**TRƯỜNG ĐẠI HỌC PHENIKAA**

**KHOA CÔNG NGHỆ THÔNG TIN**

**⸎⸎⸎⸎⸎**

A logo of a university

AI-generated content may be incorrect.

**BÁO CÁO BÀI TẬP LỚN**

**HỌC PHẦN KĨ THUẬT PHẦN MỀM**

**Đề tài : Xây dựng website thương mại bản nhạc**

**cho trò chơi Sky: Children of the Light**

Thành viên : Lê Ngọc Khánh – 23010546

Nguyễn Khắc Long – 23010418

Nguyễn Anh Tài – 23010584

Giáo viên hướng dẫn : Th.S Vũ Quang Dũng

Th.S Nguyễn Xuân Quế

Lớp tín chỉ : N05

**HÀ NỘI, THÁNG 10/2025**

**BẢNG PHÂN CÔNG CÔNG VIỆC**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Mã SV** | **Họ tên** | **Công việc** | **Tỉ lệ đóng góp** |
| 1 | 23010546 | Lê Ngọc Khánh | * Lên kế hoạch, phân công công việc, kiểm tra tiến độ, tổng hợp kết quả. * Thiết kế giao diện nhóm chức năng… * Thiết kế backend nhóm chức năng … * Làm slide thuyết trình … * Viết báo cáo chương …. * Thuyết trình nội dung ….. |  |
| 2 | 23010584 | Nguyễn Anh Tài |  |  |
| 3 | 23010418 | Nguyễn Khắc Long |  |  |

**NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**

..............................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................

# Lời mở đầu

Trong bối cảnh bùng nổ của kỷ nguyên số, âm nhạc đã vượt ra khỏi ranh giới đời thực để trở thành một yếu tố không thể thiếu trong không gian ảo, đặc biệt là các trò chơi điện tử mang đậm tính nghệ thuật như Sky: Children of the Light. Tựa game này không chỉ là một thế giới để khám phá mà còn là một sân khấu để người chơi thể hiện cảm xúc, giao tiếp và kết nối thông qua âm nhạc. Một trong những điểm đặc sắc nhất của game là hệ thống nhạc cụ ảo, cho phép người chơi trình diễn những bản nhạc độc đáo, từ đó hình thành một cộng đồng sáng tạo và chia sẻ bản nhạc vô cùng sôi nổi.

Nhận thấy nhu cầu thực tế từ cộng đồng, nhóm chúng em đã quyết định thực hiện đề tài "Xây dựng website thương mại bản nhạc cho trò chơi Sky: Children of the Light". Mục tiêu của dự án là phát triển một nền tảng chuyên biệt cho việc chia sẻ và mua bán các bản nhạc được chuyển soạn riêng cho hệ thống nhạc cụ trong game. Điểm khác biệt của bản nhạc Sky so với bản nhạc truyền thống nằm ở định dạng đặc thù, được thiết kế để tương thích với giao diện và cơ chế chơi nhạc trong game. Điều này đặt ra những thách thức riêng về cách trình bày, lưu trữ và phân phối.

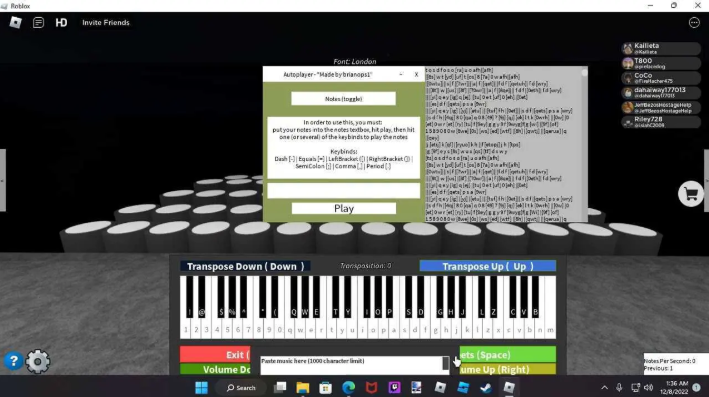
Trong khuôn khổ môn học Phân tích Thiết kế Phần mềm, dự án này là cơ hội để nhóm áp dụng các kiến thức đã học vào một quy trình phát triển hoàn chỉnh, từ khâu phân tích yêu cầu, thiết kế hệ thống, đến triển khai và kiểm thử sản phẩm. Báo cáo sẽ trình bày chi tiết toàn bộ quá trình thực hiện, những thách thức kỹ thuật đã đối mặt, giải pháp của nhóm, và định hướng phát triển trong tương lai. Chúng em kỳ vọng sản phẩm không chỉ giải quyết được bài toán thực tế mà còn góp phần thúc đẩy một cộng đồng sáng tạo bền vững trong thế giới Sky.

**Chương 1: Giới thiệu và Lập kế hoạch**

**1.1. Giới thiệu đề tài**

Trong thế giới game hiện đại, âm nhạc không chỉ còn là yếu tố phụ trợ mà đã trở thành một phần cốt lõi của trải nghiệm, một phương tiện để người chơi tương tác và thể hiện bản thân. Nhiều tựa game đã tích hợp cơ chế chơi nhạc cụ ảo hoặc các yếu tố nhịp điệu, cho phép người chơi biến không gian game thành sân khấu của riêng mình. Có thể kể đến các ví dụ như:

* Roblox: Là một nền tảng khổng lồ với vô số trò chơi do người dùng tạo ra, trong đó có nhiều game chuyên về âm nhạc như các trình giả lập piano ảo hoặc game nhịp điệu như "Funky Friday".



Hình 1.1

* Mini World: Một tựa game sandbox sáng tạo, cung cấp các công cụ để người chơi xây dựng thế giới riêng, bao gồm cả các khối nốt nhạc đơn giản để tạo ra giai điệu.

A screenshot of a video game

AI-generated content may be incorrect.

Hình 1.2

* Genshin Impact: Người chơi có thể sử dụng các nhạc cụ như "Phong Cổ Cầm" để trình diễn các bản nhạc. Cộng đồng game này cũng có những công cụ và website riêng để chia sẻ các bản nhạc được soạn cho game.



Hình 1.3

Mặc dù các tựa game trên đều có những yếu tố âm nhạc và nhịp điệu thú vị, nhóm quyết định chọn **Sky: Children of the Light** làm trọng tâm cho dự án vì những lý do khác biệt và tiềm năng sau:

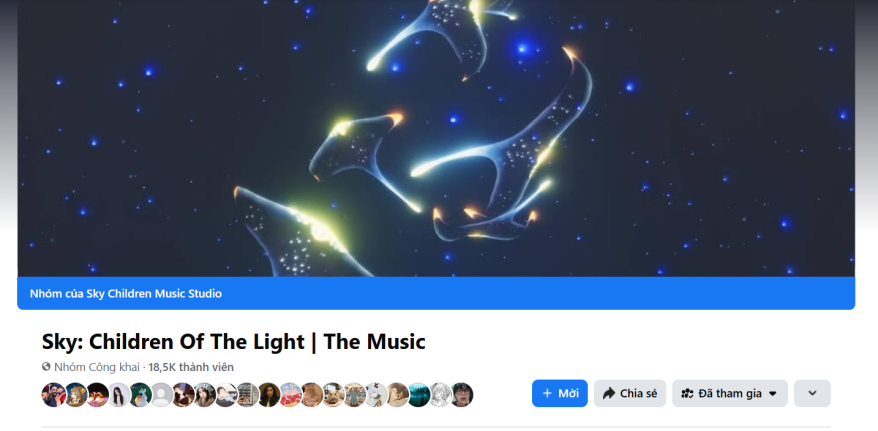
1. Âm nhạc là ngôn ngữ giao tiếp cốt lõi: Sky: Children of the Light không phải là game đầu tiên có chức năng chơi đàn ảo. Nhưng, Sky đã rất thành công trong việc đưa tính năng này trở thành một phương tiện giao tiếp và kết nối cộng đồng một cách trực quan, dễ tiếp cận và đầy cảm xúc. Không giống các game khác nơi âm nhạc là một tính năng phụ, trong Sky, âm nhạc là phương tiện giao tiếp chính. Người chơi dùng âm nhạc để kết bạn, giải đố và thể hiện cảm xúc. Điều này tạo ra một nhu cầu tự nhiên và mạnh mẽ hơn về việc học và chia sẻ các bản nhạc.

A screenshot of a video game

AI-generated content may be incorrect.

Hình 1.4

1. Cộng đồng sáng tạo khổng lồ và gắn kết: Sky sở hữu một trong những cộng đồng người chơi sáng tạo và thân thiện nhất. Họ không chỉ chơi game mà còn liên tục sáng tác, chuyển soạn và chia sẻ hàng ngàn bản nhạc trên các nền tảng mạng xã hội. Tuy nhiên, việc chia sẻ này hiện còn manh mún, thiếu một nền tảng tập trung, có tổ chức để tìm kiếm, lưu trữ và thương mại hóa các tác phẩm.



Hình 1.5

1. Hệ thống nhạc cụ độc đáo và dễ tiếp cận: Hệ thống nhạc cụ của Sky được thiết kế với giao diện đơn giản hóa, trực quan, có nhiều loại nhạc cụ đa dạng và không đòi hỏi kiến thức nhạc lý phức tạp. Điều này giúp một lượng lớn người chơi, kể cả những người không chuyên, có thể dễ dàng tham gia vào việc chơi nhạc và sáng tạo. Chính sự đơn giản này đã tạo ra một "chuẩn" ký hiệu bản nhạc riêng cho game, không giống với bất kỳ hệ thống nào khác.

A screenshot of a cell phone

AI-generated content may be incorrect.

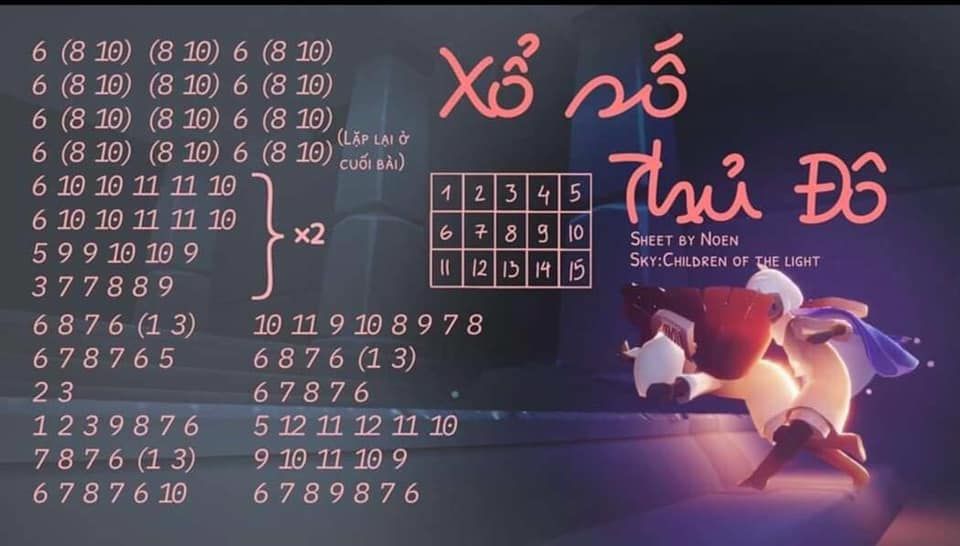
Hình 1.6

Từ việc nhận thấy khoảng trống và nhu cầu thực tế đó, đề tài **"Xây dựng website thương mại bản nhạc cho game Sky: Children of the Light"** được hình thành. Dự án không chỉ nhằm mục đích áp dụng kiến thức kỹ thuật phần mềm mà còn mong muốn giải quyết một bài toán có thật, tạo ra một sản phẩm ý nghĩa và đóng góp cho sự phát triển bền vững của một cộng đồng game đầy sáng tạo.

**1.2. Phân tích bài toán & Giải pháp đề xuất**

Để làm rõ hơn tính cấp thiết của dự án, trước hết cần phải phân biệt rõ sự khác biệt nền tảng giữa bản nhạc thông thường và bản nhạc được sáng tạo riêng cho game. Bản nhạc truyền thống sử dụng hệ thống ký âm nhạc lý chuẩn hóa toàn cầu (nốt, khuông nhạc, khóa son/fa), đòi hỏi người đọc phải có kiến thức nhạc lý nhất định. Ngược lại, bản nhạc trong game như Sky được thiết kế để bất kỳ ai cũng có thể chơi được, sử dụng một hệ thống ký hiệu riêng biệt, trực quan và tương ứng trực tiếp với giao diện của nhạc cụ trong game. Trong bối cảnh đó, có hai định dạng chính để chia sẻ các bản nhạc:

Đầu tiên là **dạng số**. Đây là một định dạng trực quan và dễ tiếp cận, thường được dùng trong các video hướng dẫn hoặc khi người chơi muốn chia sẻ nhanh một đoạn nhạc. Các nốt nhạc được thể hiện bằng một chuỗi số từ 1 đến 15 (hoặc ký tự đơn giản khác), tương ứng với vị trí 15 phím đàn trên giao diện 3x5 trong game.



Hình 1.7

Tuy nhiên, nhược điểm lớn nhất của dạng này là không thể hiện rõ trường độ hay nhịp điệu, khiến người chơi phải tự cảm nhận dựa trên giai điệu gốc. Vì vậy, một định dạng khác đã ra đời và trở thành phương pháp phổ biến và mạnh mẽ nhất trong cộng đồng người chơi sử dụng ứng dụng soạn nhạc, đó là **dạng sheet dữ liệu**.

Đây là một file dữ liệu có cấu trúc (tương tự JSON) không dành cho người đọc trực tiếp, mà để máy tính và các công cụ xử lý. Nó có cấu trúc chứa thông tin chi tiết như tên bản nhạc, tác giả, nhịp độ (bpm). Quan trọng nhất, mỗi nốt nhạc được ghi lại với hai thuộc tính: "key" (phím đàn) và "time" (thời điểm chơi chính xác tính bằng mili giây). Nhờ đó, nó ghi lại hoàn hảo nhịp điệu của cả bản nhạc, khắc phục hoàn toàn nhược điểm của sheet số.

Sự tồn tại song song của hai định dạng này cho thấy một nhu cầu đa dạng trong cộng đồng: một bên cần sự đơn giản, trực quan để tự luyện tập, một bên cần sự chính xác tuyệt đối để chia sẻ các bản nhạc phức tạp hoặc sử dụng cho các công cụ auto-play. Nhận thấy tiềm năng và tính ưu việt của sheet dạng dữ liệu, dự án sẽ tập trung hoàn toàn vào việc xây dựng một nền tảng chuyên biệt cho định dạng này. Nền tảng của quyết định này đến từ việc sheet dữ liệu là một "nguồn" gốc chứa đựng mọi thông tin cần thiết. Từ đó, việc tự động chuyển đổi sang dạng sheet số trực quan để người chơi dễ dàng theo dõi là hoàn toàn khả thi và có thể được tích hợp như một tính năng của website. Cách tiếp cận này đảm bảo tính chính xác và đồng bộ của dữ liệu, đồng thời tạo ra một nền tảng vững chắc, cho phép kết nối những nhà soạn nhạc tài năng với cộng đồng người chơi một cách hiệu quả nhất.

**1.3. Mục tiêu dự án**

Dự án "Xây dựng website thương mại bản nhạc cho game Sky: Children of the Light" được thực hiện nhằm hướng đến các mục tiêu cụ thể về cả sản phẩm và kỹ năng.

**Mục tiêu chính - Xây dựng sản phẩm thực tế**

Mục tiêu cốt lõi của dự án là xây dựng thành công một website hoạt động ổn định, đáp ứng nhu cầu chia sẻ và thương mại hóa bản nhạc trong cộng đồng game Sky. Sản phẩm cuối cùng phải đảm bảo các chức năng cơ bản sau:

* Cho phép người dùng đăng ký, đăng nhập và quản lý tài khoản cá nhân.
* Cung cấp tính năng tải lên (upload) bản nhạc cho người bán và tìm kiếm, xem trước cho người mua.
* Hoàn thiện quy trình mua bán cơ bản, cho phép người dùng thêm sản phẩm vào giỏ hàng và thực hiện giao dịch.

**Mục tiêu phụ: Phát triển kỹ năng chuyên môn**

Bên cạnh việc tạo ra sản phẩm, dự án còn là cơ hội để các thành viên trong nhóm đạt được các mục tiêu học thuật và kỹ năng mềm, bao gồm:

* Áp dụng kiến thức môn học: Vận dụng các lý thuyết và phương pháp của môn Phân tích Thiết kế Phần mềm vào một bài toán thực tế, từ khâu phân tích yêu cầu, vẽ biểu đồ UML, thiết kế kiến trúc cho đến kiểm thử.
* Rèn luyện kỹ năng làm việc nhóm: Nâng cao khả năng phối hợp, trao đổi thông tin, và giải quyết xung đột khi làm việc chung.
* Thực hành quản lý dự án: Tập làm quen với việc lập kế hoạch, phân chia công việc và quản lý tiến độ bằng các công cụ hỗ trợ Git/GitHub.

**1.4. Phạm vi dự án**

Để đảm bảo dự án được hoàn thành đúng tiến độ và tập trung vào các giá trị cốt lõi, nhóm đã xác định rõ ràng các chức năng sẽ được xây dựng và những tính năng sẽ tạm thời lược bỏ trong khuôn khổ bài tập lớn này.

**Các chức năng trong phạm vi phát triển**

Hệ thống sẽ được xây dựng để phục vụ ba nhóm đối tượng chính: Người dùng (User), Người bán (Seller), và Quản trị viên (Admin).

1. Đối với Người dùng (User):
   * Khi chưa đăng nhập (Guest): Có thể tìm kiếm và xem thông tin chi tiết của các bản nhạc, xem các bài viết công khai, thực hiện đăng ký tài khoản mới hoặc đăng nhập.
   * Khi đã đăng nhập (Member): Kế thừa các quyền của Guest, đồng thời có thể quản lý thông tin cá nhân, thực hiện quy trình mua hàng (thêm vào giỏ, thanh toánĐ.
2. Đối với Người bán (Seller):
   * Kế thừa tất cả các quyền của một Người dùng đã đăng nhập.
   * Tải lên các file bản nhạc, điền thông tin mô tả, đặt giá, và quản lý các sản phẩm đã đăng bán trên trang cá nhân của mình.
3. Đối với Admin:
   * Truy cập bảng điều khiển (Dashboard) để xem các thống kê tổng quan về hoạt động của website.
   * Quản lý tài khoản của tất cả người dùng, bao gồm xem danh sách và thực hiện các thao tác như khóa/mở khóa tài khoản.
   * Kiểm duyệt và quản lý toàn bộ nội dung trên website, bao gồm bản nhạc, đơn hàng và bài viết.

**Các chức năng ngoài phạm vi phát triển**

Để tránh việc dự án trở nên quá tải và không thể hoàn thành, nhóm quyết định sẽ không thực hiện các chức năng phức tạp sau trong phiên bản này:

* Hệ thống đánh giá và bình luận sản phẩm: Chức năng cho phép người mua để lại review, xếp hạng sao cho bản nhạc.
* Live chat hỗ trợ: Công cụ chat trực tiếp giữa người mua, người bán hoặc với quản trị viên.
* Tích hợp cổng thanh toán thực tế: Quy trình thanh toán sẽ được giả lập để hoàn thành luồng chức năng, không tích hợp với các dịch vụ thanh toán bên thứ ba như MoMo, ZaloPay,...
* Hệ thống gợi ý sản phẩm: Tự động đề xuất các bản nhạc tương tự dựa trên lịch sử xem hoặc mua hàng của người dùng.
* Chức năng xã hội: Các tính năng như theo dõi (follow) người bán, tạo danh sách yêu thích,...

**1.5. Lựa chọn mô hình quy trình phát triển**

Sau khi xem xét các mô hình phát triển phần mềm phổ biến, nhóm chúng em đã quyết định lựa chọn và áp dụng Mô hình phát triển linh hoạt (Agile), cụ thể là khung làm việc Scrum.

Scrum là một khung làm việc (framework) cho phép giải quyết các bài toán phức tạp nhưng vẫn đảm bảo bàn giao sản phẩm có giá trị cao nhất một cách hiệu quả và sáng tạo. Dự án sẽ được chia thành các giai đoạn lặp lại ngắn, gọi là Sprint (thường kéo dài 1-2 tuần), với mục tiêu là tạo ra một phần của sản phẩm có thể hoạt động được sau mỗi Sprint.

**Lý do lựa chọn mô hình Scrum**

Mô hình này được lựa chọn dựa trên sự phù hợp với đặc thù của dự án và quy mô của nhóm, cụ thể như sau:

1. Phù hợp với quy mô nhóm nhỏ (3 thành viên): Scrum được thiết kế để phát huy hiệu quả tối đa với các nhóm nhỏ (từ 3 đến 9 thành viên). Với 3 thành viên, việc giao tiếp và trao đổi thông tin diễn ra nhanh chóng, giúp các cuộc họp hằng ngày (Daily Scrum) trở nên hiệu quả, đảm bảo mọi người luôn nắm được tiến độ và khó khăn của nhau.
2. Linh hoạt và dễ dàng thích ứng với thay đổi: Đối với một dự án bài tập lớn, các yêu cầu ban đầu có thể chưa hoàn toàn chi tiết hoặc có thể thay đổi sau khi nhận được phản hồi từ giảng viên hướng dẫn. Scrum cho phép nhóm linh hoạt điều chỉnh các ưu tiên và chức năng sau mỗi Sprint, giảm thiểu rủi ro đi sai hướng và đảm bảo sản phẩm cuối cùng đáp ứng tốt nhất mục tiêu đề ra.
3. Phân chia công việc rõ ràng và quản lý tiến độ hiệu quả: Toàn bộ các chức năng của website sẽ được đưa vào một danh sách chung (Product Backlog). Đầu mỗi Sprint, nhóm sẽ cùng nhau chọn ra các công việc quan trọng nhất để thực hiện (Sprint Backlog). Cách làm này giúp 3 thành viên dễ dàng phân chia công việc, tự quản lý nhiệm vụ của mình và theo dõi tiến độ chung của toàn đội một cách trực quan.
4. Khuyến khích sự hợp tác và bàn giao sản phẩm liên tục: Thay vì chờ đến cuối kỳ mới có sản phẩm, Scrum hướng nhóm đến việc tạo ra các phần nhỏ của website sau mỗi Sprint. Điều này không chỉ giúp nhóm nhận được phản hồi sớm để cải thiện mà còn tạo động lực khi thấy được kết quả rõ ràng sau từng giai đoạn ngắn, đồng thời đảm bảo sản phẩm luôn trong trạng thái có thể demo được.

Với những ưu điểm trên, nhóm tin rằng việc áp dụng Scrum sẽ là phương pháp tối ưu để quản lý dự án một cách hiệu quả, đảm bảo sự hợp tác chặt chẽ giữa các thành viên và bàn giao sản phẩm cuối kỳ đạt chất lượng tốt nhất.

**Chương 2: Phân tích yêu cầu**

**2.1. Yêu cầu chức năng**

Dựa trên việc phân tích các đối tượng sẽ tương tác với hệ thống, các yêu cầu chức năng được xác định và phân nhóm theo 3 tác nhân chính: Người dùng (User), Người bán (Seller), và Quản trị viên (Admin).

**2.1.1. Nhóm chức năng dành cho người dùng**

**Chức năng khi chưa đăng nhập**

Đây là các chức năng mà bất kỳ ai truy cập vào website đều có thể sử dụng mà không cần tài khoản.

|  |  |  |
| --- | --- | --- |
| **ID** | **Tên chức năng** | **Mô tả chi tiết** |
| **F01** | Xem Trang chủ | Hiển thị trang chủ với các nội dung chính, sản phẩm nổi bật. |
| **F02** | Xem trang Shop | Hiển thị danh sách các sản phẩm đang bán. Người dùng có thể xem hình ảnh và giá cả. |
| **F03** | Tìm kiếm/Lọc sản phẩm | Cho phép người dùng tìm kiếm sản phẩm theo tên hoặc sử dụng các bộ lọc (ví dụ: theo giá, theo danh mục). |
| **F04** | Xem chi tiết sản phẩm | Khi bấm vào một sản phẩm, người dùng có thể xem trang chi tiết với mô tả đầy đủ, hình ảnh và thông số kỹ thuật. |
| **F05** | Xem trang Cộng đồng | Hiển thị danh sách các bài viết, thảo luận công khai từ cộng đồng người dùng. |
| **F06** | Xem chi tiết bài viết | Cho phép người dùng đọc nội dung của một bài viết cụ thể trong mục Cộng đồng. |
| **F07** | Truy cập trang hỗ trợ | Cho phép người dùng xem các trang thông tin tĩnh như Câu hỏi thường gặp (FAQ), Hướng dẫn sử dụng, Chính sách. |
| **F08** | Đăng ký tài khoản | Cung cấp form để người dùng tạo một tài khoản mới. |
| **F09** | Đăng nhập | Cung cấp form để người dùng đăng nhập vào tài khoản đã có. |
| **F10** | Quên mật khẩu | Cung cấp form để người dùng reset lại mật khẩu mới |

**Bảng 1.1**

**Chức năng bổ sung khi đã đăng nhập**

Sau khi đăng nhập, thành viên sẽ có thêm các quyền hạn và chức năng sau, áp dụng cho cả vai trò người mua và người bán.

|  |  |  |
| --- | --- | --- |
| **ID** | **Tên chức năng** | **Mô tả chi tiết** |
| **F11** | Đăng xuất | Cho phép thành viên đăng xuất khỏi tài khoản hiện tại. |
| **F12** | Quản lý thông tin cá nhân | Cho phép thành viên xem và chỉnh sửa các thông tin cá nhân (ví dụ: ảnh đại diện, mật khẩu). |
| **F13** | Thêm vào giỏ hàng | Cho phép thành viên thêm một hoặc nhiều bản nhạc vào giỏ hàng để chuẩn bị mua. |
| **F14** | Quản lý giỏ hàng | Cung cấp trang xem danh sách các sản phẩm trong giỏ hàng, có thể thay đổi số lượng hoặc xóa sản phẩm. |
| **F15** | Mua bản nhạc | Thực hiện quy trình mua các bản nhạc có trong giỏ hàng. Sau khi mua thành công, thành viên có quyền truy cập vào file bản nhạc. |
| **F16** | Nạp tiền vào tài khoản | Cung cấp giao diện để thành viên có thể nạp tiền vào tài khoản của mình thông qua một quy trình giả lập. |
| **F17** | Xem số dư và lịch sử giao dịch | Cho phép thành viên xem số dư hiện tại và lịch sử các giao dịch (nạp xu, mua bản nhạc) của mình. |

**Bảng 1.2**

**2.1.3. Nhóm chức năng dành cho Người bán**

Đây là các chức năng đặc thù mà chỉ những thành viên có vai trò là Người bán mới có thể thực hiện.

|  |  |  |
| --- | --- | --- |
| **ID** | **Tên chức năng** | **Mô tả chi tiết** |
| **F18** | Tải lên bản nhạc | Cung cấp giao diện để người bán tải lên file bản nhạc, điền thông tin mô tả, đặt giá và đăng bán sản phẩm. |
| **F19** | Quản lý bản nhạc đã đăng | Cung cấp trang quản lý cá nhân để xem danh sách các bản nhạc đã đăng, có thể chỉnh sửa thông tin hoặc gỡ bán. |
| **F20** | Yêu cầu rút tiền | Cho phép người bán tạo yêu cầu rút tiền từ doanh thu đã tích lũy trong tài khoản chờ rút của họ. |

**Bảng 1.3**

**2.1.3. Nhóm chức năng dành cho Quản trị viên (Admin)**

Đây là các chức năng đặc quyền dành cho người quản trị hệ thống để đảm bảo website vận hành ổn định và đúng quy định.

|  |  |  |
| --- | --- | --- |
| **ID** | **Tên chức năng** | **Mô tả chi tiết** |
| **F21** | Xem Bảng điều khiển (Dashboard) | Hiển thị một trang tổng quan với các số liệu thống kê chính, bao gồm: Tổng doanh thu, tổng số đơn hàng, tổng số người dùng, và tổng số bản nhạc. Trang này cũng có biểu đồ doanh thu và danh sách các bản nhạc bán chạy nhất. |
| **F22** | Quản lý sản phẩm | Cho phép Admin truy cập vào mục "Quản lý bản nhạc" để xem, thêm, sửa, hoặc xóa các bản nhạc trên toàn hệ thống. |
| **F23** | Quản lý đơn hàng | Cho phép Admin truy cập vào mục "Đơn hàng" để xem danh sách và chi tiết các giao dịch đã được thực hiện trên website. |
| **F24** | Quản lý người dùng | Cho phép Admin truy cập vào mục "Người dùng" để xem danh sách người dùng, thông tin chi tiết và thực hiện các tác vụ quản lý như khóa/mở khóa tài khoản. |
| **F25** | Xem thống kê | Cho phép Admin truy cập vào mục "Thống Kê" để xem các báo cáo và phân tích chi tiết hơn về hoạt động của website (ví dụ: doanh thu theo thời gian, xu hướng người dùng). |
| **F26** | Cài đặt Hệ thống | Cho phép Admin truy cập vào mục "Cài Đặt" để cấu hình các thiết lập chung cho trang web. |
| **F27** | Quản lý bài viết | Cho phép Admin đăng bài và quản lý tất cả bài viết đã đăng (thông báo từ admin) |
| **F28** | Quản lý yêu cầu rút tiền | Cho phép Admin xem, duyệt, hoặc từ chối các yêu cầu rút tiền từ Người bán. |

**Bảng 1.4**

**2.2. Yêu cầu phi chức năng**

Bên cạnh các yêu cầu về chức năng, hệ thống cũng cần phải đáp ứng các yêu cầu về chất lượng, hiệu suất và trải nghiệm người dùng. Các yêu cầu phi chức năng chính của dự án bao gồm:

**Hiệu năng (Performance)**

* Tốc độ tải trang: Thời gian tải các trang quan trọng (như trang chủ, trang chi tiết sản phẩm) phải dưới 3 giây trong điều kiện mạng thông thường để đảm bảo trải nghiệm người dùng mượt mà và tránh việc người dùng rời bỏ trang.
* Khả năng đáp ứng của máy chủ: Hệ thống phải có khả năng xử lý đồng thời một lượng truy cập nhất định mà không gây ra tình trạng quá tải hay treo hệ thống.

**Bảo mật (Security)**

* Mã hóa mật khẩu: Mật khẩu của tất cả người dùng phải được mã hóa bằng các thuật toán băm (hashing) hiện đại trước khi lưu trữ vào cơ sở dữ liệu để ngăn chặn việc lộ thông tin ngay cả khi cơ sở dữ liệu bị xâm nhập.
* Phân quyền truy cập: Hệ thống phải có cơ chế phân quyền truy cập rõ ràng. Người dùng thường (người mua) không thể truy cập các chức năng của người bán (ví dụ: tải lên bản nhạc). Chỉ có Quản trị viên mới có quyền truy cập vào các chức năng quản trị hệ thống.

**Khả năng sử dụng (Usability)**

* Giao diện thân thiện: Giao diện người dùng (UI) cần được thiết kế sạch sẽ, trực quan và nhất quán trên toàn bộ trang web để người dùng có thể dễ dàng tìm thấy thông tin và thực hiện các thao tác mong muốn mà không cần hướng dẫn phức tạp.
* Thiết kế đáp ứng (Responsive Design): Website phải tương thích và hiển thị tốt trên nhiều loại thiết bị khác nhau, từ máy tính để bàn, máy tính bảng cho đến điện thoại di động, đảm bảo trải nghiệm người dùng không bị gián đoạn.

**Khả năng bảo trì (Maintainability)**

* Mã nguồn có cấu trúc: Mã nguồn của dự án cần được tổ chức theo các module rõ ràng, tuân thủ các quy chuẩn lập trình và có các chú thích cần thiết để các thành viên trong nhóm và các nhà phát triển trong tương lai có thể dễ dàng đọc hiểu, sửa lỗi và phát triển thêm các tính năng mới.

**2.3. Mô hình hóa yêu cầu**

**2.3.1. Biểu đồ use case**

Để thể hiện một cách tổng quan các tương tác chính giữa người dùng và hệ thống, nhóm đã xây dựng biểu đồ use case. Biểu đồ này được thiết kế dựa trên các yêu cầu chức năng đã được xác định trong mục 2.1.

**Xác định các use case chính**

Từ danh sách các yêu cầu chức năng (F01-F28), các use case chính được gom nhóm lại thành 6 luồng nghiệp vụ chính như sau:

* UC1: Xác thực người dùng
* UC2: Tìm kiếm & Xem thông tin
* UC3: Quản lý Profile cá nhân
* UC4: Quản lý Mua hàng
* UC5: Quản lý Bán hàng
* UC6: Quản trị Hệ thống

|  |  |  |
| --- | --- | --- |
| **Use Case (UC)** | **Mô tả** | **Yêu cầu chức năng (F)** |
| **UC1: Đăng ký/ đăng nhập** | Bao gồm các hoạt động đăng ký, đăng nhập, và quên mật khẩu để xác thực và truy cập hệ thống. | F08: Đăng ký tài khoản  F09: Đăng nhập  F10: Quên mật khẩu |
| **UC2: Tìm kiếm & Xem thông tin** | Bao gồm tất cả các hoạt động xem và tìm kiếm thông tin công khai trên website. | F01: Xem Trang chủ  F02: Xem Shop  F03: Tìm kiếm/Lọc sản phẩm  F04: Xem chi tiết sản phẩm  F05: Xem trang Cộng đồng  F06: Xem chi tiết bài viết  F07: Truy cập trang hỗ trợ |
| **UC3: Quản lý tài khoản cá nhân** | Cho phép thành viên quản lý thông tin, phiên đăng nhập, và các hoạt động tài chính liên quan đến tài khoản. | F11: Đăng xuất  F12: Quản lý thông tin cá nhân  F16: Nạp tiền vào tài khoản  F17: Xem số dư và lịch sử giao dịch |
| **UC4: Mua hàng** | Bao gồm toàn bộ luồng nghiệp vụ mua một bản nhạc của thành viên. | F13: Thêm vào giỏ hàng  F14: Quản lý giỏ hàng  F15: Mua bản nhạc |
| **UC5: Bán hàng** | Các chức năng dành riêng cho Người bán để quản lý sản phẩm và doanh thu của họ. | F18: Tải lên bản nhạc  F19: Quản lý bản nhạc đã đăng  F20: Yêu cầu rút tiền |
| **UC6: Quản trị Hệ thống** | Các chức năng quản trị cấp cao của Admin để vận hành và kiểm soát website. | F21: Xem Bảng điều khiển  F22: Quản lý sản phẩm  F23: Quản lý đơn hàng  F24: Quản lý người dùng  F25: Xem thống kê  F26: Cài đặt Hệ thống  F27: Quản lý bài viết  F28: Quản lý yêu cầu rút tiền |

**Bảng 1.5**

**Biểu đồ Use Case (Use Case Diagram):**

**A diagram of a website

AI-generated content may be incorrect.**

**Hình 2.1**

* + 1. **Đặc tả Use Case**

Dưới đây là đặc tả chi tiết cho 3 luồng nghiệp vụ (Use Case) quan trọng nhất của hệ thống, bao gồm: Xác thực người dùng, Quản lý Mua hàng, và Quản lý Bán hàng.

**UC1: Xác thực người dùng**

Mô tả: Bao gồm các hoạt động đăng ký tài khoản mới, đăng nhập vào hệ thống, và khôi phục mật khẩu đã quên để xác thực và cấp quyền truy cập cho người dùng.

**Luồng sự kiện:**

* + **Luồng cơ bản 1 – Đăng nhập:**
    1. Use Case bắt đầu khi người dùng muốn đăng nhập vào hệ thống.
    2. Người dùng truy cập giao diện trang "Đăng nhập".
    3. Hệ thống hiển thị form đăng nhập, yêu cầu người dùng nhập thông tin (Tên đăng nhập/Email, Mật khẩu).
    4. Người dùng nhập đầy đủ và chính xác thông tin, sau đó nhấn nút "Đăng nhập".
    5. Hệ thống kiểm tra thông tin đăng nhập trong cơ sở dữ liệu.
    6. Hệ thống xác thực thông tin hợp lệ, tạo phiên làm việc (session) và chuyển hướng người dùng vào hệ thống với vai trò tương ứng (trang chủ cho Member/Seller, trang Dashboard cho Admin).
    7. Use Case kết thúc thành công.
  + **Luồng cơ bản 2 – Đăng ký:**
    1. Use Case bắt đầu khi người dùng muốn tạo tài khoản mới.
    2. Người dùng chọn chức năng "Đăng ký".
    3. Hệ thống hiển thị form đăng ký.
    4. Người dùng điền các thông tin theo yêu cầu và gửi form.
    5. Hệ thống kiểm tra tính hợp lệ của thông tin (ví dụ: email chưa tồn tại).
    6. Hệ thống tạo tài khoản mới, lưu vào cơ sở dữ liệu và hiển thị thông báo thành công.
    7. Use Case kết thúc thành công, người dùng có thể tiến hành đăng nhập.
  + **Luồng phụ:**
    1. **S-1: Quên mật khẩu**
       1. Tại bước 4 của Luồng cơ bản 1, người dùng chọn chức năng "Quên mật khẩu" trên giao diện đăng nhập.
       2. Hệ thống chuyển hướng sang trang "Khôi phục mật khẩu", yêu cầu người dùng nhập địa chỉ email đã đăng ký.
       3. Người dùng nhập email và nhấn nút "Gửi yêu cầu".
       4. Hệ thống xác thực email có tồn tại trong cơ sở dữ liệu.
       5. Hệ thống gửi một email chứa đường link đặt lại mật khẩu đến cho người dùng.
       6. Hệ thống hiển thị thông báo "Yêu cầu đã được gửi, vui lòng kiểm tra email của bạn". Use Case con này kết thúc.
  + **Luồng thay thế:**
    1. **A-1: Thông tin đăng nhập không hợp lệ**
       1. Tại bước 5 của Luồng cơ bản 1, nếu hệ thống xác thực thông tin không hợp lệ (sai tên đăng nhập hoặc mật khẩu).
       2. Hệ thống sẽ hiển thị thông báo lỗi: "Tên đăng nhập hoặc mật khẩu không chính xác."
       3. Người dùng ở lại trang Đăng nhập và luồng sự kiện quay lại bước 4.

**Tiền điều kiện:**

* + Đối với chức năng "Đăng nhập" và "Quên mật khẩu", người dùng phải có một tài khoản đã được đăng ký trong hệ thống.
  + Hệ thống đang hoạt động và sẵn sàng xử lý yêu cầu.

**Hậu điều kiện:**

* + **Thành công:** Người dùng đăng nhập thành công và được chuyển hướng đến giao diện phù hợp với vai trò của mình. Một phiên làm việc (session) được tạo ra. Hoặc, người dùng đăng ký thành công và có một tài khoản mới trong hệ thống.
  + **Thất bại:** Người dùng không thể truy cập hệ thống và nhận được thông báo lỗi tương ứng. Trạng thái hệ thống không thay đổi.



Hình 2.2

A diagram with text and images

AI-generated content may be incorrect.

A diagram with text and images

AI-generated content may be incorrect.

**UC4: Mua hàng**

Mô tả: Bao gồm toàn bộ luồng nghiệp vụ mua một bản nhạc của thành viên, từ việc thêm sản phẩm vào giỏ hàng, quản lý giỏ hàng cho đến khi thực hiện thanh toán thành công.

**Luồng sự kiện:**

* + **Luồng cơ bản:**
    1. Use Case bắt đầu khi người dùng thêm một bản nhạc vào giỏ hàng và truy cập vào trang "Giỏ hàng".
    2. Hệ thống hiển thị danh sách các bản nhạc, tổng số "Xu" phải thanh toán và số dư "Xu" hiện tại của Member.
    3. Member nhấn nút "Thanh toán".
    4. Hệ thống kiểm tra và xác nhận số dư đủ để thực hiện giao dịch.
    5. Hệ thống trừ số "Xu" tương ứng từ tài khoản của người mua, đồng thời cộng doanh thu (sau khi trừ phí) vào tài khoản của Seller.
    6. Hệ thống tạo một bản ghi Đơn hàng mới và cấp quyền truy cập file bản nhạc cho Member.
    7. Hệ thống hiển thị thông báo "Thanh toán thành công!" và dọn sạch giỏ hàng.
    8. Use Case kết thúc thành công.
  + **Luồng phụ S1 - Nạp thêm Xu**
    - 1. Tại bước 2 của Luồng cơ bản, nếu Member thấy số dư không đủ, họ có thể nhấn vào nút "Nạp Xu".
      2. Hệ thống chuyển hướng đến trang Nạp Xu.
      3. Sau khi Member nạp Xu thành công, Hệ thống đưa họ trở lại trang Giỏ hàng.
  + **Luồng thay thế:**
    1. A-1: Số dư không đủ khi thanh toán
       1. Tại bước 4 của Luồng cơ bản, nếu số dư của Member không đủ (ví dụ: do có một giao dịch khác diễn ra song song), Hệ thống sẽ hiển thị thông báo lỗi "Số dư không đủ, vui lòng nạp thêm!".
       2. Luồng sự kiện quay lại bước 2.
    2. A-2: Lỗi xử lý giao dịch
       1. Tại bước 6 của Luồng cơ bản, nếu có lỗi xảy ra (ví dụ: lỗi cơ sở dữ liệu), Hệ thống sẽ hủy bỏ mọi thay đổi (rollback).
       2. Hệ thống hiển thị thông báo "Đã có lỗi xảy ra, vui lòng thử lại sau." và Use Case kết thúc thất bại.

**Tiền điều kiện:**

* + Member phải đăng nhập vào hệ thống.
  + Giỏ hàng của Member phải có ít nhất một sản phẩm.

**Hậu điều kiện:**

* + **Thành công:** Một Đơn hàng mới được tạo. Số dư của Member và Seller được cập nhật. Member có quyền truy cập sản phẩm đã mua.
  + **Thất bại:** Trạng thái hệ thống (số dư, đơn hàng) không thay đổi. Người dùng nhận được thông báo lỗi tương ứng.

A screenshot of a computer

AI-generated content may be incorrect.

**Chương 3: Thiết kế Hệ thống**

Chương này trình bày chi tiết về quá trình thiết kế hệ thống cho website thương mại bản nhạc. Quá trình thiết kế được chia thành hai phần chính: thiết kế kiến trúc ở mức độ tổng quan để xác định cấu trúc hệ thống, và thiết kế chi tiết sử dụng các biểu đồ UML để mô hình hóa các thành phần và tương tác bên trong.

**3.1. Thiết kế kiến trúc**

Thiết kế kiến trúc là bước nền tảng, xác định cấu trúc cấp cao của hệ thống, cách các thành phần chính được tổ chức và tương tác với nhau. Lựa chọn một kiến trúc phù hợp là yếu tố then chốt, ảnh hưởng trực tiếp đến các yêu cầu phi chức năng như khả năng bảo trì, mở rộng, hiệu năng và bảo mật của sản phẩm.

**3.1.1. Lựa chọn Mô hình Kiến trúc**

Dựa trên quy mô và các yêu cầu đã xác định của dự án, các mô hình kiến trúc phổ biến như Monolithic (kiến trúc nguyên khối), Microservices, và Three-Tier (kiến trúc 3 tầng) đã được xem xét để đưa ra lựa chọn tối ưu.

* **Kiến trúc Monolithic:** Toàn bộ ứng dụng được xây dựng như một khối duy nhất. Mô hình này có ưu điểm là đơn giản trong việc phát triển và triển khai ở giai đoạn đầu. Tuy nhiên, khi hệ thống phát triển phức tạp hơn, việc bảo trì, sửa lỗi và cập nhật một thành phần nhỏ có thể ảnh hưởng đến toàn bộ hệ thống, đi ngược lại yêu cầu về "Khả năng bảo trì" (Maintainability) đã đề ra.
* **Kiến trúc Microservices:** Hệ thống được chia thành nhiều dịch vụ nhỏ, độc lập. Mô hình này mang lại sự linh hoạt và khả năng mở rộng vượt trội. Tuy nhiên, việc triển khai và quản lý một hệ thống Microservices đòi hỏi sự phức tạp cao về mặt hạ tầng và vận hành, không phù hợp với một dự án có quy mô vừa và nhỏ với đội ngũ phát triển gồm 3 thành viên.
* **Kiến trúc Three-Tier (3-Lớp):** Mô hình này phân tách ứng dụng thành ba tầng logic riêng biệt: Trình bày, Nghiệp vụ và Dữ liệu. Nó cung cấp một sự cân bằng lý tưởng giữa tính đơn giản và khả năng tổ chức, phù hợp với các ứng dụng web truyền thống và đáp ứng tốt các yêu cầu của dự án.

Sau khi phân tích, mô hình kiến trúc **Three-Tier (3-Lớp)** được lựa chọn làm kiến trúc mục tiêu cho hệ thống.

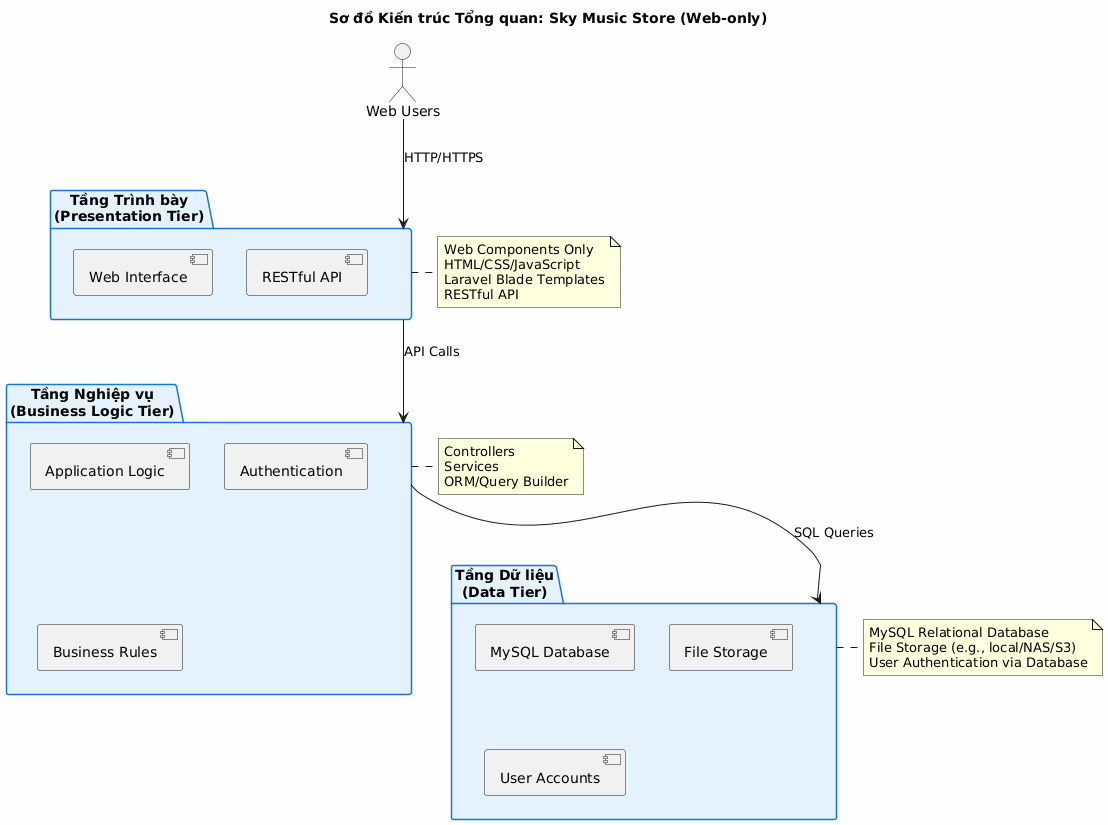
**Lý do lựa chọn:**

* **Phân tách rõ ràng (Clear Separation):** Kiến trúc 3-Lớp phân chia hệ thống thành ba tầng logic độc lập: Tầng Trình bày (Presentation), Tầng Nghiệp vụ (Business Logic), và Tầng Dữ liệu (Data). Sự phân tách này hoàn toàn phù hợp với yêu cầu về "Mã nguồn có cấu trúc" và "Khả năng bảo trì". Nó cho phép các thành viên trong nhóm có thể phát triển song song trên các tầng khác nhau (ví dụ: một thành viên tập trung vào giao diện người dùng ở Tầng Trình bày, trong khi các thành viên khác xây dựng logic ở Tầng Nghiệp vụ) mà ít gây ảnh hưởng chéo, từ đó tối ưu hóa hiệu suất làm việc nhóm.
* **Tăng cường bảo mật:** Việc cô lập Tầng Dữ liệu khỏi sự truy cập trực tiếp từ client (Tầng Trình bày) là một nguyên tắc bảo mật cơ bản. Mọi yêu cầu từ người dùng phải đi qua Tầng Nghiệp vụ để được xác thực và kiểm tra quyền hạn trước khi có thể tương tác với cơ sở dữ liệu. Điều này trực tiếp đáp ứng yêu cầu "Phân quyền truy cập" đã được xác định, ngăn chặn các truy cập trái phép vào dữ liệu nhạy cảm.
* **Phù hợp với công nghệ hiện đại:** Kiến trúc này tương thích hoàn hảo với các framework phát triển web hiện đại. Tầng Trình bày có thể được xây dựng bằng ReactJS, trong khi Tầng Nghiệp vụ có thể sử dụng các framework mạnh mẽ như Node.js hoặc Laravel, vốn là những công nghệ được đề xuất trong kế hoạch phát triển của dự án.
* **Khả năng mở rộng hợp lý:** Mặc dù không linh hoạt bằng Microservices, kiến trúc 3-Lớp vẫn cho phép mở rộng từng tầng một cách độc lập khi cần thiết. Ví dụ, khi lưu lượng truy cập tăng, hệ thống có thể bổ sung thêm máy chủ web ở Tầng Trình bày hoặc nhân bản máy chủ ứng dụng ở Tầng Nghiệp vụ mà không ảnh hưởng đến các tầng khác.

**3.1.2. Sơ đồ Kiến trúc Tổng quan và Mô tả các Tầng**

Sơ đồ dưới đây minh họa kiến trúc 3-Lớp của hệ thống và luồng tương tác giữa các thành phần, từ người dùng cuối đến cơ sở dữ liệu.

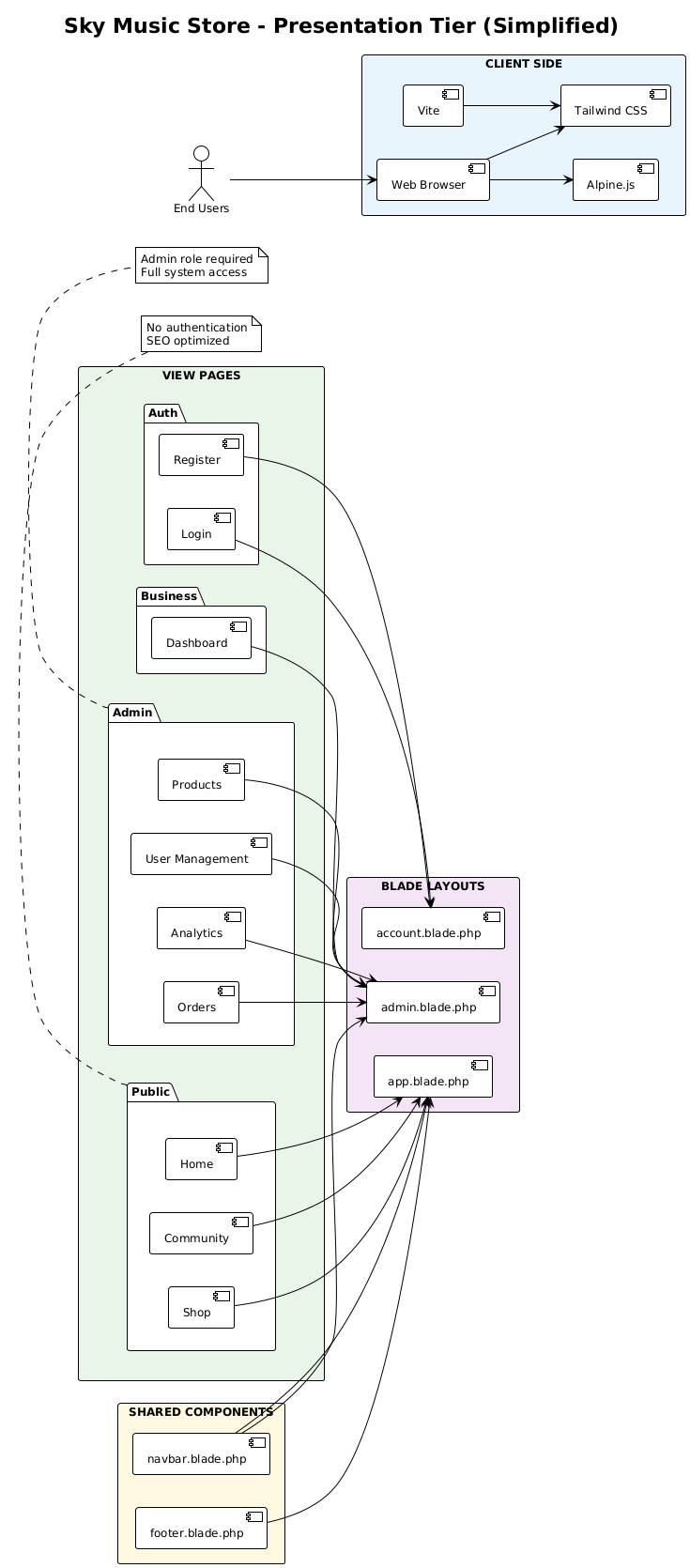
*Sơ đồ kiến trúc tổng quan 3 tầng*

**

**Mô tả chi tiết các tầng:**

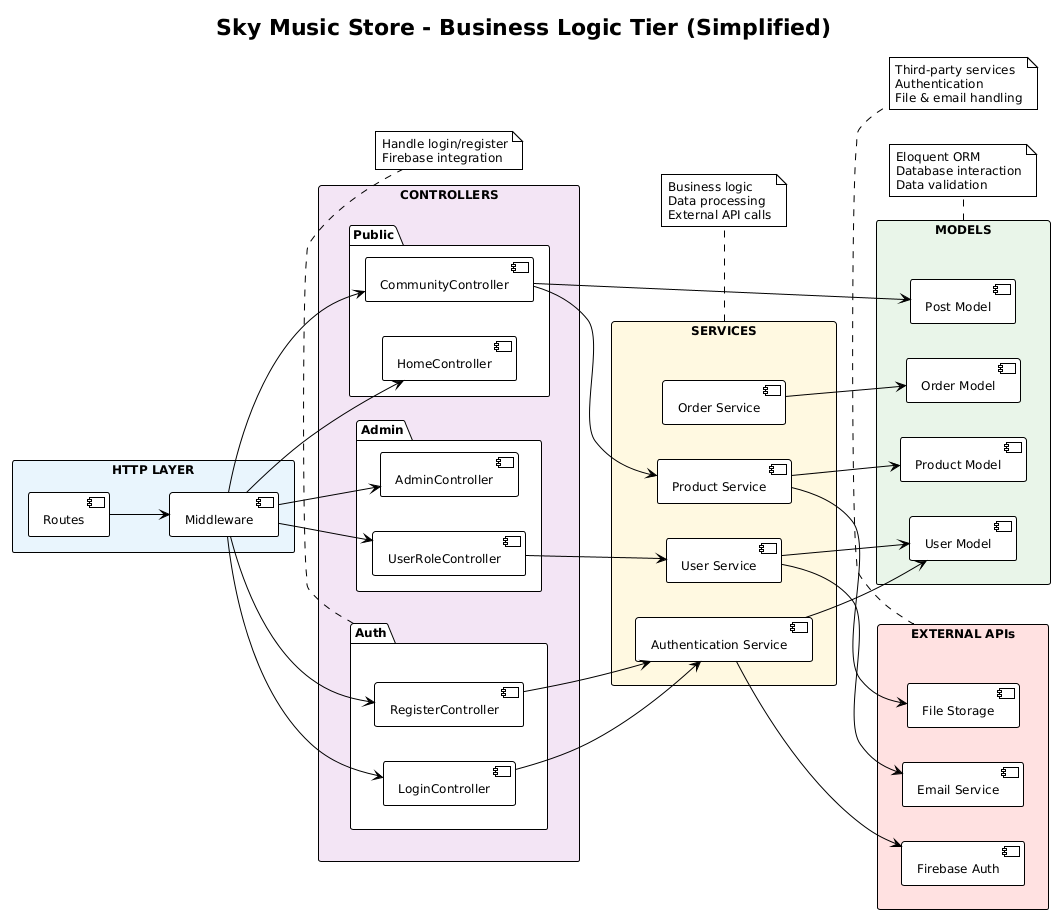
* **Tầng Trình bày (Presentation Tier)**
  + **Trách nhiệm:** Đây là tầng giao diện người dùng (UI), chịu trách nhiệm chính trong việc hiển thị dữ liệu cho người dùng và thu thập các thông tin đầu vào từ họ. Tầng này được thiết kế để không chứa bất kỳ logic nghiệp vụ phức tạp nào, đảm bảo sự tách biệt rõ ràng.
  + **Thành phần:** Bao gồm các thành phần giao diện được xây dựng bằng HTML, CSS, và JavaScript, sử dụng một framework hiện đại như ReactJS để tạo ra trải nghiệm người dùng tương tác và linh hoạt. Tầng này sẽ render các trang chính của website như Trang chủ (F01), Trang Shop (F02), Trang chi tiết sản phẩm (F04), Giỏ hàng (F13), cũng như các biểu mẫu chức năng như Đăng ký (F08) và Đăng nhập (F09).
  + **Tương tác:** Tầng Trình bày giao tiếp với Tầng Nghiệp vụ thông qua các lời gọi API (Application Programming Interface), điển hình là RESTful API qua giao thức HTTP/HTTPS. Ví dụ, khi người dùng thực hiện tìm kiếm sản phẩm, Tầng Trình bày sẽ gửi một yêu cầu HTTP GET đến một endpoint API cụ thể trên Tầng Nghiệp vụ để lấy về danh sách sản phẩm phù hợp.

Sơ đồ kiến trúc tầng Trình bày



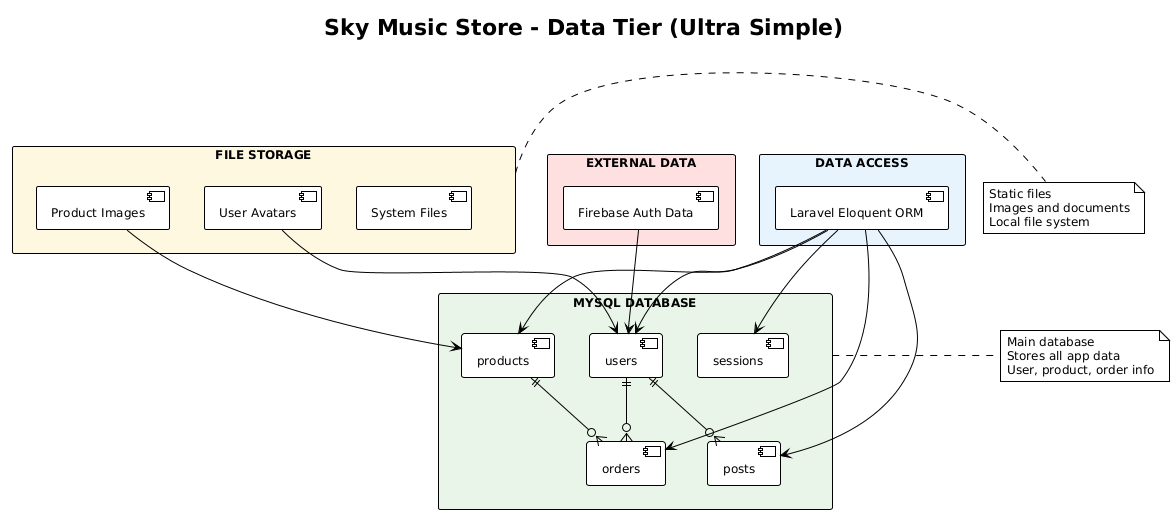
* **Tầng Nghiệp vụ (Business Logic Tier / Application Tier)**
  + **Trách nhiệm:** Đây là "bộ não" của hệ thống, nơi xử lý tất cả các quy tắc nghiệp vụ, logic xử lý dữ liệu, và điều phối luồng công việc của ứng dụng. Mọi yêu cầu từ Tầng Trình bày đều được xử lý tại đây trước khi tương tác với Tầng Dữ liệu.
  + **Thành phần:** Bao gồm các máy chủ ứng dụng (Application Servers) chạy mã nguồn backend (ví dụ: Node.js, PHP/Laravel). Tầng này được tổ chức thành các module logic để xử lý từng nhóm chức năng cụ thể:
    - *Module Quản lý Xác thực và Phân quyền:* Xử lý các nghiệp vụ đăng ký (UC1), đăng nhập (UC2), và thực thi cơ chế phân quyền, đảm bảo người dùng chỉ có thể truy cập các chức năng phù hợp với vai trò của mình (Member, Seller, Admin).
    - *Module Xử lý Nghiệp vụ Thương mại điện tử:* Quản lý giỏ hàng, tạo và xử lý đơn hàng (UC5), và thực hiện các giao dịch thanh toán bằng đơn vị tiền tệ ảo "Xu".
    - *Module Quản lý Bản nhạc:* Xử lý việc tải lên (UC6), kiểm duyệt nội dung, và cung cấp dữ liệu sản phẩm cho Tầng Trình bày.
    - *Dịch vụ Giao dịch "Xu":* Do hệ thống sử dụng đơn vị tiền tệ ảo "Xu" cho mọi giao dịch , một module dịch vụ giao dịch chuyên biệt sẽ được thiết kế tại tầng này. Module này không chỉ đơn thuần cập nhật số dư của người dùng, mà còn chịu trách nhiệm ghi lại mọi giao dịch (nạp "Xu", mua hàng, nhận doanh thu) vào một sổ cái giao dịch. Cách tiếp cận này đảm bảo tính toàn vẹn, minh bạch và khả năng truy vết của dữ liệu tài chính trong hệ thống.
  + **Tương tác:** Nhận yêu cầu từ Tầng Trình bày, thực thi logic nghiệp vụ, sau đó giao tiếp với Tầng Dữ liệu để thực hiện các thao tác đọc/ghi thông tin cần thiết.

Sơ đồ kiến trúc tầng Nghiệp vụ:



* **Tầng Dữ liệu (Data Tier)**
  + **Trách nhiệm:** Chịu trách nhiệm lưu trữ và quản lý toàn bộ dữ liệu của ứng dụng một cách bền vững và an toàn (persistent storage).
  + **Thành phần:** Bao gồm hệ quản trị cơ sở dữ liệu quan hệ (ví dụ: MySQL, PostgreSQL) để lưu trữ các dữ liệu có cấu trúc như thông tin người dùng, sản phẩm, đơn hàng. Ngoài ra, tầng này còn có thể bao gồm các giải pháp lưu trữ file (File Storage) như một thư mục trên máy chủ hoặc các dịch vụ lưu trữ đám mây (ví dụ: Amazon S3) để lưu trữ các file bản nhạc do người dùng tải lên.
  + **Tương tác:** Tầng Dữ liệu được thiết kế để chỉ giao tiếp trực tiếp với Tầng Nghiệp vụ. Việc ngăn chặn mọi kết nối trực tiếp từ Tầng Trình bày đến Tầng Dữ liệu là một biện pháp bảo mật quan trọng, đảm bảo dữ liệu luôn được xử lý thông qua các lớp logic đã được kiểm duyệt.

Sơ đồ kiến trúc tầng Dữ liệu:



**3.2. Thiết kế chi tiết (UML)**

Sau khi xác định kiến trúc tổng thể ở mức vĩ mô, bước tiếp theo là đi sâu vào thiết kế chi tiết các thành phần bên trong hệ thống bằng Ngôn ngữ Mô hình hóa Thống nhất (UML - Unified Modeling Language). Quá trình này chuyển đổi từ góc nhìn kiến trúc cấp cao sang một bản thiết kế chi tiết, hướng đối tượng cho hệ thống. Việc sử dụng các biểu đồ UML, theo các thông lệ tiêu chuẩn trong ngành kỹ thuật phần mềm , giúp trực quan hóa cấu trúc tĩnh và hành vi động của hệ thống. Đây là tài liệu định hướng cuối cùng cho giai đoạn triển khai, đảm bảo mã nguồn được phát triển tuân thủ chặt chẽ các yêu cầu đã phân tích và kiến trúc đã lựa chọn.

**3.2.1. Biểu đồ Lớp (Class Diagram)**

Biểu đồ Lớp mô tả cấu trúc tĩnh của hệ thống, bao gồm các lớp đối tượng chính, các thuộc tính (attributes), phương thức (methods) của chúng, và các mối quan hệ (kế thừa, kết hợp, tập hợp) giữa các lớp. Đây là bản thiết kế chi tiết cho cấu trúc cơ sở dữ liệu và các lớp mã nguồn của ứng dụng, được suy ra trực tiếp từ các yêu cầu chức năng (F01-F28) và các ca sử dụng (UC1-UC6) đã được xác định trong Chương 2.

*A diagram of a computer

AI-generated content may be incorrect.*

**Mô tả các Lớp chính:**

**Lớp User (Người dùng)**

Lớp User đại diện cho thực thể cơ sở của bất kỳ cá nhân nào tương tác với hệ thống sau khi đã xác thực. Nó đóng gói các thuộc tính và hành vi chung cho tất cả các vai trò. Hệ thống định nghĩa ba vai trò riêng biệt: Người dùng (User/Member), Người bán (Seller), và Quản trị viên (Admin). Thay vì một thiết kế tạo ra ba lớp riêng biệt không liên quan, một cách tiếp cận hướng đối tượng mạnh mẽ và dễ bảo trì hơn là sử dụng tính kế thừa. Theo đó,  Seller và Admin sẽ là các lớp con chuyên biệt hóa của lớp User.

Seller *là một* User, một Admin *là một* User) là định nghĩa kinh điển của kế thừa trong lập trình hướng đối tượng. Do đó, việc tạo một lớp cơ sở User để chứa dữ liệu chung (tên đăng nhập, email, mật khẩu) và các phương thức chung (login, logout, updateProfile) giúp tránh trùng lặp mã và tạo ra một hệ thống phân cấp logic, rõ ràng. Lựa chọn thiết kế này trực tiếp hỗ trợ yêu cầu phi chức năng về "Khả năng bảo trì" (Maintainability).

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính (Attribute)** | **Kiểu dữ liệu (Type)** | **Mô tả (Description)** | **Yêu cầu liên quan (Related Requirement)** |
| **userId** | int | Khóa chính, định danh duy nhất cho mỗi người dùng. | - |
| **username** | String | Tên đăng nhập duy nhất của người dùng. | F09 |
| **passwordHash** | String | Mật khẩu đã được băm (hashed) để lưu trữ an toàn. | F09, Bảo mật |
| **email** | String | Địa chỉ email duy nhất, dùng để đăng nhập và liên lạc. | F08, F10 |
| **avatarUrl** | String | Đường dẫn đến ảnh đại diện của người dùng. | F12 |
| **balance** | Decimal | Số dư "Xu" hiện tại trong tài khoản. | F16, F17 |
| **role** | Enum | Vai trò của người dùng (MEMBER, SELLER, ADMIN). | Phân quyền truy cập |
| **createdAt** | Timestamp | Thời điểm tài khoản được tạo. | - |
| **updatedAt** | Timestamp | Thời điểm thông tin tài khoản được cập nhật lần cuối. | F12 |

**Bảng: Thuộc tính lớp User**

|  |  |
| --- | --- |
| **Phương thức (Method)** | **Mô tả (Description)** |
| register(username, password, email) | Xử lý việc tạo tài khoản mới (F08). |
| login(username, password) | Xác thực thông tin đăng nhập và tạo phiên làm việc (F09). |
| logout() | Hủy phiên làm việc hiện tại (F11). |
| updateProfile(details) | Cập nhật thông tin cá nhân (F12). |
| changePassword(oldPassword, newPassword) | Thay đổi mật khẩu tài khoản. |
| deposit(amount) | Nạp "Xu" vào tài khoản (F16). |

**Lớp Seller (Người bán)**

Đây là một loại User chuyên biệt với khả năng bổ sung là tải lên, quản lý và bán bản nhạc. Lớp này kế thừa tất cả các thuộc tính và phương thức từ lớp User.

* **Thuộc tính:** (Kế thừa từ User) + payoutInfo (String), totalRevenue (Decimal).
* **Phương thức:** (Kế thừa từ User) + uploadSheetMusic(file, details), manageSheetMusic(), requestWithdrawal(amount).

**Lớp Admin (Quản trị viên)**

Đây là một User chuyên biệt với cấp độ đặc quyền cao nhất, chịu trách nhiệm giám sát hệ thống, kiểm duyệt nội dung và quản lý người dùng.

* **Thuộc tính:** (Kế thừa từ User).
* **Phương thức:** (Kế thừa từ User) + manageUser(userId, action), approveSheetMusic(sheetId), rejectSheetMusic(sheetId), viewDashboard(), processWithdrawal(requestId).

**Lớp SheetMusic (Bản nhạc)**

Lớp này đại diện cho một sản phẩm duy nhất (một tệp bản nhạc) có sẵn để bán trên nền tảng. Vòng đời của một đối tượng SheetMusic phức tạp hơn một mục sản phẩm thông thường vì nó liên quan đến một quy trình kiểm duyệt. Yêu cầu F18 cho phép Seller tải lên một bản nhạc, trong khi yêu cầu F27 quy định rằng Admin phải "Kiểm duyệt và quản lý toàn bộ nội dung". Điều này ngụ ý có một khoảng thời gian và một quy trình giữa lúc tải lên và lúc sản phẩm được đăng bán. Một thuộc tính

status với các trạng thái như PENDING\_APPROVAL, APPROVED, REJECTED là cần thiết để mô hình hóa quy trình nghiệp vụ này một cách rõ ràng trong thiết kế.

* **Thuộc tính:** sheetId (int), title (String), description (String), price (Decimal), author (String), gameVersion (String), filePath (String - đường dẫn đến "Sheet Dữ liệu"), previewContent (String - "Sheet Số" được tạo ra), status (Enum), sellerId (int - Khóa ngoại đến User).
* **Phương thức:** generatePreview(), updateDetails(details).

**Lớp Order (Đơn hàng) và OrderItem (Chi tiết Đơn hàng)**

Lớp Order đại diện cho một giao dịch đã hoàn tất, nơi một người dùng đã mua một hoặc nhiều bản nhạc. Một Order được cấu thành từ một hoặc nhiều đối tượng OrderItem. Đây là một mối quan hệ Hợp thành (Composition) vì một OrderItem không thể tồn tại nếu không có Order cha của nó.

* **Thuộc tính Order:** orderId (int), buyerId (int - FK đến User), orderDate (Timestamp), totalAmount (Decimal).
* **Thuộc tính OrderItem:** orderItemId (int), orderId (int - FK đến Order), sheetId (int - FK đến SheetMusic), priceAtPurchase (Decimal).

**Lớp Transaction (Giao dịch)**

Lớp này được thiết kế để ghi lại mọi sự dịch chuyển của đơn vị tiền tệ ảo "Xu" trong hệ thống, đảm bảo khả năng kiểm toán và tính toàn vẹn dữ liệu. Đơn vị tiền "Xu" là huyết mạch của nền kinh tế trên nền tảng. Việc chỉ đơn thuần tăng/giảm một cột balance trong bảng User rất dễ gặp phải các điều kiện tranh chấp (race conditions) và khiến việc kiểm toán trở nên bất khả thi.

Hệ thống có nhiều hành động tài chính: nạp tiền (F16), mua hàng (F15), người bán nhận doanh thu, và rút tiền (F20). Một lỗi xảy ra trong quá trình mua hàng (ví dụ: máy chủ gặp sự cố sau khi đã trừ tiền của người mua nhưng trước khi cộng tiền cho người bán) sẽ dẫn đến hỏng dữ liệu. Giải pháp chuyên nghiệp là sử dụng một sổ cái giao dịch bất biến. Mỗi sự kiện tài chính sẽ tạo ra một hoặc nhiều bản ghi

Transaction. Số dư của người dùng không còn là một thuộc tính được lưu trữ trực tiếp mà là một giá trị được *tính toán* (tổng của tất cả các giao dịch của họ). Điều này đảm bảo số dư luôn nhất quán với lịch sử sự kiện và cung cấp một dấu vết kiểm toán đầy đủ.

* **Thuộc tính:** transactionId (int), userId (int - FK đến User), type (Enum: DEPOSIT, PURCHASE, SALE\_REVENUE, WITHDRAWAL), amount (Decimal), relatedEntityId (int - ví dụ: orderId), timestamp (Timestamp).

**Lớp WithdrawalRequest (Yêu cầu Rút tiền)**

Đại diện cho một yêu cầu chính thức từ Seller để rút số tiền họ kiếm được. Đối tượng này cũng có một vòng đời riêng.

* **Thuộc tính:** requestId (int), sellerId (int - FK đến User), amount (Decimal), status (Enum: PENDING, APPROVED, REJECTED), requestDate (Timestamp), processedDate (Timestamp).

**3.2.2. Biểu đồ Trình tự (Sequence Diagram)**

Biểu đồ Trình tự được sử dụng để mô hình hóa sự tương tác giữa các đối tượng trong một khoảng thời gian nhất định. Theo yêu cầu của đề bài , biểu đồ sẽ được vẽ cho hai chức năng phức tạp nhất, đại diện cho hai luồng giá trị cốt lõi của hệ thống: luồng tiêu thụ (mua hàng) và luồng tạo lập (đăng bán).

* **Lựa chọn chức năng:**
  1. **Thanh toán Đơn hàng (Use Case UC4):** Chức năng này là trọng tâm của mọi nền tảng thương mại điện tử, liên quan đến nhiều đối tượng, xác thực tài chính và cập nhật quyền sở hữu.
  2. **Tải lên và Đăng bán Bản nhạc (Function F18):** Thay vì chức năng "Quản lý bản quyền" vốn trừu tượng và không được định nghĩa rõ, chức năng F18 được chọn. Chức năng này phức tạp vì nó bao gồm việc xử lý tệp, thực thi logic nghiệp vụ đặc thù (chuyển đổi định dạng bản nhạc), và thay đổi trạng thái của đối tượng (SheetMusic) để chờ kiểm duyệt.

**Luồng 1: Thanh toán Đơn hàng**

Biểu đồ trình tự này minh họa quy trình từng bước khi một Member hoàn tất việc mua hàng từ giỏ hàng, như được mô tả trong Use Case UC4.

A screenshot of a computer

AI-generated content may be incorrect.

* **Mô tả chi tiết luồng sự kiện:**
  1. **Actor Member** nhấn nút "Thanh toán" trên giao diện Giỏ hàng (CartUI).
  2. **CartUI** gửi một yêu cầu checkout() đến **OrderController**.
  3. **OrderController** gọi phương thức processOrder(userId, cartItems) của **OrderService**.
  4. **OrderService** đầu tiên gọi getUser(userId) đến **UserRepository** để lấy thông tin người mua, bao gồm cả số dư.
  5. **OrderService** tính toán totalAmount từ cartItems.
  6. **OrderService** thực hiện kiểm tra nghiệp vụ: if (user.balance >= totalAmount).
  7. **[Nếu đủ số dư]** OrderService bắt đầu một giao dịch cơ sở dữ liệu để đảm bảo tính toàn vẹn (all-or-nothing).
  8. OrderService gọi phương thức executePurchase(buyerId, sellerId, amount, orderId) của **TransactionService**.
  9. TransactionService là chìa khóa cho tính toàn vẹn tài chính. Thay vì chỉ cập nhật số dư một cách trực tiếp, nó đóng gói logic để tạo các mục ghi sổ cái bất biến. Đối với một giao dịch mua hàng, ít nhất hai giao dịch được tạo ra: một giao dịch loại PURCHASE ghi nợ cho người mua và một giao dịch loại SALE\_REVENUE ghi có cho người bán. Cách tiếp cận này đảm bảo rằng ngay cả khi hệ thống gặp lỗi, nhật ký giao dịch vẫn nhất quán và số dư có thể được tính toán lại một cách chính xác.
  10. **TransactionService** tạo hai đối tượng Transaction:
      + Một bản ghi ghi nợ (debit) cho người mua (type = PURCHASE, amount là số âm).
      + Một bản ghi ghi có (credit) cho người bán (type = SALE\_REVENUE, amount là số dương).
  11. TransactionService lưu cả hai bản ghi vào cơ sở dữ liệu thông qua **TransactionRepository**.
  12. Sau khi TransactionService trả về thành công, **OrderService** tạo một đối tượng Order mới và các đối tượng OrderItem tương ứng.
  13. OrderService lưu Order và OrderItem vào CSDL thông qua **OrderRepository**.
  14. OrderService cấp quyền truy cập tệp cho người mua.
  15. OrderService trả về kết quả thành công cho **OrderController**, sau đó giao diện hiển thị thông báo "Thanh toán thành công!".
  16. **[Luồng thay thế: Không đủ số dư]** Tại bước 6, nếu số dư không đủ, OrderService sẽ trả về lỗi, và giao diện sẽ hiển thị thông báo "Số dư không đủ".

**Luồng 2: Tải lên và Đăng bán Bản nhạc**

Biểu đồ này cho thấy quy trình một Seller thực hiện để tải lên một tệp bản nhạc mới để bán, tương ứng với chức năng F18.

* *A screenshot of a computer

  AI-generated content may be incorrect.***Mô tả chi tiết luồng sự kiện:**
  1. **Actor Seller** điền thông tin (tiêu đề, giá, mô tả) và chọn tệp "Sheet Dữ liệu" trên giao diện Tải lên (UploadUI).
  2. **UploadUI** gửi yêu cầu uploadSheet(details, file) đến **SheetMusicController**.
  3. **SheetMusicController** gọi phương thức createSheetMusic(sellerId, details, file) của **SheetMusicService**.
  4. **SheetMusicService** đầu tiên gọi **FileStorageService** để lưu tệp gốc vào hệ thống lưu trữ. FileStorageService trả về filePath.
  5. Việc chuyển đổi từ "Sheet Dữ liệu" sang "Sheet Số" là một bước logic nghiệp vụ cốt lõi phải là một phần của quy trình tải lên. Mô tả dự án nhấn mạnh khả năng của hệ thống trong việc tự động tạo ra một bản xem trước thân thiện với người dùng ("Sheet Số") từ tệp dữ liệu phức tạp ("Sheet Dữ liệu"). Việc chuyển đổi này không thể xảy ra ở phía client vì lý do bảo mật và nhất quán; nó phải là một quy trình ở backend.
  6. Do đó, sau khi tệp được lưu trữ an toàn, **SheetMusicService** gọi phương thức convert(filePath) của **SheetMusicConversionService**. Dịch vụ này đọc tệp, phân tích cú pháp và trả về chuỗi previewContent (định dạng "Sheet Số").
  7. **SheetMusicService** tạo một đối tượng SheetMusic mới với tất cả thông tin, bao gồm filePath, previewContent, và quan trọng là đặt status = 'PENDING\_APPROVAL'.
  8. SheetMusicService lưu đối tượng SheetMusic mới vào CSDL thông qua **SheetMusicRepository**.
  9. SheetMusicService trả về kết quả thành công cho **SheetMusicController**.
  10. Giao diện hiển thị thông báo "Tải lên thành công! Bản nhạc của bạn đang chờ kiểm duyệt."

**3.3. Thiết kế Cơ sở dữ liệu**

Phần này trình bày mô hình dữ liệu quan hệ, là sự cụ thể hóa của Biểu đồ Lớp (3.2.1) thành một lược đồ cơ sở dữ liệu có thể triển khai được. Thiết kế này đảm bảo tính toàn vẹn, nhất quán và hiệu quả truy xuất dữ liệu.

**3.3.1. Sơ đồ Quan hệ Thực thể (Entity-Relationship Diagram - ERD)**

Sơ đồ ERD dưới đây mô tả các thực thể chính trong cơ sở dữ liệu, các thuộc tính của chúng và mối quan hệ (một-nhiều, nhiều-nhiều) giữa chúng.

*A screenshot of a computer

AI-generated content may be incorrect.*Một điểm quan trọng trong thiết kế ERD là việc chuyển đổi mô hình kế thừa trong lập trình hướng đối tượng sang lược đồ cơ sở dữ liệu quan hệ. Biểu đồ Lớp cho thấy Seller và Admin kế thừa từ User. Việc tạo các bảng riêng biệt users, sellers, và admins trong cơ sở dữ liệu quan hệ là không hiệu quả và dẫn đến sự phức tạp khi cần kết nối dữ liệu. Mẫu hình tiêu chuẩn trong ngành là "Single Table Inheritance" (Kế thừa trên một bảng duy nhất). Theo đó, một bảng users duy nhất được sử dụng cho tất cả các vai trò. Một cột role sẽ phân biệt giữa chúng. Các thuộc tính chỉ dành riêng cho một lớp con (như payout\_info cho Seller) sẽ trở thành các cột có thể nhận giá trị NULL trong bảng users. Do đó, ERD sẽ chỉ hiển thị một thực thể users, thể hiện sự hiểu biết thực tế về các mẫu thiết kế cơ sở dữ liệu tương ứng với các khái niệm hướng đối tượng.

**3.3.2. Mô tả Chi tiết các Bảng (Data Dictionary)**

Dưới đây là mô tả chi tiết cho từng bảng trong cơ sở dữ liệu.

**Bảng users**

Đây là bảng trung tâm cho việc xác thực và phân quyền, lưu trữ thông tin cho tất cả các tác nhân của hệ thống. Thiết kế của nó ảnh hưởng trực tiếp đến bảo mật và quản lý người dùng.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên cột** | **Kiểu dữ liệu** | **Ràng buộc** | **Mô tả** |
| **user\_id** | INT | PRIMARY KEY, AUTO\_INCREMENT | Khóa chính |
| **username** | VARCHAR(50) | UNIQUE, NOT NULL | Tên đăng nhập |
| **password\_hash** | VARCHAR(255) | NOT NULL | Mật khẩu đã băm |
| **email** | VARCHAR(100) | UNIQUE, NOT NULL | Email |
| **avatar\_url** | VARCHAR(255) | NULL | URL ảnh đại diện |
| **balance** | DECIMAL(10, 2) | NOT NULL, DEFAULT 0.00 | Số dư "Xu" |
| **role** | ENUM('MEMBER', 'SELLER', 'ADMIN') | NOT NULL, DEFAULT 'MEMBER' | Vai trò người dùng |
| **payout\_info** | TEXT | NULL | Thông tin nhận tiền cho Seller |
| **created\_at** | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | Ngày tạo |

**Bảng sheet\_music**

|  |  |  |  |
| --- | --- | --- | --- |
| Tên cột | Kiểu dữ liệu | Ràng buộc | Mô tả |
| sheet\_id | INT | PRIMARY KEY, AUTO\_INCREMENT | Khóa chính |
| seller\_id | INT | FOREIGN KEY (users.user\_id) | ID người bán |
| title | VARCHAR(255) | NOT NULL | Tên bản nhạc |
| description | TEXT | NULL | Mô tả chi tiết |
| price | DECIMAL(10, 2) | NOT NULL | Giá bán (bằng "Xu") |
| file\_path | VARCHAR(255) | NOT NULL | Đường dẫn tệp "Sheet Dữ liệu" |
| preview\_content | TEXT | NULL | Nội dung "Sheet Số" để xem trước |
| status | ENUM('PENDING', 'APPROVED', 'REJECTED') | NOT NULL, DEFAULT 'PENDING' | Trạng thái kiểm duyệt |
| created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | Ngày đăng |

**Bảng orders**

|  |  |  |  |
| --- | --- | --- | --- |
| Tên cột | Kiểu dữ liệu | Ràng buộc | Mô tả |
| order\_id | INT | PRIMARY KEY, AUTO\_INCREMENT | Khóa chính |
| buyer\_id | INT | FOREIGN KEY (users.user\_id) | ID người mua |
| order\_date | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | Ngày đặt hàng |
| total\_amount | DECIMAL(10, 2) | NOT NULL | Tổng giá trị đơn hàng |

**Bảng order\_items**

|  |  |  |  |
| --- | --- | --- | --- |
| Tên cột | Kiểu dữ liệu | Ràng buộc | Mô tả |
| order\_item\_id | INT | PRIMARY KEY, AUTO\_INCREMENT | Khóa chính |
| order\_id | INT | FOREIGN KEY (orders.order\_id) | ID đơn hàng |
| sheet\_id | INT | FOREIGN KEY (sheet\_music.sheet\_id) | ID bản nhạc |
| price\_at\_purchase | DECIMAL(10, 2) | NOT NULL | Giá tại thời điểm mua |

**Bảng transactions**

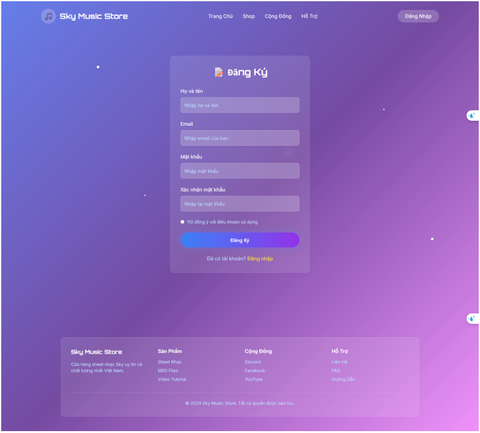
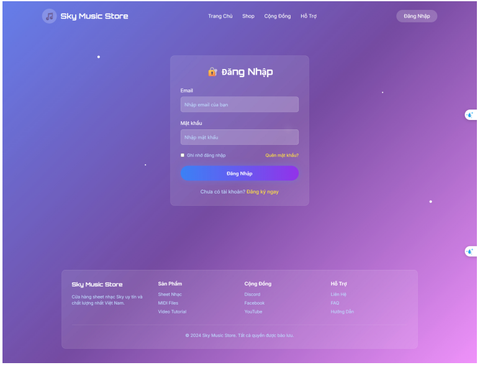
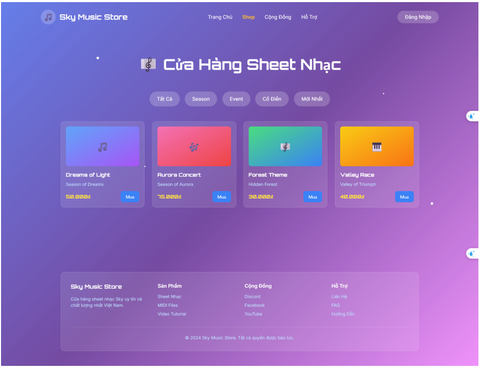
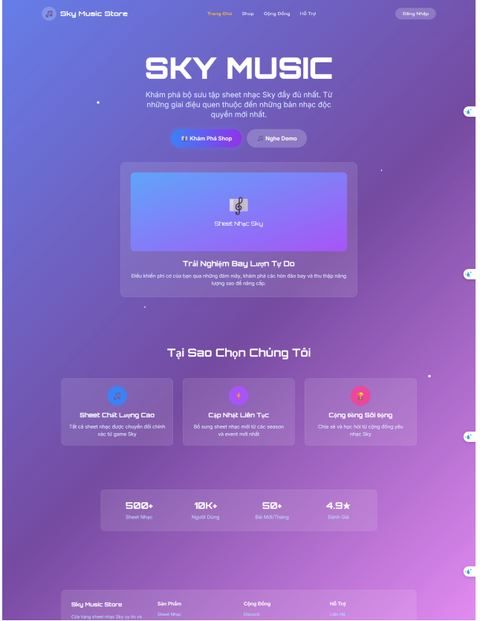
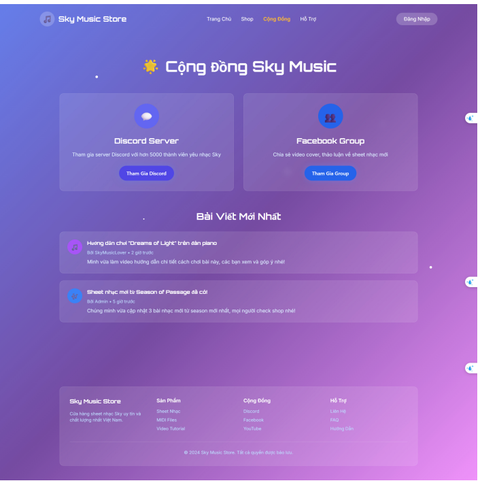
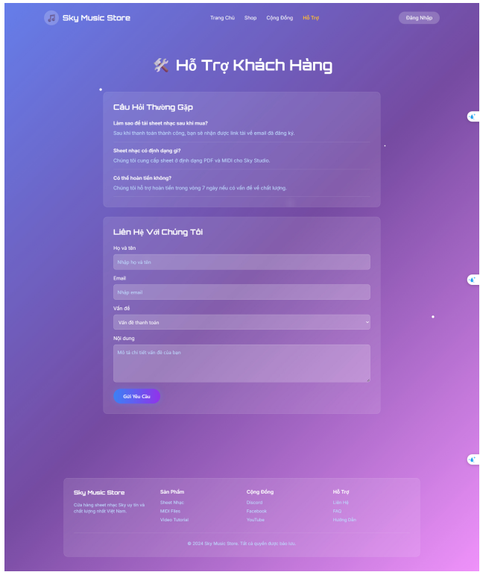
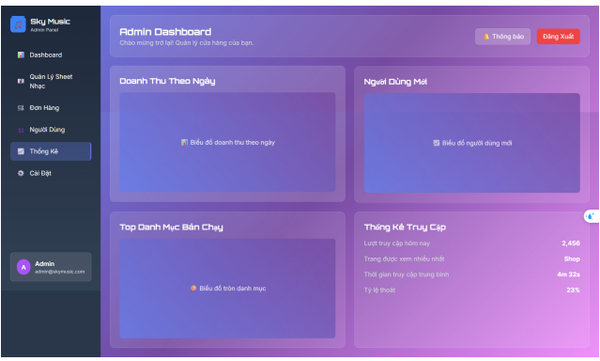
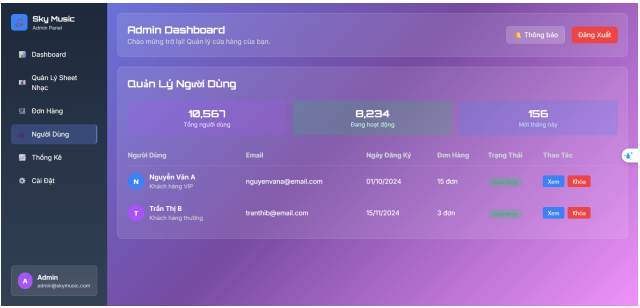
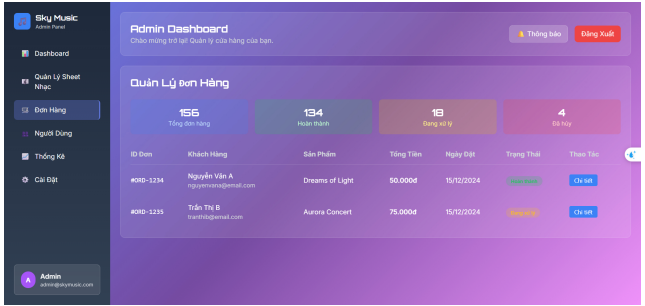
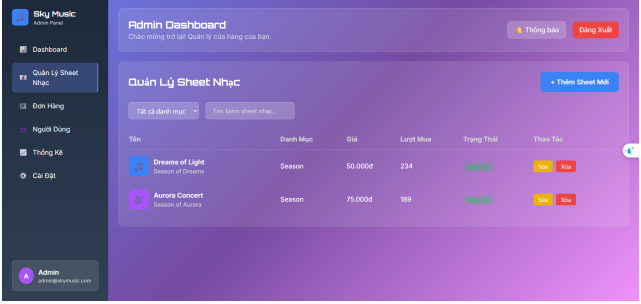
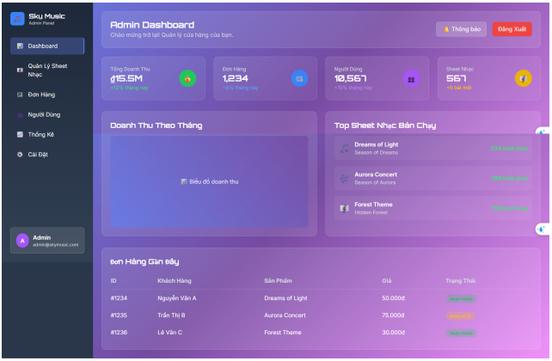
Bảng này là nền tảng của tính toàn vẹn tài chính trong hệ thống. Nó cung cấp một dấu vết kiểm toán đầy đủ, bất biến cho mọi sự dịch chuyển của "Xu", làm cho hệ thống trở nên mạnh mẽ và đáng tin cậy.

|  |  |  |  |
| --- | --- | --- | --- |
| Tên cột | Kiểu dữ liệu | Ràng buộc | Mô tả |
| transaction\_id | INT | PRIMARY KEY, AUTO\_INCREMENT | Khóa chính |
| user\_id | INT | FOREIGN KEY (users.user\_id) | ID người dùng liên quan |
| type | ENUM('DEPOSIT', 'PURCHASE', 'SALE\_REVENUE', 'WITHDRAWAL') | NOT NULL | Loại giao dịch |
| amount | DECIMAL(10, 2) | NOT NULL | Số tiền (âm là chi, dương là thu) |
| related\_entity\_id | INT | NULL | ID của thực thể liên quan (vd: order\_id) |
| created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | Thời điểm giao dịch |

**Bảng withdrawal\_requests**

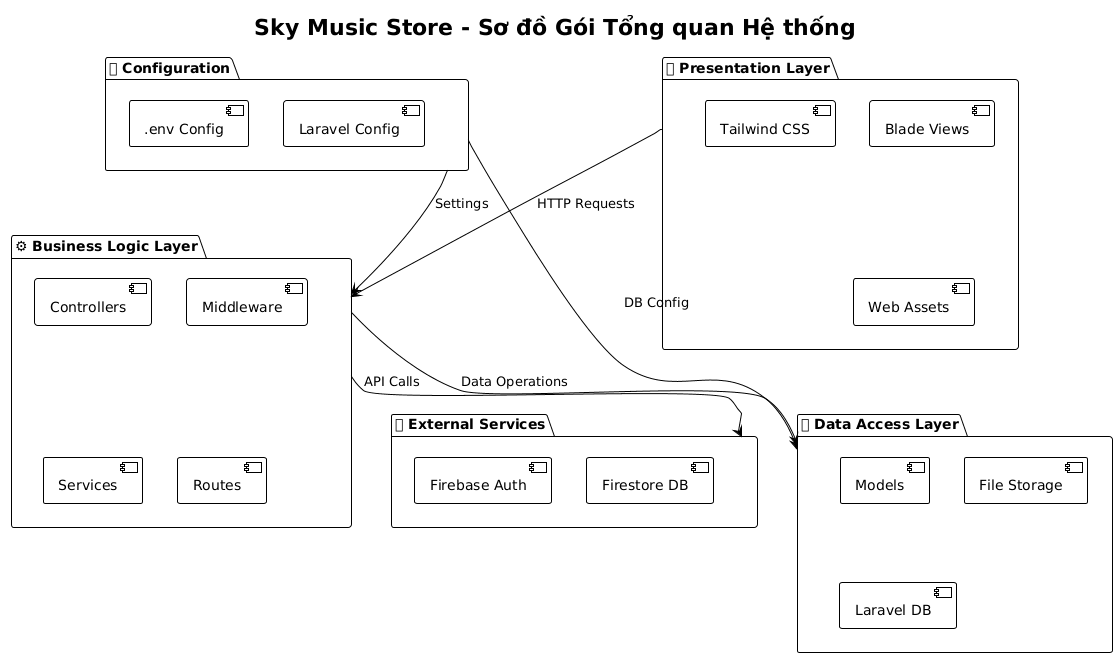
|  |  |  |  |
| --- | --- | --- | --- |
| Tên cột | Kiểu dữ liệu | Ràng buộc | Mô tả |
| request\_id | INT | PRIMARY KEY, AUTO\_INCREMENT | Khóa chính |
| seller\_id | INT | FOREIGN KEY (users.user\_id) | ID người bán yêu cầu |
| amount | DECIMAL(10, 2) | NOT NULL | Số tiền yêu cầu rút |
| status | ENUM('PENDING', 'APPROVED', 'REJECTED') | NOT NULL, DEFAULT 'PENDING' | Trạng thái yêu cầu |
| created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | Ngày yêu cầu |
| processed\_at | TIMESTAMP | NULL | Ngày xử lý |

**3.4. Thiết kế giao diện (UI/UX)**

* *Chèn một vài bản thiết kế (mockup/prototype) cho các trang chính như Trang chủ, Trang chi tiết sản phẩm, Giỏ hàng.*
* 
* 
* 
* 
* 
* 
* 

3.5. Package Diagram

3.5.1. System level package (Sơ đồ gói tổng quan hệ thống)



Hệ thống **Sky Music Store** được tổ chức theo mô hình bốn tầng mở rộng dựa trên nguyên tắc Separation of Concerns và 3-tier architecture truyền thống, bao gồm tầng Dịch vụ Bên ngoài, tầng Giao diện, tầng Xử lý Nghiệp vụ và tầng Truy cập Dữ liệu. Ngoài ra, hệ thống còn có tầng Cấu hình đóng vai trò hỗ trợ xuyên suốt. Cách tổ chức này giúp hệ thống đảm bảo khả năng mở rộng, bảo trì dễ dàng và tối ưu hóa hiệu suất trong quá trình vận hành.

Tầng Dịch vụ Bên ngoài (External Services) đóng vai trò cung cấp các dịch vụ lõi cho toàn hệ thống. Firebase Authentication được sử dụng để quản lý toàn bộ quy trình xác thực và phân quyền người dùng thông qua custom claims. Firestore Database hoạt động như một cơ sở dữ liệu NoSQL chính, lưu trữ metadata của các sản phẩm âm nhạc, thông tin người bán và các dữ liệu phi cấu trúc khác. Việc tận dụng các dịch vụ của Firebase giúp hệ thống Sky Music Store kế thừa được khả năng mở rộng và độ tin cậy cao của Google Cloud Platform, đồng thời giảm thiểu chi phí và độ phức tạp khi triển khai hạ tầng.

Tầng Giao diện (Presentation Layer) chịu trách nhiệm xử lý toàn bộ các tương tác với người dùng cuối. Blade Templates của Laravel cung cấp cơ chế render HTML động, giúp hiển thị dữ liệu từ backend một cách linh hoạt và hiệu quả. Tailwind CSS được tích hợp như một framework CSS hiện đại theo hướng utility-first, giúp xây dựng giao diện thân thiện, responsive và dễ tùy chỉnh. Các tệp JavaScript và CSS được biên dịch, tối ưu hóa qua Vite nhằm đảm bảo tốc độ tải nhanh và trải nghiệm mượt mà. Giao diện hệ thống được chia thành các module riêng biệt cho từng nhóm người dùng như admin, seller, user và guest, từ đó tăng cường tính bảo mật và kiểm soát phân quyền ngay từ tầng giao diện.

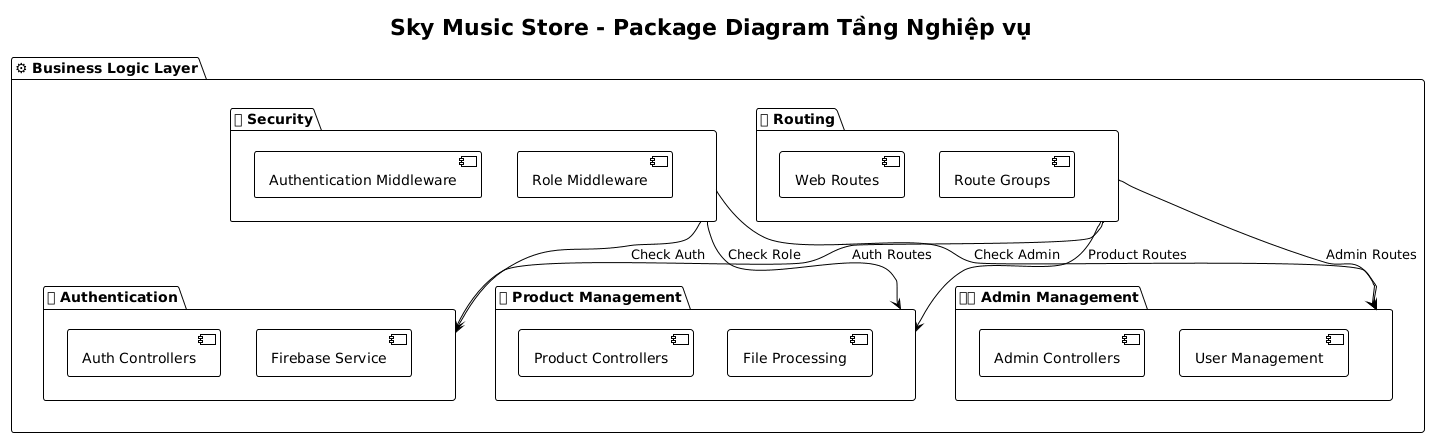
Tầng Xử lý Nghiệp vụ (Business Logic Layer) là trung tâm điều phối hoạt động của hệ thống, nơi chứa toàn bộ logic xử lý. Controllers đóng vai trò tiếp nhận và xử lý các HTTP request, kiểm tra tính hợp lệ của dữ liệu và điều phối các nghiệp vụ. Middleware Stack bao gồm các lớp như EnsureFirebaseAuthenticated và RoleMiddleware, giúp kiểm soát quyền truy cập và đảm bảo chỉ người dùng hợp lệ mới có thể sử dụng tài nguyên tương ứng. Services Layer chứa các nghiệp vụ phức tạp hơn, chẳng hạn như FirestoreSimple service để trừu tượng hóa các thao tác với Firestore, File Processing Service để xử lý upload và phân tích file nhạc, cùng Authentication Service để quản lý session và ánh xạ vai trò người dùng. Hệ thống Routes định nghĩa các RESTful endpoints và áp dụng middleware tương ứng cho từng nhóm chức năng.

Tầng Truy cập Dữ liệu (Data Access Layer) quản lý mọi hoạt động lưu trữ và truy xuất dữ liệu. Các Models đóng vai trò là lớp trừu tượng đại diện cho các thực thể dữ liệu. Product Model được tích hợp với Firestore thông qua FirestoreSimple service, trong khi User Model sử dụng Eloquent ORM của Laravel để làm việc với cơ sở dữ liệu truyền thống. File Storage được tổ chức khoa học, phân loại theo khu vực địa lý và loại file, giúp quản lý, sao lưu và phục hồi dữ liệu hiệu quả hơn. Các thành phần như Laravel DB Components đảm nhiệm việc quản lý session, caching và queue jobs, hỗ trợ xử lý thời gian thực với hiệu suất cao.

Tầng Cấu hình (Configuration Layer) giữ vai trò quan trọng trong việc cung cấp và kiểm soát các thông số hệ thống. File .env chứa các biến môi trường quan trọng như Firebase credentials, database connections và API keys, được bảo vệ khỏi hệ thống quản lý phiên bản để đảm bảo bảo mật. Các file cấu hình của Laravel cung cấp thông tin cấu hình có cấu trúc rõ ràng cho từng dịch vụ, đồng thời hỗ trợ override theo từng môi trường triển khai khác nhau, giúp dễ dàng mở rộng và bảo trì.

Luồng dữ liệu chính của hệ thống bắt đầu từ các HTTP request tại tầng Giao diện, được xử lý qua tầng Xử lý Nghiệp vụ và cuối cùng tương tác với tầng Truy cập Dữ liệu hoặc tầng Dịch vụ Bên ngoài để thực hiện các tác vụ cần thiết. Toàn bộ các tầng được thiết kế theo nguyên tắc one-way dependency, trong đó tầng cao hơn chỉ phụ thuộc vào tầng thấp hơn. Điều này giúp đảm bảo tính độc lập giữa các tầng, giảm thiểu sự phụ thuộc chặt chẽ và tăng tính linh hoạt trong phát triển. Kiến trúc này không chỉ mang lại khả năng mở rộng và bảo trì tốt mà còn giúp việc kiểm thử (testing) trở nên dễ dàng hơn thông qua khả năng mock các dependencies giữa các tầng.

3.5.2. Sơ đồ gói Tầng xử lý nghiệp vụ (Business Logic Package Diagram)



Tầng **Xử lý Nghiệp vụ (Business Logic Layer)** của hệ thống **Sky Music Store** được tổ chức thành năm package chính, mỗi package đảm nhận một nhóm chức năng cụ thể và có sự tương tác chặt chẽ với nhau để tạo thành một hệ thống xử lý nghiệp vụ hoàn chỉnh. Kiến trúc này được xây dựng dựa trên nguyên tắc Single Responsibility Principle, đảm bảo mỗi package có trách nhiệm riêng biệt, rõ ràng và giúp hệ thống đạt được khả năng mở rộng, bảo trì cũng như kiểm thử hiệu quả.

**Authentication Package** đóng vai trò là trung tâm xác thực của toàn bộ hệ thống, bao gồm các thành phần chính là Auth Controllers và Firebase Service. Các Auth Controllers chịu trách nhiệm xử lý những request liên quan đến đăng nhập, đăng ký và quên mật khẩu thông qua các controller chuyên biệt như LoginController, RegisterController và ForgotPasswordController. Firebase Service đảm nhận việc tích hợp với Firebase Authentication API, thực hiện xác thực phía server và quản lý custom claims phục vụ cho phân quyền người dùng. Package này được triển khai trong thư mục Auth và tương tác trực tiếp với Firebase thông qua thư viện kreait/firebase-php, đảm bảo tính bảo mật, độ tin cậy và khả năng mở rộng cao cho hệ thống xác thực.

**Product Management Package** chịu trách nhiệm quản lý toàn bộ vòng đời của sản phẩm âm nhạc trong hệ thống. Các Product Controllers bao gồm ProductController và SalerController, xử lý các tác vụ CRUD cho sản phẩm, quản lý metadata và thông tin người bán. Bên cạnh đó, File Processing Component đóng vai trò đặc biệt quan trọng trong việc xử lý upload file nhạc, phân tích nội dung từ các định dạng file như JSON và text, cũng như giải quyết các vấn đề liên quan đến encoding như phát hiện BOM hoặc chuyển đổi bảng mã ký tự. Package này được triển khai chủ yếu tại thư mục Seller, tích hợp với Firestore để lưu trữ metadata của sản phẩm, đồng thời quản lý file lưu trữ cục bộ tại thư mục seller\_files.

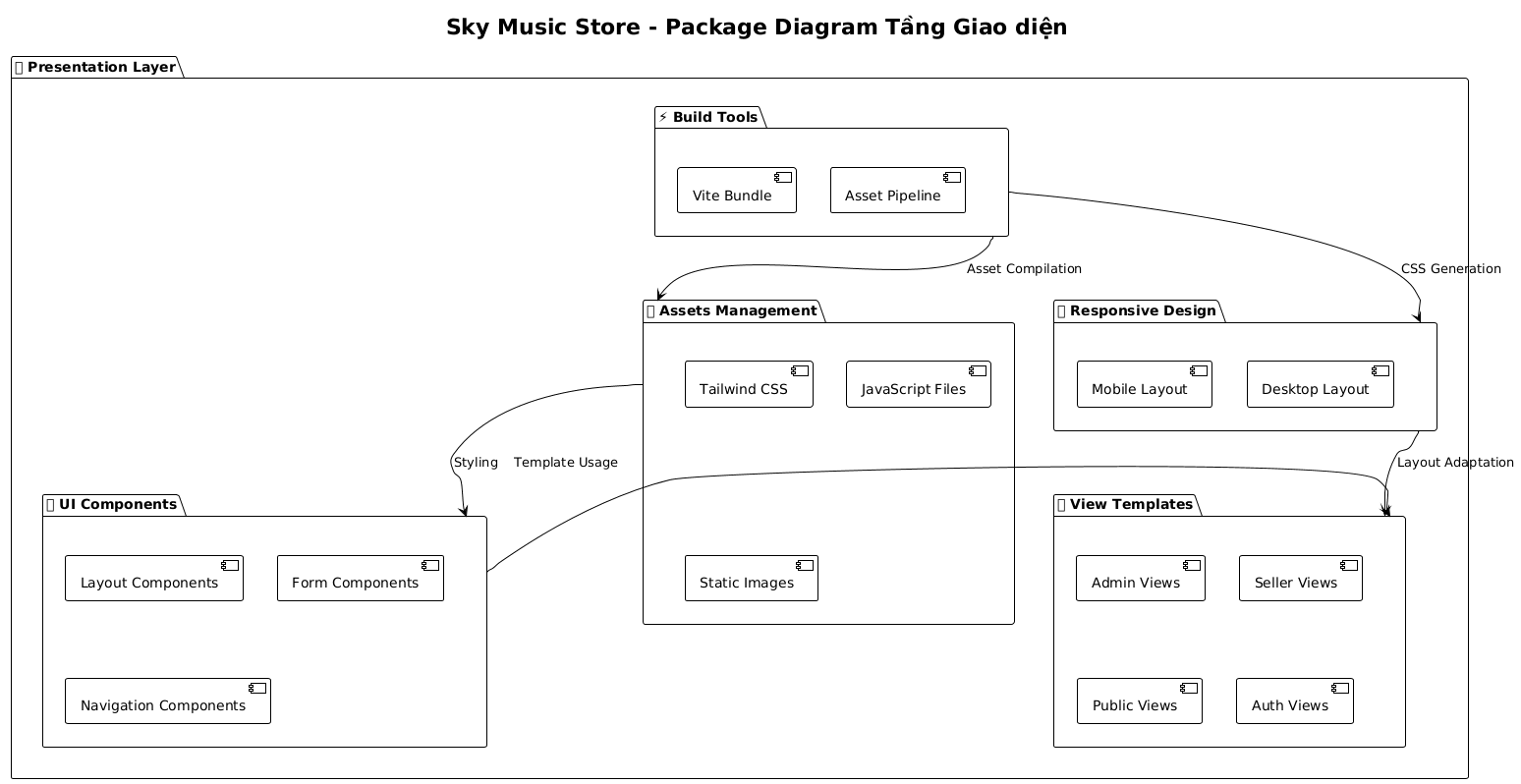
**Admin Management Package** cung cấp các chức năng phục vụ công tác quản trị hệ thống và quản lý người dùng. Các Admin Controllers chịu trách nhiệm xử lý các request đến từ admin dashboard, bao gồm quản lý sản phẩm, đơn hàng và thống kê hoạt động của hệ thống. Thành phần User Management Component, được triển khai thông qua UserRoleController, cho phép admin thực hiện việc thay đổi vai trò của người dùng bằng cách cập nhật custom claims trên Firebase Authentication. Package này đảm bảo chỉ có người dùng có quyền admin mới có thể truy cập và thao tác với các chức năng quan trọng, nhờ sự tích hợp chặt chẽ với Security Package.

**Security Package** giữ vai trò cốt lõi trong việc bảo vệ toàn bộ hệ thống thông qua cơ chế middleware, bao gồm Authentication Middleware và Role Middleware. Middleware EnsureFirebaseAuthenticated kiểm tra sự tồn tại của firebase\_uid trong session để đảm bảo người dùng đã được xác thực, trong khi RoleMiddleware thực hiện kiểm tra vai trò của người dùng nhằm bảo đảm quyền truy cập phù hợp với từng loại tài khoản. Các middleware này được khai báo tại thư mục Middleware và được tự động áp dụng cho các nhóm route thông qua định nghĩa trong file web.php, tạo thành một lớp bảo vệ thống nhất và an toàn cho toàn bộ ứng dụng.

**Routing Package** đảm nhiệm vai trò định tuyến và tổ chức các route của hệ thống theo từng nhóm chức năng. Các Web Routes định nghĩa các endpoint RESTful cho ứng dụng, trong khi Route Groups được sử dụng để nhóm các route theo các prefix như /admin, /saler và /account, mỗi nhóm route đều được gắn với middleware phù hợp để đảm bảo an toàn và kiểm soát truy cập. Package này được triển khai trực tiếp trong file web.php và đóng vai trò là điểm đầu vào của toàn bộ hệ thống, điều phối các request đến đúng controller và áp dụng các lớp bảo mật cần thiết.

Mối quan hệ giữa các package được thiết kế theo mô hình layered architecture, trong đó Routing Package đóng vai trò là entry point, điều phối request đến các package chức năng như Authentication, Product Management và Admin Management. Song song đó, Security Package hoạt động như một cross-cutting concern, bảo vệ toàn bộ quy trình xử lý thông qua hệ thống middleware. Cấu trúc này giúp hệ thống duy trì được sự tách biệt giữa các chức năng (separation of concerns), đồng thời cho phép kiểm thử độc lập từng package và dễ dàng mở rộng thêm tính năng mới mà không ảnh hưởng đến các thành phần còn lại của tầng xử lý nghiệp vụ.

3.5.3. Sơ đồ gói tầng giao diện (Presentation Layer package diagram)



Tầng **Giao diện (Presentation Layer)** của hệ thống **Sky Music Store** được thiết kế theo kiến trúc component-based, chia thành năm package chính nhằm đảm bảo khả năng tái sử dụng cao, dễ bảo trì và mở rộng trong quá trình phát triển. Kiến trúc này tuân thủ chặt chẽ các nguyên tắc Separation of Concerns và DRY (Don’t Repeat Yourself), giúp xây dựng giao diện người dùng hiệu quả, nhất quán và tối ưu trên toàn bộ ứng dụng.

**View Templates Package** đóng vai trò là lớp cao nhất trong tầng giao diện, chịu trách nhiệm tổ chức các view theo từng nhóm người dùng cụ thể. Admin Views được thiết kế cho giao diện quản trị hệ thống, bao gồm các trang như dashboard analytics, quản lý người dùng và cấu hình hệ thống, được triển khai trong thư mục admin. Seller Views cung cấp giao diện cho người bán, tập trung vào các chức năng quản lý sản phẩm, tải lên file nhạc và theo dõi doanh số, được tổ chức trong thư mục saler. Public Views bao gồm các trang công khai như homepage, shop, community và support, được lưu trữ trong thư mục page. Auth Views xử lý các trang liên quan đến xác thực như đăng nhập, đăng ký và đặt lại mật khẩu, được đặt trong thư mục auth. Mỗi nhóm view đều được thiết kế với bố cục, màu sắc và chức năng phù hợp với từng vai trò người dùng, đồng thời đảm bảo tính bảo mật và phân quyền từ tầng giao diện.

**UI Components Package** cung cấp các thành phần giao diện có thể tái sử dụng trong toàn hệ thống. Layout Components bao gồm các layout chính như app.blade.php, admin.blade.php và account.blade.php, định nghĩa khung bố cục chung cho các trang khác nhau. Form Components cung cấp các phần tử biểu mẫu tiêu chuẩn như ô nhập liệu, nút bấm, hộp chọn và thông báo lỗi với thiết kế và validation đồng nhất. Navigation Components quản lý các thành phần điều hướng như menu, breadcrumb và thanh điều hướng (navigation bar), giúp người dùng có trải nghiệm mượt mà và nhất quán khi tương tác. Các thành phần này được triển khai tại thư mục components và layouts, tuân theo hệ thống Blade Component của Laravel, cho phép dễ dàng mở rộng và duy trì.

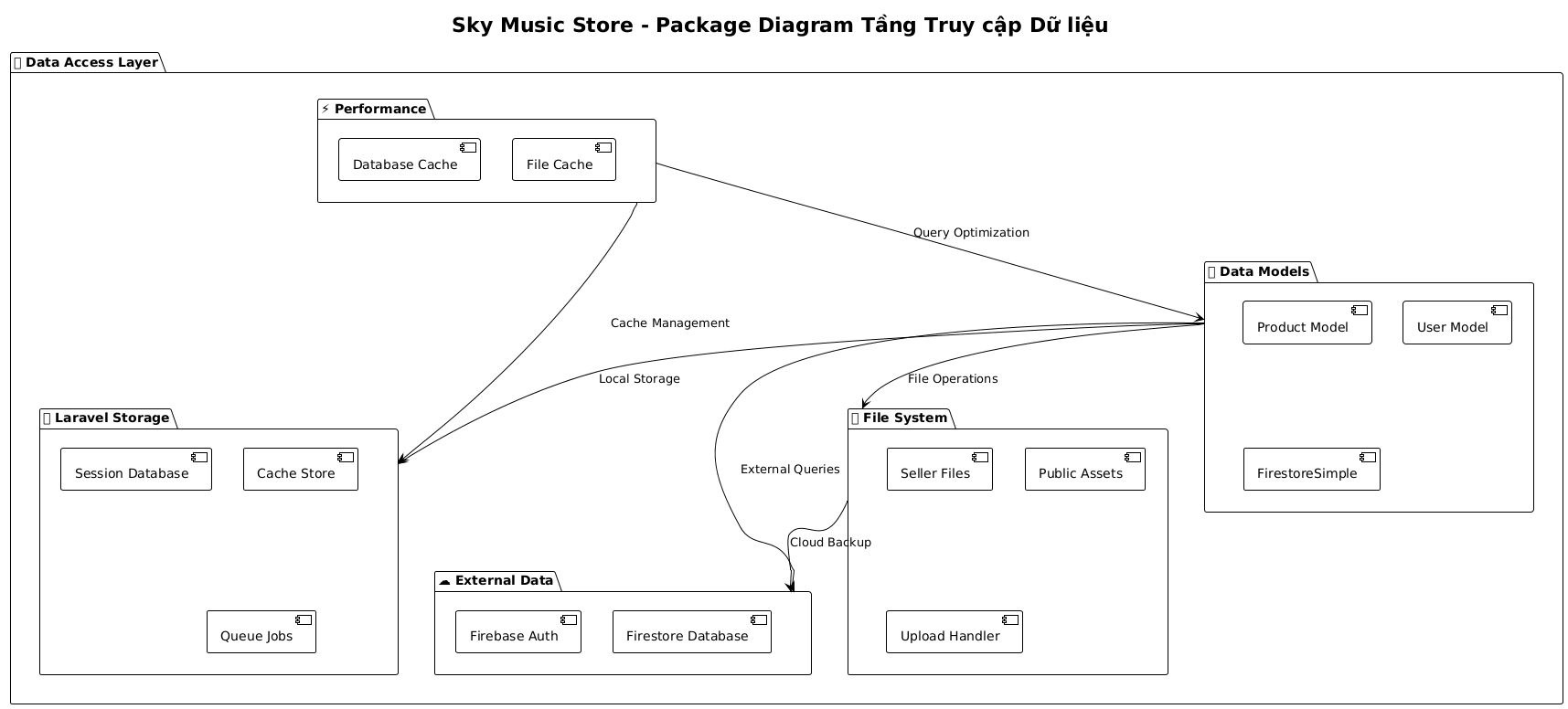
**Assets Management Package** chịu trách nhiệm quản lý toàn bộ tài nguyên tĩnh của ứng dụng. Tailwind CSS được sử dụng làm framework chính theo hướng utility-first, mang lại khả năng tùy chỉnh cao và đảm bảo thiết kế responsive cho toàn bộ giao diện thông qua file app.css. Các file JavaScript quản lý các tác vụ tương tác phía client như kiểm tra dữ liệu biểu mẫu, tải nội dung động và xử lý hiệu ứng người dùng, được tổ chức tại thư mục js. Thư mục img lưu trữ hình ảnh tĩnh như logo, biểu tượng, hình nền và avatar mặc định, được tối ưu hóa để đạt hiệu năng cao khi hiển thị trên web.

**Build Tools Package** đảm nhận vai trò biên dịch, tối ưu hóa và gói (bundle) các tài nguyên trước khi đưa vào môi trường production. Vite Bundle hoạt động như công cụ build hiện đại, cung cấp tính năng hot module replacement trong giai đoạn phát triển và tối ưu hóa asset cho sản xuất. Asset Pipeline thực hiện biên dịch Sass/CSS, nén JavaScript, tối ưu hình ảnh và xử lý cache busting thông qua versioning. Gói này được cấu hình thông qua các file vite.config.js và package.json, tích hợp trực tiếp với hệ thống biên dịch tài nguyên của Laravel để đảm bảo quy trình build nhanh và nhất quán.

**Responsive Design Package** đảm bảo giao diện của hệ thống hoạt động tốt trên mọi loại thiết bị và kích thước màn hình. Mobile Layout được phát triển theo hướng mobile-first, tối ưu hóa cho các thao tác cảm ứng và băng thông hạn chế. Desktop Layout cung cấp trải nghiệm phong phú hơn với bố cục nhiều cột, thanh điều hướng bên và các thành phần giao diện nâng cao. Package này tận dụng các tiện ích responsive utilities của Tailwind CSS cùng CSS Grid và Flexbox để xây dựng các bố cục linh hoạt, thích ứng và hiệu năng cao trên nhiều nền tảng.

Mối quan hệ giữa các package trong tầng giao diện được thiết kế theo một hệ thống phụ thuộc có thứ bậc rõ ràng. Build Tools Package đóng vai trò nền tảng, chịu trách nhiệm biên dịch và tối ưu hóa các tài nguyên từ Assets Management Package, sau đó cung cấp cho UI Components Package. View Templates Package sử dụng các thành phần giao diện từ UI Components để xây dựng các trang hoàn chỉnh, trong khi Responsive Design Package đóng vai trò là yếu tố cắt ngang (cross-cutting concern), ảnh hưởng đến toàn bộ các tầng còn lại nhằm đảm bảo khả năng hiển thị và trải nghiệm người dùng tối ưu trên mọi thiết bị. Kiến trúc này không chỉ mang lại hiệu suất cao mà còn hỗ trợ phát triển song song giữa các nhóm lập trình viên, dễ bảo trì và đáp ứng tốt các tiêu chuẩn về progressive enhancement cũng như web accessibility.

3.5.4. Sơ đồ gói tầng Dữ liệu (Data Access Layer)



Tầng truy cập dữ liệu của **Sky Music Store** được thiết kế theo kiến trúc **hybrid**, kết hợp giữa các giải pháp **cloud-native** và **traditional database** nhằm tận dụng ưu điểm của cả hai mô hình. Cấu trúc này được chia thành năm **package chính**, mỗi package đảm nhận một nhóm chức năng cụ thể trong việc **quản lý, lưu trữ và truy xuất dữ liệu**, đảm bảo hiệu suất cao, khả năng mở rộng tốt và dễ dàng tích hợp với các thành phần khác của hệ thống.

**External Data Package** đóng vai trò là nền tảng cho các hoạt động **authentication** và **product data storage**. Firebase Auth quản lý toàn bộ tài khoản người dùng, thông tin xác thực và phân quyền dựa trên **role-based permissions** thông qua cơ chế **custom claims**, mang lại mức độ bảo mật cao và khả năng mở rộng linh hoạt mà không cần duy trì hạ tầng xác thực phức tạp. Firestore Database được sử dụng làm **primary data store** cho metadata của sản phẩm, thông tin người bán và các thực thể kinh doanh khác, hỗ trợ **real-time synchronization** và khả năng phân phối dữ liệu toàn cầu. Các thành phần này được tích hợp thông qua **kreait/firebase-php library** và **FirestoreSimple service**, giúp việc kết nối với Laravel diễn ra mượt mà trong khi vẫn duy trì lợi thế của mô hình cloud-native.

**Laravel Storage Package** cung cấp các chức năng cơ sở dữ liệu truyền thống cho các nghiệp vụ cần **ACID properties** và **complex queries**. **Session Database** chịu trách nhiệm quản lý phiên đăng nhập của người dùng, đảm bảo tính ổn định và tuân thủ các tiêu chuẩn bảo mật của Laravel. **Cache Store** sử dụng cơ chế lưu trữ dữ liệu trong cơ sở dữ liệu để tăng tốc độ phản hồi và giảm số lần truy cập API bên ngoài, đặc biệt hữu ích cho các dữ liệu được truy cập thường xuyên như hồ sơ người dùng hoặc danh sách sản phẩm. Hệ thống **Queue Jobs** xử lý các tác vụ bất đồng bộ như xử lý tệp, gửi email hay đồng bộ dữ liệu, được cấu hình để sử dụng driver database thông qua các thiết lập như SESSION\_DRIVER=database, CACHE\_STORE=database, và QUEUE\_CONNECTION=database.

**File System Package** chịu trách nhiệm quản lý toàn bộ quá trình lưu trữ tệp và **digital asset management** của nền tảng. Thư mục **Seller Files** được tổ chức theo cấu trúc phân cấp tại public/seller\_files/{seller\_id}/{category}/, giúp việc tổ chức và kiểm soát quyền truy cập trở nên hiệu quả hơn. **Public Assets** chứa các tài nguyên tĩnh như CSS, JavaScript bundles, hình ảnh và phông chữ, được tối ưu hóa cho việc phân phối web và tích hợp CDN. Thành phần **Upload Handler** xử lý các nghiệp vụ phức tạp liên quan đến tệp, bao gồm xác thực định dạng, nhận diện mã hóa, trích xuất metadata và lưu trữ an toàn – đặc biệt quan trọng đối với các tệp nhạc có nhiều định dạng và tiêu chuẩn mã hóa khác nhau.

**Data Models Package** đóng vai trò là lớp **abstraction layer** và tích hợp logic nghiệp vụ trong quá trình truy xuất dữ liệu. **Product Model** triển khai cơ chế tương tự ORM để tương tác với Firestore, cung cấp các thao tác CRUD, phương thức truy vấn và quản lý mối quan hệ giữa các thực thể sản phẩm. **User Model** sử dụng **Eloquent ORM** của Laravel cho các hoạt động liên quan đến cơ sở dữ liệu truyền thống, đóng vai trò như nền tảng cho các yêu cầu dữ liệu nội bộ trong tương lai. **FirestoreSimple service** giữ vai trò trung gian, cung cấp **standardized interface** cho các thao tác Firestore như CRUD tài liệu, thực thi truy vấn và chuyển đổi dữ liệu giữa định dạng Firestore và đối tượng PHP.

**Performance Package** tập trung tối ưu hóa hiệu suất truy cập dữ liệu thông qua chiến lược **multi-layered caching**. **Database Cache** lưu trữ kết quả truy vấn để giảm tải cho cơ sở dữ liệu và tăng tốc độ phản hồi cho các dữ liệu được truy cập thường xuyên. **File Cache** quản lý việc lưu tạm dữ liệu metadata của tệp và các tài nguyên tĩnh nhằm giảm số lần truy cập hệ thống file, nâng cao hiệu suất phục vụ. Package này tích hợp chặt chẽ với hệ thống cache của Laravel, có thể mở rộng dễ dàng với **Redis** hoặc **Memcached** trong môi trường production, đồng thời cung cấp cơ chế **cache invalidation** để đảm bảo tính nhất quán của dữ liệu.

Mối quan hệ giữa các package được tổ chức theo **optimized data flow pattern**, đảm bảo dòng dữ liệu di chuyển hợp lý và hiệu quả. **Data Models Package** đóng vai trò trung tâm, điều phối các yêu cầu đến **External Data** và **Laravel Storage** tùy theo yêu cầu nghiệp vụ. **Performance Package** hoạt động như một lớp trong suốt, tự động bắt và cache các truy vấn để cải thiện hiệu suất tổng thể. **File System Package** đảm nhận các thao tác tệp đặc biệt và có khả năng tích hợp với **External Data** để hỗ trợ **cloud backup** và **CDN distribution**. Kiến trúc này không chỉ đảm bảo **tính nhất quán và hiệu suất dữ liệu**, mà còn mang lại **tính linh hoạt cao**, cho phép dễ dàng mở rộng hoặc di chuyển giữa các giải pháp lưu trữ khác nhau trong tương lai, đồng thời duy trì sự tách biệt rõ ràng giữa **business logic** và **data access layer**.

**Chương 4: Triển khai và Kiểm thử (20% điểm)**

**4.1. Công nghệ và Môi trường phát triển**

* **Ngôn ngữ lập trình:** *JavaScript, PHP,...*
* **Framework:** Javascript *(Frontend), Laravel (Backend),...*
* **Cơ sở dữ liệu:** *MySQL, MongoDB,...*
* **Công cụ khác:** *Git/GitHub, VS Code, Trello,...*

**4.2. Kế hoạch kiểm thử (Test Cases)**

* *Xây dựng các kịch bản kiểm thử hộp đen (Black-box testing) cho các chức năng chính (ví dụ: Test case cho chức năng đăng nhập, tìm kiếm,...).*

**4.3. Kết quả kiểm thử**

* *Trình bày kết quả thực hiện các test case đã nêu: Passed/Failed và hình ảnh minh họa nếu cần.*

**Chương 5: Tổng kết và Hướng phát triển (10% điểm)**

**5.1. Kết quả đạt được**

* *Tổng kết các chức năng đã hoàn thành và so sánh với mục tiêu ban đầu.*

**5.2. Hạn chế và Bài học kinh nghiệm**

* *Nêu những khó khăn gặp phải trong quá trình làm (kỹ thuật, làm việc nhóm,...) và những điểm còn hạn chế của sản phẩm.*
* *Bài học kinh nghiệm rút ra được sau khi thực hiện dự án.*

**5.3. Hướng phát triển trong tương lai**

* *Đề xuất các tính năng có thể bổ sung để hoàn thiện sản phẩm (ví dụ: Hệ thống rating/review, gợi ý bản nhạc, tích hợp diễn đàn,...).*

**Lời kết**

**Tài liệu tham khảo**

<https://sky.hhgame.vn/tin-tuc/hanh-trinh-am-nhac-buoi-hoa-nhac-mua-aurora>

<https://sky-children-of-the-light.fandom.com/wiki/Instruments>