

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ ДЛЯ
ВПОРЯДКУВАННЯ ДОМАШНЬОЇ ФОНОТЕКИ
SOFTWARE IMPLEMENTATION OF AN APPLICATION
FOR ORGANIZING HOME MUSIC LIBRARY

Виконав: студент 4 курсу, групи КНТ-111сп
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

БАБАНСЬКИЙ Я.С.

(ПРИЗВИЩЕ та ініціали)

Керівник ЛЕОЩЕНКО С.Д.

(ПРИЗВИЩЕ та ініціали)

Рецензент СКРУПСЬКИЙ С.Ю.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. СУББОТІН
« » 2024 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

БАБАНСЬКОГО Ярослава Сергійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація застосунку для впорядкування домашньої фонотеки. Software Implementation of an Application for Organizing Home Music Library

керівник проєкту (роботи) д-р філософ., ст.викл. Леощенко Сергій Дмитрович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “24” квітня 2024 року
№ 168

2. Строк подання студентом проєкту (роботи) 04 травня 2024 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ЛЕОЩЕНКО С.Д., ст.викл.		
Нормоконтроль	КАМІНСЬКА Ж.К., асистент		

7. Дата видачі завдання « 04 » квітня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Ярослав БАБАНСЬКИЙ
(Імя ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Сергій ЛЕОЩЕНКО
(Імя ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
96 с., 26 рис., 3 табл., 4 дод., 24 джерел.

АУДИО, ФОНОТЕКА, МУЗИЧНА КОЛЕКЦІЯ,
КРОСПЛАТФОРМНИЙ ЗАСТОСУНОК, РОЗРОБКА, МОВА
ПРОГРАМУВАННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Об'єкт дослідження – процес програмної реалізації застосунку для впорядкування домашньої фонотеки.

Предмет дослідження – програмні засоби для реалізації застосунку для впорядкування домашньої фонотеки.

Метою роботи є програмна реалізація застосунку для впорядкування домашньої фонотеки.

Матеріали, методи та технічні засоби: мова програмування C# та фреймворк .NET Framework, система керування базами даних Microsoft SQL Server, засоби моделювання UML.

Результати. Розроблено застосунок – впорядкування домашньої фонотеки.

Практична цінність роботи полягає у програмній реалізації застосунку для впорядкування домашньої фонотеки у вигляді кросплатформного застосунку.

Висновки. Реалізовано застосунок для впорядкування домашньої фонотеки у вигляді кросплатформного застосунку.

Галузь використання – поціновувачі музики.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 96 pages, 26 figures, 3 tables, 4 appendixes, 24 sources.

AUDIO, MUSIC LIBRARY, MUSIC COLLECTION, CROSS-PLATFORM APPLICATION, DEVELOPMENT, PROGRAMMING LANGUAGE, SOFTWARE

The object of research is the process of software implementation of an application for organizing your home music library.

The subject of the research is software tools for implementing an application for organizing your home music library.

The purpose of this work is to programmatically implement an application for organizing your home music library.

Materials, methods, and technical tools: C# programming language and .NET Framework, Microsoft SQL Server Database Management System, UML modeling tools.

Results. An application has been developed-organizing your home music library.

The practical value of the work lies in the software implementation of an application for organizing your home music library in the form of a cross-platform application.

Conclusions. Implemented an application for organizing your home music library as a cross-platform application.

Field of use – music lovers.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	7
Вступ	8
1 Аналіз предметної області	11
1.1 Поняття фонотеки	11
1.2 Використання програмних засобів для відтворення та впорядкування домашньої фонотеки	15
1.3 Порівняння існуючих аналогів	19
1.4 Постановка завдання	25
1.5 Висновки за розділом 1	29
2 Матеріали і методи	30
2.1 Вибір мови програмування	30
2.2 Вибір середовища розробки	37
2.3 Структурна схема розробки	43
2.4 Висновки за розділом 2	44
3 Основні рішення щодо реалізації компонентів системи	45
3.1 Проєктування моделей системи	45
3.2 Проєктування інтерфейсу	54
4 Експлуатація, тестування та експериментальне дослідження програми	59
4.1 Призначення й умови застосування програми	59
4.2 Вимоги до інтерфейсу	59
4.3 Умови виконання програми	60
4.4 Методика та результати тестування	60
Висновки	64
Перелік джерел посилання	66
Додаток А Технічне завдання	69
Додаток Б Опис програми	74
Додаток В Текст програми	80
Додаток Д Слайди презентації	90

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ОС – операційна система;

ПЗ – програмне забезпечення;

СКБД – система керування базами даних;

GUI – graphical user interface;

MVC – model-view-controller;

UML – unified modeling language.

ВСТУП

Розробка програми для організації домашньої музичної бібліотеки – фонотеки може бути нагальним завданням з кількох причин, кожна з яких пов'язана з ускладненням управління цифровими музичними колекціями та зростаючим попитом на персоналізовані музичні можливості [1]–[3].

Зростаючий обсяг цифрової музики – з розповсюдженням цифрових музичних платформ, потокових сервісів та інтернет-магазинів користувачі отримують доступ до постійно розширюваного каталогу пісень, альбомів та списків відтворення. Як результат, управління особистою музичною бібліотекою та її організація стають дедалі складнішими, що диктує необхідність ефективних організаційних інструментів [1]–[3].

Різноманітні музичні джерела та формати – любителі музики часто купують музику з різних джерел, включаючи цифрові завантаження, потокові послуги, компакт-диски, вінілові платівки та особисті записи. Кожне джерело може використовувати різні формати файлів, стандарти метаданих та рівні якості, що призводить до фрагментації та невідповідності в музичній бібліотеці. Додаток, який може уніфікувати та стандартизувати музичні метадані та формати, може вирішити цю проблему [1]–[3].

Персоналізоване музичний супровід – багато слухачів музики воліють добірки, які відображають їх унікальні смаки, уподобання і настрої. Вони можуть створювати власні списки відтворення, класифікувати музику за жанрами або настроєм, а також підбирати колекції на основі певних тем або подій. Додаток, який полегшує індивідуальне кураторство та організацію, може підвищити якість прослуховування та задоволеність користувачів [1]–[3].

Потреба в ефективному пошуку та виявленні – зі збільшенням розміру та різноманітності музичних колекцій користувачам потрібні ефективні механізми пошуку та виявлення, щоб швидко та легко знаходити та вивчати музику. Розширені фільтри пошуку, інтелектуальні рекомендації і параметри

перегляду допоможуть користувачам знаходити нову музику, заново відкривати старі улюблені і знайомитися зі схожими виконавцями або жанрами [1]–[3].

Важливість управління метаданими – точні та комплексні метадані необхідні для ефективної організації та категоризації музики в бібліотеці. Метадані включають таку інформацію, як імена виконавців, назви альбомів, номери треків, дати виходу, жанри та обкладинки альбомів. Застосунок, що дозволяє користувачам керувати та редагувати метадані, забезпечує послідовність та точність у музичній бібліотеці [1]–[3].

Можливості резервного копіювання та синхронізації – любителі музики часто витрачають багато часу і зусиль на створення своїх музичних колекцій, що робить вкрай важливим захист від втрати даних і забезпечення доступності на декількох пристроях. Застосунок, що пропонує можливості резервного копіювання та синхронізації як за допомогою хмарного сховища, так і за допомогою локальної синхронізації, забезпечує спокій і зручність для користувачів [1]–[3].

Підтримка декількох платформ та пристроїв – користувачі очікують, що їх програма для організації музики буде доступна на різних платформах, включаючи настільні комп'ютери, ноутбуки, смартфони та планшети, а також різні операційні системи, такі як Windows, macOS, iOS та Android. Кросплатформна сумісність забезпечує гнучкість і доступність для користувачів незалежно від бажаних ними пристроїв [1]–[3].

Спільнота та соціальні функції – музика часто використовується в соціальних мережах, де користувачі діляться та обговорюють свої улюблені треки, альбоми та списки відтворення з друзями, родиною та онлайн-спільнотами. Включення соціальних функцій, таких як обмін інформацією, лайки, коментарі та спільні плейлисти, підвищує залученість і розвиває почуття спільності у користувачів [1]–[3].

Можливості налаштування та персоналізації – кожен слухач музики має унікальні уподобання та звички, коли справа доходить до організації та

використання музики. Надання таких можливостей налаштування, як вибір теми, Налаштування макета та користувацькі теги чи мітки, дозволяє користувачам адаптувати додаток до своїх індивідуальних потреб та уподобань [1]–[3].

На закінчення слід зазначити, що розробка програмного забезпечення (ПЗ) для організації домашньої музичної бібліотеки є актуальним завданням у зв'язку з ускладненням управління цифровими музичними колекціями, зростаючим попитом на персоналізований музичний досвід і потребою в ефективних механізмах організації, пошуку і виявлення інформації. Вирішуючи ці завдання і пропонуючи інноваційні функції і можливості, розробники можуть створити цінний інструмент, який поліпшить якість прослуховування музики користувачами і задовольнить їх зростаючі потреби і очікування [1]–[3].

Отже актуальною задачею залишається програмна реалізація застосунку для впорядкування домашньої фонотеки.

Для виконання такої задачі необхідно виконати наступні завдання:

- виконати дослідження проблемної області накопичення та впорядкування фонотеки;
- виконати дослідження переваг та недоліків існуючих аналогів для впорядкування фонотеки;
- виконати вибір технічних засобів реалізації запропонованого застосунку та обґрунтувати його;
- спроектувати архітектуру нового застосунку;
- програмно реалізувати новий застосунок;
- провести тестування розробки;
- сформулювати висновки на основі виконаної роботи.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Перший розділ пояснювальної записки до роботи буде присвячено ознайомленню із проблемною областю накопичення та впорядкування фонотеки, визначенням фонотеки, як музичної колекції та ознайомленню із існуючими аналогічними рішеннями впорядкування фонотеки. Після аналітичної частини буде наведено перелік функціональних та нефункціональних вимог до ПЗ, що буде розроблятися під час роботи.

1.1 Поняття фонотеки

Домашня музична бібліотека, також відома як домашня фонотека, – це колекція аудіозаписів, які люди чи сім'ї збирають та зберігають для особистого задоволення, збереження культури та художнього сприйняття [4]–[6]. Як правило, вона включає в себе широкий спектр музичних жанрів, виконавців, альбомів і треків, підібраних з урахуванням особистих смаків, уподобань та інтересів. Основна мета домашньої музичної бібліотеки-забезпечити джерело розваг, натхнення та емоційного зв'язку за допомогою музики [4]–[6].

Визначення та компоненти:

– домашня музична бібліотека включає в себе різні типи аудіозаписів, включаючи альбоми (повні колекції пісень або композицій, виданих виконавцем або групою), сингли (окремі треки або пісні, випущені незалежно один від одного або в складі збірки), мікстейпи/плейлисти (добірки пісень, підібрані колекціонером за тематикою, настроєм або жанрами), концертні записи (концертні виступи, студійні сесії або контрабандні записи, які фіксують досвід прослуховування живої музики), саундтреки (музика, написана для фільмів, телевізійних шоу, відеоігор або інших засобів масової інформації);

– формати – аудіозаписи можуть зберігатися у фізичних форматах, таких як вінілові платівки, компакт-диски, касети, або в цифрових форматах, таких як MP3, FLAC, WAV, або в потокових сервісах [4]–[6].

Причини для збору:

– особисте задоволення: багато людей збирають музику просто тому, що їм подобається її слухати. Створення ретельно підібраної колекції улюблених пісень, альбомів і виконавців забезпечує нескінченні години насолоди і розваг [4]–[6];

– емоційний зв'язок: музика здатна викликати сильні емоції та спогади. Збір пісень, які мають особисте значення або перегукуються з конкретними переживаннями, може стати способом зберегти і переосмислити дорогі серцю моменти життя [4]–[6];

– збереження культурної спадщини: музика відображає культурну спадщину, традиції та різноманітність суспільств у всьому світі. Збір та збереження музичних записів сприяє документуванню та збереженню культурної ідентичності та художнього вираження [4]–[6];

– художнє сприйняття: любителі музики часто цінують артистизм, креативність і майстерність виконання музичних композицій. Створення різноманітної музичної бібліотеки дозволяє людям вивчати різні жанри, стилі та періоди музичної історії [4]–[6];

– відкриття та дослідження: Колекціонування музики відкриває можливості для знайомства з новими виконавцями, жанрами та звуками. Це дозволяє людям розширити свій музичний кругозір, поглибити розуміння різноманітних музичних традицій і вступити в контакт зі світовим співтовариством любителів музики [4]–[6];

– соціальні зв'язки: спільне прослуховування музики з друзями, родиною та іншими любителями музики сприяє встановленню соціальних зв'язків. Обговорення улюблених пісень, обмін рекомендаціями та спільне відвідування концертів чи музичних заходів зміцнюють стосунки та сприяють обміну досвідом [4]–[6].

Способи збору інформації:

– купівля: колекціонери можуть купувати музичні записи різними способами, включаючи придбання фізичних форматів, таких як вінілові платівки, компакт-диски або касети, у музичних магазинах, інтернет-магазинах роздрібної торгівлі або на музичних фестивалях. Вони також можуть придбати цифрові завантаження або потокові підписки з онлайн-платформ, таких як iTunes, Amazon Music, Bandcamp, Spotify або Apple Music [4]–[6];

– торгівля та обмін: деякі колекціонери обмінюються музичними записами з друзями, родичами або колегами-колекціонерами. Це може включати в себе обмін фізичними копіями альбомів, мікстейпів або контрабандних записів, а також обмін цифровими файлами по електронній пошті, на файлообмінних платформах або в соціальних мережах [4]–[6];

– оцифровка аналогових форматів: колекціонери можуть оцифровувати аналогові музичні формати, такі як вінілові платівки, касети або компакт-диски, щоб перетворити їх у цифрові файли для спрощення зберігання, відтворення та збереження. Цей процес передбачає використання аудіообладнання, програмного забезпечення або спеціалізованих сервісів для перекладу аудіозаписів в цифровий формат [4]–[6].

Організація та кураторство:

– каталогізація: колекціонери часто організовують свою музичну бібліотеку, використовуючи системи каталогізації або програмні програми, які дозволяють їм класифікувати, маркувати та індексувати музичні записи на основі метаданих, таких як імена виконавців, назви альбомів, жанри, дати виходу та тривалість треків. Це полегшує ефективний пошук, перегляд і витяг музичних записів [4]–[6];

– спеціальні списки відтворення: створення спеціальних списків відтворення або мікстейпів дозволяє колекціонерам створювати персоналізовані колекції пісень, адаптовані до конкретних тем, настроїв або

подій. Вони можуть розташувати пісні в певній послідовності, жанрі або темпі, щоб створити єдине враження від прослуховування [4]–[6];

- обкладинки альбомів: багато колекціонерів цінують обкладинки альбомів як невід'ємну частину процесу прослуховування музики. Вони можуть організувати свою музичну бібліотеку так, щоб у ній поряд з аудіозаписами були представлені обкладинки альбомів, вкладиші та інші візуальні елементи, що підвищує естетичну привабливість і ностальгічну цінність колекції [4]–[6].

Збереження та обслуговування:

- зберігання: правильне зберігання музичних записів має важливе значення для збереження їх якості та довговічності. Колекціонери можуть використовувати спеціалізовані рішення для зберігання, такі як ящики для записів, підставки для компакт-дисків або архівні коробки, щоб захистити фізичні формати від пилу, вологи та пошкоджень. Цифрові файли зазвичай зберігаються на жорстких дисках, хмарних платформах зберігання даних або медіасерверах для легкого доступу та резервного копіювання [4]–[6];

- технічне обслуговування: регулярне технічне обслуговування музичних записів включає в себе очищення, відновлення і підтримання фізичних форматів для запобігання зносу з плином часу. Для цифрових файлів регулярне резервне копіювання даних, оновлення метаданих і організація файлів в логічну структуру каталогів забезпечують довгострокове збереження і доступність [4]–[6].

Таким чином, домашня музична бібліотека або звуковий архів - це ретельно підібрана колекція аудіозаписів, яку люди збирають і зберігають для особистого задоволення, збереження культури і художнього сприйняття. Будь то придбання фізичних форматів, оцифровка аналогових записів або вивчення цифрових потокових платформ, Колекціонування музики дозволяє людям спілкуватися зі своїми улюбленими виконавцями, відкривати нові звуки та створювати значущі враження за допомогою музики [4]–[6].

1.2 Використання програмних засобів для відтворення та впорядкування домашньої фонотеки

ПЗ відіграють вирішальну роль у відтворенні та організації домашньої музичної бібліотеки, надаючи користувачам засоби для ефективного управління своєю колекцією аудіозаписів, вивчення її та насолоди нею. Ці інструменти пропонують широкий спектр функцій, призначених для покращення якості прослуховування музики та спрощення організації музичних колекцій [7], [8].

Медіаплеєри – це програмні програми, призначені для відтворення аудіо-та відеофайлів на комп'ютерах, смартфонах, планшетах та інших цифрових пристроях. Вони підтримують різні аудіоформати, включаючи MP3, FLAC, WAV, AAC та OGG, що дозволяє користувачам відтворювати свою музичну бібліотеку з високоякісним звуком [7], [8].

Медіаплеєри пропонують такі функції, як управління відтворенням (відтворення, пауза, зупинка, пропуск), регулювання гучності, режими повтору/перемішування та настроювані списки відтворення [7], [8].

Популярне ПЗ для медіаплеєрів включає:

- «Winamp» – класичний медіаплеєр, відомий своїм настроюваним інтерфейсом, широкою підтримкою плагінів і розширеними можливостями відтворення [7], [8];
- медіаплеєр Windows – вбудований медіаплеєр для операційних систем Windows, що пропонує основні функції відтворення та інструменти управління бібліотекою;
- «iTunes» – медіаплеєр Apple і цифрова вітрина для macOS і Windows, що надають доступ до великої бібліотеки музики, фільмів, телешоу і подкастів [7], [8];
- «VLC Media Player» – медіаплеєр з відкритим кодом, відомий своєю підтримкою широкого формату, легким дизайном та сумісністю між платформами [7], [8].

ПЗ для управління музичною бібліотекою допомагає користувачам ефективно організовувати, класифікувати та керувати своїми музичними колекціями. Ці інструменти зазвичай надають такі функції, як редагування метаданих, управління обкладинками альбомів, автоматична організація файлів та пакетне перейменування файлів [7], [8].

ПЗ для управління музичною бібліотекою також пропонує розширені можливості пошуку та фільтрації, що дозволяє користувачам швидко знаходити певні пісні, альбоми чи виконавців у своїй колекції [7], [8].

Популярне ПЗ для управління музичною бібліотекою включає:

- «MusicBee» – комплексний застосунок для управління музикою для Windows, який пропонує функції управління бібліотекою, відтворення та додавання тегів, а також підтримку перетворення аудіо та копіювання компакт-дисків [7], [8];

- «Foobar2000» – легко налаштовується музичний програвач і менеджер бібліотек для Windows, який підтримує широкий спектр аудіоформатів, включаючи аудіо високої роздільної здатності, і пропонує розширені функції створення тегів і організації [7], [8];

- «JRiver Media Center» – мультимедійний програмний комплекс для Windows і macOS, який забезпечує відтворення мультимедіа, управління бібліотекою і потокову передачу, а також підтримує відтворення аудіо аудіофільського рівня і можливості настройки [7], [8].

Редактори музичних тегів та метаданих дозволяють користувачам редагувати та керувати метаданими, пов'язаними з їх музичними файлами, включаючи імена виконавців, назви альбомів, номери треків, жанри та обкладинки альбомів [7], [8]. Ці інструменти дозволяють користувачам забезпечувати послідовність та точність у своїй музичній бібліотеці шляхом стандартизації метаданих у своїй колекції [7], [8].

Редактори музичних тегів і метаданих можуть також пропонувати такі функції, як автоматичний пошук метаданих з онлайн-баз даних, пакетне додавання тегів і вибір обкладинок [7], [8].

ПЗ для редагування популярних музичних тегів та метаданих включає:

- «Mp3tag» – універсальний редактор метаданих для Windows, який підтримує широкий спектр аудіоформатів і пропонує функції пакетної розмітки, перейменування файлів і пошуку в онлайн-базі даних [9], [10];
- «Kid3» – редактор метаданих з відкритим кодом, доступний для Windows, macOS та Linux, який підтримує редагування тегів ID3, коментарів Vorbis та тегів MP4, а також можливості пакетної розмітки та сценаріїв [11], [12];
- «MusicBrainz Picard» – програма для створення музичних тегів з відкритим кодом, яка використовує базу даних MusicBrainz для автоматичного відображення та позначення музичних файлів, забезпечуючи точність та послідовність метаданих у всій бібліотеці [13], [14].

Цифрові аудіостанції (digital audio workstation – DAW) – це програмні програми, які використовуються для запису, редагування, змішування та створення музики та аудіовмісту.

Хоча DAW в основному використовуються музикантами, продюсерами та звукорежисерами для професійного створення музики, вони також можуть використовуватися ентузіастами для організації та управління їх музичною бібліотекою [7], [8].

DAW пропонує розширені інструменти редагування та мастерингу, підтримку MIDI, плагіни для віртуальних інструментів та можливості обробки ефектів [7], [8].

Популярне ПЗ DAW включає в себе:

- «Ableton Live» – універсальна DAW, відома своїм інтуїтивно зрозумілим інтерфейсом, можливостями роботи в режимі реального часу і великою бібліотекою вбудованих інструментів і ефектів [7], [8];
- «Logic Pro X» – професійна DAW від Apple для macOS, що пропонує потужні інструменти для запису, редагування і мікшування, а також велику колекцію віртуальних інструментів, синтезаторів і звукових ефектів [7], [8];

– «FL Studio» – популярна DAW-програма для Windows і macOS, що користується популярністю у виробників електронної музики за інноваційний робочий процес, секвенування на основі шаблонів і широкий спектр вбудованих плагінів і інструментів [7], [8].

Послуги потокової передачі музики. Сервіси потокової передачі музики надають доступ до великих бібліотек музичного вмісту для потокової передачі на вимогу та відтворення в режимі офлайн. Ці послуги пропонують персоналізовані рекомендації, списки відтворення та радіостанції на основі уподобань користувачів та звичок прослуховування [7], [8].

Хоча багато поточних сервісів використовуються в основному для потокової передачі музики, вони також пропонують функції для організації та управління особистими музичними бібліотеками, такими як створення списків відтворення, Синхронізація бібліотек та завантаження в автономному режимі [7], [8].

Популярні сервіси потокової музики включають:

– «Spotify» – провідний сервіс потокової передачі музики, відомий своїм великим каталогом, персоналізованими списками відтворення та функціями обміну в соціальних мережах, що пропонує як безкоштовну, так і преміум-підписку;

– «Apple Music» – потоковий сервіс Apple, що пропонує доступ до великої музичної бібліотеки, ретельно відібраних списків відтворення та ексклюзивного вмісту, а також інтеграцію з бібліотекою iTunes для безперешкодного доступу до особистих музичних колекцій.

– «Музика з YouTube» – Google і стрімінговий сервіс, що надає доступ до широкого спектру музичних кліпів, офіційних треків, живих виступів, і призначений для користувача контент, а також персональні рекомендації та опції в автономному режимі відтворення.

Таким чином, ПЗ відіграють життєво важливу роль у відтворенні та організації домашньої музичної бібліотеки, пропонуючи функції для відтворення, управління бібліотекою, редагування метаданих та створення

музики [7], [8]. Незалежно від того, чи використовуються медіаплеєри, програмне забезпечення для управління бібліотекою, редактори тегів, DAW або потокові сервіси, користувачі мають широкий спектр можливостей для поліпшення якості прослуховування музики і забезпечення впорядкованості і доступності своєї музичної колекції [7], [8].

1.3 Порівняння існуючих аналогів

Mr3tag, Kid3 та MusicBrainz Picard – це популярні інструменти для редагування та управління музичними метаданими, кожен з яких має унікальні функції. Mr3tag відомий своїми широкими можливостями створення тегів і варіантами настройки, в той час як Kid 3 надає зручний інтерфейс і основні функції створення тегів. MusicBrainz Picard спеціалізується на автоматичному пошуку метаданих і точного маркування з використанням бази даних MusicBrainz і технології акустичної дактилоскопії. Вибір між цими інструментами залежить від уподобань користувача, сумісності з платформою та конкретних вимог до управління музичною бібліотекою.

Mr3tag (рис. 1.1) надає користувачам простий у використанні інструмент для редагування тегів у різних популярних аудіофайлах [9], [10]. Дозволяє здійснювати правки в ID3-тегах, в коментарях MP3 і OGG файлів, проводити заміну спеціальних символів або слів. З інших можливостей програми можна виділити функції імпорту/експорту інформації, призначені для користувача сценарії, створення і складання плейлистів, підтримку Unicode, роботу з онлайн-базою даних freedb і здійснення групового перейменування файлів на основі інформації в тегах [9], [10].

MP3tag може правити мультимедійні файли як в одиночному, так і в пакетному режимі. Утиліта підтримує наступні аудіоформати: AAC, ALAC, AIFF (.aif/.aifc/.aiff), Direct Stream Digital Audio (.dsf), FLAC, Matroska (.mka/.mkv), Monkey's Audio (.ape), MP3, MPEG - 4 (.mp4, .m4a, m4b, m4v, iTunes сумісний), Musepack (.mpc), Ogg Vorbis (.ogg), OPUS, OptimFROG

(ofr/ofs), Speex (.spx), TAK, True Audio (.tta), Windows Media Audio (.wma), WavPack (.wv), Waveform Audio File Format (.wav).

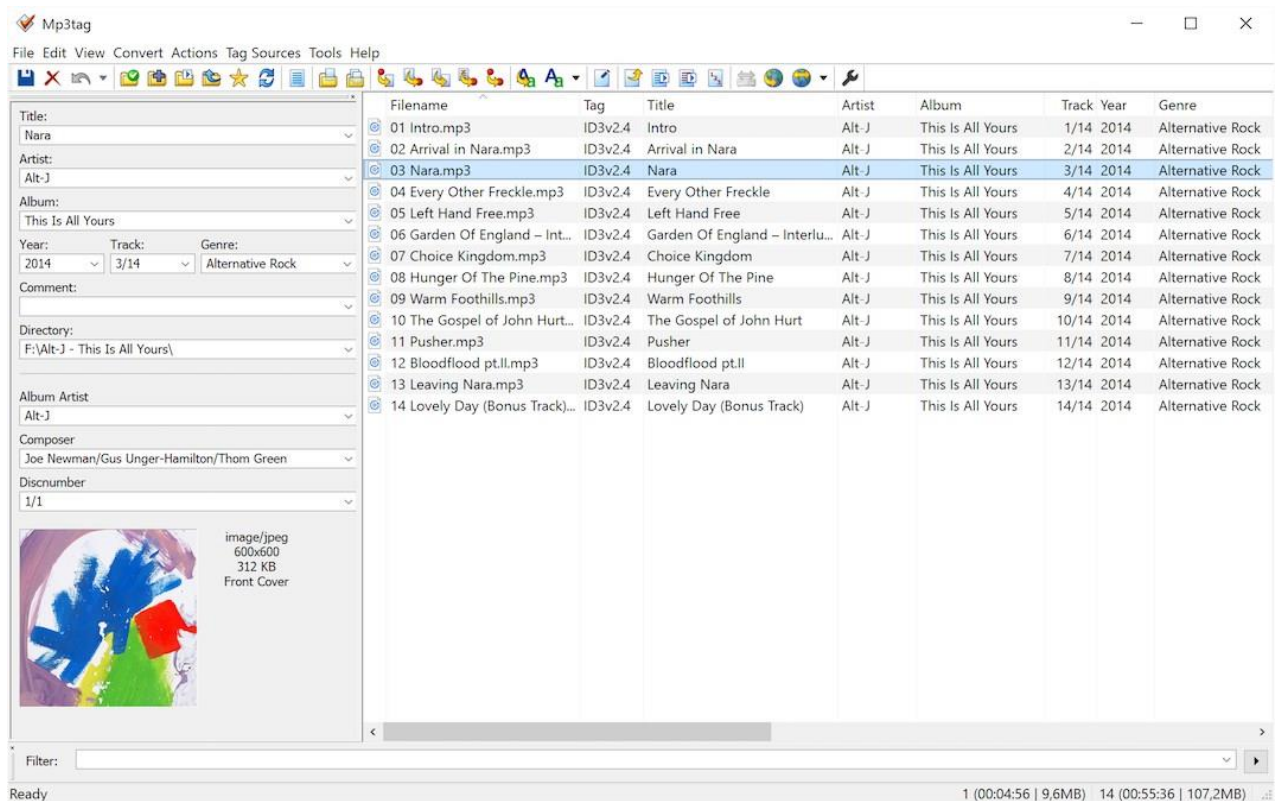


Рисунок 1.1 – Приклад Mp3tag [9]

До всього іншого Mp3tag вміє автоматично проводити оновлення тегів і обкладинок альбомів, які вбудовані в них, а також оновлювати їх за допомогою онлайн ресурсів Freedb, Amazon або Discogs. Інтерфейс Mp3tag переведений на більш ніж 40 мов [9], [10].

Kid3 – це кроплатформний редактор звукових тегів з відкритим кодом для багатьох форматів аудіофайлів (рис. 1.2). Він підтримує DSF, MP3, Ogg, FLAC, MPC, MPEG-4 (mp4/m4a/m4b), AAC, Opus, SPX, True Audio, APE, WavPack, WMA, WAV, AIFF, трекерні модулі [11], [12].

Особливість:

- редагування та перетворення тегів ID3v1.1, ID3v2.3 та ID3v2.4;
- доступ до всіх полів тегів;
- пакетне редагування тегів декількох файлів;

- створення тегів на основі імен файлів або вмісту інших полів тегів;
- генерація імен файлів з тегів, перейменування та створення каталоги з тегів;
- створення файлів плейлистів;
- автоматичне перетворення верхній і нижній регістри та заміна рядків;
- пошук та фільтрація файлів за тегами;
- імпорт з gnudb.org, TrackType.org, MusicBrainz, Discogs, Amazon та інших джерел даних про альбоми [11], [12];
- автоматичний пакетний імпорт метаданих і обкладинок для декількох альбомів з різних джерел;
- імпортування за допомогою acoustic fingerprints та AcoustID;
- експорт тегів в форматі CSV, HTML, плейлисти, Kover XML і в інших форматах.

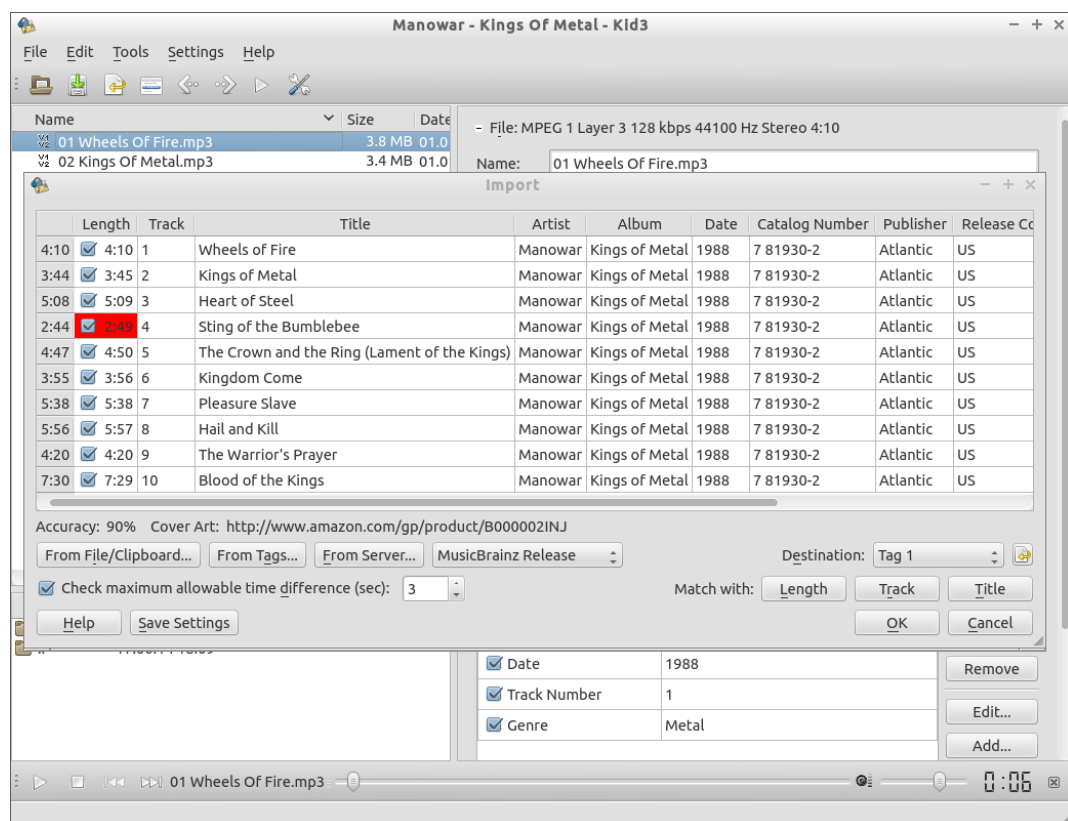


Рисунок 1.2 – Приклад Kid3 Audio Tagger [11]

MusicBrainz – це проєкт MetaBrainz, метою якого є створення спільної музичної бази даних, подібної до проєкту freedb [13], [14]. Компанія MusicBrainz була заснована у відповідь на обмеження, накладені на базу даних компакт-дисків, базу даних програмних програм для пошуку інформації про аудіодиски в Інтернеті. MusicBrainz розширив свої цілі, вийшовши за рамки сховища метаданих CD (це інформація про виконавців, артистів, авторів пісень і т.д.), і перетворившись в структуровану онлайн-базу даних (рис. 1.3) про музику [13], [14].

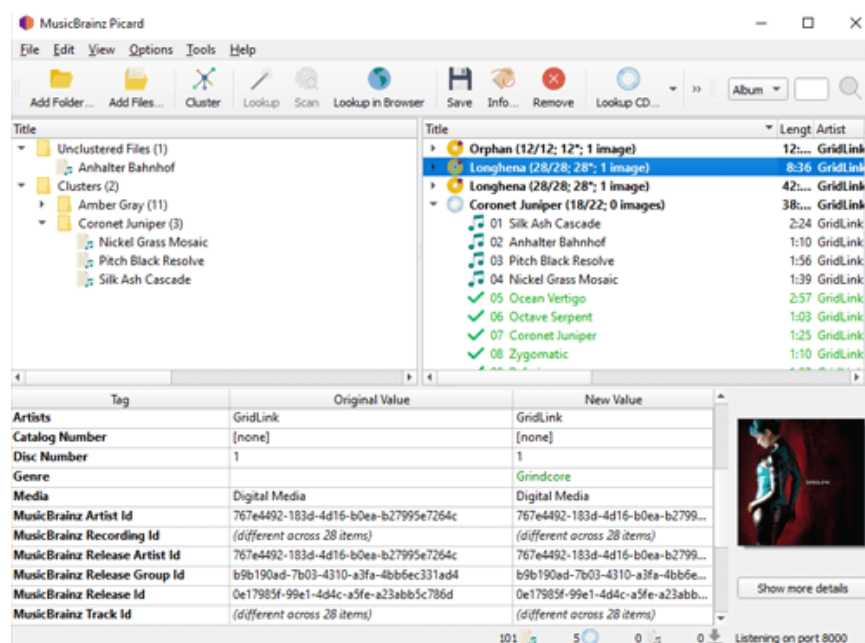


Рисунок 1.3 – Приклад MusicBrainz

MusicBrainz збирає інформацію про виконавців, їх записані твори та стосунки між ними. Записи про записані твори містять, як мінімум, назву альбому, назви треків і тривалість кожного треку. Ці записи підтримуються редакторами-добровольцями, які дотримуються рекомендацій спільноти щодо стилю. Записані твори можуть також містити інформацію про дату випуску та країну, ідентифікатор компакт-диска, обкладинку, акустичний відбиток, текст анотації у вільній формі та інші метадані. Станом на жовтень 2023 року MusicBrainz містить інформацію про приблизно 2,2 мільйона виконавців, 3,9 мільйона релізів та 30,4 мільйона записів. Кінцеві користувачі можуть

використовувати програмне забезпечення, що взаємодіє з MusicBrainz, для додавання тегів метаданих до своїх цифрових медіа, таких як ALAC, FLAC, MP3, Ogg Vorbis або AAC [13], [14].

Порівняємо розглянуті рішення та представимо результати у вигляді таблиці – табл. 1.1. Для порівняння використаємо наступні критерії:

- сумісність з платформами – цей критерій вказує, які операційні системи (ОС) підтримує ПЗ. Це важливо, оскільки користувачі повинні переконатися, що програмне забезпечення сумісне з операційною системою їх пристрою;

- підтримувані формати файлів – цей критерій визначає типи форматів аудіофайлів, які може обробляти ПЗ. Користувачам важливо знати, чи підтримуються їх улюблені формати файлів, щоб забезпечити сумісність з їх музичною бібліотекою;

- особливості та функціонал – цей критерій включає в себе цілий ряд можливостей і інструментів, пропонованих ПЗ. Це включає редагування метаданих, пакетне маркування, параметри налаштування та будь-які унікальні функції, які відрізняють ПЗ від інших;

- простота використання – цей критерій відображає зручність використання ПЗ та те, наскільки легко користувачам орієнтуватися в ньому та використовувати його функції. Важливо, щоб користувачі відчували себе комфортно та ефективно під час використання програмного забезпечення;

- спільнота та підтримка – цей критерій враховує доступність підтримки спільноти, форумів, документації та регулярних оновлень. Ефективна система підтримки гарантує, що користувачі зможуть знайти допомогу та ресурси, коли це необхідно, і що ПЗ регулярно підтримується та вдосконалюється;

- інтеграція з музичними бібліотеками – цей критерій дозволяє оцінити, наскільки добре ПЗ інтегрується з онлайн-музичними базами даних для пошуку метаданих і як воно підтримує різні завдання управління

музичною бібліотекою. Інтеграція з музичними бібліотеками підвищує функціональність і зручність використання ПЗ.

Таблиця 1.1 – Програмні засоби для впорядкування домашньої фонотеки

Критерії порівняння	Mp3tag	Kid3	MusicBrainz Picard
Сумісність з платформою	Windows	Windows, macOS, Linux	Windows, macOS, Linux
Підтримувані формати файлів	MP3, FLAC, OGG, WMA, AAC, WAV та інші	MP3, FLAC, OGG та інші	MP3, FLAC, OGG, AAC, WMA та інші
Особливості та функціонал	Комплексне редагування метаданих, пакетне створення тегів, налаштування полів тегів, підтримка сценаріїв	Редагування тегів ID3, коментарів Vorbis і тегів MP4, пакетне створення тегів, пошук в базі даних в режимі онлайн, можливості налаштування	Автоматичний пошук метаданих, пакетне додавання тегів, перейменування файлів, підтримка акустичної дактилоскопії
Простота використання	Інтуїтивно зрозумілий інтерфейс, зручний дизайн	Простий інтерфейс, основні функції створення тегів	Більш швидкий процес навчання, використання бази даних MusicBrainz і акустичної дактилоскопії
Спільнота та підтримка	Велика спільнота користувачів, активні форуми, велика документація	Форуми спільноти, посібники користувача, регулярні оновлення	Спільнота MusicBrainz, форуми, документація, регулярні оновлення
Інтеграція з музичними бібліотеками	Інтеграція з Discogs та MusicBrainz для пошуку метаданих, підтримка скриптів користувачів для онлайн-баз даних	Інтеграція з онлайн-базами даних для пошуку метаданих, підтримка користувачьких налаштувань тегування та скриптів	Використання бази даних MusicBrainz для пошуку метаданих, підтримка акустичної дактилоскопії

Розглядаючи ці критерії, користувачі можуть оцінити ПЗ, виходячи зі своїх конкретних потреб, уподобань і пріоритетів в області редагування музичних метаданих і управління ними. Кожна функція надає цінну інформацію про можливості програмного забезпечення та його придатність для різних випадків використання.

У цій таблиці представлено коротке порівняння Mp3tag, Kid3 і MusicBrainz Picard з різних аспектів, включаючи сумісність з платформами, підтримувані формати файлів, можливості і функціонал, простоту використання, співтовариство і підтримку, а також інтеграцію з музичними бібліотеками. Користувачі можуть використовувати цю інформацію для прийняття обґрунтованого рішення на основі їх конкретних потреб та уподобань щодо редагування та управління музичними метаданими.

В більшості існуючих рішень досить ідентичні функції та особливості. Проте під час аналізу, помітно, що усі рішення обмежені окремим інформаційними базами. А у випадку, якщо система все таки використовує онлайн-бази, користувачі відзначають досить хаотичне заповнення даних.

Саме тому, актуальною задачею залишається програмна реалізація нового застосунку для впорядкування домашньої фонотеки.

1.4 Постановка завдання

Розробка ПЗ для організації домашньої музичної бібліотеки є актуальним завданням у зв'язку з ускладненням управління цифровими музичними колекціями, зростаючим попитом на персоналізований музичний досвід і потребою в ефективних механізмах організації, пошуку і виявлення інформації. Вирішуючи ці завдання і пропонуючи інноваційні функції і можливості, розробники можуть створити цінний інструмент, який поліпшить якість прослуховування музики користувачами і задовольнить їх зростаючі потреби і очікування.

Отже актуальною задачею залишається програмна реалізація застосунку для впорядкування домашньої фонотеки.

1.4.1 Функціональність системи

Система має реалізовувати наступні функціональні вимоги [15], [16]:

- аутентифікація користувачів і управління профілями. Програма повинна дозволяти користувачам створювати облікові записи та безпечно входити в систему, а також надавати функціональні можливості для управління профілями, такі як редагування інформації користувача і переваг;
- управління музичною бібліотекою: підтримка додавання, видалення та упорядкування музичних файлів; дозволяє користувачам імпортувати музику з різних джерел, включаючи локальні сховища, онлайн-потоківі послуги та зовнішні пристрої; можливість класифікувати музику за виконавцями, альбомами, жанрами, роком випуску та іншими метаданими; можливість створювати власні списки відтворення та організовувати музичні колекції на основі особистих уподобань;
- редагування метаданих: дозволяє користувачам редагувати метадані, пов'язані з музичними файлами, включаючи імена виконавців, назви альбомів, номери треків, жанри та обкладинки альбомів; надає можливості пакетного редагування для оновлення метаданих для декількох файлів одночасно; підтримує автоматичний пошук метаданих з онлайн-баз даних для спрощення процесу створення тегів;
- відтворення та потокове передавання: надає вбудований музичний плеєр для відтворення аудіофайлів у додатку; підтримує відтворення різних аудіоформатів, включаючи MP3, FLAC, WAV, AAC і OGG;
- інтеграція з популярними поточковими сервісами для забезпечення безперешкодного доступу до потокового музичного контенту;
- пошук і навігація: реалізована надійна функція пошуку, що дозволяє користувачам шукати музичні файли за ключовими словами, виконавцем,

альбомом, жанром чи іншими критеріями; надайте параметри фільтрації та сортування для полегшення перегляду та навігації по музичній бібліотеці; пропонує зручний інтерфейс з інтуїтивно зрозумілими елементами навігації і налаштованим видами;

- налаштування та персоналізація: дозволяє користувачам налаштовувати зовнішній вигляд програми, включаючи теми, кольірні схеми та параметри макета; надає налаштування для налаштування параметрів відтворення, таких як налаштування еквалайзера, швидкість відтворення та підтримка різноманітних ефектів; підтримує користувальницькі теги, написи і рейтинги для спрощення персоналізованої організації та категоризації музичних файлів;

- резервне копіювання та синхронізація: пропонує функції резервного копіювання та синхронізації для забезпечення цілісності даних та доступності на декількох пристроях; дозволяє користувачам синхронізувати музичні бібліотеки між різними пристроями, включаючи комп'ютери, смартфони та планшети; надає можливості автоматичного резервного копіювання і синхронізації за розкладом, щоб звести до мінімуму ризик втрати даних;

- інтеграція із зовнішніми пристроями: підтримка інтеграції з зовнішніми пристроями, такими як USB-накопичувачі, Зовнішні жорсткі диски і пристрої мережевого зберігання даних; дозволяє користувачам отримувати доступ до музичних файлів, що зберігаються на зовнішніх пристроях, і керувати ними безпосередньо з програми;

- доступність і локалізація: забезпечення спеціальних можливостей, таких як підтримка читання з екрану, навігація з клавіатури і перетворення тексту в мову, для користувачів з обмеженими можливостями; підтримка декількох мов і варіантів локалізації для користувачів з різним мовним досвідом. Також необхідно забезпечити спеціальні можливості, такі як підтримка читання з екрану, навігація з клавіатури і перетворення тексту в мову, для користувачів з обмеженими можливостями; підтримка декількох мов і варіантів локалізації для користувачів з різним мовним досвідом.

Ці функціональні вимоги описують основні функції та можливості, якими повинна володіти програма для організації домашньої музичної бібліотеки. Виконання цих вимог забезпечить всебічний і зручний для користувача досвід управління музичними колекціями та користування ними.

1.4.2 Вимоги до інтерфейсу

Інтерфейс системи повинен відрізнятися лаконічністю, інтуїтивною зрозумілістю та зручністю для користувачів. Дизайн має бути адаптивним для зручності роботи на будь-якому дисплеї [15], [16].

1.4.3 Вимоги до безпеки

Забезпечення конфіденційності та безпеки даних забезпечується дотримання відповідних правил та рекомендацій щодо захисту даних користувачів та захисту інформації користувача.

1.4.4 Не функціональні вимоги до продукту

Апаратні вимоги до сервера:

- центральний процесор – від 8 фізичних ядер та 16 потоків з тактовою частотою від 4 GHz;
- оперативна пам'ять формату DDR4, з тактовою частотою від 2600 MHz та об'ємом від 128 Гб;
- накопичувач (формату HDD або SSD) з вільним простором від 10 Гб (для розгортання усіх дамів музичних баз даних).

Програмні вимоги до сервера:

- розгорнута серверна операційна система (ОС), наприклад: Windows Server, Linux Server або macOS Server;
- розгорнута SQL-сумісна СКБД.

Апаратні вимоги до клієнта:

- центральний процесор – від 2 фізичних ядер та 2 потоків з тактовою частотою від 1,7 GHz (з можливістю збільшення під час навантаження);
- оперативна пам'ять об'ємом від 4 Гб;
- накопичувач (формату HDD або SSD) з вільним простором від 0,5 Гб.

Програмні вимоги до сервера:

- наявність веббраузера актуальної версії.

1.5 Висновки за розділом 1

Перший розділ пояснювальної записки до роботи було присвячено ознайомленню із проблемною областю накопичення та впорядкування фонотеки, визначенням фонотеки, як музичної колекції та ознайомленню із існуючими аналогічними рішеннями впорядкування фонотеки. Після аналітичної частини було наведено перелік функціональних та нефункціональних вимог до ПЗ, що буде розроблятися під час роботи.

2 МАТЕРІАЛИ І МЕТОДИ

Другий розділ пояснювальної записки до роботи буде присвячено висвітленню матеріалів та методів, що буде використано для програмної реалізації застосунку для впорядкування домашньої фонотеки. Буде досліджено, обрано та аргументовано технологію розробки, мову програмування у зв'язці із програмними надбудовами, та патерну структури застосунку.

2.1 Вибір мови програмування

C# – це потужна сучасна мова програмування, розроблена Microsoft. Він є частиною .Екосистема NET, яка є надійною та універсальною платформою для створення різних типів додатків, включаючи вебдодатки. «.Net» надає розробникам єдину платформу розробки, пропонуючи інструменти, бібліотеки та фреймворки для оптимізації процесу розробки [17], [18].

C# – це статично типізована мова, що означає, що змінні повинні бути оголошені із зазначенням їх типів даних під час компіляції. Такий підхід дає кілька переваг:

- виявлення помилок під час компіляції – оскільки помилки типу виявляються під час компіляції, розробники можуть виявляти та виправляти проблеми на ранніх стадіях процесу розробки, зменшуючи ймовірність виникнення помилок під час виконання [17], [18];

- надійність і обслуговуваність коду – сувора типізація підвищує надійність і ремонтпридатність коду за рахунок застосування строгих правил перевірки типів. Це полегшує розробникам розуміння, модифікацію та налагодження коду, особливо у великомасштабних проєктах [17], [18].

Продуктивність є критичним фактором при розробці вебзастосунків, особливо для додатків з високим трафіком і високими робочими

навантаженнями. .Net Core, крос-платформна версія .Net з відкритим кодом, забезпечує чудову оптимізацію продуктивності, що робить її придатною для створення високопродуктивних вебзастосунків [17], [18]:

- оптимізоване виконання – .Net Core використовує розширену оптимізацію середовища виконання, компіляцію вчасно та ефективні методи управління пам'яттю для забезпечення більш високої продуктивності порівняно з іншими платформами, такими як Node.js або PHP [17], [18];

- масштабованість – характеристики продуктивності .Net Core забезпечують високий ступінь масштабованості, дозволяючи вебдодаткам ефективно обробляти велику кількість одночасних користувачів і запитів.

Єдина платформа розробки. Однією з ключових сильних сторін .Net є її здатність служити уніфікованою платформою розробки для створення різних типів додатків [17], [18]. Розробники можуть використовувати C# та .NET для створення не лише вебзастосунків, а й настільних, мобільних та хмарних додатків. Ця уніфікована платформа має ряд переваг:

- повторне використання коду – розробники можуть повторно використовувати код у різних типах проєктів, скорочуючи час та зусилля на розробку. Це особливо вигідно для організацій з різноманітними вимогами до застосунків [17], [18];

- досвід послідовної розробки – оскільки розробники можуть використовувати однакові інструменти, мови та платформи для різних типів додатків, вони можуть підтримувати послідовний робочий процес розробки, що призводить до підвищення продуктивності та ефективності [17], [18].

Багата екосистема. Екосистема .Net пропонує безліч інструментів, бібліотек та фреймворків для полегшення розробки вебзастосунків [17], [18]. Деякі з ключових компонентів екосистеми .Net включають:

- ASP.NET Core – це високопродуктивний кросплатформний фреймворк для створення сучасних вебзастосунків. Він надає такі функції, як шаблон Model-View-Controller (MVC), введення залежностей та проміжне

програмне забезпечення, що полегшує розробку надійних та масштабованих вебзастосунків [17], [18];

- Entity Framework Core – це об'єктно-реляційна платформа відображення, яка полегшує доступ до бази даних у програмах .NET. це забезпечує зручний спосіб взаємодії з базами даних за допомогою сильно набраних об'єктів, зменшуючи кількість стандартного коду, необхідного для операцій доступу до даних [17], [18];

- стандартні бібліотеки. Net – .Net Standard – це набір API, які визначають загальний набір функціональних можливостей для всіх мережових реалізацій. Це дозволяє розробникам створювати бібліотеки, які можуть використовуватися на різних платформах .Net, забезпечуючи сумісність і інтероперабельність [17], [18].

Кросплатформна сумісність. З появою .Net Core C# та .NET стали кросплатформними, підтримуючи розробку в Windows, Linux та macOS. Така кросплатформна сумісність дає ряд переваг:

- гнучкість платформи – розробники можуть вибирати платформу, яка найкраще відповідає їхнім потребам та інфраструктурі, не прив'язуючись до конкретної операційної системи. Така гнучкість дозволяє організаціям оптимізувати свої інвестиції в існуючу інфраструктуру та адаптуватися до різних середовищ розгортання [17], [18];

- єдиний досвід розробки – незалежно від цільової платформи розробники можуть використовувати однакові інструменти, мови та фреймворки для створення та розгортання мережових застосунків. Така послідовність спрощує процес розробки та скорочує час навчання, пов'язаний із технологіями, що залежать від платформи [17], [18].

Безпека є головним пріоритетом при розробці вебзастосунків, і .NET надає кілька вбудованих функцій, які допомагають розробникам вирішувати поширені загрози безпеці [17], [18]. Серед ключових функцій безпеки .Net є:

- перевірка запитів – ASP.NET Core включає вбудовані механізми перевірки запитів для захисту від поширених вебвразливостей, таких як

міжсайтовий сценарій (XSS) та введення SQL. Розробники можуть налаштувати правила перевірки запитів для очищення вхідних даних та запобігання шкідливим атакам [17], [18];

- шифрування даних – .Net надає надійні бібліотеки шифрування для захисту конфіденційних даних під час передачі та спокою. Розробники можуть використовувати ці бібліотеки для реалізації алгоритмів шифрування, таких як AES та RSA, для захисту даних від несанкціонованого доступу [17], [18];

- аутентифікація та авторизація – ASP.NET Core включає вбудовані механізми аутентифікації та авторизації для управління доступом до вебзастосунків. Розробники можуть реалізовувати схеми аутентифікації, такі як JWT і OAuth, і визначати політики авторизації для обмеження доступу до певних ресурсів на основі ролей і дозволів користувачів [17], [18].

C# та .NET мають велику та активну спільноту розробників, що забезпечує ряд переваг:

- обмін знаннями – розробники можуть використовувати колективні знання та досвід спільноти для вивчення нових методів, усунення несправностей та бути в курсі останніх тенденцій та найкращих практик у розробці вебзастосунків [17], [18];

- внесок у розробку з відкритим кодом – екосистема .Net підтримується активною спільнотою розробників з відкритим кодом, яка сприяє розробці бібліотек, фреймворків та інструментів. Ці спільні зусилля розширюють можливості .Net та сприяють інноваціям у розробці вебзастосунків [17], [18];

- ресурси спільноти – існує безліч онлайн-форумів, груп користувачів та ресурсів спільноти, присвячених розробці на C# та .NET. ці ресурси містять багато посібників, статей та прикладів коду, які допомагають розробникам вдосконалювати свої навички та створювати кращі вебпрограми [17], [18].

В решті решт, C# та .NET пропонують привабливе поєднання функцій, продуктивності та підтримки екосистеми, що робить їх одними з найкращих рішень для розробки вебзастосунків. .Net надає розробникам всі необхідні інструменти для створення надійних, масштабованих і безпечних веб - додатків-від суворої типізації і безпеки до крос-платформної сумісності і захищеності. Завдяки процвітаючій спільноті та багатій екосистемі бібліотек та фреймворків, C# та.NET продовжують розвиватися та впроваджувати інновації, визначаючи майбутнє веброботи [17], [18].

Обґрунтуємо вибір, виконавши порівняння, найбільш актуальних та відомих зв'язок мов програмування та надбудов (фреймворків) за такими критеріями:

- суворе введення тексту – вказує, чи забезпечує мова, що використовується в стеку, суворе введення тексту, що може вплинути на надійність та зручність супроводу коду;
- продуктивність – відноситься до характеристик продуктивності стека технологій, включаючи такі фактори, як швидкість та ефективність використання ресурсів;
- масштабованість – відображає здатність стека технологій ефективно масштабуватися, пристосовуючись до збільшеного робочого навантаження і кількості користувачів;
- кросплатформність – вказує, чи підтримує стек технологій розробку та розгортання в декількох операційних системах;
- підтримка спільноти – описує розмір і рівень активності спільноти розробників, пов'язаного зі стеком технологій, які можуть впливати на доступність ресурсів і підтримки;
- багата екосистема – вказує на доступність інструментів, бібліотек та фреймворків, які доповнюють стек технологій, полегшуючи розробку та розширюючи функціональність;

- функції безпеки – відноситься до вбудованих механізмів безпеки та функцій, що надаються стеком технологій для захисту від поширених вразливостей;
- крива навчання – відображає легкість або складність вивчення та освоєння стека технологій, що може вплинути на продуктивність розробників та терміни реалізації проєктів;
- підтримка баз даних – вказує діапазон баз даних, підтримуваних технологічним стеком для зберігання та управління прикладними даними;
- підтримка паралелізму – відображає підтримку паралельного програмування і одночасної обробки декількох завдань в рамках технологічного стеку;
- модуль шаблонізації – позначає модуль шаблонізації або мову розмітки, що використовується для створення динамічного вмісту на вебсторінках;
- підтримка проміжного програмного забезпечення – вказує, чи надає стек технологій компоненти проміжного програмного забезпечення для обробки запитів та логіки програми;
- управління пакетами – вказує на систему управління пакетами, що використовується в стеку технологій для управління залежностями та сторонніми бібліотеками;
- параметри розгортання – вказує на доступні варіанти розгортання та платформи, що підтримуються стеком технологій для розгортання вебзастосунків;
- популярні компанії/проєкти – наведені приклади відомих компаній або проєктів, що використовують стек технологій, а також інформація про його впровадження та використання в реальних умовах.

Враховуючи ці аспекти, розробники можуть приймати обґрунтовані рішення при виборі технологічного стека для своїх проєктів вебзастосунків, беручи до уваги такі фактори, як продуктивність, масштабованість, безпека та підтримка спільноти. Порівняння мов програмування наведено у табл. 2.1.

Таблиця 2.1 – Порівняння мов програмування

	C# + .NET	JavaScript + Node.js	Python + Django	PHP + Symfony
Суворі типізація	Так	Ні	Так	Ні
Уявлення	Високий	Помірний	Помірний	Помірний
Масштабованість	Високий	Помірний	Помірний	Помірний
Кросплатформність	Так (з. NET Core)	Так	Так	Так
Підтримка спільноти	Великий	Великий	Великий	Великий
Багата екосистема	Так	Так	Так	Так
Функції безпеки	Сильний	Помірний	Помірний	Помірний
Процес навчання	Помірний	Помірний	Помірний	Помірний
Підтримка баз даних	Широкий діапазон	Широкий діапазон	Широкий діапазон	Широкий діапазон
Підтримка паралелізму	Так (з асинхронністю / очікуванням)	Так (з архітектурою , керованою подіями)	Так (за допомогою async / await)	Так (з асинхронністю / очікуванням)
Механізм створення шаблонів	Razor	Різний	Шаблони Django	Twig
Підтримка проміжного програмного забезпечення	Так	Так	Так	Так
Управління пакетами	NuGet	, npm	, pip	Composer
Варіанти розгортання	Різні (Docker, Azure, AWS і т. д.)	Різні (Docker, AWS, Heroku і т. д.)	Різні (Docker, AWS, Heroku і т. д.)	Різні (Docker, AWS, Heroku і т. д.)
Популярні компанії / проекти	Microsoft, Stack Overflow, GitHub і т. д.	Netflix, PayPal, LinkedIn і т. д.	Instagram, Spotify, Dropbox, etc.	BlaBlaCar, Trivago, Spotify, etc.

У цій таблиці представлений всебічний огляд сильних сторін і характеристик кожного технологічного стека, що допомагає розробникам приймати обґрунтовані рішення, засновані на їх вимогах і перевагах до проєкту.

2.2 Вибір середовища розробки

Visual Studio широко відомий як одне з найкращих інтегрованих середовищ розробки (integrated development environment – IDE) для розробки на C# та.NET, і з поважних причин [19], [20].

Надійний набір функцій. Visual Studio може похвалитися повним набором функцій, розроблених спеціально для розробки на C# та .NET. до них відносяться:

- редактор коду – потужний редактор коду з такими функціями, як підсвічування синтаксису, заповнення коду, інструменти рефакторингу та можливості навігації кодом, що полегшує розробникам ефективно написання та управління кодом [19], [20];
- інструменти налагодження – розширені засоби налагодження з такими функціями, як точки зупинки, вікна спостереження, стеки викликів та перевірка змінних у режимі реального часу, що дозволяє розробникам швидко виявляти та виправляти проблеми в процесі розробки [19], [20];
- інтегрована підтримка Git – вбудована інтеграція Git з такими функціями, як контроль версій, розгалуження, злиття та вирішення конфліктів, дозволяє розробникам безперешкодно співпрацювати та ефективно керувати вихідним кодом [19], [20];
- модульні тестові платформи – інтеграція з популярними модульними тестовими платформами, такими як MSTest, NUnit та xUnit, дозволяє розробникам писати та виконувати модульні тести безпосередньо в IDE та забезпечувати якість та надійність коду [19], [20];

- менеджер пакетів NuGet – вбудована підтримка NuGet, менеджера пакетів для .NET, що дозволяє розробникам легко знаходити, встановлювати та керувати сторонніми бібліотеками та залежностями [19], [20];
- інструменти аналізу коду – вбудовані інструменти аналізу коду, такі як Codelens та Code Metrics, дають уявлення про якість коду, продуктивність та ремонтпридатність, а також допомагають розробникам приймати обґрунтовані рішення в процесі розробки [19], [20];
- IntelliSense – це потужна функція завершення коду, яка забезпечує контекстно-залежні пропозиції та автоматичне завершення елементів коду, зменшуючи помилки друку та прискорюючи виконання завдань з написання коду [19], [20].

Visual Studio доступний на декількох платформах, включаючи Windows та macOS, що дозволяє розробникам гнучко вибирати середовище, яке найкраще відповідає їхнім потребам та уподобанням. Завдяки Visual Studio для Mac розробники можуть насолоджуватися тим самим багатим досвідом розробки та функціями, що підвищують продуктивність, як у macOS, забезпечуючи безперебійну співпрацю та спільний доступ до коду на різних платформах [19], [20].

Visual Studio забезпечує надійну підтримку розширень, дозволяючи розробникам налаштовувати та розширювати IDE відповідно до їх конкретних вимог [19], [20]. Visual Studio Marketplace пропонує широкий спектр розширень, включаючи інструменти, шаблони та доповнення, створені корпорацією Майкрософт та спільнотою розробників, що охоплюють різні аспекти розробки, такі як продуктивність, аналіз коду, тестування та розгортання. Незалежно від того, чи потрібна розробникам додаткова мовна підтримка, інтеграція зі сторонніми сервісами або спеціалізовані інструменти для вирішення конкретних завдань, вони можуть легко знайти і встановити розширення для поліпшення робочого процесу розробки [19], [20].

Visual Studio легко інтегрується з Microsoft Azure, платформою хмарних обчислень та сервісами, пропонованими Microsoft. Ця інтеграція

дозволяє розробникам створювати, розгортати та керувати хмарними програмами безпосередньо з IDE, використовуючи такі послуги Azure, як служба додатків Azure, функції Azure, база даних SQL Azure та сховище Azure [19], [20]. Завдяки таким функціям, як шаблони Azure Resource Manager, Cloud Explorer і інтеграція з Azure DevOps, розробники можуть спростити розробку і розгортання хмарних додатків, відстежувати продуктивність додатків і ефективно управляти ресурсами в звичному середовищі Visual Studio [19], [20].

Visual Studio надає надійні інструменти співпраці, які дозволяють розробникам ефективно працювати над проєктами незалежно від їх розташування чи розміру команди. Такі функції, як Live Share, дозволяють декільком розробникам співпрацювати в режимі реального часу, редагувати код, налагоджувати та ділитися термінальними сесіями без необхідності спільного використання екрана або синхронізації коду [19], [20]. Крім того, Visual Studio легко інтегрується з Azure DevOps, надаючи потужні інструменти для керування версіями, відстеження робочих елементів, безперервної інтеграції та безперервної доставки (CI/CD), дозволяючи командам впроваджувати гнучкі методи та ефективно надавати високоякісне програмне забезпечення [19], [20].

Visual Studio призначена для простої роботи з великими і складними кодовими базами, забезпечуючи чудову продуктивність і масштабованість. IDE використовує різні оптимізації для забезпечення безперебійної роботи навіть при роботі з великими проєктами, що містять тисячі файлів і залежностей. Такі функції, як фонові компіляція, поетапні збірки та аналіз усього рішення, допомагають скоротити час збірки та підвищити загальну продуктивність, дозволяючи розробникам зосередитися на написанні коду, а не чекати завершення збірки [19], [20].

Інтегрований досвід розробки. Visual Studio надає інтегровані можливості розробки, які охоплюють весь життєвий цикл розробки програмного забезпечення, від створення проєкту до розгортання і далі.

Розробники можуть виконувати такі завдання, як написання коду, налагодження, тестування, профілювання та розгортання додатків у єдиному, згуртованому середовищі. Плавна інтеграція різних інструментів і робочих процесів усуває необхідність перемикатися між різними додатками або середовищами, скорочуючи час перемикання контексту і оптимізуючи процес розробки [19], [20].

IDE користується перевагами динамічного і активного співтовариства розробників, MVP Microsoft (найбільш цінних фахівців) і інженерів Microsoft, які вносять свій внесок в її постійний розвиток і вдосконалення [19], [20]. Документація Visual Studio є всеосяжною та регулярно оновлюється, надаючи цінні ресурси, навчальні посібники та посібники з усунення несправностей, які допоможуть розробникам розпочати роботу та подолати труднощі. Крім того, форуми Visual Studio, репозиторії GitHub та Stack Overflow є цінними ресурсами, де розробники можуть звертатися за допомогою, ділитися знаннями та співпрацювати з іншими членами спільноти [19], [20].

В решті решт, Visual Studio, безсумнівно, є однією з найкращих IDE для розробки на C# та .NET, пропонуючи багатий набір функцій, сумісність між платформами, розширюваність, інтеграцію Azure, інструменти співпраці, продуктивність та масштабованість. Завдяки інтуїтивно зрозумілому інтерфейсу, надійним інструментам розробки та повній інтеграції з екосистемою Microsoft Visual Studio дозволяє розробникам ефективно створювати високоякісне програмне забезпечення, що робить його кращим вибором для розробників у всьому світі [19], [20].

Виконаємо порівняння IDE Visual studio, з іншими середовищами, що підтримують розробку на C#. Результати у вигляді таблиці наведено нижче (табл. 2.2). Вибираючи аспекти для порівняння, я прагнув охопити широкий спектр факторів, важливих для розробників при виборі IDE для розробки C#. Ці аспекти охоплюють різні аспекти, такі як функціональність, підтримка платформи, продуктивність, підтримка спільноти, простота використання, інтеграція з іншими інструментами та технологіями та ціни.

Таблиця 2.2 – Порівняння фремоворків

Критерій	Visual Studio	JetBrains Rider	Eclipse
Підтримувані платформи	Windows, mac OS	Windows, macOS, Linux	Windows, macOS, Linux
Основна мова	C#	C#	C#
IntelliSense	Так	Так	Так
Налагоджувач	Так	Так	Так
Рефакторинг коду	Так	Так	Так
Навігація по коду	Так	Так	Так
Інтеграція модульного тестування	Так(вбудована підтримка MSTest, NUnit, xUnit)	Так(вбудована підтримка MSTest, NUnit, xUnit)	Так(з такими плагінами, як NUnit, xUnit)
Інтеграція контролю версій	Так (вбудована підтримка Git, TFVC)	Так (вбудована підтримка Git)	Так (з такими плагінами, як EGit)
Екосистема плагінів	Vast (Visual Studio Marketplace)	Moderate (репозиторій плагінів JetBrains)	Vast (Eclipse Marketplace)
Уявлення	Відмінно	Відмінно	Помірний
Споживання ресурсів	Висока (особливо на вікнах)	Помірний (відносно легкий)	Помірний
Час запуску	Довгий	Помірний	Помірний
Підтримка спільноти	Великий	Великий	Великий
Простота використання	Підходить для початківців	Підходить для початківців	Підходить для початківців
Інтеграція Azure	Відмінно	Обмежено (Azure Toolkit for Rider)	Обмежений (Azure Toolkit для Eclipse)
Кросплатформна Сумісність	Обмежено (переважно в Windows)	Так	Так

Серед критеріїв порівняння будемо розглядати наступні:

- підтримувані платформи – ОС, на яких може бути встановлена і використана середовище IDE;
- основна мова – основна мова програмування, для якої оптимізовано IDE;

- IntelliSense – функція завершення коду в IDE, яка надає контекстно-залежні пропозиції та автоматичне завершення для елементів коду;
- Debugger – вбудований інструмент налагоджувача для виявлення та виправлення проблем у коді;
- рефакторинг коду – інструменти для реструктуризації існуючого коду без зміни його зовнішньої поведінки;
- навігація по коду – функції для зручної навігації по коду, такі як перехід до визначення, пошук всіх посилань і перехід до реалізації;
- інтеграція модульного тестування – вбудована підтримка популярних платформ модульного тестування, таких як MSTest, NUnit та xUnit;
- інтеграція з системою контролю версій – інтеграція з системами контролю версій, такими як Git або Team Foundation version Control (TFVC);
- екосистема плагінів – доступність плагінів та розширень для розширення функціональності IDE;
- продуктивність – оперативність і швидкість роботи IDE при виконанні таких завдань, як редагування коду, налагодження та збірка;
- споживання ресурсів – кількість системних ресурсів (процесор, пам'ять), споживаних IDE під час використання;
- час запуску – час, необхідний IDE для запуску та готовності до використання;
- підтримка спільноти – розмір та рівень активності спільноти розробників, пов'язаної з IDE;
- простота використання – зручність для користувача і інтуїтивність інтерфейсу IDE і робочого процесу;
- інтеграція з Azure – інтеграція зі службами Microsoft Azure для хмарної розробки та розгортання;

– сумісність між платформами – можливість IDE працювати в декількох операційних системах.

У цій таблиці представлено всебічний огляд сильних сторін та характеристик кожної зі сторін, що дозволяє розробникам приймати обґрунтовані рішення на основі своїх уподобань, вимог проекту та уподобань робочого процесу.

2.3 Структурна схема розробки

Загальна схема системи за обраною темою на рис. 2.1.

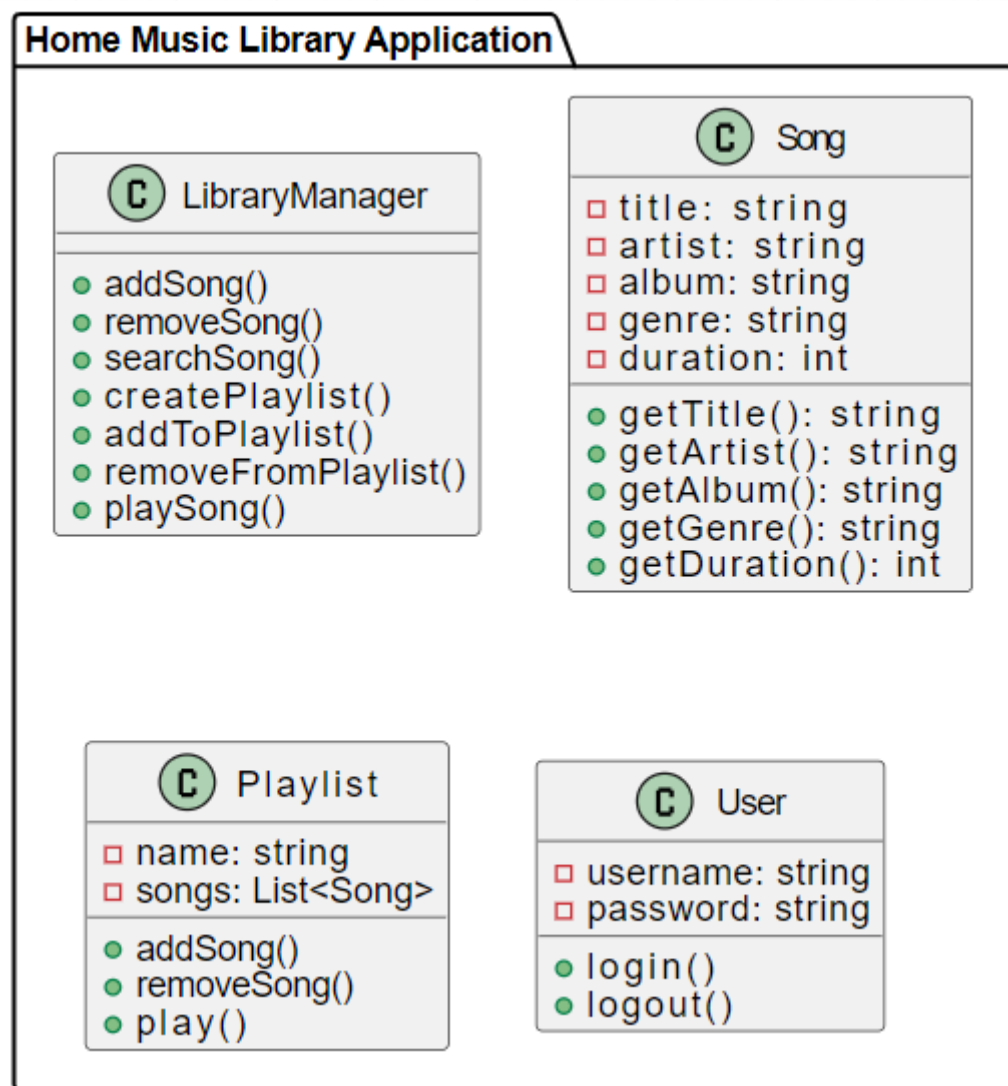


Рисунок 2.1 – Загальна схема системи за обраною темою

Ця структурна схема описує основні компоненти програми, включаючи клас `LibraryManager`, відповідальний за управління піснями та списками відтворення, клас `Song`, що представляє окремі пісні в бібліотеці, клас `Playlist` для організації пісень у списки відтворення та клас `User` для автентифікації користувача та управління сеансами.

Клас `LibraryManager` надає методи додавання, видалення, пошуку та відтворення пісень, а також створення та керування списками відтворення. Клас пісні містить інформацію про окремі пісні, такі як назва, виконавець, альбом, жанр та тривалість. Клас `Playlist` - це колекція пісень, організована під певною назвою, з методами додавання, видалення та відтворення пісень у списку відтворення. Нарешті, клас `User` відповідає за автентифікацію користувача та управління сеансами, а також за методи входу та виходу з системи.

2.4 Висновки за розділом 2

Другий розділ пояснювальної записки до роботи було присвячено висвітленню матеріалів та методів, що буде використано для програмної реалізації застосунку для впорядкування домашньої фонотеки. Було досліджено, обрано та аргументовано технологію розробки, мову програмування у зв'язці із програмними надбудовами, та патерну структури застосунку.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Проєктування моделей системи

Unified Modeling Language (UML) служить потужним інструментом для моделювання додатків у процесі розробки, пропонуючи безліч переваг, які покращують взаємодію, полегшують розуміння та спрощують життєвий цикл розробки [21], [22].

Візуалізація структури та поведінки системи. UML забезпечує стандартизоване позначення для візуального представлення структури та поведінки систем, що полегшує розуміння та передачу складних концепцій. За допомогою різних діаграм, таких як діаграми класів, діаграми дій та діаграми послідовностей, розробники можуть візуалізувати різні аспекти програми, включаючи її компоненти, взаємозв'язки, взаємодії та робочі процеси [21], [22].

Чітка взаємодія між зацікавленими сторонами. Діаграми UML служать загальною мовою, що полегшує спілкування між зацікавленими сторонами, залученими до процесу розробки, включаючи розробників, дизайнерів, тестувальників, менеджерів проєктів та клієнтів. Використовуючи діаграми UML для вираження вимог, дизайнерських рішень та поведінки системи, зацікавлені сторони можуть ефективно передавати свої ідеї, уточнювати очікування та забезпечувати узгодженість протягом усього життєвого циклу розвитку [21], [22].

Виявлення та аналіз вимог. На етапі виявлення та аналізу вимог UML може допомогти зафіксувати та змодельовати функціональні та нефункціональні вимоги програми. Діаграми випадків використання дозволяють зацікавленим сторонам визначити та візуалізувати функціональність системи з точки зору учасників та випадків використання, тоді як діаграми дій допомагають відобразити послідовність дій та бізнес-процесів [21], [22].

Проектування та архітектурне моделювання. UML служить потужним інструментом для проектування та моделювання архітектури програмних систем. Діаграми класів дозволяють розробникам визначати структуру системи, включаючи класи, атрибути, методи та їх взаємозв'язки. Діаграми компонентів і пакетів допомагають організувати і представити модульну структуру системи, полегшуючи розробку на основі компонентів і архітектурно-дизайнерські рішення [21], [22].

Поведінкове моделювання та моделювання. UML надає різні діаграми для моделювання динамічної поведінки систем, дозволяючи розробникам моделювати та аналізувати поведінку системи в різних сценаріях. Діаграми стану відображають життєвий цикл об'єктів та переходи між станами, тоді як діаграми послідовностей ілюструють взаємодію між об'єктами з часом, допомагаючи розробникам зрозуміти поведінку системи та виявити потенційні проблеми [21], [22].

Генерація та реалізація коду. Інструменти UML часто пропонують функції для генерації коду на основі діаграм UML, що дозволяє розробникам автоматично перетворювати моделі дизайну у виконуваний код. Це сприяє узгодженості етапів проектування і реалізації, скорочує витрати на ручне кодування і мінімізує ризик помилок і невідповідностей [21], [22].

Управління документацією та знаннями. Діаграми UML служать цінними артефактами документації, які відображають проектні рішення, архітектуру та поведінку програми. Вони забезпечують структуроване та наочне представлення системи, яке може бути використано для залучення нових членів команди, полегшення передачі знань та ведення точного обліку еволюції проекту з часом [21], [22].

Ітеративна розробка та доопрацювання. UML підтримує ітеративні та інкрементальні процеси розробки, дозволяючи розробникам моделювати, візуалізувати та вдосконалювати системні вимоги та проектувати ітеративно. Розробники можуть створювати та вдосконалювати UML-діаграми спільно,

враховуючи зворотний зв'язок, вносячи корективи та постійно вдосконалюючи дизайн та реалізацію програми протягом усього її життєвого циклу [21], [22].

Тестування та валідація. UML-діаграми можуть допомогти в тестуванні та валідації програмних систем, забезпечуючи чітке розуміння поведінки та взаємодій системи. Тестувальники можуть використовувати UML-діаграми для створення тестових кейсів, перевірки функціональності системи та забезпечення відповідності впровадженої системи заданим вимогам і проєктним обмеженням [21], [22].

Інтеграція з інструментами та процесами розробки. UML легко інтегрується з різними інструментами та процесами розробки, включаючи системи контролю версій, системи відстеження проблем та IDE. Розробники можуть використовувати інструменти UML для співпраці, управління артефактами проєкту, відстеження змін та інтеграції моделей UML з іншими артефактами розробки та документацією [21], [22].

Таким чином, UML служить цінною мовою моделювання для розробки програмного забезпечення, пропонуючи стандартизовані позначення для візуалізації, обміну інформацією та управління складністю програмних систем. Використовуючи UML протягом усього процесу розробки, команди можуть покращити взаємодію, покращити взаєморозуміння та, зрештою, створити високоякісне програмне забезпечення, яке відповідає потребам зацікавлених сторін [21], [22].

Діаграма класів (рис. 3.1) ілюструє основні класи та їх взаємозв'язки в застосунку:

- `LibraryManager` – відповідає за управління бібліотекою, включаючи додавання, видалення та пошук пісень, створення списків відтворення та відтворення композицій;

- `Song` – представляє окрему пісню в бібліотеці з такими атрибутами, як назва, виконавець, альбом, жанр та тривалість;

- Playlist – представляє колекцію пісень, організовану під певною назвою, із способами додавання, видалення та відтворення пісень у списку відтворення;
- User – представляє Користувача програми зі способами входу та виходу з системи. Кожен користувач пов'язаний з екземпляром LibraryManager, щоб керувати своєю особистою музичною бібліотекою.

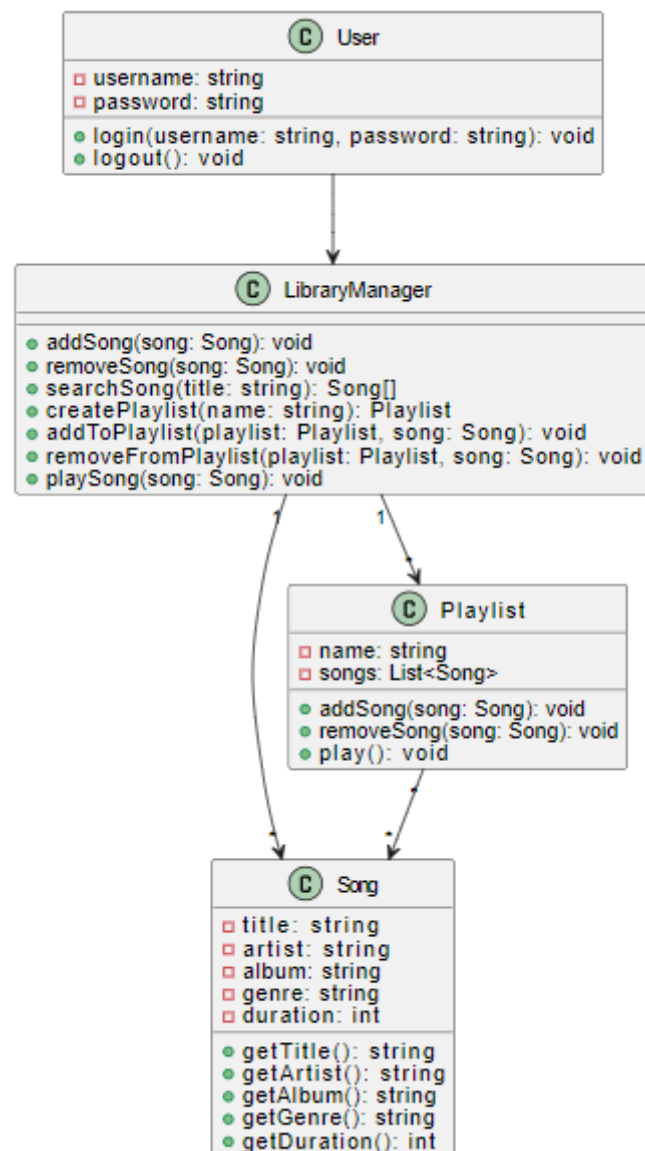


Рисунок 3.1 – Діаграма класів

Відносини між класами відображаються за допомогою асоціацій, що вказують, як екземпляри одного класу пов'язані з екземплярами іншого класу. Наприклад, `LibraryManager` має асоціації з `Song`, `Playlist` та `User`, що вказує на те, що він взаємодіє з екземплярами цих класів для виконання різних операцій у програмі. Аналогічно, список відтворення пов'язаний з `Song`, що вказує на те, що `Playlist` містить кілька пісень.

Діаграма пакетів (рис. 3.2) ілюструє організацію програми у вигляді трьох основних пакетів:

- **Managers**: містить клас `LibraryManager`, відповідальний за управління домашньою музичною бібліотекою;
- **Entities**: містить класи `Song` та `Playlist`, що представляють об'єкти музичної бібліотеки;
- **Users**: містить клас `User`, що представляє користувачів програми.

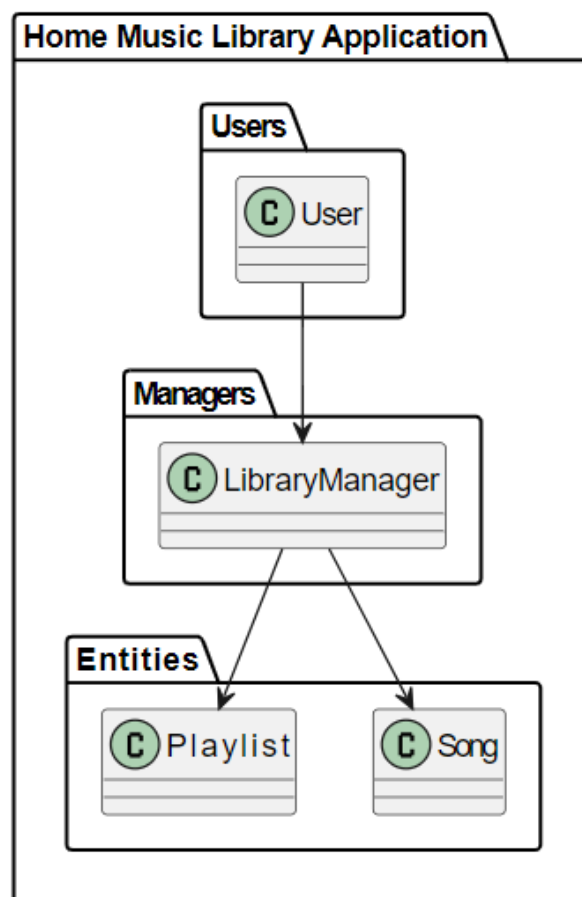


Рисунок 3.2 – Діаграма пакетів

Діаграма показує взаємозв'язки між пакетами, які показують, що LibraryManager взаємодіє як з об'єктами Song, так і з об'єктами Playlist, тоді як користувач взаємодіє з LibraryManager для управління музичною бібліотекою.

UML діаграма прецедентів (рис. 3.3) ілюструє різні дії, які користувач може виконувати в програмі для організації домашньої музичної бібліотеки:

- додати пісню: дозволяє користувачеві додати нову пісню до музичної бібліотеки;

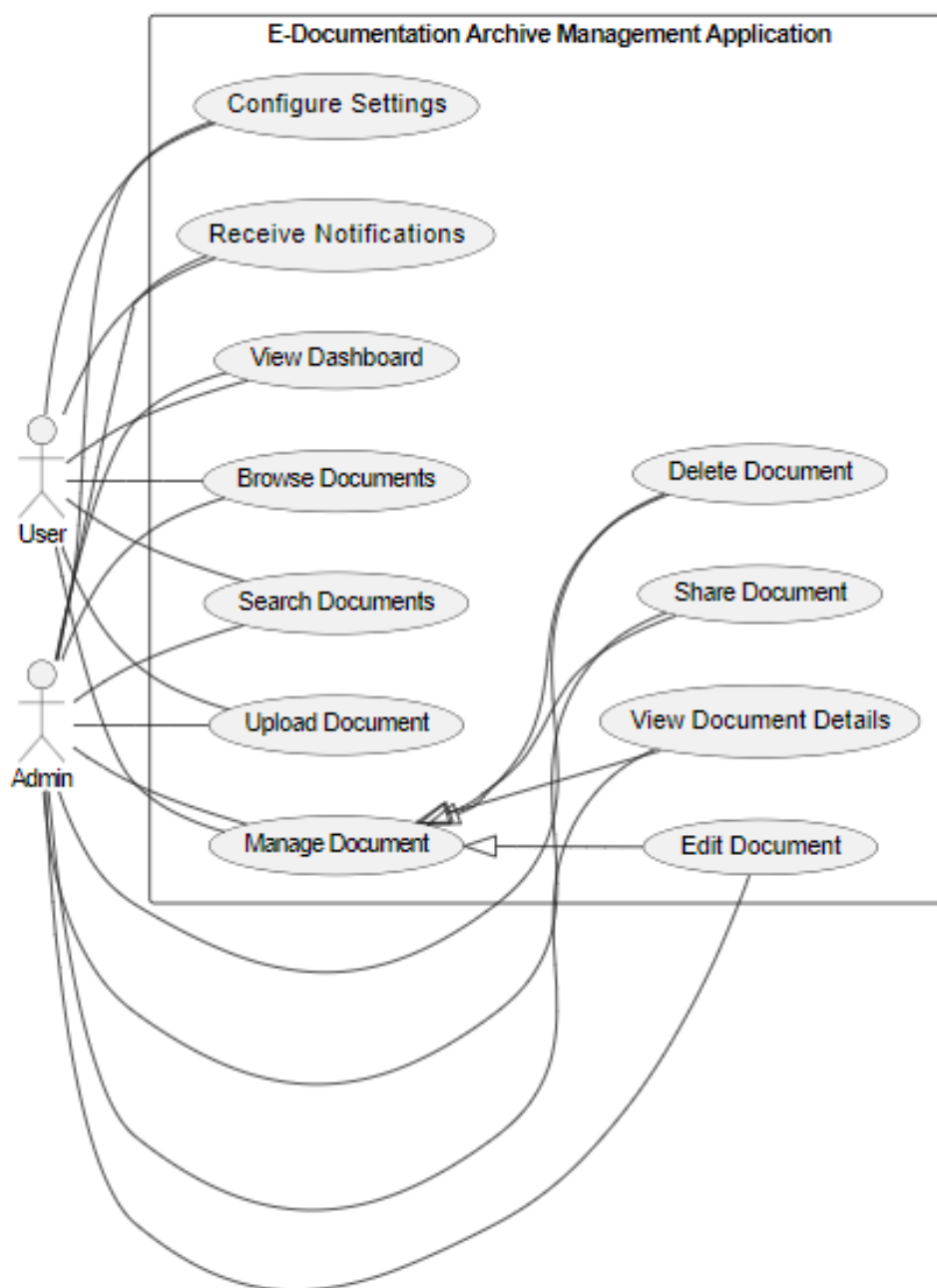


Рисунок 3.3 – Діаграма прецедентів

- видалити пісню: дозволяє користувачеві видалити пісню з музичної бібліотеки;
- пошук пісні: дозволяє користувачеві шукати пісню в музичній бібліотеці;
- створити список відтворення: дозволяє користувачеві створити новий список відтворення;
- додати до списку відтворення: дозволяє користувачеві додати пісню до існуючого списку відтворення;
- видалити зі списку відтворення: дозволяє користувачеві видалити пісню зі списку відтворення;
- відтворення пісні: дозволяє користувачеві відтворити вибрану пісню з музичної бібліотеки.

На діаграмі показано взаємодію між Користувачем та різними випадками використання, наданими додатком, та показано функціональність, доступну користувачеві для організації та управління домашньою музичною бібліотекою.

Ця діаграма станів (рис. 3.4) показує різні стани та переходи в програмі організації домашньої музичної бібліотеки:

- початковий стан – LoggedOut, при якому користувач не входить в систему;
- після успішного входу користувач переходить у стан LoggedIn;
- стан LoggedIn забезпечує доступ до головного меню, де користувач може переглядати бібліотеку, шукати пісні, створювати списки відтворення та виходити з системи;
- кожен пункт головного меню веде до іншого стану або дії: ViewingLibrary – режим перегляду бібліотеки дозволяє користувачеві переглядати пісні в бібліотеці; Searching – режим пошуку дозволяє користувачеві виконувати пошук пісень; ViewingLibrary – режим перегляду бібліотеки дозволяє користувачеві переглядати пісні в бібліотеці; Searching – режим пошуку дозволяє користувачеві виконувати пошук пісень;

- стан CreatingPlaylist допомагає користувачеві в процесі створення нового списку відтворення;
- стан PlayingSong дозволяє користувачеві відтворювати вибрану композицію з параметрами відтворення, паузи та зупинки відтворення;
- переходи між станами відбуваються залежно від дій користувача, таких як вхід, вибір параметрів меню, перехід назад або виконання елементів керування відтворенням.

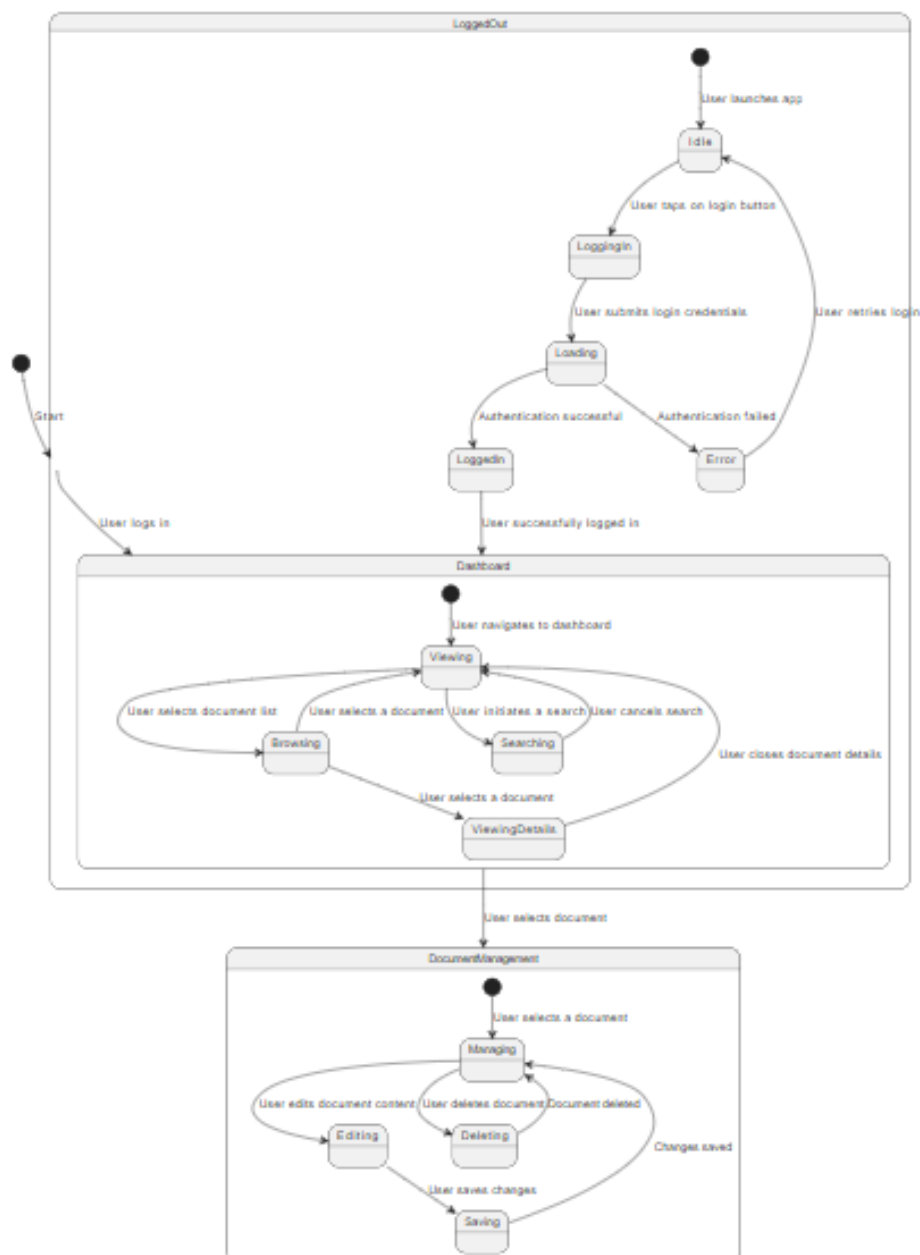


Рисунок 3.4 – Діаграма станів системи

Ця діаграма станів забезпечує візуальне представлення поведінки програми та її роботи, ілюструючи можливі стани та переходи, які користувач може відчувати під час взаємодії з домашньою музичною бібліотекою.

Діаграма діяльностей (рис. 3.5) ілюструє робочий процес програми для організації домашньої музичної бібліотеки:

- процес починається з входу користувача в застосунок.
- якщо користувач успішно виконав вхід, йому будуть представлені параметри головного меню;

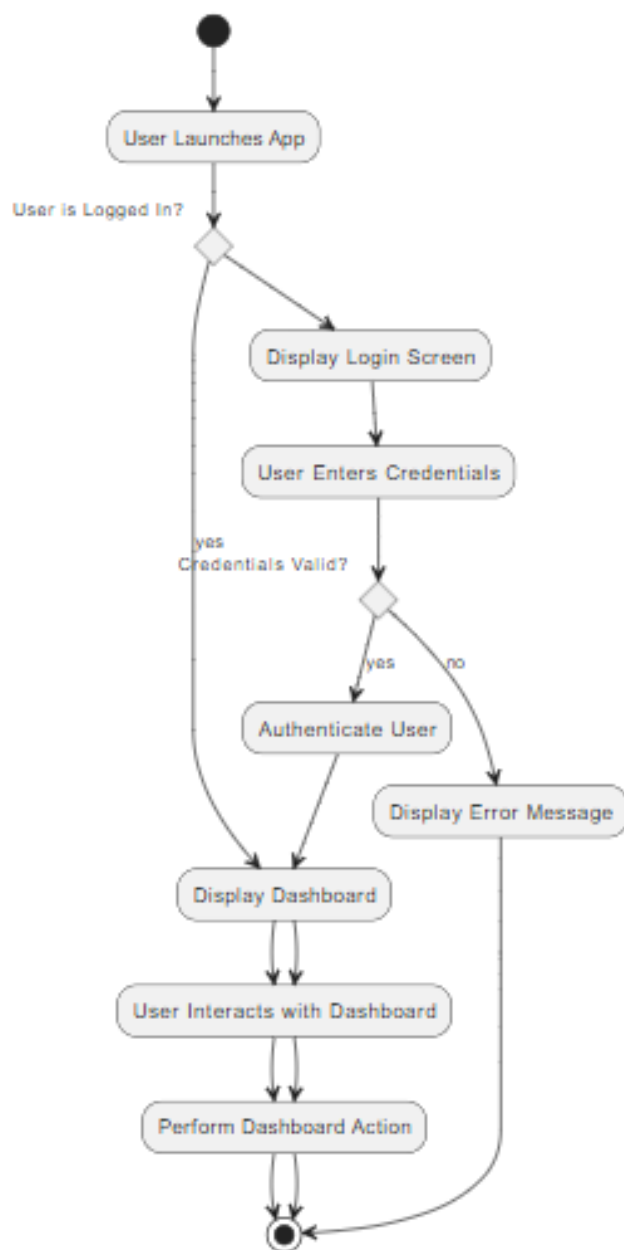


Рисунок 3.5 – Діаграма діяльності

– залежно від обраної опції додаток виконує різні дії: якщо користувач вибирає «Перегляд бібліотеки», програма відображає вміст бібліотеки; якщо користувач вибирає «Пошук», йому пропонується ввести пошуковий термін, і програма шукає відповідні пісні в бібліотеці, відображаючи результати пошуку; якщо користувач вибере «Створити список відтворення», він зможе створити новий список відтворення, додати до нього пісні та зберегти список відтворення; якщо користувач вибере «Вихід», він вийде з програми; якщо Користувач не увійшов у систему, програма відобразить екран входу.

Ця діаграма дій забезпечує наочне представлення робочого процесу програми, ілюструючи послідовність кроків, пов'язаних із взаємодією Користувача з домашньою музичною бібліотекою.

3.2 Проектування інтерфейсу

Розробка графічного інтерфейсу користувача (graphical user interface – GUI) для музичного застосунку вимагає ретельного розгляду різних функцій, щоб забезпечити безперебійну та приємну роботу користувача [23], [24].

Інтуїтивно зрозуміла навігація. Слід використовувати зрозумілі та інтуїтивно зрозумілі елементи навігації, такі як меню, вкладки або кнопки, щоб користувачі могли легко переглядати різні розділи програми, такі як списки відтворення, альбоми, виконавці та жанри [23], [24].

Елементи керування відтворенням. Слід увімкнути такі елементи керування відтворенням, як відтворення, пауза, пропуск, перемотування назад і регулювання гучності, що дозволить користувачам легко керувати відтворенням музичних композицій. Ці елементи управління повинні бути легко доступні і розташовуватися на видному місці в інтерфейсі [23], [24].

Функціональність пошуку. Слід реалізувати панель пошуку або піктограму пошуку, яка дозволяє користувачам швидко шукати певні пісні, альбоми, виконавців або списки відтворення в музичній бібліотеці.

Використовувати фільтри або розширені параметри пошуку для уточнення результатів пошуку та підвищення точності [23], [24].

Візуальний зворотний зв'язок. Використовувати візуальний зворотний зв'язок, такий як анімація, індикатори прогресу або зміни кольору, щоб відобразити стан дій, які виконує користувач, наприклад, завантаження музичних доріжок, буферизація або завершення відтворення [23], [24].

Відображення обкладинки альбому на видному місці в інтерфейсі для підвищення візуальної привабливості програми та надання користувачам візуальних підказок про відтворювану в даний момент пісню або альбом.

Параметри налаштування –: дозволяють користувачам налаштовувати зовнішній вигляд і компонування інтерфейсу, наприклад, змінювати теми оформлення, налаштовувати колірні схеми або переставляти елементи відповідно до своїх уподобань [23], [24].

Керування списками відтворення – дозволяє користувачам створювати, редагувати та керувати списками відтворення, включаючи можливість додавати або видаляти пісні, змінювати порядок композицій та перейменовувати списки відтворення. Надає функції для організації списків відтворення за категоріями або папками для кращої організації [23], [24].

Доступність – слід переконатися, що GUI доступний для користувачів з обмеженими можливостями, включивши такі функції, як сумісність із зчитувачем екрана, комбінації клавіш та налаштовані розміри шрифту або параметри контрастності [23], [24].

Включивши ці функції в GUI музичного застосунку, розробники можуть створити зручний і захоплюючий інтерфейс, який підвищує задоволення від прослуховування музики і сприяє залученню і задоволенню користувачів.

На рис. 3.6 наведено концепт головної сторінки, а на рис. 3.7 зображено концепт відкриття однієї композиції у майбутньому застосунку.

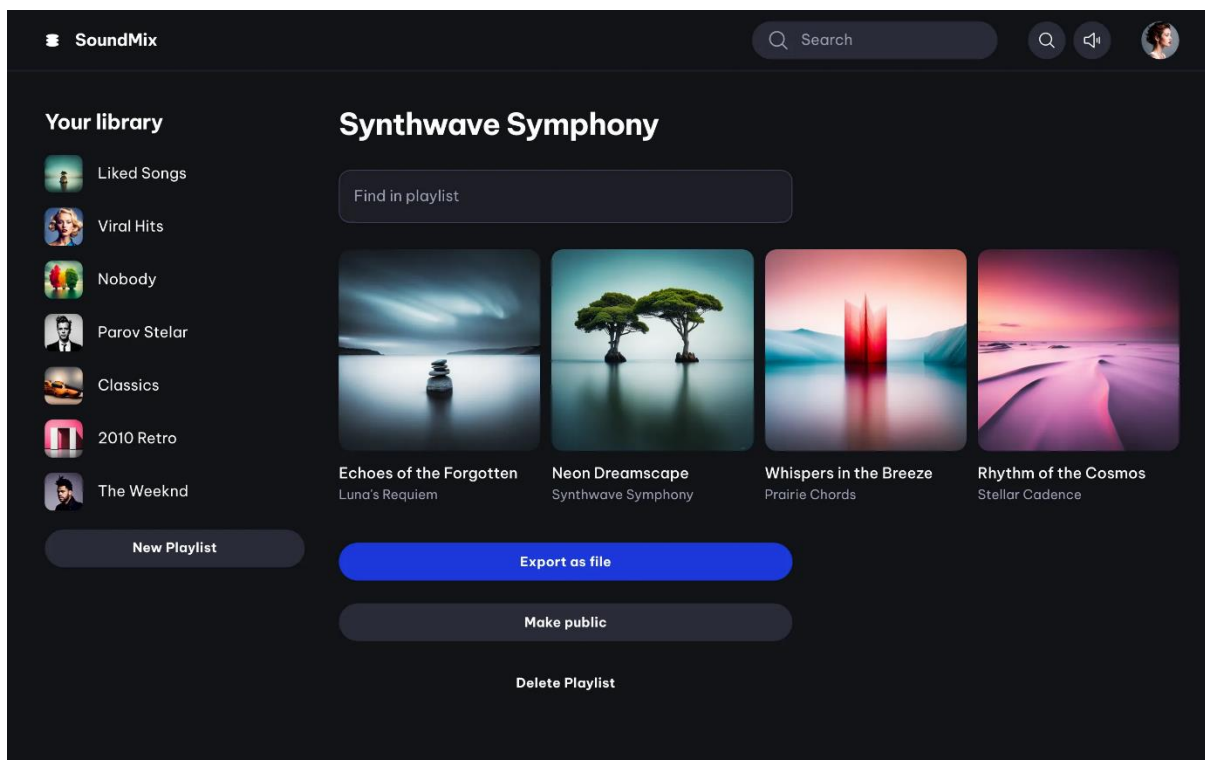


Рисунок 3.6 – Концепт головної сторінки

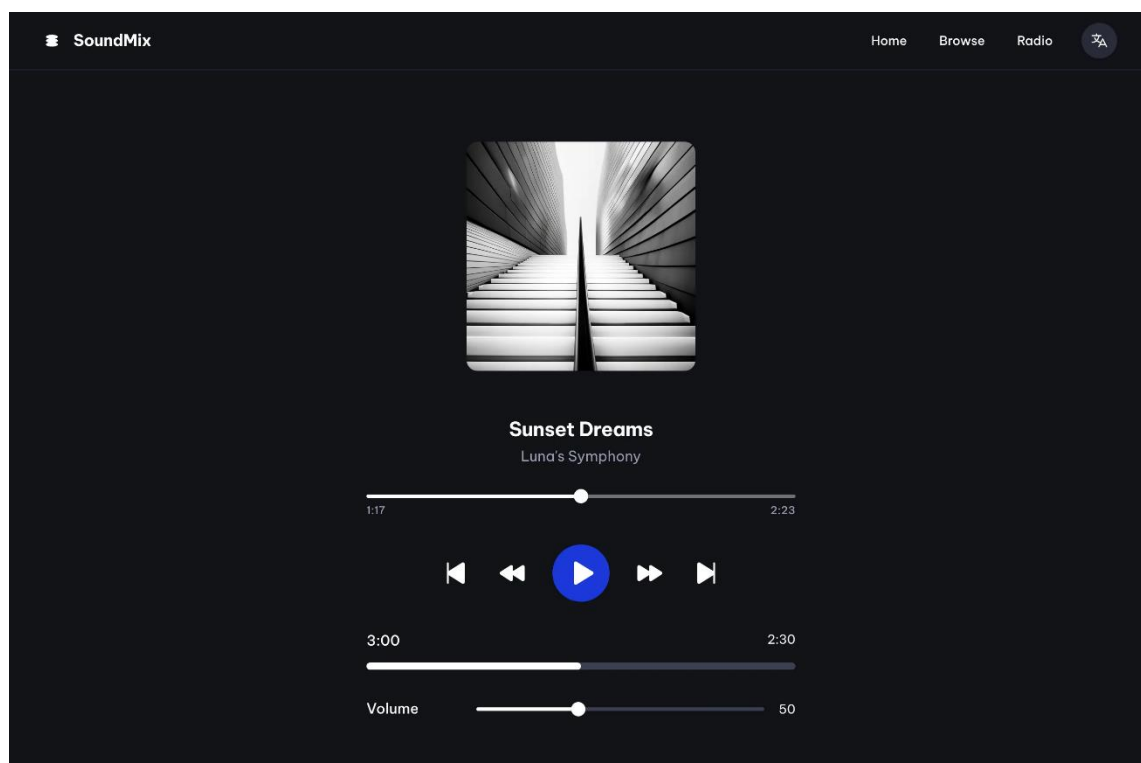


Рисунок 3.7 – Концепт відкриття однієї композиції

На рис. 3.8 наведено концепт сторінки бібліотеки.

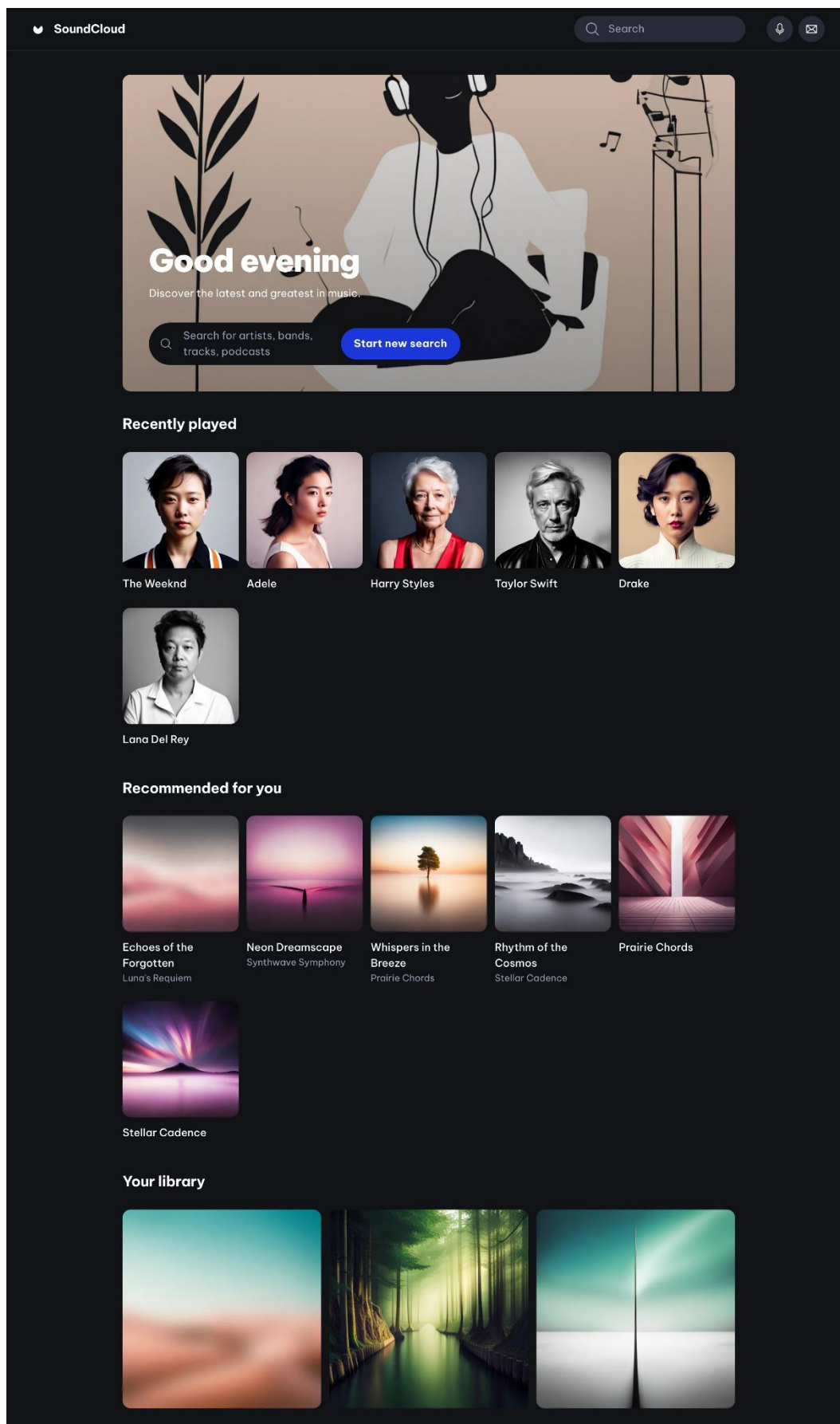


Рисунок 3.8 – Концепт сторінки бібліотеки

На рис. 3.9 наведено вікно оновлення плейлисту.

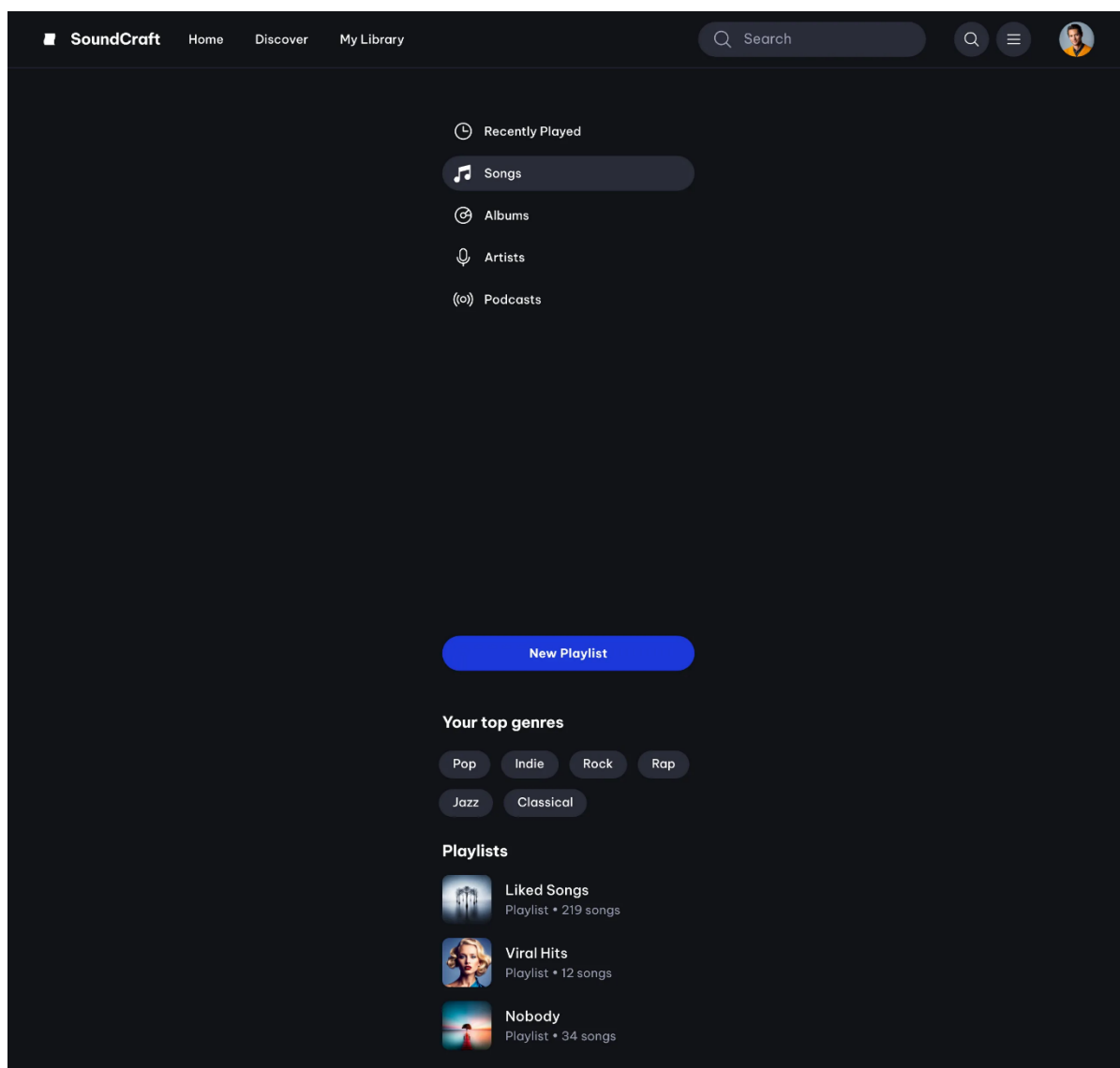


Рисунок 3.9 – Вікно оновлення плейлисту

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Розробка ПЗ для організації домашньої музичної бібліотеки є актуальним завданням у зв'язку з ускладненням управління цифровими музичними колекціями, зростаючим попитом на персоналізований музичний досвід і потребою в ефективних механізмах організації, пошуку і виявлення інформації. Вирішуючи ці завдання і пропонуючи інноваційні функції і можливості, розробники можуть створити цінний інструмент, який поліпшить якість прослуховування музики користувачами і задовольнить їх зростаючі потреби і очікування.

Отже актуальною задачею залишається програмна реалізація застосунку для впорядкування домашньої фонотеки.

Система має реалізовувати наступні функціональні вимоги:

- аутентифікація користувачів і управління профілями;
- управління музичною бібліотекою;
- редагування метаданих;
- відтворення та потокове передавання;
- пошук і навігація;
- налаштування та персоналізація;
- резервне копіювання та синхронізація;
- доступність і локалізація.

4.2 Вимоги до інтерфейсу

Інтерфейс системи повинен відрізнятися лаконічністю, інтуїтивною зрозумілістю та зручністю для користувачів. Дизайн має бути адаптивним для зручності роботи на будь-якому дисплеї.

4.3 Умови виконання програми

Апаратні вимоги до сервера:

- центральний процесор – від 8 фізичних ядер та 16 потоків з тактовою частотою від 4 GHz;
- оперативна пам'ять формату DDR4, з тактовою частотою від 2600 MHz та об'ємом від 128 Гб;
- накопичувач (формату HDD або SSD) з вільним простором від 10 Гб (для розгортання усіх дамів музичних баз даних).

Програмні вимоги до сервера:

- розгорнута серверна ОС, наприклад: Windows Server, Linux Server або macOS Server;
- розгорнута SQL-сумісна СКБД.

Апаратні вимоги до клієнта:

- центральний процесор – від 2 фізичних ядер та 2 потоків з тактовою частотою від 1,7 GHz (з можливістю збільшення під час навантаження);
- оперативна пам'ять об'ємом від 4 Гб;
- накопичувач (формату HDD або SSD) з вільним простором від 0,5 Гб.

Програмні вимоги до сервера:

- наявність веббраузера актуальної версії.

4.4 Методика та результати тестування

Під час тестування було обрано стратегію поетапного виконання за сценарієм основного алгоритму користування застосунком..

Тож на рис. 4.1 наведено головну сторінку застосунку.

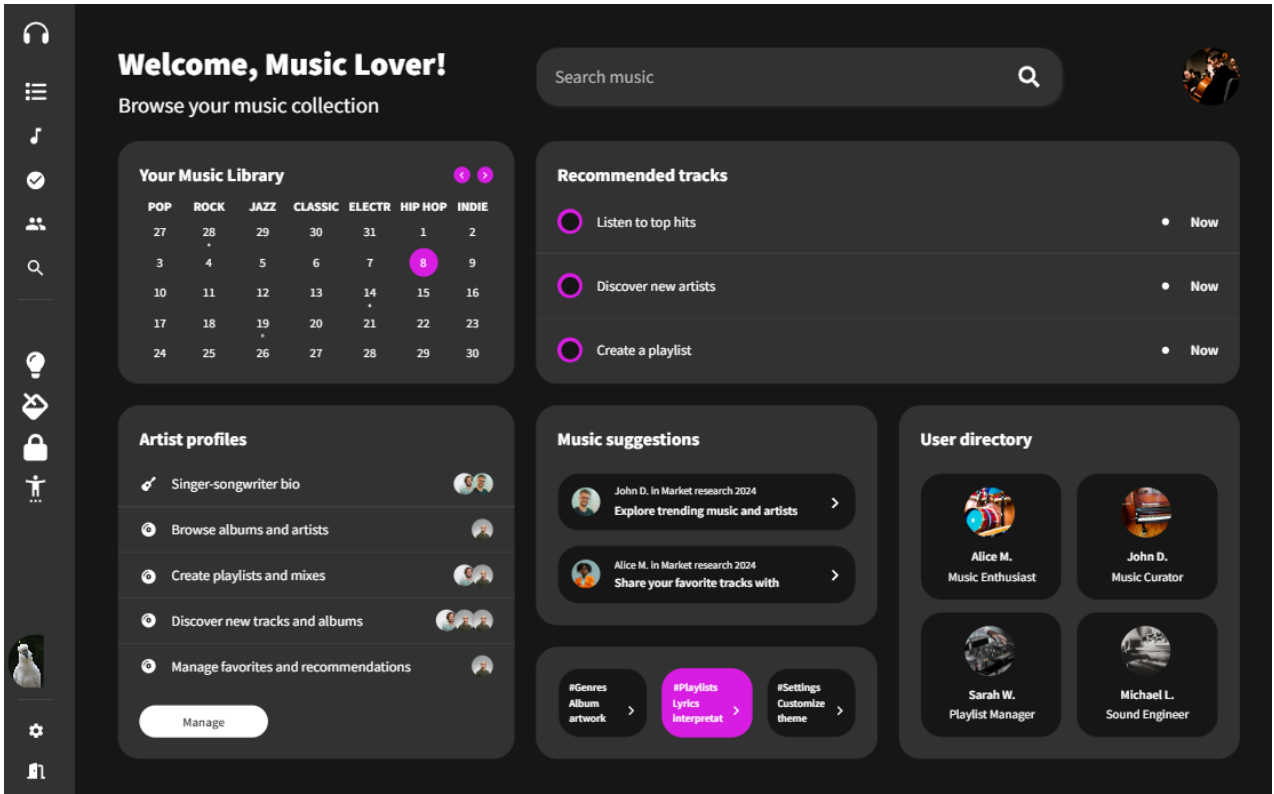


Рисунок 4.1 – Головний екран (привітання) застосунку

На рис. 4.2 наведено приклад представлення фонотеки користувача.

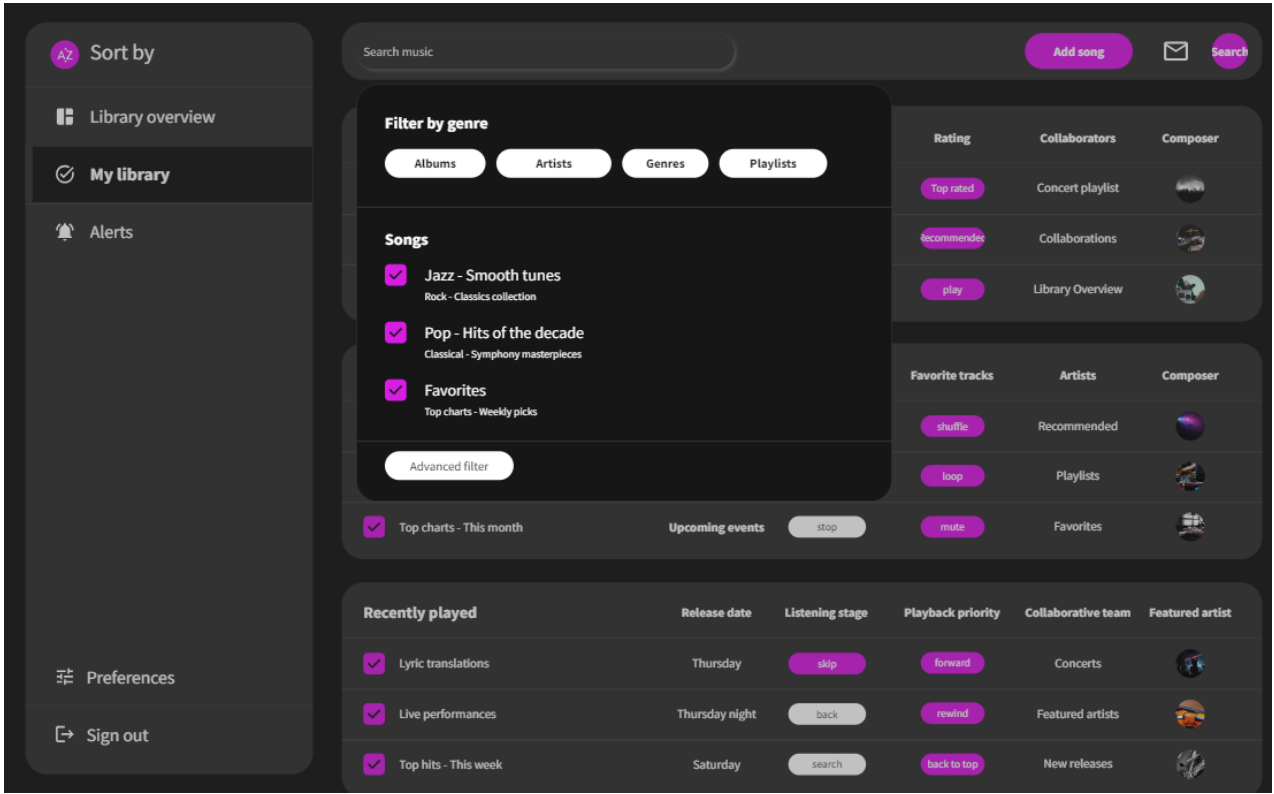


Рисунок 4.2 – Представлення фонотеки користувача

На рис. 4.3 наведено відображення плейлистів користувача.

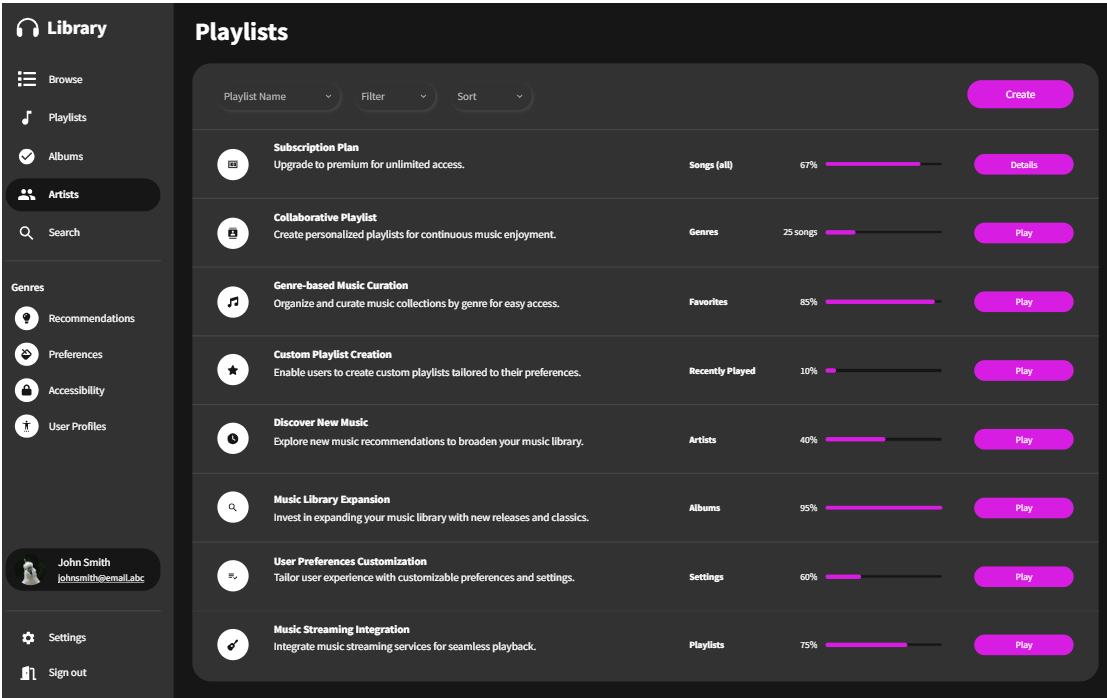


Рисунок 4.3 – Плейлисти користувача

На рис. 4.4 наведено відкриття та програвання плейлисту.

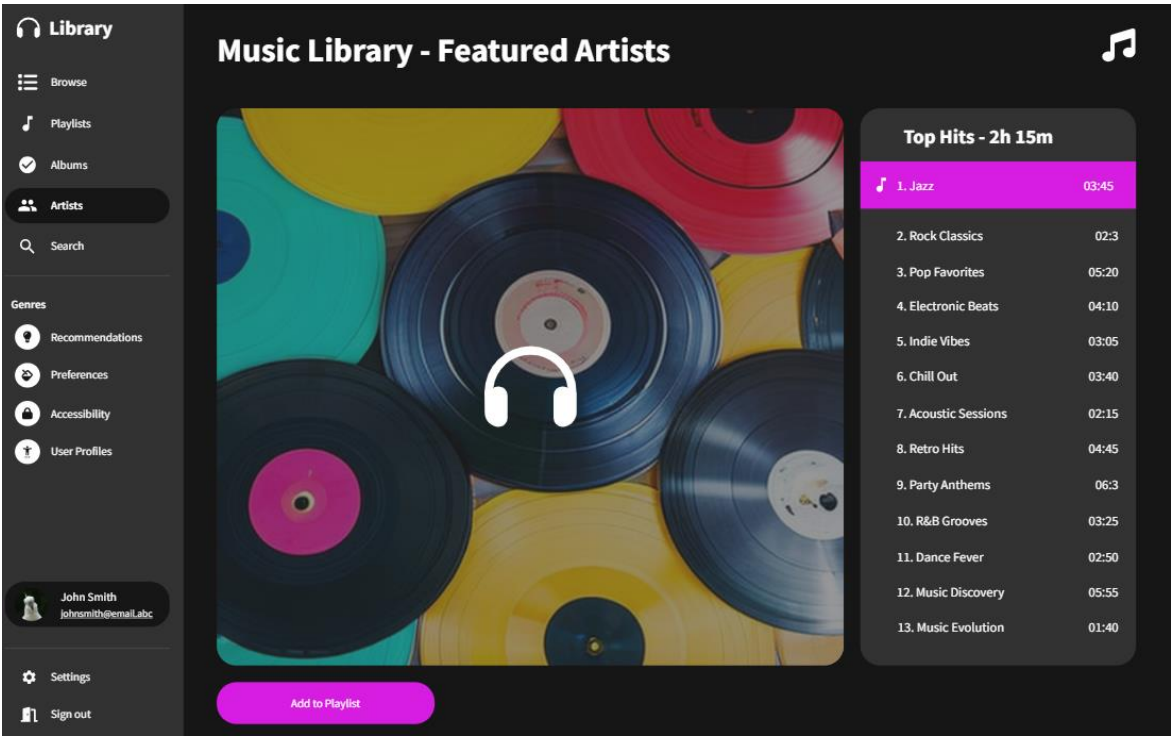


Рисунок 4.4 – Відображення плейлисту

На рис. 4.5 наведено вікно налаштувань даних про користувача.

Library

- Browse
- Playlists
- Albums
- Artists**
- Search

Genres

- Recommendations
- Preferences
- Accessibility
- User Profiles

John Smith
johnsmith@email.abc

Settings

Sign out

Profile Settings

Change profile
Display name

Profile Information

Display name
Lorem Ipsum
JohnDoe

Email
johndoe@email.com

Phone number
+00 111 222 333 444

Date of birth
01/01/1990

Country of origin
USA

Preferred languages
English, Spanish, Italian

Address
123 Music Street, City, Country

Emergency contact
+11 222 333 444

Favorite genres
Rock, Jazz, Electronic

Music preferences
Shuffle, Repeat, Equalizer

Music Documents

Playlist.pdf	View
Lyrics.docx	View
Concert Tickets.pdf	View
Equipment Manual	View

Identification Documents

Driver's License	Edit
Passport	Edit
Utility Bill	Edit

Upload new document

Рисунок 4.5 – Вікно персоналізації налаштувань

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було виконано програмну реалізацію застосунку для впорядкування домашньої фонотеки. Для цього було виконано ряд завдань:

- виконано дослідження проблемної області накопичення та впорядкування фонотеки;
- виконано дослідження переваг та недоліків існуючих аналогів для впорядкування фонотеки;
- виконано вибір технічних засобів реалізації запропонованого застосунку та обґрунтувати його;
- спроектовано архітектуру нового застосунку;
- програмно реалізовано новий застосунок;
- проведено тестування розробки;
- сформовано висновки на основі виконаної роботи.

Розроблений застосунок був реалізований із використанням: мови програмування C# та фреймворку .Net, засоби моделювання UML.

Застосунок було протестовано відповідно до сценарію його використання. Тестування засвідчило коректність роботи.

Застосунок реалізує такі функціональні вимоги:

- аутентифікація користувачів і управління профілями;
- управління музичною бібліотекою;
- редагування метаданих;
- відтворення та потокове передавання;
- пошук і навігація;
- налаштування та персоналізація;
- резервне копіювання та синхронізація;
- доступність і локалізація.

В результаті розробки реалізовано систему, що відповідає всім вимогам визначеним на початку розробки у вигляді функціональних вимог. В майбутньому можливо вдосконалити систему додавши реалізацію еквалайзера для налаштувань відтворення композицій.

Усі завдання дипломної кваліфікаційної роботи бакалавра повністю виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Everything You Need to Build Your Very Own Digital Music Collection / [Electronic resource] – <https://pitchfork.com/features/article/everything-you-need-to-build-your-own-digital-music-collection/>
2. How to build your own digital music library / [Electronic resource] – <https://www.whathifi.com/advice/how-to-build-your-own-digital-music-library>
3. Music Collection: Home / [Electronic resource] – <https://fredonia.libguides.com/music>
4. How to build a local digital music library / [Electronic resource] – <https://www.soundguys.com/how-to-build-local-digital-music-library-68789/>
5. How to stream your personal music library wherever you are / [Electronic resource] – <https://thenextweb.com/news/how-to-stream-your-personal-music-library-wherever-you-are>
6. The Ultimate Guide to Organizing Your Music Library / [Electronic resource] – <https://disctopia.com/the-ultimate-guide-to-organizing-your-music-library/>
7. Organize your Spotify music collection by any of a wide range of musical attributes including genre, mood, decade of release and more / [Electronic resource] – <http://organizeyourmusic.playlistmachinery.com/>
8. The most effective way to organize your music/artists? / [Electronic resource] – https://www.reddit.com/r/LetsTalkMusic/comments/145ypyx/the_most_effective_way_to_organize_your/
9. Mp3tag / [Electronic resource] – <https://www.mp3tag.de/en/>
10. Mp3tag Development Build Status / [Electronic resource] – <https://community.mp3tag.de/t/mp3tag-development-build-status/455>
11. Kid3 Audio Tagger / [Electronic resource] – <https://kid3.kde.org/>
12. Kid3 Audio Tag Editor 3.6.0, an efficient tag editor / [Electronic resource] – <https://ubunlog.com/en/kid3-audio-tag-editor/>

13. MusicBrainz - the open music encyclopedia / [Electronic resource] – <https://musicbrainz.org/>
14. How to Get a MusicBrainz Page for Your Band / [Electronic resource] – <https://bandzoogle.com/blog/how-to-get-a-musicbrainz-page-for-your-band>
15. Functional and Nonfunctional Requirements: Specification and Types / [Electronic resource] – <https://www.altexsoft.com/blog/functional-and-non-functional-requirements-specification-and-types/>
16. Functional vs Non Functional Requirements / [Electronic resource] – <https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/>
17. Керівництво C# / [Electronic resource] – <https://learn.microsoft.com/dotnet/csharp/>
18. C# Початок роботи / [Electronic resource] – https://w3schoolsua.github.io/cs/cs_getstarted.html#gsc.tab=0
19. Visual Studio 2022 / [Electronic resource] – <https://visualstudio.microsoft.com/>
20. Welcome to Visual Studio / [Electronic resource] – https://education.launchcode.org/intro-to-programming-csharp/chapters/vs-setup/setup.html?utm_term=&utm_campaign=&utm_source=adwords&utm_medium=ppc&hsa_acc=4368208516&hsa_cam=17572421468&hsa_grp=135984040497&hsa_ad=605748003303&hsa_src=g&hsa_tgt=dsa-19959388920&hsa_kw=&hsa_mt=&hsa_net=adwords&hsa_ver=3&gad_source=1&gclid=CjwKCAjw9IayBhBJEiwAVuc3fogRui2Nkw55uYRzTbhXmuyzSFKAIXkww5qFDmEc5o81O2l0ukiLdhoCpv4QAvD_BwE
21. Як будувати UML-діаграми. Розбираємо три найпопулярніші варіанти / [Electronic resource] – <https://dou.ua/forums/topic/40575/>
22. Елементи UML / [Electronic resource] – <https://docs.kde.org/trunk5/uk/umbrello/umbrello/uml-elements.html>
23. Що таке дизайн графічного інтерфейсу користувача GUI? / [Electronic resource] – <https://training.qatestlab.com/blog/technical-articles/what-is-graphical-user-interface-design/>

24. Принципи вебдизайну інтерфейсу користувача / [Electronic resource] – <https://stfalcon.com/uk/blog/post/user-interface-web-design-principles>

ДОДАТОК А
Технічне завдання

Вступ

Даний застосунок присвячений для впорядкування домашньої музичної колекції (бібліотеки) – фонотеки.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмна реалізація застосунку для впорядкування домашньої фонотеки», затверджене наказом Національного університету «Запорізька політехніка» № 168 від 24 квітня 2024 р.

A.2 Основні вимоги до програми, що розробляється

Розробка ПЗ для організації домашньої музичної бібліотеки є актуальним завданням у зв'язку з ускладненням управління цифровими музичними колекціями, зростаючим попитом на персоналізований музичний досвід і потребою в ефективних механізмах організації, пошуку і виявлення інформації. Вирішуючи ці завдання і пропонуючи інноваційні функції і можливості, розробники можуть створити цінний інструмент, який поліпшить якість прослуховування музики користувачами і задовольнить їх зростаючі потреби і очікування.

Отже актуальною задачею залишається програмна реалізація застосунку для впорядкування домашньої фонотеки.

A.2.1 Вимоги до функціональних характеристик

– аутентифікація користувачів і управління профілями. Програма повинна дозволяти користувачам створювати облікові записи та безпечно

входити в систему, а також надавати функціональні можливості для управління профілями, такі як редагування інформації користувача і переваг;

- управління музичною бібліотекою: підтримка додавання, видалення та упорядкування музичних файлів; дозволяє користувачам імпортувати музику з різних джерел, включаючи локальні сховища, онлайн-потоківі послуги та зовнішні пристрої; можливість класифікувати музику за виконавцями, альбомами, жанрами, роком випуску та іншими метаданими; можливість створювати власні списки відтворення та організовувати музичні колекції на основі особистих уподобань;

- редагування метаданих: дозволяє користувачам редагувати метадані, пов'язані з музичними файлами, включаючи імена виконавців, назви альбомів, номери треків, жанри та обкладинки альбомів; надає можливості пакетного редагування для оновлення метаданих для декількох файлів одночасно; підтримує автоматичний пошук метаданих з онлайн-баз даних для спрощення процесу створення тегів;

- відтворення та потокове передавання: надає вбудований музичний плеєр для відтворення аудіофайлів у додатку; підтримує відтворення різних аудіоформатів, включаючи MP3, FLAC, WAV, AAC і OGG;

- інтеграція з популярними поточковими сервісами для забезпечення безперешкодного доступу до потокового музичного контенту;

- пошук і навігація: реалізована надійна функція пошуку, що дозволяє користувачам шукати музичні файли за ключовими словами, виконавцем, альбомом, жанром чи іншими критеріями; надає параметри фільтрації та сортування для полегшення перегляду та навігації по музичній бібліотеці; пропонує зручний інтерфейс з інтуїтивно зрозумілими елементами навігації і налаштованими видами;

- налаштування та персоналізація: дозволяє користувачам налаштовувати зовнішній вигляд програми, включаючи теми, кольорні схеми та параметри макета; надає налаштування для налаштування параметрів відтворення, таких як налаштування еквалайзера, швидкість відтворення та

підтримка різноманітних ефектів; підтримує користувальницькі теги, написи і рейтинги для спрощення персоналізованої організації та категоризації музичних файлів;

- резервне копіювання та синхронізація: пропонує функції резервного копіювання та синхронізації для забезпечення цілісності даних та доступності на декількох пристроях; дозволяє користувачам синхронізувати музичні бібліотеки між різними пристроями, включаючи комп'ютери, смартфони та планшети; надає можливість автоматичного резервного копіювання і синхронізації за розкладом, щоб звести до мінімуму ризик втрати даних;

- інтеграція із зовнішніми пристроями: підтримка інтеграції з зовнішніми пристроями, такими як USB-накопичувачі, Зовнішні жорсткі диски і пристрої мережевого зберігання даних; дозволяє користувачам отримувати доступ до музичних файлів, що зберігаються на зовнішніх пристроях, і керувати ними безпосередньо з програми;

- доступність і локалізація: забезпечення спеціальних можливостей, таких як підтримка читання з екрану, навігація з клавіатури і перетворення тексту в мову, для користувачів з обмеженими можливостями; підтримка декількох мов і варіантів локалізації для користувачів з різним мовним досвідом. Також необхідно забезпечити спеціальні можливості, такі як підтримка читання з екрану, навігація з клавіатури і перетворення тексту в мову, для користувачів з обмеженими можливостями; підтримка декількох мов і варіантів локалізації для користувачів з різним мовним досвідом.

A.2.2 Вимоги до інтерфейсу програми

Інтерфейс системи повинен відрізнятися лаконічністю, інтуїтивною зрозумілістю та зручністю для користувачів. Дизайн має бути адаптивним для зручності роботи на будь-якому дисплеї.

A.2.3 Вимоги до надійності

Забезпечення конфіденційності та безпеки даних забезпечується дотримання відповідних правил та рекомендацій щодо захисту даних користувачів та захисту інформації користувача.

A.2.4 Вимоги до складу та параметрів технічний засобів

Апаратні вимоги до сервера:

- центральний процесор – від 8 фізичних ядер та 16 потоків з тактовою частотою від 4 GHz;
- оперативна пам'ять формату DDR4, з тактовою частотою від 2600 MHz та об'ємом від 128 Гб;
- накопичувач (формату HDD або SSD) з вільним простором від 10 Гб (для розгортання усіх дамів музичних баз даних).

Програмні вимоги до сервера:

- розгорнута серверна ОС, наприклад: Windows Server, Linux Server або macOS Server;
- розгорнута SQL-сумісна СКБД.

Апаратні вимоги до клієнта:

- центральний процесор – від 2 фізичних ядер та 2 потоків з тактовою частотою від 1,7 GHz (з можливістю збільшення під час навантаження);
- оперативна пам'ять об'ємом від 4 Гб;
- накопичувач (формату HDD або SSD) з вільним простором від 0,5 Гб.

Програмні вимоги до сервера:

- наявність веббраузера актуальної версії.

ДОДАТОК Б
Опис програми

Б.1 Загальні відомості

Розробка ПЗ для організації домашньої музичної бібліотеки є актуальним завданням у зв'язку з ускладненням управління цифровими музичними колекціями, зростаючим попитом на персоналізований музичний досвід і потребою в ефективних механізмах організації, пошуку і виявлення інформації. Вирішуючи ці завдання і пропонуючи інноваційні функції і можливості, розробники можуть створити цінний інструмент, який поліпшить якість прослуховування музики користувачами і задовольнить їх зростаючі потреби і очікування.

Отже актуальною задачею залишається програмна реалізація застосунку для впорядкування домашньої фонотеки.

Б.2 Функціональне призначення

Система має реалізовувати наступні функціональні вимоги:

- аутентифікація користувачів і управління профілями. Програма повинна дозволяти користувачам створювати облікові записи та безпечно входити в систему, а також надавати функціональні можливості для управління профілями, такі як редагування інформації користувача і переваг;

- управління музичною бібліотекою: підтримка додавання, видалення та упорядкування музичних файлів; дозволяє користувачам імпортувати музику з різних джерел, включаючи локальні сховища, онлайн-потоківі послуги та зовнішні пристрої; можливість класифікувати музику за виконавцями, альбомами, жанрами, роком випуску та іншими метаданими; можливість створювати власні списки відтворення та організовувати музичні колекції на основі особистих уподобань;

- редагування метаданих: дозволяє користувачам редагувати метадані, пов'язані з музичними файлами, включаючи імена виконавців, назви альбомів, номери треків, жанри та обкладинки альбомів; надає можливості пакетного

редагування для оновлення метаданих для декількох файлів одночасно; підтримує автоматичний пошук метаданих з онлайн-баз даних для спрощення процесу створення тегів;

- відтворення та потокове передавання: надає вбудований музичний плеєр для відтворення аудіофайлів у додатку; підтримує відтворення різних аудіоформатів, включаючи MP3, FLAC, WAV, AAC і OGG;

- інтеграція з популярними потоковими сервісами для забезпечення безперешкодного доступу до потокового музичного контенту;

- пошук і навігація: реалізована надійна функція пошуку, що дозволяє користувачам шукати музичні файли за ключовими словами, виконавцем, альбомом, жанром чи іншими критеріями; надайте параметри фільтрації та сортування для полегшення перегляду та навігації по музичній бібліотеці; пропонує зручний інтерфейс з інтуїтивно зрозумілими елементами навігації і налаштованими видами;

- налаштування та персоналізація: дозволяє користувачам налаштовувати зовнішній вигляд програми, включаючи теми, кольорні схеми та параметри макета; надає налаштування для налаштування параметрів відтворення, таких як налаштування еквалайзера, швидкість відтворення та підтримка різноманітних ефектів; підтримує користувальницькі теги, написи і рейтинги для спрощення персоналізованої організації та категоризації музичних файлів;

- резервне копіювання та синхронізація: пропонує функції резервного копіювання та синхронізації для забезпечення цілісності даних та доступності на декількох пристроях; дозволяє користувачам синхронізувати музичні бібліотеки між різними пристроями, включаючи комп'ютери, смартфони та планшети; надає можливості автоматичного резервного копіювання і синхронізації за розкладом, щоб звести до мінімуму ризик втрати даних;

- інтеграція із зовнішніми пристроями: підтримка інтеграції з зовнішніми пристроями, такими як USB-накопичувачі, Зовнішні жорсткі диски і пристрої мережевого зберігання даних; дозволяє користувачам

отримувати доступ до музичних файлів, що зберігаються на зовнішніх пристроях, і керувати ними безпосередньо з програми;

– доступність і локалізація: забезпечення спеціальних можливостей, таких як підтримка читання з екрану, навігація з клавіатури і перетворення тексту в мову, для користувачів з обмеженими можливостями; підтримка декількох мов і варіантів локалізації для користувачів з різним мовним досвідом. Також необхідно забезпечити спеціальні можливості, такі як підтримка читання з екрану, навігація з клавіатури і перетворення тексту в мову, для користувачів з обмеженими можливостями; підтримка декількох мов і варіантів локалізації для користувачів з різним мовним досвідом.

Ці функціональні вимоги описують основні функції та можливості, якими повинна володіти програма для організації домашньої музичної бібліотеки. Виконання цих вимог забезпечить всебічний і зручний для користувача досвід управління музичними колекціями та користування ними.

Б.3 Опис логічної структури

Загальна схема системи за обраною темою на рис. Б.1. Ця структурна схема описує основні компоненти програми, включаючи клас `LibraryManager`, відповідальний за управління піснями та списками відтворення, клас `Song`, що представляє окремі пісні в бібліотеці, клас `Playlist` для організації пісень у списки відтворення та клас `User` для автентифікації користувача та управління сеансами. Клас `LibraryManager` надає методи додавання, видалення, пошуку та відтворення пісень, а також створення та керування списками відтворення. Клас `Song` містить інформацію про окремі пісні, такі як назва, виконавець, альбом, жанр та тривалість. Клас `Playlist` - це колекція пісень, організована під певною назвою, з методами додавання, видалення та відтворення пісень у списку відтворення. Нарешті, клас `User` відповідає за автентифікацію користувача та управління сеансами, а також за методи входу та виходу з системи.

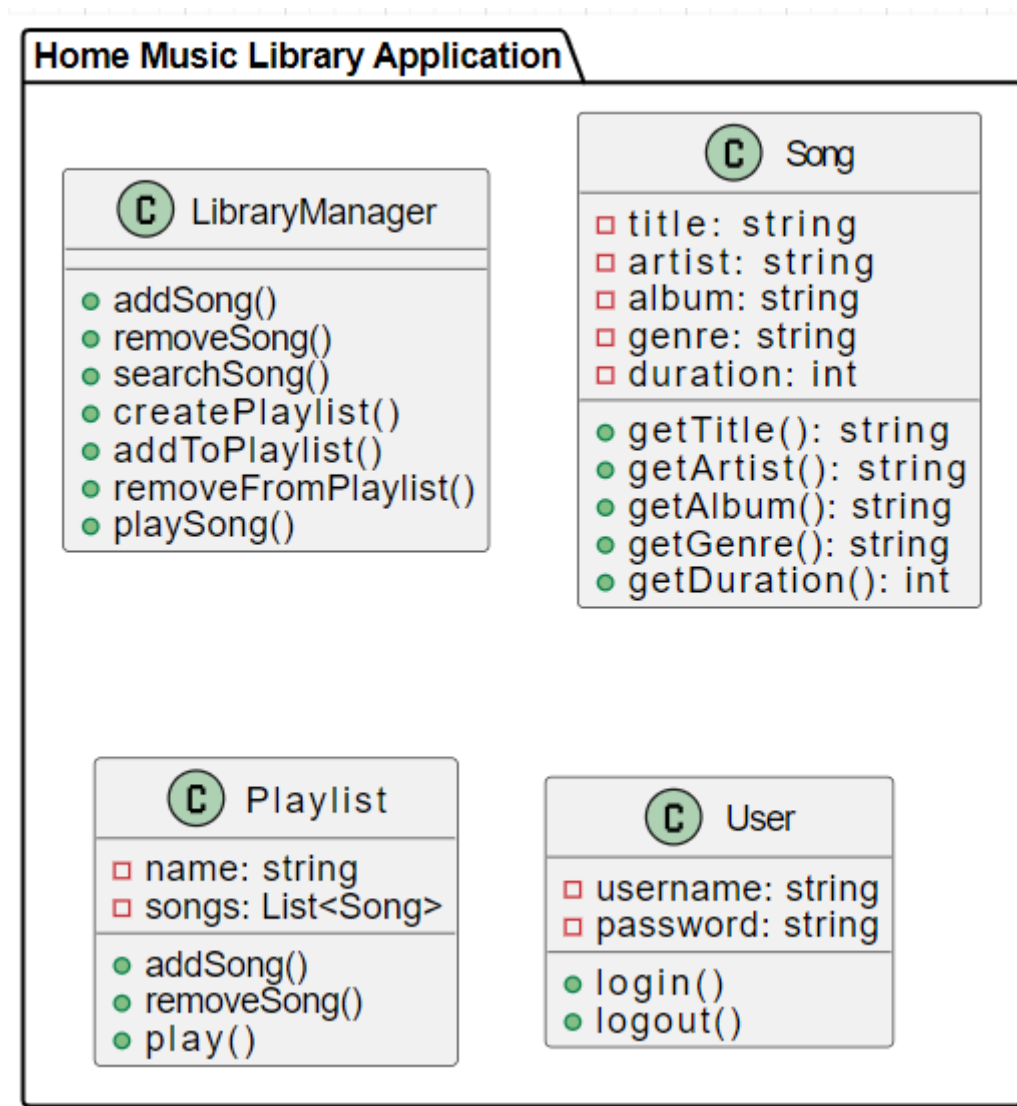


Рисунок Б.1 – Загальна схема системи за обраною темою

Б.4 Використані технічні засоби

Апаратні вимоги до сервера:

- центральний процесор – від 8 фізичних ядер та 16 потоків з тактовою частотою від 4 GHz;
- оперативна пам'ять формату DDR4, з тактовою частотою від 2600 MHz та об'ємом від 128 Гб;
- накопичувач (формату HDD або SSD) з вільним простором від 10 Гб (для розгортання усіх дамів музичних баз даних).

Програмні вимоги до сервера:

- розгорнута серверна ОС, наприклад: Windows Server, Linux Server або macOS Server;

- розгорнута SQL-сумісна СКБД.

Апаратні вимоги до клієнта:

- центральний процесор – від 2 фізичних ядер та 2 потоків з тактовою частотою від 1,7 GHz (з можливістю збільшення під час навантаження);

- оперативна пам'ять об'ємом від 4 Гб;

- накопичувач (формату HDD або SSD) з вільним простором від 0,5 Гб.

Програмні вимоги до сервера:

- наявність веббраузера актуальної версії.

ДОДАТОК В
Текст програми

```

<webDesk:webDeskApplicationPage
    mc:Ignorable="d"
    fntFaml="{ststRes webDeskfntFamlNormal}"
    fntSize="{ststRes webDeskfntSizeNormal}"
    Foreground="{ststRes webDeskForegroundBrush}"
    supOrient="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True"
    toolkit:TiltEffect.IsTiltEnabled="True">

    <shell:SystemTray.ProgressIndicator>
        <shell:ProgressIndicator
x:Name="ProgressIndicator"          IsIndeterminate="True"
IsVisible="{Bnd ProgressIndicatorVisible}" />
    </shell:SystemTray.ProgressIndicator>

    <toolkit:TransNModeService.NaviInTransNMode>
        <toolkit:NaviInTransNMode>
            <toolkit:NaviInTransNMode.Backward>
                <toolkit:TurnstileTransNMode
Mode="BackwardIn"/>
            </toolkit:NaviInTransNMode.Backward>
            <toolkit:NaviInTransNMode.Forward>
                <toolkit:TurnstileTransNMode
Mode="ForwardIn"/>
            </toolkit:NaviInTransNMode.Forward>
        </toolkit:NaviInTransNMode>
    </toolkit:TransNModeService.NaviInTransNMode>
    <toolkit:TransNModeService.NaviOutTransNMode>
        <toolkit:NaviOutTransNMode>
            <toolkit:NaviOutTransNMode.Backward>

```

```

        <toolkit:TurnstileTransNMode
Mode="BackwardOut"/>
    </toolkit:NaviOutTransNMode.Backward>
    <toolkit:NaviOutTransNMode.Forward>
        <toolkit:TurnstileTransNMode
Mode="ForwardOut"/>
    </toolkit:NaviOutTransNMode.Forward>
</toolkit:NaviOutTransNMode>
</toolkit:TransNModeService.NaviOutTransNMode>

<!--LayoutRoot is the root Grd where all page
content is placed-->
    <Grd x:Name="LayoutRoot">

        <Grd.Background>
            <img_srcBrush
img_srcsrcPack="Assets/music_explorer_page_bg.jpg"
Stretch="Fill"/>
        </Grd.Background>

        <!--Pivot Control-->
        <webDesk:Pivot          x:Name="TitleTextCapt"
Title="MUSIC EXPLORER">

            <!--Details Item-->
            <webDesk:PivItem
x:Name="DetailsPivItem" Header="details">
                <Grd>
                    <Grd.RowDefinitions>

```

```

                                <txtBlock
TextCapt="{Bnd                                SelectedArtist.Name}"
TextCaptWrapping="Wrap"                                Margin="10,0,0,0"
Style="{ststRes webDeskTextCaptLargeStyle}"/>
                                <txtBlock
TextCapt="{Bnd      SelectedArtist.Country}"      Margin="10,-
2,0,0" Style="{ststRes webDeskTextCaptNormalStyle}"/>
                                <txtBlock
TextCapt="{Bnd      SelectedArtist.Genres}"      Margin="10,-
2,0,0" Style="{ststRes webDeskTextCaptNormalStyle}"/>
                                </stckPan>
                                </stckPan>

                                <controls:TiltGrd      Grd.Row="1"
Margin="0,6,0,6">
                                <Grd
Tap="OnShowArtistClick">
                                <Grd.clmnDefs>
                                    <clmnDef
Width="Auto"/>
                                    <clmnDef
Width="*/>
                                </Grd.clmnDefs>
                                <Rectangle
Grd.Column="0" Grd.ColumnSpan="2" Fill="Transparent"/>
                                <Rectangle
Grd.Column="0"                                Margin="12,0,0,0"
HorizontalAlignment="Left"      Fill="Black"      Width="85"
Height="85"></Rectangle>

```

```

<img_src Grd.Column="0"
Margin="10,0,0,0"          Width="50"          Height="50"
Stretch="UniformToFill"
srcPack="/Assets/music_explorer_note.png"></img_src>

<txtBlock
Grd.Column="1"          Width="360"          Margin="12,0,0,0"
VerticalAlignment="Center"          fontSize="24"
IsHitTestVisible="False"  TextCapt="Show artist in Nokia
Music"></txtBlock>

</Grd>
</controls:TiltGrd>

<controls:TiltGrd  Grd.Row="2"
Margin="0,6,0,6">

    <Grd Tap="OnPlayMixClick">
        <Grd.clmnDefs>
            <clmnDef
Width="Auto"/>

            <clmnDef
Width="*/>

        </Grd.clmnDefs>
        <Rectangle
Grd.Column="0" Grd.ColumnSpan="2" Fill="Transparent"/>
        <Rectangle
Grd.Column="0"          Margin="12,0,0,0"
HorizontalAlignment="Left"  Fill="Black"  Width="85"
Height="85"></Rectangle>

        <img_src Grd.Column="0"
Margin="10,0,0,0"          Width="50"          Height="50"

```

```

Stretch="UniformToFill"
srcPack="/Assets/music_explorer_note.png"></img_src>
        <txtBlock
Grd.Column="1"          Width="360"          Margin="12,0,0,0"
VerticalAlignment="Center"          fntSize="24"
IsHitTestVisible="False"  TextCapt="Play artist mix in
Nokia Music"></txtBlock>

        </Grd>
    </controls:TiltGrd>

    <controls:TiltGrd Grd.Row="3"
    <webDesk:lngLstSelect.ItemTplt>
        <DataTplt>
            <Grd>
                <stckPan
Orientation="Horizontal"          HorizontalAlignment="Left"
Margin="12,2,0,4" Height="105" Width="432">
                    <img_src
Width="100" Height="100" srcPack="{Bnd ThumbUri}"/>
                    <stckPan
Width="311" Margin="8,4,0,0">

    <txtBlock  TextCapt="{Bnd  Name}"  Margin="10,-2,0,0"
Style="{ststRes          webDeskTextCaptNormalStyle}"
FontWeight="Bold"/>

    <txtBlock  TextCapt="{Bnd  TakenFrom}"  Margin="10,-2,0,0"
Style="{ststRes  webDeskTextCaptNormalStyle}"/>
                </stckPan>
            </stckPan>

```

```

<img_src
Width="30"      Height="30"      HorizontalAlignment="Left"
VerticalAlignment="Top"      Margin="92,2,0,0"
srcPack="Assets/music_explorer_note.png"/>
</Grd>
</DataTmpl>

</webDesk:lngLstSelect.ItemTmpl>
</webDesk:lngLstSelect>
</Grd>
</webDesk:PivItem>

<!--Albums Item-->
<webDesk:PivItem x:Name="AlbumsPivItem"
Header="albums">
<Grd>
<txtBlock      Margin="0,50,0,0"
fntSize="24"      TextCapt="No      albums      available"
HorizontalAlignment="Center"      Visibility="{Bnd
NoAlbumsVisibility}"></txtBlock>
<webDesk:lngLstSelect
x:Name="AlbumsList" Margin="0,-7,0,2" ItemsSrcPack="{Bnd
AlbumsForArtist}"
SelectionChanged="OnAlbumSelectionChanged">

<webDesk:lngLstSelect.ItemTmpl>
<DataTmpl>
<Grd>

```

```

                                <stckPan
Orientation="Horizontal"      HorizontalAlignment="Left"
Margin="12,2,0,4" Height="105" Width="432">
                                <img_src
Width="100" Height="100" srcPack="{Bnd ThumbUri}"/>
                                <stckPan
Width="311" Margin="8,4,0,0">

<txtBlock TextCapt="{Bnd Name}" Margin="10,-2,0,0"
Style="{ststRes webDeskTextCaptNormalStyle}"
FontWeight="Bold"/>

<txtBlock TextCapt="{Bnd TrackCount}" Margin="10,-2,0,0"
Style="{ststRes webDeskTextCaptNormalStyle}"/>
                                </stckPan>
                                </stckPan>
                                <img_src
Width="30" Height="30" HorizontalAlignment="Left"
VerticalAlignment="Top" Margin="92,2,0,0"
srcPack="Assets/music_explorer_note.png"/>
                                </Grd>
                                </DataTmpl>

</webDesk:lngLstSelect.ItemTmpl>
    </webDesk:lngLstSelect.ItemTmpl>
        </webDesk:lngLstSelect>
            </Grd>
                </webDesk:PivItem>

                <!--Similar Item-->

```

```

        <webDesk:PivItem
x:Name="SimilarPivItem" Header="similar">
            <Grd>
                <txtBlock      Margin="0,50,0,0"
fntSize="24"  TextCapt="No  similar  artists  available"
HorizontalAlignment="Center"      Visibility="{Bnd
NoSimilarVisibility}"/>
                <lstBx      Margin="0,-7,      0,2"
x:Name="SimilarList"      ItemsSrcPack="{Bnd
SimilarForArtist}"
SelectionChanged="OnSimilarArtistsSelectionChanged">
                    <lstBx.ItemsPanel>
                        <ItemsPanelTmpl>

<toolkit:WrapPanel/>

                        </ItemsPanelTmpl>
                    </lstBx.ItemsPanel>

                    <lstBx.ItemTmpl>
                        <DataTmpl>
                            <controls:TileItem
Height="{Bnd ItemHeight}"

Width="{Bnd ItemWidth}"

ItemWidth="{Bnd ItemWidth}"

PrimaryTextCapt="{Bnd Name}"

SecondaryTextCapt="{Bnd Country}"

```

```
img_src="{Bnd ThumbUri}"
```

```
Noteimg_src="True"/>
```

```
</DataTmpl>
```

```
</lstBx.ItemTmpl>
```

```
</lstBx>
```

```
</Grd>
```

```
</webDesk:PivItem>
```

```
</webDesk:Pivot>
```

```
</Grd>
```

```
</webDesk:webDeskApplicationPage>
```

ДОДАТОК Д
Слайди презентації



Рисунок Д.1 – Слайд 1



Рисунок Д.2 – Слайд 2

ПОРІВНЯННЯ ІСНУЮЧИХ АНАЛОГІВ

Критерії порівняння	Mp3tag	Kid3	MusicBrainz Picard
Сумісність з платформою	Windows	Windows, macOS, Linux	Windows, macOS, Linux
Підтримувані формати файлів	MP3, FLAC, OGG, WMA, AAC, WAV та інші	MP3, FLAC, OGG та інші	MP3, FLAC, OGG, AAC, WMA та інші
Особливості та функціонал	Комплексне редагування метаданих, пакетне створення тегів, налаштування полів тегів, підтримка сценаріїв	Редагування тегів ID3, коментарів Vorbis і тегів MP4, пакетне створення тегів, пошук в базі даних в режимі онлайн, можливості налаштування	Автоматичний пошук метаданих, пакетне додавання тегів, перейменування файлів, підтримка акустичної дактилоскопії
Простота використання	Інтуїтивно зрозумілий інтерфейс, зручний дизайн	Простий інтерфейс, основні функції створення тегів	Більш швидкий процес навчання, використання бази даних MusicBrainz і акустичної дактилоскопії
Спільнота та підтримка	Велика спільнота користувачів, активні форуми, велика документація	Форуми спільноти, посібники користувача, регулярні оновлення	Спільнота MusicBrainz, форуми, документація, регулярні оновлення
Інтеграція з музичними бібліотеками	Інтеграція з Discogs та MusicBrainz для пошуку метаданих, підтримка скриптів користувачів для онлайн-баз даних	Інтеграція з онлайн-базами даних для пошуку метаданих, підтримка користувацьких налаштувань тегування та скриптів	Використання бази даних MusicBrainz для пошуку метаданих, підтримка акустичної дактилоскопії

3

Рисунок Д.3 – Слайд 3

ПОСТАНОВКА ЗАВДАННЯ

Розробка програмного забезпечення для організації домашньої музичної бібліотеки є актуальним завданням у зв'язку з ускладненням управління цифровими музичними колекціями, зростаючим попитом на персоналізований музичний досвід і потребою в ефективних механізмах організації, пошуку і виявлення інформації. Вирішуючи ці завдання і пропонуючи інноваційні функції і можливості, розробники можуть створити цінний інструмент, який поліпшить якість прослуховування музики користувачами і задовольнить їх зростаючі потреби і очікування.

Отже актуальною задачею залишається програмна реалізація застосунку для впорядкування домашньої фонотеки.

4

Рисунок Д.4 – Слайд 4

ПОРІВНЯННЯ МОВ ПРОГРАМУВАННЯ				
	C# + .NET	JavaScript + Node.js	Python + Django	PHP + Symfony
Суворі типізація	Так	Ні	Так	Ні
Уявлення	Високий	Помірний	Помірний	Помірний
Масштабованість	Високий	Помірний	Помірний	Помірний
Кросплатформність	Так (з .NET Core)	Так	Так	Так
Підтримка спільноти	Великий	Великий	Великий	Великий
Багата екосистема	Так	Так	Так	Так
Функції безпеки	Сильний	Помірний	Помірний	Помірний
Процес навчання	Помірний	Помірний	Помірний	Помірний
Підтримка баз даних	Широкий діапазон	Широкий діапазон	Широкий діапазон	Широкий діапазон
Підтримка паралелізму	Так (з асинхронністю / очікуванням)	Так (з архітектурою, керованою подіями)	Так (за допомогою async / await)	Так (з асинхронністю / очікуванням)
Механізм створення шаблонів	Razor	Різний	Шаблони Django	Twig
Підтримка проміжного програмного забезпечення	Так	Так	Так	Так
Управління пакетами	NuGet	, npm	, pip	Composer

5

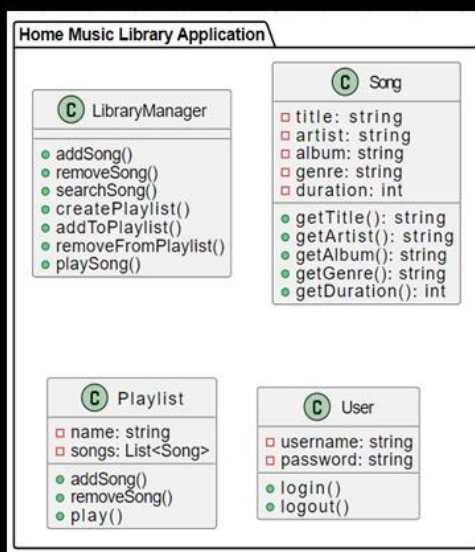
Рисунок Д.5 – Слайд 5

ПОРІВНЯННЯ СЕРЕДОВИЩ РОЗРОБКИ			
Критерій	Visual Studio	JetBrains Rider	Eclipse
Підтримувані платформи	Windows, macOS	Windows, macOS, Linux	Windows, macOS, Linux
Основна мова	C#	C#	C#
IntelliSense	Так	Так	Так
Налагоджувач	Так	Так	Так
Інтеграція модульного тестування	Так (вбудована підтримка MSTest, NUnit, xUnit)	Так (вбудована підтримка MSTest, NUnit, xUnit)	Так (з такими плагінами, як NUnit, xUnit)
Інтеграція контролю версій	Так (вбудована підтримка Git, TFVC)	Так (вбудована підтримка Git)	Так (з такими плагінами, як EGit)
Екосистема плагінів	Vast (Visual Studio Marketplace)	Moderate (репозиторій плагінів JetBrains)	Vast (Eclipse Marketplace)
Уявлення	Відмінно	Відмінно	Помірний
Споживання ресурсів	Висока (особливо на вікнах)	Помірний (відносно легкий)	Помірний
Час запуску	Довгий	Помірний	Помірний
Підтримка спільноти	Великий	Великий	Великий
Простота використання	Підходить для початківців	Підходить для початківців	Підходить для початківців
Інтеграція з Azure	Відмінно	Обмежено (Azure Toolkit for Rider)	Обмежений (Azure Toolkit для Eclipse)

6

Рисунок Д.6 – Слайд 6

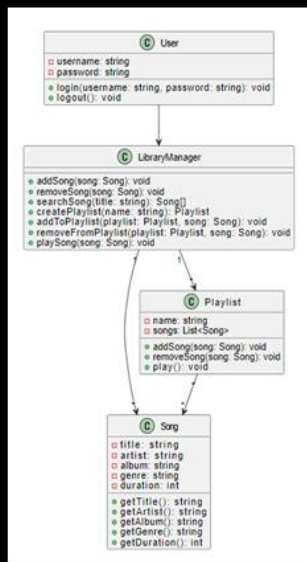
ЗАГАЛЬНА СХЕМА ЗАСТОСУНКУ



7

Рисунок Д.7 – Слайд 7

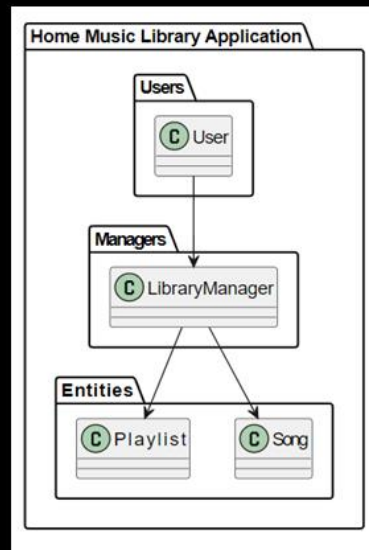
ДІАГРАМА КЛАСІВ



8

Рисунок Д.8 – Слайд 8

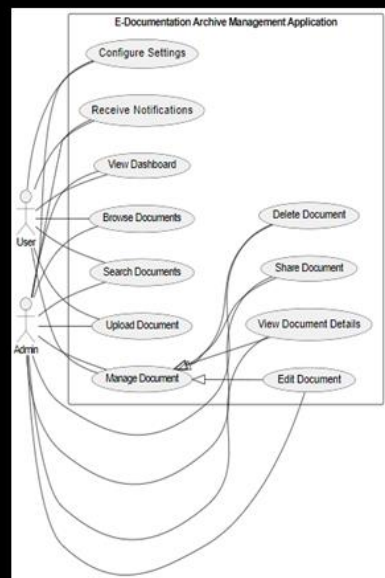
ДІАГРАМА ПАКЕТІВ



9

Рисунок Д.9 – Слайд 9

ДІАГРАМА ПРЕЦЕДЕНТІВ



10

Рисунок Д.10 – Слайд 10

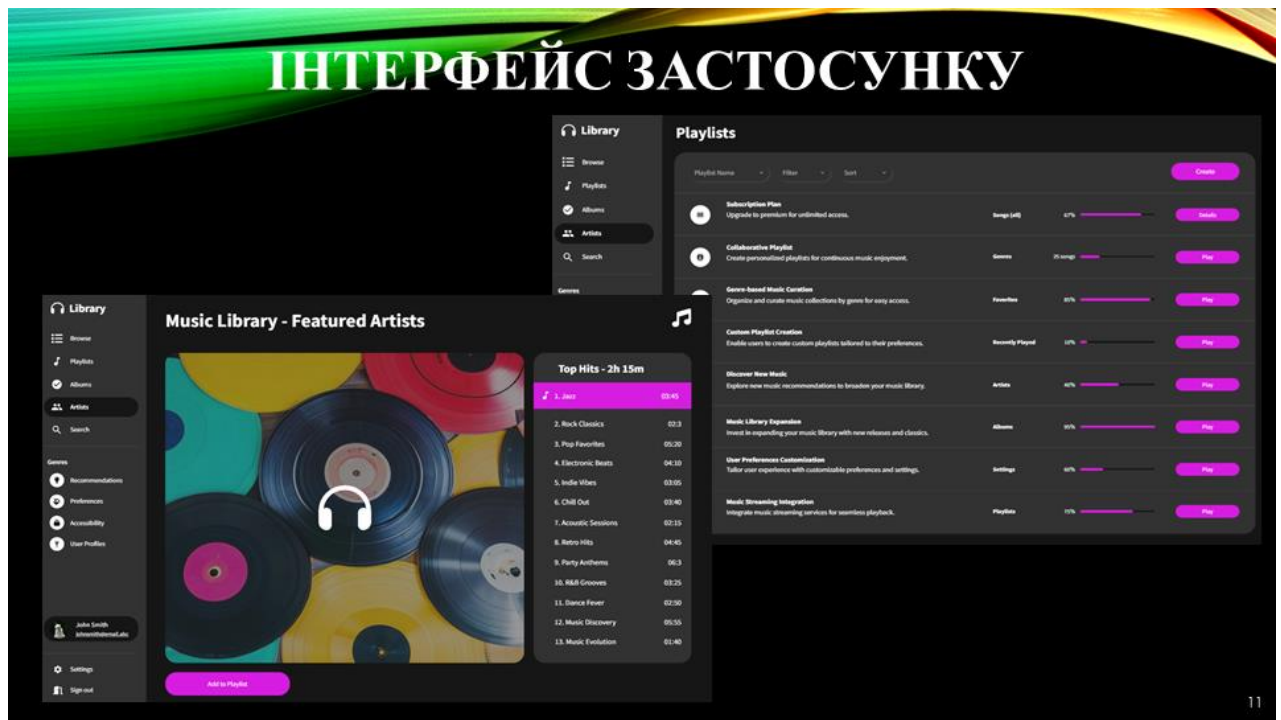


Рисунок Д.11 – Слайд 11

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було виконано програмну реалізацію застосунку електронного ведення архіву документації. Для цього було виконано ряд завдань:

- виконано дослідження проблемної області накопичення та впорядкування фонотеки;
- виконано дослідження переваг та недоліків існуючих аналогів для впорядкування фонотеки;
- виконано вибір технічних засобів реалізації запропонованого застосунку та обґрунтувати його;
- спроектовано архітектуру нового застосунку;
- програмно реалізовано новий застосунок;
- проведено тестування розробки;
- сформовано висновки на основі виконаної роботи.

В результаті виконання завдання було отримано систему, що відповідає всім вимогам визначеним на початку розробки та закріпленим в технічному завданні. Розроблена система є зручною для використання та виконує всі необхідні функції.

Усі завдання дипломної кваліфікаційної роботи бакалавра повністю виконано.

Рисунок Д.12 – Слайд 12