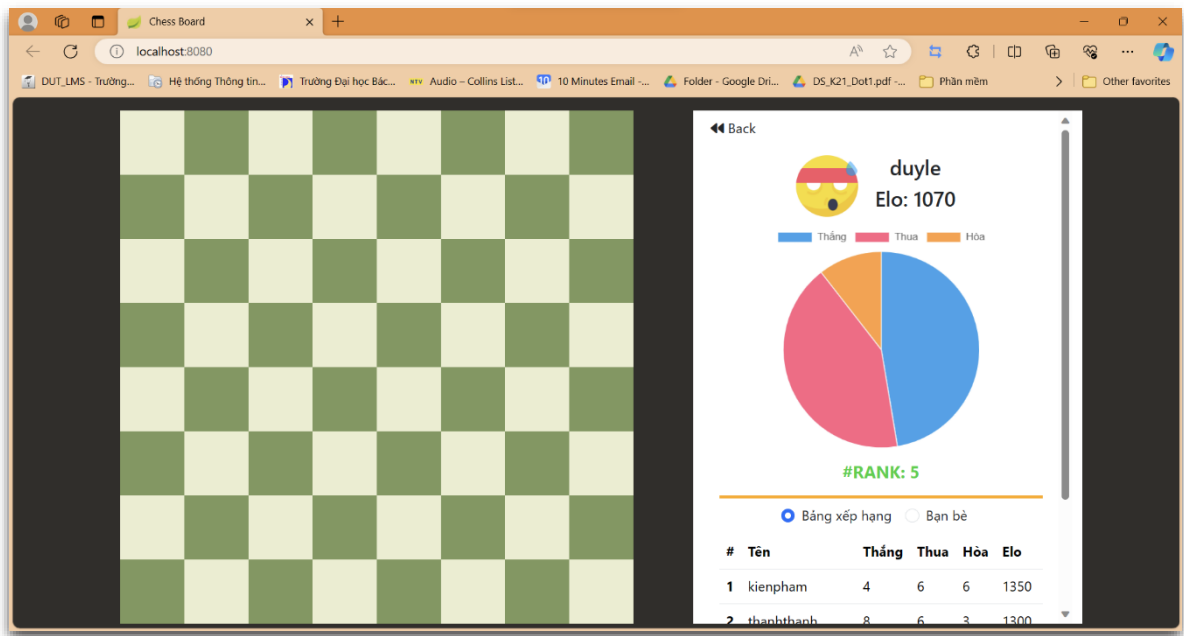


Hình 27: Giao diện chức năng đăng nhập.

Màn hình	Giao diện đăng nhập.	
Mô tả	Sử dụng cho chức năng đăng nhập.	
Nội dung màn hình		
Mục	Kiểu	Mô tả
Tên tài khoản	Text	Tên tài khoản người truy cập.
Mật khẩu	Text	Mật khẩu người truy cập (tối thiểu 5 kí tự).
Ok	Button	Đăng nhập vào tài khoản người dùng nếu hợp lệ và thông báo lỗi nếu thất bại.

Bảng 6: Mô tả giao diện đăng nhập.

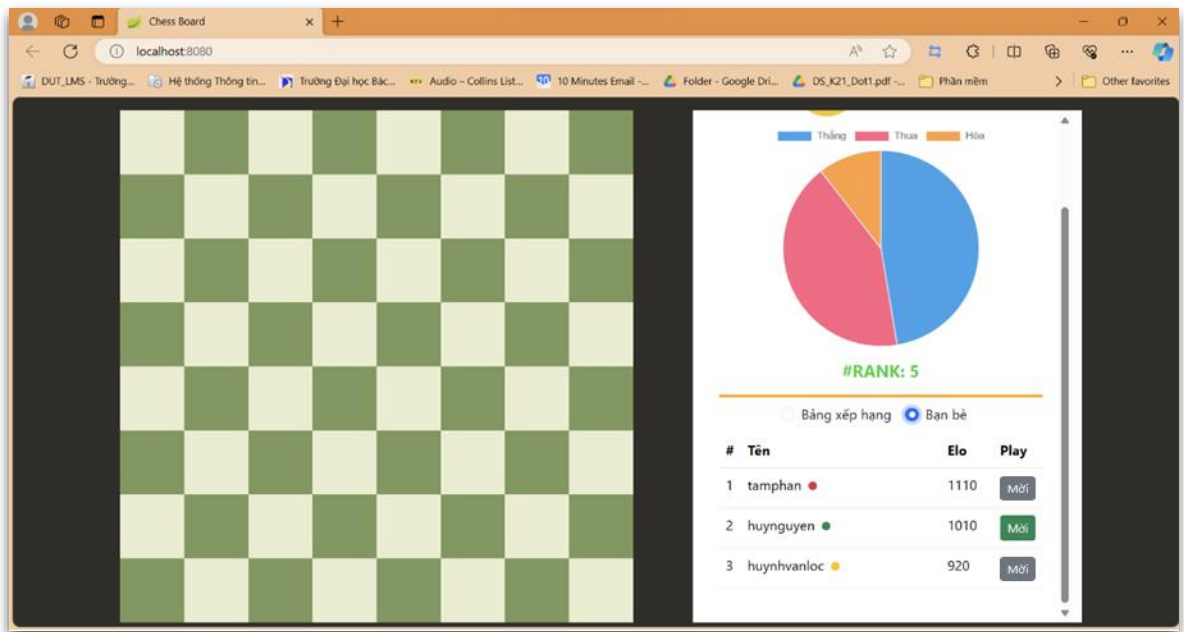
3.2.2. Chức năng xem thông tin cá nhân



Hình 28: Giao diện xem thông tin cá nhân và bảng xếp hạng.

Màn hình	Giao diện thông tin cá nhân.
Mô tả	Sử dụng cho chức năng xem thông tin cá nhân.
Nội dung màn hình	<ul style="list-style-type: none"> - Hiện thị tên, elo, avatar tài khoản. - Thống kê tổng số trận thắng, thua, hòa. - Hiện thị thứ hạng tài khoản. - Bảng xếp hạng người chơi (sắp xếp theo elo, bao gồm tên, tổng số trận thắng, thua, hòa).

Bảng 7: Mô tả giao diện xem thông tin cá nhân và bảng xếp hạng.

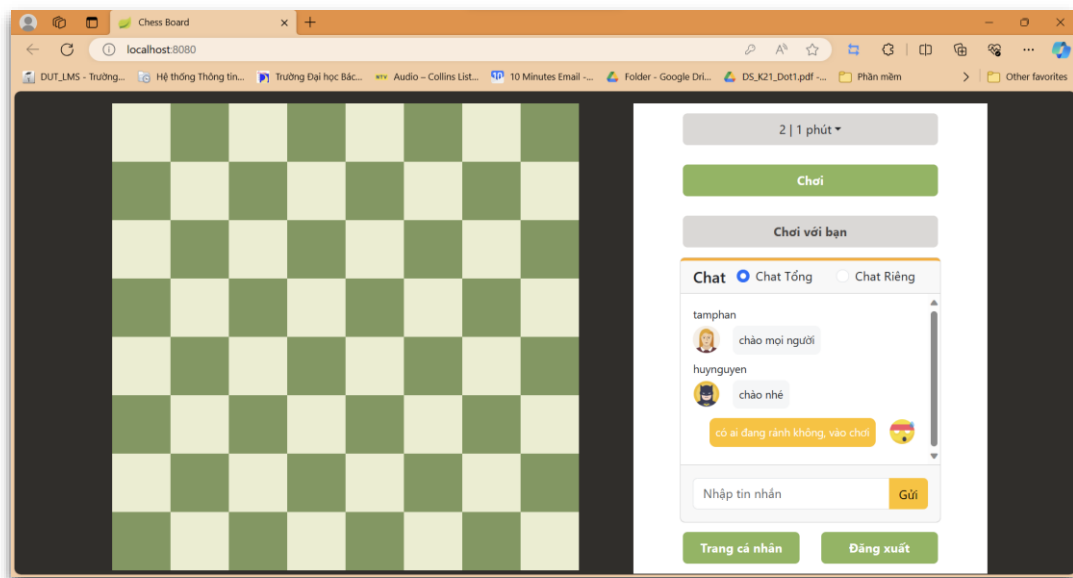


Hình 29: Giao diện xem thông tin bạn bè.

Màn hình	Giao diện xem thông tin bạn bè.
Mô tả	Sử dụng cho chức năng xem thông tin bạn bè.
Nội dung màn hình	<ul style="list-style-type: none"> - Hiển thị danh sách bạn bè (bao gồm thông tin về tên, elo, trạng thái hoạt động). - Trạng thái hoạt động được hiển thị bằng 3 màu sắc: màu vàng (người chơi đang không hoạt động), màu đỏ (người chơi đang trong trận đấu), màu xanh (người chơi đang hoạt động). - Chức năng mời chỉ đối với người chơi đang hoạt động (nhưng không trong trận).

Bảng 8: Mô tả giao diện xem thông tin bạn bè.

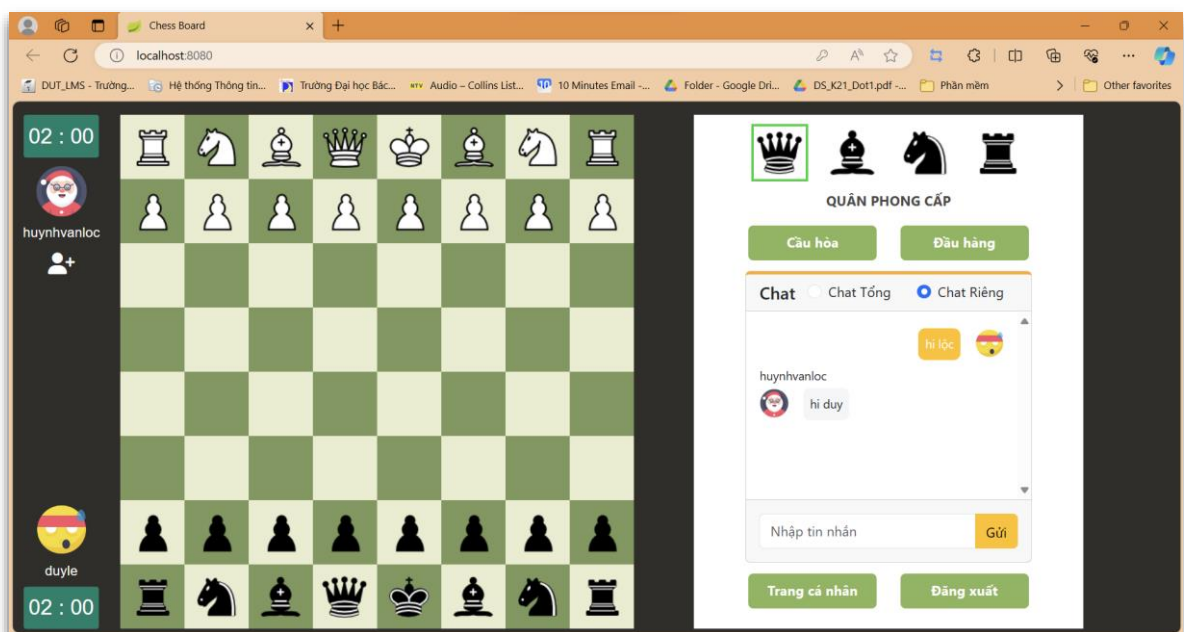
3.2.3. Chức năng nhắn tin tổng, nhắn tin riêng



Hình 30: Giao diện chức năng nhắn tin tổng.

Màn hình	Giao diện chức năng nhắn tin tổng.
Mô tả	Sử dụng để nhắn tin đến toàn bộ người chơi trên hệ thống.
Nội dung màn hình	<ul style="list-style-type: none"> - Hiển thị avatar bản thân. - Hiển thị tên, avatar tài khoản nhắn đến. - Hiển thị nội dung tin nhắn.

Bảng 9: Mô tả giao diện nhắn tin tổng.

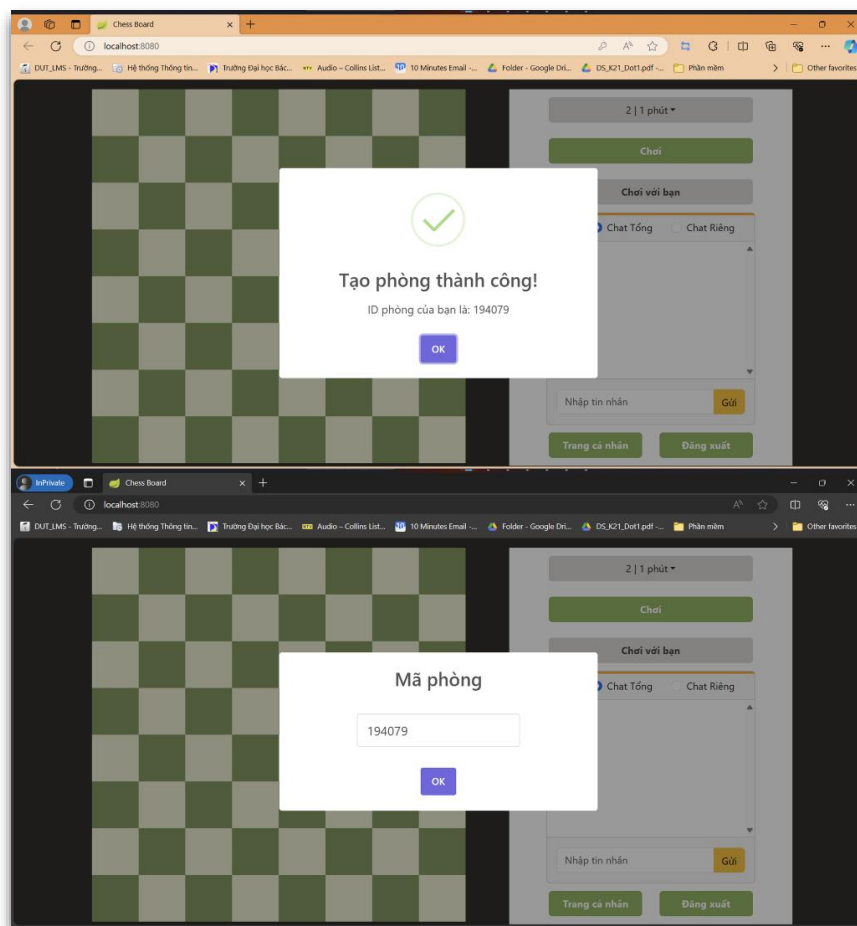


Hình 31: Giao diện chức năng nhắn tin riêng.

Màn hình	Giao diện chức năng nhắn tin riêng.
Mô tả	Sử dụng để nhắn tin đến đối thủ đang trong trận.
Nội dung màn hình	<ul style="list-style-type: none"> - Hiện thị avatar bản thân. - Hiện thị tên, avatar tài khoản đối thủ. - Hiện thị nội dung tin nhắn.

Bảng 5.2.6: Mô tả giao diện nhắn tin riêng.

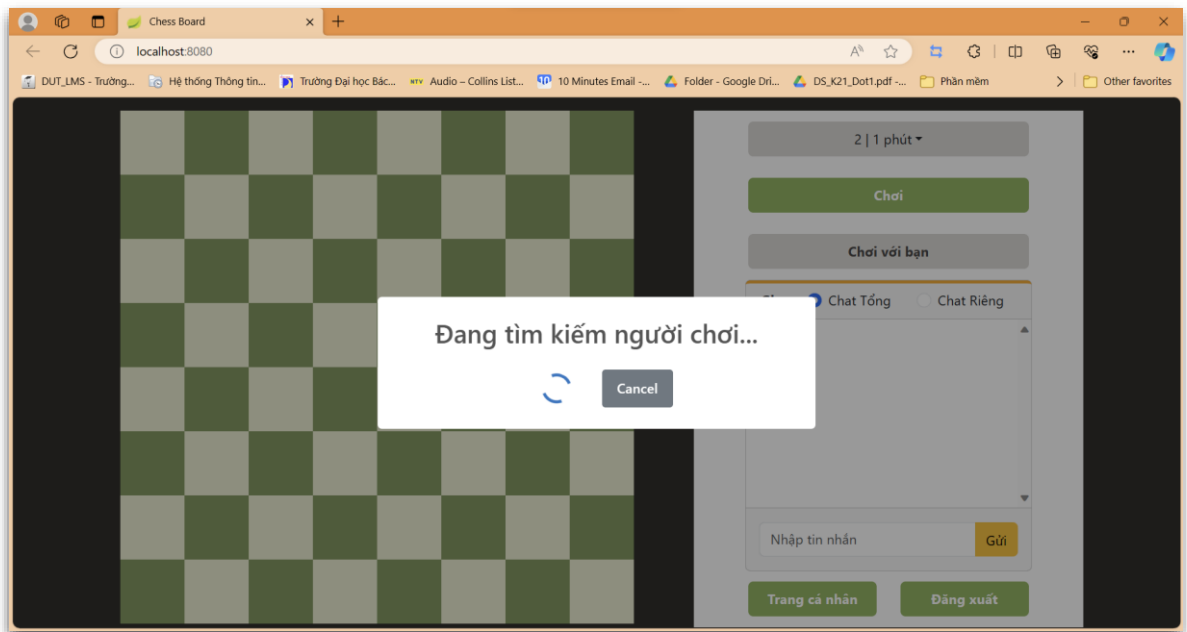
3.2.4. Chức năng chơi cờ



Hình 32: Giao diện chức năng chơi với bạn.

Màn hình	Giao diện chức năng chơi với bạn.
Mô tả	Người chơi có thể tự tạo phòng để mời người khác tham gia.
Nội dung màn hình	<ul style="list-style-type: none"> - Người chơi tạo phòng, hiển thị ID phòng. - Đối thủ nhập ID phòng để tham gia trận.

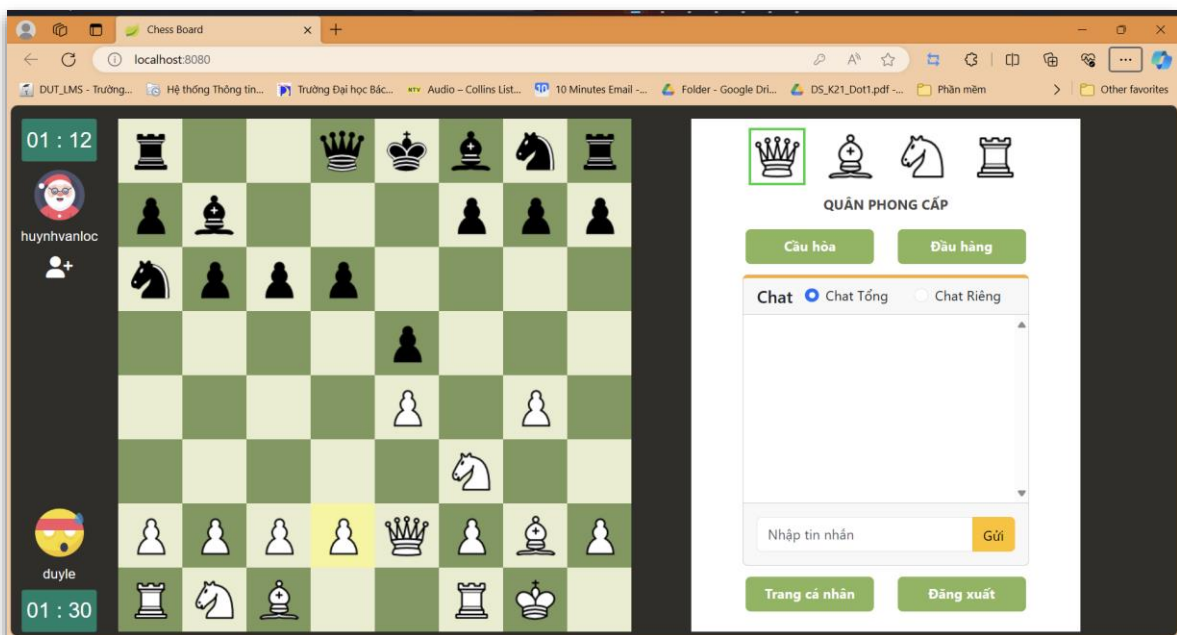
Bảng 10: Mô tả chức năng chơi với bạn.



Hình 33: Giao diện chức năng chơi thể giới.

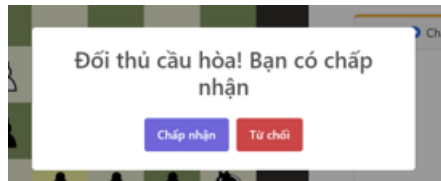
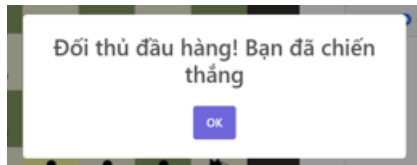

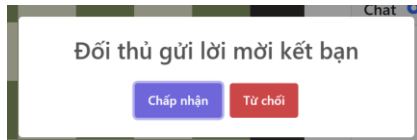
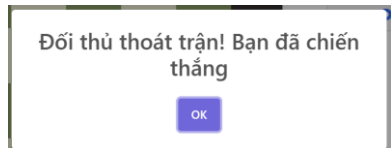
Màn hình	Giao diện chức năng chơi thể giới.
Mô tả	Hệ thống tự động tạo phòng dựa trên số người đang tham gia.
Nội dung màn hình	<ul style="list-style-type: none"> Sau 20 giây, nếu chưa phòng chưa được tạo sẽ quay về giao diện trang chủ.

Bảng 11: Mô tả chức năng chơi thể giới.



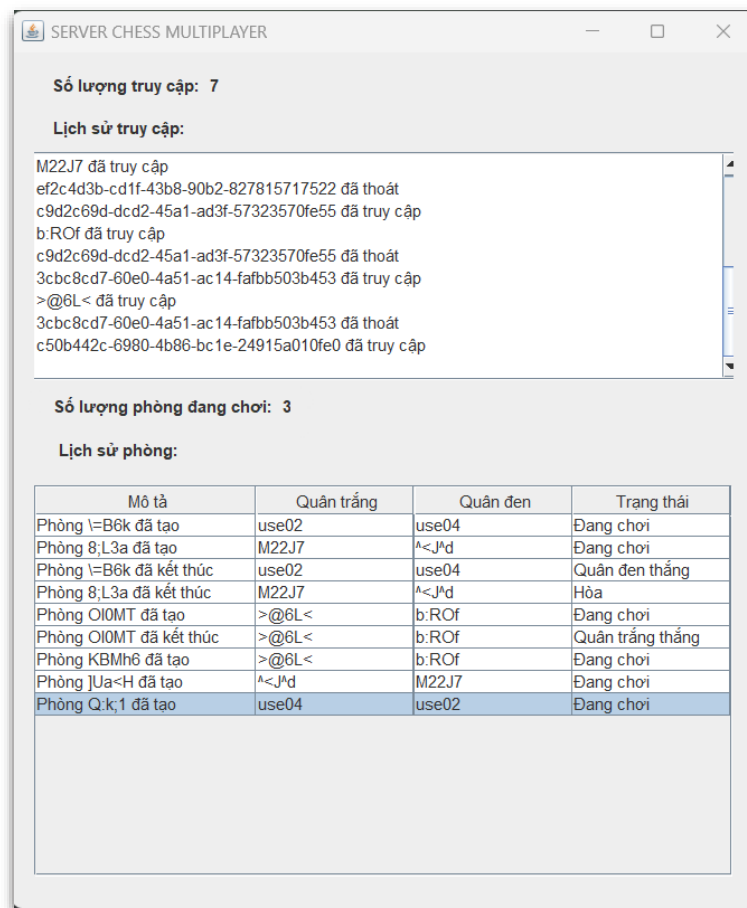
Hình 34: Giao diện khi trong trận.

PBL4: DỰ ÁN HỆ ĐIỀU HÀNH & MẠNG MÁY TÍNH

Màn hình	Giao diện trong trận đấu.		
Mô tả	Giao diện khi 2 đối thủ trong trận, cùng các chức năng kèm theo.		
Nội dung màn hình			
Mục	Kiểu	Ghi chú	Mô tả
Quân phong cấp	RadioButton	Khi người dùng kích chọn, quân tốt sẽ được phong cấp thành quân tương ứng (mặc định sẽ phong quân hậu).	Gồm các lựa chọn quân phong cấp khi quân tốt đi đến hàng cuối phía bên kia bàn cờ.
Cầu hòa	Button	Khi người dùng kích chọn, đối thủ sẽ nhận thông báo chấp nhận hoặc từ chối cầu hòa.	
Đầu hàng	Button	Khi người dùng kích chọn, đối thủ sẽ nhận thông báo chiến thắng do đầu hàng.	
Thêm bạn bè 	Button	Khi người dùng kích chọn, đối thủ sẽ nhận thông báo chấp nhận hoặc từ chối lời mời kết bạn.	
Trang cá nhân	Button		Hiển thị trang cá nhân (trận đấu vẫn tiếp tục).
Đăng xuất	Button	Đăng xuất tài khoản, đối thủ nhận được thông báo thắng trận.	

Bảng 12: Mô tả giao diện trong trận đấu.

3.2.5. Chức năng phía server



Hình 35: Giao diện server.

Màn hình	Giao diện server.	
Nội dung màn hình		
Mục	Kiểu	Mô tả
Số lượng truy cập	Text	Hiển thị số người dùng đang truy cập.
Lịch sử truy cập	Text	Hiển thị thông tin người dùng đang truy cập.
Số lượng phòng đang chơi	Text	Hiển thị số phòng đang tạo.
Lịch sử phòng	Table	Hiển thị thông tin trạng thái phòng.

Bảng 13: Mô tả giao diện server.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết luận

Trong quá trình nghiên cứu và thực hiện đồ án, chúng em đã tìm hiểu và nắm vững hơn các kiến thức liên quan về lập trình mạng, giao thức WebSocket truyền tin thời gian thực,... Ngoài ra nhóm chúng em cũng học hỏi, ứng dụng được nhiều công nghệ như: mã hóa password, Java Spring Boot, Typescript, sử dụng các thư viện, công cụ hỗ trợ thiết kế web (Bootstrap, SweetAlert2, Snowpack), ...

Chương trình đáp ứng được nhu cầu của đề tài, với các chức năng cơ bản, giao diện người dùng thân thiện, dễ sử dụng. Tuy nhiên, đề tài này vẫn còn mang tính chất học hỏi, trao đổi và bắt đầu làm quen với thực tế, nên thuật toán còn chưa được tối ưu.

2. Hướng phát triển

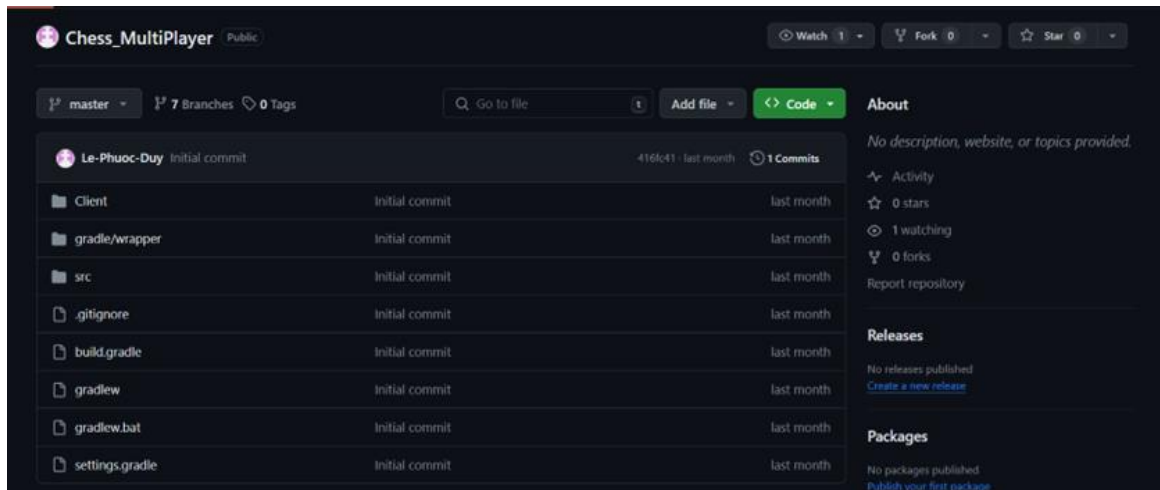
- Thêm các chức năng đánh với máy để đem đến những trải nghiệm tốt hơn cho người dùng muốn cải thiện khả năng chơi cờ vua.
- Mở rộng chức năng voice chat đối với người chơi đang trong trận.
- Tích hợp các chức năng đăng nhập nhanh bằng tài khoản Google, Facebook, Instagram,...
- Ứng dụng thêm các thuật toán tối ưu, rút ngắn thời gian khi chơi cờ, giảm thiểu khối lượng xử lý của Server để nâng cao khả năng đa luồng, đa nhiệm.

TÀI LIỆU THAM KHẢO

- [1]. TS. Lê Thị Mỹ Hạnh, Slide bài giảng “Phân tích thiết kế hướng đối tượng”.
- [2]. ThS. Mai Văn Hà, Slide bài giảng “Chương 3 Các giao thức cơ bản”.
- [3]. ThS. Mai Văn Hà, Slide bài giảng “Chương 4 Lập trình Websocket với TCP”.
- [4]. Hà Mạnh Đào, giáo trình “Lập trình mạng”.
- [5]. Jan Graba, giáo trình “An Introduction to Network Programming with Java”.
- [6]. Hướng dẫn lập trình Typescript <https://www.typescriptlang.org/docs/>
- [7]. STOMP documents:
<https://docs.spring.io/springframework/reference/web/websocket/stomp.html>

PHỤ LỤC A

- Source code: https://github.com/Le-Phuoc-Duy/Chess_MultiPlayer
- Quản lý mã nguồn :



Hình 36: Quản lý mã nguồn

PHỤ LỤC B

- **Chức năng đánh cờ :**

```
@MessageMapping("/chessMove")
public void chessMove(ChessGame message) {
    String AccId = accountService.getAccID(message.getUserName());
    String UserId = userService.getIdUserByIdAcc(AccId);
    String AccOppId =
accountService.getAccID(message.getUserReceiveName());
    String UserOppId = userService.getIdUserByIdAcc(AccOppId);
    int mode = roomService.getRoomById(message.getIdRoom()).getMode();
    System.out.println("mode: " + mode);
    ChessGame chessGameUserReceive = new ChessGame();
    chessGameUserReceive.setIDUserSend(UserOppId);
    chessGameUserReceive.setIDRoom(message.getIdRoom());
    chessGameUserReceive.setUserSendAva(message.getUserSendAva());
    message.setUserSendAva(userService.getUserById(UserOppId).getAva());
    if(getRoomuserByIdByRoomIdAndUserId(message.getIdRoom(),
UserOppId)!=null){

chessGameUserReceive.setIDRoomUser(getRoomuserByIdByRoomIdAndUserI
d(message.getIdRoom(), UserOppId));
    chessGameUserReceive.setChessMove(message.getChessMove());
    chessGameUserReceive.setBoard(reverseString(message.getBoard()));
    chessGameUserReceive.setColor(!message.getColor());

chessGameUserReceive.setUserReceiveName(userService.getUsernameByUs
erID(UserId));
    if(containsCountdownTimerWithIdUser(message.getIdRoomUser()) &&
containsCountdownTimerWithIdUser(chessGameUserReceive.getIdRoomUs
er())){
        switch (mode){
            case -1 -> {

stopCountdownTimerWithIdRoomUser(message.getIdRoomUser(), 2);
            }
            case -2 -> {

stopCountdownTimerWithIdRoomUser(message.getIdRoomUser(), 3);
            }
            default -> {

stopCountdownTimerWithIdRoomUser(message.getIdRoomUser(), 1);
            }
        }
    }
}
```

```
startCountdownTimerWithIdRoomUser(chessGameUserReceive.getIdRoomUser());
    }else{
        switch (mode){
            case -1 -> {
                createCountdownTimer(120, message.getIdRoomUser(),false,
chessGameUserReceive.getIdRoomUser(), message.getiDRoom(), UserId,
UserOppId);
                createCountdownTimer(120,
chessGameUserReceive.getIdRoomUser(),true, message.getIdRoomUser(),
message.getiDRoom(), UserOppId, UserId);
            }
            case -2 -> {
                createCountdownTimer(180, message.getIdRoomUser(),false,
chessGameUserReceive.getIdRoomUser(), message.getiDRoom(), UserId,
UserOppId);
                createCountdownTimer(180,
chessGameUserReceive.getIdRoomUser(),true, message.getIdRoomUser(),
message.getiDRoom(), UserOppId, UserId);
            }
            case -3 -> {
                createCountdownTimer(300, message.getIdRoomUser(),false,
chessGameUserReceive.getIdRoomUser(), message.getiDRoom(), UserId,
UserOppId);
                createCountdownTimer(300,
chessGameUserReceive.getIdRoomUser(),true, message.getIdRoomUser(),
message.getiDRoom(), UserOppId, UserId);
            }
            case -4 -> {
                createCountdownTimer(600, message.getIdRoomUser(),false,
chessGameUserReceive.getIdRoomUser(), message.getiDRoom(), UserId,
UserOppId);
                createCountdownTimer(600,
chessGameUserReceive.getIdRoomUser(),true, message.getIdRoomUser(),
message.getiDRoom(), UserOppId, UserId);
            }
            default ->{
                createCountdownTimer(mode, message.getIdRoomUser(),false,
chessGameUserReceive.getIdRoomUser(), message.getiDRoom(), UserId,
UserOppId);
                createCountdownTimer(mode,
chessGameUserReceive.getIdRoomUser(),true, message.getIdRoomUser(),
message.getiDRoom(), UserOppId, UserId);
            }
        }
    }
}
```



```
message.setUserCountdownValue(getCountdownTimerWithIdRoomUser(message.getIdRoomUser()));
```

```
message.setOppCountdownValue(getCountdownTimerWithIdRoomUser(chessGameUserReceive.getIdRoomUser()));
    System.out.println("countDown: " +
getCountdownTimerWithIdRoomUser(chessGameUserReceive.getIdRoomUser()));
```

```
chessGameUserReceive.setUserCountdownValue(getCountdownTimerWithIdRoomUser(chessGameUserReceive.getIdRoomUser()));
```

```
chessGameUserReceive.setOppCountdownValue(getCountdownTimerWithIdRoomUser(message.getIdRoomUser()));
```

```
    messagingTemplate.convertAndSendToUser(UserId,
"/queue/chessMoveSuccess",message);
    messagingTemplate.convertAndSendToUser(UserOppId,
"/queue/chessMove",chessGameUserReceive );
    }else{
        messagingTemplate.convertAndSendToUser(UserId,
"/queue/chessMove", null);
    }
}
```

- **Chức năng trò chuyện:**

```
@MessageMapping("/chatRoom")
public void chatRoom(ChatRoom message) {
```

```
    roomuserService.updateChatById(message.getIdRoomUser(),message.getChat());
```

```
    ChatRoom chatRoomUserReceive = new ChatRoom();
```

```
    String AccOppId =
```

```
accountService.getAccID(message.getUserReceiveName());
```

```
    String UserOppId = userService.getIdUserByIdAcc(AccOppId);
```

```
    chatRoomUserReceive.setIdUserSend(UserOppId);//ban than
```

```
chatRoomUserReceive.setUserSendName(userService.getUsernameByUserID(message.getIdUserSend()));
```

```
chatRoomUserReceive.setUserSendAva(userService.getUserById(message.getIdUserSend()).getAva());
```

```
    chatRoomUserReceive.setIdRoom(message.getIdRoom());
```

```
chatRoomUserReceive.setUserReceiveName(userService.getUsernameByUserID(message.getIdUserSend()));//doi thu
```

```
        if(getRoomuserIdByRoomIdAndUserId(message.getIdRoom(),
        UserOppId)!=null){

        chatRoomUserReceive.setIdRoomUser(getRoomuserIdByRoomIdAndUserId
        (message.getIdRoom(), UserOppId));
        chatRoomUserReceive.setChat(message.getChat());
        messagingTemplate.convertAndSendToUser(UserOppId,
        "/queue/chatRoom",chatRoomUserReceive );
        }else{
        messagingTemplate.convertAndSendToUser(UserOppId,
        "/queue/chatRoom", null);
        }
    }
}

@MessageMapping("/publicChat")
@SendTo("/topic/publicChat")
public publicChat publicChat(publicChat message) {
    if(userService.getUserById(message.getIdDUserSend())!=null){

        message.setUserSendName(userService.getUsernameByUserID(message.getIdDUs
        erSend()));

        message.setAva(userService.getUserById(message.getIdDUserSend()).getAva());
        return message;

    }else{
        return null;
    }
}
```