BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC BÀ RỊA – VŨNG TÀU

BARIA VUNGTAU
UNIVERSITY
CAP SAINT JACQUES

ĐỒ ÁN TỐT NGHIỆP

GAME ĐỐI KHÁNG WIBUPIXEL

Trình độ đào tạo: Đại học chính quy

Ngành: Công nghệ thông tin

Chuyên ngành: Lập trình Ứng dụng di động & Game

Giảng viên hướng dẫn: ThS. Nguyễn Thị Minh Nương

Sinh viên thực hiện: Đào Phú Thịnh

Mã số sinh viên: 20035398

Lớp: DH20LT

Thành phố Vũng Tàu, ngày 01 tháng 04 năm 2024

LỜI NÓI ĐẦU

Tôi vô cùng phấn khích và tự hào khi được chia sẻ với bạn đọc về dự án tốt nghiệp của mình - một dự án mà tôi đã dành nhiều tháng, thậm chí là nhiều năm để hoàn thiện. Trong những dòng này, tôi muốn kể về hành trình học hỏi, sáng tạo và thách thức mà tôi đã trải qua khi xây dựng tựa game đối kháng online mang tên "WibuPixel".

Từ lâu, tôi đã ấp ủ ước mơ tạo ra một trò chơi mang tính cách mạng, không chỉ hấp dẫn người chơi bởi cốt truyện và gameplay sáng tạo mà còn giúp họ kết nối với nhau thông qua mạng lưới trực tuyến. Với sự hỗ trợ của công nghệ Unity và Photon2, cùng với sự sáng tạo từ Photoshop, tôi tin rằng dự án này sẽ là một bước đột phá trong sự nghiệp của mình.

Trong quá trình phát triển, tôi đã đối mặt với nhiều thách thức và khó khăn, từ việc thiết kế nhân vật, lập trình gameplay cho đến tối ưu hóa hiệu suất. Nhưng với đam mê, kiên trì và sự hỗ trợ từ bạn bè, giáo viên và cộng đồng lập trình viên, tôi đã vượt qua mọi trở ngại và hoàn thành dự án với niềm tự hào và sự hài lòng.

LỜI CẢM ƠN

Trong quá trình thực hiện đồ án tốt nghiệp này, tôi đã nhận được rất nhiều sự giúp đỡ, hỗ trợ và động viên từ nhiều người. Đây là nguồn động lực to lớn giúp tôi vượt qua những khó khăn và hoàn thành dự án một cách tốt nhất.

Trước hết, tôi xin bày tỏ lòng biết ơn sâu sắc đến người hướng dẫn Đồ án tốt nghiệp - cô Nguyễn Thị Minh Nương. Cô không chỉ truyền đạt kiến thức mà còn hướng dẫn, chỉ bảo và động viên tôi trong suốt quá trình nghiên cứu và hoàn thành đồ án. Sự tận tâm và kiên nhẫn của cô đã giúp tôi vượt qua nhiều thách thức và trưởng thành hơn trong con đường học vấn.

Đặc biệt, tôi xin gửi lời cảm ơn đến gia đình và bạn bè đã luôn ủng hộ, động viên và khích lệ tôi trong suốt quá trình học tập và nghiên cứu. Sự hỗ trợ tinh thần và vật chất của mọi người là động lực lớn lao giúp tôi vượt qua những khó khăn và thử thách.

Cuối cùng, tôi xin cảm ơn cộng đồng lập trình viên, những người bạn đồng hành trên các diễn đàn và nhóm lập trình đã chia sẻ kinh nghiệm, kiến thức và cùng tôi giải quyết những vấn đề gặp phải trong quá trình phát triển dự án. Sự chia sẻ và hỗ trợ của các bạn là nguồn tài nguyên vô giá giúp tôi hoàn thiện sản phẩm này.

Mặc dù đã cố gắng rất nhiều, nhưng đồ án vẫn không tránh khỏi những thiếu sót. Rất mong nhận được sự thông cảm, chỉ dẫn và góp ý của quý thầy cô và các bạn để đồ án được hoàn thiện hơn.

Xin chân thành cảm ơn!

Thành phố Vũng Tàu, ngày 01 tháng 04 năm 2024

Sinh viên thực hiện

Đào Phú Thịnh

NHẬN XÉT CỦA GIẢNG VIÊN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Thành phố Vũng Tàu, ngày … tháng … năm 202…

Giảng viên xác nhận

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

DANH MỤC CÁC TỪ VIẾT TẮT/TỪ TIẾNG ANH

Wibu: là thuật ngữ để chỉ những người hâm mộ văn hóa Nhật Bản (manga, anime, games) đặc biệt là những người yêu thích anime. Từ này có nguồn gốc từ "Weeaboo" trong tiếng Anh, nhưng được viết lại theo phát âm tiếng Việt.

Pixel: Pixel là đơn vị nhỏ nhất của một hình ảnh số, thường được sử dụng trong game để tạo ra đồ họa dạng điểm ảnh. Trò chơi có phong cách pixel thường sử dụng hình ảnh có độ phân giải thấp, tạo ra cảm giác retro và độc đáo.

Gameplay: là thuật ngữ chỉ cách mà người chơi tương tác với một trò chơi, bao gồm cả cách chơi, cơ chế và cảm nhận khi tham gia trải nghiệm game.

Engine: là một phần mềm hoặc hệ thống phần mềm được sử dụng để phát triển và chạy trò chơi điện tử hoặc ứng dụng đa phương tiện.

Consoles: là các hệ máy chơi game dành riêng cho việc chơi game, như PlayStation, Xbox, Nintendo Switch, và các thiết bị tương tự.

Framework: là một cấu trúc hoặc một tập hợp các công cụ và nguyên tắc được sử dụng để phát triển phần mềm hoặc ứng dụng.

Sprite: là một hình ảnh hoặc đối tượng được sử dụng trong trò chơi điện tử, thường là các nhân vật, vật phẩm hoặc cảnh vật.

Indie games: là các trò chơi được phát triển bởi các công ty độc lập hoặc nhóm phát triển nhỏ, thường không thuộc về các hãng game lớn.

Kick: là một thuật ngữ trong game để chỉ hành động đuổi người chơi khác ra khỏi phòng.

FPS: là viết tắt của "Frames Per Second", là số lượng hình ảnh mà màn hình hiển thị trong một giây. Trong game, FPS thường được sử dụng để đo lường hiệu suất của trò chơi.

Ping: là thời gian mà một gói dữ liệu mất để đi từ máy tính của người chơi đến máy chủ game và trở lại. Ping thấp thường chỉ ra một kết nối mạng ổn định và ít lag trong game.

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1. Đặt vấn đề

Trong thời đại công nghệ thông tin phát triển mạnh mẽ, việc phát triển trò chơi điện tử không chỉ là một lĩnh vực giải trí mà còn là một phần không thể thiếu của ngành công nghiệp giải trí toàn cầu. Trong số các thể loại trò chơi, các trò chơi đối kháng luôn là điểm sáng và thu hút sự quan tâm của hàng triệu người chơi trên khắp thế giới.

Tuy nhiên, trên thị trường hiện nay, vẫn còn thiếu đi một số lượng đáng kể các trò chơi đối kháng dành cho người chơi trực tuyến, đặc biệt là trong lĩnh vực game dành cho cộng đồng yêu thích văn hóa Nhật Bản - hay còn được gọi là "Wibu". Để phát triển một trò chơi đối kháng không chỉ đơn giản là việc kết hợp giữa sự sáng tạo trong thiết kế nhân vật, cốt truyện và cảnh quan mà còn đòi hỏi sự hiểu biết sâu sắc về lập trình và công nghệ. Đồng thời, việc xây dựng một cộng đồng trò chơi mạnh mẽ cũng là một yếu tố không thể thiếu để giữ cho trò chơi sống động và bền vững.

Hầu hết các trò chơi trên thị trường thường áp đặt các loại phí khi chơi như nạp tiền vào game, xem quảng cáo hoặc đầu tư thời gian "cày cuốc" để có được các vật phẩm mong muốn. Điều này đã tạo ra một khoảng trống mà dự án WibuPixel nhắm đến - tạo ra một cộng đồng chơi game WibuPixel, nơi mà không yêu cầu quá nhiều thời gian, tiền bạc, mà chỉ cần sự đam mê và niềm vui trong việc tham gia trải nghiệm game.

1.2. Ý tưởng Game

Trong thế giới của WibuPixel, người chơi sẽ bước vào một cuộc phiêu lưu huyền bí, nơi các nhân vật manga huyền thoại được triệu hồi để tham gia vào các trận đấu đỉnh cao. Ý tưởng của trò chơi là một quyển sách kể về cuộc chiến giữa các nhân vật manga, nơi mà sức mạnh, kỹ năng và chiến thuật được thể hiện thông qua các trận đấu đồng đội hoặc đối kháng.

Người chơi sẽ được hòa mình vào một thế giới tuyệt vời, nơi mà họ có thể điều khiển các nhân vật yêu thích từ các tác phẩm manga nổi tiếng: Từ các chiến binh

Saiyan mạnh mẽ, các Ninja sử dụng các kết ấn đẹp mắt hay các hải tặc ra khơi tìm kho báu của mình, mọi người chơi đều có cơ hội để chứng tỏ sức mạnh của mình trong các trận đấu nảy lửa.

Với đồ họa pixel độc đáo, trò chơi sẽ tái hiện lại không khí hấp dẫn và huyền bí của thế giới manga, tạo nên một trải nghiệm đầy kích thích và thú vị cho người chơi. Cuộc phiêu lưu trong thế giới manga sẽ đưa người chơi đến những cuộc chiến đỉnh cao, nơi mà họ phải thể hiện bản lĩnh và chiến đấu để trở thành kẻ mạnh nhất.

1.3. Hướng giải quyết

Để giải quyết vấn đề thiếu hụt các trò chơi đối kháng dành cho cộng đồng yêu thích văn hóa Nhật Bản và để tạo ra một không gian trò chơi không yêu cầu quá nhiều thời gian và tiền bạc từ người chơi, WibuPixel sẽ tiếp cận một số giải pháp chính sau đây:

1.3.1. Phát triển trò chơi đối kháng "WibuPixel":

- Game được phát triển là một trò chơi đối kháng trực tuyến mang tên "WibuPixel" với phong cách thiết kế và gameplay chủ đạo dựa trên văn hóa Nhật Bản.
- Trò chơi sẽ được xây dựng trên nền tảng công nghệ Unity và sử dụng tính năng mạng lưới trực tuyến của Photon2 để cho phép người chơi tham gia trận đấu với nhau từ mọi nơi trên thế giới.
- Việc tạo ra trải nghiệm trò chơi mượt mà, ổn định và hấp dẫn sẽ là ưu tiên hàng đầu của dự án.

1.3.2. Tạo cộng đồng game mạnh mẽ:

- Game sẽ tạo ra một cộng đồng trò chơi mạnh mẽ và sôi động bằng cách thúc đẩy sự tương tác giữa các thành viên thông qua diễn đàn trò chơi, cổng thông tin trò chơi và mạng xã hội.
- Sự tích hợp của các tính năng xã hội như chia sẻ thành tích, thách đấu giữa bạn bè và hệ thống giải thưởng sẽ tạo động lực cho người chơi tham gia và gắn kết với cộng đồng.

Commented [NTM1]: Phần này em đã làm chưa?

Commented [PT2R1]: Em sẽ làm 1 page facebook, push 1 vài bài viết lên đó để làm tính năng cộng đồng. Em sẽ thêm một đường dẫn vào website của em.

Commented [PT3R1]:

1.3.3. Không áp đặt các loại phí khi chơi:

- Cam kết tạo ra một mô hình kinh doanh công bằng và minh bạch bằng cách không áp đặt các loại phí khi chơi, không buộc người chơi phải chi tiền hoặc thời gian để có được trải nghiệm đầy đủ của trò chơi.
- Trong trường hợp cần thiết, sẽ xem xét các hình thức tài trợ hoặc hợp tác quảng cáo để duy trì hoạt động và phát triển của trò chơi một cách bền vững.

Commented [NTM4]: Cái này là hướng đến hay đã lên phương án?

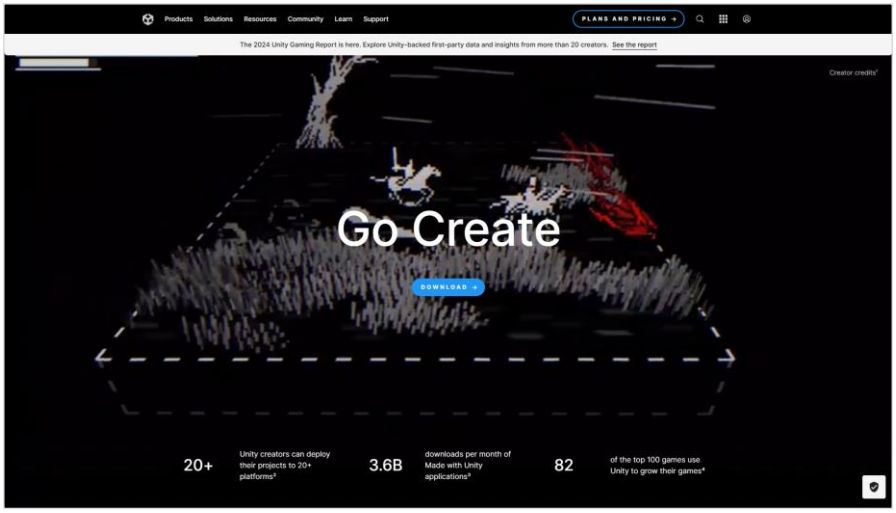
Commented [PT5R4]: Cái này là về sau rồi ạ. Nếu cộng đồng người chơi mà lớn nhưng lại không có nguồn thu từ game mà phải cần phí để duy trì game thì mới cần tới các nhà đầu tư quảng cáo hoặc là donate để duy trì game.

Commented [PT6R4]:

1.4. Các công nghệ được lựa chọn

1.4.1. Unity

Unity là một game engine đa nền tảng được phát triển bởi Unity Technologies, mà chủ yếu để phát triển video game cho máy tính, consoles và điện thoại. Lần đầu tiên nó được công bố chạy trên hệ điều hành OS X, tại Apple's Worldwide Developers Conference vào năm 2005, đến nay đã mở rộng 27 nền tảng.



Hình 1 Trang chủ Unity

6 phiên bản chính của phần mềm này đã được phát hành. Tại triển lãm WWDC năm 2006, Apple đã trao thưởng giải Best Use of Mac OS X Graphics cho ứng dụng này.

Các đặc trưng của Unity:

- **Đa nền tảng:** Unity hỗ trợ phát triển trò chơi và ứng dụng trên nhiều nền tảng khác nhau như Windows, macOS, Linux, iOS, Android, các thiết bị game console như PlayStation và Xbox, cũng như các thiết bị thực tế ảo và thực tế tăng cường.
- **Cộng tác và đa người chơi:** Unity cho phép nhiều nhà phát triển làm việc cùng nhau trên cùng một dự án thông qua tích hợp công cụ cộng tác. Nó cũng cung cấp các tính năng mạng lưới để phát triển các trò chơi đa người chơi trực tuyến.
- **Đồ họa 2D và 3D:** Unity hỗ trợ phát triển cả trò chơi 2D và 3D với các công cụ và tính năng phong phú. Nó cung cấp các công cụ mạnh mẽ để tạo ra và quản lý đồ họa động và tĩnh.
- **Physic Engine:** Unity tích hợp một physic engine mạnh mẽ cho phép mô phỏng và xử lý các hiệu ứng vật lý như va chạm, chuyển động và gravitation trong trò chơi.
- **Hệ thống Animation:** Unity cung cấp một hệ thống animation linh hoạt cho phép tạo và quản lý các animation cho nhân vật, vật thể và các thành phần khác của trò chơi.
- **Scripting:** Unity hỗ trợ việc viết mã thông qua nhiều ngôn ngữ lập trình khác nhau như C#, JavaScript, và Boo. Điều này cho phép nhà phát triển sử dụng ngôn ngữ lập trình ưa thích của họ để tạo ra trò chơi.
- **Asset Store:** Unity có một cửa hàng Asset Store phong phú, cung cấp các tài nguyên như mẫu nhân vật, cảnh quan, hiệu ứng đặc biệt, và các plug-in để giúp nhà phát triển tăng tốc quá trình phát triển và tối ưu hóa trò chơi của họ.
- **Tích hợp AR/VR:** Unity hỗ trợ phát triển ứng dụng và trò chơi thực tế ảo (VR) và thực tế tăng cường (AR) trên nhiều nền tảng khác nhau, bao gồm Oculus Rift, HTC Vive, Microsoft HoloLens và thiết bị di động.

1.4.2. Photon

Photon là một nền tảng mạng lưới phổ biến được sử dụng để phát triển trò chơi đa người chơi trực tuyến. Nền tảng này cung cấp các công cụ và dịch vụ để quản lý kết nối mạng lưới giữa người chơi, đồng bộ hóa dữ liệu trò chơi, và tạo ra trải nghiệm trò chơi mượt mà và ổn định.

Hình 2 Trang chủ Photon

Photon có nhiều phiên bản khác nhau, trong đó hai phiên bản phổ biến nhất là Photon Realtime và Photon Unity Networking (PUN).

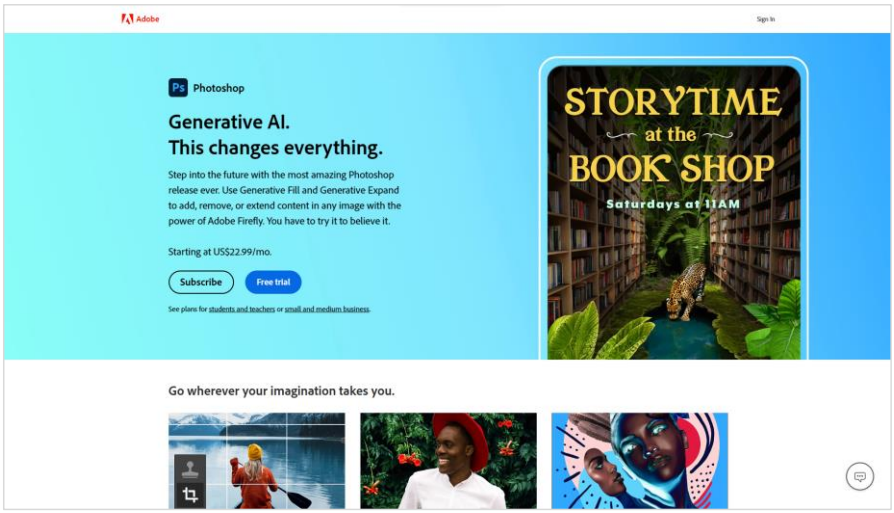
- **Photon Realtime:** Đây là một dịch vụ cốt lõi của Photon, cung cấp giải pháp mạng lưới phân tán cho việc tạo ra và quản lý các phòng trò chơi, truyền dữ liệu giữa người chơi và đồng bộ hóa trạng thái trò chơi. Photon Realtime thường được sử dụng để phát triển các trò chơi trực tuyến đa người chơi.
- **Photon Unity Networking (PUN):** PUN là một bộ công cụ tích hợp sẵn cho Unity, giúp nhà phát triển tạo ra trò chơi đa người chơi trực tuyến một cách dễ dàng. PUN cung cấp các lớp và phương thức để kết nối, tạo phòng và xử lý dữ liệu mạng lưới một cách hiệu quả. PUN thường được sử dụng trong việc phát triển trò chơi trực tuyến trên nền tảng Unity.

Các đặc trưng của Photon:

- **Hỗ trợ đa nền tảng:** Photon 2 hỗ trợ phát triển trò chơi trực tuyến trên nhiều nền tảng khác nhau như Windows, macOS, Linux, iOS, Android, PlayStation, Xbox và WebGL.
- **Realtime:** Photon Realtime là một trong những dịch vụ cốt lõi của Photon 2, cung cấp giải pháp mạng lưới phân tán cho việc tạo ra và quản lý các phòng trò chơi, truyền dữ liệu giữa người chơi và đồng bộ hóa trạng thái trò chơi.
- **PUN (Photon Unity Networking):** PUN là một bộ công cụ tích hợp sẵn cho Unity, giúp nhà phát triển tạo ra trò chơi đa người chơi trực tuyến một cách dễ dàng. PUN cung cấp các lớp và phương thức để kết nối, tạo phòng và xử lý dữ liệu mạng lưới một cách hiệu quả.
- **Photon Voice:** Photon Voice là một tính năng của Photon 2 cho phép trò chơi hỗ trợ chat thoại trong game. Nó cung cấp các API để ghi âm, truyền và phát lại âm thanh giữa các người chơi trong cùng một phòng trò chơi.
- **Photon Chat:** Photon Chat là một dịch vụ cho phép giao tiếp trò chuyện giữa người chơi trong trò chơi. Nó cung cấp các tính năng như phòng chat, tin nhắn riêng tư và nhóm, và khả năng tùy chỉnh.
- **Cloud Functions:** Photon 2 cung cấp Cloud Functions, cho phép nhà phát triển thêm logic xử lý dữ liệu phía máy chủ của mình một cách dễ dàng thông qua các hàm JavaScript.
- **Độ tin cậy và hiệu suất:** Photon 2 được thiết kế để có độ tin cậy và hiệu suất cao, giúp trò chơi chạy mượt mà và ổn định ngay cả trong môi trường mạng không ổn định.

1.4.3. Photoshop

Adobe Photoshop là một phần mềm chỉnh sửa ảnh và đồ họa chuyên nghiệp được phát triển bởi hãng phần mềm Adobe Systems. Được phát hành lần đầu vào năm 1988, Photoshop đã trở thành một công cụ phổ biến và quan trọng trong nhiều lĩnh vực như thiết kế đồ họa, nhiếp ảnh, quảng cáo, và nghệ thuật số.



Hình 3 Trang chủ Photoshop

Photoshop cung cấp một loạt các công cụ và tính năng mạnh mẽ cho việc chỉnh sửa và tạo ra hình ảnh, bao gồm:

- **Công cụ chỉnh sửa:** Cho phép người dùng cắt, chèn, xoay, biến dạng và điều chỉnh kích thước của các hình ảnh.
- **Công cụ vẽ và sơn:** Cung cấp bút vẽ, bút cọ, và các công cụ sơn để tạo ra và chỉnh sửa các phần tử đồ họa.
- **Lớp và lớp điều chỉnh:** Cho phép người dùng làm việc với nhiều lớp ảnh và điều chỉnh các hiệu ứng, màu sắc và ánh sáng của từng lớp.
- **Công cụ lọc và hiệu ứng:** Cung cấp một loạt các bộ lọc và hiệu ứng để thêm sự động cho ảnh, bao gồm làm mờ, làm sáng, tạo ra hiệu ứng màu sắc, và nhiều hơn nữa.
- **Chỉnh sửa văn bản:** Cho phép người dùng thêm, sửa đổi và biến đổi văn bản trực tiếp trên hình ảnh.
- **Công cụ đồ họa vector:** Photoshop cung cấp một số công cụ để tạo ra đồ họa vector như hình elip, hình chữ nhật, v.v.

1.4.4. Thiết kế Website deploy sản phẩm (HTML, Tailwind, Javascript, VPS)

1.4.4.1. *HTML (HyperText Markup Language):*

- HTML là ngôn ngữ đánh dấu được sử dụng để tạo cấu trúc và định dạng nội dung trên trang web.
- HTML sử dụng các thẻ (tag) để xác định các phần tử trên trang web, ví dụ như tiêu đề, đoạn văn bản, hình ảnh, liên kết, và bảng.
- HTML là ngôn ngữ cơ bản và quan trọng nhất trong việc xây dựng trang web.

1.4.4.2. *Tailwind CSS:*

- Tailwind CSS là một framework CSS, cung cấp một bộ công cụ và lớp CSS được đặt tên mô tả để thiết kế giao diện người dùng trên web.
- Tailwind CSS không phải là một framework CSS truyền thống, mà thay vào đó nó cung cấp các lớp CSS được đặt tên mô tả để áp dụng các kiểu thiết kế.
- Với Tailwind CSS, người phát triển có thể tạo ra giao diện độc đáo và tùy chỉnh dễ dàng bằng cách sử dụng các lớp CSS có sẵn.

1.4.4.3. *JavaScript:*

- JavaScript là một ngôn ngữ lập trình thông dịch và đa nền tảng được sử dụng chủ yếu cho việc phát triển ứng dụng web tương tác.
- JavaScript được sử dụng để thêm các tính năng động và tương tác vào trang web, bao gồm xử lý sự kiện, thao tác với DOM (Document Object Model), và giao tiếp với máy chủ thông qua AJAX.
- JavaScript cũng được sử dụng phổ biến trong phát triển ứng dụng di động và các ứng dụng máy chủ bằng Node.js.

1.4.4.4. *VPS (Virtual Private Server):*

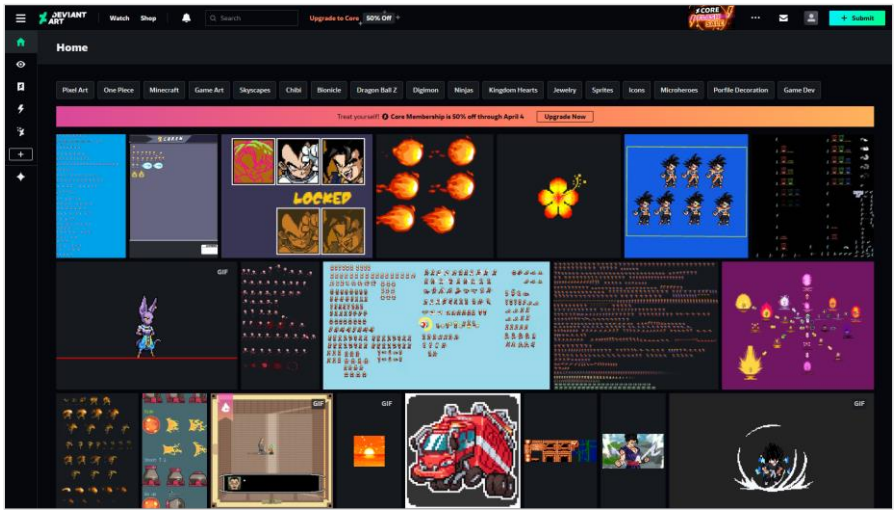
- VPS là một dạng máy chủ ảo được cung cấp bởi một nhà cung cấp dịch vụ hosting.

- Trong môi trường VPS, người dùng có thể thuê một phần của một máy chủ vật lý và cấu hình máy chủ ảo của họ theo nhu cầu riêng của họ.
- VPS cung cấp các tài nguyên độc lập và cách ly từ các máy chủ khác, cho phép người dùng có quyền truy cập và kiểm soát hoàn toàn máy chủ ảo của mình.

1.5. Assets

1.5.1. Deviantart

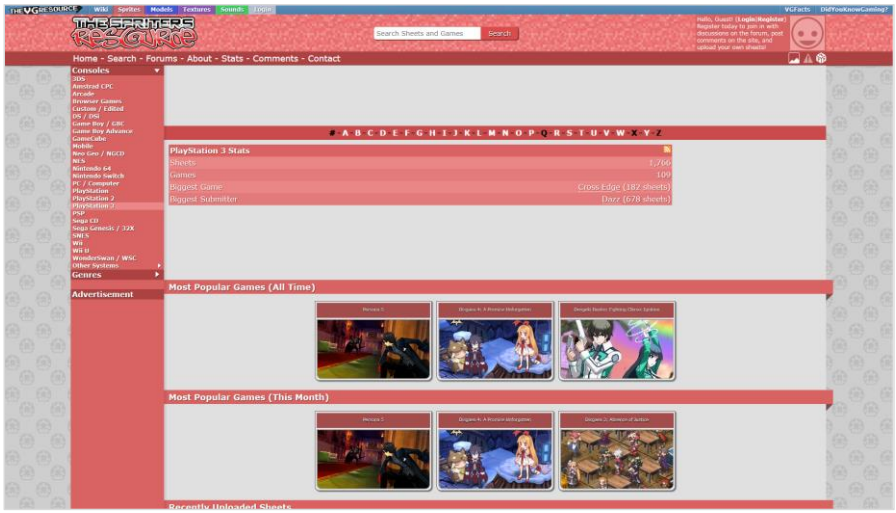
Deviantart là một cộng đồng trực tuyến nổi tiếng cho người yêu nghệ thuật và thiết kế. Trên Deviantart, người dùng có thể tải lên và chia sẻ các tài nguyên như hình ảnh, hình vẽ, mẫu nhân vật, và các tài nguyên khác có thể được sử dụng trong phát triển trò chơi, ứng dụng, hoặc các dự án nghệ thuật khác.



Hình 4 Trang chủ Deviantart

1.5.2. The Spriters Resource

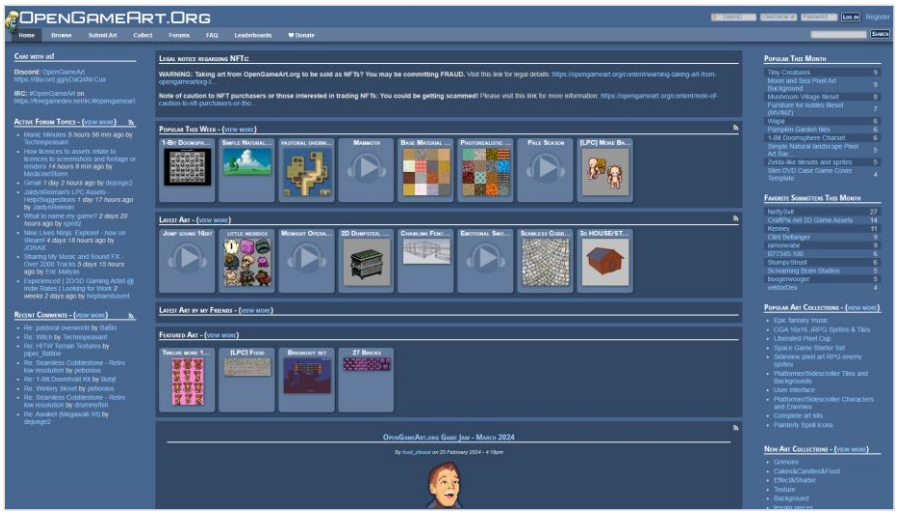
The Spriters Resource là một trang web nổi tiếng chuyên cung cấp các tài nguyên sprite và hình ảnh cho các trò chơi điện tử. Trên The Spriters Resource, người dùng có thể tìm thấy và tải về các sprite, background, và các loại hình ảnh khác được sử dụng trong phát triển trò chơi 2D và các dự án liên quan.



Hình 5 Trang chủ The Spriters Resource

1.5.3. Open Game Art

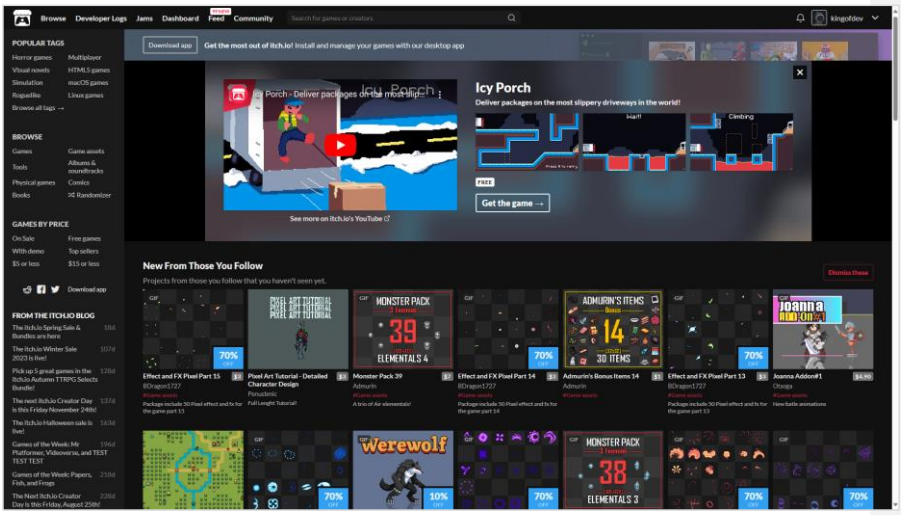
Open Game Art là một cộng đồng trực tuyến cho phép người dùng chia sẻ các tài nguyên miễn phí dành cho phát triển trò chơi. Các tài nguyên trên Open Game Art bao gồm các sprite, âm thanh, hiệu ứng âm thanh, background, và nhiều loại tài nguyên khác được sử dụng để tạo ra trò chơi đa dạng và đa dạng hơn.



Hình 6 Trang chủ Open Game Art

1.5.4. itch.io

itch.io là một nền tảng kỹ thuật số cho phép người dùng tải xuống và chơi trò chơi độc lập (indie games), cũng như nền tảng cho phép các nhà phát triển trò chơi độc lập đăng ký và bán trò chơi của họ.



Hình 7 Trang chủ itch.io

1.6. Kịch bản game WibuPixel

Đây là một quyển manga không giống với bất kì một quyển manga nào mà bạn đã đọc trước đây. Đây sẽ là một quyển truyện dành cho những fan cuồng nhiệt của văn hóa Nhật Bản mà còn là nơi mà sức mạnh của từng nhân vật manga được thử thách và thể hiện thông qua các trận đấu đối kháng đỉnh cao. Tất cả các nhân vật Manga, từ những anh hùng đến những kẻ phản diện, từ những nhân vật chính đến những nhân vật phụ, đều sẽ được triệu hồi tại đây và tham gia chiến đấu với nhau để tìm ra kẻ mạnh nhất.

Hành trình bắt đầu khi người chơi được đưa vào thế giới của WibuPixel, một thế giới được hình thành từ sự kết hợp của các thế giới manga nổi tiếng. Từ âm thanh quen thuộc như "Gomu gomu no…" trong "One Piece" hay các nhẫn thuật nổi tiếng trong "Naruto". Mọi thứ đều có trong WibuPixel.

Người chơi sẽ được đưa vào vai một người hùng có khả năng điều khiển các nhân vật manga mạnh mẽ. Với khả năng hấp thụ các loại ngọc, các phép bổ trợ và kỹ

năng độc nhất dành riêng cho mỗi nhân vật. Người dùng sẽ có thể tìm ra một lối chơi phù hợp với mình nhất. Với những thử thách và chiến đấu với những nhân vật mạnh mẽ nhất từ các tác phẩm manga khác nhau để thu thập sức mạnh cho mình.

Nhưng hành trình không chỉ là về chiến đấu mà người chơi còn có cơ hội tìm hiểu về tính cách, quá khứ và mục tiêu của từng nhân vật mà họ triệu hồi. Mỗi nhân vật đều mang trong mình một câu chuyện riêng, và bằng cách tương tác với họ, người chơi sẽ có cơ hội khám phá thêm về thế giới manga đầy màu sắc này.

Trên con đường của mình, người chơi sẽ gặp gỡ những đối thủ mạnh mẽ, những thử thách cam go và cơ hội vươn lên trở thành nhà vô địch trong cuộc thi WibuPixel. Họ sẽ phải thể hiện khả năng chiến đấu, sự sáng tạo và khả năng lập kế hoạch chiến thuật trong từng trận đấu để đạt được vị trí cao nhất trên bảng xếp hạng.

WibuPixel không chỉ là nơi thách thức sức mạnh cá nhân, mà còn là nơi tạo ra cơ hội cho sự kết nối và tương tác giữa cộng đồng người chơi. Người chơi có thể thách đấu với bạn bè, tham gia vào các hoạt động cộng đồng, và chia sẻ thành tích của mình trên các diễn đàn và mạng xã hội. Mọi người chơi đều là một phần của một cộng đồng đam mê, nơi mà họ có thể tận hưởng niềm vui của trò chơi mà không gặp phải bất kỳ rào cản nào.

Chuẩn bị cho một cuộc hành trình đầy mạo hiểm và kích thích trong thế giới của WibuPixel, nơi mà sức mạnh, kỹ năng và tình bạn sẽ là chìa khóa cho sự thành công. Hãy sẵn lòng đối mặt với những thử thách, khám phá những bí mật và tận hưởng những niềm vui trong thế giới manga đầy màu sắc này. Hãy cùng nhau tạo nên một câu chuyện mới, một huyền thoại mới trong thế giới của WibuPixel!

CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

2.1. Khảo sát sơ bộ

2.1.1. Đối tượng sử dụng

Tất cả người dùng, mọi lứa tuổi.

2.1.2. Dòng thiết bị

Dòng thiết bị mà game này hướng đến là tất cả các máy tính chạy hệ điều hành Windows. Việc chọn dòng thiết bị này là vì Windows có một lượng người dùng rất lớn trên toàn cầu, và nền tảng này cung cấp môi trường phát triển phong phú và đa dạng cho ứng dụng và trò chơi.

Các máy tính chạy Windows có mức độ đa dạng rất cao, từ các máy tính cấu hình cao đến các máy tính cấu hình thấp hơn. Do đó, khi thiết kế và phát triển trò chơi, cần đảm bảo rằng trò chơi có thể hoạt động một cách mượt mà và ổn định trên nhiều loại hệ thống và cấu hình khác nhau. Điều này đòi hỏi phải tối ưu hóa hiệu suất của trò chơi và thực hiện kiểm tra kỹ lưỡng trên nhiều loại máy tính để đảm bảo sự tương thích và trải nghiệm tốt nhất cho người chơi.

WibuPixel không quá đòi hỏi về phần cấu hình vì đồ họa pixel nên không yêu cầu quá cao về cấu hình. Nhưng nếu càng cao thì đồ họa trong game sẽ càng sắc nét, tăng trải nghiệm của người chơi với game.

2.2. Phân tích Hệ thống

Chuyên mục này được triển khai bằng UML – Ngôn ngữ mô hình hoá thống nhất (https://www.tutorialspoint.com/uml/uml_standard_diagrams.htm).

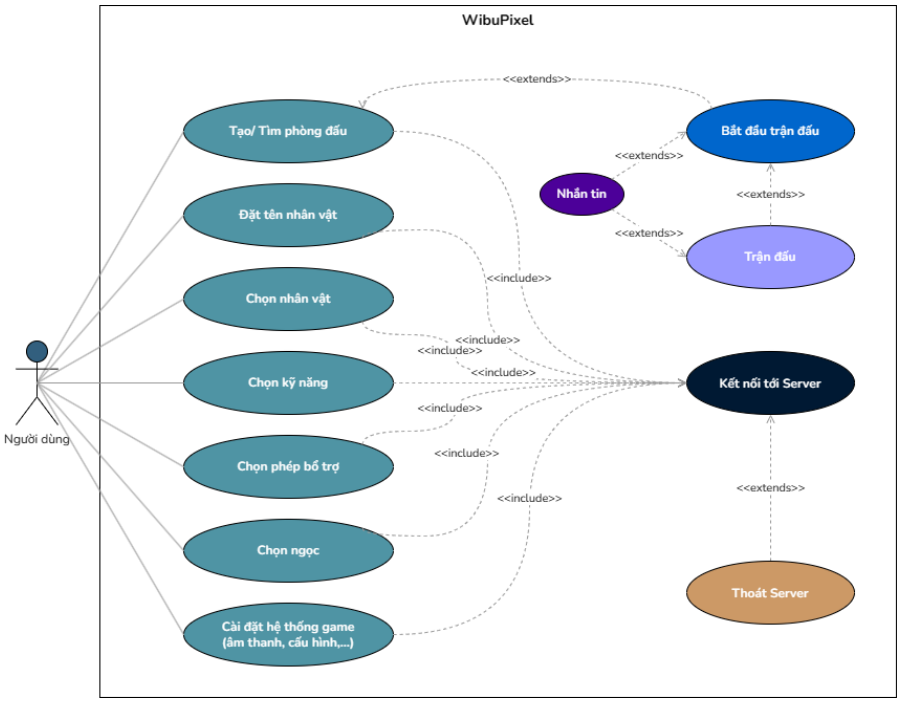
Các dạng Biểu đồ được thể hiện cho mỗi chức năng bao gồm:

- UseCase Specification (Đặc tả chức năng).
- UseCase Diagram: Mô tả các mối quan hệ và sự phụ thuộc giữa các chức năng (https://www.tutorialspoint.com/uml/uml_use_case_diagram.htm).
- Sơ đồ tuần tự (Sequence Diagram): Sơ đồ này minh họa các tương tác giữa các đối tượng trong một chuỗi thời gian xác định. Nó mô tả cách các đối tượng giao tiếp với nhau trong một kịch bản cụ thể hoặc trong quá trình thực hiện một chức năng cụ thể.

2.2.1. Các tác nhân

Trong hệ thống này, tác nhân trực tiếp sử dụng game là: Người dùng (Tương tác với tất cả các UI/UX có trong hệ thống game).

2.2.2. Usecase tổng quát



Hình 8 Usecase Tổng quát

2.2.3. Các Usecase chi tiết

2.2.3.1. Chức năng Đặt tên nhân vật

a. Đặc tả chức năng

Tác nhân: Người dùng.

Mô tả chức năng: Chức năng Đặt tên nhân vật cho phép người chơi đặt tên cho nhân vật của mình khi kết nối với server lần đầu tiên. Nếu lần tiếp theo kết nối tới server mà muốn đổi tên thì phải click vào biểu tượng "Bút chì".

Quy trình:

Commented [NTM7]: Này là tên của nút nhấn? Nếu k có tên thì chụp ảnh của nút

Commented [PT8R7]: Đã thêm Icon

Commented [PT9R7]:

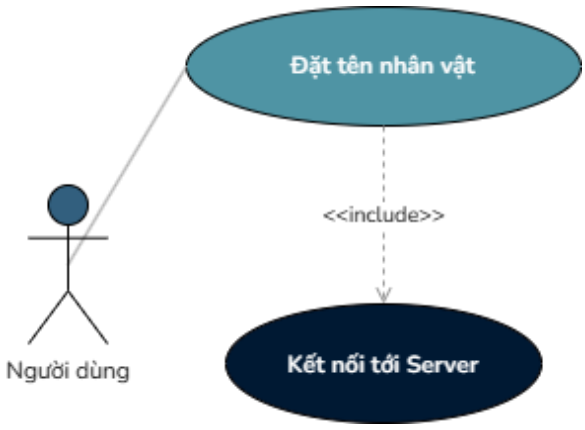
1. Người chơi kết nối với server hoặc mở ứng dụng trò chơi lần đầu tiên.
2. Hệ thống nhận diện người chơi là người mới và yêu cầu họ đặt tên cho nhân vật của mình.
3. Một giao diện hoặc hộp thoại hiển thị cho người chơi để nhập tên cho nhân vật.
4. Người chơi nhập tên vào ô văn bản và xác nhận.
5. Hệ thống kiểm tra tính hợp lệ của tên (ví dụ: không chứa ký tự đặc biệt, không được ngắn hơn 3 và dài hơn 15 kí tự) và lưu trữ tên này trong cơ sở dữ liệu của trò chơi.
6. Sau khi tên được lưu trữ thành công, người chơi sẽ tiếp tục vào trò chơi hoặc giao diện chính của trò chơi.

Tiền điều kiện: Người chơi kết nối với server hoặc mở ứng dụng trò chơi lần đầu tiên.

Yêu cầu đặc biệt: Không có.

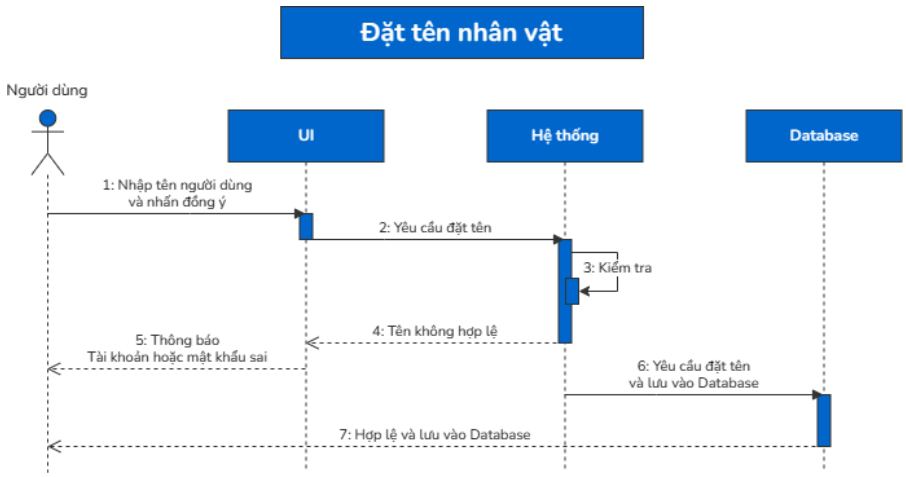
Hậu điều kiện: Người chơi đã đặt tên cho nhân vật của mình và có thể sử dụng tên này khi tham gia vào các trận đấu hoặc hoạt động khác trong trò chơi.

b. Biểu đồ Usecase



Hình 9 Usecase Đặt tên nhân vật

c. Sơ đồ tuần tự



Hình 10 Sơ đồ tuần tự Đặt tên nhân vật

2.2.3.2. Chức năng Chọn nhân vật

a. Đặc tả chức năng

Tác nhân: Người dùng.

Mô tả chức năng: Chức năng Chọn nhân vật cho phép người chơi chọn nhân vật mà họ muốn sử dụng trong trò chơi.

Quy trình:

1. Người chơi truy cập vào màn hình chọn nhân vật từ giao diện chính của trò chơi.
2. Hệ thống hiển thị danh sách các nhân vật có sẵn để chọn, bao gồm hình ảnh và mô tả ngắn về mỗi nhân vật.
3. Người chơi chọn một nhân vật từ danh sách bằng cách nhấp vào hình ảnh hoặc tên của nhân vật.
4. Hệ thống xác nhận lựa chọn của người chơi và cập nhật nhân vật được chọn trong dữ liệu trò chơi.
5. Người chơi được chuyển đến giao diện chính của trò chơi hoặc màn hình chờ đợi, đã có nhân vật của mình để bắt đầu trải nghiệm trò chơi.

Tiền điều kiện:

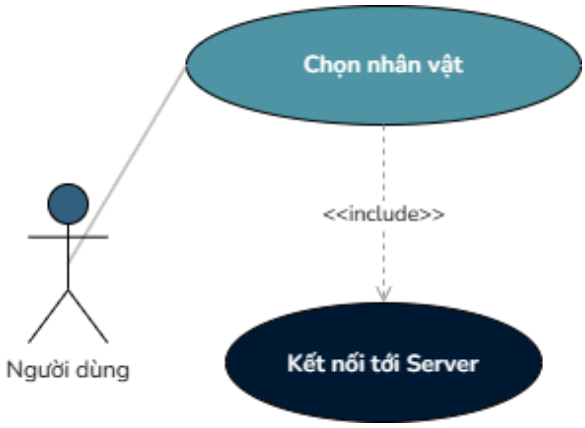
- Kết nối mạng ổn định để tìm và vào phòng.

- Người dùng đã đặt tên cho nhân vật

Yêu cầu đặc biệt: Không có.

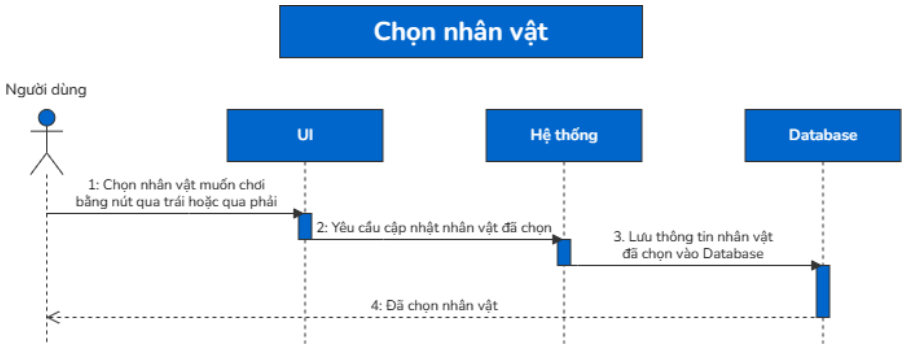
Hậu điều kiện: Người chơi đã chọn được nhân vật của mình và có thể bắt đầu trải nghiệm trò chơi với nhân vật đã chọn.

b. Biểu đồ Usecase



Hình 11 Usecase Chọn nhân vật

c. Sơ đồ tuần tự



Hình 12 Sơ đồ tuần tự Chọn nhân vật

2.2.3.3. Chức năng Chọn kỹ năng

a. Đặc tả chức năng

Tác nhân: Người dùng.

Mô tả chức năng: Chức năng Chọn kỹ năng cho phép người chơi chọn một kỹ năng mới hoặc nâng cấp kỹ năng hiện tại trong trò chơi. Mỗi kỹ năng có thể được nâng cấp tối đa 3 lần.

Quy trình:

1. Người chơi truy cập vào màn hình chọn kỹ năng từ giao diện chính của trò chơi.
2. Hệ thống hiển thị danh sách các kỹ năng có sẵn để chọn, bao gồm cả mô tả và hiệu ứng của mỗi kỹ năng.
3. Người chơi chọn một kỹ năng từ danh sách bằng cách nhấp vào tên hoặc biểu tượng của kỹ năng.
4. Hệ thống xác nhận lựa chọn của người chơi và cập nhật kỹ năng mới hoặc kỹ năng đã nâng cấp trong dữ liệu trò chơi.
5. Nếu người chơi chọn nâng cấp kỹ năng hiện tại:
   - Hệ thống kiểm tra xem kỹ năng đã nâng cấp đạt đến giới hạn tối đa chưa.
   - Nếu chưa, hệ thống tăng cường hiệu ứng hoặc sức mạnh của kỹ năng theo mức độ nâng cấp.
6. Người chơi được chuyển đến giao diện chính của trò chơi hoặc màn hình tiếp theo, đã có kỹ năng mới hoặc kỹ năng đã nâng cấp để sử dụng trong trò chơi.

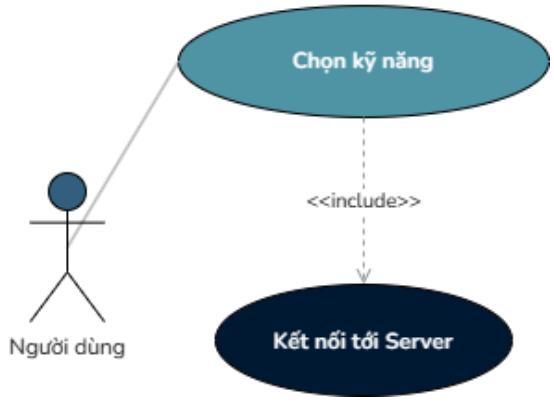
Tiền điều kiện:

- Kết nối mạng ổn định để tìm và vào phòng.
- Người dùng đã đặt tên cho nhân vật

Yêu cầu đặc biệt: Không có.

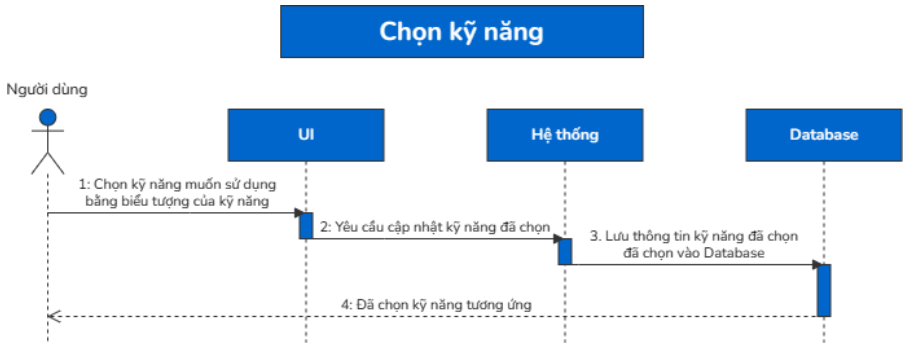
Hậu điều kiện: Người chơi đã chọn được kỹ năng mới hoặc kỹ năng đã nâng cấp và có thể sử dụng chúng trong trò chơi.

b. Biểu đồ Usecase



Hình 13 Usecase Chọn kỹ năng

c. Sơ đồ tuần tự



Hình 14 Usecase Chọn kỹ năng

2.2.3.4. Chức năng Chọn phép bổ trợ

a. Đặc tả chức năng

Tác nhân: Người dùng.

Mô tả chức năng: Chức năng Chọn phép bổ trợ cho phép người chơi chọn một trong 9 phép bổ trợ khác nhau để sử dụng trong trận đấu.

Danh sách các phép bổ trợ:

1. Hồi máu: Hồi phục một phần máu tối đa của nhân vật.
2. Hồi năng lượng: Hồi phục một phần năng lượng tối đa của nhân vật.
3. Tốc biến: Tăng tốc độ di chuyển lên phía trước.

4. Triple Jump: Cho phép nhân vật nhảy 3 lần liên tiếp.
5. Kháng: Giảm sát thương nhận được, miễn nhiễm bạo kích và các hiệu ứng khác trong một khoảng thời gian nhất định.
6. Lá chắn: Tạo ra một lớp lá chắn để giảm sát thương nhận được trong một khoảng thời gian nhất định.
7. Đòn kết liễu: Tăng sát thương và khả năng bạo kích của nhân vật trong một khoảng thời gian nhất định.
8. Khát máu: Hút máu từ sát thương gây ra cho đối thủ trong một khoảng thời gian nhất định.
9. Sát thương chuẩn: Gây sát thương mà bỏ qua giáp và khả năng né tránh của đối thủ trong một khoảng thời gian nhất định.

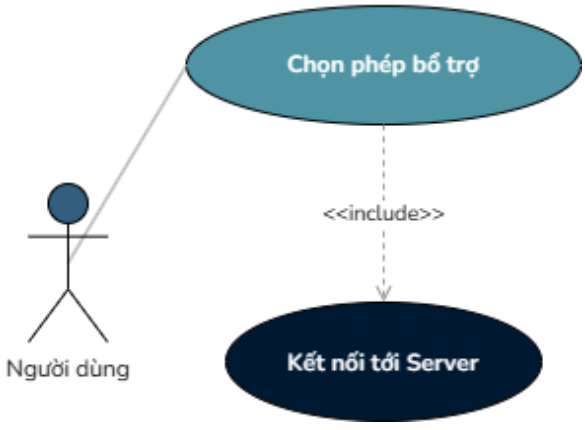
Tiền điều kiện:

- Kết nối mạng ổn định để tìm và vào phòng.
- Người dùng đã đặt tên cho nhân vật

Yêu cầu đặc biệt: Không có.

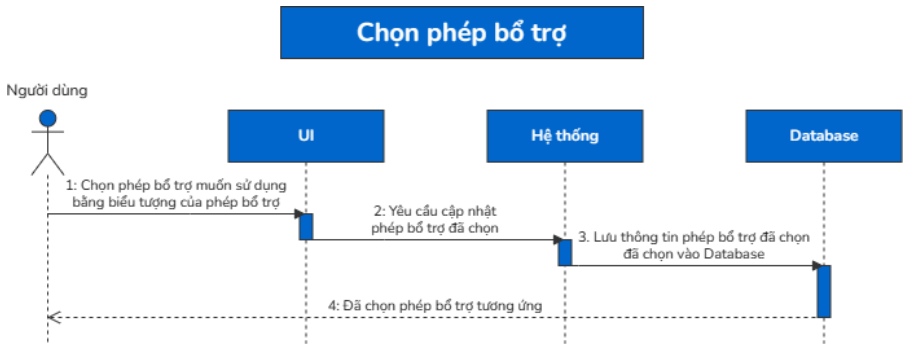
Hậu điều kiện: Người chơi đã chọn được phép bổ trợ mong muốn và có thể sử dụng chúng trong trận đấu.

a. Biểu đồ Usecase



Hình 15 Usecase Chọn phép bổ trợ

b. Sơ đồ tuần tự



Hình 16 Usecase Chọn phép bổ trợ

2.2.3.5. Chức năng Chọn ngọc

a. Đặc tả chức năng

Tác nhân: Người dùng.

Mô tả chức năng: Chức năng "Chọn Ngọc" cho phép người chơi chọn từ 18 loại ngọc khác nhau để tăng cường các chỉ số của nhân vật. Mỗi loại ngọc sẽ cộng 1 chỉ số khác nhau, và người chơi có thể chọn tối đa 6 loại ngọc, có thể trùng lặp, để bắt đầu trận đấu.

Danh sách các loại ngọc:

1. Ngọc Sát thương
2. Ngọc Phòng thủ
3. Ngọc Máu
4. Ngọc Hồi máu
5. Ngọc Năng lượng
6. Ngọc Hồi năng lượng
7. Ngọc Bạo kích
8. Ngọc Sát thương Bạo kích
9. Ngọc Né tránh
10. Ngọc Hút máu
11. Ngọc Giảm hồi máu
12. Ngọc Tốc độ

13.Ngọc Kháng

14.Ngọc Đâm xuyên

15.Ngọc Tốc độ đánh

16.Ngọc Giảm thời gian

17.Ngọc Choáng

18.Ngọc Thường

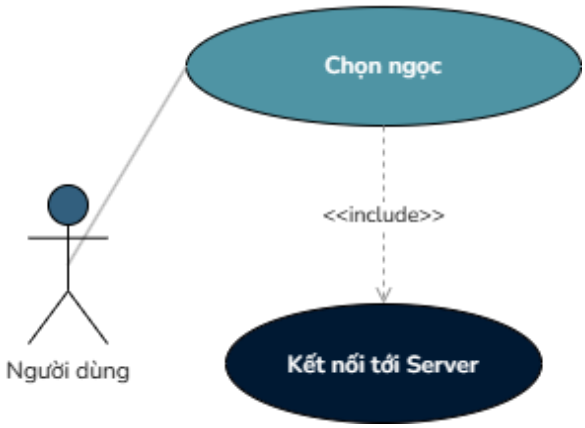
Tiền điều kiện:

- Kết nối mạng ổn định để tìm và vào phòng.
- Người dùng đã đặt tên cho nhân vật

Yêu cầu đặc biệt: Không có.

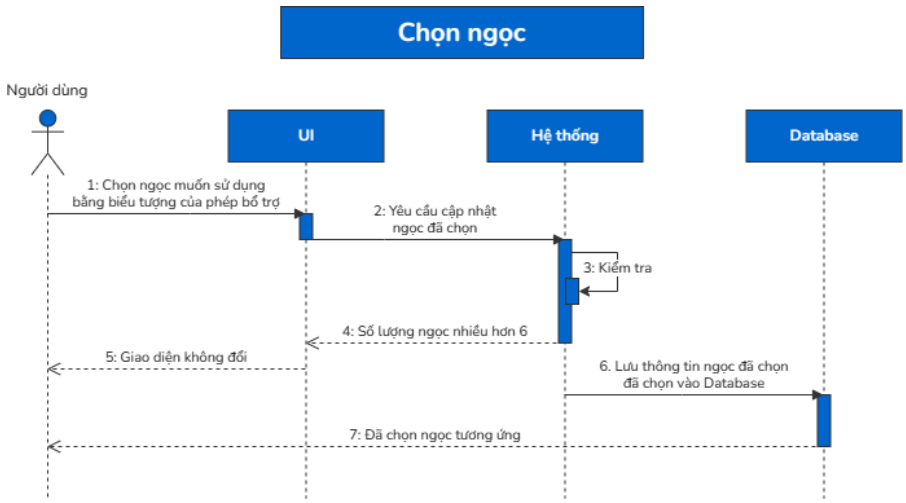
Hậu điều kiện: Người chơi đã chọn được tối đa 6 loại ngọc và các chỉ số của nhân vật đã được tăng cường tương ứng.

a. Biểu đồ Usecase



Hình 17 Usecase Chọn ngọc

b. Sơ đồ tuần tự



Hình 18 Sơ đồ tuần tự Chọn ngọc

2.2.3.6. Chức năng Cài đặt

a. Đặc tả chức năng

Tác nhân: Người dùng.

Mô tả chức năng: Chức năng "Cài đặt" cho phép người chơi tinh chỉnh các thiết lập liên quan đến trải nghiệm chơi game theo ý muốn của mình.

Danh sách các thiết lập:

- **Độ phân giải:** Cho phép người chơi chọn độ phân giải màn hình phù hợp với thiết bị của họ.
- **Toàn màn hình:** Cho phép chuyển đổi giữa chế độ toàn màn hình và chế độ cửa sổ.
- **Ẩn tên người chơi:** Tùy chọn ẩn tên của người chơi trên màn hình.
- **Ẩn tên nhân vật:** Tùy chọn ẩn tên của nhân vật trên màn hình.
- **Ẩn sát thương:** Tùy chọn ẩn hiệu ứng số sát thương gây ra khi trong trận.
- **Ẩn tin nhắn:** Tắt giao diện tin nhắn trong game.
- **Nhạc nền:** Cho phép người chơi điều chỉnh âm lượng của nhạc nền.
- **Nhạc hiệu ứng:** Cho phép người chơi điều chỉnh âm lượng của nhạc hiệu ứng trong game.

- **Nhạc giọng nói:** Cho phép người chơi điều chỉnh âm lượng của nhạc giọng nói trong game.
- **Điều chỉnh phím di chuyển:** Cho phép người chơi tùy chỉnh phím di chuyển để di chuyển nhân vật trong trận đấu.

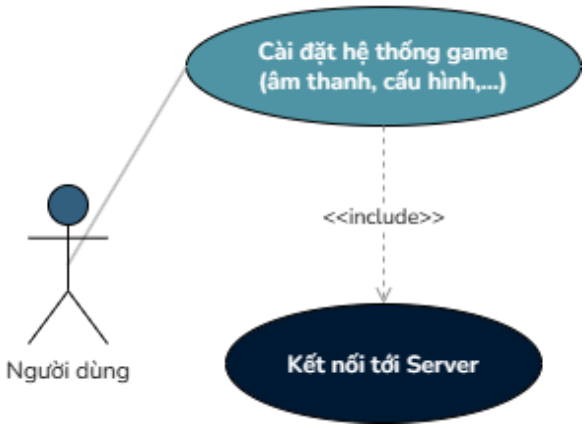
Tiền điều kiện:

- Kết nối mạng ổn định để tìm và vào phòng.
- Người dùng đã đặt tên cho nhân vật

Yêu cầu đặc biệt: Không có.

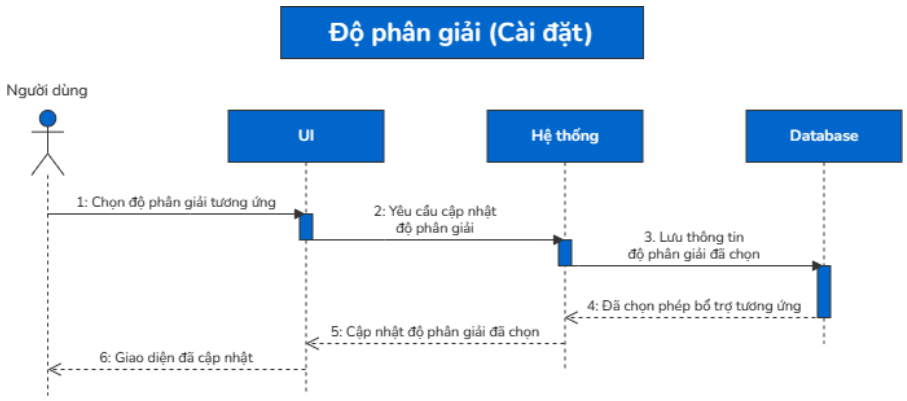
Hậu điều kiện: Các thiết lập cài đặt được lưu và áp dụng trực tiếp.

b. Biểu đồ Usecase

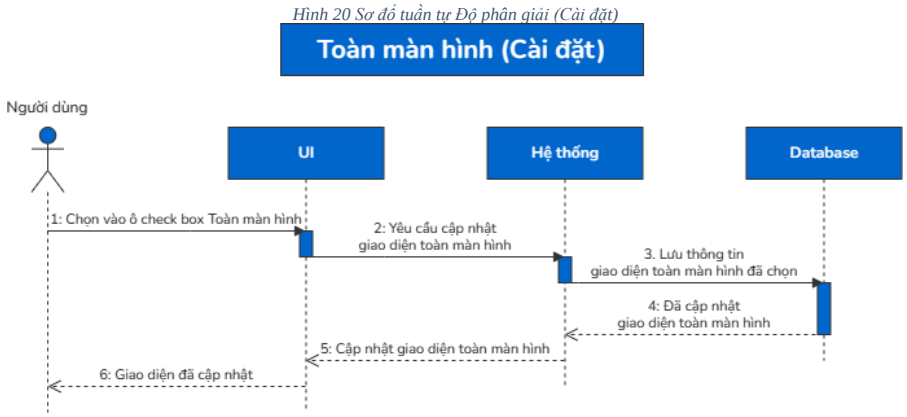


Hình 19 Usecase Cài đặt hệ thống game

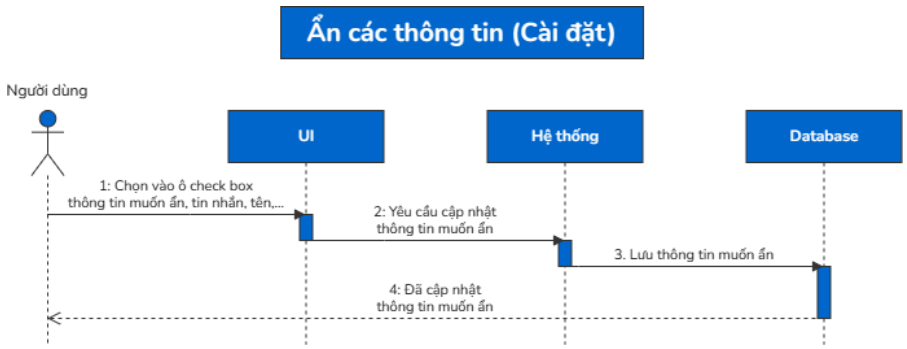
c. Sơ đồ tuần tự



Hình 20 Sơ đồ tuần tự Độ phân giải (Cài đặt)



Hình 21 Sơ đồ tuần tự Toàn màn hình (Cài đặt)



Hình 22 Sơ đồ tuần tự Ẩn các thông tin (Cài đặt)

2.2.3.7. Chức năng Tạo phòng đấu

a. Đặc tả chức năng

Tác nhân: Người dùng.

Mô tả chức năng: Chức năng Tạo phòng đấu cho phép người chơi tạo ra một phòng chơi mới để mời bạn bè hoặc người chơi khác tham gia vào trận đấu. Sau khi chọn tạo phòng, người chơi hiện tại sẽ được chuyển đến một giao diện "Phòng đấu" mới, đóng vai trò là chủ phòng. Mỗi "Phòng đấu" sẽ được gán một ID riêng biệt để phân biệt với các phòng khác.

Chủ phòng sẽ có 2 đặc quyền là riêng biệt:

- Kick người chơi khác ra khỏi phòng của mình.
- Vào trận đấu.
- Sẽ hiển thị chữ "Chủ phòng" ở nhân vật của họ.

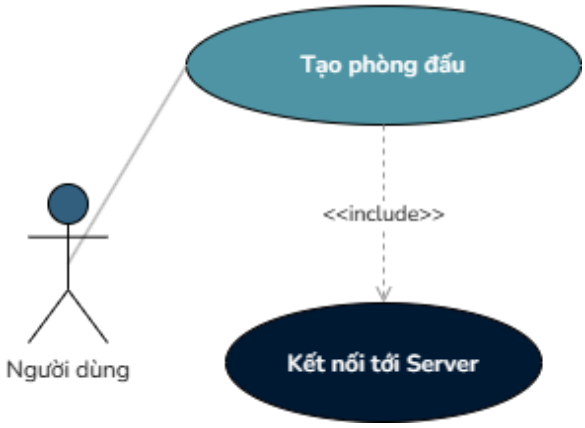
Tiền điều kiện:

- Kết nối mạng ổn định.
- Người dùng đã đặt tên cho nhân vật.

Yêu cầu đặc biệt: Không có.

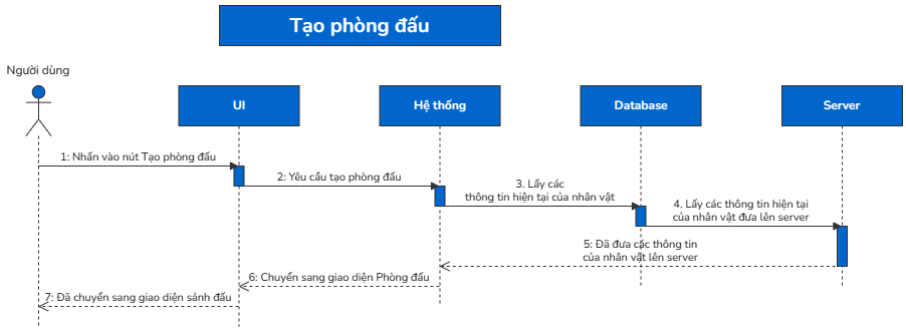
Hậu điều kiện: Khi phòng đấu được tạo thành công và người chơi trở thành chủ phòng, họ có thể quản lý phòng và khởi đầu trận đấu một cách thuận tiện và linh hoạt.

b. Biểu đồ Usecase



Hình 23 Usecase Tạo phòng đấu

c. Sơ đồ tuần tự



Hình 24 Sơ đồ tuần tự Tạo phòng đấu

2.2.3.8. Chức năng Tìm phòng đấu

a. Đặc tả chức năng

Tác nhân: Người dùng.

Mô tả chức năng: Chức năng Tìm phòng đấu cho phép người chơi khác vào phòng chơi thông qua ba cách khác nhau:

Chọn phòng từ danh sách phòng có sẵn:

- Người chơi có thể xem danh sách các phòng đấu hiện có và chọn phòng mà họ muốn tham gia.
- Danh sách các phòng có thể hiển thị các thông tin như tên phòng, số lượng người chơi hiện tại và tối đa, cũng như thông tin về người chủ phòng và cài đặt của phòng.
- Người chơi chọn một phòng từ danh sách và thực hiện thao tác để vào phòng.

Nhập ID phòng:

- Người chơi có thể nhập ID phòng mà người chủ phòng đã gửi cho họ.
- ID phòng là một chuỗi ký tự đặc biệt để phân biệt các phòng, mà người chủ phòng sẽ cung cấp cho người chơi khác.
- Sau khi nhập ID phòng, người chơi có thể xác nhận để vào phòng.

Tìm phòng ngẫu nhiên:

- Người chơi có thể chọn tùy chọn "Tìm phòng ngẫu nhiên" để tự động tham gia vào một phòng đấu ngẫu nhiên từ danh sách các phòng trống.
- Hệ thống sẽ chọn ngẫu nhiên một phòng trống và chuyển người chơi vào phòng đó một cách tự động.

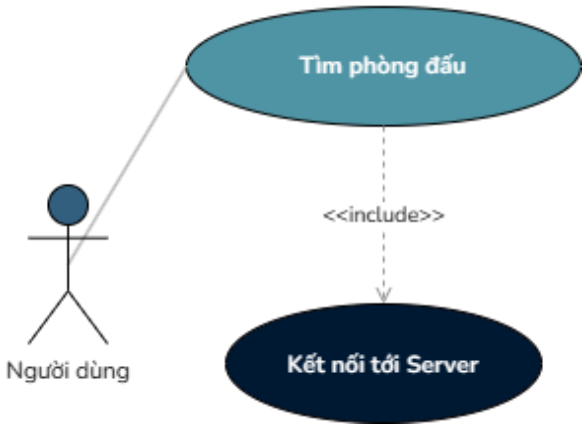
Tiền điều kiện:

- Kết nối mạng ổn định để tìm và vào phòng.
- Người dùng đã đặt tên cho nhân vật

Yêu cầu đặc biệt: Phòng đã chọn vẫn còn tồn tại.

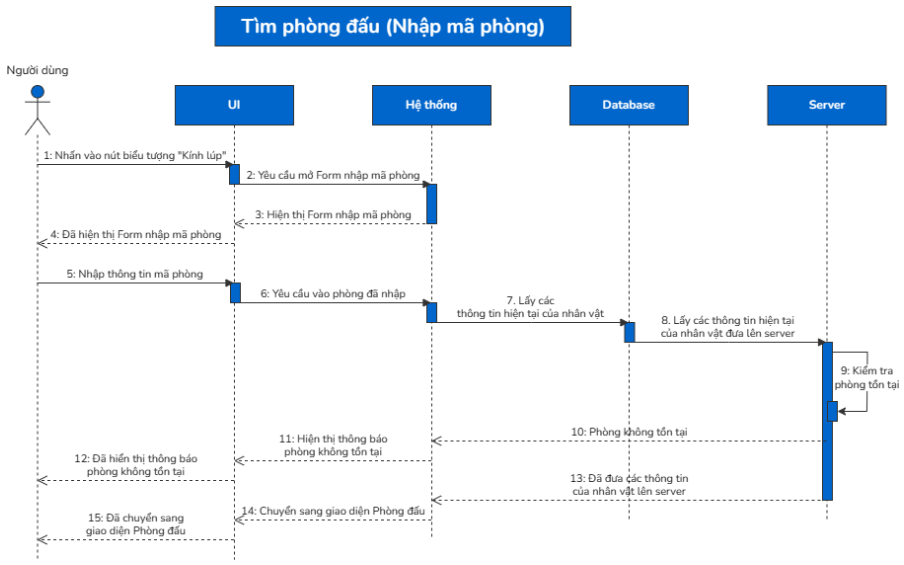
Hậu điều kiện: Khi thực hiện thành công chức năng, người chơi sẽ được chuyển đến phòng đấu mà họ đã chọn hoặc được tự động chuyển đến phòng đấu ngẫu nhiên để tham gia vào trận đấu.

b. Biểu đồ Usecase

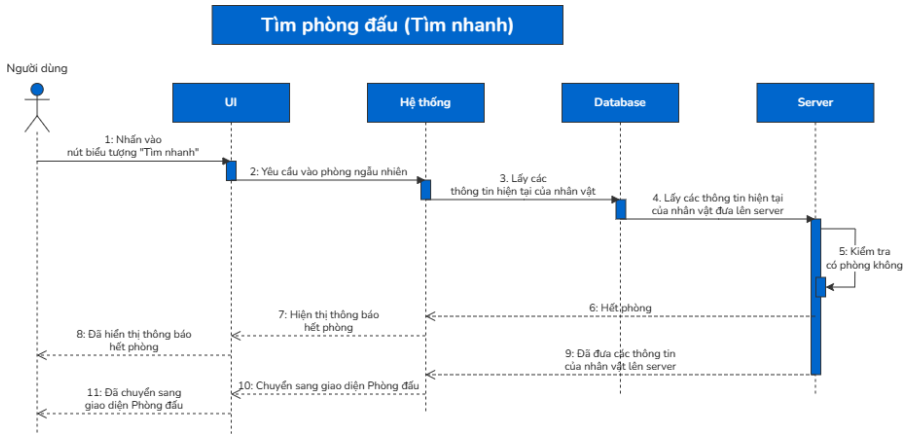


Hình 25 Tìm phòng đấu

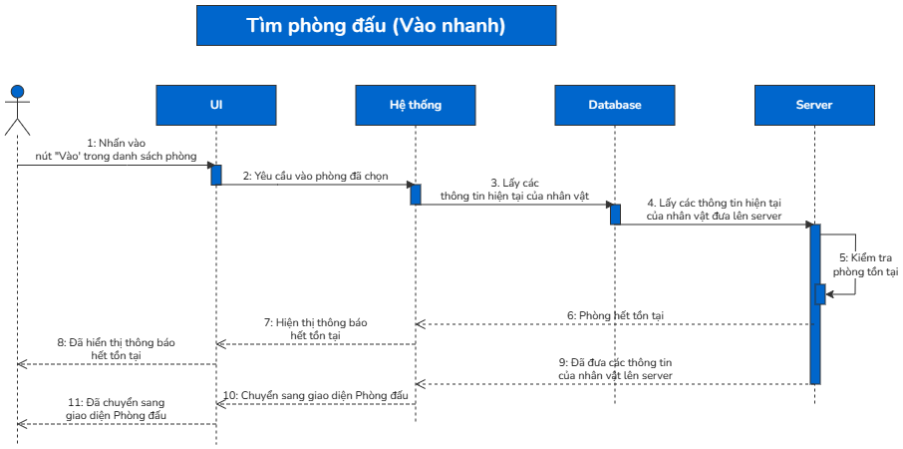
c. Sơ đồ tuần tự



Hình 26 Sơ đồ tuần tự Tìm phòng (Nhập mã phòng)



Hình 27 Sơ đồ tuần tự Tìm phòng (Tìm nhanh)



Hình 28 Sơ đồ tuần tự Tìm phòng (Vào nhanh)

2.2.3.9. Chức năng Bắt đầu trận đấu

a. Đặc tả chức năng

Tác nhân: Người dùng.

Mô tả chức năng: Chức năng "Bắt đầu trận đấu" cho phép người chơi khởi đầu trận đấu sau khi đã sẵn sàng trong phòng đấu.

Các hoạt động trong chức năng:

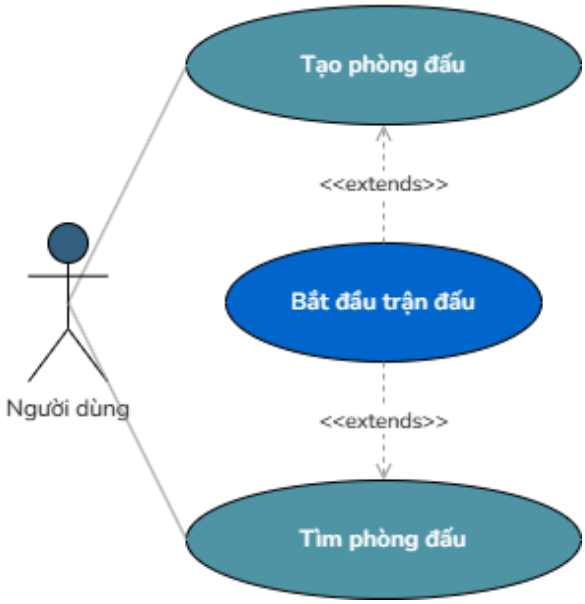
1. **Kiểm tra sẵn sàng:** Hệ thống sẽ kiểm tra xem đã đủ số lượng người chơi có trong phòng hay chưa.
2. **Bắt đầu trận đấu:** Sau khi đã đủ số lượng người trong phòng, người chủ phòng sẽ có thể bắt đầu trận đấu.
3. **Chuyển sang màn hình trận đấu:** Hệ thống sẽ chuyển tất cả người chơi sang màn hình trận đấu để bắt đầu thực hiện các nhiệm vụ và hoạt động trong trận đấu.

Tiền điều kiện: Phòng đó phải đủ số lượng người chơi.

Yêu cầu đặc biệt: Chỉ người chủ phòng mới có thể bắt đầu trận đấu.

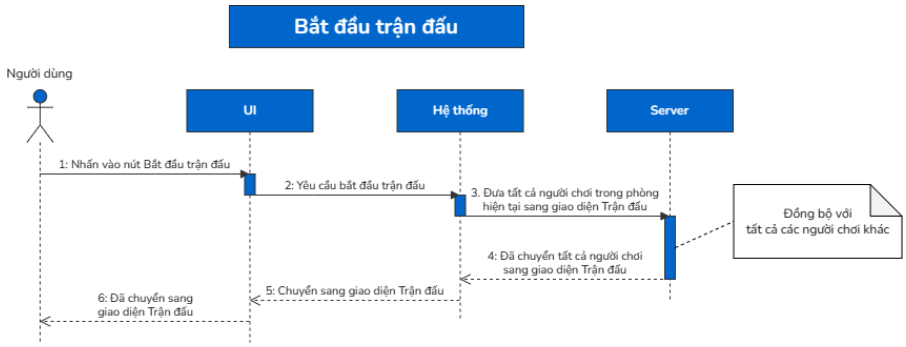
Hậu điều kiện: Trận đấu bắt đầu và tất cả người chơi được chuyển sang màn hình trận đấu để tham gia vào trận đấu.

b. Biểu đồ Usecase



Hình 29 Usecase Bắt đầu trận đấu

c. Sơ đồ tuần tự



Hình 30 Sơ đồ tuần tự Bắt đầu trận đấu

2.2.3.10. Các chức năng trong Trận đấu

a. Đặc tả chức năng

Tác nhân: Người dùng.

Mô tả chức năng: Chức năng "Trận đấu" cho phép hai người chơi tham gia vào một trận đấu, sử dụng các kỹ năng, di chuyển, sử dụng phép bổ trợ và cập nhật các chỉ số mà họ đã chọn từ ngọc.

Các hoạt động trong chức năng:

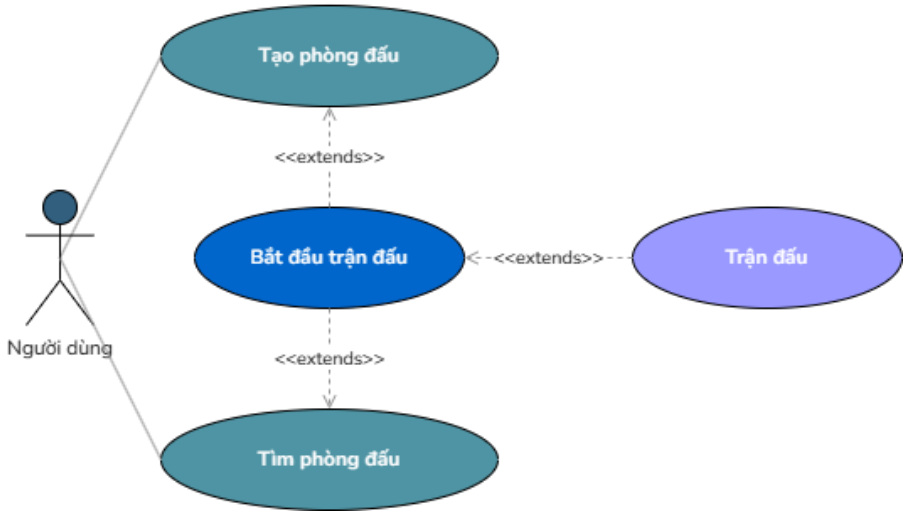
1. **Sử dụng kỹ năng:** Người chơi có thể sử dụng các kỹ năng được gán cho nhân vật của mình để tấn công hoặc phòng thủ.
2. **Di chuyển:** Người chơi có thể di chuyển nhân vật của mình trong không gian trận đấu để tránh các đòn tấn công hoặc tìm cơ hội tấn công.
3. **Sử dụng phép bổ trợ:** Người chơi có thể sử dụng các phép bổ trợ đã chọn để cải thiện hiệu suất của nhân vật trong trận đấu.
4. **Cập nhật các chỉ số từ ngọc:** Các chỉ số mà người chơi đã cài đặt từ ngọc sẽ được cập nhật và ảnh hưởng đến hiệu suất của nhân vật trong trận đấu.
5. **Sử dụng chức năng từ cài đặt:** Các chức năng được cài đặt như ẩn tên người chơi, ẩn sát thương, và các thiết lập khác sẽ được áp dụng trong trận đấu.
6. **Hiển thị FPS, Ping, thời gian trận đấu:** Thông tin về FPS (khung hình mỗi giây), Ping (độ trễ mạng) và thời gian trận đấu sẽ được hiển thị để người chơi có thể theo dõi và đánh giá hiệu suất trận đấu.
7. **Kết thúc trận đấu:** Nếu một trong hai người chơi hết máu trước, họ sẽ thua và trận đấu sẽ kết thúc. Người thua sẽ được chuyển về lại màn hình chính để chọn trận đấu mới hoặc thực hiện các hoạt động khác.

Tiền điều kiện: Hai người chơi đã sẵn sàng và bắt đầu trận đấu.

Yêu cầu đặc biệt: Không có.

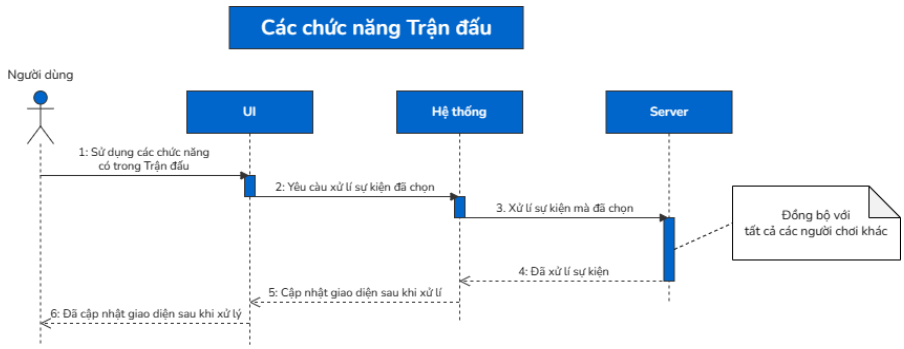
Hậu điều kiện: Trận đấu kết thúc và người chơi được chuyển về lại màn hình chính.

b. Biểu đồ Usecase



Hình 31 Usecase Trận đấu

c. Sơ đồ tuần tự



Hình 32 Sơ đồ tuần tự Các chức năng trong trận đấu

2.2.3.11. Chức năng Nhắn tin

a. Đặc tả chức năng

Tác nhân: Người dùng.

Mô tả chức năng: Chức năng "Nhắn tin" cho phép người chơi giao tiếp với nhau trong trận đấu hoặc trong sảnh đấu.

Các hoạt động trong chức năng:

1. **Gửi tin nhắn:** Người chơi có thể gửi tin nhắn văn bản cho người chơi khác trong cùng một trận đấu hoặc sảnh đấu.

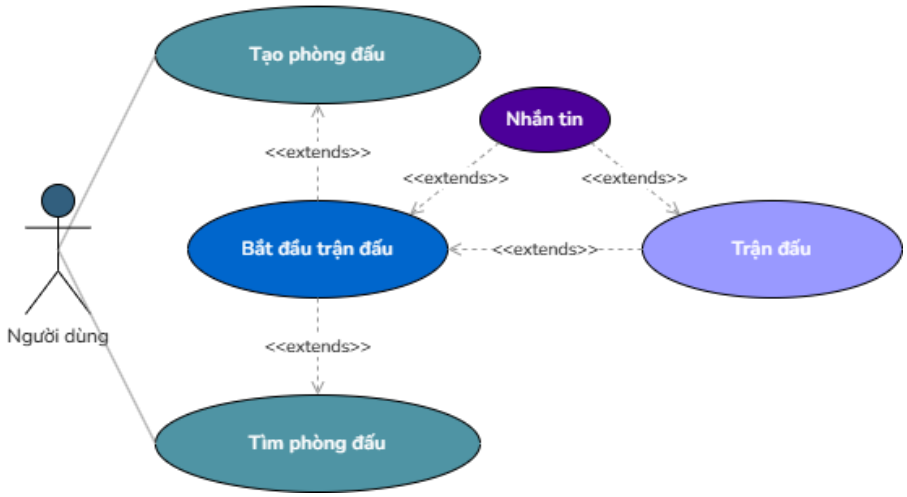
2. **Nhận tin nhắn:** Người chơi sẽ nhận và hiển thị các tin nhắn từ người chơi khác trong cùng một trận đấu hoặc sảnh đấu.

Tiền điều kiện: Người chơi đã tham gia vào trận đấu hoặc sảnh đấu.

Yêu cầu đặc biệt: Không có.

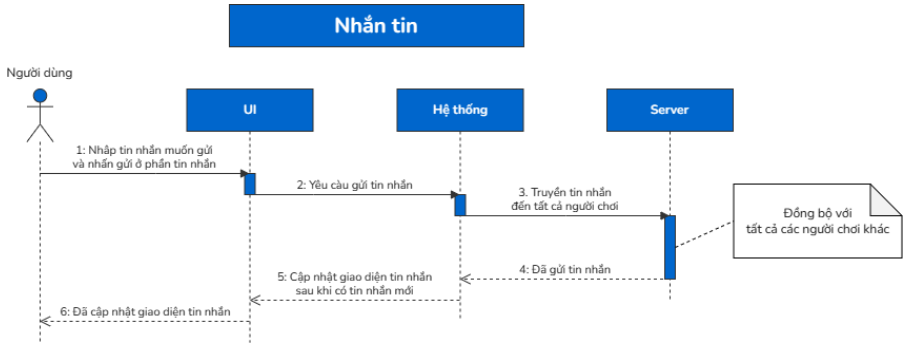
Hậu điều kiện: Tin nhắn đã được gửi và nhận thành công, góp phần tạo ra một môi trường giao tiếp tích cực và thoải mái cho người chơi.

b. Biểu đồ Usecase



Hình 33 Usecase Nhắn tin

c. Sơ đồ tuần tự



Hình 34 Sơ đồ tuần tự nhắn tin

2.2.4. Thiết kế Hệ thống

2.2.4.1. Thiết kế Giao diện

a. Ngôn ngữ thiết kế

Ngôn ngữ thiết kế UI của Unity chủ yếu là ngôn ngữ C# (C Sharp), một ngôn ngữ lập trình được sử dụng rộng rãi trong việc phát triển trò chơi và ứng dụng trên nền tảng Unity. Tuy nhiên, khi thiết kế UI trong Unity, chúng ta thường không phải viết mã từ đầu mà thay vào đó sử dụng Editor của Unity để kéo và thả các phần tử UI và thiết lập các thuộc tính cần thiết. Unity cung cấp một số thành phần UI được xây dựng sẵn như Button, Text, Image, Slider, và nhiều thành phần khác, giúp việc thiết kế UI trở nên dễ dàng hơn.

Để tạo ra các tương tác và chức năng cho UI, có thể sử dụng mã C# để xử lý các sự kiện như nhấn nút, kéo thả, hoặc nhập liệu từ người dùng. Unity cung cấp các hàm và sự kiện sẵn có để có thể kết nối các hành động UI với mã của mình.

Ví dụ, để xử lý sự kiện khi người dùng nhấn vào một nút trong UI, có thể sử dụng các sự kiện như ***onClick*** hoặc sử dụng các phương thức của Unity như ***Button.onClick.AddListener()*** để thêm hành động vào nút đó.

Dưới đây là một ví dụ đơn giản về cách xử lý sự kiện nhấn nút trong Unity bằng C#:

```
using UnityEngine;
using UnityEngine.UI;

public class ButtonHandler : MonoBehaviour
{
    // Reference to the button
    public Button myButton;

    void Start()
    {
        // Add listener for button click event
        myButton.onClick.AddListener(OnClick);
    }

    // Function to handle button click
    void OnClick()
    {
        Debug.Log("Button Clicked!");
```

```
        // Add your desired functionality here
    }
}
```

Bảng 1 Ví dụ về Unity

b. Phong cách thiết kế

WibuPixel được thiết thiết theo phong cách Pixel. Phong cách thiết kế Pixel là một lựa chọn tuyệt vời cho trò chơi, mang lại một cảm giác retro, độc đáo và gợi nhớ những kỷ niệm về thời kỳ game 8-bit và 16-bit. Với phong cách này, các đồ họa sẽ được tạo ra bằng cách sử dụng các hình ảnh pixelated, có độ phân giải thấp, tạo nên một diện mạo cổ điển và đầy cảm xúc.

Các ưu điểm của phong cách thiết kế Pixel bao gồm:

- **Sự độc đáo:** Pixel art mang lại sự khác biệt và độc đáo cho trò chơi, giúp nó nổi bật giữa các tựa game khác.
- **Gợi nhớ ký ức:** Phong cách này thường gợi nhớ về những kỷ niệm tuổi thơ với các trò chơi cổ điển, tạo ra một liên kết tinh thần với người chơi.
- **Tính nhẹ nhàng:** Các hình ảnh pixel thường có kích thước nhỏ và đơn giản, giúp giảm thiểu tải trọng đồ họa trên hệ thống và tối ưu hóa hiệu suất trò chơi.
- **Dễ dàng tạo ra:** Pixel art không đòi hỏi các kỹ năng đồ họa cao cấp, nên nó có thể được tạo ra một cách dễ dàng bởi các nhà phát triển độc lập hoặc các nhóm phát triển nhỏ.

Với sự kết hợp của phong cách thiết kế Pixel và gameplay hấp dẫn, WibuPixel sẽ có sức hút đặc biệt và tạo ra một trải nghiệm độc đáo và thú vị cho người chơi.

2.2.4.2. Thiết kế Cơ sở dữ liệu

Trong quá trình phát triển trò chơi, việc lưu trữ dữ liệu là một phần quan trọng để đảm bảo trải nghiệm người chơi được cải thiện và ổn định. Tuy nhiên, với WibuPixel, không có quá nhiều trường dữ liệu cần lưu trữ, và do đó không cần thiết kế một cơ sở dữ liệu phức tạp. Thay vào đó, WibuPixel có thể tận dụng tính linh hoạt và tiện ích của Player Prefs trong Unity.

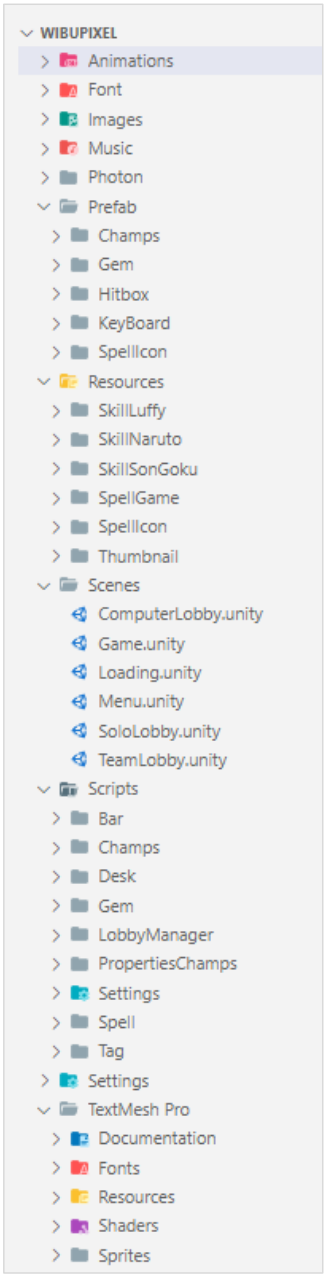
Player Prefs là một cơ chế đơn giản để lưu trữ và truy cập các cặp key-value với số lượng dữ liệu nhỏ. Với tính linh hoạt và đơn giản này, có thể lưu trữ các thông tin quan trọng như cài đặt người chơi, tiến trình của họ trong trò chơi và các thông tin tùy chỉnh khác mà không cần phải triển khai một hệ thống lưu trữ dữ liệu phức tạp.

Bằng cách sử dụng Player Prefs, có thể tiết kiệm thời gian và công sức trong việc xây dựng và duy trì một cơ sở dữ liệu lớn, đồng thời vẫn đảm bảo rằng dữ liệu của người chơi được lưu trữ một cách an toàn và hiệu quả. Điều này cũng giúp tập trung hơn vào việc phát triển các tính năng và nội dung của trò chơi mà không phải lo lắng về vấn đề lưu trữ dữ liệu.

Vì vậy, trong WibuPixel, đã sử dụng Player Prefs để lưu trữ các thông tin cần thiết của người chơi một cách tiện lợi và hiệu quả, đồng thời tối ưu hóa quá trình phát triển và duy trì trò chơi.

CHƯƠNG 3. XÂY DỰNG PHẦN MỀM

3.1. Cấu trúc dự án



Hình 35 Cấu trúc dự án

Chú thích nội dung các thư mục:

- **Animations:** Thư mục chứa các tệp tin hoạt hình, bao gồm các animation clips được sử dụng trong trò chơi.
- **Font:** Thư mục chứa các tệp tin font chữ được sử dụng để hiển thị văn bản trong trò chơi.
- **Images:** Thư mục chứa các tệp tin hình ảnh được sử dụng trong trò chơi, bao gồm các hình ảnh nền, hình ảnh nhân vật, và các hình ảnh khác liên quan.
- **Music:** Thư mục chứa các tệp tin âm nhạc và nhạc nền được sử dụng trong trò chơi.
- **Photon:** Thư mục chứa các tệp tin và tài nguyên liên quan đến việc tích hợp và sử dụng Photon, một nền tảng dịch vụ mạng đa nền tảng cho trò chơi.
- **Prefab:** Thư mục chứa các Prefab, là các đối tượng đã được thiết kế trước và có thể tái sử dụng trong trò chơi.
- **Resources:** Thư mục này thường chứa các tài nguyên không thay đổi và được sử dụng trong trò chơi, nhưng không được tải vào bộ nhớ khi khởi động.
- **Scripts:** Thư mục chứa các tệp tin mã nguồn (scripts) của trò chơi, bao gồm các tập lệnh được viết bằng ngôn ngữ lập trình như C# để điều khiển các hành vi và chức năng trong trò chơi.
- **Settings:** Thư mục chứa các tệp tin cài đặt và cấu hình của trò chơi, bao gồm các tùy chọn về âm thanh, đồ họa, điều khiển và các thiết lập khác.
- **TextMesh Pro:** Thư mục chứa các tệp tin và tài nguyên liên quan đến việc sử dụng TextMesh Pro, một công cụ mạnh mẽ cho việc hiển thị văn bản trong trò chơi với nhiều tùy chọn định dạng và hiệu ứng.

3.2. Các màn hình chức năng

3.2.1. Website của Game

Vì trò chơi không được phát hành trên bất kỳ cửa hàng nào như App Store, CH Play hay Steam, việc có một trang web riêng để truy cập và tải trò chơi là cực kỳ quan trọng. Trang web này không chỉ cung cấp một nơi để người chơi tìm hiểu về

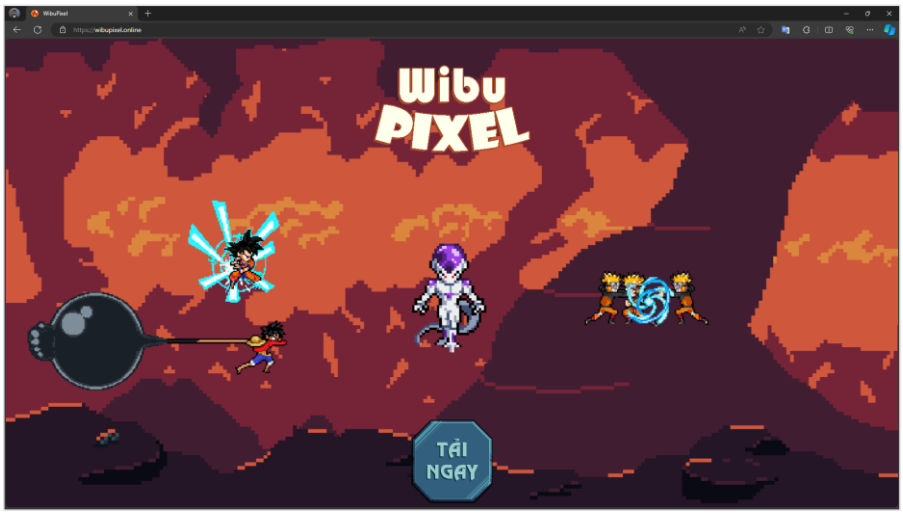
Commented [NTM10]: Đã suy nghĩ để đăng lên các cửa hàng chưa? Ưu điểm? Khuyết điểm

Commented [PT11R10]: Game của em là game trên PC, chỉ có thể chơi được trên PC vì cần có nhiều thao tác không thuận tiện cho việc chơi trên điện thoại, nên không thể up lên App Store và Ch Play được. Còn trên Steam thì phí để up lên đó khá cao ~2tr/1 game và Steam kiểm duyệt rất cao. Với lại 1 phần em làm Game này chủ yếu là để học kỹ năng làm game online, làm game để giải trí chứ không có nhu cầu kiếm tiền từ Game này nên em sẽ không up lên các cửa hàng.

Commented [PT12R10]:

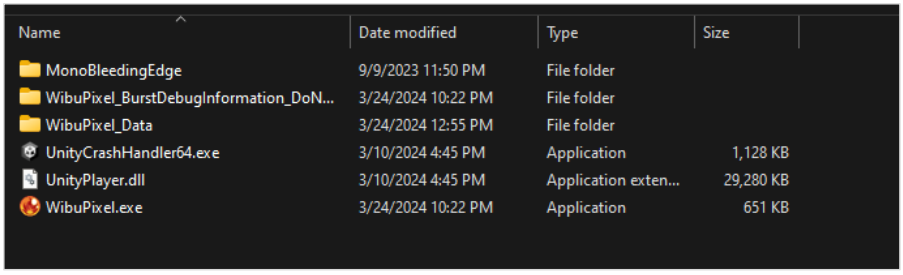
trò chơi và tải xuống, mà còn là một cơ hội để tạo ra sự hiện diện trực tuyến cho dự án.

Website game *wibupixel.online* sẽ không chỉ là nơi để người chơi tải trò chơi xuống, mà còn là một trung tâm truy cập thông tin và cộng đồng cho trò chơi.



Hình 36 wibupixel.online

Sau khi tải game xuống sẽ có các thư mục của game như hình. Để bắt đầu game, chọn tệp WibuPixel.exe và chạy tệp.



Hình 37 File Game sau khi tải về

3.2.1. Kết nối với server

Sau khi khởi động tệp, game sẽ trực tiếp kết nối người chơi lên server của game.

Hình 38 Giao diện Loading

3.2.2. Trang chủ

Khi kết nối với server thành công, người chơi sẽ được đưa đến giao diện Trang chủ của game. Vì game được lấy ý tưởng là một câu chuyện các nhân vật trong manga nên giao diện của game sẽ xoay quanh chủ đề 1 quyển sách.

Hình 39 Giao diện sau khi Loading thành công

Sau khi quyển sách mở ra sẽ chuyển sang giao diện trang chủ.

Hình 40 Giao diện Trang chủ

Giải thích chức năng có trong giao diện

- **Tên của người chơi**

Hình 41 Giao diện tên nhân vật

Khu vực này sẽ hiện thị tên của nhân vật, như trong hình chúng ta có thể thấy tên của người chơi là "test". Nếu họ không thích tên này nữa có thể nhấn vào biểu tượng "Bút chì" bên cạnh để đổi tên nhân vật.

- **Đổi tên nhân vật**

Commented [NTM13]: Chụp hình icon

Commented [PT14R13]: Đã thêm icon

Commented [PT15R13]:

Hình 42 Giao diện đổi tên nhan vật

- **Thông tin nhân vật**

Hình 43 Giao diện thông tin nhân vật

Khu vực này sẽ hiển thị thông tin hiện tại của nhân vật như là kỹ năng, chỉ số của nhân vật, phép bổ trợ,…

- **Danh sách phòng đấu**

Hình 44 Giao diện danh sách phòng

Khu vực này sẽ chưa các phòng đấu hiện đang tồn tại trong server. Người chơi có thể tìm kiếm phòng nhanhh bằng các cách

- **Tất cả:** Tất cả các phòng
- **Solo:** Chỉ tìm phòng 1vs1
- **Team:** Chỉ tìm phòng 2vs2

- **Tạo, tìm phòng đấu**

Hình 45 Giao diện chức năng Tạo, Tìm phòng

Khu vực này sẽ hiện thị các chức năng tìm phòng và tạo phòng.

- **Đơn:** Chế độ 1vs1
- **Đôi:** Chế độ 2vs2
- **Máy:** Chế độ luyện tập
- **Kính lúp:** Tìm phòng đấu bằng mã phòng

Khi nhấn vào button có biểu tượng  "Kính lúp". Giao diện sẽ hiện thị lên một Form giúp người dùng nhập mã phòng.

Commented [NTM16]: Chụp hình icon

Commented [PT17R16]: Đã thêm Icon

Commented [PT18R16]:

Hình 46 Giao diện Tìm phòng theo mã

3.2.3. Chọn nhân vật

Hình 47 Giao diện Chọn nhân vật

Ở trang bên trái, khi người dùng muốn đổi nhân vật có thể click vào button "trái" hoặc "phải" để đổi nhân vật muốn chơi.

Ở trang bên phải hiển thị thông tin kỹ năng hiện tại của nhân vật, bao gồm: tên kỹ năng, mô tả về kỹ năng, thời gian hồi chiêu, lượng năng lượng tiêu thụ và biểu tượng của kỹ năng.

3.2.4. Chọn phép bổ trợ

Hình 48 Giao diện chọn phép bổ trợ

Ở trang bên phải sẽ hiển thị các thông tin về tất cả phép bổ trợ, bao gồm: tên phép bổ trợ, mô tả về phép bổ trợ, thời gian hồi chiêu và biểu tượng của phép bổ trợ. Người dùng muốn chọn phép bổ trợ nào chỉ cần nhấn vào biểu tượng của phép bổ trợ đó.

3.2.5. Chọn ngọc

Hình 49 Giao diện Chọn ngọc

Ở trang bên trái sẽ hiển thị các loại ngọc mà người dùng đã chọn và tổng số chỉ số sẽ cộng vào cho nhân vật của người chơi.

Ở trang bên phải sẽ hiển thị các loại ngọc mà người chơi có thể chọn. Người dùng muốn chọn loại ngọc nào chỉ cần nhấn vào biểu tượng của loại ngọc đó. Người dùng chỉ có thể chọn tối đa 6 loại ngọc.

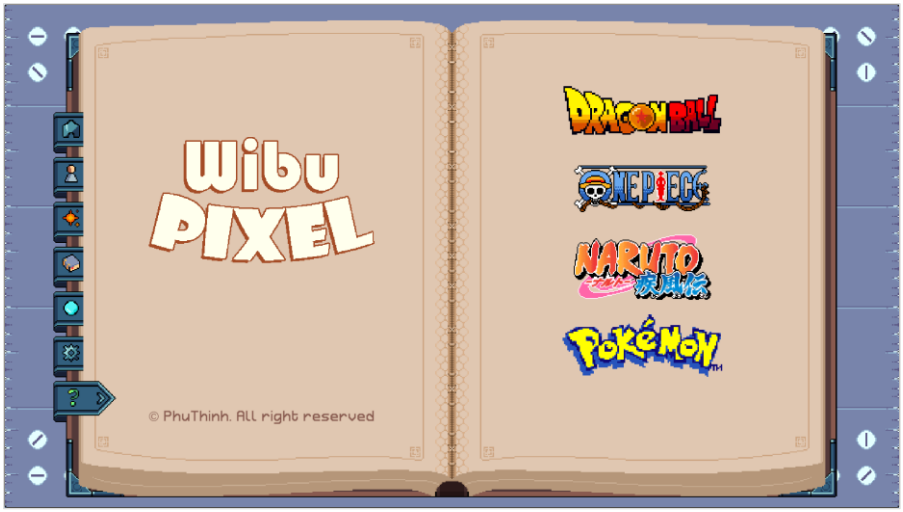
3.2.6. Cài đặt

Hình 50 Giao diện Cài đặt

- Ở trang biên trái sẽ hiển thị các chức năng của chế độ cài đặt. Bao gồm: **Độ phân giải:** Cho phép người chơi chọn độ phân giải màn hình phù hợp với thiết bị của họ.
- **Toàn màn hình:** Cho phép chuyển đổi giữa chế độ toàn màn hình và chế độ cửa sổ.
- **Ẩn tên người chơi:** Tùy chọn ẩn tên của người chơi trên màn hình.
- **Ẩn tên nhân vật:** Tùy chọn ẩn tên của nhân vật trên màn hình.
- **Ẩn sát thương:** Tùy chọn ẩn hiệu ứng số sát thương gây ra khi trong trận.
- **Ẩn tin nhắn:** Tắt giao diện tin nhắn trong game.
- **Nhạc nền:** Cho phép người chơi điều chỉnh âm lượng của nhạc nền.
- **Nhạc hiệu ứng:** Cho phép người chơi điều chỉnh âm lượng của nhạc hiệu ứng trong game.
- **Nhạc giọng nói:** Cho phép người chơi điều chỉnh âm lượng của nhạc giọng nói trong game.

Ở trang bên phải sẽ hiển thị chức năng cho đổi phím. Cho phép người chơi tùy chỉnh phím di chuyển để di chuyển nhân vật trong trận đấu.

3.2.7. Thông tin game



Hình 51 Giao diện thông tin Game

Giao diện này sẽ hiện thị thông tin của game như là tên game, tác giả, các bộ Anime, Manga được sử dụng trong game.

3.2.8. Phòng đấu

Hình 52 Gaio diện phòng đấu

Sau khi tạo phòng hoặc tìm phòng, người chơi sẽ được chuyển sang giao diện Phòng đấu.

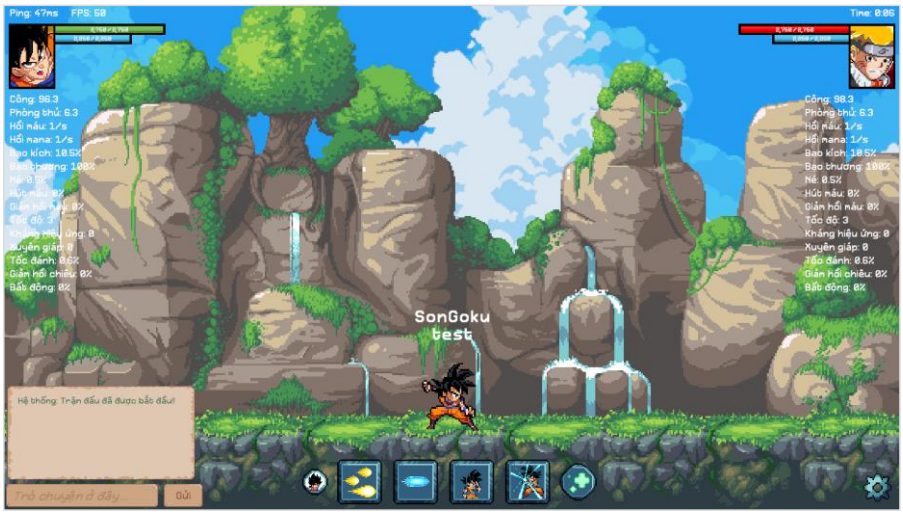
Ở giữa màn hình sẽ tại ra các nhân vật đã vào phòng.

Góc trên bên phải sẽ hiện thị thông tin mã phòng, số người chơi hiện tại đang có trong phòng và một nút "Copy" cho phép người dùng sao chép mã phòng hiện tại để có thể thuận tiện chia sẽ mã phòng cho người chơi khác.

Góc dưới bên trái sẽ là chức năng tin nhắn, truyền tải nội dung mà mình muốn nhắn gửi đến đối thủ của họ.

Góc dưới bên phải sẽ là chức năng vào trận đấu và thoát phòng.

3.2.9. Trận đấu



Hình 53 Giao diện Trận đấu

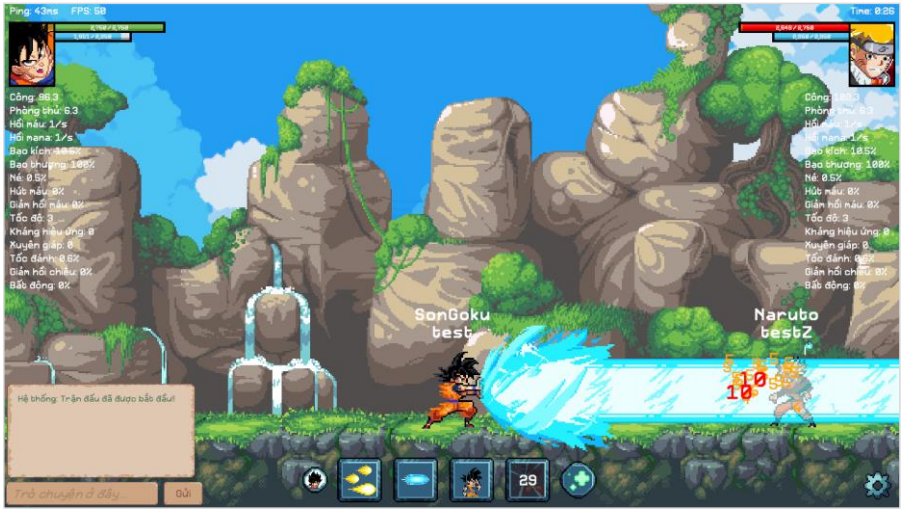
Đây là giao diện khi vào trong trận đấu. Nhân vật của người chơi và đối thủ của họ sẽ xuất hiện.

Góc trên sẽ hiện thị các thông tin hiện tại của game như FPS, Ping, thời gian trận đấu. Đồng thời hiện thị các thông tin của người chơi và đối thủ như là Thanh máu, năng lượng, các chỉ số của nhân vật.

Góc dưới ở giữa sẽ hiện thị các kỹ năng của nhân mà mà người dủng đã chọn. Người dùng có thể xem thông tin chi tiết của kỹ năng bằng cách di chuyển chuột vào biểu tượng của kỹ năng đó.

Tương tự như giao diện Phòng đấu. Góc dưới bên trái sẽ là chức năng tin nhắn, truyền tải nội dung mà mình muốn nhắn gửi đến đối thủ của họ.

Góc dưới bên trái sẽ là chức năng cài đặt của game.



Hình 54 Giao diện sử dụng kỹ năng trong Trận đấu

3.2.10. Cài đặt trong Game

Hình 55 Giao diện Cài đặt trong trận đấu

Tương tự như phần cài đặt ở Trang chủ, phần cài đặt trong Trận đấu cũng tương tự như trong Trang chủ.

3.2.11. Kết thúc trận đấu

Sau khi một trong 2 người chơi hết máu thì giao diện “Bạn Chiến Thắng” và “Bạn Đã Thua” sẽ hiện thị tương ứng.

Nhấn vào nút Thoát để quay trở lại Trang chủ.

Hình 56 Giao diện Bạn Chiến Thắng

Hình 57 Giao diện Bạn Đã Thua

CHƯƠNG 4. KẾT LUẬN

4.1. Kết quả đạt được

Trong quá trình phát triển và hoàn thiện, dự án đã đạt được những thành tựu đáng kể. Dưới đây là những điểm nổi bật về thành tựu của dự án:

Chất lượng tổng thể: Game đã đạt được một mức độ chất lượng ổn định và tốt. Từ giao diện người dùng (UI) đến trải nghiệm người chơi (UX), từ âm thanh đến hiệu suất hoạt động, mọi khía cạnh đều được chăm chút và cải thiện.

Giao diện và trải nghiệm người chơi: UI/UX được thiết kế tỉ mỉ và linh hoạt, tạo ra trải nghiệm thú vị và dễ dàng cho người chơi. Tính tương tác và khả năng điều hướng trong game đã được cải thiện, giúp người chơi tham gia và tận hưởng trò chơi một cách thoải mái.

Hiệu suất và tối ưu hóa: Hiệu suất của trò chơi được tối ưu hóa, đảm bảo trải nghiệm chơi mượt mà và không gặp lag. Việc tối ưu hóa này cải thiện trải nghiệm người chơi trên nhiều loại thiết bị.

Website Game: Việc phát triển và triển khai trang web riêng cho trò chơi đã tạo ra một kênh quảng bá và phân phối hiệu quả. Website cung cấp nền tảng để người chơi tìm hiểu và tải trò chơi, đồng thời tạo ra sự hiện diện trực tuyến cho dự án, thu hút sự chú ý từ cộng đồng game thủ.

Tóm lại, dự án đã đạt được những kết quả tích cực và đáng tự hào. Sự nỗ lực và cam kết trong quá trình phát triển đã mang lại một trò chơi tốt và một nền tảng phân phối đáng tin cậy. Dự án này có tiềm năng phát triển và thu hút người chơi trong tương lai.

4.2. Hạn chế còn tồn đọng

Mặc dù đã đạt được những thành tựu đáng kể, nhưng dự án vẫn còn một số hạn chế cần được giải quyết:

- **Chưa có trên nhiều nền tảng:** Hiện tại, game chưa có mặt trên các cửa hàng game phổ biến như CH Play, Steam, hoặc itch.io, điều này giới hạn khả năng tiếp cận của người chơi.

- **Lỗi trong quá trình chơi:** Do thời gian kiểm thử có hạn, vẫn có thể tồn tại những lỗi hoặc vấn đề trong quá trình chơi mà chưa được phát hiện hoặc xử lý.
- **Tính năng chưa hoàn thiện:** Một số tính năng như nâng cấp kỹ năng, khả năng vào lại phòng sau khi mất kết nối, vẫn chưa được triển khai hoàn chỉnh, cần phải tiếp tục phát triển và hoàn thiện.
- **Thiếu phương tiện kiếm doanh thu:** Hiện tại chưa có phương tiện nào để kiếm doanh thu từ trò chơi, điều này có thể ảnh hưởng đến khả năng duy trì và phát triển của dự án.
- **Bảo mật chưa cao:** Database lưu trữ ở local của người chơi có thể tạo ra rủi ro bảo mật, cần phải xem xét và cải thiện.

4.3. Hướng phát triển trong tương lai

Trong tương lai, để nắm bắt cơ hội và đối mặt với thách thức, dự án có thể phát triển theo các hướng sau:

- **Mở rộng nền tảng:** Đẩy mạnh việc phát triển trên nhiều nền tảng khác nhau như CH Play, Steam, iOS, và web để tăng khả năng tiếp cận và thu hút đa dạng người chơi.
- **Kiểm thử và cải thiện:** Dành thời gian và nguồn lực để kiểm thử kỹ lưỡng hơn, sửa chữa các lỗi và tối ưu hóa trải nghiệm người chơi, từ đó nâng cao chất lượng tổng thể của trò chơi.
- **Hoàn thiện tính năng:** Tiếp tục phát triển và hoàn thiện các tính năng chưa được triển khai hoàn chỉnh như nâng cấp kỹ năng, khả năng vào lại phòng sau khi mất kết nối, để cung cấp trải nghiệm chơi game đầy đủ và mượt mà hơn.
- **Xây dựng mô hình kinh doanh:** Tìm kiếm và phát triển các phương tiện kiếm doanh thu như quảng cáo trong trò chơi, bán vật phẩm ảo, hoặc tích hợp hệ thống trả phí để đảm bảo sự bền vững của dự án.
- **Cải thiện bảo mật:** Nghiên cứu và triển khai các biện pháp bảo mật hiệu quả hơn để đảm bảo an toàn cho dữ liệu người chơi và tránh rủi ro về lỗ hổng bảo mật.

- **Tích hợp tính năng mạng xã hội:** Kết nối với các nền tảng mạng xã hội để người chơi có thể chia sẻ trải nghiệm, mời bạn bè tham gia, và tạo ra một cộng đồng chơi game sôi động.
- **Liên tục cập nhật và bảo trì:** Tiếp tục cập nhật nội dung mới, sửa lỗi, và cải thiện tính năng để duy trì sự hấp dẫn và tạo ra những trải nghiệm mới cho người chơi.

Bằng cách này, dự án có thể phát triển và mở rộng, thu hút đông đảo người chơi và đạt được thành công trong tương lai.

TÀI LIỆU THAM KHẢO

1. **Unity**

https://docs.unity.com/

https://docs.unity3d.com/Manual/index.html

https://docs.unity3d.com/Manual/AssetStore.html

https://docs.unity3d.com/ScriptReference/index.html

2. **Photon**

https://doc.photonengine.com/pun/current/getting-started/pun-intro

https://doc-api.photonengine.com/en/pun/current/index.html

https://doc-api.photonengine.com/en/pun/current/general.html

3. **Dragonball Pixel (old my game)**

https://phuthinh.dev/demo_dragonballpixel

4. **ChatGPT**

https://chat.openai.com/

PHỤ LỤC

1. File code Loading

Gồm các chức năng: Kết nối đến server

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using Photon.Pun;
using Photon.Realtime;
using UnityEngine.SceneManagement;
public class Loadding : MonoBehaviourPunCallbacks
{
    public void Start()
    {
        PhotonNetwork.ConnectUsingSettings();
    }
    public override void OnConnectedToMaster()
    {
        PhotonNetwork.JoinLobby();
    }
    public override void OnJoinedLobby()
    {
        PhotonNetwork.LoadLevel("Menu");
    }
}
```

Bảng 2 File code Loading

2. File code ChampsManager

- Gồm các chức năng:
- Đổi nhân vật
- Chọn nhân vật

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;
using Photon.Pun;
using Photon.Realtime;

public class ChampsManager : MonoBehaviour
{
    private const string SelectedChampKey = "SelectedChamp";
```

```
public GameObject ChampContainer;
public TextMeshProUGUI _champNameText;

public Transform[] champs;

public Button _nextChampButton;
public Button _previousChampButton;

private int currentChampIndex = 0;

private string[] champNames;

private void Start()
{
    champNames = new string[champs.Length];
    for (int i = 0; i < champNames.Length; i++)
    {
        champNames[i] = champs[i].name;
    }

    if (PlayerPrefs.HasKey(SelectedChampKey))
    {
        currentChampIndex = PlayerPrefs.GetInt(SelectedChampKey);
        ChooseChamp();
    }
    else
    {
        ChooseChamp();
    }

    champs = new Transform[ChampContainer.transform.childCount];
    for (int i = 0; i < ChampContainer.transform.childCount; i++)
    {
        champs[i] = ChampContainer.transform.GetChild(i);
        champs[i].gameObject.SetActive(i == currentChampIndex);
    }

    _nextChampButton.onClick.AddListener(ShowNextChamp);
    _previousChampButton.onClick.AddListener(ShowPreviousChamp);
}

public void ShowNextChamp()
{
    champs[currentChampIndex].gameObject.SetActive(false);
    currentChampIndex = (currentChampIndex + 1) % champs.Length;
    champs[currentChampIndex].gameObject.SetActive(true);
    ChooseChamp();
}
```

```
    public void ShowPreviousChamp()
    {
        champs[currentChampIndex].gameObject.SetActive(false);
        currentChampIndex = (currentChampIndex - 1 + champs.Length) % champs.Length;
        champs[currentChampIndex].gameObject.SetActive(true);
        ChooseChamp();
    }

    public void ChooseChamp()
    {
        int selectedChampIndex = currentChampIndex;

        ExitGames.Client.Photon.Hashtable playerProperties = new
ExitGames.Client.Photon.Hashtable();
        playerProperties["SelectedChampIndex"] = selectedChampIndex;
        playerProperties["SelectedChampName"] = champNames[selectedChampIndex];
        PlayerPrefs.SetInt(SelectedChampKey, selectedChampIndex);
        PhotonNetwork.LocalPlayer.SetCustomProperties(playerProperties);
        _champNameText.text = champNames[selectedChampIndex];
    }

}
```

Bảng 3 File code ChampsManager

3. File code Menu

Gồm các chức năng:

- Đặt tên cho nhân vật
- Tạo phòng
- Tìm phòng
- Sắp xếp các phòng đang có theo danh sách

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;
using Photon.Pun;
using Photon.Realtime;
using System.Text;
using System.Linq;

public class Menu : MonoBehaviourPunCallbacks
{
    private const string PlayerNameKey = "PlayerName";
    public TextMeshProUGUI _playerNameText;
```

```
    public Button _editNameButton;
    public TMP_InputField _setNameInput;
    public Button _applyNameButton;
    public Button _closeUiChangeNameButton;
    public TextMeshProUGUI _errorNameText;
    public GameObject _uiChangeName;
    public GameObject _showNameText;
    public GameObject _closeNameUI;
    public Button _CreateSoloLobbyButton;
    public Button _CreateTeamLobbyButton;
    public Button _CreateComputerLobbyButton;
    public Button _joinRandomLobbySoloButton;
    public Button _joinRandomLobbyTeamButton;
    public Button _joinNameRoomButton;
    public Button _closeUiJoinRoomButton;

    public Button _joinLobbyButton;
    public GameObject _uiJoinRoom;

    public Button _sortAllButton;
    public Button _sortSoloButton;
    public Button _sortTeamButton;
    public TMP_InputField _setIdLobbyInput;
    public TextMeshProUGUI _errorJoinNameText;
    public TextMeshProUGUI _errorJoinRoomText;

    private bool isNameValid = false;
    private List<RoomInfo> roomList = new List<RoomInfo>();
    public GameObject roomInfoPrefab;
    public Transform roomListContent;
    private List<GameObject> instantiatedRoomInfos = new List<GameObject>();
    public GameObject _scrollTop;
    public GameObject _scrollBottom;

    public Animator animatorTagHome;
    public Animator animatorTagUser;
    public Animator animatorTagSkill;
    public Animator animatorTagSpell;
    public Animator animatorTagGem;
    public Animator animatorTagSettings;
    public Animator animatorTagInstruct;
    public Animator animatorCloseBook;
    public GameObject[] _changeScene;

    public GameObject _uiHome;
    public GameObject _uiUser;
    public GameObject _uiSkill;
    public GameObject _uiSpell;
    public GameObject _uiGem;
    public GameObject _uiSettings;
```

```
    public GameObject _uiInstruct;

    public GameObject _champ;
    public GameObject _skill;

    private void Start()
    {
        UpdateName();

        _scrollTop.SetActive(false);
        _scrollBottom.SetActive(false);

        _applyNameButton.onClick.AddListener(ApplyName);
        _CreateSoloLobbyButton.onClick.AddListener(OnCreateSoloLobbyButtonClick);
        _CreateTeamLobbyButton.onClick.AddListener(CreateTeamLobbyButton);
        _CreateComputerLobbyButton.onClick.AddListener(OnCreateComputerLobbyButtonClick);
        _joinRandomLobbySoloButton.onClick.AddListener(JoinSoloLobbyButton);
        _joinRandomLobbyTeamButton.onClick.AddListener(JoinTeamLobbyButton);

        _joinLobbyButton.onClick.AddListener(JoinLobby);

        _joinNameRoomButton.onClick.AddListener(() =>
        {
            _uiJoinRoom.SetActive(true);
        });
        _closeUiJoinRoomButton.onClick.AddListener(() =>
        {
            _uiJoinRoom.SetActive(false);
        });
        _editNameButton.onClick.AddListener(() =>
        {
            _uiChangeName.SetActive(true);
        });
        _closeUiChangeNameButton.onClick.AddListener(() =>
        {
            _uiChangeName.SetActive(false);
        });

        _sortAllButton.onClick.AddListener(SortAll);
        _sortSoloButton.onClick.AddListener(SortSolo);
        _sortTeamButton.onClick.AddListener(SortTeam);

        PhotonNetwork.JoinLobby();
        PhotonNetwork.AddCallbackTarget(this);
    }
    private IEnumerator DelayShowChangeName()
    {
        yield return new WaitForSeconds(3.0f);
        _showNameText.SetActive(true);
        _uiChangeName.SetActive(true);
```

```
        _closeNameUI.SetActive(false);
    }
    private void UpdateName()
    {
        if (PlayerPrefs.HasKey(PlayerNameKey))
        {
            string playerName = PlayerPrefs.GetString(PlayerNameKey);
            _setNameInput.text = playerName;
            _playerNameText.text = playerName;
            ApplyName();
        }
        else
        {
            StartCoroutine(DelayShowChangeName());
        }
    }

    private void SortAll()
    {
        foreach (GameObject roomInfo in instantiatedRoomInfos)
        {
            roomInfo.SetActive(true);
        }
    }

    private void SortSolo()
    {
        foreach (GameObject roomInfo in instantiatedRoomInfos)
        {
            if (roomInfo.transform.GetChild(2).GetComponent<TextMeshProUGUI>().text ==
"Solo Room")
            {
                roomInfo.SetActive(true);
            }
            else
            {
                roomInfo.SetActive(false);
            }
        }
    }
    private void SortTeam()
    {
        foreach (GameObject roomInfo in instantiatedRoomInfos)
        {
            if (roomInfo.transform.GetChild(2).GetComponent<TextMeshProUGUI>().text ==
"Team Room")
            {
                roomInfo.SetActive(true);
            }
            else
```

```
            {
                roomInfo.SetActive(false);
            }
        }
    }
    private void ApplyName()
    {
        string playerName = _setNameInput.text;
        if (playerName.Length > 2)
        {
            isNameValid = true;
            _errorNameText.text = "";
            PhotonNetwork.NickName = playerName;
            PlayerPrefs.SetString(PlayerNameKey, playerName);
            _playerNameText.text = playerName;
            _uiChangeName.SetActive(false);
            _showNameText.SetActive(false);
            _closeNameUI.SetActive(true);
        }
        else
        {
            _errorNameText.text = "Tên phải nhiều hơn 2 ký tự!";
            isNameValid = false;
        }
    }
    private void ChangeScene()
    {
        animatorCloseBook.SetBool("isClose", true);
        animatorTagHome.SetBool("isEnd", true);
        animatorTagUser.SetBool("isEnd", true);
        animatorTagSkill.SetBool("isEnd", true);
        animatorTagSpell.SetBool("isEnd", true);
        animatorTagGem.SetBool("isEnd", true);
        animatorTagSettings.SetBool("isEnd", true);
        animatorTagInstruct.SetBool("isEnd", true);
        animatorTagHome.SetBool("isStart", true);
        RandomChangeScene();
    }
    void RandomChangeScene()
    {
        int random = Random.Range(0, _changeScene.Length);
        _changeScene[random].SetActive(true);
    }
    private void HiddenAllUI()
    {
        _uiHome.SetActive(false);
        _uiUser.SetActive(false);
        _uiSkill.SetActive(false);
        _uiSpell.SetActive(false);
        _uiGem.SetActive(false);
```

```
        _uiSettings.SetActive(false);
        _uiInstruct.SetActive(false);
        _champ.SetActive(false);
        _skill.SetActive(false);
    }
    private void OnCreateComputerLobbyButtonClick()
    {
        ChangeScene();
        HiddenAllUI();
        StartCoroutine(CreateComputerLobbyButton());
    }
    private IEnumerator CreateComputerLobbyButton()
    {
        yield return new WaitForSeconds(3.0f);
        System.Random random = new System.Random();
        int randomValueFront = random.Next(0, 999);
        int randomValueBack = random.Next(0, 999);
        string timeNow = System.DateTime.Now.ToString("mmss");
        string lobbyName = randomValueFront.ToString("D3") + timeNow +
randomValueBack.ToString("D3");
        ExitGames.Client.Photon.Hashtable roomProperties = new
ExitGames.Client.Photon.Hashtable { { "Host", PhotonNetwork.NickName } };
        RoomOptions roomOptions = new RoomOptions
        {
            MaxPlayers = 1,
            CustomRoomProperties = roomProperties,
            CustomRoomPropertiesForLobby = new string[] { "Host" }
        };
        PhotonNetwork.CreateRoom(lobbyName, roomOptions);
    }
    private void OnCreateSoloLobbyButtonClick()
    {
        ChangeScene();
        HiddenAllUI();
        StartCoroutine(CreateSoloLobbyButton());
    }

    private IEnumerator CreateSoloLobbyButton()
    {
        yield return new WaitForSeconds(3.0f);
        System.Random random = new System.Random();
        int randomValueFront = random.Next(0, 999);
        int randomValueBack = random.Next(0, 999);
        string timeNow = System.DateTime.Now.ToString("mmss");
        string lobbyName = randomValueFront.ToString("D3") + timeNow +
randomValueBack.ToString("D3");
        ExitGames.Client.Photon.Hashtable roomProperties = new
ExitGames.Client.Photon.Hashtable { { "Host", PhotonNetwork.NickName } };
        RoomOptions roomOptions = new RoomOptions
        {
```

```
            MaxPlayers = 2,
            CustomRoomProperties = roomProperties,
            CustomRoomPropertiesForLobby = new string[] { "Host" }
        };
        PhotonNetwork.CreateRoom(lobbyName, roomOptions);
    }
    private void CreateTeamLobbyButton()
    {
        System.Random random = new System.Random();

        int randomValueFront = random.Next(0, 999);
        int randomValueBack = random.Next(0, 999);

        string timeNow = System.DateTime.Now.ToString("mmss");

        string lobbyName = randomValueFront.ToString("D3") + timeNow +
randomValueBack.ToString("D3");

        ExitGames.Client.Photon.Hashtable roomProperties = new
ExitGames.Client.Photon.Hashtable { { "Host", PhotonNetwork.NickName } };
        RoomOptions roomOptions = new RoomOptions
        {
            MaxPlayers = 4,
            CustomRoomProperties = roomProperties,
            CustomRoomPropertiesForLobby = new string[] { "Host" }
        };

        PhotonNetwork.CreateRoom(lobbyName, roomOptions);
    }

    private void JoinSoloLobbyButton()
    {
        List<RoomInfo> soloRooms = roomList.Where(room => room.MaxPlayers == 2 &&
room.PlayerCount < room.MaxPlayers).ToList();

        if (soloRooms.Count > 0)
        {
            int randomIndex = Random.Range(0, soloRooms.Count);
            RoomInfo selectedRoom = soloRooms[randomIndex];

            JoinRoom(selectedRoom.Name);
        }
        else
        {
            _errorJoinRoomText.text = "Không có phòng Đơn nào trống!";
        }
    }

    private void JoinTeamLobbyButton()
    {
```

```
        List<RoomInfo> teamRooms = roomList.Where(room => room.MaxPlayers == 4 &&
room.PlayerCount < room.MaxPlayers).ToList();

        if (teamRooms.Count > 0)
        {
            int randomIndex = Random.Range(0, teamRooms.Count);
            RoomInfo selectedRoom = teamRooms[randomIndex];

            JoinRoom(selectedRoom.Name);
        }
        else
        {
            _errorJoinRoomText.text = "Không có phòng Đội nào trống!";
        }
    }
    private void JoinLobby()
    {
        if (!isNameValid)
            return;

        string lobbyId = _setIdLobbyInput.text;
        if (string.IsNullOrEmpty(lobbyId))
        {
            _errorJoinNameText.text = "Vui lòng nhập mã phòng!";
            return;
        }

        if (!PhotonNetwork.IsConnectedAndReady)
            return;

        List<RoomInfo> roomChoose = roomList.Where(room => room.Name == lobbyId &&
room.PlayerCount == room.MaxPlayers).ToList();

        if (roomChoose.Count > 0)
        {
            _errorJoinRoomText.text = "Phòng đã đầy!";
            return;
        }

        PhotonNetwork.JoinRoom(lobbyId);
    }

    private void JoinRoom(string roomId)
    {
        if (!isNameValid)
            return;

        string lobbyId = roomId;
        if (string.IsNullOrEmpty(lobbyId))
        {
```

```
            _errorJoinNameText.text = "Vui lòng nhập mã phòng!";
            return;
        }

        if (!PhotonNetwork.IsConnectedAndReady)
            return;

        List<RoomInfo> roomChoose = roomList.Where(room => room.Name == lobbyId &&
room.PlayerCount == room.MaxPlayers).ToList();

        if (roomChoose.Count > 0)
        {
            _errorJoinRoomText.text = "Phòng đã đầy!";
            return;
        }

        PhotonNetwork.JoinRoom(lobbyId);
    }

    public override void OnJoinedRoom()
    {
        if (PhotonNetwork.CurrentRoom.MaxPlayers == 1)
        {
            PhotonNetwork.LoadLevel("ComputerLobby");
        }
        if (PhotonNetwork.CurrentRoom.MaxPlayers == 2)
        {
            PhotonNetwork.LoadLevel("SoloLobby");
        }
        if (PhotonNetwork.CurrentRoom.MaxPlayers == 4)
        {
            PhotonNetwork.LoadLevel("TeamLobby");
        }
    }

    public override void OnRoomListUpdate(List<RoomInfo> roomList)
    {
        foreach (RoomInfo room in roomList)
        {
            bool roomExists = false;
            for (int i = 0; i < this.roomList.Count; i++)
            {
                if (this.roomList[i].Name == room.Name)
                {
                    roomExists = true;
                    this.roomList[i] = room;
                    break;
                }
            }
```

```
            if (!roomExists && room.PlayerCount > 0)
            {
                this.roomList.Add(room);
            }
        }

        foreach (GameObject roomInfo in instantiatedRoomInfos)
        {
            Destroy(roomInfo);
        }
        instantiatedRoomInfos.Clear();

        foreach (RoomInfo room in this.roomList)
        {
            GameObject newRoomInfo = Instantiate(roomInfoPrefab, roomListContent);
            newRoomInfo.SetActive(true);

            TextMeshProUGUI idLobbyText =
newRoomInfo.GetComponentInChildren<TextMeshProUGUI>();
            TextMeshProUGUI nameHostText =
newRoomInfo.transform.GetChild(1).GetComponent<TextMeshProUGUI>();
            TextMeshProUGUI typeText =
newRoomInfo.transform.GetChild(2).GetComponent<TextMeshProUGUI>();
            TextMeshProUGUI activeText =
newRoomInfo.transform.GetChild(3).GetComponent<TextMeshProUGUI>();
            Button joinButton = newRoomInfo.transform.GetChild(4).GetComponent<Button>();

            idLobbyText.text = "Mã: " + room.Name;

            if (room.MaxPlayers == 2)
            {
                typeText.text = "Phòng Đơn";
            }
            else if (room.MaxPlayers == 4)
            {
                typeText.text = "Phòng Đội";
            }
            else
            {
                typeText.text = "";
            }

            if (room.PlayerCount >= room.MaxPlayers)
            {
                activeText.text = $"{room.PlayerCount}/{room.MaxPlayers}(Max)";
            }
            else
            {
                activeText.text = $"{room.PlayerCount}/{room.MaxPlayers}";
            }
```

```
            if (room.CustomProperties.ContainsKey("Host"))
            {
                nameHostText.text = "Chủ phòng: " +
room.CustomProperties["Host"].ToString();
            }
            else
            {
                nameHostText.text = "Đã bị hủy";
                activeText.text = "";
            }
            joinButton.onClick.AddListener(() => JoinRoom(room.Name));
            instantiatedRoomInfos.Add(newRoomInfo);
        }
        if (roomListContent.childCount > 6)
        {
            _scrollTop.SetActive(true);
            _scrollBottom.SetActive(true);
        }
        else
        {
            _scrollTop.SetActive(false);
            _scrollBottom.SetActive(false);
        }
    }
    public override void OnLeftRoom()
    {
        roomList.Clear();
    }

    private void OnDestroy()
    {
        PhotonNetwork.RemoveCallbackTarget(this);
    }
    private void OnEnable()
    {
        PhotonNetwork.AddCallbackTarget(this);
    }
    private void OnDisable()
    {
        PhotonNetwork.RemoveCallbackTarget(this);
    }
    private bool RoomNameExists(string roomName)
    {
        foreach (RoomInfo room in roomList)
        {
            if (room.Name == roomName)
            {
                return true;
            }
```

```
        }
        return false;
    }

}
```

Bảng 4 File code Menu

4. File code Lobby

Gồm các chức năng:

- Vào trận đấu
- Kich người chơi khác
- Xác định chủ phòng
- Hiển thị nhân vật tương ứng với nhân vật người chơi đã chọn
- Nhắn tin

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;
using Photon.Pun;
using Photon.Realtime;
using System.Text;
using System.Linq;

public class SoloLobby : MonoBehaviourPunCallbacks
{
    private const string PlayerNameKey = "PlayerName";
    private const string SelectedChampKey = "SelectedChamp";
    private PhotonView _photonView;
    public Button _startButton;
    public Button _leaveButton;
    public TextMeshProUGUI _playerCountText;
    public TextMeshProUGUI _idLobbyText;
    public Button _copyIdLobbyButton;
    public GameObject[] _playerObjects;
    public TextMeshProUGUI[] _playerNameTexts;
    public TextMeshProUGUI[] _champNameTexts;
    public GameObject[] _champImageObjects;
    public GameObject[] _champImagePrefabs;
    public Button[] _kickButtons;
    public GameObject[] _kickButtonsHostOnly;
    public GameObject[] _hostTextsHostOnly;
```

```
    public TMP_InputField _chatInput;
    public Button _chatButton;
    public ScrollRect _chatListView;
    public Transform _chatListContent;
    public GameObject _contentChatPrefab;
    private List<GameObject> _chatMessages = new List<GameObject>();

    private Dictionary<Player, string> playerChampMap = new Dictionary<Player, string>();

    private bool isHost = false;
    private bool isKicked = false;

    private void Start()
    {
        _photonView = GetComponent<PhotonView>();

        _idLobbyText.text = PhotonNetwork.CurrentRoom.Name;

        _startButton.onClick.AddListener(StartGame);
        _copyIdLobbyButton.onClick.AddListener(CopyLobbyId);
        _leaveButton.onClick.AddListener(LeaveLobby);
        _chatInput.onEndEdit.AddListener(OnChatInputEndEdit);
        _chatButton.onClick.AddListener(SendChatMessage);

        UpdateName();

        InitializePlayerObjects();

        PlayerPrefs.SetString("Scene", "SoloLobby");
    }

    private void UpdateName()
    {
        string playerName = PlayerPrefs.GetString(PlayerNameKey);
        string newPlayerName = playerName;
        if (PlayerNameExists(playerName))
        {
            newPlayerName += "Z";
            PhotonNetwork.NickName = newPlayerName;
        }
        else
        {
            PhotonNetwork.NickName = playerName;
        }
    }

    private bool PlayerNameExists(string playerName)
    {
        Player[] players = PhotonNetwork.PlayerList;
        foreach (Player player in players)
```

```
        {
            if (player.NickName == playerName && player != PhotonNetwork.LocalPlayer)
            {
                return true;
            }
        }
        return false;
    }

    private void Update()
    {
        if (isKicked)
        {
            isKicked = false;
            StartCoroutine(DelayedSceneChange());
        }
    }

    private void Awake()
    {
        for (int i = 0; i < _kickButtons.Length; i++)
        {
            int playerIndex = i;
            _kickButtons[i].onClick.AddListener(() => KickPlayer(playerIndex));
        }
    }

    private void CopyLobbyId()
    {
        GUIUtility.systemCopyBuffer = PhotonNetwork.CurrentRoom.Name;
    }

    private void InitializePlayerObjects()
    {
        isHost = PhotonNetwork.IsMasterClient;

        for (int i = 0; i < PhotonNetwork.CurrentRoom.PlayerCount; i++)
        {
            Player player = PhotonNetwork.PlayerList[i];
            _playerObjects[i].SetActive(true);
            _playerNameTexts[i].text = player.NickName;

            if (player.CustomProperties.TryGetValue("SelectedChampName", out object
selectedChampObj) && selectedChampObj is string selectedChampName)
            {
                _champNameTexts[i].text = selectedChampName;

                GameObject champImageObject = _champImageObjects[i];
                for (int j = 0; j < _champImagePrefabs.Length; j++)
                {
```

```
                    if (champImageObject.transform.childCount == 0 &&
_champImagePrefabs.Length > j && _champImagePrefabs[j] != null &&
_champImagePrefabs[j].name == selectedChampName)
                    {
                        GameObject instantiatedChampImagePrefab =
Instantiate(_champImagePrefabs[j], champImageObject.transform);
                        instantiatedChampImagePrefab.transform.SetParent(champImageObject
.transform, false);
                    }
                }
                playerChampMap[player] = selectedChampName;
            }
            else
            {
                _champNameTexts[i].text = "No Champ";
            }
        }

        for (int i = PhotonNetwork.CurrentRoom.PlayerCount; i < _playerObjects.Length;
i++)
        {
            _playerObjects[i].SetActive(false);
        }
        for (int i = 0; i < _kickButtonsHostOnly.Length; i++)
        {
            _kickButtonsHostOnly[i].SetActive(isHost);
            _kickButtonsHostOnly[0].SetActive(false);
        }
        for (int i = 0; i < _hostTextsHostOnly.Length; i++)
        {
            _hostTextsHostOnly[i].SetActive(false);
            _hostTextsHostOnly[0].SetActive(true);
        }
        UpdateStartButtonState();
        UpdatePlayerCountText();
    }

    private void UpdateStartButtonState()
    {
        if(PhotonNetwork.CurrentRoom.PlayerCount > 1 && isHost)
            _startButton.interactable = true;
        else
            _startButton.interactable = false;
    }

    public void StartGame()
    {
        Player[] players = PhotonNetwork.PlayerList;
```

```
        ExitGames.Client.Photon.Hashtable CustomProperties = new
ExitGames.Client.Photon.Hashtable();

        Dictionary<string, ExitGames.Client.Photon.Hashtable> playerData = new
Dictionary<string, ExitGames.Client.Photon.Hashtable>();

        for(int i = 0; i < players.Length; i++)
        {
            Player player = players[i];
            ExitGames.Client.Photon.Hashtable playerProperties = new
ExitGames.Client.Photon.Hashtable();
            playerProperties["PlayerName"] = player.NickName;

            if (player.CustomProperties.TryGetValue("SelectedChampName", out object
selectedChampObj) && selectedChampObj is string selectedChampName)
            {
                playerProperties["ChampName"] = selectedChampName;
            }
            else
            {
                playerProperties["ChampName"] = "No Champ";
            }
            playerData[player.NickName] = playerProperties;
        }

        CustomProperties["PlayerData"] = playerData;
        PhotonNetwork.CurrentRoom.SetCustomProperties(CustomProperties);

        PhotonNetwork.CurrentRoom.IsOpen = false;
        PhotonNetwork.CurrentRoom.IsVisible = false;

        StartCoroutine(DelayedStartGame());
    }

    private IEnumerator DelayedStartGame()
    {
        yield return new WaitForSeconds(2f);
        _photonView.RPC("RequestSceneChangeToGame", RpcTarget.All);
    }
    private IEnumerator DelayedPlayerEnteredRoom()
    {
        yield return new WaitForSeconds(0.3f);
        InitializePlayerObjects();
    }

    public override void OnPlayerEnteredRoom(Player newPlayer)
    {
        InitializePlayerObjects();
        _photonView.RPC("SendNoticationJoinRoom", RpcTarget.All, newPlayer);
        StartCoroutine(DelayedPlayerEnteredRoom());
```

```
    }

    public override void OnPlayerLeftRoom(Player otherPlayer)
    {
        UpdateName();
        Player[] players = PhotonNetwork.PlayerList;
        if (players.Length > 0)
        {
            Player newHost = players[0];
            PhotonNetwork.SetMasterClient(newHost);

            ExitGames.Client.Photon.Hashtable CustomProperties = new
ExitGames.Client.Photon.Hashtable();
            CustomProperties["Host"] = newHost.NickName;
            PhotonNetwork.CurrentRoom.SetCustomProperties(CustomProperties);
        }

        for (int i = 0; i < _champImageObjects.Length; i++)
        {
            GameObject champImageObject = _champImageObjects[i];
            for (int j = 0; j < champImageObject.transform.childCount; j++)
            {
                Destroy(champImageObject.transform.GetChild(j).gameObject);
            }
        }

        for (int i = 0; i < PhotonNetwork.CurrentRoom.PlayerCount; i++)
        {
            Player player = PhotonNetwork.PlayerList[i];
            string selectedChampName;

            if (player.CustomProperties.TryGetValue("SelectedChampName", out object
selectedChampObj) && selectedChampObj is string)
            {
                selectedChampName = (string)selectedChampObj;
            }
            else
            {
                selectedChampName = "No Champ";
            }

            GameObject champImageObject = _champImageObjects[i];
            for (int j = 0; j < _champImagePrefabs.Length; j++)
            {
                if (_champImagePrefabs[j] != null && _champImagePrefabs[j].name ==
selectedChampName)
                {
                    GameObject instantiatedChampImagePrefab =
Instantiate(_champImagePrefabs[j], champImageObject.transform);
```

```
                    instantiatedChampImagePrefab.transform.SetParent(champImageObject.tra
nsform, false);
                }
            }
        }
        _photonView.RPC("SendNoticationLeaveRoom", RpcTarget.All, otherPlayer);
        InitializePlayerObjects();
    }

    private void UpdatePlayerCountText()
    {
        int currentPlayerCount = PhotonNetwork.CurrentRoom.PlayerCount;
        int maxPlayer = PhotonNetwork.CurrentRoom.MaxPlayers;
        _playerCountText.text = $"Số người trong phòng:
{currentPlayerCount}/{maxPlayer}";
    }

    private void LeaveLobby()
    {
        PhotonNetwork.LeaveRoom();
        StartCoroutine(DelayedSceneChange());
    }

    private IEnumerator DelayedSceneChange()
    {
        yield return new WaitForSeconds(1f);
        PhotonNetwork.LoadLevel("Menu");
    }

    [PunRPC]
    private void RequestSceneChangeToGame()
    {
        PhotonNetwork.LoadLevel("Game");
    }

    public void KickPlayer(int playerIndex)
    {
        if (playerIndex >= 0 && playerIndex < PhotonNetwork.PlayerList.Length)
        {
            Player playerToKick = PhotonNetwork.PlayerList[playerIndex];
            if (playerToKick != null)
            {
                if (isHost)
                {
                    _photonView.RPC("KickPlayerRPC", playerToKick);
                }
            }
        }
    }
```

```
    [PunRPC]
    private void KickPlayerRPC()
    {
        isKicked = true;
        PhotonNetwork.LeaveRoom();
    }

    private void OnChatInputEndEdit(string text)
    {
        if (Input.GetKeyDown(KeyCode.Return) || Input.GetKeyDown(KeyCode.KeypadEnter))
        {
            if (!string.IsNullOrWhiteSpace(text))
            {
                SendChatMessage();
            }

            StartCoroutine(SelectChatInputNextFrame());
        }
    }

    private void SendChatMessage()
    {
        string message = _chatInput.text;
        if (!string.IsNullOrWhiteSpace(message))
        {
            _photonView.RPC("AddChatMessage", RpcTarget.All, message,
PhotonNetwork.LocalPlayer);
            _chatInput.text = "";
            _chatInput.Select();
        }
    }

    [PunRPC]
    private void AddChatMessage(string message, Player sendingPlayer)
    {
        GameObject messageObject = Instantiate(_contentChatPrefab, _chatListContent);
        TextMeshProUGUI messageText =
messageObject.GetComponentInChildren<TextMeshProUGUI>();

        string playerName = sendingPlayer.NickName;
        string champName = "No Champ";
        if (playerChampMap.TryGetValue(sendingPlayer, out string selectedChampName))
        {
            champName = selectedChampName;
        }

        if (message.Length > 40)
        {
            message = message.Substring(0, 40) + "...";
        }
```

```
        messageText.text = $"{playerName} ({champName}): {message}";

        _chatMessages.Add(messageObject);

        StartCoroutine(DelayedScrollToBottom());
    }

    [PunRPC]
    private void SendNoticationJoinRoom(Player newPlayer)
    {
        GameObject messageObject = Instantiate(_contentChatPrefab, _chatListContent);
        TextMeshProUGUI messageText =
messageObject.GetComponentInChildren<TextMeshProUGUI>();

        Color orangeColor = new Color(1.0f, 0.5f, 0.0f);
        messageText.color = orangeColor;

        messageText.text = $"Hệ thống: {newPlayer.NickName} đã vào phòng!!";

        _chatMessages.Add(messageObject);

        StartCoroutine(DelayedScrollToBottom());
    }

    [PunRPC]
    private void SendNoticationLeaveRoom(Player ortherPlayer)
    {
        GameObject messageObject = Instantiate(_contentChatPrefab, _chatListContent);
        TextMeshProUGUI messageText =
messageObject.GetComponentInChildren<TextMeshProUGUI>();

        Color redColor = new Color(1.0f, 0.0f, 0.0f);
        messageText.color = redColor;

        messageText.text = $"Hệ thống: {ortherPlayer.NickName} đã rời phòng!!";

        _chatMessages.Add(messageObject);

        StartCoroutine(DelayedScrollToBottom());
    }

    private IEnumerator DelayedScrollToBottom()
    {
        yield return new WaitForSeconds(0.1f);
        _chatListView.verticalNormalizedPosition = 0f;
    }

    private IEnumerator SelectChatInputNextFrame()
    {
```

```
        yield return null;

        _chatInput.Select();
        _chatInput.ActivateInputField();
    }
}
```

Bảng 5 File code Lobby

5. File code Game

Gồm các chức năng:

- Parallax Background
- Tạo nhân vật tương ứng vào màn hình game
- Đồng bộ tất cả thao tác của tất cả các người chơi trên server
- Kiểm tra thắng thua bằng cách kiểm tra máu của người chơi bằng 0
- Lấy tất cả chỉ số của người chơi đưa lên server
- Lấy FPS, Ping của máy
- Hiển thị thời gian của trận đấu hiện tại
- Cài đặt
- Nhắn tin

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;
using Photon.Pun;
using Photon.Realtime;
using System.Text;
using System.Linq;

public class Game : MonoBehaviourPunCallbacks
{
    private const string ResolutionIndexKey = "ResolutionIndex";
    private const string FullscreenKey = "Fullscreen";
    private const string HiddenNameKey = "HiddenName";
    private const string HiddenPropertiesKey = "HiddenProperties";
    private const string HiddenNameChampKey = "HiddenNameChamp";
    private const string HiddenDamageKey = "HiddenDamage";
    private const string HiddenChatKey = "HiddenChat";
    private const string MusicVolumeKey = "MusicVolume";
    private const string SoundVolumeKey = "SoundVolume";
    private const string VoiceVolumeKey = "VoiceVolume";
```

```
    private PhotonView _photonView;
    public static Game Instance;
    private GameObject localPlayer;
    public GameObject cameraPosition;
    public Transform[] backgroundLayers;
    public float[] parallaxSpeeds;
    private Transform playerTransform;
    private Vector3 previousCameraPosition;
    private int playerInRoom = 0;
    private bool canZoomCamera = true;

    private float maxX = 6.1f;
    private float minX = -6.1f;
    private float maxY = 1.5f;
    private float minY = -0.8f;
    private Dictionary<Player, string> playerChampNames = new Dictionary<Player,
string>();

    public GameObject playerObjectPrefab;
    private Dictionary<Player, GameObject> playerObjectMap = new Dictionary<Player,
GameObject>();
    private Vector2[] vector2Player;

    public TMP_InputField _chatInput;
    public Button _chatButton;
    public ScrollRect _chatListView;
    public Transform _chatListContent;
    public GameObject _contentChatPrefab;
    private List<GameObject> _chatMessages = new List<GameObject>();
    private Dictionary<Player, string> playerChampMap = new Dictionary<Player, string>();
    private int layerPlayer;

    private float _damagePlayer01;
    private float _defensePlayer01;
    private float _healthMaxPlayer01;
    private float _healthPlayer01;
    private float _healingPlayer01;
    private float _manaMaxPlayer01;
    private float _manaPlayer01;
    private float _restoreManaPlayer01;
    private float _criticalPlayer01;
    private float _criticalDamagePlayer01;
    private float _dodgePlayer01;
    private float _bloodsuckingPlayer01;
    private float _reducedHealingPlayer01;
    private float _speedPlayer01;
    private float _resistancePlayer01;
    private float _penetratePlayer01;
    private float _attackSpeedPlayer01;
```

```
    private float _reducedTimePlayer01;
    private float _stunPlayer01;

    private float _damagePlayer02;
    private float _defensePlayer02;
    private float _healthMaxPlayer02;
    private float _healthPlayer02;
    private float _healingPlayer02;
    private float _manaMaxPlayer02;
    private float _manaPlayer02;
    private float _restoreManaPlayer02;
    private float _criticalPlayer02;
    private float _criticalDamagePlayer02;
    private float _dodgePlayer02;
    private float _bloodsuckingPlayer02;
    private float _reducedHealingPlayer02;
    private float _speedPlayer02;
    private float _resistancePlayer02;
    private float _penetratePlayer02;
    private float _attackSpeedPlayer02;
    private float _reducedTimePlayer02;
    private float _stunPlayer02;

    private float _hitWaitTimePlayer01;
    private float _hitWaitTimePlayer02;

    public TextMeshProUGUI _damagePlayer01Text;
    public TextMeshProUGUI _defensePlayer01Text;
    public TextMeshProUGUI _healthMaxPlayer01Text;
    public TextMeshProUGUI _healthPlayer01Text;
    public TextMeshProUGUI _healingPlayer01Text;
    public TextMeshProUGUI _manaMaxPlayer01Text;
    public TextMeshProUGUI _manaPlayer01Text;
    public TextMeshProUGUI _restoreManaPlayer01Text;
    public TextMeshProUGUI _criticalPlayer01Text;
    public TextMeshProUGUI _criticalDamagePlayer01Text;
    public TextMeshProUGUI _dodgePlayer01Text;
    public TextMeshProUGUI _bloodsuckingPlayer01Text;
    public TextMeshProUGUI _reducedHealingPlayer01Text;
    public TextMeshProUGUI _speedPlayer01Text;
    public TextMeshProUGUI _resistancePlayer01Text;
    public TextMeshProUGUI _penetratePlayer01Text;
    public TextMeshProUGUI _attackSpeedPlayer01Text;
    public TextMeshProUGUI _reducedTimePlayer01Text;
    public TextMeshProUGUI _stunPlayer01Text;

    public TextMeshProUGUI _damagePlayer02Text;
    public TextMeshProUGUI _defensePlayer02Text;
    public TextMeshProUGUI _healthMaxPlayer02Text;
    public TextMeshProUGUI _healthPlayer02Text;
```

```
    public TextMeshProUGUI _healingPlayer02Text;
    public TextMeshProUGUI _manaMaxPlayer02Text;
    public TextMeshProUGUI _manaPlayer02Text;
    public TextMeshProUGUI _restoreManaPlayer02Text;
    public TextMeshProUGUI _criticalPlayer02Text;
    public TextMeshProUGUI _criticalDamagePlayer02Text;
    public TextMeshProUGUI _dodgePlayer02Text;
    public TextMeshProUGUI _bloodsuckingPlayer02Text;
    public TextMeshProUGUI _reducedHealingPlayer02Text;
    public TextMeshProUGUI _speedPlayer02Text;
    public TextMeshProUGUI _resistancePlayer02Text;
    public TextMeshProUGUI _penetratePlayer02Text;
    public TextMeshProUGUI _attackSpeedPlayer02Text;
    public TextMeshProUGUI _reducedTimePlayer02Text;
    public TextMeshProUGUI _stunPlayer02Text;

    public Button _buttonSettting;
    public Button _buttonCloseSetting;
    public Button _buttonSaveSetting;
    public Button _buttonLeaveRoom;
    public GameObject _settingPanel;

    public TMP_Dropdown _dropdownResolution;
    public Toggle _toggleFullscreen;
    public Toggle _toggleHiddenName;
    public Toggle _toggleHiddenProperties;
    public Toggle _toggleHiddenNameChamp;
    public Toggle _toggleHiddenDamage;
    public Toggle _toggleHiddenChat;
    public Slider _sliderMusic;
    public Slider _sliderSound;
    public Slider _sliderVoice;
    public AudioSource _backgroundMusic;

    private Resolution[] _resolutions;

    private List<GameObject> namePlayerObjects;
    private List<GameObject> nameChampObjects;
    private List<GameObject> dameObjects;
    private List<GameObject> propertiesObjects;
    private List<GameObject> chatObjects;

    public HealthBar healthBarPlayer01;
    public DelayedHealthBar healthDelayBarPlayer01;
    public HealthBar healthBarPlayer02;
    public DelayedHealthBar healthDelayBarPlayer02;

    public HealthBar manaBarPlayer01;
    public DelayedHealthBar manaDelayBarPlayer01;
    public HealthBar manaBarPlayer02;
```

```
    public DelayedHealthBar manaDelayBarPlayer02;
    public Image _fillHealthBarPlayer01;
    public Image _fillGealthDelayBarPlayer01;
    public Image _fillHealthBarPlayer02;
    public Image _fillGealthDelayBarPlayer02;

    public Image _imageThumbnailPlayer01;
    public Image _imageThumbnailPlayer02;

    private TextMeshProUGUI _pingText;
    private TextMeshProUGUI _fpsText;
    private TextMeshProUGUI _timeText;
    private float _timeGame;
    private  bool _endGame = false;
    public Button _buttonLeaveGameWin;
    public Button _buttonLeaveGameLose;
    public GameObject _UIWin;
    public GameObject _UILose;

    public GameObject _dummy;

    private AudioSource[] _voiceSound;
    private AudioSource[] _effectSound;

    private GameObject[] _voiceSoundGameObject;
    private GameObject[] _effectSoundGameObject;

    private void Awake()
    {
        Instance = this;
    }

    public void SetLocalPlayer(GameObject player)
    {
        localPlayer = player;
    }

    public GameObject GetLocalPlayer()
    {
        return localPlayer;
    }

    void Start()
    {
        _photonView = GetComponent<PhotonView>();
        localPlayer = Game.Instance.GetLocalPlayer();

        _chatInput.onEndEdit.AddListener(OnChatInputEndEdit);
        _chatButton.onClick.AddListener(SendChatMessage);
```

```
        _buttonSettting.onClick.AddListener(ShowSettingPanel);
        _buttonCloseSetting.onClick.AddListener(HiddenSettingPanel);

        previousCameraPosition = Camera.main.transform.position;

        SetNameChamp();
        SpawnPlayers();
        SendNoticationStartGame();
        HiddenNamePlayer();
        HiddenNameChamp();
        StartCoroutine(HiddenChat());
        HiddenProperties();
        layerPlayer = LayerMask.NameToLayer("Player");
        Physics2D.IgnoreLayerCollision(layerPlayer, layerPlayer);

        List<Resolution> validResolutions = new List<Resolution>();
        Resolution[] allResolutions = Screen.resolutions;
        HashSet<string> uniqueResolutions = new HashSet<string>();

        foreach (Resolution res in allResolutions)
        {
            if (Mathf.Approximately(res.width / (float)res.height, 16f / 9f))
            {
                string resolutionString = $"{res.width} x {res.height}";

                if (!uniqueResolutions.Contains(resolutionString))
                {
                    uniqueResolutions.Add(resolutionString);
                    validResolutions.Add(res);
                }
            }
        }

        _resolutions = validResolutions.ToArray();

        _dropdownResolution.ClearOptions();
        List<string> resolutionOptions = new List<string>();
        foreach (Resolution res in validResolutions)
        {
            resolutionOptions.Add($"{res.width} x {res.height}");
        }
        _dropdownResolution.AddOptions(resolutionOptions);

        LoadSettings();

        _sliderMusic.onValueChanged.AddListener((value) => ChangeSliderMusic());
        ChangeSliderMusic();

        _buttonSaveSetting.onClick.AddListener(SaveSettings);
        _buttonLeaveRoom.onClick.AddListener(LeaveLobby);
```

```
        _buttonLeaveGameLose.onClick.AddListener(LeaveLobby);
        _buttonLeaveGameWin.onClick.AddListener(LeaveLobby);

        _pingText = GameObject.Find("PingText").GetComponent<TextMeshProUGUI>();
        _fpsText = GameObject.Find("FPSText").GetComponent<TextMeshProUGUI>();
        _timeText = GameObject.Find("TimeText").GetComponent<TextMeshProUGUI>();

        StartCoroutine(SetMaxBar());
        if(PlayerPrefs.GetString("Scene") == "ComputerLobby")
        {
            _dummy.SetActive(true);
            _imageThumbnailPlayer02.sprite =
Resources.Load<Sprite>("Thumbnail/ThumbnailDummy");
        }
    }

    void Update()
    {
        CameraStartZoom();
        StartCoroutine(DelayedCameraFollow());
        ParallaxBackground();
        UpdateHitTimeWait();
        StartCoroutine(MaxHhealthManaPlayer());
        HiddenDamage();
        _voiceSound = GameObject.FindGameObjectsWithTag("VoiceSound").Select(go =>
go.GetComponent<AudioSource>()).ToArray();
        _effectSound = GameObject.FindGameObjectsWithTag("EffectSound").Select(go =>
go.GetComponent<AudioSource>()).ToArray();
        _voiceSoundGameObject =
GameObject.FindGameObjectsWithTag("VoiceSound").ToArray();
        _effectSoundGameObject =
GameObject.FindGameObjectsWithTag("EffectSound").ToArray();
        _voiceSound.ToList().ForEach(sound => sound.volume = _sliderVoice.value);
        _effectSound.ToList().ForEach(sound => sound.volume = _sliderSound.value);

    }

    IEnumerator MaxHhealthManaPlayer()
    {
        yield return new WaitForSeconds(5.0f);

        if(_healthPlayer01 >= _healthMaxPlayer01)
        {
            _healthPlayer01 = _healthMaxPlayer01;
        }
        if(_healthPlayer02 >= _healthMaxPlayer02)
        {
            _healthPlayer02 = _healthMaxPlayer02;
        }
        if(_healthPlayer01 <= 0)
```

```
        {
            _healthPlayer01 = 0;
            if(PlayerPrefs.GetString("Scene") != "ComputerLobby")
            {
                _endGame = true;
                if (playerInRoom == 0)
                {
                    _UILose.SetActive(true);
                }
                if (playerInRoom == 1)
                {
                    _UIWin.SetActive(true);
                }
            }
        }
        if(_healthPlayer02 <= 0)
        {
            _healthPlayer02 = 0;
            if(PlayerPrefs.GetString("Scene") != "ComputerLobby")
            {
                _endGame = true;
                if (playerInRoom == 0)
                {
                    _UIWin.SetActive(true);
                }
                if (playerInRoom == 1)
                {
                    _UILose.SetActive(true);
                }
            }
        }

        if(_manaPlayer01 >= _manaMaxPlayer01)
        {
            _manaPlayer01 = _manaMaxPlayer01;
        }
        if(_manaPlayer02 >= _manaMaxPlayer02)
        {
            _manaPlayer02 = _manaMaxPlayer02;
        }
        if(_manaPlayer01 <= 0 && !_endGame)
        {
            _manaPlayer01 = 0;
        }
        if(_manaPlayer02 <= 0 && !_endGame)
        {
            _manaPlayer02 = 0;
        }
    }
```

```
    void FixedUpdate()
    {
        Configuration();
    }

    void Configuration()
    {
        float ping = PhotonNetwork.GetPing();
        _pingText.text = "Ping: " + ping.ToString() + "ms";
        float fps = 1.0f / Time.deltaTime;
        _fpsText.text = "FPS: " + fps.ToString("###,###.#");
        _timeGame += Time.deltaTime;
        int minutes = Mathf.FloorToInt(_timeGame / 60F);
        int seconds = Mathf.FloorToInt(_timeGame - minutes * 60);
        string niceTime = string.Format("{0:0}:{1:00}", minutes, seconds);
        _timeText.text = "Time: " + niceTime;
    }

    void ShowSettingPanel()
    {
        _settingPanel.SetActive(true);
    }

    void HiddenSettingPanel()
    {
        _settingPanel.SetActive(false);
    }

    void LoadSettings()
    {
        int resolutionIndex = PlayerPrefs.GetInt(ResolutionIndexKey);
        _dropdownResolution.value = resolutionIndex;
        _toggleFullscreen.isOn = PlayerPrefs.GetInt(FullscreenKey, 1) == 1;
        _toggleHiddenName.isOn = PlayerPrefs.GetInt(HiddenNameKey) == 1;
        _toggleHiddenProperties.isOn = PlayerPrefs.GetInt(HiddenPropertiesKey) == 1;
        _toggleHiddenNameChamp.isOn = PlayerPrefs.GetInt(HiddenNameChampKey) == 1;
        _toggleHiddenDamage.isOn = PlayerPrefs.GetInt(HiddenDamageKey) == 1;
        _toggleHiddenChat.isOn = PlayerPrefs.GetInt(HiddenChatKey) == 1;
        _sliderMusic.value = PlayerPrefs.GetFloat(MusicVolumeKey, 0.3f);
        _sliderSound.value = PlayerPrefs.GetFloat(SoundVolumeKey, 0.5f);
        _sliderVoice.value = PlayerPrefs.GetFloat(VoiceVolumeKey, 0.5f);
        _backgroundMusic.volume = PlayerPrefs.GetFloat(MusicVolumeKey);

        SetResolution(resolutionIndex);
    }

    void SetResolution(int resolutionIndex)
    {
        if (resolutionIndex >= 0 && resolutionIndex < _resolutions.Length)
        {
```

```
            Resolution selectedResolution = _resolutions[resolutionIndex];
            Screen.SetResolution(selectedResolution.width, selectedResolution.height,
_toggleFullscreen.isOn);
        }
    }

    void ChangeSliderMusic()
    {
        _backgroundMusic.volume = _sliderMusic.value;
    }

    void SaveSettings()
    {
        PlayerPrefs.SetInt("ResolutionIndex", _dropdownResolution.value);
        PlayerPrefs.SetInt("Fullscreen", _toggleFullscreen.isOn ? 1 : 0);
        PlayerPrefs.SetInt("HiddenName", _toggleHiddenName.isOn ? 1 : 0);
        PlayerPrefs.SetInt("HiddenProperties", _toggleHiddenProperties.isOn ? 1 : 0);
        PlayerPrefs.SetInt("HiddenNameChamp", _toggleHiddenNameChamp.isOn ? 1 : 0);
        PlayerPrefs.SetInt("HiddenDamage", _toggleHiddenDamage.isOn ? 1 : 0);
        PlayerPrefs.SetInt("HiddenChat", _toggleHiddenChat.isOn ? 1 : 0);
        PlayerPrefs.SetFloat("MusicVolume", _sliderMusic.value);
        PlayerPrefs.SetFloat("SoundVolume", _sliderSound.value);
        PlayerPrefs.SetFloat("VoiceVolume", _sliderVoice.value);

        _voiceSound.ToList().ForEach(sound => sound.volume = _sliderVoice.value);
        _effectSound.ToList().ForEach(sound => sound.volume = _sliderSound.value);

        PlayerPrefs.Save();
        LoadSettings();
        HiddenNamePlayer();
        HiddenNameChamp();
        HiddenProperties();
        StartCoroutine(HiddenChat());
        HiddenSettingPanel();
    }

    void HiddenNamePlayer()
    {
        if (namePlayerObjects == null)
        {
            namePlayerObjects = new
List<GameObject>(GameObject.FindGameObjectsWithTag("NamePlayer"));
        }

        bool hiddenName = PlayerPrefs.GetInt(HiddenNameKey, 0) == 1;

        foreach (GameObject go in namePlayerObjects)
        {
            go.SetActive(!hiddenName);
        }
```

```
    }

    void HiddenNameChamp()
    {
        if (nameChampObjects == null)
        {
            nameChampObjects = new
List<GameObject>(GameObject.FindGameObjectsWithTag("NameChamp"));
        }

        bool hiddenNameChamp = PlayerPrefs.GetInt(HiddenNameChampKey, 0) == 1;

        foreach (GameObject go in nameChampObjects)
        {
            go.SetActive(!hiddenNameChamp);
        }
    }

    void HiddenDamage()
    {
        if (dameObjects == null)
        {
            dameObjects = new
List<GameObject>(GameObject.FindGameObjectsWithTag("DameValue"));
        }

        bool hiddenDameObjects = PlayerPrefs.GetInt(HiddenDamageKey, 0) == 1;

        foreach (GameObject go in dameObjects)
        {
            go.SetActive(!hiddenDameObjects);
        }
    }

    void HiddenProperties()
    {
        if (propertiesObjects == null)
        {
            propertiesObjects = new
List<GameObject>(GameObject.FindGameObjectsWithTag("Properties"));
        }

        bool hiddenProperties = PlayerPrefs.GetInt(HiddenPropertiesKey, 0) == 1;

        foreach (GameObject go in propertiesObjects)
        {
            go.SetActive(!hiddenProperties);
        }
    }
```

```
    IEnumerator HiddenChat()
    {
        yield return new WaitForSeconds(1.0f);
        if (chatObjects == null)
        {
            chatObjects = new
List<GameObject>(GameObject.FindGameObjectsWithTag("Chat"));
        }

        bool hiddenChat = PlayerPrefs.GetInt(HiddenChatKey, 0) == 1;

        foreach (GameObject go in chatObjects)
        {
            go.SetActive(!hiddenChat);
        }
    }

    private void ParallaxBackground()
    {
        Vector3 playerDelta = Camera.main.transform.position - previousCameraPosition;
        for (int i = 0; i < backgroundLayers.Length; i++)
        {
            float parallaxX =+ playerDelta.x * parallaxSpeeds[i] * Time.deltaTime;
            float parallaxY =+ playerDelta.y * parallaxSpeeds[i] * Time.deltaTime;

            Vector3 backgroundTargetPosition = backgroundLayers[i].position + new
Vector3(parallaxX, parallaxY, 0f);
            backgroundLayers[i].position = Vector3.Lerp(backgroundLayers[i].position,
backgroundTargetPosition, Time.deltaTime);
        }

        previousCameraPosition = Camera.main.transform.position;
    }

    private void CameraStartZoom()
    {
        if (!canZoomCamera)
            return;

        if(Camera.main.orthographicSize >= 5.0f)
        {
            Camera.main.orthographicSize -= 1.0f * Time.deltaTime;
            if (playerInRoom == 0)
            {
                if(cameraPosition.transform.position.x > minX)
                    cameraPosition.transform.position = new
Vector3(cameraPosition.transform.position.x - (2.0f * Time.deltaTime),
cameraPosition.transform.position.y, cameraPosition.transform.position.z);
            }
            if (playerInRoom == 1)
```

```
            {
                if(cameraPosition.transform.position.x < maxX)
                    cameraPosition.transform.position = new
Vector3(cameraPosition.transform.position.x + (2.0f * Time.deltaTime),
cameraPosition.transform.position.y, cameraPosition.transform.position.z);
            }
            if(cameraPosition.transform.position.y > minY)
                cameraPosition.transform.position = new
Vector3(cameraPosition.transform.position.x, cameraPosition.transform.position.y  - (1.0f
* Time.deltaTime), cameraPosition.transform.position.z);
        }
        else
        {
            canZoomCamera = false;
            Camera.main.orthographicSize = 5.0f;
            return;
        }
    }

    private IEnumerator DelayedCameraFollow()
    {
        yield return new WaitForSeconds(4.0f);
        if (localPlayer != null)
        {
            Vector3 targetPosition = new Vector3(localPlayer.transform.position.x,
localPlayer.transform.position.y, Camera.main.transform.position.z);
            Camera.main.transform.position = Vector3.Lerp(Camera.main.transform.position,
targetPosition, 3.0f * Time.deltaTime);
        }
        if (cameraPosition.transform.position.x > maxX)
        {
            cameraPosition.transform.position = new Vector3(maxX,
cameraPosition.transform.position.y, cameraPosition.transform.position.z);
        }
        if (cameraPosition.transform.position.x < minX)
        {
            cameraPosition.transform.position = new Vector3(minX,
cameraPosition.transform.position.y, cameraPosition.transform.position.z);
        }

        if (cameraPosition.transform.position.y > maxY)
        {
            cameraPosition.transform.position = new
Vector3(cameraPosition.transform.position.x, maxY, cameraPosition.transform.position.z);
        }
        if (cameraPosition.transform.position.y < minY)
        {
            cameraPosition.transform.position = new
Vector3(cameraPosition.transform.position.x, minY, cameraPosition.transform.position.z);
        }
```

```
    }

    private void SetNameChamp()
    {
        Player[] players = PhotonNetwork.PlayerList;
        for (int i = 0; i < players.Length; i++)
        {
            Player player = players[i];
            ExitGames.Client.Photon.Hashtable CustomProperties =
PhotonNetwork.CurrentRoom.CustomProperties;
            if (CustomProperties.ContainsKey("PlayerData"))
            {
                foreach (var key in ((Dictionary<string,
ExitGames.Client.Photon.Hashtable>)CustomProperties["PlayerData"]).Keys)
                {
                    ExitGames.Client.Photon.Hashtable playerData =
(ExitGames.Client.Photon.Hashtable)((Dictionary<string,
ExitGames.Client.Photon.Hashtable>)CustomProperties["PlayerData"])[key];
                    if (playerData["PlayerName"].ToString() == player.NickName)
                    {
                        playerChampMap[player] = playerData["ChampName"].ToString();
                        playerChampNames[player] = playerData["ChampName"].ToString();
                    }
                }
            }
        }
    }

    private void SpawnPlayers()
    {
        vector2Player = new Vector2[4];
        vector2Player[0] = new Vector2(-6, -4);
        vector2Player[1] = new Vector2(6, -4);
        vector2Player[2] = new Vector2(-10, -4);
        vector2Player[3] = new Vector2(10, -4);

        Player[] players = PhotonNetwork.PlayerList;
        for (int i = 0; i < players.Length; i++)
        {
            Player player = players[i];

            if (player.Equals(PhotonNetwork.LocalPlayer))
            {
                Vector2 spawnPosition = vector2Player[i];

                GameObject playerObject =
PhotonNetwork.Instantiate(playerChampNames[player] + "Prefab", spawnPosition,
Quaternion.identity);

                Game.Instance.SetLocalPlayer(playerObject);
```

```
                playerTransform = Game.Instance.GetLocalPlayer().transform;
                playerObjectMap[player] = playerObject;

                if (i % 2 == 0)
                {
                    playerObject.transform.localScale = new Vector2(1, 1);
                }
                else
                {
                    playerObject.transform.localScale = new Vector2(-1, 1);
                }
                photonView.RPC("SetPlayerTag", RpcTarget.AllBuffered,
playerObject.GetPhotonView().ViewID, "Player0" + (i + 1).ToString());
                photonView.RPC("SetThumbnail", RpcTarget.All, i,
playerChampNames[player]);
                playerInRoom = i;
            }
            else
            {
                if(i == 0)
                {
                    _fillHealthBarPlayer01.color = new Color32(255, 0, 0, 255);
                    _fillGealthDelayBarPlayer01.color = new Color32(255, 100, 0, 255);
                }
                else
                {
                    _fillHealthBarPlayer02.color = new Color32(255, 0, 0, 255);
                    _fillGealthDelayBarPlayer02.color = new Color32(255, 100, 0, 255);
                }
            }

        }
    }

    [PunRPC]
    private void SetThumbnail(int idPlayer, string nameChamp)
    {
        if(idPlayer == 0)
            _imageThumbnailPlayer01.sprite = Resources.Load<Sprite>("Thumbnail/Thumbnail"
+ nameChamp);
        else
            _imageThumbnailPlayer02.sprite = Resources.Load<Sprite>("Thumbnail/Thumbnail"
+ nameChamp);
    }

    [PunRPC]
    private void SetPlayerTag(int viewID, string tag)
    {
        GameObject playerObject = PhotonView.Find(viewID).gameObject;
```

```
        playerObject.tag = tag;
    }

    public override void OnPlayerLeftRoom(Player otherPlayer)
    {
        _photonView.RPC("SendNoticationLeaveRoom", RpcTarget.All, otherPlayer);
    }

    private void OnChatInputEndEdit(string text)
    {
        if (Input.GetKeyDown(KeyCode.Return) || Input.GetKeyDown(KeyCode.KeypadEnter))
        {
            if (!string.IsNullOrWhiteSpace(text))
            {
                SendChatMessage();
            }

            StartCoroutine(SelectChatInputNextFrame());
        }
    }

    private void SendChatMessage()
    {
        string message = _chatInput.text;
        if (!string.IsNullOrWhiteSpace(message))
        {
            _photonView.RPC("AddChatMessage", RpcTarget.All, message,
PhotonNetwork.LocalPlayer);
            _chatInput.text = "";
            _chatInput.Select();
        }
    }

    [PunRPC]
    private void AddChatMessage(string message, Player sendingPlayer)
    {
        GameObject messageObject = Instantiate(_contentChatPrefab, _chatListContent);
        TextMeshProUGUI messageText =
messageObject.GetComponentInChildren<TextMeshProUGUI>();

        string playerName = sendingPlayer.NickName;
        string champName = "No Champ";
        if (playerChampMap.TryGetValue(sendingPlayer, out string selectedChampName))
        {
            champName = selectedChampName;
        }

        if (message.Length > 40)
        {
            message = message.Substring(0, 40) + "...";
```

```
        }

        // Color whiteColor = new Color(1.0f, 1.0f, 1.0f);
        // messageText.color = whiteColor;

        messageText.text = $"{playerName} ({champName}): {message}";

        _chatMessages.Add(messageObject);

        StartCoroutine(DelayedScrollToBottom());
    }

    private void SendNoticationStartGame()
    {
        GameObject messageObject = Instantiate(_contentChatPrefab, _chatListContent);
        TextMeshProUGUI messageText =
messageObject.GetComponentInChildren<TextMeshProUGUI>();

        Color orangeColor = new Color(0.227451f, 0.4470588f, 0.3372549f);
        messageText.color = orangeColor;

        messageText.text = $"Hệ thống: Trận đấu đã được bắt đầu!!";

        _chatMessages.Add(messageObject);

        StartCoroutine(DelayedScrollToBottom());
    }

    [PunRPC]
    private void SendNoticationLeaveRoom(Player ortherPlayer)
    {
        GameObject messageObject = Instantiate(_contentChatPrefab, _chatListContent);
        TextMeshProUGUI messageText =
messageObject.GetComponentInChildren<TextMeshProUGUI>();

        Color redColor = new Color(1.0f, 0.0f, 0.0f);
        messageText.color = redColor;

        messageText.text = $"Hệ thống: {ortherPlayer.NickName} đã rời phòng!!";

        _chatMessages.Add(messageObject);

        StartCoroutine(DelayedScrollToBottom());
    }

    private IEnumerator DelayedScrollToBottom()
    {
        yield return new WaitForSeconds(0.1f);
        _chatListView.verticalNormalizedPosition = 0f;
    }
```

```
    private IEnumerator SelectChatInputNextFrame()
    {
        yield return null;

        _chatInput.Select();
        _chatInput.ActivateInputField();
    }

    private void LeaveLobby()
    {
        PhotonNetwork.LeaveRoom();
        StartCoroutine(DelayedSceneChange());
    }

    private IEnumerator DelayedSceneChange()
    {
        yield return new WaitForSeconds(1f);
        PhotonNetwork.LoadLevel("Menu");
    }

    [PunRPC]
    public void UpdatePropertiesPlayer01(float damage, float defense, float healthMax
,float health, float healing, float manaMax, float mana, float restoreMana, float
critical, float criticalDamage, float dodge, float bloodsucking, float reducedHealing,
float speed, float resistance, float penetrate, float attackSpeed, float reducedTime,
float stun)
    {
        _damagePlayer01 = damage;
        _defensePlayer01 = defense;
        _healthMaxPlayer01 = healthMax;
        _healthPlayer01 = health;
        _healingPlayer01 = healing;
        _manaMaxPlayer01 = manaMax;
        _manaPlayer01 = mana;
        _restoreManaPlayer01 = restoreMana;
        _criticalPlayer01 = critical;
        _criticalDamagePlayer01 = criticalDamage;
        _dodgePlayer01 = dodge;
        _bloodsuckingPlayer01 = bloodsucking;
        _reducedHealingPlayer01 = reducedHealing;
        _speedPlayer01 = speed;
        _resistancePlayer01 = resistance;
        _penetratePlayer01 = penetrate;
        _attackSpeedPlayer01 = attackSpeed;
        _reducedTimePlayer01 = reducedTime;
        _stunPlayer01 = stun;
    }

    [PunRPC]
```

```
    public void UpdatePropertiesPlayer02(float damage, float defense, float healthMax
,float health, float healing, float manaMax, float mana, float restoreMana, float
critical, float criticalDamage, float dodge, float bloodsucking, float reducedHealing,
float speed, float resistance, float penetrate, float attackSpeed, float reducedTime,
float stun)
    {
        _damagePlayer02 = damage;
        _defensePlayer02 = defense;
        _healthMaxPlayer02 = healthMax;
        _healthPlayer02 = health;
        _healingPlayer02 = healing;
        _manaMaxPlayer02 = manaMax;
        _manaPlayer02 = mana;
        _restoreManaPlayer02 = restoreMana;
        _criticalPlayer02 = critical;
        _criticalDamagePlayer02 = criticalDamage;
        _dodgePlayer02 = dodge;
        _bloodsuckingPlayer02 = bloodsucking;
        _reducedHealingPlayer02 = reducedHealing;
        _speedPlayer02 = speed;
        _resistancePlayer02 = resistance;
        _penetratePlayer02 = penetrate;
        _attackSpeedPlayer02 = attackSpeed;
        _reducedTimePlayer02 = reducedTime;
        _stunPlayer02 = stun;
    }

    public IEnumerator TakePropertiesPlayer01(float damage, float defense, float
healthMax ,float health, float healing, float manaMax, float mana, float restoreMana,
float critical, float criticalDamage, float dodge, float bloodsucking, float
reducedHealing, float speed, float resistance, float penetrate, float attackSpeed, float
reducedTime, float stun)
    {
        yield return new WaitForSeconds(1.0f);
        _photonView.RPC("UpdatePropertiesPlayer01", RpcTarget.All, damage, defense,
healthMax, health, healing, manaMax, mana, restoreMana, critical, criticalDamage, dodge,
bloodsucking, reducedHealing, speed, resistance, penetrate, attackSpeed, reducedTime,
stun);
    }

    public IEnumerator TakePropertiesPlayer02(float damage, float defense, float
healthMax ,float health, float healing, float manaMax, float mana, float restoreMana,
float critical, float criticalDamage, float dodge, float bloodsucking, float
reducedHealing, float speed, float resistance, float penetrate, float attackSpeed, float
reducedTime, float stun)
    {
        yield return new WaitForSeconds(1.0f);
        _photonView.RPC("UpdatePropertiesPlayer02", RpcTarget.All, damage, defense,
healthMax, health, healing, manaMax, mana, restoreMana, critical, criticalDamage, dodge,
```

```
bloodsucking, reducedHealing, speed, resistance, penetrate, attackSpeed, reducedTime,
stun);
    }

    [PunRPC]
    public void SetPropertiesPlayer01()
    {
        _damagePlayer01Text.text = GetFormattedAttributeText("Công", _damagePlayer01);
        _defensePlayer01Text.text = GetFormattedAttributeText("Phòng thủ",
_defensePlayer01);
        _healthMaxPlayer01Text.text = _healthMaxPlayer01.ToString("###,###");
        _healthPlayer01Text.text = _healthPlayer01.ToString("###,###");
        _healingPlayer01Text.text = GetFormattedAttributeText("Hồi máu",
_healingPlayer01) + "/s";
        _manaMaxPlayer01Text.text = _manaMaxPlayer01.ToString("###,###");
        _manaPlayer01Text.text = _manaPlayer01.ToString("###,###");
        _restoreManaPlayer01Text.text = GetFormattedAttributeText("Hồi mana",
_restoreManaPlayer01) + "/s";
        _criticalPlayer01Text.text = GetFormattedAttributeText("Bạo kích",
_criticalPlayer01) + "%";
        _criticalDamagePlayer01Text.text = GetFormattedAttributeText("Bạo thương",
_criticalDamagePlayer01) + "%";
        _dodgePlayer01Text.text = GetFormattedAttributeText("Né", _dodgePlayer01) + "%";
        _bloodsuckingPlayer01Text.text = GetFormattedAttributeText("Hút máu",
_bloodsuckingPlayer01) + "%";
        _reducedHealingPlayer01Text.text = GetFormattedAttributeText("Giảm hồi máu",
_reducedHealingPlayer01) + "%";
        _speedPlayer01Text.text = GetFormattedAttributeText("Tốc độ", _speedPlayer01);
        _resistancePlayer01Text.text = GetFormattedAttributeText("Kháng hiệu ứng",
_resistancePlayer01);
        _penetratePlayer01Text.text = GetFormattedAttributeText("Xuyên giáp",
_penetratePlayer01);
        _attackSpeedPlayer01Text.text = GetFormattedAttributeText("Tốc đánh",
_attackSpeedPlayer01) + "%";
        _reducedTimePlayer01Text.text = GetFormattedAttributeText("Giảm hồi chiêu",
_reducedTimePlayer01) + "%";
        _stunPlayer01Text.text = GetFormattedAttributeText("Bất động", _stunPlayer01) +
"%";


        healthDelayBarPlayer01.SetHealth(_healthPlayer01);
        manaDelayBarPlayer01.SetHealth(_manaPlayer01);

        healthBarPlayer01.SetMaxHealth(_healthMaxPlayer01);
        healthBarPlayer01.SetHealth(_healthPlayer01);

        manaBarPlayer01.SetMaxHealth(_manaMaxPlayer01);
        manaBarPlayer01.SetHealth(_manaPlayer01);
    }
```

```
    [PunRPC]
    public void SetPropertiesPlayer02()
    {
        _damagePlayer02Text.text = GetFormattedAttributeText("Công", _damagePlayer02);
        _defensePlayer02Text.text = GetFormattedAttributeText("Phòng thủ",
_defensePlayer02);
        _healthMaxPlayer02Text.text = _healthMaxPlayer02.ToString("###,###");
        _healthPlayer02Text.text =  _healthPlayer02.ToString("###,###");
        _healingPlayer02Text.text = GetFormattedAttributeText("Hồi máu",
_healingPlayer02) + "/s";
        _manaMaxPlayer02Text.text = _manaMaxPlayer02.ToString("###,###");
        _manaPlayer02Text.text = _manaPlayer02.ToString("###,###");
        _restoreManaPlayer02Text.text = GetFormattedAttributeText("Hồi mana",
_restoreManaPlayer02) + "/s";
        _criticalPlayer02Text.text = GetFormattedAttributeText("Bạo kích",
_criticalPlayer02) + "%";
        _criticalDamagePlayer02Text.text = GetFormattedAttributeText("Bạo thương",
_criticalDamagePlayer02) + "%";
        _dodgePlayer02Text.text = GetFormattedAttributeText("Né", _dodgePlayer02) + "%";
        _bloodsuckingPlayer02Text.text = GetFormattedAttributeText("Hút máu",
_bloodsuckingPlayer02) + "%";
        _reducedHealingPlayer02Text.text = GetFormattedAttributeText("Giảm hồi máu",
_reducedHealingPlayer02) + "%";
        _speedPlayer02Text.text = GetFormattedAttributeText("Tốc độ", _speedPlayer02);
        _resistancePlayer02Text.text = GetFormattedAttributeText("Kháng hiệu ứng",
_resistancePlayer02);
        _penetratePlayer02Text.text = GetFormattedAttributeText("Xuyên giáp",
_penetratePlayer02);
        _attackSpeedPlayer02Text.text = GetFormattedAttributeText("Tốc đánh",
_attackSpeedPlayer02) + "%";
        _reducedTimePlayer02Text.text = GetFormattedAttributeText("Giảm hồi chiêu",
_reducedTimePlayer02) + "%";
        _stunPlayer02Text.text = GetFormattedAttributeText("Bất động", _stunPlayer02) +
"%";


        healthDelayBarPlayer02.SetHealth(_healthPlayer02);
        manaDelayBarPlayer02.SetHealth(_manaPlayer02);

        healthBarPlayer02.SetMaxHealth(_healthMaxPlayer02);
        healthBarPlayer02.SetHealth(_healthPlayer02);

        manaBarPlayer02.SetMaxHealth(_manaMaxPlayer02);
        manaBarPlayer02.SetHealth(_manaPlayer02);
    }

    private IEnumerator SetMaxBar()
    {
        yield return new WaitForSeconds(4.0f);
        healthDelayBarPlayer01.SetMaxHealth(_healthMaxPlayer01);
```

```
        healthDelayBarPlayer02.SetMaxHealth(_healthMaxPlayer02);

        manaDelayBarPlayer01.SetMaxHealth(_manaMaxPlayer01);
        manaDelayBarPlayer02.SetMaxHealth(_manaMaxPlayer02);
    }

    private string GetFormattedAttributeText(string attributeName, float attributeValue)
    {
        if(attributeValue == 0)
        {
            return attributeName + ": " + 0;
        }
        else if(attributeValue < 1 && attributeValue > 0)
        {
            return attributeName + ": " + "0" + attributeValue.ToString("###,###.#");
        }
        else
        {
            return attributeName + ": " + attributeValue.ToString("###,###.#");
        }
    }

    public IEnumerator UpdatePropertiesPlayer(string tag)
    {
        yield return new WaitForSeconds(1.5f);
        if (tag == "Player01")
        {
            _photonView.RPC("SetPropertiesPlayer01", RpcTarget.All);
        }
        else if (tag == "Player02")
        {
            _photonView.RPC("SetPropertiesPlayer02", RpcTarget.All);
        }
    }

    public float GetDamagePlayer01()
    {
        return _damagePlayer01;
    }

    public float GetDefensePlayer01()
    {
        return _defensePlayer01;
    }

    public float GetHealthMaxPlayer01()
    {
        return _healthMaxPlayer01;
    }
```

```
public float GetHealthPlayer01()
{
    return _healthPlayer01;
}

public float GetHealingPlayer01()
{
    return _healingPlayer01;
}
public float GetManaMaxPlayer01()
{
    return _manaMaxPlayer01;
}

public float GetManaPlayer01()
{
    return _manaPlayer01;
}

public float GetRestoreManaPlayer01()
{
    return _restoreManaPlayer01;
}

public float GetCriticalPlayer01()
{
    return _criticalPlayer01;
}

public float GetCriticalDamagePlayer01()
{
    return _criticalDamagePlayer01;
}

public float GetDodgePlayer01()
{
    return _dodgePlayer01;
}

public float GetBloodsuckingPlayer01()
{
    return _bloodsuckingPlayer01;
}

public float GetReducedHealingPlayer01()
{
    return _reducedHealingPlayer01;
}

public float GetSpeedPlayer01()
```

```
    {
        return _speedPlayer01;
    }

    public float GetResistancePlayer01()
    {
        return _resistancePlayer01;
    }

    public float GetPenetratePlayer01()
    {
        return _penetratePlayer01;
    }

    public float GetAttackSpeedPlayer01()
    {
        return _attackSpeedPlayer01;
    }

    public float GetReducedTimePlayer01()
    {
        return _reducedTimePlayer01;
    }

    public float GetStunPlayer01()
    {
        return _stunPlayer01;
    }

    public float GetDamagePlayer02()
    {
        return _damagePlayer02;
    }

    public float GetDefensePlayer02()
    {
        return _defensePlayer02;
    }

    public float GetHealthMaxPlayer02()
    {
        return _healthMaxPlayer02;
    }

    public float GetHealthPlayer02()
    {
        return _healthPlayer02;
    }

    public float GetHealingPlayer02()
```

```
    {
        return _healingPlayer02;
    }

    public float GetManaMaxPlayer02()
    {
        return _manaMaxPlayer02;
    }

    public float GetManaPlayer02()
    {
        return _manaPlayer02;
    }

    public float GetRestoreManaPlayer02()
    {
        return _restoreManaPlayer02;
    }

    public float GetCriticalPlayer02()
    {
        return _criticalPlayer02;
    }

    public float GetCriticalDamagePlayer02()
    {
        return _criticalDamagePlayer02;
    }

    public float GetDodgePlayer02()
    {
        return _dodgePlayer02;
    }

    public float GetBloodsuckingPlayer02()
    {
        return _bloodsuckingPlayer02;
    }

    public float GetReducedHealingPlayer02()
    {
        return _reducedHealingPlayer02;
    }

    public float GetSpeedPlayer02()
    {
        return _speedPlayer02;
    }

    public float GetResistancePlayer02()
```

```
    {
        return _resistancePlayer02;
    }

    public float GetPenetratePlayer02()
    {
        return _penetratePlayer02;
    }

    public float GetAttackSpeedPlayer02()
    {
        return _attackSpeedPlayer02;
    }

    public float GetReducedTimePlayer02()
    {
        return _reducedTimePlayer02;
    }

    public float GetStunPlayer02()
    {
        return _stunPlayer02;
    }

    [PunRPC]
    public void SetDamageHurtPlayer01(float damage, float hitWaitTime)
    {
        _healthPlayer01 -= damage;
        _hitWaitTimePlayer01 = hitWaitTime;
    }

    [PunRPC]
    public void SetDamageHurtPlayer02(float damage, float hitWaitTime)
    {
        _healthPlayer02 -= damage;
        _hitWaitTimePlayer02 = hitWaitTime;
    }

    public void TakeDamagePlayer01(float damage, float hitWaitTime)
    {
        _photonView.RPC("SetDamageHurtPlayer01", RpcTarget.All, damage, hitWaitTime);
    }

    public void TakeDamagePlayer02(float damage, float hitWaitTime)
    {
        _photonView.RPC("SetDamageHurtPlayer02", RpcTarget.All, damage, hitWaitTime);
    }

    [PunRPC]
    public void SetPlusHealthPlayer01(float health)
```

```
    {
        _healthPlayer01 += health;
    }

    [PunRPC]
    public void SetPlusHealthPlayer02(float health)
    {
        _healthPlayer02 += health;
    }

    public void TakePlusHealthPlayer01(float health)
    {
        _photonView.RPC("SetPlusHealthPlayer01", RpcTarget.All, health);
    }

    public void TakePlusHealthPlayer02(float health)
    {
        _photonView.RPC("SetPlusHealthPlayer02", RpcTarget.All, health);
    }

    [PunRPC]
    public void SetPlusManaPlayer01(float mana)
    {
        _manaPlayer01 += mana;
    }

    [PunRPC]
    public void SetPlusManaPlayer02(float mana)
    {
        _manaPlayer02 += mana;
    }

    public void TakePlusManaPlayer01(float mana)
    {
        _photonView.RPC("SetPlusManaPlayer01", RpcTarget.All, mana);
    }

    public void TakePlusManaPlayer02(float mana)
    {
        _photonView.RPC("SetPlusManaPlayer02", RpcTarget.All, mana);
    }

    [PunRPC]
    public void SetMinusManaPlayer01(float mana)
    {
        _manaPlayer01 -= mana;
    }

    [PunRPC]
    public void SetMinusManaPlayer02(float mana)
```

```
    {
        _manaPlayer02 -= mana;
    }

    public void TakeMinusManaPlayer01(float mana)
    {
        _photonView.RPC("SetMinusManaPlayer01", RpcTarget.All, mana);
    }

    public void TakeMinusManaPlayer02(float mana)
    {
        _photonView.RPC("SetMinusManaPlayer02", RpcTarget.All, mana);
    }

    [PunRPC]
    public void SetPlusDamPlayer01(float dam)
    {
        _damagePlayer01 += dam;
    }

    [PunRPC]
    public void SetPlusDamPlayer02(float dam)
    {
        _damagePlayer02 += dam;
    }

    public void TakePlusDamPlayer01(float dam)
    {
        _photonView.RPC("SetPlusDamPlayer01", RpcTarget.All, dam);
    }

    public void TakePlusDamPlayer02(float dam)
    {
        _photonView.RPC("SetPlusDamPlayer02", RpcTarget.All, dam);
    }

    public void UpdateHitTimeWait()
    {
        _hitWaitTimePlayer01 -= Time.deltaTime;
        _hitWaitTimePlayer02 -= Time.deltaTime;
    }

    public float GetHitTimeWaitPlayer01()
    {
        return _hitWaitTimePlayer01;
    }

    public float GetHitTimeWaitPlayer02()
    {
        return _hitWaitTimePlayer02;
```

```
    }

    public bool GetEndGame()
    {
        return _endGame;
    }
}
```

Bảng 6 File code Game