

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ПРОГРАМИ МЕДІАПЛЕЄРУ
MEDIA PLAYER SOFTWARE DEVELOPMENT

Виконав: студент(ка) 4 курсу, групи КНТ-228
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерні науки

Науменко А.С.

(прізвище та ініціали)

Керівник Сердюк С.М.

(прізвище та ініціали)

Рецензент Касьян М.М.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” _____ 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТА(КИ)

Науменко Артема Сергійовича
(прізвище, ім'я, по батькові)
1. Тема проекту (роботи) Розробка програми медіаплеєру. Media Player Software Development
керівник проекту (роботи) Сердюк Сергій Микитович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від “27” квітня 2022 року № 102
2. Строк подання студентом проекту (роботи) 7 червня 2022 року
3. Вихідні дані до проекту (роботи) рекомендована література, технічне завдання
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка структури програмного забезпечення. 3. Основні рішення щодо реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво оператора. Додатки.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Сердюк С.М., доцент		
Нормоконтроль	Белова А.В., асистент		

7. Дата видачі завдання 28 березня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2–3 тижні	Розділ 1
3	Розробка архітектури програми.	4 тиждень	Розділ 2
4	Розробка програми.	5–6 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми.	7 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	8–9 тижні	Розділ 5, Додатки
7	Захист роботи.	10 тиждень	

Студент(ка)

(підпис) Науменко А.С.
(прізвище та ініціали)

Керівник проекту (роботи)

(підпис) Сердюк С.М.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
89 с., 15 табл., 28 рис., 2 дод., 12 джерел.

WINDOWS, КОДЕКИ, QT CREATOR, C++, ОПЕРАЦІЙНА СИСТЕМА,
МЕДІА, DIRECTSHOW, MEDIA FOUNDATION, МЕДІАПЛЕЄР.

Об'єкт дослідження – процеси відтворення медіафайлів.

Предмет дослідження – методи відтворення медіафайлів.

Мета роботи – розробка програми, що дозволяє відтворення медіа-контенту.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування; мова програмування C++; середовище розробки Qt Creator; фреймворк Multimedia для роботи з мультимедійною інформацією; персональний комп'ютер з процесором з частотою 1 ГГц або вище під управлінням операційної системи Windows Vista або вище.

Результати. Розроблено програмне забезпечення, що призначене для відтворення аудіо та відео файлів найбільш поширених форматів, формування плейлистів та збереження шляхів для пошуку медіа файлів.

Галузь використання – інформаційні технології.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 89 pages, 28 figures, 15 tables, 2 appendixes, 12 sources.

WINDOWS, CODECS, QT CREATOR, C++, OPERATING SYSTEM, MEDIA, DIRECTSHOW, MEDIA FOUNDATION, MEDIA PLAYER.

Object of study are a media playback processes.

Subject of study are methods of media playback.

The purpose of the work is development of a program that allows playback of media content.

Materials, methods and technical means: object-oriented programming; programming languages C++; Qt Creator development environment; Multimedia framework for working with multimedia information; personal computer with a processor 1 GHz or higher running Windows Vista or higher.

Results. Software has been developed to play audio and video files of the most common formats, create playlists and save ways to search for media files.

The field of use is Information Technology.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ	9
1 Аналіз предметної області	11
1.1 Огляд форматів медіа-контейнерів	11
1.2 Огляд програмних аналогів	22
1.2.1 Windows Media Player	23
1.2.2 Media Player Classic та Media Player Classic HC	25
1.2.3 VideoLAN Client	27
1.3 Розробка технічного завдання	29
1.3.1 Підстава для розробки	29
1.3.2 Мета розробки та її призначення	30
1.3.3 Основні вимоги до програмного продукту	30
1.3.4 Вимоги до технічних засобів, операційної системи і мови програмування	31
1.3.5 Вимоги до інформаційної та програмної сумісності	32
1.3.6 Вимоги до програмної документації	33
2 Розробка структури програмного забезпечення	34
2.1 Аналіз технічного завдання	34
2.2 Розробка функціональної структури програми	36
2.3 Обґрунтування вибору мови програмування	38
3 Основні рішення щодо реалізації компонентів системи	41
3.1 Розробка класів Desktop-застосунку	41
3.2 Розробка графічного інтерфейсу користувача	45
4 Керівництво програміста	53
4.1 Призначення та умови застосування програми	53
4.2 Характеристики програми	53
4.3 Вхідні та вихідні дані	53
4.4 Опис необхідних модулів та класів	54

- 4.5 Повідомлення програмісту 61
- 5 Керівництво оператора..... 62
 - 5.1 Призначення та умови виконання програми 62
 - 5.2 Виконання програми 62
 - 5.3 Повідомлення оператору 64
- Висновки..... 65
- Перелік джерел посилання 66
- Додаток А. Текст програми 67
- Додаток Б. Слайди презентації..... 80

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interfaces;
MIME	– Multipurpose Internet Mail Extensions;
Гб	– гігабайт;
ГГц	– гігагерц;
ОС	– операційна система;
ПК	– персональний комп'ютер.

ВСТУП

На сьогоднішній день існує багато операційних систем (ОС) та мобільних платформ. Компанії, що займаються розробкою та підтриманням цих систем, презентують різноманітні оновлення багатьох компонентів. Не залишаються без уваги й медіа-плеєри, адже різноманітність форматів аудіо та відео файлів зростає, оновлюються версії операційних систем, їх властивості та можливості.

Медіа-файли – це фотографії, музика, аудіо, відео та документи. Комп'ютерні програми або програми можуть читати та працювати з цифровим файлом після його кодування в процесі збереження. Наприклад, формати документів можна читати та редагувати у програмах обробки текстів, таких як Microsoft Word.

Багато відео-форматів, таких як відеокамера, файли DVD, файли Quicktime, відео Windows і кілька форматів високої чіткості, перед відтворенням необхідно перетворити на певний формат, і їх не можна відтворювати безпосередньо в спочатку створеному або збереженому форматі.

Програмне забезпечення медіа-плеєра – це тип прикладного програмного забезпечення для відтворення мультимедійних комп'ютерних файлів, таких як аудіо та відеофайли.

У найпоширеніших операційних системах є, як мінімум, один медіа-плеєр за замовчуванням. Наприклад, ОС Windows поставляється з Windows Media Player, Microsoft Movies & TV та Groove Music, а macOS – з QuickTime Player та Music. Дистрибутиви Linux поставляються з різними медіаплеєрами, такими як SMPlayer, Amarok, Audacious, Banshee, MPlayer, mpv, Rhythmbox, Totem, медіаплеєр VLC та xine. Android поставляється з Google Play Music для аудіо та Google Photos для відео, а виробники смартфонів, такі як Samsung можуть встановлювати спеціальне програмне забезпечення.

Найкращим програмним продуктом є той, що легко підтримувати, той, що приносить найбільш можливу користь користувачам, допомагає у вирішенні проблем, надаючи необхідний функціонал. Звичайному користувачу може бути складно обрати медіа-плеєр, адже кожен аналог вирізняється з-поміж інших функціями, можливостями, інноваціями та пропозиціями.

Отже, створення простого, універсального медіа-плеєру, що є базовим інструментом для маніпулювання медіа-контентом для будь-якого користувача, найбільш улюблена операційна система якого є Windows, є актуальним рішенням.

Мета даної кваліфікаційної роботи бакалавра полягає у розробці програмного забезпечення, для відтворення медіа-контенту з використанням сучасних спеціалізованих бібліотек.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд форматів медіа-контейнерів

Контейнерний формат – це спосіб зберігання цифрової інформації у комп'ютерній системі. Іншими словами, відео контейнерний формат – це тип формату файлу, який містить різні типи даних, стиснутих стандартизованими кодеками. Формати контейнерів насправді є обгортками, оскільки вони не визначають, який кодек використовує формат контейнера, а швидше визначають, як відео, аудіо та інші дані зберігаються в контейнері.

Двома основними мультимедійними форматами є формати мультимедійних контейнерів та формати єдиного кодування.

Мультимедійний контейнер – формат файлів, що може містити дані різних типів та дозволяє зберігати аудіо, відео і текстову інформацію в єдиному файлі.

Комп'ютерні системи використовують контейнерні формати для декодування файлів, щоб користувачі могли бачити дані, що зберігаються у конкретному файлі. Це стосується аудіо- та відеофайлів. Перший загальний формат контейнера називався форматом обміну файлу або IFF.

Формати мультимедійних контейнерів містять аудіо кодеки, які є спеціалізованим програмним забезпеченням для кодування та декодування. Кодек використовується в процесі форматування, щоб змінити звукову частину файлів цифрових даних для їх найбільш ефективного зберігання. Аудіо кодек також запрограмований для координації з відео, тому що вони синхронізуються при перегляді.

Існує величезна кількість форматів медіа-контейнерів. Деякі підтримують лише аудіо, інші підтримують і аудіо, і відео. Найбільш поширеними типами форматів контейнерів є 3GPP, ASF та AVI [1].

Партнерський проект третього покоління, або 3GPP, є контейнерним форматом, що використовується в основному в мобільних телефонах. 3GPP є спільним проектом Європейського інституту телекомунікаційних стандартів,

Китайської асоціації стандартів зв'язку, Асоціації телекомунікаційних технологій та Альянсу галузевих рішень для телекомунікацій. Ця багатостороння співпраця забезпечує глобальний доступ та єдиний формат для відтворення та запису відео на мобільних телефонах [2].

Найбільш часто, для мультимедіа в Інтернеті, використовуються такі контейнери, як MPEG-4 (MP4), QuickTime Movie (MOV) та формат аудіофайлу Wavefile (WAV). Однак користувач також може зустріти MP3, Ogg, WebM, AVI та інші формати. Проте не всі з них широко підтримуються браузерами; деяким комбінаціям контейнера з кодеком іноді присвоюються власні розширення файлів і типи MIME (медіа тип, який є стандартом, описує природу та формат документа, файлу або набору байтів) для зручності або їх повсюдного поширення. Наприклад, файл Ogg, що містить тільки звукову доріжку Opus, іноді називається файлом Opus і навіть може мати розширення .opus. Але насправді це лише файл Ogg [2].

Розглянемо найбільш поширені типи медіа контейнерів (табл. 1.1).

Таблиця 1.1 – Найбільш поширені типи медіа контейнерів

Скорочене ім'я кодека	Повне ім'я кодека	Сумісність із браузером
1	2	3
3GP	Third Generation Partnership	Firefox для Android
ADTS	Audio Data Transport Stream	Firefox Доступний лише в середовищі носія базової операційної системи.
FLAC	Free Lossless Audio Codec	Chrome 56, Edge 16, Firefox 51, Safari 11
MPEG / MPEG-2	Moving Picture Experts Group (1 and 2)	—
MPEG-4 (MP4)	Moving Picture Experts Group 4	Chrome 3, Edge 12, Firefox, Internet Explorer 9, Opera 24, Safari 3.1
Ogg	Ogg	Chrome 3, Firefox 3.5, Edge 17 (тільки desktop), Internet Explorer 9, Opera 10.50

Продовження табл. 1.1

1	2	3
QuickTime (MOV)	Apple QuickTime movie	Тільки старіші версії Safari, а також інші браузері, що підтримують модуль Apple QuickTime.
WebM	Web Media	Chrome 6, Edge 17 (тільки desktop), Firefox 4, Opera 10.6, Safari 14.1 (macOS), Safari 15 (iOS). Edge вимагає інсталяції Web Media Extensions.

Якщо не вказано інше, браузер, вказаний у табл.1.1, має сумісність як із мобільним, так і з настільним браузером. Також мається на увазі підтримка лише самого контейнера, а не якихось конкретних кодеків.

Медіа-контейнер 3GP або 3GPP використовується для інкапсуляції аудіо та/або відео, які спеціально призначені для передачі стільниковими мережами для використання на мобільних пристроях. Формат був розроблений для використання на мобільних телефонах 3G, але його все ще можна використовувати у більш сучасних телефонах та мережах. Тим не менш, розширення смуги пропускання та збільшення обмежень на передачу даних у більшості мереж зменшили потребу у форматі 3GP. Однак цей формат, як і раніше, використовується для більш повільних мереж та для телефонів з низькою продуктивністю.

Цей формат медіа-контейнера є похідним від базового формату медіа-файлів ISO та MPEG-4, але він спеціально оптимізований для сценаріїв з нижчою пропускнуою здатністю.

Типи MIME [4], що наведені у табл. 1.2 є основними типами для медіа-контейнера 3GP. Інші типи можуть використовуватися залежно від конкретного кодека або кодеків.

Таблиця 1.2 – Типи MIME для медіа-контейнера 3GP

Audio	Video
audio/3gpp	video/3gpp
audio/3gpp2	video/3gpp2
audio/3gp2	video/3gp2

Крім того, можна додати параметр `codecs` до рядка типу MIME, щоб вказати, які кодеки використовуються для аудіо- та/або відео доріжок, а також для додаткових відомостей про профіль, рівень та/або інші особливості конфігурації кодека.

Медіа-контейнер Audio Data Transport Stream (ADTS) – це формат контейнера, зазначений у частині 3 MPEG-4 для аудіо даних, призначений для використання в потоковому аудіо, наприклад, для інтернет-радіо. По суті це майже пустий потік аудіо даних AAC, що складається з кадрів ADTS з мінімальним заголовком.

Тип MIME, який використовується для ADTS, залежить від того, які аудіо кадри містяться всередині. Якщо використовуються кадри ADTS, слід використовувати тип MIME `audio/aac`. Якщо аудіо кадри мають формат MPEG-1/MPEG-2 Audio Layer I, II або III, тип MIME слід використовувати `audio/mp3`. Аудіо кодеки, які підтримуються ADTS наведені у табл. 1.3. Підтримка Firefox для AAC залежить від медіа інфраструктури ОС і є доступною, якщо її підтримує ОС.

Таблиця 1.3 – Аудіо кодеки, що підтримуються ADTS медіа-контейнером

Кодек	Підтримуємий браузер			
	Chrome	Edge	Firefox	Safari
AAC	-	-	+	-
MP3	-	-	+	-

Медіа-контейнер Free Lossless Audio Codec (FLAC) – це безкоштовний аудіо кодек без втрат. Існує також пов'язаний з ним простий формат контейнера, що також має назву FLAC, який може містити це аудіо. Файли FLAC можуть містити лише аудіо дані FLAC.

Аудіо кодеки, які підтримуються FLAC наведені у табл. 1.4.

Таблиця 1.4 – Аудіо кодеки, що підтримуються FLAC медіа-контейнером

Кодек	Підтримуємий браузер			
	Chrome	Edge	Firefox	Safari
FLAC	+	+	+	+

Формати файлів MPEG-1 та MPEG-2 практично ідентичні. Ці формати, створені експертною групою з рухомих зображень (MPEG), широко використовуються на фізичних носіях, у тому числі як формат відео на DVD-носіях.

Найпоширеніше використання формату MPEG в Інтернеті – це звукові дані Layer_III/MP3. Результируючий файл є надзвичайно популярним файлом MP3, який використовується цифровими музичними пристроями по всьому світу. MPEG-1 та MPEG-2 широко не використовуються у веб-контенті іншим способом.

Основні відмінності між MPEG-1 та MPEG-2 полягають у форматах мультимедійних даних, а не у форматі контейнера. MPEG-1 був представлений у 1992 році; MPEG-2 з'явився у 1996 році.

Аудіо та відео кодеки, які підтримуються MPEG-1 та MPEG-2 наведені у табл. 1.5.

Медіа-контейнер MPEG-4 (MP4) – це остання версія формату MPEG. Існують дві версії формату, визначені у частинах 1 та 14 специфікації. MP4 сьогодні є популярним контейнером, так як він підтримує кілька кодеків, що найчастіше використовуються, і широко підтримується.

Таблиця 1.5 – Аудіо та відео кодеки, що підтримуються MPEG-1 та MPEG-2 медіа-контейнером

Кодек	Підтримуємий браузер			
	Chrome	Edge	Firefox	Safari
Відео кодеки				
MPEG-1 Part 2	-	-	-	-
MPEG-2 Part 2	-	-	-	-
Аудіо кодеки				
MPEG-1 Audio Layer I	-	-	-	-
MPEG-1 Audio Layer II	-	-	-	-
MPEG-1 Audio Layer III (MP3)	-	-	+	-

Оригінальний формат файлу MPEG-4 Part 1 було представлено 1999 року; формат версії 2, визначений у частині 14, було додано у 2003 році. Формат файлу MP4 є похідним від базового формату мультимедійних файлів ISO, який безпосередньо є похідним від формату QuickTime, розробленого Apple.

Такі типи MIME, як аудіо/mp4 або відео/mp4, є основними типами для медіаконтейнера MPEG-4. Інші типи MIME можуть використовуватись залежно від конкретного кодека або кодеків, які використовуються у контейнері. Крім того, можна додати параметр codecs до рядка типу MIME, щоб вказати, які кодеки використовуються для аудіо- та/або відео доріжок, а також для додаткових відомостей про профіль, рівень та/або інші особливості конфігурації кодека.

Аудіо та відео кодеки, які підтримуються MPEG-4 наведені у табл. 1.6.

Медіа-контейнерний формат Ogg – це безкоштовний та відкритий формат, який підтримується Xiph.org Foundation. Платформа Ogg також визначає формати мультимедійних даних, такі як відеокодек Theora та аудіокодеки Vorbis та Opus. Документи Xiph.org про формат Ogg доступні на

їхньому веб-сайті.

Таблиця 1.6 – Аудіо та відео кодеки, що підтримуються MPEG-4 медіа-контейнером

Кодек	Підтримуємий браузер			
	Chrome	Edge	Firefox	Safari
Відео кодеки				
AVC (H.264)	-	-	+	-
AV1	-	-	+	-
H.263	-	-	-	-
MPEG-4 Part 2 Visual	-	-	-	-
VP9	-	-	+	-
Аудіо кодеки				
AAC	-	-	-	-
FLAC	-	-	-	-
MPEG-1 Audio Layer III (MP3)	-	-	+	-
Opus	-	-	+	-

Хоча Ogg існує вже давно, він так і не отримав широкої підтримки, необхідної для того, щоб обрати його першим серед медіа контейнерів. Користувачі зазвичай обирають WebM, хоча бувають випадки, коли Ogg корисно запропонувати, наприклад, коли необхідно підтримувати старіші версії Firefox та Chrome, які ще не підтримують WebM. Наприклад, Firefox 3.5 та 3.6 підтримують Ogg, але не WebM.

Тип MIME application/ogg можна використовувати у випадку, коли не обов'язково знати, чи містить медіа файл аудіо або відео. Якщо додати параметр codecs до рядка типу MIME, то таким чином можна вказати, які кодеки використовуються для аудіо- та/або відео доріжок, а також для додаткового опису форматів мультимедіа доріжок.

Аудіо та відео кодеки, які підтримуються Ogg наведені у табл. 1.7.

Таблиця 1.7 – Аудіо та відео кодеки, що підтримуються Ogg медіа-контейнером

Кодек	Підтримуємий браузер			
	Chrome	Edge	Firefox	Safari
Відео кодеки				
Theora	+	+	+	-
VP8	+	+	+	-
VP9	+	+	+	-
Аудіо кодеки				
FLAC	+	+	+	-
Vorbis	+	+	+	-
Opus	+	+	+	-

Формат файлів QuickTime (QTFF, QT або MOV) був створений Apple для використання однойменної мультимедійної структури. Розширення .mov для цих файлів, пов'язане з тим, що цей формат спочатку використовувався для фільмів і називався форматом «QuickTime movie». Хоча QTFF послужив основою формату файлу MPEG-4, між ними є відмінності [3].

Файли QuickTime підтримують будь-які дані, засновані на часі, включаючи аудіо- та відеофайли, текстові доріжки тощо. Файли QuickTime в основному підтримуються macOS, але протягом ряду років QuickTime був доступний для Windows. Проте з початку 2016 року Apple більше не підтримує QuickTime для Windows і його не слід використовувати через відомі проблеми з безпекою. Однак програвач Windows Media тепер має вбудовану підтримку файлів QuickTime версії 2.0 та раніше. Підтримка пізніших версій QuickTime вимагає сторонніх застосунків.

У Mac OS платформа QuickTime підтримувала не лише відеофайли та кодеки у форматі QuickTime, але й широкий спектр популярних та спеціальних

аудіо- та відеокодеків, а також формати нерухомих зображень. Через QuickTime програми Mac (включаючи веб-браузери, через модуль QuickTime, що підключається або пряму інтеграцію QuickTime) могли читати та записувати аудіо формати, включаючи AAC, AIFF, MP3, PCM та Qualcomm PureVoice; та відео формати, включаючи AVI, DV, Pixlet, ProRes, FLAC, Cinepak, 3GP, H.261-H.265, MJPEG, MPEG-1 та MPEG-4 Part 2, Sorenson та багато інших.

Крім того, для QuickTime є ряд сторонніх компонентів, деякі з яких додають підтримку додаткових кодеків. Підтримка QuickTime насамперед доступна на пристроях Apple і не використовується більше широко в Інтернеті. Сама фірма Apple зараз взагалі використовує для відео формат MP4. Крім того, фреймворк QuickTime протягом деякого часу застарів на Mac і більше не доступний, починаючи з MacOS 10.15 Catalina.

Тип MIME video/quicktime є основним типом мультимедійного контейнера QuickTime. Варто відзначити, що QuickTime (середовище мультимедіа в операційних системах Mac) підтримує широкий спектр контейнерів та кодеків, тому насправді він підтримує багато інших типів MIME.

Аудіо та відео кодеки, які підтримуються QuickTime наведені у табл. 1.8.

Таблиця 1.8 – Аудіо та відео кодеки, що підтримуються QuickTime медіа-контейнером

Кодек	Підтримуваний браузер			
	Chrome	Edge	Firefox	Safari
1	2	3	4	5
Відео кодеки				
AVC (H.264)	-	-	-	+
Cinepak	-	-	-	+
Component Video	-	-	-	+

Продовження табл. 1.8

1	2	3	4	5
DV				+
MPEG-2				+
Motion JPEG				+
Sorenson Video 3				+
Аудіо кодеки				
AAC	-	-	-	+
Apple Lossless (ALAC)	-	-	-	+
Microsoft ADPCM	-	-	-	+

Формат аудіо файлу Waveform (WAVE), який зазвичай називають WAV через розширення імені файлу .wav, є форматом, розробленим Microsoft і IBM для зберігання даних аудіо потоку. Він є похідним від формату файлу обміну ресурсами (RIFF) і є схожим з іншими форматами, такими як AIFF від Apple. Реєстр кодеків WAV можна знайти у RFC 2361; однак, оскільки майже всі файли WAV використовують лінійний PCM, підтримка інших кодеків незначна.

Аудіо кодеки, які підтримуються WAVE наведені у табл. 1.9 [5].

Таблиця 1.9 – Аудіо кодеки, що підтримуються WAVE медіа-контейнером

Кодек	Підтримуємий браузер			
	Chrome	Edge	Firefox	Safari
ADPCM	+	-	+	+
GSM 06.10	+	+	-	+
LPCM	-	-	+	+
MP3	+	-	+	+
μ-Law	+	-	+	+

WebM (Web Media) – це формат, заснований на Matroska, який розроблений спеціально для використання у сучасних веб середовищах. Він повністю заснований на безкоштовних та відкритих технологіях і переважно використовує кодеки, які, у свою чергу, є теж безкоштовними та відкритими.

WebM був вперше представлений у 2010 році і зараз широко підтримується. Для підтримки відеокодеків VP8 та VP9 та аудіо кодеків Vorbis та Opus потрібні сумісні реалізації WebM. Формат контейнера WebM та необхідні для нього кодеки доступні за відкритими ліцензіями. Для використання інших кодеків може знадобитися ліцензія.

Аудіо та відео кодеки, які підтримуються WebM наведені у табл. 1.10.

Таблиця 1.10 – Аудіо та відео кодеки, що підтримуються WebM медіа-контейнером

Кодек	Підтримуємий браузер			
	Chrome	Edge	Firefox	Safari
Відео кодеки				
AV1	+	+	+	+
VP8	+	+	+	+
VP9	+	+	+	+
Аудіо кодеки				
Vorbis	+	+	+	+
Opus	+	+	+	+

Існує кілька факторів, які слід враховувати, вибираючи оптимальний контейнер для медіа. Відносна важливість кожного залежить від потреб користувача, вимог до ліцензії та вимог до сумісності вашої цільової аудиторії. Наприклад, якщо користувач збирається маніпулювати медіа даними, використання стисненого формату може підвищити продуктивність, тоді як використання стисненого формату без втрат принаймні запобігає накопиченню шуму, оскільки артефакти стиснення множаться з кожним повторним

стисненням.

Рекомендації щодо вибору оптимального медіа контейнера наведені у табл. 1.11 [3].

Таблиця 1.11 – Рекомендації щодо вибору оптимального медіа контейнера

Потреба користувача	Рекомендація
Тільки аудіо файли	
Відтворення стиснутих файлів загального призначення	MP3 (MPEG-1 Audio Layer III)
Стиснуті файли без втрат	FLAC з ALAC
Нестиснені файли	WAV
Відео файли	
Відео загального призначення, бажано у відкритому форматі	WebM з MP4
Відео загального призначення	MP4 з WebM
Висока ступінь стиснення, оптимізована для повільних з'єднань	3GP з MP4
Сумісність зі старими пристроями/браузерами	QuickTime з AVI і/або MPEG-2

Розглянувши найпоширеніші формати медіа-контейнерів перейдемо до огляду програм-аналогів застосунку, що розробляється у даній роботі.

1.2 Огляд програмних аналогів

Найпершим кроком у проектуванні та розробці є аналіз існуючих програмних засобів. На сьогоднішній день існує багато аналогів програмного забезпечення, що проектується, проте, для порівняння було взято найвідоміші медіа плеєри, функціональні можливості яких найкраще відповідають можливостям, що необхідно якісно та швидко реалізувати. Усі розглянуті

програмні засоби є безкоштовними.

1.2.1 Windows Media Player

Програваач Windows Media Player (WMP) — це програмний застосунок Microsoft, який використовується для відтворення, зберігання та організації цифрового звуку, зображень та відео [6]. Попередні версії WMP були випущені для ПК з різними операційними системами, включаючи Windows Mobile.

WMP використовує формати файлів WMV, WMA та ASF для кодування аудіо та відео, завантажених з Інтернету або скопійованих з компакт-дисків або інших носіїв. Останні версії WMP також включають можливість синхронізації цифрового мультимедіа з портативним медіа плеєром, таким як Zune від Microsoft або Zen від Creative, а також можливість купувати або купувати напрокат контент в інтернет-магазинах мультимедіа. Оскільки WMP використовує DRM Windows Media від Microsoft, а не FairPlay DRM, файли, закодовані за допомогою WMP, несумісні з iPod.

Програваач Windows Media пропонується безкоштовно для завантаження з веб-сайту Microsoft, за винятком мобільної версії програмного забезпечення, яка постачається лише з пристроями Windows Mobile. Існує декілька плагінів, що розширюють можливості медіаплеєру, встановлюючи фільтри для декодування обробки звуку в визначеному плагіном форматі.

Існують функції, які розвивалися з кожною версією медіапрограваача Windows. Крім того, що він є медіаплеєром, він може забезпечувати трансляцію радіостанцій за допомогою Інтернету. Він здатний впорядковувати та шукати медіафайли на жорсткому диску. Він може не тільки відтворювати DVD-диски та компакт-диски, але також може створювати власні компакт-диски та копіювати файли на портативні пристрої. Він також може копіювати музику з DVD та компакт-дисків та допомагати в синхронізації контенту з пристроями. Він може впорядковувати аудіо- та відеофайли за різними категоріями, такими як жанр, рік, рейтинг, альбом або виконавець. Однією з важливих можливостей

медіапрогравача Windows є його здатність створювати списки відтворення. Починаючи з Windows Media Player 9, надається розширене кодування. Користувачі можуть захоплювати та кодувати мультимедіа з таких пристроїв, як камери, за допомогою медіапрогравача Windows. Створення фільмів також можливе за допомогою медіапрогравача Windows, можливості якого немає в інших популярних програвачах мультимедіа.

Головна сторінка даного медіа плеєру зображена на рис. 1.1.

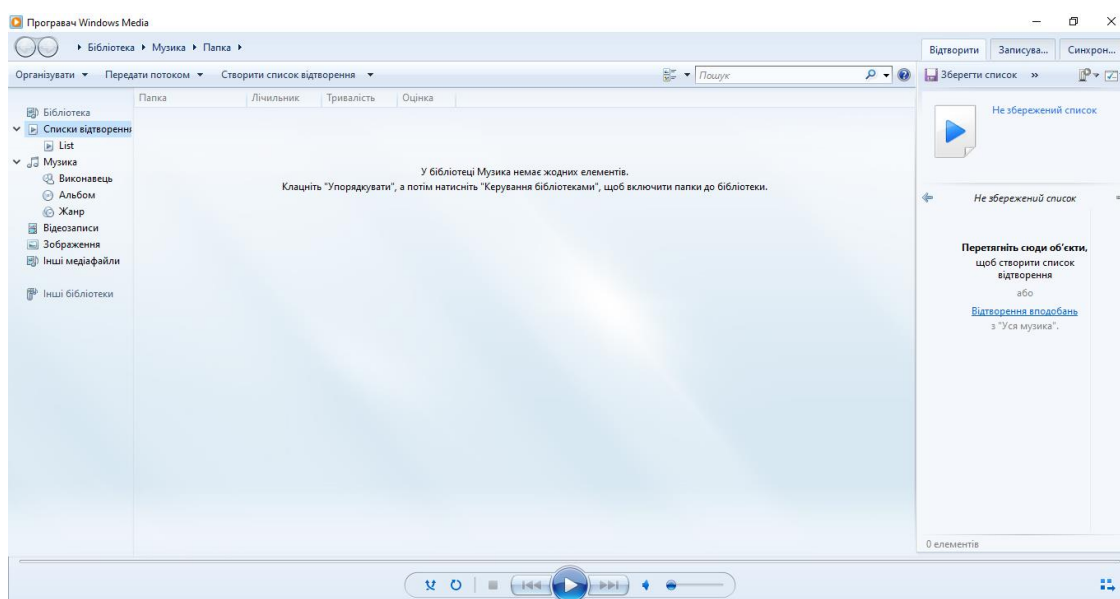


Рисунок 1.1 – Головна сторінка Windows Media Player

До переваг даного плеєру можна віднести:

- керування бібліотекою шляхом редагування плей-листів, сортування медіа файлів за різними параметрами;
- можливість видозмінити загальний вигляд програми, налаштувати елементи управління в зручному варіанті;
- копіювання медіа файлів з компакт-дисків та запис дисків в Audio CD форматі.

До недоліків даного плеєру можна віднести [7]:

- досить високі вимоги до системних ресурсів, робота програми впливає на швидкість відгуку інших програм, що виконуються на ПК;

- відсутні функції пришвидшеного відтворення медіа файлу, покадрової зміни точки відтворення відео;
- нездатність правильно відтворювати потокове відео.

1.2.2 Media Player Classic та Media Player Classic HC

Програваач Media Player Classic та Media Player Classic Home Cinema (MPC HC) – це безкоштовні відео- та аудіо плеєри з відкритим вихідним кодом для Windows. MPC-HC заснований на оригінальному проєкті Guliverkli та містить безліч додаткових функцій та виправлень помилок [8]. Офіційне розвинення програм цієї серії продовжувалося до 2006 року, а інтерфейс MPC HC медіа плеєру (рис.1.2) має вигляд, аналогічний інтерфейсу Windows Media Player версії 6.4, проте з іншою кодовою базовою. У інтерфейсі відсутні жодні складні інструменти управління бібліотекою, візуалізації, що можуть надмірно завантажувати процесор, тобто це простий, функціональний інструмент для відтворення аудіо- та відеофайлів.



Рисунок 1.2 – Інтерфейс медіа плеєру Media Player Classic Home Cinema

Але в той час, як інтерфейсу вже багато років, список функцій Windows Media Player дуже актуальний. Користувач має підтримку списків відтворення, субтитрів та титрів, а також відтворення DVD. Крім того, застосунок буде працювати з будь-яким кодеком, який захоче встановити користувач. Він навіть перевершує останню версію програвача Windows Media з перемиканням аудіопотоків, поєднаннями клавіш, що налаштовуються, і дистанційним керуванням через HTTP.

Але що робить Media Player Classic особливо привабливим, то це його надзвичайно малий розмір. Він є окремим виконуваним файлом (установка не потрібна), який займає близько 4 МБ на диску. Його обсяг пам'яті під час запуску становить лише 8 МБ.

Крім того, основні переваги медіа плеєру Media Player Classic наступні:

- можливість відтворення пошкоджених файлів, або не повністю завантажених на момент відтворення у форматі .AVI;
- підтримка зчитування субтитрів;
- підтримка керування з командного рядку;
- відтворення відеороликів в форматі Adobe Flash.

Основні недоліки даного медіа плеєру:

- відсутність еквалайзера;
- відсутня функція налаштування яскравості зображення;
- не є багатоплатформним.

Нововведення Media Player Classic Home Cinema, що відсутні у версії Media Player Classic:

- підтримка декількох моніторів;
- опція видалення тремтіння відеокадру;
- пульт дистанційного керування для пристроїв Android;
- відтворення й запис телевізійних програм, якщо на ПК встановлено ТВ-тюнер.

1.2.3 VideoLAN Client

VideoLAN Client (VLC) – кросплатформний програвач мультимедіа (рис. 1.3), що розроблюється проєктом VideoLAN. Даний медіа плеєр має безліч мов інтерфейсу, дизайн застосунку легко змінюється завдяки підтримці skins2 (налаштування власного стилю користувацького інтерфейсу), користувачі, що користуються терміналом, працюють з інтерфейсом ncurses (спосіб представлення даних у вигляді інтерфейсу звичайної програми у терміналі)[9].



Рисунок 1.3 – Інтерфейс медіаплеєру VLC

Переваги VLC плеєру наступні:

- підтримка багатьох форматів, наприклад. flv (флеш-відео), mpeg, avi, 3gp;
- VLC також відтворює файли .ram з використанням реальних DLL-плеєрів, якщо використовується ОС Windows. Однак підтримуються не всі аудіо та відео версії .ram;
- якісне відтворення «битих» AVI. Він може зчитувати розрізнені кадри та намагатися відтворити їх послідовно;

- забезпечує можливість встановити різні швидкості відтворення відео за допомогою простих елементів керування на панелі інструментів;
- плавне відтворення DVD;
- забезпечує захоплення відео з веб-камер;
- графічний еквалайзер на 10 повзунків, аудіо ефекти (нормалізація, синхронізація, придушення шумів та інші);
- програму можна запускати як утиліту з USB-флеш-накопичувача.

До недоліків VLC слід віднести наступні:

- для різних швидкостей відео не можна відтворювати у зворотному напрямку;
- відкритий вихідний код, що приводить до того, що іноді програма поводить себе хаотично, з раптовим перериванням роботи програми;
- відсутність списку MRU (останні використані файли);
- відсутність налаштування еквалайзера для звуку. Присутній тільки набір опцій меню, таких як техно, бас, класика і т. д., але користувач не має змоги встановити свій власний еквалайзер.

Результати аналізу найвідоміших аналогів, що є легкими в опануванні наведені у табл. 1.12.

Таблиця 1.12 – Порівняльна характеристика відомих медіа плеєрів

Назва	VLC Media Player	Media Player Classic HC	Windows Media Player
1	2	3	4
Вартість	Безкоштовно	Безкоштовно	Безкоштовно
Потокове відтворення	Так	Так	Так
Підтримка відео	Так	Так	Так
Менеджер завантажень	Ні	Ні	Ні
Віддалене керування	Так	Так	Так

Продовження табл. 1.12

1	2	3	4
Візуалізація	Так	Ні	Так
Безкоштовність	Так	Так	Так
Сумісність з Windows	Так	Так	Так

Було з'ясовано, що у наведених у табл. 1.12 аналогів обов'язково присутній наступний функціонал :

- відтворення медіа файлів з різних тек;
- можливість відтворення відео у повноекранному режимі;
- формування плейлистів для відтворення великої кількості медіа файлів поспіль;
- регулювання швидкості, гучності відтворення медіа файлів;
- висвітлювання інформації про поточний медіа файл;
- пересування повзунка для вказання конкретного моменту часу, з якого необхідно продовжити відтворення медіа файлу.

Кожен з наведених застосунків підтримує наступні аудіо формати: MP3, WMA, RealAudio, AAC, AC-3, FLAC, ALAC, а також такі відео формати: MPEG-1, MPEG-2, MPEG-4, WMV, RealVideo.

Отже, універсальний медіа плеєр, що розроблюється має реалізувати базову функціональність, наявну у рішень-аналогів, що сприятиме швидкодії та зручності у його використанні.

1.3 Розробка технічного завдання

1.3.1 Підстава для розробки

Підставою для розробки служить завдання на дипломну кваліфікаційну роботу бакалавра на тему "Розробка програми медіаплеєру", затверджене наказом № 102 від 27 квітня 2022 р. по НУ «Запорізька політехніка».

1.3.2 Мета розробки та її призначення

Метою є розробка програмного забезпечення, що призначене для відтворення файлів аудіо та відео форматів, а також маніпулювання медіа-даними.

1.3.3 Основні вимоги до програмного продукту

Програма, що розроблюється повинна бути простою у використанні – мати інтуїтивно зрозумілий інтерфейс та розміщення усіх елементів управління згідно очікувань звичайного користувача. Програма повинна мати доступ лише до директорій, вказаних користувачем, відслідковувати медіа файли, що може відтворити та завантажувати їх в поточний перелік. Програмне забезпечення має зберігати шляхи, дозвіл на пошук медіа-контенту яких надав користувач.

Функціональні вимоги до медіа плеєру наступні:

- вказування шляхів, до яких матиме доступ медіа плеєр;
- автоматичне зберігання шляхів в файлі розширення .txt;
- перелік контейнерів файлів, що підтримується в модулі Multimedia в реалізації Media Foundation – 3GP, ASF, AVI, MP3, MPEG-4 [1];
- перелік аудіокодеків, що підтримується – AAC, MP3, GSM 6.10, а також звукові файли в форматі WMA;
- перелік відеокодеків, що підтримується Відео DV, H.264, MJPEG, MPEG-4, частина 2, MPEG-4 v1/v2/v3 та відеофайли в форматі WMV;
- використання типу медіа-потoku PCM;
- підтримання функції пересування повзунка для вказання конкретного моменту часу, з якого необхідно продовжити відтворення медіа файлу;
- плейлист формується автоматично, завантажуючи усі знайдені медіа файли в визначених користувачем медіа-тек;

- забезпечення можливості переходу в повноекранний режим під час перегляду відеофайлу;
- висвітлювання інформації про поточну композицію;
- регулювання гучності;
- перемикання наступного або попереднього файлу;
- функція призупинення відтворення;
- повторювання вказаного користувачем медіа файлу або всіх існуючих в поточному переліку;
- можливість змінити швидкість відтворення медіа файлу.

Було виділено такі вимоги до початкових даних:

- формати медіа файлів мають підтримуватися операційною системою, в іншому випадку, програма не зможе відтворити бажаний файл;
- медіа файли мають містити коректну назву та формат, не бути пошкодженими для коректного відтворення;
- файл зі шляхами може існувати, або може створюватися та змінюватися під час взаємодії користувача з програмою.

Вихідні результати роботи застосунку мають бути наступними:

- відтворення усіх файлів з усіх шляхів, встановлених користувачем;
- успішно застосовані користувацькі налаштування;
- не виявлено збійних ситуацій;
- відтворені медіа файли без системних помилок;
- відредагований або створений файл, що зберігає шляхи.

1.3.4 Вимоги до технічних засобів, операційної системи і мови програмування

Програмне забезпечення, що описується даним технічним завданням найпростіше реалізувати можливостями операційної системи Windows, починаючи з версії Windows 7. Сімейство даних ОС зручне у використанні, має власний набір базових функцій інтерфейсів (Windows API), що дозволяють без

використання додаткових бібліотек створювати складні, функціональні системи. Більшість бібліотек, наприклад, OpenAL, PortAudio, Audiere, Simple DirectMedia Layer є кросплатформними, тому вони не впливають на вибір цільової операційної системи.

Програма, що розробляється має бути написана на мові C++ у середовищі розробки Qt Creator. Програма призначена для роботи на ПК під керуванням ОС Windows, починаючи з версії Windows Vista.

Встановлено такі мінімальні вимоги, необхідні для коректного та швидкого відпрацювання програми:

- процесор з частотою 1 ГГц або SoC;
- ОЗУ 1 Гб для 32-розрядних або 2 Гб для 64-розрядних;
- жорсткий диск: 16 Гб для 32-бітної і 20 Гб для 64-бітної ОС;
- відеоадаптер з підтримкою DirectX 9;
- параметри дисплею 800 x 600;

Зазначені вимоги є офіційними вимогами Microsoft для встановлення версії Windows 10 [10] на ПК.

1.3.5 Вимоги до інформаційної та програмної сумісності

Розроблене програмне забезпечення сумісне з такими версіями Windows: Vista, 7, 8 та 10. На офіційному сайті фреймворку це пояснюється тим, що модуль Multimedia для Windows реалізований в двох плагінах: Microsoft DirectShow API та WMF (Windows Media Foundation).

API DirectShow був представлений в Windows 98, але поступово застарівав, починаючи з Windows XP. Платформа Media Foundation була представлена в Windows Vista як заміна DirectShow та інших мультимедійних API. Отже, плагін WMF, а значить й реалізація модуля Multimedia у Qt підтримується тільки для Windows Vista та пізніших версій операційної системи.

Окрім цього, мультимедійні API Qt базуються на мультимедійній структурі базової платформи [12]. Це означає, що підтримка різних кодеків або контейнерів може відрізнятися на різних машинах, залежно від того, що встановив кінцевий користувач, тому завчасно вказати формати, які можливо відтворити даною програмою не можна.

1.3.6 Вимоги до програмної документації

Програмна документація має включати: керівництво програміста, керівництво оператора, опис програми та тексти основних програмних модулів.

2 РОЗРОБКА СТРУКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз технічного завдання

Відповідно до зазначених у технічному завданні вимог до програмного забезпечення, система що розроблюється має надавати користувачеві можливість працювати з медіа файлами. Найвідоміші та найрозповсюдженіші мови програмування(C, C++, C#, Java, Python) надають вже готові для використання бібліотеки та фреймворки для маніпулювання, обробки, запису та трансляції медіа контенту. Незважаючи на це, реалізація окремих функцій програми, навіть з використанням додаткових бібліотек, нерідко викликає труднощі.

Найскладнішою частиною даного застосунку, що розроблюється є реалізація перетягування повзунка користувачем для зміни точки відтворення відео. Складність полягає в тому, що, відслідковуючи позицію медіа файлу, що відтворюється (приблизно кожну секунду), повзунок реагує на це зміною поточного значення. При цьому, змінюючи повзунок користувачем, необхідно викликати оновлення позиції медіа файлу, що відтворюється не створюючи цикл, адже, в свою чергу, повзунок знову відреагує на зміну позиції медіа файлу, і, як наслідок, встановить новий показник. Вирішити проблему дозволить така перевірка:

```
on_time_line_slider_valueChanged(int value)
{
    if(value != myplayer->player->position() / 1000)
    {
        myplayer->player->setPosition(value * 1000);
    }
}
```

Якщо при отриманні сигналу про зміну значення повзунка, його показник не буде співпадати з показником плеєру, це означає, що нове значення встановив саме користувач. Лише в такому випадку медіа плеєр змінить точку відтворення й таким чином виникнення циклу є неможливим.

Крім того, в технічному завданні було вказано, що програма повинна відтворювати визначений перелік форматів. Мультимедійні API Qt базуються на мультимедійній структурі базової платформи [12]. Це означає, що підтримка різних кодеків або контейнерів може відрізнятися на різних машинах, залежно від того, які саме встановлені на поточному комп'ютері.

Застосунок, що розроблюється має зберігати шляхи у заданому файлі. Створення бази даних в даному випадку є недоречним рішенням, адже кількість інформації, що необхідно зберегти, досить невелика. В такому разі найкраще підходять для рішення задачі файли з розширенням .XML, .CSV або .TXT. Звичайний текстовий файл є найкращим варіантом, адже функціонал для запису та зчитування найлегше реалізується саме для нього. Інші формати є досить специфічними, адже для формату .XML необхідно реалізувати спеціальну структуру, а .CSV, як правило, містить багато розділових знаків, що заважатимуть при пошуку файлів у системі. Єдиним недоліком зберігання інформації в файлі .TXT є незахищеність файлу з налаштуваннями від дій користувача.

Найголовнішим компонентом системи, що розроблюється є мультимедійний фреймворк, який використаний для розробки модуля Qt, а саме: Multimedia. До версії Qt5 модуль мав назву «Phonon», тож після оновлення середовища до нової версії, весь функціонал цього модуля було перенесено й перейменовано як «Multimedia» [12].

За увесь час існування цього модуля було представлено дві реалізації – Media Foundation та DirectShow. Різні ПК можуть працювати з різними фреймворками, проте, незалежно від реалізації, програмне забезпечення є сумісним з версіями цільової операційної системи Windows, починаючи з Vista.

Усі інші класи, що належать окремим, різним модулям, й використовуватимуться під час розробки не мають суттєвих відмін у використанні в різних версіях операційної системи.

2.2 Розробка функціональної структури програми

Визначення послідовності дій користувача має велике значення під час проектування графічного інтерфейсу програмного забезпечення. Завдяки такому підходу, під час проектування програми буде враховано де саме й як користувач зможе взаємодіяти з системою, який функціонал буде доступний, та що від цього зміниться.

Блок схема алгоритму взаємодії користувача з програмою наведена на рис. 2.1.

Програмне забезпечення має великий перелік доступних, працюючих функцій та опцій. Розглянемо основні кроки алгоритму:

- запускаємо медіа плеєр;
- залишаємо або змінюємо системні шляхи до тек, що містять медіа файли;
- відтворюємо перший медіа файл;
- змінюємо гучність відтворення;
- використовуємо будь-які з зазначених функцій - перемикаємо на попередню або наступну композицію;
- призупиняємо відтворення медіа-контенту;
- завершуємо роботу з програмою.

Для опису процесу взаємодії користувача з програмним забезпеченням, без прив'язки до часу, було розроблено діаграму прецедентів (рис. 2.2). Фактично, діаграма прецедентів, у даному випадку, це інший спосіб представлення алгоритму взаємодії користувача з системою.

З даної діаграми видно, що більшість функціоналу існує, проте, є необов'язковим у використанні. Користувач може регулювати гучність, швидкість відтворення медіа контенту, змінювати режими відтворення, переходити в повноекранний режим.

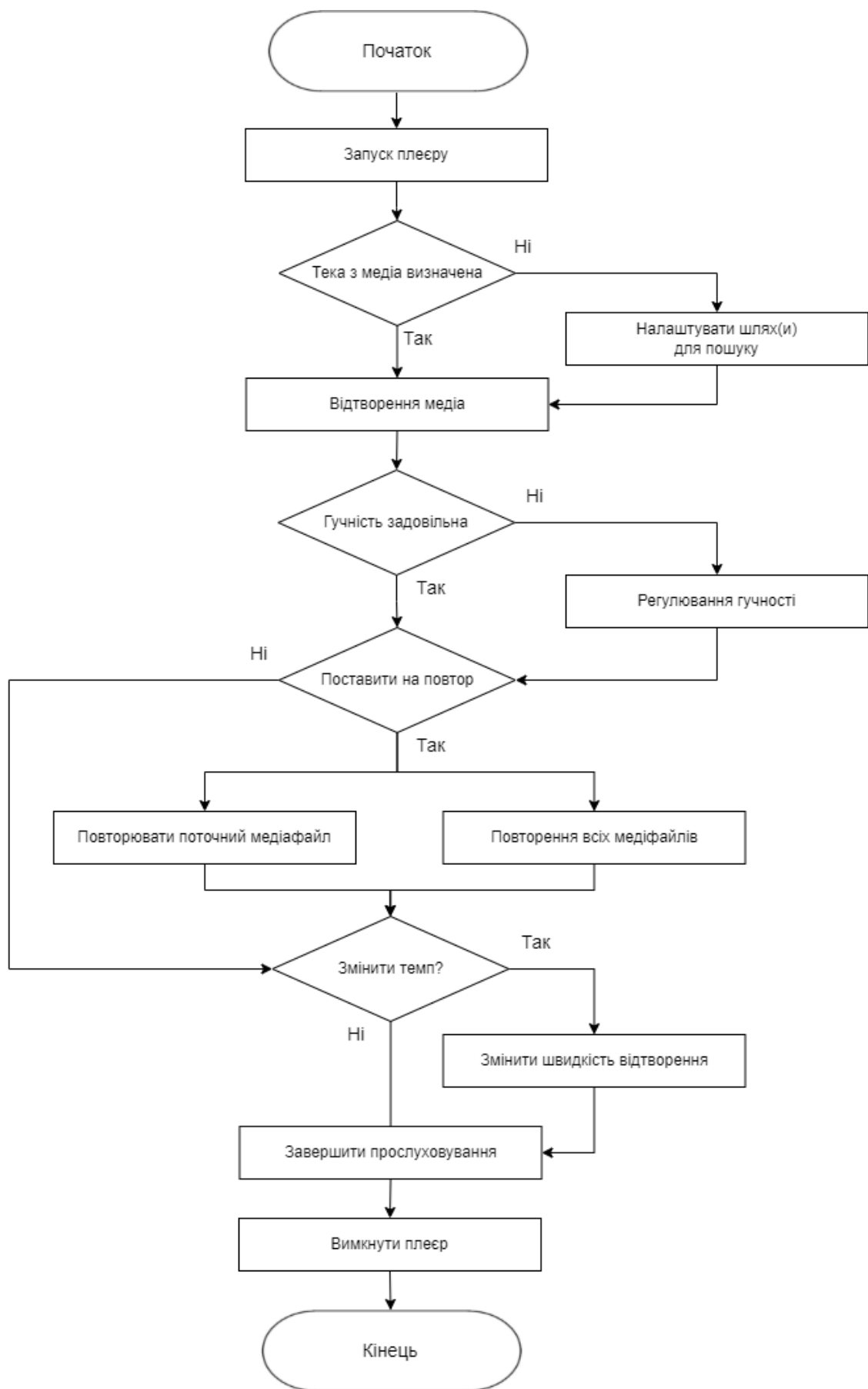


Рисунок 2.1 – Блок-схема алгоритму взаємодії користувача з програмою

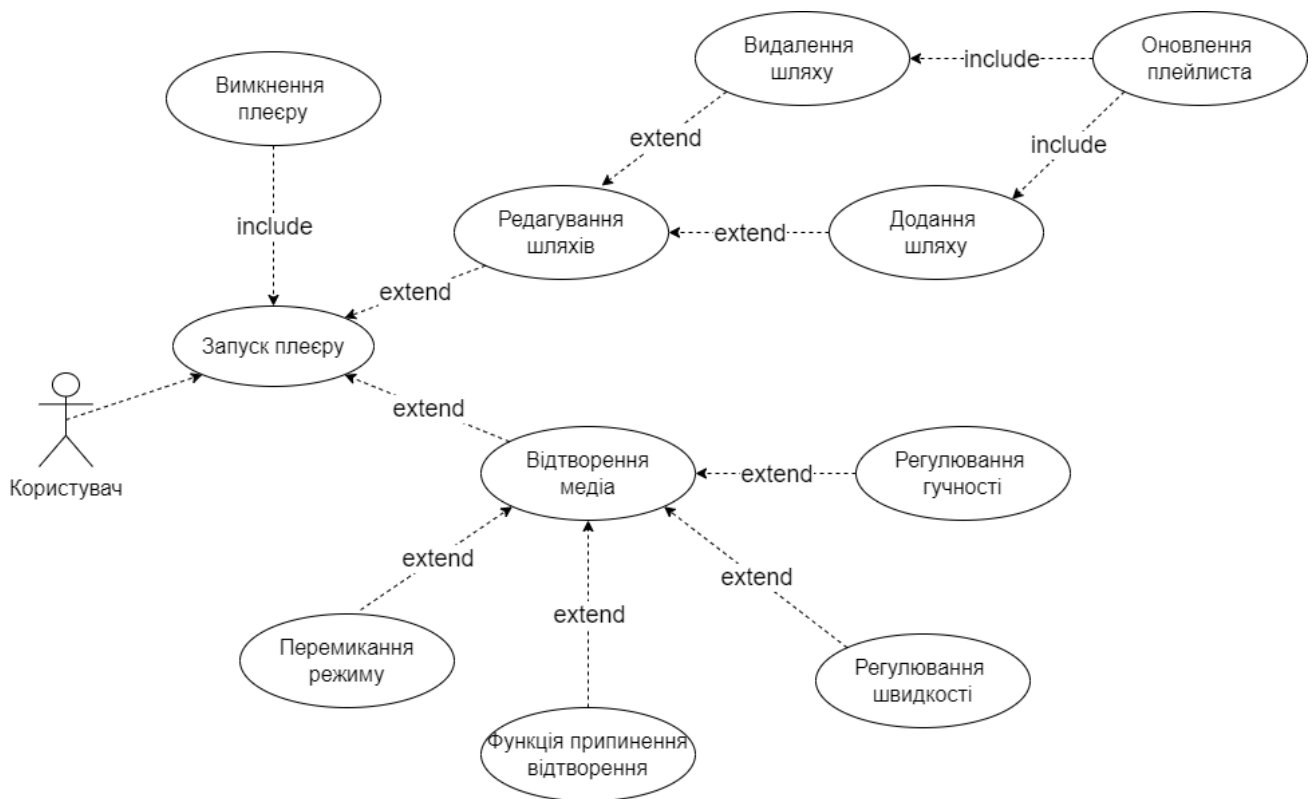


Рисунок 2.2 – Діаграм прецедентів

2.3 Обґрунтування вибору мови програмування

Серед всіх мов програмування, що можуть бути потенційними інструментами опрацювання медіа файлів, було обрано мову програмування C++. Обрати оптимальний інструмент для розробки доволі непроста задача. Тому для її вирішення, гарним рішенням буде вибір шляхом порівняння декількох мов, з урахуванням їх переваг, недоліків та особливостей.

Найпопулярнішими мовами для розробки Desktop програм є C++, C# та Python. Усі перераховані мови мають великі можливості та безліч готових бібліотек. Порівняння мов програмування наведено у табл. 2.1.

Враховуючи усі критерії оцінки при порівнянні, саме мова C++ має найбільшу кількість бібліотек для маніпулювання медіа файлами, зручну графічну оболонку Qt, програми розроблені з використанням цього фреймворку будуть кросплатформеними, хоча цільовою ОС є саме Windows. Саме обрана мова є найшвидшою у виконанні комп'ютером, надає найбільші можливості у

керуванні ресурсами комп'ютера шляхом контролю за пам'яттю, безпосередньо в кодї програми, що виділяється та звільнюється. Також, ця мова має сумісність з мовою C, тож, при необхідності, код, написаний мовою C, без проблем інтегрується в проєкт, налаштований під мову C++.

Таблиця 2.1 – Порівняння мов програмування

Критерії оцінки	Мова програмування		
	C++	C#	Python
Типізація	Статична, слабка	Статична, сильна	Динамічна, сильна
Швидкість роботи	Найшвидша з порівнюємих	Швидша за мову Python, повільніша за C++	Найповільніша з порівнюємих
Бібліотеки для роботи з медіа контентом	OpenAL PortAudio Audiere SDL Multimedia (Qt)	FFmpeg AForge.NET Стандартний клас MediaPlayer	Multimedia (Qt) GTK+ LibVLC GStreamer Python
Графічний інтерфейс	Qt WinAPI wxWidgets WinForms	WinForms WPF	Tkinter Kivy Pyforms PyQt
Кросплатформність	Підтримує (використовуючи фреймворк Qt)	Підтримує (використовуючи платформу .Net5)	Підтримує (використовуючи PyQt або Tkinter)

Недоліками обраної мови є складність у коректному використанні механізмів покажчиків, перевантажень та посилянь, в відстеженні помилок, а також складні мовні конструкції, що сповільнюють швидкість розробки та розуміння коду. Незважаючи на виявлені недоліки, обрана мова C++ є найкращим варіантом у розробці даного універсального медіа плеєру.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Розробка класів Desktop-застосунку

Обрана мова програмування C++ підтримує об'єктно-орієнтовану парадигму. Це означає, що реалізація програмного забезпечення буде виражена у вигляді взаємодії окремих класів предметної області.

Оскільки розробка програми проводиться в середовищі Qt Creator, деякі класи програми успадковуються від базових класів простору імен Qt. Дане програмне забезпечення будується на восьми класах:

- QWidget (клас для створення форми, від якого успадковує клас EditPath), призначений для відображення форми редагування шляхів пошуку медіа файлів;
- QMainWindow (клас для створення головного вікна, від якого успадковує клас MainWindow), головне вікно програми, що містить основний функціонал для маніпулювання медіа даними;
- QVideoWidget (клас, від якого успадковує клас VideoWidget);
- VideoWidget – створений з метою перевизначення функції подвійного натискання клавіші миші для виходу з повноекранного режиму;
- MyPlayer – клас, що містить об'єкти класів QMediaPlayer та QMediaPlaylist, в цьому класі реалізується весь функціонал даного медіаплеєру, в ньому також зберігається перелік форматів, що підтримує система;
- EditPath – клас, що відображає форму редагування шляхів для пошуку медіа файлів, дозволяє видаляти та додавати нові шляхи, при цьому результати дій відображаються в таблиці;
- PathFiles – службовий клас, в який розміщено функції для створення файлу, його зчитування та запису інформації.

Для визначених класів було розроблено діаграму класів, зображену на рис. 3.1.

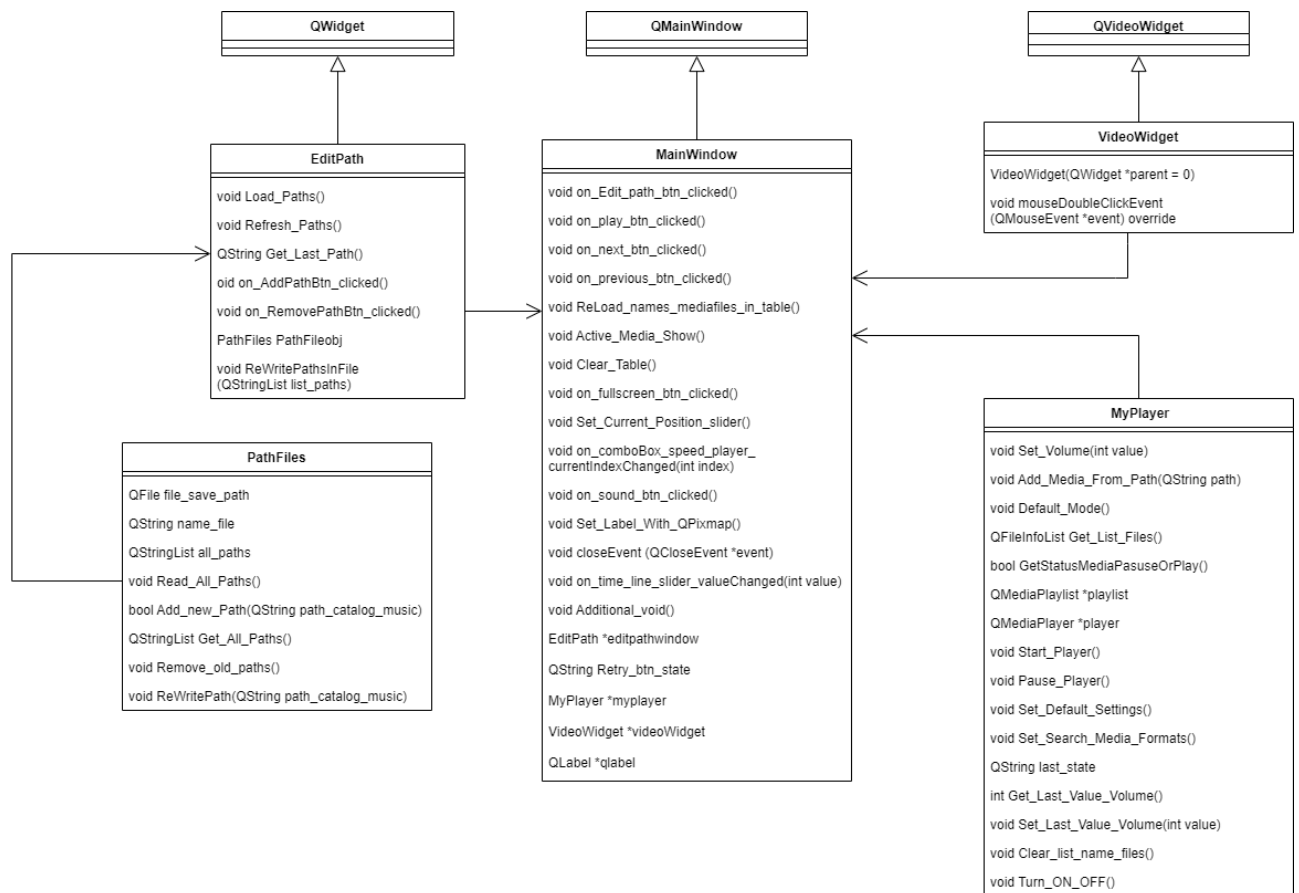


Рисунок 3.1 – Діаграма класів

Створена діаграма класів дозволяє чітко побачити та зрозуміти, як побудована програма. Головне вікно програми MainWindow успадковує клас QMainWindow. Це стандартний спосіб реалізації головного вікна в середовищі Qt Creator. Діаграма показує наявність відношень «асоціація», тобто один клас містить екземпляр іншого класу у якості поля. Цей зв'язок існує між формою MainWindow та класами EditPath, MyPlayer, VideoWidget, а також між класами EditPath та PathFiles.

Інша форма, що описана у якості класу EditPath, успадковує клас QWidget, тобто, клас для представлення «звичайного» вікна, допомагає редагувати шляхи пошуку медіа контенту. Клас VideoWidget успадковує клас QVideoWidget з метою перевизначення функції подвійного натискання клавіші миші. При розгортанні відео на весь екран, весь фокус переходить на об'єкт класу VideoWidget, тому перевизначена функція дозволяє відслідкувати подію «Подвійне натискання миші» й залишити цей режим.

Розглянемо декілька основних реалізацій функціоналу для класу MainWindow (головного вікна програми).

Найпершою створеною функцією є функція відтворення медіа контенту. Функція створена та працює виключно після натискання кнопки з іконкою «Відтворення», тобто реалізована як обробник подій «Натиснуто кнопку».

Реалізація цієї функції:

```
void MainWindow::on_play_btn_clicked()
{
    if(myplayer->playlist->isEmpty() == false)
    {
        QIcon *icon = new QIcon();
        if(myplayer->GetStatusMediaPasuseOrPlay())
        {
            icon->addPixmap(QPixmap(":/images/play.png"));
        }
        else
        {
            icon->addPixmap(QPixmap(":/images/stop.png"));
        }
        ui->play_btn->setIcon(*icon);
        myplayer->Turn_ON_OFF();
        delete icon;
    }
}
```

Функція перевіряє наявність медіа файлів в плейлисті, встановлює нове зображення play.png або stop.png й викликає функцію класу MyPlayer Turn_ON_OFF(), для вмикання або призупинення медіа файлу. Виділена пам'ять обов'язково звільнюється.

Наступною функцією є відображення поточного треку з плейлиста:

```
void MainWindow::Active_Media_Show()
```

```

{
    QModelIndex newIndex = ui->tableWidget->model()->
index(myplayer->Get_Current_Index_Playlist(), 0);
    ui->tableWidget->setCurrentIndex(newIndex);
}

```

Все, що необхідно для демонстрації активного медіа файлу, це:

- отримання поточного індексу плейлиста за допомогою значення, що повертається функції `Get_Current_Index_Playlist()`;
- встановлення «поточного» індексу таблиці з переліком треків.

Далі, завдяки використанню класу `QDir`, було розроблено функцію для зчитування усіх шляхів з файлу «`path.txt`» :

```

void PathFiles::Read_All_Paths()
{
    all_paths.clear();
    if (file_save_path.open(QIODevice::ReadOnly |
QIODevice::Text))
    {
        while (!file_save_path.atEnd())
        {
            QString line = file_save_path.readLine();
            all_paths.push_back(line);
        }
        file_save_path.close();
    }
}

```

Спершу програма починає очищення переліку існуючих шляхів функцією `clear()`, після цього, у разі успішного відкриття файлу починається «циклічне» зчитування усіх рядків та додання їх у перелік. Файл закривається й подальша робота буде відбуватися лише з цим переліком.

Розглянемо деякий функціонал класу, що описує властивості та методи медіа плеєру, а саме: MyPlayer. Усі формати, що підтримує система, автоматично завантажуються у визначений перелік, й застосовується для фільтрування файлів:

```
void MyPlayer::Set_Search_Media_Formats()
{
    list_audio_formats << V "*.mp2" << "*.mp3" << "*.mp4" << "*.m4a"
<< "*.mpeg";
    list_audio_formats << "*.asf" << "*.wma" << "*.au" << "*.wmv";
    list_audio_formats << "*.asx" << "*.wax" << "*.aiff" << "*.wmx";
    list_audio_formats << "*.flac" << "*.aac" << "*.ogg";
    list_audio_formats << "*.wmd" << "*.mpeg" << "*.wav";
    list_audio_formats << "*.mov" << "*.m4v" << "*.m2ts" << "*.cdda";
    list_audio_formats << "*.mid" << "*.midi" << "*.dvr-ms";
}
```

Дана функція викликається в конструкторі власного класу, а значить, вона буде гарантовано викликана й застосована, перш ніж завантажиться головна форма програми та користувач почне працювати з системою. Код конструктору класу з основними властивостями наступний:

```
MyPlayer::MyPlayer()
{
    player = new QMediaPlayer();
    playlist = new QMediaPlaylist();
    player->setPlaylist(playlist);
    Set_Default_Settings();
    Set_Search_Media_Formats();
    last_value = 0;
    last_state = "Sequential";
}
```

Конструктор даного класу ініціалізує плеєр, плейлист, завантажує плейлист в плеєр, встановлює функцією Set_Default_Settings() гучність в 100

відсотків, додає фільтри форматів, вказує змінній `last_value` значення 0, для запам'ятовування останнього значення гучності, якщо вона регулювалася користувачем. Також встановлюється режим «Sequential», тобто, початковий режим встановлюється як «Послідовне відтворення».

3.2 Розробка графічного інтерфейсу користувача

Застосунок що розроблюється повинен складатись з двох основних форм:

- головна форма, де користувач маніпулює медіа файлами через посередництво елементів керування;
- форма з додаванням та видаленням шляхів, які необхідно використовувати для пошуку музики та відео.

Перш за все, інтерфейс виконано у «адаптивному стилі». Елементи керування підлаштовуються під розміри вікна, якщо користувач забажає змінити їх. Відео, при цьому, зберігає пропорції, а інші елементи керування просто зберігають відносне розташування одне від одного.

Також, програмне забезпечення має бути інтуїтивно зрозумілим. Найкращим способом проектування інтерфейсу є використання досвіду взаємодії користувача з програмами-аналогами. Так, спираючись на аналіз існуючих аналогів (п. 1.2), можливо розмістити елементи керування згідно очікувань кінцевого користувача. Усі іконки для елемента «кнопка» мають схожий дизайн з головною формою застосунку «Windows Media Player» (рис. 1.1).

Для проектування форм за допомогою візуального редактору Qt Designer було застосовано основні кольори: сірий та блакитний. На головній формі розміщено віджет, в який додано елемент «QLabel» та «VideoWidget». Програма приховує віджет для відтворення відео, якщо відео є недоступним, або в цей час відтворюється музичний файл. При цьому, програма відображає

певний фоновий рисунок. Головне вікно програми при першому її запуску відображене на рис. 3.2.

Було додано таблицю (QTableWidget), що візуально відображає усі медіа файли поточного плей листа, а також автоматичне виділення активного треку в цій таблиці.

Розглянемо функціонал, який було реалізовано для маніпулювання плеєром та відображено у вигляді іконок.

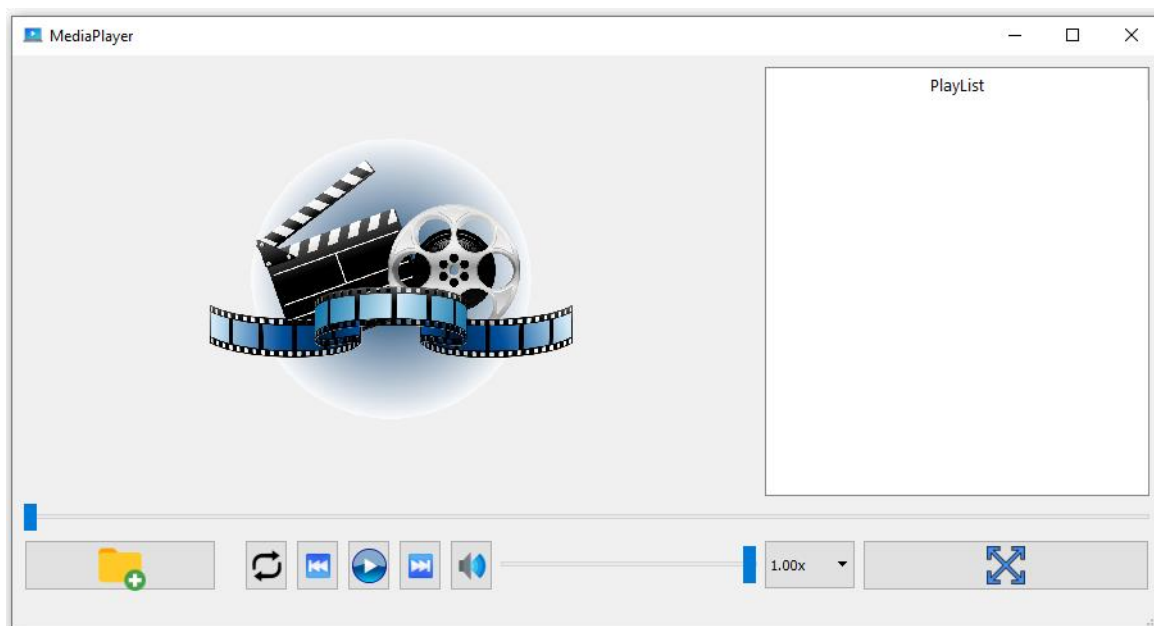


Рисунок 3.2 – Головне вікно медіа плеєру

З початку, плейлист медіа плеєру є порожнім, тому взаємодія з кнопками для маніпулювання медіа контентом є безрезультатною. Дана форма містить в собі два повзунки, один з них розташований біля елемента управління, іконка якого означає «Гучність», й інтуїтивно допомагає зрозуміти, що саме цей повзунок регулює гучність, а кнопка біля нього – вимикає звук повністю, або, відповідно, повертає його. Інший повзунок, що займає усю ширину вікна програми, відображає поточну позицію медіаплеєру під час відтворення медіафайлу. За допомогою нього можливо не тільки слідкувати за тривалістю медіа, а також встановити будь-яку позицію на ньому. Система

продовжить програвання з заданої користувачем позиції. Кожна умовна одиниця повзунка відповідає одній секунді реального часу.

Після повзунка гучності встановлені кнопки «1.00x» (для встановлення швидкості відтворення медіа файлу) й кнопка «Розгорнути на весь екран». Програма дозволяє згорнути повноекранний режим, як цією кнопкою, так і подвійним натисканням будь-якої клавіші миші.

У нижній частині вікна, по центру, зосереджені головні кнопки – їх іконки можна охарактеризувати так: «Відтворення» при неактивному стані та «Пауза» під час відтворення, кнопки «Вперед» та «Назад», що дозволяють рухатися по плейлисту у різних напрямках, а також кнопка «Повторити все», натискання на яку викликає зміну її іконки на «Повторити лише поточний файл», «Звичайний режим».

Зауважимо, що кнопки для відтворення медіа контенту, переходу до наступного або попереднього треку, перехід у повноекранний режим працюють, тільки, якщо плейлист містить хоча б один файл для відтворення.

Вигляд головного вікна при програванні відео наведено на рис. 3.3.

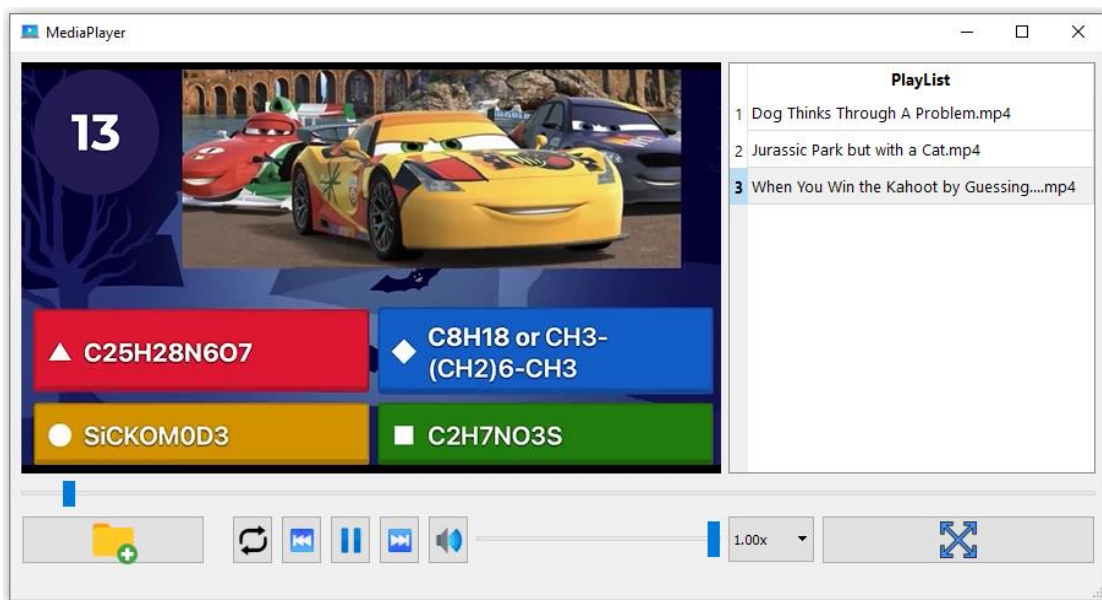


Рисунок 3.3 – Відтворення відео

Наступною була розроблена форма для редагування шляхів для пошуку. Дана форма має всього три елементи, а саме: дві кнопки «QPushButton» та одну таблицю «QTableWidget». Таблиця допускає лише виділення одного або декількох рядків для видалення вже існуючих шляхів в системі. Для додавання нового шляху, треба скористатися елементом з іконкою «Знайти». Вигляд форми зображено на рис. 3.4.

Як видно з рис. 3.4, інтерфейс даного вікна є інтуїтивно зрозумілим та досить лаконічним, елементи не містять зайвої, відволікаючої інформації, при цьому змінення розмірів форми не порушить естетичного вигляду. Як вже було визначено, для додавання нового шляху, треба скористатися елементом з іконкою «Знайти». Після цього, програма має розгорнути діалогове вікно «Вибір папки» (рис. 3.5). Система має слідкувати за доступністю вказаного каталогу, його існування й не дозволяти додавати порожній шлях до таблиці.

Інтерфейс побудовано таким чином, що, при закритті головного вікна програми, усі інші вікна також автоматично завершують свою існування.

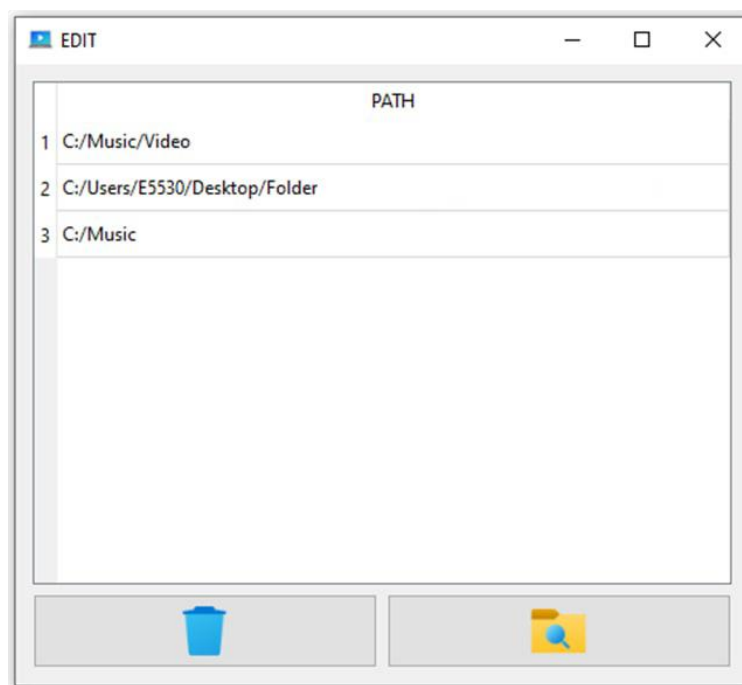


Рисунок 3.4 – Форма для редагування шляхів

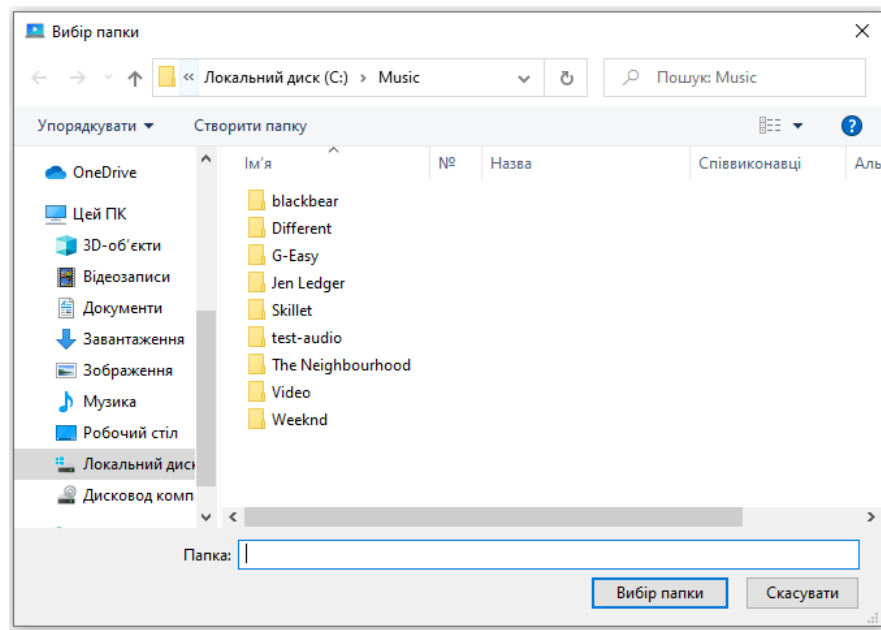


Рисунок 3.5 – Діалогове вікно «Вибір папки»

Отже, графічне оформлення програми виконано згідно очікувань користувача, кольорова гама складається переважно з двох кольорів, а значить, є досить привабливою.

Найскладнішою частиною реалізації інтерфейсу є створення функції перетягування повзунка та встановлення нового поточного значення медіа плеєру. Розглянемо наступний фрагмент коду:

```
connect(myplayer->player, &QMediaPlayer::positionChanged, this,
&MainWindow::Set_Current_Position_slider);
```

До об'єкту `myplayer`, що є одним з полів класу `MainWindow`, а саме його сигналу `positionChanged` (змінення позиції плеєру) підключаємо слот, що описує подальшу поведінку програми:

```
void MainWindow::Set_Current_Position_slider()
{
    if(ui->time_line_slider->isSliderDown() == false)
    {
```

```

        ui->time_line_slider->setValue
        (myplayer->player->position() / 1000);
    }
}

```

Даний фрагмент коду перевіряє, чи є повзунок натиснутим. У разі відсутності такої властивості, встановлюється нове значення для повзунка, що відображає поточну позицію плеєру. При цьому буде викликана така подія:

```

void MainWindow::on_time_line_slider_valueChanged(int value)
{
    if(value != myplayer->player->position() / 1000)
    {
        myplayer->player->setPosition(value * 1000);
    }
}

```

Ця подія відслідковується, а дії, які необхідно виконати, реалізовані у наведених вище функціях. Система порівнює поточне значення повзунка (завжди в секундах) та значення повзунка (в мс), поділених на 1000. Тобто, програма вирішує, змінене значення було програмно, чи користувачем, просто перевіряючи відповідність значень повзунка до значення позиції медіа плеєру.

Такі перевірки є обов'язковими, адже, вони повністю захищають систему від циклічного перемикавання позицій медіа плеєру та повзунка, що його представляє.

Наступною важливою частиною у реалізації інтерфейсу є змінення максимального значення повзунка, що допомагає орієнтуватися у тривалості поточного медіа файлу. Під час руху в плейлисті, програма щоразу змінює максимальний показник повзунка, що розглядається наступним чином:

```

connect(myplayer->player, &QMediaPlayer::durationChanged, this,
&MainWindow::Set_Max_Value_time_line_slider);

```

Підключаємо сигнал об'єкту myplayer, durationChanged до слоту (реалізованої функції вікна MainWindow) Set_Max_Value_time_line_slider() використовуючи такий код:

```
void MainWindow::Set_Max_Value_time_line_slider()
{
    if(myplayer->player->isVideoAvailable())
    {
        qlabel->hide();
        videoWidget->show();
    }
    else
    {
        videoWidget->hide();
        qlabel->show();
    }
    ui->time_line_slider->setMaximum(myplayer->player->duration()
/ 1000);
}
```

У даному коді, функцією isVideoAvailable()) перевіряємо, чи є доступним відео потік, у випадку, коли відео є доступним, приховуємо фонове зображення й показуємо відео. В іншому випадку, система повертає зображення на головну форму й приховує VideoWidget.

Розглянемо останній рядок:

```
ui->time_line_slider->setMaximum(myplayer->player->duration() / 1000);
```

Система отримує поточне значення загальної кількості мілісекунд, а потім, переводячи їх у секунди, встановлює новий максимум для повзунка.

Отже, розроблена система містить чіткий, зрозумілий код, розміщений у різних класах, добре іменовані змінні та обробники подій, що реалізують весь функціонал, визначений у функціональних вимогах технічного завдання.

Код розробленого застосунку наведено у додатку А.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення та умови застосування програми

Дане програмне забезпечення призначене для відтворення аудіо та відео файлів, формування плейлистів, збереження шляхів для пошуку медіа файлів. Для коректного функціонування програми необхідна версія ОС Windows Vista або вище. Файл з налаштуванням шляхів створюється та редагується лише програмою. Програма підтримує, вказаний в технічному завданні, перелік аудіо та відео форматів. Проте цей перелік може відрізнятись, у залежності від того, які аудіо медіа контейнери встановлені на машині користувача.

4.2 Характеристики програми

Програма допомагає користувачу у вирішенні його задач, використовуючи при цьому перевірки в різних місцях коду. Некоректні дії користувача програма розпізнає, й повідомляє, як виправити поточну ситуацію.

Для запуску програмного забезпечення необхідно виконати файл `Diplom_Project.exe`. Програма не потребує та не підтримує виклик з додатковими аргументами з командного рядка.

Програму розроблено за допомогою мови програмування високого рівня C++ в середовищі розробки Qt Creator з використанням модуля Multimedia для обробки мультимедійного контенту.

4.3 Вхідні та вихідні дані

До вхідних даних програми належать директорії, з яких медіа файли точно визначеними форматами програма може відтворювати. Фактично на вхід подається інформація з файлової системи комп'ютера, й інтерфейс програмного забезпечення для користувача.

Вихідними даними програми виступає щойно створений та заповнений переліком директорій текстовий файл у пам'яті комп'ютера. При цьому, якщо даний файл вже існує, програма знайде його та продовжить працювати з ним. До вихідних даних відноситься музика та відео, котрі відтворює програма.

4.4 Опис необхідних модулів та класів

Для забезпечення функціонування усіх функцій медіаплеєру використано наступні класи з модуля Multimedia середовища програмування Qt Creator:

- <QDir> клас, що забезпечує доступ до структур каталогів і їх вмісту, застосовано для пошуку й фільтрування усіх файлів, що можливо відтворити;
- QMessageBox клас, що дозволяє легко створювати та викликати прості діалогові вікна, призначені для інформування користувача щодо будь-якої інформації, запитів на дозволи, або попереджень;
- QtMultimedia модуль, що містить в собі окремі класи, які допомагають реалізувати функціонал плеєру. З цього модуля було взято класи «QMediaPlayer», «QMediaPlaylist», «QVideoWidget»;
- QMouseEvent клас, що містить параметри, які описують події миші. Він був використаний для перевизначення подвійного натискання миші на елемент «QWidget» для виходу з повноекранного режиму та входу в нього;
- QFile цей клас надає інтерфейс для зчитування та запису в файли, за допомогою нього реалізовано функції взаємодії з файлом, що містить шляхи.

Перелік розроблених класів програми було наведено у п. 3.1 та 3.2. Перелік атрибутів та методів кожного з розроблених класів наведено у табл. 4.1.

Таблиця 4.1 – Перелік атрибутів та методів класів

Клас	Атрибут/метод	Тип даних	Короткий опис
1	2	3	4
EditPath	Load_Paths()	void	Завантаження шляхів в колонки таблиці
	Refresh_Paths()	void	Оновлення шляхів в колонках таблиці
	Get_Last_Path()	QString	Отримання останнього доданого шляху
	last_file	QString	Зберігання останнього доданого шляху
	on_AddPathBtn_clicked()	void	Сигнал натискання кнопки AddPathBtn
	on_RemovePathBtn_clicked()	void	Сигнал натискання кнопки RemovePathBtn
	ReWritePathsInFile (QStringList list_paths)	void	Перезапис всіх шляхів
	SendSignalDelAndUpdataMainWindow()	void	Сигнал про видалення з переліку
	SendSignalAddAndUpdataMainWindow	void	Сигнал про додавання до переліку
	explicit EditPath (QWidget *parent = nullptr)	EditPath	Конструктор даного класу
EditPath	PathFiles PathFileobj	PathFiles	Екземпляр класу PathFiles
	~EditPath()	~EditPath	Деструктор даного класу
VideoWidget	(QWidget *parent = 0)	Video Widget	Конструктор класу

Продовження табл. 4.1

1	2	3	4
VideoWidget	mouseDoubleClickEvent (QMouseEvent *event) override	void	Обробник події подвійного натискання миші
Main Window	*editpathwindow;	EditPath	Екземпляр класу EditPath
	Retry_btn_state	QString	Поточний стан кнопки «повтор»
	*myplayer	MyPlayer	Екземпляр класу MyPlayer
	*videoWidget	Video Widget	Екземпляр класу VideoWidget
	*qlabel	QLabel	Надпис для вставлення зображення
	Change_Icon_Cycle_Btn	void	Перемикання іконки кнопки «Повтор»
	on_Edit_path_btn_clicked	void	Виклик меню «Edit Path»
	on_cycle_btn_clicked	void	Перемикання режиму відтворення
	on_play_btn_clicked	void	Відтворення файлу
Main Window	on_next_btn_clicked()	void	Перехід до наступної композиції
	on_previous_btn_clicked()	void	Перехід до попередньої композиції
	on_horizontalSlider_valueChan ged()	void	Обробник події повзунка
	ReLoad_names_mediafiles _in_table()	void	Перезапис всіх композицій в таблиці

Продовження табл.4.1

1	2	3	4
Main Window	Active_Media_Show()	void	Виділення активної композиції в таблиці
	Get_list_names_with_last_add_path()	void	Отримання списку композицій з останнього доданого шляху
	Clear_Table()	void	Очистити таблицю
	on_fullscreen_btn_clicked()	void	Перехід в повноекранний режим
	on_sound_btn_clicked()	void	Переключення на режимам «Без звуку»
	Set_in_column_pack_names_media_files()	void	Запис в таблицю переліку назв файлів
	Set_Label_With_QPixmap()	void	Встановлення зображення в надпис
	Init_VideoWidget()	void	Ініціалізація екземпляру класу videoWidget
Main Window	closeEvent (QCloseEvent *event)	void	Перевизначений метод закриття вікна програми
	on_time_line_slider_valueChanged(int value)	void	Обробник події «Змінено значення повзунка»
	Set_Max_Value_time_line_slider()	void	Встановлення кількості секунд в повзунку для поточної композиції

Продовження таблиці 4.1

1	2	3	4
Main Window	(QWidget *parent = nullptr)	Main Window	Конструктор даного класу
	~MainWindow()	~Main Window	Деструктор даного класу
PathFiles	file_save_path	QFile	Об'єкт для працювання з головним файлом
	name_file	QString	Назва файлу
	all_paths	QString List	Список зі зчитаними з файлу шляхами
	Read_All_Paths()	void	Зчитування всіх шляхів з файлу
	PathFiles()	PathFiles	Конструктор даного класу
	~PathFiles()	~Path Files	Деструктор даного класу
	Add_new_Path (QString path_catalog_music);	bool	Додання нового шляху
PathFiles	Get_All_Paths()	QString List	Зчитати й повернути перелік всіх шляхів
	Remove_old_paths()	void	Видалити всі застарілі директорії
	ReWritePath (QString path_catalog_music)	void	Додати новий шлях до файлу
MyPlayer	MyPlayer()	MyPlayer	Конструктор даного класу
	~ MyPlayer()	~ My Player	Деструктор даного класу

Продовження таблиці 4.1

MyPlayer	Add_Media_From_Path (QString path)	void	Додавання в плейлист усіх композицій з вказаного шляху
	Turn_ON_OFF()	void	Перемикання між режимами
	Set_Volume(int value)	void	Встановлення гучності
	Default_Mode()	void	Встановлення режиму за замовчуванням
	Retry_ALL_Mode()	void	Повторення всіх композицій
	Retry_Current_Mode()	void	Повторення поточної композиції
	Previous()	void	Перехід до попереднього треку
	Next()	void	Перехід до наступного треку
MyPlayer	Get_List_Files()	QFile InfoList	Отримати перелік композицій
	Get_Current_Index_Playlist()	int	Отримання індексу поточної композиції
	GetStatusMedia PauseOrPlay()	bool	Отримання статусу медіаплеєру
	Clear_list_name_files()	void	Очистити перелік композицій
	*playlist	QMedia Playlist	Екземпляр класу QMediaPlaylist
	*player	QMedia Player	Екземпляр класу QMediaPlayer

Продовження таблиці 4.1

MyPlayer	Get_Last_Value_Volume()	int	Дістати останню змінену гучність
	Set_Last_Value_Volume (int value)	void	Встановити останню змінену гучність
	Clear_Playlist()	void	Очистити плейлист
	list_audio_formats	QString List	Перелік доступних аудіо та відео форматів
	name_files	QFile InfoList	Перелік директорій для доступу до композицій
	last_value	int	Змінна, що вказує на останнє значення гучності
	last_state	QString	Змінна, що зберігає назву останнього обраного режиму
MyPlayer	Start_Player()	void	Увімкнення медіаплеєру
	Pause_Player()	void	Призупинення медіаплеєру
	Set_Search_Media_Formats()	void	Встановлення доступних форматів
	Set_Default_Settings()	void	Встановлення налаштувань
	Go_from_Loop()	void	Вихід з циклу для перемикавання композиції
	Go_to_Loop()	void	Повернення до циклу (режиму відтворення)

4.4 Повідомлення програмісту

Повідомлень програмісту у програмі не передбачено.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення та умови виконання програми

Програмний застосунок призначений для відтворення аудіо та відео файлів, формування плейлистів та збереження шляхів для пошуку медіафайлів.

Застосунок коректно виконується під керуванням ОС Windows Vista або вище. Файл з налаштуванням шляхів створюється, редагується лише програмою. Мінімальні апаратні вимоги до програми відповідають вимогам, що встановлені для нормальної роботи ОС Windows.

5.2 Виконання програми

Для запуску програмного забезпечення необхідно виконати файл `Diplom_Project.exe`. Програма не потребує та не підтримує виклик з додатковими аргументами з командного рядка.

Програмний продукт має дві форми, на яких розміщені елементи керування та одне діалогове вікно. Форма головного вікна, так само як і форма для редагування шляхів є адаптивною, тому користувач може легко змінювати ширину та висоту програми. Спосіб розташування елементів було запозичено в програми Windows Media Player, адже саме такий спосіб розміщення найчастіше зустрічається в програмах-аналогах.

При першому запуску програми, користувачу відкривається головне вікно медіа плеєру (рис. 3.2).

На початку роботи, плейлист медіа плеєру є порожнім, тому кнопки для маніпулювання медіа контентом є неактивними. Дана форма містить в собі два повзунка, один з них розташований біля елемента управління, іконка якого означає «Гучність». Кнопка біля нього – вимикає звук повністю, або, відповідно, повертає його. Інший повзунок, що займає усю ширину вікна програми, відображає поточну позицію медіа плеєру під час відтворення медіа

файлу. За допомогою нього можливо не тільки слідкувати за тривалістю медіа, можливо встановити будь-яку позицію на ньому.

Після повзунка гучності встановлені кнопки «1.00x» (для встановлення швидкості відтворення медіа файлу) та кнопка «Розгорнути на весь екран». Програма дозволяє згорнути повноекранний режим, як цією кнопкою, так і подвійним натисканням будь-якої клавіші миші.

У нижній частині вікна, по центру, зосереджені головні кнопки – їх іконки можна охарактеризувати так: «Відтворення» при неактивному стані та «Пауза» під час відтворення, кнопки «Вперед» та «Назад», що дозволяють рухатися по плейлисту у різних напрямках, а також кнопка «Повторити все», натискання на яку викликає зміну її іконки на «Повторити лише поточний файл», «Звичайний режим».

Також реалізовано перехід до вікна редагування шляхів. При цьому, якщо користувач закриває головне вікно, вікно з редагуванням автоматично припинить своє існування. При переході на наступну форму з іконкою «Додати» відбувається відкриття вікна «Edit» (рис. 3.4).

Дана форма має лише три елемента для керування системою. Перелік шляхів занотовується в елемент керування «TableWidget», а також динамічно оновлюється у файлі «path.txt». При цьому шляхи заборонено редагувати «вручну», тобто користувач може лише виділяти будь-яку кількість існуючих шляхів та видаляти їх, або додавати нові шляхом відкриття діалогового вікна «Вибір папки».

Елементи керування «QPushButton», що виконують роль «кнопки», не містять ніякого тексту, інформація про їх призначення отримується лише за допомогою візуального оформлення, а саме іконок.

При спробі «Додання шляху», розгортається нове діалогове вікно «Вибір папки» (рис. 3.5).

Після закриття діалогового вікна, якщо шлях не буде обрано, порожній рядок не створюється, не можна також додати шлях, якщо вже такий існує в

файлі. В іншому випадку, користувач додає новий шлях, з якого система автоматично знайде усі файли, що здатна розпізнати та відтворити (рис. 5.1).

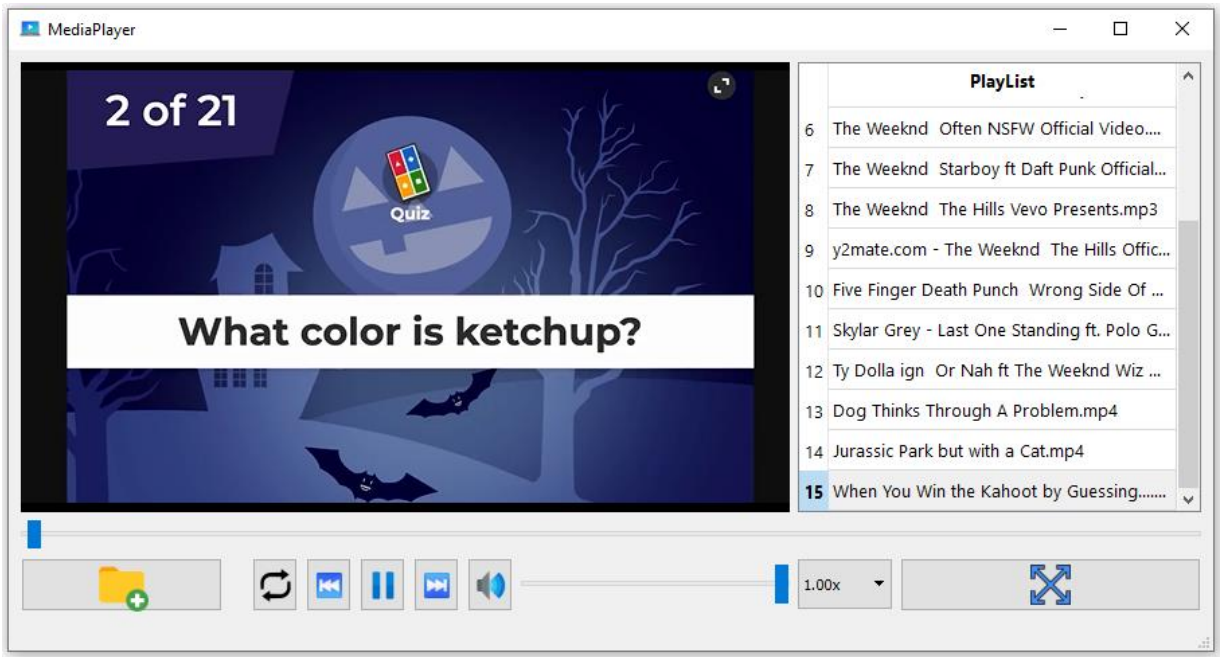


Рисунок 5.1 – Результат додавання нових шляхів

5.3 Повідомлення оператора

Було розроблено та налаштовано такі повідомлення, що може надсилати користувачу система (табл. 5.1).

Таблиця 5.1 – Повідомлення оператору

Текст	Причина появи	Подальші дії
Success add new PATH!	Успішно додано новий шлях для пошуку медіа.	
Please, input the correct PATH	Введено некоректний шлях (порожній, або існуючий).	Перевірити, чи не співпадають існуючий шлях з новим.

ВИСНОВКИ

У результаті виконання дипломної кваліфікаційної роботи бакалавра було розроблено готового до роботи медіа плеєра, призначеного для відтворення аудіо та відео файлів найбільш поширених форматів, формування плейлистів та збереження шляхів для пошуку медіафайлів.

Першим кроком до реалізації даного програмного забезпечення був детальний аналіз предметної області, а також огляд вже існуючих програмних аналогів. Визначено функціонал, що є присутнім у більшості аналогів та недоліки розглянутих безкоштовних медіа плеєрів. В результаті проходження цього етапу розробки було розроблено технічне завдання до проекту.

Аналіз технічного завдання показав, що головним компонентом системи, що розроблялась є мультимедійний фреймворк «Multimedia» середовища Qt. На етапі проектування системи, було обрано мову програмування C++, середовище розробки Qt та розроблена модель взаємодії користувача з програмою у вигляді блок-схеми алгоритму взаємодії користувача з програмою та діаграми прецедентів.

Наступним етапом стала безпосередня реалізація проекту. На цьому етапі були розроблені структура необхідних класів застосунку, діаграма класів та графічний інтерфейс користувача. Інтерфейс виконано у «адаптивному стилі», тобто елементи керування підлаштовуються під розміри вікна.

Документація щодо опису та виконання програми, що була написана в процесі розробки, а також інтуїтивно зрозумілий інтерфейс дозволяють легко ввести готовий програмний продукт в експлуатацію.

Розроблений програмний застосунок відповідає всім вимогам, поставленим в технічному завданні до проекту. Програма передбачає можливості свого подальшого розширення.

Галузь використання розробленої системи – інформаційні технології.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Sullivan, J. What is a Container Format? [Electronic resource]. – Access mode: <https://www.wise-geek.com/what-is-a-container-format.htm>.
2. Расширение файла 3GPP [Электронный ресурс] – Режим доступа: <https://www.file-extension.info/ru/format/3gpp/>.
3. Media container formats (file types) [Electronic resource]. – Access mode: <https://developer.mozilla.org/en-US/docs/Web/Media/Formats/Containers>.
4. MIME types [Electronic resource]. – Access mode: https://developer.mozilla.org/ru/docs/Web/HTTP/Basics_of_HTTP/MIME_types.
5. HTML5 Audio – состояние дел. Часть 1 [Электронный ресурс] – Режим доступа: <https://habr.com/ru/post/148368/>.
6. Windows Media Player [Электронный ресурс] – Режим доступа: https://uk.wikipedia.org/wiki/Windows_Media_Player.
7. Techopedia Explains Windows Media Player (WMP) [Electronic resource]. – Access mode: <https://www.techopedia.com/definition/979/windows-media-player-wmp>.
8. Media Player Classic Home Cinema [Электронный ресурс] – Режим доступа: https://ru.bmstu.wiki/Media_Player_Classic_Home_Cinema.
9. VLC (медиаплеер) [Электронный ресурс] – Режим доступа: [https://ru.wikipedia.org/wiki/VLC_\(медиаплеер\)](https://ru.wikipedia.org/wiki/VLC_(медиаплеер)).
10. Системные требования для ОС Windows 10 [Электронный ресурс] – Режим доступа: <https://support.microsoft.com/ru-ru/windows/системные-требования-для-windows-10-6d4e9a79-66bf-7950-467c-795cf0386715>.
11. Universal-Media-Player [Электронный ресурс] – Режим доступа: <https://github.com/Bogdan17-bit/Universal-Media-Player/blob/main/myplayer.crpoundation>.
12. Документація модуля Qt Multimedia [Электронный ресурс] – Режим доступа: <https://doc.qt.io/qt-5/qtmultimedia-index.html>.

ДОДАТОК А
Текст програми

A.1 Файл Diplom_Project.pro

```

QT      += core gui multimedia multimediawidgets

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

CONFIG += c++11

#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000

SOURCES += \
    editpath.cpp \
    main.cpp \
    mainwindow.cpp \
    myplayer.cpp \
    pathfiles.cpp \
    videowidget.cpp

HEADERS += \
    editpath.h \
    mainwindow.h \
    myplayer.h \
    pathfiles.h \
    videowidget.h

FORMS += \
    editpath.ui \
    mainwindow.ui

// Правила развертывания по умолчанию

qnx: target.path = /tmp/${TARGET}/bin
else: unix:!android: target.path = /opt/${TARGET}/bin
!isEmpty(target.path): INSTALLS += target

RESOURCES += \
    resource.qrc

DISTFILES += \
    ../../../../Desktop/-Pngtree-pause vector
icon_5432679.png

```

A.2 Файл editpath.h

```

#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QtMultimedia/QMediaPlayer>
#include <QtMultimedia/QMediaPlaylist>
#include <QtMultimediaWidgets/QVideoWidget>
#include "editpath.h"
#include "myplayer.h"
#include "QMouseEvent"
#include "videowidget.h"
#include "QLabel"

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
    EditPath *editpathwindow;
    QString Retry_btn_state;
    MyPlayer *myplayer;
    VideoWidget *videoWidget;
    QLabel *qlabel;
    void Change_Icon_Cycle_Btn();

private slots:
    void on_Edit_path_btn_clicked();

    void on_cycle_btn_clicked();

    void on_play_btn_clicked();

    void on_previous_btn_clicked();

    void on_horizontalSlider_valueChanged(int value);

```

```

void ReLoad_names_mediafiles_in_table();

void Active_Media_Show();

void on_next_btn_clicked();

void Get_list_names_with_last_add_path();

void Clear_Table();

void on_fullscreen_btn_clicked();

void Set_Max_Value_time_line_slider();

void Set_Current_Position_slider();

void on_comboBox_speed_player_currentIndexChanged(int
index);

void on_sound_btn_clicked();

void
Set_in_column_pack_names_media_files(QFileInfoList
list_names);

void Additional_void();

void Set_Label_With_QPixmap();

void Init_VideoWidget();

void on_time_line_slider_valueChanged(int value);

void closeEvent (QCloseEvent *event);

private:
    Ui::MainWindow *ui;
};
#endif // MAINWINDOW_H

```

A.3 Файл myplayer.h

```

#ifndef MYPLAYER_H
#define MYPLAYER_H

#include <QMediaPlayer>
#include <QMediaPlaylist>
#include "QFileInfoList"

class MyPlayer
{
public:
    MyPlayer();
    ~MyPlayer();

    void Add_Media_From_Path(QString path);
    void Turn_ON_OFF();
    void Next();
    void Previous();
    void Set_Volume(int value);
    void Retry_ALL_Mode();
    void Retry_Current_Mode();
    void Default_Mode();

    QFileInfoList Get_List_Files();
    int Get_Current_Index_Playlist();
    bool GetStatusMediaPasuseOrPlay();
    void Clear_list_name_files();
    QMediaPlaylist *playlist;
    QMediaPlayer *player;

    int Get_Last_Value_Volume();
    void Set_Last_Value_Volume(int value);
    bool Clear_Playlist();

private:
    QStringList list_audio_formats;
    QFileInfoList name_files;

    int last_value;
    void Start_Player();
    void Pause_Player();
    void Set_Search_Media_Formats();
    void Set_Default_Settings();

```

```

        void Go_from_Loop();
        void Go_to_Loop();

        QString last_state;
};

#endif // MYPLAYER_H

```

A.4 Файл pathfiles.h

```

#ifndef PATHFILES_H
#define PATHFILES_H
#include "QString"
#include "QFile"
#include "QDir"

class PathFiles
{
private:
    QFile file_save_path;
    QString name_file;
    QStringList all_paths;
    void Read_All_Paths();

public:
    PathFiles();
    ~PathFiles();
    bool Add_new_Path(QString path_catalog_music);
    QStringList Get_All_Paths();
    void Remove_old_paths();
    void ReWritePath(QString path_catalog_music);
};

#endif // PATHFILES_H

```

A.5 Файл main.cpp

```

#include "mainwindow.h"
#include "pathfiles.h"

#include <QApplication>

int main(int argc, char *argv[])
{

```

```

    QApplication a(argc, argv);

    MainWindow w;
    a.setWindowIcon(QIcon(":/images/laptop_icon.png"));
    w.show();
    return a.exec();
}

```

A.6 Файл myplayer.cpp

// Для реализации стандартных функций проигрывателя, использовался известный подход [11]

```

#include "myplayer.h"
#include "QLabel"
#include "QFile"
#include "QDir"
#include <QAudioDeviceInfo>

// Конструктор класса

MyPlayer::MyPlayer()
{
    player = new QMediaPlayer();
    playlist = new QMediaPlaylist();
    player->setPlaylist(playlist);
    Set_Default_Settings();
    Set_Search_Media_Formats();
    last_value = 0;
    last_state = "Sequential";
}

// Деструктор класса

MyPlayer::~MyPlayer()
{
    playlist->clear();
    player = nullptr;
    playlist = nullptr;
    delete player;
    delete playlist;
}

MainWindow::~MainWindow()

```

```

{
    delete ui;
}

// Очистка списка

bool MyPlayer::Clear_Playlist()
{
    return playlist->clear();
}

int MyPlayer::Get_Last_Value_Volume()
{
    return last_value;
}

// Управление громкостью

void MyPlayer::Set_Last_Value_Volume(int value)
{
    last_value = value;
}

// Получение индекса текущей композиции

int MyPlayer::Get_Current_Index_Playlist()
{
    return playlist->currentIndex();
}

// Повтор всех композиций

void MyPlayer::Retry_ALL_Mode()
{
    qDebug("Set retry all mode ");
    playlist->
>setPlaybackMode(QMediaPlaylist::PlaybackMode::Loop);
}

// Автоматическая загрузка поддерживаемых форматов в
требуемый перечень

void MyPlayer::Set_Search_Media_Formats()
{

```

```

        list_audio_formats << V "*.mp2" << "*.mp3" << "*.mp4"
<< "*.m4a" << "*.mpeg";
        list_audio_formats << "*.asf" << "*.wma" << "*.au" <<
        "*.wmv";
        list_audio_formats << "*.asx" << "*.wax" << "*.aiff"
<< "*.wmx";
        list_audio_formats << "*.flac" << "*.aac" << "*.ogg";
        list_audio_formats << "*.wmd" << "*.mpeg" << "*.wav";
        list_audio_formats << "*.mov" << "*.m4v" << "*.m2ts"
<< "*.cdda";
        list_audio_formats << "*.mid" << "*.midi" << "*.dvr-
ms";

    }

void MainWindow::Clear_Table()
{
    ui->tableWidget->setRowCount(0);
}

// Установка режима по умолчанию

void MyPlayer::Default_Mode()
{
    qDebug("Set default mode ");
    playlist-
>setPlaybackMode(QMediaPlaylist::PlaybackMode::Sequential);
}

// Повтор текущей композиции

void MyPlayer::Retry_Current_Mode()
{
    qDebug("Set retry current mode ");
    playlist-
>setPlaybackMode(QMediaPlaylist::PlaybackMode::CurrentItemInLo
op);
}

// Установление громкости

void MyPlayer::Set_Volume(int value)
{
    player->setVolume(value);
}

```

```

    }

    // Возвращение к режиму воспроизведения

    void MyPlayer::Go_to_Loop()
    {
        if(last_state == "Sequential")
        {
            playlist->
>setPlaybackMode(QMediaPlaylist::PlaybackMode::Sequential);
        }

        else if(last_state == "CurrentItemInLoop")
        {
            playlist->
>setPlaybackMode(QMediaPlaylist::PlaybackMode::CurrentItemInLo
op);
        }

        else
        {
            playlist->
>setPlaybackMode(QMediaPlaylist::PlaybackMode::Loop);
        }
    }

    // Установление настроек

    void MyPlayer::Set_Default_Settings()
    {
        player->setVolume(100);
        playlist->
>setPlaybackMode(QMediaPlaylist::PlaybackMode::Sequential);
    }

    // Добавление в плейлист всех композиций по указанному
пути

    void MyPlayer::Add_Media_From_Path(QString path)
    {
        QDir dir(path);
        QFileInfoList new_pack_media_files =
dir.entryInfoList(list_audio_formats, QDir::Files |
QDir::NoDotAndDotDot);
    }

```

```

        name_files += new_pack_media_files;
        foreach (QFileInfo media_file, new_pack_media_files)
        {
            playlist-
>addMedia(QUrl(media_file.absoluteFilePath())));
        }
    }

```

```

bool MyPlayer::GetStatusMediaPasuseOrPlay()
{
    if(player->state() ==
QMediaPlayer::State::PlayingState)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}

```

// Переключение между режимами

```

void MyPlayer::Turn_ON_OFF()
{
    qDebug(QString::number(playlist-
>currentIndex()).toUtf8());
    if(player->state() ==
QMediaPlayer::State::PlayingState)
    {
        Pause_Player();
    }
    else
    {
        Start_Player();
    }
}

```

// Выход из цикла для переключения композиции

```

void MyPlayer::Go_from_Loop()
{

```

```

        if(playlist->playbackMode() ==
QMediaPlayer::PlaybackMode::CurrentItemInLoop)

        {
            last_state = "CurrentItemInLoop";
        }

        else if(playlist->playbackMode() ==
QMediaPlayer::PlaybackMode::Sequential)
        {
            last_state = "Sequential";
        }

        else
        {
            last_state = "Loop";
        }
        playlist-
>setPlaybackMode(QMediaPlayer::PlaybackMode::Loop);
    }

    // Переход к следующему треку

    void MyPlayer::Next()
    {
        qDebug(QString::number(playlist-
>currentIndex()).toUtf8());
        Go_from_Loop();
        playlist->setCurrentIndex(playlist->nextIndex());

        if(playlist->currentIndex() == -1)
        {
            playlist->setCurrentIndex(playlist->nextIndex());
            player->play();
        }
        Go_to_Loop();
    }

    // Повтор предыдущего трека

    void MyPlayer::Previous()
    {
        if(playlist->currentIndex() == -1)
        {

```

```

        playlist->setCurrentIndex(0);
    }
    Go_from_Loop();
    playlist->setCurrentIndex(playlist->previousIndex());
    Go_to_Loop();
}

```

// Приостанов

```

void MyPlayer::Pause_Player()
{
    player->pause();
}

```

```

void MyPlayer::Clear_list_name_files()
{
    name_files.clear();
}

```

// Старт

```

void MyPlayer::Start_Player()
{
    player->play();
}

```

// Получить список композиций

```

QFileInfoList MyPlayer::Get_List_Files()
{
    return name_files;
}

```

ДОДАТОК Б
Слайди презентації

ДИПЛОМНА РОБОТА

РОЗРОБКА ПРОГРАМИ МЕДІАПЛЕЄРУ

Розробив: ст. групи КНТ-228
Керівник: доцент, к.т.н.

Наumenko A.C.
Сердюк С.М.

Рисунок Б.1 – Слайд №1

Мета роботи

розробка програми, що дозволяє відтворення
медіа-контенту

Об'єкт дослідження — процеси
відтворення медіафайлів

Предмет дослідження — методи
відтворення медіафайлів



Задачі, що вирішуються

- провести аналіз предметної області;
- провести огляд програмних аналогів;
- розробка технічного завдання;
- розробка структури програмного забезпечення;
- розробка моделей програмного забезпечення;
- програмна реалізація застосунку;
- розробка необхідної документації.



НУ "ЗП", Кафедра програмних засобів

3

Рисунок Б.3 – Слайд №3

Найбільш поширені типи медіа контейнерів

Скорочене ім'я кодека	Повне ім'я кодека	Сумісність із браузером
3GP	Third Generation Partnership	Firefox для Android
ADTS	Audio Data Transport Stream	Firefox Доступний лише в середовищі носія базової операційної системи.
FLAC	Free Lossless Audio Codec	Chrome 56, Edge 16, Firefox 51, Safari 11
MPEG / MPEG-2	Moving Picture Experts Group (1 and 2)	—
MPEG-4 (MP4)	Moving Picture Experts Group 4	Chrome 3, Edge 12, Firefox, Internet Explorer 9, Opera 24, Safari 3.1
Ogg	Ogg	Chrome 3, Firefox 3.5, Edge 17 (тільки desktop), Internet Explorer 9, Opera 10.50
QuickTime (MOV)	Apple QuickTime movie	Тільки старіші версії Safari, а також інші браузери, що підтримують модуль Apple QuickTime.
WebM	Web Media	Chrome 6, Edge 17 (тільки desktop), Firefox 4, Opera 10.6, Safari 14.1 (macOS), Safari 15 (iOS). Edge вимагає інсталяції Web Media Extensions.



НУ "ЗП", Кафедра програмних засобів

4

Рисунок Б.4 – Слайд №4

Рекомендації щодо вибору оптимального медіа контейнера

Потреба користувача	Рекомендація
Тільки аудіо файли	
Відтворення стиснутих файлів загального призначення	MP3 (MPEG-1 Audio Layer III)
Стиснуті файли без втрат	FLAC з ALAC
Нестиснені файли	WAV
Відео файли	
Відео загального призначення, бажано у відкритому форматі	WebM з MP4
Відео загального призначення	MP4 з WebM
Висока ступінь стиснення, оптимізована для повільних з'єднань	3GP з MP4
Сумісність зі старими пристроями/браузерами	QuickTime з AVI і/або MPEG-2



НУ "ЗП", Кафедра програмних засобів

5

Рисунок Б.5 – Слайд №5

Аналіз медіа плеєрів

Назва	VLC Media Player	Media Player Classic HC	Windows Media Player
Вартість	Безкоштовно	Безкоштовно	Безкоштовно
Потокове відтворення	Так	Так	Так
Підтримка відео	Так	Так	Так
Менеджер завантажень	Ні	Ні	Ні
Віддалене керування	Так	Так	Так
Візуалізація	Так	Ні	Так
Безкоштовність	Так	Так	Так
Сумісність з Windows	Так	Так	Так



НУ "ЗП", Кафедра програмних засобів

6

Рисунок Б.6 – Слайд №6

Аналіз мов програмування

Критерії оцінки	Мова програмування		
	C++	C#	Python
Типізація	Статична, слабка	Статична, сильна	Динамічна, сильна
Швидкість роботи	Найшвидша з порівнюємих	Швидша за мову Python, повільніша за C++	Найповільніша з порівнюємих
Бібліотеки для роботи з медіа контентом	OpenAL PortAudio Audiere SDL Multimedia (Qt)	FFmpeg AForge.NET Стандартний клас MediaPlayer	Multimedia (Qt) GTK+ LibVLC GStreamer Python
Графічний інтерфейс	Qt WinAPI wxWidgets WinForms	WinForms WPF	Tkinter Kivy Pyforms PyQt
Кросплатформність	Підтримує (використовуючи фреймворк Qt)	Підтримує (використовуючи платформу .Net5)	Підтримує (використовуючи PyQt або Tkinter)

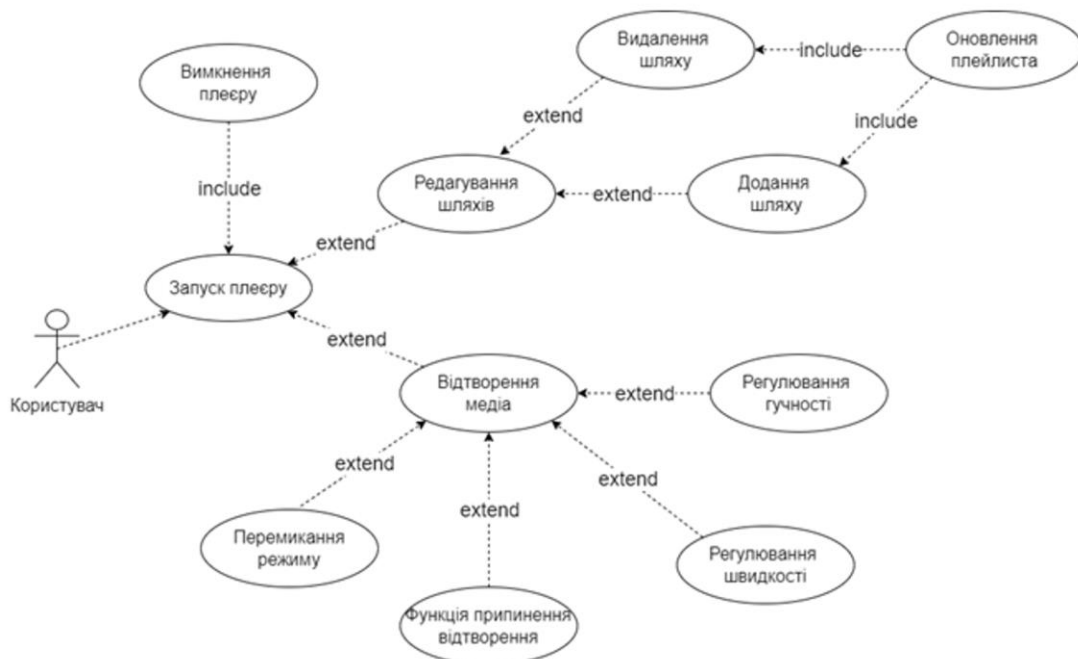


НУ "ЗП", Кафедра програмних засобів

7

Рисунок Б.7 – Слайд №7

Діаграма прецедентів

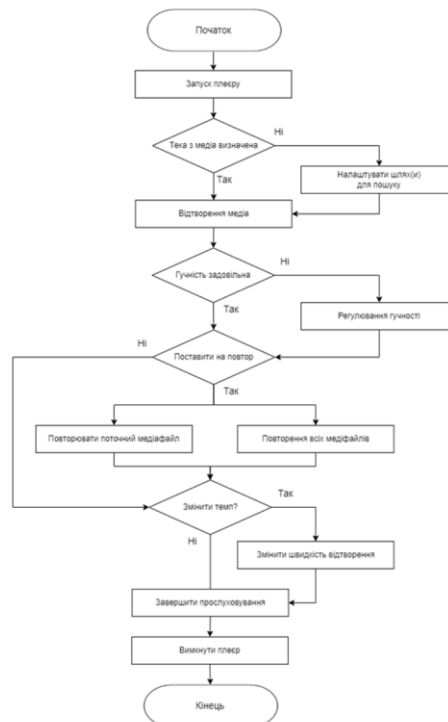


НУ "ЗП", Кафедра програмних засобів

8

Рисунок Б.8 – Слайд №8

Алгоритм взаємодії користувача з програмою

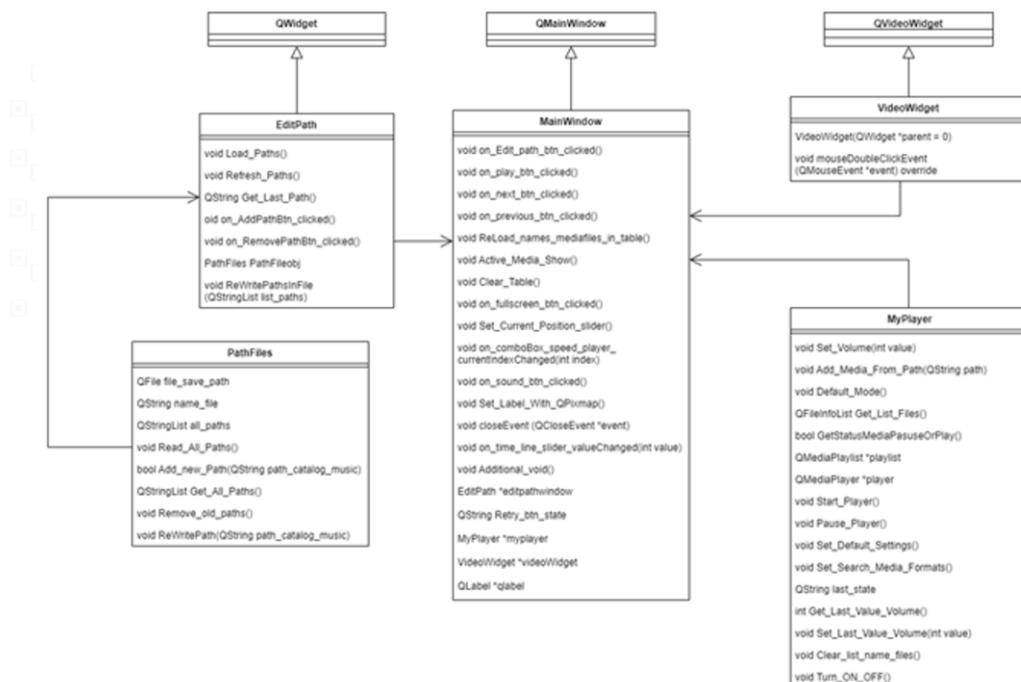


НУ "ЗП", Кафедра програмних засобів

Рисунок Б.9 – Слайд №9

9

Діаграма класів



НУ "ЗП", Кафедра програмних засобів

Рисунок Б.10 – Слайд №10

10

Приклад атрибутів та методів класу MyPlayer

Клас	Атрибут/метод	Тип даних	Короткий опис
MyPlayer	Add_Media_From_Path (QString path)	void	Додавання в плейлист усіх композицій з вказаного шляху
	Turn_ON_OFF()	void	Перемикання між режимами
	Set_Volume(int value)	void	Встановлення гучності
	Default_Mode()	void	Встановлення режиму за замовчуванням
	Retry_ALL_Mode()	void	Повторення всіх композицій
	Retry_Current_Mode()	void	Повторення поточної композиції
	Previous()	void	Перехід до попереднього треку
	Next()	void	Перехід до наступного треку



НУ "ЗП", Кафедра програмних засобів

11

Рисунок Б.11 – Слайд №11

Приклад функції відтворення медіа контенту

```
void MainWindow::on_play_btn_clicked()
{
    if(myplayer->playlist->isEmpty() == false)
    {
        QIcon *icon = new QIcon();
        if(myplayer->GetStatusMediaPasuseOrPlay())
        {
            icon->addPixmap(QPixmap(":/images/play.png"));
        }
        else
        {
            icon->addPixmap(QPixmap(":/images/stop.png"));
        }
        ui->play_btn->setIcon(*icon);
        myplayer->Turn_ON_OFF();
        delete icon;
    }
}
```

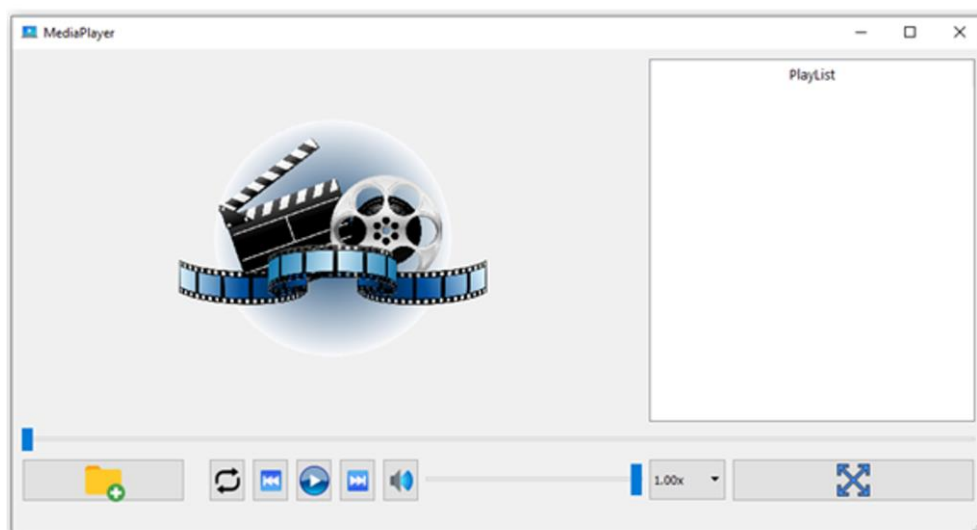


НУ "ЗП", Кафедра програмних засобів

12

Рисунок Б.12 – Слайд №12

Головне вікно медіаплеєру

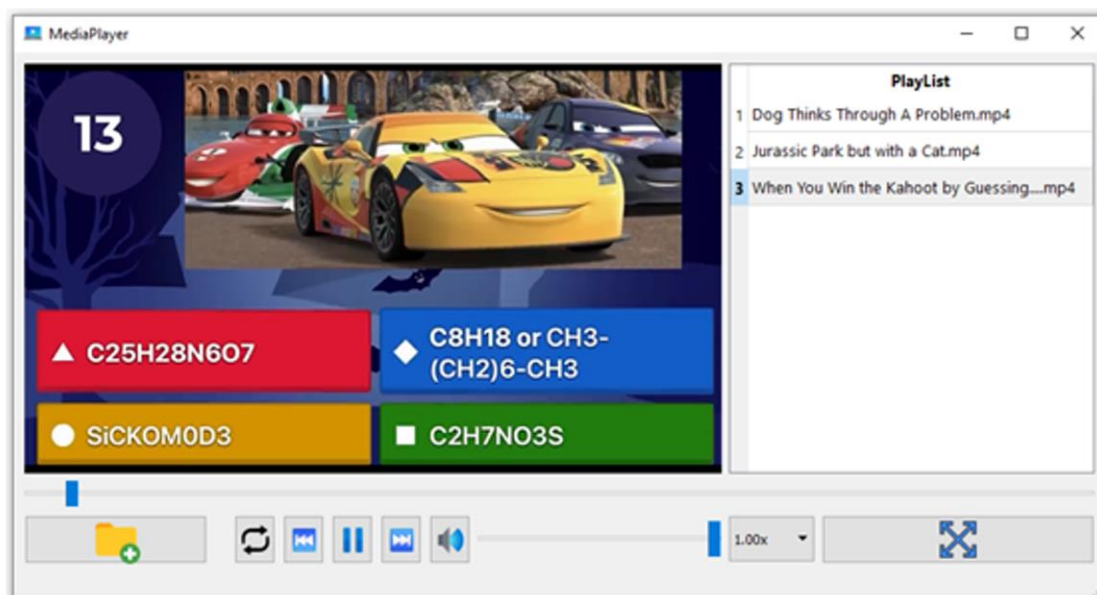


НУ "ЗП", Кафедра програмних засобів

Рисунок Б.13 – Слайд №13

13

Приклад відтворення відео

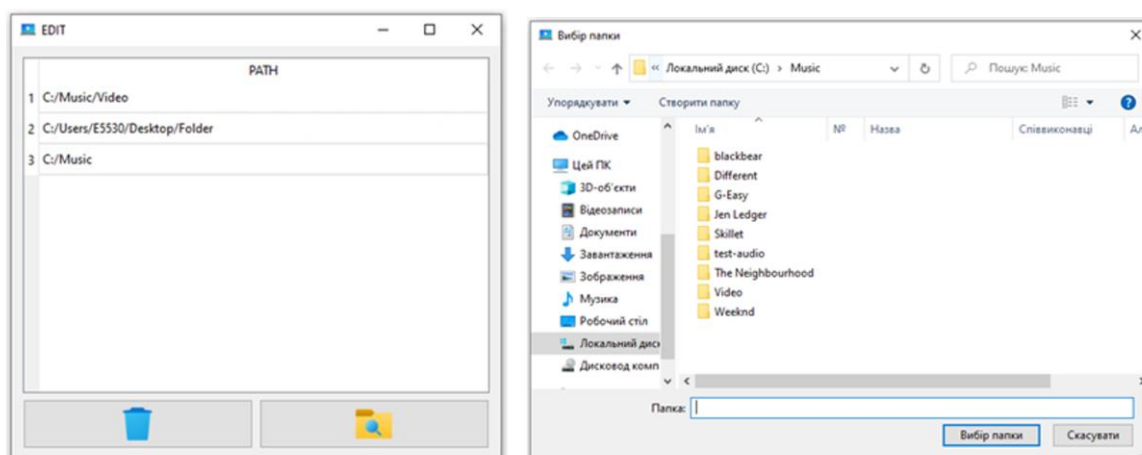


НУ "ЗП", Кафедра програмних засобів

Рисунок Б.14 – Слайд №14

14

Інші форми застосунку



Форма для редагування
шляхів

Діалогове вікно «Вибір папки»



НУ "ЗП", Кафедра програмних засобів

15

Рисунок Б.15 – Слайд №15

Висновки

- проведено аналіз предметної області;
- проведено огляд програмних аналогів;
- розроблено технічне завдання;
- розроблено структуру програмного забезпечення;
- розроблено моделі програмного забезпечення;
- програмно реалізовано застосунок;
- розроблено необхідну документацію.



НУ "ЗП", Кафедра програмних засобів

16

Рисунок Б.16 – Слайд №16

**Доклад закінчено,
дякую за увагу!**



НУ "ЗП", Кафедра програмних засобів

Рисунок Б.17 – Слайд №17