HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

---



**Đinh Thị Hương Thảo**

# NGHIÊN CỨU VÀ ỨNG DỤNG LLM CHO BÀI TOÁN TẠO NHIỆM VỤ (QUEST) TRONG GAME CASUAL

**ĐỀ ÁN TỐT NGHIỆP THẠC SĨ KỸ THUẬT**

***(Theo định hướng ứng dụng)***

**HÀ NỘI - 2025**

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

---



**Đinh Thị Hương Thảo**

# NGHIÊN CỨU VÀ ỨNG DỤNG LLM CHO BÀI TOÁN TẠO NHIỆM VỤ (QUEST) TRONG GAME CASUAL

**Chuyên ngành: Khoa học máy tính**

**Mã số: 8.48.01.01**

**ĐỀ ÁN TỐT NGHIỆP THẠC SĨ KỸ THUẬT**

***(Theo định hướng ứng dụng)***

**NGƯỜI HƯỚNG DẪN KHOA HỌC: TS. ĐẶNG HOÀNG LONG**

**HÀ NỘI - 2025**

# LỜI CAM ĐOAN

Tôi cam đoan đây là công trình nghiên cứu của riêng tôi.

Các số liệu, kết quả nêu trong đề án tốt nghiệp là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác.

**TÁC GIẢ**

**Đinh Thị Hương Thảo**

# LỜI CẢM ƠN

Tôi xin trân trọng bày tỏ lòng biết ơn sâu sắc đến tập thể các thầy, cô giáo Học viện Công nghệ Bưu chính Viễn thông đã tận tình giảng dạy, truyền đạt kiến thức và tạo điều kiện thuận lợi cho tôi trong suốt quá trình học tập và nghiên cứu tại Học viện. Đặc biệt, tôi xin gửi lời cảm ơn chân thành đến thầy giáo Tiến sĩ Đặng Hoàng Long, người đã trực tiếp hướng dẫn tôi hoàn thành đề án tốt nghiệp này. Xin được ghi nhận sự động viên và hỗ trợ từ gia đình, bạn bè và đồng nghiệp trong suốt thời gian thực hiện.

Mặc dù đã cố gắng nhiều song đề án tốt nghiệp khó tránh khỏi những hạn chế nhất định, tôi rất mong nhận được sự góp ý quý báu của quý thầy cô và các nhà nghiên cứu để công trình được hoàn thiện hơn.

Xin trân trọng cảm ơn!

**HỌC VIÊN**

**Đinh Thị Hương Thảo**

# MỤC LỤC

**MỤC LỤC** ..... **iii**
**DANH MỤC CHỮ VIẾT TẮT** ..... **vi**
**DANH MỤC BẢNG** ..... **viii**
**DANH MỤC HÌNH** ..... **ix**
**MỞ ĐẦU** ..... **1**
**CHƯƠNG 1. CƠ SỞ LÝ LUẬN** ..... **4**
**1.1 Game casual và đặc điểm nhiệm vụ (Quest)** ..... **4**
1.1.1 Định nghĩa game casual ..... 4
1.1.2 Đặc trưng của game casual ..... 5
1.1.3 Tâm lý học hành vi người chơi trong game casual ..... 7
1.1.4 Cơ sở thiết kế và phân loại nhiệm vụ trong game casual ..... 8
**1.2 Mô hình ngôn ngữ lớn (LLM)** ..... **10**
1.2.1 Kiến trúc Transformer và mô hình Decoder-only ..... 10
1.2.2 Cơ chế Self-Attention và Cửa sổ ngữ cảnh ..... 11
1.2.3 Cơ chế học trong ngữ cảnh (In-context Learning) ..... 12
1.2.4 Cơ chế tuân thủ chỉ dẫn (Instruction Following & Alignment) ..... 13
1.2.5 Kỹ thuật Prompt Engineering ..... 14
**1.3 Vấn đề nghiên cứu và nhu cầu ứng dụng LLM** ..... **15**
**CHƯƠNG 2. CHUẨN BỊ DỮ LIỆU VÀ HUẤN LUYỆN MÔ HÌNH** ..... **17**
**2.1 Chuẩn bị dữ liệu** ..... **17**
2.1.1 Quy trình thu thập dữ liệu ..... 17
2.1.2 JSON Schema và yêu cầu ràng buộc ..... 20
2.1.3 Bộ tham chiếu gameplay (Gameplay Reference Set) ..... 23
**2.2 Kiến trúc hệ thống và Lựa chọn mô hình** ..... **23**
2.2.1 Sơ đồ kiến trúc tổng thể ..... 23
2.2.2 Lựa chọn mô hình ngôn ngữ ..... 25
2.2.3 So sánh chi phí–hiệu năng ..... 26

**2.3 Quy trình huấn luyện và tối ưu ..... 27**
2.3.1 Quy trình tối ưu hóa Prompt (Prompt Optimization Pipeline)..... 27
2.3.2 Prompt V1: Phiên bản khởi đầu (Baseline Prompt)..... 27
2.3.3 Prompt V2: Bổ sung ràng buộc cấu trúc (Structure-Constrained Prompt) 28
2.3.4 Prompt V3: Tối ưu hoá ngữ cảnh (Final Prompt)..... 29
**2.4 Module kiểm định (Validation Layer) ..... 30**
2.4.1 Kiểm định cấu trúc (Structural Validation)..... 31
2.4.2 Kiểm định logic gameplay (Gameplay Logic Validation)..... 31
2.4.3 Kiểm định khả thi và cân bằng thiết kế (Gameplay Feasibility & Balancing) ..... 32
**2.5 Tích hợp thực nghiệm vào game ..... 33**
2.5.1 Thiết kế và triển khai Quest System theo mô hình ..... 33
2.5.2 Tích hợp API tạo Quest tự động bằng LLM ..... 34
2.5.3 Xây dựng giao diện Quest System trong game..... 35
**CHƯƠNG 3. KIỂM THỬ VÀ KHẢO SÁT NGƯỜI DÙNG ..... 36**
**3.1 Môi trường thực nghiệm ..... 36**
3.1.1 Mục tiêu thực nghiệm ..... 36
3.1.2 Cấu hình thực nghiệm ..... 36
3.1.3 Công cụ đo lường (Evaluation Metrics)..... 37
**3.2 Kết quả sinh nhiệm vụ từ mô hình ..... 37**
3.2.1 Hiệu quả của các phiên bản prompt ..... 38
3.2.2 Thống kê định lượng ..... 39
3.2.3 Phân tích các dạng lỗi điển hình ..... 40
**3.3 Khảo sát người dùng..... 41**
3.3.1 Mục tiêu khảo sát ..... 41
3.3.2 Phương pháp khảo sát ..... 41
3.3.3 Kết quả khảo sát..... 42
**3.4 Đánh giá hiệu quả sản xuất và chi phí ..... 43**

3.4.1 So sánh năng suất: Con người vs AI ..... 43
3.4.2 Phân tích chi phí - hiệu năng (Cost-Performance): ..... 45
**KẾT LUẬN ..... 46**
**DANH MỤC TÀI LIỆU THAM KHẢO ..... 49**
**PHỤ LỤC ..... 51**

# DANH MỤC CHỮ VIẾT TẮT

| Viết tắt | Tiếng Anh | Tiếng Việt / Giải thích |
|---|---|---|
| **AI** | Artificial Intelligence | Trí tuệ nhân tạo |
| **API** | Application Programming Interface | Giao diện lập trình ứng dụng |
| **BERT** | Bidirectional Encoder Representations from Transformers | Mô hình biểu diễn mã hóa hai chiều từ Transformer |
| **CoT** | Chain-of-Thought | Chuỗi suy luận (kỹ thuật gợi ý) |
| **DDA** | Dynamic Difficulty Adjustment | Điều chỉnh độ khó động [1] |
| **GPT** | Generative Pre-trained Transformer | Mô hình Transformer tạo sinh được huấn luyện trước |
| **IAP** | In-App Purchase | Mua hàng trong ứng dụng |
| **ICL** | In-context Learning | Cơ chế học trong ngữ cảnh |
| **JSON** | JavaScript Object Notation | Định dạng trao đổi dữ liệu đối tượng JavaScript |
| **LLM** | Large Language Model | Mô hình ngôn ngữ lớn |
| **LoRA** | Low-Rank Adaptation | Thích ứng hạng thấp (Kỹ thuật tinh chỉnh mô hình) |
| **LSTM** | Long Short-Term Memory | Mạng bộ nhớ ngắn-dài hạn |
| **MOBA** | Multiplayer Online Battle Arena | Đấu trường trận chiến trực tuyến nhiều người chơi |

| | | |
|---|---|---|
| **NLP** | Natural Language Processing | Xử lý ngôn ngữ tự nhiên |
| **QLoRA** | Quantized Low-Rank Adaptation | Thích ứng hạng thấp lượng tử hóa |
| **RLHF** | Reinforcement Learning from Human Feedback | Học tăng cường từ phản hồi của con người |
| **RNN** | Recurrent Neural Network | Mạng nơ-ron hồi quy |
| **ROI** | Return on Investment | Tỷ suất hoàn vốn |
| **RPG** | Role-Playing Game | Trò chơi nhập vai |
| **UI** | User Interface | Giao diện người dùng |

# DANH MỤC BẢNG

Bảng 2-1 Mô tả dữ liệu ứng với mỗi nhiệm vụ ..... 21
Bảng 2-2 Bảng chú giải sơ đồ pipeline hệ thống ..... 24
Bảng 2-3 Thành phần và kỹ thuật áp dụng trong Prompt V3 ..... 29
Bảng 3-1 Bảng so sánh 3 cấp độ prompt ..... 38
Bảng 3-2 Bảng tổng hợp hiệu suất kỹ thuật giữa các phiên bản Prompt ..... 39
Bảng 3-3 Bảng tổng hợp kết quả đánh giá trải nghiệm người dùng (Thang đo likert 1-5) ..... 42
Bảng 3-4 Bảng so sánh hiệu suất giữa con người và AI trong việc tạo nhiệm vụ game (Thang đo likert 1-5) ..... 44
Bảng A.1-1 Từ điển hoa quả/quái vật trong game ..... 51
Bảng A.1-2 Từ điển vật phẩm hỗ trợ trong game ..... 52
Bảng A.1-3 Từ điển hành động trong game ..... 52
Bảng A.1-4 Bộ quy tắc cân bằng Phần thưởng – Độ khó trong game ..... 53
Bảng A.1-5 Các giới hạn Gameplay (Gameplay Constraints) ..... 54
Bảng A.3-1 Các phiên bản Prompt ..... 56
Bảng A.4-1 Đầu ra của các phiên bản prompt ..... 66
Bảng A.5-1 Bảng tổng hợp lỗi giữa các phiên bản Prompt ..... 69
Bảng A.7-1 Bảng hỏi đánh giá các nội dung Quest được tạo bởi LLM ..... 73
Bảng A.8-1 Bảng so sánh về Chi phí và Hiệu năng khi sử dụng LLM để tạo quest 76

# DANH MỤC HÌNH

Hình 1-1 Một số tựa game casual nổi tiếng ..... 5
Hình 1-2 Biểu đồ so sánh hời gian chơi trung bình mỗi session theo thể loại game [Nguồn: GameAnaytics.com] ..... 6
Hình 1-3 Sơ đồ vòng lặp game tác động tới hành vi người chơi ..... 7
Hình 1-4 Hệ thống nhiệm vụ trong game casual ..... 9
Hình 1-5 Kiến trúc mô hình transformer ..... 11
Hình 1-6 Công thức Attention ..... 11
Hình 1-7 Kiến trúc In-context learning architecture ..... 12
Hình 2-1 Sơ đồ quy trình thu thập dữ liệu ..... 17
Hình 2-2 Ảnh nội dung trò chơi merge ..... 18
Hình 2-3 Cấu trúc JSON Schema của nhiệm vụ ..... 22
Hình 2-4 Sơ đồ pipeline hệ thống tích hợp LLM vào Unity Engine ..... 24
Hình 2-5 Biểu đồ so sánh tốc độ sinh token (Output Tokens/s) giữa các mô hình LLM phổ biến [Nguồn: Artificial Analysis.ai, ngày 15/6/2025]. ..... 25
Hình 2-6 Sơ đồ mô hình MVC triển khai Quest System ..... 33
Hình 2-7 Biểu đồ luồng hoạt động tích hợp LLM cho trò chơi ..... 34
Hình 2-8 Giao diện Quest System trong game ..... 35
Hình 3-1 Biểu đồ tương quan kết quả đánh giá trải nghiệm người dùng ..... 42
Hình A.2-1 Workflow triển khai huấn luyện LLM và backend bằng công cụ Dify 55
Hình A.6-1 Giao diện câu hỏi trò chơi được tạo ra bởi LLM đã tích hợp vào game 72

# MỞ ĐẦU

## 1. Lý do chọn đề tài

Theo các báo cáo thị trường trò chơi di động (game mobile) gần đây, thể loại game casual tiếp tục chiếm tỷ trọng lớn trong ngành nhờ đặc điểm dễ học, dễ chơi và phù hợp với hành vi tiêu thụ nội dung nhanh (snackable content), quỹ thời gian phân mảnh của người dùng hiện đại. Báo cáo Global Games Market 2023 của Newzoo cho thấy game mobile chiếm hơn 50% doanh thu ngành game toàn cầu, với sự tăng trưởng mạnh mẽ của nhóm casual/hyper-casual [15].

Tại Việt Nam, theo báo cáo Vietnam's Game-Winning Era của Xsolla, quy mô thị trường game đã tăng từ 192 triệu USD (2017) lên gần 1.5 tỷ USD (2024), đưa Việt Nam trở thành một trong những quốc gia có tốc độ phát triển game nhanh nhất khu vực và đứng đầu thế giới về lượng tải xuống game mobile [17].

Trong trò chơi thông thường (game casual), đặc biệt là các trò chơi vận hành theo mô hình live-service, một trong những thách thức lớn nhất là tạo mới nội dung liên tục, bao gồm các hoạt động như: thiết kế nhiệm vụ hằng ngày (Daily Quest), tạo thử thách (Challenge), xây dựng sự kiện theo mùa (Seasonal Events), duy trì động lực quay lại của người chơi (Retention). Theo nghiên cứu hành vi người chơi của Juul (2010) và Kumar (2025), nội dung nhiệm vụ (quest) đóng vai trò quan trọng trong việc kích hoạt các vòng lặp phản hồi thần kinh (dopamine feedback loop) – cơ chế tâm lý cốt lõi quyết định đến tỷ lệ duy trì (retention rate) và thời gian gắn bó của người dùng [2][3].

Tuy nhiên, quá trình tạo nhiệm vụ hiện nay phần lớn được thực hiện thủ công bởi đội ngũ thiết kế game, dẫn đến nhiều hạn chế như khả năng mở rộng (scalability) thấp, khó đáp ứng nhu cầu nội dung khổng lồ trong thời gian ngắn, nhiệm vụ dễ lặp lại, thiếu tính cá nhân hóa, tốn thời gian vận hành và chi phí nhân sự hay khó đáp ứng sự đa dạng về phong cách chơi trong game casual.

Với sự phát triển của mô hình ngôn ngữ lớn (LLM) như GPT, Claude, LLaMA mở ra khả năng tự động hóa việc tạo nội dung game. Các nghiên cứu về Transformer

[5], attention-based generation [6], few-shot prompting [14]…cho thấy LLM có thể tạo ra nội dung chất lượng cao nếu được hướng dẫn đúng bằng Prompt Engineering.

Từ bối cảnh thực tiễn và nhu cầu của ngành công nghiệp game toàn cầu nói chung và Việt Nam nói riêng, đề tài **"Nghiên cứu và ứng dụng LLM cho bài toán tạo nhiệm vụ (Quest) trong Game Casual"** được lựa chọn nhằm giải quyết bài toán tự động hóa nội dung, giảm chi phí vận hành, tăng tính đa dạng và cá nhân hóa nhiệm vụ, từ đó thúc đẩy ứng dụng AI trong ngành sáng tạo số – một định hướng đang phát triển mạnh mẽ tại Việt Nam.

## 2. Mục đích nghiên cứu

Mục tiêu tổng quát: nghiên cứu và ứng dụng các kỹ thuật prompt engineering trong việc huấn luyện mô hình tạo các nhiệm vụ trò chơi

Mục tiêu cụ thể:

(i) Nghiên cứu cơ sở lý thuyết về game casual, hành vi người chơi, mô hình ngôn ngữ lớn và các kỹ thuật Prompt Engineering.

(ii) Xây dựng pipeline sinh nhiệm vụ tự động cho game casual dựa trên LLM và lớp kiểm định (validation) đảm bảo nhiệm vụ hợp lệ, đa dạng và khả thi trong gameplay.

(iii) Kiểm thử và đánh giá chất lượng nhiệm vụ tạo sinh thông qua kết quả thực nghiệm và khảo sát người dùng.

## 3. Ý nghĩa khoa học và thực tiễn

*Ý nghĩa khoa học*

Đề tài áp dụng các kỹ thuật tiên tiến của lĩnh vực NLP như Transformer Attention [5], schema-constrained generation [6], in-context learning [14]… vào một bài toán mới: tạo nhiệm vụ game casual.

*Ý nghĩa thực tiễn*

Đề tài mang ý nghĩa thực tiễn cao, giúp giảm tải khối lượng công việc lặp lại cho đội ngũ thiết kế game; tạo ra các nhiệm vụ nhanh, đa dạng, hạn chế trùng lặp, phù hợp gameplay. Với khả năng, đề tài có tiềm năng mở rộng ứng dụng sang nhiều thể loại game hoặc lĩnh vực sáng tạo số như giáo dục, mô phỏng, gamification doanh

nghiệp, từ đó thúc đẩy việc ứng dụng AI trong ngành công nghiệp game Việt Nam – một ngành đang tăng trưởng mạnh [17].

## 4. Đối tượng và phạm vi nghiên cứu

*Đối tượng nghiên cứu*

Mô hình ngôn ngữ lớn (LLM) và khả năng tạo nhiệm vụ game casual.

*Phạm vi nghiên cứu*

Đề tài có nghiên cứu trong phạm vi:

(i) Game casual trên nền tảng di động.

(ii) Nội dung nhiệm vụ (quests) dưới dạng JSON, không nghiên cứu sinh level, hình ảnh, hội thoại.

(iii) Mô hình dùng phương pháp Prompt Engineering, không fine-tuning mô hình.

(iv) Thời gian thực hiện: năm học 2025.

## 5. Phương pháp nghiên cứu

Các phương pháp nghiên cứu áp dụng trong đề tài:

Phương pháp lý thuyết: tổng hợp, phân tích tài liệu (thu thập và phân tích lý thuyết về game casual, kiến trúc Transformer, các kỹ thuật Prompt Engineering…)

Phương pháp thực nghiệm: chạy thử nghiệm nhiều lượt tạo nhiệm vụ theo các kịch bản gameplay thực tế, đánh giá các câu hỏi tạo ra bằng Validation Layer; đánh giá chất lượng nhiệm vụ sinh ra theo tiêu chí cụ thể.

Phương pháp khảo sát – đánh giá: tổ chức khảo sát người dùng (người chơi hoặc nhà phát triển game) để đánh giá tính lợp lý của các nhiệm vụ sinh ra

Các công công cụ sử dụng: Dify, mô hình LLM (GPT/LLaMA API), Unity, Google Forms…

# CHƯƠNG 1. CƠ SỞ LÝ LUẬN

Chương này tập trung trình bày các kiến thức nền tảng cho toàn bộ đề tài. Đầu tiên, nghiên cứu sẽ giới thiệu vắn tắt đặc điểm game casual và hệ thống nhiệm vụ. Kế đến, chương cũng trình bày tổng quan về mô hình ngôn ngữ lớn và các kỹ thuật sinh nội dung. Cuối cùng, chương tiếp tục đưa ra những vấn đề còn tồn tại trong thiết kế nhiệm vụ và xác định nhu cầu ứng dụng LLM. Những nội dung này đặt cơ sở cho việc xây dựng quy trình sinh nhiệm vụ tự động ở CHƯƠNG 2.

## 1.1 Game casual và đặc điểm nhiệm vụ (Quest)

### *1.1.1 Định nghĩa game casual*

Trò chơi phổ thông, hay còn gọi là game casual, là một phân khúc đặc thù của trò chơi điện tử được định nghĩa bởi sự tối giản trong cơ chế vận hành và tính tiếp cận rộng rãi đối với đa dạng đối tượng người dùng. Về mặt thuật ngữ, game casual không chỉ đơn thuần chỉ một thể loại trò chơi cố định, mà là một phạm trù bao quát các sản phẩm giải trí số có rào cản gia nhập thấp (low barrier to entry). Đặc điểm cốt lõi của dòng game này nằm ở "tính khả dụng" (accessibility), nơi người chơi không cần sở hữu các kỹ năng thao tác phức tạp hay kiến thức chuyên môn về trò chơi trước đó để có thể bắt đầu.

Xét trên phương diện cấu trúc trải nghiệm, khái niệm này gắn liền với các vòng lặp trò chơi ngắn (short game loops), cho phép người dùng tham gia vào các phiên trải nghiệm có tính phân đoạn cao, phù hợp với xu hướng giải trí trong các khoảng thời gian trống. Khác với dòng game truyền thống (Hardcore Game) vốn đòi hỏi sự cam kết cao về thời gian và cường độ tập trung, game casual chú trọng vào việc tạo ra phản hồi tích cực tức thời thông qua hệ thống phần thưởng trực quan và âm thanh sinh động. Do đó, về mặt bản chất, đây là sự giao thoa giữa nghệ thuật thiết kế tương tác và tâm lý học hành vi, nhằm tối ưu hóa cảm giác thư giãn và giảm thiểu áp lực tâm lý cho người chơi trong suốt quá trình trải nghiệm.

### *1.1.2 Đặc trưng của game casual*

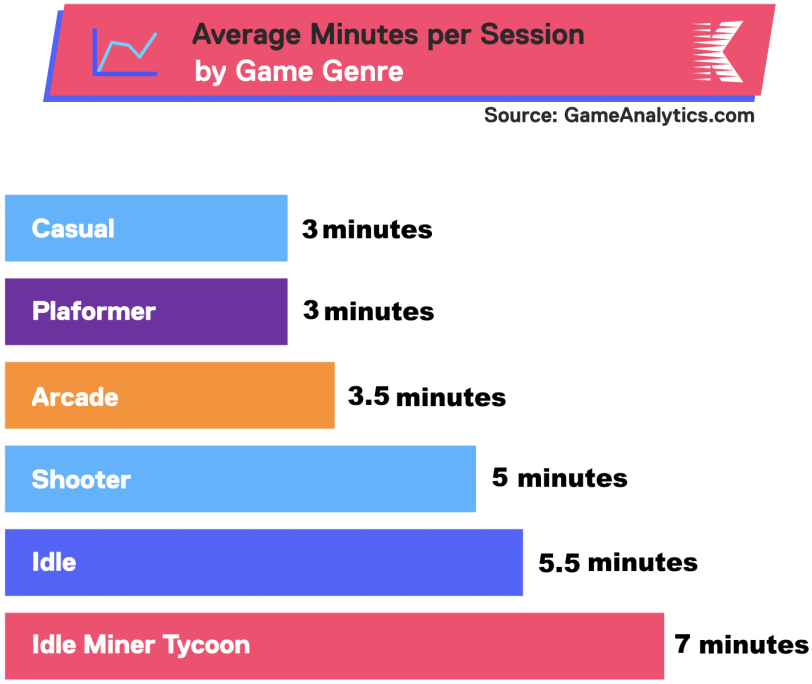
Game casual được xem là một trong những phân khúc phát triển mạnh nhất của thị trường game toàn cầu, hướng đến những trải nghiệm ngắn, nhanh và dễ tiếp cận. Theo báo cáo Vietnam's Game-Winning Era của Xsolla (2025), và Việt Nam liên tục nằm trong nhóm quốc gia xuất khẩu game mobile hàng đầu thế giới, với ~6.7 tỷ lượt tải năm 2025 [17]. Các studio Việt như Amanotes, ABI Games, Falcon Game Studio, VNG vẫn đang dẫn đầu Đông Nam Á trong mảng casual. Điều này cho thấy nhu cầu phát triển nội dung nhanh – đặc biệt là nhiệm vụ (quest/daily task) – trở thành yếu tố sống còn của các studio game Việt Nam.




**Hình 1-1 Một số tựa game casual nổi tiếng**

Lý do game casual có thể chiếm lĩnh thị trường game tỷ đô và cạnh tranh với những dòng game khác trên thị trường do chúng có những đặc trưng nổi bật sau đây.

*Thứ nhất,* game casual được thiết kế với luật chơi đơn giản, thời gian học ngắn và vòng lặp gameplay rõ ràng. Những đặc điểm này giúp game casual tiếp cận lượng lớn người dùng và tạo ra quãng chơi ngắn, lặp lại nhiều lần trong ngày [2]. Theo Li (2024), gameplay của game casual xoay quanh các hành động nhỏ được lặp liên tục. khi các thao tác đơn giản kết hợp với phần thưởng nhỏ nhưng đều đặn (micro-rewards) tạo nên chu kỳ dopamine—yếu tố then chốt giữ chân người chơi quay lại trò chơi hằng ngày với mọi lứa tuổi [1] [3].




**Hình 1-2 Biểu đồ so sánh hời gian chơi trung bình mỗi session theo thể loại game [Nguồn: GameAnaytics.com]**

*Thứ hai,* bên cạnh đó, game casual thường sở hữu luật chơi đơn giản và trực quan. Cơ chế điều khiển thường được tối ưu hoá cho thiết bị di động: chạm, kéo – thả, vuốt hoặc nhấn nhanh. Việc đơn giản hóa thao tác không chỉ giúp giảm rào cản tiếp cận mà còn tăng tốc độ đưa người chơi vào trải nghiệm cốt lõi (core gameplay). Khi các thao tác đơn giản kết hợp với phần thưởng nhỏ nhưng đều đặn (micro-rewards) tạo nên chu kỳ dopamine—yếu tố then chốt giữ chân người chơi quay lại trò chơi hằng ngày [3].

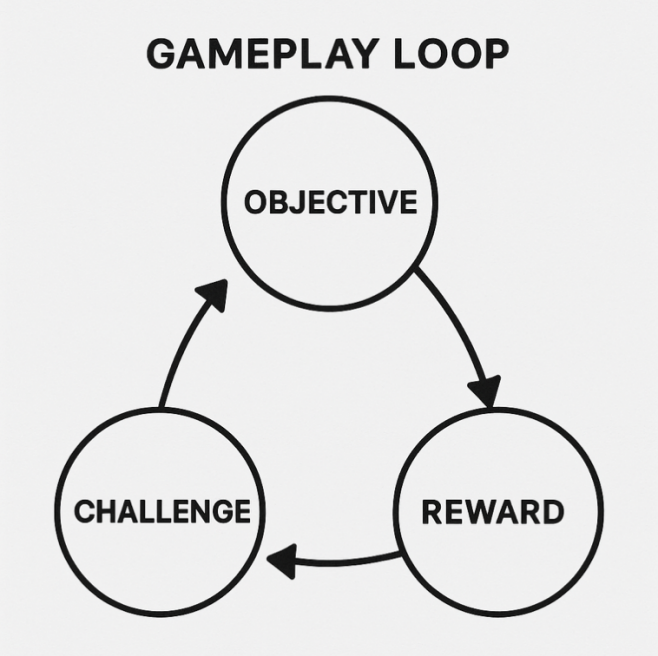
*Thứ ba*, game casual thường được thiết kế theo hướng linh hoạt và dễ mở rộng nội dung. Nhà phát triển có thể thêm nhiệm vụ mới, sự kiện theo mùa hoặc chế độ chơi đặc biệt mà không làm thay đổi lõi trò chơi. Điều này phù hợp với mô hình vận hành game live-service, nơi nội dung phải liên tục được bổ sung để duy trì tệp người chơi lâu dài. Tuy nhiên, chính yêu cầu mở rộng liên tục này lại khiến các studio phải đầu tư nhiều thời gian vào công việc tạo nội dung—một vấn đề mà LLM có tiềm năng giải quyết hiệu quả.

*Cuối cùng,* game casual cũng thường chú trọng vào tính phù hợp đại chúng (mass appeal), thể hiện ở phong cách đồ họa dễ thương, màu sắc tươi sáng, âm thanh đơn giản và tính giải trí cao. Những yếu tố này đảm bảo trò chơi phù hợp với nhiều đối tượng khác nhau và không gây áp lực cho người chơi.

Những đặc điểm này không chỉ giải thích vì sao game casual phổ biến mà còn tạo nền tảng cho quá trình thiết kế nhiệm vụ và ứng dụng LLM vào việc sinh nội dung phù hợp với hành vi người chơi, sao cho đáp ứng và tăng cường thế mạnh của thể loại game này.

### *1.1.3 Tâm lý học hành vi người chơi trong game casual*

Các công trình nghiên cứu về game casual đều nhấn mạnh rằng mọi cơ chế nhiệm vụ phải phục vụ core loop chính. Theo Juul (2010) [2] và Li (2024) [1], core loop có thể mô tả như sau:




**Hình 1-3 Sơ đồ vòng lặp game tác động tới hành vi người chơi**

Trong đó:

(i) Objective (Mục tiêu): Việc người chơi cần hoàn thành trong mỗi lượt, thường đơn giản và rõ ràng.

(ii) Challenge (Thử thách): Trở ngại hoặc mức độ khó khiến người chơi phải nỗ lực để đạt mục tiêu.

(iii) Reward (Phần thưởng): Kết quả tích cực người chơi nhận được sau khi vượt qua thử thách, thúc đẩy họ tiếp tục vòng lặp tiếp theo.

Việc thúc đẩy người chơi thông qua hệ thống nhiệm vụ góp phần tăng sức hấp dẫn cho gameplay cốt lõi. Nghiên cứu của Kumar & M.N (2025) chỉ ra rằng retention trong game casual chủ yếu đến từ "dopamine loops" – vòng phản hồi nhanh, nơi người chơi được thưởng dopamine mỗi khi hoàn thành một hành động nhỏ [3]. Những vòng lặp dopamine nhỏ nhưng liên tục này là nền tảng của trò chơi casual thành công.

Theo Juul (2010), game casual vận hành dựa trên triết lý "easy to learn, hard to master", với hành động cốt lõi được lặp lại liên tục [2]. Vì vậy, việc thiết kế nhiệm vụ hướng tới mục tiêu: dễ hiểu, dễ thực hiện và phù hợp vòng lặp hành vi hiện tại của trò chơi

### *1.1.4 Cơ sở thiết kế và phân loại nhiệm vụ trong game casual*

Trong hệ sinh thái game casual, hệ thống nhiệm vụ (Quest System) không chỉ đơn thuần là danh sách các việc cần làm, mà đóng vai trò là cơ chế then chốt để duy trì vòng đời người chơi (User Lifecycle) và tối ưu hóa chỉ số giữ chân (Retention). Các nghiên cứu về tâm lý học hành vi trong trò chơi đã chỉ ra rằng, nhiệm vụ tác động trực tiếp đến người chơi thông qua ba cơ chế chính: (1) Định hướng hành vi (Behavior Shaping) – dẫn dắt người chơi thực hiện các thao tác mong muốn; (2) Thiết lập mục tiêu ngắn hạn (Short-term Goal Setting) – tạo động lực tức thời trong các phiên chơi ngắn; và (3) Kích thích sự quay lại (Return Incentive) – tạo thói quen đăng nhập hằng ngày thông qua phần thưởng.

Xét dưới góc độ thiết kế hệ thống, Li (2024) [1] nhận định rằng nhiệm vụ trong game casual thực chất là một phần mở rộng của vòng lặp cốt lõi (Core Loop Extension). Chức năng của nó là bao bọc lấy các tương tác cơ bản (như thả trái cây, hợp nhất) để mang lại ý nghĩa và ngữ cảnh cụ thể cho những hành động lặp đi lặp lại, từ đó ngăn chặn cảm giác nhàm chán.

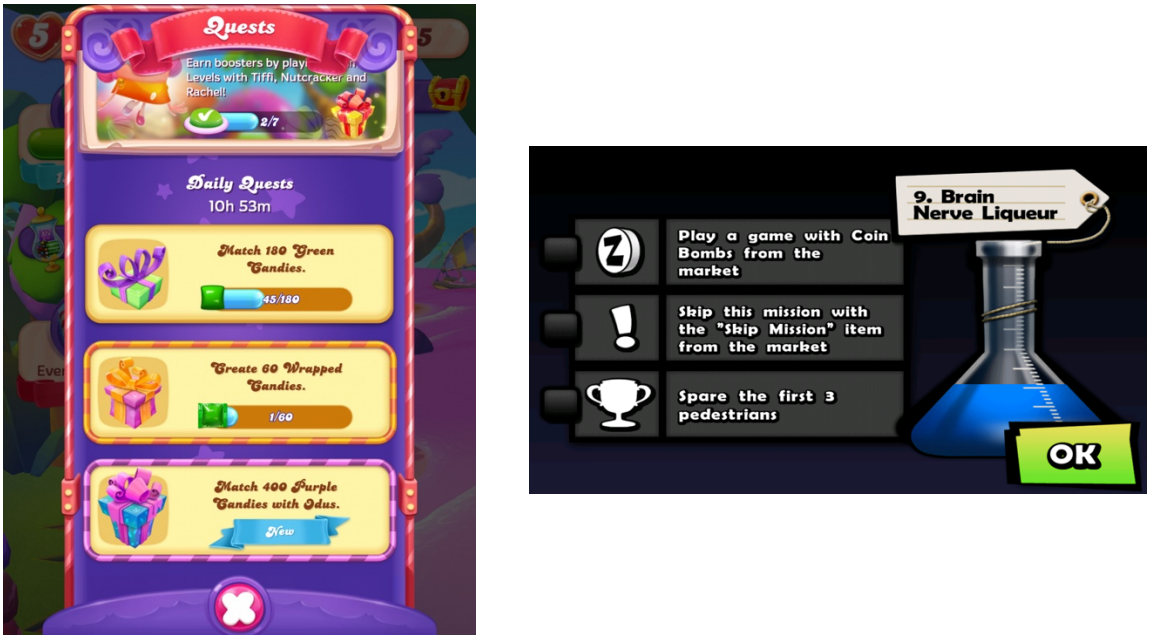
Hệ thống nhiệm vụ thường được phân loại thành ba nhóm chức năng nền tảng, bao quát toàn bộ trải nghiệm của người chơi.

Đầu tiên là nhóm nhiệm vụ định hướng hành động (Action-oriented). Các nhiệm vụ tập trung vào việc thúc đẩy các tương tác vật lý trực tiếp với trò chơi (ví dụ: thực hiện số lần hợp nhất, tạo ra các combo). Đây là nhóm nhiệm vụ giữ nhịp độ chơi (Pacing).

Thứ hai là nhóm nhiệm vụ ghi nhận tiến trình (Progression-oriented). Nhóm này tập trung vào các cột mốc thành tựu (ví dụ: đạt cấp độ quái vật cao nhất, phá kỷ lục điểm số), từ đó tạo ra cảm giác thỏa mãn và thăng tiến (Sense of Achievement).

Thứ ba là nhóm nhiệm vụ điều tiết tài nguyên (Resource-spending). Đây là nhóm nhiệm vụ khuyến khích người chơi tiêu thụ các tài nguyên tích lũy (ví dụ: sử dụng vật phẩm hỗ trợ, chi tiêu tiền vàng), đóng vai trò quan trọng trong việc cân bằng nền kinh tế trong game (Game Economy Balance).

Cấu trúc phân loại này hoàn toàn tương thích với đặc thù vận hành của dòng game casual nói chung và tựa game Merge Fruit nói riêng, nơi trải nghiệm người dùng xoay quanh việc tối ưu hóa các nước đi, quản lý tài nguyên hỗ trợ và chinh phục các giới hạn điểm số mới. Do đó, việc mô hình hóa hệ thống nhiệm vụ theo cấu trúc này là cơ sở quan trọng để xây dựng dữ liệu huấn luyện cho mô hình ngôn ngữ lớn ở các chương sau.




**Hình 1-4 Hệ thống nhiệm vụ trong game casual**

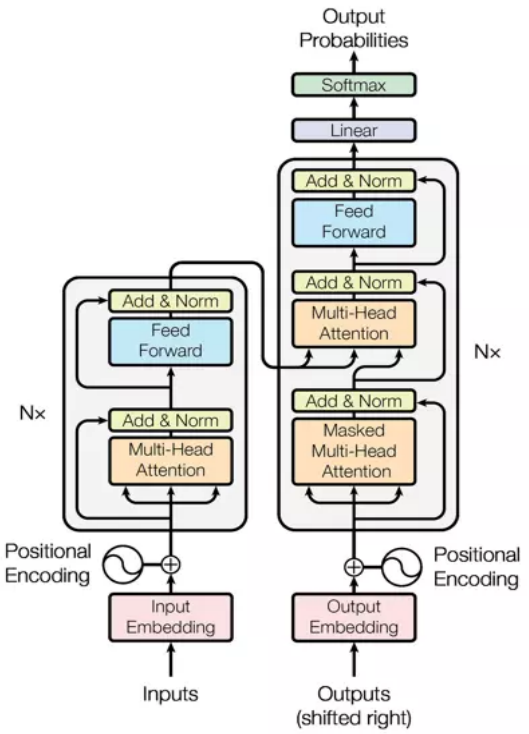
Điều này phù hợp hoàn toàn với nhu cầu của game casual, nơi người chơi liên tục thực hiện các hành động gameplay chính, dùng vật phẩm hỗ trợ đặc biệt hoặc thu thập tiền tệ trong trò chơi.

## 1.2 Mô hình ngôn ngữ lớn (LLM)

### *1.2.1 Kiến trúc Transformer và mô hình Decoder-only*

Nền tảng của hầu hết các mô hình ngôn ngữ lớn hiện nay là kiến trúc Transformer, được giới thiệu lần đầu bởi Vaswani et al. (2017) trong công trình "Attention Is All You Need" [5]. Khác với các mạng nơ-ron hồi quy (RNN) hay LSTM truyền thống vốn xử lý dữ liệu tuần tự, Transformer loại bỏ hoàn toàn các kết nối hồi quy và thay thế bằng cơ chế chú ý (attention mechanism), cho phép xử lý dữ liệu song song và nắm bắt các phụ thuộc xa trong chuỗi văn bản hiệu quả hơn đáng kể [6].

Dựa trên kiến trúc này, sự phát triển của LLM phân tách thành hai nhánh chính: Encoder-only (tiêu biểu là BERT) và Decoder-only (tiêu biểu là GPT). Theo nghiên cứu của Devlin et al. (2019), BERT sử dụng cơ chế mã hóa hai chiều (bidirectional) để tối ưu hóa khả năng hiểu ngữ cảnh, phù hợp cho các tác vụ phân loại hoặc tìm kiếm [8], [10]. Ngược lại, dòng mô hình GPT (Generative Pre-trained Transformer) do Radford et al. (2018) phát triển sử dụng kiến trúc Decoder-only với cơ chế tự hồi quy (autoregressive) [7], [9]. Trong cơ chế này, mô hình dự đoán token tiếp theo dựa trên xác suất của chuỗi token đi trước theo một chiều (unidirectional). Chính đặc tính này biến kiến trúc Decoder-only trở thành lựa chọn tối ưu cho các tác vụ tạo sinh (generative tasks), bao gồm việc sinh mã nguồn và dữ liệu cấu trúc JSON cho trò chơi, nơi yêu cầu sự liền mạch và khả năng mở rộng nội dung sáng tạo.




**Hình 1-5 Kiến trúc mô hình transformer**

### *1.2.2 Cơ chế Self-Attention và Cửa sổ ngữ cảnh*

Sức mạnh cốt lõi giúp LLM duy trì được logic của trò chơi và tuân thủ các cấu trúc dữ liệu phức tạp nằm ở cơ chế Self-Attention. Theo Vaswani et al. (2017), cơ chế này cho phép mỗi phần tử trong chuỗi đầu vào (input sequence) tương tác với tất cả các phần tử khác để tính toán trọng số quan trọng, bất kể khoảng cách vị trí giữa chúng [5]. Cụ thể, thông qua ba vector đại diện là Query (Q), Key (K) và Value (V), mô hình có thể xác định mức độ liên quan giữa các từ khóa trong chỉ dẫn (prompt) với dữ liệu cần sinh ra.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

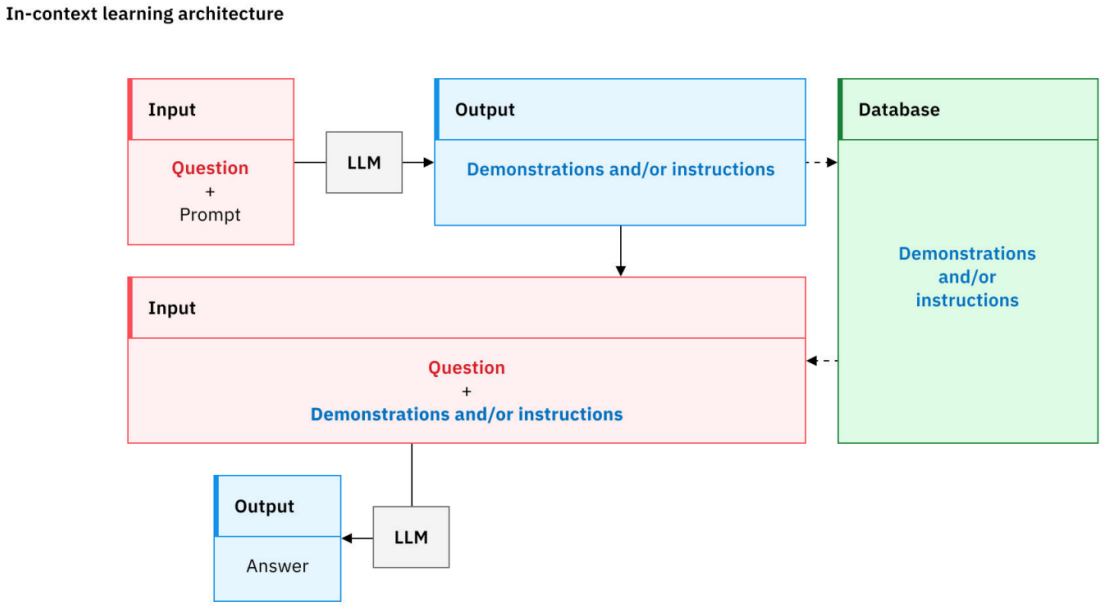
**Hình 1-6 Công thức Attention**

Trong ngữ cảnh của đề tài, cơ chế này đóng vai trò quyết định trong việc xử lý "Cửa sổ ngữ cảnh" (Context Window). Nhờ khả năng mô hình hóa các mối quan hệ phụ thuộc dài (long-range dependencies) được cải thiện so với RNN [7], mô hình

có thể ghi nhớ và liên kết các ràng buộc nằm rải rác trong prompt – ví dụ như quy tắc cân bằng game nằm ở đầu prompt và yêu cầu định dạng JSON nằm ở cuối prompt. Điều này cho phép thực hiện chiến lược "Nạp ngữ cảnh" (Context Injection), nơi toàn bộ bộ tham chiếu gameplay và luật chơi được nạp vào bộ nhớ ngắn hạn của mô hình để định hướng quá trình sinh nhiệm vụ.

### *1.2.3 Cơ chế học trong ngữ cảnh (In-context Learning)*

Một trong những bước tiến quan trọng nhất của các mô hình LLM hiện đại là khả năng thực hiện nhiệm vụ mới mà không cần cập nhật trọng số mạng nơ-ron (gradient updates), được gọi là Học trong ngữ cảnh (In-context Learning - ICL). Theo công trình nền tảng của Brown et al. (2020), các mô hình như GPT-3 thể hiện năng lực học tập "few-shot" mạnh mẽ: chỉ cần cung cấp một vài ví dụ minh họa hoặc chỉ dẫn tự nhiên trong ngữ cảnh đầu vào, mô hình có thể suy luận và thực hiện tác vụ tương ứng với độ chính xác cao [14].




**Hình 1-7 Kiến trúc In-context learning architecture**

Cơ chế này đặt nền móng lý thuyết cho việc lựa chọn giải pháp Prompt Engineering thay vì Tinh chỉnh mô hình (Fine-tuning). Mặc dù các phương pháp tinh chỉnh hiệu quả như LoRA [11] hay QLoRA [12] cho phép cập nhật mô hình với chi phí thấp hơn truyền thống, nhưng chúng vẫn yêu cầu tập dữ liệu huấn luyện lớn và

hạ tầng tính toán chuyên dụng. Ngược lại, In-context Learning cho phép hệ thống game tích hợp và điều chỉnh logic sinh nhiệm vụ một cách linh hoạt, tiết kiệm chi phí và phù hợp với quy trình phát triển nhanh (agile) của các dự án game casual, nơi luật chơi và cân bằng số liệu thay đổi thường xuyên.

### *1.2.4 Cơ chế tuân thủ chỉ dẫn (Instruction Following & Alignment)*

Bên cạnh khả năng sinh văn bản mạch lạc, năng lực cốt lõi giúp các mô hình ngôn ngữ lớn hiện đại (như GPT-4, Gemini, Claude) có thể giải quyết các tác vụ cụ thể là khả năng tuân thủ chỉ dẫn (Instruction Following). Theo Ouyang et al. (2022), các mô hình ngôn ngữ thuần túy (base models) chỉ được tối ưu hóa để dự đoán từ tiếp theo trên tập dữ liệu internet, do đó chúng thường gặp khó khăn trong việc thực hiện các mệnh lệnh của người dùng một cách chính xác [19]. Để khắc phục điều này, các kỹ thuật tinh chỉnh như Instruction Tuning (Wei et al., 2021) và Học tăng cường từ phản hồi con người (RLHF) được áp dụng nhằm "căn chỉnh" (align) mô hình với ý định của con người [20].

Cơ chế tuân thủ chỉ dẫn có vai trò đặc biệt quan trọng trong các ứng dụng yêu cầu đầu ra định dạng nghiêm ngặt, như sinh mã, sinh dữ liệu JSON, hoặc xử lý biểu mẫu. Trong khuôn khổ đề tài này, nó là nền tảng để triển khai các ràng buộc kỹ thuật trong prompt. Cụ thể, nó cho phép mô hình hiểu và thực thi các quy định như: chỉ trả về chuỗi JSON hợp lệ theo đúng schema đã định; không tạo ra giá trị nằm ngoài giới hạn enum; và loại trừ các thành phần không tồn tại trong gameplay (ví dụ: không sinh quái vật vượt quá cấp độ 11). Những yêu cầu này không thể được thỏa mãn chỉ bằng kiến thức ngữ nghĩa, mà cần đến khả năng tuân thủ chính xác cấu trúc định dạng, phản ánh rõ năng lực alignment của mô hình sau khi được instruction-tuned.

Nếu thiếu cơ chế này, LLM sẽ có xu hướng tiếp tục sinh văn bản tự do (freeform text) thay vì bám sát cấu trúc dữ liệu JSON, dẫn đến việc không thể tích hợp vào các hệ thống phần mềm tự động hóa. Do đó, Instruction Following không chỉ là điều kiện cần để cải thiện khả năng tương tác của LLM với con người, mà còn là điều kiện đủ để hiện thực hóa các hệ thống sinh dữ liệu có cấu trúc, như trong ứng dụng tạo nhiệm vụ cho trò chơi điện tử. Các kết quả từ Ouyang et al. (2022) và Wei

et al. (2021) cho thấy hiệu quả rõ rệt của việc căn chỉnh mô hình qua instruction tuning đối với chất lượng đầu ra trong các tác vụ có cấu trúc [19][20].

### *1.2.5 Kỹ thuật Prompt Engineering*

Prompt Engineering được hiểu là một phương pháp tiếp cận kỹ thuật nhằm tối ưu hóa nội dung đầu vào để điều khiển hành vi của mô hình ngôn ngữ lớn (LLM), đóng vai trò như một giải pháp huấn luyện phi tham số trong các hệ thống trí tuệ nhân tạo. Theo các nghiên cứu của Lin et al. (2022) và Brown et al. (2020), việc thiết kế prompt hợp lý không chỉ giúp định hướng đầu ra phù hợp hơn với yêu cầu người dùng mà còn đảm bảo tính chính xác, hợp lệ về mặt cú pháp và logic – đặc biệt trong các tác vụ tạo sinh dữ liệu có cấu trúc như JSON phục vụ game. Trong phạm vi đề tài này, một hệ thống các kỹ thuật Prompt Engineering đã được áp dụng để tăng cường khả năng tuân thủ định dạng và logic gameplay của mô hình.

Đầu tiên là Instruction Prompting, kỹ thuật dựa trên khả năng tuân thủ chỉ dẫn (instruction-following) của các mô hình hiện đại như GPT-4 hay Gemini, vốn đã được tinh chỉnh với dữ liệu hướng dẫn (instruction tuning). Bằng cách đặt rõ vai trò, nhiệm vụ và định dạng đầu ra trong prompt, mô hình có thể tập trung vào yêu cầu cụ thể như "Sinh nhiệm vụ dưới dạng JSON theo định dạng sau...", từ đó tránh sinh văn bản tự do không cần thiết.

Tiếp theo, Schema-Constrained Prompting đóng vai trò then chốt trong việc đảm bảo cấu trúc đầu ra đúng định dạng kỹ thuật. Lin et al. (2022) đã chỉ ra rằng việc đưa vào các mẫu dữ liệu (template) hoặc schema trong prompt giúp mô hình điều chỉnh cơ chế attention để tập trung vào các trường dữ liệu quan trọng, từ đó hạn chế đáng kể lỗi cú pháp hoặc lỗi serialize trong tác vụ sinh mã [6].

Trong các tác vụ sinh dữ liệu dạng cấu trúc, đặc biệt là tích hợp với hệ thống backend game, chỉ kiểm soát cấu trúc JSON là chưa đủ. Do đó Kỹ thuật Enum-Constrained Prompting được sử dụng để ép mô hình chỉ sinh giá trị trong một tập hợp định danh cố định (e.g., taskName từ 0–24). Theo Lin et al. (2022) [6], việc cung cấp ngữ cảnh cấu trúc và ràng buộc loại giá trị (type-level constraint) là chiến lược hiệu quả để điều hướng hành vi của LLM trong các tác vụ lập trình có cấu trúc. Kỹ thuật

này đóng vai trò như một tầng ánh xạ ngữ nghĩa (semantic mapping) giữa thế giới tự nhiên và logic hệ thống.

Bên cạnh các ràng buộc về cấu trúc, độ chính xác của nội dung sinh ra còn phụ thuộc vào khả năng tuân thủ các logic nghiệp vụ phức tạp. Kỹ thuật Rule-Based Contextualization liên quan đến việc nạp các bộ quy tắc vận hành (Operational Rulesets) – chẳng hạn như bảng cân bằng độ khó/phần thưởng hay các giới hạn vật lý của trò chơi – trực tiếp vào ngữ cảnh của prompt. Dựa trên khả năng suy luận logic (Reasoning) của các mô hình LLM hiện đại, kỹ thuật này hướng dẫn mô hình thực hiện các bước kiểm tra nội tại (internal validation) để đảm bảo dữ liệu đầu ra không chỉ đúng cú pháp mà còn khả thi và nhất quán với thiết kế hệ thống [13], [14].

Còn với Negative Prompting (Gợi ý phủ định), kỹ thuật này sẽ thiết lập các "rào chắn kiểm soát" (guardrails) bằng cách quy định rõ những điều mô hình không được làm theo Zhou et al., 2023 [18]. Trong bối cảnh sinh dữ liệu game, kỹ thuật này giúp thu hẹp không gian tìm kiếm xác suất, ngăn chặn mô hình sinh ra các thực thể không tồn tại (ảo giác) hoặc chèn các đoạn văn bản thừa làm hỏng định dạng JSON.

Trong khi đó, Few-shot Prompting – được đề xuất và chứng minh hiệu quả bởi Brown et al. (2020) – cho phép mô hình học từ một số ví dụ mẫu được đưa trực tiếp vào prompt, mà không cần tinh chỉnh mô hình. Việc sử dụng các cặp [Input] – [Ideal Output] giúp mô hình nắm bắt được các quy tắc ngầm định, chẳng hạn như mối quan hệ giữa độ khó của nhiệm vụ và phần thưởng tương ứng, hay cách giới hạn độ dài mô tả theo yêu cầu UI của game [14].

Tổng hợp lại, các cơ sở lý thuyết từ kiến trúc Transformer, cơ chế Attention, In-context Learning đến Instruction Following tạo thành khung tham chiếu vững chắc, chứng minh tính khả thi và hiệu quả của việc sử dụng LLM kết hợp với Prompt Engineering để giải quyết bài toán sinh nội dung tự động trong trò chơi điện tử.

## 1.3 Vấn đề nghiên cứu và nhu cầu ứng dụng LLM

Trong các trò chơi thuộc thể loại casual, hệ thống nhiệm vụ (quest) đóng vai trò trung tâm trong việc giữ chân người chơi, khuyến khích hành vi tích cực và tạo động lực quay lại trò chơi mỗi ngày. Tuy nhiên, quá trình thiết kế các nhiệm vụ này

hiện vẫn đang được thực hiện chủ yếu bằng phương pháp thủ công, dẫn đến nhiều hạn chế đáng kể như khó mở rộng quy mô, nội dung dễ bị lặp lại, và tốn kém chi phí vận hành do phụ thuộc vào nhân lực sáng tạo nội dung.

Sự xuất hiện của các mô hình ngôn ngữ lớn (LLM) mở ra cơ hội đột phá cho việc tự động hóa quy trình thiết kế nhiệm vụ trong game. Với khả năng sinh văn bản mượt mà, đa dạng và giàu ngữ cảnh, LLM có tiềm năng tạo ra các nhiệm vụ phong phú, linh hoạt mà vẫn đảm bảo phù hợp với logic gameplay. Tuy nhiên, để phát huy được hiệu quả trong thực tế, LLM cần được điều hướng thông qua các kỹ thuật Prompt Engineering phù hợp để đảm bảo dữ liệu đầu ra ổn định về cú pháp, đúng ngữ nghĩa và tuân thủ các ràng buộc thiết kế của trò chơi.

Trên cơ sở đó, đề tài này đặt ra câu hỏi nghiên cứu cốt lõi: Làm thế nào để ứng dụng mô hình ngôn ngữ lớn vào quy trình sinh nhiệm vụ game casual một cách hiệu quả, hợp lệ và khả thi về mặt triển khai kỹ thuật? Toàn bộ phần còn lại của luận văn sẽ tập trung vào việc thiết kế, huấn luyện và kiểm định một pipeline sinh nhiệm vụ tự động dựa trên LLM, sử dụng các kỹ thuật Prompt Engineering và cơ chế xác thực nhiều lớp nhằm đảm bảo chất lượng đầu ra phù hợp với yêu cầu của game mobile hiện đại.

# CHƯƠNG 2. CHUẨN BỊ DỮ LIỆU VÀ HUẤN LUYỆN MÔ HÌNH

Chương này đóng vai trò nền tảng kỹ thuật cho toàn bộ đề tài, đặt tiền đề cho việc kiểm thử và đánh giá kết quả ở chương tiếp theo. Các nội dung được trình bày lần lượt từ phân tích dữ liệu đầu vào (mục 2.2), đến lựa chọn mô hình phù hợp (2.3), thiết lập quy trình huấn luyện (2.4), và thiết kế module kiểm định đầu ra (2.5).

## 2.1 Chuẩn bị dữ liệu

Nội dung này trình bày phương pháp phân tích và chuẩn bị dữ liệu cho quá trình Prompt Engineering. Dữ liệu được chuẩn bị đóng vai trò chỉ dẫn mô hình LLM trả về kết quả theo yêu cầu.

### *2.1.1 Quy trình thu thập dữ liệu*

Quy trình thu thập dữ liệu được xây dựng nhằm tạo lập bộ dữ liệu nền thống nhất phục vụ cho giai đoạn mô hình hóa và sinh nhiệm vụ bằng LLM. Quy trình này bảo đảm rằng dữ liệu phản ánh chính xác logic vận hành của trò chơi, đồng thời đủ cấu trúc để hỗ trợ cơ chế In-context Learning trong Prompt Engineering (chi tiết trình bày ở mục 2.4).

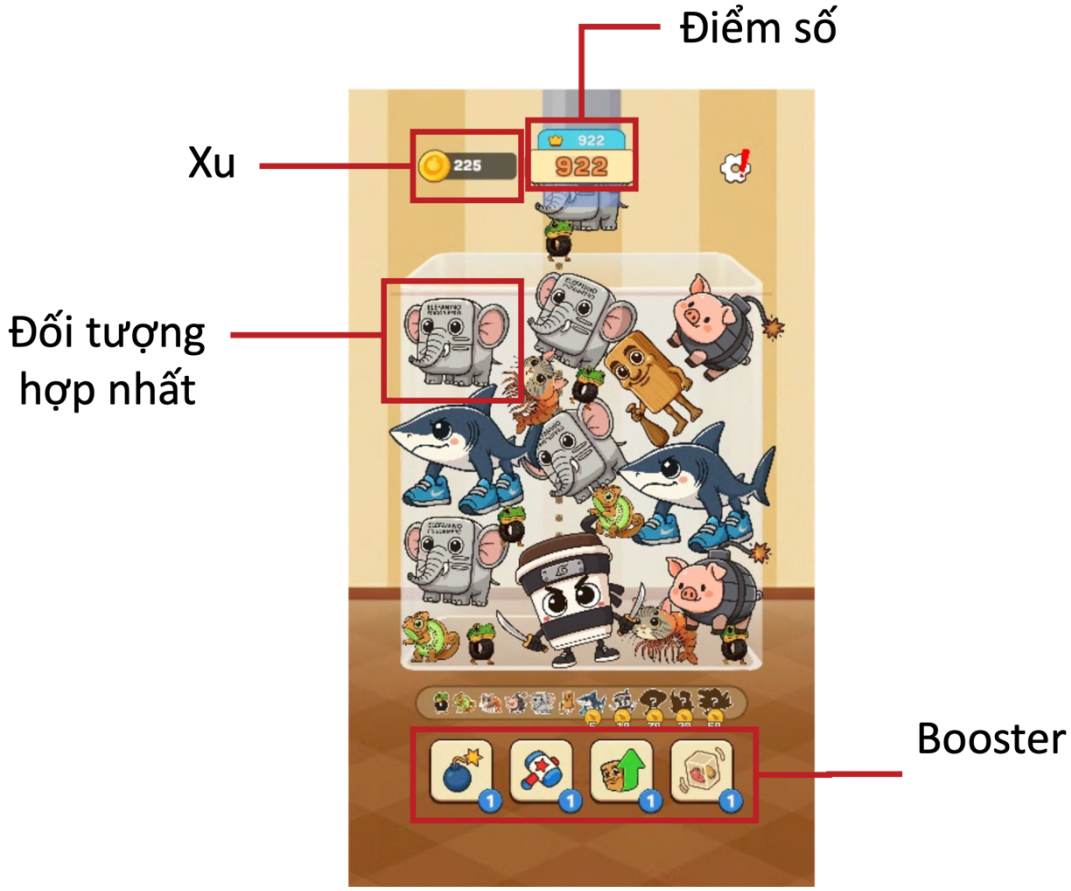
Toàn bộ quy trình được thực hiện qua ba pha chính: Thu thập dữ liệu gameplay, Chuẩn hóa thành Từ điển dữ liệu (Data Dictionary), và Thiết lập bộ quy tắc cân bằng (Balancing Ruleset). Kết quả cuối cùng của quy trình là một tập dữ liệu (Gameplay Reference Set) tinh gọn, nhất quán và sẵn sàng nạp vào pipeline tạo sinh.




**Hình 2-1 Sơ đồ quy trình thu thập dữ liệu**

### a) Thu thập dữ liệu

Dữ liệu gameplay được thu thập thông qua phương pháp phân tích tĩnh mã nguồn Unity (Static Code Analysis) – tập trung vào các file định nghĩa Enum và ScriptableObject – kết hợp với quan sát thực nghiệm và tài liệu thiết kế trò chơi (Game Design Document).




**Hình 2-2 Ảnh nội dung trò chơi merge**

Ở giai đoạn này, dữ liệu tồn tại ở dạng thô, phản ánh các đặc tính trạng thái và hành động của trò chơi, nhưng chưa có cấu trúc chuẩn hóa cho mô hình LLM. Kết quả thu thập thô này đóng vai trò nền tảng để bước chuẩn hóa ở phần tiếp theo có thể diễn ra hiệu quả và chính xác hơn.

### b) Chuẩn hóa dữ liệu

Sau giai đoạn thu thập dữ liệu thô, bước chuẩn hóa được thực hiện nhằm chuyển đổi thông tin sang định dạng cấu trúc hóa, đồng thời triệt tiêu các yếu tố nhập nhằng về mặt ngữ nghĩa. Mục tiêu cốt lõi của giai đoạn này là xây dựng sự thống nhất

tuyệt đối giữa ngôn ngữ tự nhiên và logic lập trình thông qua các thao tác xử lý cụ thể:

(i) Ánh xạ định danh (Identifier Mapping): Thiết lập cơ chế ánh xạ 1-1 giữa mỗi hành động gameplay và một mã định danh duy nhất (Enum Index) trong mã nguồn, đảm bảo tính chính xác khi tích hợp vào hệ thống.

(ii) Chuẩn hóa thuật ngữ (Terminology Normalization): Thống nhất tên gọi, phạm vi tác động và mô tả kỹ thuật của từng hành vi để loại bỏ sự đa nghĩa.

(iii) Phân loại chức năng (Functional Classification): Sắp xếp các nhiệm vụ vào các nhóm chức năng đặc thù (Economy, Core Loop, Skill, Support, Challenge) nhằm phục vụ cho việc quản lý phân phối và cân bằng game.

(iv) Định nghĩa ranh giới cú pháp (Syntax Boundary Definition): Xác định rõ các quy tắc về kiểu dữ liệu và giới hạn giá trị cho từng trường thông tin, tạo tiền đề cho việc xây dựng JSON Schema.

Kết quả của quy trình này là sự hình thành một Từ điển dữ liệu (Data Dictionary) nhất quán. Đây là tài nguyên tham chiếu quan trọng giúp mô hình ngôn ngữ lớn (LLM) thấu hiểu chính xác các khái niệm chuyên ngành, từ đó ngăn chặn hiệu quả tình trạng trùng lặp hoặc sai lệch định nghĩa trong quá trình sinh nhiệm vụ tự động.

### c) Thiết lập bộ quy tắc cân bằng

Nhằm ngăn chặn việc sinh ra các nhiệm vụ bất khả thi hoặc gây phá vỡ cấu trúc cân bằng của trò chơi, hệ thống dữ liệu cân bằng (Balancing Data) được thiết lập dựa trên sự tổng hợp giữa các giới hạn vật lý của game (như kích thước bàn chơi, tốc độ sinh quái) và dữ liệu kiểm thử nội bộ (Playtest Data). Bộ quy tắc này tập trung vào việc định lượng bốn yếu tố cốt lõi:

(i) Độ khó thực tế (Actual Difficulty): Phân cấp độ khó cho từng hành vi gameplay dựa trên tần suất xuất hiện và kỹ năng yêu cầu.

(ii) Tính khả thi theo phiên (Session Feasibility): Xác định giới hạn mục tiêu tối đa có thể đạt được trong thời lượng của một phiên chơi trung bình.

(iii) Tỷ lệ Nỗ lực – Phần thưởng (Effort-to-Reward Ratio): Thiết lập sự tương quan hợp lý giữa mức độ thử thách và giá trị phần thưởng để duy trì động lực người chơi.

(iv) Định giá kinh tế (Economic Valuation): Xác định giá trị quy đổi chuẩn của Coin và Booster trong hệ sinh thái trò chơi.

Các quy tắc cân bằng này đóng vai trò là một khung đánh giá định lượng (Quantitative Evaluation Framework), giúp xác định tính hợp lý của nhiệm vụ ngay tại thời điểm sinh, đồng thời cung cấp nền tảng logic thiết yếu cho mô-đun kiểm định (Validation Layer) ở các bước hậu xử lý.

### *2.1.2 JSON Schema và yêu cầu ràng buộc*

Việc mô hình hóa dữ liệu nhiệm vụ dưới dạng JSON Schema đóng vai trò quan trọng trong quá trình xây dựng hệ thống sinh nhiệm vụ tự động, bởi đây là phương pháp giúp chuẩn hóa cấu trúc dữ liệu và tạo ra khung ràng buộc rõ ràng cho mô hình ngôn ngữ lớn. Cách tiếp cận này phù hợp với kỹ thuật "schema-constrained generation", một hướng tiếp cận được chứng minh là hiệu quả trong Prompt Engineering nhằm kiểm soát đầu ra của LLM theo cấu trúc định sẵn [6], [14]. Việc áp dụng schema không chỉ góp phần giảm thiểu lỗi cú pháp—vốn thường xuất hiện trong các mô hình sinh văn bản tự do—mà còn hạn chế hiện tượng sai lệch ngữ nghĩa, giúp mô hình tập trung vào các trường thông tin bắt buộc thay vì tạo ra nội dung không liên quan.

Bên cạnh đó, dữ liệu có cấu trúc rõ ràng còn làm tăng tính ổn định và khả năng dự đoán của hệ thống, từ đó hỗ trợ module kiểm định (Validation Layer) hoạt động hiệu quả hơn khi tự động phát hiện và loại bỏ các nhiệm vụ không hợp lệ mà không cần can thiệp thủ công. Nhờ những lợi ích này, việc sử dụng JSON Schema trở thành nền tảng không thể thiếu trong thiết kế pipeline sinh nhiệm vụ của đề tài.

#### a) Đặc tả cấu trúc JSON

Cấu trúc nhiệm vụ được định nghĩa xoay quanh 5 trường dữ liệu cốt lõi, được tối ưu hóa cho việc xử lý máy tính:

**Bảng 2-1 Mô tả dữ liệu ứng với mỗi nhiệm vụ**

| Tên trường | Kiểu dữ liệu | Mô tả chức | Ràng buộc (Constraints) |
|---|---|---|---|
| taskName | Integer | Mã định danh loại nhiệm vụ (mapping theo enum) | Giá trị từ 0 đến 24 (Tham chiếu bảng…). |
| taskDes | String | Mô tả nhiệm vụ hiển thị trên UI | Tiếng Anh, ngắn gọn (tối đa 8 từ), văn phong vui tươi casual |
| targetValue | Integer | Giá trị mục tiêu cần đạt | Phải > 0 và khả thi trong 1 ngày chơi |
| rewardDaily | Integer | Loại phần thưởng | 0 (Coin) hoặc 1 (Booster). |
| numReward | Integer | Số lượng phần thưởng | Tuân thủ quy tắc cân bằng (Tham chiếu bảng …). |

Các trường dữ liệu (taskName, targetValue, rewardDaily, numReward…) đảm bảo tính rõ ràng – tương tự các hệ thống phân loại nhiệm vụ trong thiết kế game học thuật [1], [3].

## b) Minh hoạ JSON Schema

Từ yêu cầu của mã nguồn hệ thống nhiệm vụ trong game Merge Fruit – Hoa Quả, việc thiết kế một JSON Schema chuẩn hóa cho dữ liệu đầu ra của mô hình LLM trở thành yêu cầu bắt buộc để đảm bảo tính tương thích với toàn bộ pipeline tích hợp. Trên thực tế, Unity và backend C# yêu cầu dữ liệu nhiệm vụ phải tuân thủ nghiêm ngặt các trường thông tin và kiểu dữ liệu cố định, nếu không hệ thống có thể phát sinh lỗi runtime hoặc hiển thị sai nhiệm vụ. Do đó, JSON Schema được xây dựng như một "bản hợp đồng dữ liệu" (data contract), mô tả chính xác cấu trúc và ràng buộc cho từng nhiệm vụ mà mô hình LLM sinh ra. Mỗi phần tử trong mảng "dailyTasks" phải chứa đầy đủ các trường taskName, taskDes, targetValue,

rewardDaily và numReward, tương ứng với các thuộc tính được định nghĩa trong mã nguồn lớp DailyTask của Unity.

Cấu trúc này không chỉ phục vụ mục tiêu tích hợp hệ thống mà còn giúp LLM tuân thủ những yêu cầu ràng buộc đặc thù của gameplay. Ví dụ, taskName phải là số nguyên ánh xạ trực tiếp sang Enum DailyTaskName trong C#, nhằm tránh lỗi sai chính tả hoặc sai định danh khi mô hình sinh chuỗi văn bản tự do. Tương tự, taskDes được giới hạn tối đa tám từ để đảm bảo hiển thị đúng trong giao diện người chơi trên thiết bị di động, đồng thời giảm hiện tượng sinh mô tả dài dòng hoặc thiếu trọng tâm - một vấn đề thường gặp của các mô hình ngôn ngữ khi không có định hướng rõ ràng. Trường targetValue cũng được ràng buộc nhằm đảm bảo giá trị mô hình sinh ra phản ánh đúng mức độ khả thi của gameplay, tránh sinh mục tiêu vượt quá giới hạn vật lý của trò chơi. Cuối cùng, rewardDaily và numReward tuân theo enum RewardDaily giúp hệ thống phân loại chính xác loại phần thưởng, hỗ trợ việc cân bằng kinh tế trong game.

Dạng Schema tổng quát được sử dụng trong đề tài có thể được mô tả như sau:

```
{
  "dailyTasks": [
    {
      "taskName": <giá trị số trong enum DailyTaskName>,
      "taskDes": "1 câu mô tả ngắn gọn, rõ ràng về nhiệm
vụ, tối đa 8 từ",
      "targetValue": <giá trị mục tiêu cần đạt>,
      "rewardDaily": <giá trị số trong enum RewardDaily>,
      "numReward": <số lượng phần thưởng>
    }
  ]
}
```

**Hình 2-3 Cấu trúc JSON Schema của nhiệm vụ**

Với cấu trúc này, JSON Schema đóng vai trò là tầng ràng buộc đầu tiên giúp đảm bảo đầu ra của LLM phù hợp hoàn toàn với hệ thống thực thi nhiệm vụ, tạo nền tảng cho Validation Layer hoạt động ổn định ở các bước kiểm định tiếp theo. Qua đó,

mô hình không chỉ sinh ra dữ liệu hợp lệ mà còn tuân thủ đầy đủ các quy tắc thiết kế gameplay, tối ưu hóa hiệu quả của pipeline tạo sinh nhiệm vụ tự động.

### *2.1.3 Bộ tham chiếu gameplay (Gameplay Reference Set)*

Bộ tham chiếu gameplay (Gameplay Reference Set) là tập hợp tri thức miền (domain knowledge) được trích xuất và chuẩn hóa từ hệ thống trò chơi, dùng làm nền tảng để mô hình ngôn ngữ lớn (LLM) hiểu và sinh ra các nhiệm vụ phù hợp với ngữ cảnh của game Merge Fruit. Khác với dữ liệu thô được thu thập ban đầu, Gameplay Reference Set đại diện cho lớp dữ liệu đã qua xử lý, được tổ chức có hệ thống và có khả năng mô tả đầy đủ không gian đối tượng (entity space), không gian hành động (action space) và các giới hạn vận hành (operational constraints) của trò chơi. Theo nghiên cứu của Juul [2] và Li [1], việc tạo nhiệm vụ chỉ có ý nghĩa khi nó bám sát các giới hạn vật lý và logic của trò chơi. Xây dựng bộ tham chiếu lấy trực tiếp thông tin từ hệ thống game.

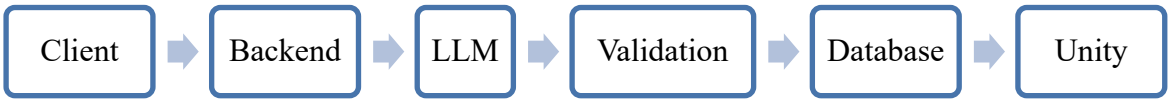
Trong kỹ thuật Prompt Engineering cho mô hình sinh có ràng buộc, việc cung cấp một tập tri thức tham chiếu rõ ràng giúp tăng độ chính xác, giảm hiện tượng "ảo giác" (hallucination), đồng thời đảm bảo đầu ra tuân thủ các yêu cầu về khả thi, cân bằng và phù hợp với trải nghiệm người chơi. Gameplay Reference Set vì thế đóng vai trò quyết định trong việc định hướng hành vi sinh nhiệm vụ của LLM.

Các bảng tham chiếu trên giúp LLM hiểu đúng môi trường trò chơi và sinh ra các nhiệm vụ hợp lệ cả về cú pháp, ngữ nghĩa lẫn logic gameplay. Đây là lớp dữ liệu quan trọng đảm bảo pipeline sinh nhiệm vụ tự động vận hành ổn định, chính xác và phù hợp với cấu trúc thiết kế của game.

## 2.2 Kiến trúc hệ thống và Lựa chọn mô hình

### *2.2.1 Sơ đồ kiến trúc tổng thể*

Hệ thống được thiết kế theo mô hình Client-Server kết hợp với kiến trúc Pipeline xử lý dữ liệu, nhằm đảm bảo tính ổn định, khả năng mở rộng (Scalability) và dễ dàng tích hợp vào Game Engine Unity. Pipeline hệ thống được xây dựng theo chu trình:




**Hình 2-4 Sơ đồ pipeline hệ thống tích hợp LLM vào Unity Engine**

Trong đó chức năng của từng khối được mô tả trong bảng sau:

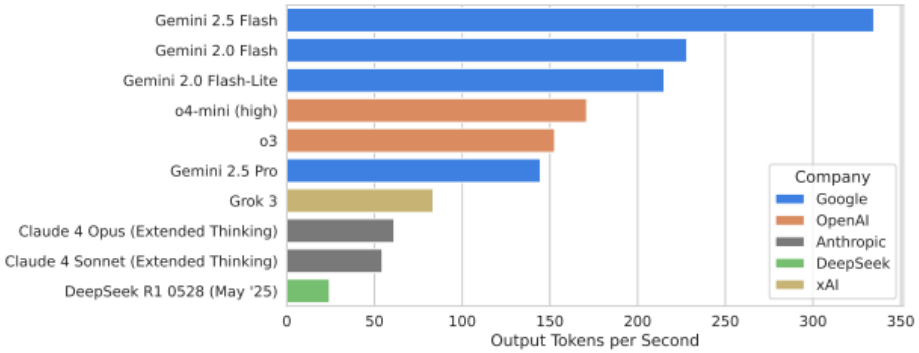
**Bảng 2-2 Bảng chú giải sơ đồ pipeline hệ thống**

| Thành phần | Vai trò / Chức năng chính | Chi tiết hoạt động |
|---|---|---|
| **Client/Unity (Unity Runtime)** | Giao diện người dùng | - Gửi tín hiệu generateDailyTasks<br>- Nhận danh sách nhiệm vụ đã kiểm định (Validated Tasks) và hiển thị UI. |
| **Backend (Middleware)** | Lớp trung gian quản lý phiên và bảo mật | - Quản lý giới hạn truy vấn (Rate Limiting).<br>- Bảo mật API Key, không lộ xuống Client. |
| **LLM Generator (Core Engine)** | Sinh nội dung nhiệm vụ | - Đầu vào: Prompt V3 (tối ưu với Data Dictionary + JSON Schema).<br>- Chiến lược Few-shot Prompting.<br>- Áp dụng Over-generation: sinh 10–12 nhiệm vụ, chọn 5 nhiệm vụ tối ưu. |
| **Validation Layer (Bộ lọc tam cấp)** | Kiểm định chất lượng đầu ra | - Structural Validation: kiểm tra JSON hợp lệ, kiểu dữ liệu đúng.<br>- Logical Validation: kiểm tra tính nhất quán (taskName vs taskDes).<br>- Gameplay Feasibility: kiểm tra độ khó, max level, phần thưởng. |
| **Database** | Lưu trữ dữ liệu | - Lưu lịch sử sinh nhiệm vụ để cache offline |

Cách thiết kế này phù hợp với các khuyến nghị triển khai hệ thống sinh nội dung bằng AI trong môi trường sản xuất (production-level content generation), được đề cập trong nhiều khảo sát về kiến trúc mô hình Transformer [6].

### *2.2.2 Lựa chọn mô hình ngôn ngữ*

Việc lựa chọn mô hình ngôn ngữ dựa trên ba tiêu chí cốt lõi: năng lực sinh nội dung, tuân thủ ràng buộc cấu trúc và hiệu quả triển khai hệ thống. Dựa trên ưu thế của kiến trúc decoder-only trong các tác vụ sinh văn bản (Radford et al., 2018; Brown et al., 2020), đề tài lựa chọn Gemini 2.5 Flash làm mô hình thực nghiệm. Đây là dòng mô hình hiện đại được tối ưu hóa cho tốc độ cao và khả năng duy trì chính xác cấu trúc JSON mà không cần tinh chỉnh (fine-tuning), giúp đảm bảo tính linh hoạt và tiết kiệm chi phí kỹ thuật trong quá trình triển khai.




**Hình 2-5 Biểu đồ so sánh tốc độ sinh token (Output Tokens/s) giữa các mô hình LLM phổ biến [Nguồn: Artificial Analysis.ai, ngày 15/6/2025].**

Với tiêu chí về khả năng tích hợp thực tế vào pipeline của hệ thống là yếu tố quan trọng trong môi trường phát triển game casual — nơi tối ưu chi phí và tốc độ triển khai là những yêu cầu cấp thiết. Việc sử dụng mô hình dạng API như Gemini 2.5 Flash mang lại lợi thế lớn bởi không đòi hỏi duy trì hạ tầng GPU chuyên dụng, cho phép triển khai nhanh chóng, dễ dàng cập nhật prompt mà không cần tinh chỉnh lại mô hình, đồng thời tiết kiệm đáng kể chi phí vận hành. Đây là những lợi ích phù hợp với bối cảnh ngành game casual, vốn được xác nhận qua các báo cáo của Sensor Tower và Newzoo về nhu cầu tối ưu hóa chi phí sản xuất trong lĩnh vực này [15], [16].

Từ các phân tích trên, đề tài đi đến lựa chọn mô hình Gemini 2.5 Flash làm mô hình thực nghiệm chính. Cách tiếp cận sử dụng few-shot prompting kết hợp với Validation Layer, thay vì fine-tuning mô hình, giúp đảm bảo sự cân bằng giữa chất lượng sinh nhiệm vụ, tính linh hoạt trong triển khai và khả năng kiểm soát chi phí. Điều này cũng phù hợp với các kết luận trong nghiên cứu của Brown et al. (2020) về hiệu quả cao của phương pháp Few-shot và Zero-shot trong các hệ thống ứng dụng thực tế sử dụng LLM [14].

### *2.2.3 So sánh chi phí–hiệu năng*

Trong thị trường game casual cạnh tranh, việc lựa chọn mô hình ngôn ngữ không chỉ dựa trên chất lượng mà còn phụ thuộc vào hiệu quả kinh tế, theo báo cáo thị trường của Newzoo và Sensor Tower [15], [16]. Sử dụng Gemini 2.5 Flash qua API mang lại lợi thế vượt trội so với phương pháp tinh chỉnh (fine-tuning) hoặc tự triển khai mô hình nội bộ nhờ lợi thế của nó.

Mô hình này tiết kiệm chi phí thông qua việc loại bỏ hoàn toàn chi phí duy trì hạ tầng GPU và giảm thiểu nguồn lực kỹ thuật cho việc huấn luyện mô hình. Đây là xu hướng chung trong các hệ thống ứng dụng LLM hiện nay, phù hợp với kết luận của Brown et al. (2020) rằng việc sử dụng prompting kết hợp rule-based validation có thể đạt hiệu quả tương đương fine-tuning trong nhiều trường hợp thực tế [14]. Đồng thời, nó cung cấp một hiệu năng hoạt động mạnh mẽ bơi tốc độ sinh JSON nhanh và ổn định, đảm bảo khả năng xử lý thời gian thực cho hệ thống backend mà không gây tắc nghẽn. Không những vậy, sự linh hoạt và có thể mở rộng bằng chiến lược kết hợp Prompt Engineering và Validation Layer tận dụng tối đa năng lực few-shot của mô hình, cho phép dễ dàng điều chỉnh pipeline mà không tốn chi phí huấn luyện lại.

Do đó, đây là điểm cân bằng lý tưởng giữa chi phí và hiệu năng, phù hợp với xu hướng ứng dụng LLM hiện đại được Brown et al. (2020) khuyến nghị.

## 2.3 Quy trình huấn luyện và tối ưu

### *2.3.1 Quy trình tối ưu hóa Prompt (Prompt Optimization Pipeline)*

Dựa trên nền tảng lý thuyết về In-context Learning (ICL) đã trình bày tại Chương 1, đề tài áp dụng chiến lược "huấn luyện phi tham số" (non-parametric training) để điều khiển hành vi của mô hình ngôn ngữ lớn. Thay vì tiêu tốn tài nguyên cho việc cập nhật trọng số mạng nơ-ron (Fine-tuning), phương pháp này tập trung vào việc tối ưu hóa không gian ngữ cảnh đầu vào (Input Context Optimization). Cụ thể, tri thức nghiệp vụ của trò chơi Merge Fruit – bao gồm cấu trúc JSON Schema (mục 2.2.2) và Bộ tham chiếu Gameplay (mục 2.2.3) – được chuyển hóa thành các chỉ dẫn (instructions) và nạp trực tiếp vào cửa sổ ngữ cảnh của mô hình Gemini Flash. Cách tiếp cận này biến mô hình tổng quát thành một tác tử chuyên biệt, có khả năng tư duy theo logic thiết kế game mà không cần trải qua quá trình đào tạo lại.

Quy trình xây dựng prompt không được thực hiện đơn lẻ mà tuân theo phương pháp tối ưu hóa lặp (Iterative Refinement Process) dựa trên vòng phản hồi thực nghiệm. Chiến lược này được chia thành ba giai đoạn phát triển kế tiếp nhau, tương ứng với mức độ kiểm soát ngày càng chặt chẽ:

1. Giai đoạn Khởi tạo (Baseline - Zero-shot): Đánh giá năng lực tự nhiên của mô hình khi chưa có ràng buộc (Prompt V1).

2. Giai đoạn Cấu trúc hóa (Structural Alignment): Tích hợp JSON Schema và Enum để chuẩn hóa định dạng đầu ra (Prompt V2).

3. Giai đoạn Ngữ cảnh hóa toàn diện (Contextual Integration): Nạp tri thức gameplay và áp dụng các kỹ thuật prompt để hoàn thiện logic thiết kế (Prompt V3).

Mỗi giai đoạn đều được đánh giá dựa trên các lỗi phát sinh thực tế để làm cơ sở cho các cải tiến ở bước tiếp theo, đảm bảo kết quả cuối cùng đạt độ chính xác cao nhất về cả cú pháp lẫn ngữ nghĩa.

### *2.3.2 Prompt V1: Phiên bản khởi đầu (Baseline Prompt)*

Prompt V1 là bước thử nghiệm ban đầu nhằm đánh giá khả năng tạo sinh nhiệm vụ của mô hình khi không có bất kỳ ràng buộc kỹ thuật hoặc ngữ cảnh hỗ trợ nào. Phiên bản này chỉ sử dụng một yêu cầu khái quát.

Mục tiêu của thiết kế này là xác định các mô hình lỗi (failure modes) đặc trưng của LLM khi thiếu chỉ dẫn, bao gồm: lỗi sai cú pháp JSON (Syntax Error), sai lệch giá trị định danh (Enum Hallucination), và tạo ra các nội dung không tồn tại trong game (Content Hallucination). Kết quả thực nghiệm từ Prompt V1 cung cấp cơ sở dữ liệu đối chứng quan trọng để đo lường hiệu quả của các kỹ thuật tối ưu hóa được áp dụng trong các phiên bản kế tiếp.

### *2.3.3 Prompt V2: Bổ sung ràng buộc cấu trúc (Structure-Constrained Prompt)*

Dựa trên phân tích các lỗi cú pháp từ phiên bản khởi đầu, Prompt V2 được phát triển với trọng tâm là kỹ thuật Schema-Constrained Prompting được đề xuất trong Lin et al. (2022) [6]. Mục tiêu của phiên bản này là chuyển đổi đầu ra từ văn bản tự do sang dữ liệu có cấu trúc định sẵn, đảm bảo khả năng tích hợp hệ thống. Các cải tiến cốt lõi giúp cải thiện đầu ra của mô hình.

*Thứ nhất*, JSON Schema (đã xây dựng tại mục 2.2.2) được tích hợp trực tiếp vào prompt ép buộc mô hình tuân thủ tuyệt đối các trường thông tin và kiểu dữ liệu, qua đó triệt tiêu lỗi cú pháp (Syntax Error).

*Thứ hai*, bảng ánh xạ enum kèm giải nghĩa (semantic enum mapping) được được bổ sung để thiết lập mối liên kết giữa các con số định danh (0–24) với hành vi người chơi cụ thể. Điều này giúp ngăn chặn việc sinh ra các giá trị nằm ngoài phạm vi cho phép (Out-of-bound).

*Thứ ba,* giới hạn độ dài mô tả nhiệm vụ cũng được bổ sung (ví dụ: yêu cầu taskDes ngắn gọn) nhằm phù hợp với không gian hiển thị UI trong game casual và giảm độ nhiễu trong quá trình sinh.

Đánh giá: Prompt V2 giải quyết hiệu quả bài toán về "hình thức" (Syntax), tuy nhiên vẫn bộc lộ hạn chế về "nội dung" (Semantics). Mô hình vẫn gặp khó khăn trong việc xử lý các ràng buộc logic phức tạp như sự cân bằng độ khó hay tính khả thi vật lý của nhiệm vụ do thiếu tri thức nền tảng.

### *2.3.4 Prompt V3: Tối ưu hoá ngữ cảnh (Final Prompt)*

Prompt V3 đại diện cho bước cải tiến cuối cùng trong chuỗi tối ưu hóa prompt, đánh dấu sự chuyển đổi từ một mô hình ngôn ngữ tổng quát sang một tác tử chuyên biệt cho thiết kế nhiệm vụ trong trò chơi (Game Design Agent). Cấu trúc của Prompt V3 không chỉ là sự cải tiến về mặt cú pháp, mà còn là kết quả của quá trình tích hợp đa tầng các cơ chế lý thuyết đã trình bày tại Chương 1. Thiết kế này kết hợp chặt chẽ giữa kỹ thuật kiểm soát đầu ra, ràng buộc hệ thống và duy trì tri thức nghiệp vụ, hướng đến một hệ thống sinh nhiệm vụ ổn định, chính xác và nhất quán.

Prompt V3 giúp mô hình ngôn ngữ phát huy khả năng của mình để có thể thiết kế game thông qua việc tích hợp các cơ chế và kỹ thuật cốt lõi. Đầu tiên, với Instruction Following thiết lập vai trò chyên gia và ép định dạng JSON Schema để đảm bảo đúng cú pháp đầu ra. Với Enum-Constrained ánh xạ ngôn ngữ tự nhiên sang mã định danh (Unity) để tương thích với backend. Chèn ngữ cảnh (Context Injection) nạp toàn bộ dữ liệu tham chiếu game có được thông qua quá trình thu thập dữ liệu nhằm triệt tiêu ảo giác của mô hình. Kế đến Học trong ngữ cảnh (In-context Learning) sử dụng ví dụ mẫu (few-shoot) để dạy quy tắc cần bằng game, ràng buộc input-output mà không cần tinh chỉnh. Cuối cùng, Ràng buộc phủ định (Negative Prompting) thiết lập rào chắn để loại bỏ các nội dung sai lệch hoặc dư thừa.

Tổng kết lại, Prompt V3 không đơn thuần là một tập hợp các chỉ dẫn, mà là một hệ thống kỹ thuật hoàn chỉnh được thiết kế dựa trên cơ sở lý thuyết vững chắc. Việc tích hợp các yếu tố cấu trúc (Schema), ngữ nghĩa (Enum Mapping) và tri thức (Contextual Knowledge) đã giúp phiên bản này đạt được mức độ chính xác, ổn định và phù hợp cao nhất với yêu cầu triển khai thực tế trong môi trường sản xuất game.

**Bảng 2-3 Thành phần và kỹ thuật áp dụng trong Prompt V3**

| Thành phần trong Prompt | Kỹ thuật áp dụng | Đánh giá hiệu quả |
|---|---|---|
| "Bạn là chuyên gia thiết kế nhiệm vụ..." | Instruction Prompting (Thiết lập Persona) | Định hình tư duy mô hình, chuyển từ chatbot sang Game Designer. |

| | | |
|---|---|---|
| JSON Template:<br>{ "dailyTasks": [ ... ] } | Schema-Constrained Prompting (Ràng buộc cấu trúc) | Cung cấp "khuôn mẫu" cứng, đảm bảo output 100% parse được vào Unity. |
| "Giải thích ý nghĩa enum DailyTaskName..."<br>*(List 0-24)* | Enum-Constrained / Semantic Mapping (Ánh xạ ngữ nghĩa) | Cầu nối giữa Ngôn ngữ tự nhiên và Mã nguồn (Code). |
| "Thông tin tham khảo...<br>Cấp độ các con…<br>Booster gồm 4 loại..."<br>*(Rotolasio đến SixSeven; 4 loại booster)* | Context Injection (Nạp ngữ cảnh) | Cung cấp "Từ điển dữ liệu" nhằm triệt tiêu hoàn toàn ảo giác (Hallucination) như việc sinh ra Level 12. |
| "Quy tắc & hướng dẫn tạo nhiệm vụ...”<br>*(Task dễ thưởng 10-20 coin...)* | Logic/Ruleset Injection (Nạp luật vận hành) | Giúp mô hình không chỉ sinh đúng cú pháp mà còn có khăng năng cân bằng Game |
| "Chỉ trả về JSON hợp lệ duy nhất, không kèm mô tả khác..." | Negative Prompting (Gợi ý phủ định) | Thiết lập rào chắn (Guardrails) để lọc bỏ các đoạn văn bản thừa. |

Tổng hợp lại, Prompt V3 (Final Prompt) là sản phẩm của quá trình kỹ thuật ngược (Reverse Engineering) từ logic game vào ngôn ngữ tự nhiên. Thông qua việc kết hợp chặt chẽ giữa Dữ liệu cấu trúc (JSON Schema/Reference Set) và Kỹ thuật điều khiển (Prompt Engineering Techniques), hệ thống đảm bảo khả năng sinh nhiệm vụ đạt chuẩn sản xuất: đúng cú pháp, vững logic và đa dạng về nội dung.

## 2.4 Module kiểm định (Validation Layer)

Trong kiến trúc hệ thống đề xuất, Module Kiểm định (Validation Layer) đóng vai trò là "hàng rào kiểm soát chất lượng" (Quality Gate) cuối cùng trước khi dữ liệu được chuyển đến người chơi. Dựa trên tính chất xác suất (probabilistic nature) của các mô hình ngôn ngữ lớn, việc tồn tại một tỷ lệ lỗi nhất định trong đầu ra là điều khó tránh khỏi, ngay cả khi đã áp dụng các kỹ thuật Prompt Engineering tối ưu [6], [14].

Do đó, việc thiết lập một lớp kiểm định độc lập, vận hành theo cơ chế tất định (deterministic), là yêu cầu thiết yếu để đảm bảo sự ổn định và an toàn cho hệ thống trò chơi. Module này được thiết kế dựa trên mô hình kiểm soát ba tầng, tương ứng với ba cấp độ toàn vẹn của dữ liệu: Cấu trúc, Ngữ nghĩa và Nghiệp vụ.

### *2.4.1 Kiểm định cấu trúc (Structural Validation)*

Tầng kiểm định đầu tiên tập trung vào xác thực tính toàn vẹn cú pháp và định dạng dữ liệu đầu ra của mô hình. Mục tiêu cốt lõi là đảm bảo rằng mọi nhiệm vụ được sinh ra đều tuân thủ nghiêm ngặt định dạng JSON, bao gồm: tên trường (field names) chính xác, kiểu dữ liệu đúng (type checking) và giá trị đầu ra nằm trong miền hợp lệ của các biến định danh (enum).

Quy trình kiểm định diễn ra theo bốn bước liên tiếp. Bước đầu tiên là phân tích cú pháp (Parse JSON), nơi các nhiệm vụ có lỗi định dạng – như thiếu dấu ngoặc hoặc lỗi trích dẫn – bị loại bỏ ngay lập tức. Tiếp theo, hệ thống tiến hành kiểm tra sự đầy đủ của các trường thông tin bắt buộc (Field Validation), bao gồm taskName, taskDes, targetValue, rewardDaily và numReward. Bước thứ ba là kiểm tra kiểu dữ liệu (Type Validation), đảm bảo rằng taskDes là chuỗi ký tự (string), trong khi các trường còn lại là số nguyên (integer). Cuối cùng, kiểm tra giới hạn giá trị (Enum Range Validation) được thực hiện để xác nhận taskName nằm trong khoảng [0, 24] và rewardDaily thuộc tập {0, 1}.

Việc tích hợp lược đồ JSON Schema ở giai đoạn này đóng vai trò như một lớp ràng buộc cú pháp cứng, được xem là một trong những kỹ thuật hiệu quả nhất để kiểm soát lỗi đầu ra của LLM theo kết quả khảo sát của Lin et al. (2022) [6]. Tầng kiểm định này đã chứng minh hiệu quả vượt trội trong việc loại bỏ nhiệm vụ lỗi cấu trúc – đặc biệt phổ biến trong các phiên bản prompt chưa tối ưu như Prompt V1 – đồng thời đảm bảo khả năng tích hợp trơn tru với hệ thống backend Unity.

### *2.4.2 Kiểm định logic gameplay (Gameplay Logic Validation)*

Sau khi vượt qua lớp kiểm định cấu trúc, dữ liệu tiếp tục được xử lý tại tầng logic để đánh giá tính khả thi và tính phù hợp với thiết kế gameplay của dòng game casual. Tầng kiểm định này tập trung vào hai tiêu chí chính: (1) tính hợp lý của giá

trị mục tiêu (targetValue) và (2) sự tương thích giữa mô tả văn bản (taskDes) với hành vi được định danh bởi taskName.

Các giới hạn vật lý trong gameplay – chẳng hạn như số lượng quái vật tối đa có thể hợp nhất hoặc số lần xem quảng cáo tối đa mỗi ngày – được sử dụng như quy tắc kiểm tra. Nếu một nhiệm vụ yêu cầu vượt quá giới hạn này, nó sẽ bị loại bỏ. Ngoài ra, nội dung mô tả (taskDes) cũng được phân tích ngữ nghĩa để đảm bảo rằng nó phản ánh đúng hành động mà taskName chỉ định, loại bỏ các trường hợp mâu thuẫn logic như "Xem quảng cáo" nhưng được định danh bằng mã số dành cho "Thu thập vật phẩm".

Những tiêu chuẩn kiểm định này không chỉ đảm bảo tính đúng đắn về mặt vận hành mà còn tuân thủ các nguyên tắc về tiến trình hợp lý trong thiết kế game casual (progression pacing) nhằm duy trì động lực của người chơi, như đã được trình bày trong các nghiên cứu của Kumar và M. (2025) [3].

### *2.4.3 Kiểm định khả thi và cân bằng thiết kế (Gameplay Feasibility & Balancing)*

Tầng kiểm định cuối cùng và quan trọng nhất là đánh giá tính phù hợp của nhiệm vụ trong bối cảnh thực tế của trò chơi. Dựa trên Bộ tham chiếu Gameplay và Quy tắc cân bằng (Balancing Ruleset - mục 2.2.3), hệ thống áp dụng các thuật toán heuristic để lọc bỏ các nhiệm vụ phá vỡ trải nghiệm người chơi:

(i) Kiểm tra giới hạn vật lý (Physical Constraints): Đảm bảo các giá trị mục tiêu (targetValue) nằm trong giới hạn khả thi của một phiên chơi. Ví dụ: Không yêu cầu số lượng Combo vượt quá khả năng thao tác trung bình của người chơi casual [1].

(ii) Kiểm soát kinh tế (Economy Check): Đối chiếu tỷ lệ giữa độ khó (Difficulty) và phần thưởng (Reward). Hệ thống sẽ tự động loại bỏ hoặc điều chỉnh các nhiệm vụ "lỗi" như: yêu cầu thực hiện tác vụ đơn giản (Login) nhưng lại trao phần thưởng quá lớn, gây lạm phát tiền tệ trong game.

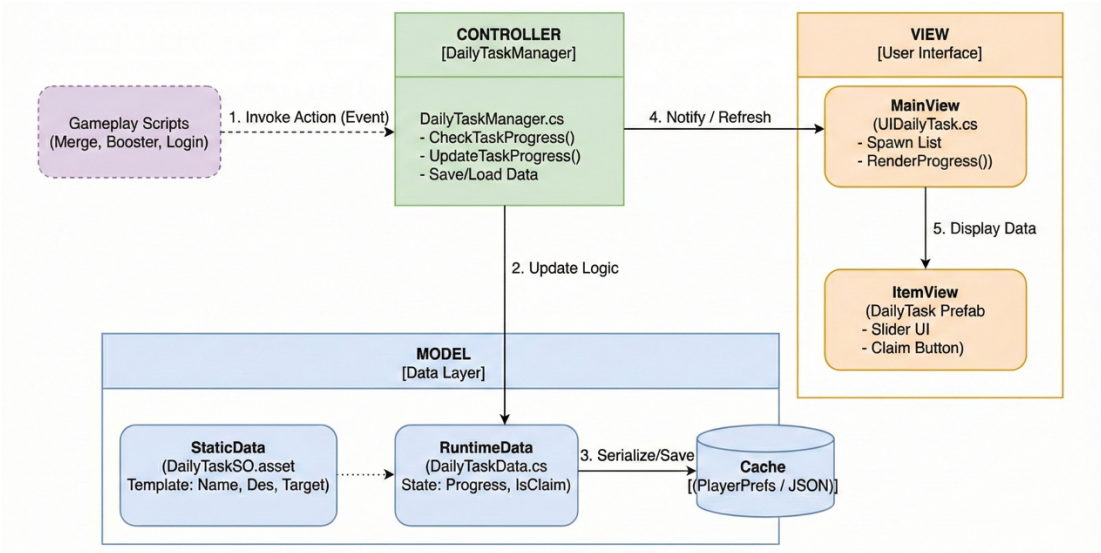
(iii) Đảm bảo tính đa dạng (Diversity Check): Trong trường hợp sử dụng chiến lược sinh dư thừa (Over-generation), tầng này chịu trách nhiệm chọn lọc ra bộ 5

nhiệm vụ tối ưu nhất, đảm bảo bao phủ đủ các nhóm chức năng (Economy, Skill, Core Loop) theo đúng mô hình hành vi người chơi đã phân tích tại Chương 1.

## 2.5 Tích hợp thực nghiệm vào game

### *2.5.1 Thiết kế và triển khai Quest System theo mô hình*

Hệ thống nhiệm vụ (Quest System) được kiến trúc và triển khai dựa trên mô hình Model-View-Controller (MVC) kết hợp với Observer Pattern (Mẫu quan sát viên) nhằm đảm bảo sự tách biệt rõ ràng giữa dữ liệu, giao diện và logic xử lý, đồng thời tối ưu hóa khả năng mở rộng.




**Hình 2-6 Sơ đồ mô hình MVC triển khai Quest System**

Trong kiến trúc này, thành phần Model được đại diện bởi lớp DailyTaskSO (ScriptableObject) và lớp bao đóng DailyTaskData. DailyTaskSO đóng vai trò là khuôn mẫu dữ liệu tĩnh, định nghĩa các thuộc tính cơ bản như tên nhiệm vụ (TaskName), mô tả và phần thưởng, giúp tối ưu hóa bộ nhớ thông qua việc chia sẻ dữ liệu tham chiếu. Ngược lại, DailyTaskData lưu trữ trạng thái động của từng nhiệm vụ tại thời điểm chạy (Runtime) như tiến độ hiện tại (Progress), trạng thái hoàn thành (IsSuccess) và trạng thái nhận thưởng (IsClaim). Việc tách biệt này cho phép hệ thống dễ dàng quản lý dữ liệu người chơi mà không làm ảnh hưởng đến cấu trúc dữ liệu gốc.

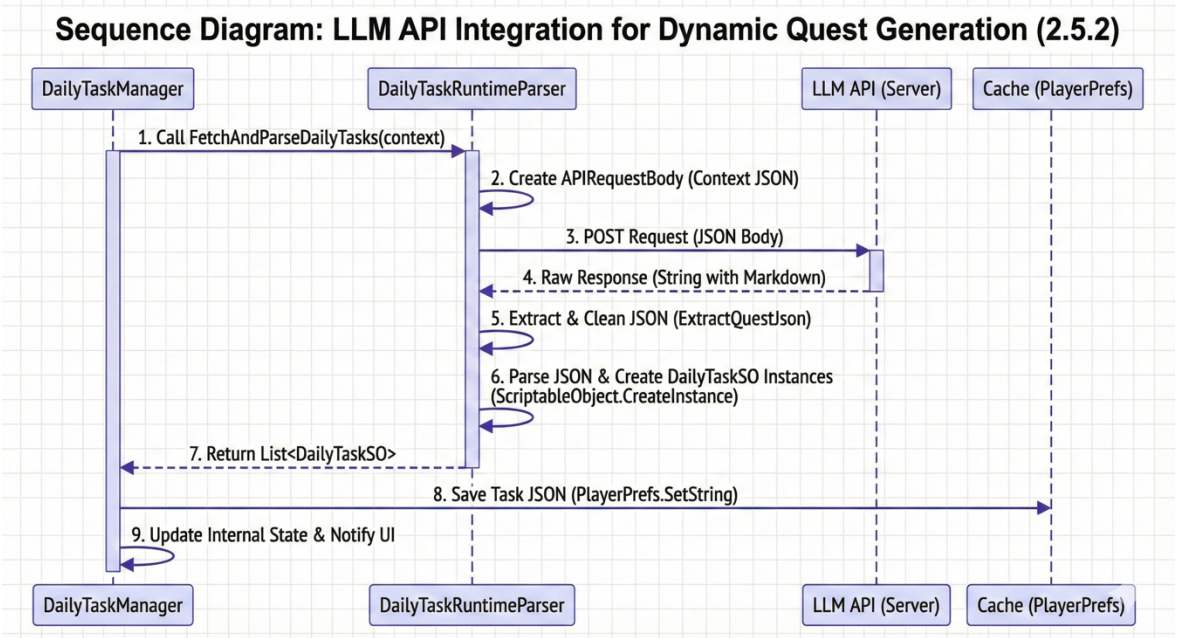
Thành phần View được đảm nhiệm bởi lớp UITaskManager (trong mã nguồn là UIDailyTask). Thành phần này không chứa logic nghiệp vụ mà chỉ chịu trách nhiệm lắng nghe dữ liệu từ Controller để hiển thị trực quan lên màn hình thông qua

các đối tượng UI như thanh trượt tiến độ (Slider), văn bản và hiệu ứng hình ảnh. Sử dụng kỹ thuật Object Pooling, View quản lý việc khởi tạo và tái sử dụng các prefab nhiệm vụ (DailyTask), đảm bảo hiệu năng khi danh sách nhiệm vụ thay đổi.

Đóng vai trò trung tâm là thành phần Controller, được thực hiện bởi lớp DailyTaskManager. Đây là bộ não của hệ thống, quản lý vòng đời của nhiệm vụ từ việc khởi tạo, tải dữ liệu (từ API hoặc bộ nhớ đệm), đến việc kiểm tra logic nghiệp vụ. Controller áp dụng cơ chế hướng sự kiện (Event-Driven) thông qua việc đăng ký lắng nghe các sự kiện ActionName từ các module gameplay khác. Khi người chơi thực hiện một hành động (ví dụ: PerformMerge hoặc UseBooster), Controller sẽ bắt sự kiện, đối chiếu với dữ liệu trong Model và cập nhật tiến độ. Nếu tiến độ đạt mục tiêu, Controller sẽ thay đổi trạng thái trong Model và thông báo cho View cập nhật giao diện, khép kín vòng lặp xử lý của mô hình MVC.

### *2.5.2 Tích hợp API tạo Quest tự động bằng LLM*

Việc tích hợp trí tuệ nhân tạo vào quy trình tạo nhiệm vụ được thực hiện thông qua module trung gian DailyTaskRuntimeParser, hoạt động như một lớp dịch vụ (Service Layer) giúp kết nối Game Client với hệ thống Large Language Model (LLM). Quy trình này giải quyết bài toán tạo nội dung động (Procedural Content Generation) mà không cần can thiệp cứng vào mã nguồn.



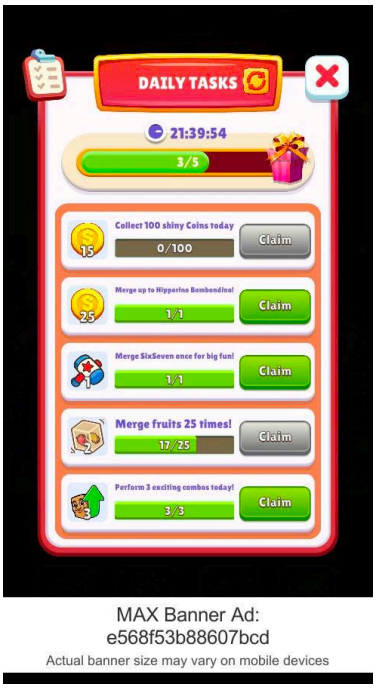

**Hình 2-7 Biểu đồ luồng hoạt động tích hợp LLM cho trò chơi**

Về mặt kỹ thuật, hệ thống sử dụng UnityWebRequest để gửi yêu cầu bất đồng bộ đến máy chủ AI và tích hợp bước tiền xử lý để làm sạch dữ liệu JSON phản hồi. Điểm đột phá nằm ở hai cơ chế được trình bày dưới đây.

Khởi tạo tại thời điểm chạy (Runtime Instantiation) giúp dữ liệu JSON được chuyển đổi trực tiếp thành đối tượng ScriptableObject qua phương thức CreateInstance<DailyTaskSO>(). Kỹ thuật này giúp các nhiệm vụ sinh ra tương thích hoàn toàn với logic xử lý sự kiện hiện có của Game Controller như một tài nguyên có sẵn.

Cơ chế lưu trữ đệm (Caching) khiến dữ liệu sau khi tải về được tuần tự hóa (Serialize) và lưu cục bộ vào PlayerPrefs. Hệ thống ưu tiên tải từ bộ nhớ đệm trong các lần đăng nhập tiếp theo, giúp giảm tải cho máy chủ và đảm bảo trải nghiệm người chơi mượt mà.

### *2.5.3 Xây dựng giao diện Quest System trong game*




**Hình 2-8 Giao diện Quest System trong game**

Xây dựng giao diện Quest System hiển thị tên câu hỏi, phần thưởng, tiến trình, các nút bấm nhận thưởng, trạng thái đã nhân và nút reset câu hỏi để cập nhật bộ câu hỏi mới qua API.

# CHƯƠNG 3. KIỂM THỬ VÀ KHẢO SÁT NGƯỜI DÙNG

## 3.1 Môi trường thực nghiệm

### *3.1.1 Mục tiêu thực nghiệm*

Mục tiêu cốt lõi của quá trình thực nghiệm là kiểm chứng tính hiệu quả của quy trình tối ưu hóa Prompt (Prompt Optimization Pipeline) đã đề xuất tại Chương 2. Cụ thể, thực nghiệm tập trung vào việc đánh giá định lượng chất lượng đầu ra của mô hình ngôn ngữ lớn qua ba phiên bản prompt khác nhau (V1, V2, V3), từ đó đo lường mức độ cải thiện về cấu trúc JSON, tính nhất quán của Enum và độ hợp lệ của logic gameplay. Bên cạnh các chỉ số kỹ thuật, nghiên cứu cũng tiến hành kiểm tra mức độ chấp nhận của người dùng (User Acceptance Testing) nhằm xác nhận rằng các nhiệm vụ được sinh ra không chỉ đúng về mặt kỹ thuật mà còn đáp ứng được các tiêu chuẩn về trải nghiệm người chơi trong dòng game casual. Phương pháp đánh giá này được xây dựng dựa trên mô hình "sinh có ràng buộc kết hợp hậu kiểm định" (constrained generation + post-validation), phù hợp với các khuyến nghị trong nghiên cứu về ứng dụng Transformer trong sản xuất nội dung tự động [6], [14].

### *3.1.2 Cấu hình thực nghiệm*

Để đảm bảo tính khách quan và khả năng tái lập của thực nghiệm, hệ thống được thiết lập với các thông số kỹ thuật cố định xuyên suốt quá trình đánh giá.

Về mô hình ngôn ngữ, đề tài sử dụng mô hình Gemini 2.5 Flash (Google) thông qua API. Lựa chọn này dựa trên ưu thế vượt trội về tốc độ suy luận (inference speed) và chi phí vận hành thấp đã phân tích tại mục 2.3.2.

Về môi trường tích hợp: Hệ thống Client được phát triển trên nền tảng Unity Engine (phiên bản Unity 6000.1.17f1), tích hợp Module Kiểm định (Validation Layer) được viết bằng C#. Môi trường này mô phỏng chính xác quy trình vận hành thực tế: gửi yêu cầu từ Client → xử lý tại Server API → nhận phản hồi JSON → kiểm định và hiển thị lên giao diện game (UI).

Về quy mô dữ liệu mẫu: Quá trình thực nghiệm thực hiện tổng cộng 150 lượt sinh dữ liệu (Generation Iterations), được chia đều thành 3 tập mẫu tương ứng với 50 lượt cho mỗi phiên bản Prompt (V1, V2, V3). Số lượng mẫu này tuân theo phương

pháp đánh giá mô hình ngôn ngữ trong các bài toán Few-shot Learning của Brown et al. (2020) [14], đảm bảo đủ độ lớn thống kê để loại trừ các yếu tố ngẫu nhiên và đưa ra kết luận tổng quát về độ ổn định của hệ thống.

### *3.1.3 Công cụ đo lường (Evaluation Metrics)*

Hiệu quả của mô hình được đánh giá thông qua hệ thống thước đo kép, bao gồm các chỉ số kỹ thuật định lượng và chỉ số trải nghiệm định tính.

*Nhóm chỉ số kỹ thuật (Technical Metrics):*

Tỷ lệ hợp lệ cấu trúc (Structural Validity Rate) đo lường phần trăm các phản hồi tuân thủ đúng định dạng JSON Schema, không mắc lỗi cú pháp (Syntax Error) và đảm bảo an toàn kiểu dữ liệu (Type Safety). Đây là chỉ số phản ánh hiệu quả của kỹ thuật Schema-Constrained Prompting.

Tỷ lệ tuân thủ ngữ nghĩa (Semantic Compliance Rate) đo lường mức độ chính xác của các giá trị Enum (ví dụ: taskName trong khoảng [0-24], rewardDaily trong {0,1}). Chỉ số này đánh giá khả năng Ánh xạ ngữ nghĩa (Semantic Mapping) của mô hình.

Tỷ lệ hợp lệ logic (Gameplay Feasibility Rate) kiểm tra xem các giá trị mục tiêu (targetValue) và phần thưởng có nằm trong ngưỡng cân bằng của game hay không (ví dụ: không sinh Level > 11). Chỉ số này phản ánh năng lực Rule-based Contextualization.

*Nhóm chỉ số trải nghiệm người dùng (User Experience Metrics)*

Dựa trên lý thuyết thiết kế game casual của Li (2024) [1] và Juul (2010) [2],các nhiệm vụ sinh ra được đánh giá bởi người dùng thật trên thang điểm Likert 5 mức độ về ba khía cạnh: Độ rõ ràng (Clarity – mức độ dễ hiểu của nhiệm vụ), Sự phù hợp (Balance – độ khó tương xứng phần thưởng), và Động lực (Motivation – khả năng kích thích người chơi quay lại).

## 3.2 Kết quả sinh nhiệm vụ từ mô hình

Phần này trình bày các kết quả thực nghiệm thu được từ quá trình sinh nhiệm vụ bằng mô hình Gemini Flash qua ba giai đoạn phát triển prompt. Các dữ liệu được phân tích dựa trên sự so sánh đối chiếu giữa năng lực cơ sở (Baseline) và năng lực sau

khi tối ưu hóa, nhằm làm rõ tác động của từng kỹ thuật Prompt Engineering đối với chất lượng đầu ra.

### *3.2.1 Hiệu quả của các phiên bản prompt*

Quá trình thực nghiệm được chia thành ba giai đoạn tương ứng với ba cấp độ can thiệp kỹ thuật.

**Bảng 3-1 Bảng so sánh 3 cấp độ prompt**

| **Đặc điểm** | **Prompt V1** | **Prompt V2** | **Prompt V3 (Final)** |
|---|---|---|---|
| **Kỹ thuật** | Zero-shot | Schema-Constrained + Enum Mapping | Context Injection + Few-shot + Ruleset |
| **Đầu ra JSON** | Lỗi cú pháp thường xuyên | Chính xác cấu trúc 100% | Chính xác cấu trúc 100% |
| **Logic Game** | Ảo giác (sinh item lạ) | Sai lệch (Level > 11, thưởng sai) | Chính xác, cân bằng, có Flavor Text |
| **Tính khả thi** | Thấp | Trung bình | Cao (Sẵn sàng cho Production) |

Tại giai đoạn 1 (Prompt V1 - Zero-shot Baseline), mô hình hoạt động dựa trên tri thức tổng quát và không có sự hỗ trợ của ngữ cảnh nghiệp vụ. Kết quả định tính cho thấy mô hình thường xuyên mắc lỗi "ảo giác" (hallucination): sinh ra các vật phẩm không tồn tại trong game (như "Sword", "Shield"), sử dụng các định dạng JSON không đồng nhất, hoặc tự ý thêm các đoạn văn bản giải thích thừa thãi. Điều này xác nhận hạn chế của phương pháp Zero-shot đối với các tác vụ yêu cầu độ chính xác cao về cấu trúc.

Bước sang giai đoạn 2 (Prompt V2 - Structural Alignment), việc áp dụng kỹ thuật Schema-Constrained Prompting, chất lượng đầu ra đã có sự cải thiện rõ rệt về mặt hình thức. Các lỗi cú pháp (Syntax Error) giảm mạnh, cấu trúc JSON trở nên ổn định và có thể phân tích tự động (parse) bởi hệ thống. Tuy nhiên, về mặt nội dung, Prompt V2 vẫn tồn tại các lỗi logic nghiêm trọng như yêu cầu cấp độ quái vật vượt quá giới hạn thiết kế (Level > 11) hoặc phân phối phần thưởng không tương xứng với độ khó, do thiếu sự liên kết ngữ nghĩa (semantic grounding) với logic game.

Cuối cùng, ở giai đoạn 3 (Prompt V3 - Contextual Optimization), phiên bản hoàn thiện tích hợp Context Injection và Rule-based Contextualization cho thấy sự vượt trội toàn diện: mô hình không chỉ tuân thủ tuyệt đối cấu trúc JSON mà còn thể hiện sự "thông hiểu" sâu sắc về gameplay. Các nhiệm vụ sinh ra sử dụng chính xác hệ thống tên gọi (Terminology) của game Merge Fruit, tuân thủ các quy tắc cân bằng kinh tế và đảm bảo tính khả thi vật lý.

### *3.2.2 Thống kê định lượng*

Để đánh giá chính xác hiệu năng kỹ thuật, đề tài tiến hành đo lường trên tập mẫu 150 lượt sinh dữ liệu (50 lượt/phiên bản). Kết quả được tổng hợp qua ba chỉ số hiệu suất chính (KPIs):

**Bảng 3-2 Bảng tổng hợp hiệu suất kỹ thuật giữa các phiên bản Prompt**

| Chỉ số đánh giá | Prompt V1 (Zero-shot) | Prompt V2 (Schema-only) | Prompt V3 (Context-aware) |
|---|---|---|---|
| **Tỷ lệ JSON hợp lệ** | 0% | 100% | **100%** |
| **Tỷ lệ Enum hợp lệ** | 0% | 100% | **100%** |
| **Tỷ lệ Logic Game hợp lệ** | 0% | 40% | **96%** |

Kết quả thống kê cho thấy sự phân hóa rõ rệt giữa các phiên bản. Đối với chỉ số cấu trúc và ngữ nghĩa, Prompt V1 ghi nhận tỷ lệ thất bại hoàn toàn (0%) do mô hình ở trạng thái Zero-shot thường xuyên trả về văn bản dẫn nhập thừa hoặc sử dụng sai tên trường dữ liệu, khiến hệ thống không thể phân tích cú pháp tự động. Ngược lại, Prompt V2 và V3 đạt độ chính xác tuyệt đối (100%) ở cả hai chỉ số này. Sự cải thiện đột biến này là minh chứng thực nghiệm cho tính hiệu quả của kỹ thuật Schema-Constrained và Semantic Mapping, khẳng định rằng việc cung cấp khuôn mẫu cứng và bảng ánh xạ giá trị giúp triệt tiêu hoàn toàn các lỗi về cú pháp và định danh.

Tuy nhiên, sự khác biệt mang tính quyết định nằm ở chỉ số logic game. Mặc dù Prompt V2 đảm bảo đúng định dạng kỹ thuật, nhưng chỉ có 40% số nhiệm vụ sinh ra là hợp lý về mặt thiết kế; phần lớn còn lại mắc các lỗi nghiêm trọng về cân bằng hoặc

yêu cầu bất khả thi. Trong khi đó, Prompt V3 đạt tỷ lệ logic hợp lệ lên tới 96%. Mức tăng trưởng vượt bậc này chứng minh tác động của kỹ thuật Context Injection và Few-shot Learning, giúp mô hình không chỉ "biết viết đúng" (cú pháp) mà còn "biết suy nghĩ đúng" (nghiệp vụ). 4% sai sót còn lại ở Prompt V3 chủ yếu là các sai lệch nhỏ về độ khó chủ quan, hoàn toàn nằm trong phạm vi xử lý của Module Kiểm định.

### *3.2.3 Phân tích các dạng lỗi điển hình*

Thông qua việc phân tích sâu các trường hợp thất bại (Failure Modes) trong tập dữ liệu thực nghiệm, nghiên cứu đã nhận diện các nhóm lỗi đặc trưng và cơ chế khắc phục tương ứng của hệ thống đề xuất.

Nhóm lỗi đầu tiên là Ảo giác nội dung (Content Hallucination), xuất hiện phổ biến ở Prompt V1. Do thiếu tri thức nền tảng, mô hình thường xuyên sinh ra các thực thể không tồn tại trong thiết kế trò chơi, ví dụ như vật phẩm "Sword" hay hành động "Share Facebook". Tại Prompt V3, lỗi này đã được loại bỏ hoàn toàn nhờ kỹ thuật Nạp ngữ cảnh (Context Injection). Bằng việc nạp toàn bộ danh sách thực thể hợp lệ (như tên các cấp độ quái vật và loại Booster) vào bộ nhớ ngắn hạn, không gian tìm kiếm của mô hình bị giới hạn chặt chẽ, ngăn chặn triệt để khả năng sáng tạo sai lệch.

Nhóm lỗi thứ hai liên quan đến Mất cân bằng kinh tế (Economic Imbalance) và Bất khả thi vật lý (Physical Infeasibility), là đặc trưng của Prompt V2. Trong giai đoạn này, mô hình tuy tuân thủ cấu trúc nhưng lại thiếu các quy tắc vận hành, dẫn đến việc sinh ra các nhiệm vụ đơn giản như "Đăng nhập" nhưng trao phần thưởng quá lớn gây lạm phát, hoặc yêu cầu đạt cấp độ quái vật vượt quá giới hạn thiết kế (Level > 11). Prompt V3 đã khắc phục vấn đề này thông qua sự kết hợp giữa Few-shot Learning và Negative Prompting. Các ví dụ mẫu giúp mô hình học được tỷ lệ quy đổi hợp lý giữa nỗ lực và phần thưởng, trong khi các rào chắn kiểm soát (Guardrails) ngăn chặn việc sinh ra các giá trị vượt ngưỡng cho phép.

Tổng hợp lại, kết quả phân tích lỗi khẳng định rằng chỉ khi kết hợp đồng bộ cả ba trụ cột kỹ thuật – Cấu trúc, Ngữ nghĩa và Logic – trong Prompt V3, hệ thống mới đạt được độ tin cậy cần thiết để triển khai trong môi trường sản xuất thực tế.

## 3.3 Khảo sát người dùng

Song song với các đánh giá định lượng về mặt kỹ thuật, việc đo lường mức độ chấp nhận của người dùng (User Acceptance) đóng vai trò then chốt để xác định tính khả thi của hệ thống khi triển khai thực tế. Mục tiêu của phần này là kiểm chứng xem các nhiệm vụ do mô hình AI sinh ra có đáp ứng được các tiêu chuẩn về trải nghiệm người chơi (Player Experience - PX) hay không, đặc biệt là trong bối cảnh game casual nơi yếu tố giữ chân người dùng (retention) phụ thuộc lớn vào sự hài lòng tức thời.

### *3.3.1 Mục tiêu khảo sát*

Khảo sát được thiết kế nhằm đánh giá chất lượng nhiệm vụ dựa trên ba tiêu chí tâm lý học hành vi cốt lõi: Mức độ dễ hiểu (Clarity), Sự phù hợp của độ khó (Difficulty Fit) và Khả năng tạo động lực (Motivation). Theo nghiên cứu của Kumar (2025) về cơ chế củng cố hành vi trong trò chơi miễn phí, một hệ thống nhiệm vụ hợp lý sẽ kích hoạt các "vòng lặp dopamine" (dopamine loops), góp phần gia tăng chỉ số giữ chân và tần suất quay lại của người chơi loops [3]. Do đó, bài toán đặt ra không chỉ là sinh đúng cú pháp JSON (như đã kiểm chứng ở mục 3.3), mà nội dung nhiệm vụ phải thực sự mang lại niềm vui và cảm giác thành tựu, tránh gây ức chế do quá khó hoặc nhàm chán do quá dễ.

### *3.3.2 Phương pháp khảo sát*

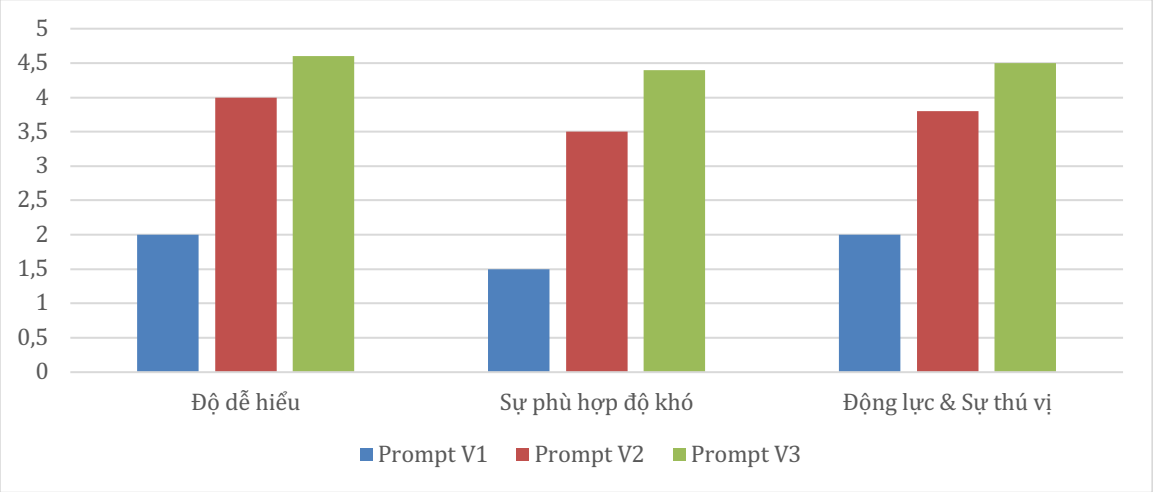
Quá trình khảo sát được thực hiện trên nhóm đối tượng mẫu gồm 20 người tham gia, bao gồm cả người chơi phổ thông, sinh viên chương trình Thiết kế và Phát triển game và chuyên gia thiết kế game (Game Designers) để đảm bảo góc nhìn đa chiều. Phương pháp đánh giá định lượng được áp dụng thông qua thang đo Likert 5 mức độ (1 - Rất tệ đến 5 - Rất tốt). Các người tham gia được trải nghiệm ngẫu nhiên (Blind Test) các bộ nhiệm vụ được sinh ra bởi Prompt V1, V2 và V3 mà không biết trước nguồn gốc, sau đó chấm điểm dựa trên trải nghiệm thực tế trong môi trường game *Merge Fruit*. Quy trình này tuân thủ các chuẩn mực về Nghiên cứu trải nghiệm người dùng (UX Research) và Phân tích hành vi người chơi thường được áp dụng trong phát triển game casual [1], [2].

### *3.3.3 Kết quả khảo sát*

Kết quả thu thập được cho thấy sự tiến bộ rõ rệt về trải nghiệm người dùng tương ứng với sự hoàn thiện của các phiên bản Prompt.

**Bảng 3-3 Bảng tổng hợp kết quả đánh giá trải nghiệm người dùng (Thang đo likert 1-5)**

| Tiêu chí đánh giá | Prompt V1 (Zero-shot) | Prompt V2 (Schema-only) | Prompt V3 (Final) | Mức độ cải thiện (V1 → V3) |
|---|---|---|---|---|
| **1. Độ dễ hiểu** | 2.0 | 4.0 | 4.6 | +130.0% |
| **2. Sự phù hợp độ khó** | 1.5 | 3.5 | 4.4 | +193.3% |
| **3. Động lực & Sự thú vị** | 2.0 | 3.8 | 4.5 | +125.0% |




**Hình 3-1 Biểu đồ tương quan kết quả đánh giá trải nghiệm người dùng**

Về mức độ dễ hiểu (Clarity), Prompt V1 chỉ đạt điểm trung bình thấp là 2.0 /5, phản ánh việc mô tả nhiệm vụ còn mơ hồ hoặc thiếu tính tự nhiên. Tuy nhiên, chỉ số này tăng lên 4.0/5 ở Prompt V2 và đạt mức xuất sắc 4.6/5 ở Prompt V3. Sự gia tăng này phù hợp với lý thuyết về tác động của kỹ thuật Few-shot Prompting trong

việc định hướng văn phong [14]. Nhờ được cung cấp các ví dụ mẫu (Golden Samples), mô hình Gemini Flash ở phiên bản V3 đã học được cách diễn đạt ngắn gọn, súc tích và sử dụng chính xác các thuật ngữ trong game (như tên riêng của quái vật), giúp người chơi dễ dàng nắm bắt mục tiêu.

Về mức độ phù hợp độ khó (Difficulty Fit), tiêu chí ghi nhận sự cải thiện mạnh mẽ nhất. Prompt V1 bị đánh giá thấp (1.5/5) do thường xuyên sinh ra các nhiệm vụ bất khả thi hoặc quá tầm thường. Ngược lại, Prompt V3 đạt điểm 4.4/5, tiệm cận với chất lượng thiết kế thủ công của con người. Kết quả này là minh chứng thực nghiệm cho hiệu quả của kỹ thuật Rule-based Contextualization (Ngữ cảnh hoá dựa trên luật). Việc nạp các quy tắc cân bằng vào ngữ cảnh đã giúp mô hình "hiểu" được ranh giới giữa thử thách và kỹ năng, đảm bảo nhiệm vụ nằm trong vùng dòng chảy (Flow Zone) của người chơi theo nguyên tắc thiết kế reward-pacing [3].

Về khả năng tạo động lực (Motivation), Prompt V3 đạt điểm số ấn tượng 4.5/5 về khả năng kích thích người chơi thực hiện nhiệm vụ. Các người tham gia phản hồi tích cực về sự đa dạng của các loại hình nhiệm vụ (từ thu thập, hợp nhất đến sử dụng vật phẩm) và giá trị phần thưởng tương xứng. Điều này khẳng định rằng hệ thống AI, khi được tối ưu hóa ngữ cảnh đầy đủ, hoàn toàn có khả năng duy trì các vòng lặp dopamine hiệu quả, đóng góp trực tiếp vào mục tiêu tăng trưởng chỉ số retention cho sản phẩm.

## 3.4 Đánh giá hiệu quả sản xuất và chi phí

Bên cạnh các chỉ số về độ chính xác kỹ thuật và trải nghiệm người dùng, tính hiệu quả trong vận hành (Operational Efficiency) là yếu tố quyết định khả năng ứng dụng của mô hình trong môi trường doanh nghiệp. Phần này tiến hành so sánh năng suất lao động giữa quy trình thiết kế thủ công truyền thống và quy trình tự động hóa bằng AI đã đề xuất.

### *3.4.1 So sánh năng suất: Con người vs AI*

Theo nghiên cứu của Li (2024), việc thiết kế nhiệm vụ trong game casual, dù không yêu cầu độ phức tạp cao về cốt truyện, nhưng lại đòi hỏi sự tỉ mỉ trong việc cân bằng các thông số số học (numerical balancing) và sự sáng tạo liên tục để tránh

lặp lại [1]. Trong quy trình truyền thống, một nhân viên thiết kế game (Game Designer) phải thực hiện thủ công các bước: lên ý tưởng, viết mô tả, tính toán chỉ số, nhập liệu vào Excel/JSON và kiểm thử.

Thực nghiệm đối chứng được tiến hành trên cùng một khối lượng công việc là tạo ra 10 nhiệm vụ hằng ngày. Kết quả ghi nhận sự chênh lệch đáng kể về thời gian và nguồn lực:

**Bảng 3-4 Bảng so sánh hiệu suất giữa con người và AI trong việc tạo nhiệm vụ game (Thang đo likert 1-5)**

| **Tiêu chí so sánh** | **Quy trình thủ công** *(Game Designer)* | **Quy trình tự động hóa** *(Gemini Flash + Validation)* | **Mức độ tối ưu** |
|---|---|---|---|
| **Thời gian hoàn thành** | 20 - 30 phút | 3 – 5 phút | **~ 90%** |
| **Tính đa dạng nội dung** | Phụ thuộc vào sức sáng tạo cá nhân (dễ bị lặp lại) | Cao (nhờ tham số Temperature và Over-generation) | **Cao** |
| **Tỷ lệ lỗi kỹ thuật** | Thấp (nhưng dễ mắc lỗi nhập liệu typo) | Thấp (sau khi qua lớp Validation Layer) | **Tương đương** |
| **Khả năng mở rộng** | Khó (tốn thêm nhân sự nếu cần 1000 task) | Dễ dàng (chỉ cần lặp lại vòng lặp API) | **Vượt trội** |

Từ bảng so sánh trên có thể thấy hệ thống AI đề xuất giúp rút ngắn khoảng 85 - 90% tổng thời gian phát triển nội dung. Nếu như con người mất trung bình 2.5 - 3 phút để hoàn thiện một nhiệm vụ từ khâu ý tưởng đến nhập liệu, thì hệ thống tích hợp Gemini Flash chỉ mất chưa đầy 20 giây cho mỗi nhiệm vụ (bao gồm cả thời gian sinh, kiểm định và lọc lỗi). Sự cải thiện năng suất này đặc biệt có ý nghĩa trong bối cảnh thị trường game casual đòi hỏi khối lượng nội dung cập nhật liên tục (Live-ops) để duy trì chỉ số Retention, phù hợp với xu hướng "AI hóa" quy trình sản xuất được ghi nhận trong báo cáo thị trường của Newzoo và Sensor Tower [15], [16].

### *3.4.2 Phân tích chi phí - hiệu năng (Cost-Performance):*

Trong bối cảnh ngành công nghiệp game ngày càng cạnh tranh, việc tối ưu hóa chi phí vận hành (OpEx) là yếu tố sống còn, đặc biệt đối với các studio quy mô vừa và nhỏ (Indie/Mid-core). Việc lựa chọn mô hình Gemini 1.5 Flash cho hệ thống sinh nhiệm vụ không chỉ xuất phát từ yêu cầu kỹ thuật về tốc độ phản hồi mà còn là một quyết định chiến lược về mặt kinh tế. Dòng mô hình này được Google thiết kế tối ưu hóa đặc biệt cho các tác vụ tần suất cao và độ trễ thấp, với mức định giá dựa trên triệu token (Cost per Million Tokens) cạnh tranh bậc nhất trên thị trường hiện nay.

Cụ thể, xét về chi phí hạ tầng AI, đơn giá token đầu vào (Input) và đầu ra (Output) của Gemini 2.5 Flash ở mức cực thấp so với các mô hình lớn như GPT-4 hay Claude 3 Opus. Giả sử trung bình mỗi yêu cầu sinh nhiệm vụ (Prompt V3) tiêu tốn khoảng 1.500 token ngữ cảnh đầu vào (bao gồm luật chơi, danh sách enum, ví dụ mẫu) và sinh ra khoảng 200 token đầu ra (JSON kết quả), tổng chi phí để tạo ra một bộ nhiệm vụ hoàn chỉnh chỉ nằm ở mức phần nghìn của một đô la Mỹ. Ngay cả khi nhân rộng quy mô lên hàng chục nghìn lượt sinh mỗi tháng để phục vụ lượng người chơi lớn (Live-ops scaling), tổng chi phí hạ tầng vẫn nằm trong ngưỡng ngân sách không đáng kể.

Khi đặt lên bàn cân so sánh với chi phí nhân sự truyền thống (Man-hour cost), lợi thế kinh tế trở nên rõ rệt. Để tạo ra một lượng nội dung tương đương 1.000 nhiệm vụ chất lượng cao, một nhân viên thiết kế game (Game Designer) với mức lương trung bình cần tiêu tốn hàng chục, thậm chí hàng trăm giờ làm việc để lên ý tưởng, nhập liệu và kiểm tra cân bằng. Chi phí lương chi trả cho khoảng thời gian này gấp hàng trăm lần so với chi phí gọi API của mô hình. Điều này cho phép các studio game giải phóng nguồn lực nhân sự khỏi các công việc lặp lại nhàm chán để tập trung vào các tác vụ sáng tạo giá trị cao hơn, như thiết kế tính năng mới hay tối ưu hóa trải nghiệm cốt lõi. Tựu trung lại, giải pháp tự động hóa này mang lại tỷ suất hoàn vốn (ROI) vượt trội, biến nó trở thành công cụ đắc lực giúp các nhà phát triển tối ưu hóa ngân sách vận hành mà vẫn đảm bảo dòng chảy nội dung phong phú cho người chơi.

# KẾT LUẬN

**Kết quả của nghiên cứu**

Trong khuôn khổ đề tài ***"Nghiên cứu và ứng dụng LLM cho bài toán tạo nhiệm vụ (Quest) trong game casual"***, tác giả đã xây dựng và triển khai thành công một hệ thống tạo nhiệm vụ tự động, góp phần chuyển đổi quy trình sản xuất nội dung game từ thủ công sang hướng tự động hóa thông minh thông qua việc kết hợp lý thuyết về mô hình ngôn ngữ lớn với các kỹ thuật thiết kế hướng cấu trúc. Qua đây, đề tài đã đạt được những kết quả nổi bật trên cả ba phương diện: lý thuyết, hệ thống và thực nghiệm.

Về mặt lý thuyết và quy trình, đề tài đã đề xuất một quy trình tối ưu hóa Prompt (Prompt Optimization Pipeline) dựa trên cơ chế Attention, In-context Learning và Instruction Following qua việc kết hợp ba trụ cột: Ràng buộc cấu trúc (Schema-Constrained) – Ánh xạ ngữ nghĩa (Semantic Mapping) – Ngữ cảnh hoá dựa trên luật (Rule-based Contextualization). Việc áp dụng các kỹ thuật này giúp kiểm soát được đặc tính ngẫu nhiên của LLM, đồng thời đảm bảo đầu ra luôn tuân thủ định dạng và quy tắc trò chơi.

Về mặt hệ thống, đề tài đã thiết kế kiến trúc tích hợp giữa mô hình Gemini 2.5 Flash và nền tảng Unity thông qua lớp trung gian Dify, mang lại hiệu năng phản hồi nhanh, chi phí thấp và khả năng tích hợp trực tiếp với hệ thống game thực tế. Trong đó, Module Kiểm định (Validation Layer) hoạt động như một hàng rào chất lượng cuối cùng, bảo đảm toàn bộ đầu ra đều hợp lệ về mặt kỹ thuật và logic gameplay.

Về hiệu quả thực nghiệm, kết quả kiểm thử thực tế cho thấy:

(i) Tỷ lệ nhiệm vụ hợp lệ đạt 96% ở phiên bản Prompt V3.

(ii) Năng suất sinh nội dung tăng xấp xỉ 90% so với phương pháp thủ công.

(iii) Mức độ chấp nhận của người chơi đạt trung bình 4.5/5 điểm về độ phù hợp và mức độ động lực.

(iv) Chi phí vận hành được tối ưu đáng kể nhờ sử dụng mô hình Flash có chi phí token thấp.

Những kết quả này không chỉ chứng minh tính khả thi của mô hình đề xuất, mà còn mở ra tiềm năng lớn trong việc ứng dụng LLM vào quy trình thiết kế nội dung trò chơi điện tử.

**Kiến nghị đề xuất**

Dựa trên quá trình nghiên cứu và thử nghiệm, đề tài đưa ra một số kiến nghị nhằm hỗ trợ các nhà phát triển game – đặc biệt là các studio Indie và Mid-core – trong việc triển khai Generative AI:

*Thứ nhất,* kết hợp mô hình kiến trúc lai (Hybrid Architecture), không nên hoàn toàn phó mặc cho AI mà cần kết hợp LLM với các lớp kiểm soát logic và cấu trúc (Validation Layer) để bảo đảm độ tin cậy của đầu ra.

*Thứ hai,* ưu tiên hướng huấn luyện phi tham số, việc đầu tư vào kỹ thuật Prompt Engineering và xây dựng dữ liệu ngữ cảnh có thể tiết kiệm chi phí và tăng tính linh hoạt hơn so với việc tinh chỉnh mô hình riêng (Fine-tuning).

*Thứ ba,* lựa chọn mô hình phù hợp với tác vụ, đối với các tác vụ có cấu trúc lặp lại, như sinh nhiệm vụ hàng ngày, nên ưu tiên sử dụng các mô hình gọn nhẹ (như Gemini Flash) để tối ưu chi phí và tốc độ xử lý.

**Hướng nghiên cứu trong tương lai**

Mặc dù hệ thống hiện tại đã xử lý hiệu quả bài toán sinh nhiệm vụ "tĩnh", hướng phát triển tiếp theo sẽ tập trung vào các kỹ thuật "siêu cá nhân hóa" (Hyper-personalization) nhằm nâng cao mức độ thích ứng với người chơi cụ thể. Cụ thể, đề tài có thể phát triển các hướng tiềm năng sau.

Nạp hồ sơ người dùng vào prompt (User Profiling Injection) là một hướng tiềm năng thông qua việc bổ sung thông tin về hành vi người chơi như tần suất chơi, mức độ tiêu dùng, tỷ lệ thắng/thua để sinh nhiệm vụ sát với từng đối tượng. Ví dụ, với người chơi "hardcore", hệ thống có thể sinh các nhiệm vụ có độ khó cao kèm phần thưởng danh hiệu; trong khi đó, người chơi có nguy cơ rời bỏ sẽ nhận nhiệm vụ dễ với phần thưởng hấp dẫn.

Điều chỉnh độ khó động (Dynamic Difficulty Adjustment) bằng cách sử dụng LLM để theo dõi chỉ số kỹ năng người chơi trong thời gian thực cũng là một hướng

đem lại ý nghĩa lớn cho nhà phát triển game, từ đó điều chỉnh targetValue phù hợp nhằm duy trì trải nghiệm mượt mà và hưng phấn (flow state).

**Tổng kết lại,** hệ thống được đề xuất không đơn thuần là một giải pháp kỹ thuật, mà còn đại diện cho tư duy thiết kế game hiện đại: chuyển từ cách làm thủ công sang tiếp cận dựa trên dữ liệu và AI, hướng tới trải nghiệm chơi game được cá nhân hóa cao và được cập nhật liên tục theo hành vi người dùng./.

# DANH MỤC TÀI LIỆU THAM KHẢO

*** Tài liệu tiếng Anh:**

1. Y. Li (2024), *Exploring the Mechanics Design of Casual Games*, Hội nghị ADDT (EAI) 2024eudl.eu.

2. J. Juul (2010), *A Casual Revolution: Reinventing Video Games and Their Players*, MIT Pressjesperjuul.net.

3. S. Kumar và M. N (2025), *"Dopamine Loops and Player Retention: A Study on Reinforcement in Free-to-Play Games"*, Journal of Communication and Management, Vol.4(Spl), pp.56–61researchgate.net.

4. (Công trình nghiên cứu về cấu trúc nhiệm vụ và vòng lặp chính của game casual, kích thích giải phóng dopamine để tăng retention – tham khảo ví dụ của Candy Crush, Clash of Clans, v.v.)

5. A. Vaswani et al. (2017), *"Attention Is All You Need"*, NIPS 2017, pp.5998–6008ar5iv.labs.arxiv.org (kiến trúc Transformer toàn phần attention, không dùng RNN).

6. T. Lin et al. (2022), *"A Survey of Transformers"*, AI Open, 3 (2022), 111–132sciencedirect.com.

7. A. Radford et al. (2018), *"Improving Language Understanding by Generative Pre-Training"* (OpenAI GPT pre-training)cdn.openai.com (mô hình sử dụng Transformer, cải thiện lưu giữ ngữ cảnh dài so với RNN).

8. J. Devlin et al. (2019), *"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"*, NAACL 2019arxiv.org (mô hình BERT dùng mã hóa hai chiều).

9. A. Radford et al. (2018), *"Improving Language Understanding by Generative Pre-Training"*, OpenAI Preprint (GPT-1)cdn.openai.com (mô hình GPT huấn luyện autoregressive một chiều).

10. E. Sharkey et al. (2024), *"A Comparative Study of BERT and GPT"*, arXiv:2405.12990arxiv.org (tóm tắt ưu nhược điểm: BERT 2 chiều, GPT 1 chiều).

11. E. J. Hu et al. (2021), *"LoRA: Low-Rank Adaptation of Large Language Models"*, arXiv:2106.09685arxiv.org (phương pháp LoRA đóng băng trọng số gốc, thêm ma trận rank thấp để tinh chỉnh).

12. T. Dettmers et al. (2023), *"QLoRA: Efficient Finetuning of Quantized LLMs"*, arXiv:2305.14314arxiv.org (giới thiệu QLoRA để tinh chỉnh mô hình lượng tử hoá 4-bit kết hợp LoRA).

13. J. Wei et al. (2022), *"Chain-of-Thought Prompting Elicits Reasoning in Large Language Models"*, NeurIPS 2022arxiv.org (đưa ra các bước suy luận trung gian trong prompt để cải thiện khả năng giải toán/phán đoán của LLM).

14. T. B. Brown et al. (2020), *"Language Models are Few-Shot Learners"*, NeurIPS 2020arxiv.org (nghiên cứu GPT-3: khả năng học ít-shot qua prompt mà không cần fine-tune).

15. Newzoo (2024), *Global Games Market Report 2023 (Free Version)*newzoo.com (dự báo tổng doanh thu toàn cầu game 2023 đạt ~183.9 tỷ USD, tăng ~0.5% so với 2022).

16. Sensor Tower (2025), *Southeast Asia: Mobile Game Market Insights 2025*sensortower.comsensortower.com (SEA Q1/2025: ~1.93 tỷ lượt cài mới, Việt Nam ~329 triệu lượt cài; doanh thu in-app (~IAP) khoảng 625 triệu USD).

17. Xsolla (2025), *"Vietnam's game-winning era"*xsolla.comxsolla.com (doanh thu thị trường game VN: từ 192 triệu USD năm 2017 lên ~1.5 tỷ USD năm 2024; VN đứng đầu thế giới về xuất khẩu game mobile với ~6.7 tỷ lượt tải năm 2025).

18. Zhou, X., Palangi, H., Zhang, L., Gao, J., & Galley, M. (2023). "Large Language Models with Guardrails: A Survey." *arXiv:2308.07905*. https://arxiv.org/abs/2308.07905

19. Ouyang, L. et al. (2022), "Training language models to follow instructions with human feedback", *arXiv preprint*, arXiv:2203.02155. [https://arxiv.org/abs/2203.02155]

20. Wei, J. et al. (2021), "Finetuned Language Models Are Zero-Shot Learners", *arXiv preprint*, arXiv:2109.01652. [https://arxiv.org/abs/2109.01652]

# PHỤ LỤC

## A.1. Bộ tham chiếu

**Bảng A.1-1 Từ điển hoa quả/quái vật trong game**

| Cấp độ (Level) | Tên định danh (Code Name) | Trạng thái (Status) | Đặc tả hành vi (Behavior Spec) |
|---|---|---|---|
| Lv 1 | Rotolasio | Common | Quái cơ bản, xuất hiện nhiều nhất. |
| Lv 2 | Cappi Coppi | Common | Tạo ra từ 2 Lv1. |
| Lv 3 | Hipporino Bombondino | Common | Tạo ra từ 2 Lv2. |
| Lv 4 | Cuppa Assassinoto!! | Common | Tạo ra từ 2 Lv3. |
| Lv 5 | Tapirino Frigorifero | Medium | Bắt đầu chiếm diện tích đáng kể. |
| Lv 6 | Tiki Taka Sahur | Medium | Mốc trung bình của game. |
| Lv 7 | Sharalelo Shalala! | Rare | Khó đạt được trong thời gian ngắn. |
| Lv 8 | Saturno Tyranino! | Rare | Kích thước lớn, dễ gây đầy hộp. |
| Lv 9 | Elephanto Toasterino! | Epic | Mục tiêu của nhiệm vụ khó. |
| Lv 10 | Croccocino Spaghettino | Epic | Cận cấp cuối. |
| Lv 11 | SixSeven | Legendary | Cơ chế đặc biệt: Khi 2 Lv11 hợp nhất, chúng sẽ phát nổ và biến mất (Clear Board). Không tạo ra Lv12. |

**Bảng A.1-2 Từ điển vật phẩm hỗ trợ trong game**

| Loại Booster | Tên gọi | Chức năng (Function) | Ngữ cảnh sử dụng (Context) |
|---|---|---|---|
| Type 1 | Boom Booster | Nổ các quả nhỏ | Dùng để phá hủy các quái Lv1–Lv2 gây kẹt. |
| Type 2 | Hammer Booster | Phá hủy chỉ định | Dùng để xóa 1 quái bất kỳ (thường là quái lớn gây cản trở). |
| Type 3 | Upgrade Booster | Nâng cấp chỉ định | Biến 1 quái Lv(N) thành Lv(N+1) mà không cần hợp nhất. |
| Type 4 | Shakebox Booster | Xáo trộn (Shuffle) | Dùng khi không còn nước đi, làm rung hộp để thay đổi vị trí vật lý. |

**Bảng A.1-3 Từ điển hành động trong game**

| Nhóm (Group) | Hành động (Action) | Mô tả | Enum Mapping |
|---|---|---|---|
| Economy (Kinh tế) | CollectCoins | Thu thập tiền vàng trong quá trình chơi. | DailyTaskName.CollectCoins |
| | WatchClaim | Xem quảng cáo để nhận thưởng/hồi sinh. | DailyTaskName.WatchClaim |
| | Login | Đăng nhập vào game. | DailyTaskName.Login |
| Core Loop (Cốt lõi) | TotalMerges | Thực hiện hành động hợp nhất bất kỳ. | DailyTaskName.TotalMerges |

| | | | |
|---|---|---|---|
| | MergeLevel[X] | Tạo ra quái cấp độ cụ thể (từ 2 đến 11). | DailyTaskName.MergeLevel[X] |
| Skill (Kỹ năng) | TotalCombos | Tạo chuỗi hợp nhất liên tiếp trong 1 lượt thả. | DailyTaskName.TotalCombos |
| | HighScore | Đạt điểm số cao trong một ván chơi. | DailyTaskName.HighScore |
| Support (Hỗ trợ) | UseBooster | Sử dụng vật phẩm hỗ trợ. | DailyTaskName.UseBooster[X] |

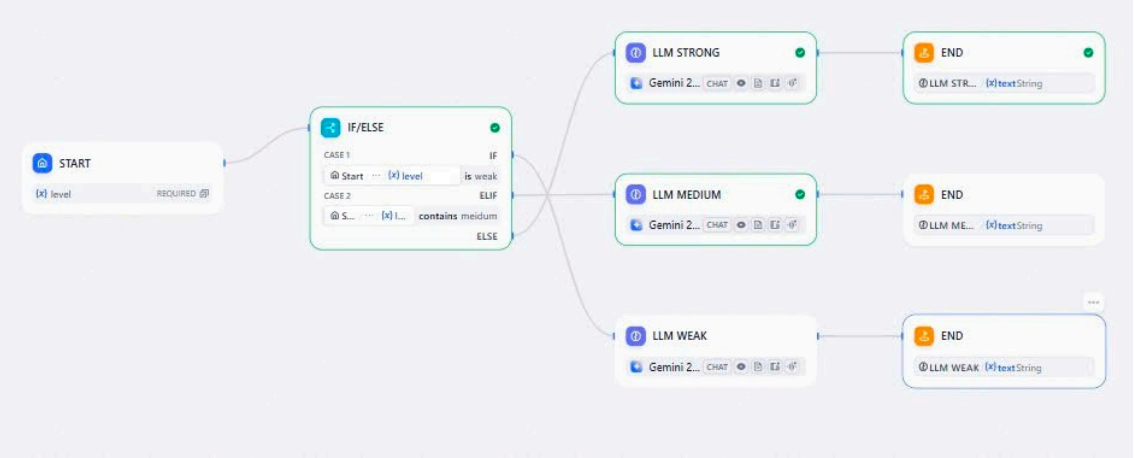
**Bảng A.1-4 Bộ quy tắc cân bằng Phần thưởng – Độ khó trong game**

| Độ khó | Mức độ | Phần thưởng |
|---|---|---|
| Dễ | Login, WatchClaim, Merge đối tượng cấp thấp | 10 – 20 coin |
| Trung bình | Combo ngắn, Merge đối tượng cấp trung bình | 20 – 30 coin |
| Khó | HighScore, MaxLevel (Lv9+), Combo dài | 30 – 50 coin hoặc 1 booster |

**Bảng A.1-5 Các giới hạn Gameplay (Gameplay Constraints)**

| Tham số (Parameter) | Giới hạn trung bình/ngày (Average Cap) | Giới hạn cứng (Hard Limit) | Ghi chú cho Validation |
|---|---|---|---|
| Max Level Reachable | Lv 9 - Lv 10 | Lv 11 (SixSeven) | targetValue cho Lv11 tối đa là 1 hoặc 2. |
| Total Merges | 100 - 150 merges | Không giới hạn | targetValue nên nằm trong khoảng 20 - 80. |
| Max Combo Chain | 3 - 5 combos | ~12 combos | targetValue cho Combo không nên vượt quá 5. |
| Ads Watchable | 3 - 5 ads | 10 ads | Không sinh nhiệm vụ "Watch 15 Ads". |
| Booster Usage | 2 - 3 boosters | Tùy thuộc số Coin | targetValue cho Booster không nên vượt quá 5. |

## A.2. Workflow triển khai bằng Dify




**Hình A.2-1 Workflow triển khai huấn luyện LLM và backend bằng công cụ Dify**

## A.3. Các phiên bản Prompt

**Bảng A.3-1 Các phiên bản Prompt**

| Phiên bản | Prompt |
|---|---|
| Prompt V1 | *Bạn là một chuyên gia thiết kế game casual. Hãy nghĩ ra 5 nhiệm vụ hằng ngày (daily quests) thú vị cho trò chơi có tên là "Merge Fruit" (trò chơi thả trái cây giống Suika Game).*<br><br>*Yêu cầu:*<br>*1. Các nhiệm vụ cần khuyến khích người chơi tương tác với game.*<br>*2. Mỗi nhiệm vụ cần có tên, mô tả, mục tiêu cần đạt và phần thưởng.*<br>*3. Hãy trả về kết quả dưới dạng JSON để tôi có thể dùng trong code.* |
| Prompt V2 | *Bạn là hệ thống sinh nhiệm vụ tự động (AI Task Generator). Nhiệm vụ của bạn là tạo ra 5 nhiệm vụ hằng ngày cho game "Merge Fruit" và trả về kết quả dưới dạng JSON hợp lệ duy nhất.*<br><br>*Vui lòng tuân thủ nghiêm ngặt cấu trúc JSON sau:*<br>*{*<br>*"dailyTasks": [*<br>*{*<br>*"taskName": <integer: giá trị enum>,*<br>*"taskDes": <string: mô tả nhiệm vụ>,*<br>*"targetValue": <integer: số lượng cần đạt>,*<br>*"rewardDaily": <integer: 0 là Coin, 1 là Booster>,*<br>*"numReward": <integer: số lượng phần thưởng>*<br>*}*<br>*]*<br>*}*<br><br>*Sử dụng bảng mã Enum (Mapping) sau cho taskName:* |

| | |
|---|---|
| | *0: WatchClaim (Xem quảng cáo)*<br>*1: Login (Đăng nhập)*<br>*2: CollectCoins (Thu thập tiền)*<br>*3: UseBooster (Dùng vật phẩm)*<br>*8: MergeFruit (Hợp nhất trái cây)*<br>*19: ReachLevel (Đạt cấp độ quái)*<br>*20: Combo (Tạo combo)*<br>*23: HighScore (Đạt điểm cao)*<br><br>*Yêu cầu ràng buộc:*<br>*- Chỉ trả về JSON thuần túy, không kèm Markdown hay giải thích.*<br>*- taskDes phải ngắn gọn (dưới 10 từ).*<br>*- Sử dụng đúng ID trong danh sách Enum ở trên.* |
| Prompt V3 | *Bạn là chuyên gia thiết kế nhiệm vụ hằng ngày (Daily Task Designer) cho game mobile thể loại Puzzle/Physics/Merge có tên là "Merge Fruit – Hoa Quả".*<br><br>*Hãy tạo ra 5 nhiệm vụ hằng ngày (daily task) mới ở dạng JSON hoàn toàn hợp lệ theo đúng cấu trúc sau:*<br>*{*<br>*"dailyTasks": [*<br>*{*<br>*"taskName": <giá trị số trong enum DailyTaskName>,*<br>*"taskDes": "1 câu mô tả ngắn gọn, rõ ràng về nhiệm vụ, tối đa 8 từ",*<br>*"targetValue": <giá trị mục tiêu cần đạt>,*<br>*"rewardDaily": <giá trị số trong enum RewardDaily>,*<br>*"numReward": <số lượng phần thưởng>*<br>*}* |

| | |
|---|---|
| | *]*<br>*}*<br><br>*Giải thích ý nghĩa enum DailyTaskName (giá trị số tương ứng index của enum):*<br><br>*0 - WatchClaim – Xem quảng cáo để nhận thưởng*<br>*Ví dụ: "Watch 3 video ads"*<br><br>*1 - Login – Đăng nhập game hằng ngày*<br>*Ví dụ: "Login once per day"*<br><br>*2 - CollectCoins – Thu thập coin trong game*<br>*Ví dụ: "Collect 250 coins"*<br><br>*3 - UseBooster1 – Dùng Boom Booster (bom nhỏ)*<br>*Ví dụ: "Use 5 boom boosters"*<br><br>*4 - UseBooster2 – Dùng Hammer Booster*<br>*Ví dụ: "Use 3 hammers to destroy fruits"*<br><br>*5 - UseBooster3 – Dùng Upgrade Booster*<br>*Ví dụ: "Use 2 fruit upgrades"*<br><br>*6 - UseBooster4 – Dùng Shakebox Booster*<br>*Ví dụ: "Use 1 shakebox"*<br><br>*7 - TotalBoostersUsed – Tổng số booster đã dùng (mọi loại)*<br>*Ví dụ: "Use 10 boosters in total"* |

| | |
|---|---|
| | *8 - TotalMerges – Tổng số lượt merge trái cây/quái*<br>*Ví dụ: "Merge 60 times"*<br><br>*9 - MergeLevel2 – Merge tạo quái Level 2 (Cappi Coppi)*<br>*Ví dụ: "Create 8 Cappi Coppi"*<br><br>*10 - MergeLevel3 – Merge tạo quái Level 3 (Hipporino Bombondino)*<br>*Ví dụ: "Create 6 Hipporino Bombondino"*<br><br>*11 - MergeLevel4 – Merge tạo quái Level 4 (Cuppa Assassinoto!!)*<br>*Ví dụ: "Create 5 Cuppa Assassinoto!!"*<br><br>*12 - MergeLevel5 – Merge tạo quái Level 5 (Tapirino Frigorifero)*<br>*Ví dụ: "Create 4 Tapirino Frigorifero"*<br><br>*13 - MergeLevel6 – Merge tạo quái Level 6 (Tiki Taka Sahur)*<br>*Ví dụ: "Create 3 Tiki Taka Sahur"*<br><br>*14 - MergeLevel7 – Merge tạo quái Level 7 (Sharalelo Shalala!)*<br>*Ví dụ: "Create 2 Sharalelo Shalala!"*<br><br>*15 - MergeLevel8 – Merge tạo quái Level 8 (Saturno Tyranino!)*<br>*Ví dụ: "Create 2 Saturno Tyranino!"* |

| | |
|---|---|
| | *16 - MergeLevel9 – Merge tạo quái Level 9 (Elephanto Toasterino!)*<br>*Ví dụ: "Create 2 Elephanto Toasterino!"*<br><br>*17 - MergeLevel10 – Merge tạo quái Level 10 (Croccocino Spaghettino)*<br>*Ví dụ: "Create 1 Croccocino Spaghettino"*<br><br>*18 - MergeLevel11 – Merge tạo quái Level 11 (SixSeven)*<br>*Ví dụ: "Create 1 SixSeven"*<br><br>*19 - MaxLevelReached – Level quái cao nhất đạt được trong ngày*<br>*Ví dụ: "Reach level 11 (SixSeven)"*<br><br>*20 - TotalCombos – Tổng số combo đạt được*<br>*Ví dụ: "Achieve 20 combos"*<br><br>*21 - Combo5Plus – Số combo có độ dài ≥ 5 merge*<br>*Ví dụ: "Achieve 10 combos ≥5"*<br><br>*22 - MaxComboReached – Combo cao nhất đạt được*<br>*Ví dụ: "Reach combo 10"*<br><br>*23 - HighScore – Điểm cao trong một lượt chơi*<br>*Ví dụ: "Score 5000 points in one game"*<br><br>*24 - GainScore – Tổng điểm tích lũy trong ngày* |

| | |
|---|---|
| | *Ví dụ: "Accumulate 10000 points total"*<br><br>*Cấu trúc tham chiếu trong code Unity:*<br>*[System.Serializable]*<br>*public enum DailyTaskName*<br>*{*<br>*WatchClaim,*<br>*Login,*<br>*CollectCoins,*<br>*UseBooster1,*<br>*UseBooster2,*<br>*UseBooster3,*<br>*UseBooster4,*<br>*TotalBoostersUsed,*<br>*TotalMerges,*<br>*MergeLevel2,*<br>*MergeLevel3,*<br>*MergeLevel4,*<br>*MergeLevel5,*<br>*MergeLevel6,*<br>*MergeLevel7,*<br>*MergeLevel8,*<br>*MergeLevel9,*<br>*MergeLevel10,*<br>*MergeLevel11,*<br>*MaxLevelReached,*<br>*TotalCombos,*<br>*Combo5Plus,* |

| | |
|---|---|
| | *MaxComboReached,*<br>*HighScore,*<br>*GainScore,*<br>*}*<br><br>*[System.Serializable]*<br>*public enum RewardDaily*<br>*{*<br>*Coin = 0,*<br>*Booster = 1*<br>*}*<br><br>*[System.Serializable]*<br>*public class DailyTask*<br>*{*<br>*public DailyTaskName taskName;*<br>*public string taskDes;*<br>*public int targetValue;*<br>*public RewardDaily rewardDaily;*<br>*public int numReward;*<br>*}*<br><br>*Quy tắc & hướng dẫn tạo nhiệm vụ:*<br>*Nội dung daily task phải bám sát gameplay của Merge Fruit – Hoa Quả, cụ thể:*<br>*- Merge trái cây / quái cùng level để tạo con lớn hơn.*<br>*- Tạo combo nhiều merge liên tiếp.*<br>*- Dùng booster để vượt tình huống khó.* |

| | |
|---|---|
| | *- Thu thập coin, đạt level quái cao, hoặc tạo quái đặc biệt (level 9–11).*<br>*- Có thể khai thác các hành động thường ngày như: đăng nhập (Login), xem quảng cáo nhận thưởng (WatchClaim), tăng điểm (GainScore), v.v.*<br><br>*Phải đa dạng hành động, gắn với enum DailyTaskName:*<br>*- Ít nhất 1 task về thu thập coin (CollectCoins).*<br>*- Ít nhất 1 task về merge (TotalMerges hoặc MergeLevel2–MergeLevel11 hoặc MaxLevelReached).*<br>*- Ít nhất 1 task về combo (TotalCombos, Combo5Plus hoặc MaxComboReached).*<br>*- Ít nhất 1 task về booster (UseBooster1–UseBooster4 hoặc TotalBoostersUsed).*<br>*- Ít nhất 1 task về tiến hóa / level cao (ưu tiên MergeLevel9–MergeLevel11 hoặc MaxLevelReached).*<br>*- Có thể thêm các task tiện lợi như: Login, WatchClaim, HighScore, GainScore để tăng tính đa dạng.*<br><br>*Độ khó tăng dần, phần thưởng hợp lý (dùng rewardDaily + numReward):*<br>*- rewardDaily = Coin:*<br>*- Task dễ: numReward từ 10–20.*<br>*- Task trung bình: numReward từ 20–30.*<br>*- Task khó: numReward từ 30–40.*<br>*- rewardDaily = Booster:*<br>*- Task dễ: numReward 1 booster.*<br>*- Task trung bình: numReward 2 boosters.*<br>*- Task khó: numReward 3 boosters.* |

| | |
|---|---|
| | *5 task phải có độ khó và mức thưởng tăng dần rõ rệt.*<br><br>*Ngôn ngữ: Tiếng Anh (ngắn gọn, dễ hiểu, vui tươi, casual-friendly).*<br>*Ví dụ:*<br>*- "Merge 80 fruits in total"*<br>*- "Reach monster level 10 once"*<br>*- "Use any boosters 5 times"*<br>*- "Make 5 combos of 5 merges"*<br>*- "Collect 300 coins today"*<br><br>*Kết quả trả về:*<br>*- Chỉ trả về JSON hợp lệ duy nhất, không kèm mô tả, ghi chú, hoặc giải thích.*<br>*- Dữ liệu có thể copy trực tiếp vào file JSON trong Unity.*<br><br>*📘 Thông tin tham khảo về game Merge Fruit – Hoa Quả:*<br>*Thể loại: Puzzle / Physics / Merge (dựa trên cơ chế Suika Game).*<br><br>*Cơ chế chính:*<br>*- Người chơi thả trái cây / quái từ trên xuống.*<br>*- Hai con cùng level va chạm → merge thành con level cao hơn.*<br>*- Con level 11 khi merge sẽ biến mất (giải phóng không gian).*<br>*- Booster gồm 4 loại:*<br>*- Nổ nhỏ – phá quả / quái level 1–2.*<br>*- Nổ chỉ định – phá quả / quái được chọn.*<br>*- Nâng cấp chỉ định – tăng level quả / quái được chọn.*<br>*- Xáo trộn – làm rung hộp và xáo trộn vị trí.*<br>*- Coin dùng để mua booster hoặc mở khóa theme.* |

| | |
|---|---|
| | *- Mục tiêu: tạo con level 11, giải phóng không gian và đạt điểm cao.*<br>*- Game kết thúc khi hộp đầy không thể spawn thêm.*<br><br>🐾 *Cấp độ các con (monster levels):*<br>*lv1 = Rotolasio*<br>*lv2 = Cappi Coppi*<br>*lv3 = Hipporino Bombondino*<br>*lv4 = Cuppa Assassinoto!!*<br>*lv5 = Tapirino Frigorifero*<br>*lv6 = Tiki Taka Sahur*<br>*lv7 = Sharalelo Shalala!*<br>*lv8 = Saturno Tyranino!*<br>*lv9 = Elephanto Toasterino!*<br>*lv10 = Croccocino Spaghettino*<br>*lv11 = SixSeven*<br>*Khi mô tả nhiệm vụ liên quan đến level quái, có thể tham chiếu tên gọi vui nhộn của từng level (ví dụ: "Reach Elephanto Toasterino once", "Create SixSeven one time").*<br>🎯 *Yêu cầu cuối:*<br>*- Tạo ra 5 daily task duy nhất, mỗi task có:*<br>*- Hành động khác nhau (khác taskName hoặc khác loại hành vi).*<br>*- Mức thưởng và độ khó tăng dần.*<br>*- taskDes (mô tả) thú vị, mang phong cách nhẹ nhàng, vui tươi, casual-friendly.*<br>*- Trả về toàn bộ kết quả duy nhất ở dạng JSON hợp lệ, không có bất kỳ văn bản mô tả nào khác.* |

## A.4. Đầu ra của mô hình ứng với 3 phiên bản prompt

**Bảng A.4-1 Đầu ra của các phiên bản prompt**

| Phiên bản | Đầu ra |
|---|---|
| **Prompt V1** | Sure! Here are 5 fun daily quests for your Merge Fruit game:<br><br>{<br>  "quests": [<br>  {<br>  "quest_name": "Merge 5 Apples",<br>  "description": "Combine small fruits to make apples",<br>  "goal": 5,<br>  "reward": "100 Gold"<br>  },<br>  {<br>  "quest_name": "Share to Facebook",<br>  "description": "Share your high score with friends",<br>  "goal": 1,<br>  "reward": "1 Diamond"<br>  },<br>  {<br>  "quest_name": "Use Sword Item",<br>  "description": "Slice a fruit using the Sword booster",<br>  "goal": 3,<br>  "reward": "50 Gold"<br>  },<br>  {<br>  "quest_name": "Login Bonus",<br>  "description": "Log in to the game",<br>  "goal": 1,<br>  "reward": "Free Spin"<br>  },<br>  {<br>  "quest_name": "Eat a fruit",<br>  "description": "Let the monster eat a fruit",<br>  "goal": 10,<br>  "reward": "20 Gold"<br>  }<br>  ]<br>} |

**Prompt V2**


```
{
  "dailyTasks": [
    {
      "taskName": 1,
      "taskDes": "Login to game",
      "targetValue": 1,
      "rewardDaily": 0,
      "numReward": 5000
    },
    {
      "taskName": 19,
      "taskDes": "Reach Level 15 Monster",
      "targetValue": 1,
      "rewardDaily": 0,
      "numReward": 10
    },
    {
      "taskName": 3,
      "taskDes": "Use 50 Boosters",
      "targetValue": 50,
      "rewardDaily": 1,
      "numReward": 1
    },
    {
      "taskName": 8,
      "taskDes": "Kill 10 monsters",
      "targetValue": 10,
      "rewardDaily": 0,
      "numReward": 20
    },
    {
      "taskName": 23,
      "taskDes": "Get High Score",
      "targetValue": 1000000000,
      "rewardDaily": 0,
      "numReward": 5
    }
  ]
}
```

**Prompt V3**


```
{
  "dailyTasks": [
    {
      "taskName": 1,
      "taskDes": "Login to the game today",
      "targetValue": 1,
      "rewardDaily": 0,
      "numReward": 15
    },
    {
      "taskName": 12,
      "taskDes": "Create 3 Tapirino Frigorifero",
      "targetValue": 3,
      "rewardDaily": 0,
      "numReward": 25
    },
    {
      "taskName": 6,
      "taskDes": "Use Shakebox booster 1 time",
      "targetValue": 1,
      "rewardDaily": 1,
      "numReward": 1
    },
    {
      "taskName": 21,
      "taskDes": "Achieve 5 combos of 5+ merges",
      "targetValue": 5,
      "rewardDaily": 0,
      "numReward": 40
    },
    {
      "taskName": 18,
      "taskDes": "Create SixSeven monster once",
      "targetValue": 1,
      "rewardDaily": 1,
      "numReward": 3
    }
  ]
}
```

## A.5. Bảng tổng hợp lỗi tương ứng với từng layer validation

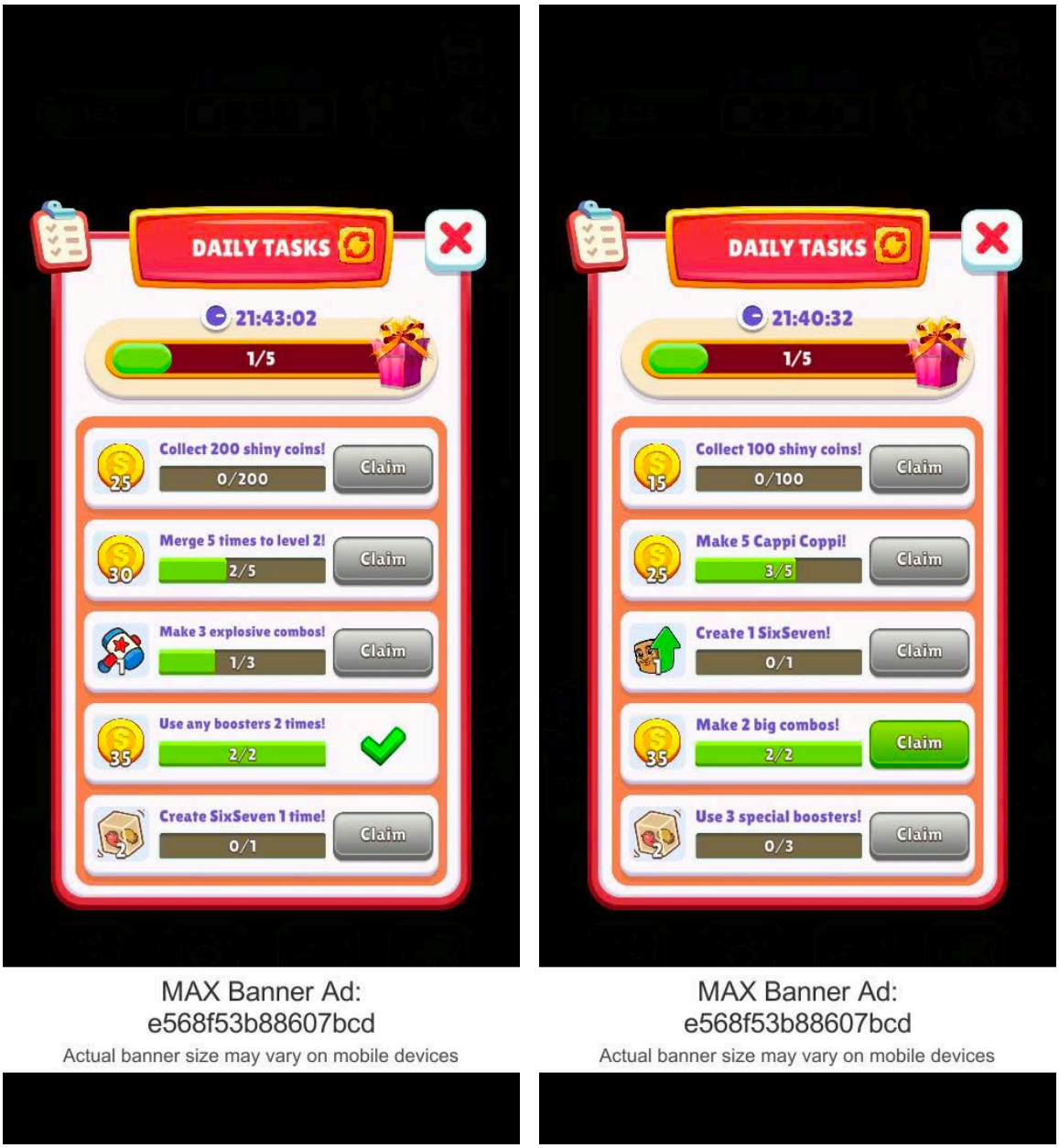
**Bảng A.5-1 Bảng tổng hợp lỗi giữa các phiên bản Prompt**

| Chỉ số kỹ thuật (Metrics) | Prompt V1 (Baseline) | Prompt V2 (Structure-Constrained) | Prompt V3 (Context-Aware) |
|---|---|---|---|
| **1. Tính hợp lệ cấu trúc** *(Structural Validity)* *Khả năng Parse JSON tự động* | **X KHÔNG ĐẠT**<br><br>**Lỗi trong mẫu:**<br>- Chứa văn bản dẫn nhập thừa (*"Sure! Here are..."*).<br>- Sai tên trường dữ liệu (quest_name thay vì taskName).<br><br>**Hệ quả:** Hệ thống Unity sẽ báo lỗi JSON Parse Error ngay lập tức. | **V ĐẠT**<br><br>**Kết quả trong mẫu:**<br>- JSON thuần túy, không có văn bản thừa.<br>- Đúng tên trường (taskName), đúng cấu trúc mảng.<br><br>**Hệ quả:** Deserialize thành công vào class C#. | **V ĐẠT**<br><br>**Kết quả trong mẫu:**<br>- JSON thuần túy, cấu trúc chuẩn xác tuyệt đối.<br><br>**Hệ quả:** Deserialize thành công, sẵn sàng sử dụng. |
| **2. Tính tuân thủ ngữ nghĩa** | **X KHÔNG ĐẠT** | **V ĐẠT** | **V ĐẠT** |

| | | | |
|---|---|---|---|
| *(Semantic Compliance)*<br>*Khả năng ánh xạ Enum (0-24)* | **Lỗi trong mẫu:**<br>- Trả về chuỗi ký tự tự do (*"Merge 5 Apples"*) thay vì mã số.<br>- Trả về phần thưởng dạng text (*"100 Gold"*).<br><br>**Hệ quả:** Code không thể ánh xạ hành động vào logic game<br><br>. | **Kết quả trong mẫu:**<br>- Sử dụng đúng số nguyên cho Enum (taskName: 1, taskName: 19).<br>- Đúng kiểu dữ liệu (rewardDaily: 0).<br><br>**Hệ quả:** Hệ thống hiểu đúng loại hành động (Ví dụ: 1 là Login).<br><br>**V ĐẠT** | **Kết quả trong mẫu:**<br>- Sử dụng chính xác mã số định danh (taskName: 12, taskName: 6).<br><br>**Hệ quả:** Hệ thống hiểu đúng và xử lý chính xác hành động. |
| **3. Tính hợp lệ Logic & Gameplay**<br>*(Gameplay Feasibility)*<br>*Tính khả thi và cân bằng* | **X KHÔNG ĐẠT**<br><br>**Lỗi trong mẫu:** | **! CẢNH BÁO (Lỗi Logic)**<br><br>**Lỗi trong mẫu:** | **V ĐẠT (Tối ưu)**<br><br>**Kết quả trong mẫu:**<br>- Chính xác: Gọi đúng tên quái Level 5 (*Tapirino* |

| | | | |
|---|---|---|---|
| | - Ảo giác (Hallucination): Sinh vật phẩm "Sword" (không tồn tại).<br>- Sai hành vi: Yêu cầu "Share Facebook" (game không có tính năng này). | - Bất khả thi: Yêu cầu đạt *Level 15* (Game max Level 11).<br>- Mất cân bằng: Task Login (dễ) thưởng *5000* (lạm phát kinh tế).<br>- Trôi ngữ nghĩa: TaskName 8 (Merge) nhưng mô tả là "Kill monsters". | *Frigorifero*) và Level 11 (*SixSeven*).<br>- Cân bằng: Task Login thưởng 15 coin (hợp lý).<br>- Đa dạng: Phối hợp tốt giữa Merge, Combo và Booster. |

## A.6. Giao diện hệ thống quest game đã tích hợp API LLM tạo Quest




**Hình A.6-1 Giao diện câu hỏi trò chơi được tạo ra bởi LLM đã tích hợp vào game**

## A.7. Bảng hỏi đánh giá nội dung Quest tạo bởi LLM

**Bảng A.7-1 Bảng hỏi đánh giá các nội dung Quest được tạo bởi LLM**

**PHIẾU KHẢO SÁT ĐÁNH GIÁ TRẢI NGHIỆM NHIỆM VỤ GAME**

**(Dành cho dự án nghiên cứu ứng dụng AI trong Game Casual)**

Xin chào bạn,

Tôi đang thực hiện nghiên cứu về việc ứng dụng trí tuệ nhân tạo (AI) để tạo nội dung tự động cho game Merge Fruit (thể loại giải đố/hợp nhất). Khảo sát này nhằm mục đích đánh giá chất lượng của các nhiệm vụ dựa trên trải nghiệm thực tế của bạn.

Thông tin của bạn sẽ được bảo mật và chỉ phục vụ cho mục đích nghiên cứu.

***A. PHẦN 1: THÔNG TIN CƠ BẢN***

*(Giúp phân loại đối tượng người chơi)*

**1. Bạn đã từng chơi các game thể loại Casual / Merge (như Suika Game, Candy Crush, 2048) chưa?**

[ ] Chưa từng chơi

[ ] Thỉnh thoảng chơi

[ ] Chơi thường xuyên

**2. Bạn quan tâm điều gì nhất khi làm nhiệm vụ hằng ngày?**

[ ] Phần thưởng (Coin/Item)

[ ] Thử thách / Độ khó

[ ] Sự rõ ràng, dễ hiểu

[ ] Cảm giác hoàn thành (Achievement)

***B. PHẦN 2: TRẢI NGHIỆM VÀ ĐÁNH GIÁ (BLIND TEST)***

*Hướng dẫn: Dưới đây là 3 nhóm nhiệm vụ (ký hiệu là Nhóm A, Nhóm B, Nhóm C) được sinh ra từ các phiên bản hệ thống khác nhau. Hãy đọc (hoặc chơi thử) và đánh giá từng nhóm dựa trên thang điểm 1 đến 5.*

**Thang điểm:**

**1:** Rất tệ / Không đồng ý **2:** Tệ **3:** Bình thường **4:** Tốt **5:** Rất tốt / Hoàn toàn đồng ý

**a) ĐÁNH GIÁ NHÓM NHIỆM VỤ A**

- **Merge 5 Apples:** Kết hợp trái cây nhỏ để tạo ra táo. (Phần thưởng: 100 Gold)
- **Share to Facebook:** Chia sẻ điểm cao lên Facebook. (Phần thưởng: 1 Diamond)

- **Use Sword Item:** Dùng kiếm chém hoa quả. (Phần thưởng: 50 Gold)
- **Eat a fruit:** Cho quái vật ăn trái cây. (Phần thưởng: 20 Gold)
- **Login Bonus:** Đăng nhập game. (Phần thưởng: Free Spin)

| Tiêu chí đánh giá | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| **1. Độ dễ hiểu (Clarity):** Ngôn ngữ rõ ràng, không gây nhầm lẫn. | ☐ | ☐ | ☐ | ☐ | ☐ |
| **2. Độ phù hợp (Difficulty Fit):** Mục tiêu khả thi, không quá vô lý. | ☐ | ☐ | ☐ | ☐ | ☐ |
| **3. Động lực (Motivation):** Nhiệm vụ thú vị, muốn làm để nhận thưởng. | ☐ | ☐ | ☐ | ☐ | ☐ |
| **4. Logic Game:** Nhiệm vụ đúng với tính năng của game Merge Fruit. | ☐ | ☐ | ☐ | ☐ | ☐ |

## b) ĐÁNH GIÁ NHÓM NHIỆM VỤ B

- **Login:** Đăng nhập vào game. (Phần thưởng: 5000 Coin)
- **Reach Level 15 Monster:** Tạo ra quái vật Level 15. (Phần thưởng: 10 Coin)
- **Use 50 Boosters:** Dùng 50 vật phẩm hỗ trợ trong một ngày. (Phần thưởng: 1 Booster)
- **Kill 10 monsters:** Tiêu diệt 10 con quái. (Phần thưởng: 20 Coin)
- **Get High Score:** Đạt 1 tỷ điểm. (Phần thưởng: 5 Coin)

| Tiêu chí đánh giá | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| **1. Độ dễ hiểu (Clarity):** Ngôn ngữ rõ ràng, không gây nhầm lẫn. | ☐ | ☐ | ☐ | ☐ | ☐ |
| **2. Độ phù hợp (Difficulty Fit):** Mục tiêu khả thi, không quá vô lý. | ☐ | ☐ | ☐ | ☐ | ☐ |
| **3. Động lực (Motivation):** Nhiệm vụ thú vị, muốn làm để nhận thưởng. | ☐ | ☐ | ☐ | ☐ | ☐ |
| **4. Logic Game:** Nhiệm vụ đúng với tính năng của game Merge Fruit. | ☐ | ☐ | ☐ | ☐ | ☐ |

## c) ĐÁNH GIÁ NHÓM NHIỆM VỤ C

- **Daily Login:** Đăng nhập game hôm nay. (Phần thưởng: 15 Coin)

- **Create Tapirino Frigorifero:** Tạo ra 3 quái Level 5 (Tapirino). (Phần thưởng: 25 Coin)
- **Use Shakebox:** Dùng vật phẩm Rung Hộp 1 lần. (Phần thưởng: 1 Booster Rung)
- **Combo Master:** Đạt 5 chuỗi Combo liên tiếp. (Phần thưởng: 40 Coin)
- **Reach SixSeven:** Tạo ra quái Level 11 (SixSeven) một lần. (Phần thưởng: 3 Boosters)

| **Tiêu chí đánh giá** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|
| **1. Độ dễ hiểu (Clarity):** Ngôn ngữ rõ ràng, không gây nhầm lẫn. | ☐ | ☐ | ☐ | ☐ | ☐ |
| **2. Độ phù hợp (Difficulty Fit):** Mục tiêu khả thi, không quá vô lý. | ☐ | ☐ | ☐ | ☐ | ☐ |
| **3. Động lực (Motivation):** Nhiệm vụ thú vị, muốn làm để nhận thưởng. | ☐ | ☐ | ☐ | ☐ | ☐ |
| **4. Logic Game:** Nhiệm vụ đúng với tính năng của game Merge Fruit. | ☐ | ☐ | ☐ | ☐ | ☐ |

## *C. PHẦN 3: SO SÁNH TỔNG QUAN*

**1. Theo bạn, nhóm nhiệm vụ nào mang lại trải nghiệm tốt nhất (giống game thật nhất)?**

[ ] Nhóm A

[ ] Nhóm B

[ ] Nhóm C

**2. Bạn có cảm thấy Nhóm C (hoặc nhóm tốt nhất bạn chọn) đủ chất lượng để đưa vào game chính thức không?**

[ ] Có, hoàn toàn đủ chất lượng.

[ ] Tạm được, cần chỉnh sửa thêm.

[ ] Không, chất lượng quá tệ.

**3. Ý kiến đóng góp thêm (nếu có):**

.....

.....

**Xin chân thành cảm ơn sự tham gia của bạn!**

## A.8. Bảng so sánh chi tiết về Chi phí và Hiệu năng khi sử dụng LLM để tạo Quest

**Bảng A.8-1 Bảng so sánh về Chi phí và Hiệu năng khi sử dụng LLM để tạo quest**

| Tiêu chí so sánh (Criteria) | Phương pháp thủ công (Game Designer) | Phương pháp tự động hóa (Gemini 2.5 Flash) | Đánh giá hiệu quả (Impact) |
|---|---|---|---|
| **1. Cơ cấu chi phí** *(Cost Structure)* | **Chi phí cố định (Salary/Man-hour):** Trả theo giờ làm việc hoặc lương tháng của nhân sự chuyên môn. | **Chi phí biến đổi (Pay-per-token):** Trả theo lượng dữ liệu sử dụng thực tế (Input/Output tokens). | **Tiết kiệm >95%** ngân sách vận hành cho các tác vụ lặp lại. |
| **2. Đơn giá định mức** *(Unit Cost)* | Ước tính trung bình: **$5 - $10 / giờ** (tùy thị trường nhân sự). | Ước tính: **~$0.001 - $0.002** cho 1.000 nhiệm vụ (dựa trên giá token Gemini Flash). | Chi phí AI gần như **không đáng kể** so với chi phí nhân sự. |
| **3. Khả năng mở rộng** *(Scalability)* | **Tuyến tính:** Muốn tăng gấp đôi sản lượng nhiệm vụ, cần tăng gấp đôi thời gian hoặc nhân sự. | **Đột biến:** Có thể sinh 10 hay 10.000 nhiệm vụ cùng lúc chỉ bằng cách gọi vòng lặp API. | Phù hợp tuyệt đối với nhu cầu **Live-ops** quy mô lớn. |
| **4. Thời gian đáp ứng** *(Latency)* | **Chậm:** Tốn thời gian suy nghĩ, tính toán cân bằng và nhập liệu thủ công (Data Entry). | **Tức thời (Real-time):** Phản hồi trong mili-giây, cho phép tạo nhiệm vụ cá nhân hóa theo thời gian thực. | Nâng cao trải nghiệm người dùng nhờ nội dung mới liên tục. |
| **5. Rủi ro vận hành** *(Operational Risk)* | Dễ mắc lỗi hành chính (typo), sai sót do mệt mỏi hoặc thiếu tập trung. | Hoạt động ổn định 24/7, độ chính xác được đảm bảo bởi lớp kiểm định (Validation Layer). | Loại bỏ lỗi chủ quan của con người. |