TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN ĐIỆN TỬ - VIỄN THÔNG

ĐỒ ÁN

TỐT NGHIỆP ĐẠI HỌC

Đề tài:

THIẾT KẾ VÀ PHÁT TRIỂN PHẦN MỀM ỨNG DỤNG ĐA PHƯƠNG TIỆN

Sinh viên thực hiện: ĐỖ TRUNG HIẾU

Lớp ĐT6 - K59

Giảng viên hướng dẫn: TS. VÕ LÊ CƯỜNG

Hà Nội, 12-2020

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN ĐIỆN TỬ - VIỄN THÔNG

ĐỒ ÁN

TỐT NGHIỆP ĐẠI HỌC

Đề tài:

THIẾT KẾ VÀ PHÁT TRIỂN PHẦN MỀM ỨNG DỤNG ĐA PHƯƠNG TIỆN

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

Hà Nội, 12-2020

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); Giỏi (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	5
2	Cập nhật kết quả nghiên cứu gần đây nhất (trong nước/quốc tế)	1	2	3	4	5
3	Nêu rõ và chi tiết phương pháp nghiên cứu/giải quyết vấn đề	1	2	3	4	5
4	Có kết quả mô phỏng/thực nghiệm và trình bày rõ ràng kết quả đạt được	1	2	3	4	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	5
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	5
7	Trong phần 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	4	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 căn lề, 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	5

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	4	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 khích trong các kỳ thi quốc gia và quốc tế khác về chuyên ngành như TI contest.	2				
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: "Thiết kế và phát triển phần mềm ứng dụng đa phương tiện".

Đồ án tốt nghiệp của em được thực hiện với mục tiêu: Vận dụng kĩ 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

Tóm tắt đồ án

Đồ án tốt nghiệp của em được thực hiện với các mục tiêu:

- Tìm hiểu và áp dụng bộ khung phần mềm Qt (Qt Framework) vào phát triển đề tài đồ án
- Tìm hiểu và áp dụng mô hình thiết kế phần mềm Model-View-Controller (MVC) vào giai đoạn thiết kế đề tài đồ án
- Phân tích tính năng, đặt ra các yêu cầu, thiết kế và lập trình sản phẩm
- Đánh giá sản phẩm sau khi hoàn thành so với các các yêu cầu đặt ra ban đầu
- Tạo ra sản phẩm phần mềm phục vụ tính năng phát file audio và video, với giao diện trực quan và cung cấp trải nghiệm người dùng tốt.

Qua quá trình thực hiện đồ án, em đã trau dồi được nhiều kĩ năng, có thể kể đến như: 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, ... Sản phẩm được tạo ra tuy vẫn còn thiếu sót và có thể cải tiến thêm, nhưng có thể coi là đáp ứng được các yêu cầu để có thể sử dụng trong thực tế.

My graduation project is implemented with the following goals:

- Learn and apply Qt Framework in project development
- Learn and apply MVC software design pattern to project design phase
- Feature analysis, create requirements, product design and programming
- Product evaluate against the original requirements
- Create software products for audio and video playback, with an intuitive, easy-to-interact interface and provide a good user experience.

Through the project implementation process, I have cultivated many skills, which can be mentioned as: understanding and evaluating the framework's capabilities, requirement analysis, design, programming, self-review and self-evaluate the product,... My product, although is still flawed and can be further improved, but can be considered to meet the requirements to be used in practice.

Mục lục

Danh mục hình ảnh

Danh mục bảng biểu

Danh mục từ viết tắt

MVC	Model-View-Controller
VideoQML	Video QML Type
Qt	Qt Framework
QML	Qt Modelling Language
OS	Operating System
RTOS	Real-Time Operating System
API	Application Programming Interface
GUI	Graphical User Interface
JSON	JavaScript Object Notation
XML	eXtensible Markup Language
OOP	Object-Oriented Programming
RAM	Random Access Memory
CPU	Central Processing Unit

Phần mở đầu

Với đề tài "Thiết kế và phát triển phần mềm ứng dụng đa phương tiện", 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 vấn đề cần được giải quyết bao gồm từ lựa chọn công cụ phát triển, phân tích yêu cầu, thiết kế kiến trúc, đến lập trình và đánh giá sản phẩm.

Giữa số lượng lớn công cụ phát triển hiện nay, em lựa chọn bộ công cụ phát triển phần mềm Qt Framework làm nền tảng cho đồ án của mình. Lựa chọn này yêu cầu lượng kiến thức nhất định về Qt để có thể sử dụng có hiệu quả.

Với mục tiêu vừa trau dồi kĩ năng cá nhân, đồng thời tạo ra sản phẩm có ứng dụng trong thực tiễn, có đủ khả năng đáp ứng nhu cầu sử dụng của người dùng thông thường, em sẽ lần lượt giải quyết các vấn đề nêu trên, chi tiết được trình bày trong các chương của đồ án, tổng quát như sau:

- Chương 1: Cơ sở lý thuyết
    - Giới thiệu và trình bày lý do lựa chọn Qt Framework làm công cụ phát triển cho đồ án
    - Tìm hiểu về mô hình thiết kế phần mềm Model-View-Controller (MVC)
- Chương 2: Thiết kế đề tài
    - 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 ứng dụng, từ cấu trúc các class (lớp), chi tiết xử lý của từng method (phương thức), tới mô tả hiển thị và hành vi của giao diện đồ họa người dùng (GUI)
- Chương 3: Kết quả và đánh giá
    - Chi tiết về sản phẩm thực tế sau khi hoàn thành giai đoạn phát triển
    - Đánh giá sản phẩm so với yêu cầu và thiết kế đề ra ban đầu

Chương 1. Cơ sở lý thuyết

Chương 1 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 khối trong mô hình và tương tác giữa các khối.

1.1. 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:

- 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
- Lập trình cảm biến
- Lập trình giao thức truyền thông (Bluetooth, serial port, NFC, ...)
- Xử lý XML (eXtensible Markup Language) và JSON (JavaScript Object Notation)
- Lập trình in ấn
- Tạo và in file PDF

Và rất nhiều các mảng khác.

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

- Hệ thống thông tin - giải trí trong xe hơi của hãng Mercedes-Benz [1]
- Hệ điều hành webOS (chủ yếu trên SmartTV) của LG [1]
- AMD Radeon Software, phần mềm điều khiển phần cứng đồ họa của hãng AMD [1]
- Hệ điều hành Ubuntu [1]

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.

1.2. 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 kí 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. 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.

1.3. Module Qt Core

1.3.1. Tổng quan về module Qt Core

Qt Core là module cơ sở nhất của Qt. Qt Core bổ sung nhiều tính năng đáng giá cho ngôn ngữ C++. Hai trong số đó là cơ chế signal - slot và hệ thống thuộc tính có thể tùy ý thêm/bớt và truy vấn cho một đối tượng.

1.3.2. Cơ chế Signal - Slot

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 kí "nghe" tín hiệu này, sẽ tự động thực hiện method tương ứng (slot).

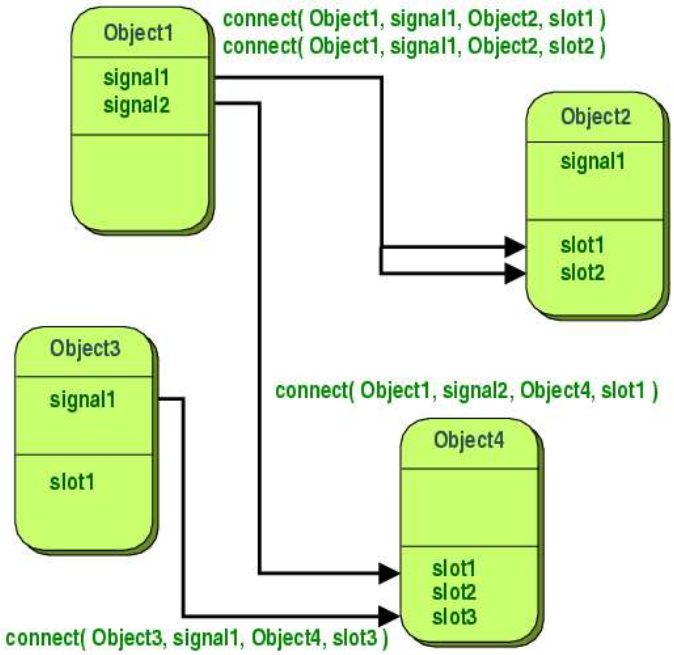
Cơ chế tự động thực thi nói trên đóng vai trò vô cùng quan trọng trong lập trình giao diện người dùng. Nhờ có cơ chế này, các sự kiện, ví dụ người dùng nhấn vào một nút bấm, có thể được xử lý bất đồng bộ, nghĩa là giao diện chỉ gửi đi signal sau đó thoát khỏi hàm và một đối tượng khác, ở luồng xử lý khác, sẽ thực hiện công việc tương ứng, sau đó trả kết quả về để cập nhật giao diện. Như vậy luồng xử lý vẽ 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 đủ 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. Ví du:

- Signal A có các tham số lần lượt:
    - argA1: int
    - argA2: bool
    - argA3: double
- Slot B có các tham số:
    - argB1: int
    - argB2: bool
    - argB3: double
- Slot C có các tham số:
    - argC1: int
    - argC2: bool
- Slot D có các tham số:
    - argD1: int
    - argD2: double

Signal A có thể kết nối với slot B và slot C, vì tham số của B trùng về cả kiểu dữ liệu và thứ tự với A, còn tham số của C thì trùng kiểu và thứ tự với phần đầu của A. Slot D không thể kết nối với A do thứ tự các tham số không thỏa mãn.



Hình 1.1 Cơ chế signal - slot của Qt [1]

Hình 1.1 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, 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 ở trên.

1.3.3. Hệ thống thuộc tính

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.

1.3.4. Class QObject

QObject là class cơ bản nhất của Qt. QObject là class cha cho tất cả các class khác mà Qt cung cấp. QObject đồng thời là cơ sở để Qt có thể triển khai cơ chế signal - slot đươc mô tả trong phần 1.2.2. QObject cung cấp các method sau để tăng cường cơ chế signal - slot:

- *connect*(): kết nối một signal với một slot
- *disconnect*(): ngắt kết nối giữa signal và slot
- *blockSignal*(): tạm thời chặn tín hiệu mà không cần phải ngắt kết nối
- *connectNotify*(): tín hiệu được phát đi khi một kết nối signal - slot được tạo thành công
- *disconnectNotify*(): tín hiệu được phát đi khi một kết nối signal - slot dược ngắt thành công

Ngoài signal - slot, QObject là cơ sở để tạo ra mối quan hệ phân tầng giữa các đối tượng. Một đối tượng thuộc class QObject hoặc các class được kế thừa từ QObject có thể được tạo ra với thông tin về đối tượng là cha của nó, và bản thân class đó sẽ tự động được thêm vào danh sách các đối tượng con của đối tượng cha. Khi đối tượng cha được giải phóng khỏi bộ nhớ, tất cả các đối tương con của nó cũng sẽ tự động được giải phóng theo, giúp hạn chế hiện tượng rò rỉ bộ nhớ.

Một đối tượng thuộc class QObject hoặc các class kế thừa từ QObject sẽ phát đi một signal sau khi được giải phóng. Lập trình viên có thể dựa vào signal này để đảm bảo các tham chiếu trước đó của đối tượng này cũng được giải phóng.

1.4. Module Qt Multimedia

1.4.1. Tổng quan về 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 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 vào từng khung hình trong video để tiếp tục xử lý

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

1.4.2. Class QMediaPlayer

QMediaPlayer là class cung cấp các tính năng điều khiển phát cho file audio. Lập trình viên được cung cấp các API để thực hiện các tính năng như:

- *play*(): Phát nội dung
- *pause*(): Tạm dừng phát
- *setMuted*(): Thiết lập tắt tiếng

- *setPlaybackRate*(): Thiết lập tốc độ phát
- *setPlaylist*(): Thiết lập danh sách phát
- *setPosition*(): Thiết lập vị trí phát
- *setVolume*(): Thiết lập âm lượng phát
- *stop*(): Dừng phát

QMediaPlayer cung cấp các thuộc tính:

- *duration:*
    - Định nghĩa: Thời lượng của nội dung đang phát, đơn vị mili giây
    - Kiểu dữ liệu: qint64 - số nguyên 64-bit
- *playbackRate:*
    - Định nghĩa: Tốc độ phát đang được sử dụng, đơn vị lần (1.0 là tốc độ chuẩn)
    - Kiểu dữ liệu: qreal - số thực
- *isMuted:*
    - Định nghĩa: Thiết lập tắt tiếng đang được sử dụng
    - Kiểu dữ liệu: boolean:
        - True: đang tắt tiếng
        - False: đang không tắt tiếng
- *position:*
    - Định nghĩa: Vị trí đang phát hiện tại, đơn vị mili giây
    - Kiểu dữ liệu: qint64 - số nguyên 64-bit
- *state:*
    - Định nghĩa: Trạng thái phát hiện tại
    - Kiểu dữ liệu: enum:
        - StoppedState: Trình phát hiện không có nội dung đang phát. Nội dung sẽ bắt đầu từ đầu khi có yêu cầu phát.
        - PlayingState: Trình phát đang phát nội dung
        - PausedState: Trình phát đang tạm dừng phát nội dung. Nội dung sẽ được tiếp tục phát từ vị trí trước khi tạm dừng khi có yêu cầu phát.

- *volume:*
    - Định nghĩa: Âm lượng đang được thiết đặt, giá trị trong khoảng từ 0 (im lặng hoàn toàn) đến 100 (âm lượng tối đa)
    - Kiểu dữ liệu: int - số nguyên

Các thuộc tính kể trên khi thay đổi đều có một signal tương ứng được phát đi. Ngoài ra QMediaPlayer còn cung cấp thêm các signal khác. Danh sách và ví dụ về trường hợp sử dụng các signal đó được liệt kê trong Bảng 1.1. dưới đây:

Bảng 1.1 Các signal của QMediaPlayer

Signal	Mô tả	Trường hợp sử dụng
currentMediaChanged()	Nội dung đang phát thay đổi	Cập nhật hiển thị tên bài hát đang phát trên giao diện
durationChanged()	Thời lượng nội dung đang phát thay đổi	Cập nhật hiển thị thông tin thời lượng file trên giao diện
error()	Có lỗi xảy ra	Thông báo lỗi cho người dùng
mutedChanged()	Thiết lập tắt tiếng đã được thay đổi	Cập nhật biểu tượng âm lượng trên giao diện
playbackRatedChanged()	Tốc độ phát đã được thay đổi	Cập nhật hiển thị thông tin tốc độ phát (1x, 2x, 4x, ...) trên giao diện
positionChanged()	Thời lượng đã phát đã thay đổi	Cập nhật hiển thị thông tin thời lượng đã phát trên giao diện
stateChanged()	Trạng thái đang phát thay đổi	Cập nhật hiển thị trạng thái nút điều khiển dừng / phát trên giao diện

volumeChanged()	Giá trị âm lượng thay đổi	Cập nhật hiển thị giá trị âm lượng trên giao diện

Các trường hợp được đề cập trong Bảng 1.1. là các tình huống tiêu biểu. Trong một dự án thực tế, các signal trên có thể xuất hiện trong nhiều trường hợp phức tạp hơn.

1.4.3. Class QMediaPlaylist

QMediaPlaylist cung cấp chức năng tạo mới và quản lý danh sách phát. Chức năng quản lý danh sách phát được cung cấp qua các API bao gồm:

- *addMedia*(): Thêm một hoặc nhiều nội dung vào cuối danh sách phát
- *insertMedia*(): Thêm một hoặc nhiều nội dung vào một ví trí bất kì trong danh sách phát
- *removeMedia*(): Xóa bỏ một hoặc nhiều nội dung khỏi danh sách phát
- *clear*(): Xoá bỏ toàn bộ nội dung trong danh sách phát
- *currentIndex*(): Chỉ số của nội dung đang được phát trong danh sách phát
- *mediaCount*(): Trả về số lượng nội dung hiện đang có trong danh sách phát
- *moveMedia*(): Thay đổi thứ tự của một nội dung trong danh sách phát
- *save*(): Lưu danh sách phát tới một địa chỉ trong bộ nhớ
- *setPlaybackMode*(): Thiết đặt chế độ phát lại. Chế độ phát lại có thể là 1 trong các giá trị sau:
    - *CurrentItemOnce*: Ngừng phát sau khi kết thúc nội dung đang phát
    - *CurrentItemInLoop*: Chỉ lặp lại nội dung đang phát
    - *Sequential*: Phát lần lượt các nội dung và ngừng phát sau khi nội dung cuối cùng trong danh sách kết thúc
    - *Loop*: Phát lần lượt các nội dung và phát lại từ nội dung đầu tên sau khi nội dung cuối cùng trong danh sách kết thúc
    - *Random*: Phát ngẫu nhiên các nội dung trong danh sách

QMediaPlaylist cung cấp các thuộc tính:

- *currentIndex:*
    - Định nghĩa: thứ tự của nội dung đang phát trong danh sách
    - Kiểu dữ liệu: int - số nguyên
- *playbackMode:*
    - Định nghĩa: chế độ phát lại đang được áp dụng
    - Kiểu dữ liệu: enum
        - *CurrentItemOnce*: Ngừng phát sau khi kết thúc nội dung đang phát
        - *CurrentItemInLoop*: Chỉ lặp lại nội dung đang phát
        - *Sequential*: Phát lần lượt các nội dung và ngừng phát sau khi nội dung cuối cùng trong danh sách kết thúc
        - *Loop*: Phát lần lượt các nội dung và phát lại từ nội dung đầu tên sau khi nội dung cuối cùng trong danh sách kết thúc
        - *Random*: Phát ngẫu nhiên các nội dung trong danh sách

Các thuộc tính kể trên khi thay đổi đều có một signal tương ứng được phát đi. Ngoài ra QMediaPlaylist còn cung cấp thêm các signal khác. Danh sách và ví dụ về trường hợp sử dụng các signal đó được liệt kê trong Bảng 1.2. dưới đây:

Bảng 1.2 Các signal của QMediaPlaylist

Signal	Mô tả	Trường hợp sử dụng
currentIndexChanged()	Thứ tự của nội dung đang phát thay đổi	Cập nhật hiển thị trạng thái phát / không phát của một nội dung trong cả danh sách được hiển thị
playbackModeChanged()	Chế độ phát lại được thay đổi	Cập nhật hiển thị của nút bấm thiết đặt chế độ phát lại

mediaInserted()	Nội dung mới được thêm thành công	Cập nhật hiển thị danh sách phát
mediaRemoved()	Nội dung đươc gỡ bỏ thành công	Cập nhật hiển thị danh sách phát

Ngoài các tình huống trên, trong phát triển dự án thực tế, có thể có nhiều tình huống khác cần đến các signal trên.

1.4.4. Video QML Type

Video QML Type (VideoQML) đơn giản hóa phương thức phát file video cho người dùng. Chỉ cần cung cấp đường dẫn tới video, các công việc còn lại, từ giải mã đến vẽ khung hình lên giao diện được VideoQML thực hiện tự động, đơn giản hóa công việc cho lập trình viên.

VideoQML cung cấp bộ API đơn giản cho việc phát file video:

- *pause():* tạm dừng phát nội dung
- *play():* bắt đầu / tiếp tục phát nội dung
- *seek():* điều chỉnh vị trí phát của nội dung
- *stop():* dừng phát nội dung

VideoQML cung cấp các thuộc tính:

- *autoLoad:*
  - Định nghĩa: Thiết lập tự động giải mã và tải dữ liệu của video ngay sau khi đường dẫn được cung cấp
  - Kiểu dữ liệu: boolean, giá trị mặc định *True*
    - *True*: tự động tải
    - *False*: không tự động tải, việc tải video chỉ được diễn ra khi có yêu cầu phát
- *autoPlay:*
  - Định nghĩa: Thiết lập tự động phát file video sau khi được cung cấp đường dẫn
  - Kiểu dữ liệu: boolean, giá trị mặc định *False*
    - *True*: tự động phát file video

        - *False*: không tự động phát file video
- *fillMode:*
    - Định nghĩa: Thiết lập phương thức hiển thị của video trên khu vực hiển thị của VideoQML trên màn hình
    - Kiểu dữ liệu: enum
        - *Stretch*: video tự điều chỉnh tỷ lệ sao vừa lấp đầy khu vực hiển thị, không cắt xén
        - *PreserveAspectFit*: video được giữ nguyên tỷ lệ ban đầu, không cắt xén
        - *PreserveAspectCrop*: video được phát lấp đầy khu vực, được cắt bớt các cạnh nếu cần
- *flushMode:*
    - Định nghĩa: Thiết lập hành động của VideoQML sau khi hoàn thành phát một video
    - Kiểu dữ liệu: enum
        - *EmptyFrame*: không hiển thị gì trên khu vực phát file video
        - *FirstFrame*: hiển thị khung hình đầu tiên của video vừa phát trên khu vực hiển thị
        - *LastFrame*: hiển thị khung hình cuối cùng của video vừa phát trên khu vực hiển thị
- *hasAudio:*
    - Định nghĩa: Mô tả nội dung hiện tại có hay không có âm thanh
    - Kiểu dữ liệu: boolean
        - *True*: có âm thanh
        - *False*: không có âm thanh
- *hasVideo:*
    - Định nghĩa: Mô tả nội dung hiện tại có hay không có video
    - Kiểu dữ liệu: boolean
        - *True*: có video
        - *False*: không có video

- *playbackState:*
    - Định nghĩa: Trạng thái phát hiện tại
    - Kiểu dữ liệu: enum
        - *PlayingState*: nội dung đang phát
        - *PausedState*: nội dung đang tạm dừng
        - *StoppedState*: nội dung đã dừng hẳn
- *position:*
    - Định nghĩa: Vị trí hiện tại của nội dung đang phát, đơn vị mili giây
    - Kiểu dữ liệu: int - số nguyên
- *seekable:*
    - Định nghĩa: Thiết lập cho phép thay đổi vị trí đang phát của nội dung hiện tại
    - Kiểu dữ liệu: boolean
        - *True*: Cho phép thay đổi vị trí đang phát
        - *False*: Không cho phép thay đổi vị trí đang phát

Bảng 1.3. liệt kê các signal được cung cấp bởi VideoQML:

Bảng 1.3 Các signal của VideoQML

Signal	Mô tả	Trường hợp sử dụng
paused()	Nội dung đã tạm ngừng phát	Cập nhật hiển thị cho nút bấm điều khiển dừng / phát
playing()	Nội dung đã được bắt đầu / tiếp tục phát	Cập nhật hiển thị cho nút bấm điều khiển dừng / phát
stopped()	Nội dung đã dừng phát	Cập nhật hiển thị cho nút bấm điều khiển dừng / phát

1.5. Mô hình thiết kế phần mềm MVC

1.5.1. 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." [3]

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. [3]

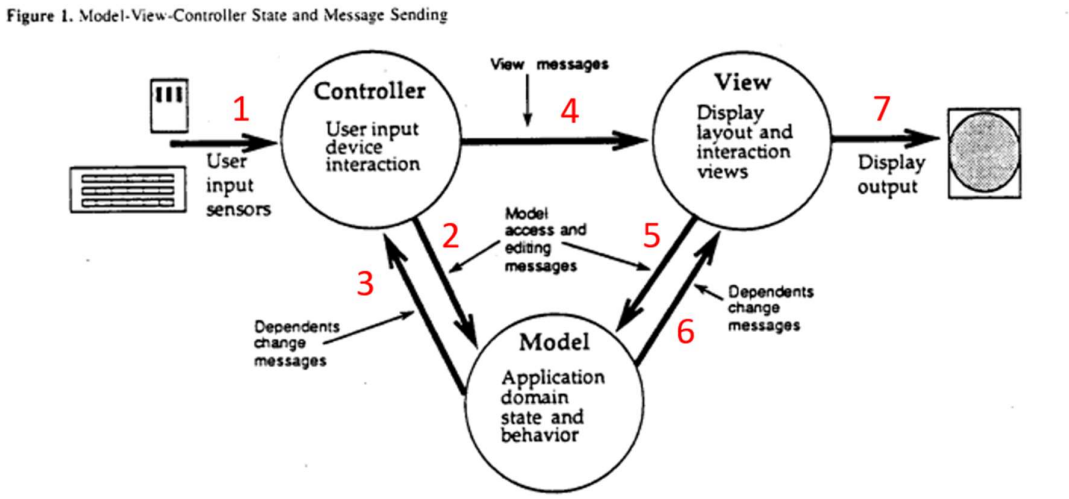
Quy trình phát triển ứng dụng MVC chia trách nhiệm của những người tham gia thành 3 vai trò để tăng tính hiệu quả của công việc, bao gồm:

- *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.
- *Intergration*: Vai trò tích hợp, là người có trách nhiệm gắn kết công việc của hai vai trò trước đó. [4]

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." [5]



Hình 1.2 Tương tác giữa các khối trong mô hình MVC [3]

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

1.5.2. 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. [6]

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. [5]

1.5.3. 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, ... [3]

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

1.5.4. 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.

Chương 1 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 2.

Chương 2. Thiết kế đề tài

Chương 2 trình bày về quá trình phân tích, thiết kế và phát triển đề tài của em. Các bước trong quá trình thực hiện thiết kề đề tài bao gồm: Phân tích yêu cầu chức năng của ứng dụng, Thiết kế kiến trúc của ứng dụng với mô hình MVC, và Thiết kế chi tiết ứng dụng với các class, thuộc tính các class, tương tác giữa các class, mô tả GUI và hành vi của GUI.

2.1. Phân tích yêu cầu chức năng

Mỗi yêu cầu sẽ được gán với một nhãn duy nhất, hỗ trợ cho việc đánh giá mức độ hoàn thiện của sản phẩm sau khi đã hoàn thành quá trình phát triển.

2.1.1. Yêu cầu chức năng cho chức năng phát file audio

- REQ_AU_1: Ứng dụng có thể phát các file có định đạng mp3.
- REQ_AU_2: Ứng dụng có thể tự động tìm và lên danh sách toàn bộ các file mp3, từ đó tạo ra danh sách phát.
- REQ_AU_3: 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
    - Hình album
- REQ_AU_4: Khi thông tin tên bài hát không có sẵn trong meta-data, thông tin này được thay thế bằng tên file.
- REQ_AU_5: Khi thông tin nghệ sĩ thể hiện không có sẵn trong meta-data, thông tin này được thay thế bằng “Unknown Artist”.
- REQ_AU_6: Khi thông tin hình album không có sẵn trong meta-data, thông tin này được thay thế bằng hình ảnh mặc định.
- REQ_AU_7: 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

- REQ_AU_8: Ứng dụng có thể hiển thị tổng thời lượng của file âm thanh đang phát, theo định dạng hh:mm:ss (giờ:phút:giây).
- REQ_AU_9: Ứng dụng có thể hiển thị thời lượng đã phát của file âm thanh, theo định dạng hh:mm:ss (giờ:phút:giây).
- REQ_AU_10: Ứng dụng có thể hiển thị tỷ lệ giữa thời lượng đã phát so với tổng thời lượng của file âm thanh đang phát.
- REQ_AU_11: Ứng dụng cung cấp khả năng điều khiển phát cho người dùng, bao gồm:
    - Điều khiển dừng / phát
    - Điều khiển chế độ lặp lại:
        - 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: Chỉ phát lại nội dung đang phát sau khi kết thúc.
    - Chế độ lặp lại mặc định sau khi khởi động là "Lặp lại toàn bộ".
    - 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.
    - Chọn phát một nội dung từ danh sách phát.
- REQ_AU_12: 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.
- REQ_AU_13: 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.

- REQ_AU_14: Khi một nội dung âm thanh kết thúc, tự động chuyển sang nội dung tiếp theo trong danh sách ngay lập tức.
- REQ_AU_15: Khi người dùng chọn một nội dung từ danh sách phát, dừng nội dung đang phát hiện tại và phát nội dung vừa được chọn từ đầu.
- REQ_AU_16: Khi chế độ lặp lại đang được áp dúng 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".
- REQ_AU_17: 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".
- REQ_AU_18: Khi chế độ lặp lại đang được áp dùng 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"
- REQ_AU_19: 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
- REQ_AU_20: Sau khi ứng dụng tắt, ở lần tiếp theo ứng dụng được khởi động, nội dung gần nhất được phát trước đó sẽ được chọn sẵn, nếu người dùng thực hiện thao tác bắt đầu phát, phát nội dung từ đầu.

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

- REQ_VID_1: Ứng dụng có thể phát các file video có định dạng mp4.
- REQ_VID_2: Ứng dụng có thể tự động tìm và lên danh sách toàn bộ các file mp4, từ đó tạo ra danh sách phát.
- REQ_VID_3: 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)

- REQ_VID_4: Ứng dụng có thể hiển thị tên của file video đang được phát.
- REQ_VID_5: Ứng dụng có thể hiển thị tổng thời lượng của file video, theo định dạng hh:mm:ss (giờ:phút:giây).
- REQ_VID_6: Ứng dụng có thể hiển thị thời lượng đã phát của file video, theo định dạng hh:mm:ss (giờ:phút:giây).
- REQ_VID_7: Ứng dụng có thể hiển thị tỷ lệ giữa thời lượng đã phát so với tổng thời lượng của video đang phát.
- REQ_VID_8: Ứng cung cung cấp khả năng điều khiển phát cho người dùng, bao gồm:
    - 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
    - Chọn phát một nội dung từ dánh sách phát
- REQ_VID_9: 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. Nều người dùng chọn phát lại, không chuyển sang nội dung tiếp theo, và phát lại nội dung vừa phát từ đầu.
- REQ_VID_10: Video được phát ở chế độ toàn màn hình.
- REQ_VID_11: Ứng dụng có thể lưu lịch sử 2 video được phát gần nhất.
- REQ_VID_12: Ứng dụng có thể hiển thị thông tin 2 video được phát gần nhất.
- REQ_VID_13: Với video được phát mới hơn trong 2 video được lưu lịch sử, hiển thị khung hình ứng với vị trí phát trước đó.
- REQ_VID_14: 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.

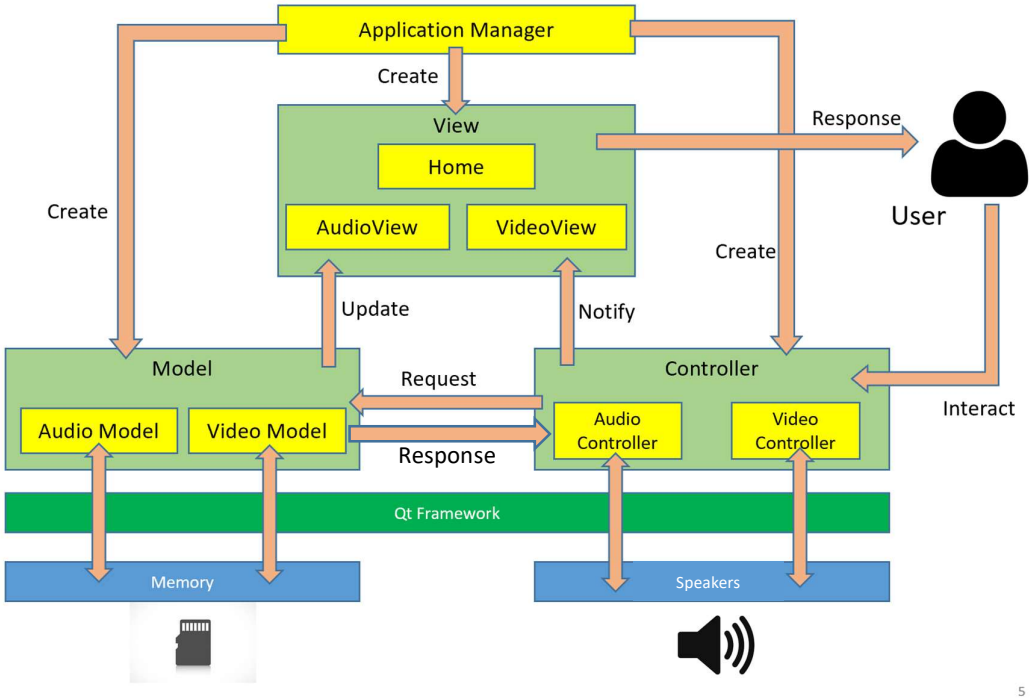
2.1.3. Yêu cầu phi chức năng:

- REQ_OTHER_1: Ứng dụng có thể hiển thị thông tin giờ

2.2. Thiết kế kiến trúc của ứng dụng

2.2.1. Sơ đồ khối

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



Hình 2.1 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.

2.2.2. 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 cung cấp các chức năng điều khiển phát file audio
    - VideoController: Khối xử lý 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

2.2.3. 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, ...)

2.2.4. 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

2.3. Thiết kế chi tiết

2.3.1. Tổ chức mã nguồn

Mã nguồn được tổ chức dạng cây thư mục như sau:

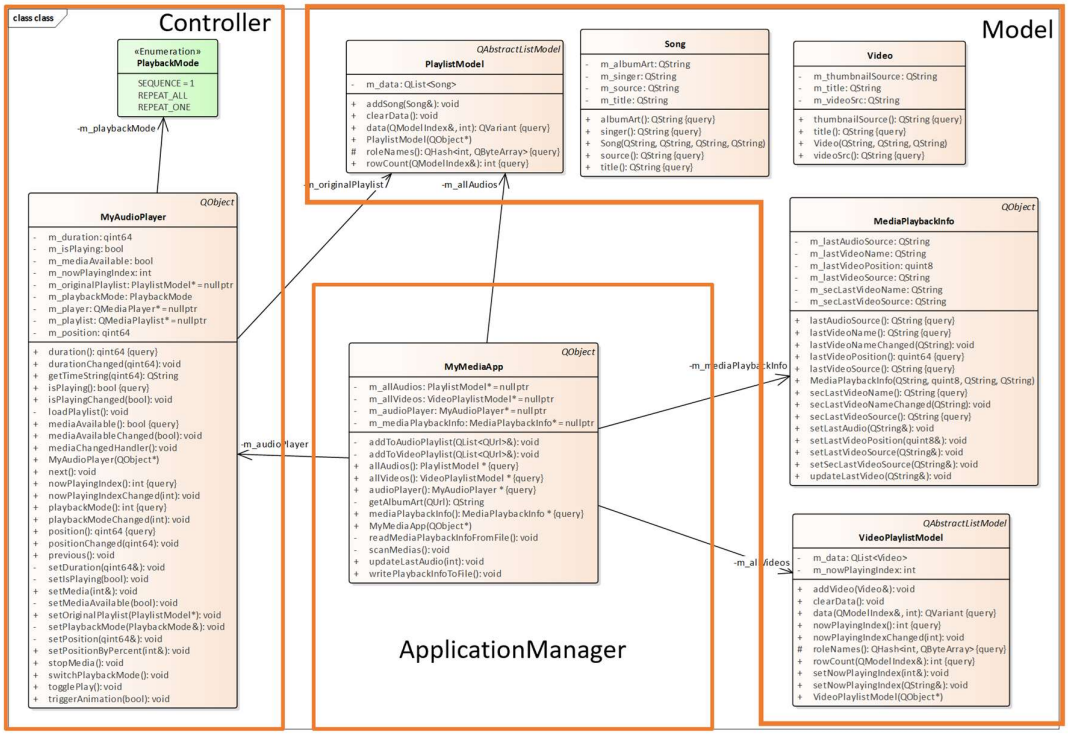
- Project_MediaApp
    - Libs
        - Taglib (thư viện dùng để truy cập meta-data)
    - Media
        - mediaplaybackinfo.cpp
        - mymediaapp.cpp
        - song.cpp
        - videoplaylistmodel.cpp
        - mediaplaybackinfo.h
        - mymediaapp.h
        - song.h
        - videoplaylistmodel.h
        - myaudioplayer.cpp
        - playlistmodel.cpp
        - video.cpp
        - myaudioplayer.h
        - playlistmodel.h
        - video.h
    - Qml
        - AudioApp.qml

- DateTimeWidget.qml
- VideoApp.qml
- AudioWidget.qml
- MyButton.qml
- VideoPlayer.qml

- Resources
- main.cpp
- main.qml
- Project_MediaApp.pro
- qml.qrc

2.3.2. Thiết kế các Class

2.3.2.1. Biểu đồ Class



Hình 2.2 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: class đóng vai trò quản lý và khởi tạo các đối tượng. Đồng thời gắn kết hoạt động của các class còn lại.

- Khối Model:
  - MediaPlaybackInfo : class lưu trữ thông tin của về nội dung được phát gần nhất của cả audio và video.
  - Song: class lưu trữ thông tin của một nội dung audio
  - Video: class lưu trữ thông tin của một nội dung video
  - PlaylistModel: class lưu trữ thông tin của danh sách phát file audio
  - VideoPlaylistModel: class lưu trữ thông tin của danh sách phát file video
- Khối Controller:
  - MyAudioApp: class cung cấp phương thức điều khiển phát file audio
  - 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 MyAudioApp của audio, vì các phương thức này đã được cung cấp sẵn bởi VideoQML (1.4.4).

Chi tiết các class được trình bày trong các phần tiếp theo của mục 2.3.2.

2.3.2.2. Class MyMediaApp

MyMediaApp là class có nhiệm vụ quản lý, khởi tạo các đối tượng trong ứng dụng. 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.

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 lên 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

2.3.2.3. Class MyAudioPlayer

MyAudioPlayer cung cấp các phương thức điều khiển phát file audio.
Chi tiết về class MyMediaApp:

- 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 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 hiện 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_mediaAvaiable 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

2.3.2.4. Class MediaPlaybackInfo

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.

Chi tiết class MediaPlaybackInfo:

- Tên: MediaPlaybackInfo
- Header: mediaplaybackinfo.h
- Source: mediaplaybackinfo.cpp
- Danh sách biến:
    - m_lastAudioSource - 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

- *lastAudioSource*() - Phạm vi: Public - Kiểu trả về: QString
  - Trả về giá trị của m_lastAudioSource
- *lastVideoName*() - Phạm vi: Public - Kiểu trả về: QString
  - Trả về giá trị của m_lastVideoName
- *lastVideoPosition*() - Phạm vi: Public - Kiểu trả về: quint64
  - Trả về giá trị của m_lastVideoPosition
- *lastVideoSource*() - Phạm vi: Public - Kiểu trả về: QString
  - Trả về giá trị của m_lastVideoSource
- *secLastVideoName*() - Phạm vi: Public - Kiểu trả về: QString
  - Trả về giá trị của m_secLastVideoName
- *secLastVideoSource*() - Phạm vi: Public - Kiểu trả về: QString
  - Trả về giá trị của m_secLastVideoSource
- *setLastAudio*() - Phạm vi: Public - Kiểu trả về: void
  - Thay đổi giá trị của m_lastAudio
- *setLastVideoPosition*() - Phạm vi: Public - Kiểu trả về: void
  - Thay đổi giá trị của m_lastVideoPosition
- *setLastVideoSource*() - Phạm vi: Public - Kiểu trả về: void
  - Thay đổi giá trị của m_lastVideoSource
- *setSecLastVideoSource*() - Phạm vi: Public - Kiểu trả về: void
  - Thay đổi giá trị của m_secLastVideoSource
- *updateLastVideo*() - Phạm vi: Public - Kiểu trả về: void
  - Thay đổi giá trị hiện tại của m_secLastVideoName bằng với giá trị hiện tại của m_lastVideoName
  - Thay đổi giá trị hiện tại của m_secLastVideoSource bằng với giá trị hiện tại của m_lastVideoSource
  - 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

2.3.2.5. Class PlaylistModel

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

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

2.3.2.6. Class Song

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

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

2.3.2.7. Class Video

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

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

2.3.2.8. Class VideoPlaylistModel

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

2.3.3. Mô tả bố cục giao diện

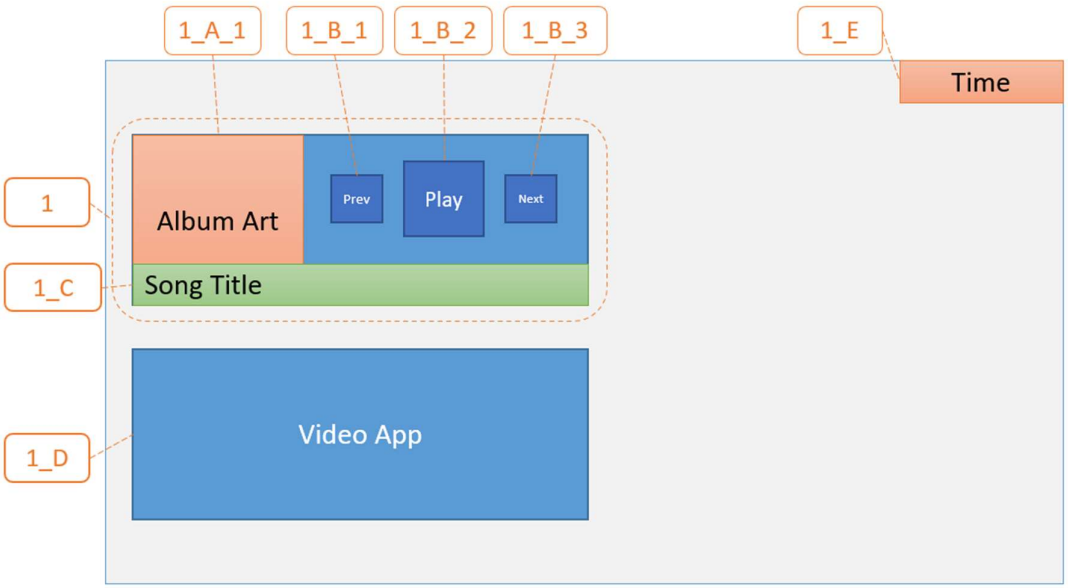
Giao diện chính là các khối con của khối View trong sơ đồ khối của ứng dụng. Các khối con bao gồm Home, AudioView và VideoView. Giao diện của ứng dụng được chia thành các màn hình hiển thị, được liệt kê tương ứng với các khối như sau:

- Khối 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 về các màn hình được mô tả trong các phần tiếp theo của mục 2.3.3

2.3.3.1. Màn hình Home



Hình 2.3 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. Màn hình Home bao gồm các thành phần:

- 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

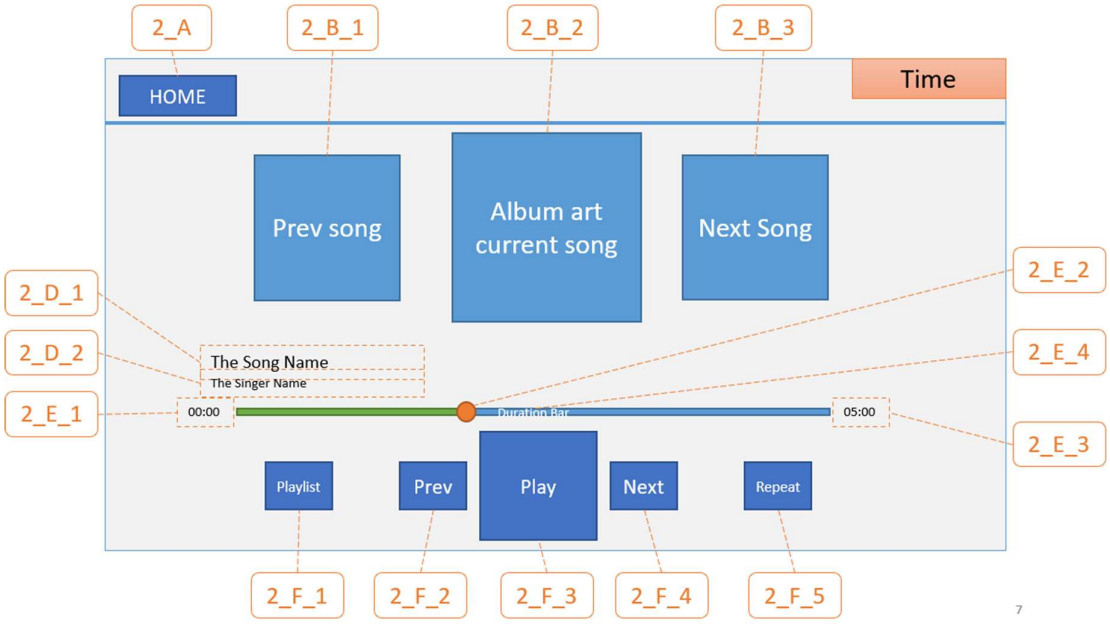
- Hiển thị nút bấm bắt đầu phát khi nội dung đang ngừng / tạm ngừng phát
- Hiện thị nút bấm tạm dừng khi nội dung đang phát

o 1_B_3: Nút bấm điều khiển chuyển sang nội dung tiếp theo

o 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)

2.3.3.2. Màn hình Audio - Player



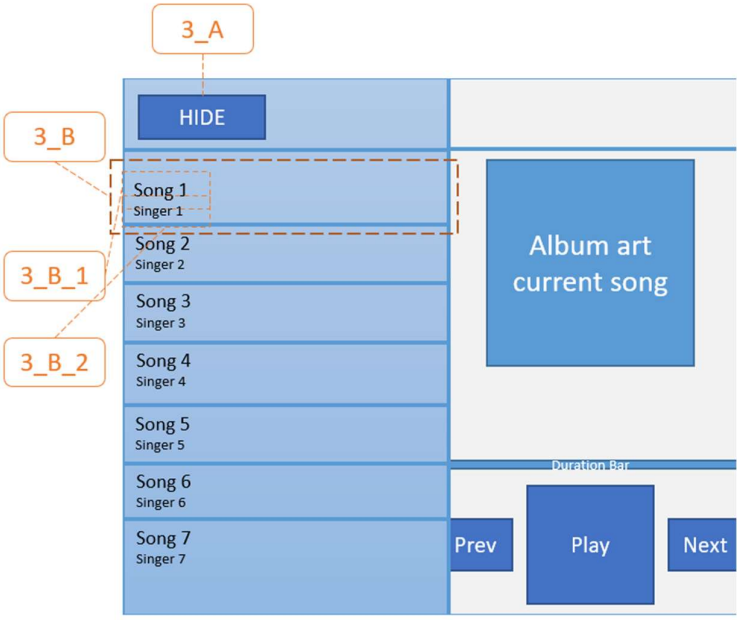
Hình 2.4 Màn hình Audio - Player

Audio - Player hiển thị các nút bấm cho phép người dùng điều khiển chức năng phát file audio, và các thông tin liên quan đến nội dung audio đang được phát. Màn hình Audio - Player gồm các thành phần sau:

- 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 hiển 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
    - Hiện 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 đọ 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

2.3.3.3. Màn hình Audio - Playlist



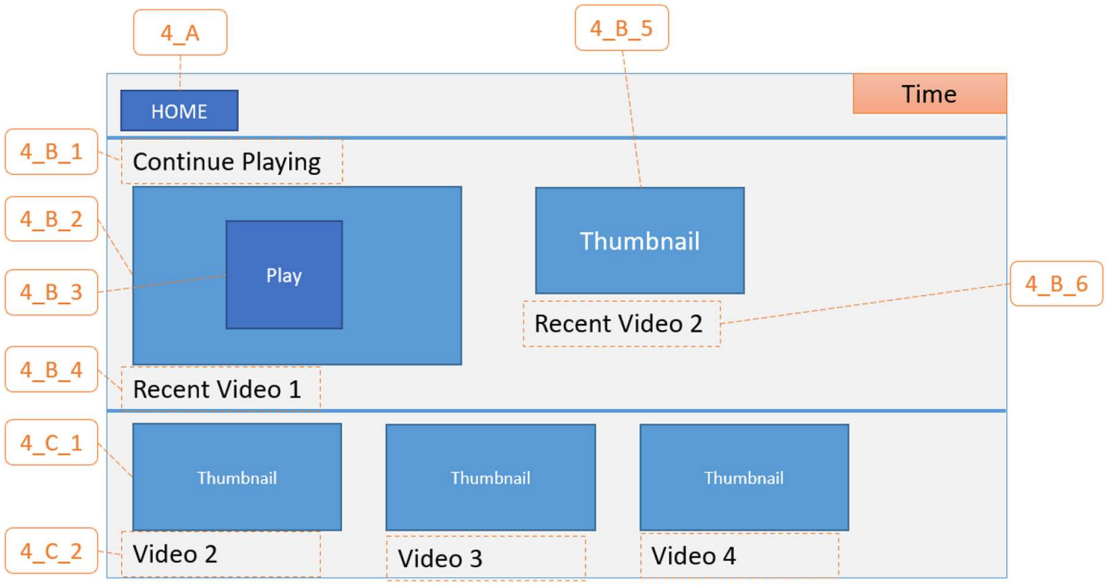
Hình 2.5 Màn hình Audio - Playlist

Audio - Playlist hiển thị danh sách các nội dung audio. Nó xuất hiện đè lên màn hình trước đó và chỉ chiếm phần bên trái khu vực hiển thị. Màn hình Audio - Playlist bao gồm:

- 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.

2.3.3.4. Màn hình Video - Playlist

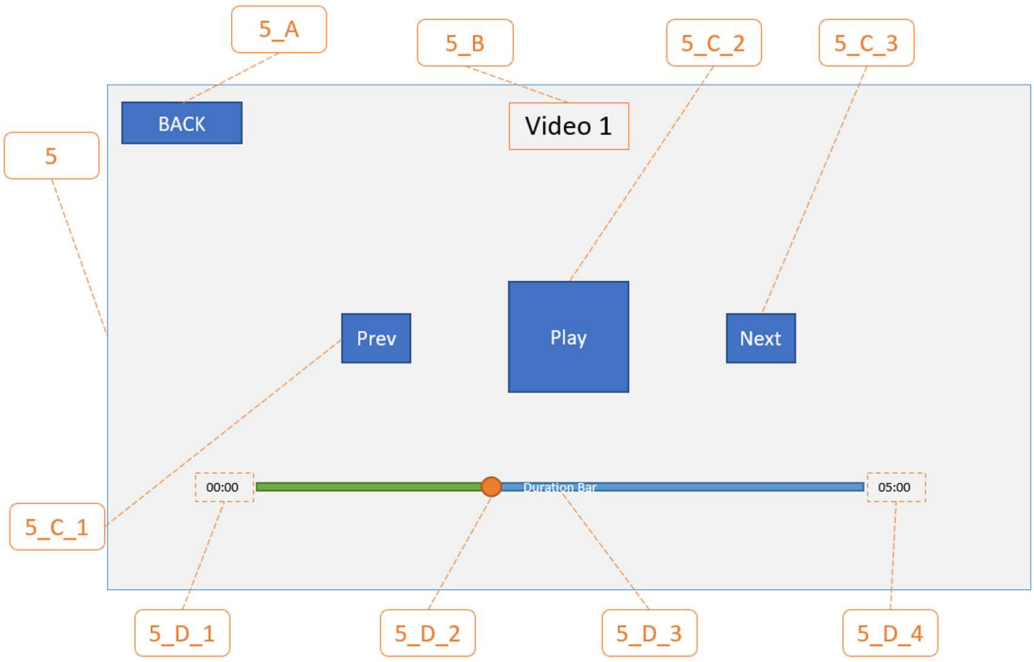


Hình 2.6 Màn hình Video - Playlist

Màn hình Video - Playlist hiển thị danh sách các nội dung video, lịch sử phát 2 video gần nhất. Màn hình Video - Playlist bao gồm các thành phần:

- 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

2.3.3.5. 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. Video - Player gồm các thành phần:

- 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

2.3.4. Mô tả hành vi của các thành phần trong View

2.3.4.1. 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

2.3.4.2. 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

2.3.4.3. Màn hình Audio - Playlist

- Chuyển đến màn mình 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

2.3.4.4. 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

2.3.4.5. 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

Với các thông tin về phân tích và thiết kế được trình bày trong Chương 2, 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/dotrunghieu96/datn.git

Sản phẩm sau khi hoàn thành được đánh giá ở Chương 3 tiếp theo.

Chương 3. Kết quả và đánh giá

Chương 3 trình bày về sản phẩm thực tế sau khi triển khai phát triển với thiết kế đã được trình bày trong Chương 2. Nội dung trong Chương 3 bao gồm:

- Hình ảnh thực tế của sản phẩm và đánh giá mức độ tuân thủ so với mô tả giao diện
- Đánh giá mức độ hoàn thiện của sản phẩm so với các yêu cầu chức năng đã được phân tích trong mục 2.1.
- Đánh giá mức độ tuân thủ mô hình thiết kế MVC được đặt ra ban đầu trong mục 1.5.

3.1. Hình ảnh thực tế của sản phẩm và đánh giá giao diện

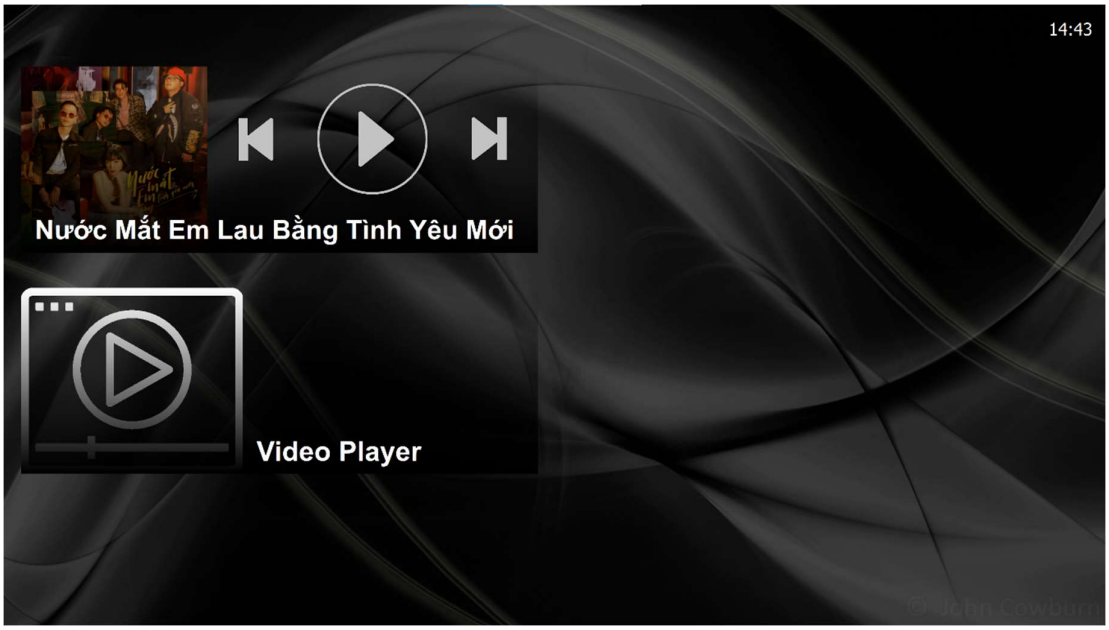
Giao diện được em tự đánh giá theo từng màn hình, với các tiêu chí:

- Đáp ứng về bố cục được mô tả trong mục 2.3.3
- Đáp ứng về hành vi được mô tả trong mục 2.3.4

Các mức độ tuân thủ bao gồm: Không đáp ứng, Đáp ứng một phần, Đáp ứng.

3.1.1. Màn hình Home

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



Hình 3.1 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 2.3.3.1, và hành vi được mô tả trong mục 2.3.4.1. Em tự đánh giá về màn hình Home như sau:

- Mức độ đáp ứng về bố cục: Đáp ứng - Các thành phần trong màn hình được sắp xếp tương tự như vị trí trong mô tả.
- Mức độ đáp ứng về hành vi: Đáp ứng - Các hành vi, bao gồm chuyển màn hình và thực hiện tính năng, được thực hiện đúng như mô tả

Đánh giá cá nhân về màn hình Home: Màn hình Home còn nhiều khoảng trống ở khu vực bên phải. Kích thước và vị trí biểu tượng video chưa hợp lý. Khu vực hiển thị thời gian nên có thêm thông tin về ngày tháng.

3.1.2. Màn hình Audio - Player

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

Hình 3.2 Màn hình Audio - Player (sản phẩm)

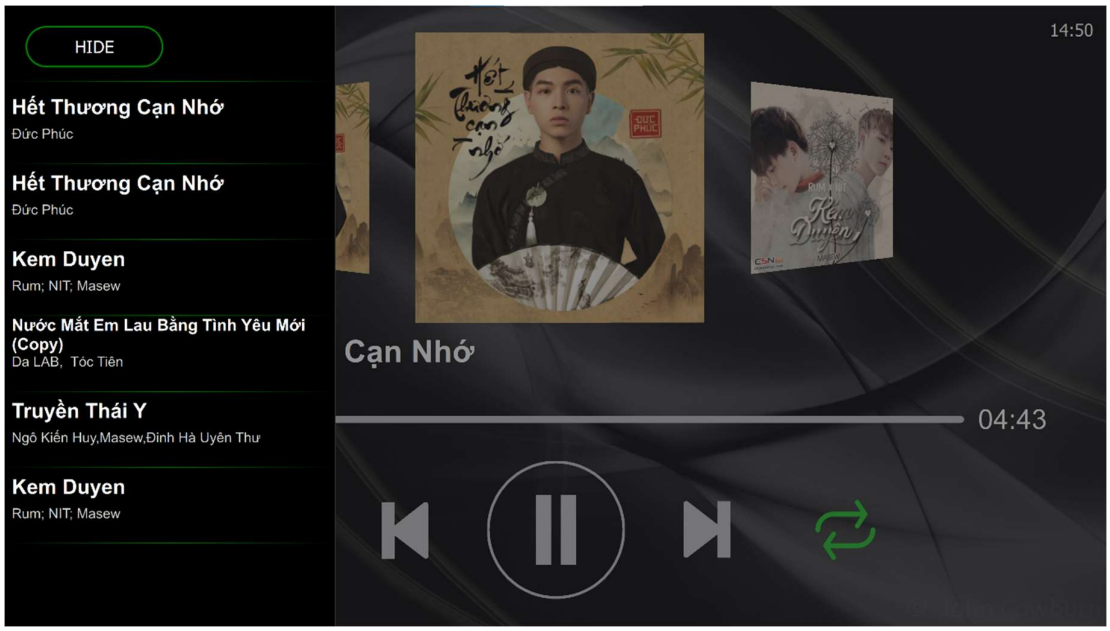
Màn hình Audio - Player được triển khai chủ đạo theo tông màu tối, các thành phần được sắp xếp theo bố cục được mô tả trong mục 2.3.3.2, và hành vi được mô tả trong mục 2.3.4.2. Em tự đánh giá về màn hình Audio - Player như sau:

- Mức độ đáp ứng về bố cục - Đáp ứng: Các thành phần trong màn hình được sắp xếp tương tự như vị trí trong mô tả.

- Mức độ đáp ứng về hành vi - Đáp ứng: Các hành vi, bao gồm chuyển màn hình và thực hiện tính năng, được thực hiện đúng như mô tả

3.1.3. Màn hình Audio - Playlist

Màn hình Audio - Playlist của sản phẩm được chụp lại trong Hình 3.3.



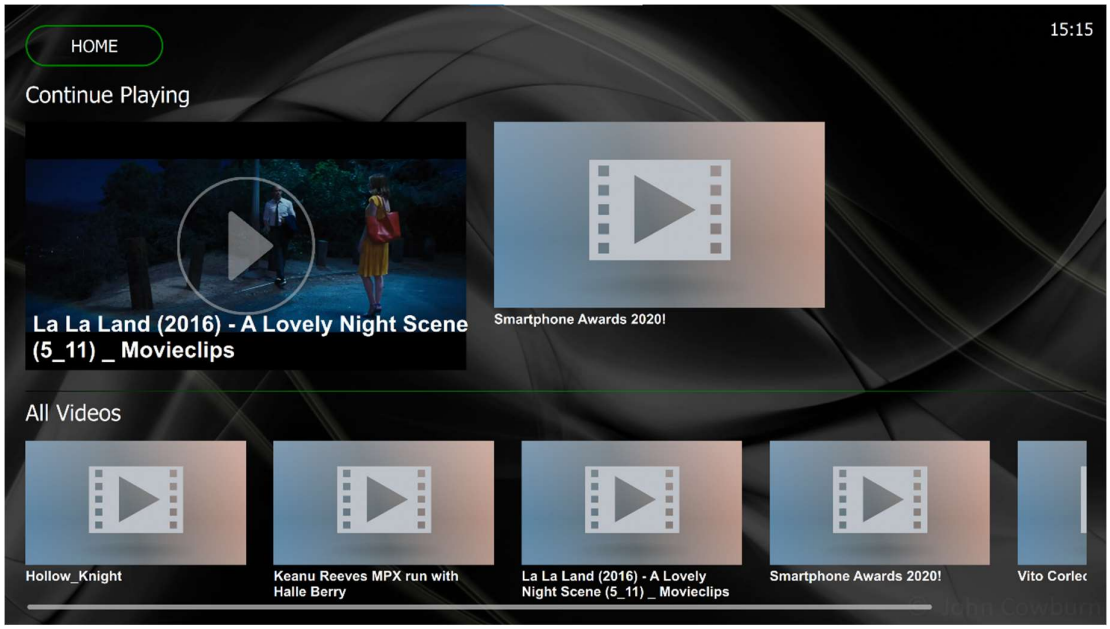
Hình 3.3 Màn hình Audio - Playlist (sản phẩm)

Màn hình Audio - Player được triển với các thành phần được sắp xếp theo bố cục được mô tả trong mục 2.3.3.3, và hành vi được mô tả trong mục 2.3.4.3. Em tự đánh giá về màn hình Audio - Playlist như sau:

- Mức độ đáp ứng về bố cục - Đáp ứng: Các thành phần trong màn hình được sắp xếp tương tự như vị trí trong mô tả.
- Mức độ đáp ứng về hành vi - Đáp ứng: Các hành vi, bao gồm chuyển màn hình và thực hiện tính năng, được thực hiện đúng như mô tả

3.1.4. Màn hình Video - Playlist

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



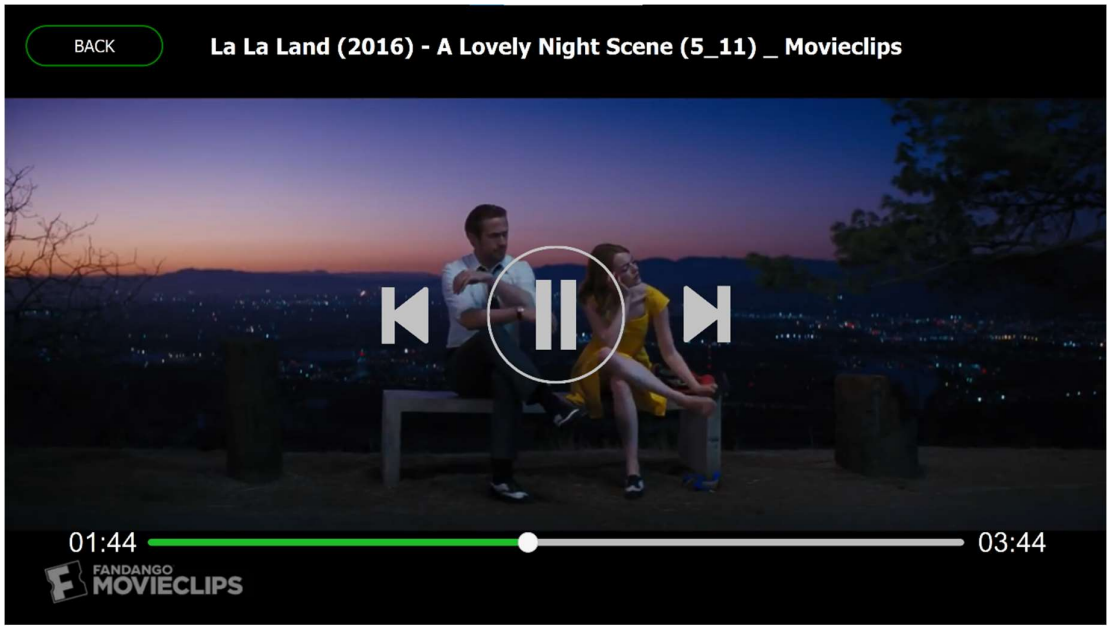
Hình 3.4 Màn hình Video - Playlist (sản phẩm)

Màn hình Video - Playlist được triển với các thành phần được sắp xếp theo bố cục được mô tả trong mục 2.3.3.4, và hành vi được mô tả trong mục 2.3.4.4. Em tự đánh giá về màn hình Video - Playlist như sau:

- Mức độ đáp ứng về bố cục - Đáp ứng: Các thành phần trong màn hình được sắp xếp tương tự như vị trí trong mô tả.
- Mức độ đáp ứng về hành vi - Đáp ứng: Các hành vi, bao gồm chuyển màn hình và thực hiện tính năng, được thực hiện đúng như mô tả

3.1.5. Màn hình Video - Player

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



Hình 3.5 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:

- Mức độ đáp ứng về bố cục - Đáp ứng: Các thành phần trong màn hình được sắp xếp tương tự như vị trí trong mô tả.
- Mức độ đáp ứng về hành vi - Đáp ứng: Các hành vi, bao gồm chuyển màn hình và thực hiện tính năng, được thực hiện đúng như mô tả

3.2. Đánh giá mức độ hoàn thiện so với yêu cầu chức năng

Về mặt chức năng, em thực hiện đánh giá theo các mục lớn tương ứng được đề ra trong mục 2.1, bao gồm chức năng phát file audio, chức năng phát file audio, và các yêu cầu phi chức năng.

3.2.1. Đánh giá mức độ hoàn thiện chức năng phát file audio

Mức độ hoàn thiện của chức năng phát file audio được đánh giá theo độ hoàn thiện của các yêu cầu được liệt kê trong mục 2.1.1. Các mức độ hoàn thiện bao gồm:

Đạt yêu cầu, Hoàn thiện một phần, Không đạt yêu cầu. Chi tiết được trình bày trong Bảng 3.1:

Bảng 3.1 Đánh giá mức độ hoàn thiện chức năng phát file audio

Mã yêu cầu	Mức độ hoàn thiện	Nhận xét
REQ_AU_1	Đạt yêu cầu	Ứng dụng có thể phát các file mp3
REQ_AU_2	Hoàn thiện một phần	Ứng dụng chỉ tự động tìm được các file mp3 trong một folder chỉ định và các folder con bên trong
REQ_AU_3	Đạt yêu cầu	Các thông tin được hiển thị
REQ_AU_4	Đạt yêu cầu	Ứng dụng có thể thay thế tên bài hát bằng tên file khi thông tin này không có sẵn trong meta-data
REQ_AU_5	Đạt yêu cầu	Ứng dụng có thể thay thế tên nghệ sĩ thể hiện bằng "Unknow Artist" khi thông tin này không có sẵn trong meta-data
REQ_AU_6	Đạt yêu cầu	Ứng dụng có thể hiển thị hình ảnh mặc định khi thông tin hình album không có sẵn trong meta-data
REQ_AU_7	Đạt yêu cầu	Các thông tin được hiển thị đủ
REQ_AU_8	Đạt yêu cầu	Thời lượng hiển thị đúng định dạng
REQ_AU_9	Đạt yêu cầu	Thời lượng đã phát hiển thị đúng định dạng
REQ_AU_10	Đạt yêu cầu	Tỷ lệ được hiển thị chính xác và có phân biệt rõ giữa phần đã phát và phần chưa phát
REQ_AU_11	Đạt yêu cầu	Các tính năng điều khiển hoạt động chính xác, đúng mong muốn
REQ_AU_12	Đạt yêu cầu	Tính năng được thực hiện chính xác
REQ_AU_13	Đạt yêu cầu	Tính năng được thực hiện chính xác
REQ_AU_14	Đạt yêu cầu	Tính năng được thực hiện chính xác
REQ_AU_15	Đạt yêu cầu	Tính năng được thực hiện chính xác
REQ_AU_16	Đạt yêu cầu	Tính năng được thực hiện chính xác

REQ_AU_17	Đạt yêu cầu	Tính năng được thực hiện chính xác
REQ_AU_18	Đạt yêu cầu	Tính năng được thực hiện chính xác
REQ_AU_19	Đạt yêu cầu	Tính năng được thực hiện chính xác
REQ_AU_20	Đạt yêu cầu	Tính năng được thực hiện chính xác

Bảng 3.1 cho thấy hầu hết các yêu cầu của chức năng phát file audio đều đã được hoàn thiện. Tuy nhiên vẫn có thiếu sót ở yêu cầu REQ_AU_2. Yêu cầu này không được hoàn thiện do nguyên nhân: Việc quét toàn bộ bộ nhớ đối với môi trường phát triển hiện tại là Windows tốn nhiều thời gian (do bộ nhớ thường có dung lượng lớn), ảnh hưởng đến thời gian khởi động của ứng dụng, nên em đã thu nhỏ phạm vi quét. Khi chuyển sang nền tảng khác, ví dụ như hệ thống giải trí trên ô tô, file được cung cấp qua thiết bị nhớ ngoài (ví dụ thẻ SD, hay USB), có thể chỉnh sửa yêu cầu sang quét toàn bộ file mp3 trong thiết bị nhớ ngoài.

3.2.2. Đánh giá mức độ hoàn thiện chức năng phát file video

Mức độ hoàn thiện của chức năng phát file video được đánh giá theo độ hoàn thiện của các yêu cầu được liệt kê trong mục 2.1.2. Các mức độ hoàn thiện bao gồm: Đạt yêu cầu, Hoàn thiện một phần, Không đạt yêu cầu. Chi tiết được trình bày trong Bảng 3.2:

Bảng 3.2 Đánh giá mức độ hoàn thiện chức năng phát file video

Mã yêu cầu	Mức độ hoàn thiện	Nhận xét
REQ_VID_1	Đạt yêu cầu	Ứng dụng có thể phát các file video định dạng mp4
REQ_VID_2	Hoàn thiện một phần	Ứng dụng chỉ tự động tìm được các file mp4 trong một folder chỉ định và các folder con bên trong
REQ_VID_3	Hoàn thiện một phần	Tên file được hiển thị chính xác, hình ảnh thumbnail không mang nhiều thông tin
REQ_VID_4	Đạt yêu cầu	Tên file đang phát được hiển thị chính xác
REQ_VID_5	Đạt yêu cầu	Thời lượng hiển thị đúng định dạng

REQ_VID_6	Đạt yêu cầu	Thời lượng đã phát hiển thị đúng định dạng
REQ_VID_7	Đạt yêu cầu	Tỷ lệ được hiển thị chính xác và có phân biệt rõ giữa phần đã phát và phần chưa phát
REQ_VID_8	Đạt yêu cầu	Các tính năng điều khiển hoạt động chính xác, đúng mong muốn
REQ_VID_9	Đạt yêu cầu	Thời gian đếm ngược đúng 5 giây, sau 5 giây, chuyển sang nội dung tiếp theo
REQ_VID_10	Đạt yêu cầu	Video phát ở chế độ toàn màn hình
REQ_VID_11	Đạt yêu cầu	Hiển thị đúng 2 video được phát gần nhất
REQ_VID_12	Đạt yêu cầu	Hiển thị đúng 2 video được phát gần nhất
REQ_VID_13	Đạt yêu cầu	Khung hình cuối cùng được phát được hiển thị trong thumbnail
REQ_VID_14	Đạt yêu cầu	Video được phát tiếp tục từ vị trí trước đó

Bảng 3.2 cho thấy hầu hết các yêu cầu của chức năng phát file video đều đã được hoàn thiện. Tuy nhiên vẫn có thiếu sót ở yêu cầu REQ_VID_2 và REQ_VID_3. Các yêu cầu này không được hoàn thiện do nguyên nhân sau:

- REQ_VID_2: Tương tự REQ_AU_2, Việc quét toàn bộ bộ nhớ đối với môi trường phát triển hiện tại là Windows tốn nhiều thời gian, nên em đã thu nhỏ phạm vi quét.
- REQ_VID_3: Em chưa tìm được giải pháp tối ưu yêu cầu này. Thông thường, trong các nền tảng video trực tuyến, hình ảnh thumbnail là do người tạo nội dung chỉ định từ đầu và là nội dung tách biệt với video (ví dụ 1 file ảnh), không phải xử lý thời gian thực để tách khung hình khỏi video. Giải pháp em từng thử nghiệm là tạo nhiều đối tượng phát file video và dừng tất cả lại ở một khung hình ngẫu nhiên, nhưng giải pháp này quá tốn RAM và yêu cầu nhiều năng lực xử lý từ CPU, là ứng dụng phản hồi chậm, đồng thời khung hình được hiển thị thường không mang nhiều ý nghĩa nên em đã bỏ qua.

3.2.3. Đánh giá mức độ hoàn thiện các yêu cầu phi chức năng

Mức độ hoàn thiện của các yêu cầu phi chức năng được đánh giá theo độ hoàn thiện của các yêu cầu được liệt kê trong mục 2.1.3. Các mức độ hoàn thiện bao gồm: Đạt yêu cầu, Hoàn thiện một phần, Không đạt yêu cầu. Chi tiết được trình bày trong Bảng 3.3:

Bảng 3.3 Đánh giá mức độ hoàn thiện các yêu cầu phi chức năng

Mã yêu cầu	Mức độ hoàn thiện	Nhận xét
REQ_ OTHER _1	Đạt yêu cầu	Thông tin giờ hiển thị chính xác

Bảng 3.3 cho thấy yêu cầu REQ_OTHER_1 đã được hoàn thiện. Tuy nhiên lượng yêu cầu phi chức năng còn rất hạn chế, còn có thể bổ sung nhiều yêu cầu khác ví dụ như các hiệu ứng hình ảnh, tổng thể chung về màu sắc, phong cách thiết kế các thành phần giao diện (mềm mại hay vuông vắn, phẳng hay nổi, sáng hay tối, ...)

3.3. Đánh giá mức độ tuân thủ mô hình MVC

Về mặt tuân thủ mô hình thiết kế MVC và sơ đồ khối 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)
- MyAudioPlayer 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).

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

- Đánh giá giao diện: Đáp ứng tốt mô tả bố cục và hành vi
- Đánh giá mức độ hoàn thiện chức năng: Hoàn thiện mức độ cao
- 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 kĩ 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 kĩ 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 kĩ năng. Các yêu cầu đặt ra đều hầu như đều đạt ở mức độ cao. Việc áp dung 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 kĩ 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] https://resources.qt.io, truy nhập cuối cùng ngày 24/12/2020

[2] https://doc.qt.io/qt-5/signalsandslots.html, truy nhập cuối cùng ngày 24/12/2020

[3] 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

[4] Kevin McArthur, *Pro PHP: Patterns, Frameworks, Testing and More*, Apress, 2008.

[5] Adam Freeman, Steven Sanderson, *Pro ASP.NET MVC 3 Framework*, Apress, 2011.

[6] W. J. Gilmore, *Easy PHP Websites*, Columbus, Ohio: W.J. Gilmore, LLC, 2009.

Bảng đối chiếu thuật ngữ Việt - Anh

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

PHỤ LỤC