

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 <i>Fortress of the Fallen</i> , 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 đền 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.

(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.
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 trong game.
Nhóm thuật ngữ về multiplayer và networking		
Multiplayer	Multiplayer	Hình thức chơi nhiều người trong cùng một thế giới game.
Real-time	Real-time	Hệ thống xử lý và phản hồi gần như ngay lập tức theo input người chơi.
Latency	Network Latency	Độ trễ mạng giữa client và server, thường đo bằng mili-giây (ms).
Tick rate	Server Tick Rate	Số lần server cập nhật trạng thái game trong một giây.
Snapshot	State Snapshot	Bản chụp trạng thái game tại một thời điểm để đồng bộ client.
Client-side Prediction	Client-side Prediction	Client dự đoán kết quả hành động trước khi server phản hồi.
Reconciliation	Reconciliation	Client điều chỉnh trạng thái dự đoán cho khớp với server.
Server Authoritative	Server Authoritative Model	Server quyết định cuối cùng mọi logic game.
WebSocket	WebSocket	Giao thức truyền thông hai chiều trên TCP cho thời gian thực.
Instance	Instance	Phiên bản riêng của một khu vực game cho nhóm người chơi.
Zone / Shard	Zone / Shard	Phân vùng logic thế giới game để chia tải.
Interest Management	Interest Management	Chỉ gửi dữ liệu cần thiết cho mỗi client để giảm tải hệ thống.

Mục lục

Danh mục thuật ngữ	1
1 Giới thiệu	10
1.1 Động lực	10
1.2 Mục tiêu	10
1.3 Phạm vi	11
1.3.1 Phạm vi trong giai đoạn 1	11
1.3.2 Ngoài phạm vi giai đoạn 1	12
1.4 Ý nghĩa của đề tài	12
1.4.1 Ý nghĩa thực tiễn	12
1.4.2 Ý nghĩa khoa học và học thuật	12
1.5 Cấu trúc báo cáo	12
2 Kiến thức nền tảng	13
2.1 Kiến thức nền tảng về Action RPG và môi trường real-time	13
2.1.1 Đặc điểm của Action RPG	13
2.1.2 Hệ thống multiplayer trong Action RPG	14
2.2 Kiến thức nền tảng về Game Design	15
2.2.1 Core Loop và Meta Loop	15
2.2.2 Progression System (Hệ thống phát triển nhân vật)	16
2.2.3 Combat Design (Thiết kế chiến đấu)	20
2.2.4 World System và Level Design	20
2.2.5 Hệ thống kinh tế (Game Economy)	21
2.2.6 UX/UI Design trong game và định hướng phong cách Pixel Art	21
2.2.7 Balancing Fundamentals (Cân bằng game)	21
2.3 Tổng kết chương	22
3 Công nghệ sử dụng	22
3.1 Công nghệ phía client	23
3.1.1 Game engine: Unity	23
3.1.2 Ngôn ngữ lập trình: C#	23
3.1.3 Nguyên tắc xử lý logic ở client	23
3.2 Công nghệ phía server	24
3.2.1 Nền tảng runtime: Node.js	24
3.2.2 Framework: NestJS và TypeScript	24
3.2.3 Tác vụ nền (background jobs)	24
3.3 Công nghệ giao tiếp mạng	24
3.3.1 HTTP/REST	24
3.3.2 WebSocket	25
3.4 Công nghệ lưu trữ dữ liệu	25
3.4.1 MongoDB	25
3.4.2 Redis	25
3.4.3 Object Storage (tùy chọn theo giai đoạn)	25
3.5 Công cụ quản trị game configuration theo hướng data-driven	25
3.6 Công cụ hỗ trợ phát triển và tài liệu hoá	26
3.6.1 Quản lý mã nguồn và phối hợp nhóm	26
3.6.2 Công cụ mô hình hoá và tài liệu	26
3.7 Tổng kết	26

4 Các công trình liên quan	27
4.1 Khung phân tích và tiêu chí lựa chọn nguồn tham khảo	27
4.2 Nguồn cảm hứng cho trực tiến trình: leo tháp, boss tầng và checkpoint	27
4.2.1 Sword Art Online (Aincrad): cấu trúc leo tầng như “xương sống” nội dung	27
4.3 Nguồn cảm hứng cho “thế giới vận hành”: NPC có vai trò và hệ thống nhân sự .	28
4.3.1 Overlord: NPC như tác nhân hệ thống và tổ chức vận hành	28
4.4 Nguồn cảm hứng cho cơ chế tuyển dụng/thu thập: gacha và meta-progression .	28
4.4.1 Pick Me Up! Infinite Gacha: gacha như vòng lặp meta và động lực sưu tầm	28
4.5 Nguồn cảm hứng cho trải nghiệm Action RPG online: nhịp combat và cảm giác điều khiển	29
4.5.1 Arcane Odyssey: ưu tiên phản hồi thao tác và nhấn mạnh kỹ năng người chơi	29
4.5.2 Soul Knight Prequel: vòng lặp phiên chơi ngắn và thưởng rõ ràng theo phiên	29
4.6 Nguồn cảm hứng cho hệ thống đảo/căn cứ: tiến trình dài hạn và quản trị tài nguyên	30
4.6.1 Clash of Clans: công trình theo thời gian và meta-progression bền vững .	30
4.7 Bảng tổng hợp: đối chiếu nguồn tham khảo và hướng áp dụng	30
4.8 Nguồn tham khảo nền tảng và kỹ thuật: cơ sở cho lập luận thiết kế và ràng buộc triển khai	31
4.9 Tổng kết chương	32
5 Phân tích yêu cầu	32
5.1 Phạm vi và giả định	32
5.1.1 Phạm vi giai đoạn 1	32
5.1.2 Giả định	32
5.2 Tác nhân và ranh giới hệ thống	33
5.2.1 Tác nhân	33
5.2.2 Ranh giới hệ thống	33
5.3 Yêu cầu chức năng	33
5.3.1 Nhóm A: Authentication & Profile	33
5.3.2 Nhóm B: Character Creation & Character Data	34
5.3.3 Nhóm C: Stats, Leveling, Skill Slots	34
5.3.4 Nhóm D: Class System và mở khoá theo điều kiện	35
5.3.5 Nhóm E: Inventory, Equipment, Resources	36
5.3.6 Nhóm F: Combat & Gameplay Modes	36
5.3.7 Nhóm G: Recruitment (Gacha)	37
5.3.8 Nhóm H: Personal Island & NPC Collection	37
5.3.9 Nhóm I: Administration	38
5.4 Yêu cầu phi chức năng	38
5.4.1 Bảo mật và an toàn dữ liệu	38
5.4.2 Hiệu năng và tính phản hồi	38
5.4.3 Tính mở rộng và bảo trì	38
5.4.4 Khả dụng và quan sát hệ thống	39
5.4.5 Yêu cầu UX/UI ở mức hệ thống	39
5.5 Yêu cầu dữ liệu	39
5.5.1 Phân loại dữ liệu	39
5.5.2 Yêu cầu mô hình dữ liệu mức logic	39
5.5.3 Yêu cầu pipeline quản trị game configuration	40

5.6	Tiêu chí chấp nhận ở mức thiết kế	40
6	Phân tích hệ thống	40
6.1	Tổng quan phân hệ và ranh giới trách nhiệm	40
6.2	Tác nhân của hệ thống (Actors)	41
6.3	Phân tích Use Case theo phân hệ	42
6.3.1	Identity & Profile	42
6.3.2	Character & Progression	43
6.3.3	Combat & Gameplay Modes	44
6.3.4	Recruitment (Gacha)	45
6.3.5	Personal Island & NPC	46
6.3.6	Administration	46
6.4	Đặc tả Use Case trọng yếu	47
6.4.1	UC-A01 – Đăng ký tài khoản	47
6.4.2	UC-A02 – Đăng nhập và tạo phiên hoạt động	47
6.4.3	UC-A03 – Tạo và chọn Game Profile	48
6.4.4	UC-P01 – Tạo nhân vật và khởi tạo dữ liệu nền	48
6.4.5	UC-P02 – Lên level và phân bổ điểm chỉ số	49
6.4.6	UC-P03 – Quản lý kỹ năng theo slot và tạo Combo Skill	49
6.4.7	UC-P04 – Mở khoá Class theo điều kiện và chuyển Class	50
6.4.8	UC-I01 – Nhận reward và cập nhật Inventory	51
6.4.9	UC-C01 – Tham gia nội dung theo phiên (Tower/Dungeon/Arena)	51
6.4.10	UC-C02 – Vòng lặp chiến đấu real-time và đồng bộ trạng thái	52
6.4.11	UC-C03 – Kết phiên, checkpoint và ghi nhận tiến trình	52
6.4.12	UC-G01 – Gacha: Summon, pity và xử lý trùng lặp	53
6.4.13	UC-S01 – Xây/Nâng cấp công trình trên đảo cá nhân	53
6.4.14	UC-S02 – Gán NPC làm worker cho công trình	54
6.4.15	UC-AD01 – Quản trị user: ban/unban và audit log	54
6.5	Quy tắc nghiệp vụ và ràng buộc hệ thống	54
6.5.1	Quy tắc Level và mở slot kỹ năng	54
6.5.2	Quy tắc Race, Stat Cap và Trait bẩm sinh	55
6.5.3	Quy tắc mở khoá Class	55
6.5.4	Quy tắc Inventory và tham chiếu cấu hình	55
6.5.5	Quy tắc Island và xây dựng công trình	56
6.5.6	Quy tắc Gacha: rate, pity và duplicate	56
6.6	Phân tích dữ liệu ở mức logic	56
6.6.1	Phân lớp dữ liệu: động và tĩnh	56
6.6.2	Các thực thể lỗi theo phân hệ	56
6.6.3	Quan hệ dữ liệu quan trọng	56
6.7	Tổng kết chương	57
7	Thiết kế hệ thống	57
7.1	Mục tiêu thiết kế	57
7.2	Kiến trúc tổng thể và triển khai	57
7.2.1	Tổng quan triển khai	57
7.2.2	Nguyên tắc server-authoritative	59
7.2.3	Phân tách giao tiếp: HTTP và WebSocket	59
7.3	Thiết kế mô-đun backend	59
7.3.1	Các thành phần dùng chung (cross-cutting)	60
7.4	Thiết kế dữ liệu	60

7.4.1	Nguyên tắc thiết kế dữ liệu	60
7.4.2	Mô hình thực thể chính (ERD mức logic)	61
7.4.3	Chỉ mục (indexes) và khoá truy vấn	62
7.5	Thiết kế hệ thống chỉ số và tăng trưởng nhân vật	62
7.5.1	Thuộc tính nền và chỉ số suy diễn	62
7.5.2	Tính toán chỉ số theo hướng data-driven	63
7.6	Thiết kế quản trị cấu hình game (Config Pipeline)	63
7.6.1	Luồng dữ liệu cấu hình	63
7.6.2	Thiết kế Config Manager	64
7.7	Thiết kế giao tiếp client-server	64
7.7.1	REST API: nhóm endpoint và quy ước	64
7.7.2	WebSocket: định dạng message và vòng đời phiên	65
7.8	Thiết kế vòng lặp real-time cho combat/instance	65
7.8.1	Mô hình instance/room	65
7.8.2	Tick loop và phát snapshot	65
7.9	Thiết kế phân hệ Administration	66
7.10	Bảo mật, nhất quán và quan sát hệ thống	66
7.10.1	Bảo mật	66
7.10.2	Nhất quán dữ liệu và giao dịch nghiệp vụ	67
7.10.3	Quan sát hệ thống (observability)	67
7.11	Tổng kết chương	67
8	Kế hoạch hiện thực prototype	67
8.1	Mục tiêu và phạm vi	67
8.2	Kế hoạch triển khai theo mốc	68
8.3	Hiện thực phía client (Unity)	68
8.3.1	Tổ chức project và nguyên tắc phân lớp	68
8.3.2	Luồng màn hình đề xuất	69
8.3.3	Networking client: REST + WebSocket	69
8.4	Hiện thực phía server (NestJS)	69
8.4.1	Cấu trúc module và luồng request	69
8.4.2	Realtime: room runtime và tick loop tối thiểu	70
8.5	Thiết kế hiện thực dữ liệu và truy xuất	71
8.5.1	Schema collections và nguyên tắc embed/reference	71
8.5.2	Nguyên tắc cập nhật an toàn cho giao dịch tiến trình	71
8.6	Hiện thực pipeline game configuration	71
8.6.1	Định dạng TSV và quy ước	71
8.6.2	Import script: upsert và log phiên bản	72
8.6.3	Config Manager runtime	72
8.7	Kế hoạch kiểm thử và tiêu chí nghiệm thu	72
8.7.1	Kiểm thử theo use case	72
8.7.2	Kiểm thử dữ liệu và tính nhất quán	72
8.7.3	Tiêu chí nghiệm thu prototype	73
8.8	Tổng kết chương	73
9	Đánh giá hệ thống	73
9.1	Mục tiêu và phương pháp đánh giá	73
9.2	Đối chiếu bao phủ yêu cầu chức năng	74
9.2.1	Ma trận bao phủ yêu cầu theo mô-đun	74
9.2.2	Đánh giá luồng nghiệp vụ trọng yếu	74

9.3	Đánh giá yêu cầu phi chức năng	75
9.3.1	Bảo mật và phân quyền	75
9.3.2	Nhất quán dữ liệu và khả năng chống cấp trùng	76
9.3.3	Hiệu năng và khả năng mở rộng	76
9.3.4	Tính bảo trì và khả năng mở rộng tính năng	76
9.3.5	Khả năng quan sát (Observability)	77
9.4	Đánh giá pipeline cấu hình game (data-driven config)	77
9.4.1	Ưu điểm	77
9.4.2	Rủi ro và kiểm soát	77
9.5	Đánh giá UI/UX theo định hướng pixel art ở mức hệ thống	77
9.6	Phân tích rủi ro và hướng giảm thiểu	78
9.7	Tổng kết chương	78
10	Kết luận và hướng phát triển	79
10.1	Tổng kết kết quả đạt được	79
10.2	Đóng góp của đồ án	80
10.3	Hạn chế	80
10.4	Hướng phát triển	80
10.4.1	Hiện thực prototype và kiểm chứng theo use case	81
10.4.2	Nâng cấp realtime cho cảm giác chiến đấu	81
10.4.3	Hoàn thiện hệ thống giao dịch và chống gian lận	81
10.4.4	Mở rộng pipeline cấu hình	81
10.4.5	Chuẩn hoá quan sát hệ thống	81
10.5	Kết luận	82
Tài liệu tham khảo		83

Danh sách hình vẽ

1	Ví dụ mô hình hoá Core Loop trong Action RPG	15
2	Mô hình hoá thuộc tính nền, chỉ số suy diễn và nhóm trait đặc biệt	17
3	Use case tổng quan cho phân hệ Identity & Profile	42
4	Use case tổng quan cho phân hệ Character & Progression	43
5	Use case tổng quan cho phân hệ Combat & Gameplay Modes	44
6	Use case tổng quan cho phân hệ Recruitment (Gacha)	45
7	Use case tổng quan cho phân hệ Personal Island & NPC	46
8	Use case tổng quan cho phân hệ Administration	46
9	Sơ đồ triển khai tổng quan hệ thống	58
10	Mô hình thuộc tính nền và chỉ số suy diễn	62

Danh sách bảng

1	Vai trò của 6 thuộc tính nền trong thiết kế build	17
2	Ví dụ quan hệ phụ thuộc Secondary Stats từ Primary Attributes	18
3	Ví dụ mốc level và phần thưởng mở slot kỹ năng	19
4	Đổi chiểu nguồn tham khảo và hướng áp dụng vào đề tài	31
5	Danh sách tác nhân và vai trò	41
6	Mốc level và cơ chế mở slot kỹ năng	55
7	Bản đồ mô-đun backend và dữ liệu liên quan	60
8	Kế hoạch hiện thực prototype theo mốc	68
9	Ma trận bao phủ yêu cầu chức năng theo mô-đun	74
10	Bảng rủi ro và hướng giảm thiểu	78

1 Giới thiệu

1.1 Động lực

Trong những năm gần đây, thị trường game trực tuyến tiếp tục tăng trưởng và xu hướng người chơi chuyển dịch mạnh sang các sản phẩm có tính tương tác và cập nhật liên tục[1]. Đặc biệt, nhóm trò chơi nhập vai hành động trực tuyến (Online Action RPG) đặt ra yêu cầu cao cả về trải nghiệm thời gian thực (real-time) lẫn tiến trình dài hạn (meta-progression). Người chơi kỳ vọng hệ thống vận hành ổn định, đồng bộ trạng thái mượt mà, phản hồi nhanh và đảm bảo tính nhất quán khi tương tác với nhiều phân hệ gameplay trong cùng một tài khoản.

Để đáp ứng các kỳ vọng đó, hệ thống phía server không chỉ xử lý xác thực và lưu trữ dữ liệu, mà còn phải đảm nhiệm các bài toán đặc thù của game online như:

- Quản lý phiên chơi và xử lý nhiều kết nối đồng thời.
- Đồng bộ trạng thái theo mô hình phù hợp để giảm sai lệch giữa client và server, đồng thời hạn chế gian lận ở mức kiên trúc[2].
- Thiết kế dữ liệu tiến trình người chơi (profile/character/inventory/island/npc) sao cho có thể mở rộng và truy vết.
- Tổ chức nhiều phân hệ nghiệp vụ (combat, progression, gacha, base-building, admin) trong một kiến trúc nhất quán, giảm phụ thuộc chéo và thuận lợi bảo trì[3].

Ngoài ra, một thách thức quan trọng trong thiết kế game là quản trị **game configuration** (dữ liệu tĩnh) như định nghĩa vật phẩm, công trình, tham số cân bằng, bảng tỉ lệ và cấu hình sự kiện. Nếu cấu hình bị gắn chặt vào mã nguồn, việc điều chỉnh nội dung và cân bằng sẽ phụ thuộc vào chu kỳ build và phát hành. Do đó, hướng tiếp cận *data-driven* thường được áp dụng để tách cấu hình khỏi code, giúp quy trình cập nhật linh hoạt hơn và giảm rủi ro “hard-code” tham số[4, 5].

Trong bối cảnh đó, đề tài *Fortress of the Fallen* được thực hiện với trọng tâm là phân tích và thiết kế hệ thống cho một game Online Action RPG. Báo cáo hướng đến mô tả đầy đủ các phân hệ cốt lõi ở mức độ án chuyên ngành, làm nền tảng cho các giai đoạn hiện thực tiếp theo.

1.2 Mục tiêu

Mục tiêu tổng quát của đề tài là xây dựng **bản thiết kế hệ thống** cho trò chơi Online Action RPG *Fortress of the Fallen*, bao gồm kiến trúc, mô hình dữ liệu, luồng nghiệp vụ và hợp đồng giao tiếp giữa các thành phần.

Trong **giai đoạn 1**, đề tài tập trung vào các mục tiêu cụ thể:

- Xác định phạm vi và mô tả tập tính năng ở mức hệ thống theo các phân hệ: Authentication & Profile, Combat & Gameplay, Character Progression, Recruitment (Gacha), Personal Island, và Administration.

- Phân tích yêu cầu chức năng, phi chức năng và yêu cầu dữ liệu; làm rõ ranh giới giữa dữ liệu tiến trình người chơi và dữ liệu cấu hình tĩnh.
- Thiết kế kiến trúc triển khai tổng thể theo mô hình client-server, bao gồm các thành phần backend và hạ tầng dữ liệu; xác định cơ chế giao tiếp phù hợp cho tác vụ request-response và real-time[2, 3].
- Thiết kế mô hình dữ liệu mức logic cho các thực thể chính: user, profile, session, character, inventory, island, npc và quan hệ giữa chúng, phục vụ truy vấn và mở rộng tính năng về sau.
- Thiết kế pipeline quản trị **game configuration (tĩnh)** theo hướng data-driven:

Google Sheets → TSV → Import vào DB (collections cấu hình) → Load vào *Config Manager* khi khởi động.

Pipeline này **chỉ áp dụng cho game configuration**

Ví dụ: CONFIG_ITEM, CONFIG_SKILL, CONFIG_ENEMY và các bảng cấu hình mở rộng, không áp dụng cho dữ liệu tiến trình người chơi.

1.3 Phạm vi

1.3.1 Phạm vi trong giai đoạn 1

Giai đoạn 1 tập trung hoàn toàn vào **phân tích và thiết kế**, bao gồm:

- **Thiết kế phân hệ và luồng nghiệp vụ:** mô tả actors, use case, luồng tổng quát cho các tác vụ chính (đăng nhập/chọn profile/chọn nhân vật; tham gia tower/dungeon/arena; loot và cập nhật inventory; nâng cấp/equip; gacha và pity; island xây dựng và worker; thao tác admin).
- **Thiết kế kiến trúc triển khai:** phân tách vai trò client và server; tổ chức backend thành các thành phần (API server, WebSocket gateway, worker) và tích hợp hạ tầng dữ liệu (DB, cache, object storage) ở mức kiến trúc.
- **Thiết kế kiến trúc phần mềm theo module:** phân rã module theo phân hệ, xác định dịch vụ dùng chung (validation, resource, RNG, inventory rules, sync/audit) nhằm giảm trùng lặp và đảm bảo nhất quán nghiệp vụ.
- **Thiết kế dữ liệu mức logic:** xây dựng mô hình dữ liệu cho tiến trình người chơi (dynamic) và mô hình cho cấu hình game (static), làm rõ điểm tham chiếu giữa item instance và item definition, giữa building instance và building definition.
- **Thiết kế hợp đồng giao tiếp:** định hướng nhóm REST API cho các thao tác không real-time và nhóm WebSocket events cho các hành vi trong phiên chơi.

- **Thiết kế pipeline cấu hình tinh:** mô tả quy trình export/import TSV và cơ chế nạp cấu hình vào Config Manager khi khởi động để phục vụ truy xuất thống nhất.

1.3.2 Ngoài phạm vi giai đoạn 1

Các nội dung sau **không thuộc phạm vi giai đoạn 1** do không phù hợp mục tiêu thiết kế ở mức đồ án:

- Hiện thực gameplay hoàn chỉnh (combat runtime đầy đủ, tối ưu điều khiển, animation/FX pipeline).
- Hoàn thiện PvP ở mức sản phẩm (matchmaking nâng cao, chống gian lận toàn diện, xếp hạng theo mùa với hệ thống phần thưởng vận hành dài hạn).
- Live-ops và monetization thương mại; cân bằng kinh tế game ở quy mô lớn.
- Benchmark tải lớn, tối ưu hiệu năng production, và triển khai đa vùng (multi-region) ở mức vận hành thực tế.

1.4 Ý nghĩa của đề tài

1.4.1 Ý nghĩa thực tiễn

Đề tài cung cấp một bản thiết kế hệ thống có tính mô-đun cho game Online Action RPG, làm rõ cách tổ chức các phân hệ gameplay và cách quản trị dữ liệu nhất quán trong một sản phẩm. Đặc biệt, cơ chế quản trị *game configuration* theo hướng data-driven tạo nền tảng cho việc mở rộng nội dung và cân bằng ở các giai đoạn sau mà không phụ thuộc hoàn toàn vào chỉnh sửa mã nguồn[4, 3].

1.4.2 Ý nghĩa khoa học và học thuật

Đề tài tạo môi trường để vận dụng tổng hợp kiến thức về kiến trúc phần mềm, hệ thống phân tán, cơ sở dữ liệu, và nguyên lý đồng bộ trạng thái trong game online. Việc mô hình hóa nghiệp vụ và dữ liệu ở mức logic giúp tăng tính hệ thống, đồng thời cung cấp cơ sở để đánh giá tính khả thi và rủi ro khi chuyển sang giai đoạn triển khai[2, 3].

1.5 Cấu trúc báo cáo

Báo cáo được tổ chức như sau:

- **Chương 2 – Kiến thức nền tảng:** Trình bày các khái niệm nền tảng về Action RPG online, core loop/meta loop, nguyên tắc client-server và hướng tiếp cận data-driven cho cấu hình[4, 5].

- **Chương 3 – Công nghệ sử dụng:** Giới thiệu các công nghệ dự kiến sử dụng và lý do lựa chọn trong bối cảnh kiến trúc hệ thống (engine, backend, networking, database, công cụ quản lý)[6, 7, 8, 9].
- **Chương 4 – Các công trình liên quan:** Tổng hợp tham khảo theo nhóm phân hệ và bảng đối chiếu để rút ra các điểm áp dụng vào thiết kế.
- **Chương 5 – Phân tích yêu cầu:** Xác định yêu cầu chức năng, phi chức năng, dữ liệu và phạm vi giai đoạn 1.
- **Chương 6 – Phân tích hệ thống:** Phân tích actors, use case, luồng nghiệp vụ và dữ liệu mức logic phục vụ thiết kế.
- **Chương 7 – Thiết kế hệ thống:** Trình bày kiến trúc triển khai, kiến trúc module, mô hình dữ liệu và hợp đồng giao tiếp; mô tả pipeline quản trị game configuration.
- **Chương 8 – Định hướng hiện thực:** Đề xuất lộ trình triển khai theo module, chiến lược tích hợp pipeline cấu hình và các điểm cần ưu tiên giảm rủi ro.
- **Chương 9 – Đánh giá hệ thống:** Đánh giá mức độ bao phủ yêu cầu, tính nhất quán của thiết kế và các rủi ro còn tồn tại.
- **Chương 10 – Kết luận:** Tổng kết kết quả thiết kế đạt được và định hướng phát triển cho các giai đoạn tiếp theo.

2 Kiến thức nền tảng

Chương này hệ thống hoá các kiến thức cơ sở phục vụ cho việc thiết kế một trò chơi Action RPG trong môi trường trực tuyến thời gian thực. Nội dung tập trung vào hai nhóm chính: (i) đặc trưng của Action RPG và các nguyên lý đồng bộ multiplayer; (ii) các khái niệm nền tảng của Game Design gồm vòng lặp chơi (loop), hệ thống tiến trình (progression), thiết kế chiến đấu, thiết kế thế giới, kinh tế trong game, UX/UI và cân bằng thông số[4, 5]. Các khái niệm ở chương này đóng vai trò “khung lý thuyết” để liên kết xuyên suốt sang phần phân tích yêu cầu (Chương 5) và phần thiết kế hệ thống (Chương 7).

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) kết hợp cơ chế nhập vai (tăng trưởng sức mạnh, xây dựng nhân vật) với chiến đấu thời gian thực (real-time combat). Các đặc điểm cốt lõi thường gặp[4, 5]:

- **Điều khiển trực tiếp và phản hồi tức thời:** hành vi chiến đấu (tấn công, né tránh, di chuyển, dùng kỹ năng) cần phản hồi nhanh để tạo cảm giác “đã tay”.

- **Chiến đấu gắn với animation:** logic đòn đánh phụ thuộc vào các mốc thời gian của animation (khi nào hitbox bật/tắt, khi nào có i-frame, thời gian hồi chiêu/khôi phục).
- **Kết hợp kỹ năng người chơi và chỉ số nhân vật:** thao tác (timing, vị trí, né đòn) quyết định hiệu quả ngắn hạn; hệ thống chỉ số, trang bị và kỹ năng quyết định “tiềm năng” dài hạn.
- **Tính công bằng trong va chạm và sát thương:** cần quy ước rõ về hitbox/hurtbox, phạm vi hiệu lực và điều kiện trúng đòn nhằm giảm cảm giác “bị oan”.

Một hành động tấn công điển hình thường được chia thành ba pha[4]:

1. **Startup:** thời gian chuẩn bị, người chơi có thể bị ngắt hoặc bị trừ phạt nếu ra đòn sai thời điểm.
2. **Active:** thời gian hitbox có hiệu lực, đòn có thể gây sát thương/hiệu ứng.
3. **Recovery:** thời gian hồi lại, tạo “cửa sổ” để đối thủ phản công.

Việc xác định hợp lý độ dài ba pha giúp kiểm soát nhịp độ combat (pacing), tạo khác biệt giữa vũ khí/kỹ năng, và là cơ sở để cân bằng.

2.1.2 Hệ thống multiplayer trong Action RPG

Với Action RPG online, bài toán kỹ thuật trung tâm là đồng bộ trạng thái trong điều kiện mạng không ổn định: độ trễ (latency), jitter, mất gói, chênh lệch tốc độ khung hình giữa các máy. Thực tế triển khai thường dựa trên mô hình **server authoritative**[2, 10].

Mô hình client-server và tính thẩm quyền (authoritative) Trong mô hình phổ biến:

- **Client** thu thập input, dự đoán (prediction) ở mức hiển thị và render hình ảnh/âm thanh.
- **Server** giữ trạng thái “chuẩn”, xử lý các luật gameplay quan trọng (xác nhận vị trí, tính sát thương, trạng thái kỹ năng, kiểm tra điều kiện) và chống gian lận ở mức hệ thống[2].

Ưu điểm của server authoritative là giảm sai lệch giữa người chơi và hạn chế cheat; đánh đổi là cần cơ chế che giấu độ trễ để giữ cảm giác điều khiển mượt.

Tick-rate và vòng lặp mô phỏng Server chạy theo chu kỳ mô phỏng (tick). Ở mỗi tick, server có thể thực hiện: nhận input, cập nhật trạng thái, xử lý va chạm/chiến đấu, sau đó gửi snapshot cho client[2]. Tick-rate cao cho phản hồi tốt hơn nhưng tăng chi phí CPU/băng thông; tick-rate thấp tiết kiệm hơn nhưng đòi hỏi client xử lý nội suy tốt để tránh giật.

Các kỹ thuật che giấu độ trễ Để gameplay cảm giác “real-time” dù có latency, các kỹ thuật nền tảng thường phối hợp[2, 10]:

- **Snapshot interpolation:** client nhận snapshot định kỳ và nội suy để hiển thị chuyển động mượt.
- **Client-side prediction:** client tạm dự đoán kết quả dựa trên input thay vì chờ phản hồi server.
- **Reconciliation:** khi server gửi trạng thái chuẩn, client điều chỉnh sai lệch một cách “êm” để tránh giật.

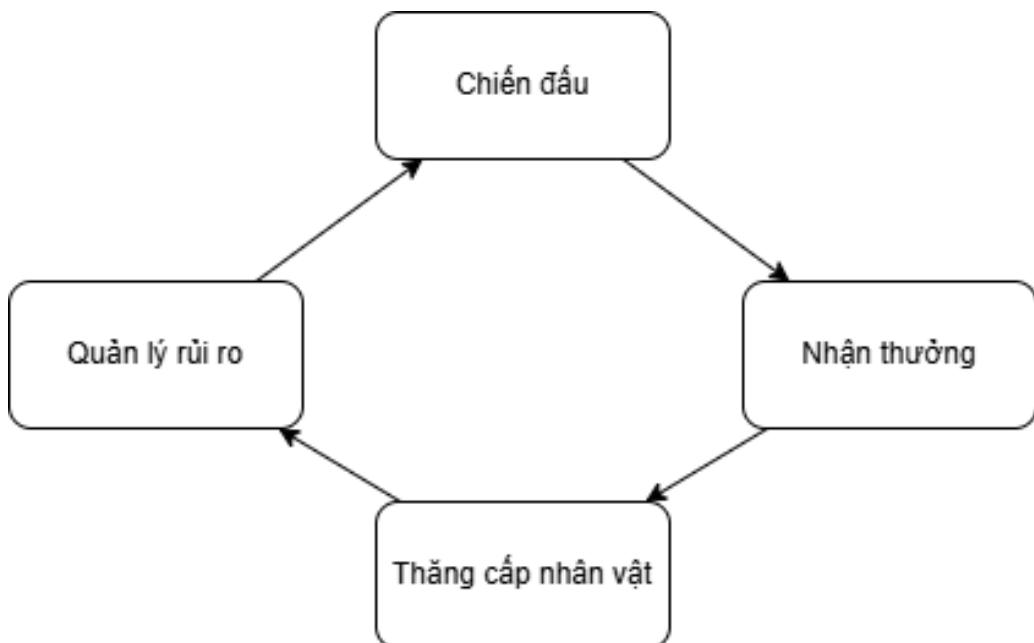
Các kỹ thuật này là nền tảng khi thiết kế combat có yêu cầu timing chính xác (né, dash, cast skill) và là tiền đề cho các quyết định ở phần thiết kế hệ thống.

2.2 Kiến thức nền tảng về Game Design

Game Design mô tả cách trò chơi vận hành: người chơi làm gì, vì sao họ muốn làm điều đó, và trò chơi phản hồi như thế nào[4, 5]. Một Action RPG online thường được phân tích theo các lớp: vòng lặp chơi, tiến trình nhân vật, chiến đấu, nội dung/không gian, kinh tế phần thưởng và UX/UI.

2.2.1 Core Loop và Meta Loop

Core Loop là vòng lặp hành vi ngắn hạn lặp lại liên tục. Với Action RPG, core loop thường xoay quanh: (i) tham gia giao tranh; (ii) nhận phần thưởng; (iii) dùng phần thưởng để mạnh lên; (iv) quay lại nội dung khó hơn[4].



Hình 1: Ví dụ mô hình hoá Core Loop trong Action RPG

Một core loop tốt cần dễ hiểu, có nhịp độ hợp lý và cung cấp “phản hồi thỏa mãn” sau mỗi vòng lặp (âm thanh, hiệu ứng, tiến bộ nhìn thấy được)[5].

Meta Loop là lớp động lực dài hạn (giữ chân người chơi): tăng cấp, mở khoá hệ thống, sưu tầm trang bị/kỹ năng, tối ưu hoá build, tham gia nội dung endgame[4]. Meta loop cần có các **môc rõ ràng** để tạo cảm giác tiến bộ, đồng thời đủ không gian cho người chơi cá nhân hoá (nhiều hướng build khác nhau).

2.2.2 Progression System (Hệ thống phát triển nhân vật)

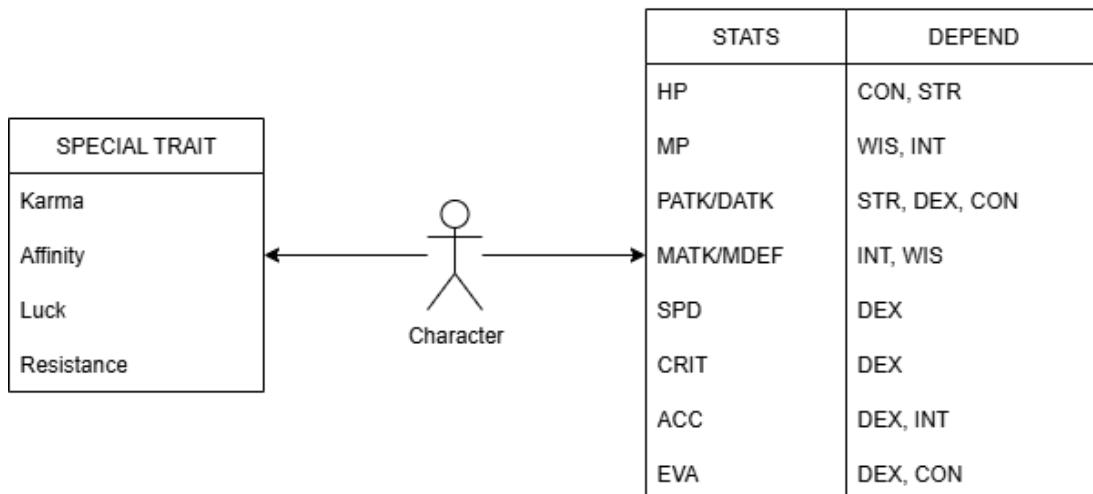
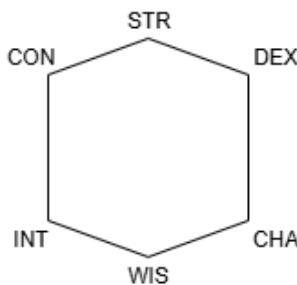
Progression là “xương sống” của Action RPG: quyết định tốc độ mạnh lên, cách người chơi ra quyết định build và cách nội dung được mở khoá theo thời gian[4, 5]. Trong phạm vi đề tài, hệ thống tiến trình được tổ chức theo ba lớp: (i) chỉ số nền (attributes/stats); (ii) tăng cấp và mở slot kỹ năng; (iii) tiến trình chủng tộc–nghề nghiệp (race/class) và điều kiện mở khoá.

Phân loại chỉ số: Primary vs. Secondary Một thiết kế phổ biến là chia chỉ số thành:

- **Primary Attributes:** người chơi phân bổ trực tiếp (ví dụ STR/DEX/CON/INT/WIS/CHA).
- **Secondary Stats:** suy ra từ primary (HP, MP, tốc độ, chí mạng, chính xác, né tránh, tấn công/phòng thủ, ...).

Cách phân lớp này giúp hệ thống dễ mở rộng và thuận lợi cho cân bằng: thay vì chỉnh trực tiếp mọi thứ, có thể điều chỉnh hệ số chuyển đổi giữa primary và secondary.

Hệ thống chỉ số 6 thuộc tính và quan hệ phụ thuộc Trong thiết kế tham chiếu của đề tài, nhân vật có 6 thuộc tính nền và các chỉ số chiến đấu suy diễn theo quan hệ phụ thuộc như Hình 10. Mục tiêu là đảm bảo *mỗi thuộc tính đều có giá trị* và hỗ trợ nhiều hướng build (tấn công, sống sót, cơ động, hỗ trợ).



Hình 2: Mô hình hoá thuộc tính nền, chỉ số suy diễn và nhóm trait đặc biệt

Bảng 1: Vai trò của 6 thuộc tính nền trong thiết kế build

Thuộc tính	Ý nghĩa thiết kế
STR	Sức mạnh thể chất; hỗ trợ sát thương cận chiến, tăng hiệu quả các hành động thiên về “đè lực”.
DEX	Sự nhanh nhẹn; hỗ trợ tốc độ, độ linh hoạt (dash/dodge), chí mạng và các hướng build cơ động.
CON	Thể lực; tăng sống sót (HP), giảm rủi ro bị “bốc hơi”, giúp cân bằng giữa tấn công và phòng thủ.
INT	Trí tuệ; hỗ trợ sát thương phép/khả năng kiểm soát, tương tác với cơ chế kỹ năng và hiệu ứng.
WIS	Minh triết; hỗ trợ tài nguyên (MP), hồi phục/kháng hiệu ứng và các vai trò hỗ trợ.
CHA	Sức hút/khả năng dẫn dắt; dùng làm biến điều kiện cho mở khoá, tương tác NPC/đội nhóm và một số hướng build đặc thù.

Bảng 2: Ví dụ quan hệ phụ thuộc Secondary Stats từ Primary Attributes

Chỉ số suy diễn	Phụ thuộc chính
HP (máu tối đa)	CON, STR
MP (năng lượng kỹ năng)	WIS, INT
PATK/DATK (tấn công vật lý)	STR, DEX, CON
MATK/MDEF (tấn công/phòng thủ phép)	INT, WIS
SPD (tốc độ)	DEX
CRIT (tỉ lệ chí mạng)	DEX
ACC (độ chính xác)	DEX, INT
EVA (né tránh)	DEX, CON

Từ góc độ triển khai, các công thức nên được cấu hình hoá (hệ số có thể thay đổi), ví dụ:

$$HP_{\max} = HP_0 + a \cdot CON + b \cdot STR, \quad MP_{\max} = MP_0 + c \cdot WIS + d \cdot INT$$

$$PATK = p_1 \cdot STR + p_2 \cdot DEX + p_3 \cdot CON, \quad SPD = s_1 \cdot DEX$$

Việc tách *hệ số* khỏi logic cho phép cân bằng mà không cần thay đổi mã nguồn, đồng thời hỗ trợ thử nghiệm nhiều “đường cong sức mạnh” (linear/exponential) ở giai đoạn tinh chỉnh[4].

Nhóm trait đặc biệt (Special Traits) và ý nghĩa thiết kế Bên cạnh hệ 6 thuộc tính, thiết kế tham chiếu còn đưa vào nhóm trait đặc biệt nhằm tăng chiều sâu meta-progression:

- **Karma:** trực lựa chọn/định hướng hành vi, có thể ảnh hưởng điều kiện mở khoá class và nội dung.
- **Affinity:** liên kết nguyên tố/hệ (Light/Dark/Nature/...), dùng cho phân nhánh class/kỹ năng và kháng.
- **Luck:** tăng biến thiên theo hướng tích cực (drop, tỉ lệ chí mạng, các cơ chế RNG), nhưng cần giới hạn để tránh phá cân bằng.
- **Resistance:** sức đề kháng/kháng hiệu ứng, giảm cảm giác bị “không chế liên hoàn” trong combat.

Nhóm trait này giúp hệ thống mở rộng theo hướng “điều kiện mở khoá” thay vì chỉ tăng số, tạo thêm mục tiêu dài hạn cho người chơi.

Leveling và mở slot kỹ năng Một lựa chọn thiết kế quan trọng là **level không trực tiếp cấp kỹ năng**, mà cấp **tài nguyên build** (điểm cộng chỉ số và slot để trang bị kỹ năng). Cách tiếp cận này tạo ra khác biệt giữa:

- **Học kỹ năng** (thu thập từ sách, nhiệm vụ, NPC, quái rơi, ...).

- **Trang bị kỹ năng** (bị giới hạn bởi slot, buộc người chơi ra quyết định).

Trong tham chiếu của đề tài: level tối đa 100; mỗi level cho +5 điểm chỉ số để phân bổ; các mốc level mở dần slot kỹ năng thường và slot kỹ năng kết hợp (combo).

Bảng 3: Ví dụ mốc level và phần thưởng mở slot kỹ năng

Mốc	Phần thưởng tiến trình
10, 20, 30	Mở dần các slot kỹ năng thường (slot 1–3), giúp người chơi hình thành bộ kỹ năng cơ bản.
40	Mở slot kỹ năng combo đầu tiên, bắt đầu giai đoạn “xây dựng lối chơi”.
50	Mốc nội dung/boss theo tuyến truyện và nới giới hạn stat cap (định hướng power spike).
60, 80, 100	Mở thêm các slot combo (2–4), hướng tới các tổ hợp mạnh và “ultimate combo”.
70, 90	Mở thêm slot thường (slot 4–5), tăng tính linh hoạt của bộ kỹ năng.

Kỹ năng kết hợp (Combo Skills) như cơ chế đa dạng hóa build Combo skill là kỹ năng hình thành từ việc kết hợp ít nhất 2 kỹ năng đã học (cơ bản hoặc theo class). Từ góc độ Game Design, combo skill tạo ra:

- **Động lực sưu tầm kỹ năng:** người chơi có lý do tìm thêm kỹ năng ngoài “meta” hiện tại.
- **Không gian thử nghiệm:** cùng một level nhưng khác bộ kỹ năng vẫn tạo trải nghiệm khác biệt.
- **Đòn bẩy endgame:** ở level cao, số slot combo tăng, cho phép build chuyên sâu.

Tuy nhiên, combo skill cũng là rủi ro cân bằng: cần giới hạn điều kiện trang bị (slot), chi phí tài nguyên (MP/stamina), và ràng buộc cooldown để tránh “một combo thống trị”.

Race/Class và cơ chế mở khoá theo điều kiện Hệ race/class là lớp meta-progression giúp người chơi định hình vai trò và phong cách chơi. Một mô hình thường gặp:

- **Race** cung cấp stat cap và trait bẩm sinh (thiên hướng dài hạn).
- **Class** cung cấp kỹ năng nghề nghiệp, cơ chế chiến đấu đặc thù và nhánh thăng tiến (định hình lối chơi).

Trong thiết kế tham chiếu, class được phân tầng (Basic/Intermediate/Advanced/Legendary/Hidden) với điều kiện mở khoá dạng **ngưỡng chỉ số** (STR/DEX/INT/...), **trait đặc biệt** (Karma/Affinity/Luck/IQ) và **thành tựu hành vi** (PvP, Arena, chỉ huy NPC, ...). Cách thiết kế này biến “mở class” thành mục tiêu gameplay thay vì chỉ là lựa chọn menu, đồng thời cho phép hệ thống nội dung gắn với điều kiện mở khoá.

2.2.3 Combat Design (Thiết kế chiến đấu)

Combat là trung tâm trải nghiệm của Action RPG; mọi hệ thống tiến trình cuối cùng đều “đổ” vào cảm giác chiến đấu[4, 5]. Vì vậy, thiết kế combat cần đồng thời kiểm soát nhịp độ, độ rõ ràng và độ công bằng.

Nhịp độ (Pacing) và cửa sổ rủi ro Pacing được tạo bởi tốc độ animation, thời gian cooldown, khoảng cách hiệu lực và mức “trừng phạt” khi người chơi mắc lỗi. Một thiết kế tốt thường đảm bảo:

- Có hành động nhanh (đánh thường/dash) để duy trì nhịp.
- Có hành động mạnh nhưng rủi ro (kỹ năng nặng, hồi chiêu dài) để tạo quyết định.
- Có cơ chế phòng thủ/cơ động (dodge/i-frame) nhưng bị giới hạn tài nguyên để tránh lạm dụng.

Tương tác giữa chỉ số và cảm giác điều khiển Khi chỉ số ảnh hưởng đến combat, cần tránh để “chỉ số thay thế kỹ năng người chơi”. Ví dụ:

- Tốc độ (SPD) và né tránh (EVA) nên tăng theo ngưỡng hợp lý để không biến combat thành “không thể trúng”.
- Chính xác (ACC) cần có vai trò đối trọng với EVA để cân bằng PvE/PvP.
- HP/Resistance tăng sống sót nhưng cần đi kèm trade-off (mất sát thương hoặc giảm cơ động).

Các nguyên tắc này liên quan trực tiếp tới thiết kế power budget và TTK trong phần cân bằng.

2.2.4 World System và Level Design

World/Level design xác định cấu trúc không gian và cách nội dung được phân phối: luồng di chuyển, mật độ giao tranh, checkpoint, nhịp nghỉ[5]. Với Action RPG online, world design còn liên quan tới:

- **Tổ chức phiên chơi:** nội dung dạng phòng (room), khu (zone) hoặc dungeon; ảnh hưởng tới cách đồng bộ multiplayer.
- **Điểm neo tiến trình:** checkpoint và mốc nội dung để tạo cảm giác “đi được một đoạn” thay vì cày vô hạn.
- **Phân tầng độ khó:** đảm bảo nội dung “vừa tầm” theo tiến trình, giảm cảm giác bế tắc.

Các nguyên lý này thường được kết hợp với progression (mốc level, mở slot, mở class) để tạo ra đường cong trải nghiệm liền mạch.

2.2.5 Hệ thống kinh tế (Game Economy)

Kinh tế trong game mô tả luồng tạo và tiêu thụ tài nguyên[4]. Về nguyên tắc, luôn cần đồng thời:

- **Nguồn (sources):** quà nhiệm vụ, rơi vật phẩm, phần thưởng theo phiêu, sự kiện.
- **Hút (sinks):** nâng cấp trang bị, chế tạo, tiêu hao vật phẩm, phí sửa chữa/duy trì, các cơ chế RNG.

Thiết kế kinh tế tốt đảm bảo người chơi luôn có mục tiêu chi tiêu hợp lý, tránh tích luỹ vô hạn hoặc cạn kiệt quá nhanh.

2.2.6 UX/UI Design trong game và định hướng phong cách Pixel Art

UX/UI trong game có nhiệm vụ: truyền tải trạng thái, hỗ trợ ra quyết định nhanh và tạo phản hồi nhất quán[5]. Với định hướng đồ họa **pixel art**, các nguyên tắc UI cần nhấn mạnh tính **đọc được** (readability) và **rõ trạng thái**:

- **Lưới pixel và căn chỉnh:** UI nên bám lưới, viền/thanh màu dùng bội số pixel để tránh mờ/nhỏe.
- **Tương phản cao:** chữ/icon cần tương phản nền đủ lớn; hạn chế gradient mịn gây giảm độ sắc.
- **Phân lớp thông tin:** trong combat chỉ hiển thị thông tin thiết yếu (HP/MP/cooldown/buff-debuff), thông tin chi tiết chuyển sang màn hình nhân vật.
- **Phản hồi hành động:** tấn công trúng, bị trúng, nhặt vật phẩm, nâng cấp thành công/thất bại cần có phản hồi hình–âm rõ ràng, nhất quán.

Ngoài ra, một bộ UI tối thiểu cho Action RPG online thường bao gồm:

- HUD chiến đấu: HP/MP, thanh kỹ năng theo slot, trạng thái cooldown, buff/debuff.
- Màn hình nhân vật: chỉ số, phân bổ điểm, class/race, trait đặc biệt.
- Inventory/Equipment: quản lý slot, trang bị, vật phẩm tiêu hao.
- Menu xã hội/cơ bản: bạn bè, party (nếu có), thiết lập (âm lượng, ngôn ngữ).

2.2.7 Balancing Fundamentals (Cân bằng game)

Cân bằng là quá trình điều chỉnh thông số để đảm bảo trải nghiệm hợp lý[4, 5]. Với Action RPG có hệ chỉ số và slot kỹ năng, các trực cân bằng thường gặp:

- **Tăng trưởng tuyến tính vs. luỹ thừa:** quyết định tốc độ mạnh lên theo thời gian và “độ chênh” giữa người chơi.

- **Time-to-Kill (TTK)**: thời gian hạ mục tiêu; quá thấp gây “bốc hơi”, quá cao gây lê thê.
- **Risk–Reward**: nội dung rủi ro cao phải thưởng tương xứng; nếu không sẽ bị bỏ qua.
- **Power Budget**: tổng sức mạnh được phân bổ qua chỉ số, trang bị, kỹ năng; cần kiểm soát để combo/skill không vượt ngân sách.

Trong bối cảnh có combo skill và điều kiện mở class theo trait, cân bằng cần chú ý thêm:

- Giới hạn khả năng “stack” (xếp chồng) các lợi thế: chí mạng + tốc độ + né tránh + kháng hiệu ứng.
- Đảm bảo nhiều hướng build đều có “đất diễn”: build cơ động mạnh nhưng mỏng; build trâu bò sống lâu nhưng thiếu bùng nổ; build phép mạnh nhưng phụ thuộc tài nguyên.

2.3 Tổng kết chương

Chương này đã trình bày các kiến thức nền tảng phục vụ thiết kế Action RPG online: đặc trưng real-time và đồng bộ multiplayer; các khái niệm game design như core loop, progression, combat, economy, UX/UI và balancing[4, 5, 2, 10]. Đồng thời, chương cũng mô hình hóa hệ chỉ số 6 thuộc tính, nhóm trait đặc biệt và cơ chế tiến trình thông qua mở slot kỹ năng/combo. Đây là cơ sở để chuyển sang phân tích yêu cầu (Chương 5) và đặc tả 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** trong đồ án để xây dựng một trò chơi Action RPG trực tuyến. Nội dung tập trung vào vai trò của từng công nghệ trong kiến trúc tổng thể, các ưu điểm/hạn chế và lý do phù hợp với bối cảnh hệ thống game online thời gian thực[3, 2].

Về mặt chức năng, hệ thống có thể chia thành các lớp chính:

- **Client/Game Engine**: hiển thị đồ họa, nhận input, mô phỏng cục bộ (visual/animation), UI và giao tiếp mạng.
- **Backend/Application Server**: xử lý nghiệp vụ game, xác thực, quản lý phiên, đồng bộ trạng thái và cung cấp API.
- **Data & Storage**: lưu trữ lâu dài (progress người chơi), cache/pub-sub, và kho dữ liệu cấu hình/tài nguyên.
- **Tooling & Process**: quản lý mã nguồn, quản lý tác vụ, tài liệu hoá và công cụ thiết kế.

3.1 Công nghệ phía client

3.1.1 Game engine: Unity

Unity là lựa chọn phù hợp để hiện thực client trong bối cảnh đồ án nhờ hệ sinh thái hoàn chỉnh và tài liệu phong phú[6]. Các lý do chính:

- **Pipeline đồ họa và nội dung:** hỗ trợ sprite/animation, quản lý scene, prefab và asset import giúp tăng tốc phát triển nội dung.
- **UI framework:** cung cấp hệ thống UI phục vụ HUD chiến đấu, inventory, menu và các màn hình meta-progression.
- **Khả năng mở rộng:** cho phép tổ chức mã theo hướng component-based; có thể mở rộng sang hướng ECS tùy mức độ phức tạp[3].

Trong giai đoạn 1, Unity được xem là nền tảng client nhằm hỗ trợ **thiết kế trải nghiệm** và **định nghĩa cấu trúc giao tiếp** với backend; chưa đặt trọng tâm vào tối ưu hiệu năng production.

3.1.2 Ngôn ngữ lập trình: C#

C# là ngôn ngữ chính khi phát triển với Unity[11]. Các đặc điểm phù hợp:

- Hỗ trợ tốt lập trình hướng đối tượng và tổ chức hệ thống gameplay theo mô-đun.
- Có thư viện tiêu chuẩn mạnh cho cấu trúc dữ liệu, toán học và xử lý bất đồng bộ ở mức client.
- Tích hợp trực tiếp với API Unity để điều khiển animation, UI và xử lý input[6].

3.1.3 Nguyên tắc xử lý logic ở client

Đối với game online, cần tách bạch rõ giữa:

- **Client logic (hiển thị):** UI, animation, hiệu ứng, nội suy chuyển động và dự đoán phản hồi để giảm cảm giác trễ.
- **Server logic (quyết định):** các kết quả quan trọng như tính sát thương, xác nhận loot, cập nhật tài nguyên và thay đổi trạng thái chính thức[2].

Nguyên tắc này giúp giảm rủi ro gian lận và đảm bảo tính nhất quán khi nhiều người chơi tương tác.

3.2 Công nghệ phía server

3.2.1 Nền tảng runtime: Node.js

Node.js phù hợp với backend game online nhờ mô hình I/O bất đồng bộ, thuận lợi khi phải xử lý đồng thời nhiều kết nối mạng (đặc biệt là WebSocket)[12]. Với đặc thù đồ án, Node.js hỗ trợ:

- Xây dựng API nhanh, dễ mở rộng theo mô-đun.
- Tích hợp tốt với hệ sinh thái thư viện cho xác thực, logging, validation và kết nối DB.

3.2.2 Framework: NestJS và TypeScript

NestJS cung cấp kiến trúc rõ ràng theo hướng module + dependency injection, phù hợp khi hệ thống cần phân rã theo phân hệ (auth/profile, session, character, inventory, ...)[7]. TypeScript giúp tăng độ an toàn của mã nguồn nhờ kiểu tường minh, giảm lỗi trong phát triển trung hạn.

Các lý do lựa chọn:

- **Tổ chức mã nguồn:** module hoá tự nhiên, dễ quản lý ranh giới nghiệp vụ và tái sử dụng service dùng chung.
- **Hỗ trợ đa giao thức:** REST cho tác vụ quản trị/truy vấn; WebSocket gateway cho tương tác thời gian thực[7].
- **Khả năng mở rộng:** có thể bắt đầu theo kiểu monolithic-module để giảm độ phức tạp, và có cơ sở để tách dịch vụ khi hệ thống tăng quy mô.

3.2.3 Tác vụ nền (background jobs)

Trong game online, một số xử lý không cần chạy trực tiếp trên request realtime, ví dụ: đồng bộ dữ liệu định kỳ, tổng hợp log, xử lý mail hệ thống, hoặc quản lý sự kiện theo lịch. Các tác vụ này có thể được tách thành **job worker** chạy độc lập với API server để giảm tải cho luồng realtime, đồng thời dễ kiểm soát retry và lịch chạy.

3.3 Công nghệ giao tiếp mạng

3.3.1 HTTP/REST

HTTP phù hợp với các tác vụ không yêu cầu độ trễ thấp hoặc không cần kết nối hai chiều liên tục:

- Đăng ký/đăng nhập, chọn profile, lấy thông tin nhân vật.
- Tải dữ liệu cấu hình khi khởi động (tùy cách triển khai Config Manager).
- Các thao tác quản trị (admin tools) và truy vấn lịch sử[7].

3.3.2 WebSocket

WebSocket phù hợp cho các tương tác realtime trong phiên chơi:

- Cập nhật trạng thái chiến đấu, sự kiện tức thời trong instance.
- Thông báo thay đổi trạng thái (loot, revive, buff/debuff) và đồng bộ theo nhịp tick.

Với mô hình server authoritative, WebSocket đóng vai trò kênh truyền nhận input/snapshot, kết hợp các kỹ thuật giảm cảm giác trễ ở client[2, 10].

3.4 Công nghệ lưu trữ dữ liệu

3.4.1 MongoDB

MongoDB phù hợp cho dữ liệu game do đặc tính document linh hoạt, dễ biểu diễn các cấu trúc lồng nhau (inventory items, island buildings, npc collection) và thích nghi khi schema thay đổi trong quá trình phát triển[8]. Trong bối cảnh đồ án:

- **Dữ liệu tiến trình (dynamic)**: user/profile/session/character/inventory/island/npc cần lưu bền vững và truy vấn theo khoá định danh.
- **Dữ liệu cấu hình (static)**: các collection cấu hình như item/building/skill có thể được import từ pipeline TSV và nạp vào bộ nhớ khi chạy.

3.4.2 Redis

Redis được định hướng sử dụng như lớp hỗ trợ hiệu năng và phối hợp realtime[9]:

- **Cache**: giảm tải đọc DB cho dữ liệu truy cập thường xuyên (ví dụ session mapping, một phần config hot, trạng thái online).
- **Pub/Sub**: chia sẻ sự kiện giữa nhiều tiến trình/instance backend khi mở rộng theo chiều ngang.

3.4.3 Object Storage (tùy chọn theo giai đoạn)

Một số dữ liệu dạng file (asset pack, snapshot log, hoặc các tệp cấu hình đóng gói) có thể lưu trong object storage theo chuẩn S3-compatible. Trong giai đoạn 1, phần này chỉ dùng ở mức định hướng kiến trúc để sẵn sàng mở rộng; chưa đặt trọng tâm vận hành production.

3.5 Công cụ quản trị game configuration theo hướng data-driven

Trong đồ án, **pipeline cấu hình** chỉ áp dụng cho **các bảng cấu hình/thông số game** (ví dụ: định nghĩa item/building/skill và các hệ số cân bằng), không áp dụng cho toàn bộ dữ liệu tiến trình người chơi.

Định hướng quy trình:

- Thiết kế bảng cấu hình bằng **Google Sheets** để thuận lợi nhập liệu và review.
- Export sang **TSV** nhằm đảm bảo format đơn giản, dễ parse và thân thiện khi version control.
- Import TSV vào các **collections cấu hình** trong DB (tách biệt với collections progress).
- Khi khởi động hoặc khi tải game, hệ thống nạp config từ DB vào **Config Manager** (in-memory) để truy xuất thông nhất trong runtime.

Lợi ích chính của cách tiếp cận data-driven:

- Giảm phụ thuộc vào hard-code tham số, hỗ trợ điều chỉnh cân bằng nhanh.
- Tăng khả năng kiểm soát thay đổi: TSV có thể được quản lý theo commit để truy vết phiên bản cấu hình.
- Tạo tiền đề cho mở rộng nội dung và kiểm thử A/B cấu hình trong tương lai.

3.6 Công cụ hỗ trợ phát triển và tài liệu hoá

3.6.1 Quản lý mã nguồn và phối hợp nhóm

Git/GitHub được sử dụng để quản lý phiên bản mã nguồn và phối hợp theo pull request; GitHub Projects dùng để quản lý tác vụ theo bảng, gán trách nhiệm và theo dõi tiến độ[13, 14].

3.6.2 Công cụ mô hình hoá và tài liệu

Các sơ đồ use case/luồng nghiệp vụ/kiến trúc có thể được xây dựng bằng công cụ UML/BPMN hoặc Mermaid để đảm bảo tính nhất quán khi trình bày[15]. Báo cáo và tài liệu hệ thống được soạn thảo bằng L^AT_EX nhằm đảm bảo định dạng học thuật và quản lý nội dung theo từng chương độc lập[16].

3.7 Tổng kết

Chương này đã trình bày các công nghệ chính được sử dụng hoặc định hướng sử dụng cho đồ án: Unity và C# ở phía client[6, 11]; Node.js, NestJS và TypeScript ở phía server[12, 7]; MongoDB và Redis cho lớp dữ liệu[8, 9]; cùng các công cụ hỗ trợ phát triển và tài liệu hoá[13, 14, 16]. Ngoài ra, chương cũng nêu định hướng pipeline quản trị **game configuration** theo hướng data-driven nhằm phục vụ thiết kế và mở rộng nội dung trong các giai đoạn tiếp theo.

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 cho cấu trúc nội dung và hệ thống gameplay; (ii) tài liệu nền tảng và tài liệu kỹ thuật giúp chuẩn hóa lập luận thiết kế, đặc biệt với bài toán game online thời gian thực.

Mục tiêu của chương là làm rõ mối liên hệ **Nguồn tham khảo** → **Bài học rút ra** → **Cách áp dụng vào đề tài**, nhằm tránh tình trạng liệt kê cảm hứng một cách chung chung, đồng thời đảm bảo các quyết định thiết kế ở các chương sau có cơ sở tham chiếu rõ ràng.

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

Nhóm lựa chọn và phân tích nguồn tham khảo theo các tiêu chí sau:

- **Có hệ thống tiến trình (progression) rõ ràng**: thể hiện được cách tăng trưởng sức mạnh, mở khoá nội dung, hoặc mở rộng năng lực nhân vật theo thời gian.
- **Có mô-đun hệ thống đặc trưng**: có thể trừu tượng hoá thành một nhóm tính năng độc lập như leo tháp/boss, gacha/tuyển dụng, căn cứ/đảo cá nhân, dungeon theo phiên (instance), hoặc hệ thống NPC vận hành.
- **Có đủ thông tin để phân rã theo lớp thiết kế**: vòng lặp chơi (loop), rủi ro-phản thường, nhịp độ (pacing), tương tác người chơi, và tổ chức nội dung.

Bên cạnh nguồn cảm hứng nội dung, nhóm sử dụng các tài liệu nền tảng để chuẩn hóa thuật ngữ và phương pháp lập luận trong thiết kế game [5, 4], kết hợp tài liệu về kiến trúc hệ thống thời gian thực [3] và networking cho game online [10, 2]. Các tài liệu công nghệ được dùng như nguồn tham khảo cho quyết định kiến trúc và ràng buộc triển khai ở mức đồ án [6, 7, 8, 9, 12].

4.2 Nguồn cảm hứng cho trực tiến trình: leo tháp, boss tầng và checkpoint

4.2.1 Sword Art Online (Aincrad): cấu trúc leo tầng như “xương sống” nội dung

Sword Art Online (Aincrad) mô tả thế giới game với cấu trúc leo tầng tuyển tính, trong đó mỗi tầng có hệ sinh thái và thử thách riêng; boss tầng đóng vai trò “cổng kiểm soát tiến trình” [17]. Ở góc độ thiết kế hệ thống, nguồn tham khảo này gợi ý ba điểm quan trọng:

- **Phân lớp nội dung theo tầng (content stratification)**: giúp thiết kế và kiểm soát độ khó theo tiến trình, đồng thời tạo cảm giác “mở rộng thế giới” theo từng mốc.
- **Boss như mốc kiểm chứng build**: yêu cầu người chơi tối ưu trang bị/kỹ năng/chỉ số, từ đó tạo động lực chuẩn bị (farm, craft, nâng cấp) trước khi vượt mốc.

- **Checkpoint tiến độ rõ ràng:** qua tầng là một cột mốc, phù hợp để gắn phần thưởng, mở khoá, và tạo cảm giác thành tựu.

Áp dụng vào đề tài: Đề tài kẽ thửa ý tưởng một trục nội dung dài hạn theo dạng “Tháp trung tâm nhiều tầng”, trong đó mỗi tầng là một mức thử thách có thể thiết kế theo nhịp tăng trưởng sức mạnh. Đề phù hợp trải nghiệm hiện đại và giảm cảm giác mệt rãnh, tiến trình được định hướng kèm checkpoint theo mốc và có thể mở rộng thêm các tầng thử thách (optional/challenge) nhằm tạo không gian cho người chơi tối ưu build mà không phá vỡ mạch tiến trình chính.

4.3 Nguồn cảm hứng cho “thế giới vận hành”: NPC có vai trò và hệ thống nhân sự

4.3.1 Overlord: NPC như tác nhân hệ thống và tổ chức vận hành

Overlord nhấn mạnh cảm giác “thế giới sống”, trong đó NPC không chỉ là đối tượng tương tác tĩnh mà có vai trò, động cơ, và cấu trúc tổ chức [18]. Khi trừu tượng hoá sang thiết kế hệ thống, các bài học chính gồm:

- **NPC như tác nhân vận hành (system agents):** NPC có thể tham gia sản xuất, thu thập, phòng thủ, hoặc thực hiện các nhiệm vụ nền; tạo lớp hệ thống “chạy ngầm” hỗ trợ meta-progression.
- **Tổ chức tạo chiêu sâu chiến lược:** thay vì chỉ tăng chỉ số nhân vật, người chơi có thêm mục tiêu tối ưu “đội ngũ vận hành” theo vai trò và độ hiếu.
- **Tính tích luỹ và bền vững:** tiến trình không nhất thiết reset theo phiên; các hệ thống nền (nhân sự, công trình, tài nguyên) tạo động lực quay lại.

Áp dụng vào đề tài: Đề tài định hướng xây dựng lớp NPC như một tài nguyên dài hạn: vừa có thể tham gia gameplay (theo mô hình thu thập/tuyển dụng), vừa có thể tham gia vận hành ở các hệ thống meta (ví dụ: đảo cá nhân, công trình, thu thập tài nguyên). Cách tiếp cận này giúp tách rõ “chiến đấu theo phiên” và “phát triển dài hạn”, đồng thời tạo không gian mở rộng nội dung về sau.

4.4 Nguồn cảm hứng cho cơ chế tuyển dụng/thu thập: gacha và meta-progression

4.4.1 Pick Me Up! Infinite Gacha: gacha như vòng lặp meta và động lực sưu tầm

Pick Me Up! Infinite Gacha cung cấp khung tham khảo cho cơ chế tuyển dụng dựa trên xác suất, kết hợp meta-progression thông qua bộ sưu tập (collection) [19]. Các điểm thiết kế có thể rút ra:

- **Rarity gắn với vai trò:** độ hiếm không chỉ là “mạnh hơn”, mà còn có thể là khác biệt về vai trò/hệ kỹ năng, tạo đa dạng chiến thuật.
- **Vòng lặp meta rõ ràng:** chơi nội dung → nhận tài nguyên → quay/tuyển dụng → tối ưu đội hình → tiếp cận nội dung khó hơn.
- **Giảm “dead-end progression”:** luôn tồn tại khả năng cải thiện thông qua thu thập và nâng cấp, ngay cả khi người chơi không vượt được một mốc khó ngay lập tức.

Áp dụng vào đề tài: Đề tài định hướng hệ thống tuyển dụng (gacha) như cơ chế tạo biến thiên tiến trình và động lực sưu tầm cho nhóm thực thể dài hạn (ví dụ NPC/đơn vị hỗ trợ). Trong phạm vi đồ án, trọng tâm là thiết kế hệ thống ở mức mô-đun và dữ liệu (tỉ lệ, độ hiếm, pity/exchange), nhằm đảm bảo về sau có thể kiểm soát cân bằng và tính minh bạch của xác suất.

4.5 Nguồn cảm hứng cho trải nghiệm Action RPG online: nhịp combat và cảm giác điều khiển

4.5.1 Arcane Odyssey: ưu tiên phản hồi thao tác và nhấn mạnh kỹ năng người chơi

Arcane Odyssey là tham chiếu về trải nghiệm combat và di chuyển trong môi trường online, nơi cảm giác điều khiển và phản hồi thao tác tác động mạnh đến mức độ “đã tay” của Action RPG [20]. Ba điểm thiết kế nổi bật:

- **Phản hồi tức thời:** hiệu ứng trúng đòn, bị đẩy lùi, hoặc né tránh cần thể hiện rõ ràng để người chơi “đọc” được kết quả thao tác.
- **Chiến đấu dựa trên vị trí và timing:** kết quả phụ thuộc nhiều vào positioning và thời điểm ra đòn/né, không thuần tuý dựa trên chỉ số.
- **Tối giản rào cản thao tác:** UI và điều khiển cần phục vụ core loop, tránh che khuất hoặc làm gián đoạn nhịp combat.

Áp dụng vào đề tài: Đề tài định hướng combat theo phong cách hack-and-slash: nhịp nhanh, có né/dash và kỹ năng chủ động. Để bảo toàn trải nghiệm trong môi trường online, phần thiết kế kiến trúc ưu tiên mô hình server-authoritative và các kỹ thuật bù trễ phù hợp [10, 2], giúp kết quả combat nhất quán và hạn chế sai lệch do latency.

4.5.2 Soul Knight Prequel: vòng lặp phiên chơi ngắn và thưởng rõ ràng theo phiên

Soul Knight Prequel là tham chiếu cho cách tổ chức vòng lặp chơi ngắn nhưng có khả năng lặp lại cao: vào phiên → chiến đấu → rơi đồ/thu hoạch → kết thúc phiên và nhận thưởng [21]. Các điểm có thể áp dụng:

- **Session ngắn, tái chơi cao:** phù hợp nhịp chơi phổ thông, dễ “quay lại thêm một ván”.
- **Reward rõ theo phiên:** phần thưởng kết phiên giúp củng cố động lực và tạo nhịp meta đều đặn.
- **Build dễ hiểu:** giảm tải học tập, cho phép người chơi thử nghiệm trang bị/kỹ năng nhanh hơn.

Áp dụng vào đề tài: Đề tài định hướng dungeon/hoạt động thử thách theo dạng instance-based, giúp đóng gói nội dung, kiểm soát tài nguyên hệ thống, và dễ mở rộng thành nhiều “kịch bản phiên” trong tương lai.

4.6 Nguồn cảm hứng cho hệ thống đảo/căn cứ: tiến trình dài hạn và quản trị tài nguyên

4.6.1 Clash of Clans: công trình theo thời gian và meta-progression bền vững

Clash of Clans là tham chiếu tiêu biểu cho hệ thống căn cứ cá nhân: người chơi thu thập tài nguyên, đầu tư vào công trình, và tiến trình chịu ảnh hưởng bởi thời gian xây dựng/nâng cấp [22]. Các bài học chính:

- **Căn cứ/đảo như “nhà” của người chơi:** tạo cảm giác sở hữu và mục tiêu dài hạn ngoài combat.
- **Kinh tế nguồn-nơi tiêu (source-sink):** thiết kế cân bằng giữa cách kiếm và cách tiêu tài nguyên để duy trì nhịp chơi hợp lý.
- **Return loop dựa trên thời gian:** cơ chế thời gian giúp hình thành thói quen quay lại, đồng thời tạo không gian cho chiến lược tối ưu hoá.

Áp dụng vào đề tài: Đề tài định hướng một lớp “đảo cá nhân” phục vụ meta-progression: xây dựng/nâng cấp công trình, quản trị tài nguyên, và có thể kết hợp với nhân sự/NPC để vận hành. Cách tiếp cận này giúp mở rộng tiến trình ngoài chiến đấu và tạo thêm mục tiêu dài hạn cho người chơi.

4.7 Bảng tổng hợp: đối chiếu nguồn tham khảo và hướng áp dụng

Bảng 4 tóm tắt mối liên hệ giữa nguồn tham khảo và các hướng áp dụng chính vào thiết kế hệ thống của đề tài.

Nguồn tham khảo	Yếu tố rút ra	Áp dụng vào đề tài
SAO (Aincrad) [17]	Leo tầng, boss tầng, checkpoint	Trục tháp nhiều tầng; phân lớp nội dung; checkpoint theo mốc
Overlord [18]	NPC có vai trò; thế giới vận hành	Thiết kế lớp NPC/nhân sự; liên kết với hệ thống meta (đảo/công trình)
Pick Me Up! [19]	Gacha, rarity, collection-driven progression	Thiết kế tuyển dụng theo xác suất; vòng lặp meta dựa trên sưu tầm/nâng cấp
Arcane Odyssey [20]	Cảm giác điều khiển; combat theo timing/vị trí	Định hướng combat real-time; ràng buộc thiết kế networking cho tính nhất quán
Soul Knight Prequel [21]	Phiên chơi ngắn; reward theo phiên	Định hướng dungeon/instance theo phiên; nhịp thưởng rõ ràng
Clash of Clans [22]	Căn cứ; tài nguyên; thời gian nâng cấp	Thiết kế đảo cá nhân; vòng lặp tài nguyên; tiến trình theo thời gian

Bảng 4: Đổi chiều nguồn tham khảo và hướng áp dụng vào đề tài

4.8 Nguồn tham khảo nền tảng và kỹ thuật: cơ sở cho lập luận thiết kế và ràng buộc triển khai

Bên cạnh nguồn cảm hứng nội dung, đề tài cần cơ sở học thuật và kỹ thuật cho các quyết định thiết kế về hệ thống, dữ liệu và vận hành:

- **Lập luận thiết kế game:** khái niệm core loop, progression, economy, UX và cách đánh giá trải nghiệm theo “lăng kính” [5, 4].
- **Kiến trúc hệ thống thời gian thực:** tổ chức các lớp hệ thống, vòng lặp cập nhật, và định hướng kiến trúc engine/real-time [3].
- **Networking cho game online:** mô hình server-authoritative và các kỹ thuật giảm ảnh hưởng của latency (prediction/interpolation/reconciliation) [10, 2].
- **Tài liệu công nghệ:** engine phía client, nền tảng backend và các hệ lưu trữ/caching phục vụ kết nối và quản lý trạng thái [6, 7, 12, 8, 9].

Các nguồn tham khảo này đóng vai trò “khung chuẩn” để đảm bảo phần thiết kế ở các chương sau nhất quán về thuật ngữ, có cơ sở khi lựa chọn mô hình kiến trúc, và có định hướng mở rộng hợp lý cho các mô-đun gameplay trong tương lai.

4.9 Tổng kết chương

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

- Nhóm nguồn cảm hứng (SAO, Overlord, Pick Me Up, Arcane Odyssey, Soul Knight Prequel, Clash of Clans) cung cấp khung ý tưởng để thiết kế trực tiến trình leo tầng, lớp NPC/nhân sự, cơ chế tuyển dụng theo xác suất, dungeon theo phiên và hệ thống đảo/căn cứ [17, 18, 19, 20, 21, 22].
- Nhóm tài liệu nền tảng/kỹ thuật cung cấp cơ sở để chuẩn hoá lập luận thiết kế và ràng buộc kiến trúc của game online thời gian thực [5, 4, 3, 10, 2, 6, 7, 8, 9].

Từ đó, các chương sau có thể triển khai phân tích yêu cầu và thiết kế hệ thống theo hướng nhất quán: hệ thống gameplay được mô-đun hoá, dữ liệu được cấu trúc để mở rộng, và kiến trúc vận hành phù hợp với bối cảnh multiplayer real-time.

5 Phân tích yêu cầu

Chương này xác định yêu cầu cho hệ thống game *Fortress of the Fallen* ở mức **phân tích hệ thống**. Nội dung bao gồm: phạm vi, tác nhân, nhóm yêu cầu chức năng theo phân hệ, yêu cầu phi chức năng và yêu cầu dữ liệu. Các yêu cầu ở chương này là cơ sở để xây dựng mô hình use case/luồng nghiệp vụ (Chương 6) và kiến trúc + thiết kế dữ liệu (Chương 7).

5.1 Phạm vi và giả định

5.1.1 Phạm vi giai đoạn 1

Giai đoạn 1 tập trung vào **thiết kế hệ thống**. Do đó, các yêu cầu được mô tả theo hướng:

- làm rõ **các tính năng cần tồn tại** và cách chúng liên kết với nhau;
- làm rõ **dữ liệu cần lưu** để đảm bảo nhất quán trạng thái người chơi;
- làm rõ **ranh giới trách nhiệm** giữa client và server cho các tác vụ request-response và real-time.

5.1.2 Giả định

- Hệ thống vận hành theo mô hình client-server; server giữ vai trò quyết định cho các kết quả quan trọng (reward/currency/progression).
- Dữ liệu trong hệ thống được chia thành:
 - **Player progression/state (động)**: thay đổi theo hành vi người chơi.
 - **Game configuration (tĩnh)**: định nghĩa và tham số gameplay (item/building/skill/rates/...).

- Pipeline cấu hình chỉ áp dụng cho game configuration, không áp dụng cho toàn bộ dữ liệu DB.

5.2 Tác nhân và ranh giới hệ thống

5.2.1 Tác nhân

- **Player:** người chơi sử dụng client để trải nghiệm gameplay, quản lý tiến trình và tài sản.
- **Admin:** người quản trị hệ thống, có quyền thao tác trên user, log, mail và cấu hình sự kiện.

5.2.2 Ranh giới hệ thống

Hệ thống được chia theo các nhóm thành phần:

- **Client:** hiển thị (render), UI, input, điều hướng màn hình, nội suy hiển thị; gửi yêu cầu/ý định hành động.
- **Server:** xác thực, quản lý phiên, xử lý nghiệp vụ, xác nhận kết quả, ghi DB.
- **Data layer:** DB cho progression/state; collections cấu hình cho configuration; cache/session runtime (định hướng).

5.3 Yêu cầu chức năng

Phần này mô tả yêu cầu chức năng theo từng phân hệ. Mỗi yêu cầu được gắn mã định danh để thuận tiện đối chiếu về sau.

5.3.1 Nhóm A: Authentication & Profile

- **FR-A01** – Đăng ký tài khoản: Player có thể đăng ký bằng username/email và mật khẩu.
- **FR-A02** – Đăng nhập: Player đăng nhập, nhận token phiên; server ghi nhận `last_login`.
- **FR-A03** – Khôi phục mật khẩu: cung cấp luồng khôi phục (tối thiểu ở mức thiết kế).
- **FR-A04** – Quản lý thiết lập người dùng: lưu *master volume*, *sfx volume*, *language*, *notifications*.
- **FR-A05** – Tạo game profile: một user có thể tạo nhiều profile (phục vụ nhiều server/nhân vật).
- **FR-A06** – Chọn game profile: chọn profile để vào game; cập nhật `last_played`.
- **FR-A07** – Quản lý session runtime: hệ thống lưu trạng thái phiên hiện hành (profile/character/room_

5.3.2 Nhóm B: Character Creation & Character Data

Yêu cầu tạo nhân vật

- **FR-B01** – Tạo nhân vật: Player tạo CHARACTER thuộc một GAME_PROFILE.
- **FR-B02** – Tuỳ chọn cơ bản: đặt tên, chọn giới tính (nếu có), chọn **race** và **class** khởi đầu.
- **FR-B03** – Tuỳ chọn ngoại hình: lưu các tham số appearance (hair/skin/face) để tái hiện nhất quán.
- **FR-B04** – Xoá nhân vật: Player có thể xoá CHARACTER theo chính sách (có thể yêu cầu xác nhận).
- **FR-B05** – Chọn nhân vật: Player chọn nhân vật để vào phiên chơi.

Race system (yêu cầu mức dữ liệu và mở khoá) Dựa trên thiết kế chủng tộc, hệ thống cần thoả:

- **FR-B06** – Race ảnh hưởng **stat cap** và **trait bẩm sinh**.
- **FR-B07** – Race hiếm có thể yêu cầu điều kiện mở khoá (ví dụ: Karma người, achievement, hoặc điều kiện tiến trình).
- **FR-B08** – Hiển thị thông tin race: client có thể xem mô tả, trait và stat cap trước khi xác nhận tạo nhân vật.

5.3.3 Nhóm C: Stats, Leveling, Skill Slots

Level system

- **FR-C01** – Level tối đa là **100**.
- **FR-C02** – Mỗi lần lên level nhận **+5 stat points** để phân bổ.
- **FR-C03** – Level không tự động cấp kỹ năng; kỹ năng được thu thập từ nguồn khác (loot/quest/NPC/...).

Stat allocation

- **FR-C04** – Player có thể phân bổ điểm vào các thuộc tính nền: STR, DEX, CON, INT, WIS, CHA.
- **FR-C05** – Hệ thống tính các chỉ số suy diễn (HP/MP/ATK/DEF/CRIT/ACC/EVA/...) từ thuộc tính nền theo công thức cấu hình.
- **FR-C06** – Ràng buộc stat cap theo race: không cho phân bổ vượt ngưỡng cho phép.

Skill slots và combo slots Để hỗ trợ lối chơi đa dạng, hệ thống áp dụng cơ chế slot:

- **FR-C07** – Có **normal skill slots** để trang bị kỹ năng thường.
- **FR-C08** – Có **combo skill slots** để trang bị kỹ năng kết hợp (combo).
- **FR-C09** – Slot được mở dần theo level (milestone). Một cấu hình milestone tham chiếu:
 - Normal slots: mở dần đến tối đa 5 slot.
 - Combo slots: mở dần tại các mốc 40/60/80/100 đến tối đa 4 slot (slot cuối là *ultimate combo*).

Thu thập và quản lý kỹ năng

- **FR-C10** – Player có thể sở hữu danh sách kỹ năng đã học (learned skills).
- **FR-C11** – Player có thể trang bị kỹ năng vào slot; hệ thống kiểm tra điều kiện (class/race/level).
- **FR-C12** – Combo skill chỉ hợp lệ nếu thoả quy tắc kết hợp (theo cấu hình).

5.3.4 Nhóm D: Class System và mở khoá theo điều kiện

Phân tầng class Hệ thống class được phân tầng để tạo meta-progression:

- **FR-D01** – Hỗ trợ nhiều nhánh class (Warrior/Archer/Mage/Healer/Rogue/Tactician).
- **FR-D02** – Hỗ trợ nhiều tầng class: Basic, Intermediate, Advanced, Legendary và Hidden.

Điều kiện mở khoá class

- **FR-D03** – Mở khoá class dựa trên điều kiện kết hợp:
 - ngưỡng chỉ số (STR/DEX/CON/INT/WIS/CHA);
 - trait đặc biệt (Karma/Affinity/Luck/Resistance/...);
 - điều kiện tiến trình/achievement (PvP, quest, sưu tầm, ...).
- **FR-D04** – Player có thể xem trước điều kiện và tiến độ đạt điều kiện để mở class.
- **FR-D05** – Khi mở class thành công, Player có thể chuyển class theo chính sách (ví dụ: cần xác nhận, giới hạn số lần, hoặc chi phí).

5.3.5 Nhóm E: Inventory, Equipment, Resources

- **FR-E01** – Inventory là 1:1 với character, có số slot tối đa và số slot đã dùng.
- **FR-E02** – Hỗ trợ currency cơ bản: gold và gem.
- **FR-E03** – Item trong inventory là **item instance** (uid/quantity/slot/enhancement/durability) và tham chiếu đến **item definition** qua **item_ref_id**.
- **FR-E04** – Stack rule: item có **max_stack** theo cấu hình.
- **FR-E05** – Equip: trang bị vũ khí/giáp theo slot (theo cấu hình), cập nhật chỉ số hiệu lực.
- **FR-E06** – Enhance: nâng cấp trang bị theo chi phí và quy tắc; ghi nhận **enhancement_level**.
- **FR-E07** – Durability (tùy chọn theo thiết kế): giảm độ bền theo sử dụng và có cơ chế sửa/khôi phục.

5.3.6 Nhóm F: Combat & Gameplay Modes

Nội dung combat

- **FR-F01** – Central Tower: chọn tầng và tham gia nội dung theo tầng.
- **FR-F02** – Dungeon: tham gia nội dung PvE theo phiên (instance).
- **FR-F03** – Arena: tham gia PvP arena (tối thiểu ở mức thiết kế).

Hành động trong combat

- **FR-F04** – Normal attack.
- **FR-F05** – Cast active skill.
- **FR-F06** – Dodge/Dash.
- **FR-F07** – Use consumable item.
- **FR-F08** – Revive (nếu có cơ chế hồi sinh).

Checkpoint và reward

- **FR-F09** – Checkpoint: lưu mốc tiến độ theo nội dung (tower/dungeon) theo chính sách.
- **FR-F10** – Loot delivery: reward sau combat được ghi vào inventory; hệ thống kiểm tra capacity/stack.
- **FR-F11** – Reward logging: các giao dịch reward/currency cần được ghi log nghiệp vụ để truy vết.

5.3.7 Nhóm G: Recruitment (Gacha)

- **FR-G01** – Banner: hiển thị danh sách banner đang hoạt động.
- **FR-G02** – Rates: hiển thị tỉ lệ rơi theo banner (minh bạch).
- **FR-G03** – Summon 1x và **FR-G04** – Summon 10x.
- **FR-G05** – Pity meter: theo dõi pity theo banner hoặc theo nhóm banner (theo cấu hình).
- **FR-G06** – Exchange: đổi pity points theo rule.
- **FR-G07** – Duplicate handling: xử lý trùng lặp theo rule (convert shard/point/...).
- **FR-G08** – Kết quả gacha được ghi vào DB và có thể truy vấn lịch sử (tối thiểu ở mức thiết kế).

5.3.8 Nhóm H: Personal Island & NPC Collection

Personal island

- **FR-H01** – Mỗi GAME_PROFILE sở hữu 1 đảo cá nhân (1:1).
- **FR-H02** – Quản lý tài nguyên đảo: wood/stone/food/iron (hoặc tập tương đương theo cấu hình).
- **FR-H03** – Xây công trình theo grid: kiểm tra vị trí hợp lệ, không chồng lấn.
- **FR-H04** – Construction timer: công trình có trạng thái và `finish_time`.
- **FR-H05** – Nâng cấp công trình: trừ tài nguyên và cập nhật level + timer.
- **FR-H06** – Harvest: thu tài nguyên theo chu kỳ/điều kiện.

NPC collection và worker assignment

- **FR-H07** – Mỗi GAME_PROFILE sở hữu 1 bộ sưu tập NPC (1:1).
- **FR-H08** – NPC có rarity/level/star/friendship/current_job và có thể tái sử dụng cấu trúc stats/resources giống nhân vật.
- **FR-H09** – Assign worker: gán NPC vào công trình; cập nhật liên kết hai chiều (`building.assigned_worker` và `npc.current_job`).
- **FR-H10** – NPC hoạt động tạo tài nguyên ở mức đơn giản (định hướng prototype), có thể mở rộng hành vi theo job.

5.3.9 Nhóm I: Administration

- **FR-I01** – Quản lý user: xem thông tin, cập nhật trạng thái (active/banned/...).
- **FR-I02** – Ban/Unban: admin có thể ban/unban; thu hồi session theo chính sách.
- **FR-I03** – System mail: gửi mail hệ thống theo user/profile/nhóm.
- **FR-I04** – Log viewer: xem log theo thời gian/severity/actor.
- **FR-I05** – Event config: tạo/sửa cấu hình sự kiện theo thời gian và phần thưởng (lưu DB).

5.4 Yêu cầu phi chức năng

5.4.1 Bảo mật và an toàn dữ liệu

- **NFR-S01** – Mật khẩu phải lưu dạng hash; không lưu plaintext.
- **NFR-S02** – Token/session có thời hạn; hỗ trợ revoke khi ban user hoặc logout.
- **NFR-S03** – Phân quyền admin theo role; mọi thao tác admin cần audit log.
- **NFR-S04** – Các thao tác reward/currency/progression phải chống cấp trùng (idempotent ở mức thiết kế).

5.4.2 Hiệu năng và tính phản hồi

- **NFR-P01** – Các API không real-time (login/profile/inventory) phải có thời gian phản hồi ổn định.
- **NFR-P02** – Kênh real-time cần hỗ trợ cập nhật trạng thái thường xuyên (snapshot/event) để client nội suy mượt.
- **NFR-P03** – Giảm truy vấn DB cho dữ liệu nóng bằng cache hoặc nạp cấu hình vào bộ nhớ (Config Manager).

5.4.3 Tính mở rộng và bảo trì

- **NFR-M01** – Hệ thống module hóa theo phân hệ, hạn chế phụ thuộc chéo.
- **NFR-M02** – Dữ liệu cấu hình tách khỏi code và hỗ trợ versioning.
- **NFR-M03** – Có khả năng mở rộng theo chiều ngang ở mức kiến trúc (nhiều instance server).

5.4.4 Khả dụng và quan sát hệ thống

- **NFR-O01** – Log request và log nghiệp vụ cho các giao dịch quan trọng (reward, gacha, admin actions).
- **NFR-O02** – Có khả năng truy vết sự cố theo user/profile/character và theo mốc thời gian.

5.4.5 Yêu cầu UX/UI ở mức hệ thống

- **NFR-U01** – UI phải hiển thị rõ trạng thái: loading/empty/error/success/disabled.
- **NFR-U02** – Với phong cách pixel art, UI cần ưu tiên readability: khung panel rõ, icon theo lưới pixel, tương phản đủ.
- **NFR-U03** – Combat HUD tối thiểu: HP/MP (hoặc tài nguyên tương đương), skill slots, cooldown, buff/debuff.

5.5 Yêu cầu dữ liệu

5.5.1 Phân loại dữ liệu

Dữ liệu tiền trình người chơi (dynamic) Bao gồm: user/profile/session, character, inventory, island state, npc collection, pity state, history logs.

Dữ liệu cấu hình game (static) Bao gồm: định nghĩa item/building/skill, bảng hệ số chỉ số, drop tables, banner rates, pity rules, event configs.

5.5.2 Yêu cầu mô hình dữ liệu mức logic

- **DR-01** – USER có settings nhúng (1:1).
- **DR-02** – USER có nhiều GAME_PROFILE (1:N).
- **DR-03** – GAME_PROFILE sở hữu nhiều CHARACTER (1:N).
- **DR-04** – CHARACTER có INVENTORY (1:1).
- **DR-05** – GAME_PROFILE có PERSONAL_ISLAND (1:1) và NPC_COLLECTION (1:1).
- **DR-06** – Item instance tham chiếu item config (item_ref_id → CONFIG_ITEM.id).
- **DR-07** – Building instance tham chiếu building config (type/id → CONFIG_BUILDING.id).

5.5.3 Yêu cầu pipeline quản trị game configuration

- **DR-08** – Nguồn nhập liệu cấu hình từ Google Sheets theo template thống nhất.
- **DR-09** – Export TSV và import vào DB theo cơ chế upsert theo khoá id.
- **DR-10** – Có *config version* để truy vết lần import và đảm bảo nhất quán runtime.
- **DR-11** – Config Manager nạp cấu hình từ DB khi khởi động và cung cấp API truy xuất theo id.

5.6 Tiêu chí chấp nhận ở mức thiết kế

Một bản thiết kế đáp ứng yêu cầu giai đoạn 1 cần thoả:

- Bao phủ đầy đủ các phân hệ đã liệt kê ở 5.3.
- Mô hình dữ liệu mức logic thể hiện rõ quan hệ user/profile/character/inventory/island/npc và mối liên kết với config.
- Có mô tả rõ về ranh giới client-server, đặc biệt với các thao tác reward/currency/gacha và luồng real-time combat.
- Pipeline cấu hình chỉ áp dụng cho game configuration và có cơ chế versioning + log import.

6 Phân tích hệ thống

Chương này trình bày phân tích hệ thống cho trò chơi *Fortress of the Fallen* ở mức logic, nhằm chuyển hoá yêu cầu (Chương 5) thành các mô tả vận hành có thể thiết kế và triển khai: tác nhân tham gia, các ca sử dụng theo từng phân hệ, luồng nghiệp vụ trọng yếu, dữ liệu phát sinh và các ràng buộc quan trọng. Kết quả phân tích là cơ sở trực tiếp cho thiết kế kiến trúc, thiết kế dữ liệu và thiết kế giao tiếp ở Chương 7.

6.1 Tổng quan phân hệ và ranh giới trách nhiệm

Hệ thống được tổ chức theo mô hình client-server, trong đó:

- **Client** chịu trách nhiệm hiển thị, UI/UX, thu thập input và gửi yêu cầu/ý định hành động.
- **Server** chịu trách nhiệm xác thực, quản lý phiên, kiểm tra luật nghiệp vụ và xác nhận các thay đổi ảnh hưởng tiến trình (reward/currency/progression/gacha/island).

Các phân hệ nghiệp vụ chính của game được phân tích trong chương này gồm:

- **Identity & Profile:** tài khoản, thiết lập người dùng, game profile và phiên hoạt động.
- **Character & Progression:** nhân vật, level/EXP, phân bổ chỉ số, mở slot kỹ năng, hệ race/class và điều kiện mở khoá.
- **Inventory & Economy:** quản lý inventory, currency, trang bị, nâng cấp, tham chiếu tới cấu hình item.
- **Combat & Gameplay Modes:** tháp trung tâm, dungeon theo phiên (instance), arena PvP; vòng lặp chiến đấu thời gian thực.
- **Recruitment (Gacha):** banner, tỉ lệ, pity, xử lý trùng lặp và lịch sử.
- **Personal Island & NPC:** đảo cá nhân, công trình, timer xây dựng/nâng cấp, tài nguyên, bộ sưu tập NPC và phân công worker.
- **Administration:** quản trị user, trạng thái, log/audit và cấu hình sự kiện.

Ngoài ra, hệ thống có lớp **game configuration** (dữ liệu tĩnh) phục vụ định nghĩa item/building/skill/rate được quản lý theo pipeline riêng và được nạp vào *Config Manager* để truy xuất nhất quán trong runtime.

6.2 Tác nhân của hệ thống (Actors)

Các tác nhân tương tác với hệ thống được xác định như sau.

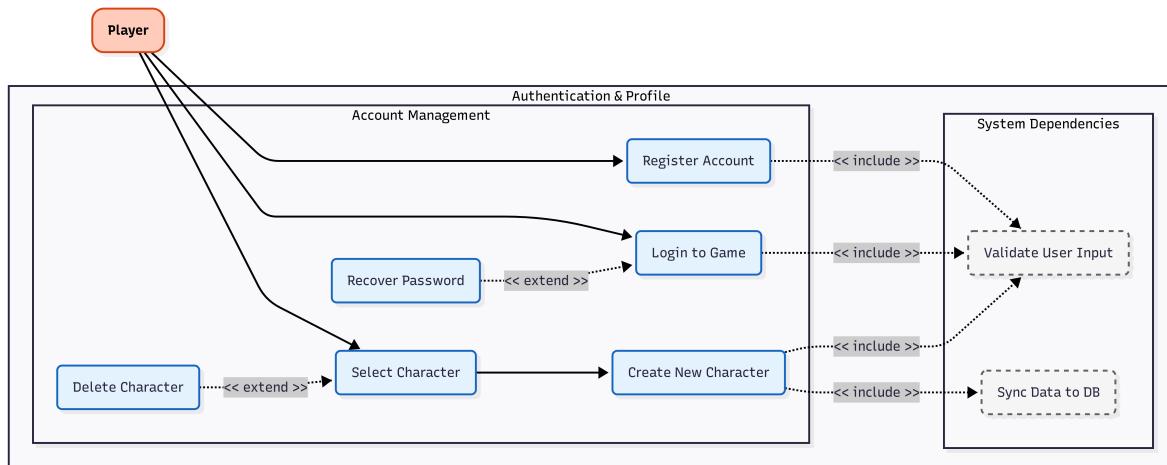
Tác nhân	Mô tả
Player	Người chơi sử dụng game client để đăng nhập, chọn profile/nhân vật, tham gia nội dung chiến đấu, quản lý tiến trình và tài sản.
Admin	Người quản trị hệ thống; có quyền thao tác trên user, log, mail hệ thống và cấu hình sự kiện.
Content Designer	Người thiết kế nội dung/thông số; quản lý bảng cấu hình (items/buildings/skills/rates) và xuất dữ liệu theo pipeline cấu hình.
Game Client	Ứng dụng chạy trên máy người chơi; hiển thị, gửi request HTTP, duy trì kênh real-time và hiển thị trạng thái.
Backend Server	Thành phần xử lý xác thực, nghiệp vụ và đồng bộ; ghi dữ liệu vào DB và cung cấp giao tiếp cho client/admin tools.
Database	Lưu trữ bền vững: dữ liệu tiến trình người chơi và các collections cấu hình.

Bảng 5: Danh sách tác nhân và vai trò

6.3 Phân tích Use Case theo phân hệ

Phần này trình bày use case theo từng phân hệ nghiệp vụ. Mỗi nhóm use case đi kèm sơ đồ tổng quan và các use case trọng yếu sẽ được đặc tả chi tiết ở Mục 6.4.

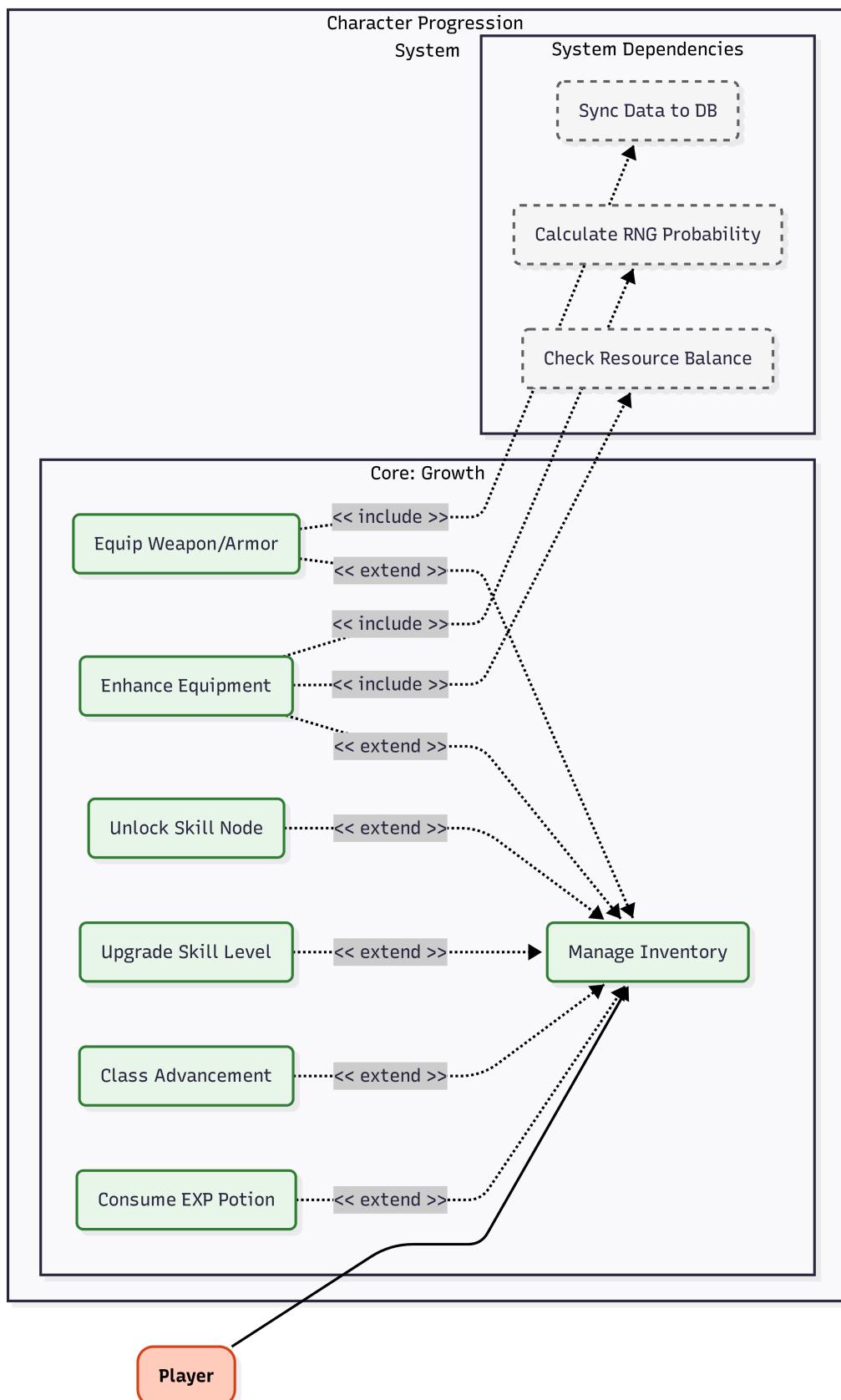
6.3.1 Identity & Profile



Hình 3: Use case tổng quan cho phân hệ Identity & Profile

Nhóm này bao gồm đăng ký/đăng nhập, quản lý thiết lập, tạo/chọn game profile và thiết lập session runtime để chuẩn bị vào phiên chơi.

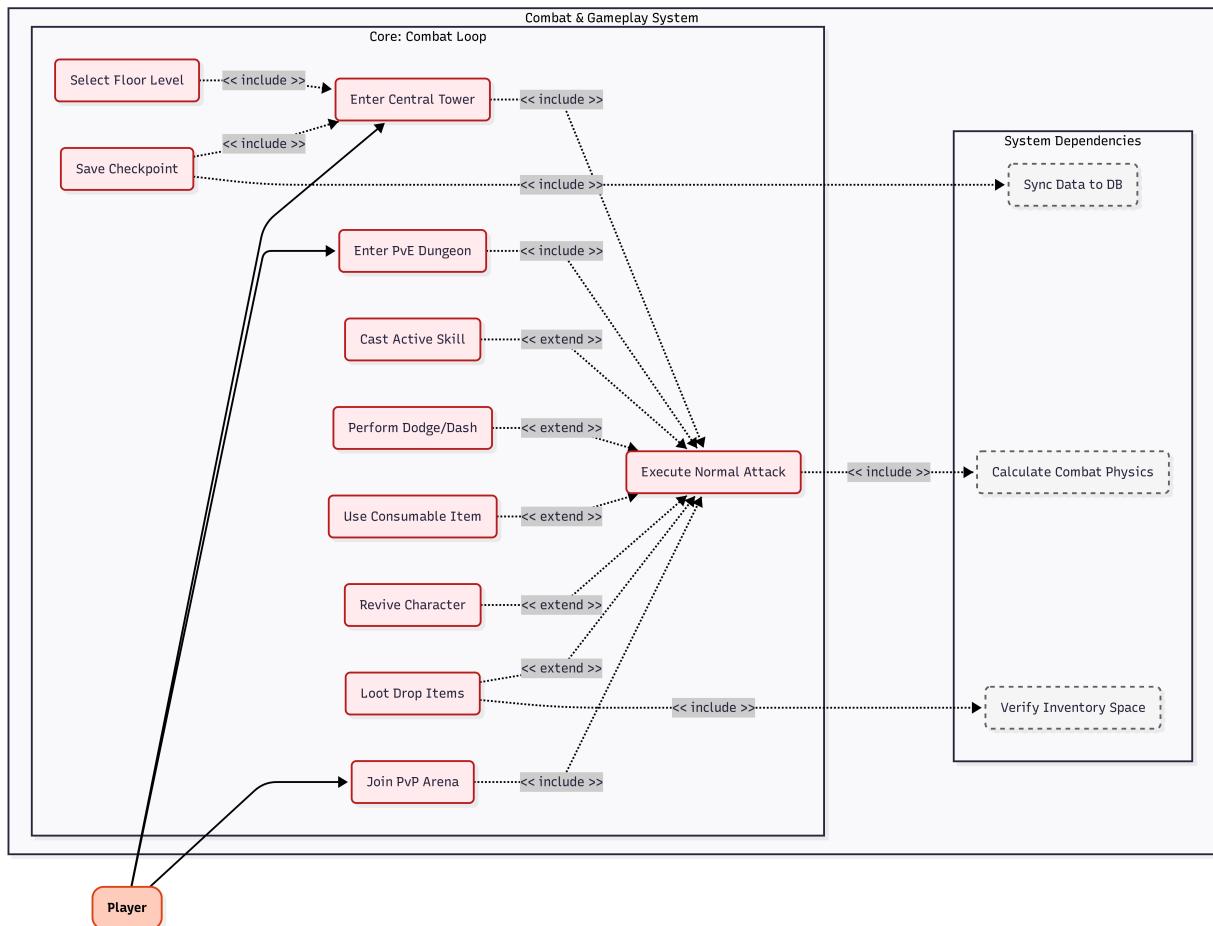
6.3.2 Character & Progression



Hình 4: Use case tổng quan cho phân hệ Character & Progression

Nhóm này tập trung vào tạo/chọn nhân vật, level/EXP, phân bổ chỉ số, quản lý kỹ năng theo slot, combo skill và mở khoá/chuyển class theo điều kiện.

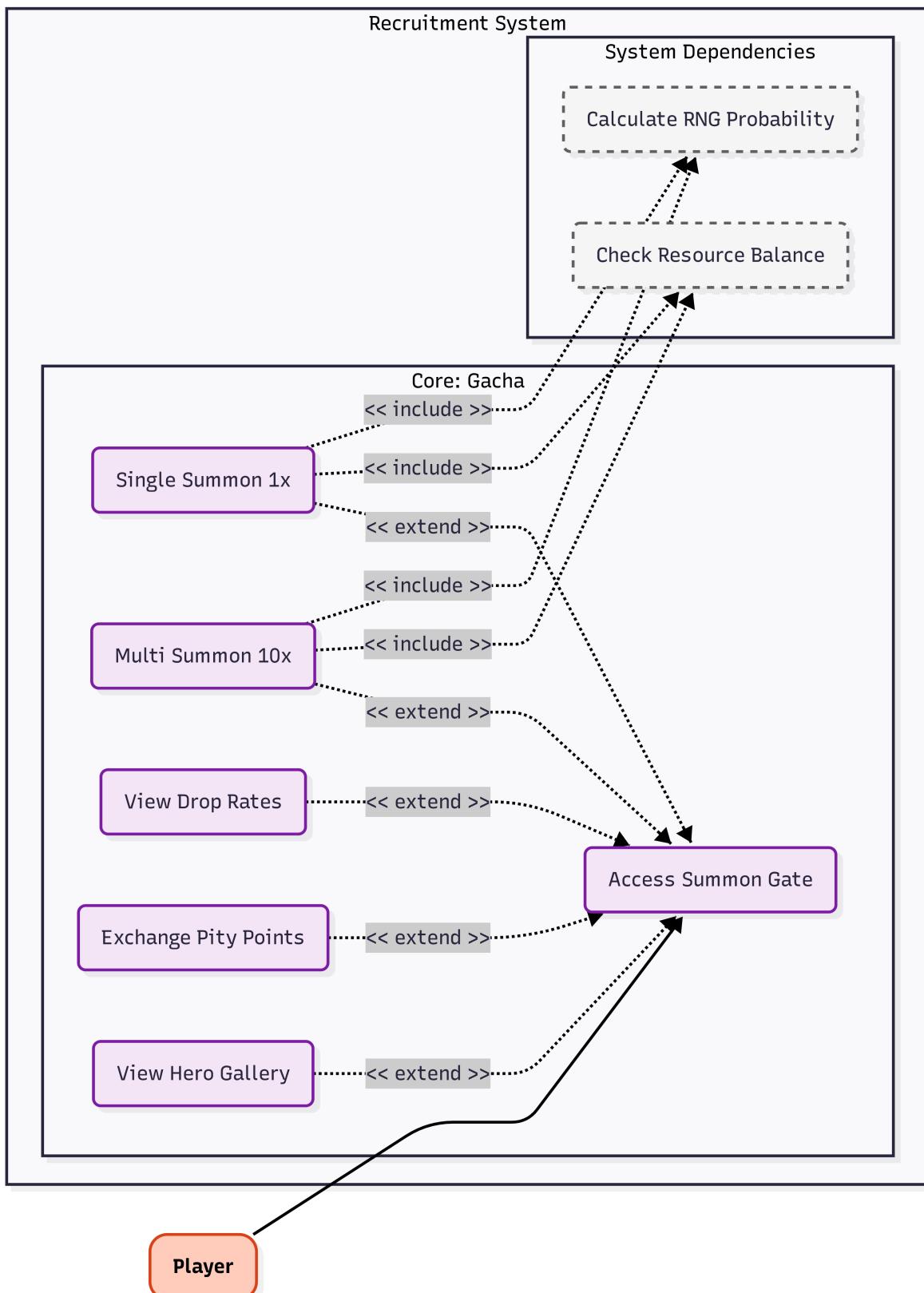
6.3.3 Combat & Gameplay Modes



Hình 5: Use case tổng quan cho phân hệ Combat & Gameplay Modes

Nhóm này mô tả hành trình tham gia nội dung (tower/dungeon/arena), vòng lặp chiến đấu real-time, xử lý kết phiên và ghi nhận reward/checkpoint.

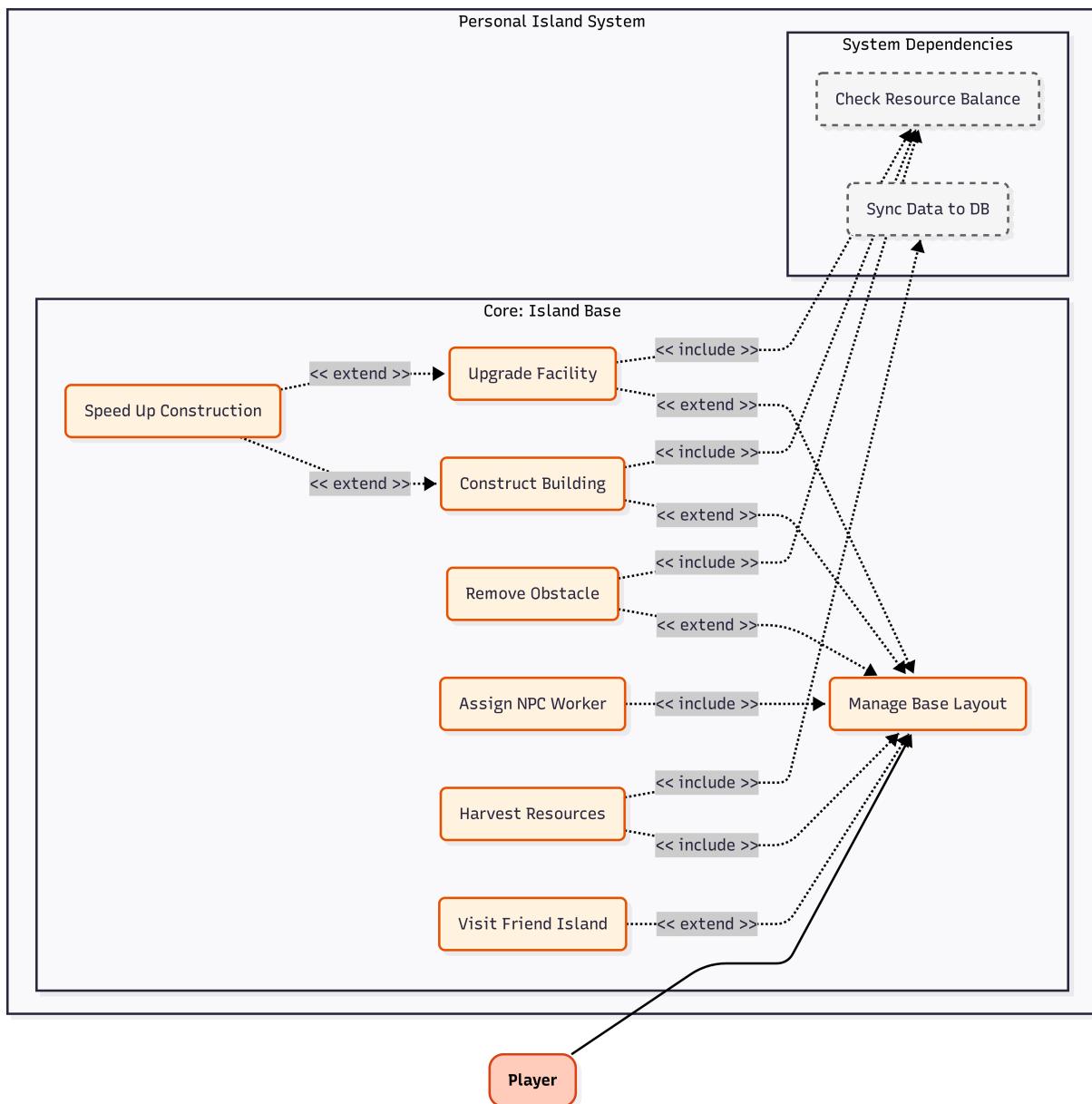
6.3.4 Recruitment (Gacha)



Hình 6: Use case tổng quan cho phân hệ Recruitment (Gacha)

Nhóm này mô tả quy trình xem banner/tỉ lệ, thực hiện summon, cập nhật pity, xử lý trùng lắp và ghi lịch sử.

6.3.5 Personal Island & NPC



Hình 7: Use case tổng quan cho phân hệ Personal Island & NPC

Nhóm này mô tả đảo cá nhân theo mô hình base-building: xây/nâng cấp công trình theo timer, quản lý tài nguyên, và phân công NPC làm worker.

6.3.6 Administration

Hình 8: Use case tổng quan cho phân hệ Administration

Nhóm này bao gồm các tác vụ quản trị: quản lý trạng thái user, ban/unban, gửi mail hệ thống, xem log/audit và cấu hình sự kiện.

6.4 Đặc tả Use Case trọng yếu

Phần này đặc tả các use case có ảnh hưởng trực tiếp tới tiến trình và tính nhất quán dữ liệu. Mỗi use case gồm: mục tiêu, tác nhân, tiền điều kiện, hậu điều kiện, luồng chính và luồng ngoại lệ.

6.4.1 UC-A01 – Đăng ký tài khoản

Mục tiêu: Player tạo tài khoản mới để sử dụng hệ thống.

Tác nhân chính: Player.

Tiền điều kiện: Player chưa đăng nhập; client có thể kết nối API.

Hậu điều kiện: USER được tạo trong DB; trạng thái tài khoản hợp lệ để đăng nhập.

Luồng chính:

1. Player nhập username/email và mật khẩu.
2. Client gửi request đăng ký.
3. Server kiểm tra định dạng, trùng lặp và chính sách mật khẩu.
4. Server băm mật khẩu và tạo USER (kèm settings mặc định).
5. Server trả kết quả thành công.

Ngoại lệ:

- Email/username trùng → trả lỗi trùng lặp.
- Dữ liệu không hợp lệ → trả lỗi theo trường.

6.4.2 UC-A02 – Đăng nhập và tạo phiên hoạt động

Mục tiêu: Xác thực Player, cấp token và thiết lập SESSION_RUNTIME để duy trì phiên.

Tác nhân chính: Player.

Tiền điều kiện: USER tồn tại và không bị khoá.

Hậu điều kiện: Token hợp lệ được cấp; server ghi nhận last_login; phiên runtime có thể tham chiếu profile/character khi Player vào game.

Luồng chính:

1. Player nhập thông tin đăng nhập.
2. Client gửi request login.
3. Server xác thực thông tin, kiểm tra trạng thái tài khoản.

4. Server cấp token (có thời hạn) và tạo hoặc cập nhật session runtime (token, socket_id nếu có, expires_at).
5. Client lưu token và điều hướng sang bước chọn profile.

Ngoại lệ:

- Sai thông tin → trả lỗi xác thực.
- Token hết hạn/không hợp lệ ở các request sau → yêu cầu login lại.

6.4.3 UC-A03 – Tạo và chọn Game Profile

Mục tiêu: Player tạo/chọn GAME_PROFILE để tách tiến trình theo nhiều profile (server/slot) trong cùng user.

Tác nhân chính: Player.

Tiền điều kiện: Đã đăng nhập (UC-A02).

Hậu điều kiện: GAME_PROFILE được tạo hoặc được chọn; session runtime cập nhật profile_id.

Luồng chính:

1. Client hiển thị danh sách profile hiện có.
2. Player tạo profile mới hoặc chọn profile.
3. Server ghi last_played khi profile được chọn.
4. Server cập nhật session runtime với profile_id.

Ngoại lệ:

- Vượt giới hạn số profile (nếu áp dụng) → trả lỗi giới hạn.

6.4.4 UC-P01 – Tạo nhân vật và khởi tạo dữ liệu nền

Mục tiêu: Player tạo CHARACTER thuộc GAME_PROFILE, kèm dữ liệu nền: level, base stats, resources, position và appearance.

Tác nhân chính: Player.

Tiền điều kiện: Đã chọn profile (UC-A03); còn slot nhân vật (theo chính sách).

Hậu điều kiện: CHARACTER và INVENTORY 1:1 được tạo; có thể chọn nhân vật để vào game.

Luồng chính:

1. Player chọn race, class khởi đầu, đặt tên và chọn appearance.
2. Client gửi request tạo nhân vật.

3. Server kiểm tra hợp lệ (tên, ràng buộc race/class).
4. Server tạo CHARACTER (level=1, exp=0), gán base stats mặc định, resources mặc định và vị trí khởi tạo.
5. Server tạo INVENTORY rỗng (max_slots mặc định, currency mặc định).
6. Server trả dữ liệu nhân vật để client hiển thị.

Ngoại lệ:

- Tên không hợp lệ/trùng → trả lỗi và yêu cầu nhập lại.
- Race/Class không hợp lệ → trả lỗi ràng buộc.

6.4.5 UC-P02 – Lên level và phân bổ điểm chỉ số

Mục tiêu: Khi nhận đủ EXP, nhân vật lên level và nhận điểm phân bổ chỉ số theo quy tắc.

Tác nhân chính: Player (qua hành động gameplay).

Tiền điều kiện: CHARACTER tồn tại; có sự kiện nhận EXP.

Hậu điều kiện: Level tăng (tối đa 100); cộng thêm **+5 stat points** mỗi level; mở slot kỹ năng theo milestone khi đạt mốc.

Luồng chính:

1. Sau khi hoàn thành nội dung, server tính EXP và cộng vào `current_exp`.
2. Nếu vượt ngưỡng, server tăng level và cộng `stat_points_available` (+5).
3. Nếu level đạt mốc mở slot, server cập nhật trạng thái slot (normal/combo).
4. Client hiển thị thông báo lên level và cho phép phân bổ điểm.

Ngoại lệ:

- Đạt level 100 → không tăng level, chỉ cộng EXP theo chính sách hoặc khoá EXP.

6.4.6 UC-P03 – Quản lý kỹ năng theo slot và tạo Combo Skill

Mục tiêu: Player trang bị kỹ năng vào slot; combo skill chỉ hợp lệ khi có slot combo và thỏa quy tắc kết hợp.

Tác nhân chính: Player.

Tiền điều kiện: Player sở hữu kỹ năng đã học; slot tương ứng đã được mở theo level.

Hậu điều kiện: Bộ kỹ năng trang bị được cập nhật; combo skill (nếu có) được lưu như một cấu hình trang bị hợp lệ.

Luồng chính:

1. Player mở màn hình kỹ năng và xem danh sách kỹ năng đã học.
2. Player kéo/thả hoặc chọn kỹ năng để gán vào normal slots.
3. Player chọn các kỹ năng thành phần để tạo combo, sau đó gán vào combo slot.
4. Server kiểm tra điều kiện (slot mở, điều kiện class/race/level, quy tắc combo).
5. Server lưu trạng thái trang bị và trả kết quả.

Ngoại lệ:

- Chưa mở combo slot → từ chối thao tác gán combo.
- Combo không hợp lệ → trả lỗi quy tắc.

6.4.7 UC-P04 – Mở khoá Class theo điều kiện và chuyển Class

Mục tiêu: Player mở khoá class mới dựa trên ngưỡng chỉ số/trait/achievement và có thể chuyển class theo chính sách.

Tác nhân chính: Player.

Tiền điều kiện: CHARACTER tồn tại; có dữ liệu chỉ số và trạng thái trait/achievement.

Hậu điều kiện: Class mới được đánh dấu đã mở; nếu chuyển class, server cập nhật class hiện tại và các ràng buộc liên quan (kỹ năng class, loadout).

Luồng chính:

1. Player xem danh sách class và điều kiện mở.
2. Player thực hiện hành động “Unlock” khi tin rằng đã đủ điều kiện.
3. Server kiểm tra điều kiện (stat threshold, karma/affinity/luck/resistance, điều kiện tiến trình).
4. Nếu đạt, server ghi nhận class đã mở và trả kết quả.
5. (Tuỳ chính sách) Player chọn “Switch class”; server cập nhật class hiện hành và hiệu lực kỹ năng liên quan.

Ngoại lệ:

- Không đủ điều kiện → trả lỗi và hiển thị phần thiêu.
- Class yêu cầu race đặc thù → từ chối nếu race không khớp.

6.4.8 UC-I01 – Nhận reward và cập nhật Inventory

Mục tiêu: Ghi nhận item/currency sau khi hoàn thành nội dung; đảm bảo quy tắc stack và dung lượng inventory.

Tác nhân chính: Player (kết quả từ gameplay).

Tiền điều kiện: Nội dung kết thúc và có reward; INVENTORY tồn tại.

Hậu điều kiện: INVENTORY được cập nhật nhất quán; item instance tham chiếu đúng item config.

Luồng chính:

1. Server xác định reward (item, gold, gem, ...).
2. Với item: server áp dụng quy tắc stack theo CONFIG_ITEM.max_stack.
3. Nếu thiếu slot, server xử lý theo chính sách (gửi mailbox, trả về kho tạm, hoặc từ chối nhận).
4. Server ghi cập nhật inventory và trả danh sách thay đổi cho client.

Ngoại lệ:

- Inventory đầy → kích hoạt chính sách overflow (tuỳ thiết kế).

6.4.9 UC-C01 – Tham gia nội dung theo phiên (Tower/Dungeon/Arena)

Mục tiêu: Player chọn nội dung, tham gia instance/room và bắt đầu vòng lặp combat real-time.

Tác nhân chính: Player.

Tiền điều kiện: Đã chọn nhân vật; kết nối real-time sẵn sàng.

Hậu điều kiện: Player được gán vào instance; trạng thái phiên được ghi vào session runtime; client nhận dữ liệu khởi tạo.

Luồng chính:

1. Player chọn chế độ (tower/dungeon/arena) và tham số (tầng, độ khó, ...).
2. Client gửi request/join event.
3. Server kiểm tra điều kiện tham gia (level, tài nguyên, vé, ...).
4. Server tạo hoặc gán vào instance, gửi thông tin spawn/room state.
5. Client load scene và hiển thị HUD chiến đấu.

Ngoại lệ:

- Không đủ điều kiện → trả lỗi điều kiện và không tạo instance.

6.4.10 UC-C02 – Vòng lặp chiến đấu real-time và đồng bộ trạng thái

Mục tiêu: Xử lý hành động combat và đồng bộ trạng thái giữa client và server trong phiên chơi.

Tác nhân chính: Player.

Tiền điều kiện: Đã ở trong instance (UC-C01) và có kênh real-time.

Hậu điều kiện: Trạng thái authoritative được cập nhật; client hiển thị nhất quán theo snapshot/event.

Luồng chính:

1. Client gửi input/intent (move, attack, cast skill, dodge) theo nhịp.
2. Server xác thực input theo luật (cooldown, tài nguyên, vị trí hợp lệ).
3. Server cập nhật trạng thái (HP/MP/buff/debuff/position) và phát snapshot/event.
4. Client nội suy/trình diễn (animation, VFX) theo trạng thái nhận được.

Ngoại lệ:

- Mất kết nối → server đóng phiên hoặc cho phép reconnect trong thời gian cho phép.

6.4.11 UC-C03 – Kết phiên, checkpoint và ghi nhận tiến trình

Mục tiêu: Sau khi kết thúc phiên, hệ thống ghi nhận kết quả: checkpoint, EXP, reward, log nghiệp vụ.

Tác nhân chính: Player.

Tiền điều kiện: Phiên kết thúc (thắng/thua/thoát).

Hậu điều kiện: Dữ liệu tiến trình được cập nhật bền vững; reward được đưa vào inventory; checkpoint được ghi theo chính sách.

Luồng chính:

1. Server xác định trạng thái kết phiên và bảng phần thưởng.
2. Server cập nhật EXP/level (UC-P02), cập nhật inventory (UC-I01).
3. Server cập nhật checkpoint (tower/dungeon) nếu đạt điều kiện.
4. Server ghi log nghiệp vụ cho các giao dịch quan trọng.
5. Client hiển thị màn hình kết quả và phần thưởng.

6.4.12 UC-G01 – Gacha: Summon, pity và xử lý trùng lặp

Mục tiêu: Player thực hiện summon theo banner; hệ thống trả kết quả theo tỉ lệ, cập nhật pity và xử lý duplicate.

Tác nhân chính: Player.

Tiền điều kiện: Banner đang hoạt động; Player có đủ tài nguyên summon.

Hậu điều kiện: Kết quả summon được ghi nhận; NPC/đơn vị mới thêm vào collection hoặc chuyển đổi theo rule; pity cập nhật nhất quán.

Luồng chính:

1. Player chọn banner và hình thức summon (1x/10x).
2. Server kiểm tra tài nguyên và trừ chi phí.
3. Server thực hiện RNG theo rate của banner; áp dụng pity nếu đạt ngưỡng.
4. Với kết quả trùng: server chuyển đổi sang shard/point theo rule.
5. Server ghi lịch sử gacha và trả danh sách kết quả cho client.

Ngoại lệ:

- Không đủ tài nguyên → từ chối summon.
- Banner hết hạn → yêu cầu refresh danh sách banner.

6.4.13 UC-S01 – Xây/Nâng cấp công trình trên đảo cá nhân

Mục tiêu: Player xây hoặc nâng cấp công trình; hệ thống trừ tài nguyên và ghi timer.

Tác nhân chính: Player.

Tiền điều kiện: PERSONAL_ISLAND tồn tại; có đủ tài nguyên; building config hợp lệ.

Hậu điều kiện: BUILDING instance được tạo/cập nhật trạng thái; `finish_time` được thiết lập; tài nguyên đảo được cập nhật.

Luồng chính:

1. Player chọn loại công trình và vị trí grid.
2. Client gửi request build/upgrade.
3. Server kiểm tra vị trí hợp lệ, không chồng lấn, thoả điều kiện level/town hall.
4. Server trừ tài nguyên theo `CONFIG_BUILDING.cost` và đặt `finish_time`.
5. Client hiển thị trạng thái *Constructing/Upgrading*.

6.4.14 UC-S02 – Gán NPC làm worker cho công trình

Mục tiêu: Player phân công NPC vào công trình để vận hành/đẩy tiến trình theo job.

Tác nhân chính: Player.

Tiền điều kiện: NPC_COLLECTION tồn tại; NPC rảnh; công trình cho phép gán worker.

Hậu điều kiện: Building lưu assigned_worker_id; NPC cập nhật current_job.

Luồng chính:

1. Player chọn công trình và chọn NPC.
2. Server kiểm tra ràng buộc (NPC rảnh, công trình chưa có worker hoặc cho phép thay thế).
3. Server cập nhật liên kết hai chiều giữa building và NPC.
4. Client hiển thị NPC đang làm việc tại công trình.

6.4.15 UC-AD01 – Quản trị user: ban/unban và audit log

Mục tiêu: Admin quản lý trạng thái user, ban/unban và đảm bảo có audit log.

Tác nhân chính: Admin.

Tiền điều kiện: Admin đã xác thực và có quyền phù hợp.

Hậu điều kiện: Trạng thái user cập nhật; phiên liên quan có thể bị thu hồi theo chính sách; hành động được ghi log.

Luồng chính:

1. Admin tìm kiếm user theo username/email/id.
2. Admin thực hiện ban/unban hoặc thay đổi trạng thái.
3. Server cập nhật USER.status và ghi audit log (actor, thời gian, hành động).
4. (Tuỳ chính sách) Server thu hồi session runtime/token hiện hành.

6.5 Quy tắc nghiệp vụ và ràng buộc hệ thống

6.5.1 Quy tắc Level và mở slot kỹ năng

Theo đặc tả hệ thống level:

- Level tối đa: **100**.
- Mỗi level nhận **+5 stat points**.
- Level không tự động cấp kỹ năng; level chủ yếu mở slot để trang bị kỹ năng đã học.

Mốc level	Phân thưởng tiến trình (slot)
10 / 20 / 30	Mở lần lượt Normal Skill Slot 1–3
40	Mở Combo Skill Slot 1
50	Mốc nội dung (boss loop) và điều chỉnh giới hạn stat cap theo thiết kế
60 / 80 / 100	Mở lần lượt Combo Skill Slot 2–4; tại level 100 mở <i>Ultimate Combo</i>
70 / 90	Mở lần lượt Normal Skill Slot 4–5

Bảng 6: Mốc level và cơ chế mở slot kỹ năng

6.5.2 Quy tắc Race, Stat Cap và Trait bẩm sinh

Race quyết định:

- **Stat cap** theo từng thuộc tính.
- **Trait bẩm sinh** (ví dụ tăng kháng, tăng hiệu quả chế tạo, hoặc điều kiện kích hoạt theo HP).

Ràng buộc quan trọng là điểm phân bổ chỉ số không được vượt stat cap theo race, và một số class đặc thù yêu cầu race tương ứng.

6.5.3 Quy tắc mở khoá Class

Class được phân tầng (Basic/Intermediate/Advanced/Legendary/Hidden). Điều kiện mở khoá có thể kết hợp:

- Nguồn chỉ số (STR/DEX/CON/INT/WIS/CHA).
- Trait đặc biệt (Karma/Affinity/Luck/Resistance, ...).
- Điều kiện hành vi/tiến trình (ví dụ thắng PvP, tham gia arena, số NPC chỉ huy).
- Điều kiện race cho các class độc quyền.

Việc mở khoá cần đảm bảo kiểm tra trên server để tránh sai lệch do client.

6.5.4 Quy tắc Inventory và tham chiếu cấu hình

- Inventory có giới hạn slot; item áp dụng quy tắc stack theo cấu hình.
- Item trong inventory là **instance** (có `unique_uid`, `enhancement_level`, `durability`) và **tham chiếu** item definition bằng `item_ref_id`.

6.5.5 Quy tắc Island và xây dựng công trình

- Đảo cá nhân quản lý tài nguyên và danh sách công trình theo grid.
- Xây/nâng cấp công trình tạo timer (`finish_time`) và trạng thái (building status).
- Worker assignment cần đồng bộ hai chiều giữa công trình và NPC để tránh trạng thái treo.

6.5.6 Quy tắc Gacha: rate, pity và duplicate

- Tỉ lệ theo banner là dữ liệu cấu hình; cần minh bạch hiển thị cho Player.
- Pity là trạng thái tiến trình, cần lưu bền vững theo banner hoặc theo nhóm banner.
- Kết quả trùng lặp được chuyển đổi theo rule (shard/point/...) để tránh “mất giá trị”.

6.6 Phân tích dữ liệu ở mức logic

6.6.1 Phân lớp dữ liệu: động và tĩnh

Dữ liệu trong hệ thống được phân thành hai nhóm:

- **Dữ liệu tiến trình (dynamic):** USER, GAME_PROFILE, SESSION_RUNTIME, CHARACTER, INVENTORY, PERSONAL_ISLAND, NPC_COLLECTION, pity state, lịch sử gacha, log nghiệp vụ.
- **Dữ liệu cấu hình (static):** CONFIG_ITEM, CONFIG_BUILDING và các bảng cấu hình mở rộng (skills, rates, drop tables, ...).

6.6.2 Các thực thể lõi theo phân hệ

- **Identity:** USER (kèm OBJECT_USER_SETTINGS), GAME_PROFILE, SESSION_RUNTIME.
- **Character:** CHARACTER (base_stats/resources/position/appearance dạng nhúng).
- **Inventory:** INVENTORY và danh sách STRUCT_INVENTORY_ITEM tham chiếu CONFIG_ITEM.
- **Island:** PERSONAL_ISLAND (resources nhúng) và STRUCT_BUILDING tham chiếu CONFIG_BUILDING.
- **NPC:** NPC_COLLECTION và STRUCT_OWNED_NPC (tái sử dụng struct stats/resources).

6.6.3 Quan hệ dữ liệu quan trọng

Các quan hệ dữ liệu có ảnh hưởng trực tiếp tới luồng nghiệp vụ:

- USER → GAME_PROFILE là quan hệ 1:N.
- GAME_PROFILE → CHARACTER là quan hệ 1:N.

- CHARACTER → INVENTORY là quan hệ 1:1.
- GAME_PROFILE → PERSONAL_ISLAND và NPC_COLLECTION là quan hệ 1:1.
- Inventory item instance tham chiếu item config; building instance tham chiếu building config.

6.7 Tổng kết chương

Chương này đã phân tích hệ thống theo các phân hệ nghiệp vụ của *Fortress of the Fallen*: xác thực và profile, tiến trình nhân vật (level/slot/combo, race/class), inventory và kinh tế, combat theo phiên, gacha, đảo cá nhân & NPC, và phân hệ quản trị. Đồng thời, chương đã xác định các use case trọng yếu, quy tắc nghiệp vụ và cấu trúc dữ liệu logic, làm tiền đề cho chương thiết kế hệ thống (Chương 7).

7 Thiết kế hệ thống

7.1 Mục tiêu thiết kế

Mục tiêu của chương này là chuyển hoá kết quả phân tích ở Chương 6 thành các quyết định thiết kế cụ thể ở mức: (i) kiến trúc tổng thể client-server và triển khai; (ii) thiết kế mô-đun backend và ranh giới trách nhiệm; (iii) thiết kế dữ liệu (progression/state và game configuration); (iv) thiết kế giao tiếp (HTTP/WebSocket) và quy ước dữ liệu; (v) thiết kế các cơ chế đảm bảo nhất quán, bảo mật và quan sát hệ thống.

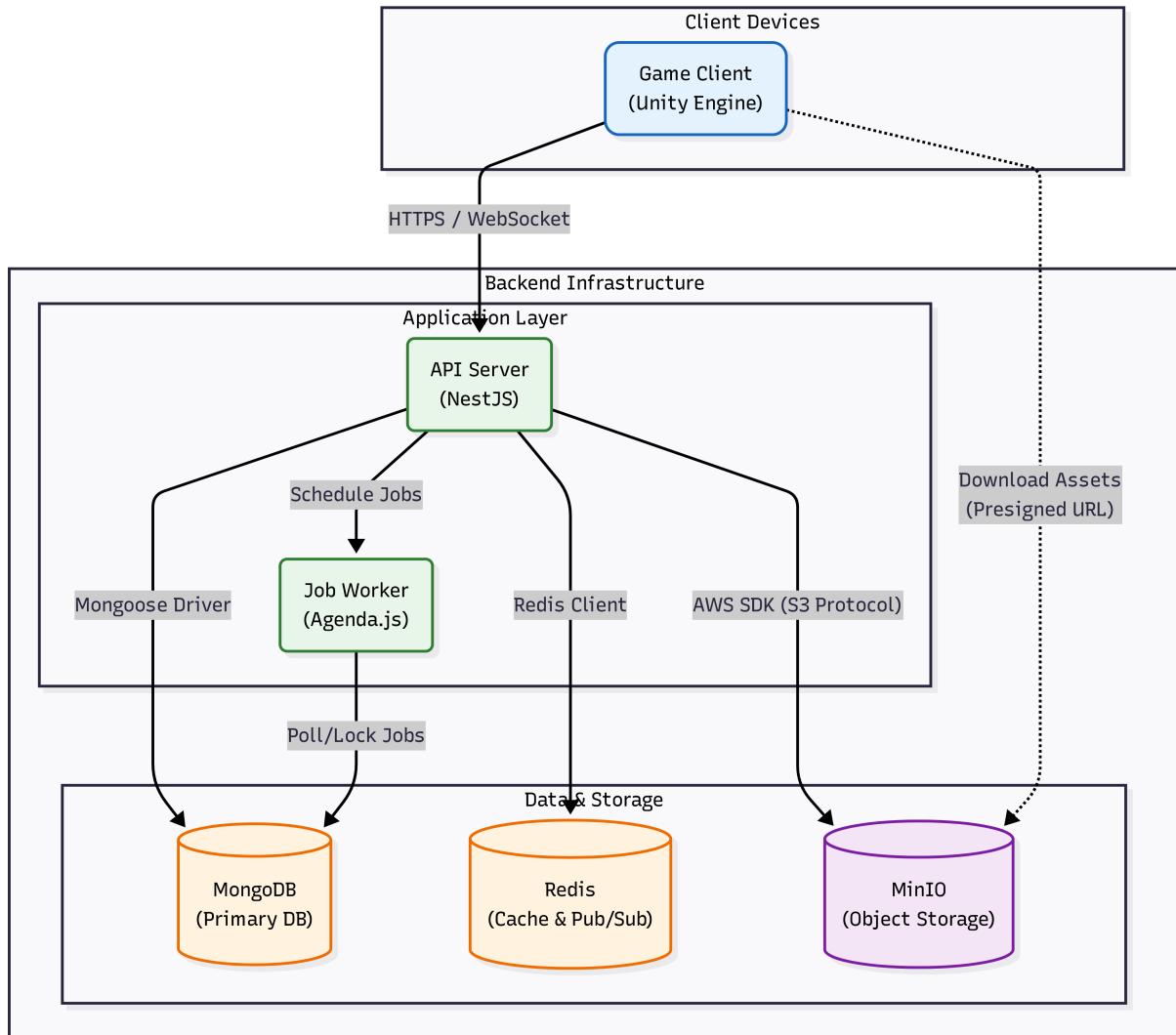
Trong giai đoạn 1, thiết kế ưu tiên:

- **Tính đúng đắn và nhất quán:** các thay đổi ảnh hưởng tiến trình (reward/currency/gacha/checkpoint) phải được server xác nhận.
- **Tính mô-đun:** phân tách rõ theo phân hệ (auth, profile, character, inventory, combat, gacha, island, admin).
- **Tính data-driven cho cấu hình:** tách dữ liệu cấu hình gameplay khỏi mã nguồn; hỗ trợ quản trị và thay đổi nhanh.
- **Phù hợp phạm vi đồ án:** triển khai ở mức hợp lý (single deployment có thể chạy được), nhưng vẫn giữ được hướng mở rộng.

7.2 Kiến trúc tổng thể và triển khai

7.2.1 Tổng quan triển khai

Hệ thống được tổ chức thành 3 lớp chính: *Client Devices*, *Backend Infrastructure* và *Data & Storage*. Hình 9 mô tả các thành phần và đường giao tiếp chính.



Hình 9: Sơ đồ triển khai tổng quan hệ thống

Các thành phần cốt lõi:

- **Game Client (Unity):** hiển thị đồ họa, UI, thu thập input; giao tiếp với server qua HTTPS và WebSocket.
- **API Server (NestJS):** lớp ứng dụng xử lý REST API, đồng thời cung cấp kênh real-time (WebSocket gateway) cho phiên chơi.
- **Job Worker:** xử lý tác vụ nền theo lịch (ví dụ: mail hệ thống, sự kiện, tổng hợp log, dọn dẹp dữ liệu runtime).
- **MongoDB (Primary DB):** lưu trữ dữ liệu tiền trinh bền vững và các collections cấu hình.
- **Redis (Cache & Pub/Sub):** lưu cache/session runtime hoặc pub-sub khi mở rộng nhiều instance.
- **MinIO (Object Storage):** lưu trữ tệp và asset bundle (nếu áp dụng); client tải qua presigned URL.

7.2.2 Nguyên tắc server-authoritative

Thiết kế áp dụng nguyên tắc **server-authoritative** cho các tác vụ ảnh hưởng tiến trình và công bằng:

- Client chỉ gửi ý định (intent), không gửi trạng thái “đã xảy ra” như “đã trúng đòn” hay “đã nhận item”.
- Server kiểm tra luật (cooldown, tài nguyên, điều kiện tham gia, dung lượng inventory) và chỉ khi hợp lệ mới ghi dữ liệu.
- Các giao dịch reward/currency/gacha/checkpoint được coi là **giao dịch nghiệp vụ** và phải có log/audit để truy vết.

Các quyết định này giúp giảm rủi ro gian lận và tăng tính nhất quán trong môi trường online [10, 2].

7.2.3 Phân tách giao tiếp: HTTP và WebSocket

HTTP/REST được dùng cho:

- đăng ký/đăng nhập, quản lý profile/nhân vật;
- tải dữ liệu UI (inventory, island state, NPC collection);
- thao tác quản trị (admin) và các tác vụ không yêu cầu độ trễ thấp.

WebSocket được dùng cho:

- tham gia instance/room và vòng lặp chiến đấu real-time;
- gửi input (move/attack/cast/dodge) theo nhịp;
- nhận snapshot/event từ server để hiển thị mượt.

7.3 Thiết kế mô-đun backend

Backend được tổ chức theo phong cách module hoá của NestJS [7]. Mỗi mô-đun tương ứng một phân hệ nghiệp vụ và sở hữu ranh giới dữ liệu rõ ràng. Bảng 7 tóm tắt các mô-đun chính.

Mô-đun	Trách nhiệm chính	Dữ liệu/Collection liên quan
Auth Module	Đăng ký/đăng nhập; cấp và xác thực token; chính sách role	USER, SESSION_RUNTIME
Profile Module	Tạo/chọn game profile; cập nhật last_played; thiết lập profile	GAME_PROFILE
Character Module	Tạo/chọn/xoá nhân vật; quản lý level/EXP; base stats, appearance	CHARACTER
Inventory Module	Quản lý slot, stack, currency; equip/enhance/durability; cập nhật reward	INVENTORY, CONFIG_ITEM
Combat Module	Tham gia tower/dungeon/arena; quản lý instance/room; vòng lặp tick	SESSION_RUNTIME, (runtime state)
Gacha Module	Banner, rate, pity, exchange; xử lý duplicate; ghi lịch sử	NPC_COLLECTION, (gacha history), config rates
Island Module	Tài nguyên đảo; xây/nâng cấp công trình; timer; layout	PERSONAL_ISLAND, CONFIG_BUILDING
NPC Module	Bộ sưu tập NPC; levelling/star/friendship; gán worker	NPC_COLLECTION
Admin Module	Ban/unban; system mail; log viewer; event config	USER, (mail/event/log)
Config Module	Nạp và cung cấp truy xuất cấu hình; versioning; cache	CONFIG_* (static)

Bảng 7: Bản đồ mô-đun backend và dữ liệu liên quan

7.3.1 Các thành phần dùng chung (cross-cutting)

Để giảm trùng lặp và tăng tính nhất quán, hệ thống dùng các thành phần dùng chung:

- **Validation:** kiểm tra input cho REST và WebSocket (schema/DTO) để chặn dữ liệu sai từ đầu.
- **Error handling:** chuẩn hoá mã lỗi và thông điệp lỗi; đảm bảo client có thể hiển thị đúng trạng thái UI (loading/error/disabled).
- **Logging & audit:** log request, log nghiệp vụ (reward/gacha/admin action), kèm correlation id.
- **Rate limit / anti-spam:** giới hạn tần suất với endpoint nhạy cảm (login, gacha) và message real-time (input spam).

7.4 Thiết kế dữ liệu

7.4.1 Nguyên tắc thiết kế dữ liệu

Hệ thống dữ liệu được chia thành 2 nhóm:

- **Progression/State (động)**: dữ liệu người chơi thay đổi theo thời gian (profile, character, inventory, island, NPC, pity).
- **Game Configuration (tĩnh)**: dữ liệu định nghĩa và thông số gameplay (item/building/skill/rates/fo

Với MongoDB, thiết kế ưu tiên:

- **Embed** khi dữ liệu có vòng đời gắn chặt với thực thể cha và truy vấn thường đi cùng nhau (ví dụ: base_stats/resources/appearance nhúng trong CHARACTER; items nhúng trong INVENTORY; buildings nhúng trong PERSONAL_ISLAND).
- **Reference** khi dữ liệu là định nghĩa tĩnh dùng chung (CONFIG_ITEM/CONFIG_BUILDING) hoặc cần tái sử dụng giữa nhiều thực thể.

7.4.2 Mô hình thực thể chính (ERD mức logic)

Trong phạm vi báo cáo, ERD được trình bày ở mức logic để làm rõ quan hệ 1:1, 1:N và các liên kết tham chiếu tới cấu hình. Mã mô tả ERD (Mermaid) có thể dùng để render thành hình khi cần:

```

1 erDiagram
2   USER ||--|{ GAME_PROFILE : "has" (1:N)
3   GAME_PROFILE ||--|{ CHARACTER : "owns" (1:N)
4   CHARACTER ||--|| INVENTORY : "has" (1:1)
5   GAME_PROFILE ||--|| PERSONAL_ISLAND : "owns" (1:1)
6   GAME_PROFILE ||--|| NPC_COLLECTION : "owns" (1:1)
7   INVENTORY ||--|{ STRUCT_INVENTORY_ITEM : "contains"
8   STRUCT_INVENTORY_ITEM }|..|| CONFIG_ITEM : "ref config"
9   PERSONAL_ISLAND ||--|{ STRUCT_BUILDING : "contains"
10  STRUCT_BUILDING }|..|| CONFIG_BUILDING : "ref config"

```

Các thực thể trọng tâm:

- **USER**: định danh tài khoản; nhúng OBJECT_USER_SETTINGS.
- **GAME_PROFILE**: tách tiền trình theo profile; là “root” cho character/island/npc.
- **SESSION_RUNTIME**: trạng thái phiên hoạt động (token, socket, profile/character đang dùng).
- **CHARACTER**: dữ liệu nhân vật và trạng thái cơ bản.
- **INVENTORY**: danh sách item instance + currency.
- **PERSONAL_ISLAND**: tài nguyên đảo + công trình theo grid.
- **NPC_COLLECTION**: danh sách NPC sở hữu; phục vụ tuyển dụng và worker assignment.

7.4.3 Chỉ mục (indexes) và khoá truy vấn

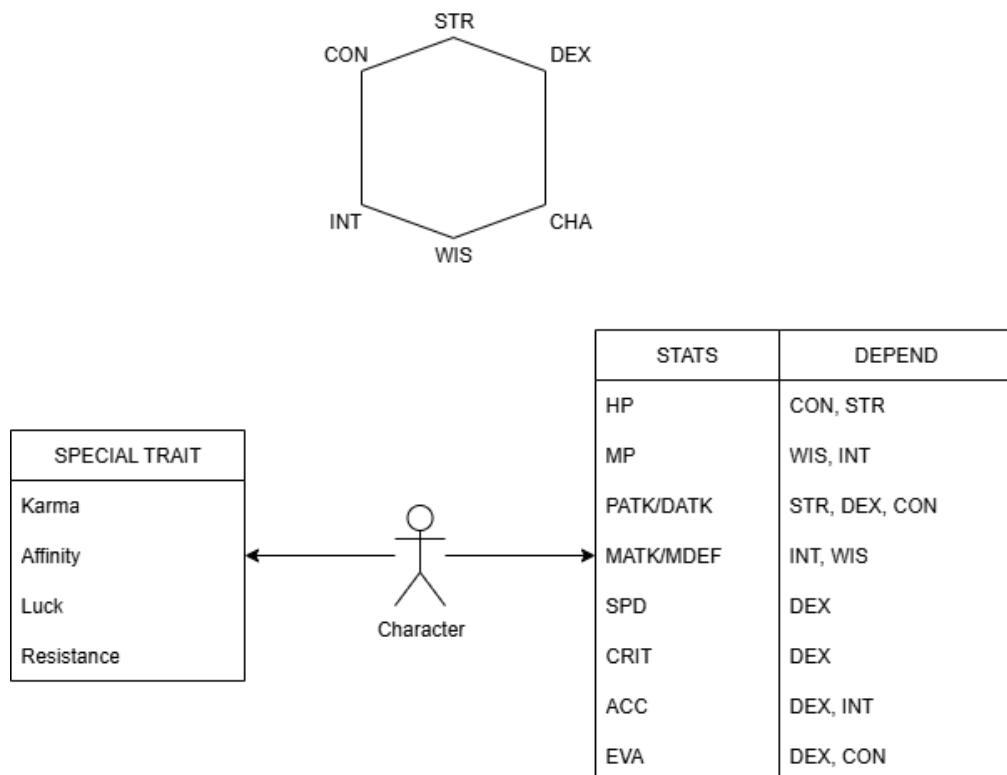
Để đảm bảo truy vấn ổn định, các chỉ mục đề xuất:

- USER: unique index cho `username`, `email`.
- GAME_PROFILE: index theo `user_id`; có thể unique theo (`user_id`, `profile_name`).
- CHARACTER: index theo `profile_id`; có thể unique theo (`profile_id`, `name`) nếu cần chống trùng trong profile.
- INVENTORY: unique index theo `character_id`.
- PERSONAL_ISLAND, NPC_COLLECTION: unique index theo `profile_id`.
- CONFIG collections: unique index theo `id` (khoá cấu hình).

7.5 Thiết kế hệ thống chỉ số và tăng trưởng nhân vật

7.5.1 Thuộc tính nền và chỉ số suy diễn

Nhân vật có 6 thuộc tính nền: STR, DEX, CON, INT, WIS, CHA. Từ đó hệ thống suy diễn các chỉ số chiến đấu như HP/MP/ATK/DEF/CRIT/ACC/EVA. Hình 10 minh họa mối phụ thuộc giữa các nhóm chỉ số.



Hình 10: Mô hình thuộc tính nền và chỉ số suy diễn

Thiết kế tách rõ:

- **Base Stats:** điểm phân bổ trực tiếp bởi người chơi theo level (+5 points/level, tối đa level 100).
- **Derived Stats:** chỉ số tính toán (HP/MP/ATK/...) dựa trên công thức và hệ số cân bằng.
- **Special Traits:** Karma, Affinity, Luck, Resistance là nhóm “đặc tính” dùng cho điều kiện mở khoá và/hoặc hiệu ứng meta.

7.5.2 Tính toán chỉ số theo hướng data-driven

Để tránh hard-code công thức, thiết kế cho phép cấu hình hoá hệ số (ví dụ: HP nhận trọng số từ CON và STR; MP nhận trọng số từ WIS và INT; SPD từ DEX). Cách tiếp cận:

- Công thức được lưu dưới dạng **config** (ví dụ bảng hệ số theo stat).
- Server tính toán derived stats khi:
 - player phân bổ điểm;
 - thay đổi trang bị;
 - thay đổi class/race có modifier.

Ở mức thiết kế, có thể biểu diễn công thức bằng dạng tuyến tính:

$$S_{derived} = \sum_i w_i \cdot A_i + b$$

trong đó A_i là thuộc tính nền (STR/DEX/CON/INT/WIS/CHA) và w_i là hệ số cấu hình.

7.6 Thiết kế quản trị cấu hình game (Config Pipeline)

Pipeline cấu hình chỉ áp dụng cho **các bảng cấu hình/thông số gameplay** (item/building/skill/rates, ...). Dữ liệu tiến trình người chơi vẫn được ghi trực tiếp bởi server trong các collections động.

7.6.1 Luồng dữ liệu cấu hình

Luồng quản trị cấu hình để xuất:

1. **Soạn thảo trên Google Sheets:** mỗi bảng là một nhóm cấu hình (CONFIG_ITEM, CONFIG_BUILDING, rates, formulas).
2. **Xuất TSV:** xuất theo template thống nhất; đảm bảo kiểu dữ liệu và khoá id.
3. **Import vào DB:** script import thực hiện *upsert* theo id; tạo bản ghi version/import log.
4. **Nạp vào Config Manager:** khi server khởi động (hoặc khi client vào game tùy chính sách), server tải config từ DB vào bộ nhớ để truy xuất nhanh.

7.6.2 Thiết kế Config Manager

Config Manager là lớp truy xuất cấu hình thống nhất cho toàn bộ service:

- **In-memory cache:** lưu map `id → config object` cho từng nhóm cấu hình.
- **Versioning:** gắn `config_version` cho lần nạp, phục vụ debug và đảm bảo nhất quán khi tính toán.
- **Validation khi nạp:** kiểm tra dữ liệu TSV sau import (schema, khoá trùng, tham chiếu hợp lệ).

Ở giai đoạn 1, cơ chế reload có thể ở mức đơn giản (restart server để nạp lại). Khi mở rộng, có thể bổ sung:

- endpoint admin “reload config” có xác thực;
- publish sự kiện qua Redis pub/sub để các instance nạp lại đồng bộ.

7.7 Thiết kế giao tiếp client-server

7.7.1 REST API: nhóm endpoint và quy ước

REST API được nhóm theo phân hệ. Mục tiêu thiết kế API:

- tài nguyên rõ ràng (resource-oriented);
- payload ổn định cho UI;
- lỗi có mã và thông điệp nhất quán.

Ví dụ nhóm endpoint (mức thiết kế):

- `/auth/register, /auth/login, /auth/recover`
- `/profiles` (list/create/select)
- `/characters` (list/create/delete/select)
- `/inventory` (get/update equip/enhance)
- `/island` (get/build/upgrade/assign-worker/collect)
- `/gacha` (banners/rates/summon/history/exchange)
- `/admin` (users/ban/mail/logs/events)

7.7.2 WebSocket: định dạng message và vòng đời phiên

Thiết kế message theo dạng envelope để dễ mở rộng:

```
1 {
2   "type": "combat.input",
3   "seq": 1024,
4   "ts": 1730000000,
5   "payload": { ... }
6 }
```

Quy ước chính:

- **type**: loại thông điệp (join, input, snapshot, event, error).
- **seq**: số thứ tự để hỗ trợ kiểm soát spam, debug và (nếu cần) reconciliation.
- **payload**: dữ liệu tùy theo type.

Vòng đời phiên:

1. Client đăng nhập và chọn profile/character qua REST.
2. Client handshake WebSocket kèm token; server xác thực và gán `socket_id`.
3. Client gửi `join` để vào instance/room; server trả dữ liệu spawn và state ban đầu.
4. Trong phiên: client gửi input theo nhịp; server phát snapshot/event.
5. Kết phiên: server trả kết quả, reward delta và trạng thái checkpoint.

7.8 Thiết kế vòng lặp real-time cho combat/instance

7.8.1 Mô hình instance/room

Hệ thống tổ chức người chơi theo **room** (đơn vị logic của một instance). Mỗi room có:

- danh sách người chơi tham gia (socket id, character id);
- trạng thái runtime (vị trí, HP/MP, trạng thái kỹ năng, entity trong phòng);
- cấu hình phiên (mode: tower/dungeon/arena; floor/seed).

7.8.2 Tick loop và phát snapshot

Thiết kế loop ở mức khái niệm:

1. Thu thập input queue từ client.
2. Xác thực input (cooldown, tài nguyên, vị trí hợp lệ, tốc độ).

3. Cập nhật trạng thái authoritative theo tick.
4. Phát snapshot định kỳ và phát event khi có thay đổi quan trọng (hit/loot/death).

Trong giai đoạn 1, cơ chế snapshot có thể thiết kế theo hướng **đơn giản và rõ ràng**:

- snapshot theo tick rate cố định (ví dụ 10–20 tick/s tùy mục tiêu kiểm thử);
- payload snapshot ưu tiên các giá trị tối thiểu để render (pos, hp/mp, state flags);
- chưa bắt buộc triển khai đầy đủ prediction/reconciliation, nhưng định dạng message chưa chỗ cho mở rộng [10, 2].

7.9 Thiết kế phân hệ Administration

Phân hệ admin phục vụ quản trị vận hành và kiểm soát rủi ro:

- **Ban/Unban user:** cập nhật USER.status; thu hồi phiên đang hoạt động.
- **System mail:** gửi thông báo/phần thưởng theo user/profile; được xử lý như job để kiểm soát retry.
- **View logs:** truy vấn log theo thời gian/actor/severity; hỗ trợ điều tra sự cố.
- **Configure events:** tạo cấu hình sự kiện và phần thưởng; dữ liệu sự kiện tách khỏi config item cơ bản.

Các thao tác admin phải có:

- phân quyền theo role;
- audit log đầy đủ (ai làm, làm gì, khi nào, trước/sau);
- cơ chế chống thao tác lặp gây cấp trùng phần thưởng (idempotent key).

7.10 Bảo mật, nhất quán và quan sát hệ thống

7.10.1 Bảo mật

Các nguyên tắc:

- mật khẩu lưu dạng hash; không lưu plaintext;
- token có thời hạn; khi ban user cần thu hồi session;
- phân quyền rõ giữa player và admin (role-based);
- giới hạn tần suất login/gacha và kiểm tra input realtime để chống spam.

7.10.2 Nhất quán dữ liệu và giao dịch nghiệp vụ

Các thao tác như reward, gacha, exchange, nâng cấp công trình cần đảm bảo:

- **atomic ở mức tài liệu** (MongoDB document update) khi cập nhật trong cùng thực thể (inventory/items/currency);
- **idempotency** cho request dễ bị gửi lại (retry do mạng);
- **log nghiệp vụ** để có thể đối soát và khôi phục khi lỗi.

7.10.3 Quan sát hệ thống (observability)

Thiết kế log tối thiểu:

- request log: endpoint, latency, status code, user id (nếu có);
- business log: gacha result, reward delivery, currency delta, admin actions;
- realtime log: join/leave room, disconnect, lỗi xác thực socket.

7.11 Tổng kết chương

Chương này đã mô tả thiết kế hệ thống ở mức kiến trúc, mô-đun, dữ liệu và giao tiếp cho *Fortress of the Fallen*. Thiết kế nhấn mạnh ranh giới trách nhiệm client-server theo hướng server-authoritative, tổ chức backend theo mô-đun rõ ràng, tách dữ liệu tiến trình khỏi dữ liệu cấu hình, và xây dựng pipeline quản trị cấu hình theo hướng data-driven. Các quyết định thiết kế này là nền tảng để mô tả hiện thực hệ thống và kiểm thử trong các chương tiếp theo [7, 8, 9, 10, 2].

8 Kế hoạch hiện thực prototype

8.1 Mục tiêu và phạm vi

Chương này mô tả kế hoạch hiện thực một **prototype** nhằm kiểm chứng các quyết định thiết kế đã trình bày ở Chương 7. Prototype được định hướng như một “bản chạy được” để xác thực:

- Luồng **Identity & Profile**: đăng ký/đăng nhập, chọn profile/nhân vật, thiết lập phiên.
- Luồng **real-time**: client gửi input, server xử lý authoritative và trả snapshot/event.
- Các mô-đun meta-progression có dữ liệu bền vững: **Inventory**, **Gacha**, **Personal Island & NPC**.
- Pipeline **game configuration** theo hướng data-driven: Google Sheets → TSV → DB → Config Manager.

Trong phạm vi prototype, các nội dung (map, quái, AI phức tạp, PvP hoàn chỉnh) không phải trọng tâm; ưu tiên là **đúng luồng, đúng dữ liệu và đúng ranh giới client-server**.

8.2 Kế hoạch triển khai theo mốc

Kế hoạch triển khai được tổ chức theo các mốc tăng dần độ phức tạp: trước tiên dựng khung kết nối và xác thực, sau đó bổ sung hệ thống dữ liệu bền vững và cuối cùng là tích hợp các mô-đun meta.

Mốc	Mục tiêu kỹ thuật	Tính năng kiểm chứng
M1	Dựng backend, DB, auth và kết nối client	Register/Login, token, chọn profile
M2	Dựng kênh real-time và room/instance runtime cơ bản	Join room, gửi input, nhận snapshot (di chuyển)
M3	Tích hợp dữ liệu bền vững và mô-đun meta tối thiểu	Inventory/currency, gacha NPC, island resources, worker assignment
M4	Hoàn thiện pipeline cấu hình và logging/audit tối thiểu	TSV import, config versioning, business logs

Bảng 8: Kế hoạch hiện thực prototype theo mốc

8.3 Hiện thực phía client (Unity)

8.3.1 Tổ chức project và nguyên tắc phân lớp

Client được tổ chức theo hướng tách biệt rõ ràng:

- **Presentation/UI:** màn hình login, profile/character select, HUD cơ bản, panel inventory/gacha/island.
- **Gameplay:** điều khiển nhân vật, state machine animation, xử lý tương tác cơ bản.
- **Networking:** lớp HTTP client (REST) và WebSocket client (real-time), quản lý token, retry và timeout.
- **Data binding:** model dữ liệu để bind ra UI, tránh UI đọc trực tiếp dữ liệu thô từ network.

Nguyên tắc chính:

- Client không tự ý “xác nhận” các thay đổi tiến trình (reward/currency/gacha).
- Client hiển thị trạng thái dựa trên phản hồi server; với realtime, ưu tiên snapshot/event authoritative.

8.3.2 Luồng màn hình đề xuất

Luồng điều hướng UI tối thiểu cho prototype:

1. **Login/Register**: nhập thông tin, gọi API, lưu token.
2. **Profile Select**: liệt kê profile, tạo/chọn profile.
3. **Character Select/Create**: liệt kê nhân vật theo profile, tạo/chọn nhân vật.
4. **Main Hub (Island)**: load scene, hiển thị HUD, mở các panel inventory/gacha/NPC/island.
5. **Join Instance (tùy mộc)**: tham gia room realtime để kiểm chứng đồng bộ di chuyển/combat đơn giản.

8.3.3 Networking client: REST + WebSocket

Client duy trì 2 kênh:

- **REST**: các thao tác quản lý (auth/profile/character/inventory/island/gacha).
- **WebSocket**: join room, gửi input, nhận snapshot/event.

Quy ước xử lý lỗi UI:

- **401/403**: token hết hạn hoặc sai quyền → điều hướng về login.
- **5xx**: lỗi hệ thống → hiển thị trạng thái retry.
- **network timeout**: hiển thị offline/reconnect và không cho thao tác tiến trình.

8.4 Hiện thực phía server (NestJS)

8.4.1 Cấu trúc module và luồng request

Backend triển khai theo module hoá để bám sát ranh giới nghiệp vụ:

- **AuthModule**: register/login, issue token, guard/role.
- **ProfileModule**: CRUD profile, chọn profile, cập nhật last_played.
- **CharacterModule**: CRUD character, load character summary.
- **InventoryModule**: get inventory, apply delta (reward, currency), equip/enhance (tùy mộc).
- **GachaModule**: banner/rate/summon, update pity, duplicate handling.
- **IslandModule**: get island, build/upgrade, resource sync.

- **NpcModule**: list NPC, assign worker, update job state.
- **RealtimeGateway**: websocket auth, join room, input handling, snapshot broadcasting.
- **ConfigModule**: load config collections, provide lookup service, expose config version.

Luồng xử lý chuẩn cho REST:

1. Guard xác thực token và gắn context (userId/profileId/characterId nếu có).
2. Validate DTO và kiểm tra ràng buộc nghiệp vụ.
3. Thực hiện cập nhật DB theo hướng atomic ở mức tài liệu (document update).
4. Ghi business log/audit log nếu là giao dịch quan trọng.

8.4.2 Realtime: room runtime và tick loop tối thiểu

Room runtime được lưu trong bộ nhớ server (in-memory) trong phạm vi prototype:

- `roomId`, `mode` (island/tower/dungeon/arena), danh sách player trong room.
- Runtime state: position, HP/MP, flags trạng thái, cooldown state tối thiểu.
- Input queue theo socket.

Tick loop tối thiểu (không tối ưu hóa) theo các bước:

1. Nhận input và đưa vào queue theo `seq`.
2. Validate input (rate limit, trạng thái hợp lệ, không vượt tốc độ).
3. Cập nhật state và tạo snapshot.
4. Broadcast snapshot theo nhịp (tick rate cấu hình).

Trong mốc đầu, snapshot chỉ cần chứa thông tin đủ để render:

$\{ entityID, pos, rot, hp, mp, stateFlags \}$

Các kỹ thuật nâng cao (prediction/reconciliation) được xem là hướng mở rộng sau khi luồng cơ bản ổn định.

8.5 Thiết kế hiện thực dữ liệu và truy xuất

8.5.1 Schema collections và nguyên tắc embed/reference

Các collections động (progression/state) ưu tiên embed khi dữ liệu gắn chặt vòng đòn:

- **CHARACTER**: nhúng base_stats/resources/position/appearance.
- **INVENTORY**: nhúng danh sách item instance.
- **PERSONAL_ISLAND**: nhúng island resources và danh sách buildings.
- **NPC_COLLECTION**: nhúng danh sách owned NPC.

Các collections cấu hình (static) dùng reference qua khoá id:

- **CONFIG_ITEM**: định nghĩa item.
- **CONFIG_BUILDING**: định nghĩa building và cost/time.
- (mở rộng) **CONFIG_SKILL**, **CONFIG_RATES**, **CONFIG_FORMULAS**, ...

8.5.2 Nguyên tắc cập nhật an toàn cho giao dịch tiến trình

Các thao tác nhạy cảm (reward/currency/gacha/build upgrade) cần:

- **Idempotency key**: mỗi request có mã định danh giao dịch để tránh cấp trùng khi retry.
- **Business log**: ghi lại delta trước/sau (hoặc sự kiện) để đối soát.
- **Atomic update**: với dữ liệu trong một document, ưu tiên update trong một thao tác DB.

8.6 Hiện thực pipeline game configuration

8.6.1 Định dạng TSV và quy ước

Mỗi TSV đại diện một bảng cấu hình, có cột id làm khoá chính. Quy ước:

- Header dòng đầu tiên; encoding UTF-8.
- Kiểu dữ liệu rõ ràng (int/float/string/bool) theo template.
- Các tham chiếu chéo dùng khoá id (ví dụ itemRefId, buildingId).

8.6.2 Import script: upsert và log phiên bản

Script import thực hiện:

1. Parse TSV theo schema (validate required fields, type).
2. Upsert theo `id` vào collection tương ứng.
3. Tạo bản ghi `CONFIG_IMPORT_LOG` gồm: thời gian import, bảng, số dòng, checksum, người thực hiện.
4. Cập nhật `config_version` (tăng dần) để runtime biết phiên bản đang chạy.

8.6.3 Config Manager runtime

Config Manager nạp dữ liệu cấu hình từ DB vào bộ nhớ khi server khởi động:

- Lưu theo map: `(tableName, id) → object`.
- Cung cấp API nội bộ: `getItem(id)`, `getBuilding(id)`, `getFormula(key)`.
- Khi trả dữ liệu về client, có thể kèm `config_version` để debug nhất quán.

8.7 Kế hoạch kiểm thử và tiêu chí nghiệm thu

8.7.1 Kiểm thử theo use case

Kiểm thử ưu tiên theo các use case trọng yếu (Chương 6):

- Auth: register/login, token expiry.
- Profile/Character: create/select, load summary.
- Realtime: join room, move input, snapshot update, disconnect handling.
- Gacha: summon 1x/10x, pity update, duplicate handling, history.
- Island/NPC: build/upgrade timer (mock), assign worker, resource tick (mô phỏng).

8.7.2 Kiểm thử dữ liệu và tính nhất quán

Các kịch bản cần kiểm chứng:

- Retry request (gacha/reward) không làm nhân đôi kết quả.
- Inventory stack và slot không vượt giới hạn.
- Worker assignment không tạo trạng thái treo (NPC vừa “rảnh” vừa “đang làm”).
- Import TSV sai schema bị từ chối và không làm “bẩn” config.

8.7.3 Tiêu chí nghiệm thu prototype

Prototype đạt yêu cầu khi thoả tối thiểu:

- Có thể đăng nhập, chọn profile/nhân vật và vào scene hub.
- Có thể tham gia room realtime và đồng bộ di chuyển ổn định.
- Gacha trả kết quả hợp lệ, lưu DB, cập nhật pity và xử lý trùng lặp.
- Island/NPC có luồng dữ liệu bền vững (lưu và tải lại đúng).
- Pipeline TSV import tạo được config version và runtime truy xuất đúng theo version.

8.8 Tổng kết chương

Chương này đã trình bày kế hoạch hiện thực prototype nhằm kiểm chứng thiết kế hệ thống: tổ chức client và server, luồng REST/WebSocket, room realtime tối thiểu, chiến lược lưu trữ dữ liệu, pipeline game configuration theo TSV và kế hoạch kiểm thử theo use case. Kết quả của chương là cơ sở để thực hiện đánh giá/đối chiếu trong chương tiếp theo khi prototype được triển khai và chạy thử.

9 Đánh giá hệ thống

9.1 Mục tiêu và phương pháp đánh giá

Mục tiêu của chương này là đánh giá mức độ phù hợp của bản thiết kế hệ thống (Chương 7) so với yêu cầu đã phân tích (Chương 5), đồng thời chỉ ra các điểm mạnh, hạn chế và rủi ro còn tồn tại. Do giai đoạn 1 tập trung vào thiết kế, phương pháp đánh giá được thực hiện theo các hướng:

- **Đối chiếu bao phủ yêu cầu (Requirement Coverage):** kiểm tra các yêu cầu chức năng/phi chức năng đã được ánh xạ tới mô-đun, dữ liệu và luồng giao tiếp hay chưa.
- **Đánh giá kiến trúc theo thuộc tính chất lượng (Quality Attributes):** nhất quán dữ liệu, bảo mật, khả năng mở rộng, tính bảo trì, khả năng quan sát.
- **Đánh giá tính khả thi triển khai (Feasibility):** dựa trên kế hoạch prototype ở Chương 8 và các kịch bản use case trọng yếu ở Chương 6.
- **Phân tích rủi ro và hướng giảm thiểu:** xác định các rủi ro kỹ thuật/thiết kế, mức ảnh hưởng và biện pháp kiểm soát.

9.2 Đổi chiểu bao phủ yêu cầu chức năng

9.2.1 Ma trận bao phủ yêu cầu theo mô-đun

Bảng 9 tổng hợp ánh xạ giữa các nhóm yêu cầu chức năng (FR) và mô-đun thiết kế ở Chương 7. Mục tiêu là đảm bảo mọi nhóm yêu cầu đều có “điểm đặt” rõ ràng trong kiến trúc.

Nhóm FR	Phân hệ	Mô-đun/Thiết kế đáp ứng
FR-A	Auth & Profile	AuthModule, ProfileModule, SessionRuntime (token, profile_id, socket_id)
FR-B	Character	CharacterModule (create/select/delete), dữ liệu nhúng appearance/base_stats/resources
FR-C	Stats/Level/Slots	CharacterModule + ConfigModule (milestone slots, công thức chỉ số data-driven)
FR-D	Class System	CharacterModule/Progression service (unlock/switch), điều kiện theo stat/trait/achievement
FR-E	Inventory/Economy	InventoryModule (stack, capacity, currency, equip/enhance), tham chiếu CONFIG_ITEM
FR-F	Combat/Modes	CombatModule + RealtimeGateway (join instance, tick, snapshot, end-session reward/checkpoint)
FR-G	Gacha	GachaModule (banner/rate/pity/duplicate/history), NPC_COLLECTION cập nhật bền vững
FR-H	Island & NPC	IslandModule (build/upgrade/timer/resources), NpcModule (assign worker, job state)
FR-I	Admin	AdminModule (ban/unban, mail, logs, event config), audit log

Bảng 9: Ma trận bao phủ yêu cầu chức năng theo mô-đun

Kết quả đổi chiểu cho thấy các nhóm tính năng chính đều đã được gắn với mô-đun và dữ liệu tương ứng, phù hợp hướng module hoá theo phân hệ của backend [7].

9.2.2 Đánh giá luồng nghiệp vụ trọng yếu

Các luồng nghiệp vụ được xem là trọng yếu vì ảnh hưởng trực tiếp tới tiến trình hoặc tính công bằng:

(1) **Auth → Profile → Character → Session** Luồng này có trạng thái phụ thuộc theo chuỗi (USER → GAME_PROFILE → CHARACTER) và có yêu cầu nhất quán ở session runtime. Thiết kế đáp ứng bằng:

- REST dùng để xác thực và chọn profile/character;

- session runtime lưu `profile_id/character_id` để WebSocket có thể xác định bối cảnh phiên;
- token có hạn và có thể revoke khi cần (ban/unban).

(2) Reward delivery và cập nhật inventory Luồng thường sau phiên (tower/dungeon) để phát sinh lỗi cấp trùng nếu người chơi retry hoặc mất kết nối. Thiết kế giảm rủi ro bằng:

- server-authoritative: client không tự ghi reward;
- cập nhật inventory theo quy tắc stack/capacity và tham chiếu config item;
- đề xuất idempotency và business log để đối soát.

(3) Gacha: rate/pity/duplicate Luồng gacha có yêu cầu “đúng xác suất theo cấu hình” và “không cấp trùng”. Thiết kế đáp ứng bằng:

- rates/banners là game configuration (tĩnh) và có thể quản trị bằng pipeline;
- pity là state (động) lưu bền vững theo banner hoặc nhóm banner;
- duplicate handling chuyển đổi theo rule, giảm cảm giác “mất giá trị”.

(4) Island build/upgrade và worker assignment Luồng island có rủi ro trạng thái treo (worker vừa rảnh vừa bận, hoặc building có worker nhưng worker không trả về building). Thiết kế đáp ứng bằng:

- cập nhật liên kết hai chiều khi assign worker;
- timer (`finish_time`) và trạng thái building rõ ràng;
- island/resources và buildings nhúng trong cùng thực thể để truy xuất nhất quán.

9.3 Đánh giá yêu cầu phi chức năng

9.3.1 Bảo mật và phân quyền

Thiết kế đáp ứng các yêu cầu bảo mật nền tảng:

- mật khẩu lưu hash (không plaintext);
- token có thời hạn; WebSocket handshake phải xác thực token;
- phân quyền admin theo role; thao tác admin có audit log;
- giới hạn tần suất thao tác nhạy cảm (login/gacha) và kiểm soát spam input realtime.

Hướng tiếp cận này phù hợp nguyên tắc hệ thống online: hạn chế tin tưởng client và đặt kiểm tra luật ở server [2].

9.3.2 Nhất quán dữ liệu và khả năng chống gấp trùng

Đối với game online, nhất quán dữ liệu quan trọng hơn tối ưu sớm. Thiết kế thể hiện các điểm tích cực:

- tách rõ dữ liệu động và tĩnh; dữ liệu tĩnh được nạp qua Config Manager để đồng bộ cách tĩnh;
- cập nhật tiến trình (inventory/gacha/island) định hướng atomic ở mức document;
- đề xuất idempotency key cho các giao dịch dễ retry (reward/gacha/build upgrade).

Hạn chế còn lại: ở mức thiết kế, cơ chế idempotency và transaction log mới dùng ở nguyên tắc, chưa có đặc tả schema log/delta chi tiết cho từng giao dịch. Đây là phần cần đặc tả thêm khi chuyển sang hiện thực.

9.3.3 Hiệu năng và khả năng mở rộng

Thiết kế ưu tiên chạy ổn định ở mức đồ án, nhưng vẫn có hướng mở rộng:

- realtime tách riêng bằng WebSocket gateway và mô hình room runtime in-memory để giảm độ trễ;
- Redis được định hướng dùng cho cache/pub-sub khi scale ngang nhiều instance;
- Config Manager nạp config vào bộ nhớ giúp giảm truy vấn DB cho dữ liệu tĩnh.

Hạn chế:

- room runtime in-memory khiến việc scale ngang cần thêm cơ chế phân vùng room (room sharding) hoặc session affinity;
- chưa có tiêu chí tick-rate/bandwidth cụ thể theo mục tiêu tải.

Các hạn chế này phù hợp với phạm vi giai đoạn 1 (thiết kế), và sẽ được xử lý khi nâng mức triển khai/benchmark.

9.3.4 Tính bảo trì và khả năng mở rộng tính năng

Tính bảo trì được hỗ trợ bởi:

- module hoá theo phân hệ rõ ràng (Auth/Profile/Character/Inventory/Combat/Gacha/Island/Admin)
- cross-cutting service cho validation, error handling, logging;
- cấu hình data-driven giúp thay đổi nội dung/thông số mà không sửa code lõi.

Điểm cần chú ý: khi số lượng phân hệ tăng, cần kiểm soát phụ thuộc chéo (đặc biệt giữa Progression/Inventory/Gacha) bằng cách chuẩn hoá service interface và event nội bộ.

9.3.5 Khả năng quan sát (Observability)

Thiết kế đã nêu các lớp log tối thiểu:

- request log (latency/status/user context);
- business log (reward/currency/gacha/admin actions);
- realtime log (join/leave room/disconnect).

Dánh giá: hướng này đủ cho giai đoạn prototype và debug nghiệp vụ. Để nâng lên vận hành thực tế, cần bổ sung thêm metric (tick time, snapshot size, error rate), tracing và cảnh báo.

9.4 Đánh giá pipeline cấu hình game (data-driven config)

Pipeline cấu hình: Google Sheets → TSV → DB → Config Manager, chỉ áp dụng cho **game configuration**.

9.4.1 Ưu điểm

- **Giảm hard-code:** các thông số (item/building/rate/formula) được thay đổi qua TSV.
- **Dễ kiểm soát phiên bản:** TSV có thể quản lý qua version control; import log giúp truy vết.
- **Tối ưu truy xuất runtime:** Config Manager nạp in-memory, phù hợp với các phép tra cứu thường xuyên.

9.4.2 Rủi ro và kiểm soát

- **Sai schema/kiểu dữ liệu:** cần validation khi import và báo lỗi rõ ràng theo dòng/cột.
- **Tham chiếu chéo sai:** cần kiểm tra referential integrity (itemRefId/buildingId/skillId).
- **Không đồng bộ phiên bản giữa instance:** nếu chạy nhiều server, cần cơ chế reload đồng bộ hoặc restart theo chiến lược triển khai.

9.5 Đánh giá UI/UX theo định hướng pixel art ở mức hệ thống

Do UI theo phong cách pixel art nhấn mạnh readability và phản hồi nhanh, thiết kế hệ thống hỗ trợ UX bằng:

- chuẩn hoá trạng thái UI (loading/error/disabled) thông qua mã lỗi nhất quán;
- ưu tiên payload “delta” hoặc snapshot nhỏ gọn cho HUD combat;
- phân tách màn hình meta (inventory/gacha/island) khỏi HUD combat để không làm gián đoạn nhịp chơi.

Hạn chế: chưa có đặc tả UI state machine chi tiết cho từng màn (ví dụ: trường hợp disconnect khi đang ở gacha hoặc đang xây building). Đây là phần có thể mở rộng ở giai đoạn hiện thực prototype.

9.6 Phân tích rủi ro và hướng giảm thiểu

Bảng 10 liệt kê các rủi ro kỹ thuật chính và hướng giảm thiểu trong phạm vi đồ án.

Rủi ro	Mức	Mô tả	Giảm thiểu
Cáp trùng reward/gacha	Cao	Retry do mạng/mất kết nối có thể tạo giao dịch lặp	Idempotency key, business log, kiểm thử retry, chốt trạng thái bằng server-authoritative
Trạng thái treo worker assignment	Trung bình	NPC và building lệch trạng thái (một chiều)	Cập nhật hai chiều, validate trước khi assign/unassign, kịch bản kiểm thử nhất quán
Sai config gây mất cân bằng/lỗi runtime	Cao	TSV sai kiểu hoặc tham chiếu chéo sai gây crash hoặc lệch gameplay	Validation import, schema strict, referential checks, config version + rollback
Realtime latency/jitter làm mất cảm giác combat	Trung bình	Snapshot không ổn định gây giật/khó điều khiển	Tick-rate hợp lý, snapshot tối thiểu, client interpolation; chừa chỗ mở rộng prediction/reconciliation
Phụ thuộc chéo giữa mô-đun tăng dần	Trung bình	Feature mới kéo theo gọi chéo nhiều service khó bảo trì	Chuẩn hóa interface, domain services rõ ràng, event nội bộ, review dependency định kỳ

Bảng 10: Bảng rủi ro và hướng giảm thiểu

9.7 Tổng kết chương

Kết quả đánh giá cho thấy bản thiết kế đã:

- bao phủ các nhóm yêu cầu chức năng chính thông qua mô-đun hoá rõ ràng và mô hình dữ liệu nhất quán;
- đáp ứng các yêu cầu phi chức năng nền tảng (bảo mật, nhất quán, bảo trì, quan sát) ở mức phù hợp với phạm vi giai đoạn 1;

- đưa ra pipeline quản trị game configuration theo hướng data-driven, giúp giảm phụ thuộc vào hard-code và tạo nền tảng cân bằng nội dung.

Các điểm còn cần đặc tả sâu hơn khi chuyển sang hiện thực gồm: cơ chế idempotency/log cho từng giao dịch, chính sách xử lý overflow inventory, tiêu chí tick-rate/bandwidth cho realtime và đặc tả UI state chi tiết cho các tình huống lỗi mạng.

10 Kết luận và hướng phát triển

10.1 Tổng kết kết quả đạt được

Đồ án *Fortress of the Fallen* được xây dựng theo định hướng một game Action RPG online với lớp meta-progression phong phú, kết hợp nhiều mô-đun gameplay: tiến trình leo tháp (tower), chiến đấu theo phiên (instance), quản lý nhân vật và chỉ số, gacha tuyển dụng thực thể dài hạn (NPC), và hệ thống đảo cá nhân theo hướng base-building.

Trong phạm vi giai đoạn 1 tập trung vào **thiết kế hệ thống**, báo cáo đã đạt được các kết quả chính:

- Xác định yêu cầu và phạm vi:** hệ thống hoá các yêu cầu chức năng theo phân hệ (auth/profile, character/progression, inventory/economy, combat, gacha, island & NPC, admin) và các yêu cầu phi chức năng (bảo mật, nhất quán, mở rộng, quan sát).
- Phân tích hệ thống theo use case và luồng nghiệp vụ:** mô tả tác nhân, use case tổng quan và đặc tả các luồng trọng yếu (đăng nhập-chọn profile, tạo nhân vật, reward delivery, gacha, island build/upgrade và worker assignment).
- Thiết kế kiến trúc client-server:** phân tách rõ REST cho thao tác quản lý và WebSocket cho realtime; áp dụng nguyên tắc server-authoritative cho các thao tác ảnh hưởng tiến trình.
- Thiết kế dữ liệu:** tổ chức thực thể lõi (USER, GAME_PROFILE, CHARACTER, INVENTORY, PERSONAL_ISLAND, NPC_COLLECTION, SESSION_RUNTIME), chỉ ra quan hệ và nguyên tắc embed/reference phù hợp cho MongoDB.
- Thiết kế pipeline game configuration theo hướng data-driven:** Google Sheets → TSV → DB → Config Manager (chỉ áp dụng cho các bảng cấu hình/thông số gameplay), giúp giảm hard-code và hỗ trợ quản trị phiên bản cấu hình.
- Kế hoạch hiện thực prototype và đánh giá thiết kế:** đề xuất các mốc triển khai để kiểm chứng thiết kế và đánh giá mức độ bao phủ yêu cầu, chất lượng kiến trúc, rủi ro và biện pháp giảm thiểu.

10.2 Đóng góp của đồ án

Các đóng góp chính của đồ án trong phạm vi thiết kế bao gồm:

- **Bộ khung thiết kế hệ thống hoàn chỉnh cho một game online theo mô-đun**, có thể mở rộng theo nhiều hướng (thêm nội dung, mở rộng server, bổ sung dịch vụ phụ trợ).
- **Mô hình dữ liệu bền vững và nhất quán** phục vụ các mô-đun meta-progression (inventory, gacha, island & NPC) đồng thời giữ được ranh giới rõ ràng giữa dữ liệu động (progress) và dữ liệu tĩnh (config).
- **Định hướng giao tiếp realtime có khả năng phát triển**: thiết kế envelope message và vòng đời phiên (handshake, join room, input, snapshot/event, end session), tạo nền cho việc bổ sung các kỹ thuật networking nâng cao.
- **Pipeline cấu hình game độc lập với dữ liệu tiến trình**, giúp nhóm nội dung có thể điều chỉnh cân bằng và mở rộng nội dung mà không phụ thuộc nhiều vào thay đổi mã nguồn.

10.3 Hạn chế

Do tập trung vào giai đoạn 1 (thiết kế), một số nội dung chưa được hiện thực hoặc chưa được kiểm chứng bằng benchmark:

- **Cơ chế giao dịch nghiệp vụ chi tiết**: idempotency/log/delta cho từng loại giao dịch (reward, gacha, upgrade) mới dừng ở mức nguyên tắc; cần đặc tả schema log và quy trình đối soát cụ thể hơn khi hiện thực.
- **Realtime nâng cao**: thiết kế đã chừa chỗ cho mở rộng, nhưng chưa triển khai và đánh giá các kỹ thuật như prediction/reconciliation, interest management và tối ưu băng thông.
- **Chính sách xử lý ngoại lệ**: các trường hợp đặc thù như overflow inventory, reconnect giữa phiên, rollback khi import config lỗi cần được đặc tả sâu và kiểm thử khi prototype chạy thật.
- **Đánh giá hiệu năng theo tải**: chưa có số đo thực nghiệm cho tick time, snapshot size, throughput hoặc độ ổn định khi tăng số lượng kết nối.

10.4 Hướng phát triển

Trong các giai đoạn tiếp theo, hệ thống có thể phát triển theo các hướng ưu tiên sau:

10.4.1 Hiện thực prototype và kiểm chứng theo use case

- Hoàn thiện prototype theo các mốc đã đề xuất: auth/profile, realtime room, meta modules (inventory/gacha/island), pipeline config.
- Xây dựng bộ kiểm thử theo kịch bản: retry giao dịch, mất kết nối, đồng bộ dữ liệu sau reconnect, consistency giữa NPC và building.

10.4.2 Nâng cấp realtime cho cảm giác chiến đấu

- Bổ sung interpolation chuẩn cho client và phân tách snapshot/event rõ ràng.
- Thử nghiệm prediction/reconciliation cho chuyển động cơ bản, sau đó mở rộng cho kỹ năng.
- Tối ưu payload snapshot (delta compression) và áp dụng interest management khi có nhiều thực thể trong room.

10.4.3 Hoàn thiện hệ thống giao dịch và chống gian lận

- Chuẩn hoá idempotency key cho các endpoint nhạy cảm (reward/gacha/upgrade).
- Thiết kế business ledger (sổ giao dịch) để truy vết thay đổi currency và phần thưởng theo thời gian.
- Tăng cường kiểm soát input realtime (anti-spam/anti-speedhack) dựa trên luật server-authoritative.

10.4.4 Mở rộng pipeline cấu hình

- Mở rộng nhóm config: skill definitions, drop tables, banner schedules, formulas và economy sinks/sources.
- Bổ sung kiểm tra tham chiếu chéo và cơ chế rollback phiên bản khi import gây lỗi.
- Khi chạy nhiều instance server: đồng bộ reload config bằng pub/sub hoặc chiến lược triển khai có kiểm soát phiên bản.

10.4.5 Chuẩn hóa quan sát hệ thống

- Bổ sung metrics (tick time, snapshot size, error rate), tracing theo request/correlation id.
- Chuẩn hóa log nghiệp vụ cho các thao tác quan trọng và dashboard quan sát cho admin.

10.5 Kết luận

Báo cáo đã trình bày đầy đủ quá trình từ xác định bài toán, phân tích yêu cầu, phân tích hệ thống đến thiết kế kiến trúc, dữ liệu và kế hoạch hiện thực prototype cho *Fortress of the Fallen*. Các quyết định thiết kế tập trung vào tính mô-đun, tính nhất quán dữ liệu và hướng data-driven cho cấu hình gameplay, tạo nền tảng vững để triển khai prototype và mở rộng hệ thống trong các giai đoạn tiếp theo.

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] Glenn Fiedler. *Gaffer on Games: Networking for Game Programmers*. Tài liệu tham khảo về server authoritative, prediction, interpolation. 2010.
- [11] Microsoft. *C# Programming Guide*. <https://learn.microsoft.com/dotnet/csharp/>. Tài liệu chính thức về ngôn ngữ C#. 2023.
- [12] OpenJS Foundation. *Node.js Documentation*. <https://nodejs.org/en/docs>. Tài liệu chính thức của Node.js. 2023.
- [13] GitHub. *GitHub Documentation*. <https://docs.github.com>. Hướng dẫn sử dụng GitHub và Git cơ bản. 2023.
- [14] 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.
- [15] Thomas Allweyer. *BPMN 2.0: Introduction to the Standard for Business Process Modeling*. Giới thiệu BPMN 2.0 dùng cho mô hình hóa quy trình. Books on Demand, 2016.
- [16] Frank Mittelbach and others. *The L^AT_EX Companion*. 2 edition. Tài liệu tham khảo về soạn thảo tài liệu bằng L^AT_EX. Addison-Wesley, 2004.

- [17] Reki Kawahara. *Sword Art Online, Vol. 1: Aincrad*. Nguồn tham khảo ý tưởng leo tầng và boss tầng. Yen Press, 2014.
- [18] Kugane Maruyama. *Overlord, Vol. 1: The Undead King*. Nguồn tham khảo cách xây dựng NPC và thế giới vận hành. Yen Press, 2016.
- [19] Pick Me Up! Infinite Gacha. *Pick Me Up! Infinite Gacha (Webtoon/Manhwa)*. Nguồn tham khảo cơ chế gacha và meta-progression. 2022.
- [20] Arcane Odyssey. *Arcane Odyssey (Game)*. Nguồn tham khảo trải nghiệm online RPG và cảm giác điều khiển. 2023.
- [21] ChillyRoom. *Soul Knight Prequel (Game)*. Nguồn tham khảo vòng lặp phiên chơi ngắn và reward theo phiên. 2023.
- [22] Supercell. *Clash of Clans (Game)*. Nguồn tham khảo hệ thống căn cứ/công trình và quản trị tài nguyên. 2012.