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

ĐỒ ÁN CHUYÊN NGÀNH
BÁO CÁO

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

"FORTRESS OF THE FALLEN"

Ngành: Khoa học Máy tính

Học kỳ 1 năm học 2025–2026 (HK251)

Giảng viên hướng dẫn: ThS. Vương Bá Thịnh

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

TP. Hồ Chí Minh, năm 2025

Danh mục thuật ngữ

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

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

(còn tiếp trang sau)

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

(còn tiếp trang sau)

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

Mục lục

Danh sách hình vẽ

Danh sách bảng

1 Giới thiệu

1.1 Động lực

Trong bối cảnh ngành công nghiệp game phát triển mạnh mẽ, các trò chơi nhập vai trực tuyến nhiều người chơi (Multiplayer Online RPG) ngày càng trở nên phổ biến. Người chơi không chỉ kỳ vọng vào đồ họa đẹp mắt, nội dung hấp dẫn mà còn yêu cầu trải nghiệm trực tuyến ổn định, mượt mà, có khả năng hỗ trợ nhiều người chơi tương tác với nhau trong thời gian thực.

Để đáp ứng các yêu cầu này, hệ thống phía server cần có khả năng xử lý lượng lớn kết nối đồng thời, đồng bộ trạng thái nhân vật, quản lý tài nguyên và dữ liệu trò chơi một cách hiệu quả, đồng thời vẫn đảm bảo độ trễ thấp. Việc nghiên cứu, thiết kế và hiện thực một hệ thống game online hoàn chỉnh đòi hỏi sinh viên phải kết hợp nhiều mảng kiến thức: lập trình game, mạng máy tính, kiến trúc client–server, cơ sở dữ liệu, cũng như thiết kế cơ chế gameplay.

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

1.2 Mục tiêu

Mục tiêu tổng quát của đề tài là thiết kế và phát triển hệ thống nền tảng cho game online nhập vai hành động nhiều người chơi "Fortress of the Fallen". Cụ thể, trong giai đoạn 1, đề tài hướng đến các mục tiêu sau:

- Nghiên cứu và tổng hợp ý tưởng thiết kế game từ các tác phẩm tham khảo (tiểu thuyết và game), từ đó định hình bối cảnh, cơ chế chơi và hệ thống tính năng cốt lõi của trò chơi.
- Xác định phạm vi, tính khả thi kỹ thuật và mức độ phù hợp của đề tài trong khuôn khổ đồ án chuyên ngành.
- Thiết kế mô hình kiến trúc hệ thống, bao gồm các thành phần: client, server backend, cơ sở dữ liệu và cơ chế giao tiếp mạng.
- Hiện thực bộ khung ban đầu (source base) cho hệ thống, sử dụng Unity cho client và NestJS kết hợp .NET cho backend, làm nền tảng để phát triển tiếp ở các giai đoạn sau.

1.3 Phạm vi

Trong phạm vi giai đoạn 1 của đồ án chuyên ngành, đề tài tập trung vào việc xây dựng *prototype nền tảng* hơn là hoàn thiện toàn bộ nội dung game. Cụ thể, phạm vi thực hiện bao gồm:

- Thiết kế kiến trúc tổng thể cho hệ thống game online nhiều người chơi.
- Thiết kế và xây dựng mô hình dữ liệu cơ bản phục vụ cho quản lý tài khoản, nhân vật, đảo cá nhân và NPC.
- Xây dựng các chức năng cơ bản trên client:
  - Giao diện đăng ký, đăng nhập.
  - Kết nối đến server, xác thực người dùng.
  - Vào đảo cá nhân và hiển thị nhân vật chính.
  - Điều khiển di chuyển nhân vật trên đảo.
- Xây dựng các chức năng nền tảng trên server:
  - Dịch vụ xác thực (authentication) và quản lý người dùng.
  - Quản lý phiên làm việc và kết nối real-time.
  - Cơ chế gacha NPC cơ bản và lưu trữ NPC vào cơ sở dữ liệu.
  - Cài đặt AI đơn giản cho NPC (ví dụ: làm việc định kỳ để tạo tài nguyên).

Các nội dung nâng cao như hệ thống leo tháp 100 tầng, 18 tầng ngục, hệ thống class–race–skill hoàn chỉnh, PvP Arena, cân bằng game và tối ưu hiệu năng nâng cao sẽ được xem xét ở các giai đoạn tiếp theo và nằm ngoài phạm vi hiện tại.

1.4 Ý nghĩa của đề tài

1.4.1 Ý nghĩa thực tiễn

Về mặt thực tiễn, đề tài giúp:

- Cung cấp một ví dụ cụ thể về quá trình xây dựng hệ thống game online nhiều người chơi từ bước ý tưởng đến kiến trúc và hiện thực.
- Minh họa cách kết hợp các công nghệ phổ biến trong phát triển game và backend như Unity, NestJS, .NET, MongoDB, Redis, ...
- Làm nền tảng để mở rộng thành một sản phẩm game hoàn chỉnh hoặc một bộ khung tham khảo cho các đồ án, dự án nghiên cứu khác liên quan đến game online.

1.4.2 Ý nghĩa khoa học và học thuật

Về mặt học thuật, đề tài mang lại các giá trị sau:

- Ứng dụng kiến thức đã học về cấu trúc dữ liệu và giải thuật, lập trình hướng đối tượng, cơ sở dữ liệu, mạng máy tính và kiến trúc phần mềm vào một bài toán thực tế.

- Tạo điều kiện cho sinh viên tiếp cận mô hình kiến trúc phân tán, xử lý kết nối đồng thời và bài toán đồng bộ trạng thái trong môi trường real-time.
- Góp phần xây dựng tài liệu tham khảo nội bộ về quy trình phân tích, thiết kế và hiện thực một hệ thống game online nhiều người chơi trong khuôn khổ chương trình Khoa học Máy tính.

1.5 Cấu trúc báo cáo

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

- **Chương 2 – Kiến thức nền tảng:** Trình bày các khái niệm, định nghĩa và kiến thức liên quan đến game Action RPG, hệ thống multiplayer, kiến trúc client–server và các cơ chế gameplay liên quan.
- **Chương 3 – Công nghệ sử dụng:** Giới thiệu các công nghệ chính được sử dụng trong đề tài như Unity, NestJS, .NET, MongoDB, Redis, cùng lý do lựa chọn.
- **Chương 4 – Các công trình liên quan:** Phân tích các tác phẩm và trò chơi tham khảo, các nghiên cứu về game networking và kiến trúc server real-time, từ đó rút ra bài học áp dụng cho đề tài.
- **Chương 5 – Phân tích yêu cầu:** Xác định người dùng hệ thống, yêu cầu chức năng, phi chức năng và yêu cầu dữ liệu cho giai đoạn 1.
- **Chương 6 – Phân tích hệ thống:** Mô tả quy trình nghiệp vụ, đặc tả use case và luồng dữ liệu logic trong hệ thống.
- **Chương 7 – Thiết kế hệ thống:** Trình bày kiến trúc tổng thể, thiết kế cơ sở dữ liệu, thiết kế API và các mô-đun chính của hệ thống.
- **Chương 8 – Hiện thực hệ thống:** Mô tả quá trình cài đặt các thành phần client và server, cơ chế giao tiếp và tích hợp.
- **Chương 9 – Đánh giá hệ thống:** Đánh giá mức độ hoàn thành so với mục tiêu, kiểm thử các chức năng chính và thảo luận các hạn chế.
- **Chương 10 – Kết luận:** Tổng kết kết quả đạt được, nêu các hạn chế và định hướng phát triển trong các giai đoạn tiếp theo.

2 Kiến thức nền tảng

Chương này trình b trình bày hệ thống các kiến thức cơ sở liên quan đến phát triển trò chơi điện tử thuộc thể loại Action RPG trong môi trường trực tuyến. Nội dung bao gồm: (1) các khái niệm nền tảng về game thời gian thực và hệ thống multiplayer; (2) kiến thức nền tảng về

thiết kế game (Game Design), bao gồm core loop, progression, combat, economy và UX. Những kiến thức này đóng vai trò quan trọng làm cơ sở cho phân tích yêu cầu và thiết kế hệ thống ở các chương tiếp theo.

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

2.1.1 Đặc điểm của Action RPG

Action RPG (Action Role-Playing Game) là thể loại game kết hợp cơ chế nhập vai với chiến đấu thời gian thực. Đặc điểm quan trọng bao gồm:

- **Tương tác thời gian thực**: mọi hành động như tấn công, di chuyển, né tránh phải được phản hồi ngay lập tức.
- **Combat gắn liền với animation**: chuyển động của nhân vật ảnh hưởng trực tiếp đến logic chiến đấu.
- **Hitbox / Hurtbox**: hệ thống kiểm tra va chạm dựa trên hình học đơn giản như AABB, Sphere, Capsule.
- **Kỹ năng người chơi và chỉ số nhân vật phối hợp**: hiệu quả chiến đấu phụ thuộc vào cả thao tác người chơi và stat.

Một đòn tấn công chuẩn thường gồm ba giai đoạn:

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

Những khái niệm này giúp đảm bảo tính logic và cảm giác "mượt" khi chiến đấu.

2.1.2 Hệ thống multiplayer trong Action RPG

Một Action RPG nhiều người chơi yêu cầu cơ chế đồng bộ trạng thái giữa các người chơi sao cho tất cả đều thấy cùng một diễn biến. Các kiến thức nền trong hệ thống multiplayer bao gồm:

Mô hình client–server Hầu hết game online dùng mô hình server authoritative:

- **Client**: thu input và hiển thị kết quả.
- **Server**: xử lý toàn bộ logic gameplay và quản lý trạng thái thật.

Ưu điểm:

- đảm bảo tính nhất quán,
- chống gian lận,
- dễ quản lý khi mở rộng.

Tick-rate và vòng lặp cập nhật Server hoạt động theo "tick", ví dụ 20 hoặc 30 lần mỗi giây. Mỗi tick:

```
- Nhận input từ người chơi
- Cập nhật vị trí và trạng thái
- Kiểm tra va chạm và xử lý chiến đấu
- Gửi snapshot cho client
```

Tick-rate càng cao, gameplay càng mượt nhưng tốn tài nguyên.

Đồng bộ trạng thái Hai kỹ thuật phổ biến:

- **Snapshot interpolation**: nội suy giữa hai snapshot để chuyển động mượt.
- **Client-side prediction**: client dự đoán tạm thời để tránh cảm giác điều khiển bị trễ.

Kết hợp:

- **Server reconciliation**: server gửi dữ liệu chính xác, client điều chỉnh sai lệch.

Những kiến thức này giúp giảm ảnh hưởng của độ trễ mạng trong môi trường real-time.

2.2 Kiến thức nền tảng về Game Design

Game Design là lĩnh vực xây dựng cơ chế hoạt động của trò chơi, quy định người chơi làm gì, cách họ tương tác và động lực khiến họ tiếp tục chơi. Một trò chơi hoàn chỉnh bao gồm nhiều lớp thiết kế: core loop, hệ thống phát triển nhân vật, combat, tài nguyên, phần thưởng và trải nghiệm người dùng.

2.2.1 Core Loop và Meta Loop

Core Loop Core loop là vòng lặp hành động cốt lõi mà người chơi thực hiện liên tục trong suốt thời gian chơi. Một ví dụ điển hình cho Action RPG:

$$\textit{Chiến đấu} \rightarrow \textit{Nhận phần thưởng} \rightarrow \textit{Tăng tiến sức mạnh} \rightarrow \textit{Quay lại chiến đấu}$$

Một core loop tốt phải:

- đơn giản,
- dễ hiểu,
- có nhịp độ nhanh,
- tạo cảm giác "hài lòng" mỗi vòng lặp.

Meta Loop Meta loop là hệ thống dài hạn thúc đẩy người chơi gắn bó lâu hơn, ví dụ:

- tăng cấp nhân vật,
- mở khóa kỹ năng mới,
- thu thập trang bị,
- khám phá khu vực mới.

Nếu core loop tạo "niềm vui ngắn hạn", meta loop tạo "động lực dài hạn".

2.2.2 Progression System (Hệ thống phát triển nhân vật)

Progression giúp định hình hành trình phát triển của nhân vật. Các yếu tố nền tảng:

- **Leveling**: lên cấp để mở khóa chỉ số hoặc kỹ năng.
- **Stats**: các chỉ số như sức mạnh, nhanh nhẹn, trí lực, máu.
- **Equipment Progression**: vũ khí, áo giáp, vật phẩm hỗ trợ.
- **Skill Tree**: hệ thống kỹ năng dạng nhánh cho phép tùy biến phong cách chơi.
- **Unlocking System**: tiến xa hơn thì mở được khu vực/tính năng mới.

Progression phải được thiết kế có nhịp độ phù hợp để tránh:

- tăng cấp quá nhanh gây nhàm,
- tăng quá chậm gây nản.

2.2.3 Combat Design (Thiết kế chiến đấu)

Combat là trung tâm của Action RPG. Các kiến thức nền gồm:

Pacing (nhịp độ) Combat có thể nhanh hoặc chậm tùy phong cách. Nhịp độ phải phù hợp:

- nhanh $\rightarrow$ nhấn mạnh phản xạ, né tránh,
- chậm $\rightarrow$ thiên về chiến thuật và quan sát.

Frame Data Mỗi đòn đánh được chia thành:

- **Startup**: chuẩn bị ra đòn,
- **Active**: hitbox có hiệu lực,
- **Recovery**: cooldown trước khi ra thêm đòn,
- **Invincibility Frame**: thời gian bất tử khi né.

Hit Detection Kiểm tra va chạm dựa trên:

- hitbox của kỹ năng,
- hurtbox của mục tiêu,
- thời điểm animation.

Enemy Pattern Quái trong Action RPG thường có:

- đòn đánh có báo hiệu (telegraph),
- nhiều phase chiến đấu,
- hành vi lặp theo pattern.

2.2.4 World System và Level Design

Trong game, "level design" quyết định cách người chơi di chuyển và tương tác. Các yếu tố quan trọng:

- bố cục không gian,
- đường di chuyển chính/phụ,
- phân bổ quái, vật phẩm,
- vùng an toàn và điểm checkpoint,
- nhịp độ giữa combat và khám phá.

Nếu level quá rối → người chơi lạc. Nếu quá đơn giản → mất hứng thú.

2.2.5 Hệ thống kinh tế (Game Economy)

Game Economy là kiến thức nền quan trọng trong mọi game RPG:

Nguồn tài nguyên (Resource Sources)

- phần thưởng theo nhiệm vụ,
- quái rơi vật phẩm,
- crafting,
- sự kiện.

Tiêu thụ tài nguyên (Resource Sinks)

- nâng cấp trang bị,
- mua vật phẩm,
- chi phí mở khóa tính năng.

Nếu nguồn thu lớn nhưng tiêu thụ ít → **lạm phát tài nguyên**. Nếu tiêu thụ nhiều nhưng khó kiếm → **người chơi bỏ game**.

2.2.6 UX/UI Design trong game

UX tốt giúp giảm tải nhận thức và tăng trải nghiệm. Các nguyên lý nền:

- **Phản hồi rõ ràng (Visual Feedback)**: hiệu ứng khi nhặt vật phẩm, đánh trúng, nhận thưởng.
- **Giao diện nhất quán**: vị trí nút bấm và layout thống nhất.
- **Giảm tải thông tin**: chỉ hiển thị những gì người chơi cần.

2.2.7 Balancing Fundamentals (Cân bằng game)

Balancing là điều chỉnh sao cho:

- nhân vật không quá mạnh hoặc quá yếu,
- progression hợp lý,
- phần thưởng tương xứng độ khó,
- thời gian hoàn thành nhiệm vụ không quá dài.

Một số khái niệm nền:

- **Linear vs Exponential Growth**,
- **Time-to-Kill (TTK)**,
- **Risk–Reward Ratio**,
- **Power Budget**.

2.3 Tổng kết chương

Chương này đã trình bày toàn bộ các nền tảng kỹ thuật và thiết kế liên quan đến Action RPG và game online. Các kiến thức về real-time gameplay, đồng bộ trạng thái, core loop, combat, progression, economy và UX sẽ được sử dụng làm cơ sở để phân tích yêu cầu và thiết kế hệ thống ở những chương tiếp theo.

3 Công nghệ sử dụng

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

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

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

Các mục sau trình bày chi tiết từng thành phần.

3.1 Kiến trúc tổng quát nhiều lớp

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

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

Việc tách bạch rõ các lớp giúp:

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

3.2 Công nghệ phía client (Game Engine)

3.2.1 Game engine cho Action RPG

Để phát triển nhanh một trò chơi Action RPG có đồ họa 2D/3D, các game engine phổ biến như Unity hoặc Unreal Engine thường được sử dụng. Một số lý do chính:

- Cung cấp sẵn hệ thống **rendering**, animation, vật lý cơ bản, quản lý scene.
- Có **asset pipeline** tương đối hoàn chỉnh (nhập mô hình 3D, texture, animation).
- Hỗ trợ tốt việc xây dựng **UI in-game**, hệ thống input đa nền tảng.
- Có **cộng đồng lớn**, thư viện tài liệu, ví dụ và asset sẵn có.

Trong bối cảnh một dự án đồ án chuyên ngành, việc sử dụng một game engine thương mại (thay vì tự viết engine từ đầu) giúp tiết kiệm thời gian, giảm rủi ro kỹ thuật, và cho phép tập trung vào thiết kế gameplay, networking và kiến trúc hệ thống.

3.2.2 Ngôn ngữ lập trình phía client

Tuỳ game engine, ngôn ngữ lập trình chủ đạo có thể là:

- **C#**: thường dùng với Unity, cú pháp hiện đại, thư viện phong phú, hỗ trợ tốt lập trình hướng đối tượng.
- **C++/Blueprint**: phổ biến trong Unreal Engine, hiệu năng cao, nhưng yêu cầu quản lý bộ nhớ và kinh nghiệm nhiều hơn.

Một số điểm kỹ thuật cần lưu ý khi lập trình phía client:

- Tách biệt **logic gameplay cục bộ** (UI, hiệu ứng) với **logic được server quyết định** (combat, trạng thái chính thức).
- Hạn chế lưu **dữ liệu nhạy cảm** phía client (ví dụ: chỉ số thật để tính sát thương) để giảm nguy cơ gian lận.
- Thiết kế kiến trúc code theo hướng **component-based** hoặc **entity–component–system (ECS)** để dễ mở rộng.

3.3 Công nghệ phía server (Backend)

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

- Dịch vụ xác thực, quản lý người dùng (authentication, account).
- Dịch vụ quản lý phiên chơi (session, room, instance).
- Dịch vụ xử lý logic game (combat, nhiệm vụ, tính điểm).
- Dịch vụ quản lý dữ liệu (đọc/ghi tiến trình người chơi).

3.3.1 Nền tảng backend phổ biến

Một số lựa chọn công nghệ backend thường gặp:

- **Node.js / TypeScript** với các framework như NestJS:
  - Hỗ trợ tốt lập trình bất đồng bộ, phù hợp với hệ thống có nhiều kết nối WebSocket.
  - Hệ sinh thái thư viện phong phú (WebSocket, REST, ORM, JWT, v.v.).
  - TypeScript giúp mã nguồn dễ bảo trì, giảm lỗi kiểu dữ liệu.
- **.NET / C#**:
  - Hiệu năng tốt, tích hợp tốt với môi trường Windows và dịch vụ Microsoft.
  - Phù hợp khi client cũng sử dụng C# (ví dụ: Unity), giúp tái sử dụng mô hình dữ liệu và một số logic phụ trợ.
- **Ngôn ngữ khác** như Java, Go, Rust:
  - Phù hợp cho các hệ thống backend có yêu cầu hiệu năng hoặc độ trễ cực thấp.
  - Tuy nhiên, thời gian học và xây dựng hạ tầng có thể cao hơn đối với nhóm sinh viên.

Trong bối cảnh đồ án, các nền tảng có cộng đồng lớn, tài liệu tốt và hỗ trợ nhanh cho WebSocket (như Node.js/NestJS hoặc .NET) thường là lựa chọn hợp lý.

3.3.2 Mô hình dịch vụ và tổ chức logic backend

Về kiến trúc, backend có thể triển khai theo:

- **Monolithic**: tất cả logic và API nằm trong một ứng dụng duy nhất.
- **Microservices**: chia thành nhiều dịch vụ nhỏ (user service, world service, combat service, v.v.).

Đối với một dự án quy mô đồ án:

- Mô hình monolithic **dễ triển khai**, dễ debug và phù hợp với nhóm nhỏ.
- Tuy nhiên trong thiết kế, vẫn có thể định hướng kiến trúc theo hướng **module hoá**, sẵn sàng tách thành microservices nếu mở rộng trong tương lai.

3.4 Công nghệ networking và giao tiếp thời gian thực

3.4.1 Giao tiếp client–server

Game online sử dụng kết hợp:

- **HTTP/REST**:

- Phù hợp cho các tác vụ không yêu cầu real-time cao như: đăng ký, đăng nhập, cấu hình tài khoản, tải dữ liệu tĩnh.

- **WebSocket**:
  - Dùng cho truyền thông hai chiều thời gian thực: cập nhật vị trí, trạng thái chiến đấu, sự kiện trong thế giới game.
  - Giúp tránh overhead của việc liên tục tạo kết nối HTTP mới.

3.4.2 Quản lý kết nối và phiên chơi

Một backend real-time cần:

- Quản lý **phiên kết nối WebSocket** cho từng người chơi.
- Ánh xạ giữa người chơi và **phòng/instance** trong game (ví dụ: phòng dungeon, map riêng, khu vực chung).
- Xử lý ngắt kết nối, timeout, reconnect.

Khi số lượng người chơi tăng, cần xem xét:

- Cân bằng tải kết nối giữa nhiều server.
- Sử dụng **message broker** hoặc **pub/sub** để đồng bộ sự kiện (ví dụ: Redis pub/sub, Kafka).

3.5 Công nghệ lưu trữ và cơ sở dữ liệu

3.5.1 Cơ sở dữ liệu NoSQL (ví dụ: MongoDB)

Các trò chơi online thường lựa chọn NoSQL, đặc biệt là mô hình document như MongoDB, vì:

- Dữ liệu game có cấu trúc linh hoạt (nhân vật, vật phẩm, trạng thái nhiệm vụ, cấu hình NPC).
- Dễ thay đổi schema khi game được mở rộng.
- Hỗ trợ tốt cho các truy vấn theo document và các trường lồng nhau.

Các nhóm dữ liệu phù hợp với NoSQL:

- Hồ sơ người chơi (tài khoản, danh sách nhân vật).
- Trạng thái nhân vật (level, stat, trang bị).
- Trạng thái thế giới (dữ liệu map riêng, đảo, dungeon).
- Cấu hình game (danh sách skill, class, hệ thống chỉ số).

3.5.2 Cơ sở dữ liệu quan hệ / SQLite

Cơ sở dữ liệu quan hệ hoặc các hệ như SQLite có thể được sử dụng cho:

- Công cụ nội bộ (internal tools).
- Lưu bảng tham chiếu tĩnh (danh mục class, chỉ số mặc định, bảng quy đổi).
- Prototyping nhanh các tính năng.

Ưu điểm:

- Mô hình dữ liệu chặt chẽ, dễ đảm bảo ràng buộc.
- Câu lệnh truy vấn SQL quen thuộc, dễ phân tích.

3.5.3 Hệ thống cache (ví dụ: Redis)

Redis thường được dùng làm:

- **Cache** cho dữ liệu truy cập thường xuyên (profile người chơi đang online, cấu hình skill).
- **Session store**: lưu token, trạng thái đăng nhập.
- **Pub/Sub** để truyền sự kiện giữa nhiều tiến trình hoặc nhiều instance server (phục vụ scale-out).

Việc kết hợp NoSQL + cache giúp:

- giảm tải cho cơ sở dữ liệu chính,
- tăng tốc độ phản hồi trong các thao tác real-time.

3.6 Công cụ hỗ trợ phát triển và vận hành

Ngoài các công nghệ chính dùng cho client, backend và dữ liệu, việc phát triển một trò chơi online hiện đại cần thêm các công cụ hỗ trợ.

3.6.1 Quản lý mã nguồn và làm việc nhóm

- **Git**: hệ thống quản lý phiên bản phân tán, cho phép nhiều thành viên cùng làm việc trên một codebase.
- **Git hosting** (GitHub, GitLab, Bitbucket): cung cấp giao diện web, issue tracking, pull request, CI/CD.

Các thực hành tốt:

- Chia nhánh theo tính năng (feature branch).
- Review code trước khi merge.
- Ghi log commit rõ ràng, nhất quán.

3.6.2 Công cụ thiết kế và tài liệu

- **Công cụ vẽ UML/BPMN**: dùng mô tả use case, luồng xử lý, kiến trúc.
- **Công cụ thiết kế giao diện** (Figma, v.v.): dùng để phác thảo UI before implement.
- **Công cụ quản lý tài liệu** (Markdown, LaTeX): dùng viết GDD, tài liệu thiết kế hệ thống, báo cáo.

3.6.3 Công cụ kiểm thử và giám sát

Ở mức độ đồ án, kiểm thử và giám sát có thể ở mức đơn giản:

- Log server (file log, console).
- Một số script kiểm thử unit và integration cơ bản.

Trong các hệ thống lớn hơn, có thể sử dụng:

- Hệ thống giám sát (Prometheus, Grafana).
- Tracking lỗi (Sentry, v.v.).

3.7 Tổng kết

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