

Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

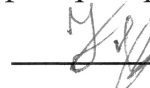
ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра
на тему: «Проект системи відтворення аудіоконтенту на
мові програмування Python»

Виконав студент 4-го курсу групи КІ-19-1
спеціальності 123 «Комп'ютерна інженерія»

 Андрій Сидорчук

Керівник професор кафедри КІ та КБ, д.т.н., доцент

 Ігор Фальковський

Рецензент старший викладач кафедри КН

 Руслан Петросян

Житомир – 2023

Державний університет «Житомирська політехніка»

Факультет інформаційно-комп'ютерних технологій

Кафедра комп'ютерної інженерії та кібербезпеки

Спеціальність 123 «Комп'ютерна інженерія»

Освітня програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач
кафедри комп'ютерної
інженерії та кібербезпеки

 Андрій Єфіменко

1 березня 2023 р.

ЗАВДАННЯ

на випускню кваліфікаційну роботу бакалавра

Студент: Сидорчук Андрій Васильович

Тема роботи: Проект системи відтворення аудіоконтенту на мові
програмування Python

затверджена Наказом університету від 1 березня 2023 р. № 98/с

Термін здачі студентом закінченої роботи 27 червня 2023 р.

Вихідні дані роботи (зазначається предмет і об'єкт дослідження):

вихідні дані для роботи;

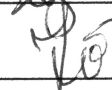
Обладнання та програмне забезпечення:

1. OC Windows 10
2. JetBrains PyCharm
3. Visual Studio Code
4. Qt Designer

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Головня О.С.	01.03.2023	01.03.2023
2	Головня О.С.	01.03.2023	01.03.2023
3	Головня О.С.	01.03.2023	01.03.2023

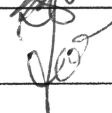
Студент  Андрій Сидорчук

Керівник  Ігор Фальковський

Календарний план

№ з/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка завдання	01.03.2023 – 01.03.2023	Виконано
2	Аналіз літературних джерел	02.03.2023 – 08.03.2023	Виконано
3	Аналіз об'єкта досліджень	09.03.2023 – 14.03.2023	Виконано
4	Аналіз програмного забезпечення для роботи	15.03.2023 – 23.03.2023	Виконано
5	Вибір технології для реалізації	24.03.2023 – 28.03.2023	Виконано
6	Розробка структури програми	29.03.2023 – 10.04.2023	Виконано
8	Написання програмного забезпечення	11.04.2023 – 14.04.2023	Виконано
9	Пошук проблем	15.04.2023 – 20.04.2023	Виконано
10	Рефакторинг коду	21.04.2023 – 09.05.2023	Виконано
11	Тестування програмного забезпечення	10.05.2023 – 13.05.2023	Виконано
12	Написання пояснювальної записки	14.05.2023 – 22.06.2023	Виконано
13	Захист	27.06.2023	Виконано

Студент  Андрій Сидорчук

Керівник  Ігор Фальковський

АНОТАЦІЯ

Випускна кваліфікаційна робота бакалавра складається з пояснювальної записки до дипломного проекту, яка складається зі вступу, трьох розділів, висновків, списку використаної літератури та додатків. Текстова частина викладена на 56 сторінках друкованого тексту. Пояснювальна записка має 22 сторінок додатків. Список використаних джерел містить 13 найменування і займає 1 сторінку. В роботі наведено 26 рисунків. Загальний обсяг роботи – 80 сторінок.

Метою кваліфікаційної роботи є розробка проекту системи відтворення аудіоконтенту на мові програмування Python.

В роботі було розглянуто призначення і принцип роботи з аудіоконтентом. Проаналізовано використання різних типів аудіоформату та існуючих рішень. Досліджено застосування популярних існуючих рішень. Розроблено концепт програми, визначено необхідне програмне забезпечення та виконано його налаштування. Результат роботи протестований та відповідає поставленим вимогам.

Ключові слова: PYTHON, AIMP, WINAMP, ООП, МЕТОДИ, ФУНКЦІЇ, ФАЙЛ, ПЛЕЙЛИСТ, ФОРМАТ, MP3, FLAC, WAV, OGG.

КІК.КР.Б – 123 – 16 – ПЗ

					КІК.КР.Б – 123 – 16 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Проект системи відтворення аудіоконтенту на мові програмування Python	Літ.	Арк.	Аркушів
Розроб.		Сидорчук А.В.		20.06.23				
Керівник		Фальковський І.Г.		20.06.23			4	80
Реценз.		Петросян Р.В.		20.06.23		ФІКТ, Гр. КІ-19-1		
Н. контр.		Єфіменко А.А.		20.06.23				
Затверд.		Єфіменко А.А.		20.06.23				

ANNOTATION

The final qualifying work of the bachelor consists of an explanatory note to the course project, which consists of an introduction, three chapters, conclusions, a list of references and appendices. The text part is set out on 56 pages of printed text. The explanatory note has 22 pages of appendices. The list of used sources contains 13 names and occupies 1 page. The work contains 26 figures. The total volume of work is 80 pages.

The purpose of qualification work is the development of a project of an audio content playback system in the Python programming language.

The work included the purpose and principle of working with audio content. The use of various types of audio format and existing solutions is analyzed. The application of popular existing solutions has been studied. The concept of the program was developed, the necessary software was determined, and its settings were made. The result of the work has been tested and meets the set requirements.

Keywords: PYTHON, AIMP, WINAMP, OOP, METHODS, FUNCTIONS, FILE, PLAYLIST, FORMAT, MP3, FLAC, WAV, OGG.

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		5

ЗМІСТ

ВСТУП.....	8
РОЗДІЛ 1 ТЕОРЕТИЧНІ АСПЕКТИ АУДІОФАЙЛІВ І РОБОТА З НИМИ.....	10
1.1 Теоретичні аспекти аудіофайлів.....	10
1.2 Огляд існуючих рішень	13
1.3 Огляд програмного забезпечення для розробки	18
РОЗДІЛ 2 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	25
2.1. Огляд модулів, які будуть використовуватися при розробці.....	25
2.2. Написання програмного забезпечення.....	34
РОЗДІЛ 3 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	46
3.1. Огляд основних можливостей	46
3.2 Взаємодія з сервісом YouTube	50
ВИСНОВКИ.....	56
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	57
ДОДАТКИ.....	58

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПК – персональний комп'ютер

MP3 – MPEG-1 Audio Layer 3

AAC – Advanced Audio Coding

WMA – Windows Media Audio

OGG – Ogg Vorbis

ПЗ – програмне забезпечення

ОС – операційна система

FLAC – Free Lossless Audio Codec

ALAC – Apple Lossless Encoder

CSS – Cascading Style Sheets

OSS – Operation Support System

GUI – Graphical user interface

QML – Qt Meta Language

URL – Uniform Resource Locator

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

ВСТУП

Актуальність теми. Аудіо у сфері комп'ютерних технологій займає дуже велику нішу. Розвиток аудіотехнологій йде крок за кроком: З поширенням потокового аудіо, підвищенням якості звуку, розширенням мультимедійних можливостей сучасних пристроїв, розробка аудіопрогравачів є актуальною. Розробники створюють інноваційні рішення для задоволення потреб користувачів у зручному та функціональному програванні аудіо. Python надає широкий спектр бібліотек та модулів, які дозволяють розробникам створювати повнофункціональні аудіопрогравачі.

Python є кросплатформеною мовою програмування, що означає, що аудіопрогравачі, розроблені на ньому, можуть працювати на різних ОС, таких як Windows, macOS, Linux і багатьох інших. Це дає можливість розробляти універсальні аудіопрогравачі, які доступні для користувачів незалежно від їхньої платформи.

Загалом програвачі володіють різними функціональними можливостями і користуються певною популярністю серед спільноти розробників та користувачів. Розвиток Python-базованих аудіопрогравачів продовжується, а розробники активно підтримують та оновлюють ці проекти, щоб відповідати вимогам сучасного аудіосередовища.

Метою кваліфікаційної роботи є дослідження особливостей розробки програмного забезпечення та використання мови програмування Python для виконання цієї цілі.

Визначена мета обумовлює наступні завдання:

- аналіз теоретичних аспектів аудіофайлів;
- дослідження існуючих рішень для відтворення аудіофайлів;
- розробка логічної і оптимізованої концепції програми;
- вибір програмного забезпечення для розробки і налаштування його під проєкт;
- тестування роботи зробленого ПЗ.

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

Об’єктом дослідження є процес проектування програмного забезпечення, здатного відтворювати аудіофайли та використанням при цьому якомога менше ресурсів ПК.

Предметом дослідження є написання проєкту системи відтворення аудіоконтенту на мові програмування Python, дослідження його на стабільність роботи, виправлення помилок.

У процесі виконання кваліфікаційної роботи було розроблено проєкт системи відтворення аудіоконтенту на мові програмування Python, досліджено і перевірено роботу програми та обраних алгоритмів.

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

РОЗДІЛ 1 ТЕОРЕТИЧНІ АСПЕКТИ АУДІОФАЙЛІВ І РОБОТА З НИМИ

1.1 Теоретичні аспекти аудіофайлів

Аудіофайли є цифровими форматами збереження звукової інформації. Вони використовуються для запису, зберігання, передачі та відтворення звукових записів. Історія створення аудіофайлів пов'язана з розвитком аудіотехнологій і поширенням цифрових пристроїв.

Початки аудіозапису сягають XIX століття, коли були винайдені перші пристрої для запису звуку. У 1877 році Томас Едісон представив фонограф - пристрій, який записував і відтворював звук на воскових циліндрах. Протягом наступних десятиліть виникали різні формати аудіозапису, такі як грамофонні пластинки, аудіокасети і компакт-диски.

З появою цифрової технології з'явилися нові формати аудіофайлів, які зберігають звук у цифровому вигляді. Одним з перших популярних цифрових форматів був MP3 (MPEG-1 Audio Layer 3), який був розроблений у 1990-х роках. MP3 забезпечував ефективне стиснення аудіоданих, що дозволяло зменшити розмір файлів, при цьому зберігаючи задовільну якість звуку.

Згодом з'явилися інші популярні формати, такі як AAC (Advanced Audio Coding), WMA (Windows Media Audio) та OGG (Ogg Vorbis). Ці формати також пропонували стиснення аудіоданих з різними рівнями якості. Вони широко використовувалися для зберігання музики на комп'ютерах і портативних пристроях.

З розвитком Інтернету та збільшення швидкості передачі даних по мережі стали популярними онлайн-аудіофайлові сервіси, такі як Spotify, Apple Music, Google Play Music та інші. Ці сервіси дозволяють стрімінгове відтворення музики без необхідності завантажувати аудіофайли.

У сучасному світі аудіофайли займають значну нішу серед інших форматів. Вони є популярним способом збереження і відтворення музики, подкастів, аудіокниг, звукових ефектів та багатьох інших звукових матеріалів. Аудіофайли

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

також широко використовуються в музичній індустрії, звукозапису, кіноіндустрії, ігровій індустрії та різних сферах мультимедіа.

Серед користувачів, аудіофайли є дуже поширеними. Завдяки доступності цифрових пристроїв, які підтримують аудіофайли, таких як комп'ютери, смартфони, плеєри та інші пристрої, майже кожен користувач може мати доступ до аудіозаписів. Онлайн-аудіосервіси також сприяють зростанню популярності аудіофайлів, оскільки вони надають можливість легко слухати музику та інші аудіоматеріали через Інтернет.

Поява цифрових аудіофайлів мала значний вплив на розвиток комп'ютерних технологій і вплинула на людство в цілому. Ось кілька ключових аспектів цього впливу:

Зміна способу споживання музики: З появою цифрових аудіофайлів люди отримали змогу зберігати, передавати і відтворювати музику на комп'ютерах та інших пристроях. Це змінило спосіб, яким ми споживаємо музику, роблячи її більш доступною та портативною. Замість фізичних носіїв, таких як грамофонні пластинки або аудіокасети, ми можемо мати доступ до великої музичної колекції на наших комп'ютерах або смартфонах.

Зручність і збереження простору: Цифрові аудіофайли дозволили людям зберігати велику кількість музики на комп'ютерах, жорстких дисках або в хмарних сховищах. Це зберігання відбувається в компактному цифровому форматі, що значно економить простір порівняно з фізичними носіями, які можуть вимагати значно більше місця. Це зручно для користувачів, які можуть мати доступ до великої музичної бібліотеки без необхідності фізичного зберігання об'ємних носіїв.

Інтернет та онлайн-музика: Поява цифрових аудіофайлів збільшила популярність Інтернету як медіа-каналу. Музичні стрімінгові сервіси, такі як Spotify, Apple Music, YouTube Music, дозволяють користувачам легко відтворювати музику в режимі реального часу без необхідності завантаження аудіофайлів на пристрій. Це змінило підхід до відтворення музики, надаючи широкий доступ до великої кількості музичного контенту з усього світу.

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

Розвиток аудіотехнологій: Поява цифрових аудіофайлів стимулювала розвиток аудіотехнологій і програмного забезпечення, спрямованого на обробку, стиснення і покращення якості звуку. Це привело до появи нових аудіоформатів з вищою якістю звуку, таких як FLAC (Free Lossless Audio Codec) і ALAC (Apple Lossless Audio Codec), які зберігають звук без втрати якості. Також були розроблені аудіоредактори, плеєри і аудіоеквалайзери, які надають користувачам можливість налаштування звуку під їхні вподобання.

Узагальнюючи, поява цифрових аудіофайлів значно полегшила доступ до музики, змінила спосіб, яким ми споживаємо аудіоконтент, сприяла розвитку Інтернету та аудіотехнологій. Ці зміни вплинули на культурні, розважальні та технологічні аспекти життя людей.

Корисно розглянути найпоширеніший аудіоформат – «.mp3». Поширеність цього аудіоформату можна пояснити декількома факторами, а особливості цього формату порівняно з іншими сприяють його популярності.

Ефективність стиснення: Основна особливість формату .mp3 полягає в його здатності ефективно стискувати аудіодані без значної втрати якості звуку. Алгоритм стиснення .mp3 використовує методи видалення незначущих аудіоданих, які людське вухо майже не чує. Це дозволяє зменшити розмір файлу, несуттєво впливаючи на якість звуку. Ця ефективність стиснення робить формат .mp3 ідеальним для збереження та передачі музики через Інтернет, де обмежені пропускні здатності і обсяги зберігання.

Універсальність та сумісність: Формат .mp3 є одним з найбільш універсальних і поширених аудіоформатів. Він підтримується більшістю аудіоплеєрів, пристроїв, операційних систем і програмного забезпечення. Це означає, що файли .mp3 можуть бути легко відтворені на різних пристроях без необхідності конвертації в інші формати.

Широкий розповсюджений медіаформат: За час свого існування, формат .mp3 став медіастандартом для зберігання й передачі аудіофайлів. Багато музичних сервісів, онлайн-магазинів та стрімінгових платформ надають контент

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

у форматі .mp3. Також значна кількість музичних пристроїв, автомобільних систем та аудіоапаратури сумісна з цим форматом.

Розмір файлів: Завдяки ефективному стисненню, файли .mp3 мають досить низький розмір порівняно з некомпресованими форматами, такими як WAV або FLAC. Це дозволяє зберігати значну кількість музичних треків на обмеженому просторі зберігання, такому як жорсткий диск або пристрій з обмеженою пам'яттю.

Усі ці особливості формату .mp3 роблять його зручним і популярним використовувати для зберігання та передачі аудіофайлів, особливо в умовах обмеженого простору і обмеженої пропускної здатності.

1.2 Огляд існуючих рішень

На сучасному ринку аудіопрогравачів існує широкий вибір програмного забезпечення, яке дозволяє відтворювати аудіофайли на різних пристроях і платформах. Ось кілька з найбільш популярних аудіопрогравачів:

iTunes є одним з найпопулярніших аудіопрогравачів, розробленим компанією Apple. Він підтримує відтворення аудіофайлів у форматі .mp3 та інших популярних форматах. iTunes також є інтегрованим магазином музики, де користувачі можуть придбати та завантажити аудіотреки.

Windows Media Player є стандартним аудіопрогравачем, який входить до складу операційної системи Windows. Він підтримує різні аудіоформати, включаючи .mp3, і має широкий функціонал для організації і відтворення музики.

Winamp. популярний аудіопрогравач, який здатний відтворювати різні формати аудіофайлів. Він має зручний інтерфейс та багато функцій, таких як рівні використання, плагіни та можливість налаштування зовнішнього вигляду.

VLC Media Player - відкритий і безкоштовний аудіопрогравач, який підтримує широкий спектр аудіоформатів, включаючи .mp3. Він має простий інтерфейс і відмінну функціональність, таку як підтримка субтитрів і стрімінгове відтворення.

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		13

Spotify: Хоча Spotify є в основному стрімінговим сервісом, він також надає аудіопрогравач для відтворення музики, яку користувачі зберігають офлайн. Spotify підтримує широкий спектр форматів і має велику музичну бібліотеку.

Важко виокремити єдину програму, яка була б домінуючою. Вибір аудіопрогравача часто залежить від особистих вподобань та потреб користувача, а також від платформи, на якій він працює. Кожен з вищезгаданих аудіопрогравачів має свої унікальні особливості і користувацький досвід, тому вибір лідера залежить від конкретних пріоритетів кожного користувача.

Розберемо детально аудіопрогравач, який найбільше схожий концепт ПЗ, яке представлено в роботі.



Рисунок 1.1 — Логотип програми WinAmp

Winamp - це аудіопрогравач, який розроблявся компанією Nullsoft та вперше випущений у 1997 році. Він став відомим своїм простим інтерфейсом, розширеннями та здатністю відтворювати різні аудіоформати. Winamp швидко набув популярності серед користувачів ПК і досяг піку своєї популярності у кінці 1990-х років та на початку 2000-х.

Протягом свого розвитку Winamp отримав ряд оновлень та нововведень, включаючи покращений звук, візуальні ефекти, можливості організації медіатеки та підтримку різних скинів для зміни зовнішнього вигляду програвача. Він також мав вбудовану підтримку радіо та потокового відтворення.

Однак, з часом популярність Winamp стала зменшуватися. У 2014 році компанія AOL, власник Nullsoft, оголосила, що припиняє активну розробку

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		14

Winamp і закриває його сайт. Проте, шанувальники програми продовжили використовувати і підтримувати Winamp. У 2018 році Radionomy, дочірня компанія українського підприємця Александра Саба (Alexandre Saboundjian), придбала Winamp разом з іншими активами компанії AOL.

На сьогоднішній день Winamp продовжує існувати і знаходиться під активною розробкою та підтримкою. Існує офіційний веб-сайт, на якому можна завантажити останню версію Winamp і отримати додаткову інформацію про продукт. Розробники продовжують випускати оновлення і вдосконалення, щоб забезпечити сумісність з останніми версіями операційних систем і забезпечити задоволення користувачів, які залишаються вірними цьому аудіопрогравачу.

Розглянемо найбільш функціональний аудіопрогравач – AIMP.



Рисунок 1.2 — Логотип програми AIMP

Це популярний аудіопрогравач, який має велику кількість функцій і особливостей, які роблять його зручним і потужним інструментом для відтворення аудіофайлів.

Особливість AIMP полягає в його широкому функціоналі та можливостях налаштування. Основні особливості включають наступне:

Відтворення різних аудіоформатів. AIMP підтримує велику кількість аудіоформатів, включаючи .mp3, .wav, .flac, .aac, .ogg та багато інших. Це дозволяє користувачам відтворювати свою улюблену музику у високій якості без необхідності конвертації форматів.

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

Музична бібліотека. АІМР має потужну музичну бібліотеку, яка дозволяє організувати та категоризувати музичні файли. Користувачі можуть створювати плейлисти, редагувати теги музичних треків і шукати музику за різними критеріями.

Еквалайзер і звукові ефекти. АІМР має вбудований еквалайзер, який дозволяє користувачам налаштовувати звукові параметри, такі як баси, середина, високі тони тощо. Крім того, він підтримує звукові ефекти, такі як реверберація, ехо, затримка та інші.

Скини та налаштування інтерфейсу. АІМР надає можливість змінювати зовнішній вигляд програвача шляхом вибору різних скинів. Крім того, користувачі можуть налаштовувати інтерфейс, розміщення елементів керування та гарячі клавіші за своїм вподобанням.

Розширені можливості управління. АІМР має багато швидких команд та гарячих клавіш для зручного управління відтворенням музики, переключення треків, регулювання гучності тощо. Також він підтримує пульт дистанційного керування.

АІМР активно розробляється і оновлюється командою розробників. Регулярно випускаються нові версії програми з виправленнями помилок, покращеннями функціональності та новими функціями. Розробники відповідають на запити та підтримують спільноту користувачів, що допомагає забезпечити стабільну роботу та розвиток АІМР.

Корисно буде розглянути аудіопрогравачі, які зв'язані зі стримінг-сервісами, наприклад Spotify.



Рисунок 1.3 — Логотип програми Spotify

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

Spotify - це популярний аудіопрогравач та стрімінгова музична платформа, розроблена компанією Spotify AB. Його основна особливість полягає у можливості безкоштовного або платного стрімінгу мільйонів музичних треків з усього світу.

Основні особливості Spotify включають наступне:

Безкоштовний та платний варіант: Spotify пропонує два варіанти використання. Безкоштовний варіант дозволяє користувачам відтворювати музику з обмеженнями, з рекламою та обмеженням на перескакування треків. Платний варіант, відомий як Spotify Premium, забезпечує безобмежений доступ до музики без реклами, можливість завантаження треків для прослуховування в автономному режимі, вищу якість звуку та інші переваги.

Велика музична бібліотека: Spotify має вражаючу колекцію музичних треків, яка охоплює різні жанри, виконавців та регіони. Користувачі можуть знайти свої улюблені пісні, альбоми та плейлисти або відкривати нові музичні відкриття за допомогою рекомендаційних систем та персоналізованих пропозицій.

Радіо та кураторство плейлистів: Spotify пропонує функцію радіо, яка дозволяє користувачам відтворювати музику на основі жанру, виконавця або певної пісні. Крім того, у Spotify є широка колекція плейлистів, які куруються власними експертами та спільнотою користувачів.

Синхронізація між пристроями: Spotify дозволяє користувачам синхронізувати свою музичну колекцію та плейлисти між різними пристроями. Ви можете почати слухати пісню на одному пристрої та продовжити на іншому без перебоїв.

Соціальна взаємодія: Spotify надає можливість спільного слухання музики з друзями, створення спільних плейлистів та обмін музичними рекомендаціями через інтегровані соціальні функції.

Spotify активно розробляється та підтримується командою розробників. Це проявляється в регулярних оновленнях програми, які включають в себе нові функції, виправлення помилок та покращення продуктивності. Команда

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

розробників також вислуховує зворотний зв'язок користувачів та враховує їхні пропозиції для подальшого розвитку платформи.

1.3 Огляд програмного забезпечення для розробки

Для створення аудіопрогравачів використовуються різні програмні засоби та технології, залежно від конкретних потреб і вимог розробників. Розглянемо кілька основних програмних засобів, які можуть використовуватися для створення аудіопрогравачів:

Програмування мовою C/C++: Багато аудіопрогравачів розробляються з використанням мови програмування C або C++. Ці мови надають широкі можливості для роботи з аудіофайлами, зчитування та декодування аудіоданих, відтворення звуку, налаштування ефектів і багато іншого.

Аудіофреймворки: Існують спеціалізовані аудіофреймворки, які допомагають в розробці аудіопрогравачів, надаючи готові інструменти та функції для роботи з аудіо. Наприклад, фреймворки, такі як PortAudio, BASS або FMOD, надають API для відтворення звуку, зчитування аудіоданих, обробки ефектів і багато іншого.

Мультимедійні бібліотеки: Деякі розробники використовують мультимедійні бібліотеки для створення аудіопрогравачів. Наприклад, бібліотека libVLC (VideoLAN) надає широкі можливості для відтворення аудіо- та відеоданих, підтримує різні формати і протоколи стрімінгу.

Розробка на основі існуючих платформ: Деякі розробники використовують існуючі платформи або середовища розробки для створення аудіопрогравачів. Наприклад, використання платформи Android або iOS дозволяє створювати мобільні аудіопрогравачі для смартфонів та планшетів.

Використання мов програмування вищого рівня: Іноді для розробки аудіопрогравачів використовуються мови програмування вищого рівня, такі як Java, C#, Python або JavaScript. Ці мови мають широку підтримку бібліотек та фреймворків, що спрощує розробку аудіопрогравачів.

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

Вибір програмного забезпечення для створення аудіопрогравача залежить від багатьох факторів, таких як цільова платформа, функціональні вимоги, екосистема розробника та особисті вподобання. Комбінація різних засобів та технологій дозволяє створити потужні та функціональні аудіопрогравачі, які задовольняють потреби користувачів.

Зважаючи на проаналізовану інформацію і можливості, які буде містити в собі аудіопрогравач, було обрано розробляти його на мові програмування Python. Розглянемо її детальніше.

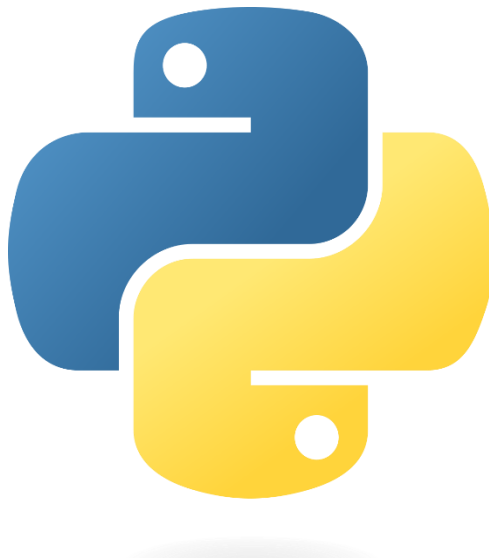


Рисунок 1.4 — Логотип мови програмування Python

Python - це високорівнева, інтерпретована, мультипарадигмальна мова програмування, яка здобула велику популярність серед розробників. Деякі відмінності Python від інших мов програмування:

- Простота та зрозумілість: Python має простий і лаконічний синтаксис, що полегшує його вивчення та розуміння.
- Широкі можливості: Python підтримує багато парадигм програмування, включаючи процедурне, об'єктно-орієнтоване, функціональне та аспектно-орієнтоване програмування.
- Велика стандартна бібліотека: Python має велику колекцію модулів та бібліотек, що спрощує розробку різних типів програм та задач.

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

- Хороша співпраця з іншими мовами: Python може бути використаний для інтеграції з кодом, написаним на C, C++, Java та інших мовах програмування.

Python був розроблений у 1989 році Гвідо ван Россумом у Нідерландах. Він прагнув створити просту та читабельну мову програмування, яка б підходила для широкого кола завдань.

Перша публічна версія Python, 0.9.0, була випущена у 1991 році. З тих пір мова пройшла кілька значних оновлень і вдосконалень, включаючи випуск стабільної версії 1.0 у 1994 році та версії 2.0 і 3.0 у 2000-х роках.

Основні принципи розробки Python - це простота, зрозумілість та зручність для розробників. Мова здобула популярність через свою ефективність, гнучкість та широкі можливості.

Існують галузі програмування, де Python лідер:

- Веб-розробка: Python має популярні фреймворки, такі як Django та Flask, які допомагають швидко розробляти веб-додатки та веб-сайти.
- Наука про дані та штучний інтелект: Python є популярним вибором для аналізу даних, машинного навчання, глибокого навчання та інших областей штучного інтелекту.
- Автоматизація та скриптинг: Python використовується для автоматизації рутинних задач, написання скриптів та розробки утиліт.
- Ігрова розробка: Python знайшов застосування в розробці ігор, особливо для прототипування, штучного інтелекту та скриптингу.

Python мав вплив на розвиток комп'ютерних технологій. Наприклад, спрощена синтаксис та зрозумілість Python зробили його більш доступним для новачків у програмуванні, що сприяло поширенню цієї мови і залученню більшої кількості розробників до індустрії програмування.

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

Python займає вагоме місце в області науки про дані, штучного інтелекту та машинного навчання, де він став основною мовою для розробки алгоритмів, моделей та аналізу даних.

Багато відкритих проектів та бібліотек Python сприяли розширенню функціональності та продуктивності програмного забезпечення в різних галузях.

Python продовжує активно розвиватися, оновлюючи свої версії та надаючи нові можливості розробникам. Він залишається популярним вибором для різних завдань програмування та має значний вплив на світ комп'ютерних технологій.

У якості IDE було обрано ПЗ PyCharm від компанії JetBrains.



Рисунок 1.5 — Логотип програми PyCharm

PyCharm - це інтегроване середовище розробки (IDE) для мови програмування Python, розроблене компанією JetBrains.

Перша версія PyCharm була випущена в 2010 році компанією JetBrains. Вона була створена як спеціалізоване IDE для розробки на мові Python з використанням функцій, які раніше були доступні в інших продуктах компанії, таких як IntelliJ IDEA. З того часу PyCharm пройшов чергу значних оновлень і покращень, включаючи підтримку нових функцій Python, покращення продуктивності та розширення функціональності.

Зазначимо деякі особливості PyCharm:

Розширений редактор коду, що надає потужний та інтуїтивно зрозумілий редактор коду з функціями підсвічування синтаксису, автодоповнення, переходу до визначень функцій та багатьма іншими.

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

Вбудована підтримка віртуальних середовищ. PyCharm інтегрується з інструментами створення та управління віртуальними середовищами, такими як `virtualenv` та `Anaconda`, що дозволяє легко створювати ізольовані середовища для розробки.

Відладка та профілювання. PyCharm надає розширені засоби для відлагодження коду, включаючи можливість встановлення точок зупину, крокування по коду та аналізу змін змінних. Воно також підтримує профілювання коду для виявлення його ефективності та оптимізації.

Інтеграція з іншими інструментами. Підтримує інтеграцію з версійним контролем, системами керування проектами, базами даних та іншими зовнішніми інструментами, що полегшує розробку та спільну роботу у команді.

PyCharm відомий своєю високою зручністю та дружнім інтерфейсом користувача, що робить його популярним серед розробників Python. Він підтримує розробку як на платформах Windows, macOS, так і на Linux, що робить його доступним для широкої аудиторії.

Це комерційний продукт, але також має безкоштовну "Community Edition" з обмеженою функціональністю для некомерційного використання.

Його поширеність серед розробників зростає, і він використовується в різних галузях програмування, таких як веб-розробка, наукова обробка даних, машинне навчання та інші.

JetBrains продовжує активно розвивати PyCharm, випускаючи нові версії та вдосконалюючи його функціональність. Це інтегроване середовище розробки стало популярним інструментом для розробки на мові Python завдяки своїм можливостям, зручному інтерфейсу та підтримці великої спільноти розробників.

Для створення графічного інтерфейсу, з підтримкою стилів(CSS) було обрано ПЗ Qt Designer.

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		22



Рисунок 1.6 — Логотип програми Qt Designer

Qt Designer - це графічний редактор інтерфейсу користувача, що постачається разом з бібліотекою Qt. Він дозволяє розробникам створювати візуальні інтерфейси програм за допомогою методу перетягування та розміщення (drag-and-drop) елементів.

Qt Designer ідеально підходить для розробки інтерфейсів для Python-програм. Він забезпечує можливість зберігати створені інтерфейси у форматі файлів ".ui", які можна імпортувати і використовувати в Python-кодi. Він надає широкий набір візуальних елементів інтерфейсу, таких як кнопки, поле введення, випадаючі списки, таблиці та інші. Розробник може вибрати та розмістити ці елементи у вікні програми за допомогою перетягування та розміщення.

Також він дозволяє встановлювати властивості елементів інтерфейсу, такі як текст, розмір, колір та інші. Також можна налаштовувати події, пов'язані з цими елементами, що дозволяє реалізувати функціональність програми.

Після створення інтерфейсу у Qt Designer, файл ".ui" можна перетворити у відповідний Python-код. Це дозволяє інтегрувати створений інтерфейс зі звичайним Python-кодом.

Qt Designer використовується разом з бібліотеками PyQt або PySide, які надають Python-інтерфейс до функціональності Qt. Ці бібліотеки дозволяють управляти елементами інтерфейсу, обробляти події, виконувати динамічні зміни та багато іншого.

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

Цей графічний редактор надає ряд інструментів для налаштування вигляду та поведінки інтерфейсу, таких як стилі оформлення (stylesheets), які дозволяють змінювати зовнішній вигляд програми без зміни коду.

Qt Designer є потужним інструментом для швидкого створення візуального інтерфейсу для Python-програм. Його можливості та зручність у використанні сприяють прискоренню розробки та полегшують роботу розробників.

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

РОЗДІЛ 2 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Огляд модулів, які будуть використовуватися при розробці

PySide6 — це офіційний модуль Python із проекту Qt для Python, який надає доступ до повного фреймворку Qt 6.0+.

Проект Qt для Python розробляється відкрито, з усіма можливостями, які розробник очікує від будь-якого сучасного проекту OSS, наприклад весь код у репозиторії git і відкритий процес проектування.

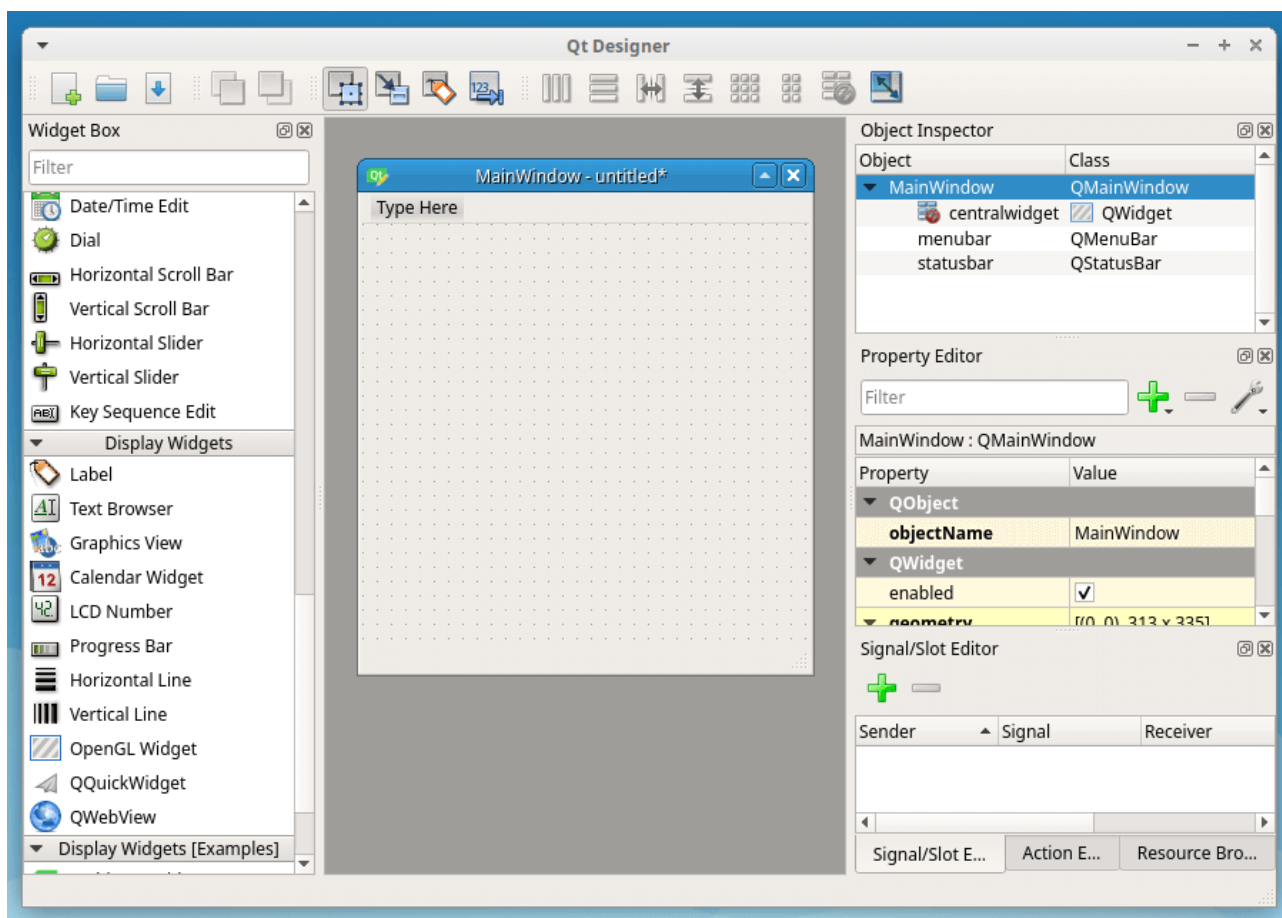


Рисунок 2.1 — Приклад створення графічного інтерфейсу, використовуючи модуль PySide6

PySide6 є бібліотекою для створення графічного інтерфейсу користувача (GUI) у мові програмування Python. Вона є офіційною прив'язкою до фреймворку Qt, як вказувалося вище, що дозволяє розробляти хмарні, настільні та мобільні програми.

Ось деякі ключові особливості модуля PySide6:

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

- Кросплатформеність: PySide6 працює на різних операційних системах, включаючи Windows, macOS, Linux, Android та iOS. Це дозволяє вам розробляти програми, які працюють на багатьох платформах без значних змін у коді.
- Повна прив'язка до Qt: PySide6 надає повний доступ до функціональності Qt, включаючи можливості створення вікон, кнопок, полів введення, таблиць, меню, діалогових вікон та багато іншого. Ви можете використовувати всі можливості Qt для створення потужних та ефективних інтерфейсів.
- Pythonic API: API PySide6 розроблено з дотриманням стилю програмування Python. Воно є інтуїтивно зрозумілим і приємним для використання, що дозволяє розробникам швидко створювати інтерфейси.
- Сигнали та слоти: PySide6 використовує механізм сигналів та слотів Qt для обробки подій. Це дає можливість ефективно реагувати на взаємодію користувача з елементами інтерфейсу та інших подій.
- Модуль QML: PySide6 також надає можливість розробки інтерфейсів за допомогою мови опису QML (Qt Meta-Object Language). QML дозволяє створювати динамічні та привабливі інтерфейси з використанням декларативного синтаксису.
- Qt Designer: PySide6 поставляється з інструментом Qt Designer, який дозволяє вам візуально створювати інтерфейси за допомогою перетягування та розміщення елементів. Ви можете експортувати створені інтерфейси у файл формату .ui, який потім можна імпортувати у ваш Python-код.

За допомогою модуля PySide6 можна розробити графічний інтерфейс аудіопрогравача на мові програмування Python. Ось кілька можливих функціональних компонентів, які можна включити до проекту:

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

Вікно програми: Можна створити вікно програми за допомогою класу QMainWindow з PySide6. Це буде головне вікно аудіопрограваача, яке можна налаштувати за своїми потребами. Вікно може містити різні елементи управління, такі як кнопки, слайдери або текст.

Елементи управління: Можна використовувати різні елементи управління для керування аудіопрограваачем. Наприклад, кнопки «Відтворити», «Пауза» та «Зупинити» можуть виконувати відповідні дії з аудіофайлами. Слайдери можуть відображати прогрес відтворення або дозволяти переміщати позицію відтворення у файлі. Крім того, можна включити елементи керування гучністю або режимами повторення/випадкового відтворення.

Перегляд списку аудіофайлів: Планується мати можливість вибирати та відтворювати різні аудіофайли. Треба включити компонент, який дозволяє переглядати список доступних файлів. Це може бути список (QListWidget) або таблиця (QTableWidget), в якому відображається назви аудіофайлів.

Функції відтворення аудіо: Для відтворення аудіофайлів можна використовувати підхід на основі модуля QMediaPlayer. Цей клас дозволяє завантажувати аудіофайли, керувати відтворенням, змінювати гучність, переміщати позицію відтворення та отримувати події, пов'язані зі станом відтворення.

Обробка подій із клавіатури або миші: Треба надати можливість користувачу керувати аудіопрограваачем за допомогою клавіш клавіатури або подій миші. Наприклад, зробити так, щоб натискання на кнопки «Відтворити», «Пауза» призупиняло і відновлювало відтворення відповідно, або щоб переміщення слайдера мишею змінювало позицію відтворення.

Стандартний модуль sys.

Модуль sys є вбудованим модулем у мові програмування Python, який надає доступ до різноманітної системної функціональності та змінних. Він надає широкі можливості для взаємодії з операційною системою, отримання інформації про виконання програми та керування виконанням самої програми.

Ось декілька важливих функцій та змінних, які надає модуль sys:

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

- `sys.argv`: Це список аргументів командного рядка, які були передані програмі при запуску. Ви можете отримати доступ до цих аргументів та їх значень для подальшої обробки в програмі.
- `sys.exit([status=0])`: Ця функція дозволяє завершити виконання програми. За замовчуванням, коли програма завершується успішно, їй присвоюється код повернення 0. Але ви також можете передати необов'язковий аргумент `status`, щоб вказати інший код повернення.
- `sys.stdin`, `sys.stdout`, `sys.stderr`: Ці змінні представляють стандартні потоки вводу, виводу та помилок відповідно. Ви можете перенаправити ці потоки або використовувати їх для зчитування вводу користувача або виведення результатів в консоль.
- `sys.modules`: Це словник, який містить назви всіх завантажених модулів у поточному виконанні програми. Ви можете використовувати його для отримання інформації про завантажені модулі або для динамічного імпорту модулів.
- `sys.path`: Це список шляхів пошуку для модулів. Він містить шляхи до папок, в яких Python шукає модулі для імпорту. Ви можете змінювати цей список, щоб додавати власні шляхи пошуку або виключати певні шляхи.
- `sys.platform`: Ця змінна містить рядок, який вказує на поточну платформу виконання. Наприклад, "win32" для Windows, "linux" для Linux, "darwin" для macOS.

Модуль `sys` має і багато інших функцій та змінних, які допомагають у взаємодії з операційною системою, конфігурацією програми та іншими системними аспектами. Він є потужним інструментом для програмістів Python, які потребують працювати з системними ресурсами під час розробки програм.

Стандартний модуль `os`.

Модуль `os` є вбудованим модулем у мові програмування Python, який надає функції для взаємодії з операційною системою. Він дозволяє працювати з

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

файловою системою, отримувати інформацію про системні ресурси та виконувати операції, пов'язані з операційною системою.

Ось декілька важливих функцій та змінних, які надає модуль `os`:

- `os.name`: Ця змінна містить ім'я операційної системи, на якій виконується програма. Наприклад, "posix" для UNIX-подібних систем (Linux, macOS) або "nt" для Windows.
- `os.getcwd()`: Ця функція повертає поточний робочий каталог (поточний шлях) вашої програми. Вона дозволяє отримати шлях до поточного робочого каталогу та працювати з файлами та папками в цьому каталозі.
- `os.listdir(path='.')`: Ця функція повертає список всіх файлів та папок у заданому каталозі `path`. Якщо не вказати `path`, вона поверне список файлів та папок у поточному робочому каталозі.
- `os.path`: Цей модуль надає функції для роботи з шляхами файлів та папок. Ви можете використовувати його для об'єднання шляхів, розбиття шляху на складові частини, отримання імені файлу або каталогу, перевірки наявності файлу або каталогу та багато іншого.
- `os.mkdir(path)`: Ця функція створює нову папку з вказаним шляхом `path`. Ви можете використовувати її для створення нових каталогів у файловій системі.
- `os.remove(path)`: Ця функція видаляє файл з вказаним шляхом `path`. Вона дозволяє видаляти файли з файлової системи.
- `os.rename(src, dst)`: Ця функція перейменовує або переміщує файл або папку з вказаного шляху `src` на новий шлях `dst`. Вона дозволяє змінювати імена файлів або переміщувати їх до інших папок.
- `os.system(command)`: Ця функція виконує команду або програму у командному рядку. Ви можете використовувати її для виконання зовнішніх команд або програм з вашої Python-програми.

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		29

Модуль `os` також містить багато інших функцій, таких як керування процесами, робота з середовищем виконання, робота з правами доступу до файлів і багато іншого. Він дозволяє вам ефективно взаємодіяти з операційною системою під час розробки програми на Python.

Стандартний модуль `asyncio`.

Модуль `asyncio` є вбудованим модулем у мові програмування Python, який надає підтримку для асинхронного програмування з використанням синтаксису `async/await`. Він дозволяє створювати асинхронні функції, задачі та корутини, які працюють паралельно та неблокуючим способом, що дозволяє ефективно виконувати велику кількість операцій без зайвого очікування.

Основні поняття та функції, пов'язані з модулем `asyncio`, включають:

- **Event Loop:** Ядро асинхронної програми на `asyncio` - це подійний цикл (`event loop`). Цикл подій обробляє інтервали часу, події вводу-виводу та запускає асинхронні функції. Він дозволяє виконувати багато задач паралельно та реагувати на події, коли вони стають доступними.
- **Асинхронні функції та корутини:** Можна створювати асинхронні функції з використанням ключового слова `async` перед оголошенням функції. Виклик асинхронної функції повертає корутину, яку можна запустити в `event loop` за допомогою `await`.
- **Задачі (Tasks):** Задача - це корутин, який виконується в `event loop`. Ви можете створити задачу з корутини за допомогою `create_task` або `ensure_future`. Задача додається до `event loop` та запускається паралельно з іншими задачами.
- **Асинхронні об'єкти вводу-виводу:** `asyncio` надає різні класи та функції для роботи з вводом-виводом. Наприклад, `asyncio.sleep` дозволяє затримати виконання корутини на певний інтервал часу. `asyncio.open_connection` дозволяє створити асинхронне з'єднання з сервером.

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		30

- Синхронізація: `asyncio` надає різні інструменти для синхронізації асинхронних задач. Наприклад, `asyncio.Lock` дозволяє захоплювати та звільняти блокування, щоб уникнути конфліктів доступу до спільних ресурсів.
- Обробка подій: `asyncio` надає можливість виконувати дії при виникненні певних подій. Наприклад, ви можете додати обробники подій для введення-виводу, таймерів або мережевих з'єднань, щоб реагувати на відповідні події.

Модуль `asyncio` є потужним інструментом для розробки асинхронних програм у Python. Він дозволяє покращити продуктивність та ефективність програм, особливо при взаємодії з мережевими операціями або введенням-виведенням з великими затримками.

Модуль `pytube`.

Модуль `pytube` є бібліотекою для завантаження відео з YouTube у мові програмування Python. Він надає простий спосіб отримання URL-адреси відео та завантаження його на локальний пристрій. `pytube` підтримує завантаження як аудіо, так і відео файлів з YouTube.

Основні функції та можливості модуля `pytube`:

Отримання відео-URL: `pytube` дозволяє отримати URL-адресу відео з YouTube за допомогою функції `YouTube()`. Потрібно передати URL відео як рядок, і він поверне об'єкт `YouTube`, який містить інформацію про відео.

- Вибір якості відео: За допомогою об'єкта `YouTube`, ви можете отримати доступні формати та якості відео за допомогою методу `streams`. Є можливість вибрати певний формат або якість відео, яке потрібно завантажити.
- Завантаження відео: Після вибору формату або якості відео, починається процес завантаження за допомогою методу `download()`. Він завантажить відео вказаного формату на локальний пристрій.

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		31

- Аудіо завантаження: `pytube` також дозволяє завантажувати лише аудіо з відео на YouTube. Можна отримати аудіоформати та якості, вибрати певний формат та завантажити його на пристрій.
- Потокове завантаження: Модуль `pytube` підтримує потокове завантаження відео, що дозволяє почати відтворення відео навіть під час його завантаження.

`Pytube` є потужним і простим у використанні інструментом для завантаження відео з YouTube у мові програмування Python.

Частина модулю `moviepy`: `moviepy.editor`.

Модуль `moviepy.editor` є частиною пакету `moviepy` і є стороннім модулем у мові програмування Python. Він надає можливості для редагування та обробки відеофайлів в Python. З допомогою цього модуля можна обрізати відео, з'єднувати кадри, додавати звукові доріжки, налаштовувати швидкість відтворення та виконувати багато інших операцій над відео.

Основні функції та можливості, які надає модуль `moviepy.editor`, включають:

- Створення об'єкту `VideoFileClip`. Дозволяє створити об'єкт `VideoFileClip`, який представляє відеофайл і дозволяє змогу працювати з ним.
- Редагування відео. Модуль `moviepy.editor` дозволяє виконувати різноманітні редагувальні операції над відео: обрізати відео за певними часовими межами, змінювати його розміри, змінювати яскравість, контрастність та інші властивості, додавати текст або зображення до відео, застосовувати фільтри та ефекти і багато іншого.
- З'єднання відео. Модуль дає можливість з'єднувати кілька відеофайлів разом для створення нового відео. Модуль `moviepy.editor` надає зручні функції для злиття відеофайлів та встановлення переходів між ними.

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		32

- Робота зі звуком. Можливість додавати звукові доріжки до відео, вилучати аудіо з відеофайлів, змінювати гучність звуку, синхронізувати аудіо та відео і багато іншого.
- Експорт відео. Модуль `moviepy.editor` дозволяє зберігати оброблене відео в різних форматах, таких як MP4, AVI, GIF та інші.

Модуль `moviepy.editor` є потужним інструментом для редагування та обробки відеофайлів у мові програмування Python. Він надає простий та зручний інтерфейс для виконання різноманітних операцій з відео, що робить його корисним для автоматизації рутинних завдань, створення відеоконтенту та багатьох інших сценаріїв.

Стандартний модуль `re`.

Модуль `re` є вбудованим модулем у мові програмування Python, який надає можливості для роботи з регулярними виразами. Він дозволяє здійснювати пошук, заміну та обробку тексту, використовуючи патерни, що відповідають певним шаблонам.

Основні функції та можливості, які надає модуль `re`, включають:

Пошук за патерном: Модуль `re` дозволяє шукати в тексті рядки, які відповідають певному патерну. Патерн визначає шаблон, який використовується для пошуку. Наприклад, ви можете шукати всі входження певного слова або фрази.

Заміна тексту: Ви можете замінювати відповідні рядки в тексті на нові значення за допомогою модуля `re`. Наприклад, замінити всі входження певного слова іншим словом або фразою.

Розбиття тексту: Модуль `re` дозволяє розбити текст на підрядки, використовуючи певний роздільник або патерн. Це корисно, коли потрібно розділити текст на окремі слова або фрази.

Групування та вилучення: Ви можете групувати патерни за допомогою дужок `()` та використовувати їх для вилучення конкретних частин тексту. Наприклад, вилучити всі електронні адреси з тексту або окремі компоненти дати.

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

Модифікатори та спецсимволи: Модуль те підтримує різні модифікатори та спецсимволи, які дозволяють розширити функціональність регулярних виразів. Наприклад, використовувати модифікатори для зроблення шаблону регістронезалежним або для пошуку на початку або кінці рядка.

Модуль те є потужним інструментом для роботи з регулярними виразами у мові програмування Python. Він дозволяє здійснювати розширений пошук, заміну та обробку тексту за допомогою патернів, що дозволяє вирішувати багато завдань, пов'язаних з маніпулюванням текстовими даними.

2.2. Написання програмного забезпечення

Спочатку було зроблено графічний інтерфейс у редаторі Qt Designer.

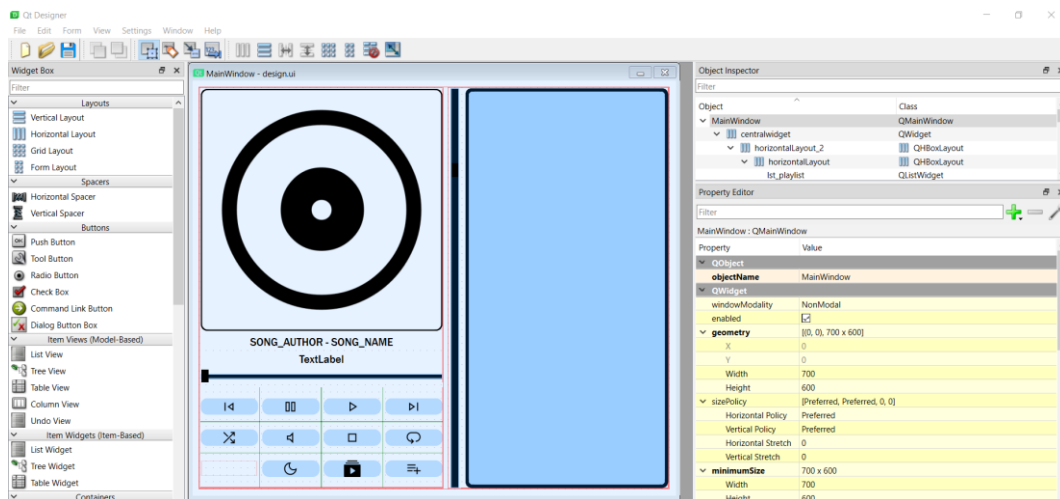


Рисунок 2.2 — Розробка дизайну головного вікна програми

Також було створене додаткове вікно, яке відкривається, коли користувач захоче завантажити файл\файли з сервісу YouTube.

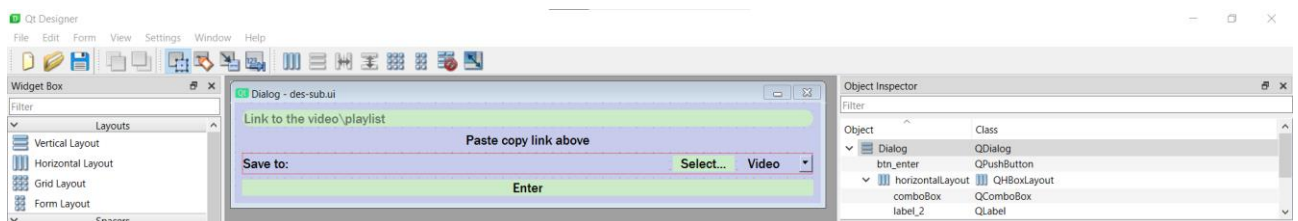


Рисунок 2.3 — Розробка дизайну додаткового вікна програми

Тепер треба конвертувати файли, які створив Qt Designer з формату «.ui» у формат «.ру». Це можливо зробити за допомогою вказаної нижче команди у терміналі.

pyside6-uic <name>.ui -o <name>.py

Головний файл main.py, де буде ініціалізуватися головне вікно буде мати такий вигляд:

```
import sys

from PySide6.QtWidgets import QApplication, QMainWindow

from PySide6.QtMultimedia import QMediaPlayer, QAudioOutput
from PySide6.QtGui import QPixmap

from design import Ui_MainWindow

import controls, playlist_controlls, work_with_file, work_with_youtube,
changing_elements, themes, show_error_mess

class Audioplayer(QMainWindow):
    def __init__(self):
        super(Audioplayer, self).__init__()
        self.ui = Ui_MainWindow() # Creating Main window
        self.ui.setupUi(self)
        self.path = '' # Variable that save one path to playlist
        self.playlist = [] # Variable that save songs in playlist
        self.img_album_cover = QPixmap('icons/album_cover_sample.png')
        self.change_album_cover()
        self.load_playlists('playlists.txt') # Loading latest playlist from
file
        self.ui.label.setText('')
        self.ui.label_time.setText('')
        self.duration_of_song = 0
        self.loop_flag = False
        self.path_to_audio_from_youtube = 'music_from_youtube'
        self.random_flag = False
        self.theme_index = 0

        # Audio
        self.player = QMediaPlayer() # Initializing audio source
        self.audio_output = QAudioOutput()
        self.player.setAudioOutput(self.audio_output)
        self.current_time = 0 # Variable that save current time in song
        self.current_song = '' # Variable that save name of song that
playing now
        self.audio_output.setVolume(self.ui.sldr_volume.value())
```

Спочатку імпортуємо необхідні модулі, створюємо клас Audioplayer, який наслідується від QMainWindow. Далі оглянемо створені змінні у цьому класі.

self.path – абсолютний шлях до каталогу плейлиста.

self.playlist – зберігає назву кожного аудіофайлу плейлиста у вигляді списку.

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

`self.img_album_cover` – фото, яке може бути обкладинкою музичного альбому.

`self.duration_of_song` – к-сть часу у мілісекундах, яка вказує на довжину відтворюваного аудіофайлу.

`self.loop_flag` – індикатор, який показує чи зациклений відтворюваний аудіофайл.

`self.path_to_audio_from_youtube` – папка, в якій будуть зберігатися завантажені файли завантажені з сервісу YouTube.

`self.random_flag` – індикатор, який показує чи включена випадкове відтворення аудіо.

`self.theme_index` – зберігає індекс теми для головного вікна.

`self.player` – об'єкт класу `QMediaPlayer`, який буде робити відповідні маніпуляції з аудіофайлами.

`self.audio_output` – об'єкт класу `QAudioOutput`, який буде керувати гучністю, тощо.

`self.current_time` – поточний прогрес по часу відтворюваного аудіо.

`self.current_song` – поточний відтворюваний аудіо файл.

Далі додамо обробку подій при взаємодії з графічним інтерфейсом.

```
# Events
self.ui.btn_add_playlist.clicked.connect(self.create_playlist)
self.ui.btn_add_playlist_from_youtube.clicked.connect(self.create_playlist_from_youtube)
self.ui.btn_play.clicked.connect(self.play_audio)
self.ui.btn_stop.clicked.connect(self.stop_audio)
self.ui.btn_pause.clicked.connect(self.pause_audio)
self.ui.lst_playlist.itemDoubleClicked.connect(self.play_audio_from_playlist)
self.ui.sldr_progress.sliderMoved.connect(self.song_progress_change)
self.ui.sldr_progress.valueChanged.connect(self.change_time_label)
self.ui.sldr_volume.sliderMoved.connect(self.change_volume)
self.ui.btn_next.clicked.connect(self.next_song)
self.ui.btn_previous.clicked.connect(self.previous_song)
self.ui.btn_loop.clicked.connect(self.change_btn_loop)
self.ui.btn_mute.clicked.connect(self.mute)
self.ui.btn_randow.clicked.connect(self.random)
self.ui.btn_theme.clicked.connect(self.change_theme)

# Events for audio source
self.player.durationChanged.connect(self.change_duration)
self.player.positionChanged.connect(self.song_position_change)
```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

Вони прив'язані до методів класу Audioplayer, а ті в свою чергу викликають функції, які знаходяться в окремих файлах. Оглянемо один за одним функції у кожному файлі окремо. Почнемо з файлу controls.py, який відповідає збільшого за події з кнопками головного вікна. Спочатку імпортуємо необхідні модулі, далі йде перша функція, яка викликається, коли користувач натиснув «Play».

```
def play_audio(self):
    """Start playing audio when button was pushed"""
    if self.player.playbackState() ==
QMediaPlayer.PlaybackState.PausedState:
        self.player.setPosition(self.current_time)
        self.player.play()
    else:
        audio_path = self.path + '/' + self.current_song
        self.player.stop()
        self.player.setSource(QUrl.fromLocalFile(audio_path))
        self.player.setPosition(0)
        self.player.play()
```

Вона відповідає за відтворення аудіо. В залежності від стану відтворення обирається подальший алгоритм дій.

Наступна функція зупиняє відтворення, коли користувач натиснув кнопку «Stop».

```
def stop_audio(self):
    self.player.stop()
```

Функція, яка відповідає за стан «Mute» виглядає так:

```
def mute(self, theme_index):
    """Muting song"""
    if self.ui.btn_mute.isFlat():
        if theme_index == 0:
            self.ui.btn_mute.setStyleSheet('background-color: #b3d9ff;')
        elif theme_index == 1:
            self.ui.btn_mute.setStyleSheet('background-color: rgb(203, 236,
199);')
        elif theme_index == 2:
            self.ui.btn_mute.setStyleSheet('background-color: #cccccc;')
        elif theme_index == 3:
            self.ui.btn_mute.setStyleSheet('background-color: # 70db70;')
        self.audio_output.setMuted(False)
        self.ui.btn_mute.setFlat(False)
    else:
        if theme_index == 0:
            self.ui.btn_mute.setStyleSheet('background-color: #0073e6;')
        elif theme_index == 1:
            self.ui.btn_mute.setStyleSheet('background-color: rgb(232, 199,
236);')
        elif theme_index == 2:
            self.ui.btn_mute.setStyleSheet('background-color: #999999;')
        elif theme_index == 3:
            self.ui.btn_mute.setStyleSheet('background-color: #33cc33;')
```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

```

self.audio_output.setMuted(True)
self.ui.btn_mute.setFlat(True)

```

Її головна роль – це робота у режимі «ключа». Таких кнопок у проєкті три і у них однаковий алгоритм роботи. Спочатку, в залежності від теми головного вікна, змінюється колір кнопки, щоб користувач розумів чи включена\виключена дана функція зараз. Далі йде алгоритм, щоб кнопка працювала у режимі «ключа».

Наступна функція ставить відтворення на паузу.

```

def pause_audio(self):
    """Pause"""
    self.current_time = self.player.position()
    self.player.pause()

```

Для переключення аудіо вперед\назад з плейлисту, були створені дві функції зі схожим алгоритмом:

```

def next_song(self):
    """Switches the song further"""
    if self.player.playbackState() ==
QMediaPlayer.PlaybackState.PausedState:
        pass
    elif self.random_flag:
        self.current_song = random.choice(self.playlist)
        self.change_song()
        self.play_audio()
    else:
        if self.current_song:
            current_index = self.playlist.index(self.current_song)
            current_index += 1
            if current_index >= len(self.playlist):
                pass
            else:
                self.current_song = self.playlist[current_index]
                self.change_song()
                self.duration_of_song = self.player.duration()
                self.play_audio()

def previous_song(self):
    """Switches the song back"""
    if self.player.playbackState() ==
QMediaPlayer.PlaybackState.PausedState:
        pass
    else:
        if self.current_song:
            current_index = self.playlist.index(self.current_song)
            current_index -= 1
            if current_index < 0:
                pass
            else:
                self.current_song = self.playlist[current_index]
                self.change_song()
                self.duration_of_song = self.player.duration()
                self.play_audio()

```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

Єдині відмінності у роботі цих функцій – це зміна індексу(додавання\віднімання) відтворюваного аудіофайлу і перевірка на включення режиму «random» у функції next_song.

Далі розглянемо файл changing_elements.py. Він відповідає за зміну стану візуальних елементів головного вікна. Наприклад перша функція змінює довжину відтворювання відповідно до аудіофайлу.

```
def change_duration(self, duration):  
    self.ui.sldr_progress.setValue(0)  
    self.ui.sldr_progress.setRange(0, duration)
```

Наступна функція зміщує прапорець на слайдері відповідно до часу.

```
def song_progress_change(self, position):  
    self.current_time = position  
    self.player.setPosition(position)
```

Далі йде функція, яка відповідає за перемотку аудіофайлу.

```
def song_position_change(self, position):  
    self.ui.sldr_progress.setValue(position)
```

Функція, яка змінює гучність виглядає так:

```
def change_volume(self, position):  
    self.audio_output.setVolume(position / 100.0)
```

Для того, щоб відображалось ім'я відтворюваного файлу у головному вікні, було створено таку функцію:

```
def change_song(self):  
    self.ui.label.setText(self.current_song)
```

Для відображення часу і реалізації функцій «random» і «loop» була створена дана функція:

```
def change_time_label(self, position):  
    seconds = (position / 1000) % 60  
    minutes = (position / 60000) % 60  
    hours = (position / 2600000) % 24  
  
    time_label = QTime(hours, minutes, seconds)  
    self.ui.label_time.setText(time_label.toString())  
    if self.ui.sldr_progress.value() == self.player.duration():  
        if self.loop_flag is True:  
            self.play_audio()  
        else:  
            if self.random_flag is True:  
                self.current_song = random.choice(self.playlist)  
                self.change_song()  
                self.play_audio()  
            else:  
                self.next_song()
```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		39

Щоб змінити фото\обкладинку відповідно до плейлиста(якщо там є фото)

написано такий код:

```
def change_album_cover(self):
    self.img_album_cover = QPixmap('icons/album_cover_sample.png')
    self.ui.lbl_photo.setPixmap(self.img_album_cover)
```

Останні дві функції схожі за своїм принципом роботи – «ключ». Тут це відноситься до кнопок «random» і «loop».

```
def change_btn_loop(self, theme_index):
    if self.ui.btn_loop.isFlat():
        if theme_index == 0:
            self.ui.btn_loop.setStyleSheet('background-color: #b3d9ff;')
        elif theme_index == 1:
            self.ui.btn_loop.setStyleSheet('background-color: rgb(203, 236, 199);')
        elif theme_index == 2:
            self.ui.btn_loop.setStyleSheet('background-color: #cccccc;')
        elif theme_index == 3:
            self.ui.btn_loop.setStyleSheet('background-color: # 70db70;')
        self.loop_flag = False
        self.ui.btn_loop.setFlat(False)
    else:
        if theme_index == 0:
            self.ui.btn_loop.setStyleSheet('background-color: #0073e6;')
        elif theme_index == 1:
            self.ui.btn_loop.setStyleSheet('background-color: rgb(232, 199, 236);')
        elif theme_index == 2:
            self.ui.btn_loop.setStyleSheet('background-color: #999999;')
        elif theme_index == 3:
            self.ui.btn_loop.setStyleSheet('background-color: #33cc33;')
        self.loop_flag = True
        self.ui.btn_loop.setFlat(True)

def random_song(self, theme_index):
    if self.ui.btn_randow.isFlat():
        if theme_index == 0:
            self.ui.btn_randow.setStyleSheet('background-color: #b3d9ff;')
        elif theme_index == 1:
            self.ui.btn_randow.setStyleSheet('background-color: rgb(203, 236, 199);')
        elif theme_index == 2:
            self.ui.btn_randow.setStyleSheet('background-color: #cccccc;')
        elif theme_index == 3:
            self.ui.btn_randow.setStyleSheet('background-color: # 70db70;')
        self.ui.btn_randow.setFlat(False)
        self.random_flag = False
    else:
        if theme_index == 0:
            self.ui.btn_randow.setStyleSheet('background-color: #0073e6;')
        elif theme_index == 1:
            self.ui.btn_randow.setStyleSheet('background-color: rgb(232, 199, 236);')
        elif theme_index == 2:
            self.ui.btn_randow.setStyleSheet('background-color: #999999;')
        elif theme_index == 3:
            self.ui.btn_randow.setStyleSheet('background-color: #33cc33;')
```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

```

self.random_flag = True
self.ui.btn_random.setFlat(True)
self.ui.random_flag = True

```

Далі йде файл `playlist_controls.py`. Відповідає за роботу з плейлистами. Перша функція дозволяє відтворити аудіо шляхом двійного кліку по назві файлу, яке відображається у таблиці головного вікна.

```

def play_audio_from_playlist(self):
    """Start playing audio from playlist after double click"""
    if self.player.playbackState() ==
QMediaPlayer.PlaybackState.PlayingState:
        self.player.stop()
        self.current_song = self.get_current_item_in_playlist()
        audio_path = self.path + '/' + self.current_song
        self.player.setSource(QUrl.fromLocalFile(audio_path))
        self.change_song()
        self.duration_of_song = self.player.duration()
        self.player.play()

```

У ній задіяна ще одна додаткова функція, яка повертає назву файлу у таблиці.

```

def get_current_item_in_playlist(self):
    """Return current item in List View"""
    return self.ui.lst_playlist.currentItem().text()

```

Для створення\додавання плейлисту було зроблено таку функцію:

```

def create_playlist(self):
    """Creating playlist and adding it to ListView"""
    self.path = QFileDialog.getExistingDirectory(self, 'Select Directory')
    temp_playlist = []
    if self.path:
        temp_playlist = os.listdir(self.path)

    if temp_playlist:
        self.playlist.clear()
        for song in temp_playlist:
            if '.mp3' in song or '.wav' in song or '.ogg' in song:
                self.playlist.append(song)

        if len(self.playlist) > 0:
            self.current_song = self.playlist[0]
            self.ui.lst_playlist.clear()
            self.ui.lst_playlist.addItem(self.playlist)
            self.save_playlists(self.path)
        else:
            self.show_error_message('This directory does\'nt have a audio
files')

        for file in temp_playlist:
            if '.jpg' in file or '.png' in file:
                self.img_album_cover = QPixmap(self.path + '\\ ' + file)
                self.ui.lbl_photo.setPixmap(self.img_album_cover)
                break
            else:
                self.change_album_cover()

    else:

```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

```
self.show_error_message('This directory does\'nt have a audio files')
```

Вона відкриває діалогове вікно, де користувач обирає папку з аудіофайлами. Після цього файли фільтруються по формату аудіофайлів і додаються у плейлист.

Якщо у папці є фото, то воно також додається до головного вікна. Якщо ж папка не містить аудіофайлів, то виводиться відповідне вікно, яке про це говорить.

```
def show_error_message(self, message):  
    """Shows error messages"""  
    show_error_mess.show_error_message(self, message)
```

Саме ця функція відповідає за повідомлення користувача у разі помилки.

Її код розташований у файлі show_error_mess.py:

```
from PySide6.QtWidgets import QMessageBox
```

```
def show_error_message(self, message):  
    """Shows error messages"""  
    msg_box = QMessageBox()  
    msg_box.setIcon(QMessageBox.Icon.Warning)  
    msg_box.setText(message)  
    msg_box.setWindowTitle('Error!')  
    msg_box.setStandardButtons(QMessageBox.StandardButton.Ok)  
    msg_box.exec()
```

Далі оглянемо файл для запису\зчитування плейлистів з файлу.

```
def save_playlists(self, path_dir):  
    """Saving all existing playlists"""  
    if os.path.exists('playlists.txt'):  
        with open('playlists.txt', 'r') as file:  
            paths = file.readlines()  
            if len(paths) > 0:  
                indicator = False  
                for existing_path in paths:  
                    if self.check_playlist_when_add(existing_path,  
path_dir):  
                        indicator = True  
                        break  
                if indicator is not True:  
                    with open('playlists.txt', 'a') as file_for_add:  
                        file_for_add.write(path_dir + '\n')  
            else:  
                with open('playlists.txt', 'a') as file_for_add: # Add  
first row of playlist  
                    file_for_add.write(path_dir + '\n')  
            else:  
                with open('playlists.txt', 'w') as file: # Create file and add  
playlist  
                    file.write(path_dir + '\n')
```

```
def load_playlists(self, path):
```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

```

        """Loading last saving playlist"""
        if os.path.exists('playlists.txt'):
            with open('playlists.txt', 'r') as file:
                playlist_names = file.readlines()
                if playlist_names:
                    last_playlist = playlist_names[-1]
                    songs = os.listdir(last_playlist[:-1])
                    self.path = last_playlist[:-1]
                    if songs:
                        for song in songs:
                            if '.mp3' in song or '.wav' in song or '.ogg' in
song:
                                self.playlist.append(song)
                            elif '.jpg' in song or '.png' in song:
                                self.img_album_cover = QPixmap(self.path + '\\\'
+ song)

self.ui.lbl_photo.setPixmap(self.img_album_cover)
                else:
                    self.change_album_cover()

                self.ui.lst_playlist.clear()
                self.ui.lst_playlist.addItem(self.playlist)

```

Функція `load_playlists` завантажує останній доданий користувачем плейлист при відкритті програми. Протилежна їй функція – `save_playlists`. Вона зберігає кожен новий доданий плейлист у файл, коли користувач обрав папку, яка задовольняє умову знаходження у ній аудіофайлів.

Розглянемо файл, який змінює теми головного вікна – `themes.py`. Він містить одну функцію.

```

def change_theme(self, theme_index):
    if theme_index == 1:
        # Ставиться відповідна тема

```

Файл великий, бо там записані стилі. Відповідно для чистоти інформації документа у цьому розділі не буде наведено повний код файлу. Алгоритм роботи простий: отримуємо індекс теми і відповідно до нього змінюємо стиль вікна.

Для роботи з сервісом YouTube було створено файл `work_with_youtube.py`. Розглянемо його функції. Для ініціалізації і виведення на екран додаткового вікна написано даний код:

```

def create_playlist_from_youtube(self):
    self.new_window = QDialog()
    self.ui_window = Ui_Dialog()
    self.ui_window.setupUi(self.new_window)

self.ui_window.pushButton.clicked.connect(self.change_path_for_youtube_music
)
    self.ui_window.btn_enter.clicked.connect(self.download_from_youtube)
    self.new_window.show()

```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

Вона викликається, коли користувач натиснув кнопку «file_from_youtube».

У додатковому вікні можна вибрати куди зберегти завантажені файли. Для реалізації цієї дії створена дана функція:

```
def change_path_for_youtube_music(self):
    path_to_audio_from_youtube =
    QFileDialog.getExistingDirectory(self.new_window, 'Select Directory')
    if path_to_audio_from_youtube:
        self.path_to_audio_from_youtube = path_to_audio_from_youtube
        self.ui_window.label_2.setText(f'Save to:
{self.path_to_audio_from_youtube}')
```

Для самого завантаження використовується така функція:

```
def download_from_youtube(self):
    url = self.ui_window.ln_url.text()
    if url:
        mode = self.ui_window.comboBox.currentText()
        asyncio.run(async_m.start_process(url, mode,
self.path_to_audio_from_youtube))
        self.ui_window.label.setText('Completed!')
```

Вона викликається, коли користувач вже вставив посилання на відео\плейлист у відповідне поле і натиснув кнопку «Enter». В ній використовується ще один файл, спеціально створений для завантаження і конвертації файлів. З рештою це буде останній файл, який буде розглядатися.

Назва файлу – async_for_youtube.py. Містить даний код:

```
async def download_data(url, mode, path):
    if mode == 'Video':
        try:
            yt = YouTube(url)
            yt.streams.filter(only_audio=True).first.download(path)
        except pytube.exceptions.RegexMatchError:
            print('Exception')
    elif mode == 'Playlist':
        try:
            p = Playlist(url)
            for video in p.videos:
                video.streams.filter(only_audio=True).first().download(path)
        except:
            pass

async def convert_data(url, path):
    print(path)
    if url:
        for file in os.listdir(path):
            if re.search('mp4', file):
                mp4 = os.path.join(path, file)
                mp3 = os.path.join(path, os.path.splitext(file)[0] + '.mp3')
                new_file = mp.AudioFileClip(mp4)
                new_file.write_audiofile(mp3, codec='libmp3lame')
                os.remove(mp4)
```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

```

async def start_process(url, mode, path):
    task_download = asyncio.create_task(download_data(url, mode, path))
    await task_download
    task_convert = asyncio.create_task(convert_data(url, path))
    await task_convert

```

Для швидшого завантаження і подальшого конвертування було використано модуль асинціо.

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

РОЗДІЛ 3 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Огляд основних можливостей

Перше, що бачить користувач при відкритті зображено на рисунку.

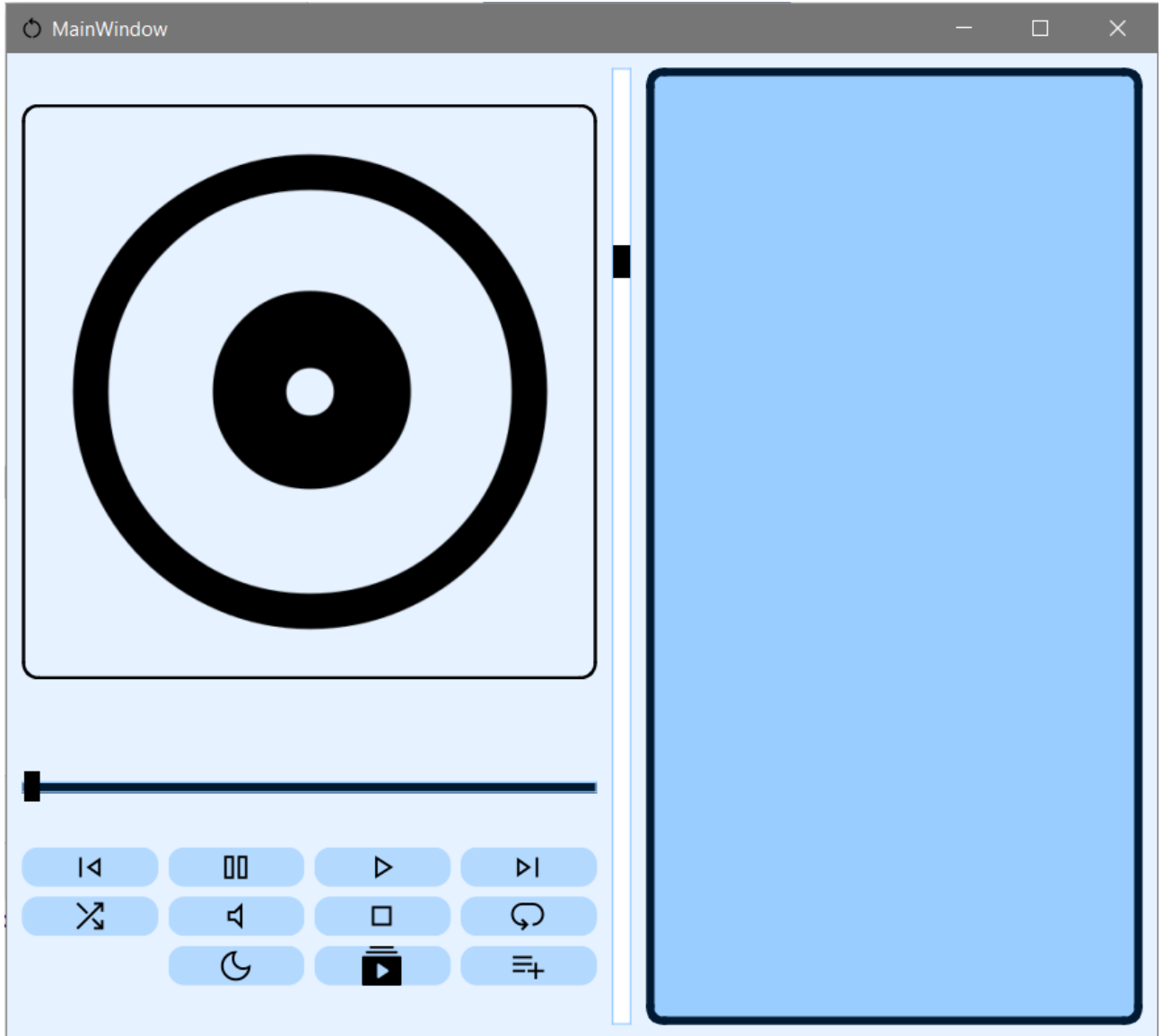


Рисунок 3.1 — Головне вікно програми

Внизу зліва розташовані кнопки управління відтворенням. Оглянемо їх в порядку справа наліво, звернувши увагу на напрямки стрілок.

Відтворити попередній по списку аудіофайл.

- Пауза.
- Відтворити.
- Відтворити наступний по списку аудіофайл.
- Режим «Random».

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

- Режим «Mute».
- Зупинити.
- Режим «Loop».
- Змінити тему.
- Створити плейлист за допомогою сервісу YouTube.
- Відкрити\створити плейлист.

Над кнопками управління розташований слайдер, який показує поточний прогрес відтворення. Далі, над ним відображається тестове зображення.

По середині розташований слайдер, який змінює гучність.

Справа знаходиться плейлист у вигляді назв відкритих аудіофайлів.

Спробуємо відкрити плейлист з файлами типу «.mp3» і відтворити аудіофайл.

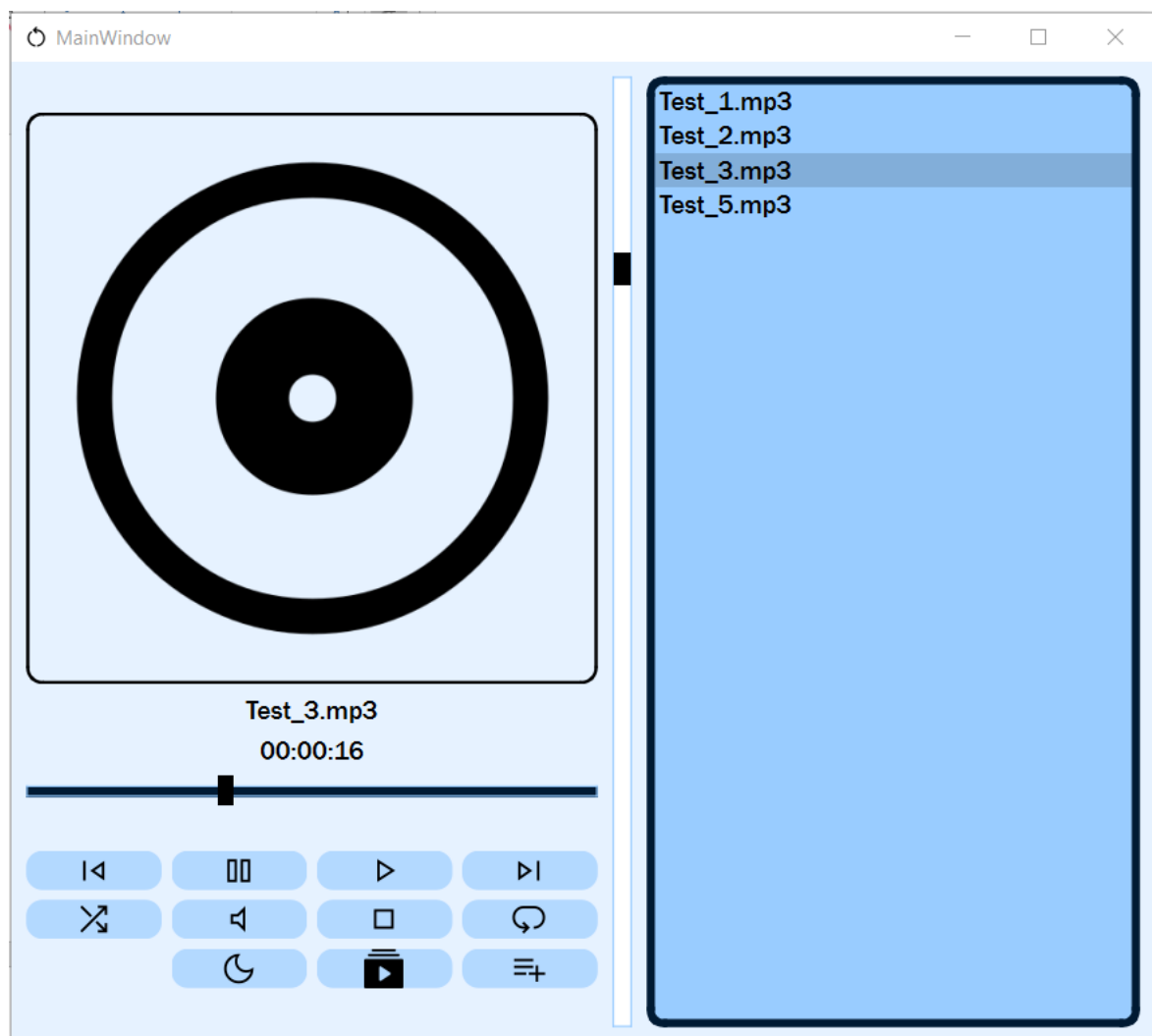


Рисунок 3.2 — Відтворення файлів типу «.mp3»

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		47

Активация доступных режимів виглядає так:

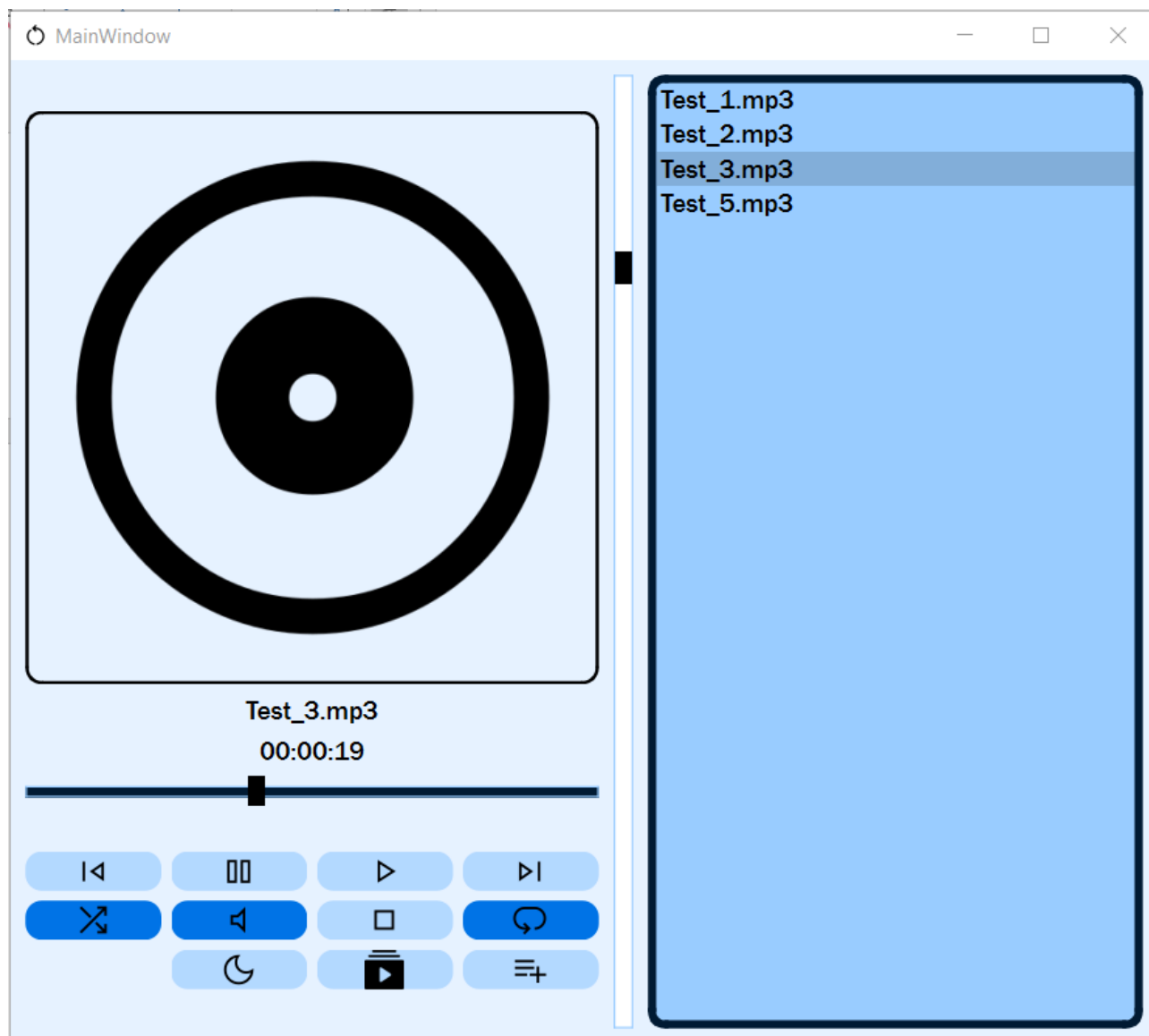
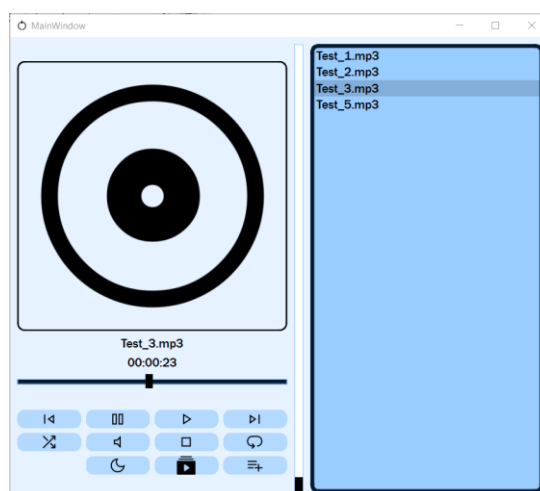


Рисунок 3.3 — Вигляд активованих кнопок, які відповідають за режими

Кнопки переключення аудіофайлів працюють справно.

Подивимося на доступні теми для головного вікна.



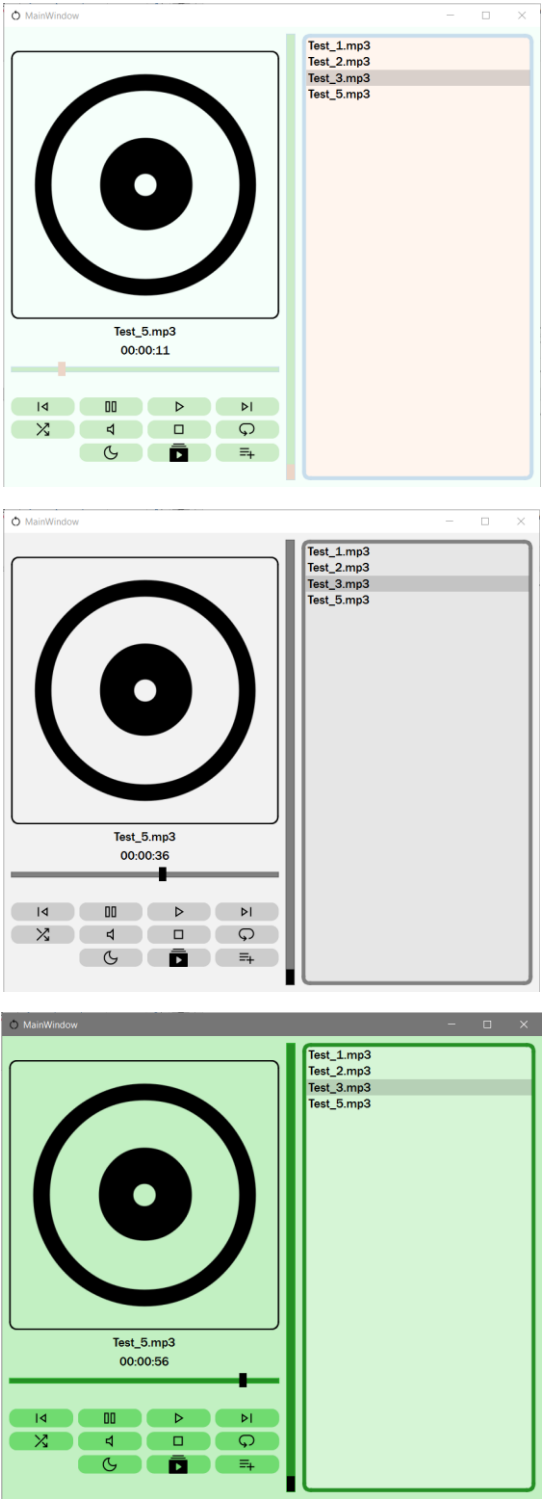


Рисунок 3.4 — Вигляд різних тем для головного вікна

Спробуємо відкрити папку з файлами типу «» і фото всередині.

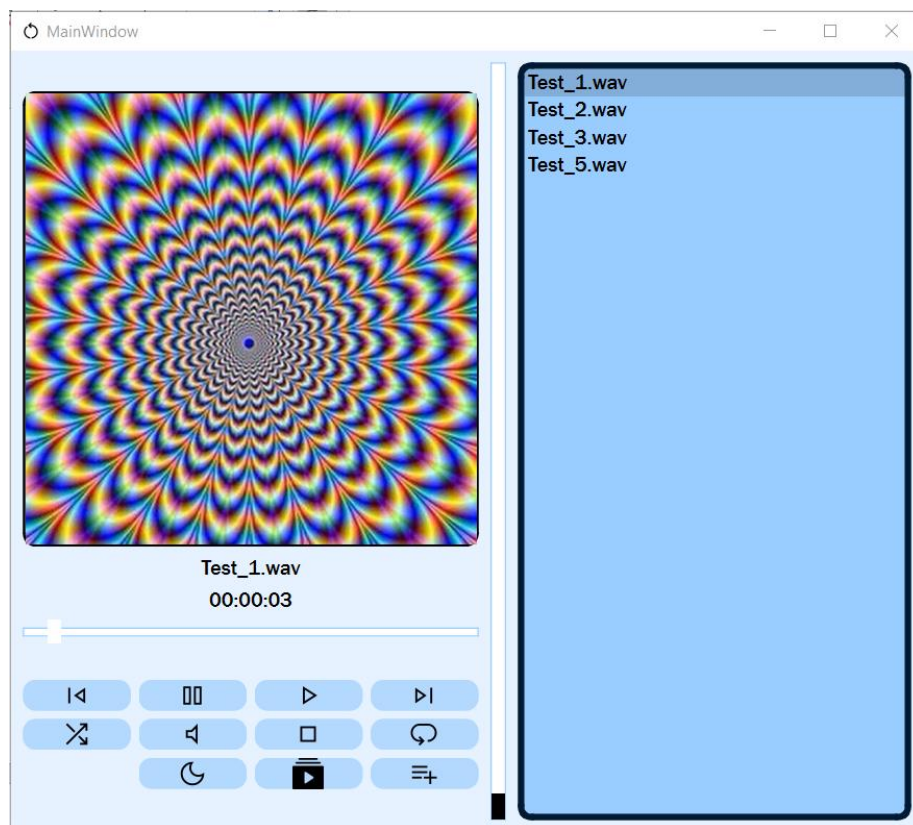


Рисунок 3.5 — Вигляд головного вікна з фото\обкладинки

Користувач може вільно маніпулювати відтворенням і насолоджуватися прослуховуванням аудіо.

3.2 Взаємодія з сервісом YouTube

Користувач натискає відповідну кнопку і з'являється додаткове вікно.



Рисунок 3.6 — Вигляд додаткового вікна

Після цього оберемо папку, куди будуть зберігатися завантажені файли.

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		50

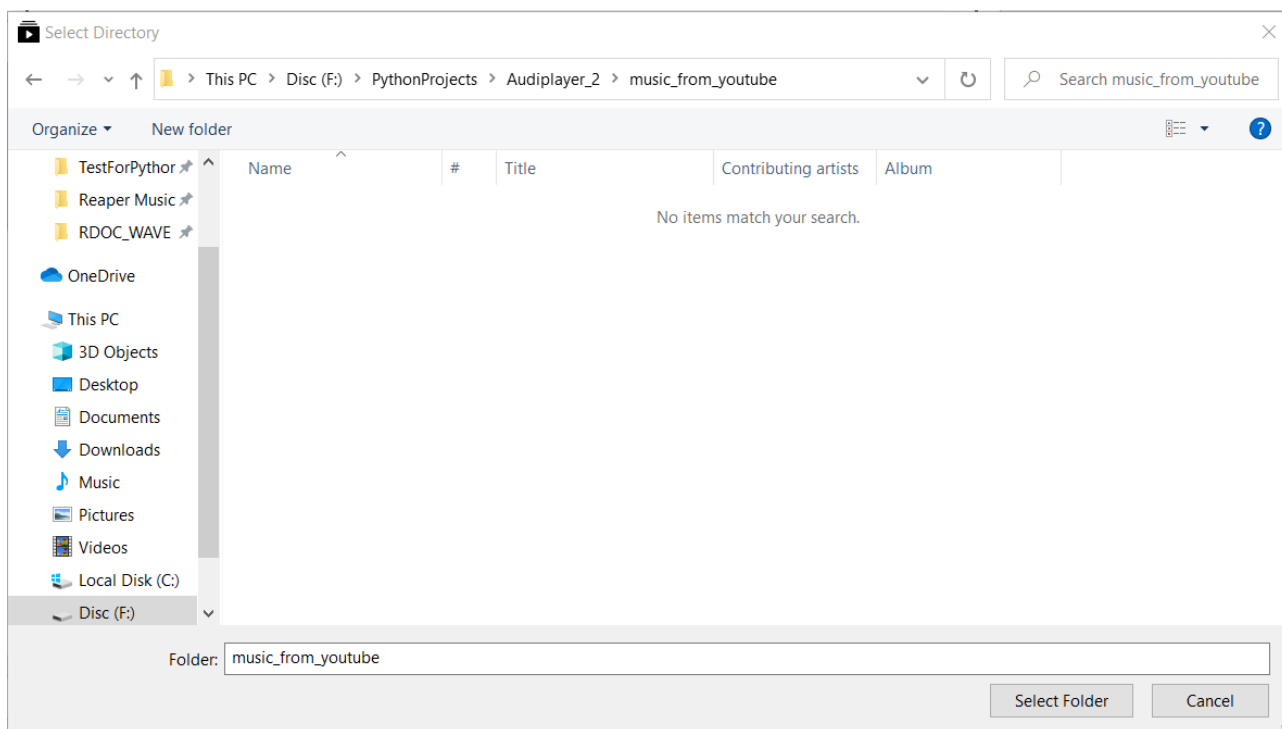


Рисунок 3.7 — Вибір папки для збереження

Тепер додаткове вікно виглядає так:

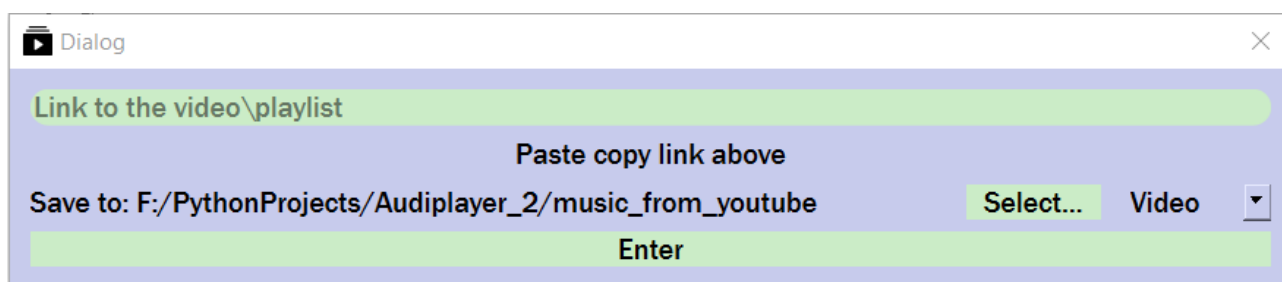


Рисунок 3.8 - Вигляд вікна після додавання папки для збереження

Користувачу потрібно вибрати тип завантажуваних даних: відео, або плейлист. Після цього копіюємо посилання. Для прикладу візьмемо відео з каналу розробника.

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

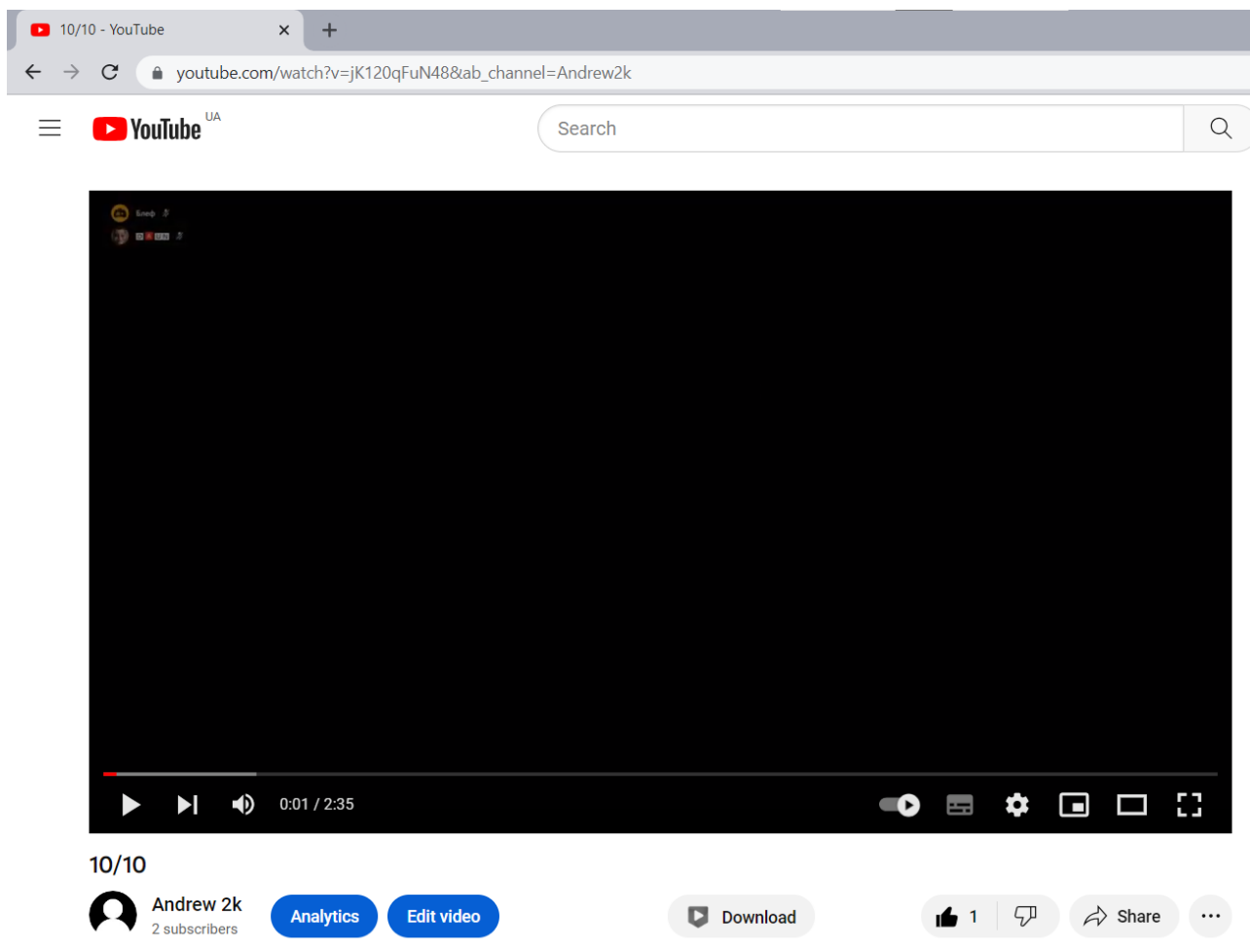


Рисунок 3.9 — Відео для прикладу

Так виглядає правильно заповнена форма для завантаження:

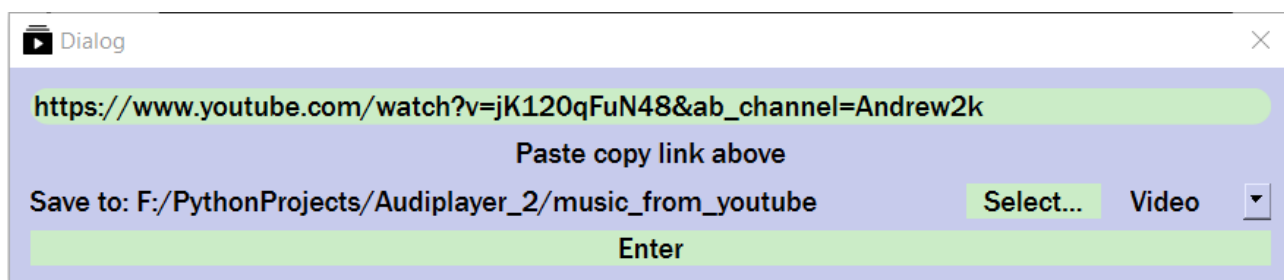


Рисунок 3.10 — Вигляд вікна до завантаження

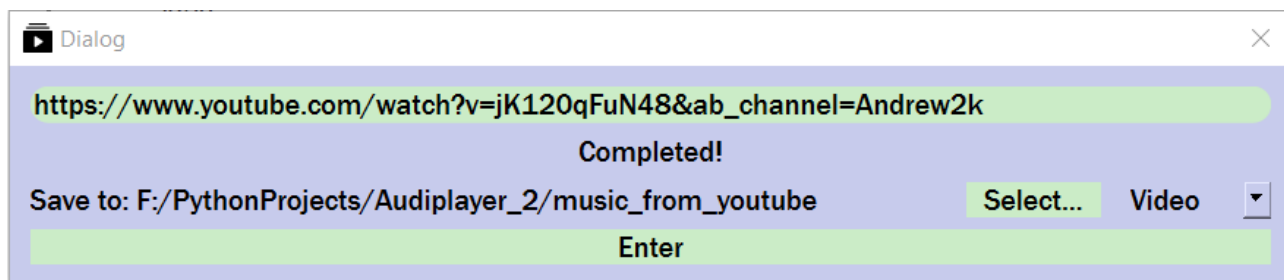


Рисунок 3.11 — Вигляд вікна після завантаження

Тепер користувач може побачити файл у папці.

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				52
Змн.	Арк.	№ докум.	Підпис	Дата		

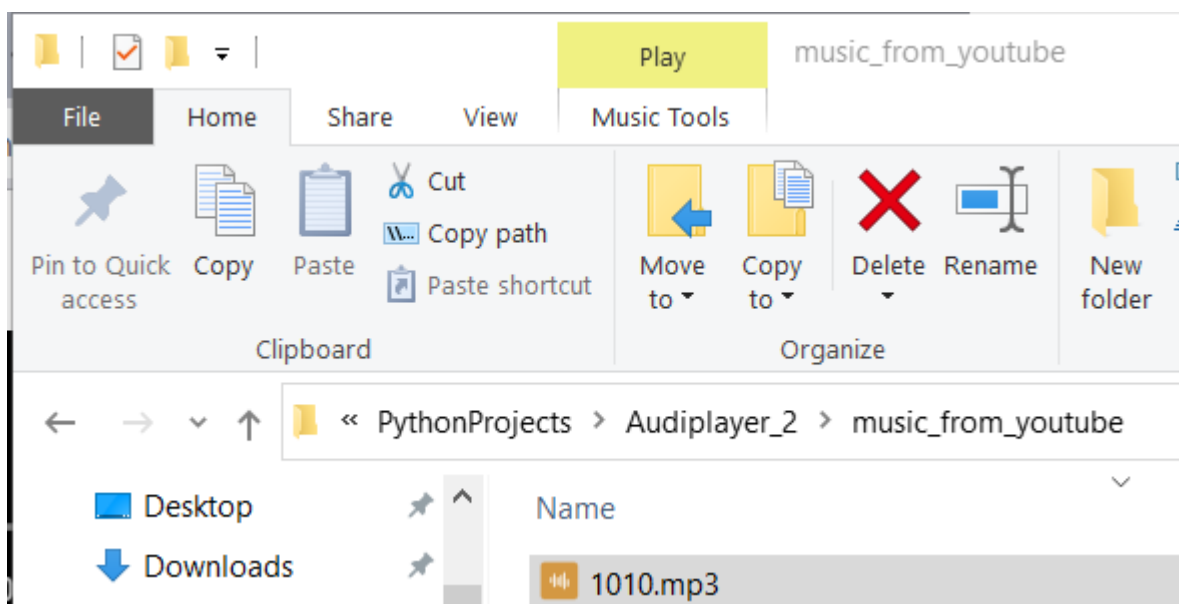


Рисунок 3.12 — Завантажений файл у папці

Відкриємо його і відтворимо.

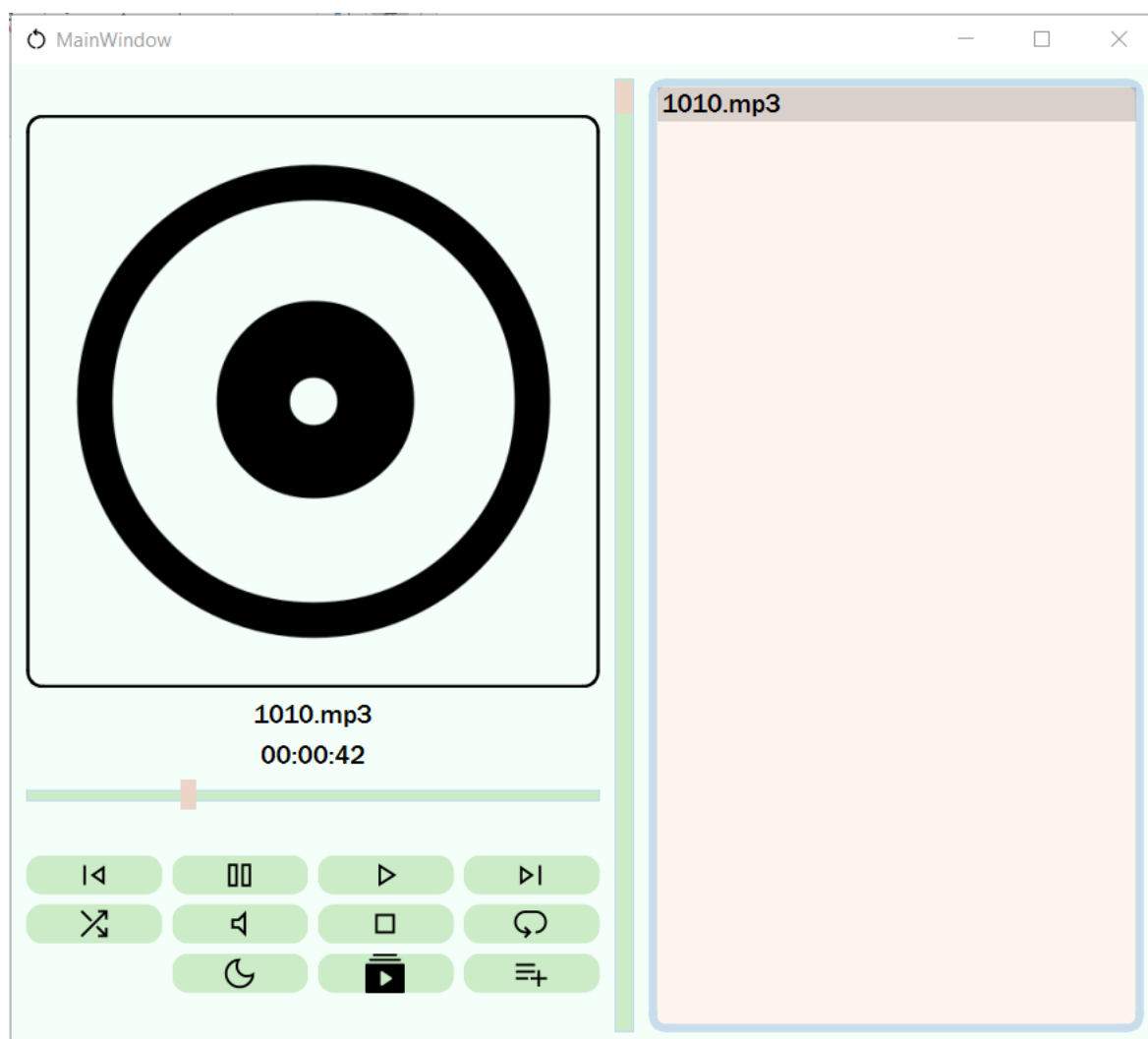


Рисунок 3.13 — Відтворювання завантаженого файлу

Тепер спробуємо завантажити плейлист.

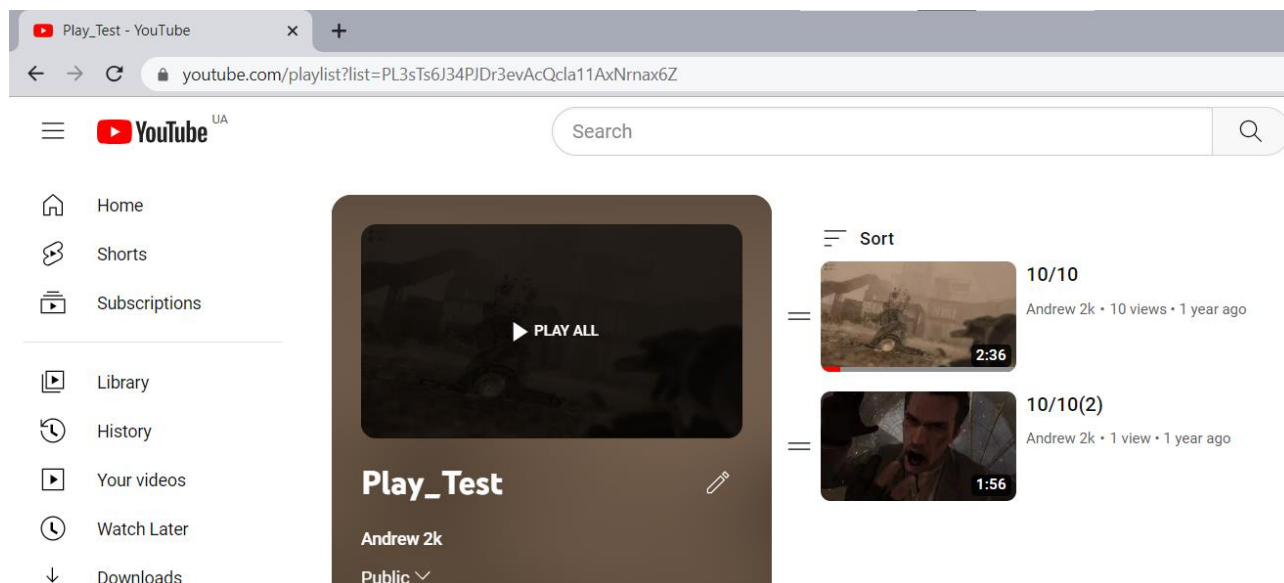


Рисунок 3.14 - Плейлист для прикладу



Рисунок 3.15. — Правильно заповнена форма для завантаження плейлисту

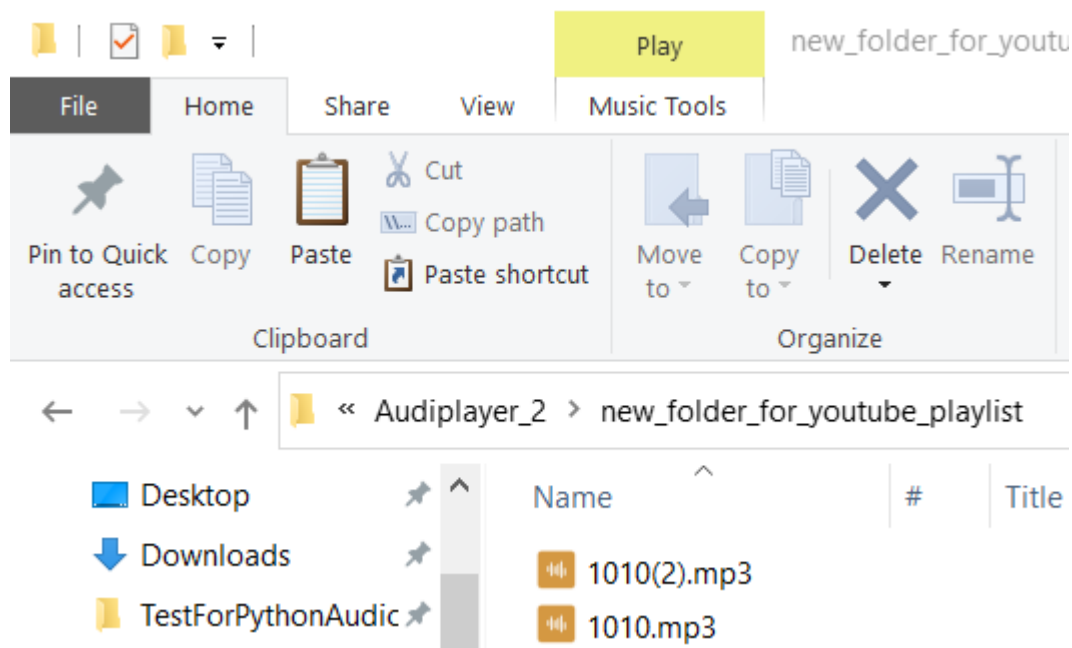


Рисунок 3.16 — Завантажуваний плейлист у новій папці

Спробуємо відтворити їх.

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				54
Змн.	Арк.	№ докум.	Підпис	Дата		

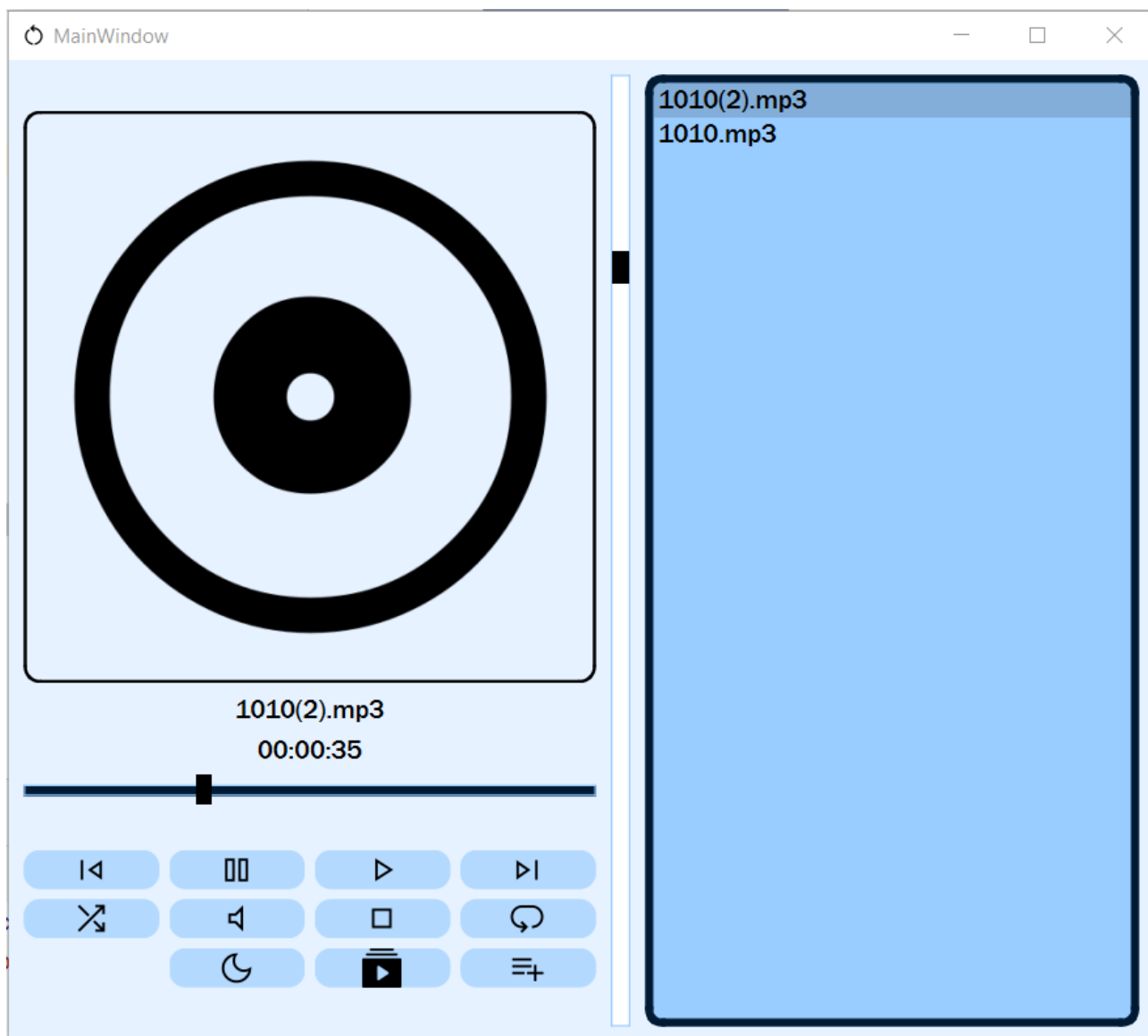


Рисунок 3.17 — Відтворення завантаженого плейлиста

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

ВИСНОВКИ

В першому розділі було визначено особливості роботи плеєрів. Було розглянуто історію і розвиток аудіофайлів, їхнє призначення і вплив на розвиток комп'ютерних технологій, а також існуючі плеєри зазначаючи їх переваги та недоліки. Детально було розглянуто програвачі WinAmp, AIMP, Spotify, оглянуто їхні можливості. Далі розглядалося питання вибору програмного забезпечення для розробки програми.

В другому розділі було розглянуто модулі для мови програмування Python, які обралися для розробки. Визначено затребувану кількість функціоналу. Розроблено алгоритми роботи і написано код програми.

В третьому розділі було оглянуто і протестовано створене програмне забезпечення, яке і задовольнило всі вимоги.

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Прискорений курс Python. Ерік Маттес – Львів: Вид. Старого Лева, 2022. – с. (Укр. мов.)
2. Офіційна сторінка Python [Електронний ресурс] – <https://www.python.org/>
3. Офіційна сторінка компанії JetBrains [Електронний ресурс] – <https://www.jetbrains.com/>
4. Офіційний сайт пошуку модулів для Python [Електронний ресурс] – <https://pypi.org/>
5. Офіційний сайт документації модулю PySide6 [Електронний ресурс] – https://wiki.qt.io/Qt_for_Python
6. Документація модулю pytube [Електронний ресурс] – <https://pytube.io/en/latest/>
7. Документація по сервісам Microsoft [Електронний ресурс] – <https://learn.microsoft.com/uk-ua/docs/>
8. Python 3 Module of the Week [Електронний ресурс] – <https://pymotw.com>
9. Додаток до навчальної книги [Електронний ресурс] – <https://nostarch.com/python-crash-course-3rd-edition>
10. Безкоштовні і не ліцензовані фото від Google [Електронний ресурс] – <https://fonts.google.com/icons>
11. Додатковий ресурс з фото [Електронний ресурс] – <https://www.flaticon.com/>
12. Офіційна документація по Python [Електронний ресурс] – <https://docs.python.org/uk/3/>
13. Курс від Державного університету «Житомирська політехніка» по мові програмування Python [Електронний ресурс] – <https://learn.ztu.edu.ua/course/view.php?id=4096>

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		57

ДОДАТКИ

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

Файл main.py

```

import sys

from PySide6.QtWidgets import QApplication, QMainWindow

from PySide6.QtMultimedia import QMediaPlayer, QAudioOutput
from PySide6.QtGui import QPixmap

from design import Ui_MainWindow

import controls, playlist_controlls, work_with_file, work_with_youtube,
changing_elements, themes, show_error_mess

class Audioplayer(QMainWindow):
    def __init__(self):
        super(Audioplayer, self).__init__()
        self.ui = Ui_MainWindow() # Creating Main window
        self.ui.setupUi(self)
        self.path = '' # Variable that save one path to playlist
        self.playlist = [] # Variable that save songs in playlist
        self.img_album_cover = QPixmap('icons/album_cover_sample.png')
        self.change_album_cover()
        self.load_playlists('playlists.txt') # Loading latest playlist from
file
        self.ui.label.setText('')
        self.ui.label_time.setText('')
        self.duration_of_song = 0
        self.loop_flag = False
        self.path_to_audio_from_youtube = 'music_from_youtube'
        self.random_flag = False
        self.theme_index = 0

        # Audio
        self.player = QMediaPlayer() # Initializing audio source
        self.audio_output = QAudioOutput()
        self.player.setAudioOutput(self.audio_output)
        self.current_time = 0 # Variable that save current time in song
        self.current_song = '' # Variable that save name of song that
playing now
        self.audio_output.setVolume(self.ui.slidr_volume.value())

        # Events
        self.ui.btn_add_playlist.clicked.connect(self.create_playlist)

        self.ui.btn_add_playlist_from_youtube.clicked.connect(self.create_playlist_fr
om_youtube)
        self.ui.btn_play.clicked.connect(self.play_audio)
        self.ui.btn_stop.clicked.connect(self.stop_audio)
        self.ui.btn_pause.clicked.connect(self.pause_audio)

        self.ui.lst_playlist.itemDoubleClicked.connect(self.play_audio_from_playlist
)
        self.ui.slidr_progress.sliderMoved.connect(self.song_progress_change)
        self.ui.slidr_progress.valueChanged.connect(self.change_time_label)
        self.ui.slidr_volume.sliderMoved.connect(self.change_volume)
        self.ui.btn_next.clicked.connect(self.next_song)

```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

```

self.ui.btn_previous.clicked.connect(self.previous_song)
self.ui.btn_loop.clicked.connect(self.change_btn_loop)
self.ui.btn_mute.clicked.connect(self.mute)
self.ui.btn_random.clicked.connect(self.random)
self.ui.btn_theme.clicked.connect(self.change_theme)

# Events for audio source
self.player.durationChanged.connect(self.change_duration)
self.player.positionChanged.connect(self.song_position_change)

def play_audio(self):
    controls.play_audio(self)

def play_audio_from_playlist(self):
    playlist_controlls.play_audio_from_playlist(self)

def get_current_item_in_playlist(self):
    return self.ui.lst_playlist.currentItem().text()

def stop_audio(self):
    controls.stop_audio(self)

def mute(self):
    controls.mute(self, self.theme_index)

def pause_audio(self):
    controls.pause_audio(self)

def next_song(self):
    controls.next_song(self)

def previous_song(self):
    controls.previous_song(self)

def create_playlist(self):
    playlist_controlls.create_playlist(self)

def check_playlist_when_add(self, input_path, checked_path):
    if input_path == checked_path + '\n':
        return True
    else:
        return False

def save_playlists(self, path_dir):
    work_with_file.save_playlists(self, self.path)

def load_playlists(self, path):
    work_with_file.load_playlists(self, path)

def create_playlist_from_youtube(self):
    work_with_youtube.create_playlist_from_youtube(self)

def download_from_youtube(self):
    work_with_youtube.download_from_youtube(self)

def change_duration(self, duration):
    changing_elements.change_duration(self, duration)

def song_progress_change(self, position):

```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		60

```

        changing_elements.song_progress_change(self, position)

def song_position_change(self, position):
    changing_elements.song_position_change(self, position)

def change_volume(self, position):
    changing_elements.change_volume(self, position)

def change_song(self):
    changing_elements.change_song(self)

def change_time_label(self, position):
    changing_elements.change_time_label(self, position)

def change_btn_loop(self):
    changing_elements.change_btn_loop(self, self.theme_index)

def change_album_cover(self):
    changing_elements.change_album_cover(self)

def random(self):
    changing_elements.random_song(self, self.theme_index)

def change_path_for_youtube_music(self):
    work_with_youtube.change_path_for_youtube_music(self)

def change_theme(self):
    if self.theme_index + 1 > 3:
        self.theme_index = 0
    else:
        self.theme_index += 1
    themes.change_theme(self, self.theme_index)

def show_error_message(self, message):
    """Shows error messages"""
    show_error_mess.show_error_message(self, message)

if __name__ == '__main__':
    app = QApplication(sys.argv)
    window = Audioplayer()
    window.show()

    sys.exit(app.exec())

```

Файл controls.py

```

import random

from PySide6.QtMultimedia import QMediaPlayer
from PySide6.QtCore import QUrl

import show_error_mess

def play_audio(self):
    """Start playing audio when button was pushed"""
    if self.player.playbackState() ==
QMediaPlayer.PlaybackState.PausedState:
        self.player.setPosition(self.current_time)

```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

```

        self.player.play()
    else:
        audio_path = self.path + '/' + self.current_song
        self.player.stop()
        self.player.setSource(QUrl.fromLocalFile(audio_path))
        self.player.setPosition(0)
        self.player.play()

def stop_audio(self):
    self.player.stop()

def mute(self, theme_index):
    """Muting song"""
    if self.ui.btn_mute.isFlat():
        if theme_index == 0:
            self.ui.btn_mute.setStyleSheet('background-color: #b3d9ff;')
        elif theme_index == 1:
            self.ui.btn_mute.setStyleSheet('background-color: rgb(203, 236,
199);')
        elif theme_index == 2:
            self.ui.btn_mute.setStyleSheet('background-color: #cccccc;')
        elif theme_index == 3:
            self.ui.btn_mute.setStyleSheet('background-color: # 70db70;')
        self.audio_output.setMuted(False)
        self.ui.btn_mute.setFlat(False)
    else:
        if theme_index == 0:
            self.ui.btn_mute.setStyleSheet('background-color: #0073e6;')
        elif theme_index == 1:
            self.ui.btn_mute.setStyleSheet('background-color: rgb(232, 199,
236);')
        elif theme_index == 2:
            self.ui.btn_mute.setStyleSheet('background-color: #999999;')
        elif theme_index == 3:
            self.ui.btn_mute.setStyleSheet('background-color: #33cc33;')
        self.audio_output.setMuted(True)
        self.ui.btn_mute.setFlat(True)

def pause_audio(self):
    """Pause"""
    self.current_time = self.player.position()
    self.player.pause()

def next_song(self):
    """Switches the song further"""
    if self.player.playbackState() ==
QMediaPlayer.PlaybackState.PausedState:
        pass
    elif self.random_flag:
        self.current_song = random.choice(self.playlist)
        self.change_song()
        self.play_audio()
    else:
        if self.current_song:
            current_index = self.playlist.index(self.current_song)
            current_index += 1

```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

```

        if current_index >= len(self.playlist):
            pass
        else:
            self.current_song = self.playlist[current_index]
            self.change_song()
            self.duration_of_song = self.player.duration()
            self.play_audio()

def previous_song(self):
    """Switches the song back"""
    if self.player.playbackState() ==
QMediaPlayer.PlaybackState.PausedState:
        pass
    else:
        if self.current_song:
            current_index = self.playlist.index(self.current_song)
            current_index -= 1
            if current_index < 0:
                pass
            else:
                self.current_song = self.playlist[current_index]
                self.change_song()
                self.duration_of_song = self.player.duration()
                self.play_audio()

```

Файл changing_elements.py

```

from PySide6.QtCore import QTime
from PySide6.QtGui import QPixmap
import random

def change_duration(self, duration):
    self.ui.sldr_progress.setValue(0)
    self.ui.sldr_progress.setRange(0, duration)

def song_progress_change(self, position):
    self.current_time = position
    self.player.setPosition(position)

def song_position_change(self, position):
    self.ui.sldr_progress.setValue(position)

def change_volume(self, position):
    self.audio_output.setVolume(position / 100.0)

def change_song(self):
    self.ui.label.setText(self.current_song)

def change_time_label(self, position):
    seconds = (position / 1000) % 60
    minutes = (position / 60000) % 60
    hours = (position / 2600000) % 24

```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

```

time_label = QTime(hours, minutes, seconds)
self.ui.label_time.setText(time_label.toString())
if self.ui.sldr_progress.value() == self.player.duration():
    if self.loop_flag is True:
        self.play_audio()
    else:
        if self.random_flag is True:
            self.current_song = random.choice(self.playlist)
            self.change_song()
            self.play_audio()
        else:
            self.next_song()

def change_btn_loop(self, theme_index):
    if self.ui.btn_loop.isFlat():
        if theme_index == 0:
            self.ui.btn_loop.setStyleSheet('background-color: #b3d9ff;')
        elif theme_index == 1:
            self.ui.btn_loop.setStyleSheet('background-color: rgb(203, 236,
199);')
        elif theme_index == 2:
            self.ui.btn_loop.setStyleSheet('background-color: #cccccc;')
        elif theme_index == 3:
            self.ui.btn_loop.setStyleSheet('background-color: # 70db70;')
        self.loop_flag = False
        self.ui.btn_loop.setFlat(False)
    else:
        if theme_index == 0:
            self.ui.btn_loop.setStyleSheet('background-color: #0073e6;')
        elif theme_index == 1:
            self.ui.btn_loop.setStyleSheet('background-color: rgb(232, 199,
236);')
        elif theme_index == 2:
            self.ui.btn_loop.setStyleSheet('background-color: #999999;')
        elif theme_index == 3:
            self.ui.btn_loop.setStyleSheet('background-color: #33cc33;')
        self.loop_flag = True
        self.ui.btn_loop.setFlat(True)

def random_song(self, theme_index):
    if self.ui.btn_randow.isFlat():
        if theme_index == 0:
            self.ui.btn_randow.setStyleSheet('background-color: #b3d9ff;')
        elif theme_index == 1:
            self.ui.btn_randow.setStyleSheet('background-color: rgb(203,
236, 199);')
        elif theme_index == 2:
            self.ui.btn_randow.setStyleSheet('background-color: #cccccc;')
        elif theme_index == 3:
            self.ui.btn_randow.setStyleSheet('background-color: # 70db70;')
        self.ui.btn_randow.setFlat(False)
        self.random_flag = False
    else:
        if theme_index == 0:
            self.ui.btn_randow.setStyleSheet('background-color: #0073e6;')
        elif theme_index == 1:
            self.ui.btn_randow.setStyleSheet('background-color: rgb(232,
199, 236);')
        elif theme_index == 2:

```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

```

        self.ui.btn_random.setStyleSheet('background-color: #999999;')
    elif theme_index == 3:
        self.ui.btn_random.setStyleSheet('background-color: #33cc33;')
    self.random_flag = True
    self.ui.btn_random.setFlat(True)
    self.ui.random_flag = True

def change_album_cover(self):
    self.img_album_cover = QPixmap('icons/album_cover_sample.png')
    self.ui.lbl_photo.setPixmap(self.img_album_cover)

```

Файл playlists_control.py

```

import os

from PySide6.QtWidgets import QDialog
from PySide6.QtMultimedia import QMediaPlayer
from PySide6.QtCore import QUrl
from PySide6.QtGui import QPixmap

import show_error_mess

def play_audio_from_playlist(self):
    """Start playing audio from playlist after double click"""
    if self.player.playbackState() ==
QMediaPlayer.PlaybackState.PlayingState:
        self.player.stop()
        self.current_song = self.get_current_item_in_playlist()
        audio_path = self.path + '/' + self.current_song
        self.player.setSource(QUrl.fromLocalFile(audio_path))
        self.change_song()
        self.duration_of_song = self.player.duration()
        self.player.play()

def get_current_item_in_playlist(self):
    """Return current item in List View"""
    return self.ui.lst_playlist.currentItem().text()

def create_playlist(self):
    """Creating playlist and adding it to ListView"""
    self.path = QDialog.getExistingDirectory(self, 'Select Directory')
    temp_playlist = []
    if self.path:
        temp_playlist = os.listdir(self.path)

    if temp_playlist:
        self.playlist.clear()
        for song in temp_playlist:
            if '.mp3' in song or '.wav' in song or '.ogg' in song:
                self.playlist.append(song)

        if len(self.playlist) > 0:
            self.current_song = self.playlist[0]
            self.ui.lst_playlist.clear()
            self.ui.lst_playlist.addItem(self.playlist)
            self.save_playlists(self.path)
        else:

```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		65

```

        self.show_error_message('This directory does\'nt have a audio
files')

    for file in temp_playlist:
        if '.jpg' in file or '.png' in file:
            self.img_album_cover = QPixmap(self.path + '\\\' + file)
            self.ui.lbl_photo.setPixmap(self.img_album_cover)
            break
        else:
            self.change_album_cover()

    else:
        self.show_error_message('This directory does\'nt have a audio
files')

def show_error_message(self, message):
    """Shows error messages"""
    show_error_mess.show_error_message(self, message)

```

Файл show_error_mess.py

```
from PySide6.QtWidgets import QMessageBox
```

```

def show_error_message(self, message):
    """Shows error messages"""
    msg_box = QMessageBox()
    msg_box.setIcon(QMessageBox.Icon.Warning)
    msg_box.setText(message)
    msg_box.setWindowTitle('Error!')
    msg_box.setStandardButtons(QMessageBox.StandardButton.Ok)
    msg_box.exec()

```

Файл themes.py

```

def change_theme(self, theme_index):
    if theme_index == 1:
        # Ставиться відповідна тема
        self.setStyleSheet(u"QWidget {\n"
            "    background-color: #F5FFFA;\n"
            "    font-family: 'Franklin Gothic Medium',\n"
            "'Arial Narrow', Arial, sans-serif;\n"
            "    font-size: 16px;\n"
            "}\n"
            "QPushButton {\n"
            "    background-color: rgb(203, 236,\n"
            "199);\n"
            "    border: none;\n"
            "    border-radius: 10px;\n"
            "}\n"
            "QPushButton:hover {\n"
            "    background-color: rgb(232, 199,\n"
            "236);\n"
            "}\n"
            "QPushButton:pressed {\n"
            "    background-color: rgb(199, 221,\n"
            "236);\n"
            "}\n"

```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				66
Змн.	Арк.	№ докум.	Підпис	Дата		

```

236);\n"

"QListView {\n"
"    background-color: #FFF5EE;\n"
"    border: 5px solid rgb(199, 221,

"    border-radius: 10px;\n"
"}\n"
"QSlider::groove:horizontal {\n"
"    border: 1px solid rgb(199, 221,

236);\n"

"    background-color: rgb(203, 236, 199);\n"
"    height: 5px;\n"
"    margin: 0px;\n"
"    }\n"
"QSlider::handle:horizontal {\n"
"    background-color: rgb(236, 213,

199);\n"

"    border: none;\n"
"    border-radius: 10px;\n"
"    height: 10px;\n"
"    width: 10px;\n"
"    margin: -15px 0px;\n"
"    }\n"
"QSlider::handle:vertical {\n"
"    background-color: rgb(236, 213,

199);\n"

""
"    border: none;\n"
"    border-radius: 10px;\n"
"    height: 20px;\n"
"    }\n"
"QSlider::groove:vertical {\n"
"    border: 1px solid rgb(199, 221,

236);\n"

"    background-color: rgb(203, 236, 199);\n"
"    width: 10px;\n"
"    margin: 0px;\n"
"    }")

elif theme_index == 2:
    self.setStyleSheet(u"QWidget {\n"
"        background-color: #f2f2f2;\n"
"        font-family: 'Franklin Gothic Medium',
'Arial Narrow', Arial, sans-serif;\n"
"        font-size: 16px;\n"
"}\n"
"QPushButton {\n"
"    background-color: #cccccc;\n"
"    border: none;\n"
"    border-radius: 10px;\n"
"}\n"
"QPushButton:hover {\n"
"    background-color: #b3b3b3;\n"
"}\n"
"QPushButton:pressed {\n"
"    background-color: #999999\n"
"}\n"
"QListView {\n"
"    background-color: #e6e6e6;\n"
"    border: 5px solid grey;\n"
"    border-radius: 10px;\n"
"}\n"

```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				67
Змн.	Арк.	№ докум.	Підпис	Дата		

```

"QSlider::groove:horizontal {\n"
"    border: 1px solid #636363;\n"
"    background-color: #808080;\n"
"    height: 5px;\n"
"    margin: 0px;\n"
"    }\n"
"QSlider::handle:horizontal {\n"
"    background-color: #000000;\n"
"    border: none;\n"
"    border-radius: 10px;\n"
"    height: 10px;\n"
"    width: 10px;\n"
"    margin: -15px 0px;\n"
"    }\n"
"QSlider::handle:vertical {\n"
"    background-color: #000000;\n"
"    border: none;\n"
"    border-radius: 10px;\n"
"    height: 20px;\n"
"    }\n"
"QSlider::g"
"roove:vertical {\n"
"    border: 1px solid #636363;\n"
"    background-color: #808080;\n"
"    width: 10px;\n"
"    margin: 0px;\n"
"    }")

elif theme_index == 3:
    self.setStyleSheet(u"QWidget {\n"
        "    background-color: #c2f0c2;\n"
        "\n"
        "    font-family: 'Franklin Gothic Medium',
'Arial Narrow', Arial, sans-serif;\n"
        "    font-size: 16px;\n"
        "}\n"
        "QPushButton {\n"
        "    background-color: #70db70;\n"
        "    border: none;\n"
        "    border-radius: 10px;\n"
        "}\n"
        "QPushButton:hover {\n"
        "    background-color: #33cc33;\n"
        "}\n"
        "QPushButton:pressed {\n"
        "    background-color: #1f7a1f;\n"
        "}\n"
        "QListView {\n"
        "    background-color: #d6f5d6;\n"
        "    border: 5px solid #248f24;\n"
        "    border-radius: 10px;\n"
        "}\n"
        "QSlider::groove:horizontal {\n"
        "    border: 1px solid #2eb82e;\n"
        "    background-color: #248f24;\n"
        "    height: 5px;\n"
        "    margin: 0px;\n"
        "    }\n"
        "QSlider::handle:horizontal {\n"
        "    background-color: #000000;\n"
        "    border: none;\n"

```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				68
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        "border-radius: 10px;\n"
        "height: 10px;\n"
        "width: 10px;\n"
        "margin: -15px 0px;\n"
        "}\n"
"QSlider::handle:vertical {\n"
"    background-color: #000000;\n"
"    border: none;\n"
"    border-radius: 10px;\n"
"    height: 20px;\n"
"    }\n"
"Q"
"Slider::groove:vertical {\n"
"    border: 1px solid #2eb82e;\n"
"    background-color: #248f24;\n"
"    width: 10px;\n"
"    margin: 0px;\n"
"    }")

elif theme_index == 0:
    self.setStyleSheet(u"QWidget {\n"
        "    background-color: #e6f2ff;\n"
        "\n"
        "    font-family: 'Franklin Gothic Medium',
'Arial Narrow', Arial, sans-serif;\n"
        "    font-size: 16px;\n"
        "}\n"
"QPushButton {\n"
"    background-color: #b3d9ff;\n"
"    border: none;\n"
"    border-radius: 10px;\n"
"    }\n"
"QPushButton:hover {\n"
"    background-color: #3399ff;\n"
"    }\n"
"QPushButton:pressed {\n"
"    background-color: #0066cc;\n"
"    }\n"
"QListView {\n"
"    background-color: #99ccff;\n"
"    border: 5px solid #001a33;\n"
"    border-radius: 10px;\n"
"    }\n"
"QSlider::groove:horizontal {\n"
"    border: 1px solid #99ccff;\n"
"    background-color: white;\n"
"    height: 5px;\n"
"    margin: 0px;\n"
"    }\n"
"QSlider::handle:horizontal {\n"
"    background-color: white;\n"
"    border: none;\n"
"    border-radius: 10px;\n"
"    height: 10px;\n"
"    width: 10px;\n"
"    margin: -15px 0px;\n"
"    }\n"
"QSlider::handle:vertical {\n"
"    background-color: #000000;\n"
"    border: none;\n"
"    border-radius: 10px;\n"

```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		69

```

"    height: 20px;\n"
"    }\n"
""
"QSlider::groove:vertical {\n"
"    border: 1px solid #99ccff;\n"
"    background-color: white;\n"
"    width: 10px;\n"
"    margin: 0px;\n"
"    }")

```

Файл work_with_file.py

```

import os

from PySide6.QtGui import QPixmap

def save_playlists(self, path_dir):
    """Saving all existing playlists"""
    if os.path.exists('playlists.txt'):
        with open('playlists.txt', 'r') as file:
            paths = file.readlines()
            if len(paths) > 0:
                indicator = False
                for existing_path in paths:
                    if self.check_playlist_when_add(existing_path,
path_dir):
                        indicator = True
                        break
                if indicator is not True:
                    with open('playlists.txt', 'a') as file_for_add:
                        file_for_add.write(path_dir + '\n')
            else:
                with open('playlists.txt', 'a') as file_for_add: # Add
first row of playlist
                    file_for_add.write(path_dir + '\n')
        else:
            with open('playlists.txt', 'w') as file: # Create file and add
playlist
                file.write(path_dir + '\n')

def load_playlists(self, path):
    """Loading last saving playlist"""
    if os.path.exists('playlists.txt'):
        with open('playlists.txt', 'r') as file:
            playlist_names = file.readlines()
            if playlist_names:
                last_playlist = playlist_names[-1]
                songs = os.listdir(last_playlist[:-1])
                self.path = last_playlist[:-1]
                if songs:
                    for song in songs:
                        if '.mp3' in song or '.wav' in song or '.ogg' in
song:
                            self.playlist.append(song)
                        elif '.jpg' in song or '.png' in song:
                            self.img_album_cover = QPixmap(self.path + '\\\
+ song)

self.ui.lbl_photo.setPixmap(self.img_album_cover)

```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        else:
            self.change_album_cover()

self.ui.lst_playlist.clear()
self.ui.lst_playlist.addItem(self.playlist)

```

Файл work_with_youtube.py

```

from PySide6.QtWidgets import QDialog, QDialogBox

from des_sub import Ui_Dialog

import asyncio
import async_for_youtube as async_m

def create_playlist_from_youtube(self):
    self.new_window = QDialog()
    self.ui_window = Ui_Dialog()
    self.ui_window.setupUi(self.new_window)

self.ui_window.pushButton.clicked.connect(self.change_path_for_youtube_music)

    self.ui_window.btn_enter.clicked.connect(self.download_from_youtube)
    self.new_window.show()

def change_path_for_youtube_music(self):
    path_to_audio_from_youtube =
QFileDialog.getExistingDirectory(self.new_window, 'Select Directory')
    if path_to_audio_from_youtube:
        self.path_to_audio_from_youtube = path_to_audio_from_youtube
        self.ui_window.label_2.setText(f'Save to:
{self.path_to_audio_from_youtube}')

def download_from_youtube(self):
    url = self.ui_window.ln_url.text()
    if url:
        mode = self.ui_window.comboBox.currentText()
        asyncio.run(async_m.start_process(url, mode,
self.path_to_audio_from_youtube))
        self.ui_window.label.setText('Completed!')

```

Файл async_for_youtube.py

```

import asyncio

import pytube.exceptions
from pytube import YouTube, Playlist
import os, re
import moviepy.editor as mp

async def download_data(url, mode, path):
    if mode == 'Video':
        try:
            yt = YouTube(url)
            yt.streams.filter(only_audio=True).first.download(path)
        except pytube.exceptions.RegexMatchError:
            print('Exception')

```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		71

```

elif mode == 'Playlist':
    try:
        p = Playlist(url)
        for video in p.videos:
            video.streams.filter(only_audio=True).first().download(path)
    except:
        pass

async def convert_data(url, path):
    print(path)
    if url:
        for file in os.listdir(path):
            if re.search('mp4', file):
                mp4 = os.path.join(path, file)
                mp3 = os.path.join(path, os.path.splitext(file)[0] + '.mp3')
                new_file = mp.AudioFileClip(mp4)
                new_file.write_audiofile(mp3, codec='libmp3lame')
                os.remove(mp4)

async def start_process(url, mode, path):
    task_download = asyncio.create_task(download_data(url, mode, path))
    await task_download
    task_convert = asyncio.create_task(convert_data(url, path))
    await task_convert

```

Файл дизайну головного вікна – design.py

```

# -*- coding: utf-8 -*-

#####
####
## Form generated from reading UI file 'design.ui'
##
## Created by: Qt User Interface Compiler version 6.5.1
##
## WARNING! All changes made in this file will be lost when recompiling UI
file!
#####
####

from PySide6.QtCore import (QCoreApplication, QDate, QDateTime, QLocale,
    QMetaObject, QObject, QPoint, QRect,
    QSize, QTime, QUrl, Qt)
from PySide6.QtGui import (QBrush, QColor, QConicalGradient, QCursor,
    QFont, QFontDatabase, QGradient, QIcon,
    QImage, QKeySequence, QLinearGradient, QPainter,
    QPalette, QPixmap, QRadialGradient, QTransform)
from PySide6.QtWidgets import (QApplication, QGridLayout, QHBoxLayout,
    QLabel,
    QListWidget, QListWidgetItem, QMainWindow, QPushButton,
    QSizePolicy, QSlider, QVBoxLayout, QWidget)
import res_rc

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        if not MainWindow.setObjectName():
            MainWindow.setObjectName(u"MainWindow")
            MainWindow.resize(700, 600)
            MainWindow.setMinimumSize(QSize(700, 600))

```

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

```

        icon = QIcon()
        icon.addFile(u":/icons/icons/loop_v1_icon.svg", QSize(),
QIcon.Normal, QIcon.Off)
        MainWindow.setWindowIcon(icon)
        MainWindow.setStyleSheet(u"QWidget {\n"
"    background-color: #e6f2ff;\n"
"    \n"
"    font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-
serif;\n"
"    font-size: 16px;\n"
"}\n"
"QPushButton {\n"
"    background-color: #b3d9ff;\n"
"    border: none;\n"
"    border-radius: 10px;\n"
"}\n"
"QPushButton:hover {\n"
"    background-color: #3399ff;\n"
"}\n"
"QPushButton:pressed {\n"
"    background-color: #0066cc;\n"
"}\n"
"QListView {\n"
"    background-color: #99ccff;\n"
"    border: 5px solid #001a33;\n"
"    border-radius: 10px;\n"
"}\n"
"QSlider::groove:horizontal {\n"
"    border: 1px solid #99ccff;\n"
"    background-color: #001a33;\n"
"    height: 5px;\n"
"    margin: 0px;\n"
"    }\n"
"QSlider::handle:horizontal {\n"
"    background-color: #000000;\n"
"    border: none;\n"
"    border-radius: 10px;\n"
"    height: 10px;\n"
"    width: 10px;\n"
"    margin: -15px 0px;\n"
"    }\n"
"QSlider::handle:vertical {\n"
"    background-color: #000000;\n"
"    border: none;\n"
"    border-radius: 10px;\n"
"    height: 20px;\n"
"    }\n"
""
        "QSlider::groove:vertical {\n"
"    border: 1px solid #99ccff;\n"
"    background-color: white;\n"
"    width: 10px;\n"
"    margin: 0px;\n"
"    }")

        self.centralwidget = QWidget(MainWindow)
        self.centralwidget.setObjectName(u"centralwidget")
        self.horizontalLayout_3 = QHBoxLayout(self.centralwidget)
        self.horizontalLayout_3.setObjectName(u"horizontalLayout_3")
        self.horizontalLayout_2 = QHBoxLayout()
        self.horizontalLayout_2.setObjectName(u"horizontalLayout_2")

```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

```

self.verticalLayout_2 = QVBoxLayout()
self.verticalLayout_2.setObjectName(u"verticalLayout_2")
self.verticalLayout = QVBoxLayout()
self.verticalLayout.setObjectName(u"verticalLayout")
self.lbl_photo = QLabel(self.centralwidget)
self.lbl_photo.setObjectName(u"lbl_photo")
sizePolicy = QSizePolicy(QSizePolicy.Preferred,
QSizePolicy.Preferred)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.lbl_photo.sizePolicy().hasHeightForWidth())

self.lbl_photo.setSizePolicy(sizePolicy)
self.lbl_photo.setMinimumSize(QSize(1, 1))
self.lbl_photo.setMaximumSize(QSize(350, 350))
self.lbl_photo.setStyleSheet(u"border: 2px solid black;\n"
"border-radius: 10px;")

self.lbl_photo.setPixmap(QPixmap(u":/icons/icons/album_cover_sample.png"))
self.lbl_photo.setScaledContents(True)
self.lbl_photo.setAlignment(Qt.AlignCenter)
self.lbl_photo.setMargin(0)

self.verticalLayout.addWidget(self.lbl_photo)

self.label = QLabel(self.centralwidget)
self.label.setObjectName(u"label")
sizePolicy1 = QSizePolicy(QSizePolicy.Ignored, QSizePolicy.Maximum)
sizePolicy1.setHorizontalStretch(0)
sizePolicy1.setVerticalStretch(0)

sizePolicy1.setHeightForWidth(self.label.sizePolicy().hasHeightForWidth())
self.label.setSizePolicy(sizePolicy1)
self.label.setAlignment(Qt.AlignCenter)

self.verticalLayout.addWidget(self.label)

self.label_time = QLabel(self.centralwidget)
self.label_time.setObjectName(u"label_time")
sizePolicy2 = QSizePolicy(QSizePolicy.Minimum, QSizePolicy.Maximum)
sizePolicy2.setHorizontalStretch(0)
sizePolicy2.setVerticalStretch(0)

sizePolicy2.setHeightForWidth(self.label_time.sizePolicy().hasHeightForWidth())

self.label_time.setSizePolicy(sizePolicy2)
self.label_time.setAlignment(Qt.AlignCenter)

self.verticalLayout.addWidget(self.label_time)

self.sldr_progress = QSlider(self.centralwidget)
self.sldr_progress.setObjectName(u"sldr_progress")
self.sldr_progress.setOrientation(Qt.Horizontal)

self.verticalLayout.addWidget(self.sldr_progress)

self.verticalLayout_2.addLayout(self.verticalLayout)

```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		74

```

self.gridLayout = QGridLayout()
self.gridLayout.setObjectName(u"gridLayout")
self.btn_stop = QPushButton(self.centralwidget)
self.btn_stop.setObjectName(u"btn_stop")
icon1 = QIcon()
icon1.addFile(u":/icons/icons/stop_icon.svg", QSize(), QIcon.Normal,
QIcon.Off)
self.btn_stop.setIcon(icon1)
self.btn_stop.setIconSize(QSize(24, 24))

self.gridLayout.addWidget(self.btn_stop, 1, 2, 1, 1)

self.btn_theme = QPushButton(self.centralwidget)
self.btn_theme.setObjectName(u"btn_theme")
icon2 = QIcon()
icon2.addFile(u":/icons/icons/theme.svg", QSize(), QIcon.Normal,
QIcon.Off)
self.btn_theme.setIcon(icon2)
self.btn_theme.setIconSize(QSize(24, 24))

self.gridLayout.addWidget(self.btn_theme, 2, 1, 1, 1)

self.btn_randow = QPushButton(self.centralwidget)
self.btn_randow.setObjectName(u"btn_randow")
icon3 = QIcon()
icon3.addFile(u":/icons/icons/random_icon.svg", QSize(),
QIcon.Normal, QIcon.Off)
self.btn_randow.setIcon(icon3)
self.btn_randow.setIconSize(QSize(24, 24))

self.gridLayout.addWidget(self.btn_randow, 1, 0, 1, 1)

self.btn_loop = QPushButton(self.centralwidget)
self.btn_loop.setObjectName(u"btn_loop")
icon4 = QIcon()
icon4.addFile(u":/icons/icons/loop_v2_icon.svg", QSize(),
QIcon.Normal, QIcon.Off)
self.btn_loop.setIcon(icon4)
self.btn_loop.setIconSize(QSize(24, 24))

self.gridLayout.addWidget(self.btn_loop, 1, 3, 1, 1)

self.btn_pause = QPushButton(self.centralwidget)
self.btn_pause.setObjectName(u"btn_pause")
icon5 = QIcon()
icon5.addFile(u":/icons/icons/pause_icon.svg", QSize(),
QIcon.Normal, QIcon.Off)
self.btn_pause.setIcon(icon5)
self.btn_pause.setIconSize(QSize(24, 24))

self.gridLayout.addWidget(self.btn_pause, 0, 1, 1, 1)

self.btn_next = QPushButton(self.centralwidget)
self.btn_next.setObjectName(u"btn_next")
icon6 = QIcon()
icon6.addFile(u":/icons/icons/next_icon.svg", QSize(), QIcon.Normal,
QIcon.Off)
self.btn_next.setIcon(icon6)
self.btn_next.setIconSize(QSize(24, 24))

```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		75

```

self.gridLayout.addWidget(self.btn_next, 0, 3, 1, 1)

self.btn_previous = QPushButton(self.centralwidget)
self.btn_previous.setObjectName(u"btn_previous")
icon7 = QIcon()
icon7.addFile(u":/icons/icons/previous_icon.svg", QSize(),
QIcon.Normal, QIcon.Off)
self.btn_previous.setIcon(icon7)
self.btn_previous.setIconSize(QSize(24, 24))

self.gridLayout.addWidget(self.btn_previous, 0, 0, 1, 1)

self.btn_add_playlist = QPushButton(self.centralwidget)
self.btn_add_playlist.setObjectName(u"btn_add_playlist")
icon8 = QIcon()
icon8.addFile(u":/icons/icons/playlist_add_icon.svg", QSize(),
QIcon.Normal, QIcon.Off)
self.btn_add_playlist.setIcon(icon8)
self.btn_add_playlist.setIconSize(QSize(24, 24))

self.gridLayout.addWidget(self.btn_add_playlist, 2, 3, 1, 1)

self.btn_mute = QPushButton(self.centralwidget)
self.btn_mute.setObjectName(u"btn_mute")
icon9 = QIcon()
icon9.addFile(u":/icons/icons/mute.svg", QSize(), QIcon.Normal,
QIcon.Off)
self.btn_mute.setIcon(icon9)
self.btn_mute.setIconSize(QSize(24, 24))

self.gridLayout.addWidget(self.btn_mute, 1, 1, 1, 1)

self.btn_play = QPushButton(self.centralwidget)
self.btn_play.setObjectName(u"btn_play")
icon10 = QIcon()
icon10.addFile(u":/icons/icons/play_icon.svg", QSize(),
QIcon.Normal, QIcon.Off)
self.btn_play.setIcon(icon10)
self.btn_play.setIconSize(QSize(24, 24))

self.gridLayout.addWidget(self.btn_play, 0, 2, 1, 1)

self.btn_add_playlist_from_youtube = QPushButton(self.centralwidget)
self.btn_add_playlist_from_youtube.setObjectName(u"btn_add_playlist_from_you
tube")
self.btn_add_playlist_from_youtube.setEnabled(True)
icon11 = QIcon()
icon11.addFile(u":/icons/icons/playlist.png", QSize(), QIcon.Normal,
QIcon.Off)
self.btn_add_playlist_from_youtube.setIcon(icon11)
self.btn_add_playlist_from_youtube.setIconSize(QSize(24, 24))

self.gridLayout.addWidget(self.btn_add_playlist_from_youtube, 2, 2,
1, 1)

self.verticalLayout_2.addLayout(self.gridLayout)

```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		76

```

self.horizontalLayout_2.addLayout(self.verticalLayout_2)

self.horizontalLayout = QHBoxLayout()
self.horizontalLayout.setObjectName(u"horizontalLayout")
self.sldr_volume = QSlider(self.centralwidget)
self.sldr_volume.setObjectName(u"sldr_volume")

sizePolicy.setHeightForWidth(self.sldr_volume.sizePolicy().hasHeightForWidth
())

self.sldr_volume.setSizePolicy(sizePolicy)
self.sldr_volume.setValue(80)
self.sldr_volume.setOrientation(Qt.Vertical)

self.horizontalLayout.addWidget(self.sldr_volume)

self.lst_playlist = QListWidget(self.centralwidget)
self.lst_playlist.setObjectName(u"lst_playlist")

self.horizontalLayout.addWidget(self.lst_playlist)

self.horizontalLayout_2.addLayout(self.horizontalLayout)

self.horizontalLayout_3.addLayout(self.horizontalLayout_2)

MainWindow.setCentralWidget(self.centralwidget)

self.retranslateUi(MainWindow)

QMetaObject.connectSlotsByName(MainWindow)
# setupUi

def retranslateUi(self, MainWindow):
    MainWindow.setWindowTitle(QCoreApplication.translate("MainWindow",
u"MainWindow", None))
    self.lbl_photo.setText("")
    self.label.setText(QCoreApplication.translate("MainWindow",
u"SONG_AUTHOR - SONG_NAME", None))
    self.label_time.setText(QCoreApplication.translate("MainWindow",
u"TextLabel", None))
    self.btn_stop.setText("")
    self.btn_theme.setText("")
    self.btn_random.setText("")
    self.btn_loop.setText("")
    self.btn_pause.setText("")
    self.btn_next.setText("")
    self.btn_previous.setText("")
    self.btn_add_playlist.setText("")
    self.btn_mute.setText("")
    self.btn_play.setText("")
    self.btn_add_playlist_from_youtube.setText("")
# retranslateUi

```

Файл дизайну допоміжного вікна – des_sub.py

```

# -*- coding: utf-8 -*-

#####
####

```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		77

```

## Form generated from reading UI file 'des-sub.ui'
##
## Created by: Qt User Interface Compiler version 6.5.1
##
## WARNING! All changes made in this file will be lost when recompiling UI
file!
#####

from PySide6.QtCore import (QCoreApplication, QDate, QDateTime, QLocale,
    QMetaObject, QObject, QPoint, QRect,
    QSize, QTime, QUrl, Qt)
from PySide6.QtGui import (QBrush, QColor, QConicalGradient, QCursor,
    QFont, QFontDatabase, QGradient, QIcon,
    QImage, QKeySequence, QLinearGradient, QPainter,
    QPalette, QPixmap, QRadialGradient, QTransform)
from PySide6.QtWidgets import (QApplication, QComboBox, QDialog,
    QHBoxLayout,
    QLabel, QLineEdit, QPushButton, QSizePolicy,
    QVBoxLayout, QWidget)
import res_sub_rc

class Ui_Dialog(object):
    def setupUi(self, Dialog):
        if not Dialog.setObjectName():
            Dialog.setObjectName(u"Dialog")
        Dialog.resize(700, 120)
        sizePolicy = QSizePolicy(QSizePolicy.Maximum, QSizePolicy.Maximum)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)

        sizePolicy.setHeightForWidth(Dialog.sizePolicy().hasHeightForWidth())
        Dialog.setSizePolicy(sizePolicy)
        Dialog.setMinimumSize(QSize(600, 120))
        icon = QIcon()
        icon.addFile(u":icons/icons/playlist.png", QSize(), QIcon.Normal,
            QIcon.Off)
        Dialog.setWindowIcon(icon)
        Dialog.setStyleSheet(u"QWidget {\n"
            "    background-color: rgb(199, 203, 236);\n"
            "    font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-
            serif;\n"
            "    font-size: 16px;\n"
            "}\n"
            "QPushButton {\n"
            "    background-color: rgb(203, 236, 199);\n"
            "    border: none;\n"
            "    border-radius: 10px;\n"
            "}\n"
            "QPushButton:hover {\n"
            "    background-color: rgb(232, 199, 236);\n"
            "}\n"
            "QPushButton:pressed {\n"
            "    background-color: rgb(199, 221, 236);\n"
            "}\n"
            "QLineEdit {\n"
            "    border: none;\n"
            "    border-radius: 10px;\n"
            "    background-color: rgb(203, 236, 199);\n"
            "}\n")

```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		78

```

"QLineEdit: hover {\n"
"    border: 1px solid rgb(232, 199, 236);\n"
"}\n"
"")

self.verticalLayout = QVBoxLayout(Dialog)
self.verticalLayout.setObjectName(u"verticalLayout")
self.ln_url = QLineEdit(Dialog)
self.ln_url.setObjectName(u"ln_url")

self.verticalLayout.addWidget(self.ln_url)

self.label = QLabel(Dialog)
self.label.setObjectName(u"label")
sizePolicy1 = QSizePolicy(QSizePolicy.Minimum, QSizePolicy.Maximum)
sizePolicy1.setHorizontalStretch(0)
sizePolicy1.setVerticalStretch(0)

sizePolicy1.setHeightForWidth(self.label.sizePolicy().hasHeightForWidth())
self.label.setSizePolicy(sizePolicy1)
self.label.setAlignment(Qt.AlignCenter)

self.verticalLayout.addWidget(self.label)

self.horizontalLayout = QHBoxLayout()
self.horizontalLayout.setObjectName(u"horizontalLayout")
self.label_2 = QLabel(Dialog)
self.label_2.setObjectName(u"label_2")

self.horizontalLayout.addWidget(self.label_2)

self.pushButton = QPushButton(Dialog)
self.pushButton.setObjectName(u"pushButton")

sizePolicy.setHeightForWidth(self.pushButton.sizePolicy().hasHeightForWidth(
))
self.pushButton.setSizePolicy(sizePolicy)
self.pushButton.setStyleSheet(u"padding-left: 10px;\n"
"padding-right: 10px;")

self.horizontalLayout.addWidget(self.pushButton)

self.comboBox = QComboBox(Dialog)
self.comboBox.addItem("")
self.comboBox.addItem("")
self.comboBox.setObjectName(u"comboBox")

sizePolicy.setHeightForWidth(self.comboBox.sizePolicy().hasHeightForWidth())
self.comboBox.setSizePolicy(sizePolicy)
self.comboBox.setStyleSheet(u"border-radius: 10px;\n"
"padding-left: 10px;\n"
"padding-right: 10px;")

self.horizontalLayout.addWidget(self.comboBox)

self.verticalLayout.addLayout(self.horizontalLayout)

self.btn_enter = QPushButton(Dialog)
self.btn_enter.setObjectName(u"btn_enter")
sizePolicy2 = QSizePolicy(QSizePolicy.Ignored, QSizePolicy.Fixed)

```

		Сидорчук А.В.			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		79

```

sizePolicy2.setHorizontalStretch(0)
sizePolicy2.setVerticalStretch(0)

sizePolicy2.setHeightForWidth(self.btn_enter.sizePolicy().hasHeightForWidth(
))

self.btn_enter.setSizePolicy(sizePolicy2)

self.verticalLayout.addWidget(self.btn_enter)

self.retranslateUi(Dialog)

QMetaObject.connectSlotsByName(Dialog)
# setupUi

def retranslateUi(self, Dialog):
    Dialog.setWindowTitle(QCoreApplication.translate("Dialog",
u"Dialog", None))
    self.ln_url.setPlaceholderText(QCoreApplication.translate("Dialog",
u"Link to the video\\playlist", None))
    self.label.setText(QCoreApplication.translate("Dialog", u"Paste copy
link above", None))
    self.label_2.setText(QCoreApplication.translate("Dialog", u"Save to:
", None))
    self.pushButton.setText(QCoreApplication.translate("Dialog",
u"Select...", None))
    self.comboBox.setItemText(0, QCoreApplication.translate("Dialog",
u"Video", None))
    self.comboBox.setItemText(1, QCoreApplication.translate("Dialog",
u"Playlist", None))

    self.btn_enter.setText(QCoreApplication.translate("Dialog",
u"Enter", None))
    # retranslateUi

```

		Сидорчук А.В..			КІК.КР.Б – 123 – 16 – ПЗ	Арк.
		Фальковський І.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		80