

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN ĐIỆN TỬ - VIỄN THÔNG



ĐỒ ÁN
TỐT NGHIỆP ĐẠI HỌC

Đề tài:

**PHÁT TRIỂN PHẦN MỀM
PHÁT AUDIO VÀ VIDEO**

Sinh viên thực hiện: **ĐỖ TRUNG HIẾU**
Lớp ĐT6 - K59
Giảng viên hướng dẫn: **TS. VÕ LÊ CƯỜNG**

Hà Nội, 1-2021

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN ĐIỆN TỬ - VIỄN THÔNG



ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC

Đề tài:

PHÁT TRIỂN PHẦN MỀM PHÁT AUDIO VÀ VIDEO

Sinh viên thực hiện: ĐỖ TRUNG HIẾU
Giảng viên hướng dẫn: Lớp ĐT6 - K59
TS. VÕ LÊ CUỜNG
Cán bộ phản biện:

Hà Nội, 1-2021

**Đánh giá quyển đồ án tốt nghiệp
(Dùng cho giảng viên hướng dẫn)**

Giảng viên đánh giá:

Họ và tên Sinh viên: Đỗ Trung Hiếu MSSV: 20141501

Tên đồ án: Thiết kế và phát triển phần mềm ứng dụng đa phương tiện

Chọn các mức điểm phù hợp cho sinh viên trình bày theo các tiêu chí dưới đây:

Rất kém (1); Kém (2); Đạt (3); Giỏi (4); Xuất sắc (5)

3. Nhận xét thêm của Thầy/Cô (giảng viên hướng dẫn nhận xét về thái độ và tinh thần làm việc của sinh viên)

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

Ngày: / /201

Người nhận xét

(Ký và ghi rõ họ tên)

Đánh giá quyển đồ án tốt nghiệp

(Dùng cho cán bộ phản biện)

Giảng viên đánh giá:

Họ và tên Sinh viên: Đỗ Trung Hiếu MSSV: 20141501

Tên đồ án: Thiết kế và phát triển phần mềm ứng dụng đa phương tiện

Chọn các mức điểm phù hợp cho sinh viên trình bày theo các tiêu chí dưới đây:

Rất kém (1); Kém (2); Đạt (3); Giới (4); Xuất sắc (5)

Có sự kết hợp giữa lý thuyết và thực hành (20)					
1	Nêu rõ tính cấp thiết và quan trọng của đề tài, các vấn đề và các giả thuyết (bao gồm mục đích và tính phù hợp) cũng như phạm vi ứng dụng của đồ án	1	2	3	4
2	Cập nhật kết quả nghiên cứu gần đây nhất (trong nước/quốc tế)	1	2	3	4
3	Nêu rõ và chi tiết phương pháp nghiên cứu/giải quyết vấn đề	1	2	3	4
4	Có kết quả mô phỏng/thực nghiệm và trình bày rõ ràng kết quả đạt được	1	2	3	5
Có khả năng phân tích và đánh giá kết quả (15)					
5	Kế hoạch làm việc rõ ràng bao gồm mục tiêu và phương pháp thực hiện dựa trên kết quả nghiên cứu lý thuyết một cách có hệ thống	1	2	3	4
6	Kết quả được trình bày một cách logic và dễ hiểu, tất cả kết quả đều được phân tích và đánh giá thỏa đáng.	1	2	3	4
7	Trong phân kết luận, tác giả chỉ rõ sự khác biệt (nếu có) giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai.	1	2	3	5
Kỹ năng viết (10)					
8	Đồ án trình bày đúng mẫu quy định với cấu trúc các chương logic và đẹp mắt (bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đề cập đến trong đồ án, có cẩn lè, dấu cách sau dấu chấm, dấu phẩy v.v), có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn đúng quy định	1	2	3	4
9	Kỹ năng viết xuất sắc (cấu trúc câu chuẩn, văn phong khoa học, lập luận logic và có cơ sở, từ vựng sử dụng phù hợp v.v.)	1	2	3	5
Thành tựu nghiên cứu khoa học (5) (chọn 1 trong 3 trường hợp)					
10a	Có bài báo khoa học được đăng hoặc chấp nhận đăng/đạt giải SVNC khoa học giải 3 cấp Viện trở lên/các giải thưởng khoa học (quốc tế/trong nước) từ giải 3 trở lên/ Có đăng ký bằng phát minh sáng chế	5			
10b	Được báo cáo tại hội đồng cấp Viện trong hội nghị sinh viên nghiên cứu khoa học nhưng không đạt giải từ giải 3 trở lên/Đạt giải khuyến	2			

	khích trong các kỳ thi quốc gia và quốc tế khác về chuyên ngành như TI contest.	
10c	Không có thành tích về nghiên cứu khoa học	0
Điểm tổng		/50
Điểm tổng quy đổi về thang 10		

3. Nhận xét thêm của Thầy/Cô

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

Ngày: / /201

Người nhận xét
(Ký và ghi rõ họ tên)

LỜI NÓI ĐẦU

Sau thời gian học tập tại trường, được sự chỉ bảo hướng dẫn nhiệt tình của thầy cô giáo thuộc Viện Điện Tử - Viễn Thông nói riêng, và trường Đại học Bách Khoa Hà Nội nói chung, em đã kết thúc chương trình học và đã tích luỹ được vốn kiến thức nhất định. Được sự đồng ý của nhà trường và thầy cô giáo trong khoa, em được giao đề tài đồ án tốt nghiệp: “Phát triển phần mềm phát audio và video”.

Đồ án tốt nghiệp của em được thực hiện với mục tiêu: Vận dụng kỹ năng các kỹ năng phân tích, thiết kế, lập trình để phát triển ứng dụng đa phương tiện, phục vụ các tính năng phát file audio và video.

Bằng sự cố gắng nỗ lực của bản thân và đặc biệt là sự giúp đỡ tận tình của giảng viên hướng dẫn – TS. Võ Lê Cường, em đã hoàn thành đồ án đúng thời hạn. Do thời gian làm đồ án có hạn và trình độ của bản thân còn nhiều hạn chế, nên không thể tránh khỏi những thiếu sót. Em rất mong nhận được sự đóng góp ý kiến của các thầy cô cũng như là của các bạn sinh viên để đề tài đồ án này hoàn thiện hơn nữa.

Em xin chân thành cảm ơn giáo viên hướng dẫn – TS. Võ Lê Cường, các thầy cô giáo thuộc viện Điện tử Viễn thông và Đại học Bách Khoa Hà Nội nói chung, đã giúp đỡ em từ những ngày học đầu tiên, đến hoàn thiện đồ án tốt nghiệp, và cả quãng đường cuộc đời sau này.

Hà Nội, ngày 26 tháng 12 năm 2020

Sinh viên thực hiện

Đỗ Trung Hiếu

LỜI CAM ĐOAN

Tôi là Đỗ Trung Hiếu, mã số sinh viên 20141501, sinh viên lớp Điện tử 6, khóa 59. Người hướng dẫn là TS. Võ Lê Cường. Tôi xin cam đoan toàn bộ nội dung được trình bày trong đồ án *Phát triển ứng dụng phát audio và video* là kết quả quá trình tìm hiểu và nghiên cứu của tôi. Các dữ liệu được nêu trong đồ án là hoàn toàn trung thực, phản ánh đúng kết quả đo đạc thực tế. Mọi thông tin trích dẫn đều tuân thủ các quy định về sở hữu trí tuệ; các tài liệu tham khảo được liệt kê rõ ràng. Tôi xin chịu hoàn toàn trách nhiệm với những nội dung được viết trong đồ án này.

Hà Nội, ngày 12 tháng 1 năm 2021

Người cam đoan

MỤC LỤC

DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT	i
DANH MỤC HÌNH VẼ	ii
DANH MỤC BẢNG BIỂU	iv
TÓM TẮT ĐỒ ÁN	v
ABSTRACT	vi
PHẦN MỞ ĐẦU.....	1
Chương 1. Cơ sở lý thuyết về chuẩn MP3 và MP4.....	3
1.1. <i>Chuẩn MP3.....</i>	3
1.1.1. Tổng quan	3
1.1.2. Mã Huffman.....	4
1.1.3. Sơ lược cấu trúc file MP3	5
1.1.4. Sơ lược giải mã file MP3	8
1.2. <i>Chuẩn MP4.....</i>	9
1.2.1. Tổng quan	9
1.2.2. Sơ lược cấu trúc file MP4	9
1.3. <i>Kết luận chương.....</i>	10
Chương 2. Giới thiệu về Qt Framework và mô hình MVC	11
2.1. <i>Giới thiệu về Qt Framework</i>	11
2.2. <i>Cơ chế Signal – Slot trong Qt</i>	12
2.1. <i>Hệ thống thuộc tính trong Qt</i>	14
2.2. <i>Giới thiệu về ngôn ngữ QML</i>	14
2.3. <i>Module Qt Multimedia</i>	16
2.4. <i>Mô hình thiết kế phần mềm MVC</i>	16
2.4.1. Tổng quan	16
2.4.2. Khối Model	18
2.4.3. Khối View	18
2.4.4. Khối Controller	18
2.5. <i>Kết luận chương.....</i>	19

Chương 3. Thiết kế đề tài	20
3.1. Phân tích yêu cầu hệ thống.....	20
3.1.1. Yêu cầu chức năng cho chức năng phát audio	20
3.1.2. Yêu cầu chức năng cho chức năng phát file video.....	22
3.1.3. Yêu cầu phi chức năng.....	23
3.2. Thiết kế kiến trúc của ứng dụng.....	23
3.2.1. Sơ đồ khái	23
3.2.2. Các khái chính	24
3.2.3. Các khái còn lại	24
3.2.4. Tương tác giữa các khái.....	24
3.3. Thiết kế chi tiết.....	25
3.3.1. Thiết kế các Class	25
3.3.2. Đặc tả các chức năng phát audio.....	27
3.3.3. Đặc tả các chức năng phát video.....	37
3.3.4. Mô tả bộ cục giao diện	39
3.4. Kết luận chương.....	43
Chương 4. Kiểm thử và đánh giá kết quả.....	44
4.1. Hình ảnh thực tế của sản phẩm	44
4.2. Kiểm thử các yêu cầu.....	47
4.2.1. Kiểm thử và đánh giá chức năng phát file audio.....	49
4.2.2. Kiểm thử và đánh giá chức năng phát file video.....	56
4.3. Đánh giá mức độ tuân thủ mô hình MVC	57
4.4. Kết luận chương.....	58
Kết luận	59
Tài liệu tham khảo	60
Bảng đổi chiểu thuật ngữ Việt - Anh.....	61
PHỤ LỤC	62
A. Chi tiết các class	62
B. Chi tiết các màn hình	74
1. Chi tiết bộ cục giao diện	74
2. Chi tiết hành vi của các màn hình	78

DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT

<i>MP3</i>	MPEG-1 Layer III
<i>MPEG</i>	Moving Pictures Expert Group
<i>ISO</i>	International Organization for Standardization
<i>MP4</i>	MPEG-4 Part 14
<i>IMDCT</i>	Inverse Modified Discrete Cosine Transform
<i>MVC</i>	Model-View-Controller
<i>VideoQML</i>	Video QML Type
<i>Qt</i>	Qt Framework
<i>QML</i>	Qt Modelling Language
<i>CSS</i>	Cascading Style Sheet
<i>OS</i>	Operating System
<i>RTOS</i>	Real-Time Operating System
<i>API</i>	Application Programming Interface
<i>GUI</i>	Graphical User Interface
<i>JSON</i>	JavaScript Object Notation
<i>XML</i>	eXtensible Markup Language
<i>OOP</i>	Object-Oriented Programming
<i>RAM</i>	Random Access Memory
<i>CPU</i>	Central Processing Unit

DANH MỤC HÌNH VẼ

Hình 1.1 Ví dụ về mã Huffman [1].....	4
Hình 1.2 Greedy Huffman Algorithm [1]	4
Hình 1.3 Cấu trúc frame trong file MP3 [1]	5
Hình 1.4 Cấu trúc Header của frame MP3	5
Hình 1.5 Cấu trúc Side Information của frame MP3 [1]	7
Hình 1.6 Cấu trúc Main Data của frame MP3 [1].....	8
Hình 1.7 Giải mã frame MP3 [1]	8
Hình 1.8 Cấu trúc file MP4 [2]	9
Hình 2.1 Cơ chế signal - slot của Qt [4]	13
Hình 2.2 Tương tác giữa các khôi trong mô hình MVC [5]	17
Hình 3.1 Sơ đồ khôi của ứng dụng.....	23
Hình 3.2 Biểu đồ Class.....	25
Hình 3.3 Chức năng lập và hiển thị danh sách phát.....	27
Hình 3.4 Cập nhật hiển thị thời lượng đã phát.....	28
Hình 3.5 Cập nhật hiển thị thời lượng tổng	29
Hình 3.6 Điều khiển dừng/phát.....	30
Hình 3.7 Chuyển về nội dung liền trước	31
Hình 3.8 Chức năng chuyển đến nội dung tiếp theo	32
Hình 3.9 Chức năng chọn phát một nội dung từ danh sách	33
Hình 3.10 Chức năng chọn phát từ vị trí bất kì.....	34
Hình 3.11 Chuyển chế độ phát lặp lại	35
Hình 3.12 Lưu thông tin phát gần nhất	36
Hình 3.13 Tự động tải file được phát gần nhất khi khởi động.....	37
Hình 3.14 Lập và hiển thị danh sách phát video	38
Hình 3.15 Tua đến 10 giây trước (sau)	39
Hình 3.16 Màn hình Home.....	40

Hình 3.17 Màn hình Audio - Player.....	41
Hình 3.18 Màn hình Audio - Playlist	42
Hình 3.19 Màn hình Video - Playlist	42
Hình 4.1 Màn hình Home (sản phẩm).....	44
Hình 4.2 Màn hình Audio - Player (sản phẩm).....	45
Hình 4.3 Màn hình Audio - Playlist (sản phẩm)	45
Hình 4.4 Màn hình Video - Playlist (sản phẩm)	46
Hình 4.5 Màn hình Video - Player (sản phẩm)	46
Hình 4.6 Ứng dụng Groove Music của Microsoft	47
Hình 4.7 Ứng dụng Movies & TV của Microsoft.....	48
Hình 4.8 Kết quả kiểm thử lập danh sách phát - 1	50
Hình 4.9 Kết quả kiểm thử lập danh sách phát - 2	51
Hình 4.10 Kiểm thử tính năng lập danh sách phát video – MyMediaApp	57
Hình 4.11 Kiểm thử tính năng lập danh sách phát video – Movies & TV	57
Hình 4.12 Màn hình Audio - Playlist	76

DANH MỤC BẢNG BIỂU

Bảng 1.1 Bitrate của các lớp trong chuẩn MPEG-1	3
Bảng 1.2 Bảng đối chiếu bitrate của chuẩn MP3	6
Bảng 4.1 Tập dữ liệu kiểm thử audio.....	49
Bảng 4.2 Kết quả kiểm thử tính năng hiển thị meta-data	52
Bảng 4.3 Kết quả kiểm thử tính năng hiển thị thời lượng đã phát.....	53
Bảng 4.4 Kết quả kiểm thử tính năng hiển thị thời lượng tổng của file	54
Bảng 4.5 Tập dữ liệu kiểm thử video.....	56

TÓM TẮT ĐỒ ÁN

Đồ án tốt nghiệp của em tìm hiểu về hai định dạng file audio và video phổ biến bậc nhất trên thế giới hiện nay: định dạng audio MP3 và định dạng video MP4, và phát triển ứng dụng có chức năng phát audio và video hỗ trợ hai định dạng file này. Luận văn giới thiệu về kiến trúc của một file MP3 và trình bày sơ bộ quy trình giải mã và tái tạo tín hiệu *analog* từ định dạng file này. Với định dạng MP4, do hạn chế về thời gian và độ phức tạp cao của định dạng, em chỉ thực hiện tìm hiểu sơ bộ về cấu trúc file của định dạng. Ứng dụng phát audio và video được phát triển qua các pha: phân tích yêu cầu chức năng và phi chức năng, thiết kế kiến trúc của ứng dụng áp dụng mô hình MVC, thiết kế giao diện của ứng dụng tham khảo theo các nền tảng nổi tiếng như *Spotify* và *Youtube*, *Netflix*, lập trình ứng dụng với công cụ *Qt Framework* (đặc biệt là module *Qt Multimedia* và *QML*), và kiểm thử ứng dụng bằng phương pháp hộp đen (*black box*) với XX kịch bản, được thiết kế dựa theo các yêu cầu chức năng (*requirement-base*). Tập dữ liệu mẫu đầu vào bao gồm 8 file MP3 và 5 file MP4. Kết quả 45/49 kịch bản được kiểm thử thành công, hoàn thiện 9/10 yêu cầu chức năng.

ABSTRACT

My graduation project focus research of the two most popular audio and video file formats in the world nowaday: MP3 audio format and MP4 video format, and development and desktop application supporting audio and video playback with these two file formats. The thesis introduces the anatomy of an MP3 file and presents a preliminary process of decoding and reconstructing analog signals from this file format. With the MP4 format, due to the time constraints and the high complexity of the format, I only did a preliminary study of the format's file structure. Application that play audio and video are developed through phases: functional and non-functional requirements analysis, architectural design of the application applying MVC model, design and implement graphics user interface referencing to well-known platforms such as *Spotify* and *Youtube*, application programming using *Qt Framework* (especially the *Qt Multimedia* module), and application testing using *black-box* method with XX scripts, designed based on functional requirement (requirement-base). The input sample data set includes 8 MP3 files and 5 MP4 files. In results, 45/49 script was tested successfully, covering 9/10 functional requirements.

PHẦN MỞ ĐẦU

Đặt vấn đề

Trong cuộc sống hiện đại, nội dung đa phương tiện, tiêu biểu là audio và video, là một phần không thể thiếu. Cùng với sự phát triển của các thiết bị điện tử như máy tính cá nhân, điện thoại, TV thông minh, các hệ thống thông tin – giải trí trên xe hơi, việc cung cấp cho người dùng khả năng tiêu thụ nội dung đa phương tiện càng được đề cao, và trở thành lĩnh vực đầy cạnh tranh, điển hình là giữa các nền tảng lớn, ví dụ như Facebook và Youtube trong mảng video, hay Spotify và Apple Music trong mảng video,...

Với mong muốn nắm được nhưng hiểu biết cơ bản về lĩnh vực đa phương tiện, cụ thể là audio và video, em đã lựa chọn đề tài “Phát triển ứng dụng phát audio và video” cho đồ án tốt nghiệp của mình.

Mục đích nghiên cứu

Với đề tài “Phát triển ứng dụng phát audio và video”, đồ án của em giới hạn phạm vi phát triển gồm hai tính năng: phát và điều khiển phát file audio, phát và điều khiển phát file video. Các yêu cầu của đề tài được đặt ra bao gồm:

- Tìm hiểu tổng quan về chuẩn nén audio MP3, cấu trúc file MP3 và sơ bộ về quy trình giải mã, tái tạo tín hiệu tương tự từ file MP3
 - Tìm hiểu tổng quan về chuẩn audio MP4 và cấu trúc file MP4
 - Phân tích yêu cầu chức năng, phi chức năng cho ứng dụng phát audio và video, tham khảo theo các chức năng của nền tảng audio Spotify và nền tảng video Youtube
 - Thiết kế kiến trúc cho ứng dụng theo mô hình MVC
 - Thiết kế giao diện đồ họa người dùng (GUI) cho ứng dụng, tham khảo Spotify và Youtube
 - Phát triển ứng dụng với công cụ Qt Framework, sử dụng các module Qt Core, Qt Multimedia, Qt QML

Phương pháp nghiên cứu

Trong đề tài này, em đã sử dụng các phương pháp nghiên cứu:

- Phương pháp tham khảo từ tài liệu: Thu thập thông tin từ các tạp chí công nghệ, các blog trên Internet.

- Phương pháp quan sát: Khảo sát giao diện và tính năng của một số nền tảng audio và video lớn, ví dụ Spotify và Youtube

Bô cục đồ án

Trong Chương 1, em trình bày cơ sở lý thuyết của đề tài, bao gồm tổng quan về chuẩn MP3 và MP4, cấu trúc file MP3 và MP4, quy trình giải mã và tái tạo tín hiệu analog từ file MP3

Chương 2, em giới thiệu về Qt Framework và trình bày lý do lựa chọn Qt Framework làm công cụ phát triển cho đồ án, trình bày các kiến thức tìm hiểu được về mô hình thiết kế phần mềm Model-View-Controller (MVC), và tổng quan về phương pháp kiểm thử black box

Chương 3 trình bày về quá trình xây dựng ứng dụng, bao gồm phân tích các yêu cầu chức năng đối với 2 tính năng phát file audio và phát file video, thiết kế kiến trúc của ứng dụng, áp dụng mô hình MVC, thiết kế chi tiết cho các chức năng, thiết kế GUI.

Chương 4 trình bày quá trình kiểm thử, bao gồm các kịch bản kiểm thử và kết quả kiểm thử, qua đó đánh giá mức độ đáp ứng yêu cầu chức năng của ứng dụng, đồng thời cung cấp một số hình ảnh của ứng dụng.

Chương 1. Cơ sở lý thuyết về chuẩn MP3 và MP4

Chương 1 trình bày cơ sở lý thuyết của đề tài, bao gồm tổng quan về chuẩn MP3 và MP4, cấu trúc file MP3 và MP4, sơ bộ quy trình giải mã và tái tạo tín hiệu analog từ file MP3.

1.1. Chuẩn MP3

1.1.1. Tổng quan

Chuẩn nén âm thanh MPEG-1 Layer III (MP3) là lớp thứ 3 của chuẩn MPEG-1, được nghiên cứu, phát triển bởi Moving Picture Experts Group (MPEG), trực thuộc tổ chức ISO. MPEG-1 được phát triển với mục tiêu trở thành một chuẩn chung cho việc nén/mã hóa ảnh động, audio và video. Tuân thủ theo MPEG-1, các nội dung audio, video có thể được nén và giảm thiểu dung lượng đi đáng kể, mà không làm giảm quá nhiều chất lượng nội dung bên trong. Chi tiết về hệ số nén của các lớp trong MPEG-1 được trình bày trong bảng sau:

Bảng 1.1 Bitrate của các lớp trong chuẩn MPEG-1

Lớp	Tỉ lệ	Bitrate
PCM CD (nguyên bản)	1:1	1,4 Mbps
Layer I	4:1	384 Kbps
Layer II	8:1	192 Kbps
Layer III	12:1	128 Kbps

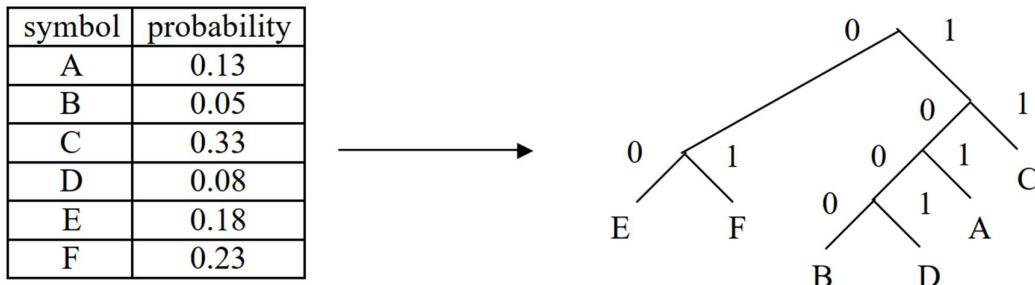
Với chuẩn MP3, nội dung âm thanh có thể được nén lại (thông thường) tới 12 lần, với lượng bitrate giảm từ 1,4 Mbps của PCM CD xuống 128 kbps. Điều này khiến MP3 trở thành chuẩn nén hiệu quả nhất trong 3 lớp, nhưng đồng thời cũng là chuẩn nén yêu cầu độ phức tạp cao nhất. Dù vậy, độ hiệu quả vẫn góp phần không nhỏ biến MP3 thành chuẩn nén audio thông dụng bậc nhất.

MPEG-1 nói chung và MP3 nói riêng tận dụng giới hạn và khả năng nghe của tai người, dựa vào đó để lọc bỏ các thông tin không cần thiết và không có nghĩa trong dữ liệu âm thanh gốc, chỉ mã hóa phần nội dung có nghĩa. Khi giải mã, phần nội dung không có nghĩa không thể được khôi phục, nhưng đối với người nghe, điều này là hoàn toàn chấp nhận được. Ngoài ra, để tăng tính hiệu quả của chuẩn nén, MPEG kết hợp quá trình lược bỏ thông tin thừa với mã hóa sử dụng mã Huffman. Quá trình mã hóa tương đối phức tạp và không thực sự phục vụ cho mục tiêu phát triển ứng dụng, nên em sẽ không đề cập đến trong luận văn.

Với mục tiêu phát triển ứng dụng phát file audio MP3, em sẽ đi sâu vào cấu trúc và phương pháp giải mã file MP3.

1.1.2. Mã Huffman

Mã Huffman được áp dụng trong mã hóa và giải mã file MP3. Cơ sở về ý tưởng của mã Huffman: phần tử có trọng số càng lớn, sẽ được mã hóa càng ngắn, giúp tối ưu dung lượng đường truyền. Trước khi thực hiện mã hóa Huffman, cần xác định được trọng số của mỗi phần tử như ví dụ trong Hình 1.1:

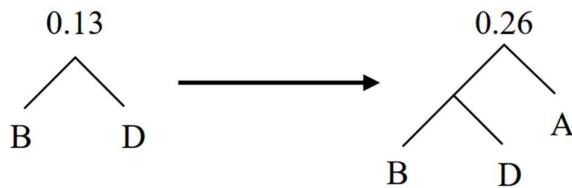


Hình 1.1 Ví dụ về mã Huffman [1]

Theo Hình 1.1, các phần tử E và F có trọng số lớn nhất, sẽ được mã hóa ngắn nhất tương ứng là 00 và 01. Ngược lại B và D có trọng số nhỏ nhất, sẽ được mã hóa tương ứng là 1000 và 1001.

Để tạo cây mã tối ưu, có thể sử dụng giải thuật tham lam (*greedy algorithm*) như sau:

- Tìm 2 phần tử có trọng số nhỏ nhất
- Gộp 2 phần tử vừa tìm được thành phần tử mới có trọng số bằng tổng trọng số của 2 phần tử
- Lặp lại 2 bước trên đến khi tất cả phần tử đều xuất hiện trong cây mã (ví dụ Hình 1.2, với bảng trọng số trong Hình 1.1)



Hình 1.2 Greedy Huffman Algorithm [1]

Hình 1.2 mô tả 2 vòng lặp trong áp dụng giải thuật tham lam Huffman: 2 phần tử trọng số nhỏ nhất được chọn đầu tiên là B và D, sau khi gộp sẽ được trọng số 0.13,

và lặp lại với phần tử trọng số nhỏ tiếp theo là A, gộp lại được trọng số 0.26. Tiếp tục vòng lặp sẽ cho kết quả cây mã như Hình 1.1.

Để có thể giải mã Huffman, cần xác định trước thông tin về trọng số.

1.1.3. Sơ lược cấu trúc file MP3

Một file MP3 được cấu thành bởi từ các khung (frame) liền kề nhau. Mỗi frame lưu trữ 1152 mẫu và kéo dài 26 mili giây (ms). Mỗi khung lại được chia thành 2 phần (granule), mỗi granule lưu trữ 576 mẫu. Hình 1.3 mô tả cấu trúc một frame:

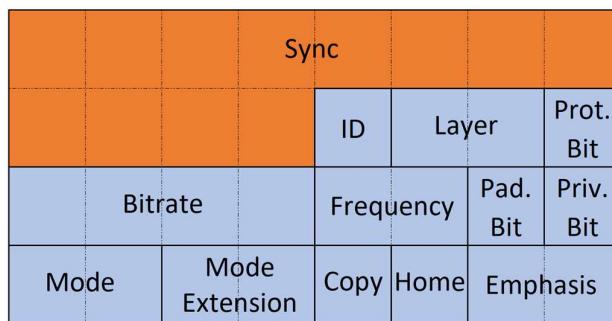
Header	CRC	Side Information	Main Data	Ancillary Data
--------	-----	------------------	-----------	----------------

Hình 1.3 Cấu trúc frame trong file MP3 [1]

Mỗi frame được cấu thành bởi các trường:

- Header: chứa các thông tin về frame
- CRC: chứa mã vòng CRC dùng cho xác định tính đúng đắn của frame
- Side Infomation: chứa thông tin dùng cho giải mã Huffman phần Main Data
- Main Data: chứa thông tin về tỉ lệ nén (*scale-factor*) và các bit dữ liệu đã được mã hóa Huffman
- Ancillary Data: chứa các thông tin bổ sung, phần này có thể có hoặc không

Trường Header có độ dài 32 bit với cấu trúc như sau:



Hình 1.4 Cấu trúc Header của frame MP3

Sync (12 bit): 12 bit đồng bộ, cả 12 bit đều có giá trị 1 (1111 1111 1111). Một receiver MP3 sẽ tìm 12 bit này để xác định điểm bắt đầu của 1 frame, từ đó có thể đọc các thông tin tiếp theo và tiến hành giải mã.

ID (1 bit): Xác định phiên bản của chuẩn MPEG được sử dụng. 1 ứng với MPEG-1, 0 ứng với MPEG-2. Với MP3 sử dụng MPEG-1, ID có giá trị bằng 1.

Layer (2 bit): Xác định layer được áp dụng trong chuẩn MPEG-1. Các giá trị bao gồm: 01 – Layer III, 10 – Layer II, 11 – Layer I. Giá trị 00 không được sử dụng. Với MP3, Layer có giá trị 01

Protection bit (1 bit): Với giá trị 1, trường CRC sẽ được sử dụng

Bit rate (4 bit): Cung cấp thông tin cho bộ giải mã về bitrate được sử dụng để mã hóa. Giá trị tương ứng của các bit được trình bày trong Bảng 1.2:

Bảng 1.2 Bảng đối chiếu bitrate của chuẩn MP3

Bit	MP3 bitrate (kbps)	Bit	MP3 bitrate (kbps)
0 0 0 0	-	1 0 0 0	112
0 0 0 1	32	1 0 0 1	128
0 0 1 0	40	1 0 1 0	160
0 0 1 1	48	1 0 1 1	192
0 1 0 0	56	1 1 0 0	224
0 1 0 1	64	1 1 0 1	256
0 1 1 0	80	1 1 1 0	320
0 1 1 1	93	1 1 1 1	-

Bảng 1.2 cho thấy MP3 hỗ trợ nén và mã hóa audio với các bitrate trong khoảng từ 32 kbps tới 320 kbps. Tuy nhiên phổ biến nhất vẫn là 128 kbps.

Frequency (2 bit): Tần số lấy mẫu của dữ liệu. Các giá trị bao gồm: 00 – 44,1 KHz, 01 – 48 KHz, 10 – 32 KHz. Giá trị 11 không được sử dụng.

Padding bit (1 bit): Giá trị bằng 1 nếu frame cần thêm các byte padding ở cuối.

Private bit (1 bit): Tùy ứng dụng sử dụng

Mode (2 bit): Chế độ kênh: 00 – Stereo, 01 – Joint Stereo, 10 – Dual Channel, 11 – Single Channel

Các bit còn lại không có ứng dụng trong giải mã nên em không đi vào chi tiết.

Trường CRC (Cyclic Redundancy Check):

Trường CRC sẽ tồn tại khi bit *Protection Bit* trong trường Header có giá trị bằng 1. Nếu được sử dụng, CRC có độ dài 16 bit. CRC được ứng dụng để kiểm tra tính đúng đắn của 2 phần dữ liệu nhạy cảm (*sensitive data*) của frame, là bit 16 đến 31 của trường Header và trường Side Infomation. Nếu bất kì phần nào trong 2 phần này bị lỗi sẽ dẫn đến cả frame đều bị lỗi. Một frame lỗi có thể bị bỏ qua (dẫn đến tắt tiếng), hoặc thay thế bởi frame trước đó (lặp lại tiếng). Do giới hạn thời gian của đồ án, em sẽ không đi vào chi tiết cách tạo và kiểm tra CRC.

Trường Side Information:

Trường Side Information chứa các thông tin dùng cho giải mã dữ liệu trong Main Data. Kích thước của Side Information phụ thuộc vào chế độ kênh được thiết lập trong Header (*Mode* bit). Nếu *Mode* được thiết lập là *Single Channel*, độ dài của Side Information là 17 bit. Với các *Mode* còn lại, Side Information có độ dài 32 bit. Cấu trúc trường Side Information được thể hiện trong Hình 1.5:

main_data_begin	private_bits	scfsi	Side_info gr. 0	Side_info gr. 1
-----------------	--------------	-------	-----------------	-----------------

Hình 1.5 Cấu trúc Side Information của frame MP3 [1]

main_data_begin (9 bit): Trong định dạng MP3, kĩ thuật *bit reservoir* được áp dụng. Kĩ thuật này cho phép các khoảng dữ liệu dư ra trong một frame có thể được sử dụng bởi frame kế tiếp. Khi thực hiện giải mã MP3, để xác định vị trí bắt đầu thực tế của dữ liệu trong một frame, bộ giải mã cần xác định giá trị của *main_data_begin*. Giá trị này là phần số byte cần dịch về bên trái (*negative offset*) tính từ bit đầu tiên của phần trường Header. Độ dài 9 bit đồng nghĩa với số bit cần dịch có thể lên tới $(2^9 - 1) * 8 = 4088$ (bit). Điều này thể hiện rằng dữ liệu của một frame có thể nằm ở *một vài* frame trước đó. Khi tiến hành dịch bit để xác định dữ liệu, cần bỏ qua các phần không thay đổi kích thước (ví dụ Header). Nếu *main_data_begin* = 0, dữ liệu của Main Data bắt đầu ngay sau Side Information.

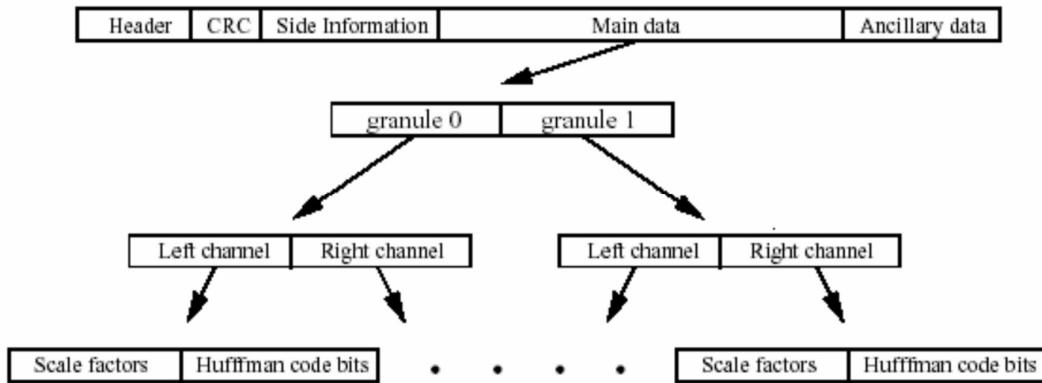
private_bits (5 bit nếu *Mode* là *Single Channel* hoặc 3 bit): Các bit sử dụng cho mục đích cá nhân hóa, ISO sẽ không sử dụng các bit này kể cả trong các phiên bản tiếp theo. Nếu

scfsi (4 bit nếu *Mode* là *Single Channel* hoặc 8 bit): *SCaleFactor Selection Information* – xác định về hệ số tỷ lệ giữa 2 granule là giống hay khác nhau..

Side_info gr.0 và *Side_info gr.1*: Chứa các thông tin phụ trợ cho việc giải mã *granule0* và *granule1* (Main Data được chia thành 2 *granule*). 2 trường có cấu trúc giống nhau, nhưng chứa thông tin riêng biệt cho từng *granule*. Các thông tin có vai trò xác định vị trí của *granule* tiếp theo (vì Main Data có thể nằm rải rác trong nhiều frame – kĩ thuật *bit reservoir*), và xác định các thông số dùng để giải mã Huffman với dữ liệu trong Main Data.

Trường Main Data:

Main Data được cấu thành từ 2 *granule*, mỗi *granule* được cấu thành từ hệ số tỷ lệ (*scalefactors*) và dữ liệu được mã hóa Huffman (*Huffman code bits*)



Hình 1.6 Cấu trúc Main Data của frame MP3 [1]

scalefactors: hệ số tỷ lệ. Scalefactors có tác dụng giảm nhiễu khi thực hiện lượng tử hóa tín hiệu.

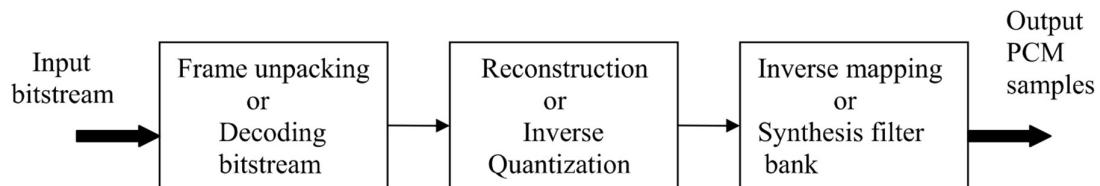
Huffman code bits: phần dữ liệu chính được mã hóa Huffman. Thông tin để giải mã được cung cấp từ trường Side Information.

Trường Ancillary Data:

Ancillary Data là các thông tin hỗ trợ và độ dài của trường này không bị giới hạn. Nếu được sử dụng, Ancillary Data có thể chứa thông tin về vị trí của trường *main_data_begin* (trong Side Information) của frame tiếp theo.

1.1.4. Sơ lược giải mã file MP3

Quá trình giải mã và tái tạo tín hiệu analog từ file MP3 tương đối phức tạp. Sơ lược quá trình này được mô tả trong Hình 1.7.



Hình 1.7 Giải mã frame MP3 [1]

Từ dòng bit đầu vào, bộ giải mã cần xác định được điểm bắt đầu của frame (dựa vào 12 bit Sync), sau đó giải mã hệ số tỷ lệ (*scalefactor*) và dữ liệu được mã hóa Huffman trong *Main Data* dựa vào thông tin có trong từ *Side Information*, và tiến hành nghịch đảo lượng tử hóa dữ liệu vừa giải mã được. Dữ liệu sau nghịch đảo lượng

tử hóa là dữ liệu trong miền tần số (*frequency-domain*), cần được sắp xếp lại. Dữ liệu sau sắp xếp được chuyển đổi sang miền thời gian bằng biến đổi nghịch đảo Cosine rời rạc được điều chỉnh (Inverse Modified Discrete Cosine Transform - IMDCT), khử chồng phỏ (*aliasing*), và được tổng hợp lại thành tín hiệu PCM ban đầu. Tín hiệu PCM được giải điều chế trở thành tín hiệu analog.

1.2. Chuẩn MP4

1.2.1. Tổng quan

MP4 là tên viết tắt thường dùng của MPEG-4 Part 14, là một phần của chuẩn MPEG-4. MPEG-4 cùng với MPEG-1 (mục 1.1.1) và MPEG-2 tạo thành **Nhóm tiêu chuẩn MPEG**. MP4 có thể được sử dụng để nén audio, video, và kết hợp cả hai. Ngoài ra, MP4 còn có thể lưu trữ các loại dữ liệu khác (ví dụ như phụ đề).

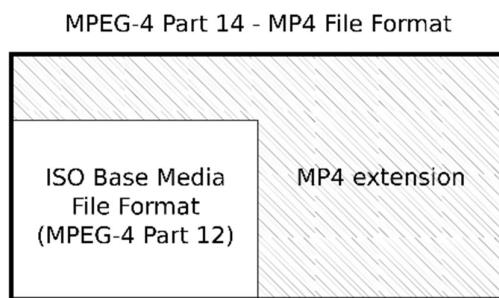
Tính đến hiện tại, MP4 có 2 phiên bản. Phiên bản đầu tiên – MPEG-4 Part 12 - được ISO giới thiệu vào năm 2001, có nhiều điểm tương đồng với định dạng QuickTime (.mov). Phiên bản 2, cũng là phiên bản được nhắc đến trong luận văn – MPEG-4 Part 14 – được giới thiệu vào năm 2003.

Điểm nổi bật của MP4 so với các định dạng nén khác là tỉ lệ nén rất cao, có thể lên đến 200 lần. Tỷ lệ nén cao đồng nghĩa với tiết kiệm không gian lưu trữ và cả đường truyền. Nhưng ở tỷ lệ này, chất lượng video sẽ bị giảm xuống quá thấp, rất khó để người dùng có thể xem được. Tỷ lệ thường được sử dụng là 50 lần.

MP4 được ISO chuẩn hóa chính thức với mã hiệu **ISO/IEC 14496-14:2003**.

1.2.2. Sơ lược cấu trúc file MP4

Hình 1.8 mô tả sơ lược cấu trúc file MP4:



Hình 1.8 Cấu trúc file MP4 [2]

Một file MP4 được cấu tạo từ 2 phần [2]:

ISO Base Media File Format (MPEG-4 Part 12): MPEG-4 Part 12 cũng là một phần của chuẩn MPEG-4. Part 12 định nghĩa một kiến trúc cơ bản cho các nội dung đa phương tiện có yêu tố thời lượng (ví dụ audio, video). Nó được thiết kế để trở thành một định dạng linh hoạt, có thể dễ dàng mở rộng, quản lý, chỉnh sửa và phát nội dung. Nội dung có thể được phát trực tiếp từ bộ nhớ hoặc qua mạng dưới dạng stream. Định dạng của part 12 không bị phụ thuộc vào giao thức mạng.[2]

MP4 Extension: Phần mở rộng đặc thù của MP4. Ngoài các dữ liệu audio và video được lưu trong Part 12, các thông tin bổ sung tùy chọn như phụ đề, mô tả bối cảnh, khung hình tĩnh có thể được lưu trong phần mở rộng này.

1.3. Kết luận chương

Chương 1 trình bày lý thuyết cơ bản về cấu trúc và giải mã file MP3, và cấu trúc file MP4. Các lý thuyết được áp dụng vào phát triển ứng dụng phát audio và video hỗ trợ hai chuẩn này.

Chương 2. Giới thiệu về Qt Framework và mô hình MVC

Chương 2 trình bày về Qt Framework (Qt), là bộ khung phần mềm (Framework) được em sử dụng để phát triển đề tài của mình, và mô thiết kế phần mềm MVC, là mô hình được em áp dụng vào thiết kế đề tài. Nội dung của Chương 1, về Qt, bao gồm giới thiệu tổng quan về Qt, giới thiệu về ngôn ngữ mô tả GUI Qt Modelling Language (QML), và các module của Qt được em sử dụng trong phát triển đề tài. Tiếp theo, nội dung về mô hình MVC bao gồm tổng quan về mô hình, định nghĩa, nhiệm vụ của từng khôi trong mô hình và tương tác giữa các khôi.

2.1. Giới thiệu về Qt Framework

Qt Framework là một framework được xây dựng nhằm hỗ trợ việc phát triển ứng dụng đa nền tảng bao gồm desktop, embedded và mobile. Hầu hết các nền tảng phổ biến đều hỗ trợ Qt, bao gồm các hệ điều hành thông thường (Standard OS), các hệ điều hành thời gian thực (RTOS), các hệ điều hành di động (Mobile OS):

- Standard OS: Linux và các biến thể (Ubuntu, CentOS, Linux Mint,...), OS X, Windows
- RTOS: VxWorks, QNX
- Mobile OS: Android, iOS, BlackBerry, SailfishOS

Qt được xây dựng trên ngôn ngữ lập trình C++. Nhiều tính năng được Qt thêm vào so với ngôn ngữ C++ cơ bản (ví dụ *signal and slots*) có thể được sử dụng qua các từ khóa mô tả (vd: *signal*, *slots*, *Q_OBJECT*, *Q_PROPERTY*,...). Trước khi mã nguồn được compile, các từ khóa này sẽ được xử lý bởi công cụ *Meta-Object Compiler* của Qt, để tạo ra các file mã nguồn tương thích với cú pháp của C++ cơ bản. Các file này sau đó có thể được compile bằng compiler C++ thông dụng như Clang, GCC, ICC, MinGW, hay MSVC.

Bản thân Qt là một framework rất mạnh. Qt cung cấp rất nhiều module trải rộng trên nhiều mảng khác nhau, có thể liệt kê sơ qua một vài lĩnh vực như: Lập trình mạng, Quản lý cơ sở dữ liệu, Lập trình đồ họa với OpenGL, Lập trình đa phương tiện (chơi nhạc, video, ...), Lập trình GUI với Widgets và QML, Lập trình web, ...

Ở thời điểm hiện tại, có rất nhiều dịch vụ, ứng dụng nổi tiếng được xây dựng trên Qt [3], có thể kể đến như:

- Hệ thống thông tin - giải trí trong xe hơi của hãng Mercedes-Benz
- Hệ điều hành webOS (chủ yếu trên SmartTV) của LG

- AMD Radeon Software, phần mềm điều khiển phần cứng đồ họa của hãng AMD
- Hệ điều hành Ubuntu

Ngoài ra, Qt được cung cấp qua cả hai phiên bản mã nguồn mở và giấy phép thương mại. Với đề tài đồ án tốt nghiệp của em, phiên bản mã nguồn mở là một lựa chọn phù hợp. Với lợi thế mã nguồn mở, Qt có lượng lập trình viên sử dụng rất lớn, dễ dàng tìm kiếm giải pháp cho các vấn đề thường gặp.

Với các ưu điểm của Qt: **đa nền tảng, hỗ trợ rộng, mã nguồn mở, dễ tìm kiếm giải pháp**, em quyết định sử dụng Qt Framework phiên bản mã nguồn mở để phát triển đề tài đồ án tốt nghiệp của mình. Tiếp sau đây là chi tiết về ngôn ngữ QML và các module, class của Qt được em sử dụng trong đề tài của mình.

2.2. Cơ chế Signal – Slot trong Qt

Signal (tín hiệu) trong Qt được mô tả dưới dạng 1 method (phương thức) của một class (lớp) mà không có phần thân hàm. Signal luôn có kiểu trả về là void. Một signal có thể mang theo nhiều kiểu dữ liệu khác nhau, được liệt kê qua các tham số của method.

Slot (khe xử lý) trong Qt là một method của đối tượng, nhưng nó được kết nối với một signal. Slot - method này sẽ được tự động thực thi khi signal mà nó kết nối với được phát đi. Slot có thể nhận nhiều dữ liệu đầu vào khác nhau, được liệt kê qua các tham số của method. Vì bản thân slot là một method bình thường, nó có thể được gọi trực tiếp.

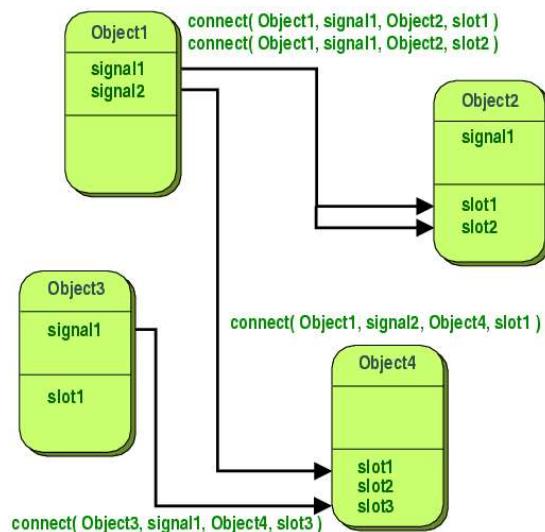
Signal và slot là cơ chế / tính năng quan trọng bậc nhất, làm nổi bật Qt với các framework khác, và được cung cấp bởi module Qt Core. Gần như toàn bộ các sản phẩm được phát triển trên Qt Framework đều phụ thuộc vào signal - slot. Signals - slots được sử dụng cho việc giao tiếp, truyền tin, truyền dữ liệu giữa các đối tượng. Trong đó, một đối tượng có thể phát ra tín hiệu (signal) khi có một sự kiện xảy ra (có thể là một thuộc tính được thay đổi, một tác vụ được hoàn thành,...). Một hoặc nhiều đối tượng khác đã được đăng ký “nghe” tín hiệu này, sẽ tự động thực hiện method tương ứng (slot).

Cơ chế tự động thực thi nói trên đóng vai trò vô cùng quan trọng trong lập trình giao diện người dùng. Nhờ có cơ chế này, các sự kiện, ví dụ người dùng nhấn vào một nút bấm, có thể được xử lý bất đồng bộ, nghĩa là giao diện chỉ gửi đi signal

sau đó thoát khỏi hàm và một đối tượng khác, ở luồng xử lý khác, sẽ thực hiện công việc tương ứng, sau đó trả kết quả về để cập nhật giao diện. Như vậy luồng xử lý về giao diện sẽ không bị đình trệ và ngừng phản ứng, đảm bảo cho trải nghiệm người dùng tốt.

Mặc dù trải nghiệm người dùng tốt hay không vẫn phụ thuộc rất nhiều vào khả năng thiết kế, thực thi ứng dụng của nhà phát triển, nhưng không thể phủ nhận signals - slots chính là bước đệm quan trọng giúp điều này trở nên khả thi.

Một signal và một slot đều điều kiện kết nối với nhau nếu các tham số đầu vào của slot trùng cả về kiểu dữ liệu và thứ tự, hoặc trùng với phần đầu của các tham số của signal.



Hình 2.1 Cơ chế signal - slot của Qt [4]

Hình 2.1 mô tả cơ chế kết nối của signal và slot. Việc kết nối được thực hiện qua method tĩnh `connect()` của class `QObject`.

Như mô tả trong Hình 2.1, một signal có thể được kết nối đến nhiều slot (ví dụ signal1 của Object1 có thể kết nối cùng lúc với slot1 và slot2 của Object2, khi signal1 được phát đi, cả 2 method slot1 và slot2 sẽ lần lượt được tự động thực thi, tùy xem method nào được kết nối trước).

Ngoài ra, một slot cũng có thể “lắng nghe” nhiều signal. Khi có bất kì signal nào trong số đó được phát đi, slot cũng sẽ được thực thi.

Kết nối giữa signal và slot không bị giới hạn bởi đối tượng. Signal của đối tượng bất kì có thể kết nối với slot của đối tượng bất kì, miễn sao thỏa mãn điều kiện về tham số.

2.1. Hệ thống thuộc tính trong Qt

Qt cung cấp hệ thống thuộc tính phong phú qua các cú pháp, từ khóa đặc biệt. Nhưng nhờ có công cụ *Meta-Object Compiler*, các cú pháp, từ khóa này được compile thành các file tương thích với chuẩn của ngôn ngữ C++ cơ bản, đảm bảo rằng Qt không phụ thuộc vào compiler.

Hệ thống thuộc tính có chức năng tiêu biểu nhất là cho phép thành phần giao diện có thể theo dõi và tự động cập nhật hiển thị theo giá trị của thuộc tính mà một class cung cấp. Thuộc tính này có thể là trạng thái của một nút bấm (không nhấn, đang nhấn,...), hay trạng thái chơi nhạc (đang chơi nhạc, đã dừng chơi nhạc, đang bật chế độ trộn bài, ...). Khi giá trị các thuộc tính này thay đổi bởi đối tượng quản lý nó, thành phần giao diện có thể tự động cập nhật theo, giúp lập trình viên tiết kiệm thời gian cập nhật một cách thủ công. Vừa tiết kiệm thời gian phát triển cũng như bảo dưỡng, vừa hạn chế lỗi, thiếu sót có thể xảy ra.

Ví dụ, trong ứng dụng có chức năng chơi nhạc, ở màn hình của chức năng này có thể có một nút bấm dùng để dừng/tiếp tục chơi nhạc. Biểu tượng hiển thị của bút bấm này phụ thuộc vào trạng thái phát: nếu đang chơi nhạc thì hiển thị biểu tượng tạm dừng, ngược lại nếu đang dừng thì hiển thị biểu tượng bắt đầu. Giả sử có tới 10 trường hợp khác nhau dẫn đến việc dừng chơi nhạc (người dùng bấm nút, xe chuyển sang trạng thái di chuyển, rút thiết bị nhớ, file bị lỗi, ...), nếu làm thủ công, lập trình viên ngoài việc đương nhiên phải tính toán đủ 10 trường hợp ở phần mã nguồn logic, còn phải thực hiện việc cập nhật tương tự phía mã nguồn giao diện, mà vẫn có thể có sai sót. Sai sót có thể kể đến như bấm nút bắt đầu chơi nhạc, lúc này nút bấm được cập nhật thủ công sang biểu tượng tạm dừng, nhưng thực tế có lỗi xảy ra dẫn đến không thể chơi nhạc, nghĩa là trạng thái hiện tại của nút bấm là không chính xác. Việc sử dụng thuộc tính để nút bấm tự động cập nhật vừa giảm thiểu phần việc phải cập nhật thủ công phía giao diện, đồng thời giảm cả khả năng xảy ra hiển thị sai trên giao diện. Giao diện khi đó sẽ không cần phải xử lý logic, chính là một trong các thuộc tính quan trọng trong thiết kế phần mềm có giao diện.

2.2. Giới thiệu về ngôn ngữ QML

QML là viết tắt của Qt Modelling Language, là ngôn ngữ dùng để mô tả (Modelling) giao diện trong Qt, cả về hiển thị và hành vi.

Về mô tả hiển thị, giao diện xây dựng bởi QML được cấu thành từ các thành phần (component), mỗi component có thể coi là một đối tượng. Một giao diện được mô tả bởi QML là một hệ thống component sắp xếp dạng cây, mỗi component có thể có các thuộc tính và các component con.

Về mô tả hành vi, QML sử dụng ngôn ngữ Javascript.

Một ứng dụng Qt có thể có phần logic được phát triển trên Qt Framework bằng ngôn ngữ C++, và phần giao diện được mô tả bằng ngôn ngữ QML. Qt hỗ trợ người phát triển ứng dụng liên kết hai thành phần này với nhau thông qua class QQmlApplicationEngine. QQmlApplicationEngine cho phép người phát triển đăng ký một đối tượng C++ với QML, nhờ đó người phát triển có thể truy xuất các thuộc tính, thực thi các method, kết nối signal từ QML với slot của đối tượng C++.

QML cung cấp cú pháp rất trực quan, dễ đọc hiểu, có nhiều nét tương đồng với CSS. Để mô tả một component bằng QML, chỉ cần thiết lập các thuộc tính theo cú pháp “thuộc tính: giá trị”. Ví dụ: “color: green” nghĩa là màu sắc (color) của component này là xanh lá (green).

QML đồng thời cung cấp một lượng lớn các component có sẵn, từ cơ bản: *Rectangle* - hình chữ nhật, *Text* - đoạn văn bản, *Button* - nút bấm,... hay hiển thị với logic đơn giản: *ScrollBar* - thanh cuộn, *ProgressBar* - thanh tiến độ, *Slider* - thanh trượt,... tới phức tạp với nhiều chức năng (cả hiển thị và hành vi): *Video* - chức năng phát và điều khiển video, *Radio* - chức năng nghe và điều khiển radio, *Map* - hiển thị và tương tác với bản đồ,...

Mỗi component khi được sử dụng đều được coi là một đối tượng, có các thuộc tính, signal, slot. Một số thuộc tính tiêu biểu, tồn tại ở hầu hết các component có thể kể đến như:

- width: chiều rộng của component, đơn vị pixel
- height: chiều cao của component, đơn vị pixel
- x: tọa độ trên trục nằm ngang của component, đơn vị pixel
- y: tọa độ trên trục nằm dọc của component, đơn vị pixel,
- z: tọa độ trên trục vuông góc với mặt phẳng hiển thị, component có tọa độ z cao hơn sẽ hiển thị đè lên component có tọa độ z thấp hơn

Một số signal cơ bản, được hỗ trợ trong hầu hết các component có thể kể đến như:

- *completed()*: component đã được khởi tạo
- *destruction()*: component bắt đầu được xóa bỏ

Ngoài các component có sẵn, người phát triển cũng có thể tự tạo ra các component đặc thù, tự khai báo, định nghĩa các thuộc tính, signal, slot, cấu trúc, hành vi, phục vụ mục đích chuyên dụng ứng với từng yêu cầu cụ thể. Component mới có thể được xây dựng từ các component có sẵn.

2.3. Module Qt Multimedia

Qt Multimedia là module hướng đến mảng đa phương tiện. Qt Multimedia cung cấp nhiều C++ class và kiểu QML chuyên dụng cho xử lý nội dung đa phương tiện (ảnh, nhạc, video, radio, ...) Với các giao diện lập trình ứng dụng (API) được cung cấp từ Qt Multimedia, người sử dụng có thể xây dựng các tính năng:

- Truy cập và điều khiển thiết bị âm thanh đang được kết nối
- Phát các hiệu ứng âm thanh với độ trễ thấp
- Tạo danh sách file audio, video
- Phát và điều khiển phát lại file audio, video trong một danh sách
- Ghi âm và nén file theo các định dạng phổ thông (mp3, aac, ...)
- Nghe và điều chỉnh tần số radio
- Truy cập vào camera của thiết bị, sử dụng các tính năng chụp ảnh, quay video
- Phát và điều chỉnh âm thanh 3D khi thiết bị có hỗ trợ
- Giải mã file audio, video từ các định dạng phổ thông (mp3, aac, mp4, v.v...) và đưa vào bộ nhớ để xử lý
- Truy cập và lấy ra từng khung hình trong video

Trong khuôn khổ của đề tài đồ án, em sử dụng các tính năng: giải mã và phát file audio, video, tạo danh sách phát, phát và điều khiển phát lại.

2.4. Mô hình thiết kế phần mềm MVC

2.4.1. Tổng quan

Model-View-Controller, viết tắt là MVC, là một mô hình thiết kế phần mềm phổ biến trong lập trình hướng đối tượng, đặc biệt là trong thiết kế phần mềm có giao diện hay thiết kế website.

Mục đích chính của MVC là để tách biệt các phần trong ứng dụng nhiều nhất có thể, giúp cho nhà phát triển ứng dụng có thể hiểu và phát triển, chỉnh sửa một phần mà không cần phải hiểu rõ các phần khác của ứng dụng.

“Isolating functional units from each other as much as possible makes it easier for the application designer to understand and modify each particular unit without having to know everything about the other units.” [5]

Một ứng dụng thiết kế theo mô hình MVC được chia thành 3 phần chính:

- *Model*: có nhiệm vụ tạo và xử lý toàn bộ dữ liệu

- *View*: có nhiệm vụ hiển thị dữ liệu lấy được từ Model cho người dùng
- *Controller*: có nhiệm vụ xử lý thao tác của người dùng. [5]

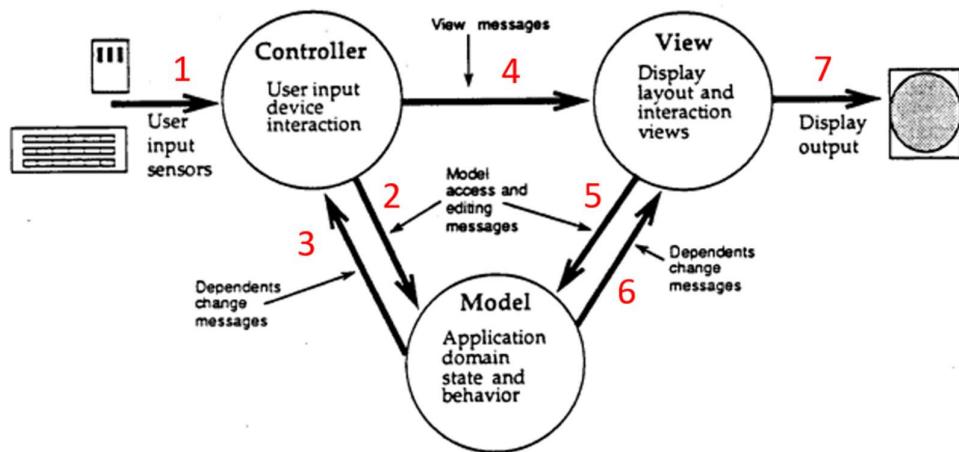
Quy trình phát triển ứng dụng MVC chia trách nhiệm của những người tham gia thành 3 vai trò để tăng tính hiệu quả của công việc, bao gồm:

- *Development*: Vai trò phát triển, thường là lập trình viên, là người triển khai logic cho ứng dụng. Công việc có thể là truy xuất dữ liệu, đóng gói và chuẩn bị dữ liệu, kiểm tra tính đúng đắn của dữ liệu,...
- *Design*: Vai trò thiết kế, là người chịu trách nhiệm về giao diện ứng dụng. Họ thực hiện việc hiển thị dữ liệu được cung cấp từ các nhà phát triển làm việc trong vai trò Development.
- *Integration*: Vai trò tích hợp, là người có trách nhiệm gắn kết công việc của hai vai trò trước đó. [6]

Tương tác của người dùng với ứng dụng MVC tuân theo một chu trình tự nhiên: người dùng thực hiện một hành động, và để phản hồi lại, ứng dụng sẽ thay đổi dữ liệu và cập nhật thông tin được cho người dùng. Chu trình này được lặp đi lặp lại trong ứng dụng.

“User interaction with an MVC application follows a natural cycle: the user takes an action, and in response the application changes its data model and delivers an updated view to the user. And then the cycle repeats.” [7]

Figure 1. Model-View-Controller State and Message Sending



Hình 2.2 Tương tác giữa các khối trong mô hình MVC [5]

Hình 1.2. mô tả tương tác giữa các khối trong mô hình MVC, tuân theo chu trình vừa mô tả ở trên:

- 1: Người dùng (*User*) tương tác với *Controller*
- 2: *Controller* gửi yêu cầu truy cập và chỉnh sửa dữ liệu đến *Model*
- 3, 6: *Model* truy cập, chỉnh sửa dữ liệu, và thông báo dữ liệu đã được cập nhật đến *Controller* và *View*
- 4: *Controller* dựa vào sự thay đổi của dữ liệu, sẽ gửi yêu cầu thay đổi cách hiển thị dữ liệu đến *View*
- 5: *View* nhận được thông báo thay đổi và yêu cầu hiển thị mới, sẽ gửi yêu cầu truy cập dữ liệu mới đến *Model*
- 6: *Model* trả về dữ liệu mới được cập nhật đến *View*
- 7: *View* hiển thị dữ liệu mới

2.4.2. Khối Model

Model là phần quản lý tất cả các tác vụ liên quan đến dữ liệu trong hệ thống: cung cấp dữ liệu, kiểm tra tính đúng đắn của dữ liệu, quản lý trạng thái, quyền truy cập dữ liệu, triển khai cấu trúc cơ sở dữ liệu. Model giúp người phát triển giảm đáng kể độ phức tạp của mã nguồn. [8]

Model chịu trách nhiệm xử lý logic và nghiệp vụ của một ứng dụng. Model sẽ đóng gói các phương thức để truy cập dữ liệu (cơ sở dữ liệu, tệp, v.v.) và sẽ cung cấp các phương thức này cho các phần cần sử dụng. Ngoài ra, Model còn có nhiệm vụ xác nhận, kiểm tra tính đúng đắn của dữ liệu và xác thực quyền truy cập dữ liệu của một yêu cầu từ khối khác. [7]

2.4.3. Khối View

View xử lý các nhiệm vụ liên quan đến hiển thị: yêu cầu truy cập dữ liệu trong Model và hiển thị dữ liệu đó cho người dùng. View không chỉ bao gồm các component chuyên dụng cho việc hiển thị, mà còn bao gồm cả các phép biến đổi đồ họa (animation), ví dụ phóng to, thu nhỏ, trượt sang trái, ... [4]

Các component trong View có thể là các nút bấm, các nhãn có nội dung, màu sắc, thanh trượt, hộp thoại, ...

2.4.4. Khối Controller

Controller có nhiệm vụ nhận sự kiện, ví dụ một tương tác từ người dùng như một thao tác nhấn chuột, một phím bấm được gõ. Controller nhận diện tương tác và xác định cách xử lý tương ứng, từ đó gửi yêu cầu cập nhật dữ liệu đến Model. Ví dụ khi người dùng nhấn chuột, Controller sẽ nhận định xem người dùng có nhấn vào nút

xóa hay không. Nếu xác định được người dùng nhấn chuột vào nút xóa, Controller gửi yêu cầu xóa đến Model, và Model sẽ thực hiện xóa dữ liệu.

2.5. Kết luận chương

Chương 2 trình bày các kiến thức em tìm hiểu được về Qt và mô hình MVC, từ đó sử dụng làm nền tảng cho quá trình phân tích, thiết kế và phát triển sản phẩm cho đề tài, được trình bày trong Chương 3.

Chương 3. Thiết kế đề tài

Chương 3 trình bày về quá trình phân tích, thiết kế và phát triển đề tài của em. Các bước trong quá trình thực hiện thiết kế đề tài bao gồm: Phân tích yêu cầu chức năng của ứng dụng, Thiết kế kiến trúc của ứng dụng với mô hình MVC, đặc tả các chức năng của ứng dụng, và mô tả bố cục giao diện người dùng của ứng dụng.

Ứng dụng được phát triển và kiểm thử trên môi trường Microsoft Windows, sử dụng công cụ Qt Creator làm môi trường phát triển tích hợp (IDE) vì Qt Creator là công cụ hỗ trợ Qt Framework tốt nhất, được chính Qt phát hành.

3.1. Phân tích yêu cầu hệ thống

3.1.1. Yêu cầu chức năng cho chức năng phát audio

a) Lập và hiển thị danh sách phát

Ứng dụng có thể tự động tìm và lập danh sách toàn bộ các file có định dạng MP3 trong bộ nhớ.

Ứng dụng có thể hiển thị danh sách các nội dung được tạo ra từ các file MP3, thông tin được hiển thị của một nội dung trong danh sách phát bao gồm:

- Tên bài hát / Tên file mp3
- Nghệ sĩ thể hiện

b) Hiển thị meta-data

Ứng dụng có thể hiển thị các dữ liệu mô tả (meta-data) đi kèm với nội dung đang phát (nếu có). Các dữ liệu này bao gồm: Tên bài hát, Nghệ sĩ thể hiện và Hình album.

Khi thông tin tên bài hát không có sẵn trong meta-data, thông tin này được thay thế bằng tên file.

Khi thông tin nghệ sĩ thể hiện không có sẵn trong meta-data, thông tin này được thay thế bằng “Unknown Artist”.

Khi thông tin hình ảnh album không có sẵn trong meta-data, thông tin này được thay thế bằng hình ảnh mặc định.

c) Hiển thị thời lượng

Ứng dụng có thể hiển thị tổng thời lượng của file âm thanh đang phát, theo định dạng hh:mm:ss (giờ:phút:giây).

Ứng dụng có thể hiển thị thời lượng đã phát của file âm thanh, theo định dạng hh:mm:ss (giờ:phút:giây).

Ứng dụng có thể hiển thị tỷ lệ giữa thời lượng đã phát so với tổng thời lượng của file âm thanh đang phát.

d) Điều khiển phát qua giao diện đồ họa

Ứng dụng cung cấp khả năng điều khiển phát cho người dùng qua giao diện đồ họa, bao gồm:

- Chọn nội dung phát từ danh sách
- Điều khiển dừng / phát
- Chọn vị trí phát bất kì.
- Chuyển sang nội dung tiếp theo.
- Chuyển về nội dung trước đó.
- Phát lại từ đầu nội dung đang phát.

Khi nội dung đã phát nhưng thời lượng ít hơn 3 giây, khi người dùng thực hiện thao tác chuyển về nội dung trước, nội dung trước đó được phát.

Trong trường hợp nội dung đã phát từ 3 giây trở lên, khi người dùng thực hiện thao tác chuyển về nội dung trước, nội dung hiện tại sẽ được phát lại từ đầu.

Khi một nội dung âm thanh kết thúc, tự động chuyển sang nội dung tiếp theo trong danh sách ngay lập tức.

e) Chọn chế độ phát lặp lại qua giao diện đồ họa

Các chế độ phát lặp lại bao gồm:

- Không lặp lại: Phát lần lượt từ nội dung được chọn đến nội dung cuối cùng trong danh sách phát. Sau khi nội dung cuối cùng kết thúc thì dừng phát.
- Lặp lại toàn bộ: Phát lần lượt từ nội dung được chọn đến nội dung cuối cùng trong danh sách phát. Sau khi nội dung cuối cùng kết thúc, phát lại từ nội dung đầu.
- Lặp lại một bài: Phát lại nội dung đang phát sau khi kết thúc.

Khi chế độ lặp lại đang được áp dụng là “Lặp lại một bài”, nếu nội dung đang phát đã phát từ ít hơn 3 giây và người dùng thực hiện chuyển về nội dung trước đó, chế độ lặp lại sẽ được thay đổi thành “Lặp lại toàn bộ”.

Khi chế độ lặp lại đang được áp dụng là “Lặp lại một bài”, nếu người dùng thực hiện chuyển đến nội dung tiếp theo, chế độ lặp lại sẽ được thay đổi thành “Lặp lại toàn bộ”.

Khi chế độ lặp lại đang được áp dụng là “Lặp lại một bài”, nếu người dùng thực hiện chọn phát một nội dung từ danh sách phát, chế độ lặp lại sẽ được thay đổi thành “Lặp lại toàn bộ”

Chế độ lặp lại được thay đổi lần lượt theo vòng khi có yêu cầu thay đổi từ người dùng: Không lặp lại => Lặp lại toàn bộ => Lặp lại một bài => Không lặp lại

f) Lưu thông tin và tự động tải nội dung phát gần nhất khi khởi động

Sau khi ứng dụng tắt, ở lần tiếp theo ứng dụng được khởi động, nội dung gần nhất được phát trước đó sẽ được tải sẵn, nếu người dùng thực hiện thao tác bắt đầu phát, phát nội dung từ đầu.

3.1.2. Yêu cầu chức năng cho chức năng phát file video

a) Lập và hiển thị danh sách phát

Ứng dụng có thể tự động tìm và lập danh sách toàn bộ các file mp4

Ứng dụng có thể hiển thị danh sách các nội dung được tạo ra từ các file mp4, thông tin được hiển thị của một nội dung trong danh sách phát bao gồm:

- Tên file mp4
- Hình ảnh thu nhỏ (thumbnail)

b) Hiển thị tên và thời lượng nội dung đang phát

Ứng dụng có thể hiển thị tên của file video đang được phát.

Ứng dụng có thể hiển thị tổng thời lượng của file video, theo định dạng hh:mm:ss (giờ:phút:giây).

Ứng dụng có thể hiển thị thời lượng đã phát của file video, theo định dạng hh:mm:ss (giờ:phút:giây).

Ứng dụng có thể hiển thị tỷ lệ giữa thời lượng đã phát so với tổng thời lượng của video đang phát.

c) Cung cấp khả năng điều khiển phát qua giao diện

Ứng dụng cung cấp khả năng điều khiển phát cho người dùng, bao gồm:

- Chọn phát một nội dung từ danh sách phát
- Điều khiển dừng / phát
- Tua đến 10 giây sau
- Tua đến 10 giây trước
- Chuyển sang nội dung tiếp theo
- Chuyển về nội dung trước đó

- Chọn vị trí phát bất kì

Khi một nội dung video kết thúc, tự động chuyển sang nội dung tiếp theo sau 5 giây. Trong thời gian đếm ngược 5 giây này, người dùng có thể chọn phát lại nội dung vừa kết thúc. Nếu người dùng chọn phát lại, không chuyển sang nội dung tiếp theo, và phát lại nội dung vừa phát từ đầu.

d) Lưu và hiển thị thông tin 2 nội dung phát gần nhất

Ứng dụng có thể lưu lịch sử và hiển thị thông tin 2 video được phát gần nhất.

Với video được phát mới hơn trong 2 video được lưu lịch sử, hiển thị khung hình ứng với vị trí phát trước đó.

Với video được phát mới hơn trong 2 video được lưu lịch sử, khi người dùng chọn phát file video này, tiếp tục phát từ vị trí trước đó.

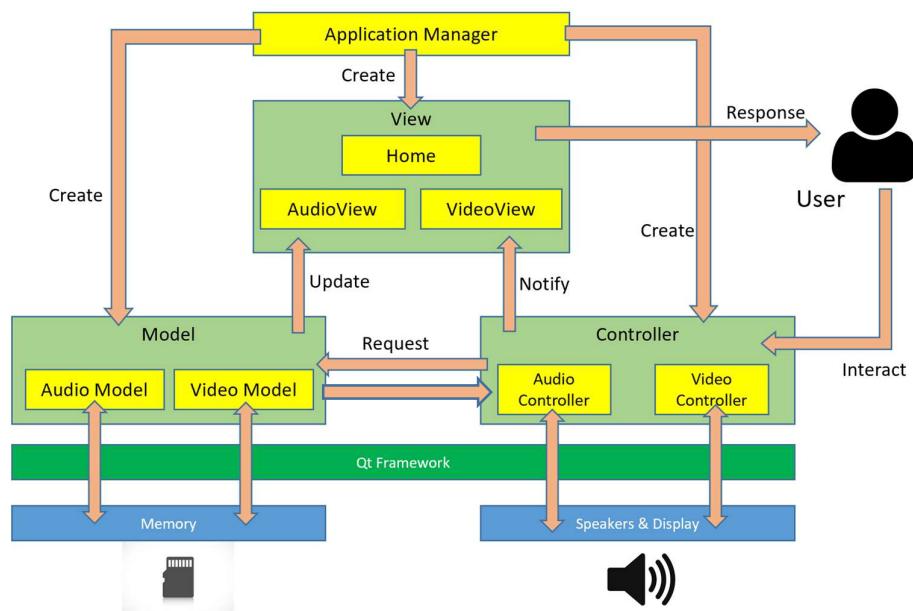
3.1.3. Yêu cầu phi chức năng

Ứng dụng có thể hiển thị thông tin giờ

3.2. Thiết kế kiến trúc của ứng dụng

3.2.1. Sơ đồ khối

Ứng dụng được thiết kế theo mô hình MVC, với các khối được sắp xếp trong Hình 2.1:



Hình 3.1 Sơ đồ khối của ứng dụng

Ngoài mô tả các khối chính: Model, View, Controller, và các khối phụ, Hình 2.1 cung cấp cái nhìn tổng quan về tương tác giữa các khối. Chi tiết về các khối và tương tác được trình bày trong các phần tiếp theo của mục 2.2.

Khối Model, Controller và ApplicationManager sẽ được xây dựng bằng ngôn ngữ lập trình C++, dựa trên Qt Framework. Khối View sẽ được xây dựng bằng ngôn ngữ QML.

3.2.2. Các khối chính

Các khối chính của ứng dụng được thiết kế theo mô hình MVC. Trong đó:

Khối **View**:

- Home: Màn hình Home
- AudioView: Các màn hình phục vụ chức năng phát file audio
- VideoView: Các màn hình phục vụ chức năng phát file video

Khối **Controller**:

- AudioController: Khối xử lý logic, cung cấp các chức năng điều khiển phát file audio
- VideoController: Khối xử lý logic, cung cấp các chức năng điều khiển phát file video

Khối **Model**:

- AudioModel: Khối xử lý tạo và quản lý danh sách phát file audio
- VideoModel: Khối xử lý tạo và quản lý danh sách phát file video

3.2.3. Các khối còn lại

Ngoài 3 khối chính, các thành phần còn lại bao gồm:

- User: Người dùng, tương tác với ứng dụng
- Application Manager: Khối quản lý, có nhiệm vụ khởi tạo và xóa bỏ các đối tượng
- Qt Framework: Các module được cung cấp sẵn từ Qt, cung cấp giao diện đơn giản phục vụ truy cập bộ nhớ và thiết bị phát âm thanh
- Memory: Bộ nhớ lưu trữ (ổ cứng, thẻ SD, USB, ...)
- Speaker: Thiết bị phát âm thanh (loa, tai nghe, ...)

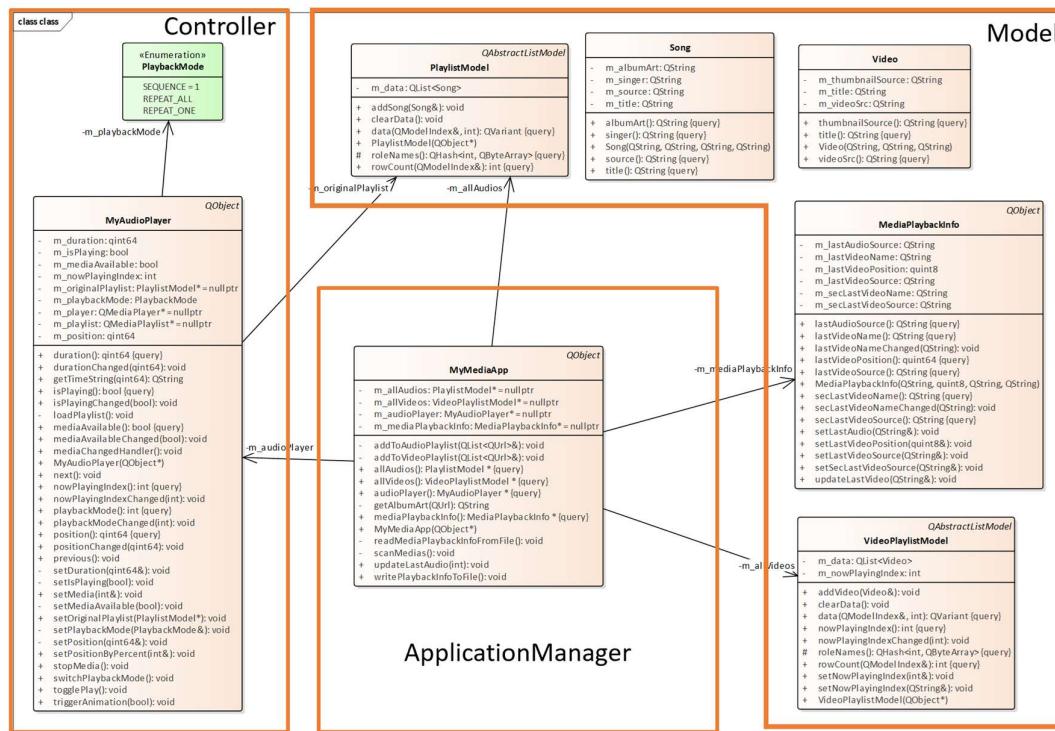
3.2.4. Tương tác giữa các khối

Tương tác giữa các khối tuân theo tương tác trong mô hình MVC, có thể mô tả đơn giản như sau:

- Application Manager khởi tạo các đối tượng
- Model truy cập vào Memory thông qua Qt Framework, tạo danh sách phát
- Model cung cấp danh sách phát để cập nhật hiển thị trên View
- User thực hiện tương tác với Controller
- Controller gửi yêu cầu truy cập nội dung đến Model
- Model gửi phản hồi đến Controller kèm theo nội dung được yêu cầu
- Controller thực hiện phát nội dung bằng Speaker thông qua Qt Framework
- Controller gửi thông báo cập nhật giao diện đến View
- Model cung cấp thông tin cập nhật giao diện đến View

3.3. Thiết kế chi tiết

3.3.1. Thiết kế các Class



Hình 3.2 Biểu đồ Class

Hình 2.2 mô tả các Class trong ứng dụng và quan hệ của chúng. Các class được tạo và sử dụng bao gồm:

Khối *ApplicationManager*:

- MyMediaApp

Khối *Model*:

- MediaPlaybackInfo
- Song
- Video
- PlaylistModel
- VideoPlaylistModel

Khối *Controller*:

- **MyAudioPlayer**
- PlaybackMode: Enum liệt kê các chế độ phát lại bao gồm:
 - SEQUENCE = 0: tương ứng với chế độ “Không lặp lại”
 - REPEAT_ALL = 1: tương ứng với chế độ “Lặp lại toàn bộ”
 - REPEAT_ONE = 2: tương ứng với chế độ “Lặp lại một bài”

Khối View không xuất hiện trong biểu đồ class vì các thành phần giao diện trong View được mô tả bằng ngôn ngữ QML, không phải class C++. Trong biểu đồ cũng không có khối cung cấp các phương thức điều khiển phát file video tương đương với **MyAudioPlayer** của audio, vì các phương thức này đã được cung cấp sẵn bởi VideoQML (1.4.4).

Tổng quan về các class như sau:

MyMediaApp là class có nhiệm vụ quản lý, khởi tạo các đối tượng trong ứng dụng. Khi khởi tạo, ngoài việc tạo ra đối tượng mới, MyMediaApp còn thực hiện truyền dữ liệu cho đối tượng, ví dụ thiết đặt danh sách phát cho MyAudioApp.

MyAudioPlayer cung cấp các phương thức điều khiển phát file audio.

MediaPlaybackInfo có nhiệm vụ lưu trữ thông tin về nội dung được phát gần nhất. Thông tin bao gồm: đường dẫn của file audio được phát gần nhất, đường dẫn đến 2 file video được phát gần nhất, thời lượng đã phát của file video được phát gần nhất.

PlaylistModel là class lưu trữ thông tin về danh sách nội dung audio

Song là class chứa thông tin của một nội dung audio.

Video là class chứa thông tin của một nội dung video.

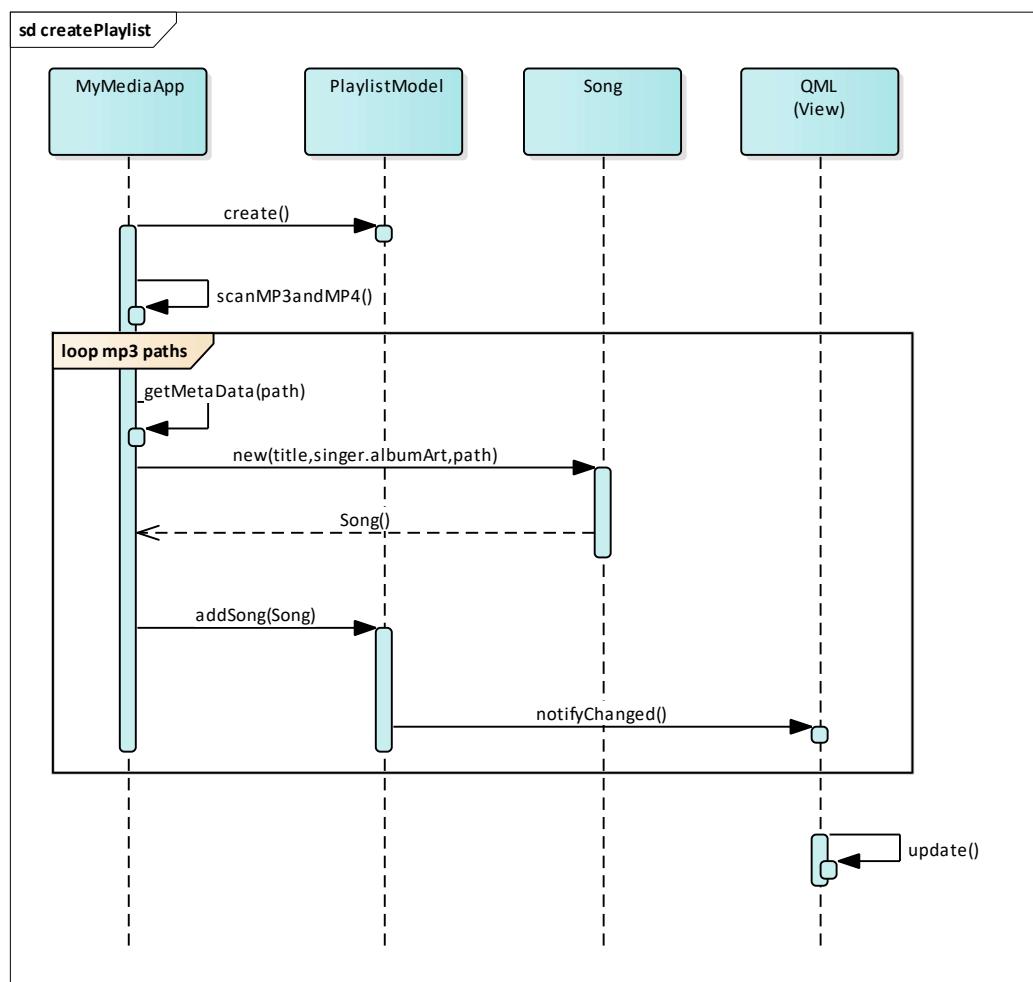
VideoPlaylistModel là class lưu trữ thông tin về danh sách nội dung video

Chi tiết các class được trình bày trong Phụ lục A.

3.3.2. ĐẶC TẢ CÁC CHỨC NĂNG PHÁT AUDIO

a) LẬP VÀ HIỂN THỊ DANH SÁCH PHÁT

Hình mô tả chức năng lập và hiển thị danh sách phát



Hình 3.3 Chức năng lập và hiển thị danh sách phát

Chức năng lập danh sách phát được thực hiện vào thời điểm khởi động ứng dụng.

Quá trình xử lý:

- MyMediaApp khởi tạo đối tượng PlaylistModel
- MyMediaApp tiến hành duyệt và lập mảng đường dẫn các file MP3
- MyMediaApp truy cập meta-data của file mp3

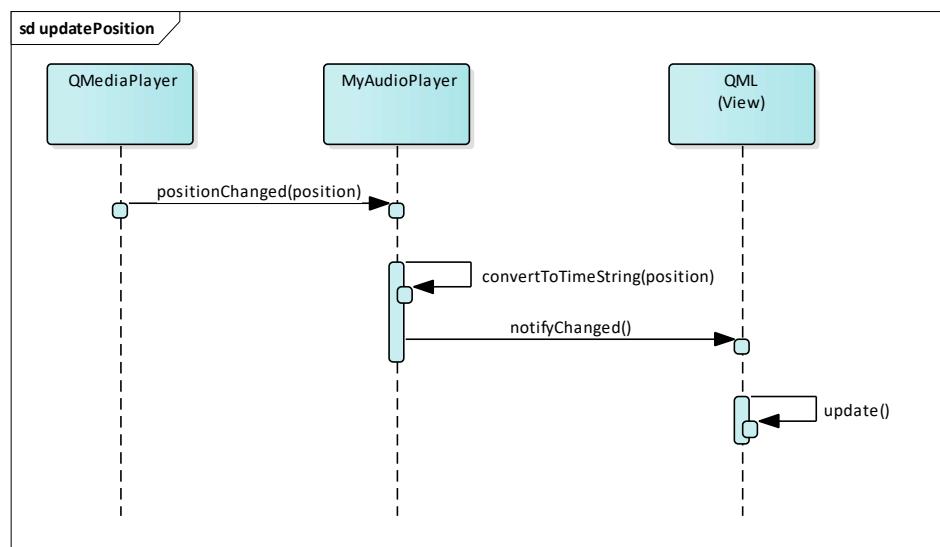
- MyMediaApp tạo đối tượng Song với các thông tin lấy được và thêm vào danh sách PlaylistModel
- Lặp lại cho tất cả phần tử trong mảng đường dẫn
- PlaylistModel gửi thông báo đến View
- View tiến hành cập nhật thông tin hiển thị trong danh sách

b) *Hiển thị meta-data*

Meta-data được cung cấp qua các đối tượng Song trong PlaylistModel, được khởi tạo khi ứng dụng khởi động. Tham khảo Hình 3.3.

c) *Hiển thị thời lượng*

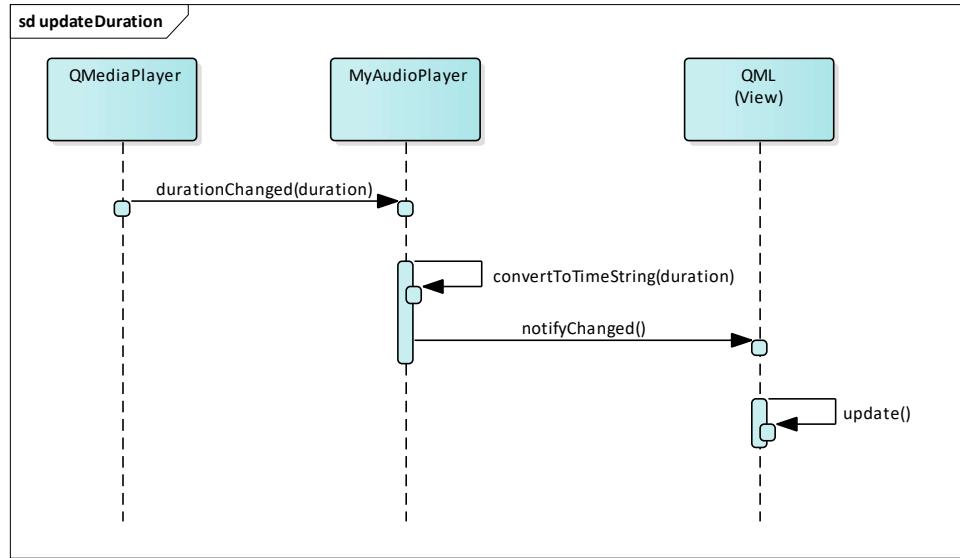
Hình 3.4 mô tả chức năng hiển thị thời lượng đã phát



Hình 3.4 Cập nhật hiển thị thời lượng đã phát

Thời lượng đã phát được cập nhật liên tục trong quá trình phát file audio. Thời lượng (đơn vị ms) được hỗ trợ tính toán bởi class QMediaPlayer của Qt Framework và được gửi bởi signal *positionChanged()*. Khi nhận được signal này, MyAudioPlayer tiến hành chuyển đổi từ đơn vị ms sang xâu có định dạng hh:mm:ss và thông báo cập nhật đến View. View sử dụng giá trị xâu vừa được cập nhật và hiển thị lên giao diện.

Hình 3.5 mô tả chức năng cập nhật hiển thị thời lượng tổng.



Hình 3.5 Cập nhật hiển thị thời lượng tổng

Thời lượng tổng được cập nhật mỗi khi thay đổi nội dung phát. Thời lượng tổng được hỗ trợ tính toán bởi class `QMediaPlayer` của Qt Framework và được cung cấp qua signal `durationChanged()`. Khi nhận được signal này, `MyAudioPlayer` tiến hành chuyển đổi từ đơn vị ms sang xâu có định dạng “hh:mm:ss” và thông báo cập nhật đến View. View sử dụng giá trị xâu vừa được cập nhật và hiển thị lên giao diện.

d) Điều khiển phát qua giao diện đồ họa

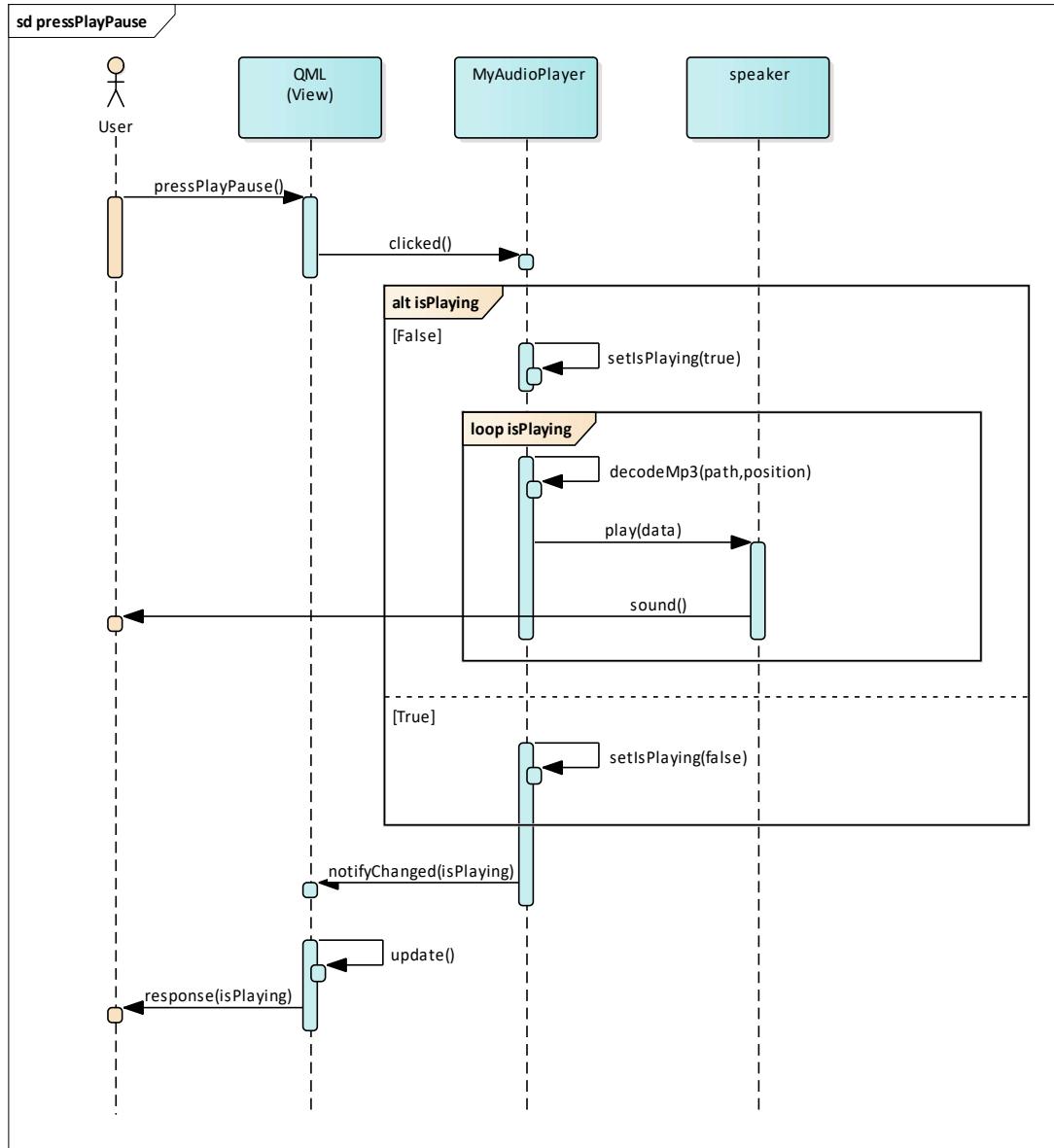
Hình 3.6 mô tả chức năng điều khiển dừng/phát qua nút bấm trên giao diện.

Khi người dùng nhấn vào nút bấm điều khiển dừng/phát, tín hiệu bấm nút được View gửi tới `MyAudioPlayer`.

Nếu trạng thái phát hiện tại là không phát, cập nhật trạng thái phát thành đang phát, và tiến hành giải mã và phát file audio đang được chọn qua thiết bị phát (tác vụ này được hỗ trợ bởi Qt Framework qua API `play()`)

Nếu trạng thái hiện tại là dừng phát, ngừng giải mã file và ngừng phát, sau đó cập nhật trạng thái phát thành không phát.

Sau khi thực hiện phát/dừng, `MyAudioPlayer` gửi thông báo cập nhật trạng thái phát tới View, View cập nhật hiển thị và phản hồi lại cho người dùng.

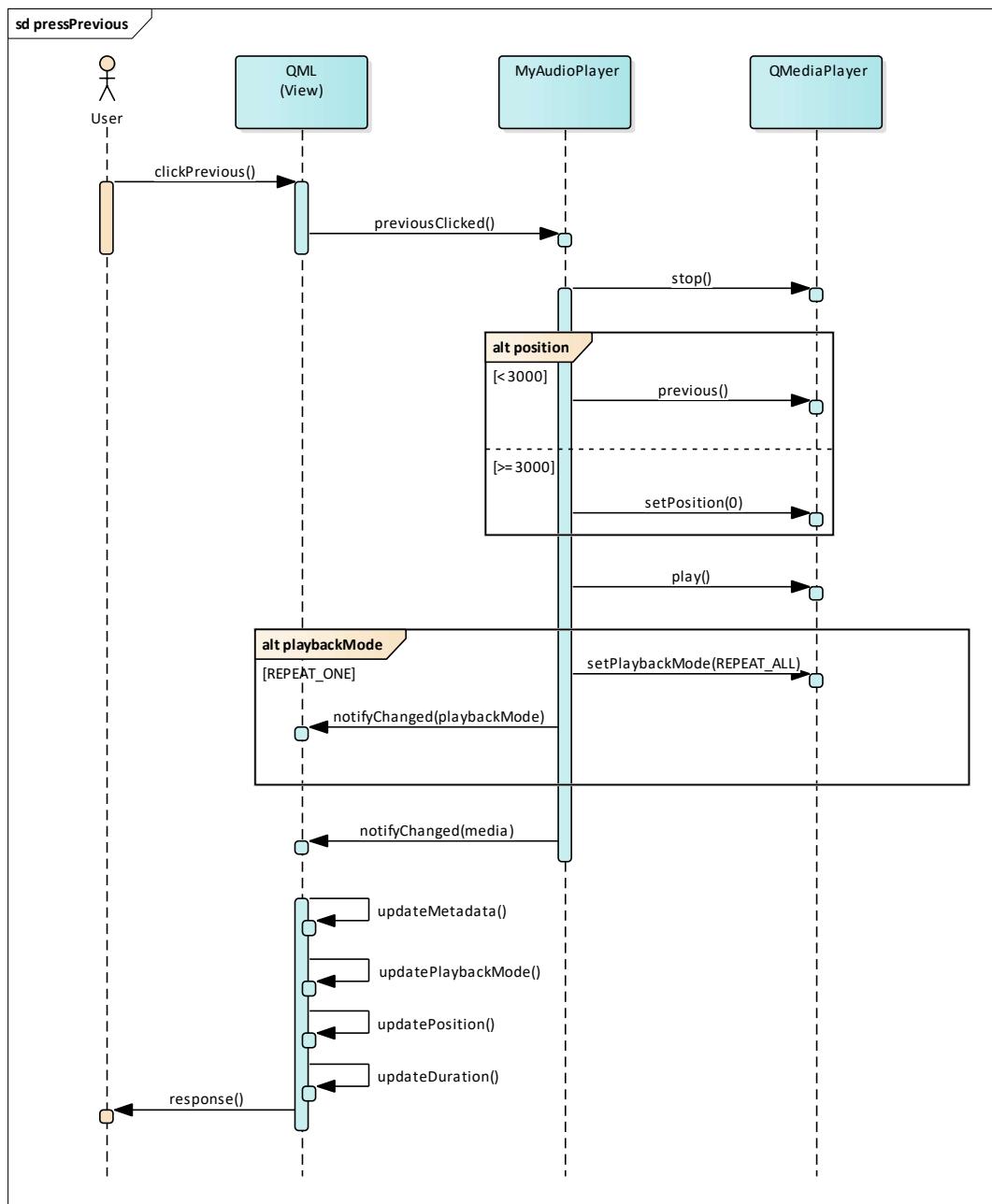


Hình 3.6 Điều khiển dừng/phát

Hình 3.7 mô tả chức năng phát lại từ đầu và chuyển về nội dung liền trước.

Khi người dùng nhấn vào nút chuyển về nội dung liền trước (previous), tiến hành dừng phát bằng API `stop()` được hỗ trợ bởi Qt

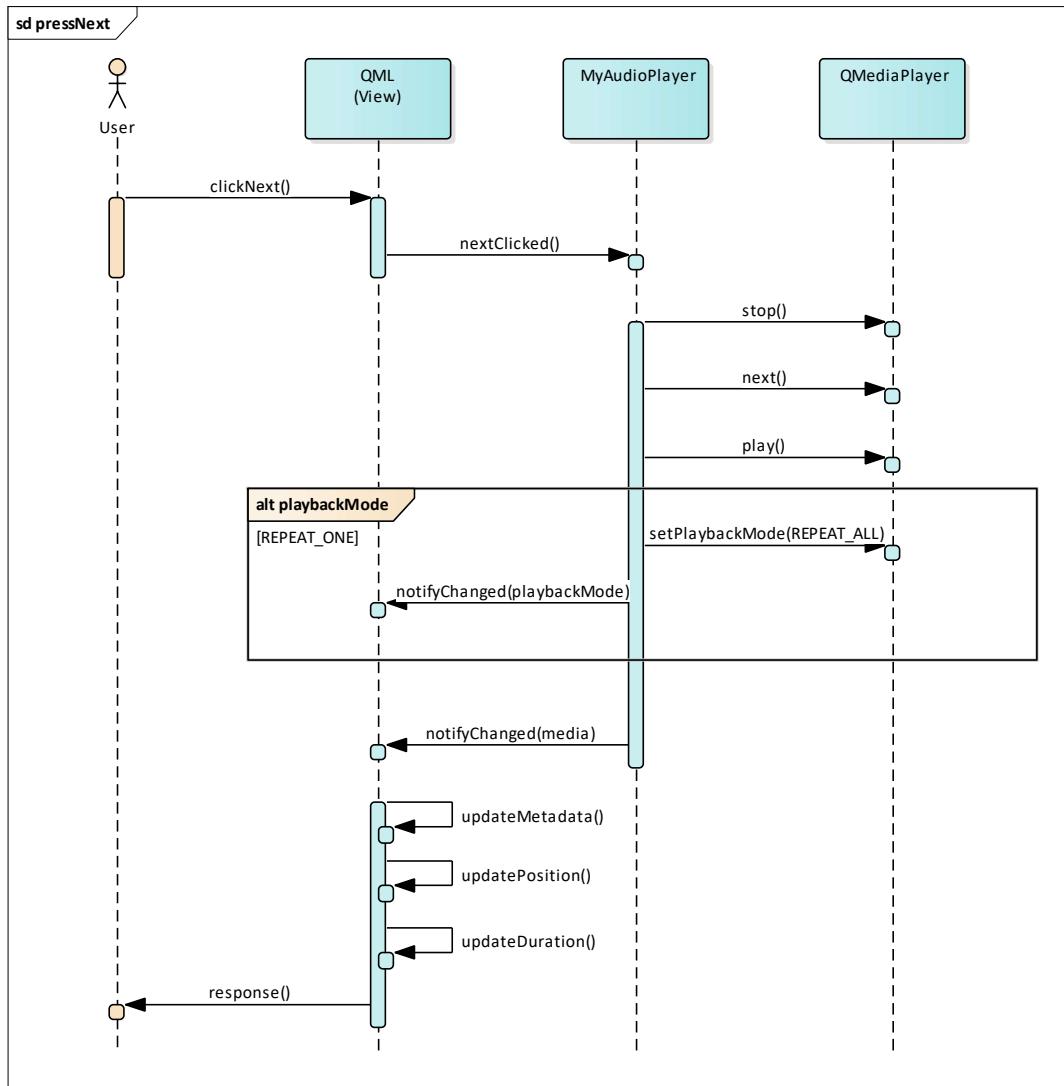
- Nếu thời lượng đã phát nhỏ hơn 3000 ms, thực hiện chuyển về bài trước đó qua API `previous()`
- Nếu thời lượng đã phát lớn hơn 3000 ms, thực hiện thiết lập vị trí phát về 0 qua API `setPosition()`



Hình 3.7 Chuyển về nội dung liền trước

Sau đó MyAudioPlayer tiếp tục phát qua API `play()` (tham khảo Hình 3.6). Nếu chế độ phát lại đang áp dụng là `REPEAT_ONE` (lặp lại 1 bài), thực hiện đổi chế độ phát lại sang `REPEAT_ALL`. MyAudioApp Gửi thông báo thay đổi đến View, View sẽ tiến hành cập nhật hiển thị để phản hồi đến người dùng.

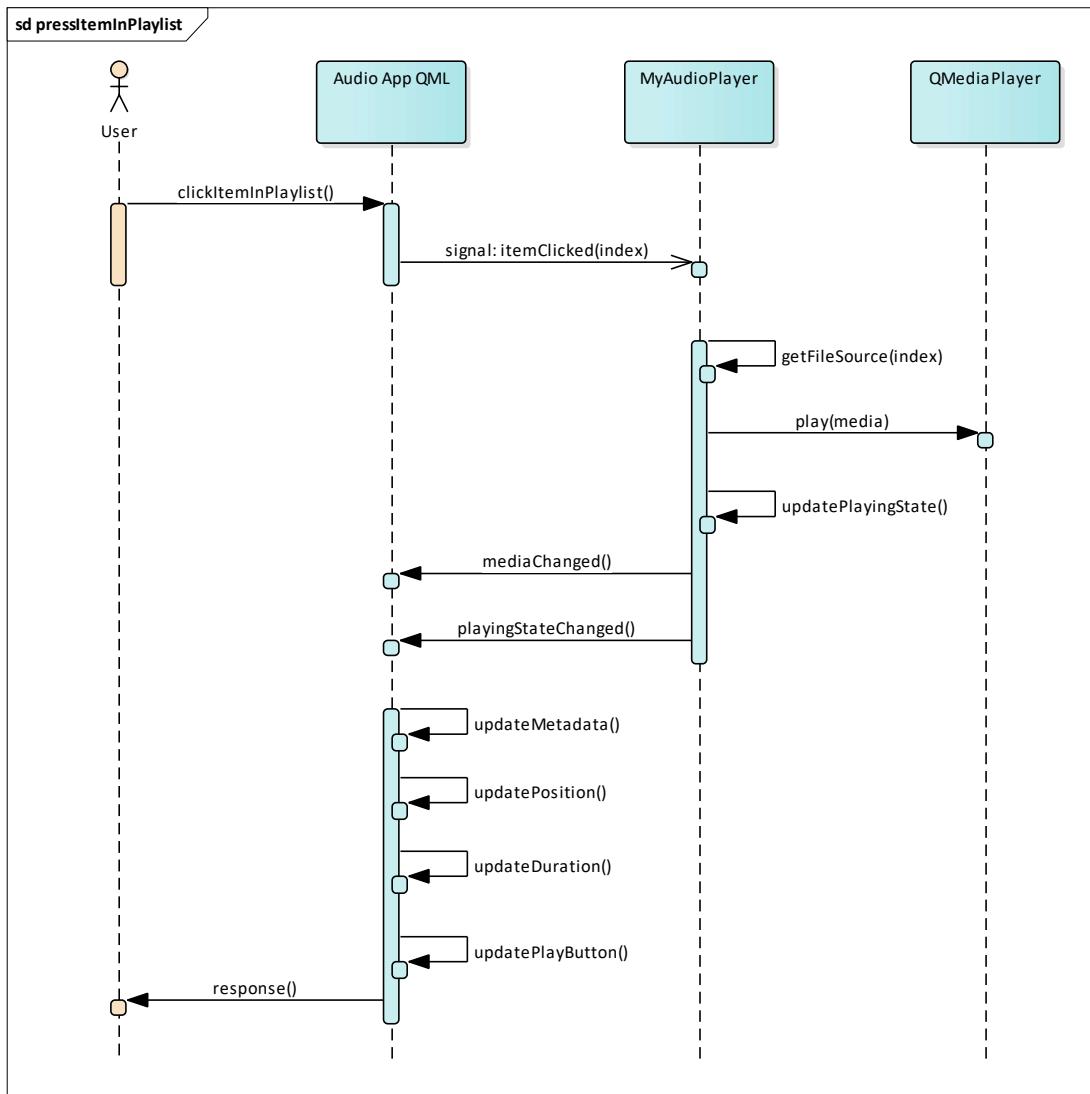
Hình 3.8 mô tả chức năng chuyển đến nội dung tiếp theo:



Hình 3.8 Chức năng chuyển đến nội dung tiếp theo

Theo Hình 3.8, khi người dùng nhấn vào nút bấm chuyển đến nội dung tiếp theo (next), MyAudioPlayer thực hiện dừng nội dung đang phát qua API `stop()`, sau đó thực hiện chuyển đến nội dung tiếp theo qua API `next()`, và phát nội dung này qua API `play()`. Sau khi thực hiện phát, nếu trạng thái phát lại đang áp dụng là `REPEAT_ONE`, tiến hành thay đổi chế độ phát lại thành `REPEAT_ALL`, sau đó thông báo thay đổi được gửi đến View, View tiến hành cập nhật thông tin hiển thị để phản hồi đến người dùng.

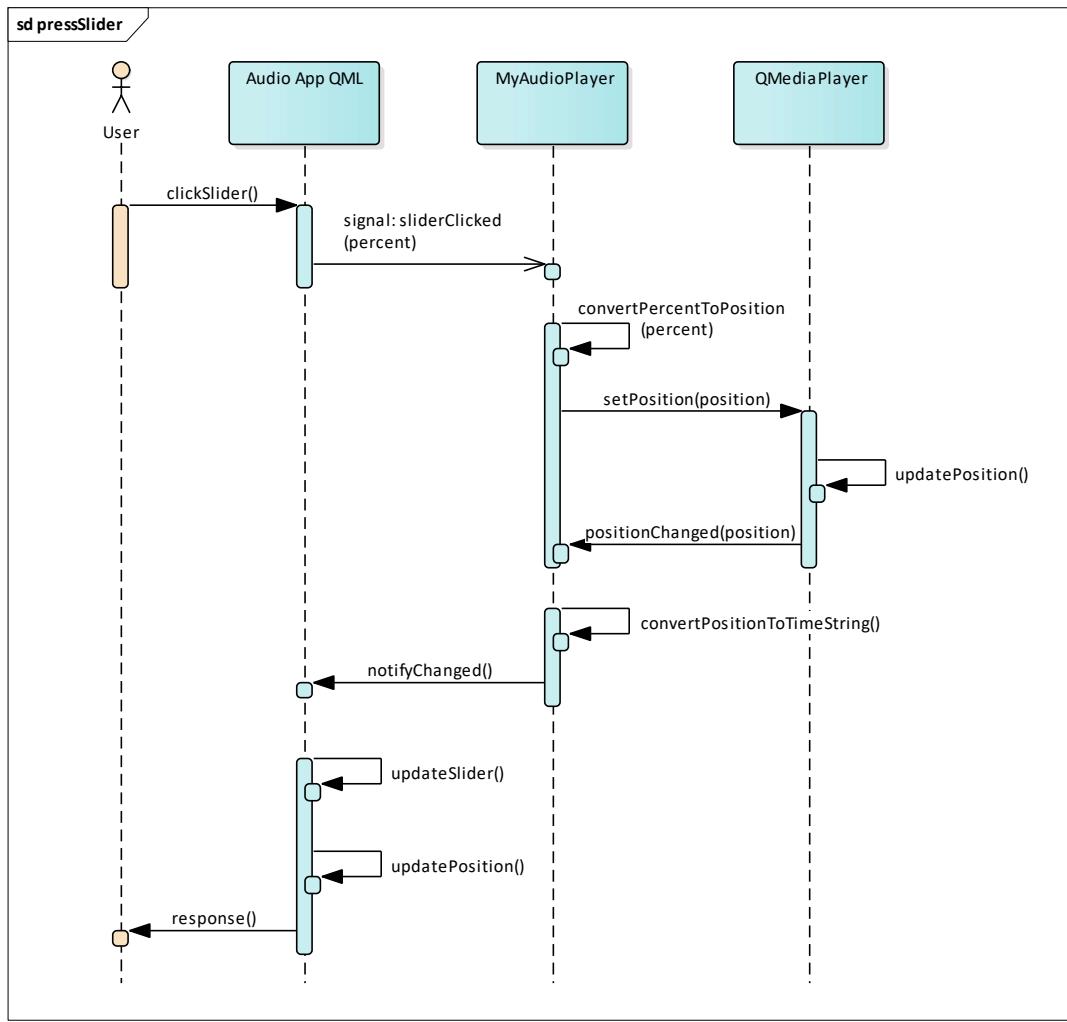
Hình 3.9 mô tả chức năng chọn phát một nội dung từ danh sách phát:



Hình 3.9 Chức năng chọn phát một nội dung từ danh sách

Theo Hình 3.9, khi người dùng nhấn chọn một nội dung trong danh sách phát, MyAudioPlayer dựa vào chỉ số của nội dung để xác định đường dẫn của file MP3 được chọn. Sau đó thực hiện giải mã và phát ra loa qua API *play()*, sau đó cập nhật thông tin về nội dung phát, trạng thái phát và gửi thông báo đến View. View thực hiện cập nhật hiển thị các thông tin được thay đổi theo thông báo để phản hồi đến người dùng.

Hình 3.10 mô tả chức năng chọn vị trí phát bất kì:

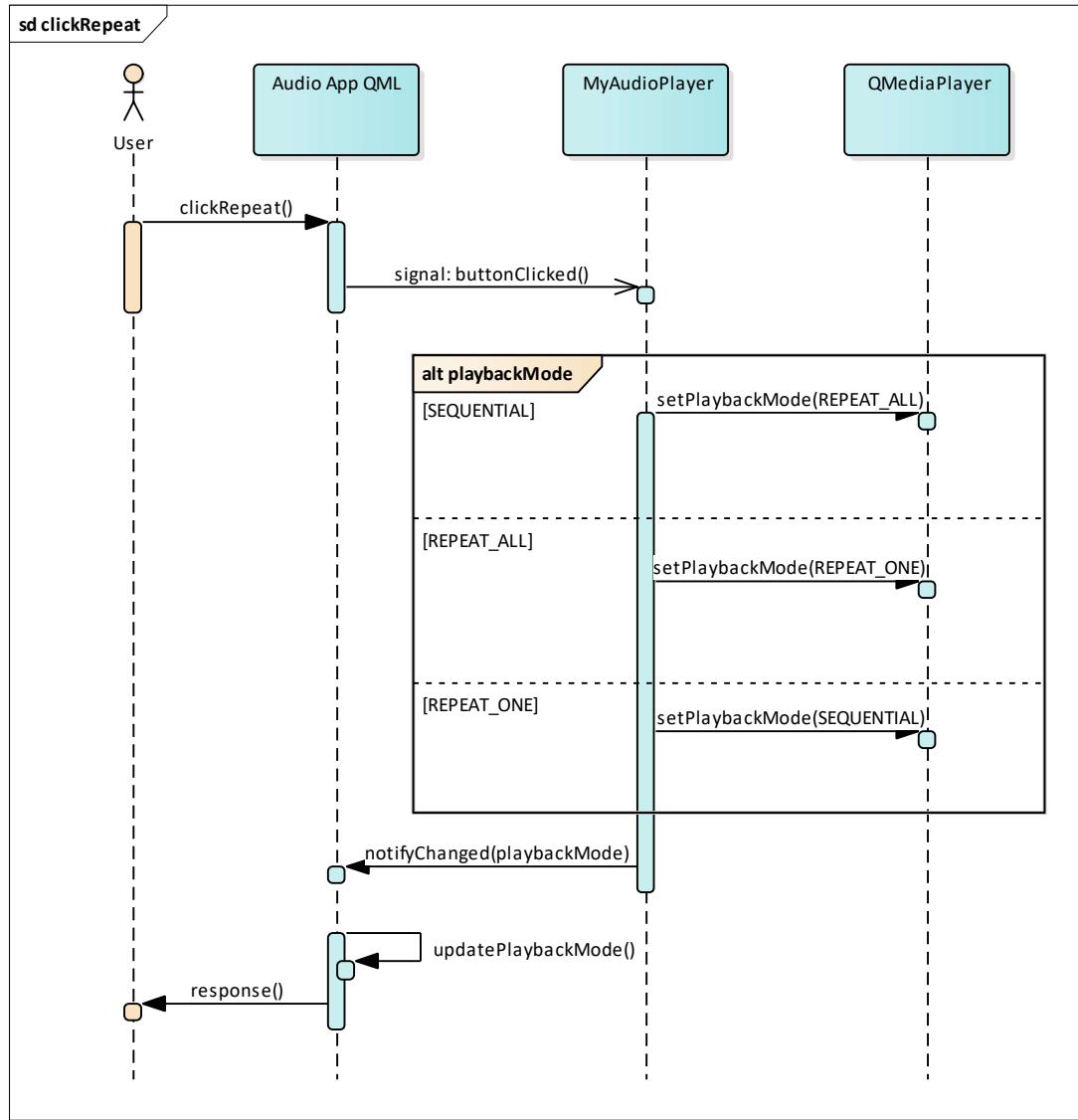


Hình 3.10 Chức năng chọn phát từ vị trí bất kì

Theo Hình 3.10, khi người dùng nhấp vào một vị trí bất kì trên thanh trượt, View sẽ gửi tín hiệu kèm theo thông tin về tỷ lệ tương đối trên thanh trượt được nhận. Dựa vào tỷ lệ tương đối này, MyAudioPlayer tiến hành tính toán ra vị trí (đơn vị ms) ứng với mong muốn của người dùng, bằng cách nhân tỉ lệ này với duration của file mp3. Sau đó thiết lập vị trí vừa tính được qua API `setPosition()`. Khi thiết lập thành công, MyAudioPlayer gửi thông báo cập nhật đến View, View sẽ tiến hành cập nhật hiển thị của thời lượng đã phát và thanh trượt để phản hồi đến người dùng.

e) Chọn chế độ phát lặp lại qua giao diện người dùng

Hình 3.11 mô tả chức năng chuyển đổi chế độ phát lặp lại qua giao diện



Hình 3.11 Chuyển chế độ phát lặp lại

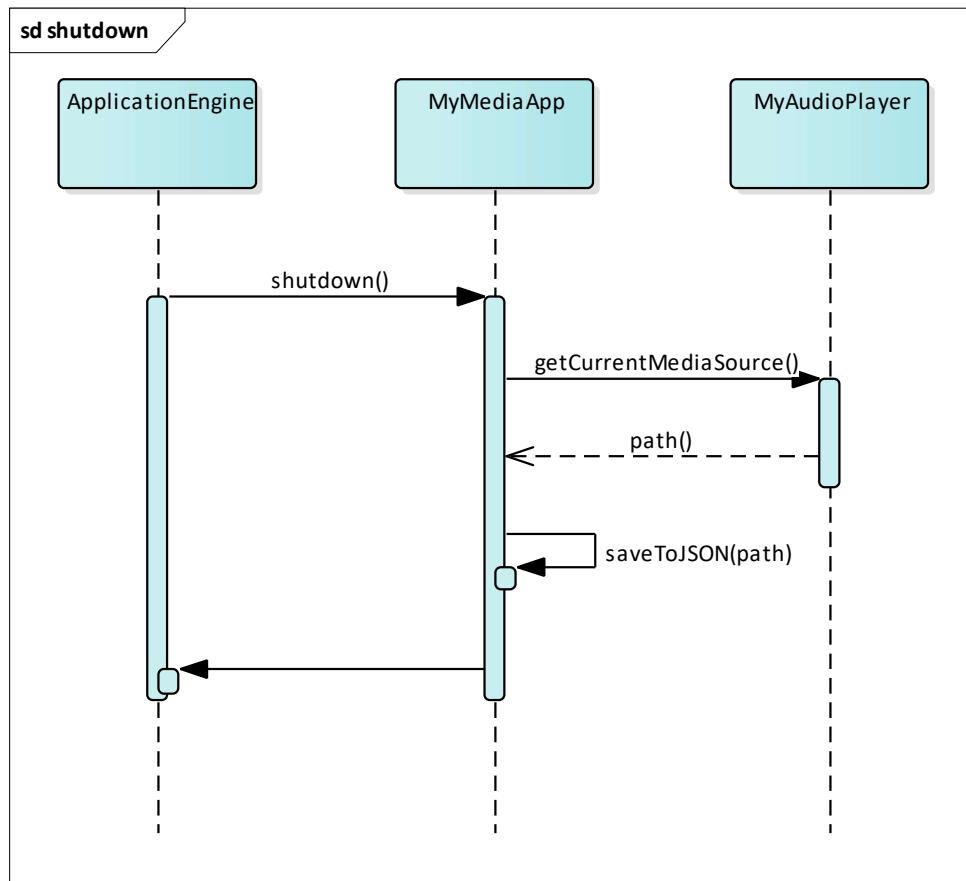
Theo Hình 3.11, khi người dùng thực hiện nhấp vào nút bấm lặp lại, yêu cầu chuyển chế độ phát lặp lại được gửi đến MyAudioPlayer.

- Nếu chế độ lặp đang áp dụng là SEQUENTIAL, chuyển sang chế độ REPEAT_ALL qua API setPlaybackMode()
- Nếu chế độ lặp đang áp dụng là REPEAT_ALL, chuyển sang chế độ REPEAT_ONE qua API setPlaybackMode()
- Nếu chế độ lặp đang áp dụng là REPEAT_ONE, chuyển sang chế độ SEQUENTIAL qua API setPlaybackMode()

Sau khi thay đổi chế độ lặp, MyAudioPlayer thông báo thay đổi đến View. View tiến hành cập nhật hiển thị để phản hồi đến người dùng

f) Lưu thông tin và tự động tải nội dung phát gần nhất khi khởi động

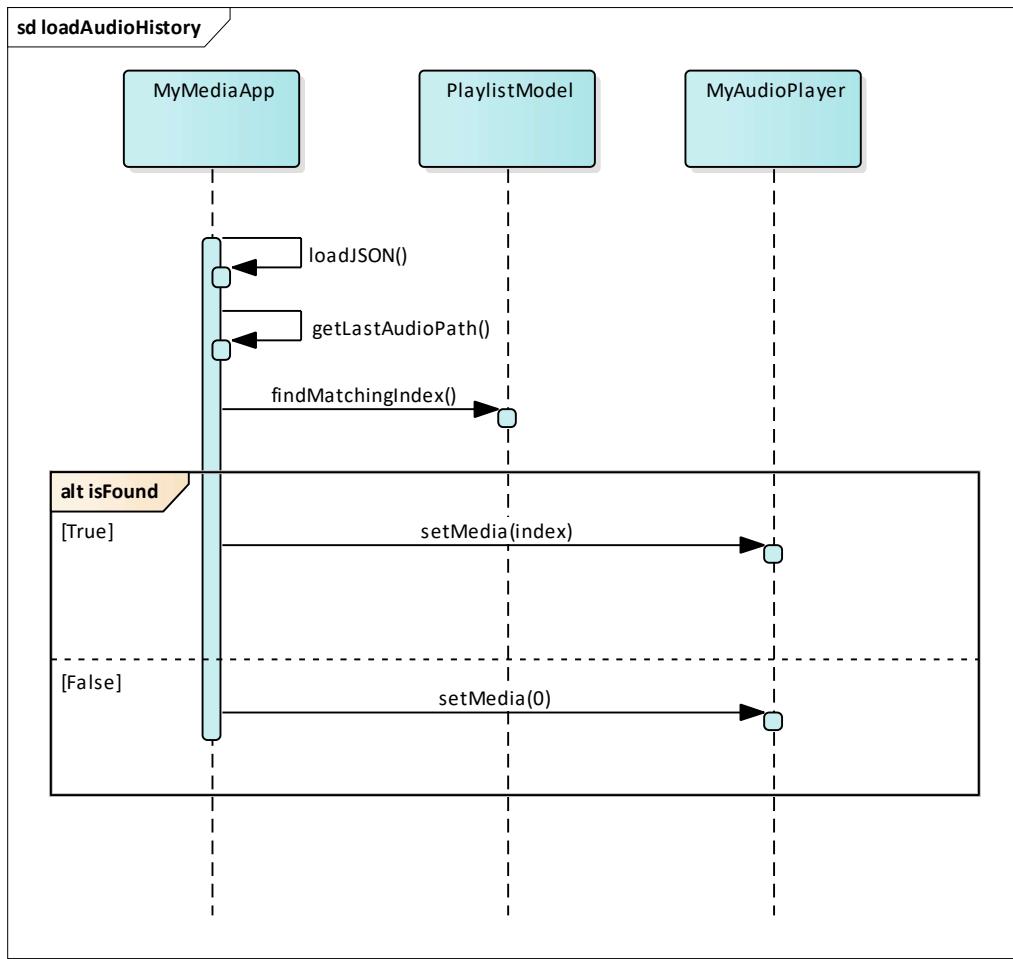
Hình 3.12 mô tả chức năng lưu thông tin nội dung audio gần nhất



Hình 3.12 Lưu thông tin phát gần nhất

Khi có yêu cầu đóng ứng dụng, MyMediaApp thực hiện lấy thông tin đường dẫn của file đang được chọn/phát từ MyAudioPlayer. Sau đó thực hiện lưu đường dẫn này vào file JSON. File JSON được định vị trong thư mục mặc định được hệ điều hành cấp riêng cho từng ứng dụng, được cung cấp bởi Qt. Tùy hệ điều hành, giá trị thực tế của đường dẫn vị trí này sẽ thay đổi.

Hình 3.13 mô tả chức năng tự động tải file được phát gần nhất khi khởi động. Theo Hình 3.13, khi ứng dụng khởi động, MyMediaApp thực hiện đọc và lấy ra đường dẫn của file được phát gần nhất được lưu trong file JSON. Sau đó tiến hành tìm chỉ số của file ứng với đường dẫn này trong PlaylistModel được tạo từ trước (tham khảo Hình 3.3).



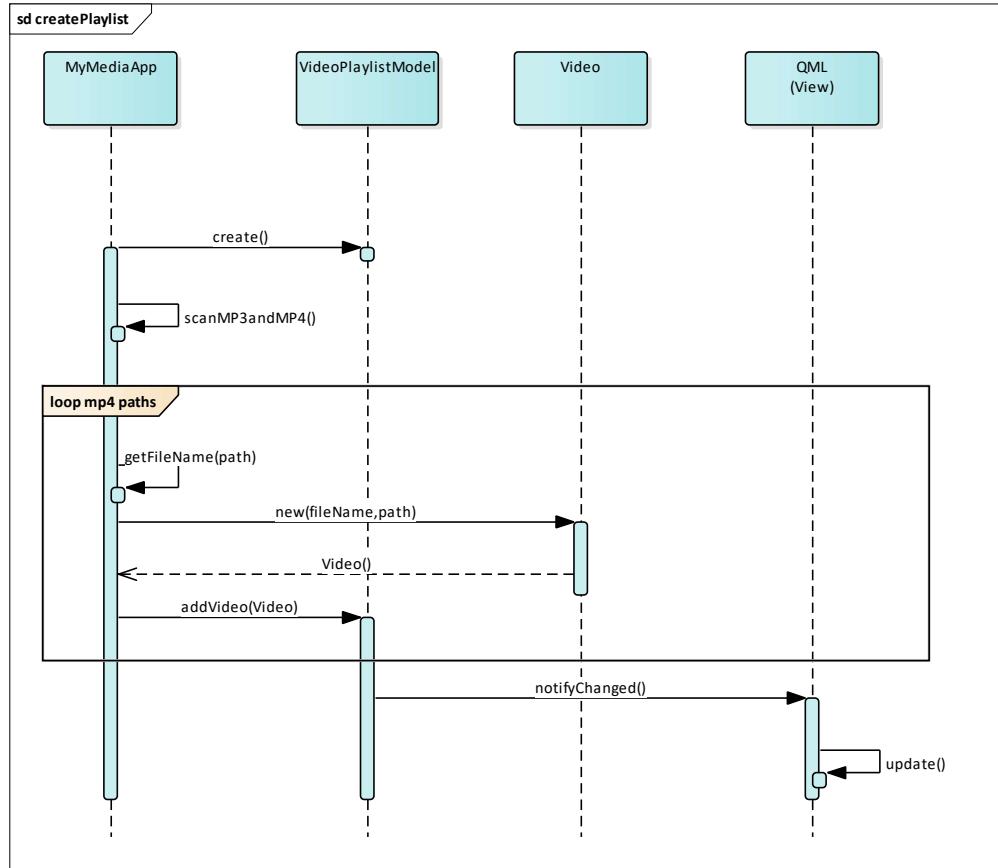
Hình 3.13 Tự động tải file được phát gần nhất khi khởi động

Nếu tìm được chỉ số, tiến hành thiết lập chỉ số này cho `MyAudioPlayer`. Nếu không, thực hiện thiết lập danh sách phát từ đầu (chỉ số 0).

3.3.3. Đặc tả các chức năng phát video

a) Lập và hiển thị danh sách phát

Quy trình tiến hành tương tự lập danh sách Audio, nhưng thay vì lấy thông tin meta-data, với file video ứng dụng chỉ tiến hành trích xuất tên file (không bao gồm phần mở rộng) và đường dẫn, sau đó tạo các đối tượng Video với hai thông tin này và thêm vào `VideoPlaylistModel`. (Hình 3.14)



Hình 3.14 Lập và hiển thị danh sách phát video

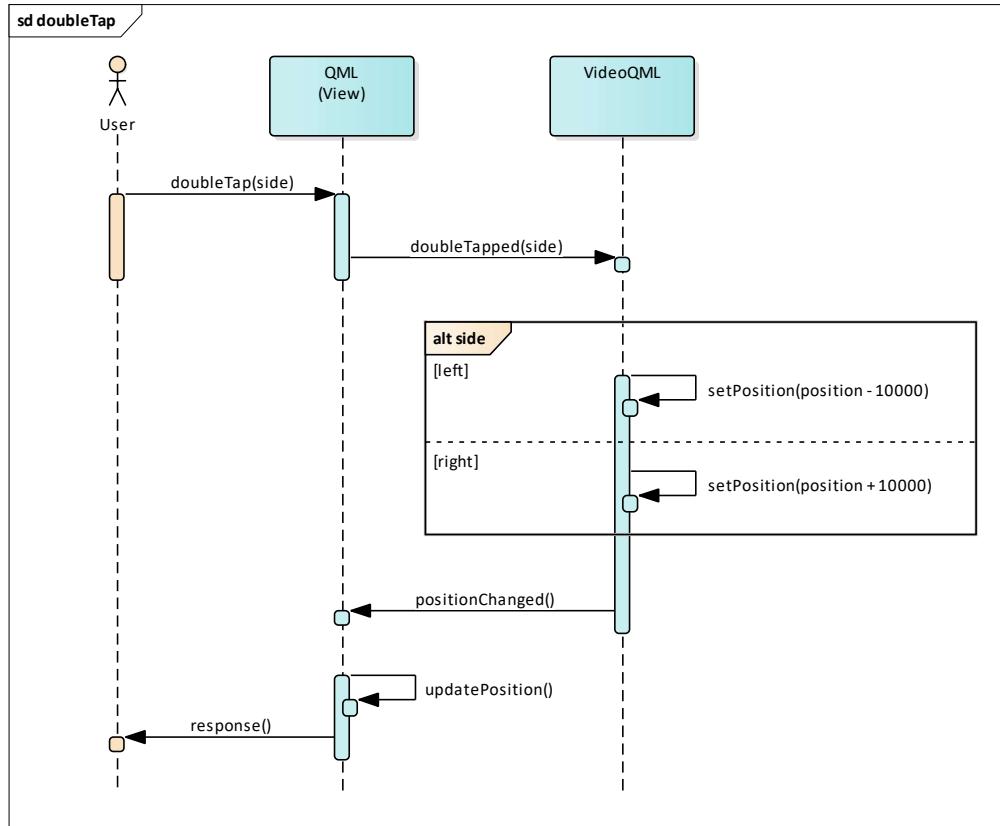
b) Hiển thị thông tin nội dung đang phát

Quy trình tương tự hiển thị nội dung audio, tuy nhiên View chỉ truy cập và hiển thị tên file được cung cấp qua các đối tượng Video trong VideoPlaylistModel.

c) Cung cấp khả năng điều khiển phát qua giao diện

Hình 3.15 mô tả chức năng tua đến 10 giây trước / sau.

Khi người dùng thực hiện nhấn đúp vào nửa màn hình bên trái hoặc bên phải, tín hiệu nhấn đúp được gửi tới VideoQML. Nếu người dùng nhấn vào bên trái màn hình, VideoQML thực hiện thiết lập position với giá trị giảm 10000 (ms).



Hình 3.15 Tua đến 10 giây trước (sau)

Nếu người dùng nhấn vào bên phải màn hình, VideoQML thiết lập position mới với giá trị tăng 10000 (ms). Sau khi thiết lập position, VideoQML thông báo thay đổi đến View, View thực hiện cập nhật hiển thị và phản hồi cho người dùng.

Các chức năng còn lại được thực hiện tương tự chức năng audio.

d) Lưu và hiển thị thông tin 2 nội dung phát gần nhất

Về tổng thể, chức năng lưu thông tin phát được thực hiện giống với chức năng lưu thông tin của audio, được lưu trong cùng file JSON ở cùng vị trí. Các thông tin được lưu của video gồm đường dẫn của 2 file được phát gần nhất và thời lượng đã phát của file phát gần nhất.

3.3.4. Mô tả bối cảnh giao diện

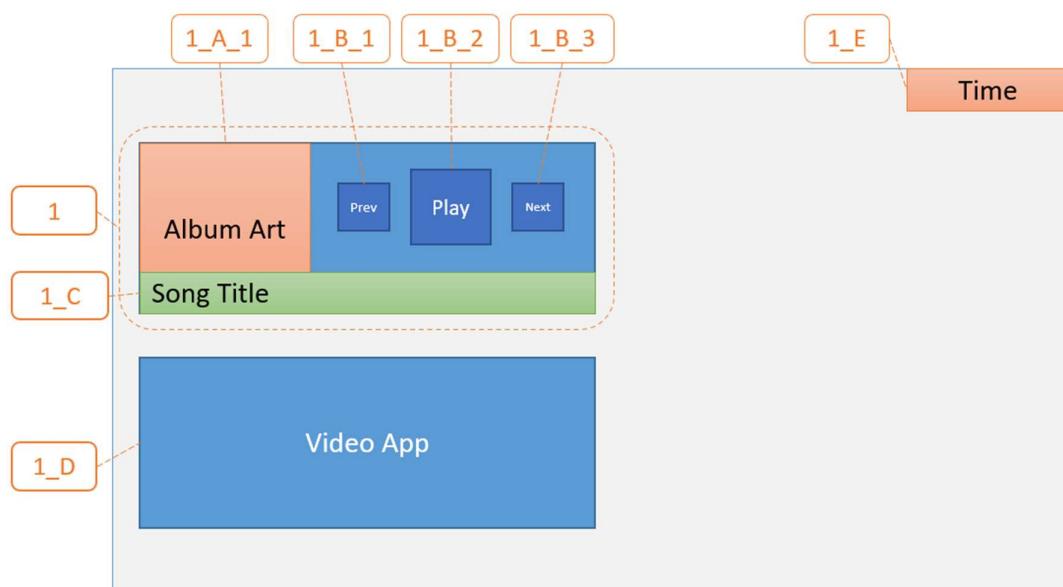
Giao diện chính là các khôi con của khôi View trong sơ đồ khôi của ứng dụng. Các khôi con bao gồm Home, AudioView và VideoView. Giao diện của ứng dụng được chia thành các màn hình hiển thị, được liệt kê tương ứng với các khôi như sau:

- Khôi Home:
 - Màn hình Home

- Khối AudioView:
 - Màn hình Audio - Player
 - Màn hình Audio - Playlist
- Khối VideoView:
 - Màn hình Video - Player
 - Màn hình Video - Playlist

Chi tiết các màn hình được trình bày trong Phụ lục B.

a) *Màn hình Home*

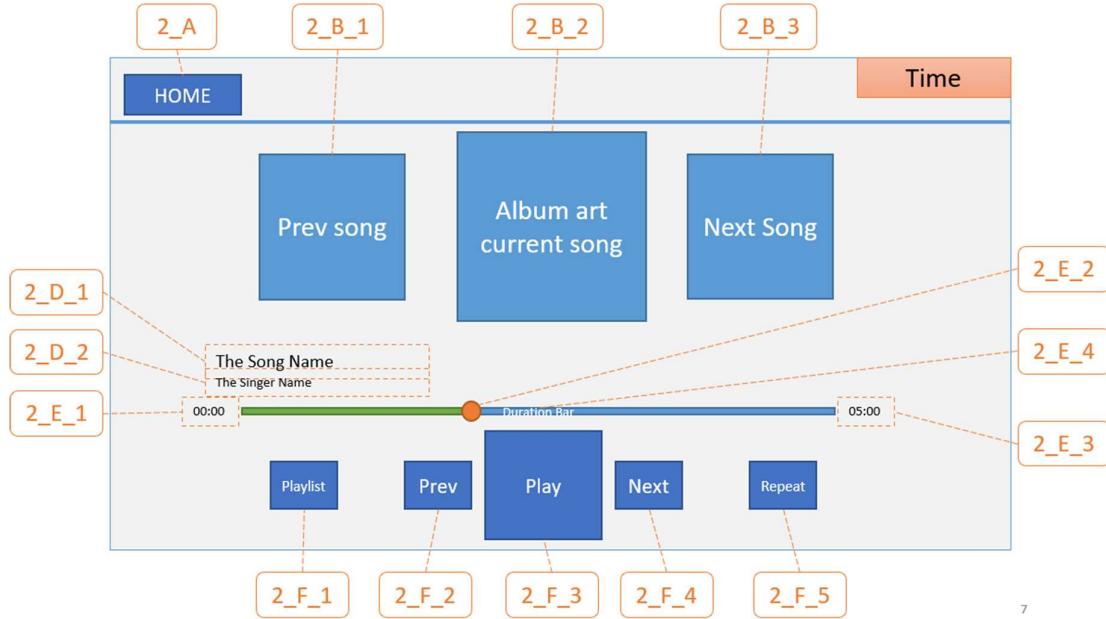


Hình 3.16 Màn hình Home

Màn hình Home đóng vai trò là trang chủ, giúp người dùng truy cập các chức năng phát file audio, phát file video và chuyển đổi qua lại giữa 2 chức năng này. Ngoài ra màn hình Home còn cung cấp khả năng xem nhanh thông tin và điều khiển dừng/phát, chuyển bài của chức năng phát audio.

b) *Màn hình Audio - Player*

Màn hình Audio - Player hiển thị các thông tin liên quan đến nội dung audio đang được phát bao gồm: tên nội dung đang phát, nghệ sĩ thể hiện, hình album của nội dung đang phát và 2 nội dung tiếp liền kề, thời lượng đã phát, tổng thời lượng, tỉ lệ thời lượng đã (qua thanh trượt)



Hình 3.17 Màn hình Audio - Player

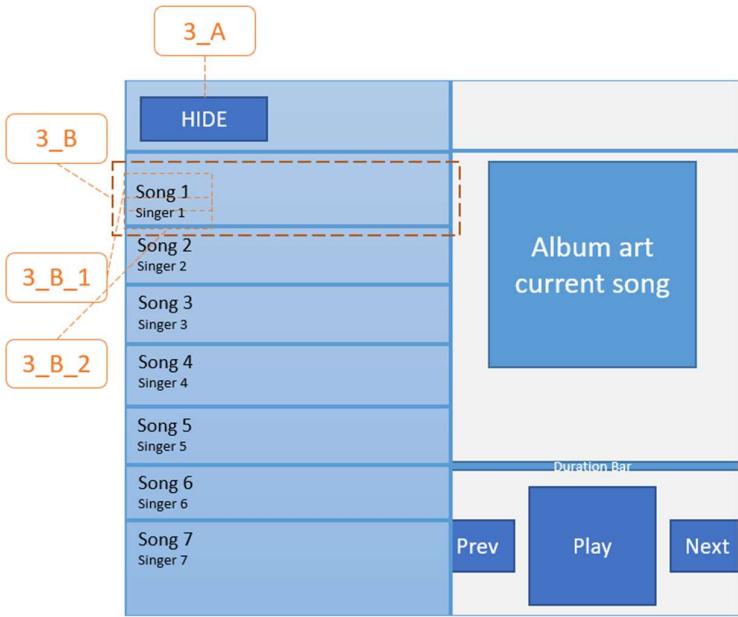
Màn hình Audio – Player cung cấp khả năng điều khiển phát file audio qua các nút bấm và thanh trượt, bao gồm: dừng/phát, chuyển bài, thay đổi chế độ lặp lại, mở danh sách phát.

c) Màn hình Audio – Playlist

Audio - Playlist hiển thị danh sách các nội dung audio, thể hiện qua các thông tin: Tên bài hát, nghệ sĩ thể hiện. Nó xuất hiện đè lên màn hình trước đó và chỉ chiếm phần bên trái khu vực hiển thị. (Hình 3.18)

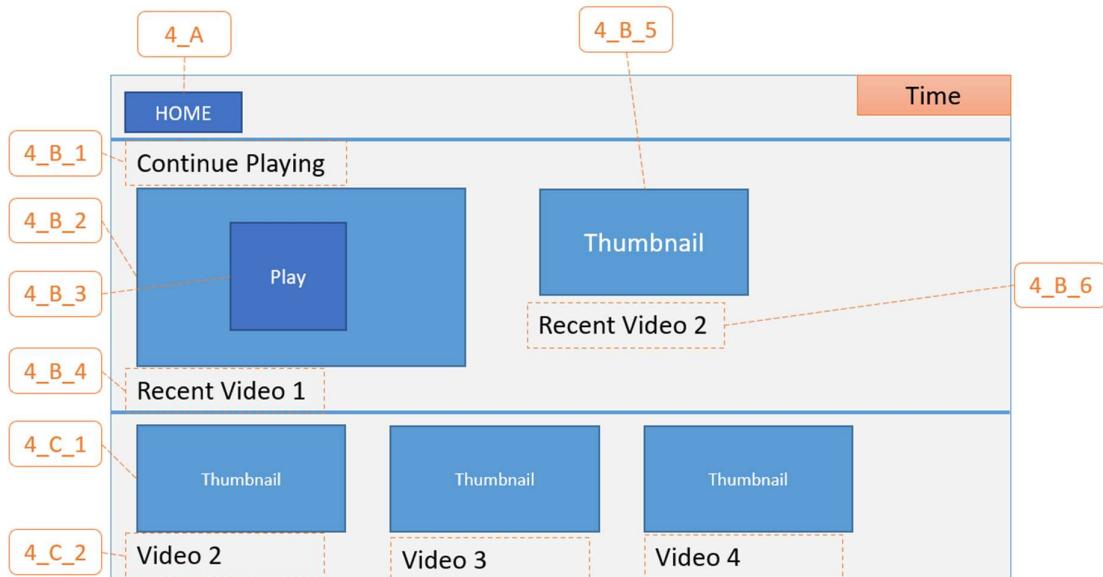
Khi người dùng nhấn vào một item trong danh sách, nội dung tương ứng với item đó sẽ được phát.

Thực tế trong quá trình phát triển, Audio – Playlist không được xếp thành màn hình riêng mà chỉ là một component của Audio – Player.



Hình 3.18 Màn hình Audio - Playlist

d) Màn hình Video - Playlist

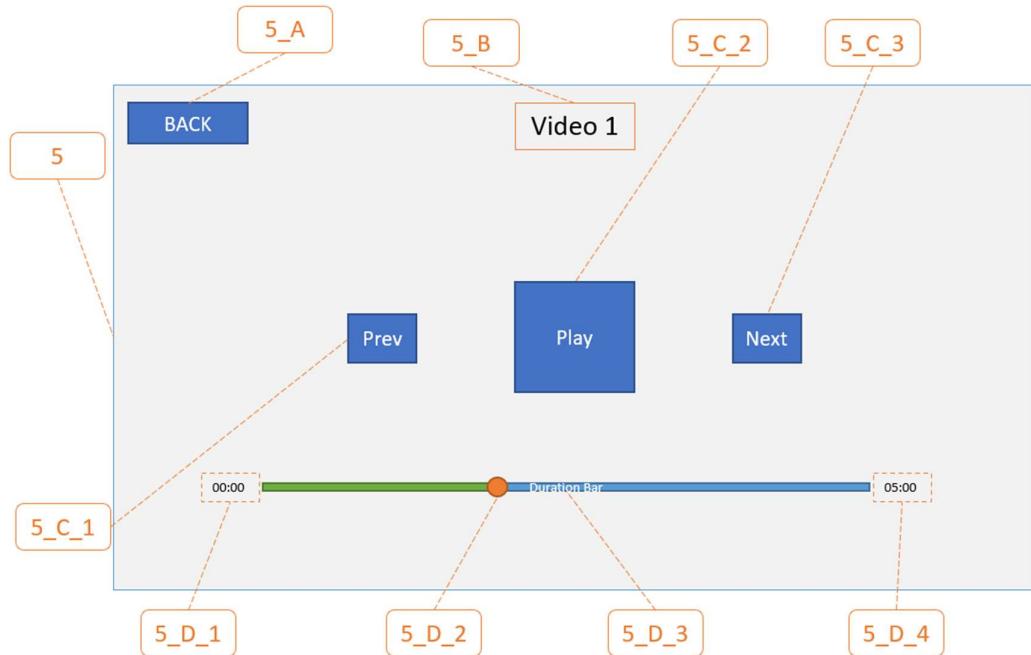


Hình 3.19 Màn hình Video - Playlist

Màn hình Video - Playlist hiển thị danh sách phát các nội dung video, lịch sử phát 2 video gần nhất. Hai nội dung gần nhất được hiển thị ở khu vực chính giữa của màn hình, nội dung gần hơn có kích thước lớn hơn.

Danh sách phát được hiển thị ở khu vực bên dưới màn hình, bao gồm hình đại diện và tên file video.

e) Màn hình Video - Player



Màn hình Video - Player có vai trò phát file video, hiển thị các nút bấm điều khiển video và hiển thị các thông tin liên quan đến video đang được phát.

Các nút bấm cung cấp chức năng điều khiển phát bao gồm: dừng/phát, chuyển video, tua trước/sau, chọn thời điểm phát trên thanh trượt

Thông tin được hiển thị về video bao gồm tên, tổng thời lượng và thời lượng đã phát.

3.4. Kết luận chương

Với các thông tin về phân tích và thiết kế được trình bày trong Chương 3, em tiến hành bước tiếp theo là lập trình sản phẩm. Mã nguồn của sản phẩm được quản lý bởi công cụ quản lý phiên bản Git, và được lưu trữ bằng dịch vụ GitHub tại đường dẫn: <https://github.com/dotrungkieu96/datn.git>

Sản phẩm sau khi hoàn thành được đánh giá ở Chương 4 tiếp theo.

Chương 4. Kiểm thử và đánh giá kết quả

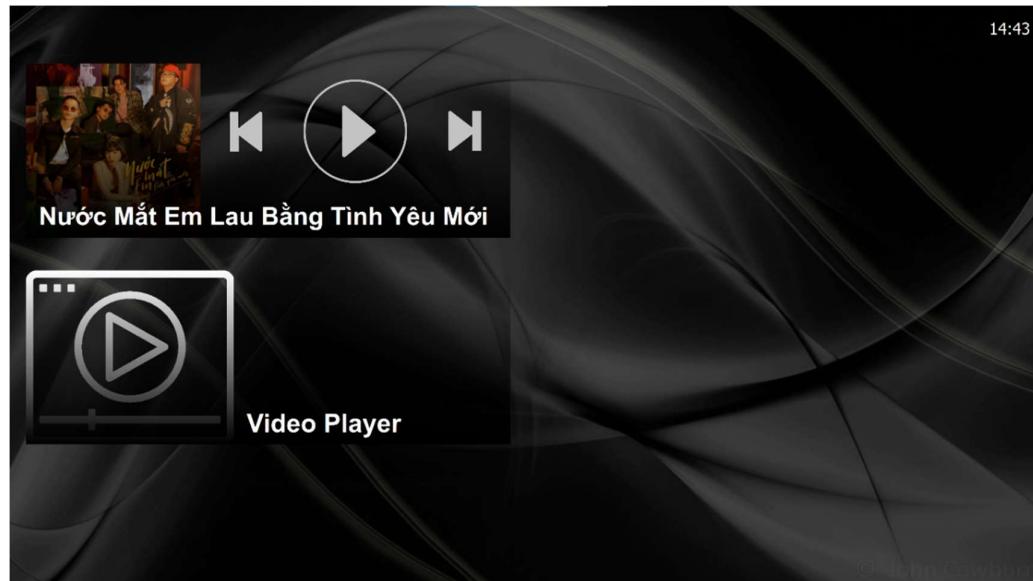
Chương 4 trình bày về quá trình kiểm thử sản phẩm sau khi đã hoàn thành giai đoạn phát triển với thiết kế đã được trình bày trong Chương 3. Nội dung trong Chương 4 bao gồm:

- Hình ảnh thực tế của sản phẩm
- Đánh giá mức độ hoàn thiện của sản phẩm so với các yêu cầu chức năng đã được phân tích trong mục 3.1.
- Đánh giá mức độ tuân thủ mô hình thiết kế MVC được đặt ra ban đầu trong mục 2.5.

4.1. Hình ảnh thực tế của sản phẩm

Phối màu của giao diện được tham khảo từ ứng dụng *Spotify* trên nền tảng Windows, với nền tối, các chi tiết sử dụng màu trắng xám kết hợp với xanh lá.

Màn hình Home của sản phẩm được chụp lại trong Hình 3.1 dưới đây.



Hình 4.1 Màn hình Home (sản phẩm)

Màn hình Home được triển khai chủ đạo theo tông màu tối, các thành phần được sắp xếp theo bố cục được mô tả trong mục 3.3.4, và hành vi được mô tả trong Phụ lục B.

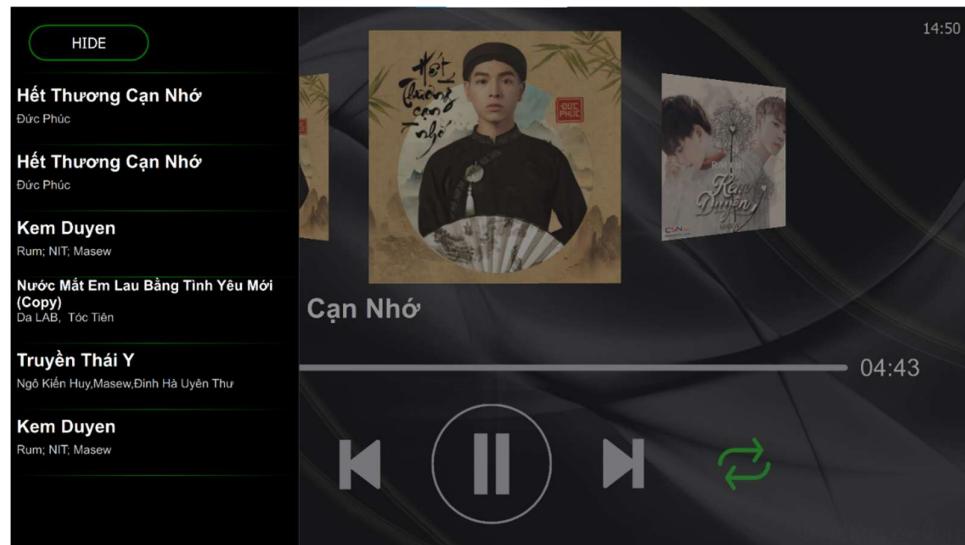
Màn hình Audio - Player của sản phẩm được chụp lại trong Hình 3.2:



Hình 4.2 Màn hình Audio - Player (sản phẩm)

Màn hình Audio - Player được triển khai chủ đạo theo tông màu tối, các thành phần được sắp xếp theo bố cục được mô tả trong mục 3.3.4, và hành vi được mô tả trong Phụ lục B.

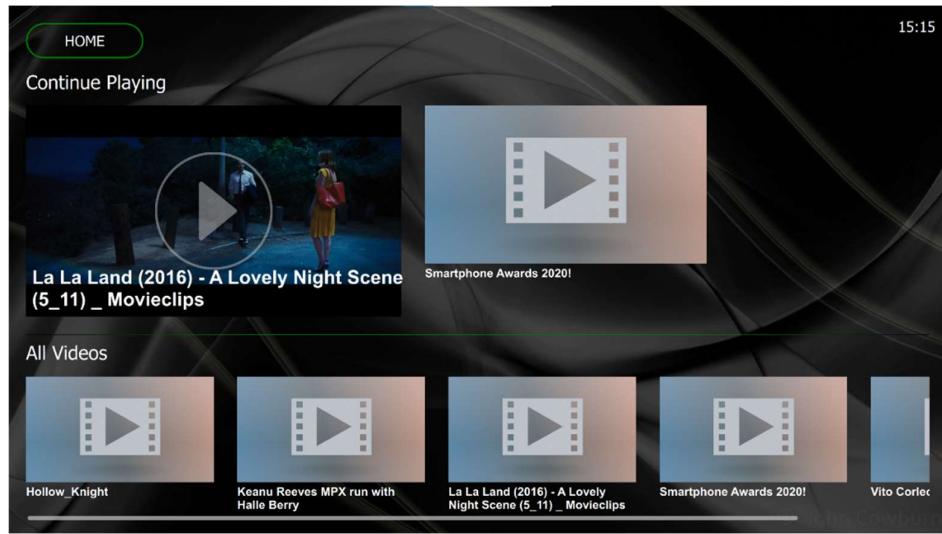
Màn hình Audio - Playlist của sản phẩm được chụp lại trong Hình 4.3.



Hình 4.3 Màn hình Audio - Playlist (sản phẩm)

Màn hình Audio - Player được triển với các thành phần được sắp xếp theo bố cục được mô tả trong mục 3.3.4, và hành vi được mô tả trong mục Phụ lục B

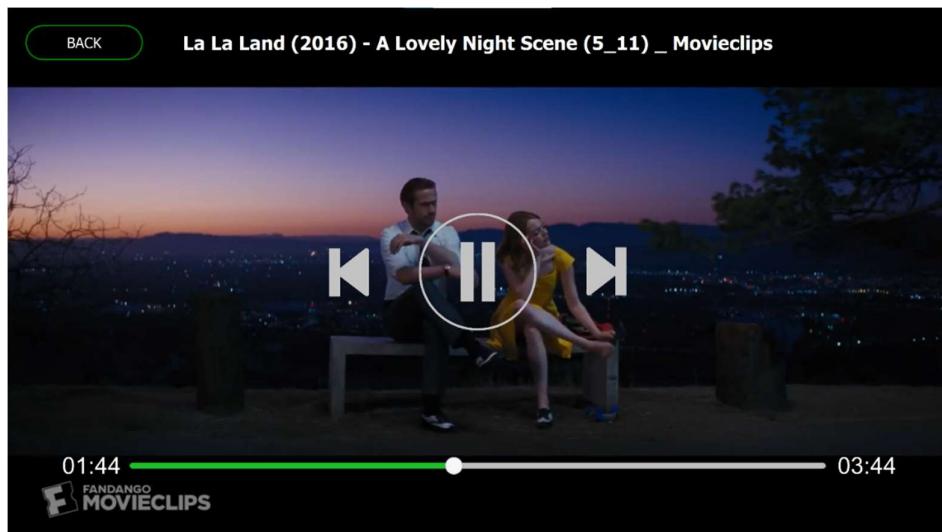
Màn hình Video - Playlist của sản phẩm được chụp lại trong Hình 3.4.



Hình 4.4 Màn hình Video - Playlist (sản phẩm)

Màn hình Video - Playlist được triển với các thành phần được sắp xếp theo bố cục được mô tả trong mục 3.3.4, và hành vi được mô tả trong Phụ lục B.

Màn hình Video - Player của sản phẩm được chụp lại trong Hình 3.5.



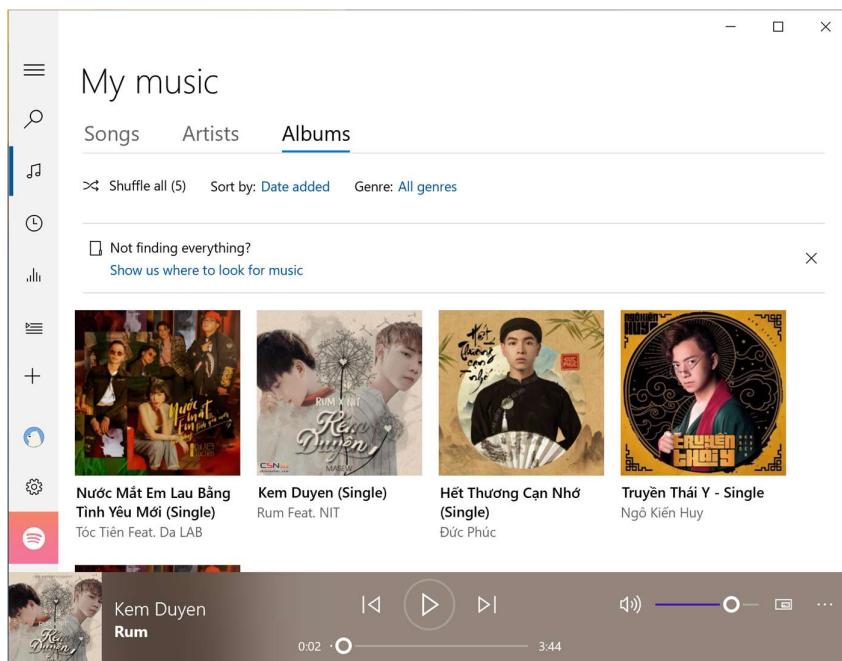
Hình 4.5 Màn hình Video - Player (sản phẩm)

Màn hình Video - Player được triển với các thành phần được sắp xếp theo bố cục được mô tả trong mục 2.3.3.5, và hành vi được mô tả trong mục 2.3.4.5. Em tự đánh giá về màn hình Video - Player như sau

4.2. Kiểm thử các yêu cầu

Về mặt chức năng của ứng dụng – tạm gọi là ứng dụng MyMediaApp, em thực hiện đánh giá theo các mục lớn tương ứng được đề ra trong mục 3.1, bao gồm chức năng phát file audio, chức năng phát file video. Các chức năng được đánh giá dựa theo các kịch bản được thiết kế cho từng yêu cầu. Phương pháp kiểm thử được áp dụng là phương pháp hộp đen (black-box) – kiểm thử trực tiếp trên sản phẩm, định nghĩa đầu vào, thực hiện thao tác và đánh giá kết quả đầu ra. Môi trường kiểm thử sản phẩm là Microsoft Windows 10.

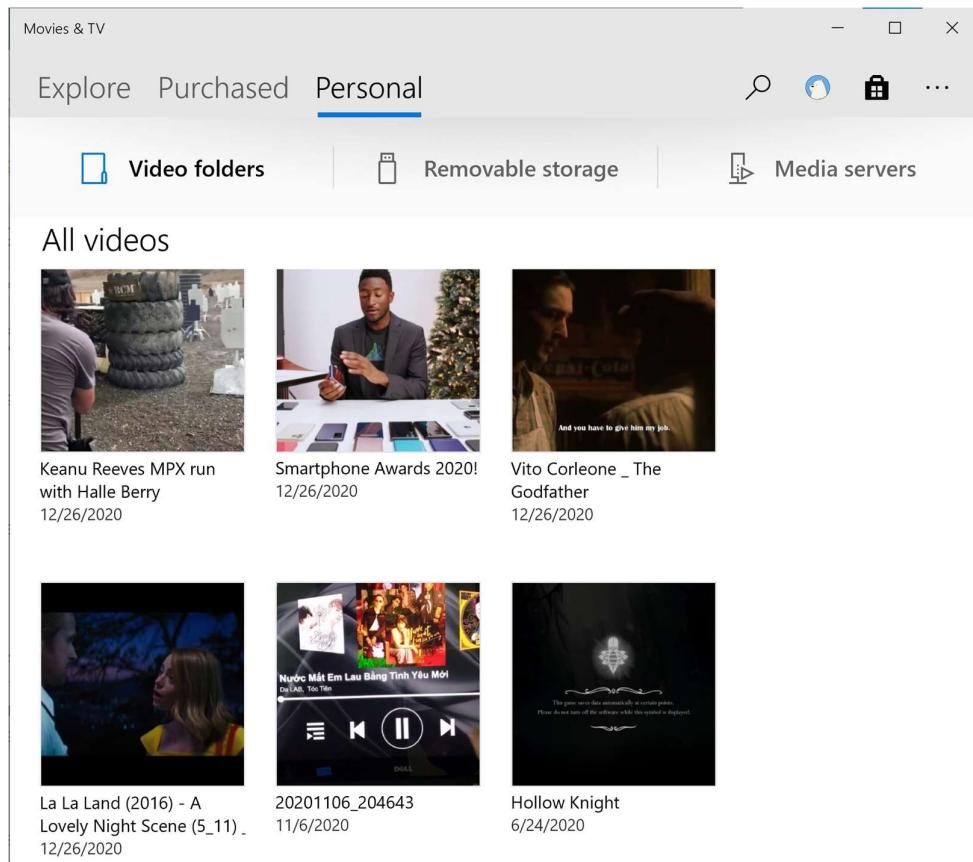
Với các tính năng liên quan đến hiển thị, một kịch bản audio được đánh giá là đạt khi kết quả đầu ra trên ứng dụng MyMediaApp tương đương với đầu ra khi thực hiện kịch bản trên ứng dụng “Groove Music” - ứng dụng phát audio mặc định của Windows 10, được phát triển và phát hành bởi Microsoft (Hình 4.6). Các tính năng liên quan đến điều khiển được đánh giá dựa vào so sánh giữa kết quả mong muốn và kết quả kiểm thử thực tế.



Hình 4.6 Ứng dụng Groove Music của Microsoft

Với các tính năng liên quan đến hiển thị, một kịch bản video được đánh giá là đạt khi kết quả đầu ra trên ứng dụng MyMediaApp tương đương với đầu ra khi thực hiện kịch bản trên ứng dụng “Movies & TV” - ứng dụng phát video mặc định của Windows 10, được phát triển và phát hành bởi Microsoft (Hình 4.7). Các tính năng

điều khiển được đánh giá bằng cách so sánh kết quả thực tế với kết quả mong muốn sau khi thực hiện kịch bản.



Hình 4.7 Ứng dụng Movies & TV của Microsoft

Một yêu cầu được đánh giá theo các mức độ:

- Hoàn thiện: Toàn bộ các kịch bản đều đạt
- Không hoàn thiện: Ít nhất 1 kịch bản thất bại

4.2.1. Kiểm thử và đánh giá chức năng phát file audio

Tập dữ liệu kiểm thử bao gồm 8 file mp3, chi tiết trong Bảng 4.1:

Bảng 4.1 Tập dữ liệu kiểm thử audio

STT	Tên file	Tiêu đề	Nghệ sĩ thể hiện	Thời lượng (mm:ss)	Hình album
1	mp3_sample_5.mp3	Hết Thương Cạn Nhớ	Đức Phúc	4:43	
2	mp3_sample_6.mp3	Kém Duyên	Rum	3:44	
3	mp3_sample_7.mp3	Nước Mắt Em Lau Băng Tình Yêu Mới	Da LAB, Tóc Tiên	4:45	
4	mp3_sample_8.mp3	Truyền Thái Y	Ngô Kiến Huy, Masew, Đinh Hà Uyên Thư	3:01	
5	mp3_sample_1.mp3	-	Unknown Artist	3:11	Mặc định
6	mp3_sample_2.mp3	-	Unknown Artist	3:08	Mặc định
7	mp3_sample_3.mp3	Galway	Kevin MacLoed	0:16	Mặc định
8	mp3_sample_4.mp3	Furious Freak	Kevin MacLoed	3:12	Mặc định

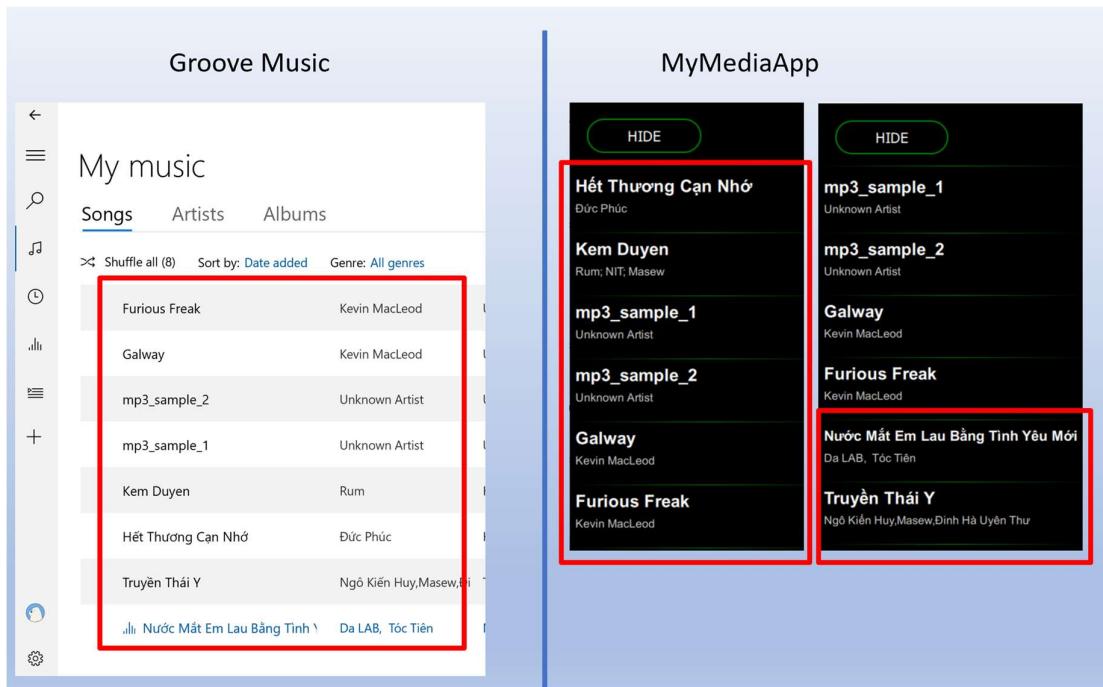
a) Lập và hiển thị danh sách phát

Với chức năng lập và hiển thị danh sách phát, em thiết kế 2 kịch bản kiểm thử, cho trường hợp cả 8 file nằm cùng cấp và trường hợp các file nằm trong các thư mục (folder) con.

Kịch bản 1: cả 8 file nằm cùng cấp

Kết quả mong muốn: Cả 8 file đều được hiển thị trong danh sách phát, các thông tin hiển thị gồm Tiêu đề và Nghệ sĩ thể hiện.

Kết quả thực tế: Cả 8 file đều xuất hiện trong danh sách phát với các thông tin yêu cầu (Hình 4.8)



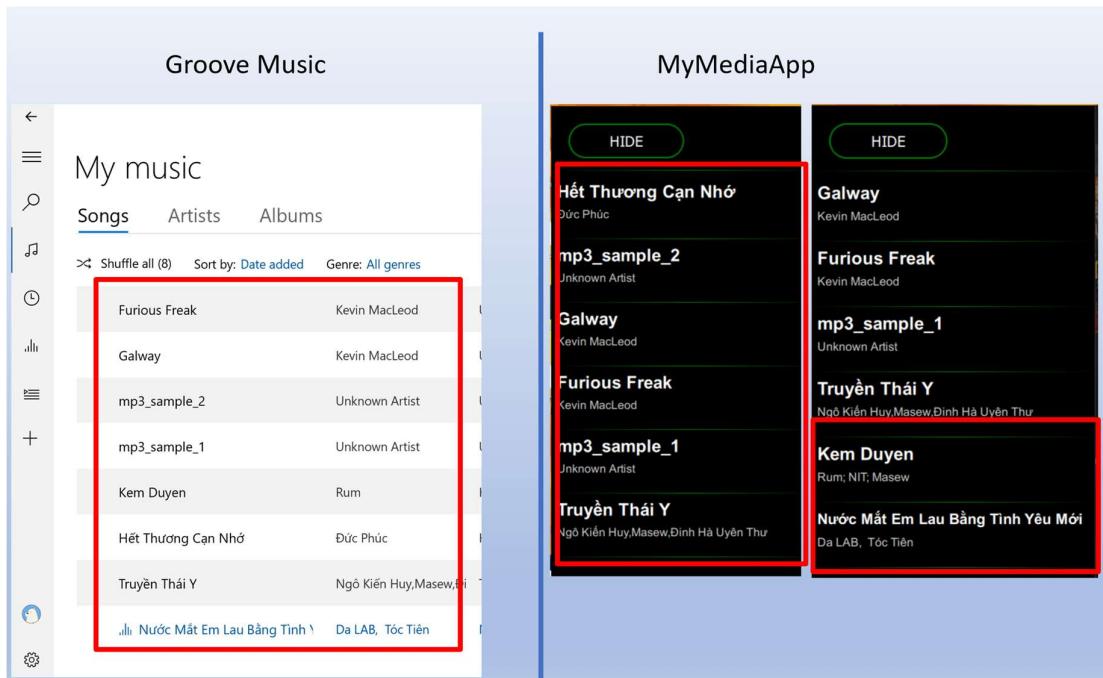
Hình 4.8 Kết quả kiểm thử lập danh sách phát - 1

Có thể thấy danh sách phát trong MyMediaApp có nội dung tương tự danh sách phát trong Groove Music mặc dù thứ tự không đồng nhất. Từ đó có thể đưa ra kết luận **Kịch bản 1 đạt**.

Kịch bản 2: 8 file được chia vào các thư mục con

Kết quả mong muốn: Cả 8 file đều được hiển thị trong danh sách phát, các thông tin hiển thị bao gồm Tiêu đề và Nghệ sĩ thể hiện.

Kết quả thực tế: Cả 8 file đều xuất hiện trong danh sách phát với các thông tin yêu cầu (Hình 4.9)



Hình 4.9 Kết quả kiểm thử lập danh sách phát - 2

Đánh giá kết quả thực tế, có thể thấy danh sách phát trong MyMediaApp có nội dung tương tự danh sách phát trong Groove Music mặc dù thứ tự không đồng nhất. Từ đó có thể đưa ra kết luận **Kịch bản 2 đạt**.

Kết luận: Yêu cầu “Lập và hiển thị danh sách phát” được đạt mức độ: **Hoàn thiện**, với 2/2 kịch bản đạt.

b) *Hiển thị meta-data*

Để kiểm thử tính năng hiển thị meta-data, các kịch bản được thực hiện giống nhau, chỉ khác ở file đầu vào và kết quả mong muốn ở đầu ra. Trong đó file đầu vào lần lượt là các file 1, 2, ..., 8 trong Bảng 4.1. Các bước thực hiện bao gồm:

- Khởi động ứng dụng
- Mở danh sách phát
- Chọn file 1, 2, ..., 8 trong danh sách

Đầu ra mong muốn: Thông tin được hiển thị bao gồm Tiêu đề file, Nghệ sĩ thể hiện, Hình album ứng với file được chọn. Chi tiết trong Bảng 4.1

Chi tiết được thể hiện trong Bảng 4.2 dưới đây:

Bảng 4.2 Kết quả kiểm thử tính năng hiển thị meta-data

MP3 File	MyMediaApp	Groove Music	Đánh giá
1	 Hết Thương Cạn Nhớ Đức Phúc	 Hết Thương Cạn Nhớ Đức Phúc	Đạt
2	 Kem Duyen Rum; NIT; Masew	 Rum Kem Duyen	Đạt
3	 Nước Mắt Em Lau Bằng Tình Yêu Mới Da LAB, Tóc Tiên	 Nước Mắt Em Lau Bằng Tình Yêu Mới Da LAB, Tóc Tiên	Đạt
4	 Truyền Thái Y Ngô Kiến Huy, Masew, Đinh Hà Uyên Thư	 Truyền Thái Y Ngô Kiến Huy, Masew, Đinh Hà Uyên Thư	Đạt
5	 mp3_sample_1 Unknown Artist	 mp3_sample_1 Unknown Artist	Đạt

6			Đạt
7			Đạt
8			Đạt

Bảng 4.2 cho thấy thông tin được hiển thị trên MyMediaApp và Groove Music là tương đồng về nội dung hiển thị. Kết luận: yêu cầu “Hiển thị meta-data” đạt mức độ: **Hoàn thiện**, với 8/8 kịch bản thành công.

c) *Hiển thị thời lượng*

Kịch bản kiểm thử tính năng hiển thị thời lượng đang phát được xây dựng với các thời gian phát khác nhau, sử dụng bộ đếm thời gian trên smartphone để xác định thời gian, và đối chiếu thời lượng hiển thị của MyMediaApp so với Groove Music.

Kết quả thực hiện kịch bản được trình bày trong Bảng 4.3:

Bảng 4.3 Kết quả kiểm thử tính năng hiển thị thời lượng đã phát

STT File	Thời gian phát thực tế (mm:ss)	Groove Music (mm:ss)	MyMediaApp (mm:ss)	Đánh giá
1	0:05	0:05	0:05	Đạt
2	0:15	0:15	0:15	Đạt
3	0:45	0:45	0:45	Đạt
4	1:30	1:30	1:30	Đạt

5	1:53	1:53	1:53	Đạt
6	3:08	3:08	3:08	Đạt
7	0:01	0:01	0:01	Đạt
8	3:11	3:11	3:11	Đạt

Dựa vào kết quả trong Bảng 4.3, tính năng hiển thị thời lượng đã phát đạt 8/8 kịch bản.

Kịch bản kiểm thử tính năng hiển thị thời lượng tổng của file audio được xây dựng với đầu vào và các bước thực hiện tương tự mục 4.2.1b – Kiểm thử tính năng hiển thị meta-data. Kết quả kiểm thử được trình bày trong Bảng 4.4 dưới đây, (kết quả vẫn có thể coi là đạt khi chênh lệch không quá 1s):

Bảng 4.4 Kết quả kiểm thử tính năng hiển thị thời lượng tổng của file

STT File	Thời lượng mong muốn	Groove Music	MyMediaApp	Đánh giá
1	4:43	4:43	04:43	Đạt
2	3:44	3:44	03:45	Đạt
3	4:45	4:45	04:45	Đạt
4	3:01	3:01	03:01	Đạt
5	3:11	3:11	06:14	Thất bại
6	3:08	3:08	03:33	Thất bại
7	0:16	0:16	00:31	Thất bại
8	3:12	3:12	06:14	Thất bại

Với kết quả được trình bày trong Bảng 4.4, tính năng hiển thị thời lượng tổng chưa đạt, chỉ 4/8 kịch bản thành công

Kết luận: yêu cầu “Hiển thị thời lượng” đạt mức độ: **Không hoàn thiện**, với 12/16 kịch bản kiểm thử đạt, 4/16 kịch bản thất bại.

d) Điều khiển phát qua giao diện đồ họa

Do danh sách phát được tạo bởi 2 ứng dụng khác nhau về thứ tự, tính năng điều khiển phát trên ứng dụng MyMediaApp được đánh giá dựa trên đầu ra thực tế so với đầu ra mong muốn được thiết kế trong kịch bản. Các kịch bản và kết quả như sau:

Kịch bản 1: Tạm dừng phát nội dung khi nhấn vào nút tạm dừng. Kết quả mong muốn: Nội dung audio ngừng phát.

Kết quả thực tế: Nội dung audio ngừng phát. Đánh giá kết quả: **Kịch bản 1 đạt.**

Kịch bản 2: Tiếp tục phát nội dung khi nhấn vào nút tiếp tục phát. Kết quả mong muốn: Nội dung tiếp tục phát

Kết quả thực tế: Nội dung tiếp tục phát. Đánh giá kết quả: **Kịch bản 2 đạt.**

Kiểm thử tính năng phát lại và chuyển về bài trước đó với 2 kịch bản như sau:

Kịch bản 3: Phát lại khi nội dung đã phát quá 3s và người dùng nhấn vào nút chuyển về bài trước đó. Kết quả mong muốn: Nội dung phát lại từ đầu, giá trị thời lượng đã phát thay đổi thành 00:00, vị trí thanh trượt ở ngoài cùng bên trái.

Kết quả thực tế: Nội dung phát lại từ đầu, giá trị thời lượng đã phát thay đổi thành 00:00, vị trí thanh trượt ở ngoài cùng bên trái. Đánh giá kết quả: **Kịch bản 3 đạt.**

Kịch bản 4: Chuyển đến nội dung trước đó khi nội dung đã phát ít hơn 3 giây và người dùng nhấn vào nút bấm chuyển về bài trước đó. Kết quả mong muốn: Nội dung trước đó – ứng với nội dung cuối cùng trong danh sách, nội dung số 8 – được phát.

Kết quả thực tế: Nội dung thứ 8 trong danh sách được phát. Đánh giá kết quả: **Kịch bản 4 đạt.**

Kiểm thử tính năng chuyển đến nội dung kế tiếp với 2 kịch bản như sau:

Kịch bản 5: Nhấn chuyển nội dung tiếp theo 1 lần khi đang phát nội dung thứ 1. Kết quả mong muốn: Nội dung thứ 2 trong danh sách được phát.

Kết quả thực tế: Nội dung thứ 2 trong danh sách được phát. Đánh giá: **Kịch bản 5 đạt**

Kịch bản 6: Nhấn chuyển nội dung tiếp theo 5 lần khi đang phát nội dung thứ 1. Kết quả mong muốn: Nội dung thứ 6 trong danh sách được phát.

Kết quả thực tế: Nội dung thứ 6 trong danh sách được phát. Đánh giá: **Kịch bản 6 đạt.**

e) *Chọn chế độ phát lặp lại qua giao diện đồ họa*

Kịch bản 1: Chọn chế độ không lặp lại và phát nội dung cuối cùng (số 8) trong danh sách. Kết quả mong muốn: Không phát nội dung tiếp theo khi kết thúc.

Kết quả thực tế: Không phát nội dung tiếp theo khi kết thúc. Đánh giá: **Kịch bản 1 đạt.**

Kịch bản 2: Chọn chế độ lặp lại toàn bộ và phát nội dung cuối cùng (số 8) trong danh sách. Kết quả mong muốn: Nội dung số 1 được phát sau khi kết thúc nội dung số 8.

Kết quả thực tế: Nội dung số 1 được phát sau khi kết thúc nội dung số 8. Đánh giá: Kịch bản 2 đạt.

Kịch bản 3: Chọn chế độ lặp lại 1 bài và phát nội dung số 5. Kết quả mong muốn: Nội dung số 5 được phát lại sau khi kết thúc.

Kết quả thực tế: Nội dung số 5 được phát lại sau khi kết thúc. Kết quả: Kịch bản 5 đạt.

Kết luận: Chức năng “Chọn chế độ phát lặp lại qua giao diện đồ họa” hoàn thiện với 3/3 kịch bản đạt.

f) *Lưu thông tin và tự động tải nội dung phát gần nhất khi khởi động*

Kịch bản 1: Phát nội dung số 4, tắt và khởi động lại ứng dụng. Kết quả mong muốn: Nội dung số 4 được chọn sẵn sau khi khởi động lại ứng dụng.

Kết quả thực tế: Nội dung số 4 được chọn sẵn. Đánh giá kết quả: **Kịch bản 1 đạt.**

Chức năng “Lưu thông tin và tự động tải nội dung phát gần nhất khi khởi động” đạt với 1/1 kịch bản đạt.

4.2.2. *Kiểm thử và đánh giá chức năng phát file video*

Tập dữ liệu kiểm thử gồm 5 file video, chi tiết trong Bảng 4.5:

Bảng 4.5 Tập dữ liệu kiểm thử video

Tên file	Thời lượng (mm:ss)
mp4_sample_1.mp4	01:45
mp4_sample_2.mp4	00:54
mp4_sample_3.mp4	25:04
mp4_sample_4.mp4	03:44
mp4_sample_5.mp4	05:24

a) *Lập và hiển thị danh sách phát*

Các kịch bản tương tự 4.2.1a. Kết quả đạt được: **2/2 kịch bản thành công.**
Chức năng hoàn thiện với kết quả trong Hình 4.10 và 4.11:



Hình 4.10 Kiểm thử tính năng lập danh sách phát video – MyMediaApp



Hình 4.11 Kiểm thử tính năng lập danh sách phát video – Movies & TV

b) *Hiển thị tên và thời lượng nội dung đang phát*

Các kịch bản tương tự 4.2.1b với 5 file trong tập dữ liệu kiểm thử. Kết quả **5/5** kịch bản đạt. Chức năng hoàn thiện.

c) *Cung cấp khả năng điều khiển phát qua giao diện*

Các kịch bản tương tự 4.2.1d với 5 file trong tập dữ liệu kiểm thử. Kết quả **5/5** kịch bản đạt. Chức năng hoàn thiện.

d) *Lưu và hiển thị thông tin 2 nội dung phát gần nhất*

Kịch bản tương tự 4.2.1f với 2 file video 1 và 2. Kết quả **1/1** kịch bản thành công. Chức năng hoàn thiện.

4.3. Đánh giá mức độ tuân thủ mô hình MVC

Về mặt tuân thủ mô hình thiết kế MVC và sơ đồ khôi được đặt ra ở mục 2.2.1, em tự đánh giá mức độ tuân thủ trung bình. Các điểm không đạt có thể kể đến:

- MyMediaApp thuộc khối Application Manager, nhưng lại thực hiện đọc và ghi dữ liệu (là công việc của Model)
- MyMediaPlayer thuộc khối Controller, nhưng lại cung cấp các thông tin trạng thái (ví dụ đang dừng / phát) để cập nhật View (các thông tin này theo lý thuyết phải nằm trong Model)

- VideoQML được cung cấp sẵn bởi Qt, vừa có chức năng hiển thị các khung hình lên giao diện (thuộc View), lại vừa cung cấp các tính năng điều khiển (thuộc Controller), cũng đồng thời lưu giữ các trạng thái, ví dụ dừng/phát, để cập nhật giao diện (thuộc Model).

4.4.Kết luận chương

Các kết quả đánh giá của Chương 3 có thể tóm tắt lại như sau:

Về đánh giá mức độ hoàn thiện chức năng: 9/10 yêu cầu chức năng đạt mức độ hoàn thiện, với 45/49 kịch bản kiểm thử đạt.

Về đánh giá mức độ tuân thủ mô hình MVC: Tuân thủ mức độ trung bình

Kết luận

Qua quá trình thực hiện đề tài, với nỗ lực của bản thân và sự hướng dẫn tận tình của giảng viên hướng dẫn – TS. Võ Lê Cường, em đã hoàn thành đề tài với nội dung: Thiết kế và phát triển phần mềm ứng dụng đa phương tiện, bao gồm cả thiết kế chi tiết và sản phẩm thực tế.

Để thực hiện đề tài, em đã vận dụng nhiều kỹ năng: tìm hiểu và đánh giá khả năng của framework, phân tích yêu cầu, thiết kế, lập trình, tự xem xét và đánh giá sản phẩm. Đây đều là các kỹ năng cần thiết cho công việc thực tế, đặc biệt đối với công việc em đang hướng tới là kỹ sư phát triển phần mềm.

Tuy đề tài không mới, nhưng em tự đánh giá là phù hợp với mục tiêu rèn dũa kỹ năng. Các yêu cầu đặt ra đều hầu như đều đạt ở mức độ cao. Việc áp dụng mô hình thiết kế giúp đỡ rất nhiều cho việc phát triển tính năng và bảo trì khi có lỗi xảy ra.

Tuy nhiên, do hiểu biết và kỹ năng có hạn, sản phẩm của em chắc chắn còn nhiều thiếu sót. Tuy nhiên em tin rằng sản phẩm có thể tiếp tục phát triển, tối ưu hóa cả về giao diện và tính năng, hoàn thiện trải nghiệm người dùng. Từ đó trở thành ứng dụng có thể thực sự phù hợp cho việc sử dụng hàng ngày.

Em xin chân thành cảm ơn sự giúp đỡ tận tình của giáo viên hướng dẫn – TS. Võ Lê Cường đã giúp đỡ em hoàn thành đề tài đồ án tốt nghiệp này.

Hà Nội, ngày 26 tháng 12 năm 2020

Sinh viên thực hiện

Đỗ Trung Hiếu

Tài liệu tham khảo

- [1] Rassol Raissi. *The Theory Behind Mp3* [Online]. Available at: <https://ceng460.cankaya.edu.tr/>
- [2] https://en.wikipedia.org/wiki/MPEG-4_Part_14, truy cập cuối cùng ngày 16/1/2021
- [3] <https://resources.qt.io>, truy nhập cuối cùng ngày 24/12/2020
- [4] <https://doc.qt.io/qt-5/signalsandslots.html>, truy nhập cuối cùng ngày 24/12/2020
- [5] Glenn E. Krasner, Stephen T. Pope, “A cookbook for using the model-view controller user interface paradigm in Smalltalk-80”, *Journal of Object-Oriented Programming*, vol. 1, no. 3, pp. 26-49, 1988
- [6] Kevin McArthur, *Pro PHP: Patterns, Frameworks, Testing and More*, Apress, 2008.
- [7] Adam Freeman, Steven Sanderson, *Pro ASP.NET MVC 3 Framework*, Apress, 2011.
- [8] W. J. Gilmore, *Easy PHP Websites*, Columbus, Ohio: W.J. Gilmore, LLC, 2009.

Bảng đối chiếu thuật ngữ Việt - Anh

<i>Qt Framework</i>	Bộ khung phần mềm Qt
<i>Model-View-Controller (MVC)</i>	Mô hình thiết kế phần mềm Model-View-Controller
<i>Operating System (OS)</i>	Hệ điều hành
<i>Real-Time Operating System (RTOS)</i>	Hệ điều hành thời gian thực
<i>Application Programming Interface (API)</i>	Giao diện lập trình ứng dụng
<i>Graphical User Interface (GUI)</i>	Giao diện đồ họa người dùng
<i>signal</i>	(Qt) tín hiệu
<i>slot</i>	(Qt) khe xử lý, method được kết nối với signal
<i>Qt Modelling Language (QML)</i>	(Qt) Ngôn ngữ mô tả QML
<i>Meta-Object Compiler</i>	Công cụ biên dịch MOC của Qt
<i>compile</i>	Biên dịch mã nguồn
<i>compiler</i>	Trình biên dịch mã nguồn
<i>Object-Oriented Programming (OOP)</i>	Lập trình hướng đối tượng
<i>Public</i>	(OOP) Chỉ định truy xuất Public
<i>Private</i>	(OOP) Chỉ định truy xuất Private
<i>Protected</i>	(OOP) Chỉ định truy xuất Protected
<i>Random Access Memory (RAM)</i>	Bộ nhớ truy cập ngẫu nhiên
<i>Central Processing Unit (CPU)</i>	Bộ xử lý trung tâm
<i>Aliasing</i>	Chồng phỏng
<i>Inverse Modified Discrete Cosine Transform</i>	Biến đổi Cosine rời rạc chỉnh sửa nghịch đảo

PHỤ LỤC

A. Chi tiết các class

Chi tiết về class MyMediaApp:

- Tên: MyMediaApp
- Header: mymediaapp.h
- Source: mymediaapp.cpp
- Danh sách biến:
 - m_allAudios
 - Phạm vi: Private (nội bộ)
 - Kiểu dữ liệu: PlaylistModel*
 - Mô tả chi tiết: Danh sách phát nội dung audio
 - m_allVideos
 - Phạm vi: Private
 - Kiểu dữ liệu: VideoPlaylistModel*
 - Mô tả chi tiết: Danh sách phát nội dung video
 - m_audioPlayer
 - Phạm vi: Private
 - Kiểu dữ liệu: MyAudioPlayer*
 - Mô tả chi tiết: Khối điều khiển phát file audio
 - m_mediaPlaybackInfo
 - Phạm vi: Private
 - Kiểu dữ liệu: MediaPlaybackInfo*
 - Mô tả chi tiết: Thông tin về các nội dung được phát gần nhất (audio và video)
- Danh sách các phương thức:
 - *MyMediaApp()* - construtor - Phạm vi: Public
 - Khởi tạo m_audioPlayer
 - Khởi tạo m_mediaPlaybackInfo
 - Khởi tạo m_allAudios
 - Khởi tạo m_allVideos
 - Kết nối signal m_audioPlayer:: nowPlayingIndexChanged() và slot updateLastAudio()
 - Đọc thông tin các file được phát gần nhất

- Quét các file audio, video và thêm vào danh sách phát bằng method `scanMedias()`
- `addToAudioPlaylist()` - Phạm vi: Private - Kiểu trả về: void
 - Nhận vào danh sách đường dẫn các file mp3
 - Duyệt lần lượt từng đường dẫn
 - Tìm thông tin tên bài hát trong meta-data của file. Nếu thông tin tên bài hát tồn tại, sử dụng làm tiêu đề, nếu không, sử dụng tên file làm tiêu đề
 - Tìm thông tin nghệ sĩ thể hiện trong meta-data của file. Nếu tồn tại, sử dụng làm thông tin nghệ sĩ, nếu không, sử dụng “Unknown Artist” làm thông tin nghệ sĩ
 - Nhận đường dẫn tới hình album trong meta-data của file bằng phương thức `getAlbumArt()`
 - Thêm đối tượng Song mới với các thông tin đường dẫn, tiêu đề, nghệ sĩ thể hiện và đường dẫn tới hình album vào `m_allAudios`
 - Từ `m_allAudios`, thiết lập danh sách phát cho `m_audioPlayer`
 - Thiết lập bài hát được phát trước đó từ thông tin đọc được từ `m_mediaPlaybackInfo`
- `addToVideoPlaylist()` - Phạm vi: Private - Kiểu trả về: void
 - Nhận vào danh sách đường dẫn các file mp4
 - Tách tên file từ đường dẫn làm tiêu đề cho nội dung
 - Tạo đối tượng Video mới từ thông tin đường dẫn và tên file
 - Thêm đối tượng video vừa tạo vào `m_allVideos`
- `allAudios()` - Phạm vi: Public - Kiểu trả về: `PlaylistModel*`
 - Trả về `m_allAudios`
- `allVideos()` - Phạm vi: Public - Kiểu trả về: `VideoPlaylistModel*`
 - Trả về `m_allVideos`
- `audioPlayer()` - Phạm vi: Public - Kiểu trả về: `MyAudioPlayer*`
 - Trả về `m_audioPlayers`
- `getAlbumArt()` - Phạm vi: Private - Kiểu trả về: `QString`
 - Nhận vào đường dẫn tới file mp3
 - Truy cập vào meta-data của file

- Nếu meta-data có tồn tại hình album, tạo một file ảnh định dạng jpg dựa vào thông tin này
 - Trả về đường dẫn tới file vừa tạo
- *mediaPlaybackInfo()* - Phạm vi: Public - Kiểu trả về: *MediaPlaybackInfo**
 - Trả về *m_mediaPlaybackInfo*
- *readMediaPlaybackInfoFromFile()* - Phạm vi: Private - Kiểu trả về: void
 - Đọc file JSON chứa các thông tin: audio được phát gần nhất, 2 video được phát gần nhất, thời lượng đã phát của video được phát gần nhất
 - Truyền các thông tin này vào *m_mediaPlaybackInfo*
- *scanMedias()* - Phạm vi: Private - Kiểu trả về: void
 - Duyệt tìm và lén danh sách toàn bộ các file mp3 và mp4 trong hệ thống
 - Gọi *addToAudioPlaylist()* và truyền vào danh sách các file mp3 để tạo dữ liệu cho *m_allAudios*
 - Gọi *addToVideoPlaylist()* và truyền vào danh sách các file mp4 để tạo dữ liệu cho *m_allVideos*
- *updateLastAudio()* - Phạm vi: Public - Kiểu trả về: void
 - Cập nhật thông tin audio được phát gần nhất trong *m_mediaPlaybackInfo*
- *writePlaybackInfoToFile()* - Phạm vi: Public - Kiểu trả về: void
 - Lưu thông tin hiện tại trong *m_mediaPlaybackInfo* vào file JSON

Chi tiết về class MyAudioPlayer:

- Tên: MyAudioPlayer
- Header: myaudioplayer.h
- Source: myaudioplayer.cpp
- Danh sách biến:
 - *m_duration* - Phạm vi: Private
 - Kiểu dữ liệu: qint64
 - Mô tả: thời lượng của nội dung đang được chọn, đơn vị mili giây

- m_isPlaying - Phạm vi: Private
 - Kiểu dữ liệu: bool
 - Mô tả: trạng thái phát / không phát
- m_mediaAvailable - Phạm vi: Private
 - Kiểu dữ liệu: bool
 - Mô tả: trạng thái có / không có sẵn nội dung
- m_nowPlayingIndex - Phạm vi: Private
 - Kiểu dữ liệu: int
 - Mô tả: chỉ số của nội dung đang phát trong danh sách phát
- m_originalPlaylist - Phạm vi: Private
 - Kiểu dữ liệu: PlaylistModel*
 - Mô tả: danh sách phát bao gồm các thông tin tiêu đề, nghệ sĩ, ...
- m_playbackMode - Phạm vi: Private
 - Kiểu dữ liệu: PlaybackMode
 - Mô tả: chế độ phát lại đang được áp dụng
- m_player - Phạm vi: Private
 - Kiểu dữ liệu: QMediaPlayer*
 - Mô tả: thực thể QMediaPlayer, cung cấp các phương thức điều khiển
- m_playlist - Phạm vi: Private
 - Kiểu dữ liệu: QMediaPlaylist*
 - Mô tả: thực thể QMediaPlaylist được tạo từ danh sách phát m_originalPlaylist, chứa các nội dung hợp lệ cho m_player
- m_position - Phạm vi: Private
 - Kiểu dữ liệu: qint64
 - Mô tả: thời lượng đã phát, đơn vị mili giây
- Danh sách các phương thức:
 - *MyAudioPlayer()* - constructor - Phạm vi: Public
 - Khởi tạo m_player
 - Khởi tạo m_playlist
 - Thiết lập m_playlist thành danh sách phát của m_player với method QMediaPlayer::setPlaylist()
 - Kết nối signal m_player::positionChanged với slot *setPosition()*
 - Kết nối signal m_player::durationChanged với slot *setDuration()*

- Kết nối signal `m_player::currentMediaChanged` với slot `mediaChangedHandler()`
- `duration()` - Phạm vi: Public - Kiểu trả về: `qint64`
 - Trả về giá trị hiện tại của `m_duration`
- `getTimeString()` - Phạm vi: Public - Kiểu trả về: `QString`
 - Nhận vào giá trị thời gian đơn vị mili giây
 - Trả về chuỗi với định dạng “hh:mm:ss” (giờ:phút:giây)
- `isPlaying()` - Phạm vi: Public - Kiểu trả về: `bool`
 - Trả về giá trị hiện tại của `m_isPlaying`
- `loadPlaylist()` - Phạm vi: Private - Kiểu trả về: `void`
 - Tạo dữ liệu cho `m_playlist` dựa vào dữ liệu đang có trong `m_originalPlaylist`
 - Với mỗi đối tượng `Song` trong `m_originalPlaylist`, thêm một nội dung tương ứng dựa trên đường dẫn trong `Song` vào `m_playlist` với method `QMediaPlaylist::addMedia()`
- `mediaAvailable()` - Phạm vi: Public - Kiểu trả về: `bool`
 - Trả về giá trị hiện tại của `m_mediaAvailable`
- `mediaChangedHandler()` - Phạm vi: Public - Kiểu trả về: `void`
 - Xử lý khi nội dung đang phát thay đổi.
 - Dựa vào chênh lệch giữa `m_nowPlayingIndex` và chỉ số hiện tại của `m_playlist`, phán đoán xem nội dung mới là nội dung xếp trước hay sau so với nội dung cũ.
 - Phát đi signal `nexted()` báo hiệu nội dung đã thay đổi, kèm theo thông tin xếp trước/sau
 - Cập nhật `m_nowPlayingIndex` đồng nhất với chỉ số hiện tại của `m_playlist`
 - Phát đi signal `nowPlayingIndexChanged()` kèm theo giá trị mới của `m_nowPlayingIndex`
 - Cập nhật `m.isPlaying`
- `next()` - Phạm vi: Public - Kiểu trả về: `void`
 - Nếu trong danh sách phát không có nội dung nào, thoát khỏi method
 - Dừng nội dung đang phát
 - Nếu `m_playbackMode` đang là `REPEAT_ONE`, cập nhật sang `REPEAT_ALL`
 - Nếu nội dung đang phát là nội dung cuối trong danh sách, phát nội dung đầu tiên trong danh sách

- Nếu nội dung đang phát không phải nội dung cuối, phát nội dung tiếp theo
- *nowPlayingIndex()* - Phạm vi: Public - Kiểu trả về: int
 - Trả về giá trị hiện tại của m_nowPlayingIndex
- *playbackMode()* - Phạm vi: Public - Kiểu trả về: int
 - Trả về giá trị hiện tại của m_playbackMode
- *position()* - Phạm vi: Public - Kiểu trả về: qint64
 - Trả về giá trị hiện tại của m_position
- *previous()* - Phạm vi: Public - Kiểu trả về: void
 - Nếu trong danh sách phát không có nội dung nào, thoát khỏi method
 - Dừng nội dung đang phát
 - Nếu m_playbackMode đang là REPEAT_ONE, cập nhật sang REPEAT_ALL
 - Nếu nội dung đang phát là nội dung đầu tiên trong danh sách và đã phát ít hơn 3 giây, chuyển về nội dung cuối cùng trong danh sách
 - Nếu nội dung đang phát không phatr là nội dung đầu tiên trong danh sách và đã phát ít hơn 3 giây, chuyển về nội dung trước đó trong danh sách
 - Nếu nội dung đang phát đã phát nhiều hơn 3 giây, phát lại nội dung từ đầu
- *setDuration()* - Phạm vi: Private - Kiểu trả về: void
 - Cập nhật giá trị của m_duration
 - Phát đi signal *durationChanged()*
- *setIsPlaying()* - Phạm vi: Private - Kiểu trả về: void
 - Cập nhật giá trị của m_isPlaying
 - Phát đi signal *isPlayingChanged()*
- *setMedia()* - Phạm vi: Public - Kiểu trả về: void
 - Nếu trong danh sách phát không có nội dung nào, thoát khỏi method
 - Dừng nội dung đang phát
 - Nếu m_playbackMode đang là REPEAT_ONE, cập nhật sang REPEAT_ALL
 - Phát nội dung với chỉ số được truyền vào
 - Cập nhật m_duration với *setDuration()*
 - Cập nhật m_position với *setPosition()*
 - Cập nhật m_isPlayingState

- Cập nhật m_nowPlayingIndex đồng nhất với chỉ số được truyền vào
- *setMediaAvailable()* - Phạm vi: Private - Kiểu trả về: void
 - Cập nhật giá trị của m_mediaAvailable
 - Phát đi signal *mediaAvailableChanged()*
- *setOriginalPlaylist()* - Phạm vi: Public - Kiểu trả về: void
 - Cập nhật m_originalPlaylist bằng dữ liệu được truyền vào
 - Tạo dữ liệu cho m_player bằng cách gọi method *loadPlaylist()*
- *setPlaybackMode()* - Phạm vi: Private - Kiểu trả về: void
 - Cập nhật giá trị của m_playbackMode
 - Phát đi signal *playbackModeChanged()*
- *setPosition()* - Phạm vi: Private - Kiểu trả về: void
 - Cập nhật giá trị của m_position
 - Phát đi signal *positionChanged()*
- *setPositionByPercent()* - Phạm vi: Public - Kiểu trả về: void
 - Phát nội dung từ vị trí tính theo tỉ lệ phần trăm so với tổng thời lượng
- *stopMedia()* - Phạm vi: Public - Kiểu trả về: void
 - Dừng phát nội dung
- *switchPlaybackMode()* - Phạm vi: Public - Kiểu trả về: void
 - Thay đổi m_playbackMode theo giá trị hiện tại của m_playbackMode
 - Nếu m_playbackMode đang là SEQUENCE thì chuyển thành REPEAT_ALL, cập nhật chế độ phát lại của m_playlist thành QMediaPlaylist::Loop
 - Nếu m_playbackMode đang là REPEAT_ALL thì chuyển thành REPEAT_ONE, cập nhật chế độ phát lại của m_playlist thành QMediaPlaylist::CurrentItemInLoop
 - Nếu m_playbackMode đang là REPEAT_ONE thì chuyển thành SEQUENCE, cập nhật chế độ phát lại của m_playlist thành QMediaPlaylist::Sequential
- *togglePlay()* - Phạm vi: Public - Kiểu trả về: void
 - Lấy trạng thái phát hiện tại của m_player qua method QMediaPlayer::state()

- Nếu trạng thái nhận được là PausedState hoặc StoppedState, bắt đầu phát nội dung bằng method QMediaPlayer::play()
 - Nếu trạng thái nhận được là PlayingState, tạm dừng phát nội dung bằng method QMediaPlayer::pause()
 - Cập nhật giá trị của m_isPlaying tương ứng
- Danh sách các signal:
 - durationChanged() - Phạm vi: Public
 - Giá trị của m_duration đã thay đổi
 - isPlayingChanged() - Phạm vi: Public
 - Giá trị của m_isPlaying đã thay đổi
 - mediaAvailableChanged() - Phạm vi: Public
 - Giá trị của m_mediaAvailable đã thay đổi
 - nowPlayingIndexChanged() - Phạm vi: Public
 - Giá trị của m_nowPlayingIndex đã thay đổi
 - playbackModeChanged() - Phạm vi: Public
 - Giá trị của m_playbackMode đã thay đổi
 - positionChanged() - Phạm vi: Public
 - Giá trị của m_position đã thay đổi
 - nexted() - Phạm vi: Public
 - Đã chuyển sang nội dung tiếp theo

Chi tiết class MediaPlaybackInfo:

- Tên: MediaPlaybackInfo
- Header: medioplaybackinfo.h
- Source: medioplaybackinfo.cpp
- Danh sách biến:
 - m_last AudioSource - Phạm vi: Private
 - Kiểu dữ liệu: QString
 - Mô tả: đường dẫn đến file audio được phát gần nhất
 - m_lastVideoName - Phạm vi: Private
 - Kiểu dữ liệu: QString
 - Mô tả: tên file video được phát gần nhất
 - m_lastVideoPosition - Phạm vi: Private
 - Kiểu dữ liệu: quint8
 - Mô tả: thời lượng đã phát của video được phát gần nhất

- m_lastVideoSource - Phạm vi: Private
 - Kiểu dữ liệu: QString
 - Mô tả: đường dẫn đến file video được phát gần nhất
- m_secLastVideoName - Phạm vi: Private
 - Kiểu dữ liệu: QString
 - Mô tả: tên file video được phát gần thứ 2
- m_secLastVideoSource - Phạm vi: Private
 - Kiểu dữ liệu: QString
 - Mô tả: đường dẫn đến file video được phát gần thứ 2
- Danh sách các method:
 - *MediaPlaybackInfo()* - Phạm vi: Public - constructor
 - Khởi tạo các biến của class theo các giá trị được truyền vào
 - *last AudioSource()* - Phạm vi: Public - Kiểu trả về: QString
 - Trả về giá trị của m_last AudioSource
 - *last Video Name()* - Phạm vi: Public - Kiểu trả về: QString
 - Trả về giá trị của m_last Video Name
 - *last Video Position()* - Phạm vi: Public - Kiểu trả về: quint64
 - Trả về giá trị của m_last Video Position
 - *last Video Source()* - Phạm vi: Public - Kiểu trả về: QString
 - Trả về giá trị của m_last Video Source
 - *sec Last Video Name()* - Phạm vi: Public - Kiểu trả về: QString
 - Trả về giá trị của m_sec Last Video Name
 - *sec Last Video Source()* - Phạm vi: Public - Kiểu trả về: QString
 - Trả về giá trị của m_sec Last Video Source
 - *set Last Audio()* - Phạm vi: Public - Kiểu trả về: void
 - Thay đổi giá trị của m_last Audio
 - *set Last Video Position()* - Phạm vi: Public - Kiểu trả về: void
 - Thay đổi giá trị của m_last Video Position
 - *set Last Video Source()* - Phạm vi: Public - Kiểu trả về: void
 - Thay đổi giá trị của m_last Video Source
 - *set Sec Last Video Source()* - Phạm vi: Public - Kiểu trả về: void
 - Thay đổi giá trị của m_sec Last Video Source
 - *update Last Video()* - Phạm vi: Public - Kiểu trả về: void
 - Thay đổi giá trị hiện tại của m_sec Last Video Name bằng với giá trị hiện tại của m_last Video Name
 - Thay đổi giá trị hiện tại của m_sec Last Video Source bằng với giá trị hiện tại của m_last Video Source

- Thay đổi giá trị hiện tại của `m_lastVideoName` thành tên của video mới được phát gần nhất
- Thay đổi giá trị hiện tại của `m_lastVideoSource` thành đường dẫn của video mới được phát gần nhất

Chi tiết về class PlaylistMode:

- Tên: PlaylistModel
- Header: playlistmodel.h
- Source: playlistmodel.cpp
- Danh sách biến:
 - `m_data` - Phạm vi: Private
 - Kiểu dữ liệu: `QList<Song>`
 - Mô tả: danh sách các đối tượng thuộc class Song, chứa thông tin về nội dung audio
- Danh sách các method:
 - `PlaylistModel()` - Phạm vi: Public - constructor
 - `addSong()` - Phạm vi: Public - Kiểu trả về: void
 - Thêm một đối tượng thuộc class Song vào `m_data`
 - `clearData()` - Phạm vi: Public - Kiểu trả về: void
 - Xóa bỏ toàn bộ các đối tượng Song hiện có trong `m_data`
 - `data()` - Phạm vi: Public - Kiểu trả về: QVariant
 - Method được triển khai theo hướng dẫn của Qt để QML có thể truy cập vào `m_data` và lấy thông tin để hiển thị
 - `roleNames()` - Phạm vi: Protected - Kiểu trả về: QHash<int, QByteArray>
 - Method được triển khai theo hướng dẫn của Qt để QML có thể truy cập vào `m_data` và lấy thông tin để hiển thị
 - `rowCount()` - Phạm vi: Public - Kiểu trả về: int
 - Số lượng đối tượng hiện có trong danh sách `m_data`

Chi tiết về class Song:

- Tên: Song
- Header: song.h
- Source: song.cpp
- Danh sách biến:

- m_albumArt - Phạm vi: Private
 - Kiểu dữ liệu: QString
 - Mô tả: đường dẫn tới file hình ảnh album
- m_singer - Phạm vi: Private
 - Kiểu dữ liệu: QString
 - Mô tả: thông tin nghệ sĩ thể hiện
- m_source - Phạm vi: Private
 - Kiểu dữ liệu: QString
 - Mô tả: đường dẫn tới file audio
- m_title - Phạm vi: Private
 - Kiểu dữ liệu: QString
 - Mô tả: thông tin tên bài hát
- Danh sách các method:
 - Song() - Phạm vi: Public - constructor
 - Khởi tạo các biến của class theo các giá trị được truyền vào
 - albumArt() - Phạm vi: Public - Kiểu trả về: QString
 - Trả về giá trị hiện tại của m_albumArt
 - singer() - Phạm vi: Public - Kiểu trả về: QString
 - Trả về giá trị hiện tại của m_singer
 - source() - Phạm vi: Public - Kiểu trả về: QString
 - Trả về giá trị hiện tại của m_source
 - title() - Phạm vi: Public - Kiểu trả về: QString
 - Trả về giá trị hiện tại của m_title

Chi tiết về class Video:

- Tên: Video
- Header: video.h
- Source: video.cpp
- Danh sách biến:
 - m_title - Phạm vi: Private
 - Kiểu dữ liệu: QString
 - Mô tả: tên file video
 - m_videoSrc - Phạm vi: Private
 - Kiểu dữ liệu: QString
 - Mô tả: đường dẫn đến file video

- Danh sách các method:
 - *Video()* - Phạm vi: Public - constructor
 - Khởi tạo các biến của class theo các giá trị được truyền vào
 - *title()* - Phạm vi: Public - Kiểu trả về: QString
 - Trả về giá trị của m_title
 - *videoSrc()* - Phạm vi: Public - Kiểu trả về: QString
 - Trả về giá trị của m_videoSrc

Chi tiết về class VideoPlaylistModel:

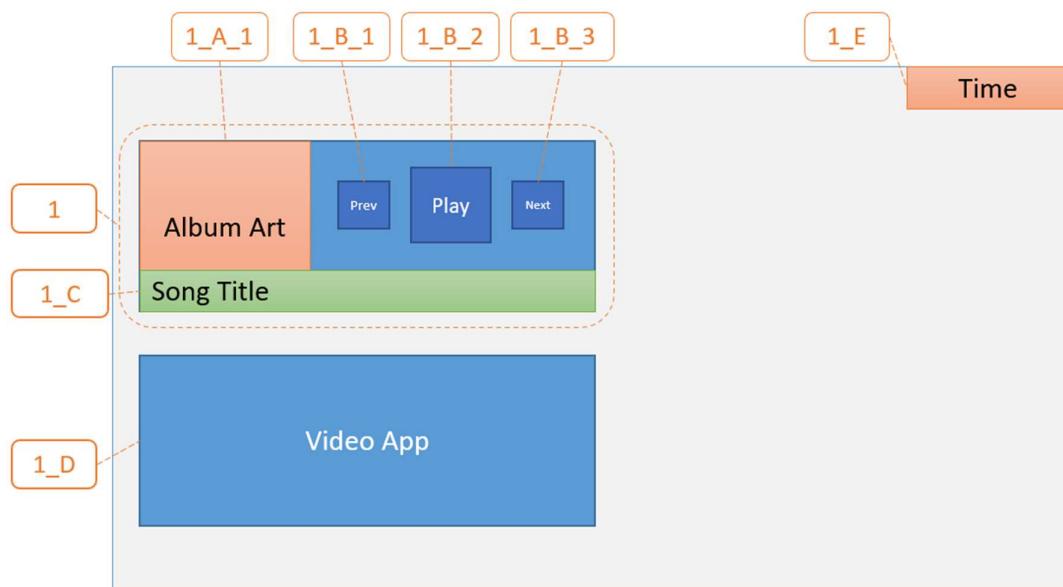
- Tên: VideoPlaylistModel
- Header: videoplaylistmodel.h
- Source: videoplaylistmodel.cpp
- Danh sách biến:
 - m_data - Phạm vi: Private
 - Kiểu dữ liệu: QList<Video>
 - Mô tả: danh sách các đối tượng thuộc class Video, chứa thông tin về nội dung video
 - m_nowPlayingIndex - Phạm vi: Private
 - Kiểu dữ liệu: int
 - Mô tả: chỉ số của nội dung đang phát trong danh sách phát
- Danh sách các method:
 - *VideoPlaylistModel()* - Phạm vi: Public - constructor
 - *addVideo()* - Phạm vi: Public - Kiểu trả về: void
 - Thêm một đối tượng thuộc class Video vào m_data
 - *clearData()* - Phạm vi: Public - Kiểu trả về: void
 - Xóa bỏ toàn bộ các đối tượng Video hiện có trong m_data
 - *data()* - Phạm vi: Public - Kiểu trả về: QVariant
 - Method được triển khai theo hướng dẫn của Qt để QML có thể truy cập vào m_data và lấy thông tin để hiển thị
 - *nowPlayingIndex()* - Phạm vi: Public - Kiểu trả về: int
 - Trả về giá trị của m_nowPlayingIndex
 - *roleNames()* - Phạm vi: Protected - Kiểu trả về: QHash<int, QByteArray>
 - Method được triển khai theo hướng dẫn của Qt để QML có thể truy cập vào m_data và lấy thông tin để hiển thị

- *rowCount()* - Phạm vi: Public - Kiểu trả về: int
 - Số lượng đối tượng hiện có trong danh sách *m_data*
- *setNowPlayingIndex()* - Phạm vi: Public - Kiểu trả về: void
 - Thiết lập giá trị của *m_nowPlayingIndex*
- Danh sách signal:
 - *nowPlayingIndexChanged()* - Phạm vi: Public - Kiểu trả về: void
 - *m_nowPlayingIndex* đã thay đổi

B. Chi tiết các màn hình

1. Chi tiết bộ cục giao diện

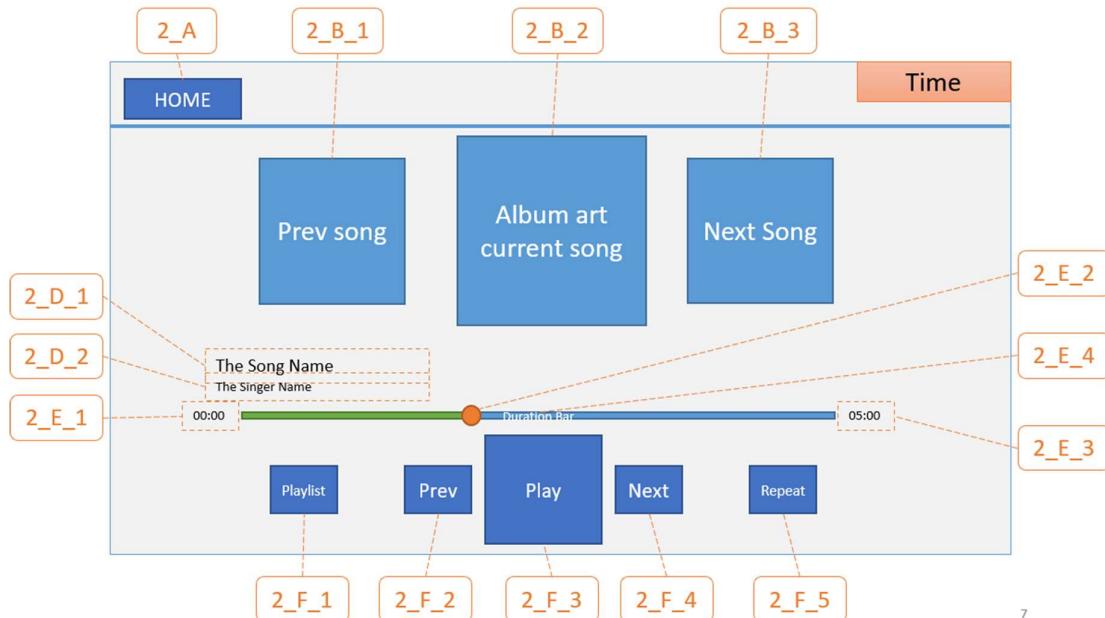
Màn hình Home:



- 1: Khu vực điều khiển thu nhỏ cho chức năng phát file audio app
 - 1_A_1: Hình album thu nhỏ của nội dung audio đang phát. Khi người dùng nhấn vào
 - 1_B_1: Nút điều khiển chuyển về nội dung trước
 - 1_B_2: Nút điều khiển dừng / phát
 - Hiển thị nút bấm bắt đầu phát khi nội dung đang ngừng / tạm ngừng phát
 - Hiện thị nút bấm tạm dừng khi nội dung đang phát
 - 1_B_3: Nút bấm điều khiển chuyển sang nội dung tiếp theo
 - 1_C: Tiêu đề của nội dung đang phát, khi người dùng nhấn vào
- 1_D: Nút bấm điều khiển mở chức năng phát file video

- 1_E: Khu vực hiển thị thời gian hiện tại (định dạng hh:mm)

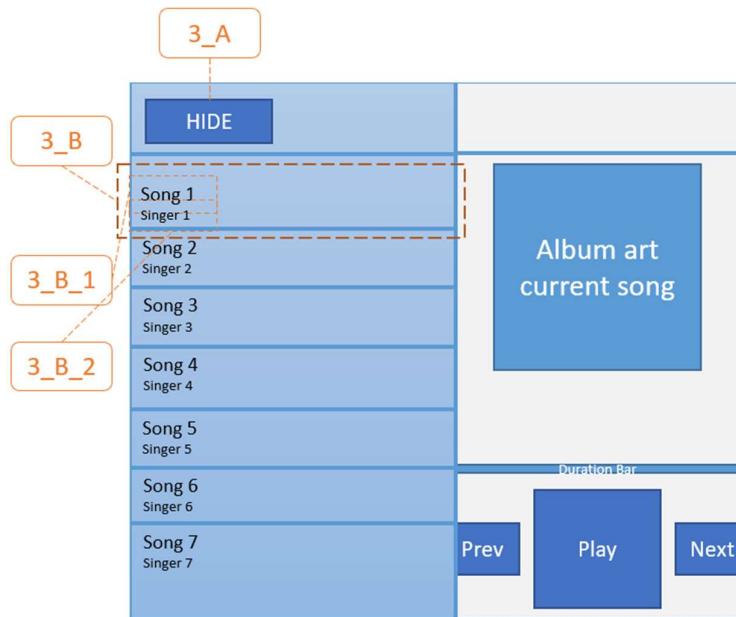
Màn hình Audio – Player:



- 2_A: Nút HOME
- 2_B_1: Hình album của nội dung trước trong danh sách
- 2_B_2: Hình album của nội dung đang phát
- 2_B_3: Hình album của nội dung tiếp theo trong danh sách
- 2_D_1: Tiêu đề của nội dung đang phát
- 2_D_2: Tên nghệ sĩ thể hiện nội dung đang phát
- 2_E_1: Thời lượng đã phát
- 2_E_2: Chỉ báo thời lượng đã phát so với tổng thời lượng
- 2_E_3: Thời lượng của nội dung đang phát
- 2_E_4: Thanh trượt chỉ báo và điều chỉnh vị trí phát
- 2_F_1: Nút bấm mở Playlist
- 2_F_2: Nút bấm chuyển về nội dung trước đó
- 2_F_3: Nút bấm điều khiển dừng / phát
 - Hiển thị biểu tượng bắt đầu phát khi nội dung đang ngừng / tạm ngừng phát
 - Hiển thị biểu tượng tạm dừng khi nội dung đang phát
- 2_F_5: Nút bấm điều khiển chế độ lặp

- Hiển thị biểu tượng không lặp khi chế độ phát lại đang áp dụng là không lặp
- Hiển thị biểu tượng lặp lại toàn bộ khi chế độ phát lại đang áp dụng là lặp lại toàn bộ
- Hiển thị biểu tượng lặp lại một bài khi chế độ phát lại đang áp dụng là lặp lại một bài

Màn hình Audio – Playlist:

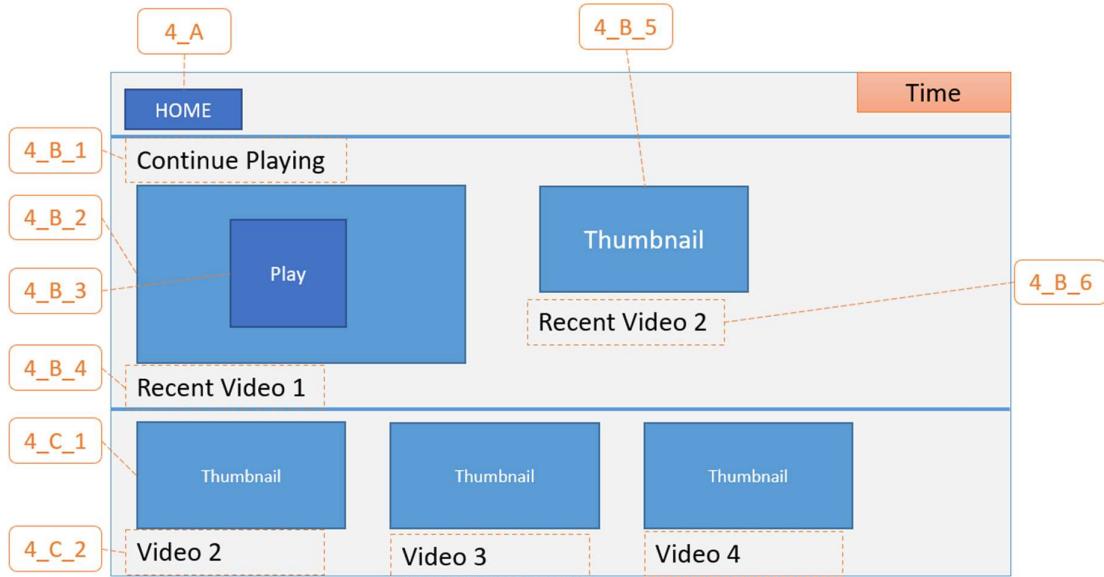


Hình 4.12 Màn hình Audio - Playlist

- 3_A: Nút HIDE
- 3_B: Một nội dung trong danh sách phát file audio
- 3_B_1: Tiêu đề của nội dung audio
- 3_B_2: Nghệ sĩ thể hiện nội dung audio

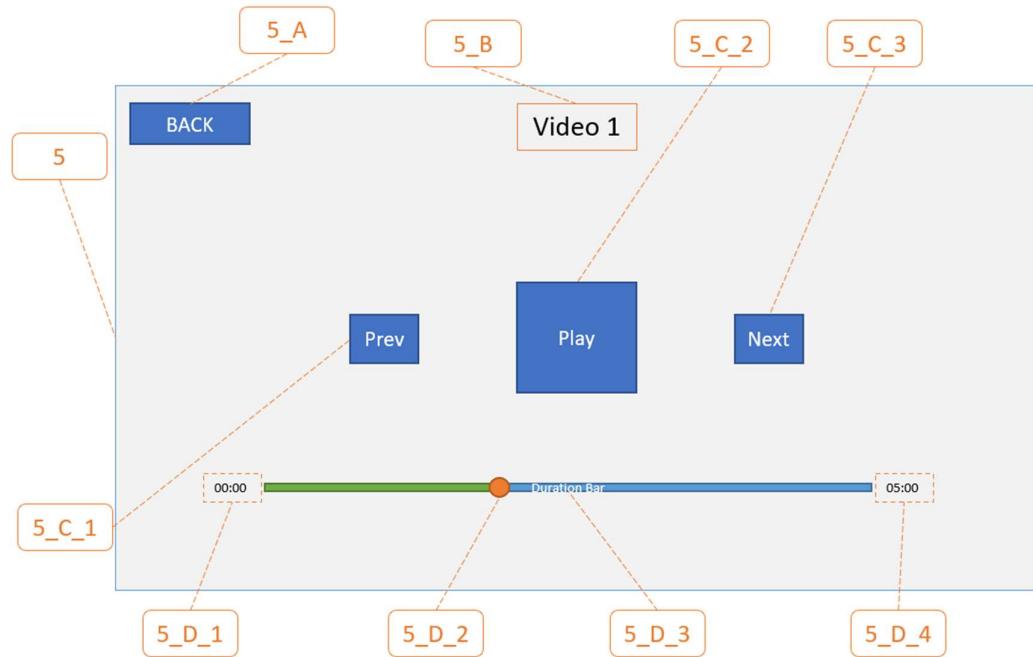
Thực tế trong quá trình phát triển, Audio – Playlist không được xếp thành màn hình riêng mà chỉ là một component của Audio – Player.

Màn hình Video – Playlist:



- **4_A**: nút HOME
- **4_B_1**: Nhãn hiển thị khu vực lịch sử phát. Luôn hiển thị “Continue Playing”
- **4_B_2**: Khung hình cuối cùng được phát của video được phát gần nhất
- **4_B_3**: Nút bấm tiếp tục phát file video gần nhất
- **4_B_4**: Tiêu đề của video được phát gần nhất
 - Nếu thông tin về video được phát gần nhất không tồn tại, hiển thị “No recently played”
- **4_B_5**: Hình đại diện mặc định cho video
- **4_B_6**: Tiêu đề của video được phát gần thứ 2
- **4_C_1**: Hình đại diện mặc định của video
- **4_C_2**: Tiêu đề của video trong danh sách phát

Màn hình Video – Player:



- 5: Khu vực phát file video, chiếm toàn bộ khu vực hiển thị, hỗ trợ bởi VideoQML (1.4.4)
- 5_A: Nút BACK
- 5_B: Tiêu đề của video đang phát
- 5_C_1: Nút bấm chuyển về video trước
- 5_C_2: Nút bấm điều khiển dừng / phát file video
 - Hiển thị biểu tượng bắt đầu phát khi nội dung đang ngừng / tạm ngừng phát
 - Hiện thị biểu tượng tạm dừng khi nội dung đang phát
- 5_C_3: Nút bấm chuyển sang nội dung tiếp theo
- 5_D_1: Thời lượng đã phát
- 5_D_2: Chỉ báo thời lượng đã phát so với thời lượng của nội dung
- 5_D_3: Thanh trượt điều khiển vị trí phát
- 5_D_4: Thời lượng của video đang phát

2. Chi tiết hành vi của các màn hình

Màn hình Home:

- Khởi động cùng ứng dụng
- Chuyển sang màn hình Audio - Player khi người dùng nhấn vào 1_A_1 hoặc 1_C

- Chuyển sang màn hình Video - Playlist khi người dùng nhấn vào 1_D
- Thực thi slot MyMediaPlayer::previous() khi người dùng nhấn vào 1_B_1
- Thực thi slot MyMediaPlayer::togglePlay() khi người dùng nhấn vào 1_B_2
- Thực thi slot MyMediaPlayer::next() khi người dùng nhấn vào 1_B_3
- Tự động cập nhật 1_A_1 và 1_C khi signal MyMediaPlayer::nowPlayingIndexChanged() được phát đi
- Thực thi slot MyMediaApp::writePlaybackInfoToFile() khi có signal destruction() được phát đi

Màn hình Audio – Player:

- Chuyển đến màn hình Home khi người dùng nhấn vào 2_A
- Chuyển đến màn hình Audio - Playlist khi người dùng nhấn vào 2_F_1
- Thực thi slot MyMediaPlayer::previous() khi người dùng nhấn vào 2_F_2
- Thực thi slot MyMediaPlayer::togglePlay() khi người dùng nhấn vào 2_F_3
- Thực thi slot MyMediaPlayer::next() khi người dùng nhấn vào 2_F_4
- Thực thi slot MyMediaPlayer::switchPlaybackMode() khi người dùng nhấn vào 2_F_5
- Thực thi slot MyMediaPlayer::setPositionByPercent() khi người dùng nhấn vào 2_E_4
- Tự động cập nhật 2_B_1, 2_B_2, 2_B_3 khi signal MyMediaPlayer::nexted() được phát đi
- Tự động cập nhật 2_D_1, 2_D_2 khi signal MyMediaPlayer::nowPlayingIndexChanged() được phát đi
- Tự động cập nhật 2_E_1 và vị trí của 2_E_2 khi signal MyMediaPlayer::positionChanged() được phát đi
- Tự động cập nhật 2_E_3 khi signal MyMediaPlayer::durationChanged() được phát đi

Màn hình Audio – Playlist:

- Chuyển đến màn hình Audio - Player khi người dùng nhấn vào 3_A
- Thực thi slot MyMediaPlayer::setMedia() khi người dùng nhấn vào một nội dung trong danh sách phát

Màn hình Video – Playlist:

- Chuyển đến màn hình Home khi người dùng nhấn vào 4_A
- Gửi yêu cầu phát file video kèm theo đường dẫn của video được chọn đến VideoQML khi người dùng nhấn vào 4_B_3, hoặc 4_B_5, hoặc 4_C_1, sau đó chuyển đến màn hình Video - Player

Màn hình Video - Player:

- Sau 5 giây không có thao tác từ người dùng, ẩn toàn bộ các thành phần trên màn hình trừ phần hiển thị video - VideoQML (mã số 5)
- Nếu các thành phần trên màn hình đang được ẩn, nếu có thao tác chạm vào màn hình từ người dùng, hiển thị lại các thành phần điều khiển
- Thực thi API VideoQML::*play()* khi người dùng nhấn vào 5_C_2 nếu trạng thái phát file video hiện tại là dừng / tạm dừng (VideoQML.playbackState bằng PausedState hoặc StoppedState)
- Thực thi API VideoQML::*pause()* khi người dùng nhấn vào 5_C_2 nếu trạng thái phát file video hiện tại là đang phát (VideoQML.playbackState bằng PlayingState)
- Cập nhật thuộc tính đường dẫn file video hiện tại của VideoQML khi người dùng nhấn vào 5_C_1 thành đường dẫn của file video trước trong danh sách và gửi yêu cầu phát đến VideoQML
- Cập nhật thuộc tính đường dẫn file video hiện tại của VideoQML khi người dùng nhấn vào 5_C_1 thành đường dẫn của file video tiếp theo trong danh sách và gửi yêu cầu phát đến VideoQML