**ĐẠI HỌC ĐÀ NẴNG**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**

A blue and white logo with arrows

Description automatically generated

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

**Đề tài: XÂY DỰNG WEBSITE TRÒ CHƠI TIẾN LÊN**

**THEO MÔ HÌNH CLIENT-SERVER**

**SINH VIÊN THỰC HIỆN:**

**Ngô Nguyễn Tấn Quân 102220033 LỚP: 22T\_KHDL NHÓM: 22.12B**

**Lưu Duy Quang 102220036 LỚP: 22T\_KHDL NHÓM: 22.12B**

**Lê Ngọc Thanh 102220041 LỚP: 22T\_KHDL NHÓM: 22.12B**

**GIẢNG VIÊN HƯỚNG DẪN: TS. Nguyễn Thị Lệ Quyên**

**Đà Nẵng 25/12/ 2024**

MỤC LỤC

[CHƯƠNG 1. CƠ SỞ LÝ THUYẾT 9](#_Toc186036421)

[1.1.Mô hình TCP/IP 9](#_Toc186036422)

* + 1. [Tầng liên kết 9](#_Toc186036423)
    2. [Tầng mạng (Internet) 9](#_Toc186036424)

[1.1.3. Tầng giao vận 9](#_Toc186036425)

[1.1.4. Tầng ứng dụng 9](#_Toc186036426)

[1.2. Mô hình Client-Server: 10](#_Toc186036427)

[1.3. Các giao thức mạng 11](#_Toc186036428)

[1.3.1. IP 11](#_Toc186036429)

[1.3.2. HTTP/HTTPS 12](#_Toc186036430)

[1.4. Socket.io 13](#_Toc186036431)

[1.5. ReactJS 14](#_Toc186036432)

[1.6. ExpressJS 15](#_Toc186036433)

[CHƯƠNG 2. THIẾT KẾ VÀ XÂY DỰNG HỆ THỐNG 16](#_Toc186036434)

[2.1. Quá trình triển khai 16](#_Toc186036435)

[1 . Quá trình triển khai và phân công nhiệm vụ 16](#_Toc186036436)

[2 . Quản lý dự án 18](#_Toc186036437)

[3. Công cụ thực hiện 19](#_Toc186036438)

[2.2. Phân tích và thiết kế hệ thống 19](#_Toc186036439)

[1. Yêu cầu chức năng 19](#_Toc186036440)

[2.Yêu cầu phi chức năng 22](#_Toc186036441)

[3. Kiến trúc hệ thống 22](#_Toc186036442)

[4. Sơ đồ Usecase tổng quan 23](#_Toc186036443)

[5. Sơ đồ triển khai cơ sở dữ liệu 23](#_Toc186036444)

[CHƯƠNG 3. DEMO ỨNG DỤNG VÀ ĐÁNH GIÁ KẾT QUẢ 24](#_Toc186036445)

[3.1. Giao diện chính của trò chơi 24](#_Toc186036446)

[3.2. Giao diện đăng nhập 24](#_Toc186036447)

[3.3. Giao diện cập nhật thông tin người chơi 25](#_Toc186036448)

[3.4. Giao diện khởi tạo trò chơi 25](#_Toc186036449)

[3.4.1. Giao diện tạo phòng 25](#_Toc186036450)

[3.4.2. Giao diện tham gia phòng đã có sẵn 26](#_Toc186036451)

[3.4.3. Giao diện hàng đợi 26](#_Toc186036452)

[3.4.4. Giao diện mời người chơi khác 27](#_Toc186036453)

[3.5. Giao diện trong ván đấu 28](#_Toc186036454)

[3.6. Giao diện chat 29](#_Toc186036455)

[3.7. Giao diện tổng quan admin dashboard 29](#_Toc186036456)

DANH SÁCH HÌNH VẼ

[Hình 1. Mô hình TCP/IP 8](#_26in1rg)

[Hình 2. Mô hình Client/Server 8](#_1ksv4uv)

[Hình 3. Cấu trúc gói tin IP 9](#_z337ya)

[Hình 4: Cơ chế hoạt động HTTP 9](#_1y810tw)

[Hình 5: Cơ chế hoạt động HTTPS 10](#_4i7ojhp)

DANH SÁCH BẢNG BIỂU

[Bảng 1. Bảng sự kiện socket 13](#_Toc186037468)

[Bảng 2. Bảng so sánh phương thức emit và on 14](#_Toc186037469)

[Bảng 3. Bảng phân công công việc 17](#_Toc186037470)

[Bảng 4. Bảng mô tả chức năng 22](#_Toc186037471)

DANH SÁCH CÁC TỪ VIẾT TẮT

No table of abbreviation entries found.

MỞ ĐẦU (GIỚI THIỆU ĐỀ TÀI)

Trong thời đại công nghệ số bùng nổ, trò chơi trực tuyến đã trở thành một phần không thể thiếu trong cuộc sống của chúng ta. Sự kết hợp hoàn hảo giữa công nghệ web và niềm đam mê game đã mở ra một thế giới giải trí vô cùng phong phú và sáng tạo. Nhận thấy tiềm năng to lớn này, nhóm chúng tôi đã quyết định xây dựng một website trò chơi Tiến Lên theo mô hình client-server, nhằm mục tiêu tạo ra một sản phẩm không chỉ giải trí mà còn là cơ hội để rèn luyện kỹ năng lập trình.

Tiến Lên, một trò chơi chiến thuật kinh điển, đã chinh phục hàng triệu người chơi trên toàn thế giới bởi luật chơi đơn giản nhưng lại vô cùng hấp dẫn. Việc đưa Tiến lên lên nền tảng web không chỉ giúp trò chơi tiếp cận được với đông đảo người chơi hơn mà còn là cơ hội để chúng tôi ứng dụng những công nghệ web hiện đại nhất. Qua dự án này, nhóm đã có dịp nghiên cứu sâu hơn về các nguyên tắc thiết kế giao diện người dùng (UI), xây dựng logic trò chơi phức tạp và quản lý dữ liệu trên server.

Chúng tôi lựa chọn mô hình client-server nhờ tính hiệu quả và khả năng mở rộng, giúp tách biệt giao diện người dùng và xử lý dữ liệu, tối ưu hiệu năng và mang lại trải nghiệm mượt mà. Nhóm sử dụng các công nghệ như JavaScript, Node.js, React và MySQL để phát triển hệ thống.

Giao diện được thiết kế trực quan, thân thiện và tăng tính hấp dẫn nhờ hiệu ứng động từ các thư viện JavaScript hiện đại. Về logic trò chơi, nhóm giải quyết các bài toán như xác định giá trị bài, kiểm tra điều kiện để được phép đánh lá bài, kiểm tra các trường hợp đôi, sảnh,… và đồng bộ hóa dữ liệu thời gian thực bằng cách áp dụng các thuật toán tìm kiếm và sắp xếp tối ưu.

Thách thức lớn nhất là đồng bộ dữ liệu giữa người chơi, nhưng nhờ sử dụng WebSocket và tối ưu thuật toán, nhóm đã đảm bảo trò chơi hoạt động mượt mà và không gián đoạn.

Sau nhiều tháng làm việc không ngừng nghỉ, chúng tôi đã hoàn thiện sản phẩm với các tính năng chính như: đăng ký và đăng nhập, tạo phòng chơi, tham gia phòng chơi và đồng bộ hóa dữ liệu thời gian thực như các hành động của người chơi được cập nhật ngay lập tức cho tất cả những người tham gia trong phòng.

Cuối cùng, chúng tôi xin gửi lời cảm ơn sâu sắc đến cô Lệ Quyên, người đã luôn đồng hành và hỗ trợ chúng tôi trong suốt quá trình thực hiện dự án. Sự hướng dẫn tận tình của cô đã giúp chúng tôi vượt qua nhiều khó khăn và hoàn thành dự án một cách tốt đẹp.

# CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

## Mô hình TCP/IP

Mô hình TCP/IP, hay còn được gọi là mô hình Internet, là một khung lý thuyết cơ bản cho việc thiết kế và triển khai mạng máy tính. Được tạo ra bởi DARPA (Defense Advanced Research Projects Agency) cho dự án ARPANET, mô hình này đã trở thành nền tảng cho Internet và nhiều mạng hiện đại. Mô hình TCP/IP chia quá trình truyền thông thành một số lớp, mỗi lớp đóng vai trò cụ thể trong quá trình này.

### Tầng liên kết

Đây là lớp thấp nhất trong mô hình, quản lý gửi và nhận dữ liệu giữa các thiết bị trực tiếp kết nối với nhau trên một mạng cục bộ. Nó thường bao gồm các giao thức như Ethernet và Wifi.

### Tầng mạng (Internet)

Lớp này quản lý việc định tuyến dữ liệu từ nguồn đến đích trên mạng. Giao thức quan trọng ở đây là IP (Internet Protocol), chịu trách nhiệm đặt địa chỉ IP cho thiết bị và chuyển gói tin từ một nút đến nút khác.

### Tầng giao vận

Lớp này quản lý kết nối giữa hai thiết bị truyền tin qua mạng. Giao thức chính ở đây bao gồm TCP (Transmission Control Protocol) và UDP (User Datagram Protocol), với TCP đảm bảo truyền tải tin cậy và UDP tập trung vào truyền tải nhanh chóng.

### Tầng ứng dụng

Lớp cuối cùng chịu trách nhiệm cho việc cung cấp các dịch vụ truy cập mạng cho người dùng. Giao thức ở đây bao gồm HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol) và SMTP (Simple Mail Transfer Protocol).

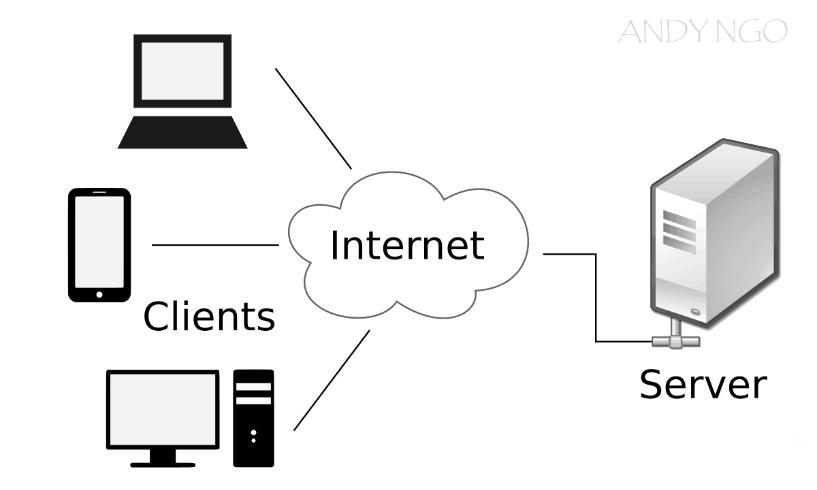
A diagram of a computer model

Description automatically generated with medium confidence

*Hình 1. Mô hình TCP/IP*

## Mô hình Client-Server:

Mô hình Client Server là mô hình mạng máy tính trong đó các máy tính con được đóng vai trò như một máy khách, chúng làm nhiệm vụ gửi yêu cầu đến các máy chủ. Để máy chủ xử lý yêu cầu và trả kết quả về cho máy khách đó.



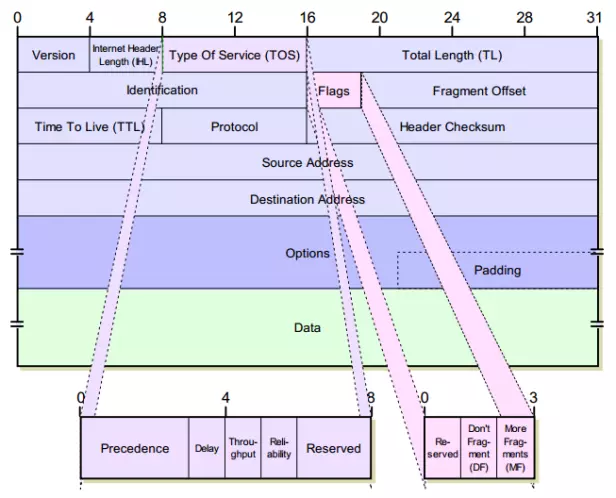
*Hình 2. Mô hình Client/Server*

**Client:** Với vai trò là máy khách, chúng sẽ không cung cấp tài nguyên đến các máy tính khác mà chỉ sử dụng tài nguyên được cung cấp từ máy chủ. Một client trong mô hình này có thể là một server cho mô hình khác, tùy thuộc vào nhu cầu sử dụng của người dùng.

**Server:** Là máy tính có khả năng cung cấp tài nguyên và các dịch vụ đến các máy khách khác trong hệ thống mạng. Server đóng vai trò hỗ trợ cho các hoạt động trên máy khách client diễn ra hiệu quả hơn.

## Các giao thức mạng

### IP



*Hình 3. Cấu trúc gói tin IP*

Internet Protocol là - Giao thức Internet (IP) là một giao thức hoặc bộ quy tắc để định tuyến và xử lý các gói dữ liệu để chúng có thể di chuyển qua các mạng máy tính và đến đích chính xác. Dữ liệu truyền qua Internet được chia thành các phần nhỏ hơn, được gọi là các gói. Thông tin IP được đính kèm vào mỗi gói và thông tin này giúp các bộ định tuyến gửi các gói đến đúng nơi. Mọi thiết bị hoặc miền kết nối với Internet đều được gán một địa chỉ IP - Internet Protocol và khi các gói được hướng đến địa chỉ IP - Internet Protocol được đính kèm với chúng, dữ liệu sẽ đến nơi cần thiết.

### HTTP/HTTPS

A computer and a phone

Description automatically generated with medium confidence

*Hình 4: Cơ chế hoạt động HTTP*

HTTP (Hypertext Transfer Protocol) là giao thức truyền tải siêu văn bản. Đây là giao thức tiêu chuẩn cho World Wide Web (www) để truyền tải dữ liệu dưới dạng văn bản, âm thanh, hình ảnh, video từ Web Server tới trình duyệt web của người dùng và ngược lại.

A computer screen shot of a computer screen

Description automatically generated

*Hình 5: Cơ chế hoạt động HTTPS*

HTTPS (Hypertext Transfer Protocol Secure) là giao thức truyền tải siêu văn bản an toàn. Thực chất, đây chính là giao thức HTTP nhưng tích hợp thêm Chứng chỉ bảo mật SSL nhằm mã hóa các thông điệp giao tiếp để tăng tính bảo mật. Có thể hiểu, HTTPS là phiên bản HTTP an toàn, bảo mật hơn. HTTPS hoạt động tương tự như HTTP, tuy nhiên được bổ sung thêm chứng chỉ SSL (Secure Sockets Layer – tầng ổ bảo mật) hoặc TLS (Transport Layer Security – bảo mật tầng truyền tải). Hiện tại, đây là các tiêu chuẩn bảo mật hàng đầu cho hàng triệu website trên toàn thế giới.

## Socket.io

Socket.IO là một thư viện JavaScript cho phép giao tiếp realtime, hai chiều giữa web clients và server. Nó bao gồm:

* Server-side library cho Node.js
* Client-side library cho browser

**Đặc điểm chính:**

* Hỗ trợ real time bidirectional communication
* Tự động reconnect khi mất kết nối
* Hỗ trợ binary streaming
* Multiplexing (namespaces và rooms)
* Automatic fallback từ WebSocket sang HTTP long-polling

**Các sự kiện chính trong Socket.io:**

|  |  |
| --- | --- |
| Sự kiện | Mô tả |
| connect | Xảy ra khi client kết nối thành công với server. |
| disconnect | |  | | --- | | Xảy ra khi client ngắt kết nối khỏi server. | |
| connect\_error | |  | | --- | | Xảy ra khi có lỗi trong quá trình kết nối (timeout, server không phản hồi, v.v.). | |
| message | |  | | --- | | Sự kiện mặc định để gửi và nhận tin nhắn giữa client và server. | |
| custom\_event | |  | | --- | | Sự kiện tùy chỉnh do lập trình viên định nghĩa, dùng để trao đổi thông tin cụ thể. | |
| disconnecting | |  | | --- | | Xảy ra ngay trước khi client ngắt kết nối (ít dùng hơn disconect). | |
| reconnect | |  | | --- | | Xảy ra khi client tự động kết nối lại sau khi ngắt kết nối. | |
| reconnect\_error | Xảy ra khi quá trình kết nối lại gặp lỗi. |

**Bảng 1 . Bảng sự kiện socket**

**Hai phương thức quan trọng nhất trong Socket.io**

1. **Phương thức emit**

**Định nghĩa:** Phương thức emit được sử dụng để phát (gửi) một sự kiện cùng dữ liệu đính kèm đến client hoặc server.

**Cú pháp:** socket.emit(eventName, data, callback);

**Thành phần:**

* eventName: Tên của sự kiện (kiểu chuỗi). Đây là cách định danh sự kiện để bên nhận xử lý đúng nội dung.
* data (tùy chọn): Dữ liệu đính kèm (có thể là chuỗi, đối tượng, hoặc bất kỳ kiểu dữ liệu hợp lệ nào).
* callback (tùy chọn): Hàm callback được gọi sau khi server hoặc client xử lý sự kiện (phản hồi).

1. **Phương thức on**

**Định nghĩa:** Phương thức on được sử dụng để lắng nghe và xử lý sự kiện mà client hoặc server phát.

**Cú pháp:** socket.on(eventName, callback);

**Thành phần:**

* eventName: Tên của sự kiện mà bạn muốn lắng nghe.
* callback: Hàm được gọi khi sự kiện được phát, nhận tham số là dữ liệu từ sự kiện.

|  |  |  |
| --- | --- | --- |
| Phương thức | Mục đích | Hành động |
| emit | Phát sự kiện | Gửi sự kiện và dữ liệu đến server hoặc client |
| on | Lắng nghe và xử lý sự kiện | Nhận sự kiện được gửi từ server hoặc client |

**Bảng 2. Bảng so sánh phương thức emit và on**

## 1.5. ReactJS

React (hay còn được gọi là React.js hoặc ReactJS) là một thư viện JavaScript front-end mã nguồn mở và miễn phí để xây dựng giao diện người dùng dựa trên các thành phần UI riêng lẻ. Nó được phát triển và duy trì bởi Meta (trước đây là Facebook) và cộng đồng các nhà phát triển và công ty cá nhân. React có thể được sử dụng làm cơ sở để phát triển các ứng dụng SPA (Single page application), thiết bị di động hoặc ứng dụng được kết xuất bằng máy chủ với các thư viện khác như Next.js. Tuy nhiên, React chỉ hướng tới việc quản lý trạng thái và hiển thị trạng thái đó cho DOM, vì vậy việc tạo ứng dụng bằng React thường yêu cầu sử dụng thêm các thư viện bổ sung để thực hiện định tuyến trang, cũng như thêm một số chức năng ở phía máy khách.

ReactJS là một thư viện JavaScript chuyên giúp các nhà phát triển xây dựng giao diện người dùng hay UI. Trong lập trình ứng dụng front-end, lập trình viên thường sẽ phải làm việc chính trên 2 thành phần sau: UI và xử lý tương tác của người dùng. UI là tập hợp những thành phần mà bạn nhìn thấy được trên bất kỳ một ứng dụng nào, ví dụ có thể kể đến bao gồm: menu, thanh tìm kiếm, những nút nhấn, card,… Giả sử bạn đang lập trình một website thương mại điện tử, sau khi người dùng chọn được sản phẩm ưng ý rồi và nhấn vào nút “Thêm vào giỏ hàng”, thì việc tiếp theo mà bạn phải làm đó là thêm sản phẩm được chọn vào giỏ hàng và hiển thị lại sản phẩm đó khi user vào xem => xử lý tương tác.Trước khi có ReactJS, lập trình viên thường gặp rất nhiều khó khăn trong việc sử dụng “vanilla JavaScript”(JavaScript thuần) và JQuery để xây dựng UI. Điều đó đồng nghĩa với việc quá trình phát triển ứng dụng sẽ lâu hơn và xuất hiện nhiều bug, rủi ro hơn. Vì vậy vào năm 2011, Jordan Walke – một nhân viên của Facebook đã khởi tạo ReactJS với mục đích chính là cải thiện quá trình phát triển UI.Hơn nữa, để tăng tốc quá trình phát triển và giảm thiểu những rủi ro có thể xảy ra trong khi coding, React còn cung cấp cho chúng ta khả năng Reusable Code (tái sử dụng code) bằng cách đưa ra 2 khái niệm quan trọng bao gồm:

* JSX.
* Virtual DOM.

## 1.6. ExpressJS

* Expressjs hay còn được viết là Express js, Express.js. Đây là một framework mã nguồn mở miễn phí cho Node.js. Express.js được sử dụng trong thiết kế và xây dựng các ứng dụng web một cách đơn giản và nhanh chóng.
* Vì Express js chỉ yêu cầu ngôn ngữ lập trình Javascript nên việc xây dựng các ứng dụng web và API trở nên đơn giản hơn với các lập trình viên và nhà phát triển.Expressjs cũng là một khuôn khổ của Node.js do đó hầu hết các mã code đã được viết sẵn cho các lập trình viên có thể làm việc.
* Nhờ có Expressjs mà các nhà lập trình có thể dễ dàng tạo các ứng dụng 1 web, nhiều web hoặc kết hợp. Do có dung lượng khá nhẹ, Expressjs giúp cho việc tổ chức các ứng dụng web thành một kiến trúc MVC có tổ chức hơn.Để có thể sử dụng được mã nguồn này, chúng ta cần phải biết về Javascript và HTML.
* Expressjs cũng là một phần của công nghệ giúp quản lý các ứng dụng web một cách dễ dàng hơn hay còn được gọi là ngăn xếp phần mềm MEAN.Nhờ có thư viện Javascript của Express js đã giúp cho các nhà lập trình xây dựng nên các ứng dụng web hiệu quả và nhanh chóng hơn. Expressjs cũng được sử dụng để nâng cao các chức năng của Node.js.
* Trên thực tế, nếu không sử dụng Express.js, bạn sẽ phải thực hiện rất nhiều bước lập trình phức tạp để xây dựng nên một API hiệu quả. Express js đã giúp cho việc lập trình trong Node.js trở nên dễ dàng hơn và có nhiều tính năng mới bổ sung.

# CHƯƠNG 2. THIẾT KẾ VÀ XÂY DỰNG HỆ THỐNG

## 2.1. Quá trình triển khai

### 1 . Quá trình triển khai và phân công nhiệm vụ

|  |  |  |  |
| --- | --- | --- | --- |
| **Tuần** | **Nội dung công việc** | **Sản phẩm** | **Người thực hiện** |
| **1** | Khảo sát đề tài |  | Cả nhóm |
| **2 - 3** | Chọn đề tài, đưa ra kế hoạch triển khai, phác thảo tổng quan đề tài | Kế hoạch triển khai, phân tích đề tài | Cả nhóm |
| **4 - 5** | Mô tả các chức năng cụ thể của hệ thống | Mô tả chức năng hệ thống | Thanh, Quang |
| Thảo luận và mô tả tổ chức CSDL | Sơ đồ Usecase, CSDL | Quân |
| Tạo cơ sở dữ liệu | File SQL | Quân |
| **6** | Thiết kế giao diện chương trình | Giao diện thông qua Figma | Thanh |
| **7 - 14** | Lập trình ứng dụng | File code | Cả nhóm |
| Kiểm thử chương trình |  | Cả nhóm |
| **15** | Viết báo cáo | Báo cáo | Cả nhóm |

**Bảng 3. Bảng phân công công việc**

### 2 . Quản lý dự án

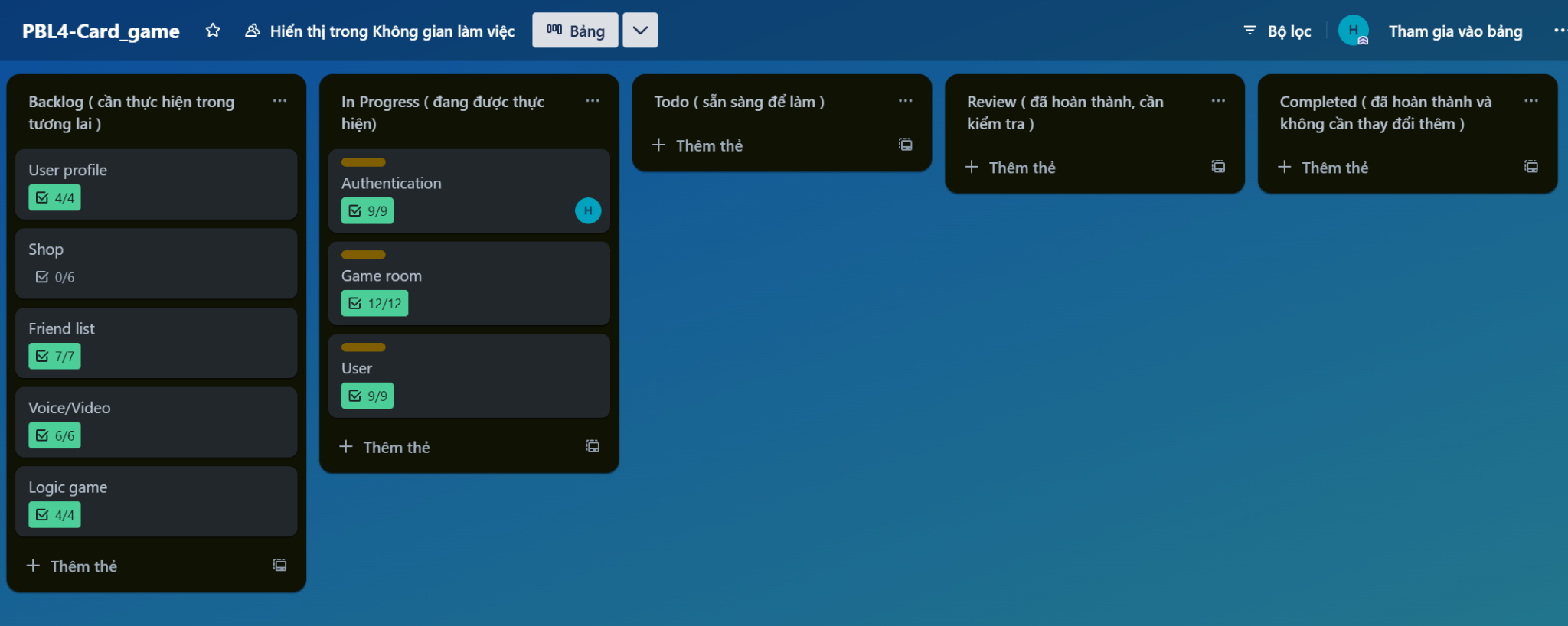
#### a. Sử dụng Git, Trello

* Link git dự án**:**

<https://github.com/Canuc80k/CardgameServer>

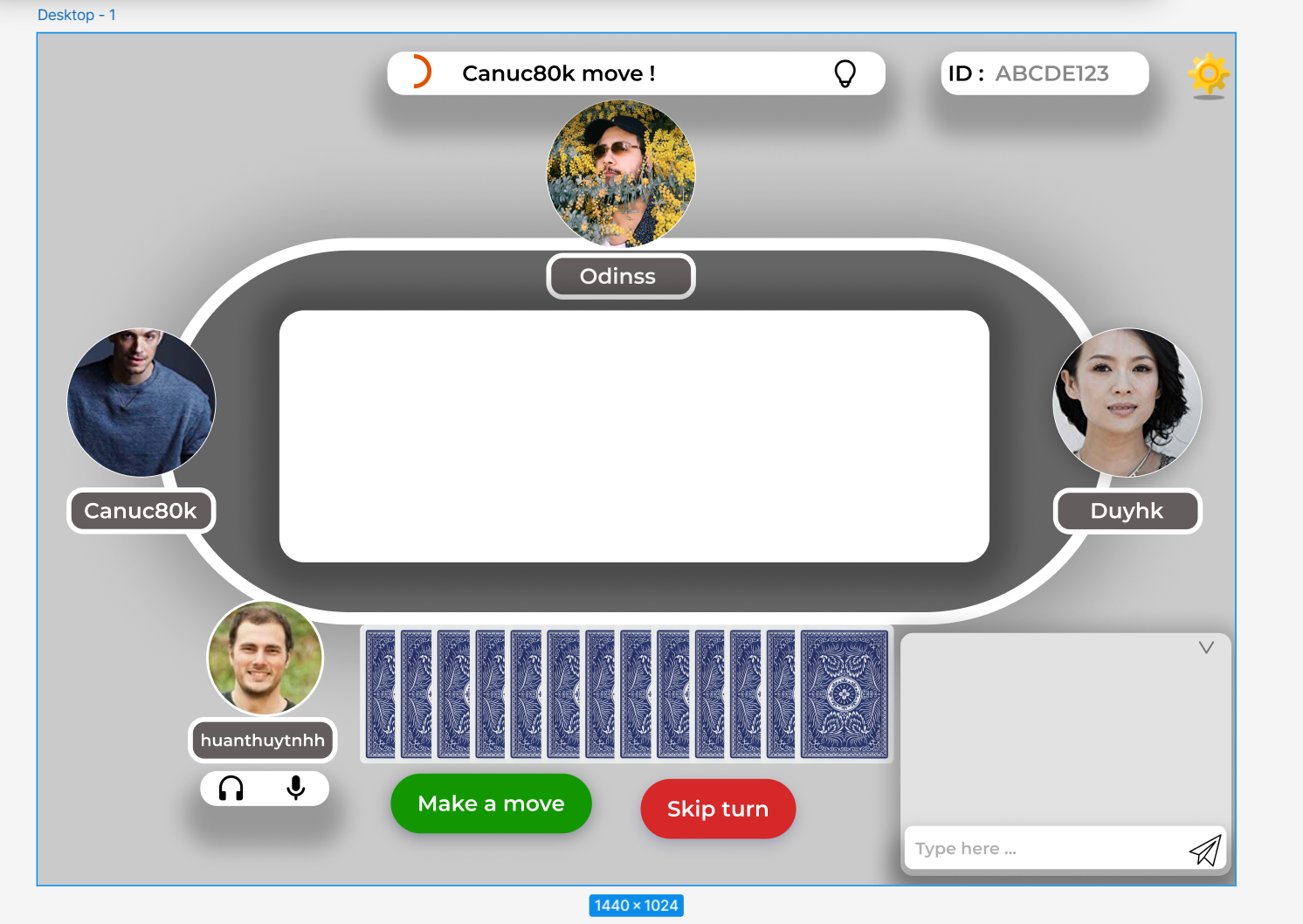
<https://github.com/Canuc80k/CardgameClient>

* Quản lý tiến độ bằng Trello

****

Hình 6: Tiến độ Trello

#### b. Thiết kế giao diện bằng Figma

****

Hình 7 : Giao diện bằng Figma

### 3. Công cụ thực hiện

**- Frontend:** Reactjs, Tailwind CSS, Shadcn UI

**- Backend:** Expressjs, Nodejs

- **Authentication**: Clerk

**- Kết nối tương tác của người dùng:** socket.io

- **Thực hiện video call**: Peerjs

**- Quản lý cơ sở dữ liệu:** MySQL

**- Quản lý tiến độ:** Trello, Notion

**- Quản lý mã nguồn:** Github

**- Thiết kế giao diện:** Figma

**- Trao đổi thông tin:** Discord, Messenger, …

## 2.2. Phân tích và thiết kế hệ thống

### 1. Yêu cầu chức năng

|  |  |
| --- | --- |
| **CHỨC NĂNG** | **MÔ TẢ** |
| **Đăng nhập và đăng ký** | Clerk authentication |
| **Chỉnh sửa thông tin, ảnh đại diện** | Clerk authentication |
| **Tạo phòng game** | Chức năng tạo phòng trong database và sau đó tự động join room |
| **Tham gia phòng game** | Chức năng gửi sự kiện socketio join room để tham gia phòng |
| **Chat real time** | Trò chuyện giữa các người dùng cùng phòng trong thời gian thực |
| **Gọi video real time** | Người chơi trong trận có thể gọi video real time cho nhau |
| **Xóa phòng game** | Khi không còn nhu cầu sử dụng sẽ xóa phòng |
| **Mời người chơi** | Thông qua socket io để gửi lời mời real time đến các người chơi đang online |
| **Phản hồi lời mời** | Thong qua socket io phản hồi lời mời để tham gia phòng hoặc từ chối tham gia |
| **Bắt đầu game** | Kiểm tra xem số lượng người trong phòng có đủ để bắt đầu game chưa , sau đó sẽ thực hiện chức năng chia bài |
| **Chia bài** | Thông qua Socket Io gửi sự kiện deal cards đến server để bắt đầu thực hiện xáo và chia bài cho người chơi sau đó thông báo cho người chơi qua sự kiện cards dealt |
| **Đánh bài** | Người chơi thông qua socket io để gửi sự kiện đánh bài play\_cards đến server và thống báo đến các player còn lại qua cards\_played |
| **Bỏ lượt** | Người chơi thông qua socket io để gửi sự kiện bỏ lượt pass\_turn đến server và thống báo đến các player còn lại qua turn\_passed |
| **Rời phòng** | Khi một người chơi rời phòng hoặc mất kết nối sẽ thông báo đến các người chơi còn lại bắt đầu lại game |
| **Kết thúc game** | Khi còn một người chơi trong phòng có số lá bài trên tay khác không thì sẽ dừng trò chơi và thông báo bảng điểm cho người chơi trong phòng |
| **Reset game** | Khởi tạo lại các dữ liệu của game đấu trước để bắt đầu một game mới |
| **Quản lý người chơi** | Admin hệ thống có thể quản lý người chơi , nâng cấp quyền cho một user thông thường lên làm admin |
| **Quản lý phòng game** | Admin hệ thống quản lý phòng chơi game |
| **Xem báo cáo** | Xem báo cáo ,thống kê trong admin dashboard |
| **Đăng xuất** | Đăng xuất tài khoản khỏi website |

**Bảng 4. Bảng mô tả chức năng**

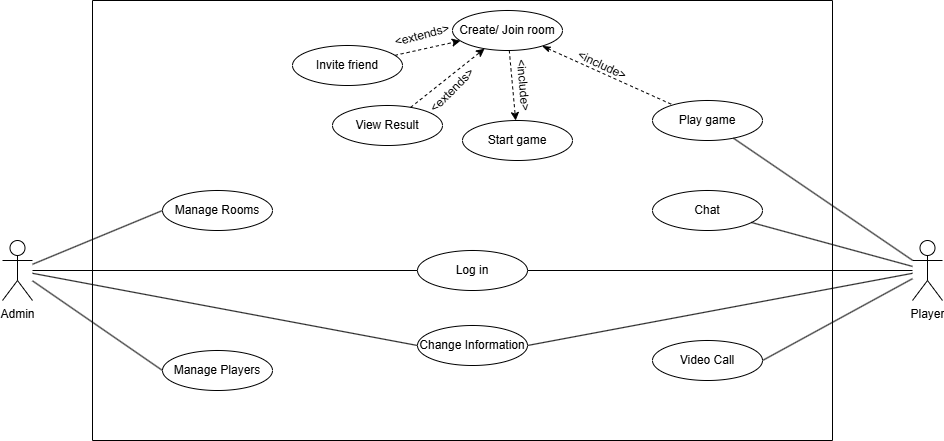
### Yêu cầu phi chức năng

* **Hiệu suất:** Đảm bảo độ trễ thấp để trải nghiệm mượt mà.
* **Bảo mật:**
  + Mã hóa dữ liệu truyền qua mạng.
  + Xác thực người chơi

### Kiến trúc hệ thống

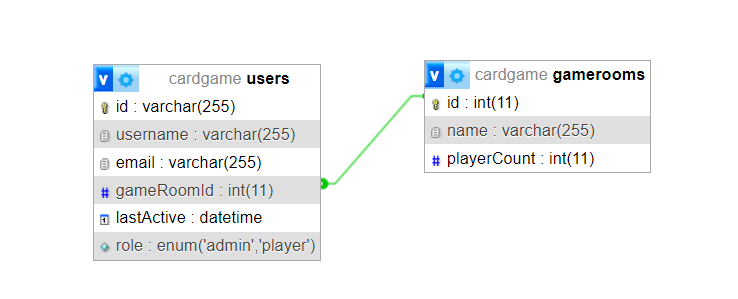
* **Mô hình client-server:**
  + **Client:** Được triển khai trên máy người chơi.
  + **Server:** Được triển khai trên một máy chủ trung tâm hoặc máy tính đóng vai trò máy chủ.
* **Giao thức truyền thông:**
  + TCP/IP hoặc WebSocket để đảm bảo kết nối liên tục.
* **Luồng hoạt động:**
  + Client gửi yêu cầu đến server.
  + Server xử lý logic và phản hồi kết quả.
  + Client nhận phản hồi và cập nhật giao diện người chơi.

### Sơ đồ Usecase tổng quan

****

Hình 7 : Sơ đồ Usecase

### Sơ đồ triển khai cơ sở dữ liệu

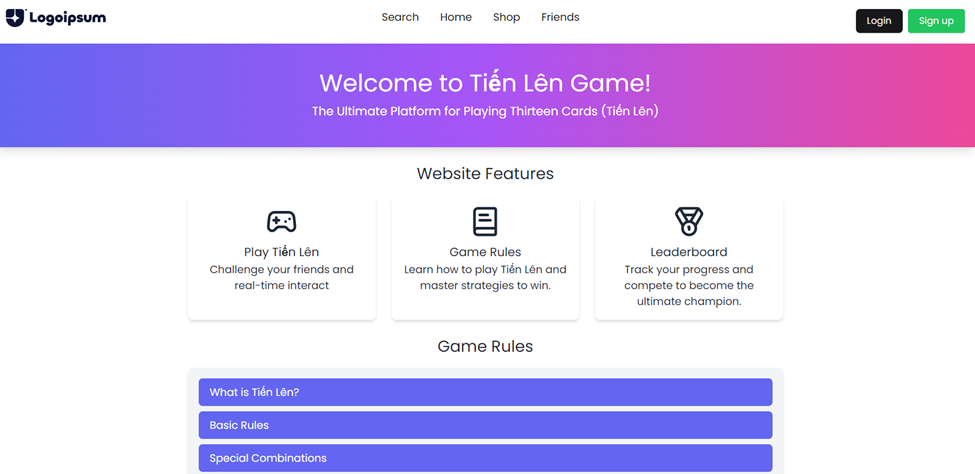


Hình 8: Quan hệ cơ sở dữ liệu

# 

# CHƯƠNG 3. DEMO ỨNG DỤNG VÀ ĐÁNH GIÁ KẾT QUẢ

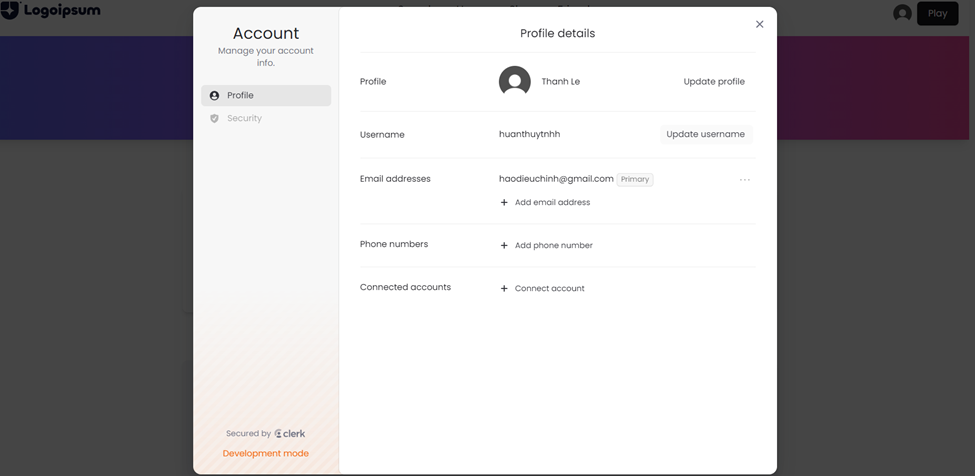
## 3.1. Giao diện chính của trò chơi



## 3.2. Giao diện đăng nhập

****

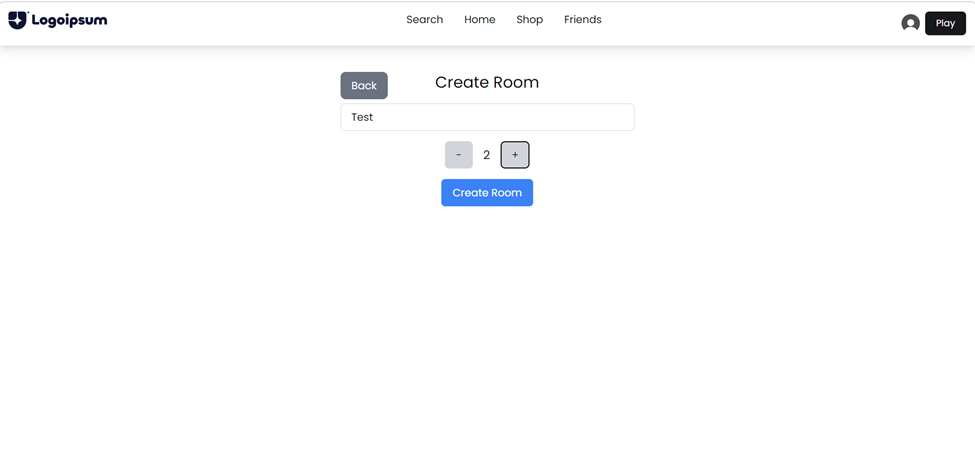
## 3.3. Giao diện cập nhật thông tin người chơi

****

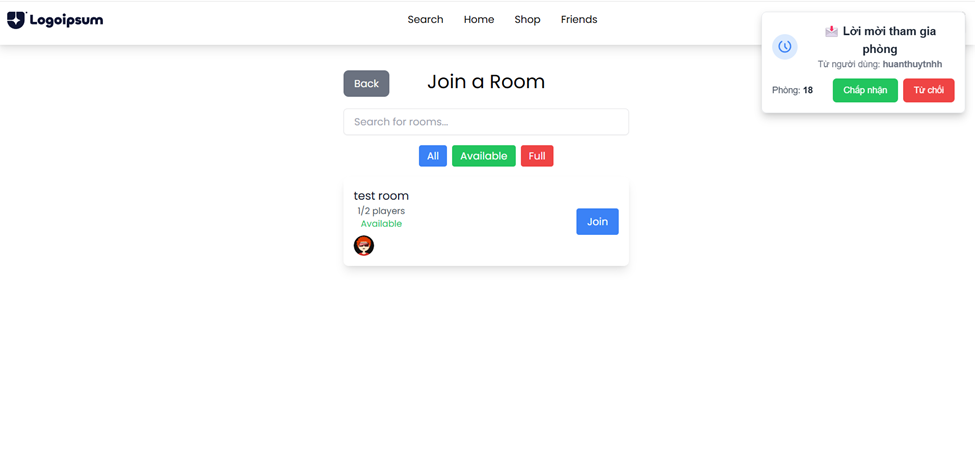
## 3.4. Giao diện khởi tạo trò chơi

* Sau khi nhấn vào nút Play ở góc trên bên phải màn hình, người chơi sẽ chọn ‘Create Room’ để tạo một phòng mới để chơi, hoặc chọn ‘Join Room’ nếu đã có bạn bè tạo sẵn phòng.

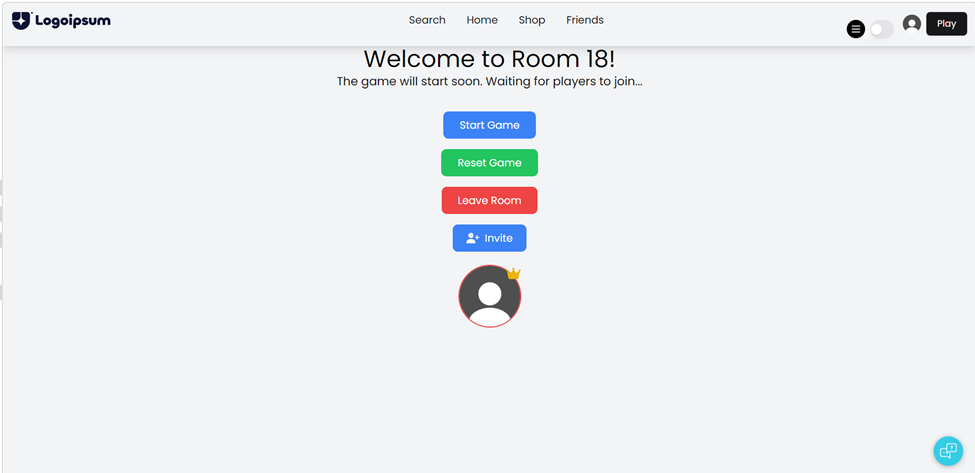
### 3.4.1. Giao diện tạo phòng

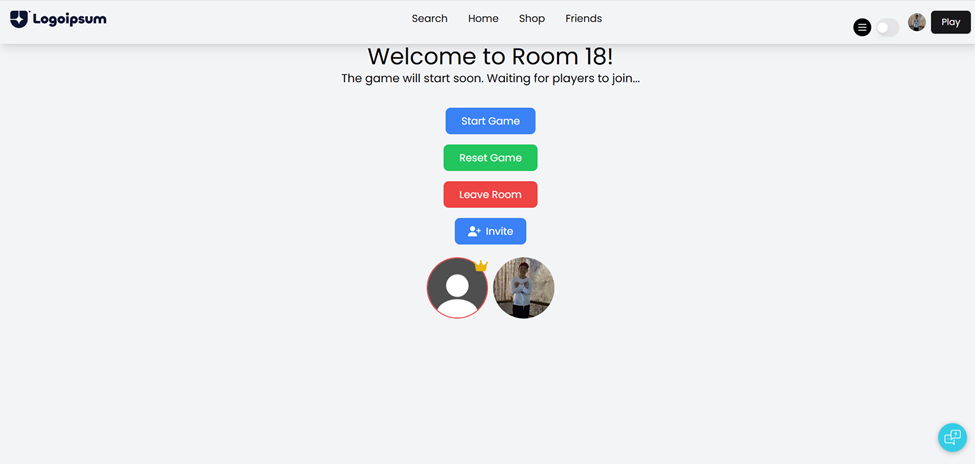
****

### 3.4.2. Giao diện tham gia phòng đã có sẵn

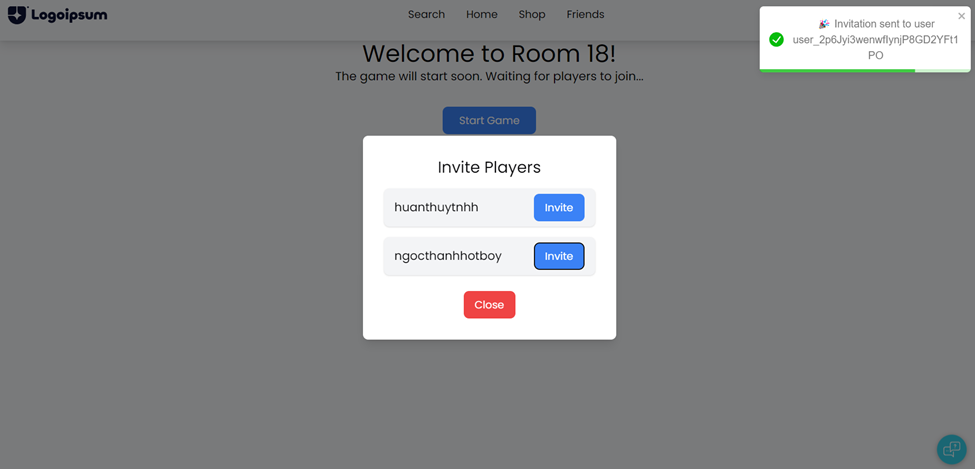
****

### 3.4.3. Giao diện hàng đợi

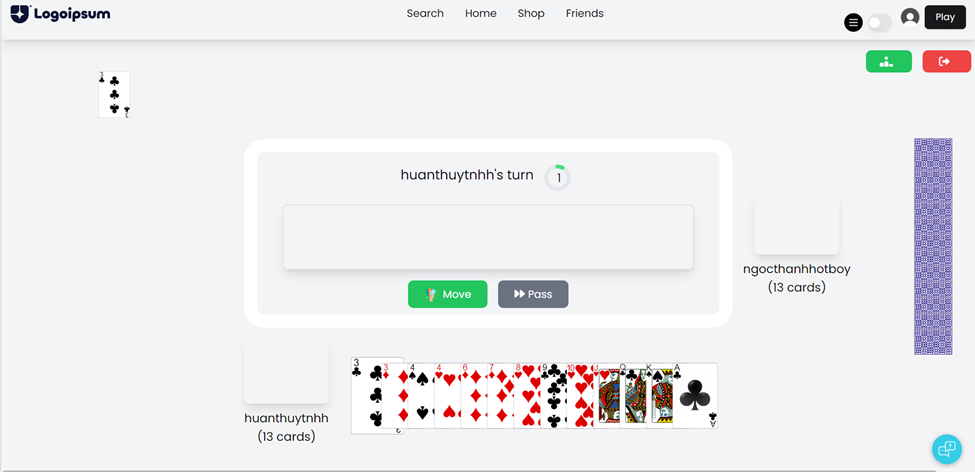
****

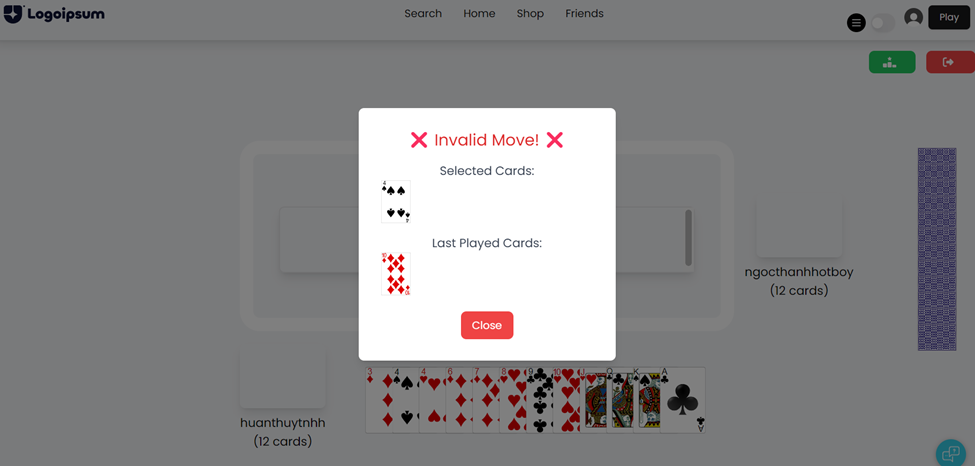
****

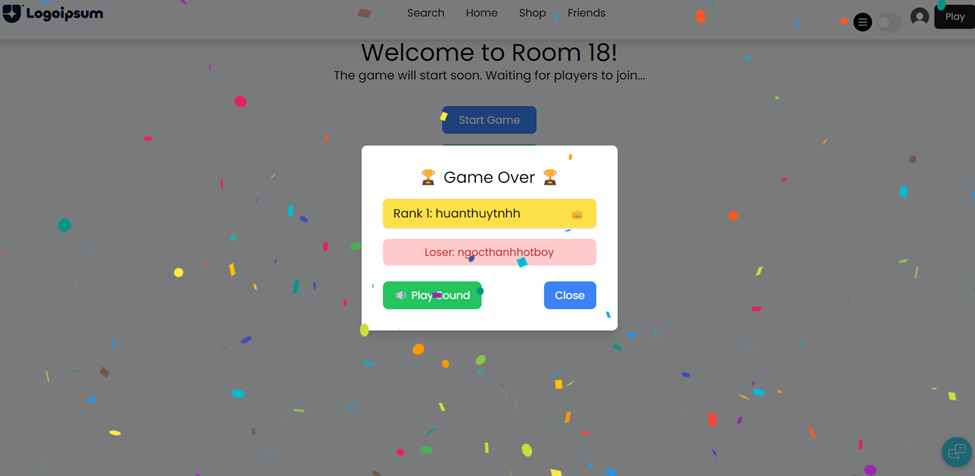
### 3.4.4. Giao diện mời người chơi khác

****

## 3.5. Giao diện trong ván đấu

****

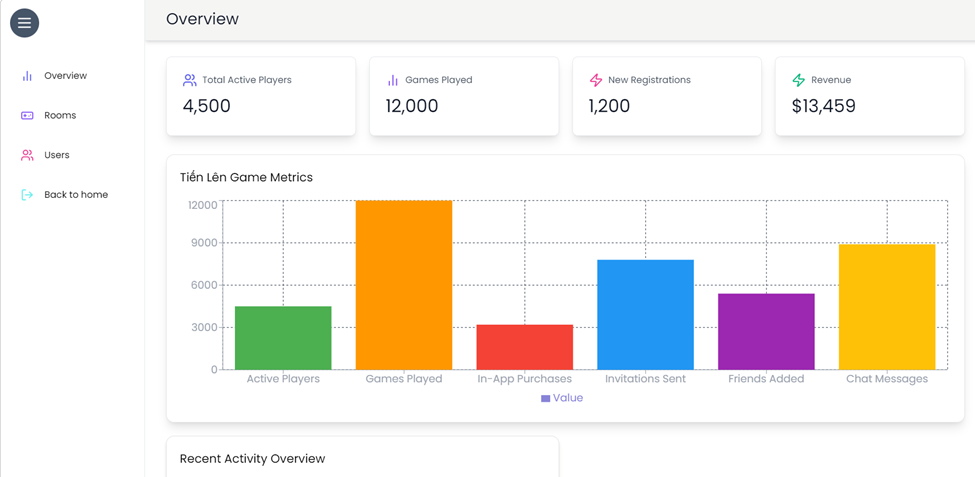
****

****

## 3.6. Giao diện chat

****

## 3.7. Giao diện tổng quan admin dashboard

****

# 

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

**I. Kết Luận**

Báo cáo trình bày quá trình thiết kế và triển khai trò chơi Tiến lên dựa trên mô hình client-server. Thông qua việc nghiên cứu và áp dụng giao thức truyền thông TCP/IP, hệ thống đã được xây dựng đảm bảo tính tương tác thời gian thực và khả năng giao tiếp giữa các client thông qua server trung gian. Trò chơi hoạt động ổn định, cung cấp trải nghiệm chơi mượt mà và đáp ứng nhanh chóng trước các lệnh từ người chơi.

**II. Hướng Phát Triển**

Trong tương lai, đề tài này có thể được mở rộng với những hướng phát triển như sau:

* Tăng cường giao diện người dùng (UI/UX): Áp dụng các thư viện đồ hoạ cải thiện trải nghiệm người chơi.
* Thêm tính năng hỗ trợ người chơi: Thay đổi vị trí các là bài theo ý muốn của người chơi.
* Hệ thống xếp hạng và thống kê: Theo dõi kết quả trận đấu, tổng số chiến thắng/thua và xếp hạng người chơi.
* Thêm chế độ chơi với máy: Xây dựng đối thủ AI với nhiều mức độ khó khác nhau, giúp người chơi có thể luyện tập trước khi thi đấu với người thật.
* Bổ sung chế độ giao tiếp trong game: Tích hợp video và voice chat giúp người chơi dễ giao lưu và tương tác trong quá trình thi đấu.
* Triển khai trên nhiều nền tảng: Phát triển phiên bản di động và web để đáp ứng nhiều thiết bị hơn.

TÀI LIỆU THAM KHẢO

**[1]** Mai Văn Hà, *Bài giảng Lập Trình Mạng*, *Công nghệ Web* Khoa Công Nghệ Thông Tin – Trường Đại học Bách Khoa – Đại học Đà Nẵng.

**[2]** [*https://socket.io/docs/v4/*](https://socket.io/docs/v4/), Socket IO, 21/10/2024

**[3]** Trịnh Công Duy, *Bài giảng Công nghệ phần mềm* Khoa Công Nghệ Thông Tin – Trường Đại học Bách Khoa – Đại học Đà Nẵng.

**[4]** Nguyễn Thế Xuân Ly, *Bài giảng Điện toán đám mây*, Khoa Công Nghệ Thông Tin – Trường Đại học Bách Khoa – Đại học Đà Nẵng.