## ‘Yêu cầu đối với BTL học phần Thiết kế phần mềm

1. Dự án phần mềm phải thuộc lĩnh vực được phân công và có thể được chọn từ một trong số các loại sau:
   1. Đồ án tốt nghiệp trong vòng 2 năm gần nhất
   2. BTL học phần Nhập môn CNPM trong vòng 2 năm gần nhất
   3. BTL học phần Thu thập và phân tích yêu cầu trong vòng 2 năm gần nhất
   4. Dự án phần mềm thực tế có thể tiếp cận hợp pháp được đầy đủ hồ sơ, mã nguồn
2. Ràng buộc với phần mềm tương tự như với học phần Nhập môn CNPM
3. Tên đề tài có dạng “Thiết kế phần mềm giải quyết bài toán <tự chọn> thuộc lĩnh vực <tên lĩnh vực được phân công>”
4. Phân công trách nhiệm phải đầy đủ (mỗi người tham gia thực hiện đủ các công việc) và ghi rõ theo từng chương
5. Quy cách trình bày báo cáo theo Quy định của khoa
6. Cấu trúc quyển báo cáo theo mẫu dưới đây

**TRƯỜNG ĐẠI HỌC MỞ HÀ NỘI**

**KHOA CÔNG NGHỆ THÔNG TIN**

## ----------------------

## 

**BÁO CÁO BÀI TẬP LỚN**

**MÔN: THIẾT KẾ PHẦN MỀM**

**ĐỀ TÀI:**

**Thiết kế phần mềm cho “phần mềm mạng xã hội chia sẻ âm nhạc SoundShare  
cho Sound Entertainment theo Công nghệ phần mềm”**

Giảng viên hướng dẫn : ThS. Lê Hữu Dũng

Nhóm sinh viên thực hiện : Nhóm SWD.G11

Ngô Minh Phương – 2110A01 Đồng Minh Trí – 2110A04

Phạm Quang Thắng – 2110A02 Nguyễn Đình Phi– 2110A04

Lê Văn Sáng – 2110A05

**Hà Nội - 2025**

## LỜI NÓI ĐẦU

Môn học Thiết kế phần mềm cung cấp cho sinh viên những kiến thức và kỹ năng cần thiết trong việc phát triển và thiết kế phần mềm. Môn học này giúp sinh viên nắm vững các phương pháp và quy trình thiết kế, từ việc thu thập yêu cầu, phân tích, thiết kế hệ thống cho đến triển khai và kiểm thử phần mềm. Mục tiêu là giúp sinh viên có thể thực hiện các công việc thiết kế và phát triển phần mềm một cách bài bản và chuyên nghiệp, đồng thời hiểu rõ các vấn đề và kỹ thuật liên quan đến quá trình này.

Với sự hướng dẫn của ThS. Lê Hữu Dũng, nhóm chúng em đã thực hiện đề tài "Thiết kế phần mềm cho phần mềm mạng xã hội chia sẻ âm nhạc SoundShare cho Sound Entertainment theo Công nghệ phần mềm". Đề tài này là cơ hội để nhóm em áp dụng các kiến thức và kỹ thuật đã học vào việc phát triển một ứng dụng mạng xã hội chia sẻ âm nhạc, nhằm đáp ứng nhu cầu ngày càng cao của người dùng về một nền tảng kết nối và chia sẻ âm nhạc.

Trong quá trình thực hiện dự án này, nhóm chúng em xin chân thành cảm ơn thầy Lê Hữu Dũng đã tận tình hướng dẫn và chia sẻ những kiến thức quý báu, giúp nhóm em hoàn thành được các giai đoạn thiết kế phần mềm. Mặc dù chúng em đã cố gắng hết mình, nhưng không thể tránh khỏi những thiếu sót và hạn chế. Chúng em rất mong nhận được những nhận xét từ thầy và các bạn để có thể hoàn thiện đề tài này hơn nữa.

Chúng em xin chân thành cảm ơn!

## 

## 

**MỤC LỤC**

## 

# 

# CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI

## Tầm quan trọng của Hoạt động Thiết kế phần mềm (phi)

Hoạt động Thiết kế phần mềm đóng vai trò rất quan trọng trong quá trình phát triển phần mềm, bởi vì nó ảnh hưởng trực tiếp đến chất lượng, hiệu quả và khả năng bảo trì của sản phẩm cuối cùng. Cụ thể:

* Đảm bảo chất lượng phần mềm: Thiết kế tốt giúp giảm thiểu lỗi và sự cố trong quá trình phát triển và triển khai phần mềm. Nó giúp xác định rõ ràng cấu trúc và luồng dữ liệu, từ đó giảm thiểu rủi ro phát sinh lỗi logic và lỗi chức năng.
* Tăng hiệu quả phát triển: Bằng cách xác định rõ ràng kiến trúc và cấu trúc phần mềm, nhóm phát triển có thể làm việc hiệu quả hơn, tránh trùng lặp và xung đột trong quá trình lập trình.
* Dễ bảo trì và mở rộng: Thiết kế phần mềm tốt đảm bảo rằng hệ thống dễ bảo trì, sửa lỗi và nâng cấp. Điều này rất quan trọng khi yêu cầu thay đổi hoặc mở rộng tính năng trong tương lai.
* Tối ưu hiệu suất: Một thiết kế hiệu quả sẽ giúp tối ưu hóa tài nguyên hệ thống như bộ nhớ, CPU, và băng thông mạng, từ đó cải thiện hiệu suất của phần mềm.
* Đảm bảo tính bảo mật: Trong bối cảnh an ninh mạng ngày càng phức tạp, thiết kế phần mềm tốt sẽ bao gồm các biện pháp bảo mật cần thiết để bảo vệ dữ liệu và quyền riêng tư của người dùng.
* Đáp ứng yêu cầu người dùng: Hoạt động thiết kế giúp chuyển đổi các yêu cầu người dùng và yêu cầu hệ thống thành các giải pháp kỹ thuật cụ thể, đảm bảo phần mềm cuối cùng đáp ứng đúng nhu cầu của người dùng.
* Cơ sở cho kiểm thử: Một thiết kế chi tiết cung cấp cơ sở để viết các kịch bản kiểm thử chính xác và đầy đủ, đảm bảo phần mềm hoạt động đúng như mong đợi.
* Giảm chi phí và thời gian phát triển: Nhờ vào việc phát hiện và giải quyết sớm các vấn đề tiềm ẩn trong giai đoạn thiết kế, chi phí và thời gian phát triển phần mềm được giảm đáng kể so với việc sửa lỗi ở giai đoạn triển khai hoặc bảo trì.

Hoạt động thiết kế phần mềm không chỉ là bước trung gian mà còn là nền tảng quyết định sự thành công của một dự án phần mềm, đảm bảo phần mềm vừa đáp ứng yêu cầu người dùng, vừa có tính ổn định, hiệu suất cao và bảo mật tốt.

## Giới thiệu bài toán (phi)

Trong thời đại kỹ thuật số, nhu cầu chia sẻ, khám phá và thưởng thức âm nhạc trực tuyến ngày càng tăng cao. Sound Entertainment mong muốn phát triển SoundShare – một mạng xã hội chia sẻ âm nhạc hiện đại, nơi người dùng có thể kết nối, tương tác và khám phá âm nhạc một cách dễ dàng và thú vị.

SoundShare cung cấp các chức năng chính như: Hệ thống tìm kiếm thông minh giúp người dùng tìm kiếm âm nhạc dễ dàng theo tên bài hát, nghệ sĩ, thể loại và nhiều tiêu chí khác; Nghe nhạc trực tuyến mọi lúc, mọi nơi, đồng thời lưu trữ và truy cập danh sách nhạc yêu thích trên mọi thiết bị; Chia sẻ âm nhạc để tạo ra sự tương tác sôi nổi giữa các thành viên trong cộng đồng; Tạo playlist theo chủ đề hoặc sở thích cá nhân và chia sẻ với người khác; Gợi ý âm nhạc cá nhân hóa dựa trên sở thích của người dùng, giúp họ khám phá âm nhạc mới một cách thú vị; Đăng ký, đăng nhập, quản lý tài khoản một cách an toàn và tiện lợi; Quản lý nhạc cá nhân bao gồm đăng, xóa, chỉnh sửa nhạc và kiểm soát quyền riêng tư khi chia sẻ.

Các thách thức của bài toán:

* Quản lý bản quyền và nội dung âm nhạc: Đảm bảo nội dung tuân thủ bản quyền số, xác minh nguồn gốc và cấp phép cho các bản nhạc được tải lên.
* Tối ưu hóa hệ thống tìm kiếm thông minh: Đảm bảo tốc độ và độ chính xác cao khi xử lý khối lượng lớn dữ liệu âm nhạc và truy vấn tìm kiếm.
* Hiệu suất và khả năng mở rộng: Đáp ứng lưu lượng người dùng lớn với tốc độ truyền tải mượt mà và dễ dàng mở rộng khi số lượng nội dung tăng nhanh.
* Bảo mật và quyền riêng tư: Bảo vệ thông tin cá nhân và quyền riêng tư của người dùng, đảm bảo an toàn khi chia sẻ và thanh toán trực tuyến (nếu có).
* Trải nghiệm người dùng (UX/UI): Giao diện thân thiện, dễ sử dụng, đảm bảo trải nghiệm nghe nhạc và tương tác xã hội mượt mà.
* Gợi ý âm nhạc cá nhân hóa: Phát triển thuật toán AI chính xác, phù hợp với sở thích âm nhạc mà không xâm phạm quyền riêng tư.
* Quản lý nội dung và tương tác cộng đồng: Đảm bảo môi trường tương tác lành mạnh, kiểm soát nội dung bình luận, bài đăng không phù hợp và bảo vệ quyền tác giả.

*(phải nêu được các thách thức (issues) của bài toán trong lĩnh vực được phân công)*

## Thông tin về dự án phần mềm được chọn

### Thông tin về tổ chức/môi trường triển khai (phương)

* Thông tin về tổ chức
  + Tổ chức: Công ty Sound Entertainment - địa chỉ: Số 63 Ngọc Hồi - Hoàng Liệt - Hoàng Mai - Hà Nội
  + Nhóm triển khai: SE.G08
  + Thành viên nhóm: Trần Khánh Hùng, Lương Thu Hoài, Bùi Kim Ngọc, Phan Công Long, Võ Thị Thanh, Đỗ Minh Quân, Nguyễn Thị Tình, Nguyễn Đức Cường, Lê Thị Thảo – 2010A02
* Môi trường triển khai:
  + Dự án sử dụng phương pháp Agile-Scrum trong phát triển phần mềm, chú trọng vào tính linh hoạt, hợp tác nhóm, và phát triển theo chu kỳ ngắn.
  + Công nghệ sử dụng bao gồm Node.js, React.js, và MongoDB để xây dựng nền tảng chia sẻ âm nhạc.

### Mục đích, mục tiêu của dự án phần mềm (phương)

* Mục đích:
  + Với người dùng:
    - Khám phá âm nhạc dễ dàng: Tạo ra một công cụ tìm kiếm thông minh, giúp người dùng nhanh chóng tìm ra các bài hát, nghệ sĩ, album yêu thích dựa trên các tiêu chí như tên bài hát, nghệ sĩ, thể loại.
    - Chia sẻ âm nhạc: Cung cấp tính năng chia sẻ âm nhạc với cộng đồng, thúc đẩy sự tương tác giữa người dùng, giúp họ kết nối và giao lưu qua âm nhạc.
    - Tạo và chia sẻ danh sách phát: Người dùng có thể tạo và chia sẻ danh sách phát nhạc của riêng mình, tạo ra các nhóm yêu thích và xây dựng cộng đồng âm nhạc.
    - Tính bảo mật thông tin: Đảm bảo bảo mật và an toàn cho dữ liệu người dùng, từ thông tin tài khoản đến các bài hát cá nhân.
  + Với Sound Entertainment (doanh nghiệp phát triển):
    - Tạo ra nền tảng độc đáo và hấp dẫn: Cung cấp một môi trường để thu hút và giữ chân người dùng, giúp xây dựng thương hiệu cho Sound Entertainment.
    - Cạnh tranh với các nền tảng âm nhạc lớn: Đưa ra các tính năng riêng biệt, như khả năng tạo cộng đồng âm nhạc và chia sẻ âm nhạc, nhằm cạnh tranh với các dịch vụ như Spotify, YouTube, hay SoundCloud.
    - Thu thập dữ liệu người dùng: Giúp công ty thu thập dữ liệu hành vi người dùng để cải tiến sản phẩm và mở rộng các dịch vụ cá nhân hóa.
  + Với các nhà phát triển phần mềm:
    - Phát triển kỹ năng công nghệ: Dự án tạo cơ hội cho các nhà phát triển sử dụng các công nghệ hiện đại như Node.js, React.js, và MongoDB trong việc xây dựng một hệ thống lớn và phức tạp.
    - Học hỏi từ quy trình phát triển Agile-Scrum: Cung cấp cơ hội làm việc theo phương pháp Agile-Scrum, giúp nhóm học hỏi cách làm việc linh hoạt và hiệu quả trong các dự án phần mềm.
  + Với người kiểm thử phần mềm:
    - Cải thiện quy trình kiểm thử phần mềm: Kiểm thử các tính năng như đăng ký, đăng nhập, phát nhạc, chia sẻ âm nhạc, giúp đảm bảo rằng sản phẩm cuối cùng đáp ứng các yêu cầu về tính năng, hiệu suất và bảo mật.
* Mục tiêu:
  + Giao diện người dùng đơn giản và dễ sử dụng: Xây dựng một giao diện người dùng thân thiện và dễ sử dụng để người dùng có thể dễ dàng tải lên và chia sẻ bài hát, tạo playlist, tìm kiếm và khám phá các bài hát mới một cách thuận tiện.
  + Cung cấp công cụ tìm kiếm nâng cao: Tích hợp các công cụ tìm kiếm nâng cao để người dùng có thể tìm kiếm bài hát theo thể loại, nghệ sĩ hoặc lời bài hát một cách chính xác và hiệu quả.
  + Tính năng tương tác: Phát triển các tính năng giúp người dùng tương tác với nhau, bao gồm theo dõi và chia sẻ các bài hát yêu thích trên các nền tảng mạng xã hội khác, tạo ra sự kết nối và giao lưu trong cộng đồng.
  + Bảo mật thông tin người dùng: Tối ưu hóa tính năng bảo mật để đảm bảo rằng thông tin cá nhân của người dùng được bảo mật và an toàn trên nền tảng.
  + Phát triển tính năng bổ sung: Liên tục phát triển và cải tiến tính năng bổ sung như lời bài hát, thông tin về nghệ sĩ và ban nhạc, tạo playlist tự động và các tính năng khác để nâng cao trải nghiệm người dùng.
  + Tối ưu hiệu suất và độ tin cậy: Tối ưu hóa hiệu suất và độ tin cậy của phần mềm để đảm bảo rằng nó có thể xử lý lượng dữ liệu lớn và đáp ứng được yêu cầu của người dùng một cách mạnh mẽ và đáng tin cậy.

### Các quy trình nghiệp vụ (sáng)

P01: Quy trình quản lý tài khoản

* Khi người dùng truy cập hệ thống, họ có thể đăng ký tài khoản bằng cách nhập thông tin như tên, email, mật khẩu.
* Sau khi đăng ký, người dùng có thể đăng nhập vào hệ thống để sử dụng các chức năng.
* Người dùng có thể đổi mật khẩu khi cần thiết.
* Khi đăng nhập thành công, người dùng có thể xem danh sách tài khoản nếu có quyền quản lý.
* Người dùng có thể đăng xuất khỏi hệ thống khi không muốn sử dụng nữa.

P02: Quy trình quản lý nhạc

* Người dùng có thể nghe nhạc trực tuyến bằng cách chọn một bài hát từ danh sách có sẵn.
* Người dùng có thể tải nhạc xuống thiết bị để nghe offline.
* Khi người dùng đăng nhập vào hệ thống, họ có thể tìm kiếm bài hát theo tên bài hát, ca sĩ hoặc thể loại.
* Người dùng có thể xem danh sách các bài hát có trên hệ thống.

P03: Quy trình quản lý bình luận

* Khi nghe nhạc, người dùng có thể thêm bình luận vào bài hát để chia sẻ cảm nhận.
* Nếu người dùng muốn thay đổi nội dung, họ có thể chỉnh sửa bình luận của mình.
* Người dùng có thể xóa bình luận nếu không muốn hiển thị nữa.
* Quản trị viên có thể quản lý bình luận, bao gồm việc xem tất cả các bình luận và xóa bình luận vi phạm.

P04: Quy trình quản lý danh sách phát

* Người dùng có thể tạo danh sách phát để lưu trữ các bài hát yêu thích.
* Danh sách phát có thể được chia sẻ với bạn bè hoặc những người dùng khác.
* Người dùng có thể xem danh sách phát của mình hoặc của những người dùng khác nếu được chia sẻ.

P05: Quy trình quản lý nhạc của tôi

* Người dùng có thể tải nhạc lên hệ thống để chia sẻ với cộng đồng.
* Sau khi tải lên, người dùng có thể chỉnh sửa thông tin bài hát, bao gồm tên bài hát, ca sĩ, thể loại.
* Nếu người dùng không muốn giữ bài hát trên hệ thống, họ có thể xóa nhạc khỏi hệ thống.
* Mỗi bài hát được quản lý theo thông tin người dùng tải lên, giúp dễ dàng theo dõi và quản lý nội dung.

### SRS của dự án phần mềm (sáng)

[SRS của dự án Phát triển phần mềm mạng xã hội chia sẻ âm nhạc SoundShare cho Sound Entertainment theo Công nghệ phần mềm](https://docs.google.com/document/d/1Z8Iun5FqFSUYd4f4UtPqwlxoN1aqlvyc/edit)

**Mục đích tài liệu**

* Mục đích của tài liệu đặc tả yêu cầu phần mềm này là cung cấp một cái nhìn tổng quan, dễ hiểu về các yêu cầu, thành phần của dự án.
* Tài liệu giúp định nghĩa, phân tích và liệt kê các yêu cầu chức năng và phi chức năng của phần mềm, bao gồm các tính năng, giao diện, dữ liệu, bảo mật, hiệu suất, tương tác với hệ thống khác và yêu cầu kỹ thuật khác cần thiết để phát triển phần mềm quản lý bán hàng cho cửa hàng điện máy.
* Tài liệu này được cung cấp như một tài liệu tham khảo cho các thành viên trong đội ngũ phát triển phần mềm để hiểu rõ hơn về các yêu cầu cần thiết của dự án, đồng thời là một công cụ quản lý dự án để giám sát và đánh giá tiến độ, chất lượng của phần mềm trong quá trình phát triển.
* Tài liệu này được viết dựa theo chuẩn của Tài liệu đặc tả yêu cầu phần mềm (Software Requirements Specifications - SRS) được giải thích trong "IEEE Recommended Practice for Software Requirements Specifications" và " IEEE Guide for Developing System Requirements Specifications".

Với cấu trúc được chia làm ba phần:

* Phần 1: Cung cấp cái nhìn tổng quan về các thành phần của SRS.
* Phần 2: Mô tả tổng quan các nhân tố, ràng buộc, đặc điểm người dùng, môi trường thực thi tác động lên hệ thống và các yêu cầu của nó. Cung cấp thông tin chi tiết các yêu cầu chức năng, cung cấp cho các nhà phát triển phần mềm thông tin để phát triển phần mềm đáp ứng được các yêu cầu đó.
* Phần 3: Các yêu cầu phi chức năng.

**Phạm vi sử dụng**

* Tài liệu đặc tả yêu cầu phần mềm này được xây dựng nhằm phục vụ cho dự án Phát triển phần mềm mạng xã hội chia sẻ âm nhạc SoundShare  
  cho Sound Entertainment theo Công nghệ phần mềm.
* Trong dự án phát triển phần mềm mạng xã hội chia sẻ âm nhạc SoundShare  
  cho Sound Entertainment theo Công nghệ phần mềm thì những đối tượng có thể truy cập và sử dụng tài liệu SRS bao gồm:
* Nhóm phát triển dự án
* Chủ sở hữu dự án

**Mục đích sử dụng**

* Với việc Phát triển phần mềm mạng xã hội chia sẻ âm nhạc SoundShare cho Sound Entertainment theo Công nghệ phần mềm sẽ cung cấp cho người dùng một công cụ mạng xã hội giúp chia sẻ âm nhạc với nhau cũng như lưu trữ lại các bài hát, ca sĩ yêu thích đồng thời bày tỏ ý kiến cá nhân với các sản phẩm âm nhạc mang nhiều lợi ích cho việc giải trí của mọi người.

[(Xem Phụ lục A)](#bookmark=id.m1n9i9ju807y)

## Đánh giá các đề tài liên quan (thắng)

Đánh giá đề tài: Phát triển kiến trúc phần mềm cho dự án quản lý bán vé xem phim cho rạp chiếu phim BHD Star Vincom Center Phạm Ngọc Thạch

Môn học : Kiến trúc phần mềm

Giảng viên hướng dẫn: ThS. Lê Hữu Dũng

Nhóm thực hiện: G20

Nguồn báo cáo: [DHL-SA.Template.202411.G20](https://docs.google.com/document/d/1-Fn7teRi5D0f6-kFtaOx5b4TrcD5SrSf/edit)

Mục đích của đề tài là thiết kế và phát triển một kiến trúc phần mềm tối ưu cho hệ thống quản lý bán vé xem phim, đảm bảo khả năng vận hành hiệu quả, bảo mật cao và có thể mở rộng dễ dàng.

Hệ thống tập trung vào việc tự động hóa quy trình đặt vé, quản lý suất chiếu, hỗ trợ thanh toán trực tuyến và cung cấp trải nghiệm người dùng mượt mà.

Hệ thống này cần đảm bảo:

* Quản lý bán vé hiệu quả, cho phép đặt vé trực tuyến và tại quầy.
* Xử lý giao dịch nhanh chóng và bảo mật, hạn chế lỗi phát sinh trong quá trình đặt vé.
* Tối ưu hóa hiệu suất, đảm bảo khả năng chịu tải khi số lượng người dùng tăng đột biến.
* Mở rộng và bảo trì dễ dàng, giúp hệ thống phát triển linh hoạt theo nhu cầu rạp chiếu phim.

Đánh giá dựa theo các tiêu chí:

* Tính hiệu năng: Hệ thống đã đáp ứng được mục tiêu xử lý giao dịch nhanh dưới 2 giây và hỗ trợ nhiều người dùng đồng thời nhờ sử dụng Layered Architecture. Việc tối ưu hóa tài nguyên và caching giúp tăng tốc độ xử lý, tuy nhiên, cần tiếp tục kiểm tra hiệu suất trong điều kiện tải cao để đảm bảo tính ổn định lâu dài.
* Tính bảo mật: Hệ thống đã triển khai các biện pháp mã hóa dữ liệu, xác thực 2 lớp (2FA) và firewall chống DDoS, đảm bảo an toàn thông tin khách hàng. Điều này phù hợp với yêu cầu bảo mật đặt ra, nhưng cần thường xuyên kiểm tra và cập nhật các chính sách bảo mật để chống lại các mối đe dọa mới.
* Tính hữu dụng: Giao diện người dùng thân thiện, hỗ trợ thao tác đặt vé dễ dàng và đảm bảo thời gian phản hồi nhanh. Hệ thống đạt được mục tiêu cung cấp trải nghiệm mượt mà, tuy nhiên, cần tiếp tục cải thiện khả năng cá nhân hóa để tăng mức độ hài lòng của người dùng.
* Tính khả biến: Hệ thống có cấu trúc module giúp dễ dàng mở rộng tính năng như thêm phương thức thanh toán hoặc đặt đồ ăn. Điều này đáp ứng tốt yêu cầu linh hoạt của hệ thống và giúp giảm thời gian tích hợp các tính năng mới.
* Tính bảo trì: Hệ thống hỗ trợ phát hiện lỗi tự động, ghi log chi tiết và có cơ chế giám sát, đảm bảo ổn định. Điều này giúp hệ thống dễ dàng duy trì và sửa lỗi kịp thời, đúng với mục tiêu đặt ra.
* Tính khả chuyển: Việc triển khai hệ thống trên cloud và on-premise mà không ảnh hưởng đến hoạt động đáp ứng tốt yêu cầu linh hoạt về môi trường chạy.
* Tính đáng tin cậy: Hệ thống có khả năng phục hồi nhanh khi gặp sự cố, đảm bảo tính chính xác và toàn vẹn dữ liệu, giúp đạt được yêu cầu về độ ổn định.
* Tính kiểm thử: Việc hỗ trợ kiểm thử tự động và phân tách các thành phần giúp phát hiện lỗi sớm, nâng cao chất lượng hệ thống, hoàn toàn phù hợp với mục tiêu kiểm thử đặt ra.
* Tính sẵn sàng: Hệ thống có phương án dự phòng và load balancing để đảm bảo hoạt động liên tục, đạt được yêu cầu về tính sẵn sàng.

## Mục đích của đề tài (thắng)

Mục đích của đề tài là phát triển một phần mềm mạng xã hội chia sẻ âm nhạc SoundShare, cung cấp nền tảng linh hoạt để người dùng có thể dễ dàng tải lên, khám phá và tương tác với âm nhạc. Hệ thống hướng đến việc tối ưu hóa trải nghiệm người dùng, đảm bảo hiệu suất cao, tính bảo mật và khả năng mở rộng, đồng thời tạo ra môi trường kết nối và chia sẻ âm nhạc tiện lợi.

Phần mềm giúp người dùng có thể dễ dàng tải lên, khám phá, tương tác và chia sẻ âm nhạc một cách thuận tiện. Hệ thống được thiết kế nhằm cung cấp trải nghiệm người dùng tối ưu, đảm bảo tính bảo mật, khả năng mở rộng và hiệu suất cao.

## Mục tiêu của đề tài (trí)

- Mục tiêu chính của đề tài phát triển phần mềm mạng xã hội chia sẻ âm nhạc là tạo ra một nền tảng linh hoạt và thuận tiện cho người dùng chia sẻ và khám phá âm nhạc. Các mục tiêu cụ thể của dự án này được liệt kê dưới đây:

* **Giao diện người dùng đơn giản và dễ sử dụng:** Xây dựng một giao diện người dùng thân thiện và dễ sử dụng để người dùng có thể dễ dàng tải lên và chia sẻ bài hát, tạo playlist, tìm kiếm và khám phá các bài hát mới một cách thuận tiện.
* **Cung cấp công cụ tìm kiếm nâng cao:** Tích hợp các công cụ tìm kiếm nâng cao để người dùng có thể tìm kiếm bài hát theo thể loại, nghệ sĩ hoặc lời bài hát một cách chính xác và hiệu quả.
* **Tính năng tương tác:** Phát triển các tính năng giúp người dùng tương tác với nhau, bao gồm theo dõi và chia sẻ các bài hát yêu thích trên các nền tảng mạng xã hội khác, tạo ra sự kết nối và giao lưu trong cộng đồng.
* **Bảo mật thông tin người dùng:** Tối ưu hóa tính năng bảo mật để đảm bảo rằng thông tin cá nhân của người dùng được bảo mật và an toàn trên nền tảng.
* **Phát triển tính năng bổ sung:** Liên tục phát triển và cải tiến tính năng bổ sung như lời bài hát, thông tin về nghệ sĩ và ban nhạc, tạo playlist tự động và các tính năng khác để nâng cao trải nghiệm người dùng.
* **Tối ưu hiệu suất và độ tin cậy:** Tối ưu hóa hiệu suất và độ tin cậy của phần mềm để đảm bảo rằng nó có thể xử lý lượng dữ liệu lớn và đáp ứng được yêu cầu của người dùng một cách mạnh mẽ và đáng tin cậy.

1. **Kế hoạch và phân công thực hiện (trí)**

Kế hoạch :

| STT | Công việc | Người thực hiện | Thời gian bắt đầu | Thời gian kết thúc |
| --- | --- | --- | --- | --- |
| 1 | Tổng quan về đề tài | Nguyễn Đình Phi  Ngô Minh Phương  Lê Văn Sáng  Phạm Quang Thắng  Đồng Minh Trí | 20/02/2025 | 20/02/2025 |
| 2 | Phân tích yêu cầu | Nguyễn Đình Phi  Ngô Minh Phương  Lê Văn Sáng  Phạm Quang Thắng  Đồng Minh Trí | 21/02/2025 | 28/02/2025 |
| 3 | Thiết kế cấu trúc của phần mềm |  | | |
| Xác định cấu trúc cơ sở dữ liệu | Ngô Minh Phương  Lê Văn Sáng  Phạm Quang Thắng | 01/03/2025 | 14/03/2025 |
| Định nghĩa chức năng và tính năng | Ngô Minh Phương  Lê Văn Sáng  Phạm Quang Thắng |
| Thiết kế giao diện người dùng | Nguyễn Đình Phi  Ngô Minh Phương  Đồng Minh Trí |
| Định nghĩa các luồng công việc và sơ đồ tương tác giữa các thành phần trong hệ thống. | Nguyễn Đình Phi  Phạm Quang Thắng  Đồng Minh Trí |
| Thiết kế kiểm soát | Đồng Minh Trí  Nguyễn Đình Phi  Lê Văn Sáng |
| 4 | Thiết kế phần mềm | Nguyễn Đình Phi  Ngô Minh Phương  Lê Văn Sáng  Phạm Quang Thắng  Đồng Minh Trí | 14/03/2025 | 21/03/2025 |
| 5 | Hoàn thiện | Nguyễn Đình Phi  Ngô Minh Phương  Lê Văn Sáng  Phạm Quang Thắng  Đồng Minh Trí | 21/03/2025 | 28/03/2025 |

Phân công :

| **Nhiệm vụ** | Nguyễn Đình Phi | Ngô Minh Phương | Lê Văn Sáng | Phạm Quang Thắng | Đồng Minh Trí |
| --- | --- | --- | --- | --- | --- |
| **Kiến trúc tổng thể hệ thống** |  | S | S | P |  |
| **Thiết kế giao diện** | S | P |  |  | S |
| **Thiết kế CSDL** |  | S | P | S |  |
| **Thiết kế sơ đồ luồng dữ liệu** | P |  |  | S | S |
| **Thiết kế kiểm soát** | S |  | S |  | P |
| **Thiết kế** | P | P | P | P | P |

## 

## 

# CHƯƠNG 2: PHÂN TÍCH YÊU CẦU

(mỗi mục lớn trong chương phải ghi rõ tên người thực hiện)

## Đánh giá SRS của dự án đã có (Phương, Trí)

Để đánh giá một tài liệu Đặc tả Yêu cầu Phần mềm (SRS - Software Requirements Specification) theo các tiêu chuẩn ISO/IEEE, cụ thể là IEEE 830-1998 (Recommended Practice for Software Requirements Specifications), có một số tiêu chí quan trọng được khuyến nghị:

* TC01: Tính Chính xác (Correctness)
* TC02: Tính Rõ ràng (Unambiguousness)
* TC03: Tính Hoàn chỉnh (Completeness)
* TC04: Tính Nhất quán (Consistency)
* TC05: Tính Xếp hạng theo Mức độ Quan trọng (Ranked for Importance/Stability)
* TC06: Tính Kiểm chứng được (Verifiability)
* TC07: Tính Có thể Sửa đổi (Modifiability)
* TC08: Tính Truy vết được (Traceability)
* TC09: Tính Khả thi (Feasibility)
* TC10: Tính Phù hợp (Appropriateness)

| **STT** | **Tiêu chí** | **Đã đạt được** | **Chưa đạt được** | **Nhận xét** |
| --- | --- | --- | --- | --- |
| 1 | TC01: Tính Chính xác (Correctness) | Tài liệu liệt kê rõ ràng các nhu cầu của người dùng (2.1) như chia sẻ nhạc, nghe nhạc, tải nhạc, quản lý danh sách phát, v.v., và các chức năng (3.1) được mô tả chi tiết, phù hợp với mục tiêu của một mạng xã hội âm nhạc. Các yêu cầu này dường như phản ánh đúng nhu cầu giải trí của người dùng mà Sound Entertainment hướng tới. | tài liệu có một số nhầm lẫn trong phần "Giới thiệu" khi đề cập đến "phần mềm quản lý bán hàng cho cửa hàng điện máy" (1.1), điều này không liên quan đến dự án SoundShare. Điều này cho thấy thiếu sự kiểm tra kỹ lưỡng, làm giảm tính chính xác. | Tính chính xác đạt mức tương đối, cần chỉnh sửa các lỗi không liên quan để đảm bảo phản ánh đúng dự án. |
| 2 | TC02: Tính Rõ ràng (Unambiguousness) | Ngôn ngữ sử dụng trong tài liệu khá dễ hiểu. Các use case (3.3) mô tả chi tiết luồng sự kiện chính với hành động và phản ứng hệ thống, giúp giảm thiểu sự mơ hồ. | Một số yêu cầu thiếu định lượng hoặc tiêu chí cụ thể, ví dụ: "Khả năng chịu tải của hệ thống khi có nhiều người truy cập phải đảm bảo hoạt động bình thường" (phi chức năng, mục 2) không nêu rõ "nhiều" là bao nhiêu người, hay "bình thường" được đo lường thế nào. | Tính rõ ràng đạt mức tốt, nhưng cần bổ sung các chỉ số định lượng để tránh hiểu nhầm. |
| 3 | TC03: Tính Hoàn chỉnh (Completeness) | Tài liệu bao gồm 18 yêu cầu chức năng (3.1) và 7 yêu cầu phi chức năng (phần 4), cùng với mô tả use case chi tiết, sơ đồ phân rã chức năng (dù chưa hoàn thiện), và các ràng buộc như tương thích đa thiết bị, bảo mật. Điều này cho thấy nỗ lực liệt kê đầy đủ các khía cạnh của hệ thống. | Một số phần chưa được hoàn thiện, ví dụ: "Sơ đồ phân rã chức năng" và "Sơ đồ luồng dữ liệu (DFD)" chỉ có ký hiệu mà chưa có nội dung cụ thể. Ngoài ra, không có mô tả về giao diện người dùng (UI) hoặc yêu cầu phần cứng cụ thể, điều này có thể gây khó khăn cho nhà phát triển. | Tính hoàn chỉnh ở mức tương đối, cần bổ sung các phần còn thiếu để đạt yêu cầu đầy đủ. |
| 4 | TC04: Tính Nhất quán (Consistency) | Các yêu cầu chức năng và phi chức năng không xuất hiện mâu thuẫn. Các thuật ngữ như "Quản lý tài khoản", "Quản lý nhạc" được sử dụng nhất quán trong các phần khác nhau. Bảng định nghĩa và viết tắt (1.4) hỗ trợ duy trì tính đồng bộ. | Có sự không nhất quán trong cách trình bày use case: một số use case lặp lại nội dung không cần thiết (ví dụ: UC01\_QLTK lặp 3 lần giống hệt nhau), và phần "Giới thiệu" lại nhắc đến "cửa hàng điện máy" không liên quan đến SoundShare. | Tính nhất quán đạt đạt mức tương đối, cần loại bỏ các nội dung trùng lặp và không liên quan. |
| 5 | TC05: Tính Xếp hạng theo Mức độ Quan trọng (Ranked for Importance/Stability) |  | Tài liệu không cung cấp bất kỳ thông tin nào về mức độ ưu tiên hoặc tính ổn định của các yêu cầu. Ví dụ, không rõ chức năng nào (đăng ký, nghe nhạc, tải nhạc) là quan trọng nhất hoặc có thể thay đổi trong tương lai. | Tính xếp hạng ở mức không đạt, cần bổ sung phân loại ưu tiên để hỗ trợ quản lý dự án. |
| 6 | TC06: Tính Kiểm chứng được (Verifiability) | Một số yêu cầu phi chức năng có tiêu chí đo lường, ví dụ: "Thời gian tối đa để xử lý yêu cầu là 10 giây" (mục 2, tính hiệu năng) hoặc "phục hồi 100% trong 24h" (mục 1, tính tin cậy). Các use case cũng mô tả kết quả đầu ra cụ thể (thành công/thất bại), hỗ trợ kiểm thử. | Nhiều yêu cầu khác thiếu tiêu chí kiểm chứng, ví dụ: "Giao diện hệ thống phải dễ nhìn, dễ sử dụng" (mục 3, tính hữu dụng) không có cách đo lường cụ thể, hoặc "Khả năng chịu tải phải đảm bảo hoạt động bình thường" không định lượng số lượng người dùng. | Tính kiểm chứng đạt mức tương đối, cần bổ sung tiêu chí đo lường cho các yêu cầu còn lại. |
| 7 | TC07: Tính Có thể Sửa đổi (Modifiability) | Tài liệu được tổ chức theo bố cục rõ ràng với các mục được đánh số (1.1, 2.1, 3.1, v.v.) và sử dụng bảng biểu, tiêu đề phụ, giúp dễ dàng chỉnh sửa từng phần mà không làm rối toàn bộ tài liệu. | Một số phần lặp lại không cần thiết (ví dụ: use case UC01\_QLTK) có thể gây khó khăn khi chỉnh sửa nếu không được tinh gọn. | Tính có thể sửa đổi đạt mức tốt, nhưng cần tối ưu hóa để giảm trùng lặp. |
| 8 | TC08: Tính Truy vết được (Traceability) | Các yêu cầu chức năng được đánh số rõ ràng (01-18) và gom nhóm theo chức năng mức 1, mức 2 (3.1.2), hỗ trợ truy vết từ yêu cầu người dùng (2.1) đến chức năng cụ thể. Use case cũng có mã (UC01, UC02, v.v.), giúp liên kết với thiết kế và kiểm thử sau này. | Không có thông tin cụ thể về nguồn gốc yêu cầu (ví dụ: từ khách hàng, khảo sát, hay nhóm phát triển), làm giảm khả năng truy ngược hoàn toàn. | Tính truy vết đạt mức tốt, nhưng cần bổ sung nguồn gốc yêu cầu để hoàn thiện. |
| 9 | TC09: Tính Khả thi (Feasibility) | Các chức năng như đăng ký, nghe nhạc, tải nhạc, quản lý danh sách phát đều nằm trong khả năng của công nghệ hiện tại (dùng mô hình MVC, Agile). Yêu cầu tương thích đa thiết bị cũng khả thi với các framework phổ biến. | Một số yêu cầu phi chức năng như "phục hồi 100% trong 24h" hoặc "xử lý yêu cầu trong 10 giây" có thể khó đảm bảo nếu không có thông tin về hạ tầng phần cứng hoặc đội ngũ kỹ thuật. Tài liệu không đề cập đến ràng buộc tài nguyên, làm khó đánh giá tính khả thi đầy đủ. | Tính khả thi ở mức chưa thể khẳng định, cần bổ sung thông tin về tài nguyên và hạ tầng. |
| 10 | TC10: Tính Phù hợp (Appropriateness) | Tài liệu cung cấp thông tin chi tiết về chức năng, use case, và yêu cầu phi chức năng, phù hợp cho nhà phát triển và tester. Ngôn ngữ tiếng Việt dễ hiểu, phù hợp với khách hàng và nhóm dự án tại Việt Nam. | Thiếu mô hình giao diện hoặc ví dụ trực quan có thể gây khó khăn cho khách hàng không chuyên về kỹ thuật trong việc hình dung sản phẩm. | Tính phù hợp đạt mức tốt, nhưng nên bổ sung hình ảnh hoặc mockup giao diện để tăng tính trực quan. |

## Phát biểu lại (đạt chất lượng) các Yêu cầu (FR, NFR) của dự án (Thắng, Sáng Phi)

2.2.1. Phát biểu lại các Yêu cầu chức năng (Thắng)

| **Đối tượng** | **Yêu cầu chức năng** | **Mô tả** | **Đánh giá** |
| --- | --- | --- | --- |
| Quản lý tài khoản | (01) Đăng ký | Người dùng cung cấp thông tin như email, số điện thoại hoặc tài khoản mạng xã hội để đăng ký và bắt đầu sử dụng hệ thống. | Có xác thực thông tin, kiểm tra đầu vào.  Bổ sung kiểm tra email/số điện thoại trùng lặp trước khi gửi dữ liệu. |
| (02) Đăng nhập | Người dùng nhập tên đăng nhập và mật khẩu để vào hệ thống. | Kiểm tra thông tin hợp lệ, xử lý lỗi.  Cần bổ sung xác thực 2FA để tăng bảo mật. |
| (03) Đăng xuất | Người dùng chọn đăng xuất để kết thúc phiên sử dụng. | Đăng xuất an toàn.  Bổ sung xác nhận trước khi đăng xuất để tránh thao tác nhầm. |
| (4) Đổi mật khẩu | Người dùng nhập mật khẩu cũ và mật khẩu mới để thay đổi. | Cần có các quy tắc bảo mật như xác nhận bằng OTP/email trước khi đổi mật khẩu. |
| (5) Xem danh sách tài khoản | Hệ thống hiển thị thông tin các tài khoản đã đăng ký. | Chỉ nên cấp quyền này cho admin.  Bổ sung phân quyền chặt chẽ. |
| Quản lý nhạc | (6) Nghe nhạc | Hệ thống cung cấp giao diện điều khiển với các chức năng như phát, tạm dừng, chuyển bài, lặp lại, trộn ngẫu nhiên và điều chỉnh âm lượng. | Hỗ trợ phát nhạc trực tuyến.  Xử lý lỗi khi bài hát không phát được, hiển thị thông báo cụ thể. |
| (7) Tải nhạc xuống | Người dùng chọn bài hát để tải xuống ở định dạng .mp3 để nghe ngoại tuyến. | Hỗ trợ tải nhạc.  Cần kiểm soát bản quyền nhạc, tránh sao chép trái phép. |
| (8) Tìm kiếm nhạc | Người dùng nhập từ khóa hoặc sử dụng bộ lọc để tìm bài hát, nghệ sĩ, album; kết quả hiển thị rõ ràng. | Tìm kiếm nhanh, có phân loại.  Nên có gợi ý tìm kiếm thông minh dựa trên lịch sử nghe. |
| (9) Xem danh sách các bài nhạc | Hệ thống hiển thị các bài hát mới nhất hoặc phổ biến nhất. | Cung cấp danh sách bài hát.  Bổ sung sắp xếp, lọc theo thể loại, nghệ sĩ. |
| Quản lý bình luận | (10) Thêm bình luận | Người dùng để lại ý kiến dưới bài hát hoặc danh sách phát. | Hỗ trợ tương tác.  Cần kiểm tra nội dung để tránh bình luận không phù hợp. |
| (11) Sửa bình luận | Người dùng thay đổi nội dung bình luận đã đăng. | Cho phép sửa lỗi.  Cần lưu lịch sử chỉnh sửa để tránh thay đổi nội dung tiêu cực. |
| (12) Xóa bình luận | Người dùng xóa bình luận của mình; quản trị viên xóa bình luận vi phạm. | Có xác nhận trước khi xóa.  Nên bổ sung tính năng báo cáo bình luận xấu. |
| (13) Xem tất cả các bình luận | Hệ thống hiển thị bình luận theo bài hát hoặc thời gian. | Cung cấp lịch sử bình luận.  Nên có bộ lọc theo bài hát, người dùng, thời gian. |
| Quản lý danh sách phát | (14) Tạo danh sách phát nhạc | Người dùng thêm và sắp xếp bài hát vào danh sách phát cá nhân. | Cho phép tùy chỉnh danh sách.  Nên hỗ trợ kéo thả bài hát để sắp xếp. |
| (15) Chia sẻ danh sách phát nhạc | Người dùng tạo liên kết để chia sẻ danh sách phát với người khác. | Hỗ trợ chia sẻ.  Cần có quyền riêng tư (công khai, chỉ bạn bè, riêng tư). |
| Quản lý nhạc của tôi | (16) Tải nhạc lên | Người dùng thêm bài hát kèm thông tin mô tả. | Đảm bảo người dùng có thể tải nhạc lên.  Cần kiểm tra định dạng file hợp lệ (.mp3, .wav, .flac). |
| (17) Chỉnh sửa thông tin nhạc | Người dùng thay đổi tên bài hát hoặc mô tả sau khi tải lên. | Hỗ trợ cập nhật thông tin.  Cần lưu lịch sử chỉnh sửa để theo dõi thay đổi. |
| (18) Xoá nhạc | Người dùng loại bỏ bài hát đã tải lên khỏi hệ thống. | Có xác nhận trước khi xóa.  Nên có chế độ khôi phục trong thời gian nhất định để tránh xóa nhầm. |

2.2.2. Phát biểu lại các Yêu cầu phi chức năng của dự án (NFR)

NFR1 : Yêu cầu về độ tin cậy

* + **Định nghĩa(ISO) :** mức độ mà một hệ thống, sản phẩm hoặc thành phần thực hiện các chức năng cụ thể trong những điều kiện cụ thể trong một khoảng thời gian cụ thể
  + **Thành phần :** Theo tiêu chuẩn ISO, tính tin cậy thường được phân tích theo các thành phần sau :
    - Độ trưởng thành(**maturity**) : Mức độ mà một hệ thống, sản phẩm hoặc thành phần đáp ứng nhu cầu về độ tin cậy trong quá trình hoạt động bình thường
    - Tính sẵn sàng(**availability**) : Mức độ mà một hệ thống, sản phẩm hoặc thành phần hoạt động và có thể truy cập khi cần sử dụng
    - Khả năng chịu lỗi(**fault tolerance**) : Mức độ mà một hệ thống, sản phẩm hoặc thành phần hoạt động như dự định bất chấp sự hiện diện của lỗi phần cứng hoặc phần mềm
    - Khả năng phục hồi(**recoverability**) : Mức độ mà, trong trường hợp xảy ra gián đoạn hoặc hỏng hóc, một sản phẩm hoặc hệ thống có thể khôi phục dữ liệu bị ảnh hưởng trực tiếp và tái lập trạng thái mong muốn của hệ thống
  + **Nguồn :** [**ISO/IEC 25010:2011**](https://www.iso.org/obp/ui/?fbclid=IwY2xjawGsDo5leHRuA2FlbQIxMAABHdqLFb-EJo5aWXsetspU5ipXAbiL3swyCURDQa43vbVf8oRkxnvjcp3Mcg_aem_0fAED_FQhF3N8gUdC10KvQ#iso:std:iso-iec:25010:ed-1:v1:en)
  + **Tính tin cậy (NFR cũ của dự án) :**
    - Thông tin hệ thống đưa ra phải có tính xác thực được người dùng tin cậy
    - Khi xảy ra các sự cố làm ngừng vận hành hệ thống, hệ thống phải đảm bảo phục hồi 100% trong vòng 24h
  + **Phát biểu lại:** 
    - Hệ thống SoundShare phải đảm bảo cung cấp thông tin chính xác, xác thực và đáng tin cậy cho người dùng. Trong trường hợp xảy ra sự cố (do lỗi phần cứng, lỗi phần mềm, tấn công từ bên ngoài, v.v…), hệ thống cần kích hoạt cơ chế phục hồi tự động, khôi phục đầy đủ các chức năng và dữ liệu, đảm bảo phục hồi 100% trong vòng 24 giờ.
  + **Phân tích theo kịch bản 6 thành phần Bass :**
    - Nguồn kích hoạt (Stimulus Source): Các tác nhân gây ra sự cố, chẳng hạn như lỗi phần cứng, lỗi phần mềm, tấn công mạng, hoặc các sự cố môi trường (ví dụ: mất điện, quá tải hệ thống).
    - Kích hoạt (Stimulus): Sự kiện bất ngờ làm gián đoạn hoạt động của hệ thống, dẫn đến việc không cung cấp được thông tin đáng tin cậy cho người dùng.
    - Ngữ cảnh (Environment): Hệ thống hoạt động trong môi trường vận hành bình thường cũng như trong các điều kiện có khả năng gặp sự cố (ví dụ: môi trường có tải cao, điều kiện vận hành không ổn định do yếu tố bên ngoài).
    - Hệ thống (Artifact): Toàn bộ hệ thống SoundShare, bao gồm các thành phần phần mềm, cơ sở dữ liệu, máy chủ và các module phục hồi chức năng.
    - Phản ứng (Response): Khi nhận được kích hoạt (sự cố), hệ thống sẽ tự động kích hoạt cơ chế phục hồi: khôi phục dữ liệu bị ảnh hưởng, tái lập trạng thái dịch vụ và đảm bảo tiếp tục cung cấp thông tin chính xác cho người dùng.
    - Đo lường phản ứng (Response Measure):
      * Thời gian phục hồi: Hệ thống phải khôi phục 100% các chức năng và dữ liệu trong vòng 24 giờ.
      * Mức độ phục hồi: Đảm bảo rằng sau quá trình phục hồi, toàn bộ các chức năng, thông tin và dữ liệu của hệ thống được tái lập hoàn toàn như trạng thái ban đầu, không mất mát hoặc sai lệch.

NFR 2 : Yêu cầu về hiệu năng

* + **Định nghĩa(ISO)** : Hiệu năng của một hệ thống, sản phẩm, hoặc thành phần được đánh giá dựa trên hiệu quả hoạt động liên quan đến lượng tài nguyên đã sử dụng trong các điều kiện nhất định
  + **Thành phần** : Theo tiêu chuẩn ISO, yêu cầu về hiệu năng thường được phân tích theo các thành phần sau :
    - Hành vi thời gian(**time behaviour**): mức độ mà các thời gian phản hồi và xử lý và tốc độ thông lượng của sản phẩm hoặc hệ thống, khi thực hiện chức năng của nó, đáp ứng các yêu cầu
    - Sử dụng tài nguyên(**resource utilization**) : mức độ mà các loại và số lượng tài nguyên được sản phẩm hoặc hệ thống sử dụng, khi thực hiện chức năng của nó, đáp ứng các yêu cầu
    - khả năng(**capacity**) : mức độ mà các giới hạn tối đa của các tham số của sản phẩm hoặc hệ thống đáp ứng các yêu cầu
  + **Nguồn :** [**ISO/IEC 25010:2011**](https://www.iso.org/obp/ui/?fbclid=IwY2xjawGsDo5leHRuA2FlbQIxMAABHdqLFb-EJo5aWXsetspU5ipXAbiL3swyCURDQa43vbVf8oRkxnvjcp3Mcg_aem_0fAED_FQhF3N8gUdC10KvQ#iso:std:iso-iec:25010:ed-1:v1:en)
  + **Tính hiệu năng (NFR cũ) :**
    - Khả năng chịu tải của hệ thống khi có nhiều người truy cập phải đảm bảo hoạt động bình thường
    - Thời gian tối đa cho phép để xử lý yêu cầu từ người dùng và trả về kết quả tìm kiếm trong hệ thống là 10 giây tính từ lúc người dùng gửi yêu cầu (dựa vào đường truyền mạng, mức độ ổn định của CPU)
  + **Phát biểu lại :** 
    - Thời gian phản hồi khi tìm kiếm nội dung âm nhạc:
* Hệ thống phải hiển thị kết quả tìm kiếm bài hát, nghệ sĩ, album, hoặc playlist trong vòng 3 giây cho 90% lượt tìm kiếm của người dùng khi sử dụng trong điều kiện bình thường.
* Thời gian tải trang giao diện người dùng:
* Trang chủ, trang cá nhân và trang chia sẻ nội dung của người dùng phải được tải và hiển thị đầy đủ trong vòng 3 giây, đảm bảo trải nghiệm mượt mà khi truy cập.
* Thời gian xử lý và đăng tải nội dung:
* Khi người dùng tải lên nội dung (như bài hát, video giới thiệu, hình ảnh bìa…), hệ thống phải hoàn tất quá trình xử lý và hiển thị thông báo thành công trong vòng 5 giây, kể cả khi số lượng người dùng đồng thời lớn.
* Cập nhật theo thời gian thực các tương tác:
* Hệ thống phải cập nhật và hiển thị số lượt thích, bình luận, chia sẻ hay thông báo tin nhắn (như yêu cầu kết bạn, nhắn tin) trong vòng 5 giây kể từ khi có giao dịch hoặc tương tác mới phát sinh.
* Thời gian xử lý và lưu trữ dữ liệu lịch sử hoạt động:
* Trong quá trình sao lưu và xử lý dữ liệu lịch sử hoạt động của người dùng (bao gồm bài đăng, tương tác, tin nhắn, …), hệ thống phải hoàn thành mỗi tác vụ trong vòng 30 phút mà không ảnh hưởng đến hiệu suất của các chức năng chính phục vụ người dùng.
* Tối ưu hóa sử dụng tài nguyên:
* Hệ thống phải đảm bảo việc sử dụng tài nguyên (CPU, bộ nhớ, băng thông mạng) được tối ưu hóa, hỗ trợ khả năng mở rộng khi số lượng người dùng và khối lượng dữ liệu tăng cao, đảm bảo các yêu cầu thời gian phản hồi được duy trì trong mọi điều kiện vận hành.
  + **Phân tích theo kịch bản 6 thành phần Bass :**
    - Nguồn kích hoạt (Stimulus Source):
      * Người dùng: Các cá nhân truy cập SoundShare để tìm kiếm, chia sẻ hoặc tương tác với nội dung âm nhạc.
      * Sự kiện tăng tải: Các chiến dịch quảng bá, ra mắt tính năng mới hoặc các sự kiện âm nhạc đặc biệt gây ra lượng truy cập tăng đột biến.
    - Kích hoạt (Stimulus):
      * Yêu cầu tìm kiếm: Người dùng gửi yêu cầu tìm kiếm bài hát, nghệ sĩ, album hay playlist.
      * Yêu cầu đăng tải nội dung: Người dùng tải lên nội dung âm nhạc hoặc video giới thiệu.
      * Tương tác thời gian thực: Người dùng thực hiện các hành động như thích, bình luận, chia sẻ hay gửi tin nhắn.
      * Yêu cầu tải trang: Khi người dùng truy cập vào trang chủ, trang cá nhân hay các trang nội dung chia sẻ.
    - Ngữ cảnh (Environment):
      * Điều kiện sử dụng bình thường: Hệ thống hoạt động trong môi trường mạng ổn định với lưu lượng truy cập tiêu chuẩn.
      * Điều kiện tải cao: Hệ thống đối mặt với đột biến số lượng truy cập do các sự kiện đặc biệt, ra mắt tính năng mới hoặc chiến dịch quảng bá, đồng thời có thể chịu ảnh hưởng từ biến động về chất lượng đường truyền hoặc tài nguyên hệ thống.
    - Hệ thống (Artifact):
      * Hệ thống SoundShare: Bao gồm giao diện người dùng (web và/hoặc mobile), máy chủ ứng dụng, cơ sở dữ liệu lưu trữ thông tin âm nhạc, hồ sơ người dùng, dữ liệu tương tác và các module hỗ trợ chức năng tìm kiếm, đăng tải, cập nhật thời gian thực và sao lưu dữ liệu.
      * Cơ chế tối ưu và mở rộng: Các công nghệ hỗ trợ như caching, load balancing và auto-scaling được tích hợp để đảm bảo hệ thống duy trì hiệu suất ổn định khi đối mặt với tải cao.
    - Phản ứng (Response):
      * Xử lý yêu cầu tìm kiếm: Hệ thống truy xuất dữ liệu âm nhạc và trả về kết quả tìm kiếm trong vòng 3 giây đối với 90% lượt truy vấn.
      * Xử lý đăng tải nội dung: Khi nhận yêu cầu đăng tải, hệ thống xử lý dữ liệu và xác nhận thành công trong vòng 5 giây, ngay cả khi số lượng giao dịch tăng đột biến.
      * Tải trang giao diện: Trang chủ, trang cá nhân và các trang nội dung được tải đầy đủ và hiển thị trong vòng 3 giây, đảm bảo giao diện trực quan và nhanh nhạy.
      * Cập nhật tương tác theo thời gian thực: Sau mỗi tương tác (thích, bình luận, chia sẻ, tin nhắn), hệ thống cập nhật thông tin trên giao diện người dùng trong vòng 5 giây.
      * Xử lý dữ liệu lịch sử: Các tác vụ sao lưu và xử lý dữ liệu lịch sử hoạt động được thực hiện trong vòng 30 phút mà không làm gián đoạn các tính năng thời gian thực phục vụ người dùng.
    - Đo lường phản ứng (Response Measure):
      * Thời gian phản hồi tìm kiếm: Đo lường thời gian từ khi nhận yêu cầu tìm kiếm đến khi hiển thị kết quả, với mục tiêu 90% lượt tìm kiếm hoàn thành dưới 3 giây.
      * Thời gian tải trang: Thời gian tải và hiển thị trang chủ, trang cá nhân và trang chia sẻ phải luôn dưới 3 giây.
      * Thời gian xử lý đăng tải: Đo lường thời gian từ khi người dùng nhấn “Đăng tải” đến khi nhận thông báo thành công, đảm bảo không vượt quá 5 giây.
      * Thời gian cập nhật tương tác: Thời gian cập nhật số liệu tương tác (lượt thích, bình luận, chia sẻ) phải dưới 5 giây sau mỗi giao dịch tương tác mới.
      * Thời gian sao lưu và xử lý dữ liệu: Mỗi tác vụ liên quan đến sao lưu và xử lý dữ liệu lịch sử phải hoàn thành trong vòng 30 phút, đồng thời không ảnh hưởng đến hiệu suất của các chức năng chính.

NFR 3 : Yêu cầu về tính bảo mật:

* + **Định nghĩa (ISO):** là mức độ mà một sản phẩm hoặc hệ thống bảo vệ thông tin và dữ liệu để người hoặc các sản phẩm hoặc hệ thống khác có mức độ truy cập dữ liệu phù hợp với loại và cấp độ ủy quyền của họ
  + **Thành phần:** Theo tiêu chuẩn ISO, tính bảo mật thường được phân tích dựa trên các thành phần sau:
    - * Bảo mật thông tin (**Confidentiality**) : Mức độ mà một sản phẩm hoặc hệ thống đảm bảo rằng dữ liệu chỉ có thể truy cập bởi những người có quyền truy cập hợp lệ
      * Tính toàn vẹn (**Integrity**) : Mức độ mà một hệ thống, sản phẩm hoặc thành phần ngăn chặn truy cập trái phép hoặc thay đổi trái phép đối với các chương trình máy tính hoặc dữ liệu.
      * Không chối bỏ (**Non-repudiation) :** Mức độ mà các hành động hoặc sự kiện có thể được chứng minh là đã xảy ra, để các sự kiện hoặc hành động đó không thể bị phủ nhận sau này.
      * Trách nhiệm **(Accountability)** : Mức độ mà các hành động của một thực thể có thể được truy xuất một cách duy nhất tới thực thể đó.
      * Tính xác thực (**Authenticity**) : Mức độ mà danh tính của một đối tượng hoặc tài nguyên có thể được chứng minh là danh tính mà nó tuyên bố.
    - **Nguồn :** [**ISO/IEC 25010:2011**](https://www.iso.org/obp/ui/?fbclid=IwY2xjawGsDo5leHRuA2FlbQIxMAABHdqLFb-EJo5aWXsetspU5ipXAbiL3swyCURDQa43vbVf8oRkxnvjcp3Mcg_aem_0fAED_FQhF3N8gUdC10KvQ#iso:std:iso-iec:25010:ed-1:v1:en)
* **Tính bảo mật (NFR cũ của dự án):**
  + - Mã hóa dữ liệu quan trọng: Dữ liệu nhạy cảm như thông tin cá nhân và thông tin thanh toán phải được mã hóa trước khi lưu trữ vào cơ sở dữ liệu.
    - Kiểm soát quyền truy cập: Muốn truy cập các chức năng ngoài quyền của người dùng thông thường cần có sự đồng ý và ủy quyền của quản trị viên (có văn bản giấy tờ xác nhận).
  + **Phát biểu lại :** Hệ thống SoundShare phải đảm bảo bảo vệ toàn diện thông tin và dữ liệu của người dùng, cũng như các giao dịch và nội dung được chia sẻ trên nền tảng. Điều này bao gồm việc ngăn chặn truy cập trái phép, bảo vệ tính toàn vẹn của dữ liệu và đảm bảo rằng mọi hành động trên hệ thống đều có thể được truy xuất rõ ràng. Cụ thể:
    - Bảo mật thông tin (Confidentiality):
      * Mã hóa dữ liệu: Tất cả dữ liệu nhạy cảm như thông tin cá nhân, dữ liệu tài khoản và thông tin thanh toán phải được mã hóa trước khi lưu trữ vào cơ sở dữ liệu.
      * Kiểm soát truy cập: Chỉ những người dùng có quyền truy cập hợp lệ mới được phép xem và thao tác với dữ liệu nhạy cảm. Các chức năng vượt quá quyền thông thường của người dùng phải được truy cập sau khi có sự xác nhận và ủy quyền của quản trị viên (có bằng chứng văn bản).
    - Tính toàn vẹn (Integrity):
      * Bảo vệ dữ liệu: Hệ thống phải đảm bảo rằng dữ liệu và nội dung không bị thay đổi hoặc làm giả trong quá trình lưu trữ, truyền tải và xử lý.
      * Cơ chế kiểm tra: Áp dụng các biện pháp kiểm tra và xác minh (ví dụ: checksum, digital signatures) để phát hiện và ngăn chặn những thay đổi trái phép.
    - Không chối bỏ (Non-repudiation):
      * Ghi nhận hành động: Mọi giao dịch và tương tác của người dùng trên hệ thống cần được ghi lại một cách đầy đủ, nhằm đảm bảo rằng các bên liên quan không thể phủ nhận việc đã thực hiện hành động đó sau này.
    - Trách nhiệm (Accountability):
      * Theo dõi và truy xuất: Hệ thống cần có cơ chế giám sát và ghi log chi tiết các hành động của người dùng, cho phép truy xuất duy nhất và rõ ràng đến từng thực thể thực hiện hành động.
      * Quy trình kiểm duyệt: Việc truy cập vào các chức năng đặc biệt phải được quản lý nghiêm ngặt với sự phê duyệt của quản trị viên, nhằm đảm bảo mọi hành động đều minh bạch và có thể kiểm tra được.
    - Tính xác thực (Authenticity):
      * Xác minh danh tính: Các cơ chế xác thực như mật khẩu, xác thực đa yếu tố (MFA) và các phương pháp xác thực khác cần được triển khai để đảm bảo rằng danh tính của người dùng và các thành phần hệ thống được xác nhận chính xác.
      * Giao tiếp bảo mật: Tất cả các tương tác giữa người dùng và máy chủ, cũng như giao tiếp với các hệ thống bên ngoài, phải được bảo vệ qua các giao thức mã hóa (ví dụ: HTTPS) để đảm bảo tính xác thực và an toàn của dữ liệu.
  + **Phân tích theo kịch bản 6 thành phần Bass :** 
    - Nguồn kích hoạt (Stimulus Source):
* Người dùng không hợp lệ hoặc kẻ tấn công: Bao gồm các cá nhân không có quyền truy cập hợp lệ, hacker, hoặc người dùng nội bộ cố ý sử dụng sai chức năng.
* Hệ thống bên ngoài: Các ứng dụng, dịch vụ bên thứ ba có thể cố gắng truy cập dữ liệu mà không được phép.
* Kích hoạt (Stimulus):
* Yêu cầu truy cập trái phép: Ví dụ, một kẻ tấn công cố gắng truy cập vào dữ liệu nhạy cảm, thay đổi dữ liệu hoặc nghe lén thông tin trao đổi giữa người dùng và hệ thống.
* Thao tác bất hợp pháp: Cố gắng gửi yêu cầu giả mạo, thay đổi dữ liệu thông qua các giao diện không được bảo vệ hoặc vượt quá phạm vi quyền hạn của người dùng.
* Ngữ cảnh (Environment):
* Vận hành trong môi trường Internet mở: Hệ thống SoundShare hoạt động trên nền tảng web và/hoặc mobile, nơi mà thông tin được truyền qua các kênh mạng có thể bị đánh chặn hoặc giả mạo nếu không được bảo vệ.
* Điều kiện có rủi ro bảo mật: Khi có lưu lượng truy cập từ nguồn không xác định, trong các sự kiện quảng bá hay khi có hoạt động bất thường trên hệ thống, làm tăng nguy cơ tấn công.
* Hệ thống (Artifact):
* Hệ thống SoundShare: Bao gồm giao diện người dùng, máy chủ ứng dụng, cơ sở dữ liệu, các module mã hóa dữ liệu và cơ chế kiểm soát truy cập.
* Cơ chế bảo mật tích hợp: Các thành phần như xác thực đa yếu tố (MFA), HTTPS/SSL, mã hóa dữ liệu nhạy cảm, hệ thống log và giám sát an ninh.
* Phản ứng (Response):
* Xác thực và ủy quyền: Hệ thống sẽ xác minh danh tính người dùng qua các cơ chế xác thực (mật khẩu, MFA, …) và kiểm tra quyền truy cập trước khi cho phép truy cập dữ liệu.
* Mã hóa và bảo vệ dữ liệu: Dữ liệu nhạy cảm được mã hóa trong quá trình lưu trữ và truyền tải, đảm bảo chỉ những người dùng hợp lệ mới có thể giải mã.
* Ghi nhận và báo cáo: Mọi hành động, đặc biệt là các truy cập hoặc thay đổi dữ liệu bất thường, sẽ được ghi log và có thể báo cáo cho quản trị viên để kiểm tra và xử lý.
* Phản ứng tự động: Trong trường hợp phát hiện truy cập trái phép, hệ thống sẽ kích hoạt các cảnh báo an ninh, tạm khóa tài khoản và áp dụng các biện pháp bảo vệ bổ sung.
* Đo lường phản ứng (Response Measure):
* Tỷ lệ truy cập hợp lệ: Đo lường số lượng truy cập thành công so với tổng số yêu cầu, đảm bảo chỉ những truy cập được xác thực mới được phép.
* Thời gian phản ứng của cơ chế bảo mật: Thời gian từ khi phát hiện hành vi bất thường đến khi hệ thống đưa ra cảnh báo hoặc khóa truy cập phải được tối ưu hóa (ví dụ: trong vài giây).
* Mức độ mã hóa và kiểm soát truy cập: Đánh giá bằng các bài kiểm tra xâm nhập (penetration test) và các tiêu chí đánh giá bảo mật nội bộ để đảm bảo dữ liệu luôn được bảo vệ.
* Khả năng truy xuất và báo cáo: Các log an ninh cần được lưu trữ đầy đủ và cho phép truy xuất nhanh chóng, đảm bảo có thể xác minh và truy nguyên gốc của mọi hành động trên hệ thống.

NFR 4 : Yêu cầu về tính hữu dụng

* + **Định nghĩa(ISO) :** Mức độ mà một sản phẩm hoặc hệ thống có thể được sử dụng bởi những người dùng đã chỉ định để đạt được các mục tiêu đã chỉ định với hiệu quả, hiệu suất và sự hài lòng trong một bối cảnh sử dụng cụ thể.
  + **Thành phần :** Theo tiêu chuẩn ISO, tính hữu dụng thường được phân tích theo các thành phần sau :
    - Khả năng nhận biết tính phù hợp (**Appropriateness recognizability**) : Mức độ mà người dùng có thể nhận ra liệu một sản phẩm hoặc hệ thống có phù hợp với nhu cầu của họ hay không.
    - Khả năng học hỏi (**Learnability**) : Mức độ mà một sản phẩm hoặc hệ thống có thể được sử dụng bởi người dùng đã chỉ định để đạt được các mục tiêu đã chỉ định của việc học cách sử dụng sản phẩm hoặc hệ thống đó một cách hiệu quả, hiệu suất, không có rủi ro và hài lòng trong một bối cảnh sử dụng cụ thể.
    - Khả năng vận hành (**Operability**) : Mức độ mà một sản phẩm hoặc hệ thống có các đặc tính làm cho nó dễ dàng vận hành và điều khiển.
    - Bảo vệ người dùng khỏi lỗi (**User error protection**) : Mức độ mà một hệ thống bảo vệ người dùng khỏi việc mắc phải lỗi.
    - Thẩm mỹ giao diện người dùng(**User interface aesthetics**) : Mức độ mà giao diện người dùng tạo ra một sự tương tác dễ chịu và thỏa mãn cho người dùng.
    - Khả năng truy cập (**Accessibility**) : Mức độ mà một sản phẩm hoặc hệ thống có thể được sử dụng bởi những người có phạm vi đặc điểm và khả năng rộng nhất để đạt được một mục tiêu cụ thể trong một bối cảnh sử dụng cụ thể.
  + **Nguồn :** [**ISO/IEC 25010:2011**](https://www.iso.org/obp/ui/?fbclid=IwY2xjawGsDo5leHRuA2FlbQIxMAABHdqLFb-EJo5aWXsetspU5ipXAbiL3swyCURDQa43vbVf8oRkxnvjcp3Mcg_aem_0fAED_FQhF3N8gUdC10KvQ#iso:std:iso-iec:25010:ed-1:v1:en)
  + **Tính hữu dụng (NFR cũ của dự án) :**
    - Phù hợp với nhu cầu sử dụng: Hệ thống phải đáp ứng đầy đủ các yêu cầu của người dùng và nhà quản lý rạp chiếu phim.
    - Hướng dẫn sử dụng rõ ràng: Cung cấp tài liệu hướng dẫn sử dụng và vận hành hệ thống chi tiết.
    - Giao diện thân thiện: Giao diện người dùng phải trực quan, dễ sử dụng và phù hợp với thói quen sử dụng của khách hàng.
    - Truy cập nhanh và dễ dàng: Hệ thống cần đảm bảo tốc độ truy cập nhanh và dễ dàng, không bị gián đoạn do hiệu suất máy chủ hoặc tốc độ mạng chậm
  + **Phát biểu lại:** Hệ thống SoundShare phải được thiết kế sao cho người dùng có thể dễ dàng nhận biết và tiếp cận các chức năng của hệ thống để thỏa mãn nhu cầu chia sẻ và thưởng thức âm nhạc. Điều này bao gồm:
* Khả năng nhận biết tính phù hợp: Người dùng phải nhanh chóng xác định được rằng hệ thống đáp ứng đúng nhu cầu tìm kiếm, chia sẻ và tương tác với nội dung âm nhạc.
* Khả năng học hỏi: Giao diện và hướng dẫn sử dụng của SoundShare cần đơn giản, trực quan, giúp người dùng mới có thể nhanh chóng làm quen và sử dụng hiệu quả các chức năng của hệ thống.
* Khả năng vận hành: Hệ thống cần dễ dàng vận hành và điều khiển, cho phép người dùng thực hiện các thao tác như tìm kiếm, đăng tải, tương tác… một cách liền mạch và mượt mà.
* Bảo vệ người dùng khỏi lỗi: Hệ thống cần tích hợp cơ chế cảnh báo và hướng dẫn khi người dùng thực hiện sai thao tác, giảm thiểu rủi ro do lỗi thao tác.
* Thẩm mỹ giao diện người dùng: Giao diện phải thân thiện, hiện đại, tạo cảm giác dễ chịu và phù hợp với thói quen sử dụng của đối tượng người dùng mục tiêu.
* Khả năng truy cập: SoundShare cần hỗ trợ đa nền tảng (web, mobile) và được thiết kế đáp ứng các tiêu chuẩn về truy cập, bao gồm cả người dùng có nhu cầu đặc biệt.
  + **Phân tích theo kịch bản 6 thành phần Bass :**
    - Nguồn kích hoạt (Stimulus Source):
      * Người dùng SoundShare: Bao gồm cả người dùng mới và người dùng thường xuyên, từ các đối tượng có trình độ kỹ năng sử dụng công nghệ khác nhau (người dùng thông thường, người dùng chuyên sâu, người dùng có nhu cầu đặc biệt).
      * Quản trị viên và nhà phát triển: Người chịu trách nhiệm tạo nội dung, quản lý hệ thống và cập nhật hướng dẫn sử dụng.
    - Kích hoạt (Stimulus):
      * Yêu cầu tương tác: Người dùng gửi các yêu cầu tương tác như đăng nhập, tìm kiếm bài hát, chia sẻ nội dung, hay thực hiện thao tác trong quá trình duyệt và sử dụng giao diện.
      * Yêu cầu hướng dẫn: Người dùng mới cần được hướng dẫn qua các bước sử dụng hệ thống hoặc xử lý khi gặp lỗi thao tác.
    - Ngữ cảnh (Environment):
      * Môi trường sử dụng đa dạng: Hệ thống hoạt động trong bối cảnh trực tuyến trên nhiều thiết bị (máy tính, điện thoại di động, máy tính bảng) với các điều kiện mạng và mức độ quen thuộc về công nghệ khác nhau.
      * Điều kiện sử dụng bình thường: Người dùng truy cập và sử dụng SoundShare trong điều kiện mạng ổn định, cũng như trong môi trường có thể có sự gián đoạn hoặc thay đổi về tốc độ kết nối.
    - Hệ thống (Artifact):
      * Giao diện người dùng của SoundShare: Bao gồm các màn hình đăng nhập, tìm kiếm, trang cá nhân, trang chia sẻ âm nhạc, và các module hướng dẫn sử dụng.
      * Tài liệu trợ giúp và hướng dẫn sử dụng: Các nội dung hướng dẫn, FAQ, video demo và trợ giúp trực tuyến tích hợp sẵn trên hệ thống.
      * Các cơ chế hỗ trợ người dùng: Các tính năng như tooltip, trợ giúp tự động, cảnh báo khi thao tác sai và thiết kế giao diện phù hợp với nguyên tắc thiết kế thân thiện.
    - Phản ứng (Response):
      * Phản hồi trực quan: Khi người dùng thực hiện thao tác, hệ thống cung cấp phản hồi ngay lập tức qua giao diện (thông báo thành công, cảnh báo lỗi, gợi ý hướng dẫn…).
      * Hỗ trợ tương tác: Hệ thống cung cấp các chỉ dẫn, hướng dẫn trực tuyến và tài liệu chi tiết để người dùng nhanh chóng làm quen và xử lý các thao tác cần thiết.
      * Cải thiện qua phản hồi người dùng: Các thông tin đánh giá về trải nghiệm (UX) được thu thập định kỳ để cải tiến giao diện và hướng dẫn sử dụng, đảm bảo tính hữu dụng luôn được tối ưu.
    - Đo lường phản ứng (Response Measure):
      * Thời gian học hỏi: Đo lường thời gian trung bình từ lần đăng nhập đầu tiên đến khi người dùng có thể thực hiện các thao tác chính một cách độc lập mà không cần trợ giúp thêm.
      * Mức độ hài lòng: Thông qua khảo sát, đánh giá trải nghiệm người dùng (UX), chỉ số hài lòng được ghi nhận và phân tích.
      * Tỷ lệ lỗi thao tác: Đo lường số lượng lỗi người dùng gặp phải khi thao tác trên giao diện và số lần cần gọi trợ giúp, nhằm giảm thiểu lỗi qua các cải tiến giao diện.
      * Đánh giá giao diện (UI aesthetics) và khả năng truy cập (Accessibility): Sử dụng các tiêu chí định tính và định lượng từ các bài kiểm tra UX/UI để đảm bảo giao diện trực quan, thẩm mỹ và phù hợp với mọi nhóm người dùng, kể cả những người có nhu cầu đặc biệt.

NFR 5 : Yêu cầu khả năng bảo trì:

* + **Định nghĩa** **(ISO):** Mức độ hiệu quả và hiệu suất mà một sản phẩm hoặc hệ thống có thể được sửa đổi bởi các nhân viên bảo trì dự kiến.
  + **Thành phần:** Theo tiêu chuẩn ISO, tính bảo mật thường được phân tích dựa trên các thành phần sau:
    - **Modularity (Tính mô-đun)**: Mức độ mà một hệ thống hoặc chương trình máy tính được cấu thành từ các thành phần riêng biệt, sao cho thay đổi một thành phần có tác động tối thiểu đến các thành phần khác.
    - **Reusability (Tính tái sử dụng)**: Mức độ mà một tài nguyên có thể được sử dụng trong nhiều hệ thống hoặc để xây dựng các tài nguyên khác.
    - **Analysability (Khả năng phân tích)**: Mức độ hiệu quả và hiệu suất mà có thể đánh giá tác động của một thay đổi dự định đối với một hoặc nhiều phần của sản phẩm hoặc hệ thống, hoặc để chẩn đoán các thiếu sót hoặc nguyên nhân gây ra sự cố, hoặc để xác định các phần cần thay đổi.
    - **Modifiability (Khả năng sửa đổi)**: Mức độ mà sản phẩm hoặc hệ thống có thể được sửa đổi một cách hiệu quả và hiệu suất mà không gây ra lỗi hoặc làm giảm chất lượng sản phẩm hiện có.
    - **Testability (Khả năng kiểm thử)**: Mức độ hiệu quả và hiệu suất mà các tiêu chí kiểm thử có thể được thiết lập cho một hệ thống, sản phẩm hoặc thành phần và các bài kiểm thử có thể được thực hiện để xác định xem các tiêu chí đó có được đáp ứng hay không.
  + **Nguồn :** [**ISO/IEC 25010:2011**](https://www.iso.org/obp/ui/?fbclid=IwY2xjawGsDo5leHRuA2FlbQIxMAABHdqLFb-EJo5aWXsetspU5ipXAbiL3swyCURDQa43vbVf8oRkxnvjcp3Mcg_aem_0fAED_FQhF3N8gUdC10KvQ#iso:std:iso-iec:25010:ed-1:v1:en)
  + **Khả năng bảo trì (NFR cũ của dự án):**
    - Dễ dàng bảo trì và nâng cấp: Hệ thống cần có khả năng bảo trì và nâng cấp dễ dàng để thích nghi với những thay đổi trong yêu cầu kinh doanh và công nghệ.
  + **Phát biểu lại :** Hệ thống SoundShare phải được thiết kế và phát triển theo nguyên tắc mô-đun, cho phép bảo trì, nâng cấp, và sửa đổi một cách nhanh chóng và hiệu quả mà không ảnh hưởng đến chất lượng và hoạt động của các phần khác. Điều này bao gồm:
* Tính mô-đun (Modularity):  
  Hệ thống được chia thành các thành phần riêng biệt có mức độ liên kết thấp và tính độc lập cao, giúp việc thay đổi, sửa chữa hoặc nâng cấp một module không làm ảnh hưởng đến các module khác.
* Tính tái sử dụng (Reusability):  
  Các thành phần, module hoặc thư viện được xây dựng theo chuẩn và có thể sử dụng lại trong các dự án khác hoặc cho các chức năng mở rộng của SoundShare, giảm thiểu thời gian phát triển và kiểm thử.
* Khả năng phân tích (Analysability):  
  Hệ thống cần có tài liệu chi tiết, cấu trúc rõ ràng và mã nguồn dễ hiểu để các nhân viên bảo trì có thể nhanh chóng đánh giá tác động của bất kỳ thay đổi nào, phát hiện lỗi và xác định các phần cần được sửa đổi.
* Khả năng sửa đổi (Modifiability):  
  Hệ thống phải dễ dàng cho phép thực hiện các thay đổi, từ việc bổ sung tính năng mới cho đến điều chỉnh các chức năng hiện có, mà không gây ra lỗi không mong muốn hoặc ảnh hưởng đến hiệu năng tổng thể.
* Khả năng kiểm thử (Testability):  
  Các thành phần của hệ thống cần được xây dựng với các tiêu chí kiểm thử rõ ràng, cho phép thực hiện các bài kiểm thử tự động và thủ công để xác định nhanh chóng xem các thay đổi có đáp ứng được yêu cầu chức năng và phi chức năng hay không.
  + **Phân tích theo kịch bản 6 thành phần Bass :** 
    - Nguồn kích hoạt (Stimulus Source):
* Nhà phát triển và nhân viên bảo trì: Các kỹ sư, lập trình viên và nhân viên chịu trách nhiệm bảo trì, nâng cấp hệ thống SoundShare.
* Yêu cầu từ phía khách hàng hoặc thay đổi về chiến lược kinh doanh: Các yêu cầu nâng cấp chức năng, sửa lỗi hoặc bổ sung tính năng mới để đáp ứng nhu cầu thị trường.
* Kích hoạt (Stimulus):
* Yêu cầu sửa đổi hoặc nâng cấp: Phát sinh từ các báo cáo lỗi, phản hồi của người dùng hoặc các thay đổi trong yêu cầu kinh doanh.
* Sự cố hệ thống: Các lỗi phát sinh cần được khắc phục kịp thời để đảm bảo hệ thống hoạt động ổn định.
* Ngữ cảnh (Environment):
* Môi trường phát triển và vận hành: Hệ thống đang chạy trong môi trường sản xuất và được duy trì thông qua các phiên bản cập nhật định kỳ.
* Điều kiện bảo trì: Các hoạt động bảo trì, nâng cấp hoặc sửa lỗi được thực hiện trong khung thời gian quy định, thường là ngoài giờ cao điểm để không ảnh hưởng đến người dùng.
* Hệ thống (Artifact):
* Kiến trúc hệ thống SoundShare: Bao gồm các module chức năng (giao diện người dùng, xử lý dữ liệu, quản lý nội dung, …) được xây dựng theo nguyên tắc mô-đun, tài liệu thiết kế, mã nguồn và các bộ kiểm thử tự động.
* Cơ sở hạ tầng hỗ trợ: Hệ thống quản lý phiên bản, công cụ giám sát và các công cụ kiểm thử tự động giúp đảm bảo khả năng bảo trì cao.
* Phản ứng (Response):
* Cập nhật và sửa đổi: Các thay đổi được thực hiện một cách có hệ thống, với quy trình kiểm thử và đánh giá tác động trước khi triển khai vào môi trường sản xuất.
* Thông báo và ghi log: Mọi thay đổi và lỗi phát sinh đều được ghi nhận chi tiết, hỗ trợ việc theo dõi, phân tích và xác định nguyên nhân gốc rễ.
* Đo lường phản ứng (Response Measure):
* Thời gian xử lý yêu cầu bảo trì: Thời gian trung bình từ lúc phát hiện lỗi hoặc yêu cầu thay đổi đến khi hoàn thành bảo trì, với mục tiêu tối thiểu hóa thời gian gián đoạn hệ thống.
* Mức độ tác động của thay đổi: Đánh giá mức độ ảnh hưởng của thay đổi đến các module khác thông qua các bài kiểm thử tự động và thủ công, đảm bảo không gây ra lỗi phụ.
* Tỷ lệ lỗi sau bảo trì: Theo dõi số lượng lỗi phát sinh sau mỗi lần nâng cấp hoặc sửa chữa, nhằm đảm bảo các thay đổi được thực hiện chính xác và hiệu quả.
* Khả năng kiểm thử: Số lượng và độ bao phủ của các bài kiểm thử tự động được thiết lập để đánh giá chất lượng của các module và tính tương thích của hệ thống sau khi thay đổi.

NFR 6: Tính tương thích :

* + **Định nghĩa:** Mức độ mà một sản phẩm, hệ thống hoặc thành phần có thể trao đổi thông tin với các sản phẩm, hệ thống hoặc thành phần khác và/hoặc thực hiện các chức năng yêu cầu của nó, trong khi vẫn chia sẻ cùng một môi trường phần cứng hoặc phần mềm.
  + **Thành phần:**
    - Cùng tồn tại: Mức độ mà một sản phẩm có thể thực hiện các chức năng yêu cầu của nó một cách hiệu quả trong khi chia sẻ môi trường và tài nguyên chung với các sản phẩm khác, mà không gây tác động tiêu cực đến bất kỳ sản phẩm nào khác.
    - Khả năng tương tác: Mức độ mà hai hoặc nhiều hệ thống, sản phẩm hoặc thành phần có thể trao đổi thông tin và sử dụng thông tin đã được trao đổi.
  + **Nguồn:** [**ISO/IEC 25010:2011**](https://www.iso.org/obp/ui/?fbclid=IwY2xjawGsDo5leHRuA2FlbQIxMAABHdqLFb-EJo5aWXsetspU5ipXAbiL3swyCURDQa43vbVf8oRkxnvjcp3Mcg_aem_0fAED_FQhF3N8gUdC10KvQ#iso:std:iso-iec:25010:ed-1:v1:en)
  + **Tính tương thích (NFR cũ):**
    - Hệ thống cần tương thích đa thiết bị
  + **Phát biểu lại :** Hệ thống SoundShare phải có khả năng hoạt động ổn định và trao đổi thông tin hiệu quả với các sản phẩm, hệ thống hoặc thành phần khác trong cùng một môi trường phần cứng và phần mềm. Cụ thể:
* Cùng tồn tại:  
  SoundShare cần chạy mượt mà trên nhiều thiết bị và nền tảng (ví dụ: máy tính để bàn, laptop, smartphone, tablet) với các hệ điều hành và trình duyệt khác nhau mà không gây ra xung đột hay tiêu thụ tài nguyên quá mức, ảnh hưởng đến hiệu suất của các ứng dụng khác.
* Khả năng tương tác:  
  SoundShare phải có khả năng trao đổi dữ liệu và tương tác liền mạch với các hệ thống bên ngoài (chẳng hạn như các dịch vụ streaming, hệ thống thanh toán, hoặc các nền tảng mạng xã hội) để hỗ trợ chức năng chia sẻ âm nhạc và kết nối người dùng. Điều này đòi hỏi việc sử dụng các giao thức và API tiêu chuẩn, đảm bảo rằng thông tin được truyền tải và xử lý chính xác giữa các hệ thống liên quan.
  + **Phân tích theo kịch bản 6 thành phần Bass:**
    - Nguồn kích hoạt (Stimulus Source):
      * Người dùng đa thiết bị: Người dùng truy cập SoundShare qua các thiết bị khác nhau (smartphone, máy tính bảng, máy tính để bàn) và từ các môi trường phần mềm đa dạng (iOS, Android, Windows, macOS, các trình duyệt web).
      * Hệ thống bên ngoài: Các dịch vụ và ứng dụng liên kết (ví dụ: dịch vụ streaming, hệ thống thanh toán, nền tảng mạng xã hội) yêu cầu trao đổi dữ liệu với SoundShare.
    - Kích hoạt (Stimulus):
      * Yêu cầu truy cập từ nhiều thiết bị: Người dùng gửi yêu cầu truy cập, đăng nhập, tìm kiếm và tương tác với nội dung từ các nền tảng khác nhau.
      * Yêu cầu giao tiếp giữa hệ thống: Các giao thức trao đổi dữ liệu giữa SoundShare và các hệ thống bên ngoài được kích hoạt khi cần chia sẻ thông tin hoặc thực hiện các chức năng liên quan (ví dụ: thanh toán, chia sẻ bài hát).
    - Ngữ cảnh (Environment):
      * Môi trường đa nền tảng: SoundShare hoạt động trong một môi trường có sự đa dạng về thiết bị, hệ điều hành và trình duyệt.
      * Chia sẻ tài nguyên: Hệ thống hoạt động cùng lúc với các ứng dụng khác trên cùng một hạ tầng, đòi hỏi khả năng tương thích để tránh xung đột và đảm bảo hiệu năng chung.
    - Hệ thống (Artifact):
      * Hệ thống SoundShare: Bao gồm giao diện người dùng (responsive design, ứng dụng native hoặc web), máy chủ ứng dụng và các API tiêu chuẩn được thiết kế theo chuẩn giao tiếp mở để đảm bảo trao đổi thông tin với các hệ thống khác.
      * Cơ chế tương thích: Sử dụng các tiêu chuẩn giao tiếp (ví dụ: RESTful API, JSON, XML) và công nghệ đa nền tảng (responsive web design, native apps cho iOS/Android) nhằm đảm bảo hoạt động mượt mà trên nhiều môi trường.
    - Phản ứng (Response):
      * Phản hồi đa thiết bị: Khi truy cập từ các thiết bị khác nhau, SoundShare phản hồi hiển thị giao diện tối ưu và thực hiện các chức năng chính mà không gặp lỗi hay giảm hiệu suất.
      * Trao đổi dữ liệu liền mạch: Các giao thức trao đổi thông tin giữa SoundShare và các hệ thống bên ngoài được thực hiện trơn tru, đảm bảo dữ liệu được truyền tải chính xác và kịp thời mà không bị gián đoạn.
    - Đo lường phản ứng (Response Measure):
      * Độ tương thích thiết bị: Đánh giá thông qua các bài kiểm thử tương thích trên nhiều thiết bị, hệ điều hành và trình duyệt, đảm bảo SoundShare hoạt động ổn định trên ít nhất các nền tảng chủ đạo (ví dụ: iOS, Android, Windows, macOS).
      * Thời gian phản hồi khi trao đổi dữ liệu: Thời gian xử lý yêu cầu và trao đổi dữ liệu giữa SoundShare với các hệ thống bên ngoài được đo lường để đảm bảo không vượt quá giới hạn chấp nhận được (ví dụ: phản hồi trong vài giây).
      * Tỷ lệ lỗi tương thích: Theo dõi số lượng lỗi phát sinh do vấn đề tương thích (ví dụ: hiển thị không đúng, lỗi giao diện, lỗi kết nối API) và đặt mục tiêu tối thiểu hóa tỷ lệ này qua các phiên bản cập nhật.

NFR 7 : Tính văn hóa (Cultural Adaptability):

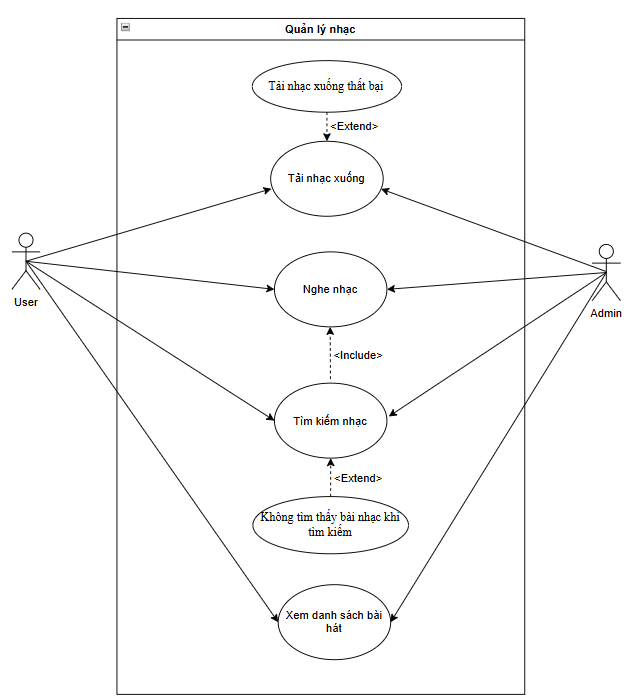
* + **Định nghĩa:** Mức độ mà hệ thống phù hợp với các khác biệt văn hóa, ngôn ngữ, hoặc yêu cầu địa phương của người dùng, đảm bảo trải nghiệm tương thích với các giá trị văn hóa của từng khu vực.  
    **Nguồn:** Marcus & Gould (2000), *Cross-cultural Design Principles*.
  + **Tính văn hóa (NFR cũ):** Ngôn ngữ chính của hệ thống là Tiếng Việt để phù hợp với đối tượng khách hàng chính là người Việt Nam.
  + **Phát biểu lại:** Hệ thống SoundShare được thiết kế với khả năng tương thích văn hóa cao, không chỉ sử dụng ngôn ngữ chính là Tiếng Việt để phục vụ đối tượng khách hàng Việt Nam mà còn có khả năng mở rộng hỗ trợ đa ngôn ngữ và điều chỉnh giao diện theo các tiêu chuẩn, giá trị văn hóa của từng khu vực. Cụ thể, giao diện và nội dung sẽ được tùy biến về mặt hình ảnh, biểu tượng, màu sắc, định dạng ngày giờ và các yếu tố tương tác khác sao cho phù hợp với đặc thù văn hóa của từng nhóm người dùng, nhằm mang lại trải nghiệm tự nhiên, thân thiện và tạo sự hài lòng cao.
* **Phân tích theo kịch bản 6 thành phần Bass:**
  + Nguồn kích hoạt (Stimulus Source):
    - Người dùng đa văn hóa: Bao gồm người dùng từ các vùng miền khác nhau (đặc biệt là người dùng Việt Nam và có tiềm năng mở rộng ra thị trường quốc tế) với những giá trị văn hóa, ngôn ngữ và phong cách giao tiếp riêng.
    - Nhà quản trị và các bên liên quan: Các chuyên gia về giao diện và nội dung văn hóa, nhà phát triển cần đảm bảo hệ thống luôn phù hợp với tiêu chuẩn địa phương.
  + Kích hoạt (Stimulus):
    - Yêu cầu hiển thị nội dung và giao diện phù hợp: Khi người dùng truy cập hệ thống, họ mong đợi được trải nghiệm giao diện phù hợp với ngôn ngữ và phong cách văn hóa của mình.
    - Yêu cầu điều chỉnh nội dung theo khu vực: Các chiến dịch marketing, sự kiện văn hóa đặc thù hoặc phản hồi từ người dùng có thể kích hoạt nhu cầu điều chỉnh giao diện, nội dung sao cho phù hợp với đặc điểm văn hóa địa phương.
  + Ngữ cảnh (Environment):
    - Môi trường sử dụng đa dạng: Hệ thống hoạt động trên nhiều nền tảng (web, mobile) và được truy cập từ các khu vực địa lý, văn hóa khác nhau với các ngôn ngữ và thói quen sử dụng khác nhau.
    - Điều kiện thay đổi theo thị trường: Khi SoundShare mở rộng ra các thị trường mới, yêu cầu về nội dung và giao diện cần linh hoạt điều chỉnh để đáp ứng văn hóa địa phương.
  + Hệ thống (Artifact):
    - Hệ thống SoundShare: Gồm giao diện người dùng, hệ thống quản lý nội dung và các module hỗ trợ định dạng, hiển thị thông tin (như định dạng ngày giờ, màu sắc, biểu tượng).
    - Cơ chế tùy biến văn hóa: Các thành phần phần mềm được thiết kế theo hướng cho phép tùy chỉnh giao diện, nội dung và thiết lập ngôn ngữ dễ dàng, có thể mở rộng hỗ trợ các ngôn ngữ và yếu tố văn hóa mới theo nhu cầu.
  + Phản ứng (Response):
    - Tùy biến giao diện và nội dung: Khi người dùng từ các khu vực khác nhau truy cập, hệ thống sẽ hiển thị giao diện và nội dung theo ngôn ngữ, định dạng và phong cách văn hóa phù hợp.
    - Tùy chọn cài đặt người dùng: Hệ thống cho phép người dùng lựa chọn ngôn ngữ và tùy chỉnh một số yếu tố giao diện, hoặc tự động nhận diện thông tin khu vực để điều chỉnh giao diện theo tiêu chuẩn văn hóa tương ứng.
    - Cập nhật nội dung văn hóa: Qua các phản hồi từ người dùng và các chiến dịch marketing, hệ thống sẽ được cập nhật, điều chỉnh nội dung và giao diện để đảm bảo luôn phù hợp với xu hướng và yêu cầu văn hóa.
  + Đo lường phản ứng (Response Measure):
    - Mức độ hài lòng của người dùng: Đo lường qua khảo sát, phản hồi người dùng về tính phù hợp của giao diện và nội dung theo văn hóa, ngôn ngữ của từng khu vực.
    - Tỷ lệ chuyển đổi ngôn ngữ: Số lượng người dùng lựa chọn sử dụng các ngôn ngữ khác ngoài Tiếng Việt (nếu mở rộng) và mức độ tương thích của giao diện với các ngôn ngữ đó.
    - Chỉ số lỗi hiển thị và định dạng: Số lỗi liên quan đến hiển thị giao diện, định dạng ngày giờ, biểu tượng, màu sắc không phù hợp với tiêu chuẩn văn hóa của từng khu vực, từ đó hướng đến mục tiêu giảm thiểu các lỗi này qua các phiên bản cập nhật.

## Phân tích và đặc tả các Use Cases

Xác định tác nhân của hệ thống

* Người dùng: Đăng nhập, đăng ký, đăng xuất, đổi mật khẩu, nghe nhạc, tải nhạc, tìm kiếm nhạc, xem danh sách các bài nhạc, thêm bình luận, sửa bình luận, xóa bình luận, xem danh sách bình luận, tạo danh sách phát nhạc, chia sẻ danh sách phát nhạc, tải nhạc lên, chỉnh sửa thông tin nhạc, xóa nhạc, tải nhạc xuống
* Admin: Đăng nhập, đăng xuất, đổi mật khẩu, xem danh sách tài khoản, nghe nhạc, tải nhạc xuống, tìm kiếm nhạc, xem danh sách các bài nhạc, xóa bình luận, xem danh sách bình luận

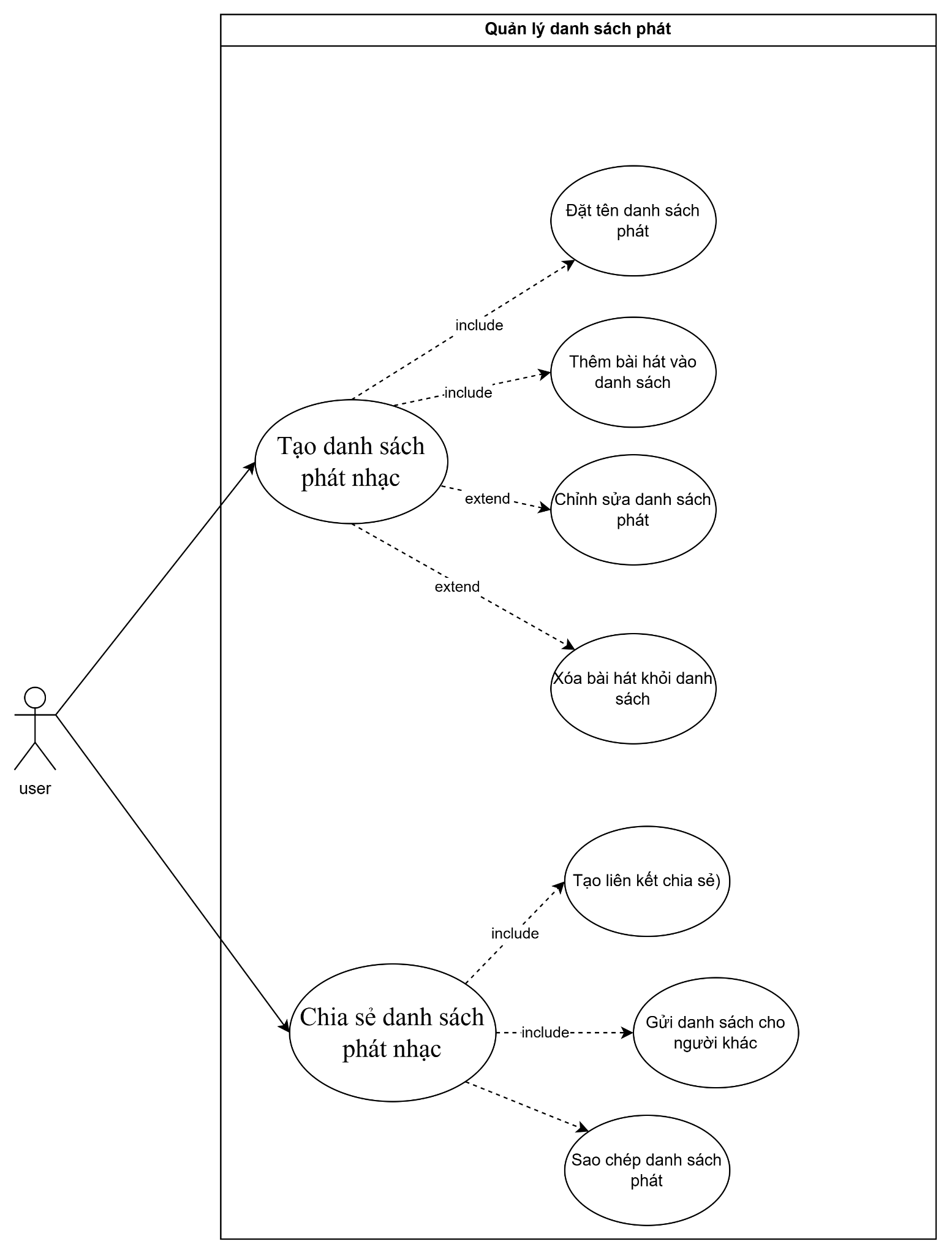
### Use case quản lý nhạc (Sáng)



Đặc tả use case:

| UC01\_QLN | Quản lý nhạc |
| --- | --- |
| Tên Use Case (Use Case Name) | Quản lý nhạc |
| Mô tả tóm tắt (Brief Description) | Hệ thống Quản lý nhạc cho phép User và Admin thực hiện các thao tác sau:   * Tải nhạc xuống * Nghe nhạc * Tìm kiếm nhạc * Xem danh sách bài hát   Các tình huống mở rộng bao gồm:   * Tải nhạc xuống thất bại khi có lỗi trong quá trình tải. * Không tìm thấy bài nhạc khi tìm kiếm nếu bài hát không tồn tại trong hệ thống. |
| Tác nhân (Actors) | Người dùng (User), Quản trị viên (Admin) |
| Tiền điều kiện (Preconditions) | * User hoặc Admin đã đăng nhập vào hệ thống. * Hệ thống có kết nối Internet để tìm kiếm và nghe nhạc trực tuyến. * Nếu tải nhạc xuống, thiết bị cần có đủ dung lượng lưu trữ. |
| Luồng sự kiện chính (Basic Flow / Main Flow) | 1. User hoặc Admin đăng nhập vào hệ thống. 2. Người dùng có thể thực hiện các thao tác sau:  * Tải nhạc xuống: Chọn bài hát từ danh sách để tải xuống thiết bị. * Nghe nhạc: Chọn bài hát để phát trực tuyến hoặc phát từ danh sách nhạc đã tải xuống. * Tìm kiếm nhạc: Nhập từ khóa để tìm bài hát mong muốn. * Xem danh sách bài hát: Hiển thị danh sách bài hát có sẵn trong hệ thống.  1. Hệ thống xử lý yêu cầu và phản hồi kết quả. |
| Luồng mở rộng (Alternative Flows) | - 2.1a: Tải nhạc xuống thất bại: Nếu có lỗi kết nối mạng hoặc bộ nhớ thiết bị đầy, hệ thống hiển thị thông báo lỗi.  - 2.3a: Không tìm thấy bài nhạc khi tìm kiếm: Nếu không có kết quả phù hợp, hệ thống sẽ hiển thị thông báo lỗi. |
| Luồng ngoại lệ (Exception Flows) | - 2.2e:Nếu quá trình phát nhạc gặp lỗi (ví dụ: bài hát bị xóa, định dạng không được hỗ trợ), hệ thống sẽ hiển thị thông báo lỗi.  - 2.4e: Nếu danh sách bài hát quá lớn, hệ thống sẽ phân trang hoặc tối ưu hiển thị để tránh quá tải dữ liệu. |
| Hậu điều kiện (Postconditions) | * Nếu tải nhạc xuống thành công, bài hát được lưu trên thiết bị. * Nếu người dùng nghe nhạc, hệ thống ghi lại lịch sử nghe nhạc. * Nếu người dùng tìm kiếm nhạc, hệ thống hiển thị kết quả phù hợp hoặc thông báo lỗi. |
| Yêu cầu đặc biệt (Special Requirements) | * Hệ thống phải hỗ trợ các định dạng nhạc phổ biến như MP3, WAV. * Khi tải nhạc xuống, cần hiển thị dung lượng file và tốc độ tải. * Khi tìm kiếm nhạc, hệ thống phải hỗ trợ tìm kiếm theo tên bài hát, ca sĩ, album. |
| Điều kiện kích hoạt (Trigger) | * Người dùng hoặc Admin thực hiện một thao tác liên quan đến quản lý nhạc (tải xuống, nghe, tìm kiếm, xem danh sách). |
| Ghi chú khác (Notes) | * Hệ thống có thể tích hợp với API nhạc trực tuyến để mở rộng dữ liệu bài hát. |

### Use case quản lý danh sách phát (Trí)



Đặc tả use case

| UC02\_QLDSP | Quản lý danh sách phát |
| --- | --- |
| Tên Use Case (Use Case Name) | Quản lý danh sách phát |
| Mô tả tóm tắt (Brief Description) | Use Case này cho phép người dùng tạo, chỉnh sửa và chia sẻ danh sách phát nhạc. Người dùng có thể thêm hoặc xóa bài hát khỏi danh sách phát, đặt tên, lưu danh sách và chia sẻ danh sách phát với người khác thông qua liên kết hoặc gửi trực tiếp. |
| Tác nhân (Actors) | Người dùng (User) |
| Tiền điều kiện (Preconditions) | -Người dùng đã đăng nhập vào hệ thống.  -Người dùng có quyền truy cập để tạo và chỉnh sửa danh sách phát. |
| Luồng sự kiện chính (Basic Flow / Main Flow) | (**A) Tạo danh sách phát**  -Người dùng truy cập trang Quản lý danh sách phát.  -Người dùng nhấn vào Tạo danh sách phát mới.  -Người dùng nhập tên danh sách phát.  -Người dùng thêm bài hát vào danh sách.  -Người dùng nhấn Lưu danh sách phát.  **(B) Chia sẻ danh sách phát**  -Người dùng chọn danh sách phát muốn chia sẻ.  -Người dùng nhấn vào Chia sẻ.  -Người dùng chọn phương thức chia sẻ và xác nhận. |
| Luồng mở rộng (Alternative Flows) | -Người dùng tìm kiếm bài hát để thêm vào danh sách phát |
| Luồng ngoại lệ (Exception Flows) | -Người dùng không nhập tên danh sách phát  Quay lại bước 4 để nhập lại tên danh sách phát.  - Người dùng không thêm bài hát nào  Nếu người dùng xác nhận, danh sách phát được lưu nhưng trống.  -Chia sẻ thất bại do lỗi mạng  Người dùng có thể thử lại hoặc chọn phương thức chia sẻ khác. |
| Hậu điều kiện (Postconditions) | -Danh sách phát đã được tạo/lưu thành công và có thể sử dụng sau này.  -Nếu chia sẻ, danh sách phát có thể được truy cập bởi người nhận qua liên kết hoặc email. |
| Yêu cầu đặc biệt (Special Requirements) | -Hệ thống cần đảm bảo bảo mật danh sách phát riêng tư, chỉ chủ sở hữu hoặc người được cấp quyền mới có thể xem. |
| Điều kiện kích hoạt (Trigger) | -Người dùng muốn tạo mới hoặc chỉnh sửa danh sách phát.  -Người dùng muốn chia sẻ danh sách phát với bạn bè hoặc công khai. |
| Ghi chú khác (Notes) | -Chức năng chia sẻ có thể tích hợp với mạng xã hội hoặc ứng dụng nhắn tin |

### Use case quản lý nhạc của tôi (Thắng)

*Sơ đồ Use Case Quản Lý Nhạc Của Tôi*

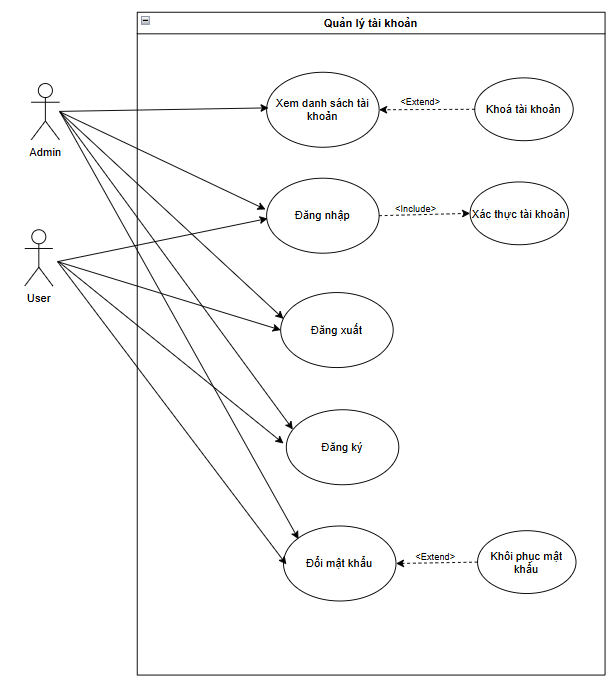
Đặc tả Use Case:

| **Mã Use Case** | UC03\_QLNCT |
| --- | --- |
| **Tên Use Case** | Quản Lý Nhạc Của Tôi |
| **Mô tả** | - Người dùng tải nhạc lên hệ thống  - Người dùng chỉnh sửa thông tin nhạc trên hệ thống  - Người dùng xóa nhạc trên hệ thống |
| **Tác nhân (Actor)** | - Người dùng |
| **Tiền điều kiện** | - Người dùng phải đăng nhập vào hệ thống |
| **Luồng sự kiện chính** | |
| **Hành động** | **Phản ứng hệ thống** |
| 1. Tải nhạc lên | |
| Người dùng nhấn vào mục “Tải nhạc lên” | Hệ thống hiển thị thông tin cho người dùng nhập vào |
| Người dùng nhập đầy đủ thông tin và bấm lưu | Hệ thống lưu trữ dữ liệu và hiển thị thông báo “Thành công” |
| 2. Chỉnh sửa thông tin nhạc | |
| Người dùng bấm vào bài nhạc đã tải lên, nhấn chọn icon “Sửa” | Hệ thống hiển thị thông tin bài nhạc |
| Người dùng sửa thông tin cần thiết và bấm “Lưu” | Hệ thống tiếp nhận thông tin và lưu trữ cập nhật lại |
| 3. Xóa nhạc | |
| Người dùng bấm chọn icon “Xóa” bên cạnh bài nhạc | Hệ thống hiển thị thông báo “Bạn có chắc muốn xóa bài nhạc?” |
| Người dùng bấm chọn “Đồng ý” | Thông tin bài nhạc được xóa khỏi hệ thống |
| **Luồng ngoại lệ** | |
| **Hành động** | **Phản ứng hệ thống** |
| 1.1. Tải nhạc lên thất bại | |
| Người dùng tải nhạc lên | Hệ thống kiểm tra định dạng và dung lượng file |
| File quá lớn hoặc không đúng định dạng | Hiển thị thông báo “Lỗi: Dung lượng file quá lớn hoặc định dạng không được hỗ trợ” |
| 2.1. Chỉnh sửa thất bại | |
| Người dùng chỉnh sửa thông tin bài nhạc | Hệ thống kiểm tra quyền hạn và trạng thái bài nhạc |
| Bài nhạc không tồn tại hoặc User không có quyền chỉnh sửa | Hiển thị thông báo "Lỗi: Bạn không có quyền chỉnh sửa bài nhạc này" |
| 3.1. Xóa nhạc thất bại | |
| Người dùng yêu cầu xóa nhạc | Hệ thống kiểm tra trạng thái bài nhạc |
| Bài nhạc đang được phát hoặc không tồn tại | Hiển thị thông báo "Lỗi: Bài nhạc đang được phát hoặc không tồn tại" |
| **Điều kiện trước** | Thông báo khi thêm, sửa, xóa thành công/thất bại. |
| **Điều kiện sau** | Tài khoản có quyền thay đổi bài nhạc của mình. |

### Use case quản lý bình luận (Phương)

| UC04\_QLBL | Quản lý bình luận |
| --- | --- |
| Tên Use Case (Use Case Name) | Quản lý bình luận |
| Mô tả tóm tắt (Brief Description) | Cho phép người dùng thực hiện các thao tác thêm, sửa, xóa và xem tất cả bình luận trên hệ thống. |
| Tác nhân (Actors) | Người dùng (User), Quản trị viên (Admin) |
| Tiền điều kiện (Preconditions) | - Người dùng hoặc quản trị viên đã đăng nhập vào hệ thống.  - Bình luận cần được thực hiện trên một bài viết hoặc nội dung hợp lệ. |
| Luồng sự kiện chính (Basic Flow / Main Flow) | 1. Người dùng truy cập chức năng "Quản lý bình luận".  2. Hệ thống hiển thị danh sách bình luận hiện có (nếu có).  3. Người dùng có thể thực hiện các thao tác:  - Thêm bình luận mới.  - Sửa bình luận của mình.  - Xóa bình luận (chỉ có thể xóa bình luận của mình hoặc Admin có quyền xóa tất cả).  - Xem tất cả bình luận.  4. Hệ thống lưu thay đổi và hiển thị trạng thái tương ứng. |
| Luồng mở rộng (Alternative Flows) | - 3.1a: Nếu bình luận vượt quá số ký tự cho phép, hệ thống sẽ thông báo lỗi.  - 3.2a: Nếu người dùng sửa bình luận quá giới hạn thời gian chỉnh sửa, hệ thống sẽ từ chối thay đổi.  - 3.3a: Nếu người dùng cố xóa bình luận của người khác, hệ thống sẽ hiển thị thông báo không có quyền. |
| Luồng ngoại lệ (Exception Flows) | - 3.1e: Nếu hệ thống gặp lỗi khi lưu bình luận, thông báo lỗi sẽ được hiển thị.  - 3.4e: Nếu danh sách bình luận quá lớn, hệ thống sẽ phân trang hoặc tối ưu hiển thị. |
| Hậu điều kiện (Postconditions) | - Bình luận mới đã được thêm vào hệ thống (nếu thêm thành công).  - Bình luận đã được cập nhật hoặc xóa khỏi hệ thống (nếu chỉnh sửa/xóa thành công). |
| Yêu cầu đặc biệt (Special Requirements) | - Hệ thống phải đảm bảo tính bảo mật, chỉ cho phép sửa/xóa bình luận của chính mình (trừ Admin).  - Phản hồi của hệ thống phải nhanh, không gây chậm trễ đáng kể. |
| Điều kiện kích hoạt (Trigger) | - Người dùng muốn thêm, chỉnh sửa, xóa hoặc xem bình luận. |
| Ghi chú khác (Notes) | - Hệ thống có thể gửi thông báo khi bình luận bị xóa hoặc chỉnh sửa bởi Admin. |

### Use case quản lý tài khoản (Phi)



| UC05\_QLTK | Quản lý tài khoản |
| --- | --- |
| Tên Use Case (Use Case Name) | Quản lý tài khoản |
| Mô tả tóm tắt (Brief Description) | Cho phép người dùng thực hiện các thao tác đăng ký, đăng nhập, đăng xuất, đổi mật khẩu và xem danh sách tài khoản trong hệ thống. |
| Tác nhân (Actors) | Người dùng (User), Quản trị viên (Admin) |
| Tiền điều kiện (Preconditions) | * Người dùng phải có tài khoản hợp lệ để đăng nhập. * Hệ thống phải có cơ sở dữ liệu lưu trữ thông tin tài khoản. |
| Luồng sự kiện chính (Basic Flow / Main Flow) | 1. Người dùng truy cập chức năng "Quản lý tài khoản". 2. Hệ thống hiển thị các tùy chọn: Đăng ký, Đăng nhập, Đăng xuất, Đổi mật khẩu, Xem danh sách tài khoản (chỉ dành cho Admin). 3. Người dùng có thể thực hiện các thao tác:    * Đăng ký tài khoản mới bằng email, số điện thoại hoặc tài khoản mạng xã hội.    * Đăng nhập bằng tên đăng nhập và mật khẩu.    * Đăng xuất khỏi hệ thống.    * Đổi mật khẩu bằng cách nhập mật khẩu cũ và mật khẩu mới.    * Xem danh sách tài khoản (chỉ Admin được phép truy cập). 4. Hệ thống lưu thay đổi và hiển thị trạng thái tương ứng. |
| Luồng mở rộng (Alternative Flows) | * Nếu email hoặc số điện thoại đã tồn tại, hệ thống sẽ thông báo lỗi khi đăng ký. * Nếu người dùng nhập sai mật khẩu khi đăng nhập quá số lần quy định, hệ thống sẽ khóa tài khoản tạm thời. * Nếu người dùng chưa đăng nhập nhưng cố đăng xuất, hệ thống sẽ hiển thị thông báo lỗi. * Nếu mật khẩu mới không đáp ứng yêu cầu bảo mật, hệ thống sẽ yêu cầu nhập lại. * Nếu người dùng không phải Admin nhưng cố truy cập danh sách tài khoản, hệ thống sẽ từ chối quyền truy cập. |
| Luồng ngoại lệ (Exception Flows) | * Nếu hệ thống gặp lỗi khi lưu thông tin đăng ký, thông báo lỗi sẽ được hiển thị. * Nếu hệ thống gặp sự cố khi xác thực tài khoản, người dùng sẽ nhận được thông báo lỗi. * Nếu hệ thống gặp lỗi khi đổi mật khẩu, quá trình thay đổi sẽ bị hủy và hệ thống hiển thị thông báo lỗi. * Nếu hệ thống gặp lỗi khi truy xuất danh sách tài khoản, hệ thống sẽ hiển thị thông báo lỗi và yêu cầu thử lại sau. |
| Hậu điều kiện (Postconditions) | * Người dùng đã có tài khoản mới (nếu đăng ký thành công). * Người dùng đã đăng nhập vào hệ thống (nếu đăng nhập thành công). * Người dùng đã đăng xuất khỏi hệ thống. * Mật khẩu của người dùng đã được thay đổi (nếu đổi mật khẩu thành công). * Quản trị viên đã xem danh sách tài khoản (nếu có quyền hợp lệ). |
| Yêu cầu đặc biệt (Special Requirements) | * Hệ thống phải đảm bảo tính bảo mật, sử dụng mã hóa cho mật khẩu. * Hệ thống phải hỗ trợ xác thực hai bước để tăng cường bảo mật. * Hệ thống phải có cơ chế phân quyền chặt chẽ, chỉ cho phép Admin xem danh sách tài khoản. * Phản hồi của hệ thống phải nhanh, không gây chậm trễ đáng kể |
| Điều kiện kích hoạt (Trigger) | Người dùng muốn đăng ký, đăng nhập, đăng xuất, đổi mật khẩu hoặc Admin muốn xem danh sách tài khoản. |
| Ghi chú khác (Notes) | * Hệ thống có thể gửi email xác nhận khi đăng ký hoặc đổi mật khẩu. * Hệ thống cần có cơ chế ghi log để theo dõi hoạt động truy cập danh sách tài khoản. |

# CHƯƠNG 3: KIẾN TRÚC PHẦN MỀM CỦA DỰ ÁN

## Kiến trúc phần mềm

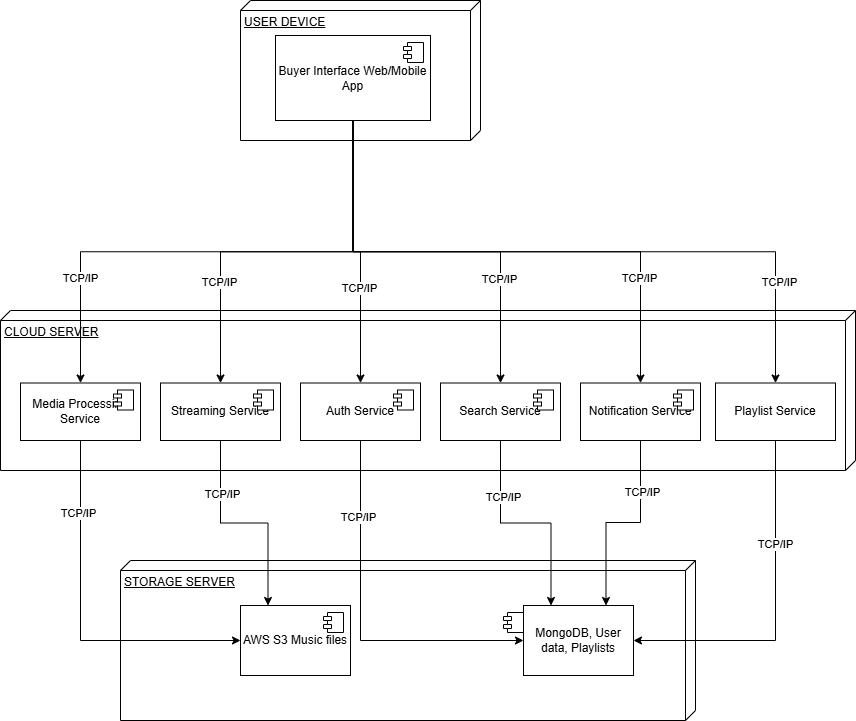
* Đặc tả kiến trúc phần mềm được chọn cho dự án (với các diagram và văn mô tả cần thiết).
* Áp dụng trình bày theo chuẩn hồ sơ kiến trúc phù hợp
* Trình bày ít nhất theo 3 kiểu cấu trúc: Allocation, C2C, Modules, … với các Views khác nhau, bằng các biểu đồ UML phù hợp
* Từng nội dung ghi rõ tên người thực hiện

**3.1.1 Kiến trúc lựa chọn: Microservices**

**3.1.2 Các kiểu kiến trúc và views**

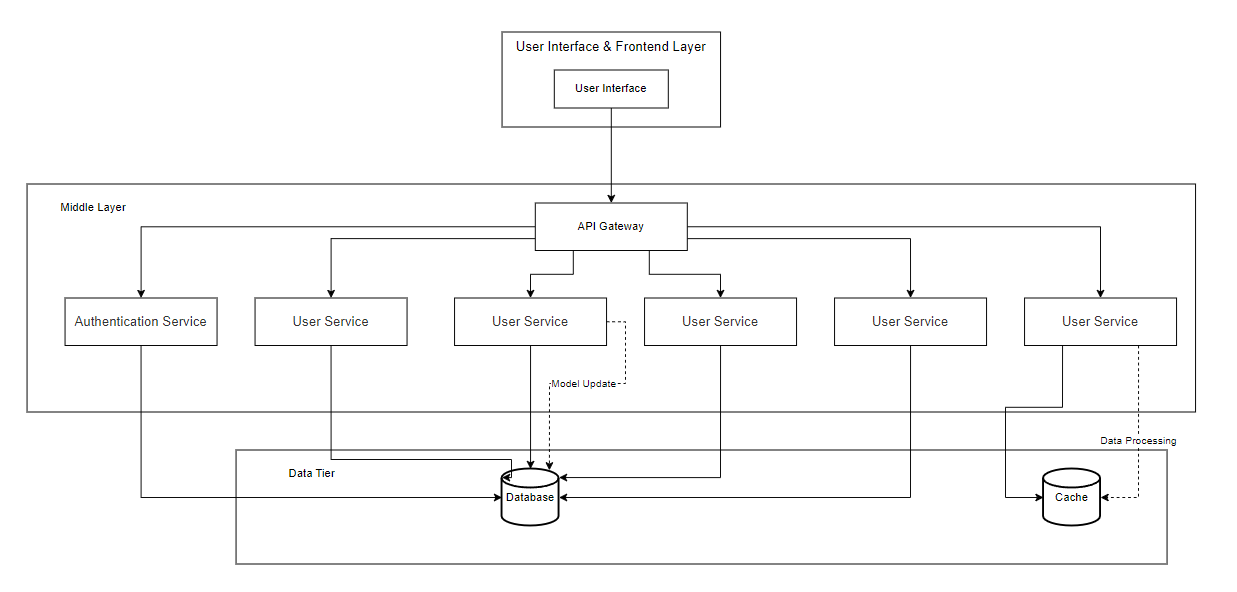
**3.1.2.1 Allocation View (Deployment diagram, implement view, Work Assignment View)**

**3.1.2.1.1. Deployment View (Phương)**

****

* Tổng Quan
  + Sơ đồ Deployment View thể hiện cách các thành phần của hệ thống triển khai trên các thiết bị vật lý và các giao thức kết nối giữa chúng. Mô hình này tập trung vào các yếu tố sau:
  + Các thiết bị (Devices): Là các máy chủ, máy khách, hoặc các thiết bị vật lý mà hệ thống chạy trên đó.
  + Các thành phần phần mềm (Software Components): Các ứng dụng hoặc dịch vụ đang chạy trên các thiết bị này.
  + Các kết nối (Connections): Mô tả cách các thành phần phần mềm giao tiếp với nhau, sử dụng các giao thức mạng.
* Chi tiết
  + Tầng giao tiếp người dùng (USER DEVICE)
    - Buyer Interface (Web/Mobile App): Ứng dụng của người dùng để đăng nhập, tìm kiếm, phát nhạc, quản lý danh sách phát.
    - Kết nối với hệ thống qua TCP/IP.
  + Tầng xử lý chính (CLOUD SERVER - Kubernetes Cluster)
    - AUTH (Authentication Service): Xác thực người dùng (OAuth 2.0, JWT).
    - SEARCH (Search Service): Cung cấp chức năng tìm kiếm bài hát, nghệ sĩ, danh sách phát.
    - MEDIA (Media Processing Service): Chuyển đổi định dạng nhạc, tối ưu dung lượng file trước khi lưu trữ.
    - NOTIF (Notification Service): Gửi thông báo khi có lượt thích, bình luận, cập nhật mới.
    - STREAM (Streaming Service): Xử lý phát nhạc trực tuyến với CDN hỗ trợ độ trễ thấp.
    - PLAYLIST (Playlist Service): Quản lý danh sách phát, yêu thích, lịch sử nghe nhạc.
  + Tầng lưu trữ dữ liệu (STORAGE SERVER)
    - AWS S3 (Music Storage): Lưu trữ file nhạc.
    - MongoDB (User Data, Playlists): Lưu tài khoản người dùng, danh sách phát, thông tin cá nhân.
  + Kết nối giữa các thành phần
    - Người dùng giao tiếp với các dịch vụ qua TCP/IP.
    - Microservices tương tác với cơ sở dữ liệu hoặc hệ thống lưu trữ (AWS S3, MongoDB) để xử lý yêu cầu.

**3.1.2.1.2. Implement View (Thắng)**

****

* **Tổng quan**
  + Hệ thống Microservices của SoundShare được thiết kế để cung cấp nền tảng mạng xã hội chia sẻ âm nhạc, đảm bảo tính linh hoạt, khả năng mở rộng và hiệu suất cao.
  + Kiến trúc này chia hệ thống thành ba lớp chính: Frontend Layer (Giao diện người dùng), Middle Layer (Tầng dịch vụ), và Data Tier (Tầng dữ liệu).
  + Việc áp dụng mô hình Microservices giúp cải thiện khả năng quản lý, bảo trì và mở rộng từng chức năng mà không ảnh hưởng đến toàn bộ hệ thống.
* **Chi tiết về hệ thống**
  + User Interface & Frontend Layer
    - User Interface: Cung cấp giao diện người dùng cho ứng dụng web và mobile, giúp người dùng tương tác với hệ thống như nghe nhạc, tạo danh sách phát, bình luận và quản lý tài khoản.
  + Middle Layer (Tầng dịch vụ)
    - API Gateway: Là điểm truy cập duy nhất cho toàn bộ hệ thống, chịu trách nhiệm chuyển tiếp yêu cầu từ frontend đến các microservices tương ứng.
    - Authentication Service: Quản lý xác thực, phân quyền người dùng, hỗ trợ đăng nhập/đăng ký và bảo mật hệ thống.
    - User Service: Quản lý thông tin người dùng, cập nhật hồ sơ và xử lý yêu cầu liên quan đến tài khoản.
    - Music Service: Quản lý nhạc, bao gồm tải lên, phát nhạc trực tuyến, và xử lý dữ liệu liên quan đến bài hát.
    - SongService: Hỗ trợ người dùng tạo, sửa đổi, và chia sẻ danh sách phát cá nhân.
    - Comment Service: Cho phép người dùng bình luận trên bài hát hoặc danh sách phát.
    - Recommendation Service: Xử lý thuật toán đề xuất bài hát dựa trên sở thích và lịch sử nghe nhạc của người dùng.
  + Data Tier (Tầng dữ liệu)
    - Database: Lưu trữ dữ liệu người dùng, danh sách phát, bài hát, và bình luận.
    - Cache: Lưu trữ tạm thời dữ liệu được truy cập thường xuyên (ví dụ: danh sách bài hát phổ biến) để giảm tải cho cơ sở dữ liệu chính và cải thiện hiệu suất hệ thống.

**3.1.2.1.3. Work Assignment View(Trí)**

a.Định nghĩa  
Work Assignment View mô tả cách công việc được phân công giữa các nhóm phát triển và cách mỗi nhóm chịu trách nhiệm cho từng phần của hệ thống. View này giúp quản lý tiến độ dự án, tránh chồng chéo nhiệm vụ và tối ưu quy trình làm việc.

b.RACI Matrix

| Module / Thành viên | Project Manager | Frontend Dev | Backend Dev | Database Eng | DevOps |
| --- | --- | --- | --- | --- | --- |
| Giao diện người dùng | I | R/A | C | I | I |
| Song  Service | I | I | R/A | C | C |
| API gateway | I | I | R/A | C | C |
| Cơ sở dữ liệu | I | I | C | R/A | I |
| Triển khai CI/CD | I | I | C | I | R/A |

Chú thích:

* R (Responsible) – Người thực hiện chính.
* A (Accountable) – Người chịu trách nhiệm chính.
* C (Consulted) – Được tư vấn, phối hợp.
* I (Informed) – Được thông báo nhưng không tham gia trực tiếp.

Cấu trúc Nhóm Phát triển & Trách nhiệm

1. Nhóm Quản lý & Điều phối

Project Manager

* Quản lý tiến độ dự án.
* Điều phối công việc giữa các nhóm.
* Đảm bảo hệ thống tuân theo kiến trúc và yêu cầu đặt ra.

2. Nhóm Phát triển Frontend

Frontend Developer

* Phát triển giao diện người dùng với React.js / Next.js.
* Xây dựng trang chi tiết danh sách phát theo chuẩn Material Design.
* Tích hợp giao diện với API từ Backend.

3. Nhóm Phát triển Backend

Backend Developer

* Xây dựng API cho Song Service, Music Service, User Service.
* Xử lý các request như thêm/xóa bài hát, phát nhạc, chia sẻ danh sách phát.
* Đảm bảo tối ưu hiệu suất và bảo mật.

4. Nhóm Cơ sở Dữ liệu & Tối ưu Truy vấn

Database Engineer

* Thiết kế Database với Sql server.
* Xây dựng các chỉ mục (index) để tối ưu truy vấn trong hệ thống.
* Đảm bảo tính nhất quán và tính sẵn sàng cao cho dữ liệu.

5. Nhóm DevOps & Triển khai hệ thống

DevOps Engineer

* Thiết lập CI/CD pipeline để tự động build và deploy.
* Đảm bảo hệ thống chạy ổn định trên cloud (AWS/GCP).

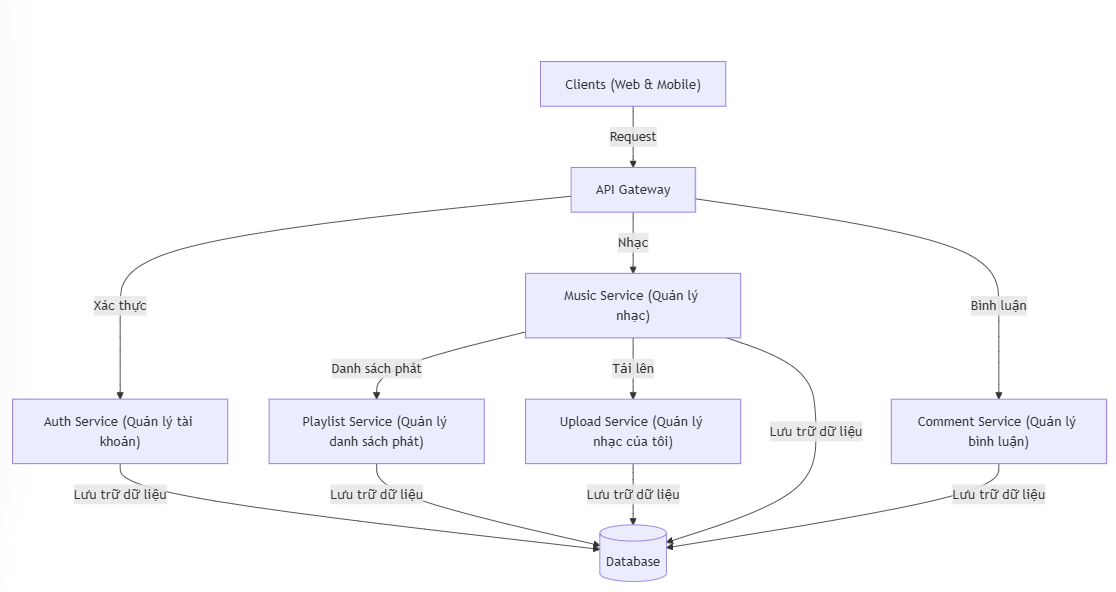
**3.1.2.2 Component and connector View**

**3.1.2.2.1.Client-Server View (Phi)**

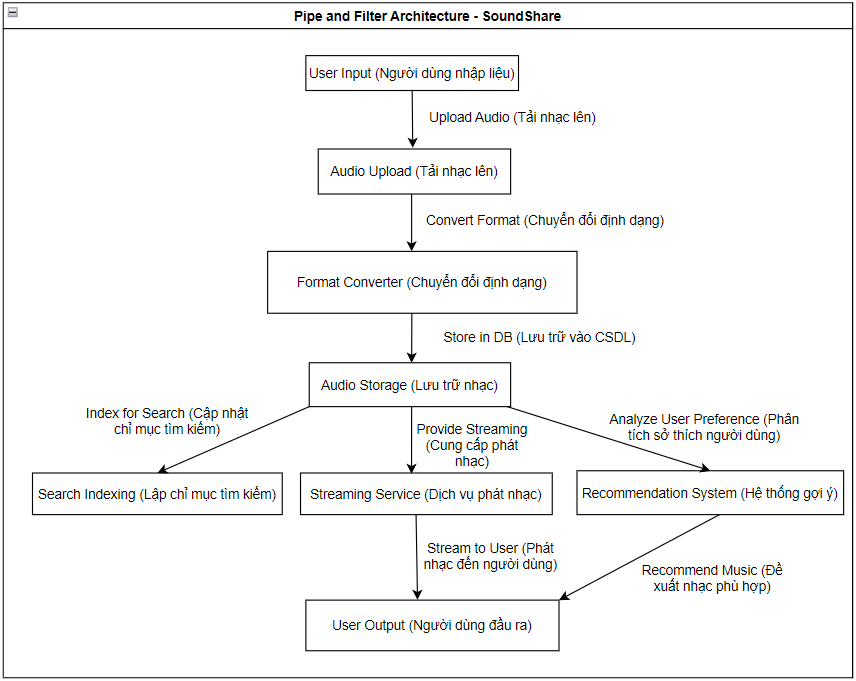
a.Định nghĩa

**Client-Server View** mô tả cách hệ thống được phân chia thành hai thành phần chính: Client (máy khách) và Server (máy chủ). View này giúp xác định cách các thành phần giao tiếp với nhau, cách dữ liệu được xử lý và truyền tải giữa client và server. Nó giúp đảm bảo hệ thống hoạt động hiệu quả, hỗ trợ mở rộng và bảo mật.

b.Biểu đồ diagram



**3.1.2.2.2.Pipe And Filter View (Sáng)**

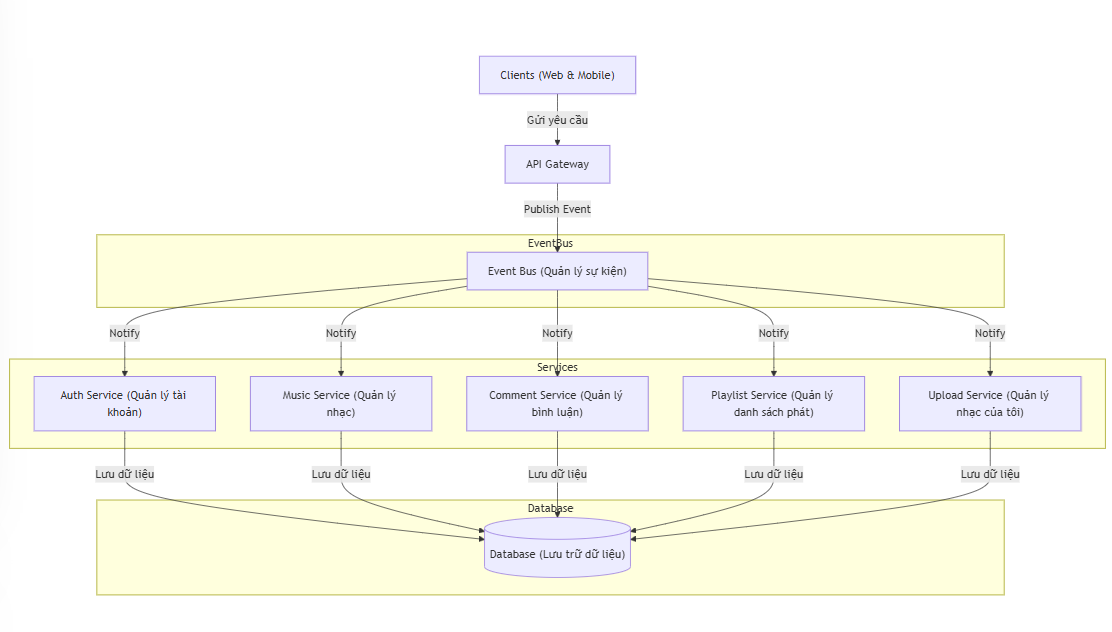


* Tổng Quan
  + Sơ đồ Pipe and Filter View của SoundShare thể hiện luồng dữ liệu từ khi người dùng tải nhạc lên hệ thống cho đến khi nhạc được lưu trữ, xử lý và phát lại cho người dùng.
  + Hệ thống được thiết kế dựa trên kiến trúc Pipe and Filter, trong đó:
    - "Pipe" (ống dẫn) giúp truyền dữ liệu từ bộ lọc này sang bộ lọc khác.
    - "Filter" (bộ lọc) xử lý dữ liệu từng bước trước khi chuyển sang bước tiếp theo.
* Chi tiết
  + User Input (Người dùng nhập liệu)
    - Đây là điểm bắt đầu của hệ thống.
    - Người dùng có thể tải nhạc lên nền tảng SoundShare.
    - Dữ liệu ban đầu là một file nhạc từ thiết bị người dùng.
  + Audio Upload (Tải nhạc lên)
    - Tiếp nhận file nhạc từ người dùng.
    - Kiểm tra tính hợp lệ của file (định dạng, dung lượng, lỗi tải lên).
    - Chuyển dữ liệu sang bước tiếp theo.
  + Format Converter (Chuyển đổi định dạng)
    - Chuyển đổi định dạng file nhạc (nếu cần) để đảm bảo tương thích với hệ thống.
    - Ví dụ: chuyển từ .wav, .flac sang .mp3 hoặc .aac để tối ưu dung lượng.
    - Sau khi xử lý xong, file nhạc được chuyển sang bộ phận lưu trữ.
  + Audio Storage (Lưu trữ nhạc)
    - Lưu trữ file nhạc vào cơ sở dữ liệu hoặc hệ thống lưu trữ cloud.
    - Đảm bảo an toàn dữ liệu và hỗ trợ truy xuất nhanh chóng.
    - Cung cấp dữ liệu cho các dịch vụ khác như Streaming Service và Search Indexing.
  + Search Indexing (Lập chỉ mục tìm kiếm)
    - Nhận dữ liệu từ Audio Storage.
    - Lập chỉ mục cho bài hát để hỗ trợ tìm kiếm nhanh chóng.
    - Người dùng có thể tìm kiếm bài hát theo tên, nghệ sĩ, album,...
  + Streaming Service (Dịch vụ phát nhạc)
    - Nhận dữ liệu từ Audio Storage.
    - Xử lý yêu cầu phát nhạc theo thời gian thực.
    - Đảm bảo tốc độ tải nhanh, giảm độ trễ khi phát nhạc.
  + Recommendation System (Hệ thống gợi ý)
    - Phân tích hành vi người dùng để gợi ý bài hát phù hợp.
    - Dựa trên lịch sử nghe nhạc, thể loại yêu thích, và xu hướng.
    - Đề xuất nhạc cho người dùng thông qua User Output.
  + User Output (Người dùng đầu ra)
    - Cung cấp nhạc đã xử lý cho người dùng.
    - Gồm hai nguồn chính:
* Streaming Service → Phát nhạc trực tiếp.
* Recommendation System → Đề xuất nhạc phù hợp.

**3.1.2.2.3.Publish-Subscribe View(phi)**

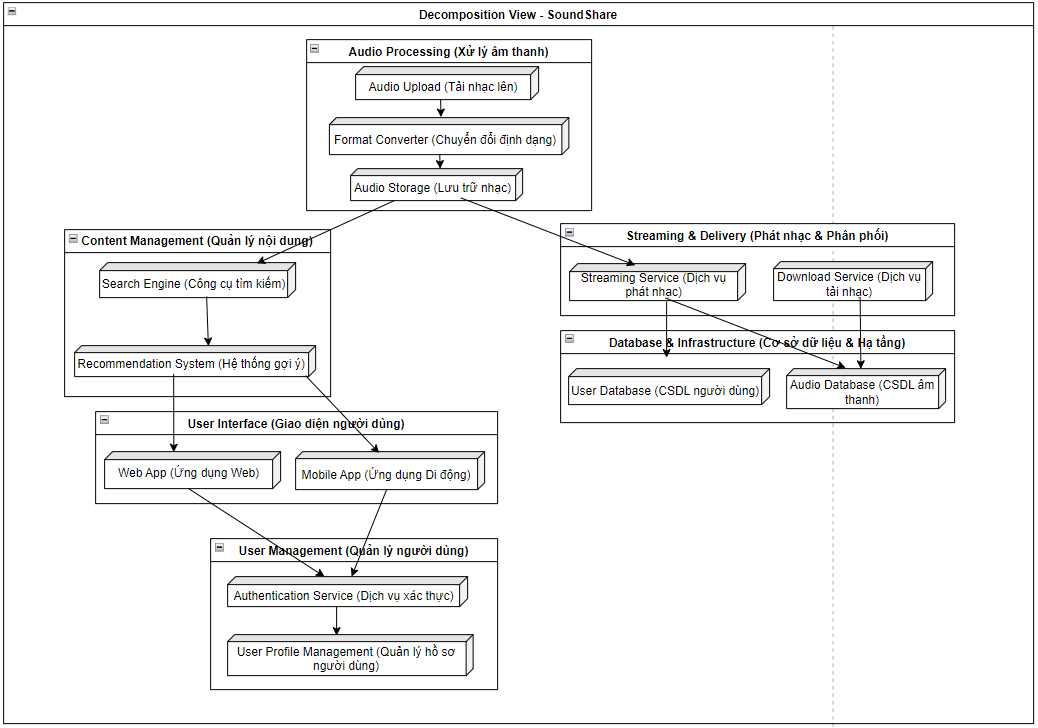
a.**Publish-Subscribe View** mô tả cách các thành phần trong hệ thống trao đổi thông tin bằng cơ chế phát (publish) sự kiện và đăng ký (subscribe) nhận sự kiện, thay vì giao tiếp trực tiếp. Mô hình này giúp hệ thống lỏng lẻo (loosely coupled), linh hoạt, dễ mở rộng và hỗ trợ xử lý sự kiện bất đồng bộ, giúp các thành phần có thể hoạt động độc lập mà không cần biết trực tiếp đến nhau.

b.Biểu đồ diagram



**3.1.2.3 Module View**

**3.1.2.3.1 Decomposition View(Sáng)**



* Tổng Quan

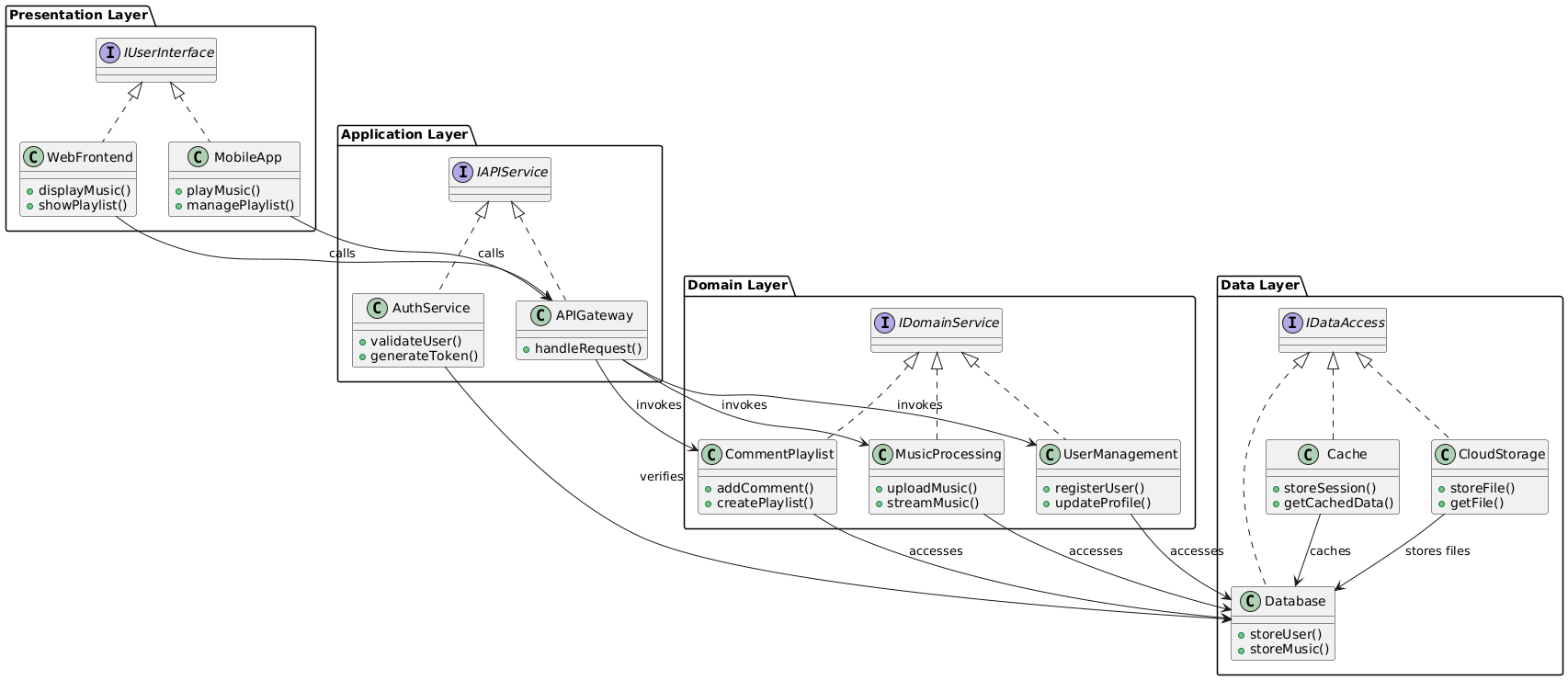
Hệ thống SoundShare được chia thành 6 phần chính, mỗi phần đảm nhận một nhiệm vụ cụ thể, giúp hệ thống dễ mở rộng, bảo trì và nâng cấp:

* + User Interface (Giao diện người dùng)
  + User Management (Quản lý người dùng)
  + Audio Processing (Xử lý âm thanh)
  + Content Management (Quản lý nội dung)
  + Streaming & Delivery (Phát nhạc & Phân phối)
  + Database & Infrastructure (Cơ sở dữ liệu & Hạ tầng)
* Chi tiết
  + User Interface (Giao diện người dùng)
    - Chức năng: Giao diện để người dùng truy cập SoundShare từ nhiều nền tảng.
    - Thành phần:
* Web App (Ứng dụng Web): Truy cập từ trình duyệt.
* Mobile App (Ứng dụng Di động): Ứng dụng chạy trên iOS/Android.
  + - Mối liên hệ:
* Liên kết với "User Management" để xử lý đăng nhập.
* Liên kết với "Content Management" để hiển thị nội dung tìm kiếm và gợi ý.
* Liên kết với "Streaming & Delivery" để phát nhạc trực tuyến hoặc tải về.
  + User Management (Quản lý người dùng)
    - Chức năng: Quản lý thông tin người dùng và xác thực danh tính..
    - Thành phần:
* Authentication Service (Dịch vụ xác thực): Xác thực đăng nhập, đăng ký.
* User Profile Management (Quản lý hồ sơ người dùng): Lưu trữ thông tin cá nhân.
  + - Mối liên hệ:
* Liên kết với "User Interface" để cho phép người dùng đăng nhập.
* Liên kết với "Database & Infrastructure" để lưu trữ dữ liệu người dùng.
  + Audio Processing (Xử lý âm thanh)
    - Chức năng: Xử lý nhạc do người dùng tải lên.
    - Thành phần:
* Audio Upload (Tải nhạc lên): Tiếp nhận file nhạc từ người dùng.
* Format Converter (Chuyển đổi định dạng): Chuyển đổi định dạng file (MP3, WAV,...).
* Audio Storage (Lưu trữ nhạc): Lưu nhạc vào cơ sở dữ liệu.
  + - Mối liên hệ:
* Liên kết với "Database & Infrastructure" để lưu trữ nhạc.
* Liên kết với "Content Management" để hỗ trợ tìm kiếm và gợi ý nhạc.
* Liên kết với "Streaming & Delivery" để phát nhạc.
  + Content Management (Quản lý nội dung)
    - Chức năng: Quản lý và xử lý nội dung để phục vụ tìm kiếm và đề xuất.
    - Thành phần:
* Search Engine (Công cụ tìm kiếm): Hỗ trợ tìm kiếm bài hát.
* Recommendation System (Hệ thống gợi ý): Đề xuất bài hát dựa trên sở thích.
  + - Mối liên hệ:
* Liên kết với "User Interface" để hiển thị kết quả tìm kiếm và đề xuất.
* Liên kết với "Streaming & Delivery" để phân tích hành vi nghe nhạc của người dùng.
* Liên kết với "Audio Processing" để lấy dữ liệu bài hát.
  + Streaming & Delivery (Phát nhạc & Phân phối)
    - Chức năng: Cung cấp dịch vụ phát nhạc trực tuyến và tải nhạc.
    - Thành phần:
* Streaming Service (Dịch vụ phát nhạc): Phát nhạc theo yêu cầu.
* Download Service (Dịch vụ tải nhạc): Cho phép người dùng tải nhạc.
  + - Mối liên hệ:
* Liên kết với "Audio Processing" để nhận dữ liệu nhạc.
* Liên kết với "Database & Infrastructure" để truy xuất dữ liệu âm thanh.
* Liên kết với "Content Management" để cung cấp dữ liệu cho hệ thống gợi ý.
  + Database & Infrastructure (Cơ sở dữ liệu & Hạ tầng)
    - Chức năng: Lưu trữ và quản lý dữ liệu hệ thống.
    - Thành phần:
* User Database (CSDL người dùng): Lưu trữ thông tin tài khoản.
* Audio Database (CSDL âm thanh): Lưu trữ dữ liệu bài hát.
  + - Mối liên hệ:
* Liên kết với "User Management" để lưu dữ liệu người dùng.
* Liên kết với "Audio Processing" để lưu nhạc tải lên.
* Liên kết với "Streaming & Delivery" để cung cấp dữ liệu khi phát nhạc.

**3.1.2.3.2 Layers View(Trí)**

a.Định nghĩa   
Layers View trong kiến trúc phần mềm là cách phân chia hệ thống thành các lớp (Layers) độc lập, mỗi lớp có trách nhiệm riêng và tương tác với lớp khác thông qua giao diện rõ ràng.

b.Biểu đồ diagram



2. Mô tả từng lớp trong kiến trúc

Presentation Layer (Lớp Giao Diện)

Chức năng chính:

* Hiển thị giao diện Trang Chi Tiết Danh Sách Phát, Trang Phát Nhạc, Trang Quản Lý Playlist.
* Nhận input từ người dùng và gửi request đến Application Layer.

Application Layer (Lớp Ứng Dụng)

Chức năng chính:

* Cung cấp các giao tiếp giữa Presentation Layer và các dịch vụ backend.
* Xử lý request từ Frontend như: Lấy danh sách phát, phát nhạc, thêm/xóa bài hát.
* Giao tiếp với Domain Layer để thực hiện logic nghiệp vụ.

Domain Layer (Lớp Miền Nghiệp Vụ)

Chức năng chính:

* Chứa logic nghiệp vụ quan trọng như quản lý danh sách phát, phát nhạc, xác thực người dùng.
* Thực hiện kiểm tra dữ liệu trước khi lưu vào Database..

Data Layer (Lớp Dữ Liệu)

Chức năng chính:

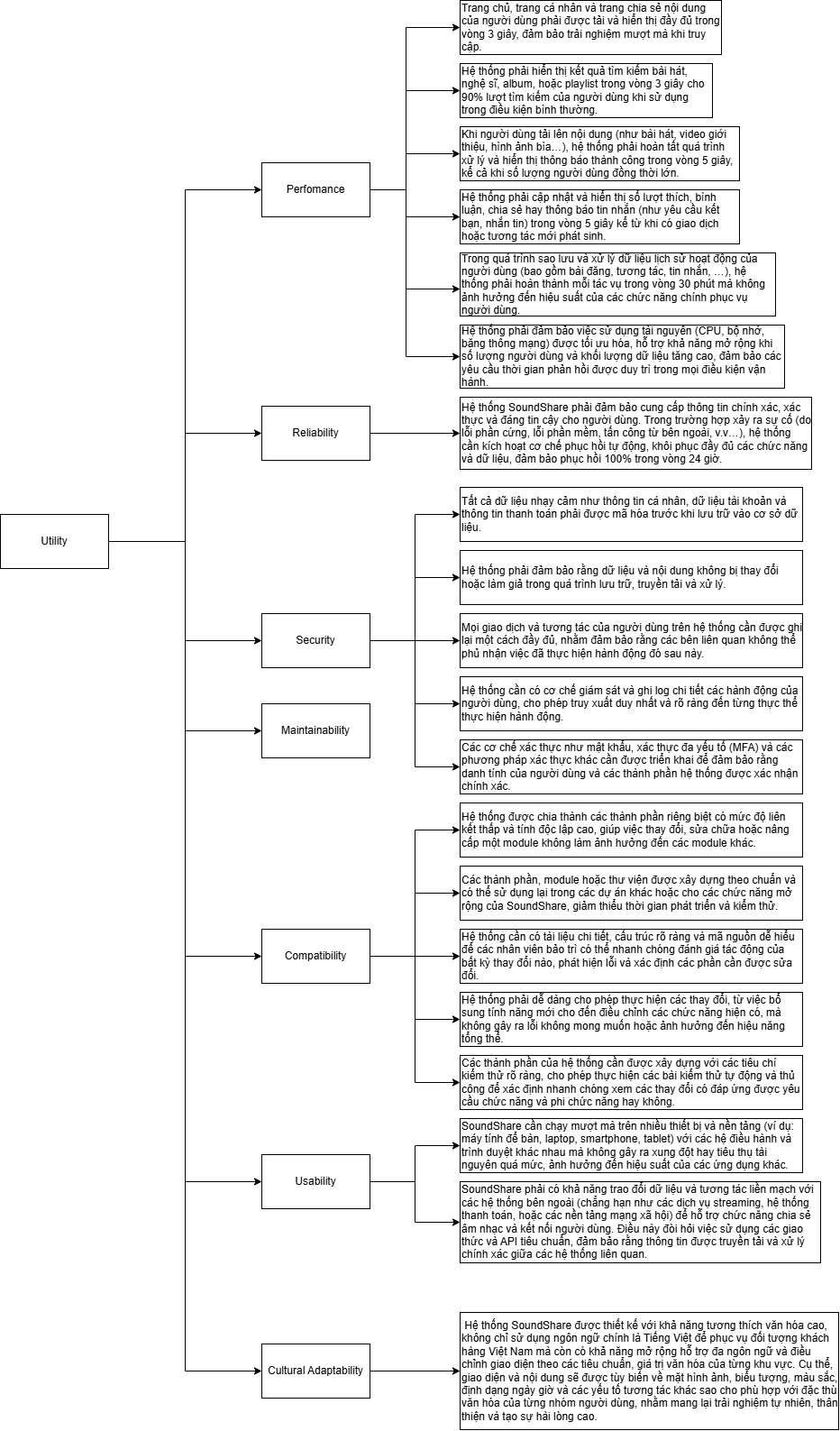
* Lưu trữ dữ liệu danh sách phát, bài hát, người dùng.
* Truy vấn và cập nhật dữ liệu khi có yêu cầu từ Domain Layer.

## Lý do lựa chọn kiến trúc phần mềm

Phân tích các lý do dẫn đến kiến trúc phần mềm (trong mục 3.1) được chọn (và kiến trúc khác không được chọn)

**3.2.1 Lý do lựa chọn kiến trúc (Phương)**

* Sơ đồ Utility tree



* Với sự ưu tiên về các NFR được thể hiện trong sơ đồ utility tree, kiến trúc microservice có khả năng đáp ứng ở mức tương đối tốt:
  + Performance: Tách riêng các dịch vụ quan trọng (Service tìm kiếm, Service phát nhạc, Service tải nhạcI). Sử dụng Load Balancer & Auto-Scaling để xử lý lưu lượng lớn. Cache dữ liệu & tối ưu truy vấn (Redis, Elasticsearch, CDN). Xử lý bất đồng bộ với Message Queue (Kafka, RabbitMQ) để giảm tải cho hệ thống chính.
  + Reliability: Mỗi dịch vụ chạy độc lập, giảm ảnh hưởng khi 1 dịch vụ gặp lỗi. Cơ chế Failover & Auto-recovery giúp tự động phục hồi khi có sự cố. Sao lưu dữ liệu & Database Replication, đảm bảo phục hồi 100% dữ liệu trong vòng 24 giờ.
  + Mỗi dịch vụ có quyền hạn riêng, tránh rò rỉ dữ liệu toàn hệ thống. Bảo mật API với OAuth2 & JWT, hạn chế truy cập trái phép. Mã hóa dữ liệu nhạy cảm (AES-256, TLS/SSL). Dùng API Gateway để kiểm soát và giám sát các request. Bảo vệ chống SQL Injection, XSS, CSRF với WAF & Security Middleware.
* Hệ thống yêu cầu khả năng chịu tải cao và tương thích đa thiết bị. Microservices cho phép chia nhỏ hệ thống thành các dịch vụ độc lập
* Hệ thống mạng xã hội chia sẻ âm nhạc có thể có lượng người dùng lớn, microservices cho phép mở rộng từng dịch vụ riêng lẻ
* Với yêu cầu hoạt động trên nhiều thiết bị, microservices có thể cung cấp API linh hoạt để tích hợp với các nền tảng khác nhau.
* Microservices hỗ trợ sử dụng các công nghệ khác nhau cho từng dịch vụ (Node.js, Python, Go), giúp tận dụng lợi thế công nghệ phù hợp với từng phần hệ thống.
* Các dịch vụ có thể được triển khai trên nhiều server/container khác nhau, tối ưu hiệu suất theo nhu cầu thực tế (ví dụ: dịch vụ streaming nhạc có thể yêu cầu tài nguyên cao hơn so với dịch vụ quản lý bình luận).

**3.2.2 Lý do không lựa chọn các kiến trúc khác (Thắng)**

| **Kiến trúc** | **Mô tả** | **Ưu điểm** | **Nhược điểm** | **Mức độ phù hợp với dự án** |
| --- | --- | --- | --- | --- |
| Monolithic (Kiến trúc đơn khối) | Tất cả các thành phần (UI, xử lý logic, cơ sở dữ liệu) được tích hợp vào một hệ thống duy nhất. | - Dễ triển khai ban đầu, không cần nhiều công cụ quản lý phức tạp.  - Dữ liệu tập trung, không cần xử lý giao tiếp phức tạp giữa các dịch vụ. | - Khó mở rộng, khi số lượng người dùng tăng thì toàn bộ hệ thống phải nâng cấp đồng thời.  - Khi sửa đổi một tính năng, cần deploy lại toàn bộ ứng dụng.  - Dễ bị ảnh hưởng khi lỗi xảy ra, gây sập toàn bộ hệ thống. | Không phù hợp – Vì SoundShare cần khả năng mở rộng và hiệu suất cao khi số lượng người dùng lớn. |
| Layered Architecture (Kiến trúc phân lớp) | Hệ thống được chia thành các lớp (UI, Business Logic, Data), giúp tách biệt xử lý và bảo trì dễ dàng hơn so với Monolithic. | - Có tính tổ chức rõ ràng, mỗi lớp thực hiện một nhiệm vụ cụ thể.  - Dễ bảo trì hơn Monolithic vì chỉ cần thay đổi lớp liên quan.  - Phù hợp với hệ thống quy mô vừa và nhỏ. | - Vẫn có sự phụ thuộc giữa các lớp, khi một lớp thay đổi có thể ảnh hưởng toàn hệ thống.  - Không dễ mở rộng theo chiều ngang như Microservices. | Không phù hợp – Dù dễ quản lý hơn Monolithic nhưng không đủ linh hoạt và khả năng chịu tải tốt như Microservices. |
| Microservices (Kiến trúc dịch vụ nhỏ) | Hệ thống được chia thành nhiều dịch vụ nhỏ, mỗi dịch vụ có thể hoạt động độc lập và giao tiếp với nhau qua API. | - Mở rộng dễ dàng theo từng dịch vụ (ví dụ: chỉ mở rộng Streaming Service khi cần).  - Mỗi phần có thể được phát triển, deploy, cập nhật độc lập.  - Giảm thiểu lỗi ảnh hưởng toàn hệ thống, một dịch vụ gặp sự cố không ảnh hưởng đến các dịch vụ khác.  - Dễ tích hợp công nghệ mới như AI gợi ý nhạc. | - Đòi hỏi kỹ thuật cao, cần có API Gateway để quản lý giao tiếp giữa các dịch vụ.  - Quản lý dữ liệu phức tạp hơn vì mỗi dịch vụ có cơ sở dữ liệu riêng. | Phù hợp – Vì đáp ứng tốt yêu cầu mở rộng, hiệu suất, bảo mật của SoundShare. |

## Chứng minh khả năng đáp ứng của KTPM với các NFR

**3.3.1 Độ tin cậy(Phi)**

#### Cung cấp thông tin chính xác, xác thực và đáng tin cậy

* Dữ liệu phân tán và nhất quán:
  + Áp dụng Event Sourcing và CQRS để tách biệt ghi và đọc dữ liệu, đảm bảo tính nhất quán trên toàn hệ thống.
  + Sử dụng Distributed Transactions (Saga Pattern) để đảm bảo dữ liệu không bị mất hoặc sai lệch trong các giao dịch phức tạp.
* Cơ chế xác thực và bảo vệ dữ liệu:
  + Sử dụng OAuth 2.0, JWT để xác thực người dùng an toàn.
  + Mã hóa dữ liệu nhạy cảm bằng AES-256 để đảm bảo bảo mật.
* Log & Monitoring:
  + Áp dụng ELK Stack (Elasticsearch, Logstash, Kibana), Prometheus, Grafana để giám sát toàn bộ hệ thống và phát hiện bất thường.

#### Cơ chế phục hồi tự động khi xảy ra sự cố

* Auto-Scaling & Load Balancing:
  + Khi một microservice gặp sự cố, hệ thống sẽ tự động scale lên các instance mới để thay thế.
  + Sử dụng API Gateway (Kong, Nginx, AWS API Gateway) để phân phối tải và tránh tình trạng quá tải cục bộ.
* Kubernetes Auto-Healing:
  + Khi một container gặp lỗi, Kubernetes sẽ tự động restart hoặc khởi tạo container mới để đảm bảo dịch vụ luôn chạy.
* Circuit Breaker & Retry Mechanism:
  + Áp dụng Resilience4j hoặc Hystrix để cô lập các service gặp lỗi, ngăn chặn lỗi lan rộng ra toàn hệ thống.
  + Retry Policy đảm bảo các request bị lỗi có thể thử lại trước khi chuyển sang trạng thái thất bại.

#### Khôi phục đầy đủ các chức năng và dữ liệu trong vòng 24 giờ

* Cơ chế backup và snapshot dữ liệu:
  + Dữ liệu được tự động sao lưu định kỳ trên AWS S3, Google Cloud Storage, với cơ chế point-in-time recovery giúp khôi phục về trạng thái gần nhất trước khi xảy ra sự cố.
* Message Queue (Kafka, RabbitMQ):
  + Các sự kiện quan trọng được lưu trữ trong hàng đợi tin nhắn, đảm bảo không bị mất dữ liệu khi một service bị lỗi.
* Event-Driven Architecture:
  + Lưu trữ toàn bộ các sự kiện của hệ thống giúp dễ dàng khôi phục trạng thái trước khi xảy ra sự cố.
* Blue-Green Deployment & Canary Release:
  + Hỗ trợ cập nhật hệ thống mà không gây gián đoạn, đảm bảo thời gian khôi phục nhanh chóng mà không ảnh hưởng đến trải nghiệm người dùng.

**3.3.2 Hiệu năng (Phương)**

* Thời gian phản hồi khi tìm kiếm nội dung âm nhạc (<3 giây cho 90% lượt tìm kiếm)
  + Tách riêng dịch vụ tìm kiếm: Một Search Service độc lập được tối ưu hóa chỉ để xử lý tìm kiếm nhạc, nghệ sĩ, album, playlist.
  + Index dữ liệu giúp truy vấn nhanh hơn so với tìm kiếm trong database truyền thống.
  + Kết quả tìm kiếm phổ biến có thể được cache để giảm tải cho database.
  + Khi có lượng lớn truy vấn tìm kiếm, service có thể tự động scale để duy trì hiệu suất.
* Thời gian tải trang giao diện người dùng (<3 giây)
  + Tối ưu API Gateway: API Gateway giúp gộp nhiều request cần thiết vào một response duy nhất, giảm số lần truy cập backend.
  + Sử dụng CDN (Content Delivery Network): Phân phối các tài nguyên tĩnh (hình ảnh, CSS, JavaScript) trên nhiều máy chủ toàn cầu giúp tải trang nhanh hơn.
  + Cải thiện hiệu suất frontend bằng cách chỉ tải những nội dung cần thiết.
  + Cho phép client yêu cầu đúng những dữ liệu cần, giảm thiểu dữ liệu dư thừa.
* Thời gian xử lý và đăng tải nội dung (<5 giây)
  + Sau khi nội dung được tải lên, hệ thống có thể lưu trữ trên Amazon S3, Google Cloud Storage để truy xuất nhanh hơn.
  + Một Media Processing Service riêng biệt có thể thực hiện transcoding, resizing, giúp giảm tải cho hệ thống chính.
* Cập nhật theo thời gian thực các tương tác (cập nhật trong vòng 5 giây)
  + Sử dụng WebSocket: Cung cấp cập nhật thời gian thực cho thông báo, bình luận, lượt thích mà không cần polling liên tục.
  + Khi có một sự kiện (ví dụ: người dùng thích bài hát), sự kiện đó sẽ được đưa ngay đến Notification Service hoặc Realtime Service để xử lý nhanh chóng.
  + Giúp đồng bộ dữ liệu nhanh giữa các service.
* Xử lý và lưu trữ dữ liệu lịch sử hoạt động (hoàn thành trong 30 phút mà không ảnh hưởng hiệu suất)
  + Background Processing: Các tác vụ nặng như sao lưu dữ liệu, phân tích lịch sử được chuyển sang backgroundworker để tránh làm chậm hệ thống chính.
  + Dữ liệu lịch sử có thể được chia nhỏ theo ngày/tháng/năm để truy xuất nhanh hơn.
  + Dữ liệu ít truy cập có thể được chuyển sang Google BigQuery, AWS Glacier để tối ưu hiệu suất.
* Tối ưu hóa sử dụng tài nguyên
  + Auto-Scaling với Kubernetes/Docker: Khi có nhiều người dùng truy cập, hệ thống có thể tự động mở rộng số lượng container để đảm bảo hiệu suất.
  + Ngăn chặn các request quá tải từ một nguồn duy nhất để bảo vệ tài nguyên hệ thống.
  + Giảm bớt các tác vụ đồng bộ bằng cách sử dụng các event (Kafka, RabbitMQ), giúp phân bổ tài nguyên hiệu quả.

**3.3.3 Bảo mật (Trí)**

* Bảo mật xác thực và phân quyền (Authentication & Authorization)
  + Sử dụng OAuth 2.0 và JWT (JSON Web Token) để xác thực người dùng.
  + Mã hóa mật khẩu bằng bcrypt trước khi lưu vào cơ sở dữ liệu.
  + Phân quyền theo vai trò (Role-Based Access Control - RBAC)
* Bảo vệ dữ liệu người dùng
  + Mã hóa dữ liệu quan trọng như email, thông tin thanh toán bằng AES-256.
  + Sử dụng HTTPS/TLS để mã hóa dữ liệu khi truyền tải.
  + Không lưu thông tin nhạy cảm như mật khẩu hoặc token trong local storage (đối với frontend).
* Bảo vệ hệ thống khỏi tấn công
  + Chống SQL Injection: Sử dụng ORM (Entity Framework) thay vì query thô.
  + Chống Cross-Site Scripting (XSS):Escape đầu ra (output escaping) để ngăn chặn mã độc được chèn vào giao diện.
  + Chống Cross-Site Request Forgery (CSRF):Dùng anti-CSRF token trong mọi request quan trọng.
  + Chống DDoS:Rate Limiting với API Gateway (ví dụ: nginx, Cloudflare).
* Bảo mật lưu trữ file nhạc
  + Các file nhạc được lưu trữ trên AWS S3 hoặc Google Cloud Storage với quyền truy cập hạn chế.
  + Sử dụng Signed URLs để chỉ cho phép người dùng hợp lệ tải nhạc.

**3.3.4 Hữu dụng(Sáng)**

* Giao diện trực quan và dễ sử dụng
  + Thiết kế UI/UX thân thiện với thanh điều hướng rõ ràng, dễ thao tác trên cả nền tảng web và mobile.
  + Các thao tác chính như nghe nhạc, tìm kiếm, chia sẻ danh sách phát được bố trí hợp lý, không gây khó khăn cho người dùng.
  + Hỗ trợ hướng dẫn sử dụng chi tiết cho người mới.
* Dễ dàng học hỏi và làm quen
  + Người dùng không cần mất nhiều thời gian để làm quen với các chức năng của hệ thống nhờ giao diện gọn gàng, rõ ràng.
  + Cung cấp hướng dẫn từng bước (onboarding) giúp người dùng mới tiếp cận hệ thống nhanh chóng.
* Hỗ trợ thao tác nhanh chóng và chính xác
  + Chức năng tìm kiếm nhạc nhanh, có gợi ý thông minh giúp người dùng tìm kiếm nội dung nhanh hơn.
  + Hệ thống phản hồi nhanh khi nghe nhạc, tạo danh sách phát, tải nhạc xuống giúp trải nghiệm mượt mà.
* Bảo vệ người dùng khỏi lỗi thao tác
  + Hiển thị cảnh báo lỗi khi nhập thông tin sai (ví dụ: đăng ký tài khoản thiếu dữ liệu, nhập sai mật khẩu).
  + Tích hợp chức năng tự động lưu tạm thời các thao tác chưa hoàn thành (ví dụ: nhập bình luận nhưng chưa gửi, đang tạo danh sách phát nhưng chưa lưu).
* Hỗ trợ đa nền tảng và khả năng truy cập tốt
  + Hoạt động tốt trên cả máy tính, điện thoại và tablet, tự động điều chỉnh giao diện phù hợp với kích thước màn hình.
  + Tương thích với nhiều trình duyệt như Chrome, Firefox, Safari, Edge.
  + Hỗ trợ phím tắt cho các thao tác quan trọng giúp người dùng thao tác nhanh hơn.

**3.3.5 Tính bảo trì (Phương)**

* Tính mô-đun (Modularity)
  + Mỗi service hoạt động độc lập, có thể thay đổi hoặc nâng cấp mà không ảnh hưởng toàn bộ hệ thống.
  + Các dịch vụ giao tiếp qua API, giảm sự phụ thuộc chặt chẽ giữa các module.
* Tính tái sử dụng (Reusability)
  + Các microservices có thể tái sử dụng trong nhiều hệ thống khác nhau (ví dụ: Authentication Service dùng chung cho nhiều ứng dụng).
  + Sử dụng thư viện chung và API tiêu chuẩn giúp mở rộng hệ thống dễ dàng.
* Khả năng phân tích (Analysability)
  + Logging & Monitoring với ELK Stack, Prometheus, Grafana giúp phân tích lỗi nhanh chóng.
  + Cấu trúc RESTful API + Documentation (Swagger, OpenAPI) giúp dev dễ hiểu và bảo trì.
* Khả năng sửa đổi (Modifiability)
  + Có thể thêm/bỏ tính năng mà không ảnh hưởng đến hệ thống nhờ kiến trúc tách biệt.
  + Dễ dàng nâng cấp từng service độc lập mà không gây downtime.
* Khả năng kiểm thử (Testability)
  + Unit Test, Integration Test, API Test trên từng service giúp phát hiện lỗi sớm.
  + CI/CD pipelines tự động kiểm thử và triển khai giúp đảm bảo chất lượng code trước khi đưa vào production.

**3.3.6 Tính tương thích(Thắng)**

* Tích hợp với các hệ thống khác
  + Hỗ trợ tích hợp với các nền tảng lưu trữ đám mây như AWS S3 và Google Cloud Storage để lưu trữ file nhạc an toàn và hiệu quả.
  + Sử dụng OAuth 2.0 để tích hợp đăng nhập qua tài khoản mạng xã hội (Google, Facebook) mà không cần phát triển hệ thống xác thực riêng.
* Hoạt động trên nhiều nền tảng và thiết bị
  + Hệ thống tương thích với các thiết bị như máy tính, điện thoại, và tablet, tự động điều chỉnh giao diện (responsive design) để phù hợp với mọi kích thước màn hình.
  + Đảm bảo hoạt động mượt mà trên các hệ điều hành phổ biến: Windows, macOS, iOS, và Android.
* Hỗ trợ đa trình duyệt
  + Tương thích với các trình duyệt phổ biến như Chrome, Firefox, Safari, Edge, đảm bảo giao diện và chức năng hoạt động đồng nhất.
  + Hỗ trợ các phiên bản trình duyệt cũ (trong phạm vi hợp lý) để tăng khả năng tiếp cận.
* Khả năng tương thích với các chuẩn công nghệ
  + Sử dụng RESTful API theo chuẩn OpenAPI để đảm bảo tích hợp dễ dàng với các hệ thống khác.
  + Hỗ trợ HTTPS/TLS để mã hóa dữ liệu truyền tải, tuân thủ các tiêu chuẩn bảo mật hiện đại.
* Hỗ trợ tích hợp với các dịch vụ bên thứ ba
  + Tích hợp với Cloudflare hoặc nginx để triển khai Rate Limiting, chống tấn công DDoS.
  + Hỗ trợ Signed URLs từ AWS S3/Google Cloud để quản lý quyền truy cập file nhạc, đảm bảo tính tương thích với các dịch vụ lưu trữ.

**3.3.7 Tính văn hóa(Sáng)**

* Hỗ trợ ngôn ngữ Tiếng Việt làm ngôn ngữ chính
  + Vì đối tượng mục tiêu ban đầu là người dùng Việt Nam, giao diện và nội dung của hệ thống được thiết kế hoàn toàn bằng Tiếng Việt.
  + Có thể dễ dàng mở rộng hỗ trợ đa ngôn ngữ khi hệ thống phát triển ra thị trường quốc tế.
* Định dạng ngày tháng, đơn vị tiền tệ theo khu vực Việt Nam
  + Định dạng ngày tháng theo chuẩn DD/MM/YYYY phù hợp với người dùng Việt.
  + Đơn vị tiền tệ hiển thị theo VND nếu có giao dịch thanh toán.
* Giao diện phù hợp với thị hiếu người dùng Việt
  + Thiết kế theo phong cách hiện đại, đơn giản nhưng trực quan, tránh sử dụng các biểu tượng không quen thuộc với người Việt.
  + Màu sắc trung tính, dễ chịu, phù hợp với thị hiếu thị trường Việt Nam..
* Hỗ trợ nội dung nhạc theo khu vực
  + Gợi ý các bài hát phổ biến dựa trên xu hướng nghe nhạc tại Việt Nam.
  + Các playlist đề xuất bao gồm các bảng xếp hạng nhạc Việt, nhạc US-UK, Kpop theo sở thích của người dùng.
* Hỗ trợ nội dung phản ánh văn hóa địa phương
  + Gợi ý người dùng tạo playlist và chia sẻ nhạc dân ca, nhạc truyền thống, giúp bảo tồn và quảng bá văn hóa âm nhạc Việt Nam.
  + Có tính năng gợi ý nhạc theo sự kiện đặc biệt như Tết Nguyên Đán, Trung Thu.

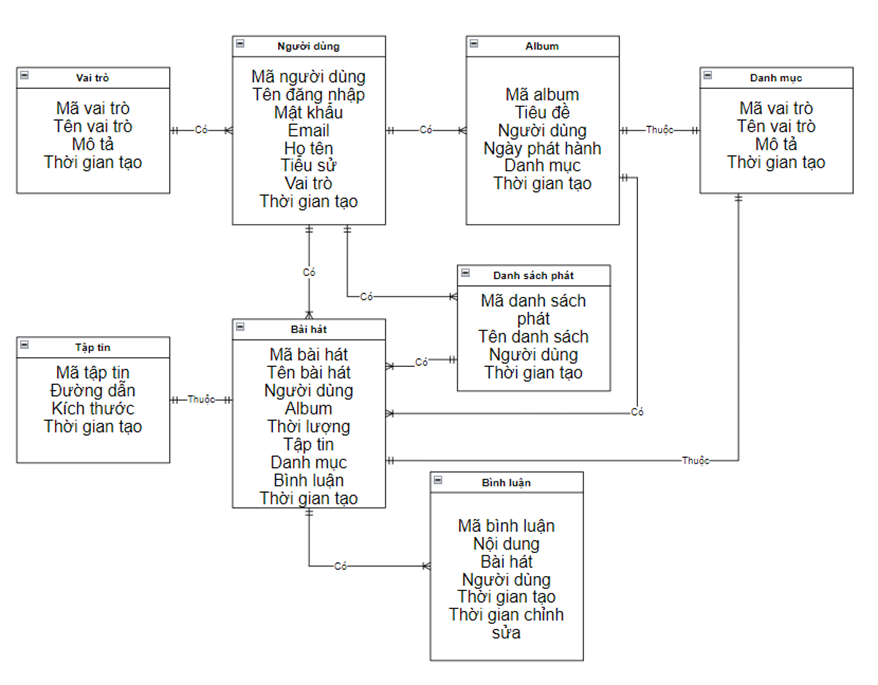
# CHƯƠNG 4: THIẾT KẾ PHẦN MỀM

## Thiết kế Dữ liệu

**4.1.1. Bảng dữ liệu (Thắng)**

* **Vai trò** (Mã vai trò, Tên vai trò, Mô tả, Thời gian tạo)
* **Người dùng** (Mã người dùng, Tên đăng nhập, Mật khẩu, Email, Họ tên, Tiểu sử, Vai trò, Thời gian tạo)
* **Danh mục** (Mã danh mục, Tên danh mục, Thời gian tạo)
* **Tập tin** (Mã tập tin, Đường dẫn, Kích thước, Thời gian tạo)
* **Album** (Mã album, Tiêu đề, Người dùng, Ngày phát hành, Danh mục, Thời gian tạo)
* **Bài hát** (Mã bài hát, Tên bài hát, Người dùng, Album, Thời lượng, Tập tin, Danh mục, Bình luận, Thời gian tạo)
* **Bình luận** (Mã bình luận, Nội dung, Bài hát, Người dùng, Thời gian tạo, Thời gian chỉnh sửa)
* **Danh sách phát** (Mã danh sách phát, Tên danh sách, Người dùng, Thời gian tạo)

**4.1.2. Sơ đồ ERD (Thắng)**

****

## Thiết kế Giao diện

### Chuẩn giao diện

* + 1. **Định nghĩa**

Chuẩn giao diện (User Interface Standard) trong thiết kế phần mềm là tập hợp các quy tắc, hướng dẫn và tiêu chuẩn nhằm đảm bảo giao diện người dùng (UI) có tính nhất quán, dễ sử dụng, dễ truy cập và tối ưu trải nghiệm người dùng (UX).

Theo tiêu chuẩn IEEE Std 1063-2001 (Software User Documentation), một giao diện người dùng cần đáp ứng các tiêu chí sau:

* Dễ học (Learnability): Người dùng có thể dễ dàng hiểu cách sử dụng hệ thống mà không cần nhiều hướng dẫn.
* Hiệu quả (Efficiency): Giúp người dùng hoàn thành tác vụ nhanh chóng và chính xác.
* Tính dễ nhớ (Memorability): Người dùng có thể quay lại sử dụng sau một thời gian mà không cần học lại.
* Tính linh hoạt (Flexibility): Hỗ trợ nhiều phong cách tương tác khác nhau (bàn phím, chuột, cảm ứng…).
* Tính nhất quán (Consistency): Các thành phần UI tuân theo một thiết kế thống nhất, giúp người dùng không bị nhầm lẫn.
* Tính thẩm mỹ (Aesthetic and Minimalist Design): Giao diện không nên có quá nhiều thông tin dư thừa gây rối mắt.
* Khả năng phản hồi (Feedback & Responsiveness): Hệ thống phải đưa ra phản hồi nhanh chóng cho hành động của người dùng.
* Khả năng truy cập (Accessibility): Đảm bảo giao diện phù hợp với mọi đối tượng, bao gồm cả người khuyết tật.
  + 1. **Các nguyên tắc**
* Nguyên tắc về Tính Nhất Quán

Sử dụng cùng một bộ màu sắc, biểu tượng, bố cục trên toàn bộ hệ thống.

Kiểu chữ, kích thước chữ và khoảng cách giữa các thành phần UI phải đồng nhất.

Hành vi của các nút bấm, menu, thanh điều hướng phải đồng bộ trên mọi trang.

* Nguyên tắc về Khả Năng Truy Cập (Accessibility)

Tuân theo tiêu chuẩn IEEE 829-1998 để đảm bảo người khiếm thị, khiếm thính có thể sử dụng giao diện.

Cung cấp tùy chọn thay đổi kích thước chữ, độ tương phản màu sắc.

Hỗ trợ điều hướng bằng bàn phím và đọc màn hình (screen reader).

* Nguyên tắc về Tương Tác Người Dùng

Theo IEEE 1471-2000, giao diện cần cung cấp phản hồi trực quan khi người dùng thực hiện thao tác.

Trạng thái của hệ thống luôn hiển thị rõ ràng (ví dụ: tiến trình tải, lỗi, thành công).

Cung cấp chức năng “Hoàn tác” (Undo) và xác nhận trước khi thực hiện hành động quan trọng.

* Nguyên tắc về Hiệu Suất và Phản Hồi (Responsiveness)

Tất cả thao tác của người dùng phải có phản hồi trong vòng 100ms để tạo cảm giác mượt mà.

Nếu cần thời gian xử lý lâu hơn, hệ thống phải hiển thị thanh tiến trình hoặc thông báo đang xử lý.

UI phải tối ưu trên nhiều kích thước màn hình và thiết bị khác nhau (desktop, tablet, mobile).

* Nguyên tắc về Đơn Giản và Tối Giản (Simplicity & Minimalism)

Tránh hiển thị quá nhiều thông tin trên một màn hình.

Chỉ giữ lại các yếu tố quan trọng, loại bỏ những yếu tố không cần thiết.

Sử dụng khoảng trắng hợp lý để tăng khả năng đọc và tập trung.

* Nguyên tắc về An Toàn và Bảo Mật (Security & Privacy)

Tuân theo tiêu chuẩn IEEE 12207 để đảm bảo dữ liệu cá nhân của người dùng được bảo vệ.

Không lưu trữ thông tin nhạy cảm trên giao diện trừ khi cần thiết.

Yêu cầu xác thực khi thực hiện các hành động quan trọng như thay đổi mật khẩu, thanh toán.

### Đặc tả Giao diện Trang chủ (Thắng)

* + 1. **Mô tả chung**
* Wireframe này là giao diện cơ bản của một ứng dụng hoặc trang web quản lý nhạc, chia thành hai phần: menu điều hướng bên trái và nội dung chính bên phải.
* Phần menu bao gồm các nút chính như Logo, Trang chủ, Danh sách phát, và Quản lý tài khoản. Phần nội dung hiển thị ô tìm kiếm, nút Tài khoản, cùng danh sách bài hát và album được sắp xếp thành hai cột, hỗ trợ người dùng duyệt và tương tác với nội dung.
  + 1. **Các chức năng chính**
* Tìm kiếm: Ô "Tìm kiếm" cho phép người dùng nhập từ khóa để lọc bài hát hoặc album.
* Xem danh sách: Hiển thị danh sách bài hát (4 mục cột trái, 5 mục cột phải) và album (4 mục mỗi cột) để chọn nghe hoặc xem chi tiết.
* Quản lý tài khoản: Nút "Tài khoản" mở menu hoặc trang quản lý thông tin cá nhân.
* Điều hướng: Nút "Trang chủ" và "Danh sách phát" chuyển hướng đến các giao diện tương ứng.
  + 1. **Wireframe**

### 

### 

* 1. **Đặc tả Giao diện Trang Nghe nhạc (Phi)**
     1. **Mô tả chung**
* Phát bài nhạc mà người dùng chọn, người dùng có thể chuyển bài, tua thời gian bài hát…
* Hiển thị danh sách nhạc mà người dùng tải lên, có nút thêm nhạc cho người dùng có thể tải nhạc của mình lên
* Có thanh các nút control để điều hướng qua các trang khác(trang chủ, trang danh sách phát …)
  + 1. **Wireframe**

****

* 1. **Đặc tả Giao diện Trang Quản lý danh sách phát (Sáng)**

4.2.4.1 Mô tả chung

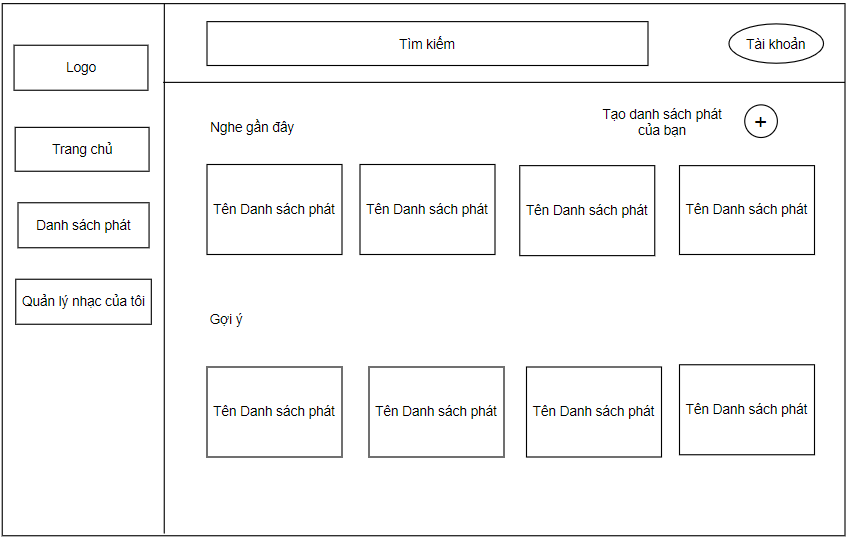
Giao diện Trang Quản lý danh sách phát gồm ba phần chính:

* Thanh điều hướng bên trái: Chứa logo và các nút chuyển trang như Trang chủ, Danh sách phát, Quản lý nhạc.
* Thanh tìm kiếm & tài khoản: Gồm ô tìm kiếm nội dung và nút tài khoản cá nhân.
* Khu vực danh sách phát: Hiển thị danh sách phát Nghe gần đây, Gợi ý và có nút "+" để tạo danh sách phát mới.

4.2.4.2 Đặc tả các chức năng chính của giao diện:

* Chuyển trang điều hướng:
* Khi nhấn vào các nút trong thanh điều hướng, người dùng sẽ được chuyển đến trang tương ứng.
* "Trang chủ" đưa người dùng về giao diện chính của ứng dụng.
* "Danh sách phát" hiển thị tất cả danh sách phát hiện có của người dùng.
* "Quản lý nhạc của tôi" giúp truy cập vào bộ sưu tập nhạc cá nhân.
* Tìm kiếm nội dung:
* Khi nhập từ khóa vào ô tìm kiếm, hệ thống sẽ hiển thị danh sách các kết quả phù hợp, có thể bao gồm bài hát, danh sách phát hoặc nghệ sĩ.
* Quản lý danh sách phát cá nhân:
* Hiển thị danh sách phát mà người dùng đã tạo hoặc đã nghe gần đây.
* Người dùng có thể chọn danh sách phát để phát nhạc, chỉnh sửa hoặc xóa.
* Tạo danh sách phát mới:
* Khi nhấn vào biểu tượng "+" bên cạnh "Tạo danh sách phát của bạn", hệ thống sẽ mở giao diện tạo danh sách phát mới, nơi người dùng có thể đặt tên và thêm bài hát
* Xem danh sách phát gợi ý:
* Hệ thống sẽ hiển thị các danh sách phát được đề xuất dựa trên lịch sử nghe nhạc của người dùng hoặc xu hướng phổ biến.

4.2.4.3 Wireframe Giao Diện

****

* 1. **Đặc tả Giao diện Trang Chi tiết danh sách phát (Trí)**

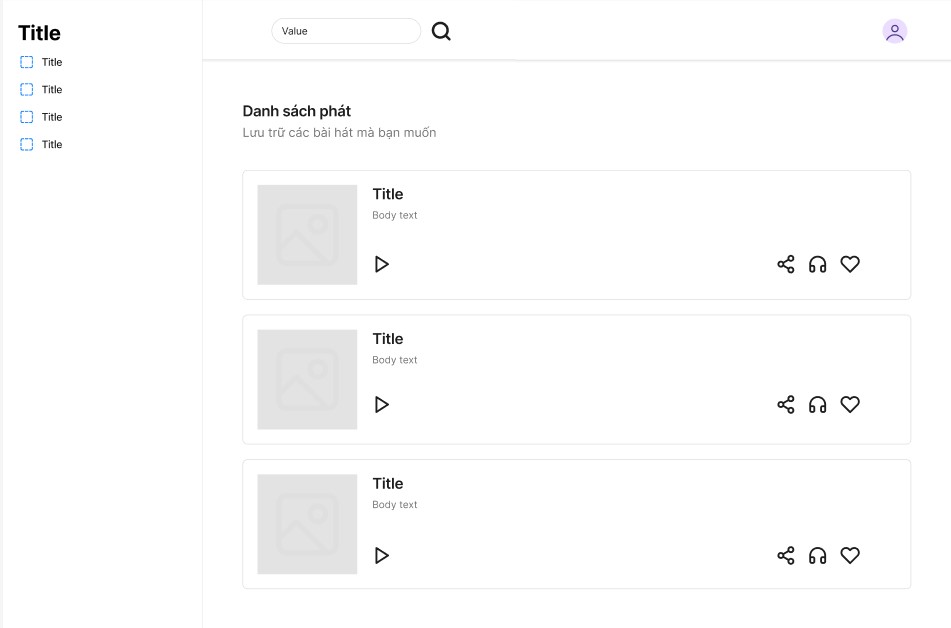
4.2.5.1.Mô tả chung

* Hiển thị danh sách bài hát trong danh sách phát của người dùng.
* Cho phép người dùng phát nhạc, thêm bài hát vào danh sách phát, chỉnh sửa hoặc xóa danh sách phát.
* Cung cấp các tùy chọn tương tác như chia sẻ danh sách phát với bạn bè

4.2.5.2.Chức năng chính

* Phát nhạc từ danh sách phát.
* Thêm bài hát vào danh sách phát.
* Xóa bài hát khỏi danh sách phát.
* Chỉnh sửa thông tin danh sách phát (tên danh sách, ảnh bìa).
* Chia sẻ danh sách phát với bạn bè qua mạng xã hội.
* Sắp xếp bài hát theo thứ tự mong muốn.

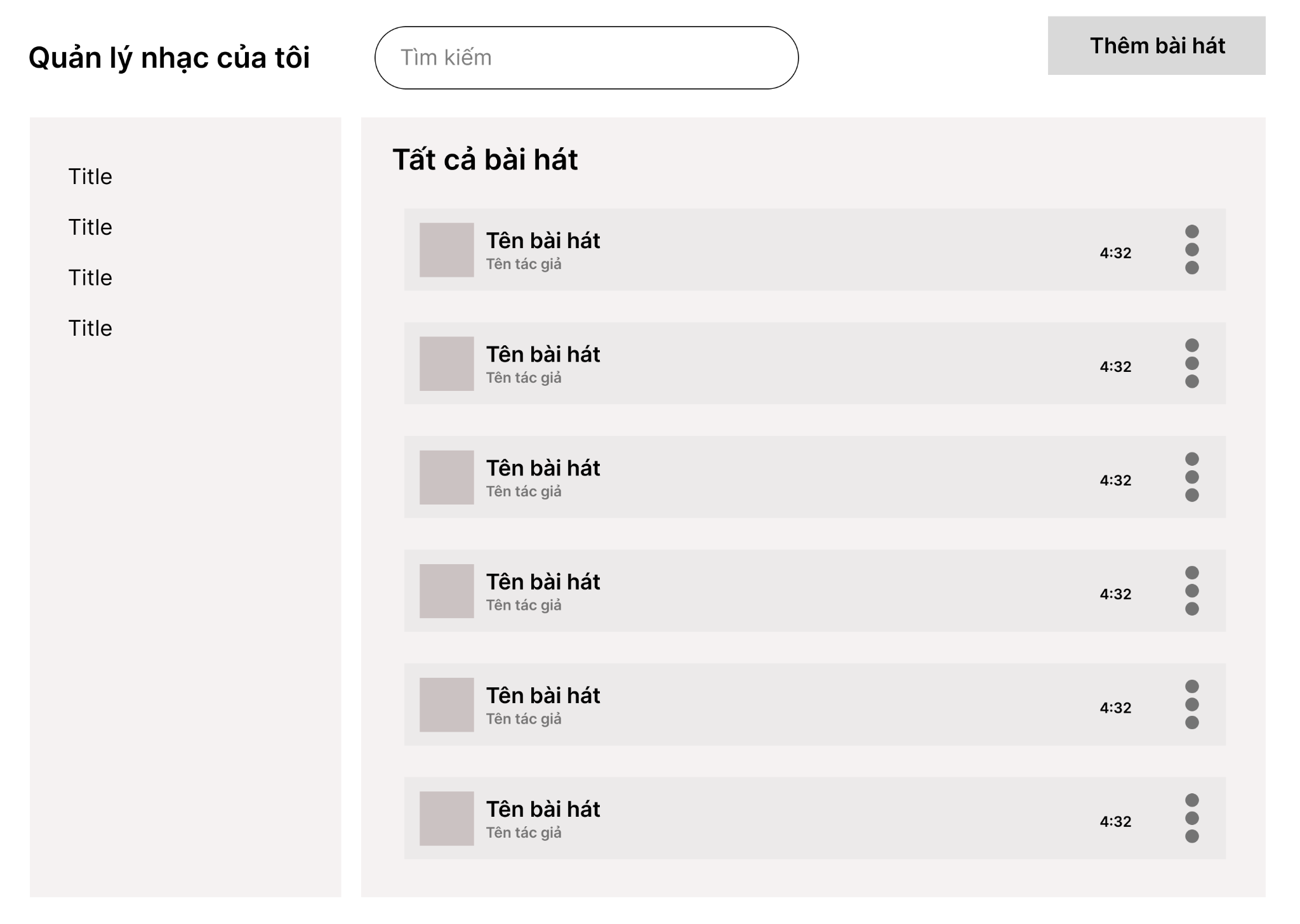
4.2.5.3.Wireframe Giao Diện



4.2.5.4.Mô tả chi tiết giao diện

| Thành phần | Mô tả |
| --- | --- |
| Ảnh bìa danh sách phát | Hiển thị ảnh đại diện của danh sách phát. |
| Tên danh sách phát | Hiển thị tiêu đề danh sách phát. Cho phép chỉnh sửa |
| Danh sách bài hát | Hiển thị danh sách các bài hát có trong danh sách phát. |
| Nút "Phát tất cả" | Cho phép phát toàn bộ danh sách phát theo thứ tự. |
| Nút "Thêm bài hát" | Hiển thị modal tìm kiếm và thêm bài hát mới vào danh sách phát. |
| Nút "Xóa" trên mỗi bài hát | Xóa bài hát khỏi danh sách phát. |
| Nút "Chia sẻ" | Chia sẻ danh sách phát qua mạng xã hội hoặc sao chép liên kết. |

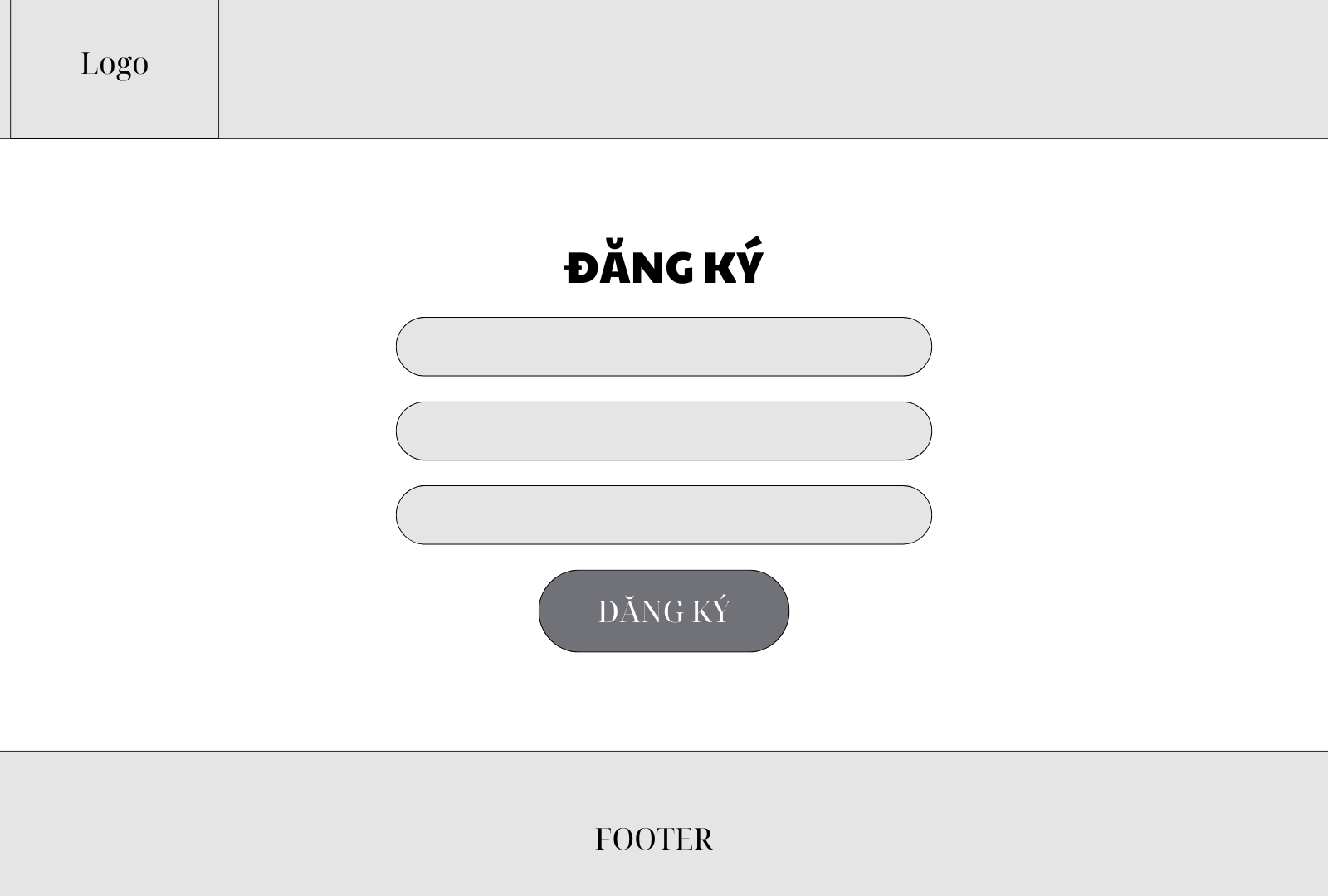
* 1. **Đặc tả Giao diện Trang Quản lý nhạc của tôi (Phương)**
* **Wireframe giao diện**

****

* Mô tả giao diện

| **Thành phần** | **Mô tả** |
| --- | --- |
| Ảnh bài hát | Hiển thị ảnh chính của bài nhạc |
| Nút “tùy chọn” | Có thể gỡ bài hát xuống hoặc chỉnh quyền riêng tư của bài hát |
| Nút “thêm bài hát” | Mở form thông tin để upload bài hát |

* 1. **Đặc tả Giao diện Trang Đăng ký (Phi)**
     1. **Mô tả chung**
* Hiển thị các khung textbox cho người dùng nhập tên người dùng, mật khẩu và nhập lại mật khẩu để đăng kí tài khoản

****

* 1. **Đặc tả Giao diện Trang Đăng nhập (Sáng)**

****

4.2.4.1 Mô tả chung

Giao diện Trang Đăng nhập có bố cục gồm :

* Khu vực logo (góc trên bên trái): Hiển thị logo của ứng dụng
* Tiêu đề "Đăng nhập" (chính giữa màn hình): Giúp người dùng nhận diện trang đăng nhập.
* Hai ô nhập liệu:
* Ô thứ nhất: Nhập tên đăng nhập hoặc email.
* Ô thứ hai: Nhập mật khẩu.
* Nút "Đăng nhập": Người dùng nhấn để xác thực thông tin và truy cập hệ thống.
* Dòng nhắc "Bạn chưa có tài khoản? Đăng ký": Liên kết giúp người dùng chuyển hướng đến trang đăng ký nếu chưa có tài khoản.

4.2.4.2 Đặc tả các chức năng chính của giao diện:

* Nhập thông tin đăng nhập:
* Người dùng nhập tên đăng nhập/email và mật khẩu vào ô tương ứng.
* Nếu nhập thiếu hoặc sai định dạng, hệ thống hiển thị thông báo lỗi.
* Xác thực đăng nhập:
* Khi nhấn nút "Đăng nhập", hệ thống kiểm tra thông tin và phản hồi
* Nếu đúng: Chuyển hướng đến trang chủ.
* Nếu sai: Hiển thị lỗi "Sai tên đăng nhập hoặc mật khẩu".
* Chuyển hướng đến trang đăng ký:
* Khi nhấn vào "Đăng k**ý"**, hệ thống chuyển hướng đến trang tạo tài khoản mới.
  1. **Đặc tả Giao diện Trang Đổi mật khẩu (Trí)**

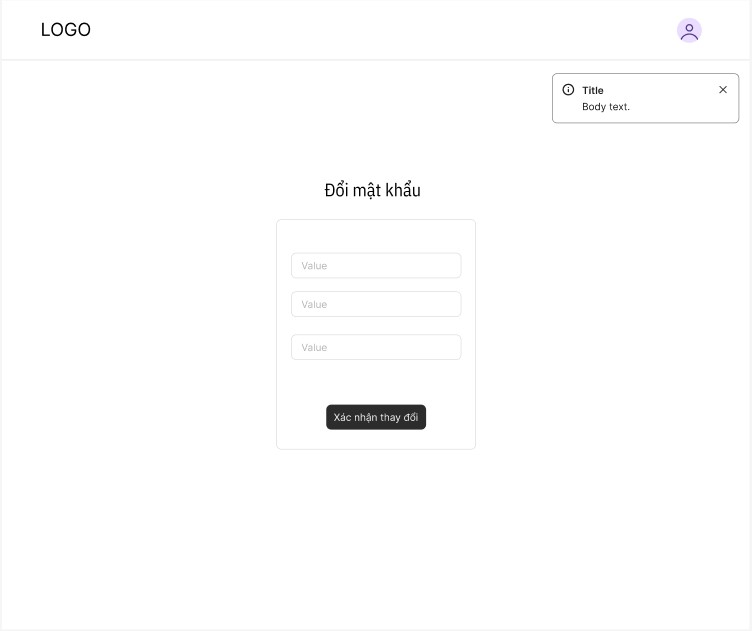
4.2.9.1.Mô tả chung

* Cung cấp giao diện cho người dùng thay đổi mật khẩu tài khoản.
* Yêu cầu nhập mật khẩu cũ, mật khẩu mới và xác nhận mật khẩu mới.
* Đảm bảo bảo mật với các quy tắc mật khẩu mạnh và xác thực người dùng.

4.2.9.2.Chức năng chính

* Nhập mật khẩu cũ để xác minh danh tính người dùng.
* Nhập và xác nhận mật khẩu mới với các yêu cầu bảo mật.
* Hiển thị thông báo lỗi nếu mật khẩu nhập không hợp lệ.
* Cho phép hiển thị/ẩn mật khẩu bằng biểu tượng mắt.
* Xác nhận thay đổi mật khẩu và đăng xuất khỏi tất cả thiết bị.
* Gửi email xác nhận khi mật khẩu được thay đổi thành công.

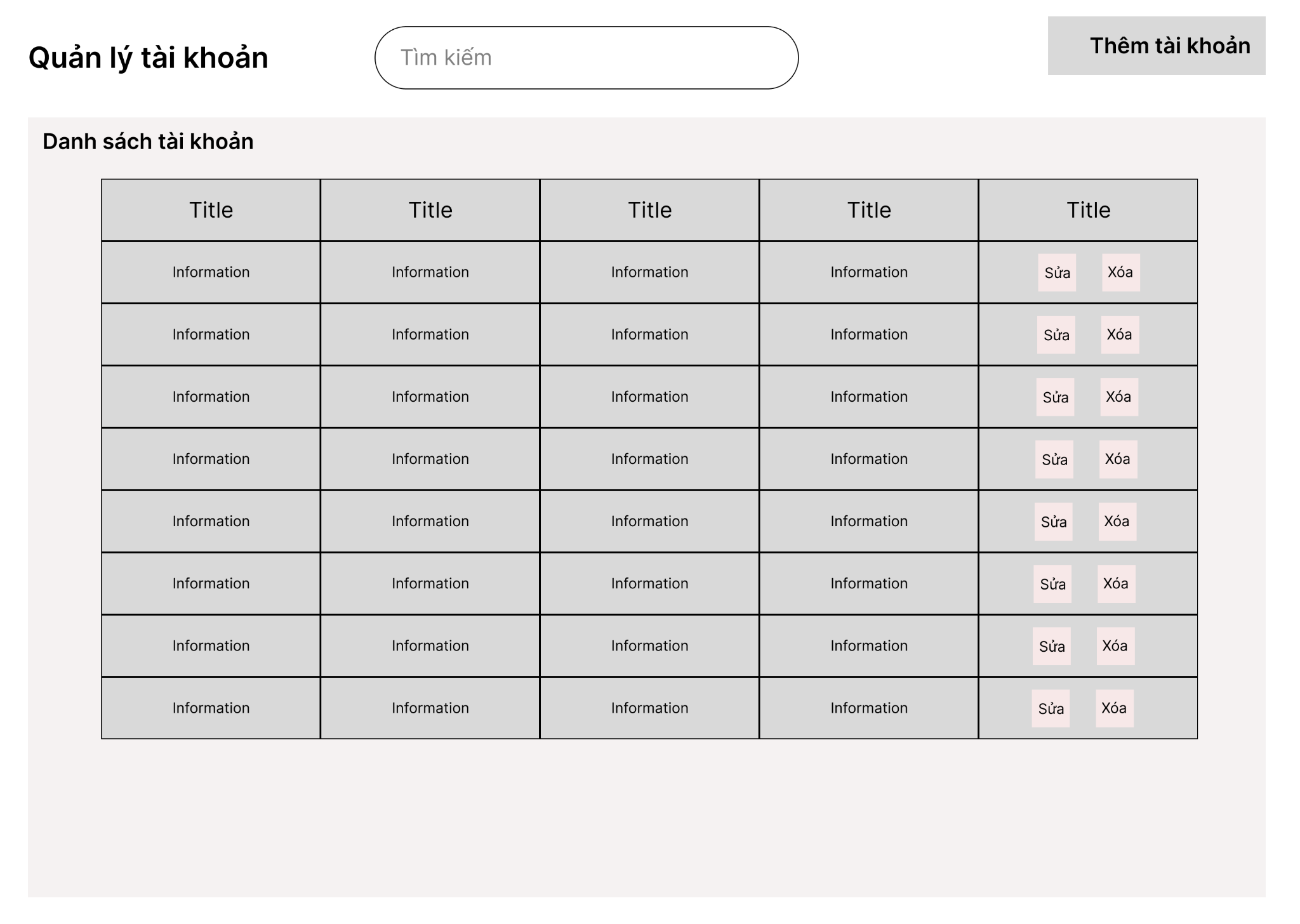
4.2.9.3.Wireframe Giao Diện



4.2.9.4.Mô tả chi tiết giao diện

| Thành phần | Mô tả |
| --- | --- |
| Tiêu đề "Đổi Mật Khẩu" | Hiển thị tiêu đề trang đổi mật khẩu. |
| Trường nhập mật khẩu cũ | Người dùng nhập mật khẩu hiện tại. |
| Trường nhập mật khẩu mới | Nhập mật khẩu mới với điều kiện bảo mật (ít nhất 8 ký tự, bao gồm chữ hoa, chữ thường, số và ký tự đặc biệt). |
| Trường xác nhận mật khẩu mới | Người dùng nhập lại mật khẩu mới để xác nhận. |
| Biểu tượng "Hiển thị/Ẩn mật khẩu" | Cho phép xem mật khẩu khi nhập. |
| Nút "Lưu thay đổi" | Xác nhận đổi mật khẩu và kiểm tra hợp lệ. |
| Thông báo lỗi | Hiển thị lỗi nếu mật khẩu cũ sai hoặc mật khẩu mới không |
| Thông báo thành công | Hiển thị khi đổi mật khẩu thành công, đồng thời đăng xuất khỏi tất cả thiết bị. |

* 1. **Đặc tả Giao diện Trang Danh sách tài khoản (Phương)**
* **Iframe**

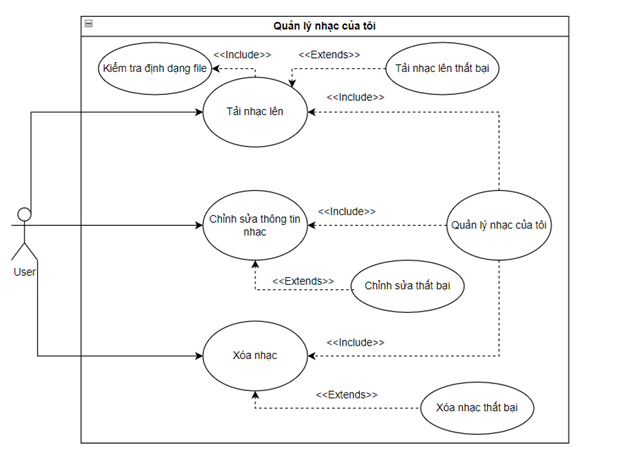
****

* **Mô tả**

| **Thành phần** | **Mô tả** |
| --- | --- |
| Nút “thêm tài khoản” | Hiển thị form để nhập thông tin tài khoản mới |
| Nút “sửa” | Sửa các thông tin của tài khoản được chọn |
| Nút “xóa” | Xóa các thông tin của tài khoản được chọn |

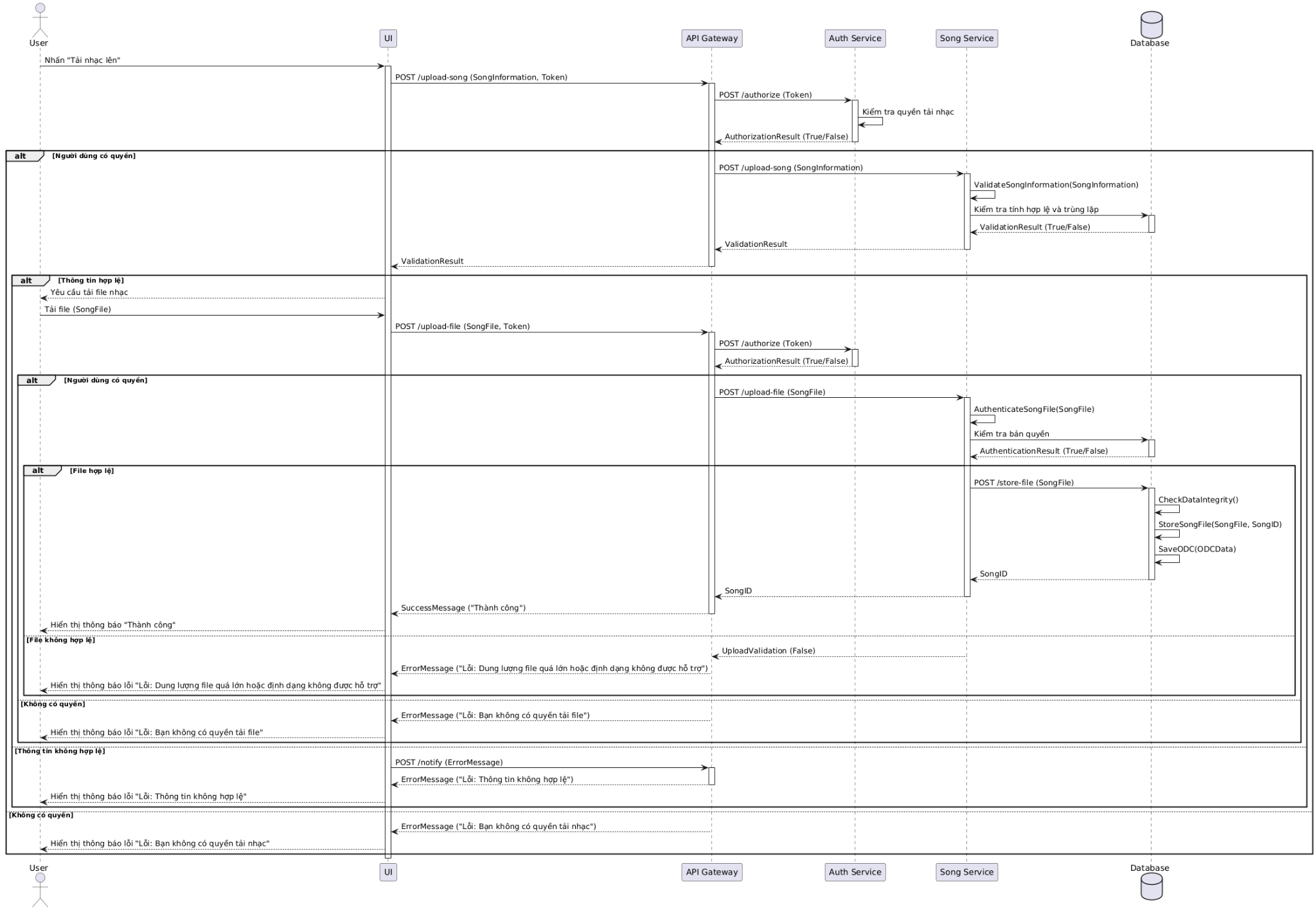
## Thiết kế Xử lý chi tiết

### Use Case Quản lý nhạc của tôi (Thắng)

****

#### 1. Tải nhạc lên

* **Sequence Diagram:**

****

**1.1. Giao diện (UI)**

Phương thức: EnterSongInformation()

* Mục đích: Cho phép người dùng nhập thông tin cơ bản về bài hát để bắt đầu quá trình tải lên.
* Input:
  1. SongInformation (Object):
     + Title (String): Tên bài hát (bắt buộc).
     + UserID (String): Mã người dùng (lấy từ token).
     + Duration (Integer): Thời lượng bài hát tính bằng giây (bắt buộc).
     + AlbumID (String): Mã album (không bắt buộc).
     + ReleaseCategoryID (String): Mã danh mục phát hành (không bắt buộc).
* Output: ValidationResult (Boolean) - True nếu thông tin hợp lệ, False nếu có lỗi.
* Quy trình :
  1. Người dùng → UI: Người dùng nhấn nút "Tải nhạc lên" (message: Nhấn "Tải nhạc lên").
  2. UI → UI: UI tự gọi EnterSongInformation() để hiển thị form nhập thông tin (self-message).
  3. UI → Người dùng: Hiển thị form để người dùng nhập thông tin (message: Hiển thị form nhập thông tin bài hát).
  4. Người dùng → UI: Người dùng nhập thông tin bài hát và nhấn "Lưu" (message: Nhập thông tin bài hát (SongInformation)).
  5. UI → API Gateway: Gửi thông tin bài hát đến API Gateway để kiểm tra quyền và tính hợp lệ (message: POST /upload-song (SongInformation, Token)).
  6. API Gateway → UI: Trả về kết quả kiểm tra (message: ValidationResult (True/False)).

Phương thức: UploadSongFile()

* Mục đích: Cho phép người dùng tải file âm thanh (mp3, wav, v.v.) lên hệ thống.
* Input:
  1. SongFile (File/Stream):
     + Path (String): Đường dẫn tạm thời của file trên client.
     + Size (Integer): Kích thước file tính bằng byte.
* Output: UploadValidation (Boolean) - True nếu file hợp lệ, False nếu định dạng hoặc kích thước không đúng.
* Quy trình :
  1. UI → Người dùng: Yêu cầu người dùng tải file nhạc (message: Yêu cầu tải file nhạc).
  2. Người dùng → UI: Người dùng chọn và tải file (message: Tải file (SongFile)).
  3. UI → API Gateway: Gửi file đến API Gateway (message: POST /upload-file (SongFile, Token)).
  4. API Gateway → UI: Trả về kết quả xác thực file (message: UploadValidation (True/False)).

Phương thức: DisplayNotification()

* Mục đích: Hiển thị thông báo kết quả (thành công hoặc lỗi) cho người dùng.
* Input: Message (String) - Tin nhắn xác nhận hoặc lỗi (ví dụ: "Thành công", "Lỗi: Dung lượng file quá lớn...").
* Output: None (void) - Chỉ hiển thị trên giao diện.
* Quy trình :
  1. API Gateway → UI: Gửi thông báo đến UI (message: SuccessMessage hoặc ErrorMessage).
  2. UI → UI: UI tự gọi DisplayNotification(Message) để hiển thị thông báo (self-message).
  3. UI → Người dùng: Hiển thị thông báo cho người dùng (message: Hiển thị thông báo (Message)).

**1.2. Dịch vụ (Services)**

**1.2.1. Auth Service**

Auth Service chịu trách nhiệm kiểm tra quyền của người dùng.

* Phương thức: AuthorizeUser()
  + Mục đích: Kiểm tra quyền của người dùng dựa trên token để đảm bảo họ có quyền thực hiện hành động (tải nhạc, tải file).
  + Input:
    1. Token (String): Token xác thực, chứa:
       - UserID (String): Mã người dùng.
       - RoleID (String): Mã vai trò (ví dụ: admin, user).
  + Output: AuthorizationResult (Boolean) - True nếu người dùng có quyền, False nếu không.
  + Quy trình :
    1. API Gateway → Auth Service: Gửi yêu cầu kiểm tra quyền (message: POST /authorize (Token)).
    2. Auth Service → Auth Service: Tự gọi AuthorizeUser(Token) để kiểm tra quyền (self-message).
    3. Auth Service → API Gateway: Trả về kết quả (message: AuthorizationResult (True/False)).

**1.2.2. Song Service**

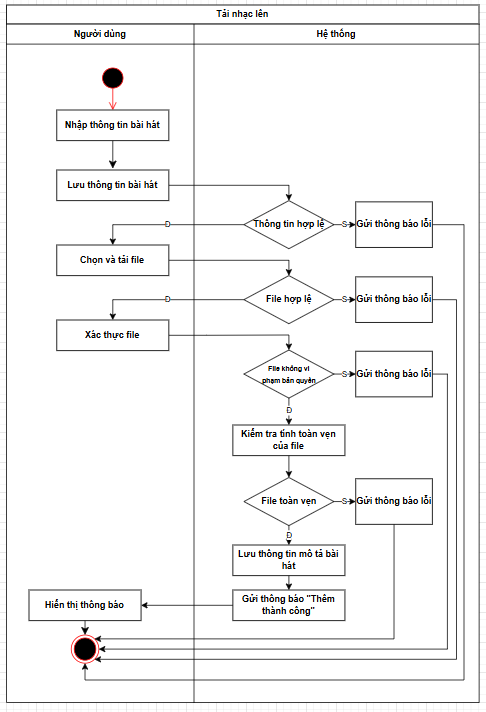
* Phương thức: ValidateSongInformation()
  + Mục đích: Kiểm tra tính hợp lệ và duy nhất của thông tin bài hát so với cơ sở dữ liệu.
  + Input:
    1. SongInformation (Object):
       - Title (String): Tên bài hát (bắt buộc).
       - UserID (String): Mã người dùng (lấy từ token).
       - Duration (Integer): Thời lượng bài hát tính bằng giây (bắt buộc).
       - AlbumID (String): Mã album (không bắt buộc).
       - ReleaseCategoryID (String): Mã danh mục phát hành (không bắt buộc).
  + Output: ValidationResult (Boolean) - True nếu hợp lệ và không trùng lặp, False nếu có lỗi.
  + Quy trình :
    1. API Gateway → Song Service: Gửi thông tin bài hát để kiểm tra (message: POST /upload-song (SongInformation)).
    2. Song Service → Song Service: Tự gọi ValidateSongInformation(SongInformation) để xử lý (self-message).
    3. Song Service → Database: Gửi yêu cầu kiểm tra trùng lặp (message: Kiểm tra tính hợp lệ và trùng lặp).
    4. Database → Song Service: Trả về kết quả kiểm tra (message: ValidationResult (True/False)).
    5. Song Service → API Gateway: Trả về kết quả (message: ValidationResult).
* Phương thức: AuthenticateSongFile()
  + Mục đích: Xác thực nội dung file bài hát để đảm bảo không vi phạm bản quyền hoặc chứa nội dung không phù hợp.
  + Input:
    1. SongFile (File/Stream):
       - Path (String): Đường dẫn tạm thời của file.
       - Size (Integer): Kích thước file tính bằng byte.
  + Output: AuthenticationResult (Boolean) - True nếu hợp lệ, False nếu vi phạm.
  + Quy trình :
    1. API Gateway → Song Service: Gửi file để xác thực (message: POST /upload-file (SongFile)).
    2. Song Service → Song Service: Tự gọi AuthenticateSongFile(SongFile) để phân tích file (self-message).
    3. Song Service → Database: Gửi yêu cầu kiểm tra bản quyền (message: Kiểm tra bản quyền).
    4. Database → Song Service: Trả về kết quả xác thực (message: AuthenticationResult (True/False)).
    5. Song Service → API Gateway: Trả về kết quả (message: UploadValidation (True/False)).
* Phương thức: StoreSong()
  + Mục đích: Chuyển file bài hát đã xác thực và thông tin bài hát đến Database để lưu trữ.
  + Input:
    1. SongFile (File/Stream):
       - Path (String): Đường dẫn tạm thời của file.
       - Size (Integer): Kích thước file tính bằng byte.
  + Output: SongID (String) - Mã định danh bài hát nếu lưu thành công (tương ứng với SongID).
  + Quy trình :
    1. Song Service → Database: Gửi file để lưu trữ (message: POST /store-file (SongFile)).
    2. Database → Song Service: Trả về mã định danh bài hát (message: SongID).
    3. Song Service → API Gateway: Chuyển SongID về API Gateway (message: SongID).

**1.3. Lưu trữ**

* Phương thức: CheckDataIntegrity()
  + Mục đích: Kiểm tra tính toàn vẹn của dữ liệu bài hát sau khi lưu trữ.
  + Input:
    1. SongFile (File/Stream):
       - Path (String): Đường dẫn tạm thời của file.
       - Size (Integer): Kích thước file tính bằng byte.
  + Output: IntegrityResult (Boolean) - True nếu dữ liệu toàn vẹn, False nếu có lỗi.
  + Quy trình :
    1. Song Service → Database: Gửi file để lưu trữ (message: POST /store-file (SongFile)).
    2. Database → Database: Tự gọi CheckDataIntegrity() để kiểm tra tính toàn vẹn (self-message).
    3. Database → Song Service: Trả về kết quả (message: IntegrityResult (True/False)).
* Phương thức: SaveODC()
  + Mục đích: Lưu thông tin mô tả (ODC - Object Data Container) của bài hát vào cơ sở dữ liệu.
  + Input:
    1. ODCData (Object):
       - SongID (String): Mã định danh bài hát.
       - Title (String): Tên bài hát.
       - UserID (String): Mã người dùng sở hữu bài hát.
       - Duration (Integer): Thời lượng bài hát.
       - FilePath (String): Đường dẫn lưu trữ file (tương ứng với Path trong File).
       - CreatedAt (DateTime): Thời gian tạo bài hát.
  + Output: SaveConfirmation (Boolean) - True nếu lưu thành công, False nếu thất bại.
  + Quy trình :
    1. Database → Database: Tự gọi SaveODC(ODCData) để lưu metadata (self-message).
    2. Database → Song Service: Trả về xác nhận (message: SaveConfirmation (True/False)).
* Phương thức: StoreSongFile()
  + Mục đích: Lưu trữ vĩnh viễn file bài hát vào hệ thống lưu trữ dài hạn (database).
  + Input:
    1. SongFile (File/Stream):
       - Path (String): Đường dẫn tạm thời của file.
       - Size (Integer): Kích thước file tính bằng byte.
    2. SongID (String): Mã định danh bài hát (SongID).
  + Output: StorageConfirmation (Boolean) - True nếu lưu thành công, False nếu thất bại.
  + Quy trình :
    1. Database → Database: Tự gọi StoreSongFile(SongFile, SongID) để lưu file (self-message).
    2. Database → Song Service: Trả về xác nhận (message: StorageConfirmation (True/False)).
    3. Database → Song Service: Trả về SongID (message: SongID).
* Phương thức: CheckDuplicateSongInformation()
  + Mục đích: Kiểm tra xem thông tin bài hát (tên, người dùng, v.v.) có bị trùng lặp trong cơ sở dữ liệu hay không.
  + Input:
    1. SongInformation (Object):
       - Title (String): Tên bài hát.
       - UserID (String): Mã người dùng.
       - Duration (Integer): Thời lượng bài hát.
       - AlbumID (String): Mã album (nếu có).
       - ReleaseCategoryID (String): Mã danh mục phát hành (nếu có).
  + Output: ValidationResult (Boolean) - True nếu không trùng lặp, False nếu trùng lặp.
  + Quy trình :
    1. Song Service → Database: Gửi yêu cầu kiểm tra trùng lặp (message: Kiểm tra tính hợp lệ và trùng lặp).
    2. Database → Database: Tự gọi CheckDuplicateSongInformation(SongInformation) để kiểm tra (self-message).
    3. Database → Song Service: Trả về kết quả (message: ValidationResult (True/False)).
* Phương thức: CheckCopyright()
  + Mục đích: Kiểm tra xem file bài hát có vi phạm bản quyền hay không.
  + Input:
    1. SongFile (File/Stream):
       - Path (String): Đường dẫn tạm thời của file.
       - Size (Integer): Kích thước file tính bằng byte.
  + Output: AuthenticationResult (Boolean) - True nếu không vi phạm, False nếu vi phạm.
  + Quy trình :
    1. Song Service → Database: Gửi yêu cầu kiểm tra bản quyền (message: Kiểm tra bản quyền).
    2. Database → Database: Tự gọi CheckCopyright(SongFile) để kiểm tra (self-message).
    3. Database → Song Service: Trả về kết quả (message: AuthenticationResult (True/False)).

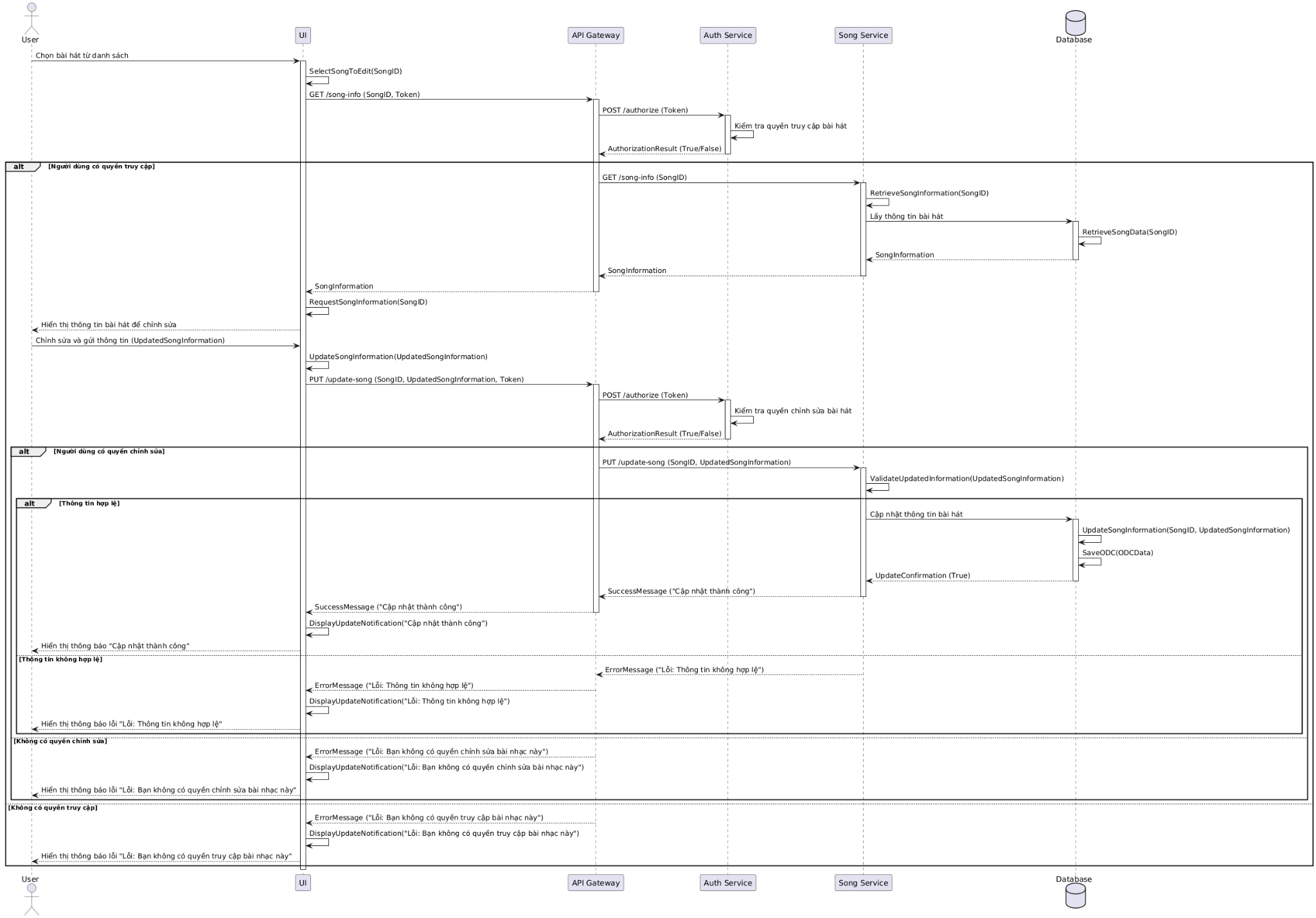
#### 

**1.4. Activity Diagram**



#### 2. Chỉnh sửa thông tin nhạc

* **Sequence Diagram:**

****

**2.1. Giao diện (UI)**

* Phương thức: SelectSongToEdit()
  + Mục đích: Cho phép người dùng chọn bài hát từ danh sách để chỉnh sửa.
  + Input:
    1. SongID (String): Mã định danh bài hát.
  + Output: None (void) - Chỉ chuyển sang bước lấy thông tin bài hát.
  + Quy trình :
    1. Người dùng → UI: Người dùng chọn bài hát từ danh sách (message: Chọn bài hát từ danh sách).
    2. UI → UI: UI tự gọi SelectSongToEdit(SongID) để ghi nhận bài hát được chọn (self-message).
* Phương thức: RequestSongInformation()
  + Mục đích: Yêu cầu thông tin chi tiết của bài hát để hiển thị cho người dùng chỉnh sửa.
  + Input:
    1. SongID (String): Mã định danh bài hát.
  + Output: SongInformation (Object) - Thông tin bài hát để hiển thị.
  + Quy trình :
    1. UI → Gateway: Gửi yêu cầu lấy thông tin bài hát (message: GET /song-info (SongID, Token)).
    2. Gateway → UI: Trả về thông tin bài hát (message: SongInformation).
    3. UI → Người dùng: Hiển thị thông tin bài hát để chỉnh sửa (message: Hiển thị thông tin bài hát để chỉnh sửa).
* Phương thức: UpdateSongInformation()
  + Mục đích: Gửi thông tin bài hát đã được chỉnh sửa để cập nhật.
  + Input:
    1. UpdatedSongInformation (Object):
       - Title (String): Tên bài hát (cập nhật).
       - UserID (String): Mã người dùng (lấy từ token).
       - Duration (Integer): Thời lượng bài hát (cập nhật).
       - AlbumID (String): Mã album (cập nhật, không bắt buộc).
       - ReleaseCategoryID (String): Mã danh mục phát hành (cập nhật, không bắt buộc).
  + Output: None (void) - Chuyển sang bước gửi yêu cầu cập nhật.
  + Quy trình :
    1. Người dùng → UI: Người dùng chỉnh sửa và gửi thông tin (message: Chỉnh sửa và gửi thông tin (UpdatedSongInformation)).
    2. UI → UI: UI tự gọi UpdateSongInformation(UpdatedSongInformation) để xử lý thông tin (self-message).
    3. UI → Gateway: Gửi yêu cầu cập nhật (message: PUT /update-song (SongID, UpdatedSongInformation, Token)).
* Phương thức: DisplayUpdateNotification()
  + Mục đích: Hiển thị thông báo kết quả (thành công hoặc lỗi) cho người dùng.
  + Input: Message (String) - Tin nhắn xác nhận hoặc lỗi (ví dụ: "Cập nhật thành công", "Lỗi: Thông tin không hợp lệ").
  + Output: None (void) - Chỉ hiển thị trên giao diện.
  + Quy trình :
    1. Gateway → UI: Gửi thông báo đến UI (message: SuccessMessage hoặc ErrorMessage).
    2. UI → UI: UI tự gọi DisplayUpdateNotification(Message) để hiển thị thông báo (self-message).
    3. UI → Người dùng: Hiển thị thông báo cho người dùng (message: Hiển thị thông báo (Message)).

**2.2. Dịch vụ (Services)**

**2.2.1. Auth Service**

* Phương thức: AuthorizeUser()
  + Mục đích: Kiểm tra quyền của người dùng dựa trên token để đảm bảo họ có quyền truy cập và chỉnh sửa bài hát.
  + Input:
    1. Token (String): Token xác thực, chứa:
       - UserID (String): Mã người dùng.
       - RoleID (String): Mã vai trò (ví dụ: admin, user).
  + Output: AuthorizationResult (Boolean) - True nếu người dùng có quyền, False nếu không.
  + Quy trình :
    1. Gateway → Auth Service: Gửi yêu cầu kiểm tra quyền (message: POST /authorize (Token)).
    2. Auth Service → Auth Service: Tự gọi AuthorizeUser(Token) để kiểm tra quyền (self-message).
    3. Auth Service → Gateway: Trả về kết quả (message: AuthorizationResult (True/False)).

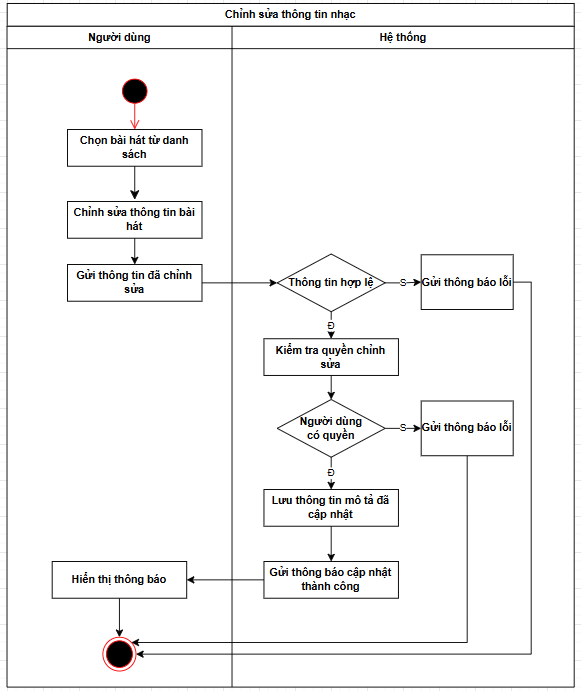
**2.2. Song Service**

* Phương thức: RetrieveSongInformation()
  + Mục đích: Lấy thông tin chi tiết của bài hát từ cơ sở dữ liệu để hiển thị cho người dùng.
  + Input:
    1. SongID (String): Mã định danh bài hát.
  + Output: SongInformation (Object) - Thông tin bài hát, bao gồm:
    1. Title (String): Tên bài hát.
    2. UserID (String): Mã người dùng sở hữu bài hát.
    3. Duration (Integer): Thời lượng bài hát.
    4. AlbumID (String): Mã album.
    5. ReleaseCategoryID (String): Mã danh mục phát hành.
  + Quy trình :
    1. Gateway → Song Service: Gửi yêu cầu lấy thông tin bài hát (message: GET /song-info (SongID)).
    2. Song Service → Song Service: Tự gọi RetrieveSongInformation(SongID) để xử lý (self-message).
    3. Song Service → Database: Gửi yêu cầu lấy dữ liệu (message: Lấy thông tin bài hát).
    4. Database → Song Service: Trả về thông tin bài hát (message: SongInformation).
    5. Song Service → Gateway: Trả về thông tin (message: SongInformation).
* Phương thức: ValidateUpdatedInformation()
  + Mục đích: Kiểm tra tính hợp lệ của thông tin bài hát đã được cập nhật.
  + Input:
    1. UpdatedSongInformation (Object):
       - Title (String): Tên bài hát (cập nhật).
       - UserID (String): Mã người dùng (lấy từ token).
       - Duration (Integer): Thời lượng bài hát (cập nhật).
       - AlbumID (String): Mã album (cập nhật, không bắt buộc).
       - ReleaseCategoryID (String): Mã danh mục phát hành (cập nhật, không bắt buộc).
  + Output: ValidationResult (Boolean) - True nếu thông tin hợp lệ, False nếu có lỗi.
  + Quy trình :
    1. Gateway → Song Service: Gửi thông tin cập nhật để kiểm tra (message: PUT /update-song (SongID, UpdatedSongInformation)).
    2. Song Service → Song Service: Tự gọi ValidateUpdatedInformation(UpdatedSongInformation) để kiểm tra (self-message).
    3. Song Service → Gateway: Trả về kết quả kiểm tra (message: ValidationResult).
* Phương thức: UpdateSongInformation()
  + Mục đích: Cập nhật thông tin bài hát trong cơ sở dữ liệu.
  + Input:
    1. SongID (String): Mã định danh bài hát.
    2. UpdatedSongInformation (Object):
       - Title (String): Tên bài hát (cập nhật).
       - UserID (String): Mã người dùng (lấy từ token).
       - Duration (Integer): Thời lượng bài hát (cập nhật).
       - AlbumID (String): Mã album (cập nhật, không bắt buộc).
       - ReleaseCategoryID (String): Mã danh mục phát hành (cập nhật, không bắt buộc).
  + Output: UpdateConfirmation (Boolean) - True nếu cập nhật thành công, False nếu thất bại.
  + Quy trình :
    1. Song Service → Database: Gửi yêu cầu cập nhật thông tin (message: Cập nhật thông tin bài hát).
    2. Database → Song Service: Trả về xác nhận (message: UpdateConfirmation (True)).
    3. Song Service → Gateway: Trả về thông báo thành công (message: SuccessMessage ("Cập nhật thành công")).

**2.3. Lưu trữ**

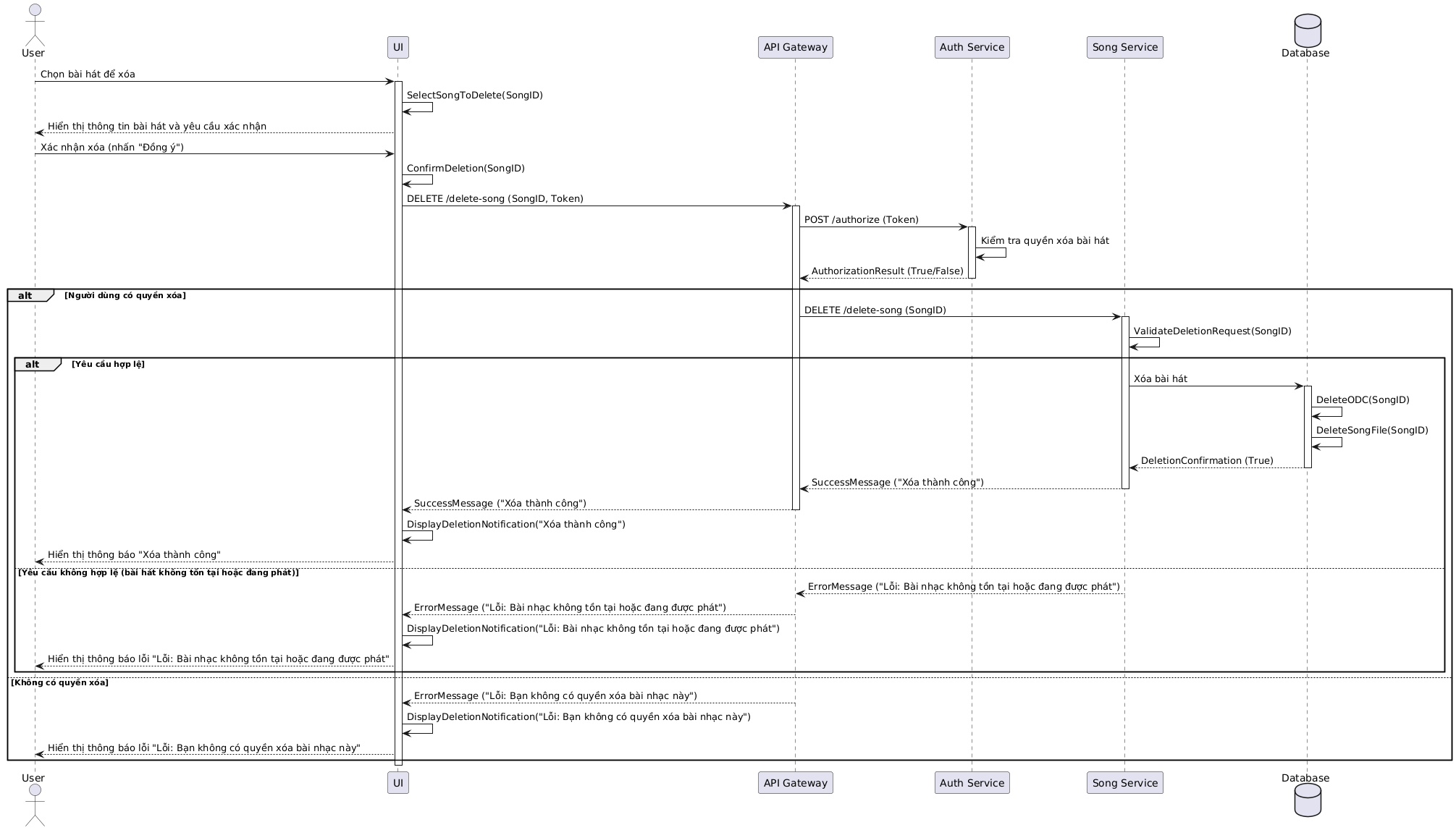
* Phương thức: RetrieveSongData()
  + Mục đích: Lấy thông tin chi tiết của bài hát từ cơ sở dữ liệu.
  + Input:
    1. SongID (String): Mã định danh bài hát.
  + Output: SongInformation (Object) - Thông tin bài hát, bao gồm:
    1. Title (String): Tên bài hát.
    2. UserID (String): Mã người dùng sở hữu bài hát.
    3. Duration (Integer): Thời lượng bài hát.
    4. AlbumID (String): Mã album.
    5. ReleaseCategoryID (String): Mã danh mục phát hành.
  + Quy trình :
    1. Song Service → Database: Gửi yêu cầu lấy thông tin (message: Lấy thông tin bài hát).
    2. Database → Database: Tự gọi RetrieveSongData(SongID) để lấy dữ liệu (self-message).
    3. Database → Song Service: Trả về thông tin bài hát (message: SongInformation).
* Phương thức: UpdateSongInformation()
  + Mục đích: Cập nhật thông tin bài hát trong cơ sở dữ liệu.
  + Input:
    1. SongID (String): Mã định danh bài hát.
    2. UpdatedSongInformation (Object):
       - Title (String): Tên bài hát (cập nhật).
       - UserID (String): Mã người dùng sở hữu bài hát.
       - Duration (Integer): Thời lượng bài hát (cập nhật).
       - AlbumID (String): Mã album (cập nhật, không bắt buộc).
       - ReleaseCategoryID (String): Mã danh mục phát hành (cập nhật, không bắt buộc).
  + Output: UpdateConfirmation (Boolean) - True nếu cập nhật thành công, False nếu thất bại.
  + Quy trình :
    1. Song Service → Database: Gửi yêu cầu cập nhật (message: Cập nhật thông tin bài hát).
    2. Database → Database: Tự gọi UpdateSongInformation(SongID, UpdatedSongInformation) để cập nhật (self-message).
    3. Database → Song Service: Trả về xác nhận (message: UpdateConfirmation (True)).
* Phương thức: SaveODC()
  + Mục đích: Lưu thông tin mô tả (ODC - Object Data Container) của bài hát vào cơ sở dữ liệu sau khi cập nhật.
  + Input:
    1. ODCData (Object):
       - SongID (String): Mã định danh bài hát.
       - Title (String): Tên bài hát.
       - UserID (String): Mã người dùng sở hữu bài hát.
       - Duration (Integer): Thời lượng bài hát.
       - FilePath (String): Đường dẫn lưu trữ file.
       - CreatedAt (DateTime): Thời gian tạo bài hát.
  + Output: SaveConfirmation (Boolean) - True nếu lưu thành công, False nếu thất bại.
  + Quy trình :
    1. Database → Database: Tự gọi SaveODC(ODCData) để lưu metadata (self-message).
    2. Database → Song Service: Trả về xác nhận (message: SaveConfirmation (True)).

**2.4. Activity Diagram**



#### 3. Xóa nhạc

* **Sequence Diagram:**

****

**3.1. Giao diện (UI)**

* Phương thức: SelectSongToDelete()
  + Mục đích: Cho phép người dùng chọn bài hát từ danh sách để xóa.
  + Input:
    1. SongID (String): Mã định danh bài hát.
  + Output: None (void) - Chỉ chuyển sang bước yêu cầu xác nhận xóa.
  + Quy trình:
    1. Người dùng → UI: Người dùng chọn bài hát để xóa (message: Chọn bài hát để xóa).
    2. UI → UI: UI tự gọi SelectSongToDelete(SongID) để ghi nhận bài hát được chọn (self-message).
    3. UI → Người dùng: Hiển thị thông tin bài hát và yêu cầu xác nhận (message: Hiển thị thông tin bài hát và yêu cầu xác nhận).
* Phương thức: ConfirmDeletion()
  + Mục đích: Xác nhận hành động xóa bài hát và gửi yêu cầu xóa đến API Gateway.
  + Input:
    1. SongID (String): Mã định danh bài hát.
  + Output: None (void) - Chuyển sang bước gửi yêu cầu xóa.
  + Quy trình:
    1. Người dùng → UI: Người dùng xác nhận xóa (message: Xác nhận xóa (nhấn "Đồng ý")).
    2. UI → UI: UI tự gọi ConfirmDeletion(SongID) để xử lý xác nhận (self-message).
    3. UI → Gateway: Gửi yêu cầu xóa (message: DELETE /delete-song (SongID, Token)).
* Phương thức: DisplayDeletionNotification()
  + Mục đích: Hiển thị thông báo kết quả (thành công hoặc lỗi) cho người dùng.
  + Input: Message (String) - Tin nhắn xác nhận hoặc lỗi (ví dụ: "Xóa thành công", "Lỗi: Bài nhạc không tồn tại hoặc đang được phát").
  + Output: None (void) - Chỉ hiển thị trên giao diện.
  + Quy trình:
    1. Gateway → UI: Gửi thông báo đến UI (message: SuccessMessage hoặc ErrorMessage).
    2. UI → UI: UI tự gọi DisplayDeletionNotification(Message) để hiển thị thông báo (self-message).
    3. UI → Người dùng: Hiển thị thông báo cho người dùng (message: Hiển thị thông báo (Message)).

**3.2. Dịch vụ**

**3.2.1. Auth Service**

* Phương thức: AuthorizeUser()
  + Mục đích: Kiểm tra quyền của người dùng dựa trên token để đảm bảo họ có quyền xóa bài hát.
  + Input:
    1. Token (String): Token xác thực, chứa:
       - UserID (String): Mã người dùng.
       - RoleID (String): Mã vai trò (ví dụ: admin, user).
  + Output: AuthorizationResult (Boolean) - True nếu người dùng có quyền, False nếu không.
  + Quy trình:
    1. Gateway → Auth Service: Gửi yêu cầu kiểm tra quyền (message: POST /authorize (Token)).
    2. Auth Service → Auth Service: Tự gọi AuthorizeUser(Token) để kiểm tra quyền (self-message).
    3. Auth Service → Gateway: Trả về kết quả (message: AuthorizationResult (True/False)).

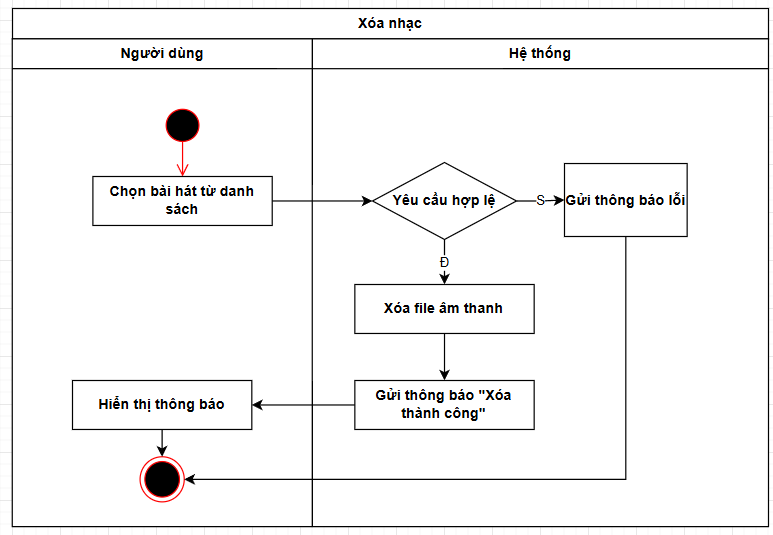
**3.2.2. Song Service**

* Phương thức: ValidateDeletionRequest()
  + Mục đích: Kiểm tra tính hợp lệ của yêu cầu xóa bài hát (bài hát có tồn tại không, có đang phát không).
  + Input:
    1. SongID (String): Mã định danh bài hát.
  + Output: ValidationResult (Boolean) - True nếu yêu cầu hợp lệ, False nếu không (bài hát không tồn tại hoặc đang phát).
  + Quy trình:
    1. Gateway → Song Service: Gửi yêu cầu xóa (message: DELETE /delete-song (SongID)).
    2. Song Service → Song Service: Tự gọi ValidateDeletionRequest(SongID) để kiểm tra (self-message).
    3. Song Service → Database: Gửi yêu cầu xóa nếu hợp lệ (message: Xóa bài hát).
* Phương thức: DeleteSong()
  + Mục đích: Gửi yêu cầu xóa bài hát đến Database.
  + Input:
    1. SongID (String): Mã định danh bài hát.
  + Output: DeletionConfirmation (Boolean) - True nếu xóa thành công, False nếu thất bại.
  + Quy trình:
    1. Song Service → Database: Gửi yêu cầu xóa (message: Xóa bài hát).
    2. Database → Song Service: Trả về xác nhận (message: DeletionConfirmation (True)).
    3. Song Service → Gateway: Trả về thông báo thành công (message: SuccessMessage ("Xóa thành công")).

**3.3. Lưu trữ**

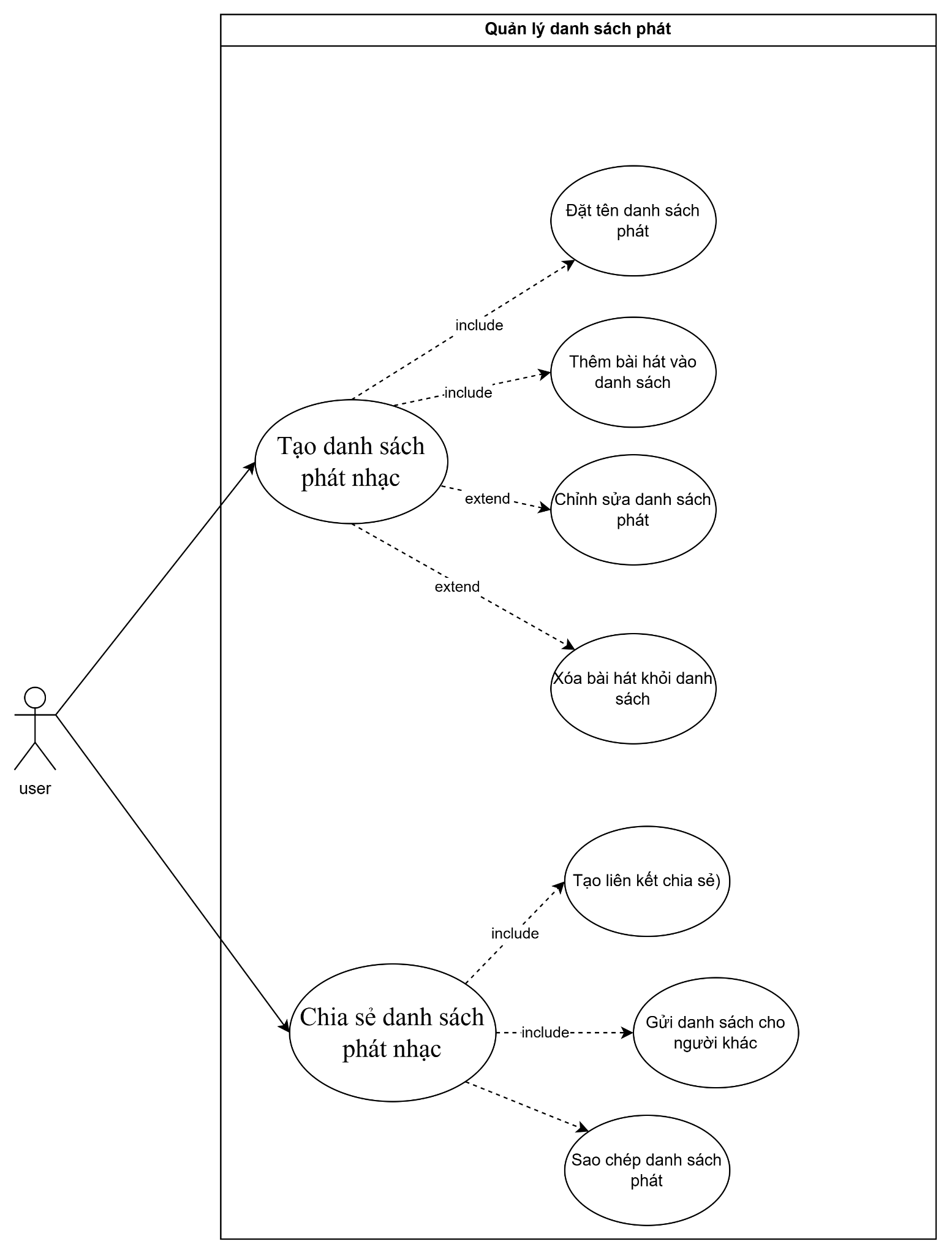
* Phương thức: DeleteODC()
  + Mục đích: Xóa thông tin mô tả (ODC - Object Data Container) của bài hát khỏi cơ sở dữ liệu.
  + Input:
    1. SongID (String): Mã định danh bài hát.
  + Output: None (void) - Chỉ thực hiện xóa metadata.
  + Quy trình:
    1. Song Service → Database: Gửi yêu cầu xóa (message: Xóa bài hát).
    2. Database → Database: Tự gọi DeleteODC(SongID) để xóa metadata (self-message).
* Phương thức: DeleteSongFile()
  + Mục đích: Xóa file bài hát khỏi hệ thống lưu trữ dài hạn.
  + Input:
    1. SongID (String): Mã định danh bài hát.
  + Output: DeletionConfirmation (Boolean) - True nếu xóa thành công, False nếu thất bại.
  + Quy trình:
    1. Database → Database: Tự gọi DeleteSongFile(SongID) để xóa file (self-message).
    2. Database → Song Service: Trả về xác nhận (message: DeletionConfirmation (True)).

**3.4. Activity Diagram**

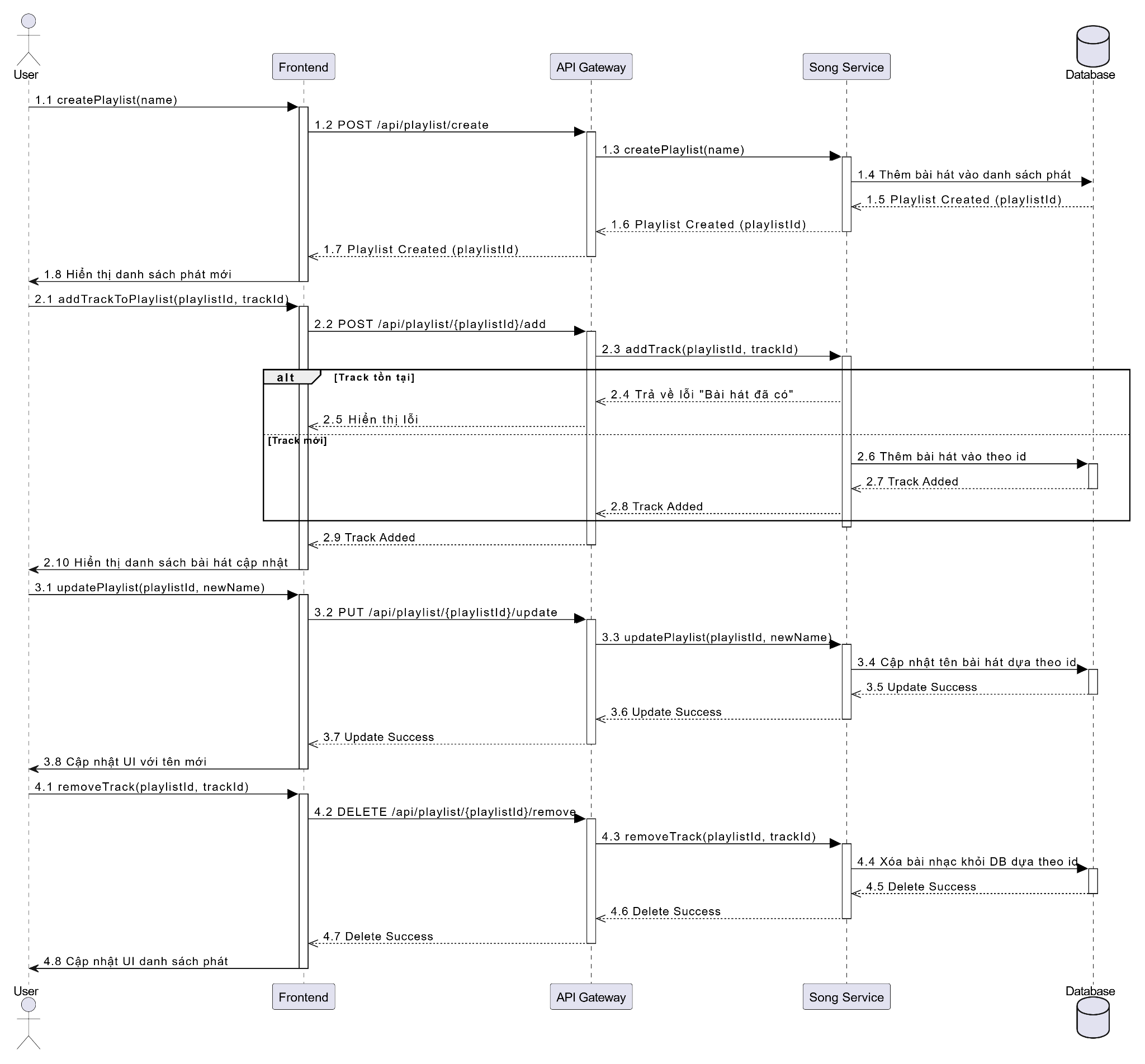


### Quản lý danh sách phát

* + 1. use case

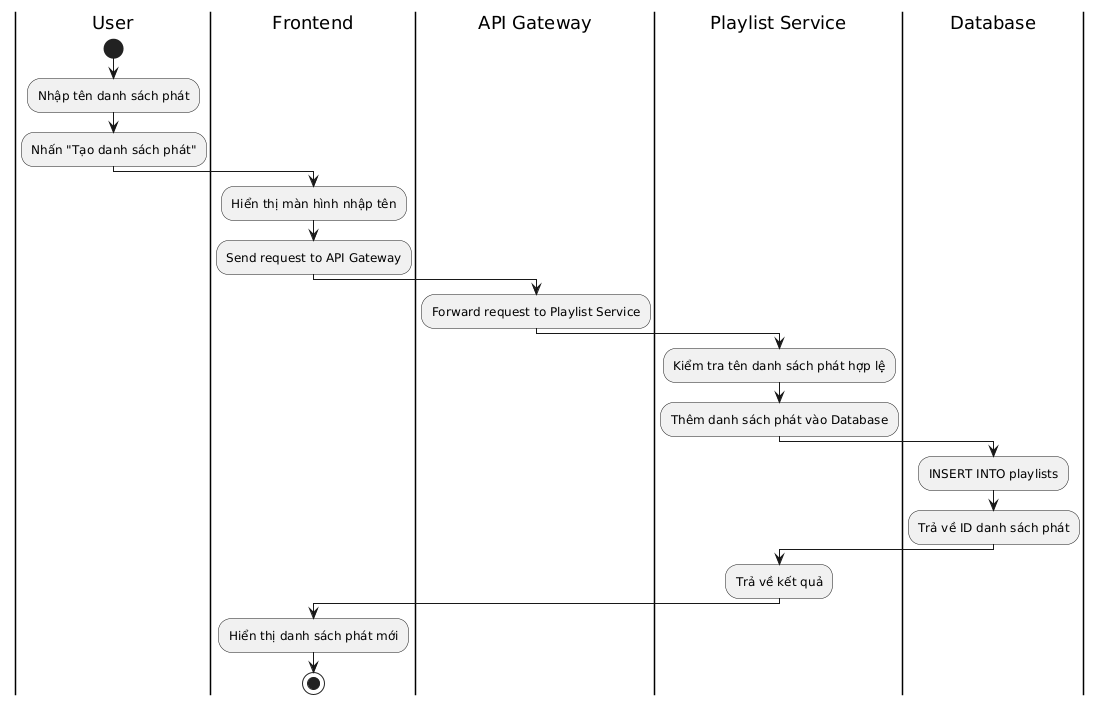


* + 1. Tạo danh sách phát nhạc

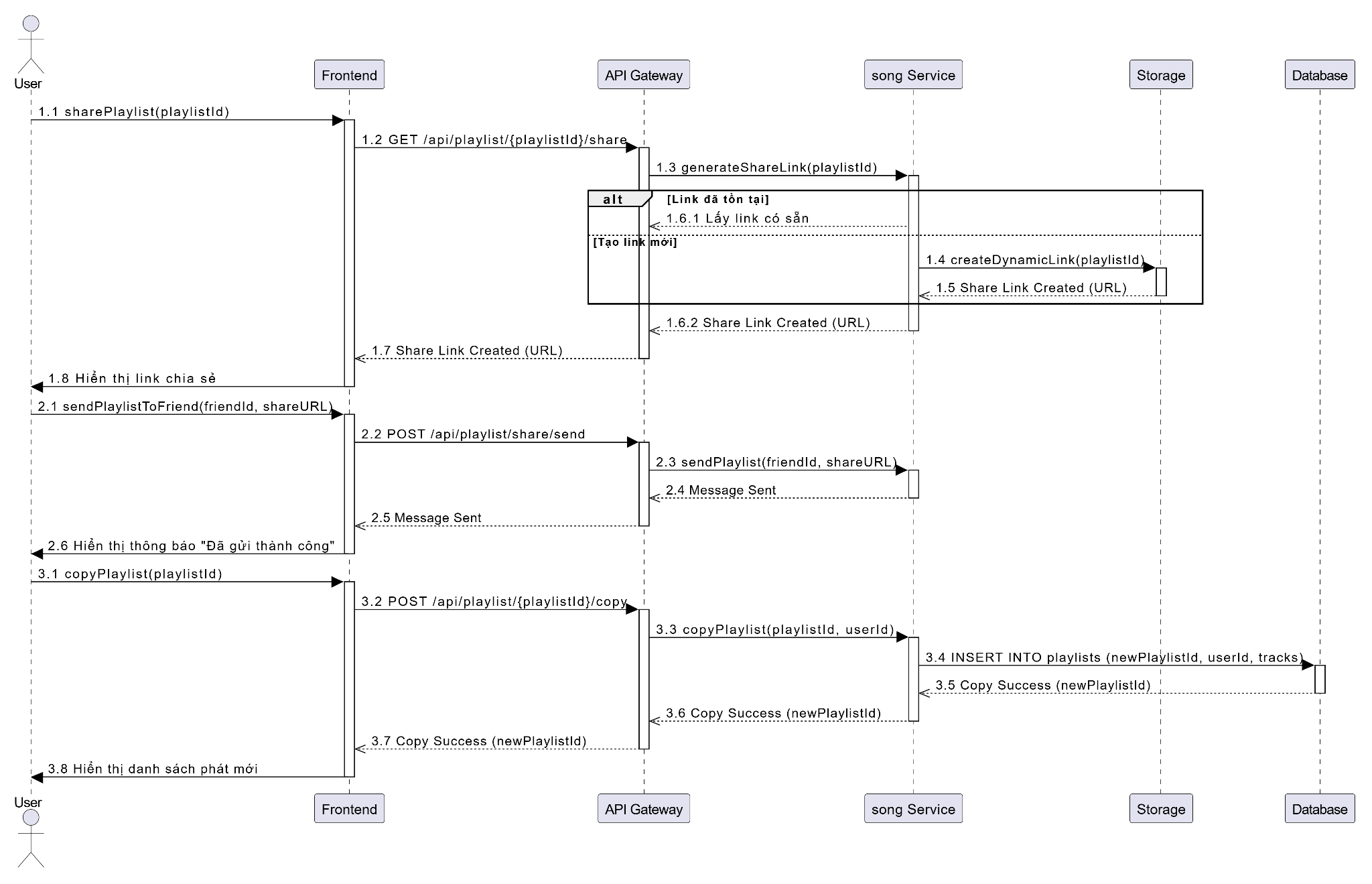


1. Front End
   1. Method: createPlaylist(name)
      1. Mục đích: Gửi yêu cầu tạo danh sách phát đến API Gateway.
      2. Input: name: String
      3. Output: playlistId: UUID
      4. Process:
         1. Người dùng nhập tên danh sách phát và nhấn “Tạo”.
         2. Frontend gửi POST /api/playlist/create đến API Gateway.
         3. Nhận phản hồi từ API và cập nhật giao diện.
   2. Method: addTrackToPlaylist(playlistId, trackId)
      1. Mục đích: Thêm bài hát vào danh sách phát.
      2. Input: playlistId: UUID, trackId: UUID
      3. Output: success: Boolean
      4. Process:
         1. Người dùng chọn bài hát để thêm vào danh sách.
         2. Gửi request POST /api/playlist/{playlistId}/add đến API Gateway.
         3. Nhận phản hồi từ API và cập nhật UI.
   3. Method: updatePlaylist(playlistId, newName)
      1. Mục đích: Chỉnh sửa tên danh sách phát.
      2. Input: playlistId: UUID, newName: String
      3. Output: success: Boolean
      4. Process:
         1. Người dùng chỉnh sửa tên danh sách phát.
         2. Gửi request PUT /api/playlist/{playlistId}/update đến API Gateway.
         3. Nhận phản hồi từ API và cập nhật giao diện.
   4. Method: removeTrack(playlistId, trackId)
      1. Mục đích: Xóa bài hát khỏi danh sách phát.
      2. Input: playlistId: UUID, trackId: UUID
      3. Output: success: Boolean
      4. Process:
         1. Người dùng nhấn xóa bài hát khỏi danh sách phát.
         2. Gửi request DELETE /api/playlist/{playlistId}/remove đến API Gateway.
         3. Nhận phản hồi và cập nhật giao diện.
2. API Gateway
   1. Method: createPlaylist(name)
      1. Mục đích: Chuyển yêu cầu tạo danh sách phát đến SongService.
      2. Input: name: String
      3. Output: playlistId: UUID
      4. Process:
         1. Nhận request từ Frontend.
         2. Gửi request POST /playlist/create đến SongService.
         3. Nhận phản hồi từ Service và gửi về Frontend.
   2. Method: addTrack(playlistId, trackId)
      1. Mục đích: Chuyển yêu cầu thêm bài hát vào SongService.
      2. Input: playlistId: UUID, trackId: UUID
      3. Output: success: Boolean
      4. Process:
         1. Nhận request từ Frontend.
         2. Gửi request POST /playlist/{playlistId}/add đến SongService.
         3. Nhận phản hồi từ Service và gửi về Frontend.
3. SongService
   1. Method: createPlaylist(name, userId)
      1. Mục đích: Lưu danh sách phát mới vào database.
      2. Input: name: String, userId: UUID
      3. Output: playlistId: UUID
      4. Process:
         1. Nhận request từ API Gateway.
         2. Gọi insertPlaylist(name, userId) trên Database.
         3. Nhận phản hồi từ Database và gửi về API Gateway.
   2. Method: addTrack(playlistId, trackId)
      1. Mục đích: Thêm bài hát vào danh sách phát.
      2. Input: playlistId: UUID, trackId: UUID
      3. Output: success: Boolean
      4. Process:
         1. Kiểm tra bài hát đã tồn tại chưa.
         2. Nếu chưa tồn tại, gọi insertTrack(playlistId, trackId) trên Database.
4. Database
   1. Method: insertPlaylist(name, userId)
      1. Mục đích: Thêm danh sách phát vào bảng playlists.
      2. Input: name: String, userId: UUID
      3. Output: playlistId: UUID
      4. Process:
         1. Thực hiện lệnh thêm mới
   2. Method: insertTrack(playlistId, trackId)
      1. Mục đích: Thêm bài hát vào danh sách phát.
      2. Input: playlistId: UUID, trackId: UUID
      3. Output: success: Boolean
      4. Process:
         1. Thực hiện câu lệnh chỉnh sửa

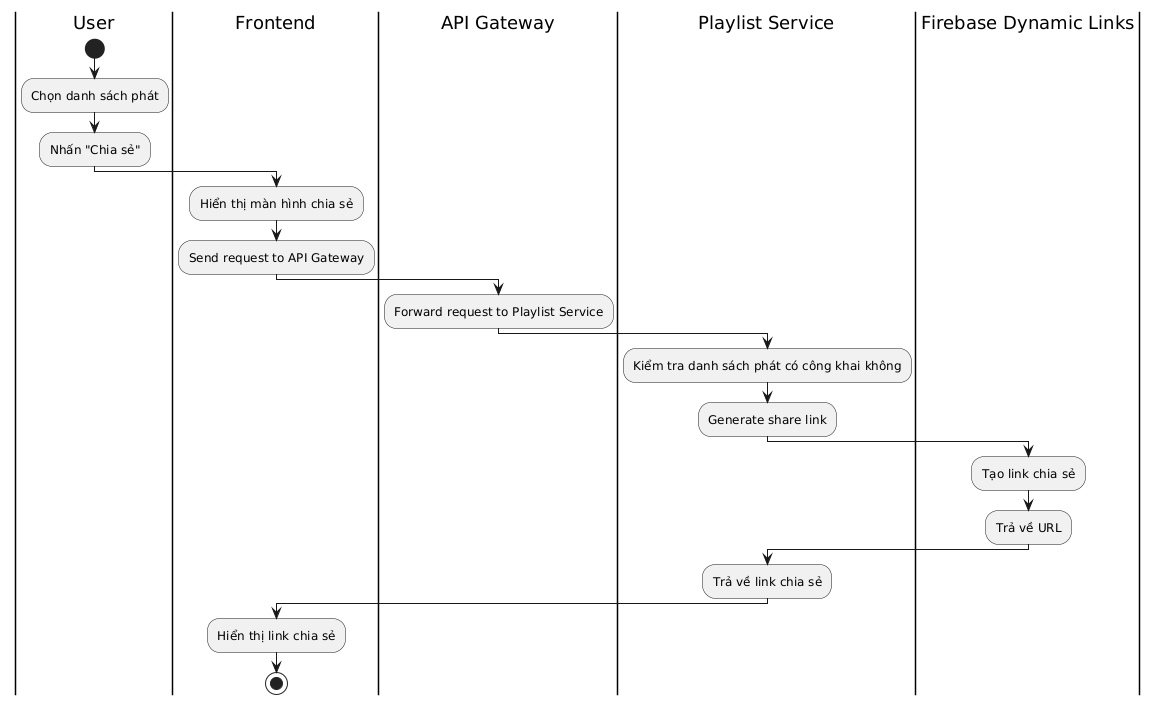
Activity



* + 1. Chia sẻ danh sách phát



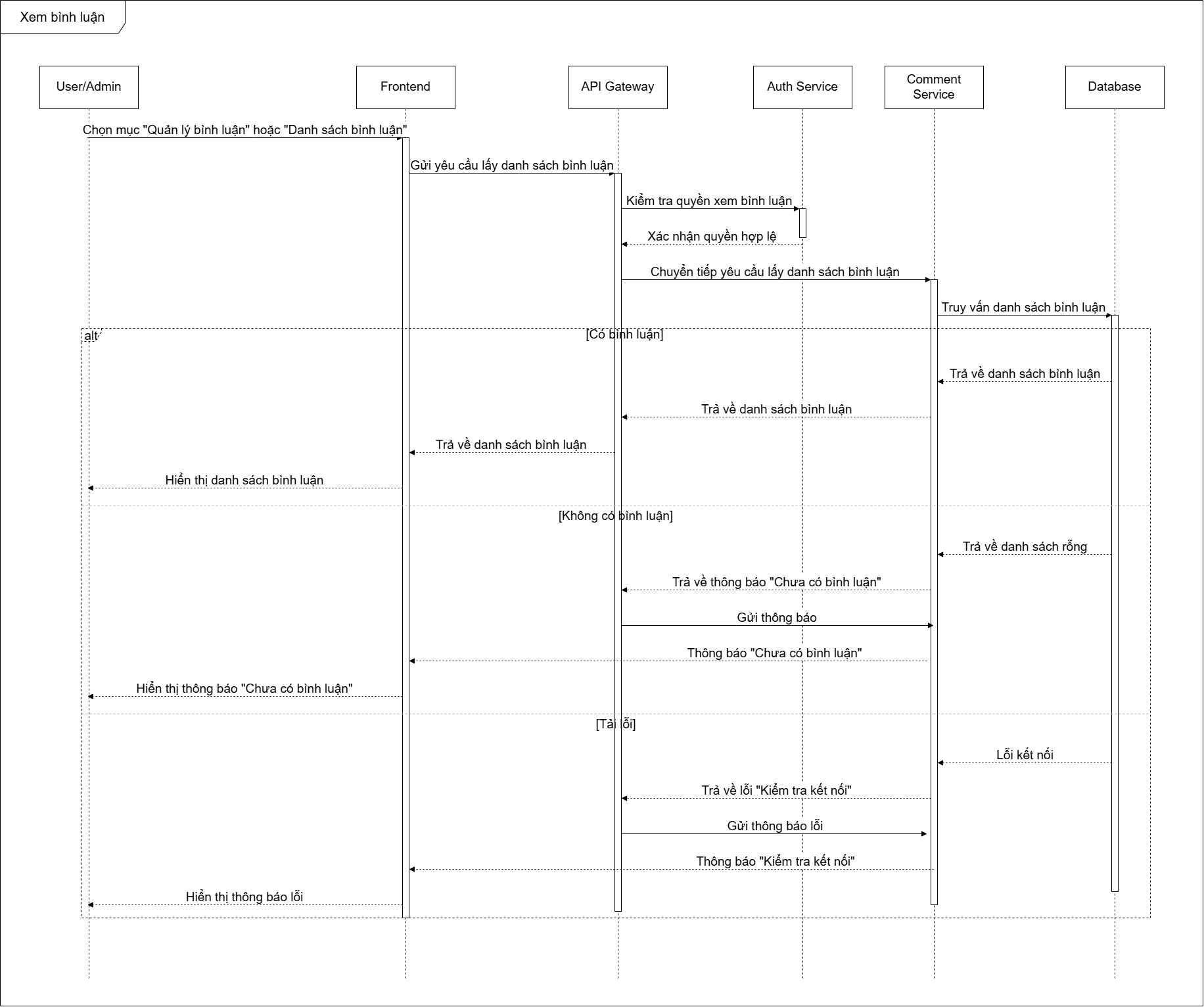
1. Frontend
   1. Tên method : sharePlaylist(playlistId)
   2. Input: playlistId: UUID
   3. Output: shareURL: String
   4. Process:
      1. Gửi request GET /api/playlist/{playlistId}/share đến API Gateway.
      2. Nhận phản hồi và hiển thị link trên UI.
2. API Gateway
   1. Method: generateShareLink(playlistId)
   2. Mục đích: Chuyển yêu cầu tạo link đến SongService.
   3. Input: playlistId: UUID
   4. Output: shareURL: String
   5. Process:
      1. Nhận request từ Frontend.
      2. Gửi request GET /playlist/{playlistId}/share đến SongService.
3. SongService
   1. Tên method : generateShareLink(playlistId)
   2. Mục đích: Tạo link chia sẻ danh sách phát.
   3. Input: playlistId: UUID
   4. Output: shareURL: String
   5. Process:
      1. Kiểm tra xem danh sách có công khai không.
      2. Nếu có, gọi createDynamicLink(playlistId).
4. Storage (Firebase )
   1. Tên method : createDynamicLink(playlistId)
   2. Mục đích: Tạo link động để chia sẻ.
   3. Input: playlistId: UUID
   4. Output: shareURL: String
   5. Process:
      1. Gửi request đến Firebase Dynamic Links.
      2. Firebase trả về link ngắn để chia sẻ.

activity  


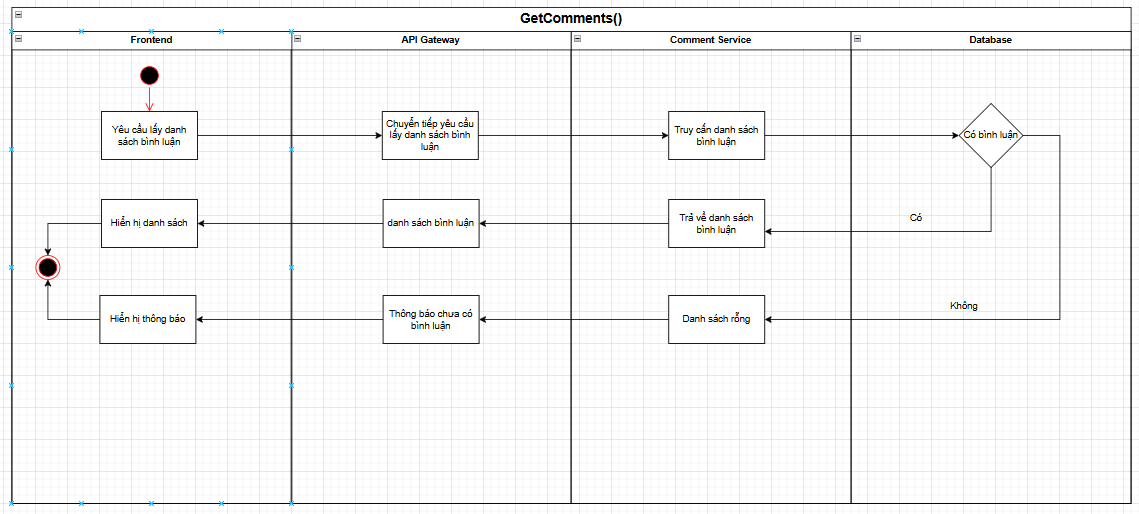
* 1. **Quản lý bình luận**
     1. **Use case**

### 

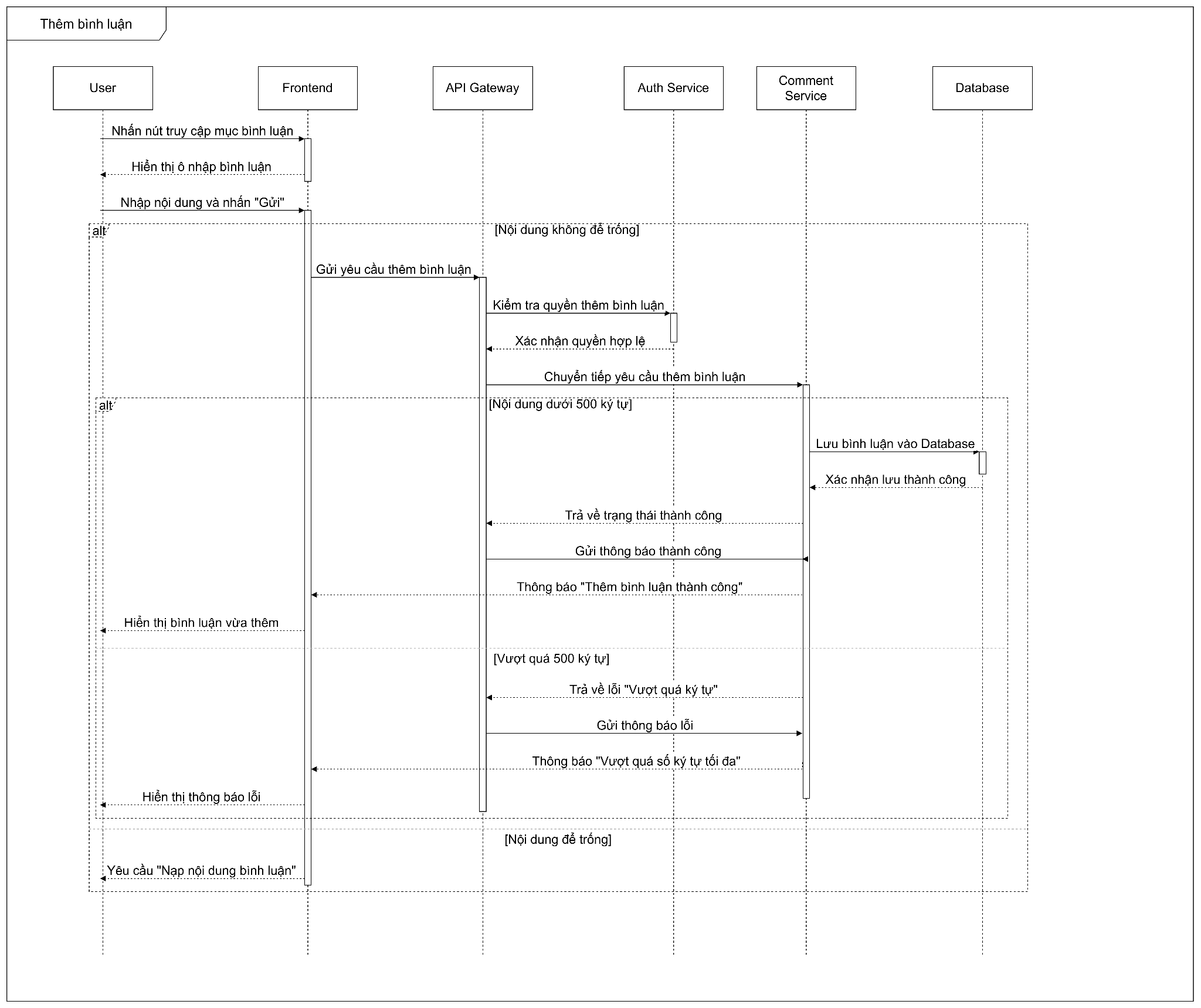
* + 1. Xem tất cả bình luận



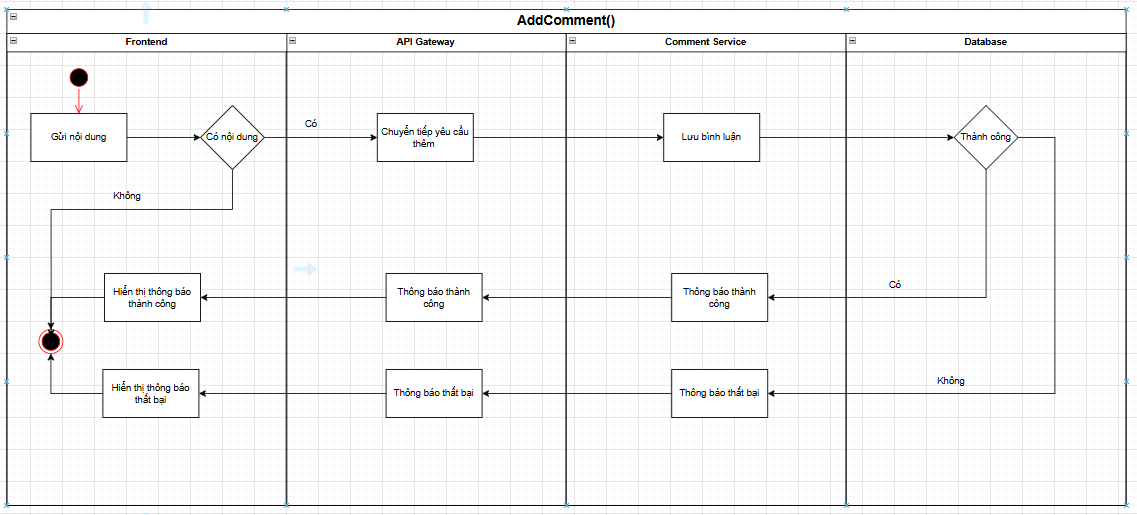
* Tên Method (Mục đích, Input, Output, Process) của Frontend
  + Tên Method: GetAllComments
  + Mục đích: Gửi yêu cầu để lấy danh sách tất cả bình luận của một bài viết.
  + Input:
    - ID của bài viết (postId) để xác định bài viết cần lấy bình luận.
    - Ví dụ: { postId: "123" }.
  + Output:
    - Danh sách các bình luận của bài viết (nếu thành công) hoặc thông báo lỗi (nếu thất bại).
    - Ví dụ:
    - Thành công: { status: "success", comments: [{ id: "1", content: "Bình luận 1", userId: "456", createdAt: "2025-03-12T10:00:00Z" }, ...] }.
    - Thất bại: { status: "error", message: "Bài viết không tồn tại" } hoặc { status: "error", message: "Không tìm thấy bài" }.
  + Process:
    - Frontend nhận yêu cầu từ người dùng (ví dụ: người dùng nhấn vào nút "Xem bình luận" của một bài viết).
    - Gửi yêu cầu HTTP (GET) đến API Gateway với thông tin bài viết (postId).
    - Nhận phản hồi từ API Gateway, bao gồm danh sách bình luận hoặc thông báo lỗi.
    - Hiển thị danh sách bình luận hoặc thông báo lỗi cho người dùng trên giao diện.
* HTTP Method
  + Dựa trên hành động "xem tất cả bình luận", đây là thao tác truy vấn dữ liệu (không thay đổi trạng thái của hệ thống).
  + HTTP Method: GET
    - Lý do: GET được sử dụng để lấy dữ liệu từ server, phù hợp với việc truy vấn danh sách bình luận.
* Backend
  + Backend: Comment Service
    - Trong sơ đồ, Comment Service là thành phần xử lý logic nghiệp vụ liên quan đến bình luận, bao gồm việc kiểm tra trạng thái bài viết và lấy dữ liệu từ Comment DB.
    - API Gateway chỉ đóng vai trò định tuyến, nên Comment Service chính là backend xử lý yêu cầu.



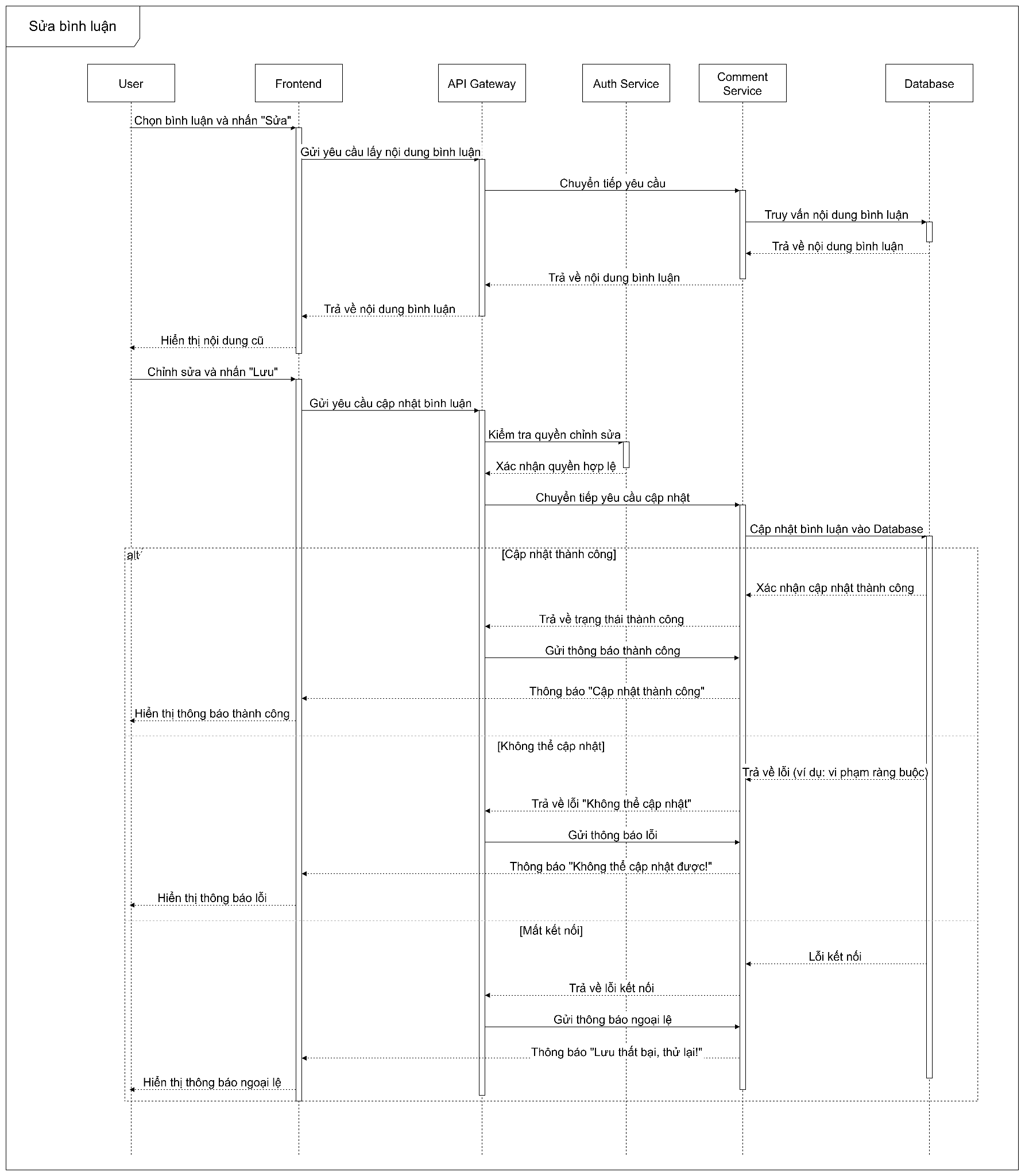
* + 1. **Thêm bình luận**



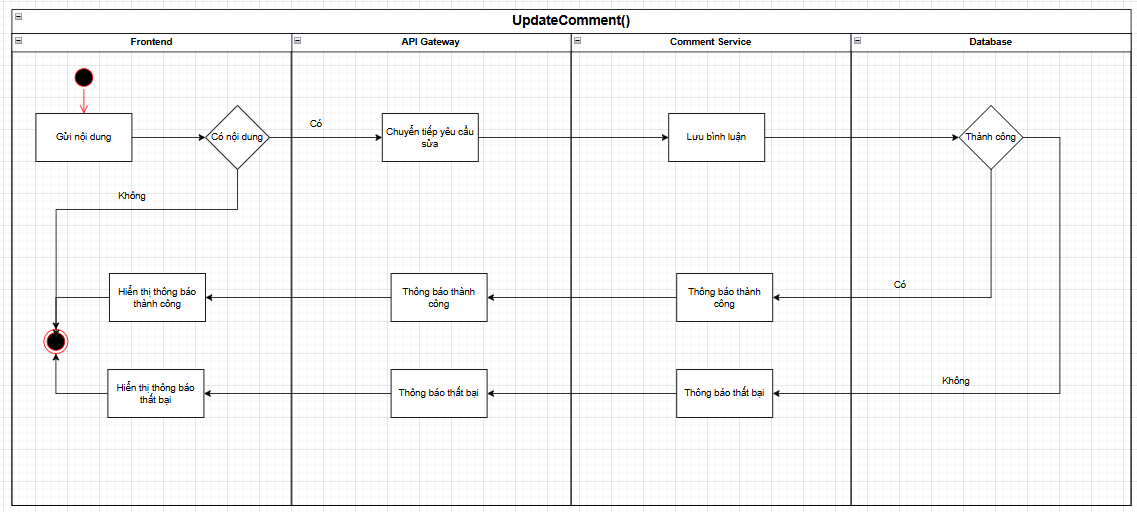
* Tên Method (Mục đích, Input, Output, Process) của Frontend
  + Tên Method: CreateComment
  + Mục đích: Gửi yêu cầu để thêm một bình luận mới cho bài viết.
  + Input:
    - Dữ liệu bình luận (bao gồm nội dung bình luận, ID bài viết, ID người dùng, thời gian tạo, v.v.).
    - Ví dụ: { postId: "123", userId: "456", content: "Đây là bình luận mới", createdAt: "2025-03-12T10:00:00Z" }.
  + Output:
    - Kết quả thành công hoặc thất bại từ API Gateway (thường là mã trạng thái HTTP và thông báo).
    - Ví dụ:
    - Thành công: { status: "success", message: "Bình luận đã được thêm" }.
    - Thất bại: { status: "error", message: "Thêm bình luận thất bại" }.
  + Process:
    - Frontend nhận dữ liệu bình luận từ người dùng (qua form hoặc giao diện nhập liệu).
    - Gửi yêu cầu HTTP (POST) đến API Gateway với dữ liệu bình luận.
    - Nhận phản hồi từ API Gateway và hiển thị kết quả cho người dùng (thông báo thành công hoặc lỗi).
* HTTP Method
  + Dựa trên hành động "thêm bình luận mới", đây là thao tác tạo mới dữ liệu.
  + HTTP Method: POST
    - Lý do: POST được sử dụng để tạo mới tài nguyên (resource) trên server, phù hợp với việc thêm một bình luận mới.
* Backend
  + Backend: Comment Service
  + Trong sơ đồ, Comment Service là thành phần xử lý logic nghiệp vụ liên quan đến bình luận, bao gồm việc kiểm tra dữ liệu và lưu bình luận mới vào Comment DB.
    - API Gateway chỉ đóng vai trò định tuyến, nên Comment Service chính là backend xử lý yêu cầu.



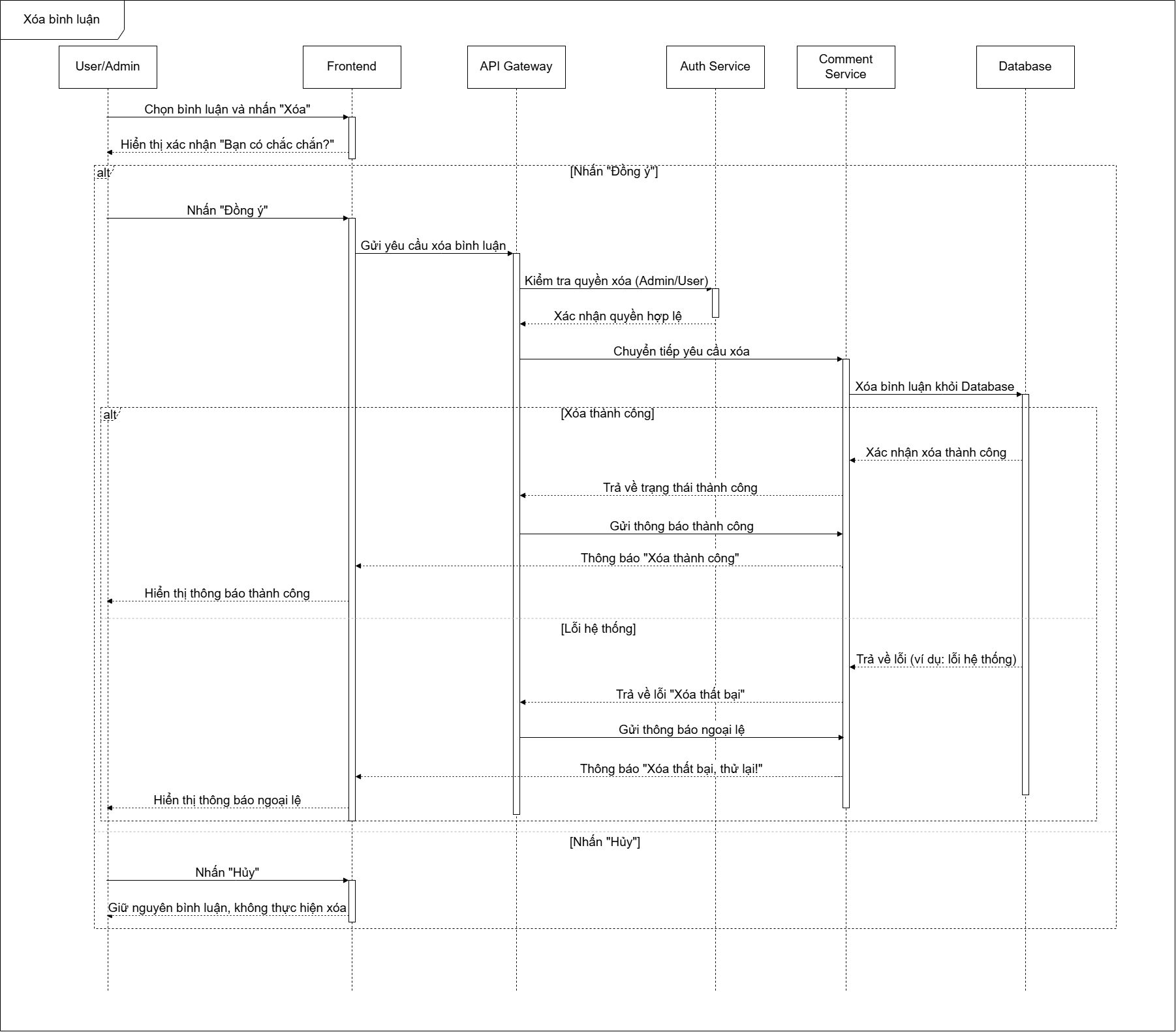
* + 1. **Sửa bình luận**



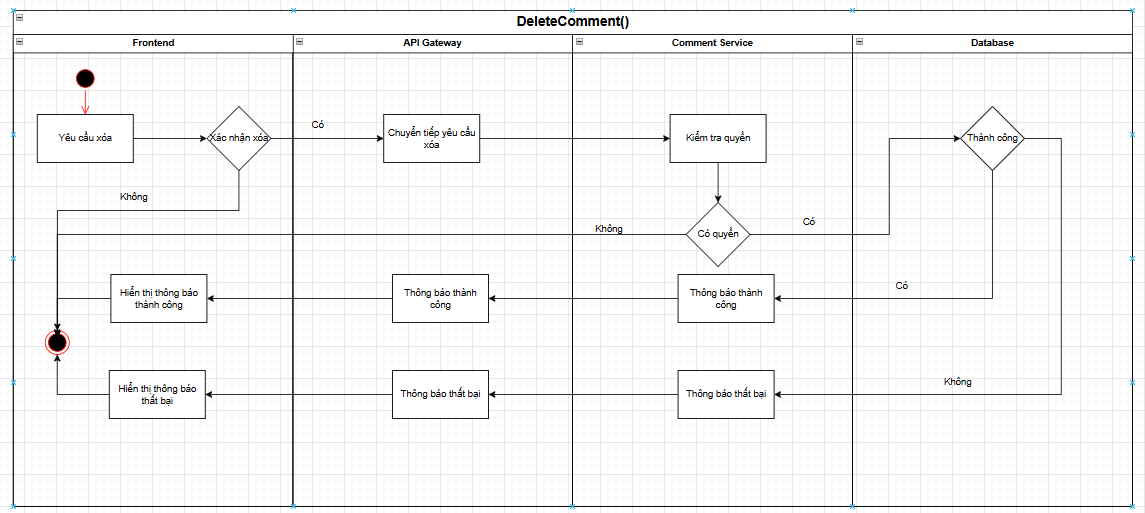
* Tên Method (Mục đích, Input, Output, Process) của Frontend
  + Tên Method: UpdateComment
  + Mục đích: Gửi yêu cầu để chỉnh sửa một bình luận đã có trong hệ thống.
  + Input:
    - ID của bình luận (commentId) để xác định bình luận cần sửa.
    - Dữ liệu bình luận đã chỉnh sửa (có thể bao gồm nội dung bình luận mới, thời gian cập nhật, v.v.).
    - Ví dụ: { commentId: "789", content: "Nội dung bình luận đã sửa", updatedAt: "2025-03-12T10:00:00Z" }.
  + Output:
    - Kết quả thành công hoặc thất bại từ API Gateway (thường là mã trạng thái HTTP và thông báo).
    - Ví dụ:
    - Thành công: { status: "success", message: "Bình luận đã được sửa" }.
    - Thất bại: { status: "error", message: "Không tìm thấy bình luận" }.
  + Process:
    - Frontend nhận dữ liệu bình luận đã chỉnh sửa từ Admin (qua form hoặc giao diện chỉnh sửa).
    - Gửi yêu cầu HTTP (PUT hoặc PATCH) đến API Gateway với dữ liệu bình luận đã chỉnh sửa.
    - Nhận phản hồi từ API Gateway và hiển thị kết quả cho Admin (thông báo thành công hoặc lỗi).
    - Cập nhật giao diện (nếu cần) để hiển thị nội dung bình luận đã sửa.
* HTTP Method
  + Dựa trên hành động "sửa bình luận", đây là thao tác cập nhật dữ liệu.
  + HTTP Method: PUT hoặc PATCH
  + Lý do:
    - PUT được sử dụng để cập nhật toàn bộ tài nguyên (thay thế toàn bộ dữ liệu bình luận).
    - PATCH được sử dụng để cập nhật một phần dữ liệu của tài nguyên (chỉ thay đổi các trường cần sửa, ví dụ: chỉ cập nhật nội dung bình luận).
    - Trong ngữ cảnh này, vì sơ đồ không chỉ rõ mức độ cập nhật, tôi sẽ chọn PUT như phương thức phổ biến hơn cho việc cập nhật toàn bộ bình luận. Tuy nhiên, nếu chỉ cập nhật một phần (ví dụ: chỉ nội dung), PATCH cũng có thể được sử dụng.
* Backend
  + Backend: Student Service
  + Trong sơ đồ, Student Service là thành phần xử lý logic nghiệp vụ liên quan đến bình luận (mặc dù tên gọi có thể gây nhầm lẫn, nhưng trong ngữ cảnh này, nó xử lý việc sửa bình luận).
  + Student Service kiểm tra tính hợp lệ và lưu dữ liệu vào Student DB.
  + API Gateway chỉ đóng vai trò định tuyến, nên Student Service chính là backend xử lý yêu cầu.



* + 1. **Xóa bình luận**

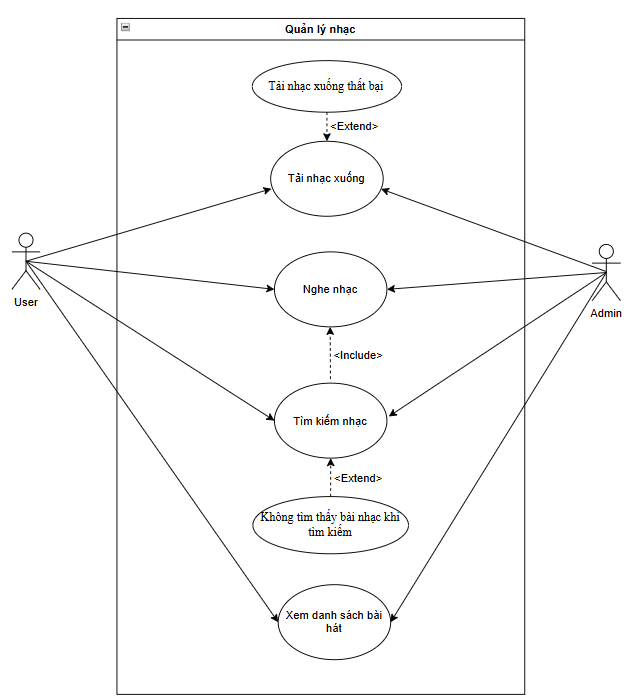


* Tên Method (Mục đích, Input, Output, Process) của Frontend
  + Tên Method: DeleteComment
  + Mục đích: Gửi yêu cầu để xoá một bình luận đã có trong hệ thống.
  + Input:
    - ID của bình luận (commentId) để xác định bình luận cần xoá.
    - Ví dụ: { commentId: "789" }.
  + Output:
    - Kết quả thành công hoặc thất bại từ API Gateway (thường là mã trạng thái HTTP và thông báo).
    - Ví dụ:
    - Thành công: { status: "success", message: "Bình luận đã được xoá" }.
    - Thất bại: { status: "error", message: "Không tìm thấy bình luận" }.
  + Process:
    - Frontend nhận yêu cầu xoá bình luận từ Admin (qua nút xoá hoặc giao diện tương tác).
    - Gửi yêu cầu HTTP (DELETE) đến API Gateway với ID của bình luận.
    - Nhận phản hồi từ API Gateway và hiển thị kết quả cho Admin (thông báo thành công hoặc lỗi).
    - Cập nhật giao diện (nếu cần) để phản ánh việc bình luận đã được xoá.
* HTTP Method
  + Dựa trên hành động "xoá bình luận", đây là thao tác xóa dữ liệu.
  + HTTP Method: DELETE
  + Lý do: DELETE được sử dụng để xóa một tài nguyên (resource) trên server, phù hợp với việc xoá một bình luận.
* Backend
  + Backend: Comment Service
  + Trong sơ đồ, Comment Service là thành phần xử lý logic nghiệp vụ liên quan đến bình luận, bao gồm việc kiểm tra và xoá bình luận khỏi Comment DB.
  + API Gateway chỉ đóng vai trò định tuyến, nên Comment Service chính là backend xử lý yêu cầu.



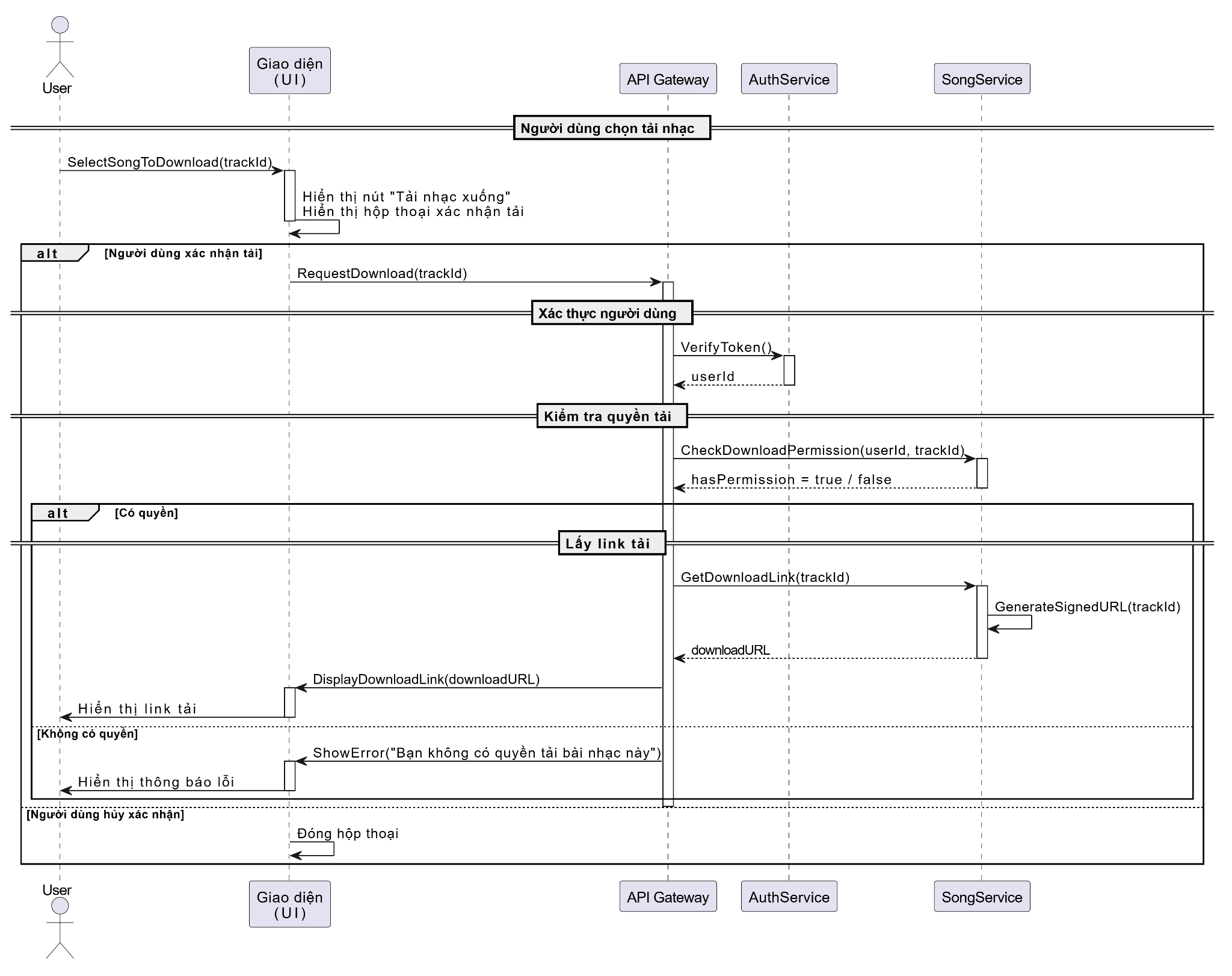
**4.3.4 Quản lý nhạc**

**Use Case quản lý nhạc**



**4.3.4.1 Tải nhạc xuống**

**1.**  **Sequence Diagram:**

****

**2. Giao diện**

Các phương thức

* Method:SelectSongToDownload(trackId)  
  + Mục đích: Cho phép người dùng chọn một bài hát và yêu cầu tải về.
  + Input: trackId – ID của bài hát được chọn.
  + Output: Giao diện hiển thị hộp thoại xác nhận.
  + Process:
    1. Người dùng chọn bài hát từ danh sách nhạc.
    2. Giao diện hiển thị nút “Tải nhạc xuống”.
    3. Khi người dùng nhấn vào nút đó, một hộp thoại xác nhận tải nhạc được hiển thị.
    4. Nếu người dùng xác nhận, giao diện sẽ gửi yêu cầu đến API Gateway để tiếp tục xử lý việc tải
* Method: DisplayDownloadLink(downloadURL)  
  + Mục đích: Hiển thị đường dẫn tải nhạc cho người dùng sau khi được xác thực và cấp phép.
  + Input: downloadURL – Đường dẫn tải nhạc hợp lệ do hệ thống trả về.
  + Output: Nút hoặc link tải để người dùng bấm vào và tải nhạc.
  + Process:
    1. Nhận downloadURL từ API Gateway.
    2. Giao diện hiển thị nút tải nhạc tương ứng cho người dùng.
    3. Khi người dùng nhấn vào, file nhạc sẽ được tải xuống thiết bị.

**3. Dịch vụ**

Các phương thức

* Method: RequestDownload(trackId)  
  + Mục đích: Tiếp nhận yêu cầu tải nhạc, xác thực người dùng và kiểm tra quyền truy cập.
  + Input: trackId, userId (lấy từ token sau xác thực).
  + Output: downloadURL hoặc thông báo lỗi.
  + Process:
    1. API Gateway gọi AuthService để xác thực token → lấy userId.
    2. Gọi SongService.CheckDownloadPermission(userId, trackId) để kiểm tra quyền tải.
    3. Nếu được phép: tiếp tục gọi SongService.GetDownloadLink(trackId) để lấy link tải.
    4. Nếu không: trả lỗi về giao diện.
* Method: CheckDownloadPermission(userId, trackId)  
  + Mục đích: Kiểm tra người dùng có được phép tải bài hát hay không.
  + Input: userId, trackId
  + Output: true hoặc false
  + Process:Kiểm tra quyền sở hữu bài hát, quyền mua/nghe offline hoặc các ràng buộc bản quyền.
* Method: GetDownloadLink(trackId)  
  + Mục đích: Lấy đường dẫn tải nhạc (có thể có thời hạn).
  + Input: trackId.
  + Output: downloadURL.
  + Process:Gọi đến module tạo link tải nhạc nội bộ (GenerateSignedURL) và trả về.

**4. Lưu trữ**

Các phương thức

* Method: GenerateSignedURL(trackId)
  + Mục đích: Tạo một đường dẫn tải nhạc có thời hạn bảo mật.
  + Input: trackId
  + Output: signedURL – link tải nhạc dạng tạm thời
  + Process:
    1. Truy xuất thông tin file nhạc tương ứng với trackId.
    2. Sử dụng dịch vụ lưu trữ như AWS S3 / Google Cloud Storage để tạo một URL tải nhạc có thời gian sống ngắn (1h, 24h,...).
    3. Trả link về cho SongService.

**5. Activity Diagram**

****

**4.3.4.2. Nghe nhạc**

**1. Sequence Diagram**



**2. Giao diện**

Phương thức

* Method: SelectSongToPlay(trackId)
  + Mục đích: Cho phép người dùng chọn một bài hát để nghe trực tuyến.
  + Input: trackId
  + Output: Giao diện hiển thị nút phát nhạc.
  + Process:
    1. Người dùng chọn một bài hát.
    2. Hiển thị nút “Phát”.
    3. Khi nhấn, gửi yêu cầu lên hệ thống qua API Gateway.
* Method: DisplayPlayer(streamURL)
  + Mục đích: Hiển thị trình phát nhạc với URL được cung cấp.
  + Input: streamURL
  + Output: Giao diện phát nhạc
  + Process:
    1. Nhận streamURL từ hệ thống.
    2. Hiển thị trình phát nhạc hoặc nhúng player.
    3. Tự động phát hoặc chờ người dùng nhấn play.

**3. Dịch vụ**

Phương thức

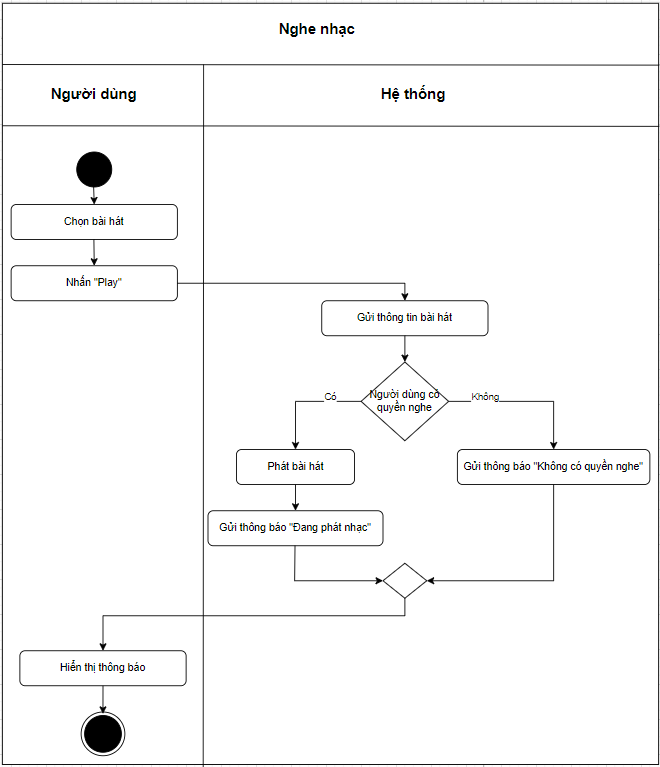
* Method: RequestStream(trackId)
  + Mục đích: Tiếp nhận yêu cầu nghe nhạc từ người dùng.
  + Input: trackId, userId (xác định qua token)
  + Output: streamURL hoặc lỗi
  + Process:
    1. Xác thực người dùng qua AuthService.
    2. Kiểm tra quyền nghe bài hát.
    3. Nếu hợp lệ, gọi GetStreamURL(trackId) để lấy link stream.
    4. Trả kết quả về cho UI.
* Method: CheckStreamPermission(userId, trackId)
  + Mục đích: Kiểm tra người dùng có quyền nghe bài hát không.
  + Input: userId, trackId
  + Output: true hoặc false
  + Process:Kiểm tra các điều kiện về quyền truy cập hoặc bài hát miễn phí.
* Method: GetStreamURL(trackId)
  + Mục đích: Tạo đường dẫn phát nhạc dạng tạm thời (nội bộ).
  + Input: trackId
  + Output: streamURL
  + Process:Gọi hàm GenerateSignedStreamURL(trackId) và trả về kết quả.

**4. Lưu trữ**

Phương thức

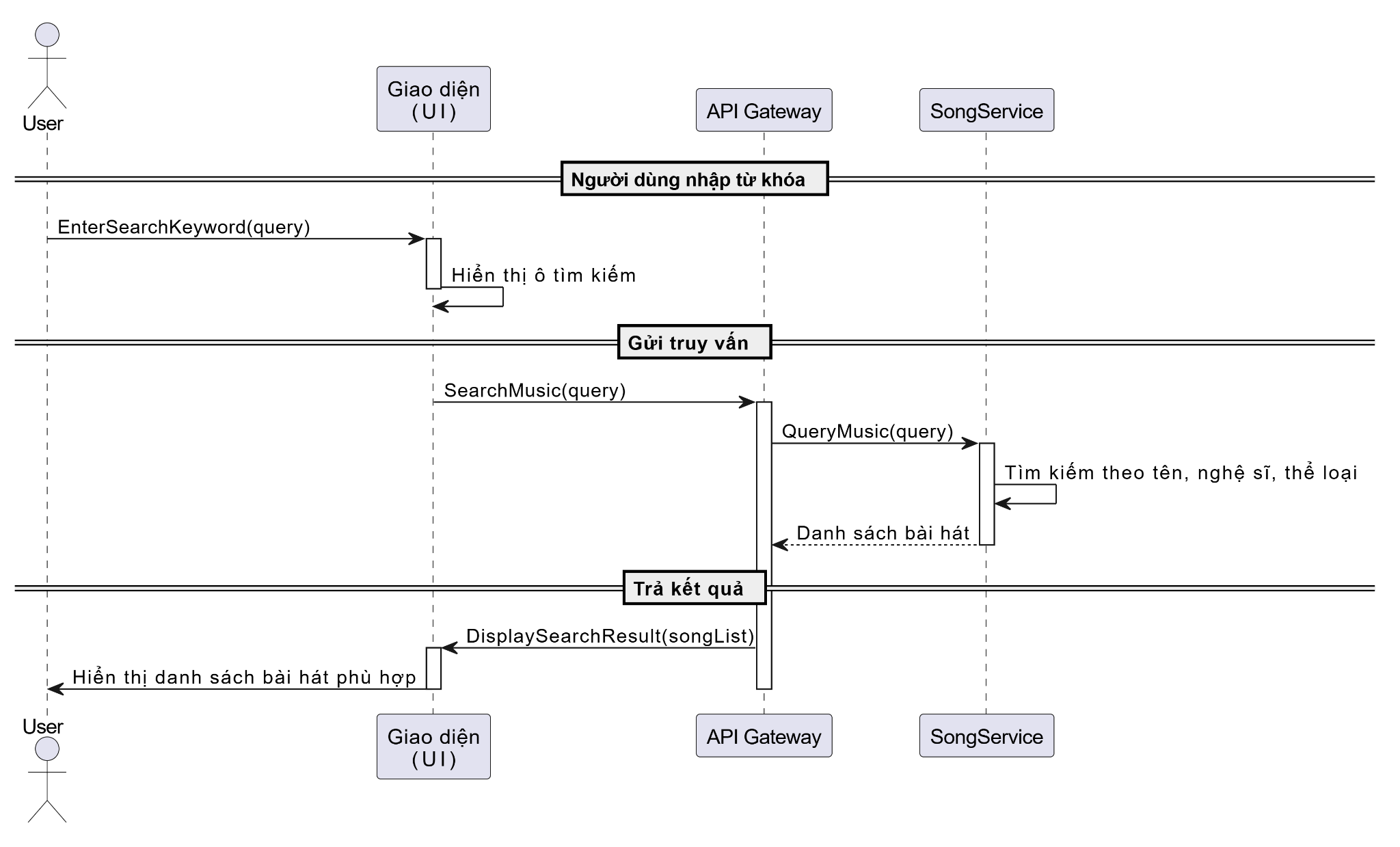
* Method: GenerateSignedStreamURL(trackId)
  + Mục đích: Tạo một URL phát nhạc bảo mật và có thời hạn.
  + Input: trackId
  + Output: streamURL
  + Process:
    1. Truy xuất thông tin bài hát tương ứng.
    2. Sinh ra URL từ hệ thống lưu trữ (nội bộ).
    3. Trả về URL dùng trong player.

**5. Activity Diagram**

****

**4.3.4.3. Tìm kiếm nhạc**

**1. Sequence Diagram**



**2. Giao diện**

Các phương thức

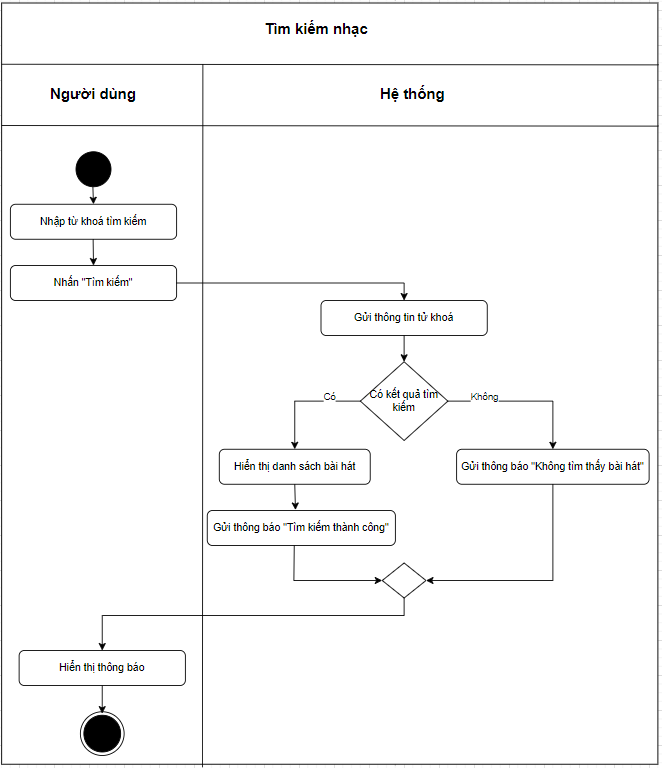
* Method: EnterSearchKeyword(query)
  + Mục đích: Cho phép người dùng nhập từ khóa để tìm nhạc.
  + Input: query – từ khóa tìm kiếm.
  + Output: Giao diện kết quả hoặc thông báo không tìm thấy.
  + Process:
    1. Hiển thị ô nhập tìm kiếm.
    2. Gửi truy vấn khi người dùng nhấn "Enter" hoặc click nút tìm.
    3. Hiển thị kết quả tìm kiếm hoặc thông báo lỗi.
* Method: DisplaySearchResult(songList)
  + Mục đích: Hiển thị danh sách bài hát phù hợp.
  + Input: songList – danh sách bài hát từ hệ thống.
  + Output: Danh sách bài hát hiển thị trong UI.
  + Process: Giao diện hiển thị danh sách bài hát có thể phát hoặc tải.

**3. Dịch vụ**

Các phương thức

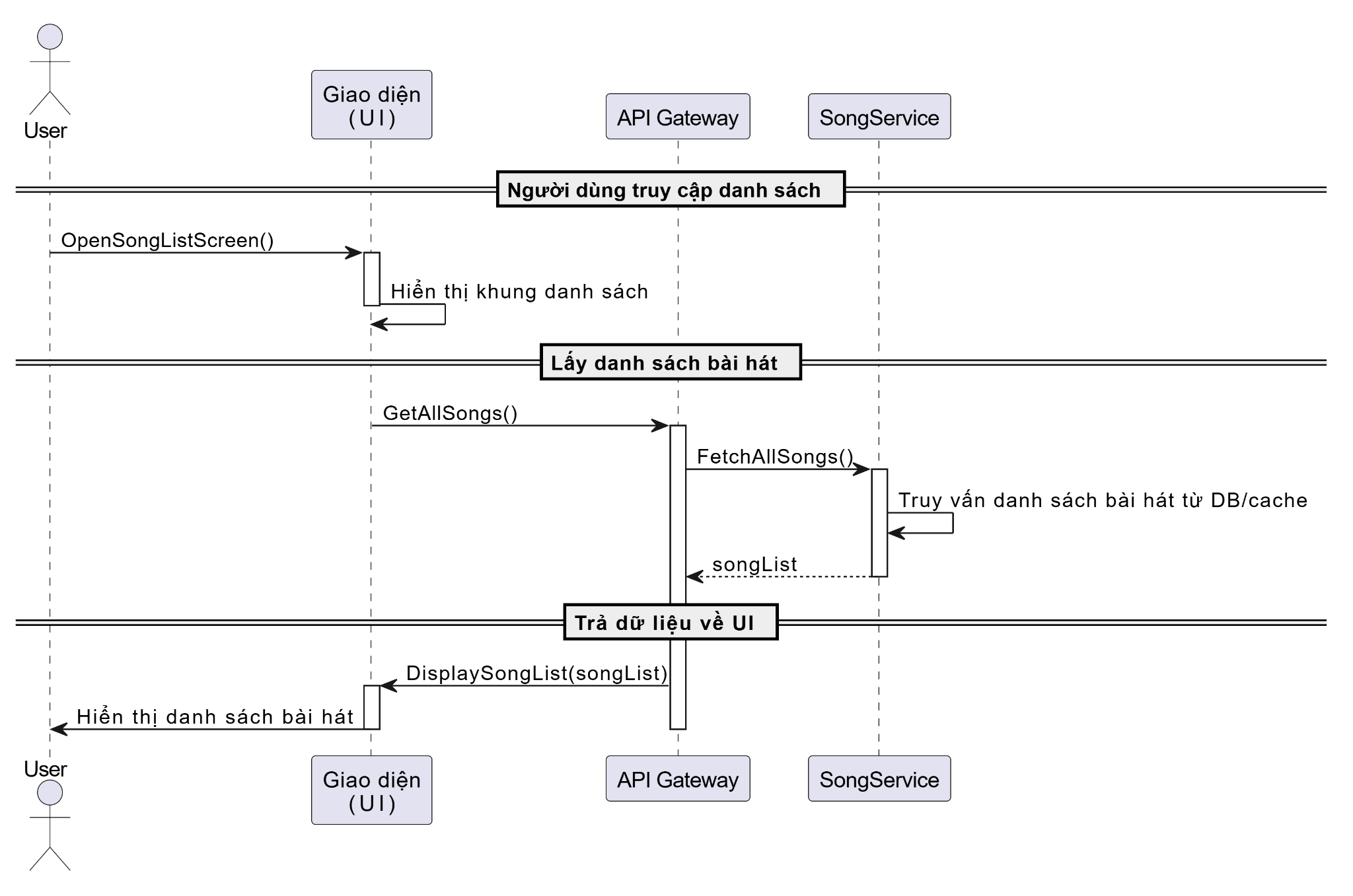
* Method: SearchMusic(query)
  + Mục đích: Xử lý yêu cầu tìm kiếm từ người dùng.
  + Input: query – chuỗi ký tự nhập từ giao diện.
  + Output: songList – kết quả phù hợp.
  + Process:
    1. Nhận truy vấn từ UI thông qua API Gateway.
    2. Gọi QueryMusic(query) trong SongService.
* Method: QueryMusic(query)
  + Mục đích: Tìm kiếm nhạc theo từ khóa.
  + Input: query
  + Output: Danh sách bài hát.
  + Process:
    1. Tìm kiếm trong hệ thống theo tên bài hát, nghệ sĩ, thể loại.
    2. Có thể ưu tiên gợi ý thông minh hoặc phân trang kết quả.
    3. Trả danh sách phù hợp về cho API Gateway.

**4. Activity Diagram**

****

**4.3.4.4 Xem danh sách bài hát**

**1. Sequence Diagram**

****

**2. Giao diện**

Các phương thức

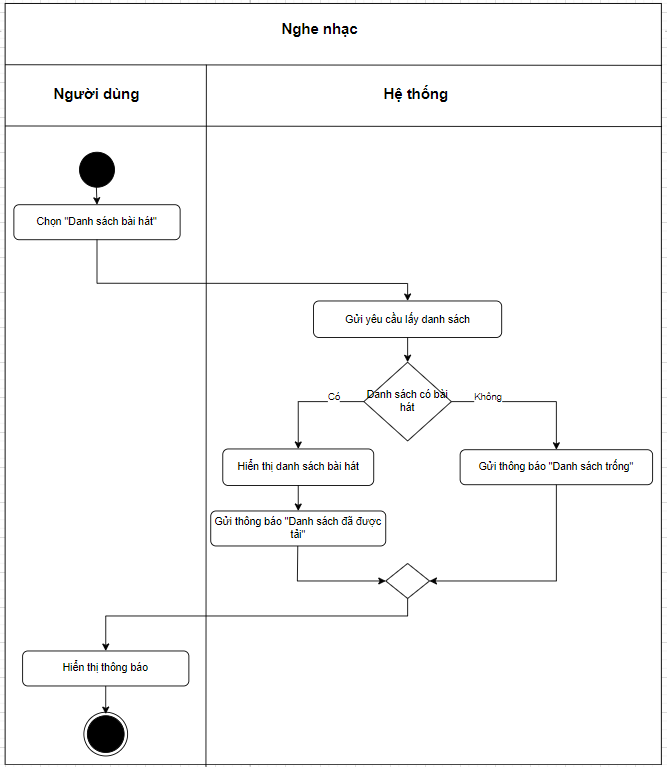
* Method: OpenSongListScreen()
  + Mục đích: Hiển thị danh sách bài hát cho người dùng.
  + Input: Không có, được kích hoạt khi mở trang.
  + Output: Giao diện danh sách.
  + Process:
    1. Giao diện gửi yêu cầu lấy danh sách bài hát khi người dùng mở trang.
    2. Gửi GetAllSongs() đến API Gateway.
* Method: DisplaySongList(songList)
  + Mục đích: Hiển thị danh sách bài hát nhận được từ hệ thống.
  + Input: songList – danh sách bài hát.
  + Output: Danh sách hiển thị trên UI.
  + Process:Hiển thị các bài hát theo grid hoặc list, có thể bấm để nghe hoặc tải.

**3. Dịch vụ**

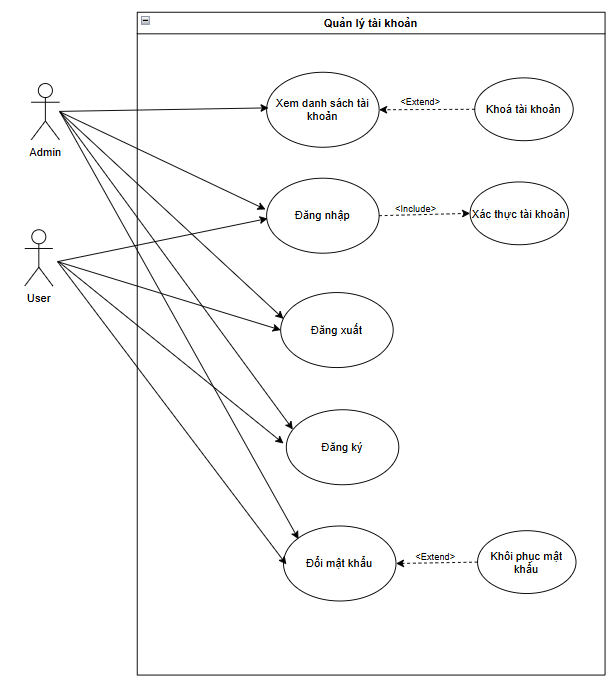
Các phương thức

* Method: GetAllSongs()
  + Mục đích: Yêu cầu danh sách tất cả các bài hát có sẵn.
  + Input: Không có.
  + Output: songList
  + Process:Gửi yêu cầu đến SongService.FetchAllSongs().
* Method: FetchAllSongs()
  + Mục đích: Truy vấn danh sách bài hát từ hệ thống.
  + Input: Không có.
  + Output: songList
  + Process:
    1. Kiểm tra cache để lấy dữ liệu nhanh nếu có.
    2. Nếu không có cache, truy vấn trực tiếp từ cơ sở dữ liệu.
    3. Trả về danh sách bài hát.

**4. Activity Diagram**

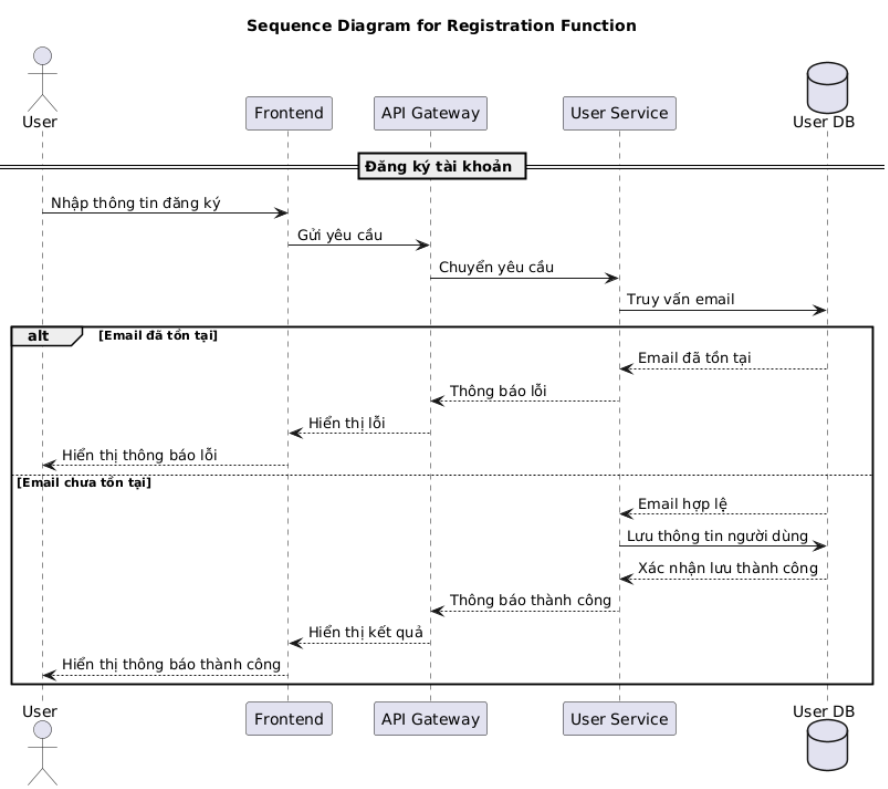
****

## 4.3.5 Quản lý tài khoản(phi)



### 4.3.5.1 Đăng ký

1. **Sequence diagram.**

****

Tên Method: RegisterUser()

Mục đích:

* Gửi yêu cầu đăng ký tài khoản mới của người dùng cho server.

Input:

* Dữ liệu đăng ký từ người dùng, bao gồm các thông tin như:
  + Email: Địa chỉ email của người dùng.
  + Password: Mật khẩu do người dùng nhập.
  + Name: Tên của người dùng (tùy chọn).
  + SDT: Số điện thoại của người dùng (tùy chọn).

Output:

* Kết quả trả về từ hệ thống, bao gồm:
  + Trường hợp thành công:
    - Trạng thái: "success".
    - Thông báo: "Đăng ký thành công".
    - Dữ liệu: Thông tin người dùng đã đăng ký (ví dụ: userId, email).
  + Trường hợp thất bại:
    - Trạng thái: "error".
    - Thông báo: Lý do thất bại (ví dụ: "Email đã tồn tại").

Process:

1. Frontend nhận dữ liệu đăng ký từ người dùng:
   * Người dùng điền thông tin đăng ký (email, mật khẩu, tên, số điện thoại, v.v.) qua form hoặc giao diện nhập liệu.
2. Gửi yêu cầu HTTP (POST) đến API Gateway:
   * Frontend gửi yêu cầu HTTP POST đến API Gateway với dữ liệu đăng ký đã nhập.
3. Nhận phản hồi từ API Gateway:
   * Frontend nhận phản hồi từ API Gateway, bao gồm kết quả thành công hoặc thất bại.
   * Hiển thị thông báo kết quả cho người dùng:
     + Nếu thành công, thông báo "Đăng ký thành công".
     + Nếu thất bại, thông báo lỗi (ví dụ: "Email đã tồn tại").

HTTP Method:

* HTTP Method: POST

Backend:

* Backend: User Service
  + User Service là thành phần xử lý logic nghiệp vụ liên quan đến đăng ký user, bao gồm:
    - Kiểm tra tính hợp lệ của thông tin đăng ký (ví dụ: email có đúng định dạng không, các trường bắt buộc có được điền đầy đủ không).
    - Kiểm tra xem email đã tồn tại trong hệ thống chưa bằng cách truy vấn cơ sở dữ liệu.
    - Mã hóa mật khẩu (nếu cần) và lưu thông tin user vào cơ sở dữ liệu.
  + API Gateway chỉ đóng vai trò định tuyến, chuyển yêu cầu từ Frontend đến User Service.

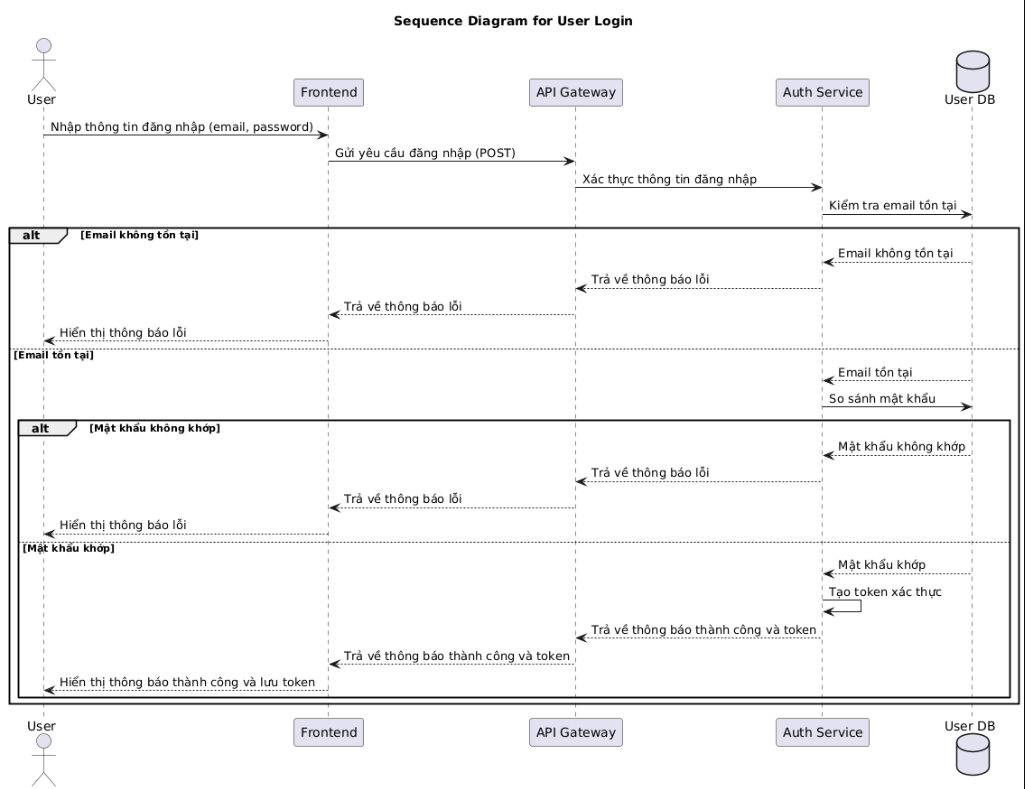
Luồng xử lý chi tiết:

1. Người dùng nhập thông tin đăng ký:
   * Người dùng điền thông tin vào form đăng ký và nhấn nút "Đăng ký".
2. Frontend gửi yêu cầu đến API Gateway:
   * Frontend thu thập dữ liệu từ form và gửi yêu cầu HTTP POST đến API Gateway.
3. API Gateway chuyển yêu cầu đến User Service:
   * API Gateway nhận yêu cầu và chuyển tiếp đến User Service để xử lý.
4. User Service kiểm tra tính hợp lệ:
   * User Service kiểm tra xem email đã tồn tại trong cơ sở dữ liệu chưa.
   * Nếu email đã tồn tại, trả về thông báo lỗi "Email đã tồn tại".
   * Nếu email hợp lệ, tiếp tục xử lý.
5. Lưu thông tin user vào cơ sở dữ liệu:
   * User Service mã hóa mật khẩu và lưu thông tin user vào cơ sở dữ liệu.
   * Trả về thông báo thành công nếu quá trình lưu thành công.
6. Trả về kết quả cho người dùng:
   * User Service trả về kết quả cho API Gateway.
   * API Gateway chuyển tiếp kết quả đến Frontend.
   * Frontend hiển thị thông báo kết quả cho người dùng.

Kết quả:

* Nếu thành công, người dùng nhận được thông báo "Đăng ký thành công" và có thể đăng nhập bằng tài khoản vừa tạo.
* Nếu thất bại, người dùng nhận được thông báo lỗi (ví dụ: "Email đã tồn tại") và cần nhập lại thông tin.

### 4.3.5.2 Đăng nhập

****

Tên Method: LoginUser

Mục đích:

* Cho phép người dùng đăng nhập vào hệ thống bằng cách xác thực thông tin đăng nhập (email và mật khẩu).

Input:

* Dữ liệu đăng nhập từ người dùng, bao gồm:
  + Email: Địa chỉ email của người dùng.
  + Mật khẩu: Mật khẩu do người dùng nhập.

Output:

* Kết quả trả về từ hệ thống, bao gồm:
  + Trường hợp thành công:
    - Trạng thái: "success".
    - Thông báo: "Đăng nhập thành công".
    - Dữ liệu: Thông tin người dùng (ví dụ: userId, email, token xác thực).
  + Trường hợp thất bại:
    - Trạng thái: "error".
    - Thông báo: Lý do thất bại (ví dụ: "Email hoặc mật khẩu không đúng").

Process:

1. Frontend nhận dữ liệu đăng nhập từ người dùng:
   * Người dùng điền thông tin đăng nhập (email và mật khẩu) qua form hoặc giao diện nhập liệu.
2. Gửi yêu cầu HTTP (POST) đến API Gateway:
   * Frontend gửi yêu cầu HTTP POST đến API Gateway với dữ liệu đăng nhập đã nhập.
3. Nhận phản hồi từ API Gateway:
   * Frontend nhận phản hồi từ API Gateway, bao gồm kết quả thành công hoặc thất bại.
   * Hiển thị thông báo kết quả cho người dùng:
     + Nếu thành công, thông báo "Đăng nhập thành công" và chuyển hướng người dùng đến trang chính.
     + Nếu thất bại, thông báo lỗi (ví dụ: "Email hoặc mật khẩu không đúng").

HTTP Method:

* HTTP Method: POST
* Lý do: POST được sử dụng để gửi dữ liệu nhạy cảm (như mật khẩu) và yêu cầu xác thực từ server.

Backend:

* Backend: Auth Service
  + Auth Service là thành phần xử lý logic nghiệp vụ liên quan đến đăng nhập, bao gồm:
    - Kiểm tra xem email có tồn tại trong hệ thống không.
    - Xác thực mật khẩu (so sánh mật khẩu nhập vào với mật khẩu đã mã hóa trong cơ sở dữ liệu).
    - Tạo token xác thực (ví dụ: JWT) nếu đăng nhập thành công.
  + API Gateway chỉ đóng vai trò định tuyến, chuyển yêu cầu từ Frontend đến Auth Service.

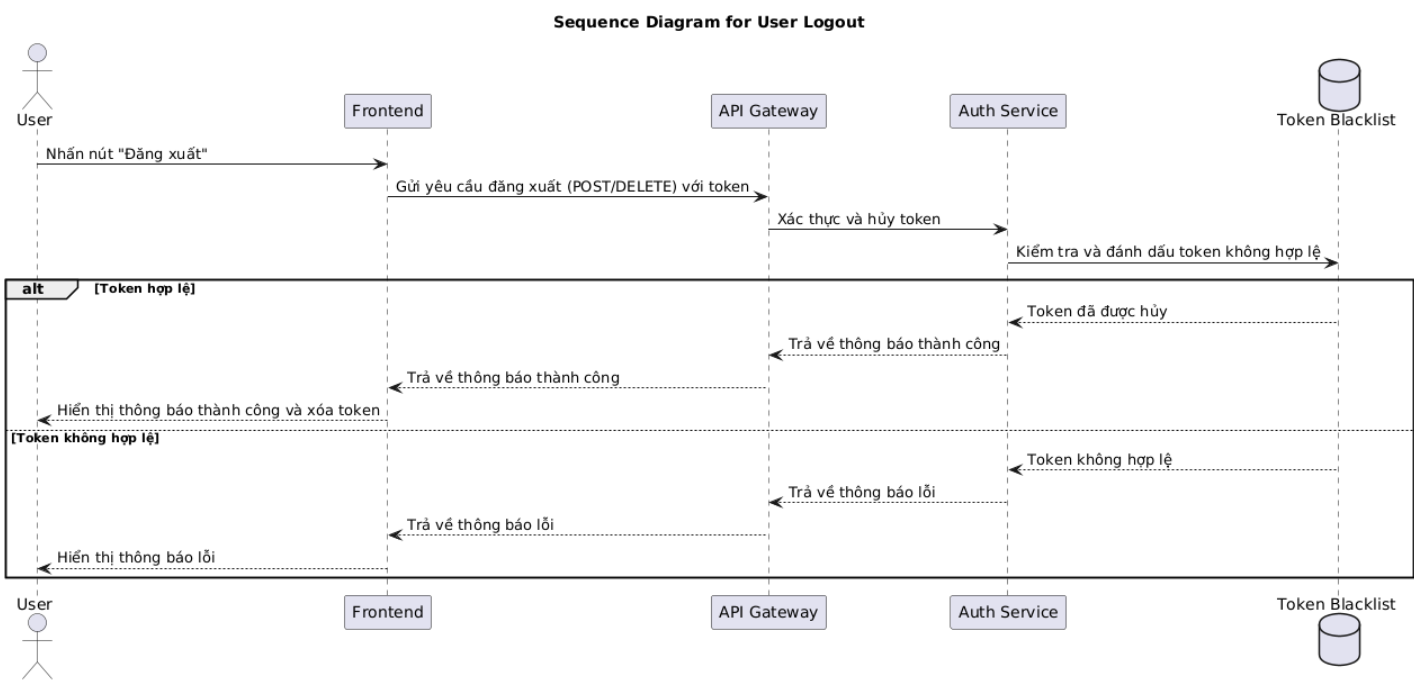
Luồng xử lý chi tiết:

1. Người dùng nhập thông tin đăng nhập:
   * Người dùng điền email và mật khẩu vào form đăng nhập và nhấn nút "Đăng nhập".
2. Frontend gửi yêu cầu đến API Gateway:
   * Frontend thu thập dữ liệu từ form và gửi yêu cầu HTTP POST đến API Gateway.
3. API Gateway chuyển yêu cầu đến Auth Service:
   * API Gateway nhận yêu cầu và chuyển tiếp đến Auth Service để xử lý.
4. Auth Service kiểm tra thông tin đăng nhập:
   * Auth Service kiểm tra xem email có tồn tại trong cơ sở dữ liệu không.
   * Nếu email không tồn tại, trả về thông báo lỗi "Email hoặc mật khẩu không đúng".
   * Nếu email tồn tại, so sánh mật khẩu nhập vào với mật khẩu đã mã hóa trong cơ sở dữ liệu.
     + Nếu mật khẩu không khớp, trả về thông báo lỗi "Email hoặc mật khẩu không đúng".
     + Nếu mật khẩu khớp, tiếp tục xử lý.
5. Tạo token xác thực:
   * Auth Service tạo token xác thực (ví dụ: JWT) để xác thực người dùng trong các yêu cầu tiếp theo.
   * Trả về thông báo thành công cùng với token và thông tin người dùng.
6. Trả về kết quả cho người dùng:
   * Auth Service trả về kết quả cho API Gateway.
   * API Gateway chuyển tiếp kết quả đến Frontend.
   * Frontend hiển thị thông báo kết quả cho người dùng:
     + Nếu thành công, lưu token vào localStorage hoặc cookie và chuyển hướng người dùng đến trang chính.
     + Nếu thất bại, hiển thị thông báo lỗi.

**Kết quả:**

* Nếu thành công, người dùng nhận được thông báo "Đăng nhập thành công" và được chuyển hướng đến trang chính.
* Nếu thất bại, người dùng nhận được thông báo lỗi (ví dụ: "Email hoặc mật khẩu không đúng") và cần nhập lại thông tin.

### 4.3.5.3 Đăng xuất

****

Tên Method: LogoutUser()

Mục đích:

* Cho phép người dùng đăng xuất khỏi hệ thống, hủy bỏ phiên làm việc hiện tại và xóa token xác thực.

Input:

* Token xác thực (JWT hoặc session token) của người dùng hiện tại.

Output:

* Kết quả trả về từ hệ thống, bao gồm:
  + Trường hợp thành công:
    - Trạng thái: "success".
    - Thông báo: "Đăng xuất thành công".
  + Trường hợp thất bại:
    - Trạng thái: "error".
    - Thông báo: Lý do thất bại (ví dụ: "Token không hợp lệ").

Process:

1. Frontend gửi yêu cầu đăng xuất:
   * Khi người dùng nhấn nút "Đăng xuất", Frontend gửi yêu cầu HTTP (POST hoặc DELETE) đến API Gateway kèm theo token xác thực.
2. API Gateway chuyển yêu cầu đến Auth Service:
   * API Gateway nhận yêu cầu và chuyển tiếp đến Auth Service để xử lý.
3. Auth Service xác thực và hủy token:
   * Auth Service kiểm tra tính hợp lệ của token.
   * Nếu token hợp lệ, hủy token (đánh dấu token không còn hiệu lực).
   * Nếu token không hợp lệ, trả về thông báo lỗi.
4. Trả về kết quả cho người dùng:
   * Auth Service trả về kết quả cho API Gateway.
   * API Gateway chuyển tiếp kết quả đến Frontend.
   * Frontend hiển thị thông báo kết quả cho người dùng:
     + Nếu thành công, xóa token khỏi localStorage hoặc cookie và chuyển hướng người dùng đến trang đăng nhập.
     + Nếu thất bại, hiển thị thông báo lỗi.

HTTP Method:

* HTTP Method: POST hoặc DELETE
* Lý do: POST hoặc DELETE được sử dụng để hủy phiên làm việc hoặc token xác thực.

Backend:

* Backend: Auth Service
  + Auth Service là thành phần xử lý logic nghiệp vụ liên quan đến đăng xuất, bao gồm:
    - Kiểm tra tính hợp lệ của token.
    - Hủy token (đánh dấu token không còn hiệu lực).
  + API Gateway chỉ đóng vai trò định tuyến, chuyển yêu cầu từ Frontend đến Auth Service.

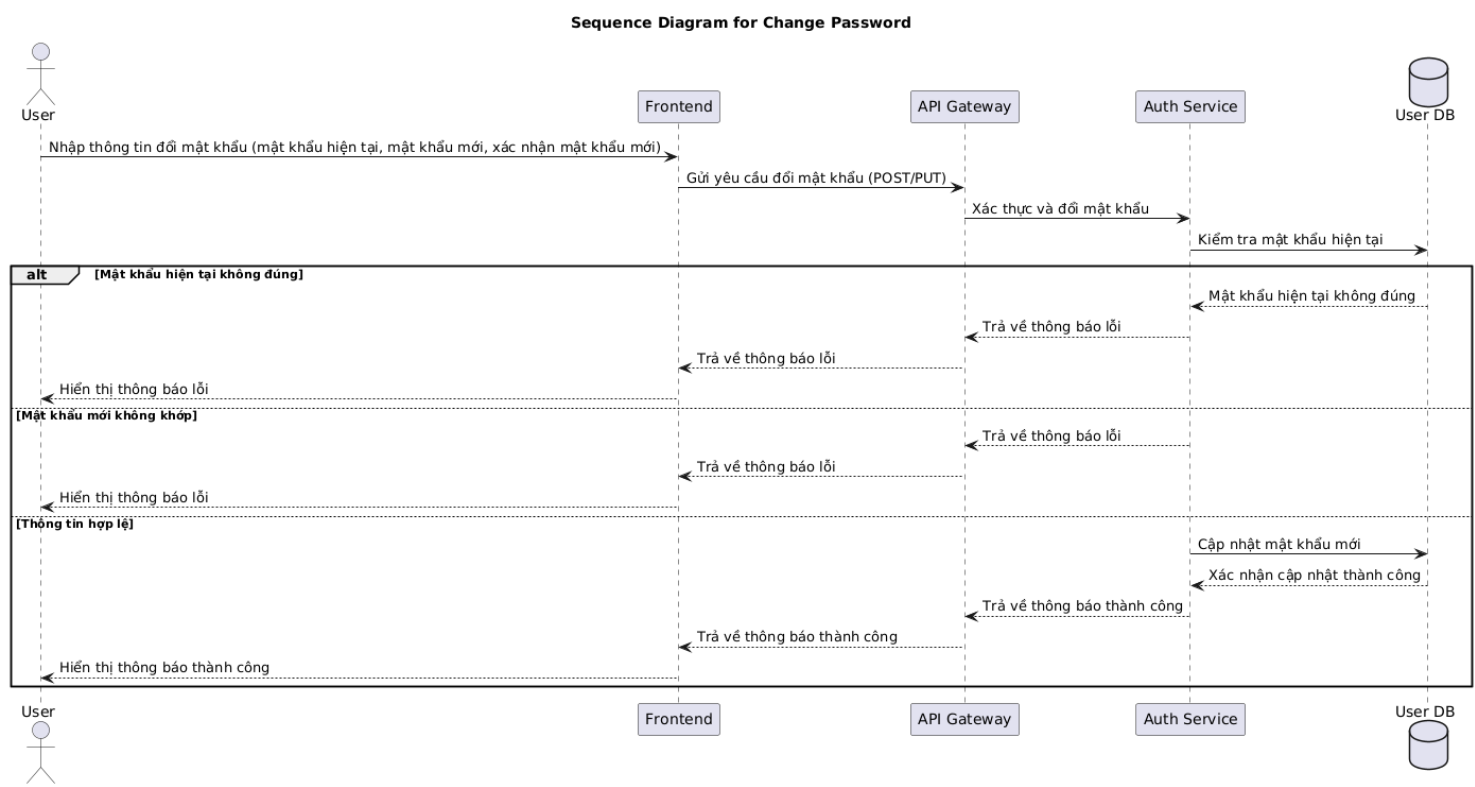
Luồng xử lý chi tiết:

1. Người dùng nhấn nút "Đăng xuất":
   * Người dùng nhấn nút "Đăng xuất" trên giao diện.
2. Frontend gửi yêu cầu đến API Gateway:
   * Frontend gửi yêu cầu HTTP (POST hoặc DELETE) đến API Gateway kèm theo token xác thực.
3. API Gateway chuyển yêu cầu đến Auth Service:
   * API Gateway nhận yêu cầu và chuyển tiếp đến Auth Service để xử lý.
4. Auth Service kiểm tra và hủy token:
   * Auth Service kiểm tra tính hợp lệ của token.
   * Nếu token hợp lệ, hủy token (đánh dấu token không còn hiệu lực).
   * Nếu token không hợp lệ, trả về thông báo lỗi.
5. Trả về kết quả cho người dùng:
   * Auth Service trả về kết quả cho API Gateway.
   * API Gateway chuyển tiếp kết quả đến Frontend.
   * Frontend hiển thị thông báo kết quả cho người dùng:
     + Nếu thành công, xóa token khỏi localStorage hoặc cookie và chuyển hướng người dùng đến trang đăng nhập.
     + Nếu thất bại, hiển thị thông báo lỗi.

Kết quả:

* Nếu thành công, người dùng nhận được thông báo "Đăng xuất thành công" và được chuyển hướng đến trang đăng nhập.
* Nếu thất bại, người dùng nhận được thông báo lỗi (ví dụ: "Token không hợp lệ").

### 4.3.5.4 Đổi mật khẩu

****

Tên Method: ChangePassword

Mục đích:

* Cho phép người dùng thay đổi mật khẩu hiện tại của họ.

Input:

* Dữ liệu đổi mật khẩu từ người dùng, bao gồm:
  + Mật khẩu hiện tại: Mật khẩu hiện tại của người dùng.
  + Mật khẩu mới: Mật khẩu mới mà người dùng muốn đổi.
  + Xác nhận mật khẩu mới: Nhập lại mật khẩu mới để xác nhận.

Output:

* Kết quả trả về từ hệ thống, bao gồm:
  + Trường hợp thành công:
    - Trạng thái: "success".
    - Thông báo: "Đổi mật khẩu thành công".
  + Trường hợp thất bại:
    - Trạng thái: "error".
    - Thông báo: Lý do thất bại (ví dụ: "Mật khẩu hiện tại không đúng", "Mật khẩu mới không khớp").

Process:

1. Frontend nhận dữ liệu đổi mật khẩu từ người dùng:
   * Người dùng điền thông tin đổi mật khẩu (mật khẩu hiện tại, mật khẩu mới, xác nhận mật khẩu mới) qua form hoặc giao diện nhập liệu.
2. Gửi yêu cầu HTTP (POST hoặc PUT) đến API Gateway:
   * Frontend gửi yêu cầu HTTP (POST hoặc PUT) đến API Gateway với dữ liệu đổi mật khẩu đã nhập.
3. API Gateway chuyển yêu cầu đến Auth Service:
   * API Gateway nhận yêu cầu và chuyển tiếp đến Auth Service để xử lý.
4. Auth Service xác thực và đổi mật khẩu:
   * Auth Service kiểm tra tính hợp lệ của mật khẩu hiện tại:
     + Nếu mật khẩu hiện tại không đúng, trả về thông báo lỗi "Mật khẩu hiện tại không đúng".
     + Nếu mật khẩu mới và xác nhận mật khẩu mới không khớp, trả về thông báo lỗi "Mật khẩu mới không khớp".
     + Nếu thông tin hợp lệ, tiến hành đổi mật khẩu.
5. Cập nhật mật khẩu mới vào cơ sở dữ liệu:
   * Auth Service mã hóa mật khẩu mới và cập nhật vào cơ sở dữ liệu.
   * Trả về thông báo thành công nếu quá trình cập nhật thành công.
6. Trả về kết quả cho người dùng:
   * Auth Service trả về kết quả cho API Gateway.
   * API Gateway chuyển tiếp kết quả đến Frontend.
   * Frontend hiển thị thông báo kết quả cho người dùng:
     + Nếu thành công, thông báo "Đổi mật khẩu thành công".
     + Nếu thất bại, hiển thị thông báo lỗi.

HTTP Method:

* HTTP Method: POST hoặc PUT
* Lý do: POST hoặc PUT được sử dụng để cập nhật thông tin (trong trường hợp này là mật khẩu).

Backend:

* Backend: Auth Service
  + Auth Service là thành phần xử lý logic nghiệp vụ liên quan đến đổi mật khẩu, bao gồm:
    - Kiểm tra tính hợp lệ của mật khẩu hiện tại.
    - Kiểm tra sự khớp nhau giữa mật khẩu mới và xác nhận mật khẩu mới.
    - Mã hóa mật khẩu mới và cập nhật vào cơ sở dữ liệu.
  + API Gateway chỉ đóng vai trò định tuyến, chuyển yêu cầu từ Frontend đến Auth Service.

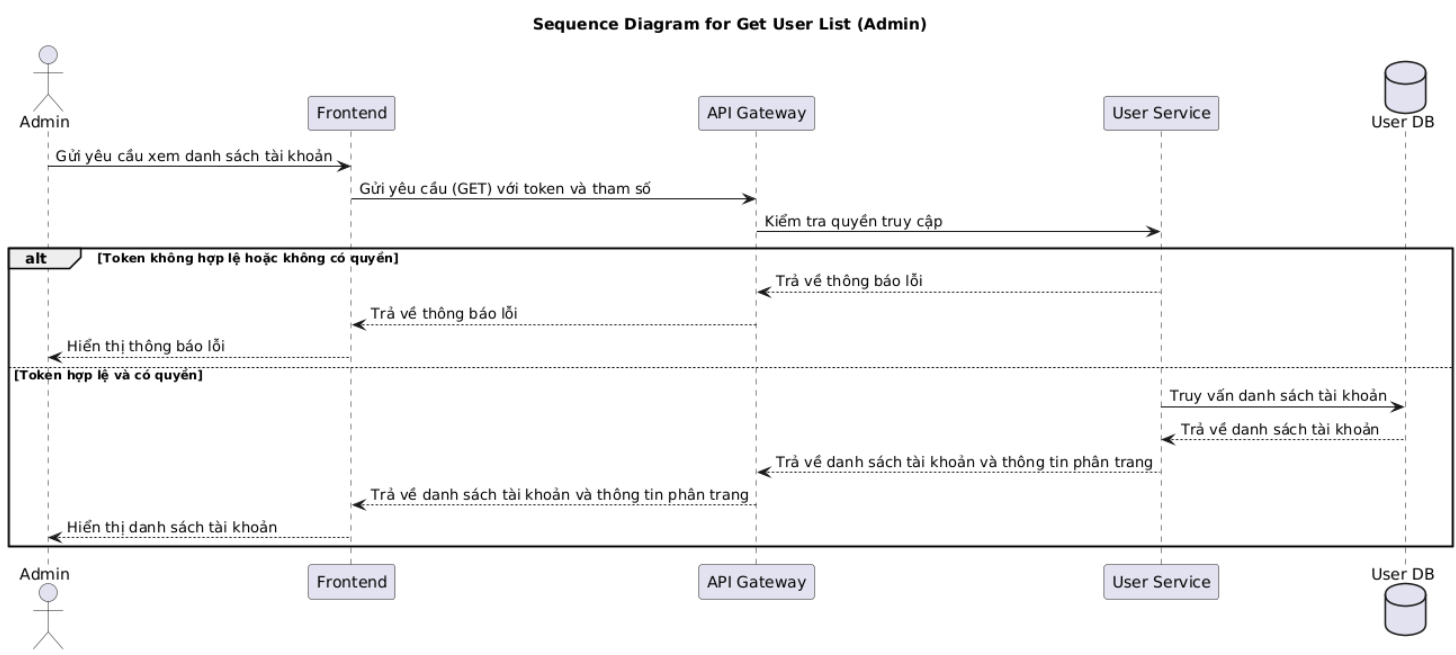
Luồng xử lý chi tiết:

1. Người dùng nhập thông tin đổi mật khẩu:
   * Người dùng điền mật khẩu hiện tại, mật khẩu mới và xác nhận mật khẩu mới vào form đổi mật khẩu.
2. Frontend gửi yêu cầu đến API Gateway:
   * Frontend gửi yêu cầu HTTP (POST hoặc PUT) đến API Gateway với dữ liệu đổi mật khẩu.
3. API Gateway chuyển yêu cầu đến Auth Service:
   * API Gateway nhận yêu cầu và chuyển tiếp đến Auth Service để xử lý.
4. Auth Service kiểm tra và đổi mật khẩu:
   * Auth Service kiểm tra tính hợp lệ của mật khẩu hiện tại và sự khớp nhau giữa mật khẩu mới và xác nhận mật khẩu mới.
   * Nếu thông tin hợp lệ, tiến hành đổi mật khẩu và cập nhật vào cơ sở dữ liệu.
5. Trả về kết quả cho người dùng:
   * Auth Service trả về kết quả cho API Gateway.
   * API Gateway chuyển tiếp kết quả đến Frontend.
   * Frontend hiển thị thông báo kết quả cho người dùng:
     + Nếu thành công, thông báo "Đổi mật khẩu thành công".
     + Nếu thất bại, hiển thị thông báo lỗi.

**Kết quả:**

* Nếu thành công, người dùng nhận được thông báo "Đổi mật khẩu thành công" và có thể đăng nhập bằng mật khẩu mới.
* Nếu thất bại, người dùng nhận được thông báo lỗi (ví dụ: "Mật khẩu hiện tại không đúng" hoặc "Mật khẩu mới không khớp") và cần nhập lại thông tin.

### 4.3.5.5 Xem danh sách tài khoản (admin)

****

Tên Method: GetUserList

Mục đích:

* Cho phép admin xem danh sách tất cả tài khoản người dùng trong hệ thống.

Input:

* Token xác thực của admin (JWT hoặc session token) để xác nhận quyền truy cập.
* Các tham số tùy chọn (nếu có):
  + page: Trang hiện tại (phân trang).
  + limit: Số lượng tài khoản hiển thị trên mỗi trang.
  + search: Từ khóa tìm kiếm (ví dụ: email, tên người dùng).

Output:

* Kết quả trả về từ hệ thống, bao gồm:
  + Trường hợp thành công:
    - Trạng thái: "success".
    - Dữ liệu: Danh sách tài khoản người dùng (ví dụ: userId, email, tên, vai trò, v.v.).
    - Thông tin phân trang (nếu có): Tổng số trang, tổng số tài khoản.
  + Trường hợp thất bại:
    - Trạng thái: "error".
    - Thông báo: Lý do thất bại (ví dụ: "Không có quyền truy cập", "Token không hợp lệ").

Process:

1. Admin gửi yêu cầu xem danh sách tài khoản:
   * Admin truy cập giao diện quản lý tài khoản và gửi yêu cầu xem danh sách tài khoản.
2. Frontend gửi yêu cầu HTTP (GET) đến API Gateway:
   * Frontend gửi yêu cầu HTTP GET đến API Gateway kèm theo token xác thực của admin và các tham số tùy chọn (nếu có).
3. API Gateway chuyển yêu cầu đến User Service:
   * API Gateway nhận yêu cầu và chuyển tiếp đến User Service để xử lý.
4. User Service kiểm tra quyền truy cập:
   * User Service kiểm tra token xác thực của admin để xác nhận quyền truy cập.
   * Nếu token không hợp lệ hoặc admin không có quyền, trả về thông báo lỗi.
5. Lấy danh sách tài khoản từ cơ sở dữ liệu:
   * User Service truy vấn danh sách tài khoản từ User DB dựa trên các tham số (nếu có).
   * Trả về danh sách tài khoản và thông tin phân trang (nếu có).
6. Trả về kết quả cho admin:
   * User Service trả về kết quả cho API Gateway.
   * API Gateway chuyển tiếp kết quả đến Frontend.
   * Frontend hiển thị danh sách tài khoản và thông tin phân trang (nếu có) cho admin.

HTTP Method:

* HTTP Method: GET
* Lý do: GET được sử dụng để truy vấn dữ liệu mà không làm thay đổi trạng thái của hệ thống.

Backend:

* Backend: User Service
  + User Service là thành phần xử lý logic nghiệp vụ liên quan đến quản lý tài khoản, bao gồm:
    - Kiểm tra quyền truy cập của admin.
    - Truy vấn danh sách tài khoản từ cơ sở dữ liệu.
    - Xử lý phân trang và tìm kiếm (nếu có).
  + API Gateway chỉ đóng vai trò định tuyến, chuyển yêu cầu từ Frontend đến User Service.

Luồng xử lý chi tiết:

1. Admin gửi yêu cầu xem danh sách tài khoản:
   * Admin truy cập giao diện quản lý tài khoản và gửi yêu cầu xem danh sách tài khoản.
2. Frontend gửi yêu cầu đến API Gateway:
   * Frontend gửi yêu cầu HTTP GET đến API Gateway kèm theo token xác thực của admin và các tham số tùy chọn (nếu có).
3. API Gateway chuyển yêu cầu đến User Service:
   * API Gateway nhận yêu cầu và chuyển tiếp đến User Service để xử lý.
4. User Service kiểm tra quyền truy cập:
   * User Service kiểm tra token xác thực của admin để xác nhận quyền truy cập.
   * Nếu token không hợp lệ hoặc admin không có quyền, trả về thông báo lỗi.
5. Lấy danh sách tài khoản từ cơ sở dữ liệu:
   * User Service truy vấn danh sách tài khoản từ User DB dựa trên các tham số (nếu có).
   * Trả về danh sách tài khoản và thông tin phân trang (nếu có).
6. Trả về kết quả cho admin:
   * User Service trả về kết quả cho API Gateway.
   * API Gateway chuyển tiếp kết quả đến Frontend.
   * Frontend hiển thị danh sách tài khoản và thông tin phân trang (nếu có) cho admin.

**Kết quả:**

* Nếu thành công, admin nhận được danh sách tài khoản và thông tin phân trang (nếu có).
* Nếu thất bại, admin nhận được thông báo lỗi (ví dụ: "Không có quyền truy cập" hoặc "Token không hợp lệ").

## Requirement Traceability Matrix

## 

| **Mã yêu cầu** | **Tên yêu cầu** | **Test Case id** | **Mô tả test case** |
| --- | --- | --- | --- |
| RQ01 | Người dùng có thể đăng ký tài khoản | TC-01 | Kiểm tra đăng ký tài khoản với thông tin hợp lệ và không hợp lệ |
| RQ02 | Người dùng có thể đăng nhập vào hệ thống | TC-02 | Kiểm tra đăng nhập với tài khoản hợp lệ và không hợp lệ |
| RQ03 | Người dùng có thể đổi mật khẩu | TC-03 | Kiểm tra đổi mật khẩu với mật khẩu cũ đúng/sai, mật khẩu mới mạnh/yếu |
| RQ04 | Người dùng có thể quên mật khẩu và đặt lại | TC-04 | Kiểm tra quy trình đặt lại mật khẩu qua email hoặc số điện thoại |
| RQ05 | Người dùng có thể cập nhật thông tin cá nhân | TC-05 | Kiểm tra cập nhật thông tin như tên, email, số điện thoại |
| RQ06 | Người dùng có thể tải nhạc lên | TC-06 | Kiểm tra tải nhạc lên với các định dạng hợp lệ và không hợp lệ |
| RQ07 | Người dùng có thể tạo danh sách phát nhạc cá nhân | TC-07 | Kiểm tra tạo danh sách phát, thêm/xóa bài hát |
| RQ08 | Người dùng có thể xem danh sách tài khoản (Admin) | TC-08 | Kiểm tra admin có thể xem danh sách tài khoản trong hệ thống |
| RQ09 | Người dùng có thể bình luận bài hát | TC-09 | Kiểm tra đăng, chỉnh sửa, xóa bình luận |
| RQ10 | Người dùng có thể tìm kiếm nhạc | TC-10 | Kiểm tra tìm kiếm nhạc theo tên, nghệ sĩ, thể loại |
| RQ11 | Người dùng có thể nghe nhạc | TC-11 | Kiểm tra phát nhạc từ danh sách |
| RQ12 | Người dùng có thể tải nhạc xuống | TC-12 | Kiểm tra tải nhạc xuống |
| RQ13 | Người dùng có thể quản lý nhạc của mình | TC-13 | Kiểm tra quản lý nhạc của tôi |
| RQ14 | Người dùng có thể chia sẻ danh sách phát nhạc | TC-14 | Kiểm tra chia sẻ danh sách phát nhạc |
| RQ15 | Người dùng có thể xem danh sách phát nhạc của mình | TC-15 | Kiểm tra xem danh sách phát nhạc của tôi |
| RQ16 | Người dùng có thể chỉnh sửa thông tin nhạc | TC-16 | Kiểm tra chỉnh sửa thông tin nhạc |
| RQ17 | Người dùng có thể xóa nhạc | TC-17 | Kiểm tra xóa nhạc |
| RQ18 | Đảm bảo tính chính xác, xác thực và đáng tin cậy của hệ thống | TC-18 | Kiểm tra tính chính xác và xác thực của hệ thống |
| NFR1 | Yêu cầu về độ tin cậy | TC-NFR-01 | Kiểm tra tính xác thực của thông tin hệ thống cung cấp và khả năng phục hồi 100% trong vòng 24h. |
| NFR2 | Yêu cầu về hiệu năng | TC-NFR-02 | Kiểm tra thời gian phản hồi của hệ thống trong tìm kiếm, tải trang, đăng tải nội dung và cập nhật tương tác theo yêu cầu hiệu năng. |
| NFR3 | Yêu cầu về tính bảo mật | TC-NFR-03 | Kiểm tra việc mã hóa dữ liệu quan trọng, kiểm soát quyền truy cập, xác thực người dùng và ghi log truy xuất hệ thống theo tiêu chuẩn bảo mật. |
| NFR4 | Yêu cầu về tính hữu dụng | TC-NFR-04 | Kiểm tra tính trực quan của giao diện, khả năng học hỏi của người dùng mới, khả năng truy cập đa nền tảng và mức độ hỗ trợ bảo vệ người dùng khỏi lỗi thao tác. |
| NFR5 | Yêu cầu về khả năng bảo trì | TC-NFR-05 | Kiểm tra tính mô-đun của hệ thống, khả năng tái sử dụng mã nguồn, khả năng phân tích tác động của thay đổi, mức độ dễ dàng trong việc sửa đổi và thực hiện kiểm thử hệ thống. |
| NFR6 | Yêu cầu về tính tương thích | TC-NFR-06 | Kiểm tra khả năng hoạt động ổn định trên nhiều nền tảng (Windows, macOS, Android, iOS), trình duyệt khác nhau và khả năng tương tác với các hệ thống bên ngoài qua API hoặc dịch vụ tích hợp. |
| NFR7 | Yêu cầu về tính văn hóa | TC-NFR-07 | Kiểm tra khả năng hỗ trợ ngôn ngữ Tiếng Việt là mặc định, đảm bảo hệ thống có thể mở rộng hỗ trợ đa ngôn ngữ, tùy chỉnh định dạng ngày giờ, biểu tượng, màu sắc và các yếu tố giao diện phù hợp với đặc thù văn hóa của từng khu vực. |

## 

# CHƯƠNG 5: SẢN PHẨM PROTOTYPE

* Lựa chọn **01 nghiệp vụ quan trọng nhất** để triển khai lập trình thành phần mềm
* Với mỗi sản phẩm triển khai cần chứng minh sự tuân thủ thiết kế tương ứng và đo lường các NFR/Quality Attributes để đánh giá hiệu quả
* Ghi rõ tên người thực hiện từng sản phẩm thiết kế, lập trình cụ thể

## 

## KẾT LUẬN

(kết luận theo mục tiêu đã đề ra của đề tài)

## TÀI LIỆU THAM KHẢO

## PHỤ LỤC

Phụ lục A : SRS của dự án Phát triển phần mềm mạng xã hội chia sẻ âm nhạc SoundShare cho Sound Entertainment

# Giới thiệu

## 1.1 Mục đích tài liệu

* Mục đích của tài liệu đặc tả yêu cầu phần mềm này là cung cấp một cái nhìn tổng quan, dễ hiểu về các yêu cầu, thành phần của dự án.
* Tài liệu giúp định nghĩa, phân tích và liệt kê các yêu cầu chức năng và phi chức năng của phần mềm, bao gồm các tính năng, giao diện, dữ liệu, bảo mật, hiệu suất, tương tác với hệ thống khác và yêu cầu kỹ thuật khác cần thiết để phát triển phần mềm quản lý bán hàng cho cửa hàng điện máy.
* Tài liệu này được cung cấp như một tài liệu tham khảo cho các thành viên trong đội ngũ phát triển phần mềm để hiểu rõ hơn về các yêu cầu cần thiết của dự án, đồng thời là một công cụ quản lý dự án để giám sát và đánh giá tiến độ, chất lượng của phần mềm trong quá trình phát triển.
* Tài liệu này được viết dựa theo chuẩn của Tài liệu đặc tả yêu cầu phần mềm (Software Requirements Specifications - SRS) được giải thích trong "IEEE Recommended Practice for Software Requirements Specifications" và " IEEE Guide for Developing System Requirements Specifications".

Với cấu trúc được chia làm ba phần:

* Phần 1: Cung cấp cái nhìn tổng quan về các thành phần của SRS.
* Phần 2: Mô tả tổng quan các nhân tố, ràng buộc, đặc điểm người dùng, môi trường thực thi tác động lên hệ thống và các yêu cầu của nó. Cung cấp thông tin chi tiết các yêu cầu chức năng, cung cấp cho các nhà phát triển phần mềm thông tin để phát triển phần mềm đáp ứng được các yêu cầu đó.
* Phần 3: Các yêu cầu phi chức năng.

## 1.2 Phạm vi sử dụng

* Tài liệu đặc tả yêu cầu phần mềm này được xây dựng nhằm phục vụ cho dự án Phát triển phần mềm mạng xã hội chia sẻ âm nhạc SoundShare  
  cho Sound Entertainment theo Công nghệ phần mềm.
* Trong dự án phát triển phần mềm mạng xã hội chia sẻ âm nhạc SoundShare  
  cho Sound Entertainment theo Công nghệ phần mềm thì những đối tượng có thể truy cập và sử dụng tài liệu SRS bao gồm:
* Nhóm phát triển dự án
* Chủ sở hữu dự án

## 1.3 Mục đích sử dụng

* Với việc Phát triển phần mềm mạng xã hội chia sẻ âm nhạc SoundShare cho Sound Entertainment theo Công nghệ phần mềm sẽ cung cấp cho người dùng một công cụ mạng xã hội giúp chia sẻ âm nhạc với nhau cũng như lưu trữ lại các bài hát, ca sĩ yêu thích đồng thời bày tỏ ý kiến cá nhân với các sản phẩm âm nhạc mang nhiều lợi ích cho việc giải trí của mọi người.

## 1.4 Định nghĩa và viết tắt

| STT | Thuật ngữ | Chi tiết |
| --- | --- | --- |
| 1 | Software Requirements Specifications-  SRS | Đặc tả yêu cầu phần mềm |
| 2 | Use Case(s) | Biểu đồ mô tả những yêu cầu của hệ thống |
| 3 | MVC | Model - View - Controller |

* Design Convention

| STT | Type | Convention |
| --- | --- | --- |
| 1 | Label | lb |
| 2 | Textbox | tb |
| 3 | Button | btn |
| 4 | Checkbox | ck |
| 5 | Radio button | rb |
| 6 | ComboBox | cb |
| 7 | ListBox | lb |
| 8 | DataGridView | dvg |
| 9 | Panel | panel |

* Database Convention

| STT | Type | Quy tắc đặt tên |
| --- | --- | --- |
| 1 | Database | db\_ |
| 2 | Table | tbl |
| 3 | Int | i |
| 4 | Float | f |
| 5 | Nvarchar | s |
| 6 | Varchar | s |
| 7 | Nchar | s |
| 8 | Char | s |
| 9 | Text | t |
| 10 | Date | d |
| 11 | Datetime | d |
| 12 | bit | b |
| 13 | decimal | m |
| 14 | Primary Key | PK\_ |
| 15 | Foreign Key | FK\_ |
| 16 | Store Procedure | sp\_ |
| 17 | View | vw\_ |

# Nội dung

## 2.1 Nhu cầu của người dùng

* Chia sẻ các bài nhạc của cá nhân với mọi người
* Nghe nhạc từ nhiều người dùng
* Nghe nhạc ngay cả khi tắt màn hình
* Tải nhạc về máy để nghe, cài đặt làm nhạc chuông…
* Thoải mái bày tỏ cảm xúc với các bài nhạc của cá nhân và của mọi người
* Thiết lập danh sách phát để lưu trữ những bài nhạc yêu thích

## 2.2 Tổng quan hệ thống

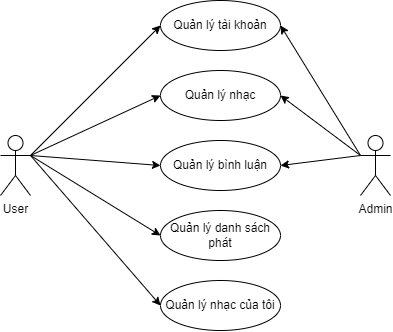
### 2.2.1 Mục tiêu và phạm vi của hệ thống

Phát triển phần mềm mạng xã hội chia sẻ âm nhạc SoundShare cho Entertainment theo Công nghệ phần mềm giúp phục vụ nhu cầu giải trí của người dùng

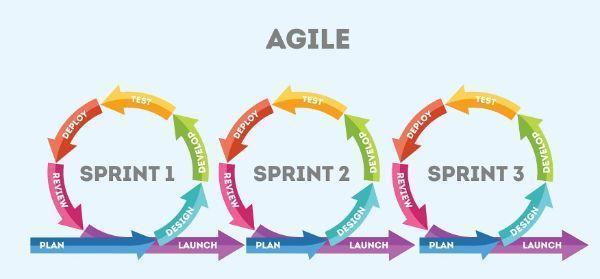
Các chức năng xây dựng bao gồm như sau:

* Quản lý tài khoản
* Quản lý nhạc
* Quản lý bình luận
* Quản lý danh sách phát
* Quản lý nhạc của tôi

### 2.2.2 Mô hình hệ thống

****

### 2.2.3 Mô hình phần mềm



Lý do chọn mô hình Agile cho dự án

* Tính linh hoạt: Agile cho phép thay đổi và điều chỉnh yêu cầu dự án trong quá trình phát triển. Điều này cho phép tiếp thu phản hồi từ khách hàng và thích ứng với thay đổi nhanh chóng trong môi trường kinh doanh.
* Tập trung vào giá trị: Agile tập trung vào việc cung cấp giá trị sớm và thường xuyên. Dự án được chia thành các phần nhỏ hơn gọi là "sprints" hoặc "iterations", trong đó thành phẩm có thể được phân phối cho khách hàng nhanh chóng.
* Kiểm soát chất lượng: Agile đặt sự chú trọng vào kiểm soát chất lượng liên tục thông qua việc thực hiện kiểm thử và phản hồi ngay từ giai đoạn sớm của dự án. Việc này giúp giảm thiểu rủi ro và đảm bảo chất lượng sản phẩm cuối cùng.

# Các yêu cầu chức năng

## 3.1 Mô hình hóa chức năng nghiệp vụ

### 3.1.1 Xác định các chức năng chi tiết

(01) Đăng ký

(02) Đăng nhập

(03) Đăng xuất

(04) Đổi mật khẩu

(05) Xem danh sách tài khoản

(06) Nghe nhạc

(07) Tải nhạc xuống

(08) Tìm kiếm nhạc

(09) Xem danh sách các bài nhạc

(10) Thêm bình luận

(11) Sửa bình luận

(12) Xóa bình luận

(13) Xem tất cả các bình luận

(14) Tạo danh sách phát nhạc

(15) Chia sẻ danh sách phát nhạc

(16) Tải nhạc lên

(17) Chỉnh sửa thông tin nhạc

(18) Xóa nhạc

### 3.1.2 Gom nhóm chức năng

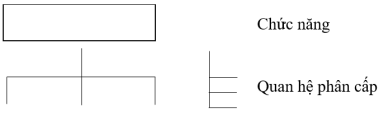
### 

| **STT** | **Chức năng mức 2** | **Chức năng mức 1** | **Chức năng mức 0** |
| --- | --- | --- | --- |
| **Tên chức năng** |
| 1 | (01) Đăng ký | Quản lý tài khoản | Phát triển phần mềm mạng xã hội chia sẻ âm nhạc SoundShare cho Sound Entertainment theo Công nghệ phần mềm |
| 2 | (02) Đăng nhập |
| 3 | (03) Đăng xuất |
| 4 | (4) Đổi mật khẩu |
| 5 | (5) Xem danh sách tài khoản |
| 6 | (6) Nghe nhạc | Quản lý nhạc |
| 7 | (7) Tải nhạc xuống |
| 8 | (8) Tìm kiếm nhạc |
| 9 | (9) Xem danh sách các bài nhạc |
| 10 | (10) Thêm bình luận | Quản lý bình luận |
| 11 | (11) Sửa bình luận |
| 12 | (12) Xóa bình luận |
| 13 | (13) Xem tất cả các bình luận |
| 14 | (14) Tạo danh sách phát nhạc | Quản lý danh sách phát |
| 15 | (15) Chia sẻ danh sách phát nhạc |
| 16 | (16) Tải nhạc lên | Quản lý nhạc của tôi |
| 17 | (17) Chỉnh sửa thông tin nhạc |
| 18 | (18) Xóa nhạc |

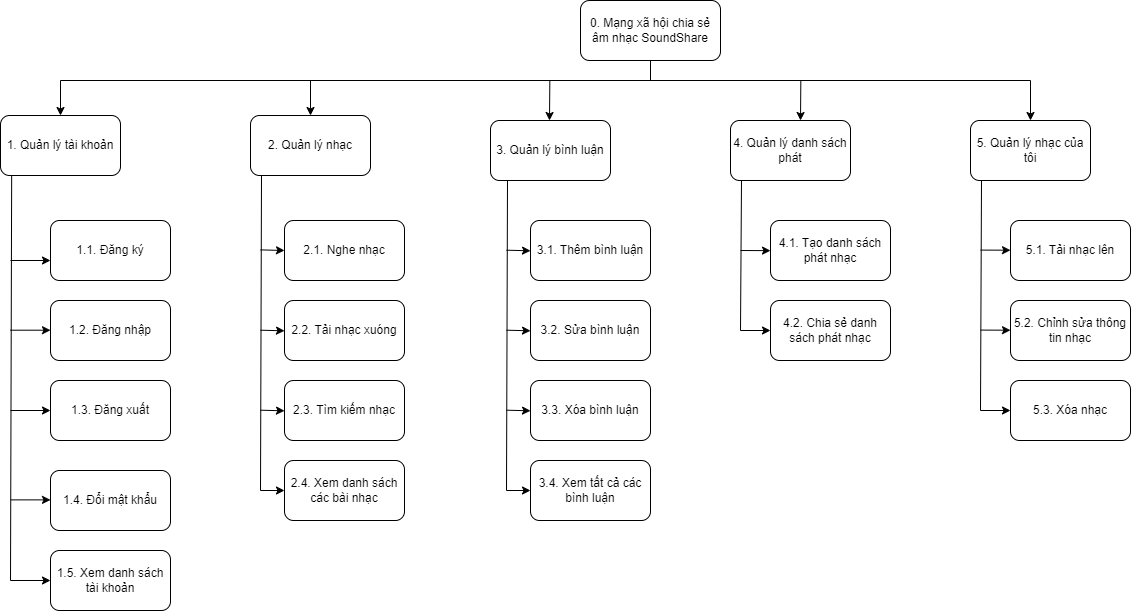
### 

### 3.1.3 Sơ đồ phân rã chức năng

* Ký hiệu sử dụng

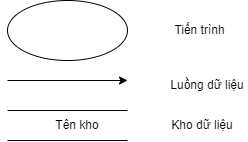


* Sơ đồ phân rã chức năng:

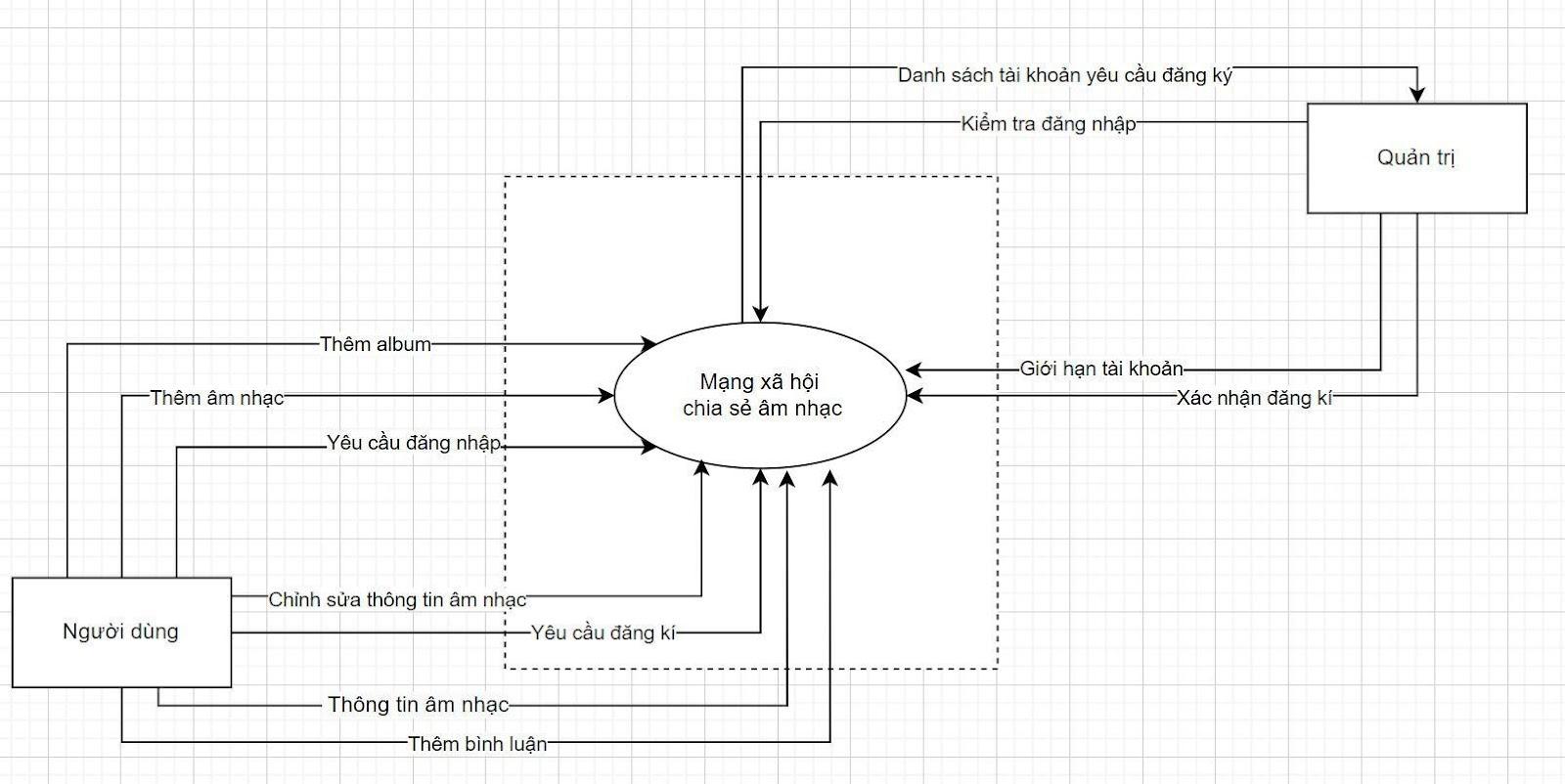
****

## 3.2 Mô hình hóa tiến trình nghiệp vụ

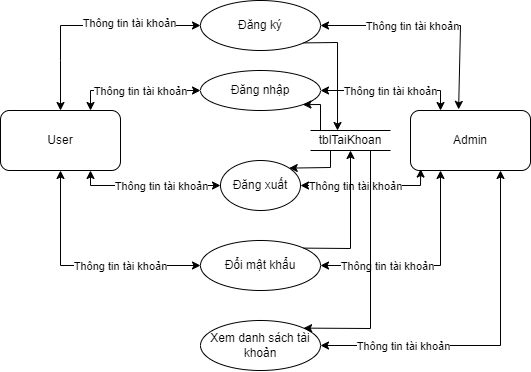
Ký hiệu sử dụng



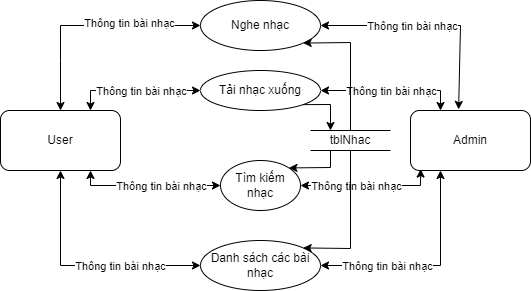
### Sơ đồ luồng dữ liệu



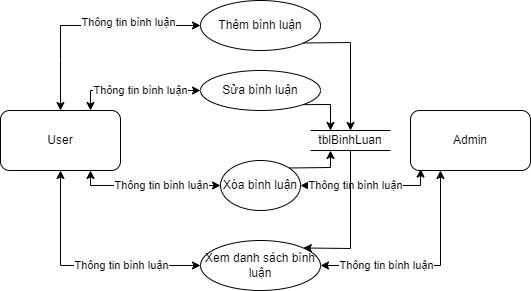
### 3.2.1 DFD Quản lý tài khoản



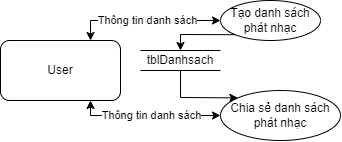
### 3.2.2 DFD Quản lý nhạc



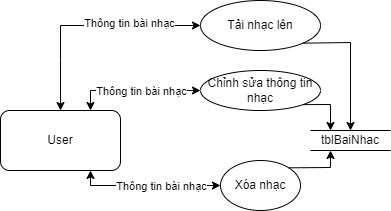
### 3.2.3 DFD Quản lý bình luận



### 3.2.4 DFD Quản lý danh sách phát



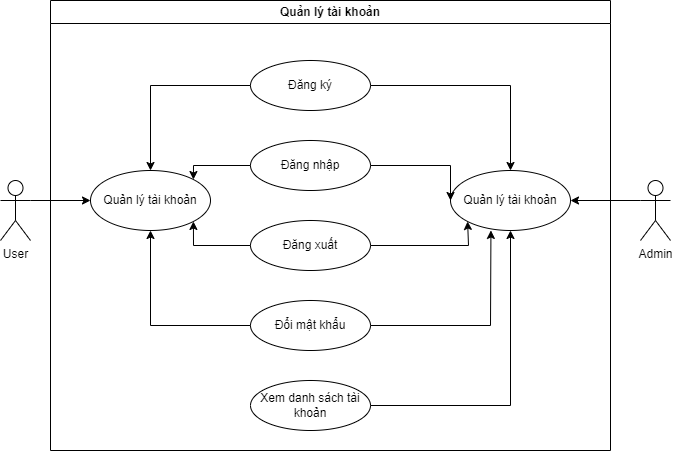
### 3.2.5 DFD Quản lý nhạc của tôi



## 3.3 Đặc tả usecase

Để có thể hình dung rõ hơn về từng yêu cầu chức năng của hệ thống bằng cách mô hình hóa chúng dưới các sơ đồ đặc tả use cases, các sơ đồ sẽ được trình bày phía dưới

### 3.3.1 Quản lý tài khoản

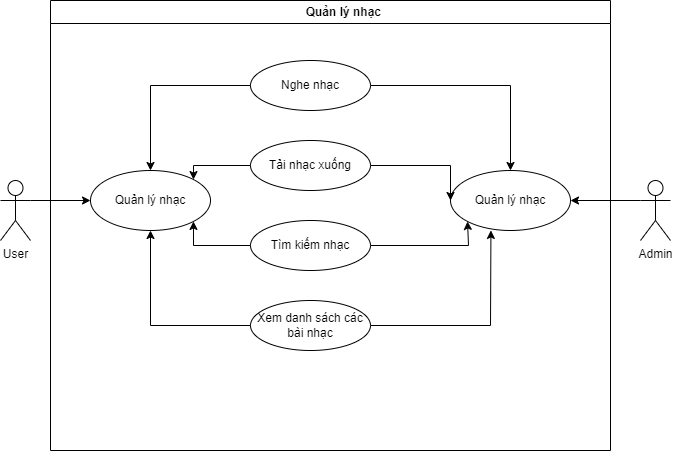


| **Mã use case** | UC01\_QLTK | | |
| --- | --- | --- | --- |
| **Tên use case** | Quản lý tài khoản | | |
| **Actor** | Quản lý, Người dùng | | |
| **Mô tả** | Use case cho phép  - Quản lý xem danh sách tài khoản, đăng nhập, đăng ký, đăng xuất và đổi mật khẩu  - Người dùng đăng nhập, đăng ký, đăng xuất, đổi mật khẩu | | |
| **Luồng sự kiện chính** | | | |
|  | ***Hành động*** | | ***Phản ứng hệ thống*** |
| ***Đăng nhập*** | 1. Tại trang chủ, chọn mục Đăng nhập | | 2. Hiển thị giao diện đăng nhập |
| 3. Nhập thông tin đăng nhập | |  |
| 4. Ấn nút đăng nhập | | 5. Kiểm tra hợp lệ của dữ liệu |
|  | | 6. Đưa ra thông báo đăng nhập thành công/ thất bại |
| ***Đăng ký*** | 1. Tại giao diện đăng nhập, chọn Đăng ký | | 2. Hiển thị giao diện đăng ký |
| 3. Nhập thông tin đăng ký | |  |
| 4. Ấn nút Đăng ký | | 5. Kiểm tra tính hợp lệ của dữ liệu |
|  | | 6. Đưa ra thông báo đăng ký thành công/ thất bại |
| ***Đổi mật khẩu*** | 1. Tại giao diện đăng nhập, chọn Đổi mật khẩu | | 2. Hiển thị giao diện đổi mật khẩu |
| 3. Nhập thông tin tài khoản | |  |
| 4. Ấn nút Đổi mật khẩu | | 5. Kiểm tra tính hợp lệ của dữ liệu |
|  | | 6. Đưa ra thông báo đổi mật khẩu thành công/ thất bại |
| ***Đăng xuất*** | 1. Tại giao diện bất kỳ, người dùng nhấn chọn vào avatar góc bên phải màn hình | | 2. Hệ thống hiển thị “Đăng xuất” |
| 3. Người dùng bấm chọn “Đăng xuất” | | 4. Hệ thống hiển thị thông báo “Bạn có chắc chắn muốn đăng xuất” |
| 5. Người dùng chọn “Đồng ý” | | 6. Hệ thống thoát tài khoản khỏi màn hình và trở về trang chủ |
| ***Xem danh sách tài khoản*** | 1. Nhấn vào danh mục Tài khoản | | 2. Hiện danh sách tài khoản |
| **Các điều kiện cụ thể** | | | |
| ***Điều kiện trước*** | | Truy cập vào hệ thống | |
| ***Điều kiện sau*** | | * Tài khoản khi thêm mới/ đăng ký được lưu trữ trong cơ sở dữ liệu của hệ thống. * Tài khoản khi đổi mật khẩu cần được cập nhật trong cơ sở dữ liệu * Thông báo lỗi khi thực hiện các chức năng không thành công | |
| ***Các điều kiện mở rộng*** | | Không có | |
|  |  |  |  |

* ·  **Các tình huống sử dụng**

| **STT** | **Đối tượng** | **Hoạt động** | **Thông tin đầu vào** | **Thông tin đầu ra** |
| --- | --- | --- | --- | --- |
| 1 | Quản lý, Người dùng | Đăng nhập | Thông tin đăng nhập của tài khoản | Đăng nhập thành công |
| 2 | Quản lý, Người dùng | Đăng ký | Thông tin đăng ký của tài khoản | Đăng ký thành công |
| 3 | Quản lý, Người dùng | Đổi mật khẩu | Thông tin tài khoản cần đổi mật khẩu | Đổi mật khẩu thành công |
| 4 | Quản lý, người dùng | Đăng xuất | Đã đăng nhập vào hệ thống | Đăng xuất tài khoản khỏi hệ thống thành công |
| 5 | Quản lý | Xem danh sách các tài khoản | Danh sách các tài khoản trong hệ thống | Xem được danh sách các tài khoản |

### 3.3.2 Quản lý nhạc

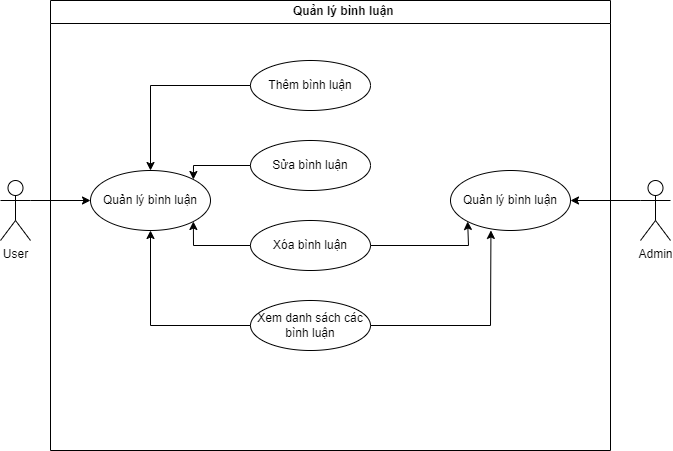
****

| **Mã use case** | | UC02\_QLN | |
| --- | --- | --- | --- |
| **Tên use case** | | Quản lý nhạc | |
| **Actor** | | Admin, user | |
| **Mô tả** | | Use case cho phép   * Admin và user: có thể nghe nhạc, tải nhạc xuống, tìm kiếm nhạc và xem danh sách các bài nhạc | |
| **Luồng sự kiện chính** | | | |
|  | ***Hành động*** | | ***Phản ứng hệ thống*** |
| **Nghe nhạc** | 1. Nhấn vào danh mục âm nhạc | | 2. Hệ thống hiển thị danh sách các bài nhạc |
| 3. Người dùng bấm chọn bài nhạc bất kỳ | | 4. Hệ thống thực hiện phát bài nhạc người dùng đã chọn |
| **Tải nhạc xuống** | 1. Tại giao diện trang nghe nhạc, người dùng bấm chọn icon tải xuống | | 2. Hệ thống tải xuống bài nhạc và lưu trữ vào máy theo dạng bainhac.mp3 |
|
| **Tìm kiếm nhạc** | 1. Tại trang danh sách các bài nhạc, người dùng bấm tìm kiếm bài nhạc theo tên bài, ca sĩ… | | 2. Hệ thống trả về thông tin gợi ý từ người dùng nhập vào |
|
|
|
|
| **Xem danh sách các bài nhạc** | 1. Tại trang chủ, người dùng bấm chọn danh mục “Danh sách bài nhạc” | | 2. Hệ thống hiển thị ra danh sách các bài nhạc |
| **Các điều kiện cụ thể** | | | |
| **Điều kiện trước** | | * Đã đăng nhập vào hệ thống | |
| **Điều kiện sau** | | * Hệ thống hiển thị thông báo thành công / thất bại khi tải nhạc xuống | |
| **Các điều kiện mở rộng** | | Không có | |

**Các tình huống sử dụng:**

| **STT** | **Đối tượng** | **Hoạt động** | **Thông tin đầu vào** | **Thông tin đầu ra** |
| --- | --- | --- | --- | --- |
| 1 | Quản lý/ User | Nghe nhạc | Đã chọn bài nhạc cần nghe | Bài nhạc được phát |
| 2 | Quản lý/ User | Tải nhạc xuống | Đã đăng nhập vào hệ thống, đã chọn bài nhạc cần tải | Bài nhạc được tải về và lưu trữ dưới dạng .mp3 |
| 3 | Quản lý/ User | Tìm kiếm nhạc | Thông tin cần tìm kiếm ví dụ như tìm kiếm theo ca sĩ, tên bài nhạc, thể loại… | Hệ thống trả về thông tin đúng theo thông tin tìm kiếm |
| 4 | Quản lý/ User | Xem danh sách các bài nhạc | Đã có các bài nhạc do người dùng tài lên | Hiển thị danh sách các bài nhạc |

### 3.3.3 Quản lý bình luận

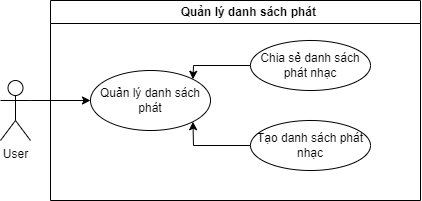


| **Mã use case:** | UC03\_QLBL | |
| --- | --- | --- |
| **Tên use case**: | Quản Lý Bình Luận | |
| **Actor**: | Admin, User | |
| **Mô tả**: | Use case cho phép   * User có thể xem, thêm, sửa, xóa bình luận. * Admin có thể xem tất cả các bình luận và xóa bình luận | |
| **Luồng sự kiện chính** | | |
|  | **Hành động** | **Phản ứng hệ thống** |
| **Xem danh sách bình luận** | 1. Nhấn vào mục quản lý bình luận | 2. Hiện lên danh sách các bình luận của người dùng |
| **Thêm** | 1. Nhấn nút thêm bình luận | 2. Hiển thị giao diện để nhập bình luận mới |
| 3. Nhập thông tin bình luận và đăng bình luận. | 4. Lưu thông tin bình luận lên hệ thống và thông báo nếu đăng thành công. |
| **Sửa** | 1. Chọn một bình luận cần sửa, nhấn nút sửa. | 2. Hiển thị giao diện để sửa thông tin bình luận. |
| 3. Nhập các thông tin cần thay đổi rồi nhấn đăng. | 4. Thông tin thay đổi được update lên hệ thống và thông báo sửa thành công |
| **Xóa** | 1. Chọn bình luận cần xóa, nhấn nút xóa | 2. Hiện thông báo xác nhận xóa. |
| 3. Chọn xác nhận xóa. | 4. Thông báo xóa thành công |
| **Các điều kiện cụ thể** | | |
| **Điều kiện trước:** Đã đăng nhập vào hệ thống | | |
| **Điều kiện sau:** Thông báo khi thêm, sửa, xóa thành công, thất bại | | |
| **Các điều kiện mở rộng:** Không có | | |

**Các tình huống sử dụng**

| **STT** | **Đối tượng** | **Hoạt động** | **Thông tin đầu vào** | **Thông tin đầu ra** |
| --- | --- | --- | --- | --- |
| 1 | Admin/User | Xem danh sách bình luận | Thống kê bình luận hiện đã có ở bài hát | Danh sách bình luận bài hát |
| 2 | User | Thêm bình luận | Đã đăng nhập tài khoản vào hệ thống, nhập thông tin cần bình luận | Bình luận được hiển thị theo bài hát |
| 3 | User | Sửa bình luận | Đã có bình luận, người dùng sửa thông tin bình luận | Thông tin bình luận được hiển thị theo nội dung đã sửa |
| 4 | Admin/User | Xóa bình luận | Đã có bình luận, bình luận vi phạm tiêu chuẩn cộng đồng | Bình luận được xóa khỏi hệ thống |

### 3.3.4 Quản lý danh sách phát



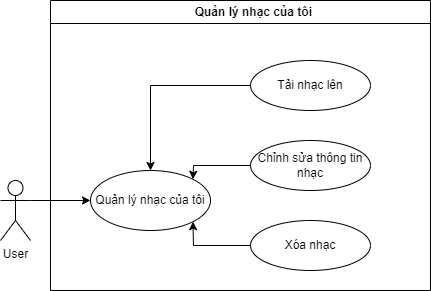
| **Mã use case:** | UC04\_QLDSP | |
| --- | --- | --- |
| **Tên use case**: | Quản Lý Danh Sách Phát | |
| **Actor**: | User | |
| **Mô tả**: | Use case cho phép   * User có thể tạo và chia sẻ danh sách phát | |
| **Luồng sự kiện chính** | | |
|  | **Hành động** | **Phản ứng hệ thống** |
| **Tạo danh sách phát** | 1. Người dùng chọn 1 bài nhạc bất kỳ, chọn dấu 3 chấm phía trên cùng bên phải màn hình | 2. Hệ thống hiển thị Thêm vào danh sách phát |
| 3. Người dùng bấm chọn “Thêm vào danh sách mới” | 4. Hệ thống hiển thị cho người dùng nhập tên danh sách phát |
| 5. Người dùng nhập tên danh sách vào và bấm Lưu | 6. Hệ thống lưu trữ bài nhạc vào danh sách phát đã được tạo |
| **Chia sẻ danh sách phát** | 1. Tại danh mục danh sách phát, người dùng bấm chọn Chia sẻ | 2. Hệ thống thực hiện chia sẻ danh sách phát cho người dùng khác |
| **Các điều kiện cụ thể** | | |
| **Điều kiện trước:** Đã đăng nhập vào hệ thống | | |
| **Điều kiện sau:** Thông báo khi tạo mới danh sách phát thành công/thất bại | | |
| **Các điều kiện mở rộng:** Không có | | |

**Các tình huống sử dụng**

| **STT** | **Đối tượng** | **Hoạt động** | **Thông tin đầu vào** | **Thông tin đầu ra** |
| --- | --- | --- | --- | --- |
| 1 | User | Tạo danh sách phát | Bài nhạc đã có trên hệ thống | Danh sách phát có chứa bài nhạc đã thêm |
| 2 | User | Chia sẻ danh sách phát | Danh sách phát đã được tạo | Danh sách được chia sẻ cho đối tượng khác |

### 

### 3.3.5 Quản lý nhạc của tôi



| **Mã use case:** | UC05\_QLNCT | |
| --- | --- | --- |
| **Tên use case**: | Quản Lý Nhạc Của Tôi | |
| **Actor**: | User | |
| **Mô tả**: | Use case cho phép   * User có thể tải nhạc lên, chỉnh sửa thông tin và xóa nhạc | |
| **Luồng sự kiện chính** | | |
|  | **Hành động** | **Phản ứng hệ thống** |
| **Tải nhạc lên** | 1. Người dùng nhấn vào mục Tải nhạc lên | 2. Hệ thống hiển thị thông tin cho người dùng nhập vào |
| 3. Người dùng nhập đầy đủ thông tin và bấm Lưu | 4. Hệ thống lưu trữ dữ liệu và hiển thị lên thông báo “Thành công” |
| **Chỉnh sửa thông tin nhạc** | 1. Người dùng bấm vào bài nhạc đã tải lên, nhận chọn icon Sửa | 2. Hệ thống hiển thị thông tin bài nhạc |
| 3. Người dùng sửa thông tin cần thiết và bấm lưu | 4. Hệ thống tiếp nhận thông tin và lưu trữ cập nhật lại |
| **Xóa nhạc** | 1. Người dùng bấm chọn icon Xóa bên cạnh bài nhạc | 2. Hệ thống hiển thị thông báo “Bạn có chắc chắn muốn xóa bài nhạc?” |
| 3. Người dùng bấm chọn “Đồng ý” | 4. Thông tin bài nhạc được xóa khỏi hệ thống |
| **Các điều kiện cụ thể** | | |
| **Điều kiện trước:** Đã đăng nhập vào hệ thống | | |
| **Điều kiện sau:** Thông báo khi thêm, sửa, xóa thành công, thất bại | | |
| **Các điều kiện mở rộng:** Không có | | |

**Các tình huống sử dụng**

| **STT** | **Đối tượng** | **Hoạt động** | **Thông tin đầu vào** | **Thông tin đầu ra** |
| --- | --- | --- | --- | --- |
| 1 | User | Tải nhạc lên | Thông tin về bài nhạc bao gồm file audio, tên bài, ca sĩ, thể loại… | Dữ liệu bài nhạc được tải lên hệ thống |
| 2 | User | Chỉnh sửa thông tin nhạc | Đã có bài nhạc trên hệ thống | Thông tin bài nhạc đã được chỉnh sửa |
| 3 | User | Xóa nhạc | Đã có bài nhạc trên hệ thống | Bài nhạc được xóa khỏi hệ thống |

### 

# Các yêu cầu phi chức năng

| **STT** | **Nội dung** | **Chi tiết tối thiểu cần có** |
| --- | --- | --- |
| 1 | Tính tin cậy | Thông tin hệ thống đưa ra phải có tính xác thực được người dùng tin cậy  Khi xảy ra các sự cố làm ngừng vận hành hệ thống, hệ thống phải đảm bảo phục hồi 100% trong vòng 24h |
| 2 | Tính hiệu năng | Khả năng chịu tải của hệ thống khi có nhiều người truy cập phải đảm bảo hoạt động bình thường  Thời gian tối đa cho phép để xử lý yêu cầu từ người dùng và trả về kết quả tìm kiếm trong hệ thống là 10 giây tính từ lúc người dùng gửi yêu cầu (dựa vào đường truyền mạng, mức độ ổn định của CPU) |
| 3 | Tính hữu dụng | Hệ thống phải phù hợp với nhu cầu sử dụng  Có tài liệu hướng dẫn sử dụng, vận hành hệ thống  Giao diện hệ thống phải dễ nhìn, dễ sử dụng  Truy cập hệ thống phải dễ dàng, nhanh chóng |
| 4 | Tính bảo mật | Các thông tin như thông tin người dùng (gồm email, password…) phải được hệ thống bảo vệ nghiêm ngặt chống rò rỉ thông tin ra bên ngoài  Yêu cầu xác thực khi đăng nhập ở một thiết bị mới |
| 5 | Khả năng bảo trì | Hệ thống có thể nâng cấp, bảo trì khi cần thiết hoặc có sự cố xảy ra |
| 6 | Tính văn hóa | Ngôn ngữ sử dụng hướng vào người dùng là người Việt Nam (vì phạm vi người dùng chỉ là toàn quốc) |
| 7 | Tính tương thích | Hệ thống cần tương thích đa thiết bị |

#### 

# Đánh giá tài liệu SRS

## 5.1 Các tiêu chí đánh giá

| **STT** | **Tiêu chí** | **Nội dung** |
| --- | --- | --- |
| 1 | Tính đầy đủ | SRS phải bao gồm tất cả các yêu cầu chức năng và phi chức năng của phần mềm  Mọi thông tin cần thiết cho việc phát triển phần mềm phải được trình bày rõ ràng và đầy đủ |
| 2 | Tính chính xác | Mọi thông tin trong SRS phải chính xác và cập nhật  Cần kiểm tra kỹ lưỡng để tránh các lỗi và thiếu sót |
| 3 | Tính rõ ràng | SRS phải được viết bằng ngôn ngữ dễ hiểu, súc tích và không mơ hồ  Tránh sử dụng các thuật ngữ chuyên môn quá phức tạp |
| 4 | Tính tổ chức | SRS phải được tổ chức một cách logic và dễ theo dõi  Sử dụng các tiêu đề, phụ đề và bảng để sắp xếp thông tin một cách khoa học |
| 5 | Tính nhất quán | Mọi thông tin trong SRS phải nhất quán với nhau  Tránh mâu thuẫn giữa các phần khác nhau của tài liệu |
| 6 | Tính khả thi | Các yêu cầu trong SRS phải có thể thực hiện được với công nghệ và nguồn lực sẵn có  Tránh đưa ra các yêu cầu không thể thực hiện hoặc quá tốn kém |
| 7 | Tính có thể kiểm tra | Các yêu cầu trong SRS phải có thể kiểm tra được để đảm bảo rằng phần mềm đáp ứng các yêu cầu đề ra  Cần xác định các tiêu chí kiểm tra cho từng yêu cầu |
| 8 | Tính truy cập | SRS phải dễ dàng truy cập bởi tất cả các bên liên quan trong dự án  Cần sử dụng các công cụ phù hợp để quản lý và chia sẻ tài liệu |

## 5.2 Kết quả đánh giá

| **STT** | **Tiêu chí** | **Đánh giá** | **Nhận xét** |
| --- | --- | --- | --- |
| 1 | Tính đầy đủ | Tương đối | SRS đã bao gồm đủ 18 chức năng và 7 yêu cầu phi chức năng |
| 2 | Tính chính xác | Chưa thể khẳng định | Mọi thông tin chưa thể khẳng định chắc chắn đúng hết |
| 3 | Tính rõ ràng | Tốt | SRS được viết bằng tiếng Việt dễ hiểu, có chú thích rõ ràng các từ ngữ viết tắt |
| 4 | Tính tổ chức | Tương đối | SRS đã sử dụng tiêu đề, phụ đề và bảng sắp xếp thông tin một cách khoa học |
| 5 | Tính nhất quán | Tương đối | Mọi thông tin trong SRS đã nhất quán với nhau |
| 6 | Tính khả thi | Chưa thể khẳng định | Chưa thể khẳng định được tất cả các yêu cầu trong SRS có thể thực hiện được với nguồn lực hiện tại |
| 7 | Tính có thể kiểm tra | Tương đối | Các yêu cầu trong SRS có thể kiểm tra được |
| 8 | Tính truy cập | Tốt | SRS đã dễ dàng truy cập |