

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
«ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ В ЕКОНОМІЦІ»**

Кафедра інформаційних систем в економіці

галузь знань 12 «Інформаційні технології»
спеціальність 122 «Комп'ютерні науки та інформаційні технології»

Форма навчання: денна

БАКАЛАВРСЬКИЙ ДИПЛОМНИЙ ПРОЕКТ

на тему

**ПРОЕКТУВАННЯ СОЦІАЛЬНО-КОМЕРЦІЙНОЇ ЦИФРОВОЇ ПЛАТФОРМИ
МУЗИЧНОЇ СПІЛЬНОТИ**

Студента Бевзюка Дмитрія Андрійовича _____

Науковий керівник:

к.е.н., доцент

_____ Денісова О. О.

**Бакалаврський дипломний проект
допущений до захисту в Екзаменаційній
комісії з атестації здобувачів вищої освіти**

В.о. завідувача кафедри:

к.е.н., доцент.

_____ Тішков Б. О.

Київ 2022

Міністерство освіти і науки України
Державний вищий навчальний заклад
«Київський національний економічний університет
імені Вадима Гетьмана»
Навчально-науковий інститут
«Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці

галузь знань 12 «Інформаційні технології»
 спеціальність 122 «Комп'ютерні науки та інформаційні технології»
 денна форма навчання

Затверджую:

в.о. завідувача кафедри _____ Тішков Б.О.
“ _____ ” _____ 2022 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на бакалаврський дипломний проект
на тему: «ПРОЕКТУВАННЯ СОЦІАЛЬНО КОМЕРЦІЙНОЇ ЦИФРОВОЇ
ПЛАТФОРМИ МУЗИЧНОЇ СПІЛЬНОТИ»

Бевзюка Дмитрія Андрійовича

Тему бакалаврського дипломного проекту затверджено наказом ректора від «17» березня 2021 р. № 385-ст.

Бакалаврський дипломний проект виконується на матеріалах

План бакалаврського дипломного проекту

Розділ 1	Характеристика та аналіз предметної галузі (назва розділу)
Розділ 2	Розробка вимог і моделювання інформаційної системи/підсистеми (назва розділу)
Розділ 3	Проектування компонентів системи (назва розділу)
Об'єкт дослідження:	Процес проектування соціально комерційної цифрової платформи музичної спільноти
Предмет дослідження:	Платформа музичної спільноти
Мета бакалаврського дипломного проекту:	Створення програмного продукту для автоматизації процесів платформи музичної спільноти

Конкретні завдання, які студент повинен виконати для досягнення поставленої мети:

У розділі 1 необхідно надати характеристику предметної галузі – платформи музичної спільноти

привести параметри об'єкта

дослідження, які необхідні для побудови інформаційної системи.

Провести аналіз літературних джерел і практичного досвіду

побудови підсистеми та використання комп'ютерних технологій в предметній галузі.

У розділі 2 визначити та обґрунтувати вимоги до інформаційних підсистем, провести їх

аналіз та специфікацію. Виконати постановку та розробити алгоритми розв'язання задачі.

Розробити модель інформаційної системи управління.

У розділі 3 виконати проектування компонентів інформаційної підсистеми, зокрема інформаційно-технічного та програмного забезпечення.

інформаційного, технічного та програмного забезпечення. Виконати

реалізацію програмних модулів і привести результати, вказав можливість та доцільність застосування їх на практиці.

Завдання підготував

Науковий керівник

_____ (підпис)

_____ Денісова О. О.

_____ (ініціали, прізвище)

« 10» березня 2021р.

Завдання одержав студент

_____ (підпис)

_____ Бевзюк Д. А.

_____ (ініціали, прізвище)

«10» березня 2021р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на бакалаврський дипломний проект
на тему: **«ПРОЕКТУВАННЯ СОЦІАЛЬНО КОМЕРЦІЙНОЇ
ЦИФРОВОЇ ПЛАТФОРМИ МУЗИЧНОЇ СПІЛЬНОТИ»**

Тему бакалаврського дипломного проекту затверджено наказом ректора від «23» березня 2020 р. № 421-ст.

Бакалаврський дипломний проект виконується на матеріалах

наукових публікацій та публікацій з мережі інтернет

План бакалаврського дипломного проекту

Розділ I Характеристика та аналіз предметної галузі.

Розділ II Розробка вимог і моделювання інформаційної системи

Розділ III Проектування та реалізація компонентів інформаційної системи

Об'єкт дослідження:

Процес управління комерційної цифрової платформи музичної спільноти

Предмет дослідження:

Проектування соціально комерційної цифрової платформи музичної спільноти

Мета бакалаврського дипломного проекту полягає в тому щоб запропонувати альтернативну систему на базі існуючих інновацій.

Конкретні завдання, які студент повинен виконати для досягнення поставленої мети:

У розділі I Охарактеризувати предметну галузь та об'єкт дослідження.

Проаналізувати літературні джерела та практичний досвід використання інформаційних систем автоматизованого обліку діяльності комерційної цифрової платформи музичної спільноти.

У розділі III Проаналізувати вимоги до інформаційної системи та описати її властивості, характеристики та функції.

Зробити постановку задачі обліку діяльності комерційної цифрової платформи музичної спільноти та розробити алгоритм її розв'язання.

Побудувати візуальні моделі інформаційної системи, що визначають її функції, структуру та поведінку.

У розділі III Розробити інформаційне, програмне та технічне забезпечення інформаційної системи комерційної цифрової платформи музичної спільноти

Викласти суть пропозицій, що лежать в основі виконаного проекту та результати їх апробації на контрольному прикладі.

АНОТАЦІЯ

бакалаврського дипломного проекту студента 4 курсу

Навчально-наукового інституту

«Інститут інформаційних технологій в економіці»

Бевзюка Дмитрія Андрійовича, виконаного на тему:

**«ПРОЕКТУВАННЯ СОЦІАЛЬНО КОМЕРЦІЙНОЇ ЦИФРОВОЇ
ПЛАТФОРМИ МУЗИЧНОЇ СПІЛЬНОТИ»**

Київ: кафедра інформаційних систем в економіці, 2021 р.

Бакалаврський дипломний проект присвячений проблемі, яка наразі є досить актуальною - автоматизації процесів комерційної цифрової музикальної спільноти, ціль якої удосконалення використанням сучасних інформаційних технологій.

Бакалаврський дипломний проект складається з трьох розділів, пов'язаних між собою.

В першому розділі дана характеристика предметної галузі й об'єкта дослідження, наведено аналіз задач, що розв'язуються, а також наводиться перелік існуючих програмних засобів даної предметної галузі.

Другий розділ є проектним і присвячений обґрунтуванню методу проектування системи, розробленню її архітектури, виконанню постановки та розробленню алгоритму розв'язання задач.

Третій розділ – містить інформаційне, технічне програмне та організаційне забезпечення для коректної роботи системи автоматизації роботи комерційної цифрової музикальної спільноти. Виділені питання щодо організації інформаційного забезпечення: розроблена структура інформаційного забезпечення, описано форми вхідних та вихідних документів. Обґрунтований комплекс технічних засобів, а також програмне забезпечення, що використовується для створення системи. Вказані структури інформаційних масивів, які використовуються під час вирішення задачі.

Висновки містять рекомендації щодо доцільності розроблення та впровадження системи автоматизації процесів у комерційних цифрових музикальних спільнот

РЕФЕРАТ

Бакалаврський дипломний проект містить 870 сторінок, 9 таблиць, 19 рисунків, список літератури з 20 найменувань, 1 додаток.

ПРОЕКТУВАННЯ СОЦІАЛЬНО КОМЕРЦІЙНОЇ ЦИФРОВОЇ ПЛАТФОРМИ МУЗИЧНОЇ СПІЛЬНОТИ

Перелік ключових слів: музика, платформа, музична спільнота, цифрова музична платформа, автоматизація, програмне забезпечення, база даних, даталогічна модель, багатоплатформний додаток.

Предметом дослідження є комерційна цифрова музикальна спільнота.

Об'єктом дослідження є процес автоматизації управління роботою комерційної цифрової музикальної спільноти.

Мета бакалаврського дипломного проекту полягає в створенні програмного продукту для автоматизації процесів комерційної цифрової музикальної спільноти.

Завданнями бакалаврського дипломного проекту є глибоке дослідження предметної області та створення програмного продукту для її автоматизації.

Апаратні та програмні засоби, що використовувались на при проектуванні: процесор

- Процесор Intel Core i7-8650U 1.90ГГц - 2.11ГГц;
- Оперативна пам'ять - 8Гб;
- Накопичувач – SSD, 512 Гб;
- Екран – IPS, 15.6.

Результати досягнуті в процесі роботи: створено програмний продукт для автоматизації процесів комерційної цифрової музикальної спільноти.

Одержані результати можуть бути використані для розробки програмного забезпечення задля автоматизації процесів комерційної цифрової музикальної спільноти.

Рік виконання бакалаврського дипломного проекту – 2021 р.

ЗМІСТ

ВСТУП	10
РОЗДІЛ 1. ХАРАКТЕРИСТИКА ТА АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ	12
1.1. Характеристика предметної галузі та об'єкта дослідження	12
1.2. Аналіз літературних джерел та практичного досвіду використання ІС і технологій в предметній галузі.....	15
РОЗДІЛ 2. РОЗРОБКА ВИМОГ І МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ	20
2.1. Аналіз і специфікація вимог до інформаційної системи/підсистеми	20
2.2. Постановка та алгоритм розв'язання задачі.....	23
2.3. Моделювання інформаційної підсистеми	27
РОЗДІЛ 3. ПРОЕКТУВАННЯ КОМПОНЕНТІВ СИСТЕМИ	31
3.1. Інформаційне забезпечення	31
3.2 Технічне забезпечення.....	39
3.3. Програмне забезпечення	40
3.4 Результати реалізації інформаційної підсистеми	42
ВИСНОВКИ.....	45
ДОДАТКИ.....	48

ВСТУП

Колись ми слухали музику на касетах і компакт-дисках. Колекція серйозного меломана могла займати в квартирі не менше місця, ніж домашня бібліотека. Потім прийшов час mp3, і пара сотень улюблених альбомів легко вмістилася в плеєрі. Тепер і цей спосіб прослуховування музики застарів: йому на зміну прийшли комерційні цифрові музичні платформи.

Це інтелектуальні мережеві музичні бібліотеки з безліччю корисних функцій, які миттєво надають матеріал будь-якого жанру. Комерційні цифрові музичні платформи дозволяють програвати музику на смартфоні, планшеті, ноутбучі або іншому подібному гаджеті. Немає сенсу завантажувати альбоми на HDD, всямузика доступна за невелику абонплату в мережевий бібліотеці в зручному вигляді.

Академічні дебати довгий час оточували термін «спільнота», вперше визначений як соціологічна конструкція теоретиком Фредеріком Тонном (1887), а пізніше досліджений іншими (Дюркгейм 1893, Вебер 1947). У 1990 - х роках широко розповсюджене використання Інтернету порушило ці ранні уявлення про те, що визначає та обмежує поняття «спільнота», але зараз існує загальна думка вчених про те, що групи за інтересами в Інтернеті також можуть функціонувати як спільноти, включаючи групи, орієнтовані на будь-яку кількість різних жанрів музики. Для членів спільноти в Інтернеті приналежність до такої групи може відігравати таку ж важливу роль у їхньому житті, як і фізична місцева спільнота.

Музичні спільноти в Інтернеті проявляються або як «автономні» онлайн-спільноти самі по собі (тобто вони існують лише у віртуальному просторі без перекриття із співвідносною офлайн-спільнотою), або як активне онлайн-місце, яке існує у додатку до вже створеної офлайн-музичної спільноти (тобто остання є "Конвергентна спільнота" - вона займає місця як у локальних, так і в автономних місцях). Однак, незалежно від їхнього

поодинокого чи подвійного розташування, дух принципів може бути, і часто є, тиражований у групах за інтересами в Інтернеті.

РОЗДІЛ 1. ХАРАКТЕРИСТИКА ТА АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1. Характеристика предметної галузі та об'єкта дослідження

Відкриті Інтернет -спільноти (ВІС) - це групи за інтересами Інтернету, веб -сайти яких є безкоштовними та відкритими для перегляду громадськістю; немає вимоги реєструватися - це безкоштовно - якщо хтось не бажає розміщувати повідомлення, коментар на сайті. На сьогодні більшість людей знайомі зі стандартним зручним для користувачів форматом ВІС, який складається з форумів-дошок оголошень-чатів, сховищ YouTube, посилань на пов'язані сайти та "правил", розміщених на головній сторінці сайту.

Учасники можуть вільно розміщувати повідомлення, спілкуватися в чаті, співпрацювати, обговорювати, задавати питання, давати поради та ділитися вмістом, створеним користувачами, на форумах спільноти.

Щоб зрозуміти, як ВІС перетворилися на свій нинішній формат, а також функціонують як вільні простори співпраці та обміну, потрібне коротке історичне дослідження їх еволюції з середини 80 -х років, коли вони вперше з'явилися.

Те, що зараз впізнається як "стандартний" формат ВІС - тобто веб - сайт з URL -адресами з кількома дошками оголошень/форумами, де члени вільно обмінюються інформацією, вмістом та дискурсом - сягає своїм корінням у рух контркультури 60 -х років, зокрема в ідеалах а також дії мислителя/підприємця Стюарта Брендта. Бренд об'єднав: дуже різні контркультурні, академічні та технологічні спільноти в єдиний текстовий простір. Цей простір, у свою чергу, став мережевим форумом - місцем, де члени цих спільнот збиралися, обмінювалися ідеями та легітимністю, а в процесі синтезували нові інтелектуальні рамки та нові соціальні мережі.

Швидкий веб-пошук відкритих музичних спільнот в Інтернеті, які вже досліджували дослідники музичної освіти, демонструє ступінь впливу дизайну WELL, обговореного вище, як і раніше, включаючи Віртуальний хор Еріка Уітакре (<http://ericwhitacre.com/thevirtual-chor>), сайт народної музики

The Mudcat Café (<http://mudcat.org>), IndabaMusic.com (<https://www.indabamusic.com>) та Hangout Hangouts (<http://www.banjohangout.org>). Не всі дизайнерські ініціативи WELL можна зрозуміти на скріншоті нижче зі сторінки дискусійного форуму Hangouts Hangouts, але чітко видновідкритість та її еволюцію сайту.

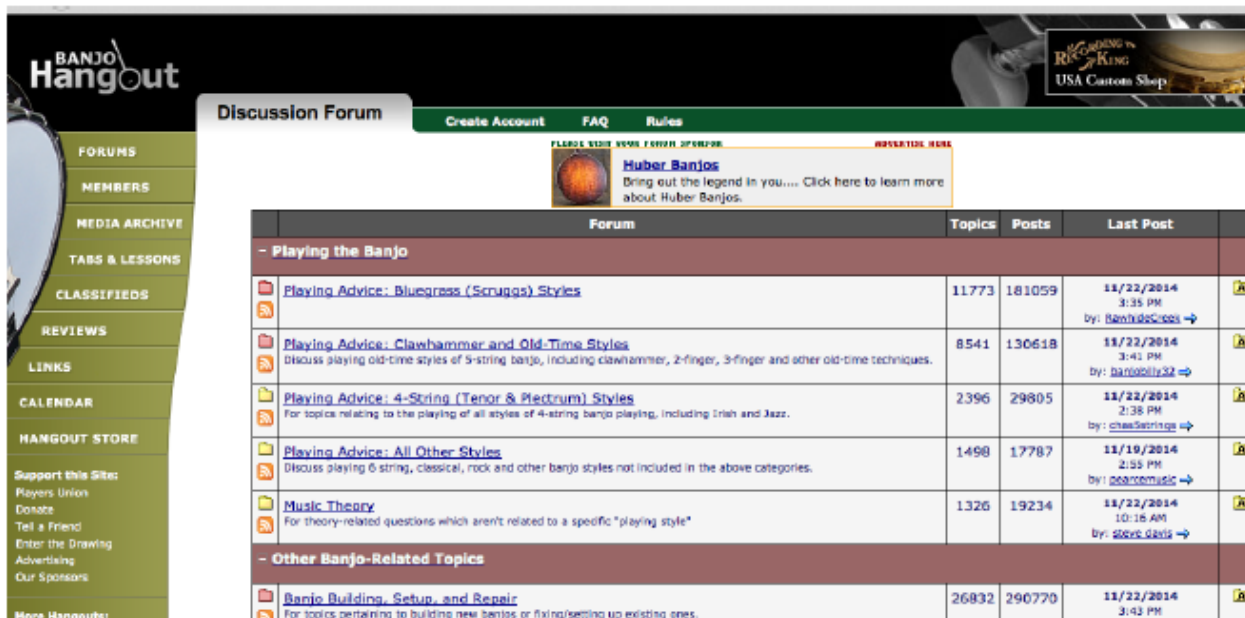


Рис. 1.1. Дискусійний форум Hangouts Hangouts

Дослідники нових медіа та комунікацій впродовж останніх двох десятиліть почали передбачати нові шляхи як для дослідження, так і для розвитку онлайн -спільнот, і цей твір має потенційно значне застосування для дослідників спільноти музики та практиків. Науковець з нових медіа Ненсі Бейм пояснює, чому дослідження "в Інтернеті" кидає виклик традиційним уявленням про дослідження:

«Важко зрозуміти, наскільки старі теоретичні та методологічні рамки можуть бути застосовані для розуміння сучасних суспільних утворень [таких як Інтернет -спільноти та/або системи соціальних медіа]. Чи можемо ми все -таки спиратися на теорії, які були розроблені в більш ранній епосі для обґрунтування нашого розслідування та пояснення наших висновків? Як застосувати процедурні моделі до дослідження, коли ці моделі більше не

підходять? Як ми можемо покращити, не лише документуючи нове, аби розповісти про цінні речі про явища, які так швидко змінюються?»

Не секрет, що пандемія COVID-19 повністю знищила музичну індустрію всього світу. Багато робочих місць у промисловості більше неможливі, і артисти, які покладаються на гастролі та живу музику, щоб заробляти на прожиток, тепер опиняються без доходу. Кожному, хто займається музичною індустрією, доводилося швидко повертатися, щоб вижити, переймаючи нові ідеї та випробовуючи нові моделі.

Наприклад, автори пісень Americana із Нешвілла "Wild Ponies" нещодавно відкрили фургон з продуктами харчування, який також є "мобільним громадським центром та платформою для творчості". «Ми не покидаємо музичну індустрію, - сказав мені у розмові Дуг Вільямс з Wild Ponies, - вона просто не існує зараз». Wild Ponies - виняток у тому, що вони створюють фізичний простір для збору людей. Натомість більшість артистів звертаються до цифрових просторів для створення музики, як і фестивалі, музичні майданчики та музичні конференції.

Незважаючи на те, що прямі трансляції можуть забезпечити частину досвіду живої музики, втраченого через обмеження COVID, головне питання - зробити їх фінансово спроможними. Станом на осінь 2020 року Facebook дозволить придбати квитки на їх прямі трансляції. На жаль, прямі трансляції у Facebook зазвичай грають небагатьох глядачів завдяки незрозуміло агресивному алгоритму платформи. Instagram, який належить Facebook, бачить низьку аудиторію у своїх прямих трансляціях. Оскільки потоки звертаються до продажу квитків як основної форми монетизації живих виступів, незрозуміло, чи будуть глядачі платити за перегляд музики в прямому ефірі так само, як і за концерт. Зараз потрібна нова модель прямих трансляцій, яка створює спільноту, засновану на фінансуванні виконавців.

Перспектива платформи прямих трансляцій, яка забезпечує кращий відеовміст та взаємодію з шанувальниками, що спрямована на постійну монетизацію, є великою для виконавців. Однак, щоб досягти успіху ,

музикант повинен адаптуватися до культури, будувати спільноту, а не намагатися донести її ззовні.

1.2. Аналіз літературних джерел та практичного досвіду використання ІС і технологій в предметній галузі

Інтернет назавжди змінив ландшафт музичної індустрії. Це загальновідомо для артистів, які використовували ефективні цифрові технології, такі як обмін файлами та цифровий маркетинг через нескінченну кількість каналів у соціальних мережах. Хоча все це може здатися очевидною інформацією, факт залишається фактом, що багато музикантів все ще борються з вибором правильних онлайн -платформ для завантаження, обміну та просування своєї музики.

Нескінченний перелік пропозицій може бути величезним, але важливо мати чітке розуміння того, як вирішувати ці три питання:

- Які спільноти є найбільш популярними серед вашої цільової аудиторії?
- Чому вони потужні (тобто які їх можливості)?
- Які обставини підходять для того, щоб втілити їх у життя?

Список 10 найкращих платформ цифрової музики

1. SoundCloud

Заснована в 2007 році, SoundCloud залишається однією з провідних світових платформ для розповсюдження аудіо та залучає понад 175 мільйонів унікальних відвідувачів щомісяця. З точки зору безкоштовного потокового передавання, SoundCloud був давнім гігантом, і це не зміниться найближчим часом. Оголошуючи свої особливості комунікабельності/спільності, переважна більшість як відомих виконавців, так і зростаючих талантів звертаються до SoundCloud, щоб завантажити свою музику.

Стиль "стрічки новин" домашньої сторінки, опції репосту/улюбленого/списку відтворення, а також послідовники/спільноти груп відіграють величезну роль у розширенні охоплення величезної

бібліотеки SoundCloud. Тож, хоча є багато людей, які користуються системою, намагаючись збільшити численні (але переважно уявні) читачі, SoundCloud-це простий варіант, який дає шанувальникам можливість безкоштовно транслювати ваш останній сингл.

2. Audiomack

Не надто відстаючи від SoundCloud, Audiomack зростає як зручна потокова платформа. Audiomack дозволяє артистам оцінювати охоплення їхньої музики за допомогою різноманітних рейтингів на основі тенденцій. Точна кількість п'єс на день/тиждень/місяць доступна, а домашня сторінка "що є в тренді" дозволяє споживачам музики знати, які пісні та альбоми гідні їхнього цінного часу. Між менш жорсткою політикою щодо авторських прав та легко засвоюваною аналітикою, Audiomack-це добре позиціонована платформа та розумний вибір для незалежних виконавців, які хочуть повністю бачити маркетингові зусилля свого нового альбому.

3. iTunes

Кожен, хто слухав музику цього століття, знає про iTunes. Якщо ви є членом вибраної групи споживачів, які щиро вірять у фінансову підтримку творчості своїх улюблених виконавців, iTunes - це ваша музична мекка. Платформа продовжує розвиватися, коли Apple постійно впроваджує інновації як технологічний гігант, і хоча потокова передача поточного короля споживання музики, музичні бібліотеки давніх користувачів iTunes все ще бачать багато дій.

Якщо ви незалежний музикант, який хоче надати шанувальникам можливість купувати їхню музику, завантаження в iTunes все ще є основним способом зробити це. Не сподівайтесь здобути золото, але одного лише визнання торгової марки достатньо, щоб завоювати деяких шанувальників.

4. Spotify

Протягом останніх кількох років Spotify працював над тим, щоб стати надзвичайно потужним цифровим музичним сервісом і став королем у бізнесі потокового передавання музики. Велика бібліотека музики від великих

виконавців зробила Spotify привабливим варіантом, і хоча безкоштовні користувачі обтяжені частою рекламою, користувачі преміум-класу насолоджуються прослуховуванням без реклами, а також у режимі офлайн/мобільних додатків.

5. YouTube

Якщо ви не починаючий музикант, вам слід приділити якісний час і зусилля для створення професійно створених музичних кліпів. І коли настає час поділитися своїми візуальними зображеннями зі світом, ви завантажуйте їх на YouTube. Інтернет-сайт для обміну відео зробив величезну різницю для інді-виконавців, які іноді піднімаються на національне визнання не обов'язково за силу пісні, а за візуальну ефективність, яка вдихає нове життя в альбом. Між простотою використання та проникливими інструментами вимірювання, YouTube є основною платформою, яка привертає нерозділену увагу більш професійно налаштованого сегменту молодих художників.

6. Bandcamp

Bandcamp, мабуть, найцікавіший вибір із цього списку музичних платформ. Безумовно, веб-сайт орієнтований на незалежних музикантів, але він, як правило, залучає більше нішу як для виконавців, так і для шанувальників. Як платформа, яка сприяє безкоштовній трансляції, а також фінансовій підтримці виконавця, Bandcamp має функцію "все в одному", яка може виявитися корисною для музикантів. Є багато успішних виконавців, які ефективно рекламували свій матеріал на Bandcamp, але будьте обережні, тому що, якщо ваша музика не перегукується з певною демографічною групою, вона може загубитися в суміші.

7. Vimeo

Ми вже встановили, що YouTube є основним гравцем у світі музичних кліпів. Vimeo, однак, займає друге місце та має власні спеціальності для художників, які знімають різноманітний візуальний контент. Коли музиканти йдуть більш різноманітним маршрутом з відеоматеріалами, такими як документальні фільми та закулісні кадри, не погана ідея поекспериментувати

з Vimeo через його добре обізнану творчу спільноту. Ні в якому разі не розумно переносити всі свої відео з облікового запису YouTube на Vimeo, але не бійтеся випробувати його, продовжуючи збільшувати свою базу прихильників.

8. Tidal

Будучи ще молодим у своєму існуванні, останні кілька місяців Tidal робив заголовки у світі потокової музики. Платформа пишається тим, що пропонує високоякісне аудіо/відео для своїх користувачів, і з великою групою виконавців-суперзірок, які спільно підписують послуги, багато шанувальників стрибають на хвилю. Хоча багато хто в галузі вважає, що Tidal-це лише егоцентрична спроба великих артистів лейблу отримати кращі прибутки у своїх потоках, інді-артисти все одно повинні стежити за кожним кроком від Tidal, оскільки він продовжує зростати (особливо Tidal Rising та Tidal Discovery) .

9. Google Play

Незважаючи на те, що Google Play отримує менше уваги, не варто ігнорувати музичний сервіс однієї з найпотужніших компаній світу. Якщо ви хочете, щоб уся ваша база шанувальників мала можливість фінансово підтримувати вашу музику, важливо переконатися, що у вас є присутність у Google Play. Є багато користувачів мобільних телефонів Android, які покладаються на Google Play, а не на iTunes, тому, якщо ви хочете, щоб усі ваші треки були охоплені, не залишайте їх осторонь.

10. Персональний веб -сайт

Незважаючи на всі ці кваліфіковані музичні платформи, всім серйозним музикантам потрібен особистий веб -сайт, на якому помітно відображається їхня найновіша та найкраща музика. Швидше за все, ви будете використовувати один із вищезгаданих варіантів для завантаження або прослуховування, але наявність музики на кожній із цих дев'яти платформ, а не на власному веб -сайті - це рецепт катастрофи. Як зростаючого виконавця, розумно мати централізовану музику в кількох

видатних місцях, а не кидати десяток різних посилань на шанувальників та пресу. І з усіх цих варіантів підтримка чіткого, оновленого веб-сайту, мабуть, є найважливішим.

РОЗДІЛ 2. РОЗРОБКА ВИМОГ І МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Аналіз і специфікація вимог до інформаційної системи/підсистеми

Вимоги до системи – це сукупність вимог до властивостей, якості та функцій системи, розробленої у даному проекті.

У даному розділі описано:

- Бізнес-вимоги
- Функціональні вимоги
- Нефункціональні вимоги

Бізнес-вимоги до життєвого циклу розробки програмного забезпечення стосуються вимог чи бажань організації, що дозволяє бізнесу досягати кінцевих цілей, бачення та цілей.

Бізнес-вимоги високого рівня описують, що повинна робити система чи рішення і чому. Вони визначають ступінь бізнес -потреб або проблем, які має вирішити конкретний проект чи завдання..

Бізнес-вимоги комерційної цифрової музичної платформи зображені на рис. 2.1 за допомогою менеджера специфікацій *Enterprise Architect*.

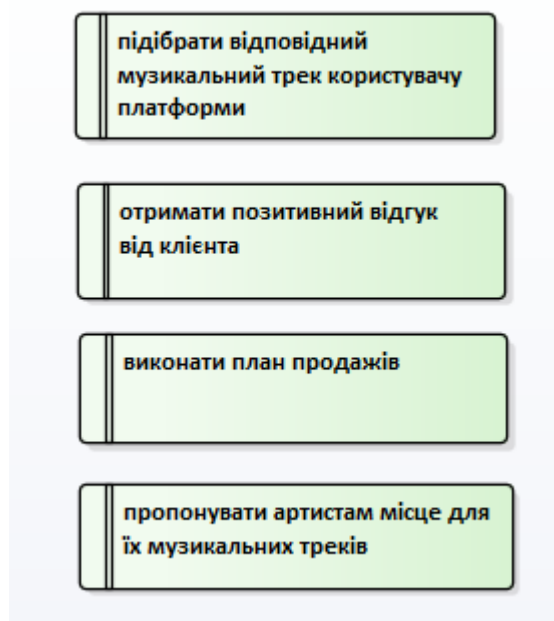


Рис. 2.1. - Бізнес-вимоги комерційної цифрової музичної платформи

Функціональні вимоги до системи

Функціональні вимоги визначають «як» реалізувати продукт.

Діаграма функціональних вимог для роботи комерційної цифрової музичної платформи реалізована за допомогою менеджера специфікацій *Enterprise Architect* та показана на рис. 2.2

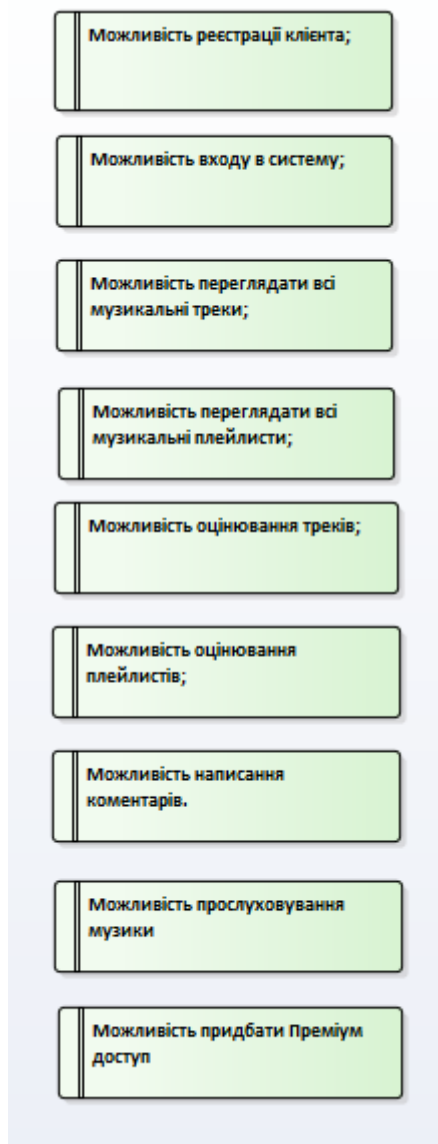


Рис. 2.2. - Діаграма функціональних вимог для комерційної цифрової музичної платформи

Нефункціональні вимоги до системи

Нефункціональні вимоги - вимоги до поведінки системи

Діаграма нефункціональних вимог для системи роботи комерційної цифрової музичної платформи реалізована за допомогою менеджера специфікацій *Enterprise Architect* та показана на рис. 2.3.

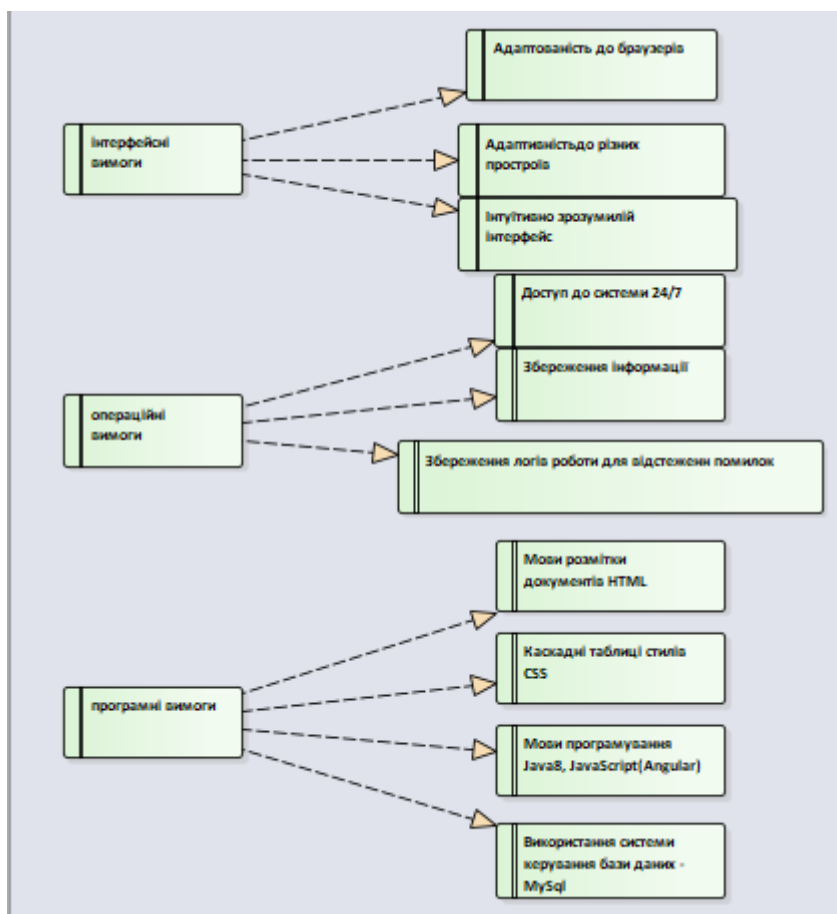


Рис. 2.3. - Діаграма нефункціональних вимог для комерційної цифрової музичної платформи

2.2. Постановка та алгоритм розв'язання задачі

Процес автоматизації комерційної цифрової музичної платформи передбачає під собою безліч задач, однією із найважливіших із яких є функція прийому замовлень від клієнтів.

Сильні сторони проекту:

- Масивна база користувачів
- Персоналізація музики та рекомендовані канали на основі уподобань користувачів за допомогою машинного навчання
- Мінімальна плата за права

Слабкі сторони проекту:

- Величезна залежність від підключення до Інтернету
- Складна схема виплати ускладнює прогнозування суми, що підлягає сплаті за ліцензійними угодами

Можливості:

- Виробництво власної музики
- Модель B2B, створення партнерства з мобільним телефоном
- Вихід на ринки з великою кількістю населення

Загрози проекту:

- відносно висока конкуренція на ринку;
- Важкі передумови для ліцензування для залучення компаній-виконавців або телекомунікаційних компаній

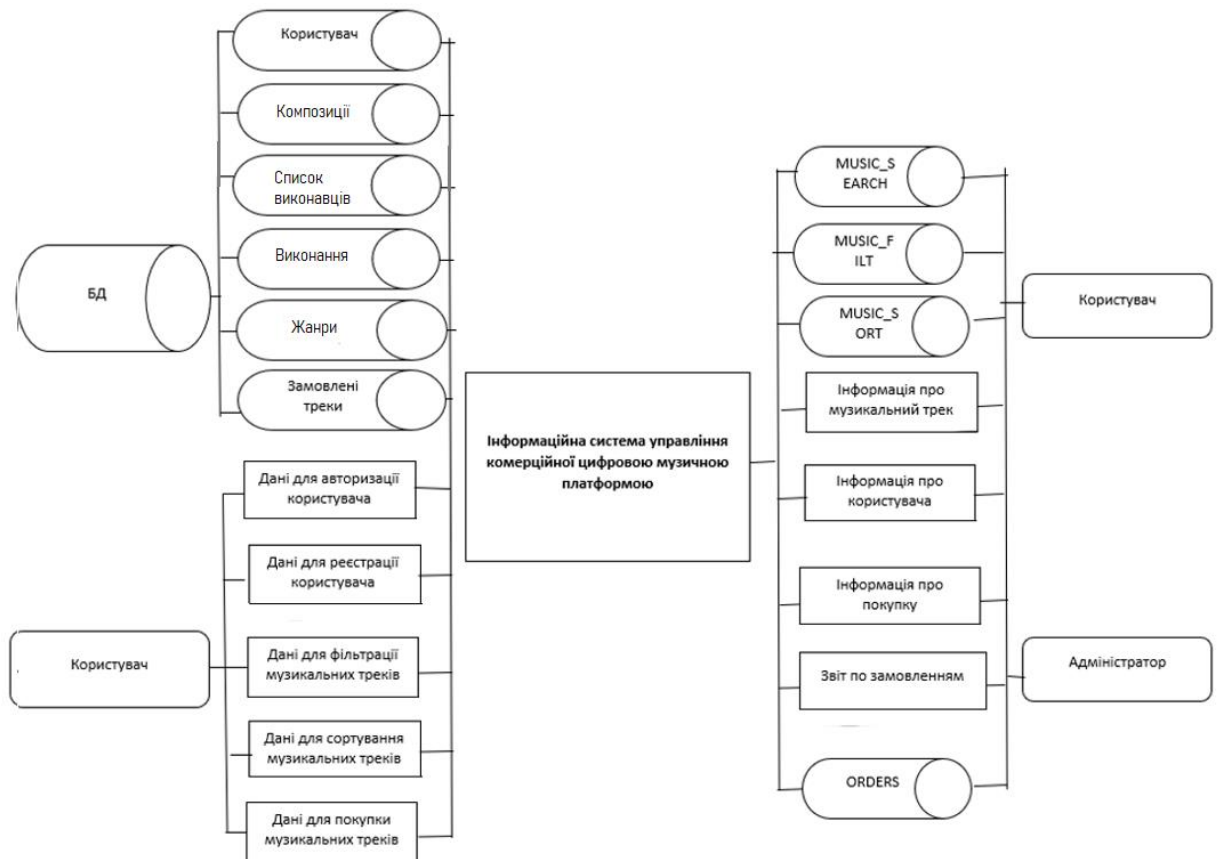


Рис 2.4. - Інформаційна модель розв'язання задачі контролю даних у системі

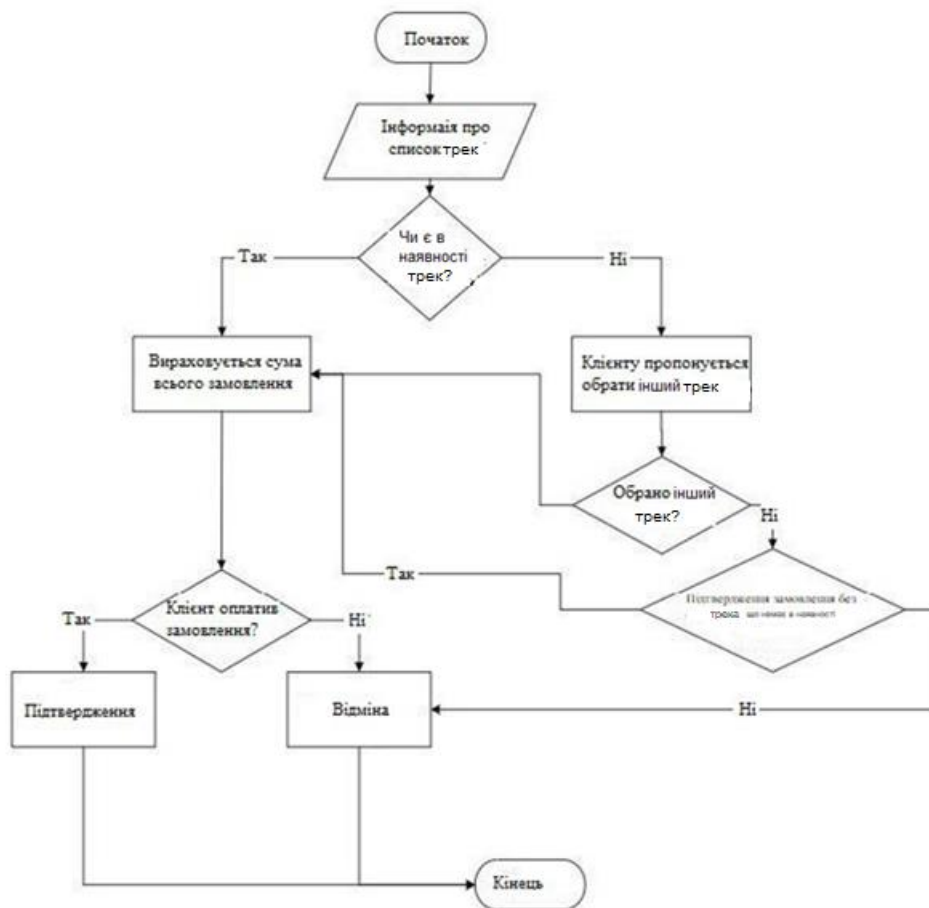


Рис 2.5. - Блок-схема процесу купівлі треку

Використовувана інформація

Інформація, що зберігається в масивах, використовується для коректного виведення інформації про доступні товари та категорії вибору для користувача.

Перелік використовуваних масивів:

- Довідник користувачі
- Довідник композиції
- Довідник Жанр
- Довідник замовлені треки
- Довідник Виконання
- Довідник Список виконавців

Таблиця 2.1 - Перелік масивів використовуваної інформації

Масив	Ідентифікатор	Максимальна кількість записів
Довідник Користувачі	USERS	100000
Довідник Композиції	MUSIC_TRACK	100000
Довідник Жанр	MUSIC_CATEGORY	1000
Довідник Замовлені треки	ORDER_MUSIC	10000
Довідник Виконання	PLAY	100000
Довідник Список виконавців	SINGERS	100000

Результати розв'язання

Перелік і опис масивів результатної інформації подано в табл. 2.2.

Таблиця 2.2 - Перелік масивів результатної інформації

Масив	Ідентифікатор	Максимальна кількість записів
Довідник знайдених треків	MUSIC_SEARCH	100000
Довідник відфільтрованих треків	MUSIC_FILT	100000
Довідник відсортованих треків	MUSIC_SORT	10000
Довідник Замовлених треків	ORDERS	10000

Табл. 2.3 - Перелік вихідної інформації інформації

Назва вихідного повідомлення	Ідентифікатор	Форма подання і вимоги до неї	Періодичність видання	Термін видання і допустимий час затримки	Користувач інформації
Пошук трека	MUSIC_SEARCH	Масив даних	За запитом	Відразу за запитом	Користувач платформи
Сортування треків	MUSIC_SORT	Масив даних	За запитом	Відразу за запитом	Користувач платформи
Фільтрація треків	MUSIC_FILTER	Масив даних	За запитом	Відразу за запитом	Користувач платформи
Інформація про трек	MUSIC_INF	Текстова	За запитом	Відразу за запитом	Користувач платформи
Інформація про користувача	USERS	Текстова	За запитом	Відразу за запитом	Модератор
Інформація про покупку	ORDER_INF	Текстова	За запитом	Відразу за запитом	Модератор
Звіт по замовленням	ORDER_ZVIT	Текстова	За запитом	Відразу за запитом	Модератор
Замовлення	ORDERS	Масив даних	За запитом	Відразу за запитом	Модератор

2.3. Моделювання інформаційної підсистеми

UseCaseDiagram - це спосіб узагальнити деталі системи та користувачів у цій системі. Зазвичай це зображено як графічне зображення взаємодій між різними елементами системи. Діаграми випадків використання визначають події в системі та те, як ці події протікають, однак діаграма випадків використання не описує, як ці події реалізовані.

На діаграмі зображено систему та акторів, що з нею взаємодіють

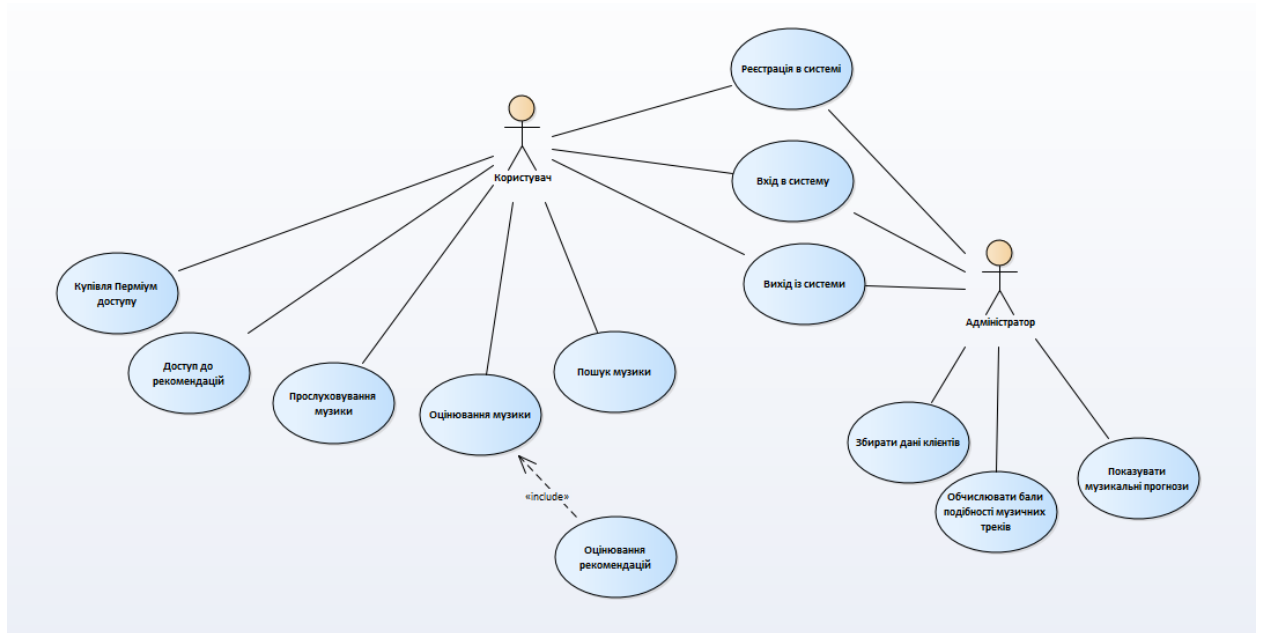


Рис. 2.6- Діаграма прецедентів

Діаграма класів (class diagram) діаграми класів - один із шести типів структурних діаграм.

Діаграми класів - це схеми вашої системи або підсистеми. Ви можете використовувати діаграми класів для моделювання об'єктів, що складають систему, для відображення зв'язків між об'єктами та для опису того, що ці об'єкти роблять, та послуг, які вони надають.

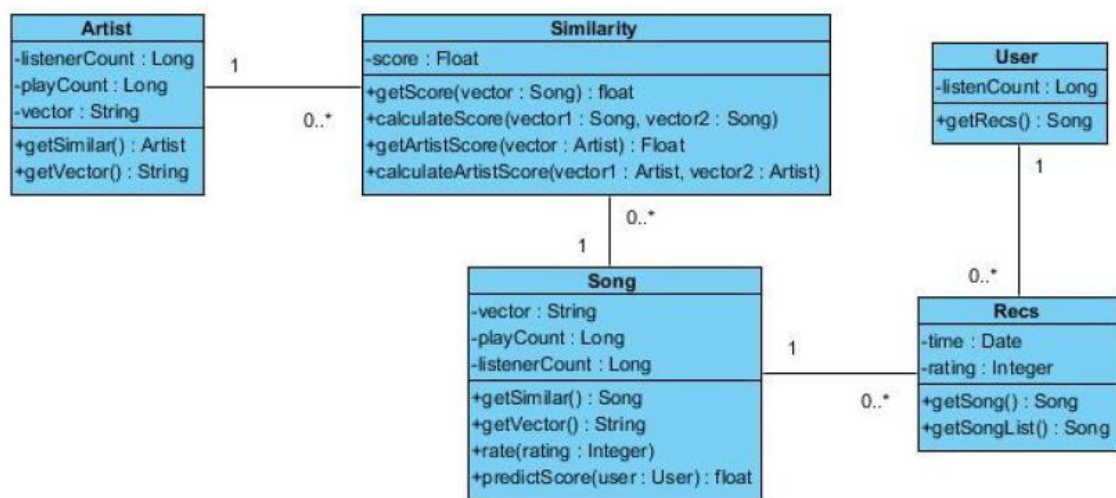


Рис. 2.7. Діаграма класів ІС музичної цифрової платформи

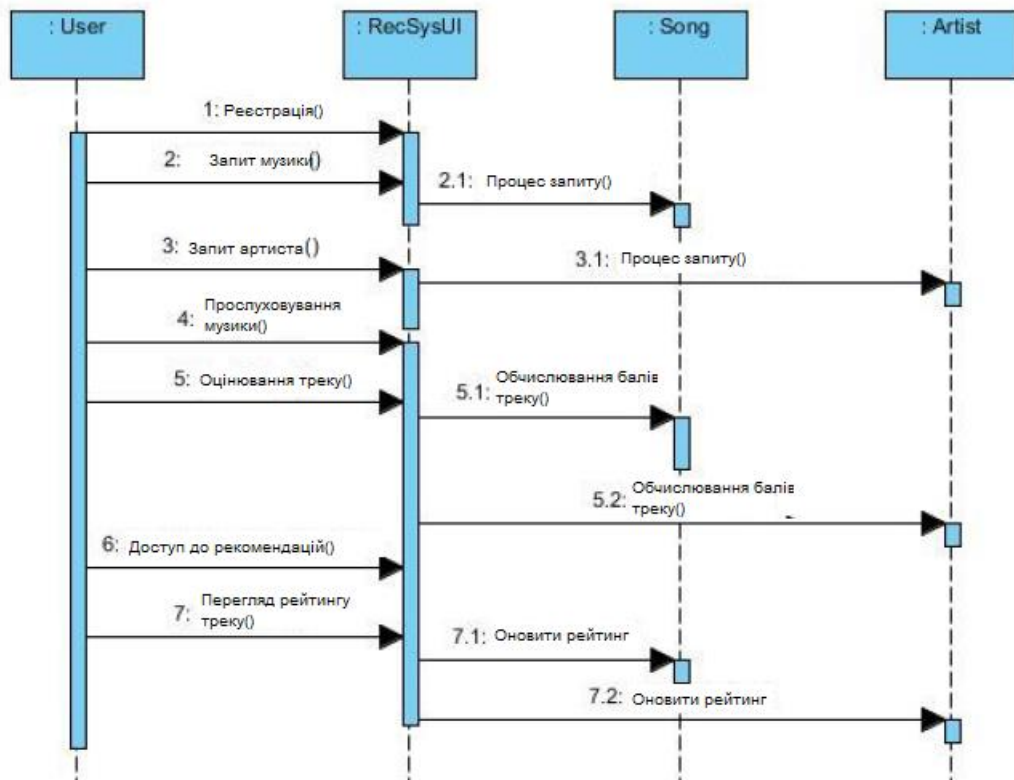


Рис. 2.9. Діаграма послідовності процесу використання музичної цифрової платформи

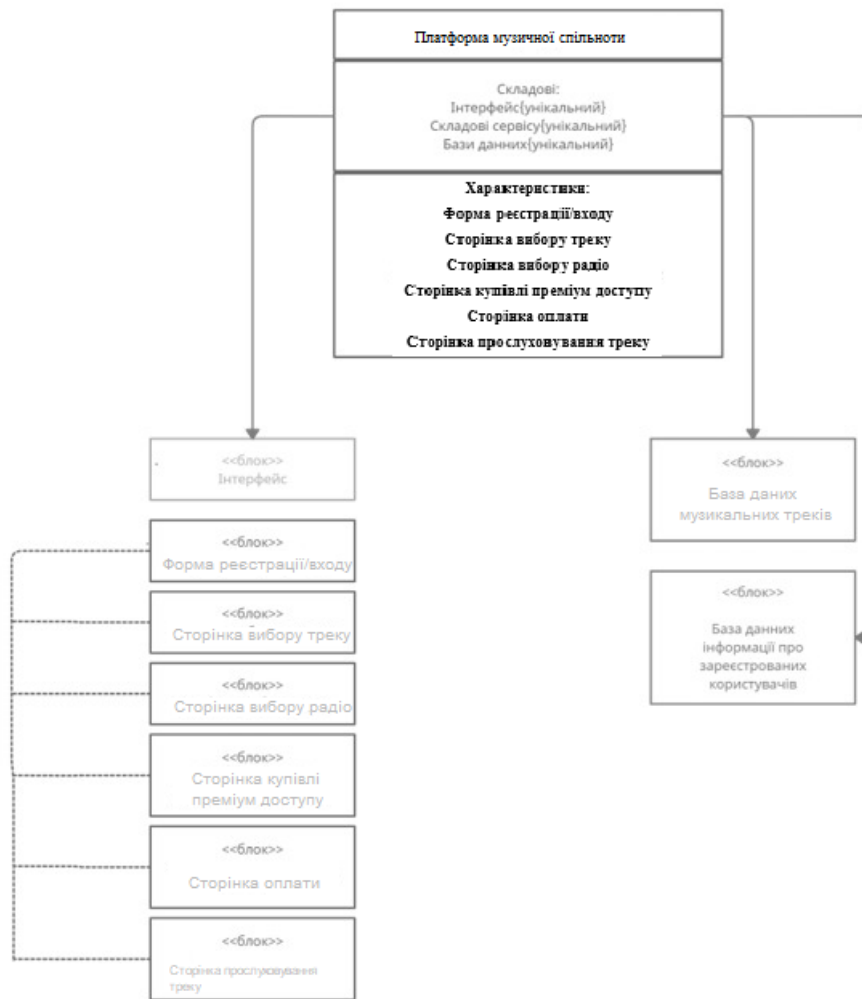


Рис. 2.10. - Діаграма bdd для проектованої системи.



Рис. 2.11. - Діаграма idd для проектованої системи.

РОЗДІЛ 3. ПРОЕКТУВАННЯ КОМПОНЕНТІВ СИСТЕМИ

3.1. Інформаційне забезпечення

Загальна характеристика інформаційного забезпечення.

Інформаційна система - це сукупність технологій по обробці, передачі і обміну даними, що відповідає за безперервне інформаційне забезпечення організації.

Використання єдиної інформаційної системи забезпечує управління робочими процесами, дозволяє зменшити витрати часу, збільшити прибуток і підвищити виробничі показники.

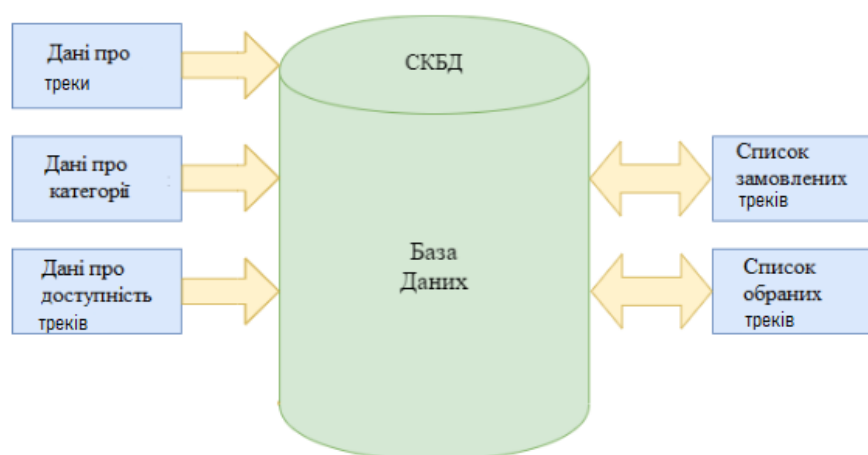


Рис. 3.1 - Структура інформаційного забезпечення

Структура інформаційних масивів

Інформаційний масив - структурна одиниця інформації, що представляє набір даних в формі з усіма їхніми значеннями, або поєднання таких наборів даних, що відносяться до однієї задачі в підсистемі

Процес управління контентом музичної цифрової платформи базується на таких інформаційних об'єктах: Користувач, Жанри, Композиції, Посилання, Виконання, Виконавці

Таблиця 3.1 – Опис масиву Користувач

Опис масиву

Найменування масиву - Користувач

Ідентифікатор масиву - USER

Найменування носія інформації - МД

Максимальний об'єм масиву - 99999

Довжина запису - 50 символів

Метод організації послідовний

Ключі упорядкування KSS

Найменування	Ідентифікатор	Умовне позначення у формулах	Формат	Бізнес-правила				Логічні та систематичні зв'язки
				Первинний(втор.) ключ	Умова на значення	Обов'язкове поле	Індексне поле	
Код користувача	USER_CODE	-	INT(20)	PK	-	Так	-	Zamovlenmya
Ім'я користувача	USER_NAME	-	VARCHAR(50)	-	-	Так	-	-
Логін	USER_LOGIN	-	VARCHAR(50)	-	-	Так	-	-
Телефон	USER_PHONE	-	VARCHAR(11)	-	-	Так	-	-

Таблиця 3.2 – Опис масиву Жанри

Опис масиву

Найменування масиву - Жанр

Ідентифікатор масиву - CATEGORY

Найменування носія інформації - МД

Максимальний об'єм масиву - 99999

Довжина запису - 50 символів

Метод організації послідовний

Ключі упорядкування KSS

Найменування	Ідентифікатор	Умовне позначення у формулах	Формат	Бізнес-правила				Логічні та систематичні зв'язки
				Первинний(втор.) ключ	Умова на значення	Обов'язкове поле	Індексне поле	
Код жанру	KOD_CATEGORY	-	INT(20)	PK	-	Так	-	Song
Назва жанру	NAME_GENRE	-	VARCHAR(50)	-	-	Так	-	-

Таблиця 3.3 -Опис масиву Композиції

Опис масиву
 Найменування масиву - Композиції
 Ідентифікатор масиву - SONG
 Найменування носія інформації - МД
 Максимальний об'єм масиву - 99999
 Довжина запису - 50 символів
 Метод організації послідовний
 Ключі упорядкування KSS

Найменування	Ідентифікатор	Умовне позначення у формулах	Формат	Бізнес-правила				Логічні та систематичні зв'язки
				Первинний(втор.) ключ	Умова на значення	Обов'язкове поле	Індексне поле	
Код композиції	KOD_SONG	-	INT(20)	PK	-	Так	-	LINK, VYKONANNYA
Код жанру	KOD_CATEGORY	-	INT(20)	FK	-	Так	-	CATEGORY
Композиція	SONG	-	VARCHAR(50)	-	-	Так	-	-
Альбом	ALBUM	-	VARCHAR(20)	-	-	Так	-	-

Таблиця 3.4 - Посилання

Опис масиву
 Найменування масиву - Посилання для завантаження
 Ідентифікатор масиву - LINKS
 Найменування носія інформації - МД
 Максимальний об'єм масиву - 99999
 Довжина запису - 100 символів
 Метод організації послідовний
 Ключі упорядкування KSS

Найменування	Ідентифікатор	Умовне позначення у формулах	Формат	Бізнес-правила				Логічні та систематичні зв'язки
				Первинний(втор.) ключ	Умова на значення	Обов'язкове поле	Індексне поле	
Код посилання	KOD_LINK	-	INT(20)	PK	-	Так	-	LINKS
Код композиції	KOD_SONG	-	INT(20)	-	-	Так	-	SONG
Посилання	LINK	-	VARCHAR(100)	-	-	Так	-	-
Розмір	SIZE_LINK	-	VARCHAR(20)	-	-	Так	-	-
Дата	DATA_LINK	-	VARCHAR(20)	-	-	Так	-	-
Оцінка	RATE_LINK	-	VARCHAR(20)	-	-	Так	-	-

Таблиця 3.5 - Виконання

Опис масиву
 Найменування масиву - Виконання
 Ідентифікатор масиву - VYKONANNYA
 Найменування носія інформації - МД
 Максимальний об'єм масиву - 99999
 Довжина запису - 50 символів
 Метод організації послідовний
 Ключі упорядкування KSS

Найменування	Ідентифікатор	Умовне позначення у формулах	Формат	Бізнес-правила				Логічні та систематичні зв'язки
				Первинний(втор.) ключ	Умова на значення	Обов'язкове поле	Індексне поле	
Код виконавця	KOD_SING	-	INT(20)	PK	-	Так	-	SINGER
Код композиції	KOD_SONG	-	INT(20)	PK	-	Так	-	-

Опис масиву

Найменування масиву - Виконавці
 Ідентифікатор масиву - SINGERS
 Найменування носія інформації - МД
 Максимальний об'єм масиву - 99999
 Довжина запису - 50 символів
 Метод організації послідовний
 Ключі упорядкування KSS

Найменування	Ідентифікатор	Умовне позначення у формулах	Формат	Бізнес-правила				Логічні та систематичні зв'язки
				Первинний(втор.) ключ	Умова на значення	Обов'язкове поле	Індексне поле	
Код виконавця	KOD_SING	-	INT(20)	PK	-	Так	-	VYKONANNYA
Виконавець	SINGER	-	INT(20)	-	-	Так	-	-

Вибір СКБД.

MySQL - це система управління реляційними базами даних, заснована на запитах SQL (мова структурованих запитів). Це одна з найпопулярніших мов доступу до записів у таблиці та керування ними. MySQL-це відкрите та безкоштовне програмне забезпечення під ліцензією GNU. Компанія Oracle підтримує дану СКБД.

Найважливіші функції MySQL:

- Система управління реляційними базами даних

MySQL - це система управління реляційними базами даних. Ця мова бази даних базується на запитах SQL для доступу до записів таблиці та керування ними.

- Простий у використанні

MySQL простий у використанні. Ми повинні отримати лише базові знання SQL. Ми можемо створювати та взаємодіяти з MySQL, використовуючи лише кілька простих операторів SQL.

- Це безпечно

MySQL складається з надійного рівня безпеки даних, який захищає конфіденційні дані від зловмисників. Крім того, паролі шифруються в MySQL.

- Архітектура клієнта/ сервера

MySQL стежить за роботою архітектури клієнт/сервер. Існує сервер баз даних (MySQL) і доволіно багато клієнтів (прикладних програм), які спілкуються з сервером; тобто вони можуть запитувати дані, зберігати зміни тощо.

- Безкоштовне завантаження

MySQL можна безкоштовно завантажити з офіційного веб -сайту MySQL.

- MySQL масштабований

MySQL підтримує багатопоточність, що робить його легко масштабованим. Він може обробляти практично будь -який обсяг даних, аж до 50 мільйонів рядків або більше. Обмеження розміру файлу за замовчуванням становить близько 4 ГБ. Однак ми можемо збільшити цю кількість до теоретичної межі 8 ТБ даних.

- Швидкість

MySQL вважається однією з найшвидших мов баз даних, підкріплена великою кількістю бенчмарк -тестів.

- Висока гнучкість

MySQL підтримує велику кількість вбудованих програм, що робить MySQL дуже гнучким.

- Сумісний з багатьма операційними системами

MySQL сумісний з багатьма операційними системами, такими як Novell NetWare, Windows* Linux*, багатьма різновидами UNIX* (наприклад, Sun* Solaris*, AIX та DEC* UNIX), OS/2, FreeBSD* та іншими. MySQL також надає можливість клієнтам працювати на тому ж комп'ютері, що і сервер, або на іншому комп'ютері (зв'язок через локальну мережу або Інтернет).

- Дозволяє відкат

MySQL дозволяє виконувати відкат транзакцій, фіксацію та відновлення після аварійного завершення роботи.

- Ефективність пам'яті

Його ефективність висока, оскільки у нього дуже низька проблема витоку пам'яті.

- Висока працездатність

MySQL швидше, надійніше та дешевше завдяки своїй унікальній архітектурі двигуна зберігання. Він забезпечує дуже високопродуктивні результати в порівнянні з іншими базами даних, не втрачаючи істотної функціональності програмного забезпечення. Він має утиліти швидкого завантаження через різну кеш -пам'ять.

- Висока продуктивність

MySQL використовує тригери, збережені процедури та перегляди, які дозволяють розробнику підвищити продуктивність.

- Платформа Незалежна

Він можезавантажувати, встановлювати та виконувати на більшості доступних операційних систем.

- Підтримка графічного інтерфейсу

MySQL надає єдиний інструмент графічного інтерфейсу візуальної бази даних під назвою "MySQL Workbench" для роботи з архітекторами баз даних, розробниками та адміністраторами баз даних. MySQL Workbench пропонує розробку SQL, моделювання даних, міграцію даних та комплексні інструменти адміністрування для конфігурації сервера, адміністрування користувачів, резервного копіювання та багато іншого. MySQL повністю підтримує графічний інтерфейс від MySQL Server версії 5.6 і вище.

- Підтримка подвійного пароля

MySQL версії 8.0 забезпечує підтримку подвійних паролів: один - це поточний пароль, а інший - вторинний, що дозволяє нам перейти на новий пароль.

Даталогічна модель бази даних

Моделювання даних - це процес створення візуального уявлення або цілої інформаційної системи, або її частин для зв'язку між точками даних та структурами. Мета полягає в тому, щоб проілюструвати типи даних, що використовуються та зберігаються в системі, взаємозв'язки між цими типами даних, способи їх групування та упорядкування та їх формати та атрибути.

Моделі даних побудовані відповідно до потреб бізнесу. Правила та вимоги визначаються заздалегідь шляхом зворотного зв'язку із зацікавленими сторонами бізнесу, щоб вони могли бути включені до проектування нової системи або адаптовані до ітерації існуючої.

Дані можна моделювати на різних рівнях абстракції. Процес починається зі збору інформації про бізнес - вимоги від зацікавлених сторін та кінцевих користувачів. Потім ці бізнес - правила перетворюються на структури даних, щоб сформулювати конкретний дизайн бази даних. Модель даних можна порівняти з дорожньою картою, планом архітектора або будь - якою офіційною схемою, яка полегшує глибше розуміння того, що проектується.

Моделювання даних використовує стандартизовані схеми та формальні методи. Це забезпечує загальний, послідовний та передбачуваний спосіб визначення та управління ресурсами даних у всій організації або навіть за її межами.

Даталогічне проектування - створення схеми бази даних на основі конкретної моделі даних, наприклад, реляційної моделі даних. Для реляційної моделі даних даталогічна модель - набір схем відносин, зазвичай із зазначенням первинних ключів, а також «зв'язків» між відносинами, що представляють собою зовнішні ключі.

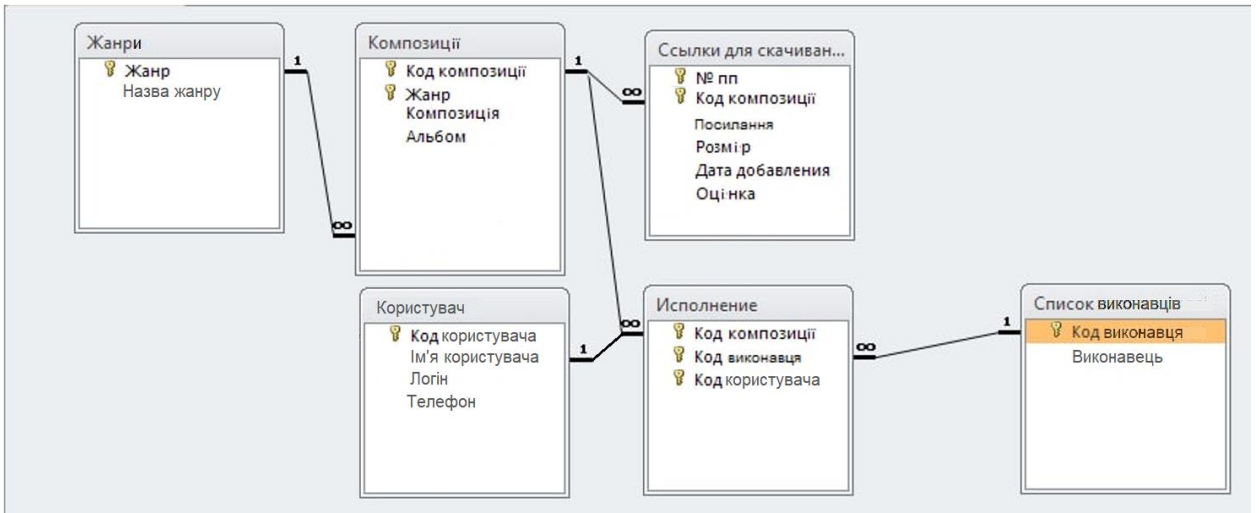


Рис. 3.1. - Даталогічна модель

3.2 Технічне забезпечення

Загальні положення та схема автоматизації.

Технічне забезпечення інформаційної системи включає технології, механізми, які реалізують заданий рівень інформаційної безпеки кожної конкретної мережі, ресурсу, інформаційної системи, компонентів інформаційної інфраструктури шляхом виконання комплексу організаційно-технічних заходів.

При виконанні заходів забезпечення інформаційної системи повинні бути реалізовані вимоги інформаційної системи до інформаційно-технологічних процесів обробки інформації.

Потребують захисту технологічні процеси обробки інформації мають бути однозначно визначені в нормативно-методичних документах відповідної організації фінансової системи.

Результати технологічних операцій з обробки інформації, що захищаються технологічних процесів повинні бути контрольовані і засвідчені уповноваженими особами. Особи, які здійснюють обробку критичної інформації і контроль (перевірку) результатів обробки, повинні бути незалежні один від одного.

Інформаційна система повинна забезпечуватися на всіх стадіях життєвого циклу інформаційних систем, що автоматизують технологічні процеси обробки фінансової інформації.

При замовленні інформаційної системи (ІС) стадії, етапи робіт і процеси, пов'язані з життєвим циклом, рекомендується визначати відповідно до ГОСТ. Власник ІС в процесі її життєвого циклу повинен забезпечити виконання заходів в області захисту інформації з урахуванням встановлених вимог.

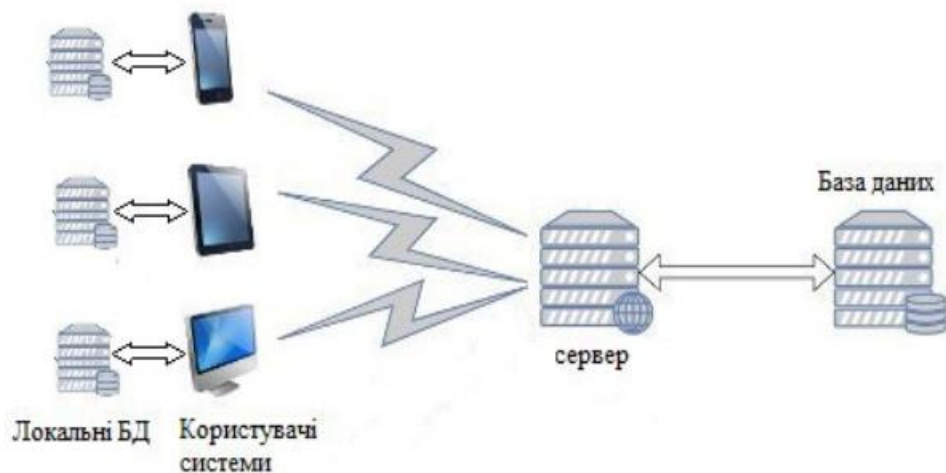


Рис 3.2. - Загальна схема автоматизації

3.3. Програмне забезпечення

Структура програмного забезпечення

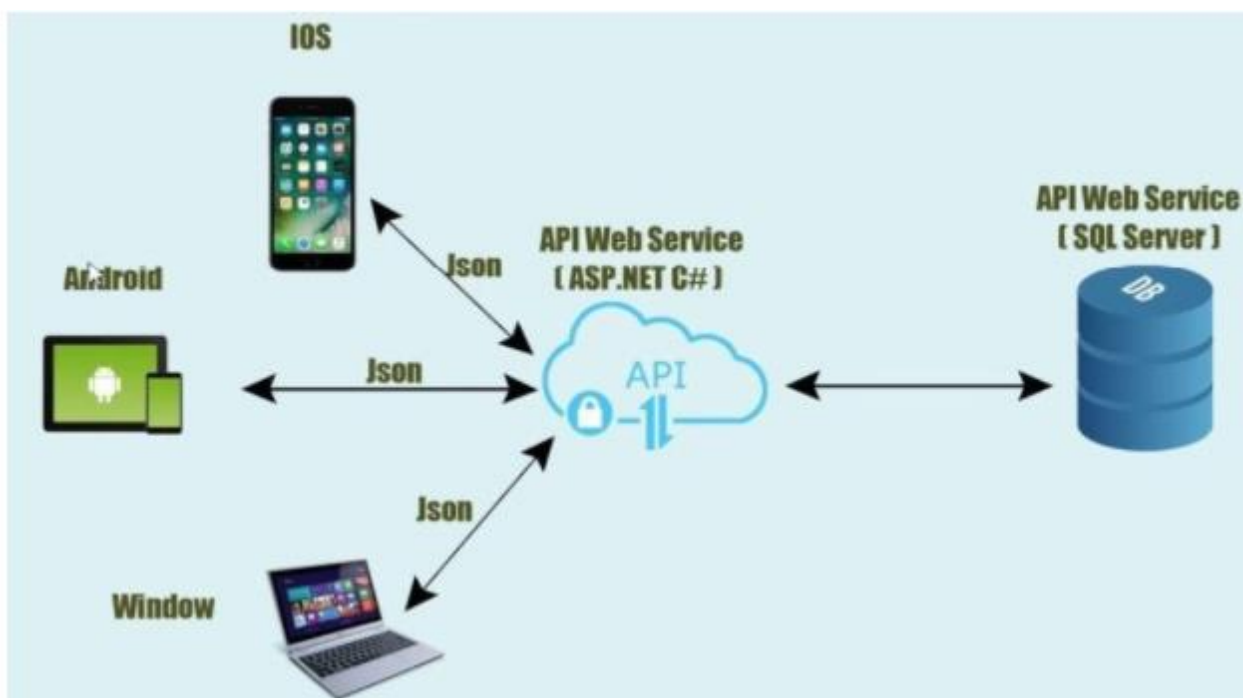


Рис 3.3. - Структура програмного забезпечення

Інструментом автоматизації збірки даної системи буде Gradle. Gradle-це інструмент автоматизації збірки з відкритим кодом, розроблений таким чином, щоб бути достатньо гнучким для створення практично будь-якого типу забезпечення. Нижче наведено огляд на високому рівні деяких найважливіших його особливостей:

- **Висока працездатність**

Gradle уникає непотрібної роботи, лише виконуючи завдання, які необхідно виконати, оскільки їхні вхідні або вихідні дані змінилися. Ви також можете використовувати кеш збірки, щоб увімкнути повторне використання результатів завдань із попередніх запусків або навіть з іншої машини (зі спільним кешем збірки).

- **Конвенції**

Gradle бере листок з книги Maven і спрощує створення загальних типів проектів, таких як проекти Java, шляхом реалізації конвенцій.

- **Підтримка IDE**

Кілька основних IDE дозволяють імпортувати збірки Gradle та взаємодіяти з ними: Android Studio, IntelliJ IDEA, Eclipse та NetBeans. Gradle також підтримує створення файлів рішень, необхідних для завантаження проекту у Visual Studio.

Бібліотека класів .NET Framework-це всебічна об'єктно-орієнтована колекція типів багаторазового використання, яку можна використовувати для розробки програм. Бібліотека класів .NET Framework містить ADO.NET, ASP.NET та Windows Forms..NET Framework - це система розробки програмного забезпечення, розроблена та підтримувана корпорацією Майкрософт для спрощення інженерії веб -додатків. Це популярна безкоштовна платформа, яка зараз використовується для багатьох різних типів додатків, оскільки вона забезпечує середовище програмування для більшості етапів розробки програмного забезпечення.

.NET Framework базується на об'єктно-орієнтованому програмуванні (ООП). ООП - це модель розробки для розбиття програмного забезпечення на менші частини, які легше управляти та поєднувати. ООП розділяє дані на об'єкти, тобто поля даних, і описує поведінку та вміст об'єктів за допомогою оголошення класів.

Модульна структура допомагає визначити поведінку об'єктів та їх взаємодію без управління внутрішніми атрибутами об'єкта. ООП спрощує

програмування, роблячи код керованим, легшим для тестування та реагуванням на повторювані проблеми.

3.4 Результати реалізації інформаційної підсистеми

У ході роботи була розроблена інформаційна система комерційної цифрової музикальної платформи. Головна сторінка виглядаю наступним чином:

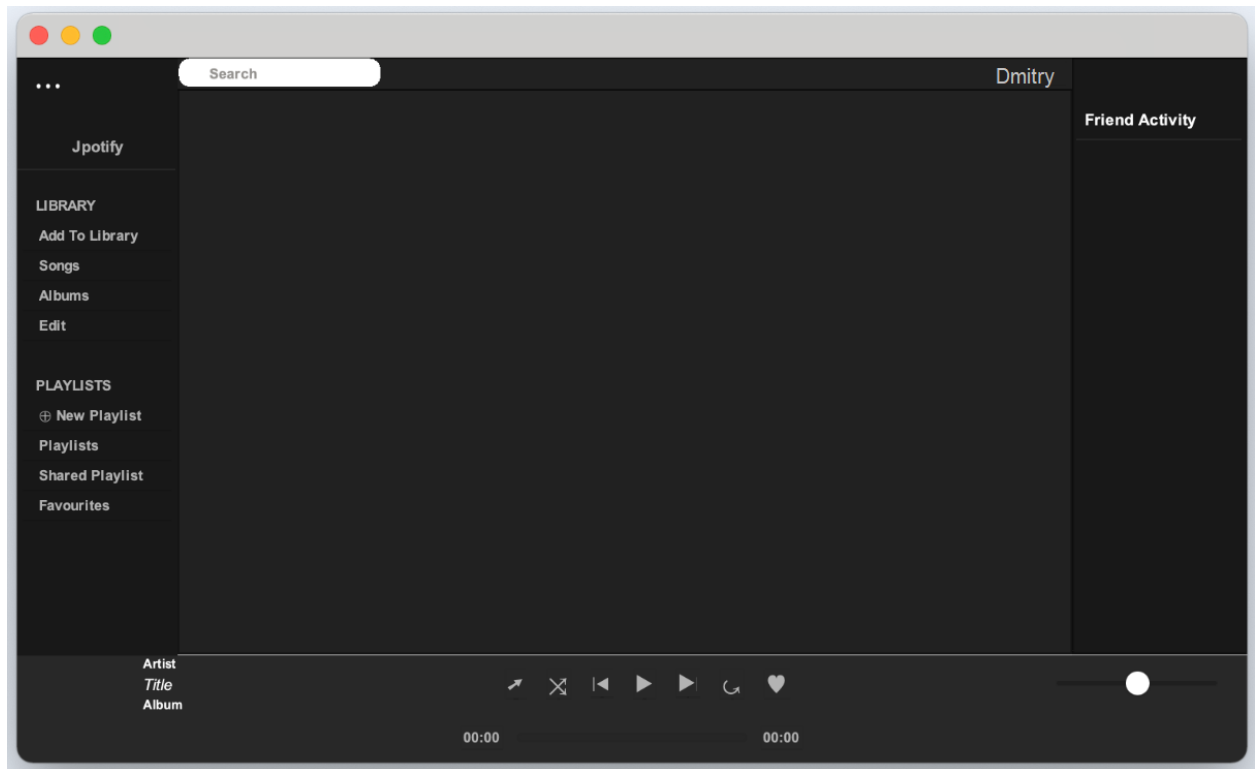


Рис 3.5. - Головна сторінка

Користувач може обрати потрібний йому альбом на сторінці з альбомами:

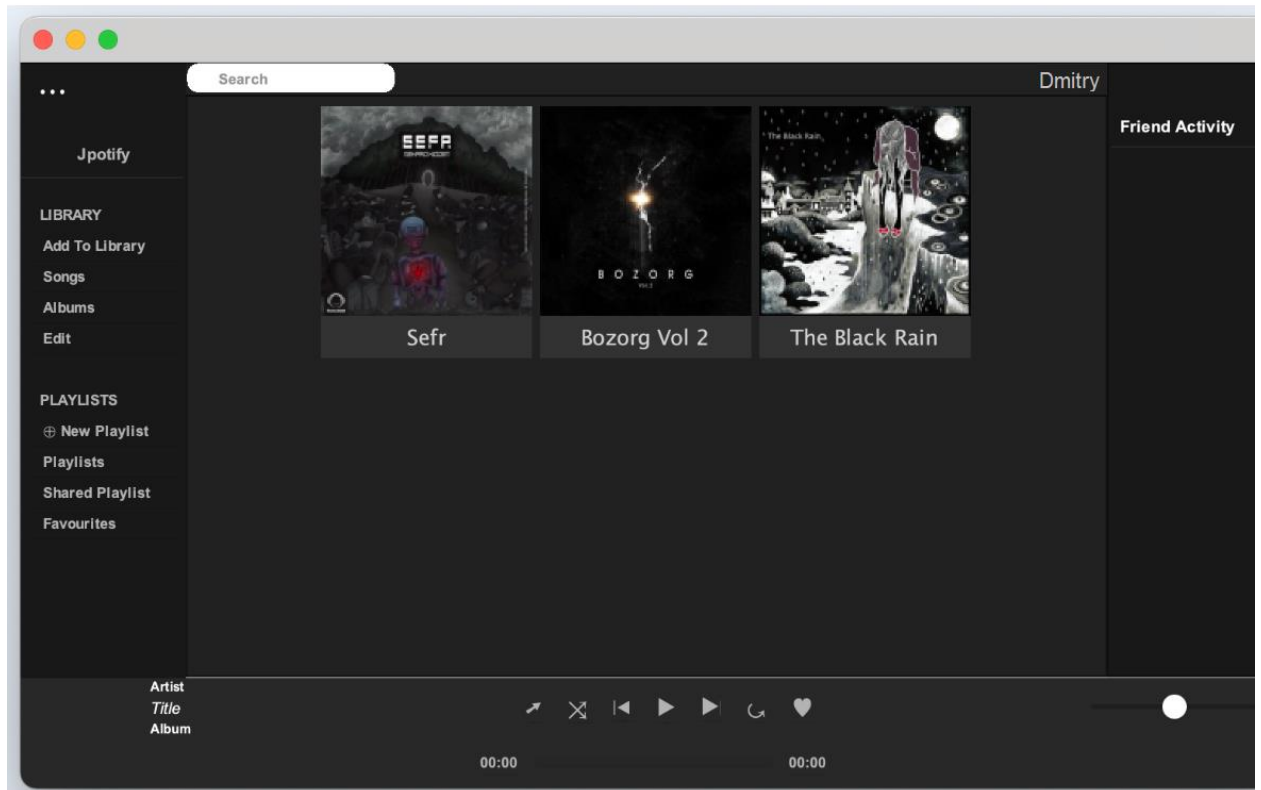


Рис.3.6 - Сторінка вибору альбому

Вікно з програвачем виглядає наступним чином:

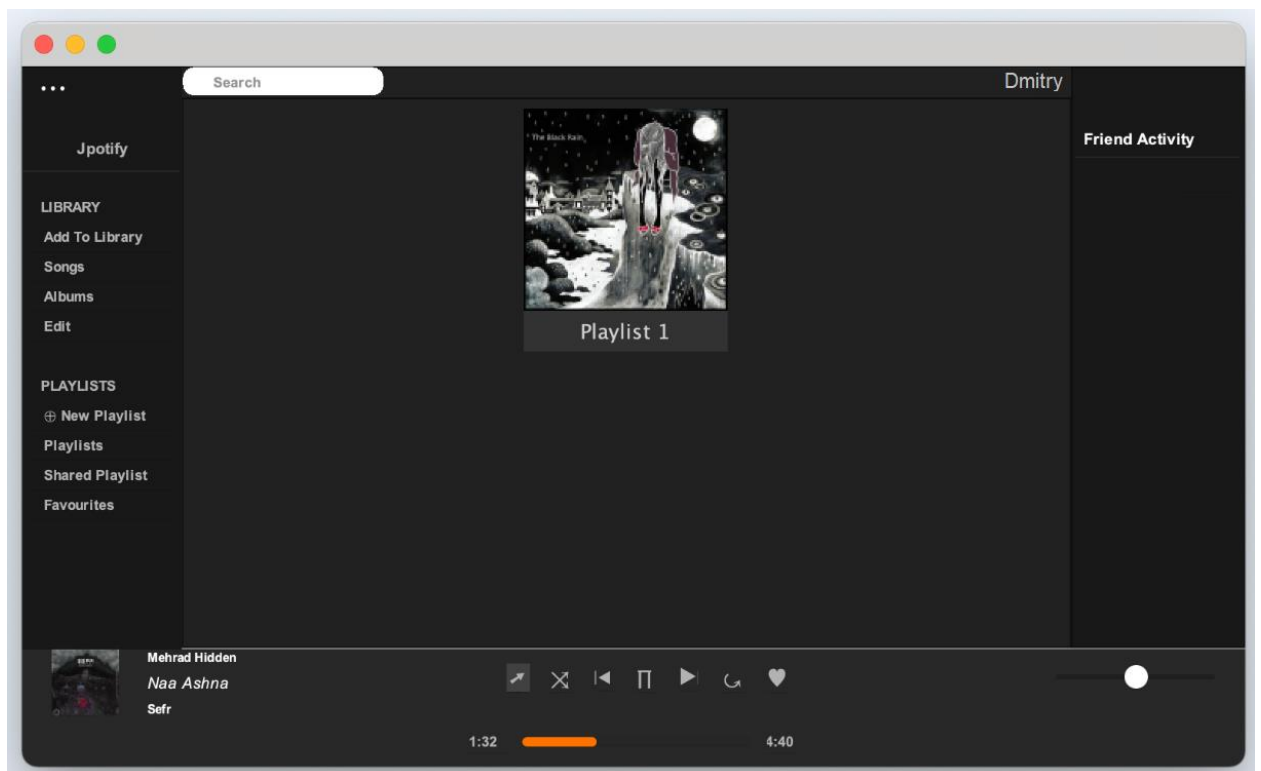


Рис.3.7- Вікно програвача

Вікно вибору пісень:

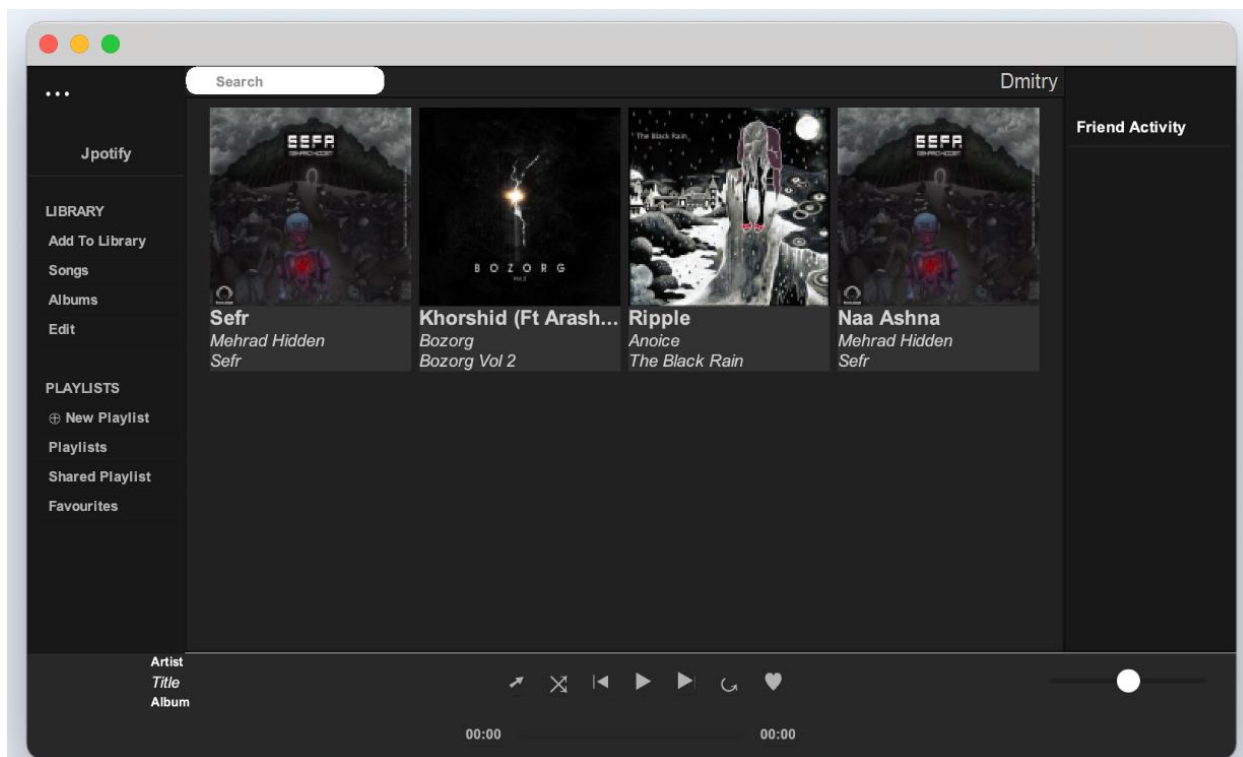


Рис.3.8- Вікно вибору пісень

ВИСНОВКИ

Інтернет назавжди змінив ландшафт музичної індустрії. Це загальновідомо для артистів, які використовували ефективні цифрові технології, такі як обмін файлами та цифровий маркетинг через нескінченну кількість каналів у соціальних мережах. Хоча все це може здатися очевидною інформацією, факт залишається фактом, що багато музикантів все ще борються з вибором правильних онлайн -платформ для завантаження, обміну та просування своєї музики.

У ході роботи було проведено дослідження існуючих інформаційних систем музикальних цифрових платформ, які зараз є досить популярними. Проаналізовано історію музикальних цифрових спільнот. Проведено аналіз технологій, за допомогою яких розроблюються електроні музикальні платформи. Також було визначено характеристики існуючих інформаційних систем. У ході роботи було спроектовано та побудовано БД для інформаційної системи комерційної музикальної платформи.

За допомогою розробленого алгоритму роботи серверної частини Веб-додатку можна редагувати, зберігати і отримувати із бази даних дані. Розроблений зручний інтерфейс для користувачів системи.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Автоматизовані інформаційні системи, бази і банки даних. Вступний курс: підруч. посіб. – М. : Геліос АРВ, 2002. – 368 с.
2. Басс Л. Архітектура програмного забезпечення на практиці / Л. Басс, П. Клементс, Р. Кацман. – СПб.: Пітер, 2006 – 576 с.
3. Береза А.М., Козак І.А. Проектування систем оброблення інформації: Навч. посіб. – К.: КНЕУ, 2008. – 448 с.
4. Бондаренко М. Ф. Моделирование и проектирование бизнес-систем: методы, стандарты, технологии: учебн. пособ. / М. Ф. Бондаренко, С. И. Маторин, Е. А. Соловьев. – Х.: Компания СМИТ, 2004. – 272 с.
5. Вигерс К. Разработка требований к программному обеспечению / К. Вигерс; пер. англ. – М.: Изд.-торговый дом "Русская редакция", 2004. – 576 с.
6. Гриценко В. Г. UML-моделювання інформаційно-аналітичної системи / В. Г. Гриценко, Г. В. Луценко. // Черкаський національний університет імені Богдана Хмельницького. – 2011.
7. Денісова О. О. Автоматизоване проектування інформаційних систем : Навчальний посібник – Київ: КНЕУ, 2011. – с.400-413
8. Павлиш В. А. Основи інформаційних технологій і систем: Навчальний посібник. / Павлиш В. А., Гліненко Л. К. - Львів: Видавництво Львівської політехніки, 2013. – 500 с.
9. Системи оброблення економічної інформації: навч. посіб. /за заг. ред. М.А. Сендзюка, М.І. Татарчука. – К. :КНЕУ, 2010. – 455с.
10. Ситник Н.В. Банківські інформаційні системи: Навч. посіб. – К.: КНЕУ, 2008, – 384с.
11. Ситник Н.В., Краснюк М.Т. Проектування баз і сховищ даних: Навчально-методичний посібник для самостійного вивчення дисципліни – Київ: КНЕУ, 2005. – 264 с.
12. Ситник Н.В. Проектування баз і сховищ даних: Нав. посіб. – К.:2004. – 348

13. Снитюк В. Е. Прогнозирование. Модели, Методы, Алгоритмы / В. Е. Снитюк. – К.: «Маклаут», 2008. – 364 с.
14. Структура інформаційної системи: апаратна та інформаційна складові, їх взаємодія [Електронний ресурс]. – Режим доступу: <http://www.eduforme.org/mod/page/view.php?id=721>
15. Design Patterns: Elements of Reusable Object-Oriented Software / E.Gamma, J. Vlissides, R. Johnson, R. Helm., 1994. – 395 с.
16. Larman C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development / Craig Larman., 2004. – 736с.
17. Risby F. Functional programming in javascript / Franklin Risby.
18. Wikipedia. Інформаційне забезпечення [Електронний ресурс] / Wikipedia. – 2020. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Інформаційне_забезпечення.
19. Wikipedia. Apache Maven [Електронний ресурс] / Wikipedia. – 2020. – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Apache_Maven
20. Wikipedia. Git [Електронний ресурс] / Wikipedia. – 2020. – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/Git>

ДОДАТКИ

Додаток 1

Програмний код інформаційної системи

```
package model;

import java.util.ArrayList;

/**
 * albums are managed through album class
 */
public class Album {

    private String albumName;
    ArrayList<Song> albumSongs = new ArrayList<>();

    public Album(String albumName) {
        this.albumName = albumName;
    }

    public String getAlbumName() {
        return albumName;
    }

    public ArrayList<Song>getAlbumSongs() {
        return albumSongs;
    }

    public void setAlbumName(String albumName) {
        this.albumName = albumName;
    }

    public void setAlbumSongs(ArrayList<Song> albumSongs) {
        this.albumSongs = albumSongs;
    }

    /**
     * add a song to album's songs arraylist
     *
     * @param song
     */
    public void addSong(Song song) {
        albumSongs.add(song);
    }

    public void removeSong(Song song) {
        if (albumSongs != null && albumSongs.contains(song)) {
            albumSongs.remove(song);
        }
    }
}
```

```

        }

        /**
        * this method gets first song of album for album artwork
        *
        * @return
        */
        public Song getFirstSong() {
            if (getAlbumSongs().size() != 0)
                return albumSongs.get(0);
            else {
                return null;
            }
        }
    }

}

package model;

import java.io.Serializable;

/**
* playlist is actually a tag not a class.
* and each song can be added to many playlists so each song has a playlist arraylist
*/
public class Playlist implements Serializable {

    private String playlistName;

    public Playlist(String playlistName) {
        this.playlistName = playlistName;
    }

    /**
    * @return the name of playlist
    */
    public String getPlaylistName() {
        return playlistName;
    }

    public void setPlaylistName(String playlistName) {
        this.playlistName = playlistName;
    }

}

package controller;

import javax.sound.sampled.*;
import javax.sound.sampled.FloatControl.Type;
import javax.swing.*;

```

```

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

import static javax.sound.sampled.AudioSystem.getClip;

public class AudioController {

    public static void setMasterOutputVolume(float value) {
        if (value <0 || value >1)
            throw new IllegalArgumentException(
"Volume can only be set to a value from 0 to 1. Given value is illegal: " + value);
        Line line = getMasterOutputLine();
        if (line == null) throw new RuntimeException("No Master Output");
        boolean opened = open(line);
        try {
            FloatControl control = getVolumeControl(line);
            if (control == null)
                throw new RuntimeException("No Volume Control : " + toString(line));
            control.setValue(value);
        } finally {
            if (opened) line.close();
        }
    }

    public static Float getMasterOutputVolume() {
        Line line = getMasterOutputLine();
        if (line == null) return null;
        boolean opened = open(line);
        try {
            FloatControl control = getVolumeControl(line);
            if (control == null) return null;
            return control.getValue();
        } finally {
            if (opened) line.close();
        }
    }

    public static Line getMasterOutputLine() {
        for (Mixer mixer : getMixers()) {
            for (Line line : getAvailableOutputLines(mixer)) {
                if (line.getLineInfo().toString().contains("SPEAKER")) return line;
            }
        }
        return null;
    }

    public static FloatControl getVolumeControl(Line line) {

```

```

if (!line.isOpen()) throw new RuntimeException("Line is closed: " + toString(line));
return (FloatControl) findControl(FloatControl.Type.VOLUME, line.getControls());
    }

```

```

private static Control findControl(Control.Type type, Control... controls) {
    if (controls == null || controls.length == 0) return null;
    for (Control control : controls) {
        if (control.getType().equals(type)) return control;
        if (control instanceof CompoundControl) {
            CompoundControl compoundControl = (CompoundControl) control;
            Control member = findControl(type, compoundControl.getMemberControls());
            if (member != null) return member;
        }
    }
    return null;
}

```

```

public static List<Mixer>getMixers() {
    Mixer.Info[] infos = AudioSystem.getMixerInfo();
    List<Mixer> mixers = new ArrayList<Mixer>(infos.length);
    for (Mixer.Info info : infos) {
        Mixer mixer = AudioSystem.getMixer(info);
        mixers.add(mixer);
    }
    return mixers;
}

```

```

public static List<Line>getAvailableOutputLines(Mixer mixer) {
    return getAvailableLines(mixer, mixer.getTargetLineInfo());
}

```

```

private static List<Line>getAvailableLines(Mixer mixer, Line.Info[] lineInfos) {
    List<Line> lines = new ArrayList<Line>(lineInfos.length);
    for (Line.Info lineInfo : lineInfos) {
        Line line;
        line = getLineIfAvailable(mixer, lineInfo);
        if (line != null) lines.add(line);
    }
    return lines;
}

```

```

public static Line getLineIfAvailable(Mixer mixer, Line.Info lineInfo) {
    try {
        return mixer.getLine(lineInfo);
    } catch (LineUnavailableException ex) {
        return null;
    }
}

```

```

public static boolean open(Line line) {
    if (line.isOpen()) return false;
}

```

```

        try {
            line.open();
        } catch (LineUnavailableException ex) {
            return false;
        }
        return true;
    }

    public static String toString(Control control) {
        if (control == null) return null;
        return control.toString() + " (" + control.getType().toString() + ")";
    }

    public static String toString(Line line) {
        if (line == null) return null;
        Line.Info info = line.getLineInfo();
        return info.toString();
    }

    public static String toString(Mixer mixer) {
        if (mixer == null) return null;
        StringBuilder sb = new StringBuilder();
        Mixer.Info info = mixer.getMixerInfo();
        sb.append(info.getName());
        sb.append(" ").append(info.getDescription()).append(" ");
        sb.append(mixer.isOpen() ? "[open]" : "[closed]");
        return sb.toString();
    }

    }

    package controller;

    import javazoom.jl.decoder.JavaLayerException;
    import javazoom.jl.player.Player;

    import java.io.InputStream;

    public class PlayPartController {

        private final static int NOTSTARTED = 0;
        private final static int PLAYING = 1;
        private final static int PAUSED = 2;
        private final static int FINISHED = 3;

        // the player actually doing all the work
        private final Player player;

        // locking object used to communicate with player thread
        private final Object playerLock = new Object();

```

```

// status variable what player thread is doing/supposed to do
private int playerStatus = NOTSTARTED;

public PlayPartController(final InputStream inputStream) throws JavaLayerException {
    this.player = new Player(inputStream);
}

/**
 * Starts playback (resumes if paused)
 */
public void play( ) throws JavaLayerException {
    synchronized (playerLock) {
        switch (playerStatus) {
            case NOTSTARTED:
                final Runnable r = new Runnable() {
                    public void run() {
                        playInternal();
                    }
                };
                final Thread t = new Thread(r);
                t.setDaemon(true);
                t.setPriority(Thread.MAX_PRIORITY);
                playerStatus = PLAYING;
                t.start();
                break;
            case PAUSED:
                resume();
                break;
            default:
                break;
        }
    }
}

/**
 * Pauses playback. Returns true if new state is PAUSED.
 */
public boolean pause() {
    synchronized (playerLock) {
        if (playerStatus == PLAYING) {
            playerStatus = PAUSED;
        }
        return playerStatus == PAUSED;
    }
}

/**
 * Resumes playback. Returns true if the new state is PLAYING.
 */
public boolean resume() {
    synchronized (playerLock) {

```

```

        if (playerStatus == PAUSED) {
            playerStatus = PLAYING;
            playerLock.notifyAll();
        }
        return playerStatus == PLAYING;
    }
}

/**
 * Stops playback. If not playing, does nothing
 */
public void stop() {
    synchronized (playerLock) {
        playerStatus = FINISHED;
        playerLock.notifyAll();
    }
}

private void playInternal( ) {
    while (playerStatus != FINISHED) {
        try {
            if (!player.play(1)) {
                break;
            }
        } catch (final JavaLayerException e) {
            break;
        }
        // check if paused or terminated
        synchronized (playerLock) {
            while (playerStatus == PAUSED) {
                try {
                    playerLock.wait();
                } catch (final InterruptedException e) {
                    // terminate player
                    break;
                }
            }
        }
        close();
    }
}

/**
 * Closes the player, regardless of current state.
 */
public void close() {
    synchronized (playerLock) {
        playerStatus = FINISHED;
    }
    try {
        player.close();
    } catch (final Exception e) {

```

```

// ignore, we are terminating anyway
    }
    }

    }
package network.client;

import model.Song;

import java.io.*;
import java.net.Socket;

public class FileClient {

    private Socket socket;
    private Song song;

public FileClient(String host, int port, String file) {

    song = new Song(file);
    try {
socket = new Socket(host, port);
    sendFile(file);
    } catch (Exception e) {
e.printStackTrace();
    }
    }

public void sendFile(String file) throws IOException {

    PrintWriter printWriter = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()));
    printWriter.println(song.getTitle());
    printWriter.flush();

    printWriter.println(song.getBytes());
    printWriter.flush();

DataOutputStream dos = new DataOutputStream(socket.getOutputStream());
    FileInputStream fis = new FileInputStream(file);

    byte[] buffer = new byte[song.getBytes()];

    while (fis.read(buffer) >0) {
        dos.write(buffer);
    }

    fis.close();
    dos.close();
    }
}

```

```

    }
    package network.server;

    import view.Friends.FriendActivity;

    import java.io.*;
    import java.net.Socket;

    public class ClientHandler implements Runnable {

        private Socket client;
        private FriendActivity friendActivity;

    public ClientHandler(Socket client, FriendActivity friendActivity) {
        this.client = client;
        this.friendActivity = friendActivity;
    }

        @Override
        public void run() {

            try {
                saveFile(client, friendActivity);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

    private void saveFile(Socket client, FriendActivity friendActivity) throws IOException {

        BufferedReader bufferedReader = null;
        try {
            bufferedReader = new BufferedReader(new
            InputStreamReader(client.getInputStream()));
        } catch (IOException e) {
            e.printStackTrace();
        }

        String input = null;
        try {
            input = bufferedReader.readLine();
        } catch (IOException e) {
            e.printStackTrace();
        }

        int bytes = 0;

        try {
            bytes = Integer.parseInt(bufferedReader.readLine());
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

```

```

        }

        DataInputStream dis = new DataInputStream(client.getInputStream());

        FileOutputStream fos = new FileOutputStream(input + ".mp3");
        byte[] buffer = new byte[bytes];

        int filesize = bytes; // Send file size in separate msg
        int read = 0;
        int totalRead = 0;
        int remaining = filesize;
        while ((read = dis.read(buffer, 0, Math.min(buffer.length, remaining))) > 0) {
            totalRead += read;
            remaining -= read;
            fos.write(buffer, 0, read);
        }

        friendActivity.addNewParticipant("/Users/apple/IdeaProjects/Jpotify1/" + input + ".mp3");

        fos.close();
        dis.close();

    }
}

package network.server;

import view.Friends.FriendActivity;

import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;

public class FileServer extends Thread {

    private ServerSocket socket;
    private FriendActivity friendActivity;

    public FileServer(int port, FriendActivity friendActivity) {
        this.friendActivity = friendActivity;
        try {
            socket = new ServerSocket(port);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void run() {
        while (true) {
            try {
                Socket client = socket.accept();

                ClientHandler clientHandler = new ClientHandler(client, friendActivity);

```

```

Thread td = new Thread(clientHandler);
    td.start();

    } catch (IOException e) {
        e.printStackTrace();
    }
    }
}

package view.Center;

import model.Album;
import model.Song;
import view.Player.PlayerPart;

import javax.swing.*;
import java.awt.*;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.util.ArrayList;
import java.util.Objects;

public class AlbumBtn extends JPanel {

private ArrayList<String> playlistNames;
    private String playlistName;
    private PlayerPart playerPart;
    private ArrayList<Song> songs;
private ArrayList<Song> showSongs;
    private Color background;
    private Color foreground;
    private Color pressedBackground;
    private Font font1;
    private JLabel name;
    private JPanel data;
    private Album album;
    private JLabel icon;

public AlbumBtn(ArrayList<Song> songs, Album album, PlayerPart playerPart, ShowPanel
    showPanel) {

    super();

    this.album = album;
    this.playerPart = playerPart;

    this.songs = songs;
    this.showSongs = new ArrayList<>();

```

```

this.setPreferredSize(new Dimension(130, 155));

    background = new Color(50, 50, 50);
    foreground = new Color(210, 210, 210);
    pressedBackground = new Color(65, 65, 65);
    this.setBackground(background);
    this.setLayout(new BorderLayout());
    this.font1 = new Font("Arial", Font.BOLD, 13);

    data = new JPanel(new FlowLayout());
    data.setBackground(background);
    data.setForeground(foreground);

    name = new JLabel(album.getAlbumName());
    data.add(name);
    name.setForeground(foreground);
    name.setBackground(background);

    icon = new JLabel();

ImageIcon albumImgIcon = new ImageIcon("/Users/apple/Desktop/userIcon.png");

    if (album.getFirstSong() != null) {

        if (album.getFirstSong().getArtwork() != null) {
            albumImgIcon = new ImageIcon(album.getFirstSong().getArtwork());
        } else {
            albumImgIcon = new ImageIcon("/Users/apple/Desktop/userIcon.png");
        }
        Image img = albumImgIcon.getImage().getScaledInstance(130, 130,
            java.awt.Image.SCALE_SMOOTH);
        Icon icon1 = new ImageIcon(img);
        icon.setIcon(icon1);

    }

    for (int i = 0; i < songs.size(); i++) {

        if (Objects.equals(songs.get(i).getAlbumName(), album.getAlbumName())) {
            showSongs.add(songs.get(i));
            if (songs.get(i).getArtwork() != null) {
                albumImgIcon = new ImageIcon(songs.get(i).getArtwork());
            }
        }

    }

    Image img = albumImgIcon.getImage().getScaledInstance(130, 130,
        java.awt.Image.SCALE_SMOOTH);
    Icon icon1 = new ImageIcon(img);
    icon.setIcon(icon1);

```

```

        this.add(icon, BorderLayout.CENTER);

this.addMouseListener(new MouseListener() {
    @Override
    public void mouseClicked(MouseEvent e) {
        }

        @Override
    public void mousePressed(MouseEvent e) {
        name.setBackground(pressedBackground);
        data.setBackground(pressedBackground);
        }

        @Override
    public void mouseReleased(MouseEvent e) {
        name.setBackground(background);
        data.setBackground(background);
        showPanel.removeAll();
        showPanel.repaint();
        showPanel.setSongs(showSongs);
        showPanel.revalidate();
        }

        @Override
    public void mouseEntered(MouseEvent e) {
        }

        @Override
    public void mouseExited(MouseEvent e) {
        }
    });

    this.add(data, BorderLayout.SOUTH);
    }

}
package view.Center;

import view.Library.LibraryPart;
import view.Player.PlayerPart;

import javax.swing.*;
import java.awt.*;

/**
 * CentralPanel at the center of the MainPage frame
 * this panel includes a search bar and a main panel for showing songs and albums and other
 * features of the program
 */
public class CentralPanel extends JPanel {

```

```

        private ToolBar toolBar;
        private ShowPanel showPanel;

    public CentralPanel(String userName, PlayerPart playerPart, LibraryPart libraryPart) {
        super();

        this.setLayout(new BorderLayout());
        this.setBackground(new Color(33, 33, 33));

        showPanel = new ShowPanel(playerPart);
        JScrollPane jScrollPane = new JScrollPane(showPanel);
        jScrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);
        jScrollPane.setBackground(new Color(33, 33, 33));
        this.add(jScrollPane, BorderLayout.CENTER);

        toolBar = new ToolBar(userName, libraryPart, showPanel);
        toolBar.setSize(new Dimension(50, 10));
        this.add(toolBar, BorderLayout.NORTH);

    }

    public void setShowPanel(ShowPanel showPanel) {

        this.showPanel = showPanel;
    }

    public ShowPanel getShowPanel() {
        return showPanel;
    }
}
package view.Center;

import model.Song;
import view.Library.EditPlaylist;
import view.Library.LibraryPart;
import view.Player.PlayerPart;

import javax.swing.*;
import java.awt.*;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.util.ArrayList;
import java.util.Objects;

/**
 * playlists are shown in the center through jpanels
 */
public class PlaylistBtn extends JPanel {

    private JLabel icon;

```

```

private ArrayList<String> playlistNames;
    private String playlistName;
    private PlayerPart playerPart;
    private ArrayList<Song> songs;
private ArrayList<Song> showSongs;
    private Color background;
    private Color foreground;
    private Color pressedBackground;
    private Font font1;
    private JLabel name;
    private JPanel data;
    private EditPlaylist editPlaylist;

public PlaylistBtn(ArrayList<Song> songs, String playlistName, PlayerPart playerPart,
    ShowPanel showPanel, LibraryPart libraryPart) {

    super();
    this.playerPart = playerPart;
    this.playlistName = playlistName;
    this.songs = songs;
    this.showSongs = new ArrayList<>();

    this.setPreferredSize(new Dimension(130, 155));

    background = new Color(50, 50, 50);
    foreground = new Color(210, 210, 210);
    pressedBackground = new Color(65, 65, 65);
    this.setBackground(background);
    this.font1 = new Font("Arial", Font.BOLD, 13);

    data = new JPanel(new FlowLayout());
    data.setBackground(background);
    data.setForeground(foreground);
    this.setLayout(new BorderLayout());

    name = new JLabel(playlistName);
    data.add(name);
    name.setForeground(foreground);
    name.setBackground(background);

    for (int i = 0; i < songs.size(); i++) {

        for (int j = 0; j < songs.get(i).getPlaylists().size(); j++) {

            if (songs.get(i).getPlaylists().get(j) != null) {
            if (Objects.equals(songs.get(i).getPlaylists().get(j), playlistName)) {
                showSongs.add(songs.get(i));
            }
        }
    }
}
}
}

```

```

        }

        icon = new JLabel();

        if (showSongs.size() != 0) {
            ImageIcon albumImgIcon;
            if (showSongs.get(0).getArtwork() != null) {
                albumImgIcon = new ImageIcon(showSongs.get(0).getArtwork());
            } else {
                albumImgIcon = new ImageIcon("/Users/apple/Desktop/userIcon.png");
            }
            Image img = albumImgIcon.getImage().getScaledInstance(130, 130,
                java.awt.Image.SCALE_SMOOTH);
            Icon icon1 = new ImageIcon(img);
            icon.setIcon(icon1);
        }
        this.add(icon, BorderLayout.CENTER);

        this.addMouseListener(new MouseListener() {
            @Override
            public void mouseClicked(MouseEvent e) {
                }

                @Override
                public void mousePressed(MouseEvent e) {
                    name.setBackground(pressedBackground);
                    data.setBackground(pressedBackground);
                }

                @Override
                public void mouseReleased(MouseEvent e) {
                    name.setBackground(background);
                    data.setBackground(background);
                    showPanel.removeAll();
                    showPanel.repaint();
                    showPanel.setSongs(showSongs);
                    showPanel.revalidate();
                }

                @Override
                public void mouseEntered(MouseEvent e) {
                }

                @Override
                public void mouseExited(MouseEvent e) {
                }
            });

        this.addMouseListener(new MouseAdapter() {

```

```

        @Override
        public void mousePressed(MouseEvent me) {
            if (me.getButton() == MouseEvent.BUTTON3) {
                editPlaylist = new EditPlaylist(songs, playlistName, libraryPart);
            }
        }
    });

    this.add(data, BorderLayout.SOUTH);

}

public ArrayList<Song>getShowSongs() {
    return showSongs;
}

public PlaylistBtn getPlaylistBtnItSelf() {
    return this;
}

public String getPlaylistName() {
    return playlistName;
}

public void setPlaylistName(String playlistName) {
    this.playlistName = playlistName;
}

}

package view.Center;

import javax.swing.*;
import java.awt.*;
import java.awt.event.FocusEvent;
import java.awt.event.FocusListener;
import java.awt.geom.RoundRectangle2D;

/**
 * this is programs search bar
 */
public class RoundTextField extends JTextField {
    private static JTextField empty;
    private Color color;
    private Color backgroundColor;

    private Shape shape;

    public RoundTextField(int size) {
        super(size);
    }
}

```

```

        empty = new JTextField();
        backgroundColor = new Color(33, 33, 33);

        color = new Color(148, 146, 143);
        setOpaque(false);
        RoundTextField.super.setText("🔍 Search");
RoundTextField.super.setFont(new Font("Arial", Font.CENTER_BASELINE, 9));
        RoundTextField.super.setForeground(color);
    }

    @Override
    protected void paintComponent(Graphics g) {
        g.setColor(Color.white);
        g.fillRect(0, 0, getWidth() - 1, getHeight() - 1, 15, 15);
        super.paintComponent(g);
    }

    @Override
    protected void paintBorder(Graphics g) {
        g.setColor(backgroundColor);
        g.drawRoundRect(0, 0, getWidth() - 1, getHeight() - 1, 15, 15);
    }

    @Override
    public boolean contains(int x, int y) {
        if (shape == null || !shape.getBounds().equals(getBounds())) {
            shape = new RoundRectangle2D.Float(0, 0, getWidth() - 1, getHeight() - 1, 15, 15);
        }
        return shape.contains(x, y);
    }

    public void setDefaultText(String s) {
        super.addFocusListener(new FocusListener() {
            public void focusGained(FocusEvent e) {
                RoundTextField.super.setText("");
                RoundTextField.super.setForeground(color);
            }
        });

        public void focusLost(FocusEvent e) {
            RoundTextField.super.setText(s);
            RoundTextField.super.setForeground(color);
        }
    }

    empty.requestFocus();
}

package view.Center;

import model.Album;
import model.Song;
import view.Library.LibraryPart;
import view.Player.PlayerPart;

```

```

import javax.swing.*;
import java.awt.*;
import java.util.ArrayList;

/**
 * this class is a container which shows songs, albums, playlists and etc in the middle part of the
 program
 * also it connects to program's player and library.
 */
public class ShowPanel extends JPanel {

    private Color background;
    private JButton[] albumButtons;
    private PlaylistBtn[] playlistBtns;
    private SongBtn[] songBtns;
    private AlbumBtn[] albumBtns;
    private PlayerPart playerPart;
    ArrayList<Song> songs = new ArrayList<>();
    ArrayList<Album> albums = new ArrayList<>();
    private ArrayList<String> playlistNames;

    public ShowPanel(PlayerPart playerPart) {

        super();
        this.playerPart = playerPart;
        background = new Color(33, 33, 33);
        this.setLayout(new FlowLayout());
        this.setBackground(background);

    }

    /**
     * shows song
     *
     * @param songs
     */
    public void setSongs(ArrayList<Song> songs) {

        this.setPreferredSize(new Dimension(300, ((songs.size() / 4) + 1) * 170));

        this.songs = null;
        this.songs = songs;
        this.songBtns = null;
        this.songBtns = new SongBtn[songs.size()];

        for (int i = 0; i < songs.size(); i++) {

            songBtns[i] = new SongBtn(playerPart, songs, i);
            add(songBtns[i]);
        }
    }
}

```

```

        }
        playerPart.setAlbums(albums);
        setVisible(true);
    }

    /**
     * shows albums
     *
     * @param albums
     */
    public void setAlbums(ArrayList<Album> albums, ArrayList<Song> songs) {

        this.setPreferredSize(new Dimension(300, ((songs.size() / 4) + 1) * 155));

        this.albums = albums;
        albumBtns = new AlbumBtn[albums.size()];

        for (int i = 0; i < albums.size(); i++) {

            albumBtns[i] = new AlbumBtn(songs, albums.get(i), playerPart, this);
            add(albumBtns[i]);

        }
        playerPart.setAlbums(albums);
        this.setVisible(true);
    }

    /**
     * shows playlists
     *
     * @param playlistNames
     * @param songs
     */
    public void setPlaylists(ArrayList<String> playlistNames, ArrayList<Song> songs, LibraryPart
        libraryPart) {

        this.setPreferredSize(new Dimension(300, ((songs.size() / 4) + 1) * 155));

        this.playlistNames = playlistNames;
        playlistBtns = new PlaylistBtn[playlistNames.size()];

        for (int i = 0; i < playlistNames.size(); i++) {

            playlistBtns[i] = new PlaylistBtn(songs, playlistNames.get(i), playerPart, this,
                libraryPart);
            add(playlistBtns[i]);

        }
        this.setVisible(true);
    }
}
package view.Center;

```

```

import model.Song;
import view.Player.PlayerPart;

import javax.swing.*;
import java.awt.*;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.util.ArrayList;

/**
 * this jpanel connects songs to player and songs can be played by clicking them here
 */
public class SongBtn extends JPanel {

private ArrayList<Song> songs;
private Song song;
private int index;
private PlayerPart playerPart;
private JLabel icon;
private JLabel title;
private JLabel album;
private JLabel artist;
private JPanel data;
private Color background;
private Color foreground;
private Color pressedBackground;
private Font font1;
private Font font2;

public SongBtn(PlayerPart playerPart, ArrayList<Song> songs, int index) {

super();

this.songs = songs;
this.index = index;
this.song = songs.get(index);
this.playerPart = playerPart;

background = new Color(50, 50, 50);
foreground = new Color(210, 210, 210);
pressedBackground = new Color(65, 65, 65);

this.setLayout(new BorderLayout());
this.setBackground(background);
this.font1 = new Font("Arial", Font.BOLD, 13);
this.font2 = new Font("Arial", Font.ITALIC, 11);
this.setPreferredSize(new Dimension(130, 170));

icon = new JLabel();

```

```

icon.setHorizontalAlignment(SwingConstants.CENTER);
    ImageIcon imgIcon;
    if (song.getArtwork() != null) {
        imgIcon = new ImageIcon(song.getArtwork());
    } else {
imgIcon = new ImageIcon("/Users/apple/Desktop/userIcon.png");
    }
    Image img = imgIcon.getImage().getScaledInstance(130, 130,
        java.awt.Image.SCALE_SMOOTH);
    Icon icon1 = new ImageIcon(img);
    icon.setIcon(icon1);
    icon.setForeground(foreground);
    this.add(icon, BorderLayout.CENTER);

    data = new JPanel();
    data.setLayout(new BorderLayout());
    data.setBackground(background);

    title = new JLabel(song.getTitle());
    title.setForeground(foreground);
    title.setFont(font1);
    data.add(title, BorderLayout.NORTH);

    artist = new JLabel(song.getArtistName());
    artist.setForeground(foreground);
    artist.setFont(font2);
    data.add(artist, BorderLayout.CENTER);

    album = new JLabel(song.getAlbumName());
    album.setForeground(foreground);
    album.setFont(font2);
    data.add(album, BorderLayout.SOUTH);

    this.add(data, BorderLayout.SOUTH);

this.addMouseListener(new MouseListener() {
    @Override
    public void mouseClicked(MouseEvent e) {
    }

    @Override
    public void mousePressed(MouseEvent e) {
        icon.setBackground(pressedBackground);
        data.setBackground(pressedBackground);
    }

    @Override
    public void mouseReleased(MouseEvent e) {
        icon.setBackground(background);
        data.setBackground(background);
        playerPart.setSongs(songs, index);
    }
}

```

```

        @Override
        public void mouseEntered(MouseEvent e) {
            }

        @Override
        public void mouseExited(MouseEvent e) {
            }
        });

    }

}

package view.Center;

import model.Song;
import view.Library.LibraryPart;

import javax.swing.*;
import java.awt.*;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.util.ArrayList;

/**
 * ToolBar panel at the top of the CentralPanel and MainPage
 */
public class ToolBar extends JPanel {

    private RoundTextField searchBox;
    private JTextField userNameTextField;
    private ArrayList<Song> searchedSongs = new ArrayList<>();

    public ToolBar(String userName, LibraryPart libraryPart, ShowPanel showPanel) {
        super();
        this.setLayout(new BorderLayout(15, 15));
        this.setBackground(new Color(33, 33, 33));
        this.setPreferredSize(new Dimension(400, 20));
        this.setMaximumSize(new Dimension(160, 700));

        searchBox = new RoundTextField(15);
        searchBox.setDefaultText("🔍 Search");
        this.add(searchBox, BorderLayout.WEST);

        this.userNameTextField = new JTextField(userName);
        userNameTextField.setFont(new Font("TimesNewRoman", Font.LAYOUT_RIGHT_TO_LEFT,
            9));
        userNameTextField.setForeground(new Color(179, 179, 179));
        userNameTextField.setBackground(new Color(33, 33, 33, 0));
    }
}

```

```

        userNameTextField.setEditable(false);
this.add(userNameTextField, BorderLayout.EAST);

searchBox.addKeyListener(new KeyAdapter() {
    @Override
    public void keyPressed(KeyEvent e) {
        super.keyPressed(e);
        if (e.getKeyCode() == KeyEvent.VK_ENTER) {
            searchedSongs = null;
            searchedSongs = new ArrayList<>();
            String search = searchBox.getText();
            if (search.equals("") || search.equals("🔍 Search")) {
                showPanel.removeAll();
                showPanel.repaint();
                showPanel.revalidate();
            } else {

                for (Song song : libraryPart.getSongs()) {
                    if (song.getTitle().contains(search.toString()) ||
                        song.getAlbumName().contains(search.toString()) ||
                        song.getArtistName().contains(search.toString())) {
                        searchedSongs.add(song);
                    }
                }
                showPanel.removeAll();
                showPanel.repaint();
                showPanel.setSongs(searchedSongs);
                showPanel.revalidate();
            }
        }
    }
});
revalidate();
}
}

```