ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

ĐỒ ÁN CHUYÊN NGÀNH
BÁO CÁO

PHÁT TRIỂN HỆ THỐNG GAME ONLINE NHẬP VAI HÀNH ĐỘNG NHIỀU NGƯỜI CHƠI

"FORTRESS OF THE FALLEN"

Ngành: Khoa học Máy tính

Học kỳ 1 năm học 2025–2026 (HK251)

Giảng viên hướng dẫn: ThS. Vương Bá Thịnh

Sinh viên thực hiện	**MSSV**
Hà Thái Toàn	2213524
Trần Minh Khang	2211472

TP. Hồ Chí Minh, năm 2025

Danh mục thuật ngữ

Bảng dưới đây liệt kê các thuật ngữ và viết tắt thường được sử dụng trong báo cáo, đặc biệt liên quan tới phát triển game online, kiến trúc client–server, hệ thống AI NPC và cơ sở dữ liệu. Các thuật ngữ này sẽ được dùng xuyên suốt các chương sau.

Thuật ngữ	**Tên đầy đủ**	**Giải thích**
Nhóm thuật ngữ về game và gameplay		
RPG	Role-Playing Game	Game nhập vai, trong đó người chơi điều khiển một hoặc nhiều nhân vật, phát triển sức mạnh thông qua hệ thống level, chỉ số, kỹ năng, trang bị và cốt truyện.
Action RPG	Action Role-Playing Game	Game nhập vai hành động, chiến đấu diễn ra theo thời gian thực (real-time), người chơi điều khiển trực tiếp nhân vật (tấn công, né, dùng kỹ năng) thay vì theo lượt (turn-based).
PvP	Player vs Player	Chế độ người chơi chiến đấu trực tiếp với người chơi khác. Yêu cầu đồng bộ trạng thái nhanh, cơ chế xử lý độ trễ và chống gian lận mạnh.
PvE	Player vs Environment	Chế độ người chơi đối đầu với quái, boss hoặc môi trường do hệ thống điều khiển. Trong *Fortress of the Fallen*, phần lớn nội dung dungeon, tháp, đảo cá nhân ban đầu thiên về PvE.
Boss	Boss	Kẻ địch đặc biệt có chỉ số cao, cơ chế tấn công phức tạp, thường là trọng tâm của một dungeon hoặc tầng tháp.
Dungeon	Dungeon	Khu vực (thường là instance riêng) tập trung chiến đấu, vượt thử thách. Trong đề tài, dungeon có thể là các tầng Tinh Hà Trung Tâm hoặc các khu vực phụ.
Hitbox	Hitbox	Vùng hình học trong game đại diện cho phạm vi đòn tấn công có hiệu lực (ví dụ: vùng chém của kiếm, vùng nổ của kỹ năng).

(còn tiếp trang sau)

Thuật ngữ	Tên đầy đủ	Giải thích
Hurtbox	Hurtbox	Vùng hình học đại diện cho khu vực trên nhân vật có thể nhận sát thương. Va chạm giữa hitbox và hurtbox thường được sử dụng để quyết định trúng đòn.
Cooldown (CD)	Cooldown	Thời gian chờ sau khi sử dụng kỹ năng trước khi kỹ năng đó có thể được sử dụng lại.
Animation	Animation	Chuỗi hình ảnh mô phỏng chuyển động (chạy, tấn công, né, trúng đòn...). Trong Action RPG, animation gắn chặt với gameplay.
Nhóm thuật ngữ về multiplayer và networking		
Multiplayer	Multiplayer	Hình thức chơi nhiều người, cho phép nhiều người chơi cùng tồn tại và tương tác trong một thế giới game chung (ví dụ: cùng ở một đảo, dungeon, tầng tháp).
Real-time	Real-time	Thời gian thực: hệ thống xử lý và phản hồi gần như ngay lập tức theo input của người chơi.
Latency	Network Latency	Độ trễ mạng, là thời gian gói tin đi từ client tới server. Thường đo bằng mili-giây (ms).
Tick rate	Server Tick Rate	Số lần server cập nhật trạng thái game trong một giây (ví dụ: 20 tick/s, 30 tick/s).
Snapshot	State Snapshot	Bản "ảnh chụp" trạng thái game tại một thời điểm (vị trí nhân vật, HP, trạng thái kỹ năng...), được server gửi cho client để đồng bộ.
Client-side Prediction	Client-side Prediction	Kỹ thuật client dự đoán trước kết quả chuyển động, hành động dựa trên input người chơi, nhằm giảm cảm giác trễ.
Reconciliation	Reconciliation	Quá trình đồng bộ lại trên client: so sánh trạng thái dự đoán với trạng thái do server gửi về và điều chỉnh để khớp.

(còn tiếp trang sau)

Thuật ngữ	**Tên đầy đủ**	**Giải thích**
Server Authoritative	Server Authoritative Model	Mô hình trong đó server nắm quyền quyết định cuối cùng về logic game (di chuyển hợp lệ, sát thương, kết quả combat).
WebSocket	WebSocket	Giao thức truyền thông hai chiều trên nền TCP, cho phép client và server giữ kết nối lâu dài và gửi/nhận dữ liệu thời gian thực.
Instance	Instance	Phiên bản riêng biệt của một khu vực game (ví dụ: một dungeon hoặc một tầng tháp tạo riêng cho một nhóm người chơi).
Zone / Shard	Zone / Shard	Khu vực logic trong thế giới game, dùng để chia tải hoặc phân vùng nội dung.
Interest Management	Interest Management	Kỹ thuật lọc và chỉ gửi dữ liệu cần thiết cho mỗi client để giảm băng thông và tải xử lý.

Mục lục

Danh sách hình vẽ

Danh sách bảng

1 Giới thiệu

1.1 Động lực

Trong bối cảnh ngành công nghiệp game phát triển mạnh mẽ, các trò chơi nhập vai trực tuyến nhiều người chơi (Multiplayer Online RPG) ngày càng trở nên phổ biến, với doanh thu và lượng người chơi tăng trưởng đều hằng năm[1]. Người chơi không chỉ kỳ vọng vào đồ hoạ đẹp mắt và nội dung phong phú, mà còn đòi hỏi một trải nghiệm trực tuyến ổn định, mượt mà, cho phép nhiều người chơi tương tác với nhau trong thời gian thực.

Để đáp ứng các kỳ vọng này, hệ thống phía server phải có khả năng:

- Xử lý lượng lớn kết nối đồng thời.
- Đồng bộ trạng thái nhân vật và thế giới trò chơi.
- Quản lý tài nguyên và dữ liệu một cách hiệu quả.
- Đồng thời duy trì độ trễ thấp để không ảnh hưởng đến cảm nhận điều khiển[2].

Việc nghiên cứu, thiết kế và hiện thực một hệ thống game online hoàn chỉnh đòi hỏi sự kết hợp của nhiều mảng kiến thức: lập trình game, mạng máy tính, kiến trúc client–server, cơ sở dữ liệu, đồng bộ trạng thái thời gian thực, cũng như thiết kế cơ chế gameplay và hệ thống game[3, 4].

Đề tài *"Fortress of the Fallen"* được lựa chọn với mục tiêu vừa thoả mãn niềm yêu thích đối với thể loại Action RPG trực tuyến, vừa tạo cơ hội để nhóm sinh viên rèn luyện kỹ năng phân tích, thiết kế và xây dựng một hệ thống phần mềm phức tạp với nhiều thành phần tương tác chặt chẽ.

1.2 Mục tiêu

Mục tiêu tổng quát của đề tài là thiết kế và phát triển nền tảng kỹ thuật cho một trò chơi nhập vai hành động nhiều người chơi trực tuyến *"Fortress of the Fallen"*. Trong giai đoạn 1, đề tài tập trung vào các mục tiêu cụ thể sau:

- Nghiên cứu và tổng hợp ý tưởng thiết kế game từ các tác phẩm tham khảo (tiểu thuyết, trò chơi cùng thể loại), từ đó định hình bối cảnh, cơ chế chơi và các hệ thống tính năng cốt lõi[5, 4].
- Xác định phạm vi và tính khả thi kỹ thuật của đề tài trong khuôn khổ đồ án chuyên ngành, đảm bảo cân đối giữa mức độ phức tạp và nguồn lực thực hiện.
- Thiết kế mô hình kiến trúc hệ thống tổng thể, bao gồm các thành phần: client, server backend, cơ sở dữ liệu và cơ chế giao tiếp mạng[3, 2].

- Hiện thực một bộ khung ban đầu (source base) cho hệ thống, sử dụng Unity cho client và NestJS cho backend, làm nền tảng để phát triển và mở rộng trong các giai đoạn tiếp theo[6, 7].

1.3 Phạm vi

Trong phạm vi giai đoạn 1, đề tài tập trung vào việc xây dựng *prototype nền tảng* thay vì hoàn thiện toàn bộ nội dung game. Cụ thể, phạm vi thực hiện bao gồm:

- Thiết kế kiến trúc tổng thể cho hệ thống game online nhiều người chơi theo mô hình client–server[2].
- Thiết kế và xây dựng mô hình dữ liệu cơ bản phục vụ cho:
  - Quản lý tài khoản và người dùng.
  - Quản lý thông tin nhân vật và các thuộc tính cốt lõi liên quan đến đăng nhập, đăng xuất và lựa chọn nhân vật.
- Xây dựng các chức năng cơ bản phía client:
  - Giao diện đăng ký, đăng nhập.
  - Kết nối đến server và thực hiện xác thực người dùng.
  - Tải và hiển thị nhân vật sau khi đăng nhập.
  - Điều khiển di chuyển nhân vật trong một môi trường thử nghiệm đơn giản (prototype) trên client.
- Xây dựng các chức năng nền tảng phía server:
  - Dịch vụ xác thực (authentication) và quản lý người dùng.
  - Quản lý phiên làm việc và kết nối real-time cho người chơi.
  - Thiết kế hạ tầng dữ liệu cơ bản, sẵn sàng để mở rộng cho các tính năng nâng cao trong giai đoạn sau.

Các nội dung nâng cao như:

- Hệ thống xây dựng đảo cá nhân.
- Hệ thống NPC và cơ chế gacha NPC.
- Hệ thống leo tháp 100 tầng và 18 tầng ngục.
- Hệ thống class–race–skill hoàn chỉnh.
- Chế độ PvP Arena.

- Cân bằng game ở quy mô lớn và tối ưu hiệu năng nâng cao.

sẽ được xem xét ở các giai đoạn tiếp theo và **không thuộc phạm vi hiện thực hoá ở giai đoạn 1**.

1.4 Ý nghĩa của đề tài

1.4.1 Ý nghĩa thực tiễn

Về mặt thực tiễn, đề tài mang lại các ý nghĩa sau:

- Cung cấp một ví dụ cụ thể về quá trình xây dựng hệ thống game online nhiều người chơi từ bước hình thành ý tưởng, phân tích, thiết kế cho đến hiện thực hoá bộ khung kỹ thuật.
- Minh hoạ cách kết hợp các công nghệ phổ biến trong phát triển game và backend như Unity, NestJS, MongoDB, Redis trong một kiến trúc thống nhất[6, 7, 8, 9].
- Đặt nền tảng để phát triển tiếp thành một sản phẩm game hoàn chỉnh hoặc một bộ khung tham khảo cho các đồ án và dự án nghiên cứu khác liên quan đến game online.

1.4.2 Ý nghĩa khoa học và học thuật

Về mặt học thuật, đề tài góp phần:

- Tạo môi trường thực tế để sinh viên vận dụng tổng hợp các kiến thức đã học: cấu trúc dữ liệu và giải thuật, lập trình hướng đối tượng, cơ sở dữ liệu, mạng máy tính, kiến trúc phần mềm, phát triển ứng dụng phân tán.
- Giúp sinh viên tiếp cận các bài toán đặc thù của game online như xử lý kết nối đồng thời, đồng bộ trạng thái thời gian thực, quản lý phiên chơi và mô hình hoá thế giới ảo[2, 3].
- Đóng góp thêm tài liệu tham khảo nội bộ về quy trình phân tích, thiết kế và hiện thực một hệ thống game online nhiều người chơi trong khuôn khổ chương trình Khoa học Máy tính.

1.5 Cấu trúc báo cáo

Phần còn lại của báo cáo được tổ chức như sau:

- **Chương 2 – Kiến thức nền tảng:** Trình bày các khái niệm, định nghĩa và kiến thức liên quan đến game Action RPG, hệ thống multiplayer, kiến trúc client–server và các khái niệm cơ bản trong thiết kế game[4, 5].
- **Chương 3 – Công nghệ sử dụng:** Giới thiệu các công nghệ chính được sử dụng trong đề tài như Unity, NestJS, MongoDB, Redis, GitHub, GitHub Projects, cùng phân tích lý do lựa chọn[6, 7, 8, 9, 10, 11].

- **Chương 4 – Các công trình liên quan:** Phân tích các tác phẩm và trò chơi tham khảo, cũng như các nghiên cứu về game networking và kiến trúc server real-time, từ đó rút ra những bài học áp dụng cho đề tài[2, 4].
- **Chương 5 – Phân tích yêu cầu:** Xác định đối tượng người dùng hệ thống, yêu cầu chức năng, phi chức năng và yêu cầu dữ liệu cho giai đoạn 1.
- **Chương 6 – Phân tích hệ thống:** Mô tả quy trình nghiệp vụ, đặc tả use case và luồng dữ liệu logic trong hệ thống.
- **Chương 7 – Thiết kế hệ thống:** Trình bày kiến trúc tổng thể, thiết kế cơ sở dữ liệu, thiết kế API và các mô-đun chính của hệ thống.
- **Chương 8 – Hiện thực hệ thống:** Mô tả quá trình cài đặt các thành phần client và server, cơ chế giao tiếp và tích hợp.
- **Chương 9 – Đánh giá hệ thống:** Đánh giá mức độ hoàn thành so với mục tiêu ban đầu, kiểm thử các chức năng chính và thảo luận các hạn chế.
- **Chương 10 – Kết luận:** Tổng kết kết quả đạt được, nêu các hạn chế và định hướng phát triển trong các giai đoạn tiếp theo.

2 Kiến thức nền tảng

Chương này trình bày có hệ thống các kiến thức cơ sở liên quan đến phát triển trò chơi điện tử thuộc thể loại Action RPG trong môi trường trực tuyến. Nội dung tập trung vào hai nhóm chính: (1) các khái niệm nền tảng về game thời gian thực và hệ thống multiplayer; (2) kiến thức nền tảng về thiết kế game (Game Design), bao gồm core loop, progression, combat, economy và trải nghiệm người dùng (UX)[4, 5]. Những kiến thức này làm cơ sở cho việc phân tích yêu cầu và thiết kế hệ thống ở các chương tiếp theo.

2.1 Kiến thức nền tảng về Action RPG và môi trường real-time

2.1.1 Đặc điểm của Action RPG

Action RPG (Action Role-Playing Game) là thể loại game kết hợp cơ chế nhập vai với chiến đấu thời gian thực. Một số đặc điểm quan trọng[4, 5]:

- **Tương tác thời gian thực**: mọi hành động như tấn công, di chuyển, né tránh được xử lý và phản hồi gần như tức thời.
- **Combat gắn liền với animation**: chuyển động của nhân vật (animation) liên kết trực tiếp với logic chiến đấu (thời điểm gây sát thương, thời gian bất tử, v.v.).

- **Hitbox / Hurtbox**: hệ thống kiểm tra va chạm dựa trên các hình học đơn giản như AABB, Sphere, Capsule để xác định vùng gây sát thương và vùng nhận sát thương.
- **Kỹ năng người chơi và chỉ số nhân vật phối hợp**: hiệu quả chiến đấu phụ thuộc đồng thời vào thao tác điều khiển của người chơi và chỉ số (stat) của nhân vật.

Một đòn tấn công điển hình có thể được chia thành ba giai đoạn[4]:

1. **Startup**: nhân vật chuẩn bị ra đòn.
2. **Active**: hitbox có hiệu lực, có thể gây sát thương.
3. **Recovery**: nhân vật quay về trạng thái sẵn sàng cho hành động tiếp theo.

Việc hiểu rõ ba giai đoạn này là cơ sở để thiết kế combat có nhịp độ hợp lý, tạo cảm giác mượt và công bằng.

2.1.2 Hệ thống multiplayer trong Action RPG

Một trò chơi Action RPG nhiều người chơi yêu cầu cơ chế đồng bộ trạng thái giữa các người chơi sao cho tất cả đều quan sát được diễn biến nhất quán. Các kiến thức nền tảng chính bao gồm[2, 3]:

Mô hình client–server Hầu hết game online hiện đại sử dụng mô hình **server authoritative**, trong đó:

- **Client** chịu trách nhiệm thu thập input (phím bấm, chuột, tay cầm) và hiển thị kết quả.
- **Server** xử lý toàn bộ logic gameplay quan trọng (tính sát thương, kiểm tra va chạm, xác nhận vị trí, trạng thái kỹ năng) và lưu trữ trạng thái chính thức.

Ưu điểm[2]:

- Đảm bảo tính nhất quán giữa các người chơi.
- Giảm khả năng gian lận do client không được tự ý quyết định kết quả.
- Thuận lợi cho việc mở rộng và bảo trì hệ thống về lâu dài.

Tick-rate và vòng lặp cập nhật Server vận hành theo chu kỳ cập nhật gọi là *tick*. Ví dụ, với tick-rate 20 hoặc 30 lần mỗi giây, ở mỗi tick, server sẽ[2]:

```
- Nhận và xử lý input từ người chơi
- Cập nhật vị trí và trạng thái đối tượng trong thế giới
- Kiểm tra va chạm và xử lý chiến đấu
- Gửi snapshot trạng thái đến client
```

Tick-rate càng cao, gameplay càng mượt nhưng chi phí tài nguyên càng lớn; tick-rate thấp hơn sẽ tiết kiệm tài nguyên nhưng ảnh hưởng đến độ mượt và độ chính xác của combat.

Đồng bộ trạng thái Để giảm tác động của độ trễ mạng và đảm bảo trải nghiệm mượt mà, thường kết hợp nhiều kỹ thuật[2]:

- **Snapshot interpolation**: client nhận các snapshot trạng thái định kỳ từ server và nội suy giữa các snapshot để hiển thị chuyển động mượt.
- **Client-side prediction**: client tạm thời dự đoán trạng thái tiếp theo dựa trên input của người chơi, thay vì chờ server phản hồi.
- **Server reconciliation**: server gửi lại trạng thái chính xác, và client điều chỉnh sai lệch giữa trạng thái dự đoán và trạng thái thực tế.

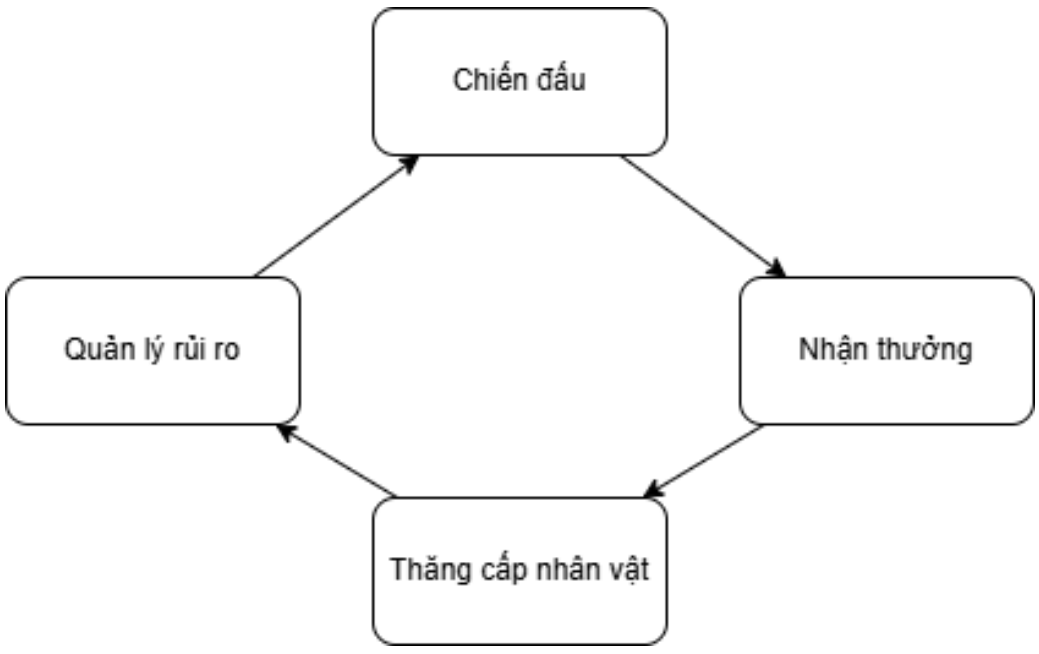
Các kỹ thuật này là nền tảng cho mọi trò chơi real-time có yếu tố cạnh tranh hoặc yêu cầu điều khiển chính xác.

2.2 Kiến thức nền tảng về Game Design

Game Design là lĩnh vực xây dựng cơ chế hoạt động của trò chơi, quy định người chơi làm gì, vì sao họ muốn làm điều đó và cách trò chơi phản hồi với hành động của họ[4, 5]. Một trò chơi hoàn chỉnh thường bao gồm nhiều lớp thiết kế: core loop, hệ thống phát triển nhân vật, hệ thống chiến đấu, hệ thống kinh tế trong game, cơ chế phần thưởng và trải nghiệm người dùng.

2.2.1 Core Loop và Meta Loop

Core Loop là vòng lặp hành động cốt lõi mà người chơi thực hiện liên tục trong suốt quá trình chơi. Với Action RPG, một dạng core loop điển hình có thể được mô tả như sau[4]:



Hình 1: Sơ đồ ví dụ về Core Loop trong Action RPG

Một core loop hiệu quả cần:

- Đơn giản, dễ hiểu ngay từ những phút chơi đầu tiên.
- Mang lại cảm giác thoả mãn sau mỗi vòng lặp.
- Có nhịp độ đủ nhanh để tránh nhàm chán nhưng không gây quá tải.

Meta Loop là các hệ thống dài hạn giữ chân người chơi trong thời gian dài hơn, ví dụ[5]:

- Tăng cấp nhân vật và mở khoá thuộc tính.
- Mở khoá kỹ năng, class hoặc khu vực mới.
- Thu thập và nâng cấp trang bị.
- Hoàn thành mục tiêu dài hạn (thành tựu, cốt truyện, mô hình hoá sức mạnh nhân vật).

2.2.2 Progression System (Hệ thống phát triển nhân vật)

Hệ thống progression định hình hành trình phát triển của nhân vật theo thời gian. Các thành phần thường gặp[4, 5]:

- **Leveling**: hệ thống lên cấp, giúp nhân vật tăng chỉ số hoặc mở khoá kỹ năng mới.
- **Stats**: các chỉ số như sức mạnh, nhanh nhẹn, trí lực, thể lực, ảnh hưởng đến khả năng chiến đấu.
- **Equipment Progression**: tiến trình nâng cấp trang bị, vũ khí, giáp và vật phẩm hỗ trợ.
- **Skill Tree**: cây kỹ năng dạng nhánh cho phép người chơi xây dựng phong cách chơi riêng.
- **Unlocking System**: cơ chế mở khoá tính năng, khu vực hoặc nội dung mới dựa trên tiến độ.

2.2.3 Combat Design (Thiết kế chiến đấu)

Chiến đấu là trung tâm của Action RPG. Các khái niệm nền tảng[4, 5]:

Pacing (nhịp độ chiến đấu) Nhịp độ chiến đấu có thể:

- **Nhanh**: nhấn mạnh phản xạ, né tránh, ra quyết định tức thời.
- **Chậm**: thiên về quan sát, ra quyết định chiến lược, tính toán khoảng cách và thời điểm.

Frame Data Mỗi hành động chiến đấu có thể được phân tích theo "frame data":

- **Startup**: thời gian chuẩn bị trước khi đòn đánh có hiệu lực.
- **Active**: khoảng thời gian hitbox có thể gây sát thương.
- **Recovery**: thời gian hồi lại trước khi có thể thực hiện hành động khác.
- **Invincibility Frames (i-frames)**: khoảng thời gian nhân vật không thể bị trúng đòn (thường xuất hiện trong các động tác lăn hoặc né).

Hit Detection Hit detection là quá trình kiểm tra xem một đòn tấn công có trúng mục tiêu hay không, dựa trên:

- Hình dạng và vị trí hitbox.
- Hình dạng và vị trí hurtbox.
- Thời điểm kích hoạt trong animation.

2.2.4 World System và Level Design

Level design quyết định cách người chơi di chuyển, tương tác và trải nghiệm không gian trò chơi. Một số yếu tố quan trọng[5]:

- Bố cục không gian (layout) rõ ràng, tránh gây lạc hướng.
- Đường di chuyển chính/phụ, nhánh rẽ tạo cảm giác khám phá.
- Phân bố quái, vật phẩm, checkpoint hợp lý.
- Xen kẽ các đoạn chiến đấu, khám phá và nghỉ ngơi để tạo nhịp độ cân bằng.

2.2.5 Hệ thống kinh tế (Game Economy)

Hệ thống kinh tế trong game quy định cách tài nguyên được tạo ra và sử dụng[4].

Nguồn tài nguyên (Resource Sources) Ví dụ:

- Phần thưởng nhiệm vụ.
- Quái rơi vật phẩm.
- Thu thập và khai thác tài nguyên.
- Hệ thống crafting.
- Sự kiện trong game.

Tiêu thụ tài nguyên (Resource Sinks) Ví dụ:

- Nâng cấp trang bị.
- Mua vật phẩm tiêu hao.
- Mở khoá tính năng hoặc khu vực mới.
- Chi phí duy trì hoặc sửa chữa trang bị.

2.2.6 UX/UI Design trong game

Trải nghiệm người dùng (UX) và giao diện người dùng (UI) đóng vai trò quan trọng trong việc truyền tải thông tin và hỗ trợ người chơi tương tác với hệ thống[5].

Các nguyên lý cơ bản:

- **Phản hồi rõ ràng (Visual Feedback)**: mọi hành động quan trọng (nhặt vật phẩm, nâng cấp, gây sát thương) cần có hiệu ứng hình ảnh và/hoặc âm thanh tương ứng.
- **Giao diện nhất quán**: vị trí, màu sắc và hành vi của các nút, menu cần được duy trì nhất quán giữa các màn hình.
- **Giảm tải thông tin**: chỉ hiển thị thông tin cần thiết trong mỗi ngữ cảnh, tránh làm người chơi bị "ngợp" vì quá nhiều chi tiết.

2.2.7 Balancing Fundamentals (Cân bằng game)

Cân bằng (balancing) là quá trình điều chỉnh các thông số trong game sao cho trải nghiệm tổng thể hợp lý[4, 5].

Các khái niệm nền tảng:

- **Linear vs Exponential Growth**: tốc độ tăng chỉ số tuyến tính hay lũy thừa.
- **Time-to-Kill (TTK)**: thời gian trung bình để hạ gục một mục tiêu.
- **Risk–Reward Ratio**: tỉ lệ giữa rủi ro mà người chơi chấp nhận và phần thưởng nhận được.
- **Power Budget**: giới hạn tổng sức mạnh cho phép trong một hệ hoặc một build nhân vật.

2.3 Tổng kết chương

Chương này đã trình bày các kiến thức nền tảng về Action RPG, hệ thống multiplayer và các khái niệm cốt lõi trong thiết kế game: core loop, progression, combat, economy và UX. Những kiến thức này sẽ được sử dụng làm cơ sở cho các chương tiếp theo, đặc biệt là trong việc phân tích yêu cầu (Chương 5) và thiết kế hệ thống (Chương 7).

3 Công nghệ sử dụng

Chương này trình bày các công nghệ và nền tảng phần mềm được sử dụng hoặc định hướng sử dụng để hiện thực hoá một trò chơi Action RPG trực tuyến nhiều người chơi. Thay vì chỉ liệt kê tên công nghệ, nội dung tập trung phân tích vai trò, ưu điểm, hạn chế và lý do các công nghệ này phù hợp với bối cảnh một hệ thống game online thời gian thực[3, 2].

Về tổng thể, một hệ thống game online có thể được chia thành các lớp chính:

- Lớp **client/game engine**: hiển thị đồ họa, nhận input và xử lý trải nghiệm người chơi.
- Lớp **backend**: xử lý logic game phía server, xác thực, quản lý người chơi, phiên chơi và đồng bộ trạng thái.
- Lớp **dữ liệu**: lưu trữ lâu dài tiến trình người chơi, cấu hình game và các dữ liệu phụ trợ.
- Lớp **hạ tầng và công cụ**: phục vụ phát triển, kiểm thử, triển khai và giám sát hệ thống.

3.1 Kiến trúc tổng quát nhiều lớp

Một trò chơi Action RPG online thường được triển khai theo kiến trúc nhiều lớp (multi-tier)[3]:

- **Lớp trình bày (Presentation Layer)**: game client chạy trên máy người chơi (PC, console hoặc mobile), được xây dựng bằng game engine (ví dụ: Unity). Lớp này chịu trách nhiệm kết xuất hình ảnh, xử lý input, hiển thị giao diện và phản hồi tương tác.
- **Lớp dịch vụ ứng dụng (Application/Backend Layer)**: tập hợp các dịch vụ server xử lý logic game, API, đồng bộ trạng thái, match-making, quản lý phiên chơi và các nghiệp vụ liên quan.
- **Lớp dữ liệu (Data Layer)**: bao gồm các cơ sở dữ liệu quan hệ hoặc NoSQL, hệ thống cache, kho lưu trữ file cấu hình, log, v.v.

Việc tách bạch các lớp này giúp:

- Dễ **mở rộng** (scale) từng lớp độc lập.
- Thuận tiện **thay thế công nghệ** ở một lớp mà không ảnh hưởng toàn hệ thống.
- Tăng **tính bảo trì** và **khả năng tái sử dụng** các thành phần.

3.2 Công nghệ phía client (Game Engine)

3.2.1 Game engine cho Action RPG

Để phát triển nhanh một trò chơi Action RPG có đồ họa 2D/3D, các game engine phổ biến như Unity hoặc Unreal Engine thường được sử dụng. Trong phạm vi đề tài, Unity được lựa chọn vì[6]:

- Cung cấp sẵn hệ thống **rendering**, animation, vật lý cơ bản và quản lý scene.
- Có **asset pipeline** tương đối hoàn chỉnh (nhập mô hình 3D, texture, animation từ các công cụ bên ngoài).
- Hỗ trợ tốt việc xây dựng **UI in-game**, hệ thống input đa nền tảng và xử lý đa phương tiện.
- Sở hữu **cộng đồng lớn**, tài liệu phong phú, nhiều plugin và asset sẵn có, phù hợp với điều kiện tự học và tra cứu.

Việc sử dụng Unity giúp nhóm phát triển tập trung hơn vào thiết kế gameplay, cơ chế mạng và kiến trúc hệ thống, thay vì phải tự xây dựng từ đầu các thành phần nền tảng phức tạp như renderer hay physics engine[3].

3.2.2 Ngôn ngữ lập trình phía client

Với Unity, ngôn ngữ lập trình chủ đạo là **C#**[12]. Một số đặc điểm nổi bật:

- Cú pháp hiện đại, hỗ trợ tốt lập trình hướng đối tượng.
- Thư viện phong phú cho xử lý cấu trúc dữ liệu, toán học, I/O, networking ở mức client.
- Tích hợp chặt chẽ với API của Unity, giúp hiện thực logic gameplay, UI và hiệu ứng một cách trực tiếp[6].

Khi lập trình phía client, cần lưu ý:

- Tách biệt **logic gameplay cục bộ** (UI, hiệu ứng hiển thị, animation) với **logic được server quyết định** (kết quả combat, trạng thái chính thức của nhân vật).
- Hạn chế lưu trữ và xử lý **dữ liệu nhạy cảm** phía client để giảm nguy cơ gian lận.
- Tổ chức mã nguồn theo hướng **component-based** hoặc **entity–component–system (ECS)** để dễ mở rộng và bảo trì[3].

3.3 Công nghệ phía server (Backend)

Backend là nơi xử lý logic game phía server, đóng vai trò "nguồn chân lý" (source of truth) cho trạng thái thế giới game[2]. Một backend cho game online thường bao gồm:

- Dịch vụ xác thực và quản lý người dùng (authentication, account management).
- Dịch vụ quản lý phiên chơi (session, room, instance).
- Dịch vụ xử lý logic game (chiến đấu, nhiệm vụ, sự kiện).
- Dịch vụ truy xuất và cập nhật dữ liệu (progress nhân vật, cấu hình game).

3.3.1 Lựa chọn nền tảng backend: Node.js và NestJS

Trong phạm vi đề tài, backend được định hướng xây dựng bằng **Node.js** kết hợp với **NestJS** và **TypeScript**[13, 7]. Lý do lựa chọn:

- Node.js hỗ trợ tốt **lập trình bất đồng bộ**, phù hợp với hệ thống có nhiều kết nối WebSocket đồng thời.
- NestJS cung cấp **kiến trúc mô-đun rõ ràng**, dựa trên decorator và dependency injection, giúp tổ chức mã nguồn backend khoa học và dễ mở rộng.
- TypeScript giúp mã nguồn **có kiểu tường minh**, hỗ trợ kiểm tra lỗi tại thời điểm biên dịch, giảm rủi ro trong phát triển trung và dài hạn.
- Hệ sinh thái phong phú: nhiều thư viện cho REST API, WebSocket, ORM, JWT, phân quyền, logging, v.v.

3.3.2 Mô hình dịch vụ và tổ chức logic backend

Về kiến trúc, backend có thể triển khai theo[2]:

- **Monolithic**: toàn bộ logic và API tập trung trong một ứng dụng NestJS duy nhất.
- **Module hoá**: bên trong ứng dụng monolithic, chia nhỏ thành các module như User, Auth, Character, Session, v.v., sẵn sàng để tách thành microservices khi hệ thống mở rộng.

3.4 Công nghệ networking và giao tiếp thời gian thực

3.4.1 Giao tiếp client–server

Game online thường sử dụng kết hợp hai cơ chế giao tiếp[2]:

- **HTTP/REST**:

  - Phù hợp với các tác vụ không yêu cầu độ trễ thấp, như: đăng ký, đăng nhập, tải cấu hình, truy vấn thông tin tĩnh.
  - Trong NestJS, có thể hiện thực nhanh bằng các controller REST tiêu chuẩn[7].

- **WebSocket**:
  - Phù hợp với truyền thông hai chiều thời gian thực: cập nhật vị trí, trạng thái chiến đấu, sự kiện trong thế giới game.
  - Giảm overhead so với việc liên tục mở và đóng kết nối HTTP.
  - NestJS cung cấp sẵn hỗ trợ WebSocket gateway, giúp tích hợp với client Unity thông qua các thư viện WebSocket tương ứng[7].

3.4.2 Quản lý kết nối và phiên chơi

Một backend real-time cần[2]:

- Quản lý **phiên kết nối WebSocket** cho từng người chơi (mapping giữa user ID và socket).
- Ánh xạ người chơi vào **phòng/instance** tương ứng (ví dụ: khu vực chung, dungeon, map thử nghiệm).
- Xử lý ngắt kết nối, timeout và cơ chế reconnect khi người chơi mất kết nối tạm thời.

3.5 Công nghệ lưu trữ và cơ sở dữ liệu

3.5.1 Cơ sở dữ liệu NoSQL (MongoDB)

Trong phạm vi đề tài, cơ sở dữ liệu chính được định hướng sử dụng là **MongoDB**[8]. Lý do:

- Dữ liệu game có cấu trúc linh hoạt (nhân vật, vật phẩm, trạng thái nhiệm vụ, cấu hình NPC, cấu hình map).
- Schema thường thay đổi trong quá trình phát triển, NoSQL cho phép thích ứng dễ dàng hơn so với các hệ quan hệ.
- Hỗ trợ tốt document lồng nhau, phù hợp với các cấu trúc dữ liệu phức hợp của nhân vật và cấu hình game.

3.5.2 Hệ thống cache (Redis)

Redis được sử dụng ở vai trò[9]:

- **Cache** cho dữ liệu truy cập thường xuyên (thông tin người chơi đang online, token, một phần cấu hình game).
- **Session store**: lưu token phiên đăng nhập và thông tin phiên làm việc.
- **Pub/Sub**: truyền sự kiện giữa nhiều instance backend trong trường hợp cần mở rộng theo chiều ngang.

3.6 Công cụ hỗ trợ phát triển và vận hành

3.6.1 Quản lý mã nguồn và làm việc nhóm

- **Git**: hệ thống quản lý phiên bản phân tán, hỗ trợ làm việc nhóm trên cùng một codebase.
- **GitHub**: nền tảng lưu trữ mã nguồn, hỗ trợ issue tracking, pull request và tích hợp CI/CD[10].
- **GitHub Projects**: hệ thống quản lý tác vụ dựa trên bảng Kanban, cho phép phân chia, gán trách nhiệm, theo dõi tiến độ và liên kết tác vụ với issue/pull request tương ứng[11].

3.6.2 Công cụ thiết kế và tài liệu

- **Công cụ vẽ UML/BPMN** (Draw.io, Mermaid): dùng để mô tả use case, luồng xử lý, kiến trúc hệ thống[14].
- **Công cụ thiết kế giao diện** (Figma, Piskel, Stable Diffusion): dùng để phác thảo giao diện và trải nghiệm người dùng trước khi hiện thực trên Unity.
- **Công cụ quản lý tài liệu** (TeX, Markdown): dùng để viết Game Design Document (GDD), tài liệu thiết kế hệ thống và báo cáo học thuật[15].

3.7 Tổng kết

Chương này đã trình bày các lớp công nghệ chính liên quan đến việc xây dựng một trò chơi Action RPG trực tuyến: game engine phía client (Unity), nền tảng backend xử lý logic (Node.js, NestJS, TypeScript), cơ sở dữ liệu NoSQL (MongoDB) và hệ thống cache (Redis), cùng các công cụ hỗ trợ phát triển và vận hành (Git, GitHub, GitHub Projects, các công cụ thiết kế và tài liệu). Việc hiểu rõ vai trò và đặc điểm của từng công nghệ giúp hình thành một kiến trúc tổng thể hợp lý, là nền tảng cho các quyết định thiết kế chi tiết ở các chương tiếp theo, đặc biệt là phân tích yêu cầu (Chương 5) và thiết kế hệ thống (Chương 7).

4 Các công trình liên quan

Chương này tổng hợp và phân tích các công trình đã tham khảo trong quá trình hình thành ý tưởng và định hướng thiết kế cho đề tài *Fortress of the Fallen*. Các nguồn tham khảo được chia làm hai nhóm: (i) tác phẩm/phim/game làm nguồn cảm hứng thiết kế nội dung và hệ thống gameplay; (ii) tài liệu kỹ thuật định hướng kiến trúc hệ thống multiplayer và công nghệ triển khai.

Mục tiêu của chương là làm rõ mối liên hệ **Nguồn tham khảo → Bài học rút ra → Cách áp dụng vào đề tài**, từ đó tránh tình trạng liệt kê cảm hứng một cách chung chung.

4.1 Khung phân tích và tiêu chí lựa chọn nguồn tham khảo

Nhóm lựa chọn các nguồn tham khảo dựa trên các tiêu chí:

- Có cơ chế **tiến trình** (progression) và **động lực dài hạn** rõ ràng, phù hợp với Action RPG online.
- Có ít nhất một hệ thống đặc trưng có thể trừu tượng hoá thành mô-đun thiết kế (ví dụ: leo tháp, gacha, căn cứ/đảo cá nhân, tổ đội, raid/boss).
- Có đủ thông tin để phân tích theo các lớp: **vòng lặp chơi (loop)**, **hệ thống phần thưởng**, **mức độ rủi ro**, **tương tác người chơi**, **tổ chức nội dung**.

Ngoài ra, nhóm sử dụng thêm các tài liệu nền tảng về game design và kiến trúc multiplayer để đối chiếu và hợp thức hoá các quyết định thiết kế ở góc độ kỹ thuật và học thuật [5, 4, 16, 6, 7, 17, 18].

4.2 Nhóm nguồn cảm hứng về cấu trúc tiến trình: Leo tháp, boss và checkpoint

4.2.1 Sword Art Online (Season 1): trục nội dung leo tầng và boss tầng

Sword Art Online mùa 1 mô tả thế giới game với cấu trúc **leo tầng tuyến tính**, trong đó mỗi tầng có hệ sinh thái, nhiệm vụ và đặc biệt là **boss tầng** đóng vai trò "cổng kiểm soát tiến trình" [19]. Từ góc độ thiết kế hệ thống, SAO gợi ý ba điểm quan trọng:

- **Phân lớp nội dung (content stratification):** nội dung được chia tầng giúp kiểm soát độ khó theo tiến trình, dễ thiết kế nhịp tăng trưởng sức mạnh.
- **Boss là mốc kiểm chứng build:** boss tầng buộc người chơi tối ưu hoá trang bị/kỹ năng, từ đó tạo động lực chuẩn bị và tương tác tổ đội.
- **Checkpoint rõ ràng:** mỗi lần qua tầng là một "thành tựu" (achievement milestone), phù hợp để gắn thưởng và mở khoá.

Áp dụng vào đề tài: Đề tài kế thừa ý tưởng **Tháp trung tâm 100 tầng** như trục nội dung dài hạn. Tuy nhiên, thay vì tuyến tính tuyệt đối, hệ thống được định hướng thiết kế theo dạng:

- **Tầng chuẩn (main floors):** đi theo tiến trình chính.
- **Tầng thử thách (challenge/optional):** tăng độ khó, thưởng cao hơn.
- **Cơ chế checkpoint:** lưu tiến độ theo mốc để giảm "cảm giác mất trắng", phù hợp trải nghiệm game hiện đại.

Trong giai đoạn 1, nội dung leo tháp được coi là **tính năng nâng cao** (không hiện thực), nhưng được đưa vào thiết kế tổng quan để định hướng kiến trúc dữ liệu và mô-đun hoá từ sớm.

4.3 Nhóm nguồn cảm hứng về thế giới sống: NPC, vận hành thế giới và vai trò tổ chức

4.3.1 Overlord: NPC có vai trò, trật tự và vận hành thế giới

Overlord nhấn mạnh cảm giác "thế giới vận hành độc lập", trong đó NPC không chỉ là đối tượng đứng yên mà là thành phần có **vai trò, nhiệm vụ và quan hệ** [20]. Các bài học thiết kế rút ra:

- **NPC như tác nhân hệ thống (system agents):** NPC có thể tham gia vận hành tài nguyên, phòng thủ, sản xuất, hoặc mở tuyến nhiệm vụ.
- **Tầng tổ chức tạo chiều sâu:** khi NPC gắn với chức trách, người chơi có động lực xây dựng đội hình/nhân sự thay vì chỉ tăng chỉ số.
- **Tính bền vững của tiến trình:** thế giới không "reset" theo phiên chơi; tiến trình có tính tích luỹ.

Áp dụng vào đề tài: Đề tài định hướng NPC như tài nguyên dài hạn phục vụ phát triển căn cứ/đảo cá nhân (mô hình "nhân sự vận hành"). Tuy nhiên, theo phạm vi giai đoạn 1, phần NPC và đảo cá nhân được xếp vào **tính năng nâng cao** và chỉ dừng ở mức thiết kế/định hướng dữ liệu.

4.4 Nhóm nguồn cảm hứng về cơ chế tuyển dụng/thu thập: Gacha và meta-progression

4.4.1 Pick Me Up! Infinite Gacha: gacha như hệ thống meta và động lực dài hạn

Pick Me Up! Infinite Gacha cung cấp khung tham khảo cho cơ chế **tuyển dụng qua gacha** và xây dựng meta-progression dựa trên bộ sưu tập (collection) [21]. Các điểm đáng chú ý:

- **Độ hiếm (rarity) và vai trò:** mỗi thực thể có độ hiếm và chức năng khác nhau, từ đó tạo động lực tối ưu đội hình.
- **Gacha tạo vòng lặp meta:** chơi để kiếm tài nguyên → quay gacha → tối ưu → chơi nội dung khó hơn.
- **Giảm "dead-end":** luôn có khả năng cải thiện thông qua hệ thống thu thập và nâng cấp.

Áp dụng vào đề tài: Đề tài sử dụng gacha như một ý tưởng cho hệ thống NPC/nhân sự (nâng cao). Về thiết kế, gacha không được dùng để ép buộc trả phí trong đồ án, mà đóng vai trò **cơ chế tạo biến thiên tiến trình** và **động lực sưu tầm**.

4.5 Nhóm nguồn cảm hứng về trải nghiệm Action RPG: nhịp combat, phản hồi và điều khiển

4.5.1 Arcane Odyssey: cảm giác điều khiển và tương tác thời gian thực trong online RPG

Arcane Odyssey là tham chiếu về cách tổ chức trải nghiệm chiến đấu và di chuyển trong môi trường online [22]. Các bài học rút ra:

- **Phản hồi tức thời (immediate feedback):** người chơi cần thấy rõ kết quả của thao tác (đòn đánh, né, hiệu ứng).
- **Nhấn mạnh kỹ năng người chơi:** kết quả phụ thuộc vào thao tác và vị trí (positioning) thay vì chỉ chỉ số.
- **Tối giản rào cản thao tác:** UI và input không được cản trở core loop.

Áp dụng vào đề tài: Đề tài định hướng combat của *Fortress of the Fallen* theo hướng hack-and-slash: nhịp nhanh, có né tránh, có hit timing. Trong giai đoạn 1, hệ thống combat chưa hiện thực đầy đủ, nhưng các yêu cầu về **real-time sync** và **độ trễ thấp** được đưa vào phần phi chức năng và kiến trúc networking.

4.5.2 Soul Knight Prequel: vòng lặp ngắn, build đơn giản và tái chơi

Soul Knight Prequel là tham chiếu về tổ chức vòng lặp chơi ngắn nhưng lặp lại nhiều lần [23]. Điểm rút ra:

- **Session ngắn:** phù hợp người chơi phổ thông, dễ quay lại.
- **Reward rõ ràng theo phiên:** kết thúc phiên có thưởng để duy trì động lực.
- **Build dễ hiểu:** cho phép người chơi thử nghiệm mà không bị quá tải.

Áp dụng vào đề tài: Đề tài định hướng nội dung dungeon/thử thách theo dạng phiên (instance-based), dễ đóng gói thành mô-đun server về sau.

4.6 Nhóm nguồn cảm hứng về căn cứ/đảo cá nhân: phát triển dài hạn và quản trị tài nguyên

4.6.1 Clash of Clans: căn cứ cá nhân và tiến trình phát triển theo tài nguyên

Clash of Clans là tham chiếu nổi bật cho cơ chế căn cứ cá nhân: người chơi đầu tư tài nguyên để nâng cấp công trình theo thời gian, tạo meta-progression bền vững [24]. Các bài học quan trọng:

- **Đảo/căn cứ như "nhà" của người chơi:** tạo gắn bó và mục tiêu dài hạn.
- **Quản trị tài nguyên:** nguồn (sources) và nơi tiêu (sinks) được thiết kế cân bằng để duy trì nhịp chơi.
- **Tăng trưởng theo thời gian:** cơ chế thời gian giúp tạo động lực quay lại (return loop).

Áp dụng vào đề tài: Đề tài đề xuất **đảo vệ tinh cá nhân** như hệ thống nâng cao để mở rộng meta-progression (kết hợp NPC vận hành). Tuy nhiên, ở giai đoạn 1, đảo cá nhân và NPC được xác định là **tính năng nâng cao**, chỉ đưa vào phạm vi thiết kế định hướng chứ không hiện thực.

4.7 Tổng hợp: Bảng đối chiếu nguồn tham khảo và phần áp dụng

Bảng 1 tóm tắt mối liên hệ giữa nguồn tham khảo và cách áp dụng vào đề tài nhằm đảm bảo tính minh bạch trong lập luận thiết kế.

Nguồn tham khảo	Yếu tố rút ra	Áp dụng vào đề tài
SAO S1 [19]	Leo tầng tuyến tính, boss tầng, checkpoint tiến trình	Định hướng tháp 100 tầng (nâng cao), phân lớp nội dung và checkpoint
Overlord [20]	NPC có vai trò, thế giới vận hành, tầng tổ chức	Định hướng NPC vận hành hệ thống (nâng cao), thiết kế dữ liệu NPC/role
Pick Me Up! [21]	Gacha tuyển dụng, rarity, meta-progression	Định hướng gacha NPC (nâng cao), vòng lặp sưu tầm và tối ưu
Arcane Odyssey [22]	Cảm giác điều khiển, phản hồi combat, real-time online	Định hướng combat hack-and-slash và yêu cầu real-time sync (phi chức năng)
Soul Knight Prequel [23]	Vòng lặp ngắn, reward rõ, build dễ hiểu	Định hướng dungeon/instance theo phiên (nâng cao)
Clash of Clans [24]	Căn cứ cá nhân, quản trị tài nguyên, tiến trình theo thời gian	Định hướng đảo cá nhân và hệ thống công trình/tài nguyên (nâng cao)

Bảng 1: Đối chiếu nguồn tham khảo và hướng áp dụng vào đề tài

4.8 Tham khảo kỹ thuật và liên hệ với kiến trúc hệ thống

Bên cạnh nguồn cảm hứng nội dung, đề tài cần nền tảng kỹ thuật để hiện thực hoá game online thời gian thực. Do đó, nhóm tham khảo các tài liệu về:

- **Game design và cách lập luận thiết kế hệ thống gameplay** [5, 4].
- **Networking multiplayer** (server authoritative, prediction, interpolation) [16].
- **Công nghệ triển khai** ở mức đồ án: Unity cho client [6], NestJS cho backend [7], MongoDB [17], Redis [18].

Các tài liệu này được dùng để: (i) chuẩn hoá thuật ngữ và lập luận trong báo cáo; (ii) làm cơ sở cho các quyết định kiến trúc ở Chương 7; (iii) đảm bảo phần hiện thực giai đoạn 1 bám theo các thực hành phổ biến thay vì triển khai tuỳ hứng.

4.9 Kết luận chương

Chương này đã phân tích các công trình liên quan theo hướng "tham khảo có kiểm soát":

- Các nguồn như SAO, Overlord, Pick Me Up, Arcane Odyssey, Soul Knight Prequel và Clash of Clans cung cấp khung ý tưởng cho hệ thống leo tháp, NPC, gacha và đảo cá nhân [19, 20, 21, 22, 23, 24].

- Các tài liệu kỹ thuật cung cấp nền tảng để hiện thực kiến trúc multiplayer và lựa chọn công nghệ phù hợp phạm vi đồ án [5, 4, 16, 6, 7, 17, 18].

Các kết quả tổng hợp là cơ sở để xác định yêu cầu (Chương 5) và thiết kế hệ thống (Chương 7) theo hướng nhất quán, có thể mở rộng.

5 Phân tích yêu cầu

Chương này trình bày quá trình phân tích yêu cầu cho hệ thống trò chơi Action RPG trực tuyến *Fortress of the Fallen*. Mục tiêu của chương là xác định rõ phạm vi hiện thực, các yêu cầu chức năng và phi chức năng của hệ thống trong giai đoạn 1, làm cơ sở cho các bước phân tích hệ thống, thiết kế và hiện thực ở các chương tiếp theo.

5.1 Mục tiêu của hệ thống

Mục tiêu của hệ thống trong giai đoạn 1 là xây dựng nền tảng kỹ thuật cho một trò chơi Action RPG trực tuyến nhiều người chơi, đóng vai trò như một prototype học thuật nhằm kiểm chứng các quyết định về kiến trúc và công nghệ.

Cụ thể, hệ thống hướng đến các mục tiêu sau:

- Xây dựng mô hình client–server cho game online thời gian thực.
- Cho phép nhiều người chơi kết nối và tương tác đồng thời trong cùng một môi trường thử nghiệm.
- Đồng bộ trạng thái nhân vật giữa client và server một cách nhất quán.
- Thiết kế kiến trúc backend và mô hình dữ liệu có khả năng mở rộng cho các giai đoạn phát triển tiếp theo.
- Đảm bảo phạm vi hiện thực phù hợp với quy mô một đồ án chuyên ngành.

Hệ thống ở giai đoạn này không đặt mục tiêu trở thành một sản phẩm game hoàn chỉnh, mà tập trung vào tính đúng đắn và hợp lý về mặt kỹ thuật.

5.2 Đối tượng sử dụng hệ thống

Hệ thống phục vụ hai nhóm đối tượng chính: người chơi và quản trị hệ thống.

5.2.1 Người chơi

Người chơi là đối tượng trực tiếp tương tác với hệ thống thông qua game client. Trong giai đoạn 1, người chơi có thể:

- Đăng ký và đăng nhập tài khoản.

- Kết nối đến server game.
- Tải dữ liệu nhân vật từ server.
- Điều khiển nhân vật di chuyển trong môi trường thử nghiệm.
- Nhận các cập nhật trạng thái theo thời gian thực từ server.

Các chức năng này chủ yếu nhằm kiểm chứng cơ chế kết nối và đồng bộ trạng thái, chưa tập trung vào nội dung gameplay phức tạp.

5.2.2 Quản trị hệ thống

Quản trị hệ thống là đối tượng ở mức kỹ thuật, bao gồm nhóm phát triển. Vai trò chính bao gồm:

- Quản lý cấu hình và vận hành server.
- Theo dõi trạng thái kết nối và log hệ thống.
- Kiểm thử và đánh giá các chức năng nền tảng.

Trong giai đoạn 1, hệ thống không yêu cầu giao diện quản trị riêng biệt.

5.3 Phạm vi hiện thực trong giai đoạn 1

5.3.1 Các chức năng thuộc phạm vi hiện thực

Trong giai đoạn 1, hệ thống tập trung hiện thực các chức năng nền tảng sau:

- Kiến trúc client–server cho game online.
- Hệ thống xác thực người dùng (đăng ký, đăng nhập).
- Kết nối real-time giữa client và server thông qua WebSocket.
- Quản lý phiên kết nối của người chơi.
- Tải và lưu trữ dữ liệu nhân vật cơ bản.
- Đồng bộ vị trí và trạng thái di chuyển của nhân vật trong môi trường thử nghiệm.

5.3.2 Các chức năng không thuộc phạm vi hiện thực

Các hệ thống sau chỉ được đề cập ở mức định hướng hoặc thiết kế, không được hiện thực trong giai đoạn 1:

- Hệ thống xây dựng và quản lý đảo cá nhân.
- Hệ thống NPC, AI NPC và cơ chế gacha.

- Hệ thống leo tháp nhiều tầng và các ngục phức tạp.
- Hệ thống class–race–skill hoàn chỉnh.
- Chế độ PvP Arena.
- Hệ thống cân bằng game ở quy mô lớn và tối ưu hiệu năng nâng cao.

Việc giới hạn phạm vi giúp đảm bảo đồ án tập trung vào các mục tiêu học thuật cốt lõi.

5.4 Yêu cầu chức năng

5.4.1 Yêu cầu chức năng phía người chơi

Hệ thống phải đáp ứng các yêu cầu chức năng sau đối với người chơi:

- Đăng ký tài khoản mới.
- Đăng nhập bằng tài khoản hợp lệ.
- Nhận phản hồi xác thực từ server.
- Tải thông tin nhân vật sau khi đăng nhập thành công.
- Tham gia môi trường game thử nghiệm.
- Điều khiển nhân vật di chuyển trong môi trường game.
- Nhận cập nhật trạng thái từ server theo thời gian thực.

5.4.2 Yêu cầu chức năng phía hệ thống

Hệ thống backend phải đảm bảo:

- Xác thực người dùng và quản lý phiên đăng nhập.
- Quản lý kết nối WebSocket cho từng người chơi.
- Nhận và xử lý input từ client.
- Cập nhật trạng thái nhân vật dựa trên input nhận được.
- Gửi snapshot trạng thái thế giới về cho client.
- Lưu trữ và truy xuất dữ liệu một cách nhất quán.

Server đóng vai trò server authoritative, là nguồn chân lý duy nhất cho trạng thái hệ thống.

5.5 Yêu cầu phi chức năng

5.5.1 Hiệu năng

- Hệ thống phải đảm bảo độ trễ chấp nhận được trong môi trường thử nghiệm.
- Có khả năng xử lý nhiều kết nối đồng thời ở quy mô phù hợp với đồ án.
- Đảm bảo trải nghiệm real-time không bị gián đoạn nghiêm trọng.

5.5.2 Bảo mật

- Mật khẩu người dùng phải được mã hoá khi lưu trữ.
- Client không được phép tự quyết định kết quả gameplay.
- Các dữ liệu nhạy cảm phải được xử lý phía server.

5.5.3 Khả năng mở rộng và bảo trì

- Kiến trúc backend phải được tổ chức theo hướng module hoá.
- Dễ dàng mở rộng thêm các hệ thống gameplay trong tương lai.
- Mã nguồn rõ ràng, dễ bảo trì.

5.5.4 Tính ổn định

- Hệ thống phải xử lý được trường hợp ngắt kết nối đột ngột.
- Không gây crash server khi một client gặp lỗi.
- Có cơ chế log để phục vụ kiểm thử và phân tích lỗi.

5.6 Yêu cầu dữ liệu

Hệ thống cần quản lý các nhóm dữ liệu chính sau:

- Thông tin tài khoản người dùng.
- Thông tin nhân vật và trạng thái cơ bản.
- Trạng thái phiên kết nối.
- Dữ liệu log phục vụ kiểm thử.

Các dữ liệu này phải được lưu trữ nhất quán và sẵn sàng mở rộng trong các giai đoạn tiếp theo.

5.7 Các ràng buộc và giả định

- Đề tài được thực hiện trong khuôn khổ đồ án chuyên ngành.
- Thời gian và nguồn lực thực hiện có hạn.
- Hệ thống không yêu cầu triển khai ở môi trường production.
- Một số quyết định thiết kế được đưa ra dựa trên tính khả thi học thuật.

5.8 Tổng kết chương

Chương này đã xác định rõ các yêu cầu chức năng và phi chức năng của hệ thống trong giai đoạn 1, đồng thời làm rõ phạm vi và các ràng buộc của đồ án. Những nội dung này là cơ sở trực tiếp cho việc phân tích hệ thống ở Chương 6 và thiết kế hệ thống ở Chương 7.

6 Phân tích hệ thống

Chương này trình bày phân tích hệ thống cho đề tài *Fortress of the Fallen* trong phạm vi giai đoạn 1. Mục tiêu của chương là chuyển hoá các yêu cầu đã nêu ở Chương 5 thành mô tả vận hành ở mức logic: hệ thống gồm những tác nhân nào, các chức năng chính được thể hiện qua các ca sử dụng (use case) nào, luồng xử lý nghiệp vụ diễn ra ra sao, dữ liệu nào được tạo/đọc/cập nhật, và các ràng buộc vận hành quan trọng. Nội dung chương là cơ sở trực tiếp để triển khai thiết kế kiến trúc, thiết kế API và thiết kế dữ liệu ở Chương 7.

6.1 Tổng quan hệ thống

Trong giai đoạn 1, hệ thống tập trung vào việc xây dựng bộ khung nền tảng cho game online nhiều người chơi theo mô hình **client–server**. Các thành phần chính gồm:

- **Game Client (Unity)**: hiển thị, xử lý input, giao diện đăng nhập và luồng vào game; gửi yêu cầu (HTTP) và trao đổi sự kiện thời gian thực (WebSocket).
- **Backend (NestJS/Node.js)**: cung cấp API xác thực, quản lý phiên, xử lý logic thời gian thực tối thiểu phục vụ prototype (ví dụ: đồng bộ vị trí/di chuyển mức cơ bản).
- **Cơ sở dữ liệu (MongoDB)**: lưu trữ dữ liệu bền vững như tài khoản, hồ sơ người chơi, thông tin nhân vật.
- **Cache/Session (Redis)**: lưu session, token hoặc dữ liệu truy cập thường xuyên (tuỳ mức hiện thực giai đoạn 1).

Nguyên tắc vận hành cốt lõi của hệ thống là **server authoritative**: server quyết định trạng thái chính thức (đặc biệt với các dữ liệu có thể ảnh hưởng đến công bằng và tính nhất quán), còn client ưu tiên trải nghiệm hiển thị và phản hồi điều khiển.

6.2 Tác nhân của hệ thống (Actors)

Các tác nhân tham gia tương tác với hệ thống được xác định như sau:

Tác nhân	**Mô tả**
Người chơi	Sử dụng game client; thực hiện đăng ký/đăng nhập; điều khiển nhân vật; tham gia phiên chơi.
Game Client (Unity)	Đại diện cho người chơi để gửi/nhận dữ liệu; xử lý UI; gửi request HTTP và sự kiện WebSocket.
Backend (NestJS)	Xử lý xác thực, cấp token, quản lý phiên và kênh real-time; xác nhận dữ liệu nhân vật; phát trạng thái cho client.
MongoDB	Lưu trữ dữ liệu bền vững: User, Character và các metadata liên quan.
Redis (tuỳ chọn)	Lưu session, token blacklist/whitelist, cache dữ liệu nóng; hỗ trợ pub/sub nếu mở rộng nhiều instance.

Bảng 2: Các tác nhân của hệ thống

6.3 Phạm vi chức năng phân tích trong giai đoạn 1

Theo định hướng giai đoạn 1, hệ thống phân tích tập trung vào các nhóm chức năng nền tảng sau:

- **Nhóm xác thực & tài khoản**: đăng ký, đăng nhập, đăng xuất, quản lý token.
- **Nhóm nhân vật**: tải dữ liệu nhân vật sau đăng nhập; chọn nhân vật (nếu có nhiều slot); khởi tạo nhân vật tối thiểu.
- **Nhóm phiên chơi & kết nối real-time**: thiết lập kết nối WebSocket; tham gia "phiên chơi" thử nghiệm; đồng bộ trạng thái cơ bản (ví dụ: di chuyển).
- **Nhóm vận hành tối thiểu**: xử lý mất kết nối, timeout, giới hạn request, log và theo dõi lỗi cơ bản.

Các tính năng nâng cao như đảo cá nhân, NPC, gacha, leo tháp/ngục, PvP, AI nâng cao **không thuộc phạm vi hiện thực của giai đoạn 1** và vì vậy chỉ xuất hiện như mục định hướng ở các chương sau, không đưa vào use case bắt buộc của giai đoạn này.

6.4 Danh sách Use Case

Danh sách ca sử dụng (use case) trong giai đoạn 1 được đề xuất như Bảng 3. Các use case được nhóm theo cụm chức năng để thuận tiện cho triển khai và kiểm thử.

Mã	Tên Use Case	Mô tả ngắn
UC01	Đăng ký tài khoản	Người chơi tạo tài khoản mới; hệ thống kiểm tra hợp lệ và lưu vào DB.
UC02	Đăng nhập	Người chơi đăng nhập; hệ thống xác thực và cấp token phiên.
UC03	Đăng xuất	Người chơi đăng xuất; hệ thống huỷ/đánh dấu token (tuỳ chính sách).
UC04	Tải hồ sơ người chơi	Client lấy thông tin tối thiểu của user (id, tên hiển thị, danh sách nhân vật).
UC05	Tạo nhân vật (tối thiểu)	Nếu chưa có nhân vật, client tạo mới với dữ liệu cơ bản.
UC06	Chọn nhân vật	Người chơi chọn nhân vật để vào phiên chơi thử nghiệm.
UC07	Thiết lập kết nối real-time	Client mở WebSocket sau đăng nhập; xác thực kênh bằng token.
UC08	Tham gia phiên chơi thử nghiệm	Server gán player vào một "room/instance" đơn giản để test.
UC09	Gửi input di chuyển	Client gửi input/ý định di chuyển theo nhịp; server xác nhận và cập nhật trạng thái.
UC10	Nhận cập nhật trạng thái	Client nhận snapshot/packet trạng thái để hiển thị vị trí và chuyển động.
UC11	Xử lý mất kết nối	Khi WebSocket rớt, server giải phóng phiên; client hiển thị trạng thái và cho phép kết nối lại.

Bảng 3: Danh sách use case giai đoạn 1

6.5 Đặc tả Use Case chi tiết

Phần này đặc tả chi tiết các use case trọng yếu để làm cơ sở cho thiết kế API, thiết kế mô-đun và kiểm thử. Mỗi use case gồm: mục tiêu, tác nhân, tiền điều kiện, hậu điều kiện, luồng chính và luồng thay thế/ngoại lệ.

6.5.1 UC01 – Đăng ký tài khoản

Mục tiêu: Cho phép người chơi tạo tài khoản mới trong hệ thống.
Tác nhân chính: Người chơi (thông qua Game Client)
Tiền điều kiện:

- Người chơi chưa đăng nhập.
- Client có thể kết nối đến API backend (HTTP).

Hậu điều kiện:

- Tài khoản mới được lưu trong MongoDB.

- Người chơi có thể thực hiện UC02 (Đăng nhập).

Luồng chính:

1. Người chơi mở màn hình đăng ký và nhập thông tin (ví dụ: email/username, mật khẩu).
2. Client gửi request đăng ký đến backend.
3. Backend kiểm tra hợp lệ (định dạng, độ dài, chính sách mật khẩu).
4. Backend kiểm tra trùng lặp tài khoản (email/username) trong DB.
5. Nếu hợp lệ, backend băm mật khẩu và lưu user mới vào DB.
6. Backend trả về kết quả thành công cho client.
7. Client hiển thị thông báo đăng ký thành công và chuyển sang màn hình đăng nhập.

Luồng thay thế/ngoại lệ:

- **E01-1:** Thông tin không hợp lệ (mật khẩu quá ngắn, email sai định dạng) → backend trả lỗi; client hiển thị lỗi theo trường.
- **E01-2:** Tài khoản đã tồn tại → backend trả lỗi trùng; client gợi ý đăng nhập.
- **E01-3:** Lỗi DB (mất kết nối, timeout) → backend trả lỗi hệ thống; client hiển thị và cho phép thử lại.

6.5.2 UC02 – Đăng nhập

Mục tiêu: Xác thực người chơi và cấp token phiên để truy cập các chức năng sau.
Tác nhân chính: Người chơi (thông qua Game Client)
Tiền điều kiện:

- Người chơi đã có tài khoản (UC01).
- Client có thể gọi API backend.

Hậu điều kiện:

- Client nhận được token (ví dụ: JWT) và lưu tạm (memory/secure storage).
- Người chơi có thể thực hiện UC04, UC07.

Luồng chính:

1. Người chơi nhập thông tin đăng nhập.
2. Client gửi request đăng nhập tới backend.

3. Backend tìm user theo username/email.
4. Backend so sánh mật khẩu (verify hash).
5. Nếu đúng, backend tạo token phiên (kèm hạn dùng) và trả về cho client.
6. Client lưu token và chuyển sang màn hình tải dữ liệu người chơi (UC04).

Luồng thay thế/ngoại lệ:

- **E02-1:** Sai tài khoản hoặc mật khẩu → trả lỗi xác thực; client hiển thị thông báo chung.
- **E02-2:** Tài khoản bị khoá (nếu có chính sách) → trả lỗi trạng thái; client hiển thị thông tin.
- **E02-3:** Backend quá tải/timeout → client cho phép thử lại.

6.5.3 UC04 – Tải hồ sơ người chơi

Mục tiêu: Client lấy dữ liệu tối thiểu của user và danh sách nhân vật để quyết định tạo/chọn nhân vật.

Tác nhân chính: Game Client

Tiền điều kiện:

- Client đã có token hợp lệ (UC02).

Hậu điều kiện:

- Client có dữ liệu profile và danh sách character.
- Nếu chưa có nhân vật: chuyển UC05. Nếu có: chuyển UC06.

Luồng chính:

1. Client gửi request lấy profile kèm token.
2. Backend xác thực token.
3. Backend truy vấn DB lấy dữ liệu user và danh sách character liên quan.
4. Backend trả dữ liệu về client.
5. Client hiển thị màn hình chọn nhân vật (UC06) hoặc tạo nhân vật (UC05).

Luồng thay thế/ngoại lệ:

- **E04-1:** Token hết hạn/không hợp lệ → backend trả lỗi; client đưa về màn hình đăng nhập.
- **E04-2:** Không tìm thấy user → lỗi hệ thống; client báo lỗi và yêu cầu đăng nhập lại.

6.5.4 UC05 – Tạo nhân vật (tối thiểu)

Mục tiêu: Cho phép người chơi khởi tạo nhân vật cơ bản để vào phiên chơi prototype.
Tác nhân chính: Người chơi (thông qua Game Client)
Tiền điều kiện:

- Đã đăng nhập và có token hợp lệ.
- Người chơi chưa có nhân vật hoặc còn slot trống.

Hậu điều kiện:

- Nhân vật mới được tạo và lưu DB.
- Client có thể chọn nhân vật để vào game (UC06).

Luồng chính:

1. Người chơi nhập thông tin nhân vật tối thiểu (tên nhân vật, lựa chọn ngoại hình cơ bản nếu có).
2. Client gửi request tạo nhân vật kèm token.
3. Backend xác thực token, kiểm tra ràng buộc (tên trống, độ dài, ký tự).
4. Backend tạo document character với các thuộc tính nền (level=1, vị trí khởi tạo, chỉ số mặc định).
5. Backend lưu vào DB và liên kết với user.
6. Backend trả kết quả thành công và dữ liệu nhân vật.
7. Client chuyển sang màn hình chọn nhân vật hoặc tự chọn nhân vật vừa tạo (UC06).

Luồng thay thế/ngoại lệ:

- **E05-1:** Tên nhân vật không hợp lệ/trùng → báo lỗi và yêu cầu nhập lại.
- **E05-2:** Hết slot nhân vật → báo lỗi giới hạn.

6.5.5 UC06 – Chọn nhân vật

Mục tiêu: Người chơi chọn một nhân vật để bắt đầu vào phiên chơi.
Tác nhân chính: Người chơi
Tiền điều kiện:

- Đã có danh sách nhân vật (UC04).

Hậu điều kiện:

- Client xác định characterId đang hoạt động.
- Chuẩn bị kết nối real-time và tham gia phiên chơi (UC07, UC08).

Luồng chính:

1. Client hiển thị danh sách nhân vật.
2. Người chơi chọn một nhân vật.
3. Client lưu characterId được chọn và chuyển sang bước kết nối real-time (UC07).

6.5.6 UC07 – Thiết lập kết nối real-time

Mục tiêu: Thiết lập kênh WebSocket để trao đổi dữ liệu thời gian thực.
Tác nhân chính: Game Client
Tiền điều kiện:

- Client có token hợp lệ và đã chọn characterId.

Hậu điều kiện:

- WebSocket được thiết lập thành công.
- Server gán socket với userId/characterId và tạo session runtime.

Luồng chính:

1. Client mở kết nối WebSocket đến backend.
2. Client gửi message "authenticate" kèm token và characterId (hoặc kèm qua query/headers tuỳ thiết kế).
3. Server xác thực token, kiểm tra quyền sở hữu characterId.
4. Nếu hợp lệ, server gán socket vào session và trả "auth_ok".
5. Client chuyển sang use case tham gia phiên chơi (UC08).

Luồng thay thế/ngoại lệ:

- **E07-1:** Token không hợp lệ/hết hạn → server từ chối, client quay lại đăng nhập.
- **E07-2:** CharacterId không thuộc user → server từ chối, client tải lại profile.
- **E07-3:** Mất kết nối khi bắt tay → client retry theo số lần giới hạn.

6.5.7 UC08 – Tham gia phiên chơi thử nghiệm

Mục tiêu: Đưa người chơi vào một phiên chơi (room/instance) đơn giản để kiểm thử đồng bộ.

Tác nhân chính: Game Client

Tiền điều kiện:

- WebSocket đã xác thực (UC07).

Hậu điều kiện:

- Người chơi được gán vào một room.
- Client nhận trạng thái khởi tạo (spawn position, snapshot ban đầu).

Luồng chính:

1. Client gửi message "join_session" (kèm thông tin map/test scene nếu cần).
2. Server tạo hoặc chọn một room thử nghiệm.
3. Server load dữ liệu nhân vật tối thiểu (vị trí, hướng, tốc độ) và tạo entity runtime.
4. Server trả message "join_ok" kèm snapshot ban đầu.
5. Client tạo nhân vật trong scene và bắt đầu vòng lặp gửi input di chuyển (UC09).

Luồng thay thế/ngoại lệ:

- **E08-1:** Room đầy (nếu có giới hạn) → server chuyển room khác.
- **E08-2:** Lỗi load dữ liệu nhân vật → server trả lỗi; client hiển thị và thử lại.

6.5.8 UC09 – Gửi input di chuyển

Mục tiêu: Cho phép client gửi ý định di chuyển để server cập nhật vị trí nhân vật theo thời gian thực.

Tác nhân chính: Game Client

Tiền điều kiện:

- Người chơi đã tham gia phiên chơi (UC08).

Hậu điều kiện:

- Server nhận input và cập nhật trạng thái nhân vật.
- Client và các client khác (nếu có) nhận được cập nhật để hiển thị.

Luồng chính:

1. Ở mỗi nhịp (tick client) hoặc khi input thay đổi, client gửi message "move_input" (hướng, tốc độ mong muốn, timestamp/seq).
2. Server nhận input và áp dụng vào vòng lặp tick của server.
3. Server kiểm tra ràng buộc (tốc độ tối đa, phạm vi hợp lệ, chống spam cơ bản).
4. Server cập nhật vị trí nhân vật trên trạng thái authoritative.
5. Server phát "state_update" định kỳ về cho client (UC10).

Luồng thay thế/ngoại lệ:

- **E09-1:** Input gửi quá nhanh (flood) → server throttle hoặc bỏ qua.
- **E09-2:** Dữ liệu sai định dạng → server cảnh báo và có thể đóng kết nối nếu lặp lại.

6.5.9 UC10 – Nhận cập nhật trạng thái

Mục tiêu: Client nhận snapshot/cập nhật trạng thái từ server để hiển thị chuyển động và vị trí nhất quán.

Tác nhân chính: Game Client

Tiền điều kiện:

- Đã kết nối và ở trong phiên (UC08).

Hậu điều kiện:

- Client cập nhật vị trí/animation theo snapshot.

Luồng chính:

1. Server định kỳ gửi message "state_update" (vị trí, vận tốc, hướng, timestamp).
2. Client nhận dữ liệu và lưu vào buffer snapshot.
3. Client nội suy (interpolation) để hiển thị mượt; nếu sai lệch lớn thì hiệu chỉnh (reconciliation) theo chính sách.

Ghi chú: Ở giai đoạn 1, có thể áp dụng cơ chế đơn giản: cập nhật thẳng vị trí theo server để giảm độ phức tạp; các kỹ thuật nội suy/dự đoán có thể nâng cấp ở giai đoạn sau.

6.5.10 UC11 – Xử lý mất kết nối

Mục tiêu: Đảm bảo hệ thống xử lý an toàn khi client mất kết nối hoặc thoát game.
Tác nhân chính: Game Client / Backend
Tiền điều kiện:

- Người chơi đang có session real-time.

Hậu điều kiện:

- Server giải phóng session runtime và tài nguyên room.
- Client hiển thị trạng thái mất kết nối và cho phép kết nối lại.

Luồng chính:

1. Kết nối WebSocket bị ngắt (do mạng hoặc người chơi thoát).
2. Server nhận sự kiện disconnect.
3. Server xoá mapping socket–user, cập nhật trạng thái "offline" runtime.
4. (Tuỳ chính sách) Server lưu vị trí/tiến trình tối thiểu vào DB nếu cần.
5. Client hiển thị thông báo mất kết nối và chuyển về màn hình đăng nhập hoặc nút "Reconnect".

Luồng thay thế/ngoại lệ:

- **E11-1:** Reconnect trong thời gian ngắn → server cho phép nối lại session nếu còn hợp lệ (tuỳ phạm vi).

6.6 Phân tích luồng nghiệp vụ tổng quát

Phần này mô tả các luồng nghiệp vụ tổng quát ở mức hệ thống, nhằm thể hiện sự liên kết giữa các use case.

6.6.1 Luồng A: Từ mở game đến vào phiên chơi

1. Người chơi mở game client.
2. Nếu chưa có tài khoản: thực hiện UC01 (Đăng ký).
3. Thực hiện UC02 (Đăng nhập) để nhận token.
4. Thực hiện UC04 (Tải hồ sơ người chơi).
5. Nếu chưa có nhân vật: UC05 (Tạo nhân vật).

6. UC06 (Chọn nhân vật).
7. UC07 (Thiết lập WebSocket và xác thực).
8. UC08 (Tham gia phiên chơi thử nghiệm).
9. Trong phiên: lặp UC09 (Gửi input di chuyển) và UC10 (Nhận cập nhật trạng thái).

6.6.2 Luồng B: Mất kết nối khi đang chơi

1. Người chơi đang trong phiên chơi.
2. Mạng mất hoặc client đóng bất thường.
3. Server kích hoạt UC11 (Xử lý mất kết nối).
4. Client hiển thị lỗi và cung cấp lựa chọn: thoát về đăng nhập hoặc thử reconnect.

6.7 Phân tích dữ liệu ở mức logic

Ở mức logic (chưa đi vào thiết kế schema chi tiết), hệ thống giai đoạn 1 cần các nhóm dữ liệu chính:

- **User**: định danh người chơi, thông tin đăng nhập (hash), trạng thái tài khoản.
- **Character**: thông tin nhân vật, thuộc tính nền (level, chỉ số cơ bản), trạng thái xuất hiện trong phiên (vị trí khởi tạo).
- **Session (runtime)**: trạng thái phiên kết nối real-time (socketId, roomId, userId, characterId); thường lưu ở bộ nhớ server và/hoặc Redis.

6.7.1 CRUD ở mức logic

Đối tượng	Create	Read	Update
User	UC01 (Đăng ký)	UC02/UC04 (xác thực, tải profile)	Cập nhật trạng thái/metadata (tuỳ phạm vi)
Character	UC05 (Tạo nhân vật)	UC04/UC06/UC08 (tải để chọn/join)	Cập nhật vị trí/tiến trình tối thiểu (tuỳ phạm vi)
Session (runtime)	UC07/UC08 (tạo session)	Trong tick real-time	UC09/UC11 (cập nhật trạng thái, giải phóng)

Bảng 4: Phân tích CRUD mức logic cho giai đoạn 1

6.8 Ràng buộc và giả định hệ thống

Các ràng buộc và giả định dưới đây được đặt ra để phù hợp nguồn lực và phạm vi giai đoạn 1:

- **Authoritative server**: server xác nhận trạng thái; client không quyết định dữ liệu quan trọng.
- **Prototype real-time**: đồng bộ di chuyển ở mức cơ bản; các cơ chế tối ưu mạng nâng cao (prediction/reconciliation hoàn chỉnh) có thể giản lược.
- **Một người chơi – một kết nối**: mỗi user chỉ duy trì một session WebSocket hoạt động tại một thời điểm (giả định), giúp đơn giản hoá quản lý phiên.
- **Giới hạn tải**: chưa tối ưu cho số lượng người chơi lớn; mục tiêu là đúng luồng và ổn định ở quy mô thử nghiệm.
- **Bảo mật tối thiểu**: áp dụng xác thực token, validate input; các cơ chế chống gian lận nâng cao sẽ xem xét ở giai đoạn sau.

6.9 Tiêu chí kiểm thử suy ra từ phân tích

Từ các use case đã phân tích, một số tiêu chí kiểm thử chức năng tối thiểu cho giai đoạn 1 gồm:

- UC01: Đăng ký thành công, đăng ký trùng, dữ liệu không hợp lệ.
- UC02: Đăng nhập đúng/sai, token trả về và lưu đúng.
- UC04: Tải profile với token hợp lệ; từ chối khi token hết hạn.
- UC05: Tạo nhân vật thành công; xử lý tên không hợp lệ.
- UC07–UC08: WebSocket auth ok; join session thành công; join fail khi token sai.
- UC09–UC10: Di chuyển gửi/nhận ổn định; trạng thái cập nhật theo nhịp.
- UC11: Mất kết nối được xử lý; server giải phóng session; client hiển thị phù hợp.

Các tiêu chí này sẽ được cụ thể hoá ở Chương 9 (Đánh giá/kiểm thử) dưới dạng test case hoặc checklist.

6.10 Tổng kết chương

Chương 6 đã phân tích hệ thống ở mức logic cho giai đoạn 1: xác định tác nhân, danh sách và đặc tả use case, luồng nghiệp vụ tổng quát, dữ liệu logic và các ràng buộc vận hành. Các kết quả phân tích là nền tảng để triển khai Chương 7 (Thiết kế hệ thống), trong đó các nội dung như kiến trúc mô-đun, thiết kế API, thiết kế room/session, thiết kế dữ liệu MongoDB/Redis và các sơ đồ (nếu có) sẽ được trình bày chi tiết.

7 Thiết kế hệ thống

7.1 Mục tiêu thiết kế

Mục tiêu của chương này là chuyển hoá các kết quả phân tích hệ thống ở Chương 6 thành các quyết định thiết kế cụ thể ở mức kiến trúc và mô-đun, làm cơ sở cho việc hiện thực hệ thống trong Chương 8.

Thiết kế hệ thống trong giai đoạn 1 hướng đến các mục tiêu chính sau:

- Xây dựng kiến trúc client–server rõ ràng, tuân theo mô hình server authoritative.
- Tổ chức hệ thống theo hướng module hoá để thuận tiện cho mở rộng.
- Đảm bảo hỗ trợ đầy đủ các yêu cầu chức năng và phi chức năng đã xác định.
- Giữ mức độ phức tạp phù hợp với phạm vi một đồ án chuyên ngành.

Thiết kế không nhằm tối ưu hoá cho môi trường production hay tải lớn, mà tập trung vào tính đúng đắn, rõ ràng và khả năng phát triển trong các giai đoạn tiếp theo.

7.2 Kiến trúc tổng thể hệ thống

7.2.1 Mô hình kiến trúc client–server

Hệ thống Fortress of the Fallen trong giai đoạn 1 được thiết kế theo mô hình client–server nhiều lớp, bao gồm các thành phần chính sau:

- **Game Client (Unity):**
  - Hiển thị đồ hoạ và giao diện người dùng.
  - Thu thập input từ người chơi.
  - Gửi request HTTP và message WebSocket đến server.
  - Hiển thị trạng thái nhận được từ server.
- **Backend Server (NestJS/Node.js):**
  - Xử lý xác thực và quản lý người dùng.
  - Quản lý phiên chơi và kết nối real-time.
  - Xử lý logic gameplay tối thiểu (đồng bộ di chuyển).
  - Đóng vai trò nguồn chân lý cho trạng thái hệ thống.
- **Data Layer (MongoDB, Redis):**
  - MongoDB lưu trữ dữ liệu bền vững (User, Character).

– Redis (tuỳ mức hiện thực) hỗ trợ session hoặc cache dữ liệu truy cập thường xuyên.

Nguyên tắc thiết kế cốt lõi của hệ thống là **server authoritative**: mọi trạng thái có ảnh hưởng đến tính nhất quán và công bằng đều do server xác nhận.

7.2.2 Phân tách theo lớp chức năng

Ở mức logic, hệ thống được phân chia thành các lớp chức năng:

- **Presentation Layer:** Game client Unity.
- **Application Layer:** Các module backend xử lý nghiệp vụ.
- **Real-time Layer:** Gateway WebSocket và logic phiên chơi.
- **Data Layer:** MongoDB và Redis.

Cách phân tách này giúp giảm phụ thuộc giữa các thành phần, đồng thời tạo điều kiện mở rộng hệ thống trong các giai đoạn sau (NPC, dungeon, PvP).

7.3 Thiết kế các mô-đun backend

Backend được tổ chức theo kiến trúc module hoá của NestJS. Trong giai đoạn 1, các mô-đun chính bao gồm:

7.3.1 Auth Module

Chức năng:

- Đăng ký tài khoản.
- Đăng nhập và cấp token phiên.
- Xác thực token cho các request tiếp theo.

Thiết kế:

- Sử dụng HTTP/REST cho đăng ký và đăng nhập.
- Token (ví dụ JWT) được gửi kèm trong header hoặc handshake WebSocket.
- Mật khẩu chỉ được lưu dưới dạng hash.

7.3.2 User và Character Module

User Module chịu trách nhiệm quản lý thông tin tài khoản người chơi và liên kết với danh sách nhân vật.

Character Module đảm nhiệm:

- Tạo nhân vật tối thiểu.
- Lưu trữ các thuộc tính nền như level, vị trí khởi tạo và chỉ số cơ bản.
- Cung cấp dữ liệu cho phiên chơi real-time.

Thiết kế hai module này hướng đến khả năng mở rộng cho các hệ thống class, skill và progression trong các giai đoạn tiếp theo.

7.3.3 Session và Room Module

Module Session quản lý trạng thái phiên chơi real-time, bao gồm ánh xạ giữa user, character và socket.

Module Room dùng để nhóm các người chơi vào các instance logic nhằm phục vụ kiểm thử đồng bộ trạng thái trong giai đoạn 1.

Các session chủ yếu tồn tại trong bộ nhớ runtime và không yêu cầu lưu bền vững.

7.3.4 Real-time Gateway

Gateway WebSocket đảm nhiệm:

- Nhận input di chuyển từ client.
- Phát snapshot trạng thái về client.
- Xử lý kết nối và ngắt kết nối.

Input từ client được xem là *ý định*, không phải trạng thái cuối cùng; server kiểm tra ràng buộc trước khi cập nhật trạng thái authoritative.

7.4 Thiết kế giao tiếp client–server

7.4.1 Giao tiếp HTTP

HTTP được sử dụng cho các chức năng không yêu cầu real-time:

- Đăng ký, đăng nhập.
- Tải hồ sơ người chơi.
- Tạo và chọn nhân vật.

Cách tiếp cận này giúp đơn giản hoá thiết kế và thuận tiện cho kiểm thử.

7.4.2 Giao tiếp WebSocket

WebSocket được sử dụng cho:

- Thiết lập phiên chơi.
- Gửi input di chuyển.
- Nhận cập nhật trạng thái.

Mỗi message WebSocket được thiết kế theo dạng message-based, gồm loại sự kiện và payload dữ liệu.

7.5 Thiết kế dữ liệu ở mức logic

7.5.1 Đối tượng User

Đối tượng User bao gồm các thuộc tính logic chính:

- userId
- username/email
- passwordHash
- metadata cơ bản

User đóng vai trò định danh và liên kết các dữ liệu khác.

7.5.2 Đối tượng Character

Đối tượng Character bao gồm:

- characterId
- ownerUserId
- level
- baseStats
- position (spawn)

Thiết kế này sẵn sàng mở rộng cho class, skill và equipment.

7.5.3 Đối tượng Session (Runtime)

Session tồn tại trong runtime server, bao gồm:

- socketId
- userId
- characterId
- roomId
- trạng thái kết nối

Session không bắt buộc lưu vào DB trong giai đoạn 1.

7.6 Thiết kế luồng xử lý real-time

Luồng xử lý di chuyển được thiết kế như sau:

1. Client gửi input di chuyển theo nhịp.
2. Server nhận input và đưa vào vòng lặp tick.
3. Server kiểm tra ràng buộc (tốc độ, spam).
4. Server cập nhật trạng thái authoritative.
5. Server gửi snapshot về client.

Ở giai đoạn 1, cơ chế snapshot được triển khai ở mức đơn giản, chưa yêu cầu prediction và reconciliation hoàn chỉnh.

7.7 Các quyết định thiết kế và giới hạn

Một số quyết định thiết kế quan trọng trong giai đoạn 1:

- Chỉ hiện thực đồng bộ di chuyển cơ bản.
- Một user duy trì tối đa một session active.
- Chưa tối ưu cho quy mô người chơi lớn.
- Ưu tiên tính rõ ràng và ổn định hơn tối ưu hiệu năng.

Các giới hạn này giúp đảm bảo đồ án phù hợp với nguồn lực và mục tiêu học thuật.

7.8 Tổng kết chương

Chương này đã trình bày thiết kế hệ thống Fortress of the Fallen ở mức kiến trúc và mô-đun cho giai đoạn 1, bao gồm kiến trúc tổng thể, thiết kế backend, giao tiếp client–server, dữ liệu logic và luồng xử lý real-time.

Các nội dung trên là cơ sở trực tiếp cho Chương 8 – Hiện thực hệ thống.

8 Hiện thực hệ thống

8.1 Mục tiêu hiện thực

Mục tiêu của chương này là mô tả quá trình hiện thực các thành phần của hệ thống Fortress of the Fallen trong giai đoạn 1, dựa trên thiết kế đã trình bày ở Chương 7. Nội dung tập trung vào cách tổ chức các thành phần client và server, cơ chế tích hợp giữa chúng, cũng như phạm vi hiện thực cụ thể của prototype.

Việc hiện thực trong giai đoạn 1 không nhằm tạo ra một sản phẩm game hoàn chỉnh, mà đóng vai trò kiểm chứng các quyết định kiến trúc, công nghệ và luồng vận hành của hệ thống.

8.2 Môi trường và công cụ triển khai

8.2.1 Môi trường phát triển

Hệ thống được hiện thực và kiểm thử trong môi trường phát triển cục bộ, bao gồm:

- Máy tính cá nhân chạy hệ điều hành Windows.
- Unity Editor cho phía client.
- Node.js runtime cho phía backend.
- MongoDB chạy dưới dạng local service hoặc container.

Hệ thống không yêu cầu triển khai trên môi trường production trong giai đoạn này.

8.2.2 Công cụ hỗ trợ

Các công cụ chính được sử dụng trong quá trình hiện thực gồm:

- Unity Editor: xây dựng client, scene và logic gameplay.
- NestJS CLI: tạo cấu trúc backend và các module dịch vụ.
- Git và GitHub: quản lý mã nguồn và phối hợp làm việc nhóm.
- Công cụ log và debug mặc định của Unity và Node.js.

8.3 Hiện thực phía client

8.3.1 Cấu trúc project Unity

Project Unity được tổ chức theo hướng tách biệt rõ các nhóm chức năng:

- **Scenes:** các scene chính như Login, CharacterSelect và TestMap.
- **Scripts:** logic điều khiển nhân vật, xử lý input, quản lý kết nối mạng.
- **UI:** các thành phần giao diện như form đăng nhập và HUD cơ bản.
- **Network:** lớp giao tiếp HTTP và WebSocket với backend.

Cách tổ chức này giúp giảm phụ thuộc giữa gameplay, UI và networking.

8.3.2 Hiện thực luồng đăng ký và đăng nhập

Luồng đăng ký và đăng nhập được hiện thực theo mô hình client–server chuẩn:

- Client hiển thị form đăng ký/đăng nhập và thu thập dữ liệu người dùng.
- Dữ liệu được gửi đến backend thông qua HTTP request.
- Client nhận phản hồi thành công hoặc lỗi và cập nhật giao diện tương ứng.

Token phiên nhận được sau khi đăng nhập được lưu tạm thời trên client để sử dụng cho các request tiếp theo và cho quá trình thiết lập kết nối real-time.

8.3.3 Hiện thực quản lý nhân vật

Sau khi đăng nhập thành công, client thực hiện:

- Gửi request tải hồ sơ người chơi và danh sách nhân vật.
- Hiển thị danh sách nhân vật hoặc giao diện tạo nhân vật mới.
- Lưu characterId được chọn để sử dụng trong phiên chơi.

Trong giai đoạn 1, dữ liệu nhân vật được sử dụng ở mức tối thiểu, chủ yếu phục vụ việc xác định danh tính và vị trí khởi tạo trong phiên chơi thử nghiệm.

8.3.4 Hiện thực điều khiển và hiển thị

Client hiện thực cơ chế điều khiển nhân vật cơ bản:

- Thu thập input di chuyển từ bàn phím.
- Gửi input dưới dạng ý định đến server thông qua WebSocket.
- Nhận snapshot trạng thái từ server và cập nhật vị trí nhân vật.

Ở giai đoạn 1, client ưu tiên hiển thị trực tiếp trạng thái do server gửi về, chưa triển khai cơ chế dự đoán hay hiệu chỉnh phức tạp.

8.4 Hiện thực phía server

8.4.1 Cấu trúc project backend

Backend được hiện thực bằng NestJS và tổ chức theo kiến trúc module hoá:

- **Auth Module:** xử lý đăng ký, đăng nhập và xác thực token.
- **User Module:** quản lý thông tin người dùng.
- **Character Module:** quản lý dữ liệu nhân vật.
- **Session/Room Module:** quản lý phiên chơi và room logic.
- **WebSocket Gateway:** xử lý giao tiếp real-time.

Cấu trúc này cho phép mở rộng thêm các module gameplay trong các giai đoạn tiếp theo.

8.4.2 Hiện thực xác thực và quản lý phiên

Backend hiện thực cơ chế xác thực theo các bước:

- Nhận request đăng nhập từ client.
- Xác thực thông tin và tạo token phiên.
- Kiểm tra token cho các request HTTP và kết nối WebSocket.

Phiên real-time được quản lý bằng cách ánh xạ giữa token, userId, characterId và socket tương ứng.

8.4.3 Hiện thực WebSocket và đồng bộ trạng thái

Gateway WebSocket chịu trách nhiệm:

- Thiết lập kết nối real-time sau khi client xác thực.
- Nhận input di chuyển từ client.
- Cập nhật trạng thái nhân vật trong vòng lặp tick đơn giản.
- Phát snapshot trạng thái định kỳ về client.

Trong giai đoạn 1, vòng lặp tick được hiện thực ở mức cơ bản nhằm kiểm chứng luồng dữ liệu và tính ổn định của kết nối.

8.4.4 Xử lý mất kết nối

Khi client mất kết nối WebSocket:

- Server nhận sự kiện disconnect.
- Giải phóng session runtime liên quan.
- Cập nhật trạng thái nhân vật về offline (ở mức logic).

Cơ chế reconnect chỉ được xem xét ở mức định hướng và chưa hiện thực đầy đủ trong giai đoạn này.

8.5 Hiện thực lưu trữ dữ liệu

8.5.1 Lưu trữ dữ liệu bền vững

MongoDB được sử dụng để lưu trữ:

- Thông tin tài khoản người dùng.
- Thông tin nhân vật và thuộc tính cơ bản.

Dữ liệu được thiết kế linh hoạt nhằm sẵn sàng mở rộng cho các hệ thống gameplay phức tạp hơn trong tương lai.

8.5.2 Dữ liệu runtime

Các dữ liệu runtime như session và trạng thái room được lưu trong bộ nhớ server. Redis có thể được tích hợp trong các giai đoạn sau nếu cần mở rộng hoặc chia tải.

8.6 Tích hợp client–server

Quá trình tích hợp được thực hiện theo các bước:

1. Client thực hiện đăng nhập và nhận token.
2. Client thiết lập kết nối WebSocket và xác thực.
3. Client tham gia phiên chơi thử nghiệm.
4. Client và server trao đổi input và snapshot trạng thái.

Việc tích hợp được kiểm thử với một số lượng nhỏ client đồng thời để đảm bảo luồng hoạt động ổn định.

8.7 Phạm vi hiện thực và giới hạn

Trong giai đoạn 1, hệ thống chỉ hiện thực các chức năng nền tảng:

- Đăng ký và đăng nhập.
- Quản lý nhân vật tối thiểu.
- Kết nối real-time và đồng bộ di chuyển.

Các hệ thống nâng cao như NPC, combat, gacha, dungeon và PvP chỉ được đề cập ở mức thiết kế và không thuộc phạm vi hiện thực của chương này.

8.8 Tổng kết chương

Chương 8 đã trình bày quá trình hiện thực hệ thống Fortress of the Fallen trong giai đoạn 1, bao gồm hiện thực client, backend, cơ chế lưu trữ dữ liệu và tích hợp client–server.

Các kết quả hiện thực này là cơ sở cho Chương 9, nơi hệ thống sẽ được đánh giá và kiểm thử dựa trên các tiêu chí đã xác định.

9 Đánh giá hệ thống

9.1 Mục tiêu đánh giá

Mục tiêu của chương này là đánh giá mức độ đáp ứng của hệ thống Fortress of the Fallen so với các mục tiêu và yêu cầu đã xác định ở Chương 5, đồng thời kiểm chứng tính đúng đắn của các quyết định thiết kế và hiện thực trong giai đoạn 1.

Việc đánh giá tập trung vào:

- Mức độ hoàn thành các yêu cầu chức năng cốt lõi.
- Khả năng vận hành ổn định của hệ thống prototype.
- Mức độ phù hợp giữa thiết kế và hiện thực.
- Các hạn chế còn tồn tại trong phạm vi đồ án.

Hệ thống không được đánh giá theo tiêu chí sản phẩm thương mại hay quy mô lớn, mà theo góc độ một prototype học thuật.

9.2 Phương pháp đánh giá

Hệ thống được đánh giá dựa trên các phương pháp sau:

- **Đối chiếu yêu cầu:** so sánh chức năng hiện thực với yêu cầu chức năng và phi chức năng đã nêu ở Chương 5.

- **Kiểm thử theo use case:** kiểm chứng các ca sử dụng đã phân tích ở Chương 6.
- **Quan sát vận hành:** theo dõi hành vi hệ thống khi chạy prototype với một số client đồng thời.
- **Phân tích định tính:** đánh giá mức độ rõ ràng, nhất quán và khả năng mở rộng.

Các phép đo định lượng chi tiết (benchmark hiệu năng, stress test) không được đặt trọng tâm trong giai đoạn này.

9.3 Đánh giá yêu cầu chức năng

9.3.1 Đăng ký và đăng nhập

Hệ thống đã hiện thực thành công các chức năng:

- Đăng ký tài khoản mới.
- Đăng nhập bằng thông tin hợp lệ.
- Từ chối đăng nhập với thông tin không hợp lệ.

Quá trình xác thực hoạt động đúng luồng phân tích, token được cấp và sử dụng cho các bước tiếp theo. Điều này đáp ứng đầy đủ các yêu cầu chức năng UC01 và UC02.

9.3.2 Quản lý nhân vật

Chức năng quản lý nhân vật ở mức tối thiểu được hiện thực gồm:

- Tải hồ sơ người chơi sau đăng nhập.
- Tạo nhân vật cơ bản khi chưa có dữ liệu.
- Chọn nhân vật để tham gia phiên chơi.

Các chức năng này đáp ứng các use case UC04, UC05 và UC06, đồng thời tạo nền tảng cho việc mở rộng hệ thống progression trong các giai đoạn tiếp theo.

9.3.3 Kết nối real-time và phiên chơi

Hệ thống đã thiết lập được kết nối WebSocket ổn định giữa client và server:

- Client xác thực kết nối real-time bằng token hợp lệ.
- Server quản lý phiên chơi và ánh xạ user–character–socket.
- Người chơi được gán vào phiên chơi thử nghiệm.

Các chức năng này đáp ứng UC07 và UC08 trong phạm vi giai đoạn 1.

9.3.4 Đồng bộ trạng thái di chuyển

Cơ chế đồng bộ di chuyển được kiểm chứng thông qua:

- Client gửi input di chuyển theo nhịp.
- Server xử lý input và cập nhật trạng thái authoritative.
- Client nhận snapshot và hiển thị vị trí nhân vật.

Hệ thống hoạt động ổn định với số lượng nhỏ client đồng thời, đáp ứng UC09 và UC10. Việc chưa triển khai prediction và reconciliation nâng cao được xem là phù hợp với phạm vi prototype.

9.3.5 Xử lý mất kết nối

Khi client mất kết nối:

- Server nhận sự kiện disconnect.
- Session runtime được giải phóng.
- Client hiển thị trạng thái mất kết nối.

Chức năng này đáp ứng UC11 ở mức cơ bản, đảm bảo hệ thống không rơi vào trạng thái không nhất quán khi có lỗi mạng.

9.4 Đánh giá yêu cầu phi chức năng

9.4.1 Hiệu năng

Trong môi trường thử nghiệm cục bộ:

- Độ trễ truyền dữ liệu ở mức chấp nhận được cho prototype.
- Hệ thống xử lý ổn định với 2–3 client đồng thời.

Hệ thống chưa được kiểm thử ở tải lớn, điều này phù hợp với phạm vi giai đoạn 1.

9.4.2 Tính ổn định

Trong quá trình kiểm thử:

- Backend không bị crash khi client ngắt kết nối đột ngột.
- Các session được giải phóng đúng cách.

Điều này cho thấy thiết kế quản lý phiên và xử lý lỗi cơ bản hoạt động đúng.

9.4.3 Bảo mật ở mức cơ bản

Các biện pháp bảo mật tối thiểu đã được áp dụng:

- Mật khẩu được mã hoá trước khi lưu trữ.
- Client không được phép tự quyết định trạng thái gameplay.
- Token được kiểm tra trước khi truy cập các chức năng quan trọng.

Các cơ chế bảo mật nâng cao chưa được triển khai trong giai đoạn này.

9.5 Đánh giá mức độ phù hợp giữa thiết kế và hiện thực

Kết quả hiện thực cho thấy:

- Các mô-đun backend được tổ chức đúng theo thiết kế ở Chương 7.
- Luồng client–server vận hành đúng theo phân tích use case.
- Không phát sinh sự lệch hướng lớn giữa thiết kế và triển khai.

Điều này chứng minh thiết kế hệ thống ban đầu có tính khả thi và phù hợp với điều kiện hiện thực của đồ án.

9.6 Hạn chế và tồn tại

Bên cạnh các kết quả đạt được, hệ thống vẫn còn một số hạn chế:

- Chưa hỗ trợ các hệ thống gameplay nâng cao (combat, NPC, gacha).
- Chưa tối ưu cho số lượng người chơi lớn.
- Chưa triển khai đầy đủ các kỹ thuật networking nâng cao.

Các hạn chế này xuất phát từ việc chủ động giới hạn phạm vi để đảm bảo tính khả thi và chất lượng học thuật của đồ án.

9.7 Tổng kết chương

Chương 9 đã đánh giá hệ thống Fortress of the Fallen trong giai đoạn 1 dựa trên các yêu cầu và use case đã phân tích. Kết quả cho thấy hệ thống prototype đáp ứng được các mục tiêu nền tảng về kiến trúc, kết nối real-time và quản lý phiên chơi.

Những kết quả và hạn chế được ghi nhận là cơ sở cho phần kết luận và định hướng phát triển trong Chương 10.

Tài liệu

[1] Newzoo. *Global Games Market Report*. Online report. Báo cáo thống kê thị trường game toàn cầu. 2023.

[2] Michael Moriarty. "Networked Physics and Latency Compensation in Online Games". **in***Proceedings of the Game Developers Conference*: Trình bày các kỹ thuật client-side prediction, interpolation, reconciliation. 2014.

[3] Jason Gregory. *Game Engine Architecture*. 3 **edition**. Kiến trúc game engine, kiến trúc nhiều lớp và real-time system. CRC Press, 2018.

[4] Ernest Adams. *Fundamentals of Game Design*. 3 **edition**. Tài liệu cơ bản về thiết kế game, core loop, progression, economy. New Riders, 2014.

[5] Jesse Schell. *The Art of Game Design: A Book of Lenses*. 3 **edition**. Các nguyên lý thiết kế trải nghiệm, level design, UX trong game. CRC Press, 2019.

[6] Unity Technologies. *Unity User Manual*. `https://docs.unity3d.com/Manual/index.html`. Tài liệu chính thức hướng dẫn sử dụng Unity. 2023.

[7] Kamil Myśliwiec **and** NestJS Team. *NestJS Documentation*. `https://docs.nestjs.com`. Tài liệu chính thức của NestJS. 2023.

[8] Kristina Chodorow. *MongoDB: The Definitive Guide*. 2 **edition**. Giới thiệu và hướng dẫn sử dụng MongoDB. O'Reilly Media, 2013.

[9] Josiah L. Carlson. *Redis in Action*. Ứng dụng Redis cho cache, pub/sub, lưu trữ session. Manning Publications, 2013.

[10] GitHub. *GitHub Documentation*. `https://docs.github.com`. Hướng dẫn sử dụng GitHub và Git cơ bản. 2023.

[11] GitHub. *GitHub Projects Documentation*. `https://docs.github.com/issues/planning-and-tracking-with-projects`. Hướng dẫn sử dụng GitHub Projects để quản lý tác vụ. 2023.

[12] Microsoft. *C# Programming Guide*. `https://learn.microsoft.com/dotnet/csharp/`. Tài liệu chính thức về ngôn ngữ C#. 2023.

[13] OpenJS Foundation. *Node.js Documentation*. `https://nodejs.org/en/docs`. Tài liệu chính thức của Node.js. 2023.

[14] Thomas Allweyer. *BPMN 2.0: Introduction to the Standard for Business Process Modeling*. Giới thiệu BPMN 2.0 dùng cho mô hình hoá quy trình. Books on Demand, 2016.

[15] Frank Mittelbach **andothers**. *The LaTeX Companion*. 2 **edition**. Tài liệu tham khảo về soạn thảo tài liệu bằng LaTeX. Addison-Wesley, 2004.

[16] Glenn Fiedler. *Gaffer on Games: Networking for Game Programmers*. Tài liệu tham khảo về server authoritative, prediction, interpolation. 2010.

[17] MongoDB. *MongoDB Manual (Documentation)*. Tài liệu tham khảo cơ sở dữ liệu MongoDB. 2025.

[18] Redis. *Redis Documentation*. Tài liệu tham khảo cache/pubsub Redis. 2025.

[19] Reki Kawahara. *Sword Art Online, Vol. 1: Aincrad*. Nguồn tham khảo ý tưởng leo tầng và boss tầng. Yen Press, 2014.

[20] Kugane Maruyama. *Overlord, Vol. 1: The Undead King*. Nguồn tham khảo cách xây dựng NPC và thế giới vận hành. Yen Press, 2016.

[21] Pick Me Up! Infinite Gacha. *Pick Me Up! Infinite Gacha (Webtoon/Manhwa)*. Nguồn tham khảo cơ chế gacha và meta-progression. 2022.

[22] Arcane Odyssey. *Arcane Odyssey (Game)*. Nguồn tham khảo trải nghiệm online RPG và cảm giác điều khiển. 2023.

[23] ChillyRoom. *Soul Knight Prequel (Game)*. Nguồn tham khảo vòng lặp phiên chơi ngắn và reward theo phiên. 2023.

[24] Supercell. *Clash of Clans (Game)*. Nguồn tham khảo hệ thống căn cứ/công trình và quản trị tài nguyên. 2012.