НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри

комп'ютерних наук

_____/Голуб Б.Л., доц., к.т.н. /

Підпис ПІБ,вчене звання і ступінь

«___»_____2025 р

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Інформаційна облікова система рецензування музики

Спеціальність: 122 - «Комп'ютерні науки»

Гарант освітньої програми

Д.е.н., професор _____ Руденський Р.А.

Керівник бакалаврської кваліфікаційної роботи

к.т.н. доцент Дудник А.О.

науковий ступінь та вчене звання підпис ПІБ

Виконав _____ Капінус Б.С.

підпис ПІБ

Київ — 2025

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ

Завідувач кафедри

Комп'ютерних наук

Голуб Б.Л., доц., к.т.н.

«___»_____2025 р

З А В Д А Н Н Я

на виконання бакалавратської кваліфікаційної роботи

Капінусу Богдану Сергійовичу

Спеціальність: 122 - «Комп'ютерні науки»
Тема бакалаврської кваліфікаційної роботи Інформаційна облікова система рецензування музики
затверджена наказом ректора НУБіП України від "16" 12 2024 р. № 2246"С"

Термін подання завершеної роботи на кафедру 2025.06.02

Вихідні дані до бакалаврської кваліфікаційної роботи:

Результати аналізу літератури, інтернет джерел.

Перелік питань, які потрібно розробити:

1. ДОСЛІДЖЕННЯ ІНФОРМАЦІЇНИХ СИСТЕМ РЕЦЕНЗУВАННЯ МУЗИКИ 2. МОДЕЛЮВАННЯ ОБЛІКОВОЇ СИСТЕМИ РЕЦЕНЗУВАННЯ МУЗИЧНОГО КОНТЕНТУ 3. СТВОРЕННЯ ПРОГРАМНОГО ПРОДУКТУ СИСТЕМИ РЕЦЕНЗАВАННЯ МУЗИКИ 4. ТЕСТУВАННЯ ТА ІНТЕГРАЦІЯ ІНФОРМАЦІЇНОЇ СИСТЕМИ 5. ВИСНОВКИ

Дата видачі завдання "16" 12 2025 р

Керівник бакалаврської кваліфікаційної роботи

к.т.н. доц. Дудник А.О.

науковий ступінь та вчене звання підпис ПІБ

Завдання прийняв до виконання Капінус Б.С.

підпис ПІБ

АНОТАЦІЯ

Бакалаврська робота на тему «Інформаційно-облікова система для рецензування музики» присвячена автоматизації обробки, управління, рецензування та оцінки контенту.

Метою роботи є розробка десктопного додатку, що забезпечує ефективне управління та оцінювання контенту, а також взаємодію з базами даних.

Для розробки я використовував мову програмування C#, фреймворк WPF (Windows Presentation Foundation) та базу даних MySQL.

В результаті було розроблено функціональний додаток, який дозволяє керувати та оцінювати контент, додавати та зберігати новий контент, а також має зручний інтерфейс з чітким дизайном.

Пояснювальна записка містить 73 сторінки, 26 рисунки, 5 таблиці, 33 список використаних джерел та 2 додатки.

ABSTRACT

The bachelor's thesis on ‘Information and Accounting System for Music Reviewing’ is dedicated to automating the processing, management, reviewing and evaluation of content.

The goal is to develop a desktop application that provides effective content management and evaluation and interaction with databases.

For development, I used the C# programming language, the WPF (Windows Presentation Foundation) framework, and the MySQL database.

As a result, a functional application was developed that allows you to manage and evaluate content, add and save new content, and has a user-friendly interface with a clear design.

The explanatory note contains 73 pages, 26 figures, 5 tables, 33 references and 2 appendices.

Зміст

Вступ

Актуальність теми: У сучасному світі щогодини з’являється нова музика, альбоми чи кліпи, тому збільшується потреба систематизації та оцінювання такого музичного контенту. З появою багатьох музичних та відео платформ та стримінгових сервісів у людей виникає потреба, необхідність в таких інструментах, що дозволяють рецензувати та оцінювати музичний контент за різними критеріями. Спеціалізованим десктопним додатком для професійного рецензування та оцінювання музичного контенту є актуальним завданням, яке реально відповідає потребам не тільки музичних критиків, а й музикантів та звичайних людей.

Метою було розробити десктопний додаток CloudRate для операційної системи Windows, що дозволяє в будь-який зручний час кожному, хто захоче, додати та прочитати рецензії, переглянути оцінки або комплексно оцінити та рецензувати той музичний контент, що він і хоче, з можливістю повністю проаналізувати різні аспекти.

Задачі дослідження:

1. Провести аналіз існуючих систем оцінювання музичного контенту та визначити їх переваги та недоліки.
2. Розробити архітектуру додатку з використанням новітніх технологій (.NET Framework, WPF).
3. Створити список критеріїв, по яким будуть оцінюватись музичні твори.
4. Реалізувати функціональність зберігання чи обробки різних видів музичних елементів.
5. Розробити зручний інтерфейс для додавання та перегляду рецензій та оцінок.
6. Забезпечити максимально надійну передачу, отримання та зберігання даних у базі даних MySQL.

7. Реалізувати систему розрахунку середніх оцінок та формування статистики.
8. Провести тестування та оптимізацію роботи додатку.

Об’єктом дослідження є процеси оцінювання за допомогою критеріїв та рецензування музики в цифровому середовищі.

Методи дослідження:

- системний аналіз предметної області та існуючих рішень;
- проектування бази даних;
- розробка інтерфейсу;
- тестування програмного забезпечення;
- новітні методи перевірки працездатності системи.

При пошуку програмного забезпечення, яке б закривало потреби користувачів у рецензуванні та оцінці контенту, було знайдено застаріле схоже програмне забезпечення.

Новизна проекту полягає в розробці спеціального додатку для комп’ютерів, який одночасно поєднав сучасні технології розробки (.NET Framework, WPF) та комплексну систему оцінювання музичного контенту, що дозволяє проводити аналіз всіх можливих аспектів музичних творів.

Розроблений додаток буде використовуватися:

- музикантами для розуміння аудиторії;
- музичними критиками для рецензій, що будуть пояснювати проблеми та гарні сторони та оцінки контенту;
- любителями музики для викладення думок з можливістю пояснень та оцінюваннями;
- музичними платформами для колаборації та впровадження системи оцінювання всередину своїх платформ;
- дослідниками для аналізу музичних трендів та вподобань різних аудиторій;
- музичними та відео продюсерами для розуміння та покращення музичних знань.

Також система сприяє розвитку культури та критичного мислення в слухачів та підвішує музичний рівень серед різноманітної аудиторії, а також допомагає менш

популярним виконавцям чи гуртам за допомогою виводу оцінок з справедливим результатом просувати свій контент.

Розроблені методи та підходи можуть бути застосовані для створення інших систем рецензування та оцінювання в майбутньому, а також для створення схожих систем для оцінювання інших видів контенту чи об’єктів.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ІНФОРМАЦІЇНИХ СИСТЕМ РЕЦЕНЗУВАННЯ МУЗИКИ

1.1 Мета розробки

На даний момент музичний ринок стрімко зростає — велика кількість нового музичного контенту (пісень, альбомів чи кліпів) з’являється щохвилини. Це вже створює великі проблеми з можливістю описати свої думки, систематизовано оцінити та написати рецензію на улюбленого артиста чи його контент не тільки звичайному користувачу, а й музичним критикам чи музичним продюсерам.

Існує багато інших додатків чи вебсервісів, що вже мають схожий функціонал, але вони або сильно спрощують систему, або, навпаки, занадто ускладнюють свій функціонал.

Основні проблеми, які вирішує моя система, — це вирішення проблеми з критеріями оцінки, яка була розроблена мною. На даний момент ще не існує такої системи, яка має єдину правильну систему оцінювання. Тому різні користувачі змушені використовувати різні платформи для найкращого оцінювання, що призводить до розділення даних, а також ускладнює аналіз.

Це і буде вирішувати моя система — вона допомагає автоматизувати процес оцінювання та зробити його максимально об’єктивним. Система, зберігаючи дані про оцінки та рецензії користувачів, дозволяє ефективно працювати з музичним контентом.

Моя система оцінює такі аспекти, як:

- музика — як звучить пісня без вокалу самого артиста;
- вокал — як звучить голос окремо від музики;
- текст — наскільки текст підходить до вибраної мелодії та голосу артиста;
- темп — який темп вибраний для цієї пісні;
- індивідуальність — наскільки артист звучить оригінально, своєрідно.
- реалізація — наскільки артист реалізував задумку та музику у цій пісні;

- атмосфера — яке відчуття отримує слухач після прослуховування вибраного треку.

Це дозволяє швидко та комплексно оцінити не лише музику, а й альбоми чи кліпи.

Функціональні можливості системи:

- Реєстрація та авторизація.
- Додавання музичних елементів (альбомів, пісень, кліпів).
- Пошук музичних елементів.
- Перегляд рецензій та оцінок.
- Додавання рецензій та оцінок.
- Рейтинг по популярності всіх музичних елементів, за рейтингом та рецензіями.

У системі я створив два основних зовнішніх акторів

1. Рецензатор (Користувач)
    1. Це зареєстрований користувач, який взаємодіє з усіма музичними елементами системи.
2. Модератор (Адміністратор)
    2. Це користувач, що слідкує за рецензаторами, їхніми діями, доданим контентом та написаними рецензіями

Для кращого розуміння я додав рис.1, на якому можна побачити діаграму варіантів використання (Use Case Diagram).

Діаграма варіантів використання (Use Case Diagram), це діаграма, яка є одним з основних інструментів для візуалізації функціональних можливостей системи в UML (Unified Modeling Language).[11]

Вона показує взаємодію між зовнішніми користувачами системи (акторів) та її функціональністю (варіантами використання) в межах певної системи. Ця діаграма слугує способом подання системи через набір зовнішніх суб’єктів, які взаємодіють з нею, виконуючи певні сценарії.

Актори — це користувачі або інші системи, що мають справу із системою. Варіанти використання описують, які дії може виконати система у відповідь на запити акторів, проте без деталізації, яким чином ці дії реалізуються технічно.

Графічно така діаграма складається з овальних елементів (що представляють варіанти використання), фігур, які позначають акторів, та межі системи, яка окреслює область відповідальності самої системи.

На моїй діаграмі варіантів рис. 1 використання можна побачити рецензатора, що може додавати новий контент, шукати потрібні елементи, переглядати елемент, переглядати популярний контент, переглядати рецензії та створювати рецензії. А також в системі є модератор, який може редагувати та видаляти елементи, видаляти користувачів та створювати звіти.

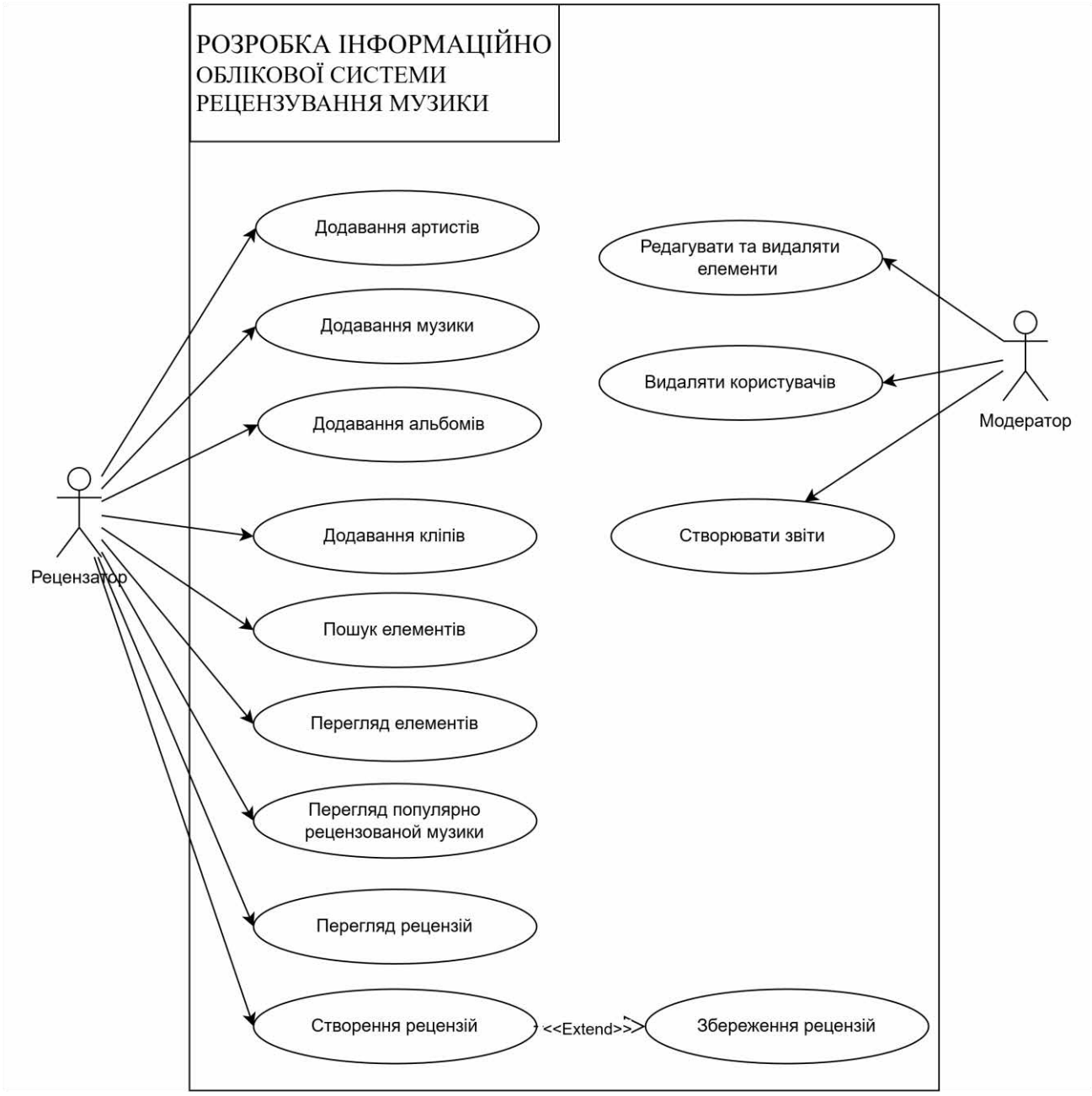


Рис.1 Діаграма послідовності (Use Case Diagram), що демонструє роботу рецензатора (користувача) та модератора (адміністратора), які взаємодіють з системою.

Діаграма діяльності (*Activity Diagram*) – це вид діаграми UML, що демонструє візуальне зображення послідовних дій або процесів що відбуваються в системі[28].

Головною метою діаграми діяльності є:

- Опис бізнес-процесів, алгоритмів чи внутрішніх процесів.
- Відображає яку логіку повинна роботи система.
- Моделює умови вибору, послідовні дій та паралельні процеси.

Елементи діаграми діяльності:

- Початковий стан (чорна коло) – початок процесу.
- Дія (прямокутник з округленням) – крок .
- Перехід (стрілка) – показує напрямок переходу між діями.
- Умова (ромб) – розгалуження за варіантами.
- Злиття (зафарбований маленький прямокутник) – об’єднання гілок після умовного розгалуження.
- Паралельні дії (зафарбований маленький прямокутник) – одночасне виконання кількох процесів.
- Кінцевий стан (чорне коло з обвідкою) – завершення процесу.

В моїй діаграмі діяльності Рис.2 я відображаю послідовність дій користувача де паралельними процесами є написання рецензій та зберігання або перегляд рецензій з можливістю почати знову паралельні процеси або закінчити дій. Діаграма показує наочний приклад роботи логічного процесу для швидкого виконання дій, а також це є важливим елементом для розробки чи аналізу функціональності системи.

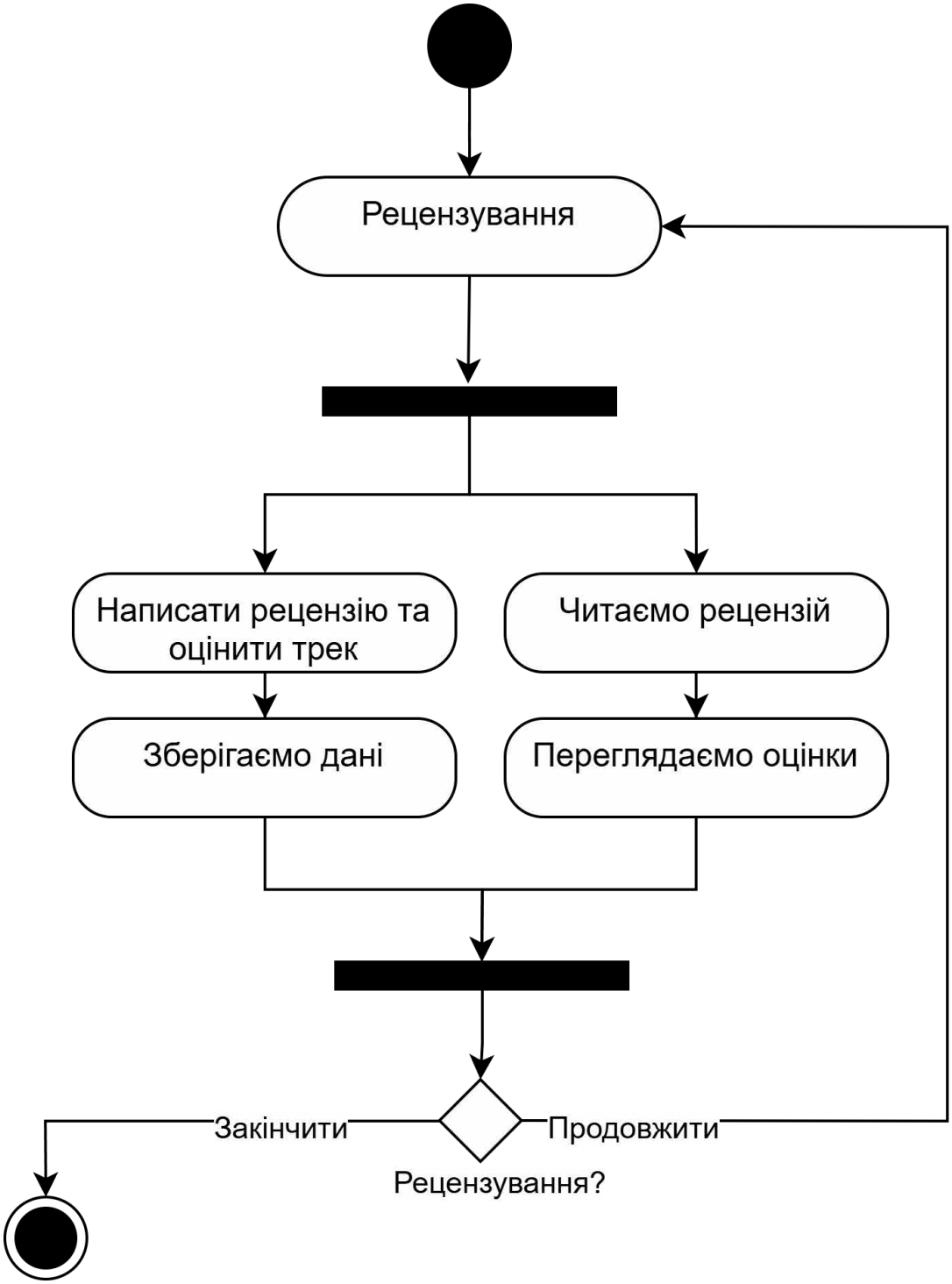


Рис.2 Діаграма діяльності процесу рецензування

1.2 Аналіз музичного ринку та потреб

Музична індустрія – це величезна складна та динамічна сфера тому щодня виходить багато нового контенту.

Кількість користувачів платних музичних стрімінгових сервісів табл.1 [5]:

Рік	Кількість користувачів	Приріст
2015	68 млн	-
2016	112 млн	+44 млн або 64,7%
2017	176 млн	+64 млн або 57,1%
2018	255 млн	+79 млн або 44,9%
2019	341 млн	+86 млн або 33,7%
2020	443 млн	+102 млн або 29,9%
2021	523 млн	+80 млн або 18,1%
2022	589 млн	+66 млн або 12,6%
2024	752 млн	+163 млн або 27,7%

Табл.1 Приріст користувачів до музичних платформ

Як ми можемо побачити на табл. 1, приріст користувачів стабільно збільшується що року. Це означає, що користувачам цікава музика, а також, що потреба в моїй системі зростає що року все більше.

Основними системи є такі категорії людей:

- Музичні критики для надання кваліфікованої оцінки, та написання професійних коментарів, або проведення детального аналізу різної категорій музичних елементів.
- Музичні блогери отримають сервіс для зручного коментування, читання, оцінки та взаємодій зі своєю аудиторією.
- Звичайні користувачі можуть викласти свою думку з оцінками, а також покращити розуміння музичної сфери, читаючи нові рецензії інших користувачів.
- Музикант отримає можливість прочитати та побачити оцінки, для розуміння, що подобається аудиторії, а що — ні, та покращення себе в майбутньому.

На сьогоднішній день існує багато сервісів для прослуховування та коментування, але вони не дають потрібну структуру для об’єктивного аналізу. Більшість з програм не мають нормальної системи критеріїв вони більше схожі на загальні підходи оцінювання такі як лайки чи звичайні коментарі.

Тому виникають такі проблеми:

- Оцінювання стає не об’єктивним;
- Інші компанії створюють схожий вид оцінювання;
- Для більш об’єктивного дослідження треба використовувати декілька сервісів.

Предметна область загалом охоплює процеси оцінювання, аналізу та взаємодії з музичним контентом, яка вимагає створенню спеціальної інформаційної системи що має зрозумілий інтерфейс, створенні критерії оцінювання та можливість гнучкої обробки даних.

Для системи рецензування та отінення було розроблене SWOT-аналіз табл.2 [32]:

Пункт	Приклад
S (Strengths) – Сильні сторони	Унікальна та деталізована система оцінок, гарний та зрозумілий інтерфейс (UI).
W (Weaknesses) – Слабкі сторони	Відсутність мобільної або веб-версії, підтримка однієї мови інтерфейсу.
O (Opportunities) – Можливості	Інтеграція в музичні платформи, розширення на інші типи контенту.
T (Threats) – Загрози	Аналоги та конкуренти, маніпуляція оцінками за допомогою ботів.

Табл.2 SWOT-аналіз

1.3 Пошук аналогів та конкурентів

У наш час існує не велика кількість систем для рецензування, особливо для рецензування музичного контенту.
Основні конкуренти моєї системи:

1. Metacritic рис. 3 [21]

- Плюсами системи можна назвати велику кількість можливих оцінок фільмів, треків, альбомів, серіалів, кліпів — майже всього, що завгодно.
- Але з мінусів можна виділити погану систему оцінювання, без деталізації до конкретного продукту, що, на мою думку, робить систему масштабною, але не високо ефективною.

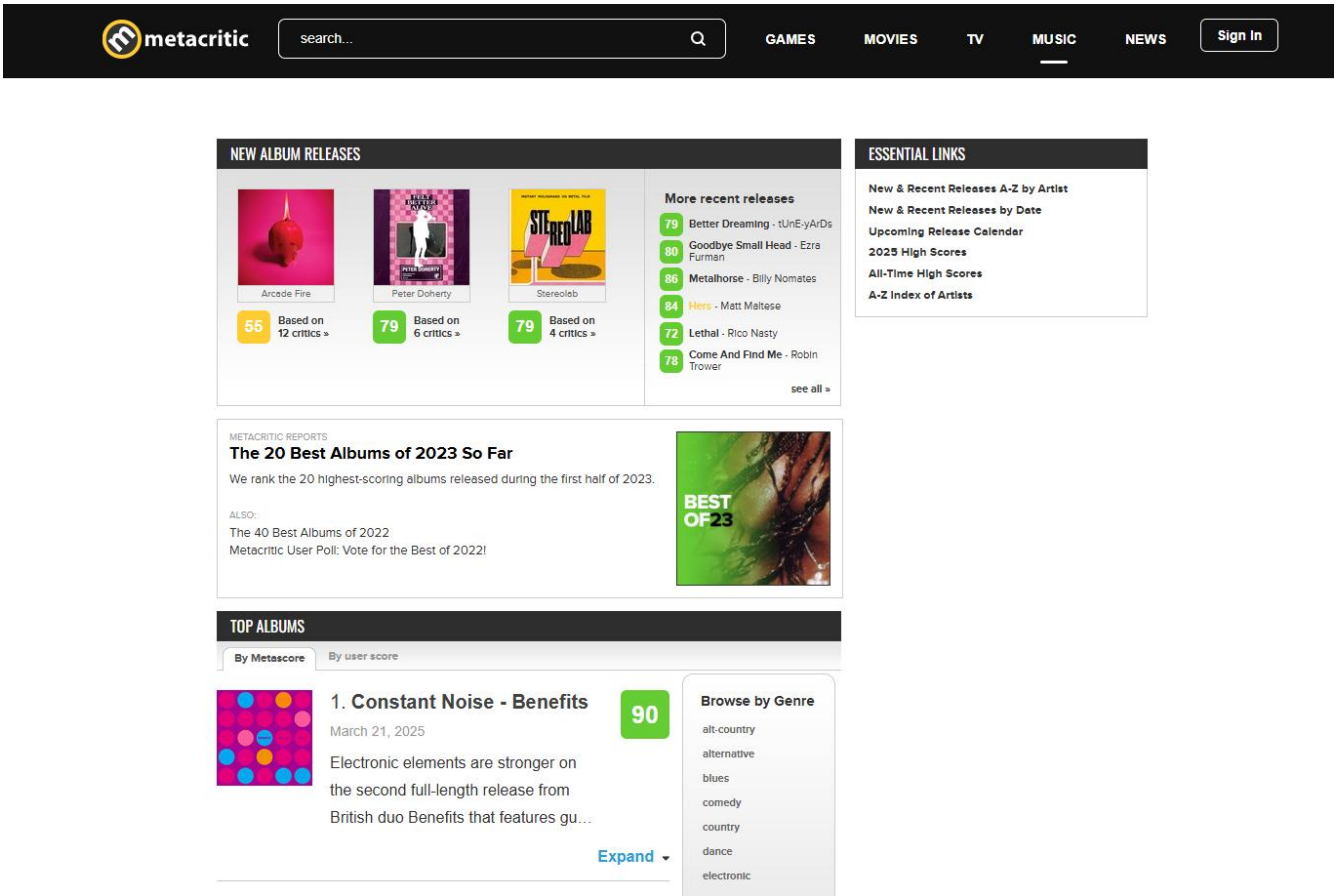


Рис. 3 Metacritic

2. Pitchfork рис. 4 [22]

- Плюсами веб-сервісу є аналізи критиків та власна 101-бальна система оцінок, що навіть може впливати на тренди.
- Але мінусами є великі проблеми з початком роботи звичайним користувачам.

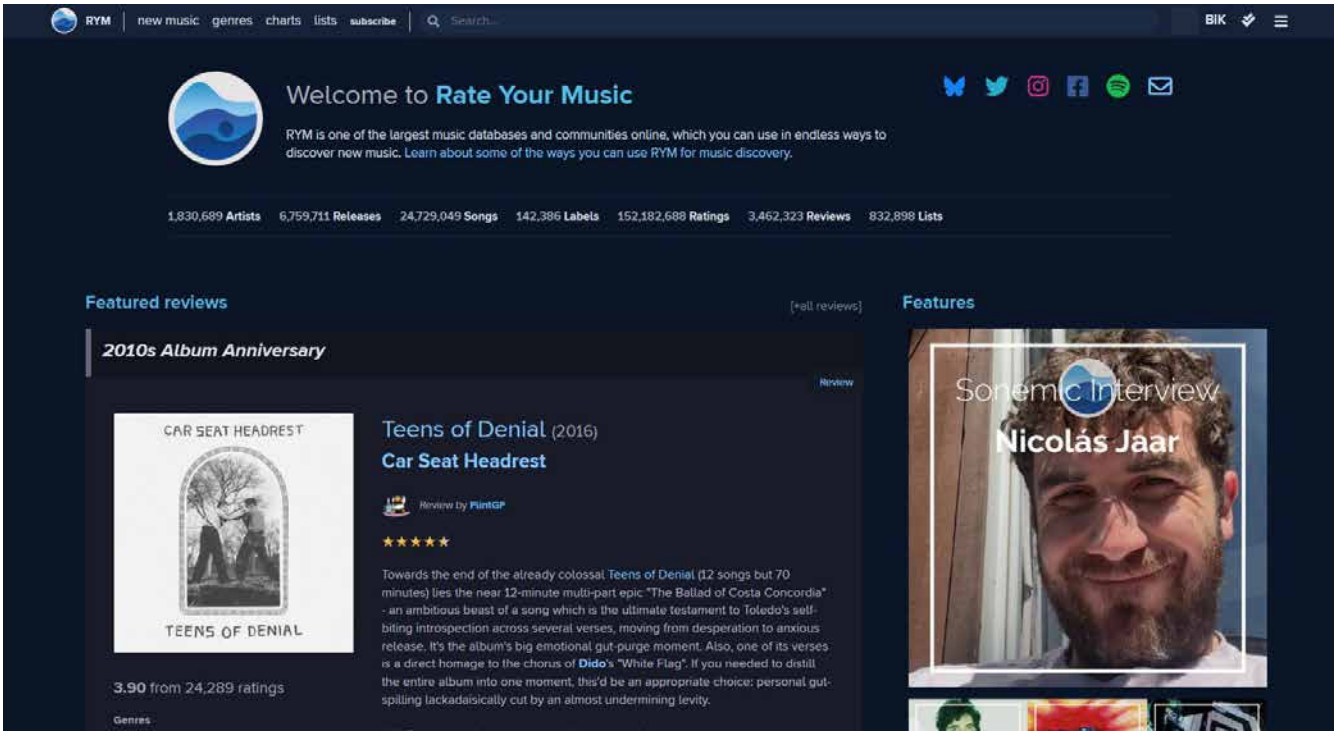


Рис. 4 Pitchfork

3. RateYourMusic рис. 5 [23]

- Плюсами цього сервісу є величезна спільнота та гарно розроблений каталог, що дає змогу швидко дістатися до потрібних музичних елементів.
- Великим мінусом є відсутність нормальних категорій для оцінки, що набагато зменшує розуміння, як правильно оцінити музичний контент.

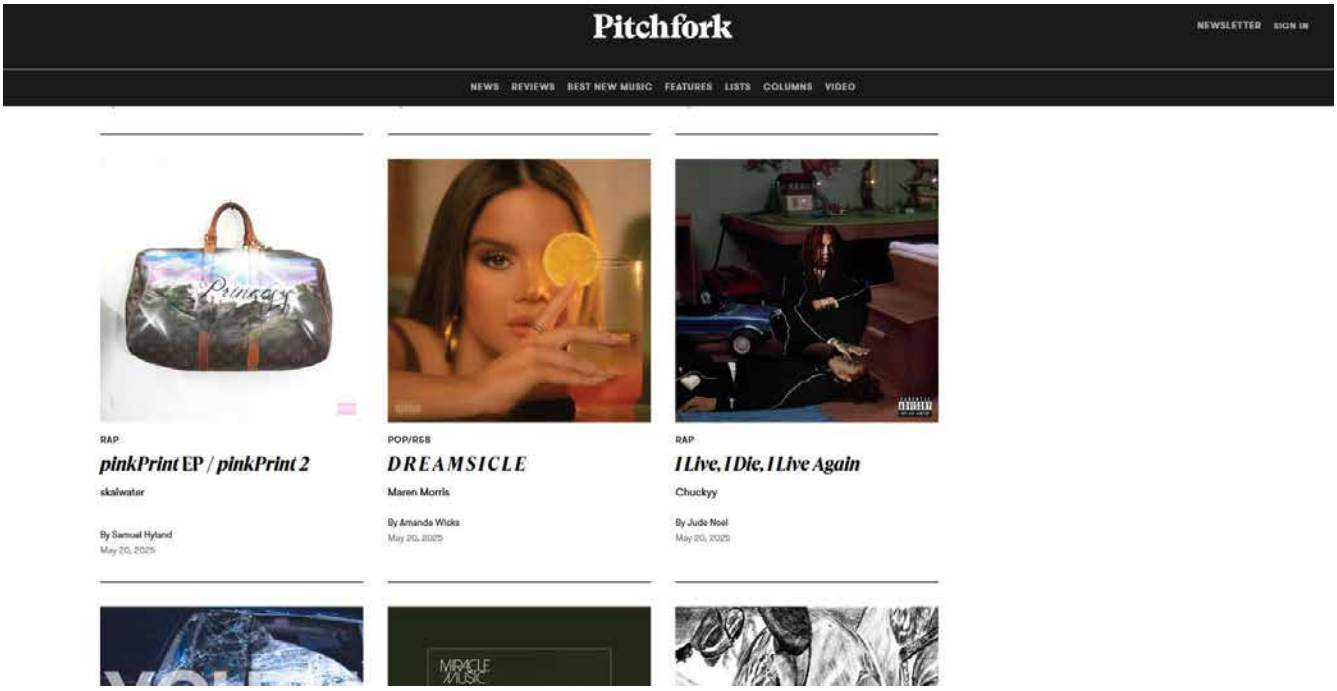


Рис. 5 RateYourMusic

Розділ 2. МОДЕЛЮВАННЯ ОБЛІКОВОЇ СИСТЕМИ РЕЦЕНЗУВАННЯ МУЗИЧНОГО КОНТЕНТУ

2.1 Створення логічної моделі

Для початку розробки системи треба створити ER-діаграму.

ER-діаграма (Entity-relationship model або entity-relationship diagram) – це модель даних, що допомагає описувати схеми певними концепціями за допомогою узагальнених блоків. Існує велика кількість моделей приставляння даних, але одним із найзручніших інструментів для уніфікованого представлення даних, незалежно від програмного забезпечення, яке його реалізує, є модель сутності-зв'язку. Важливим фактом є те, що всі існуючі моделі даних (ієрархічна, мережева, реляційна, об'єктна) можуть бути зроблені з моделі сутність-зв'язок, тому вона є найбільш загальною та вживаною.[12]

Коли ми говоримо про сутність, ми зазвичай говоримо про якийсь її аспект, який можна відрізнити від інших його аспектів у реальному світі. Сутність – це збірне поняття, яке є своєрідною абстракцією реальних предметів, процесів, явищ чи предметів. Хоча термін сутність використовується найчастіше, необхідно розрізняти поняття типу сутності та екземпляра сутності. Поняття типу сутності відноситься до однорідної групи осіб, об'єктів, подій або ідей, які діють як єдине ціле. Екземпляр сутності — це конкретна річ у колекції.

В моєму випадку я використовував розширену діаграму сутності-зв'язку, так звану (EER diagram або Enhanced Entity-Relationship diagram), яка є важливим компонентом інтерфейсу моделювання в MySQL Workbench.

Різницею ER та EER найбільше полягає в деталізації та силі опису предметної області. EER є так званою надбудовою, яка підтримує ієрархію зав'язків між сутностями, а також підходить для розробки більш складних баз даних.

На рис. 6 ми бачимо розроблену розширену діаграму сутності-зв'язку в третій нормальній формі. В системі ми маємо такі основні сутності: рецензент, артист, альбом, музика, кліпи та рецензії, також, для запоганю зав'язкам багато-до-

багатьох, було зроблені додаткові сутності, такі як артистальбом, чи артистkліп, чи артистпісня.

А зв’язки між сутностями зроблені так, щоб один рецензатор (користувач) міг створити багато рецензій, а кожна рецензія певного контенту (кліп, пісня, альбом).

А сам артист може бути пов'язаний з багатьма кліпами, піснями, альбомами, і також навпаки: один тип контенту може належить кільком артистам.

Саме така структура робить гнучке, добре відрегульовану базу даних, яка ефективно організовує інформацію про користувачів, артиста, його роботи та огляди, а також підтримує цілісність і послідовність даних у системі.

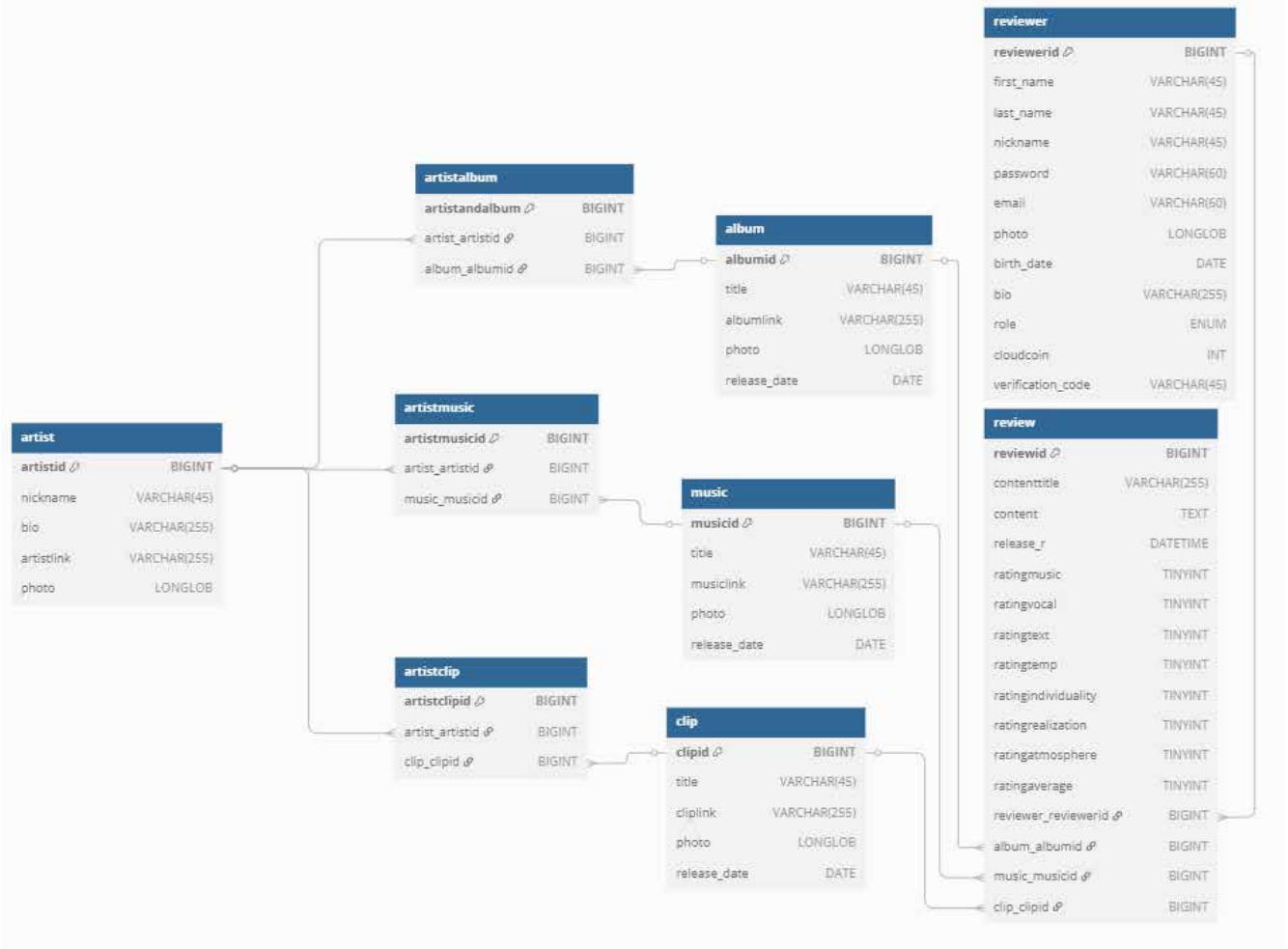


Рис.6 Діаграму сутності-зв’язку, яка показує основні сутності та зв’язки між ними, включаючи таблиці зв’язків.

2.2 Вибір система керування базами даних

Система керування базами даних (СКБД) — це набір програмних засобів, які використовуються для створення, зберігання, оновлення, отримання та керування базами даних.[13]

Перша система, схожими на системи керування базами даних, було зроблено ще в 1960 роках для автоматизації на виробництвах. В цей час з'являлися ієрархічні та мережеві бази даних, які були дуже структурованими системами з фіксованими зв'язками між елементами.

Але вже в 1970 роках відбулася велика зміна — було створено концепцію реляційної моделі, яка стала основою для сучасних СКБД. Ця модель дозволяла привести до одного стандарту, зробити його однотипним приставлянням даних за допомогою таблиць, що дуже полегшило розуміння та розробку для людей, які працюють з базами даних.

За допомогою цього підходу, в 1980 було створено першу повноцінну реляційну СКБД, такі як IBM System R та Ingres, але з часом до них підключилися й інші, такі як Oracle, MS SQL Server, PostgreSQL, MySQL та інші.

Типи систем керування базами даних за різними ознаками — в табл.3.

Типи СКБД	Опис
Ієрархічні	Дані організовані у вигляді дерева
Мережеві	Дані зберігаються в вигляді графа та можуть мати кілька зав'язків
Реляційні	Таблиці що мають фіксовану структуру
Об'єкти-орієнтовані	Об'єкти що маю в методи та властивості
Документо-орієнтовані	Збереження даних у вигляді JSON документів
Гібридні	Поєднує кілька різних моделей в одну

Табл.3 Типи систем керування базами даних

Приклад ієрархічні СКБД:

- IBM Information Management System (IMS) – це одна з перших систем керувань базами даних, яку використовували в банках, телекомунікаціях та інших.

Приклад використання:

- У банкові головною сутністю є клієнт, який має свою сутність рахунок в свою чергу він має свою сутність транзакція. Всі дані зберігаються в коректній фіксованій ієрархії.

Плюси системи:

- Швидкість роботи з даними;
- Підходи до структурованих та однакових даних.

Мінуси системи:

- Складність при потребі змінити щось в структурі;
- Не можливість створити зв’язок багато до багатьох.

Приклад мережевих СКБД:

- Integrated Data Store (IDS), IDMS (Integrated Database Management System) – використовуються для державних та корпораційних системах.

Приклад використання:

- У системах корпорації співробітники може буди прив’язаний до кількох проектів, а проекте одночасно до кількох відділів.

Плюси системи:

- Гнучкість зав’язків;
- Ефективність роботи при складних зв’язках.

Мінуси системи:

- Складна логіка запитів;
- Важка в підтримці та в маштабуванні.

Приклад реляційні СКБД:

- MySQL Workbench, PostgreSQL, Microsoft SQL Server, Oracle DB – є одними з най популярнішими видами баз даних що використовуються в веб розробці, фінансових системах, освітніх платформах, мобільних, десктопних додатках та інших

Приклад використання:

- Приклад використання в проекті є таблиці рецензент, музика, рецензії, всі рецензії пов’язані з певною музикою та рецензатором.

Плюси системи:

- Висока стабільність та надійність;
- Мова запитів SQL.

Мінуси системи:

- Ефективність роботи падає при неструктурованості чи великій кількості даних.

Приклад об’єктно-орієнтованої СКБД:

- db4o (Database for Objects), ObjectDB – використовуються як правило в вбудованих системах деяких десктопних додатків, а також в системах автоматизованого контролю.

Приклад використання:

- При створені складні десктопні додатки для яких треба мати велику кількість сутностей та внутрішніх об’єктів.

Плюси системи:

- Легке представлення складних структур;
- Висока швидкість роботи з об’єктами.

Мінуси системи:

- Важка інтеграція з реляційними системами;
- Слаба в підтримці стандартних інструментів.

Приклад документо-орієнтованої СКБД:

- MongoDB, CouchDB – використовуються для форумів, розподілених систем, інтернет речей, в іграх та для інших систем.

Приклад використання:

- При зберіганні інформації з великою кількістю змінних характеристик, що можуть змінюватися для кожного окремо.

Плюси системи:

- Гнучка структура даних;
- Висока швидкість при читанні.

Мінуси системи:

- Не ефективна робота при складних зв’язках між об’єктами;
- Відсутність схем.

Приклад гібридні СКБД:

- ArangoDB, OrientDB, Azure Cosmos DB – використовуються в веб-сервісах, систем безпеки, хмарних сервісах та в інших системах.

Приклад використання:

- При створені платформ де треба зберігати структуровані транзакції, напівструктуровані профілі та аналізу зав’язків з користувачами в одному середовищі.

Плюси системи:

- Висока гнучкість;
- Багато підходів.

Мінуси системи:

Складність у обслуговані та налаштуванні;

Не підходить для простих проектів.

А моя база даних повинна працювати з великою кількістю даних, підтримувати одночасно доступ для великої кількості користувачів, забезпечувати надійність

даних при зберіганні та легкій масштабованості, при підтримці реляційної моделі, мною було обрано MySQL, я основи СКБД, а саме MySQL Workbench – як засіб моделювати та проектування баз даних.

Основні переваги MySQL Workbench:

- Створення ER-діаграми;
- Автоматична генерація SQL-код;
- Підтримка різних стандартів SQL;
- Можливість розміщення як локального так і віддаленого серверу;
- Висока продуктивність;
- Безкоштовна та кросплатформленна для можливого створення телефонного додатку в майбутньому.

Побудована в сервісі MySQL Workbench діаграма даних рис.7.

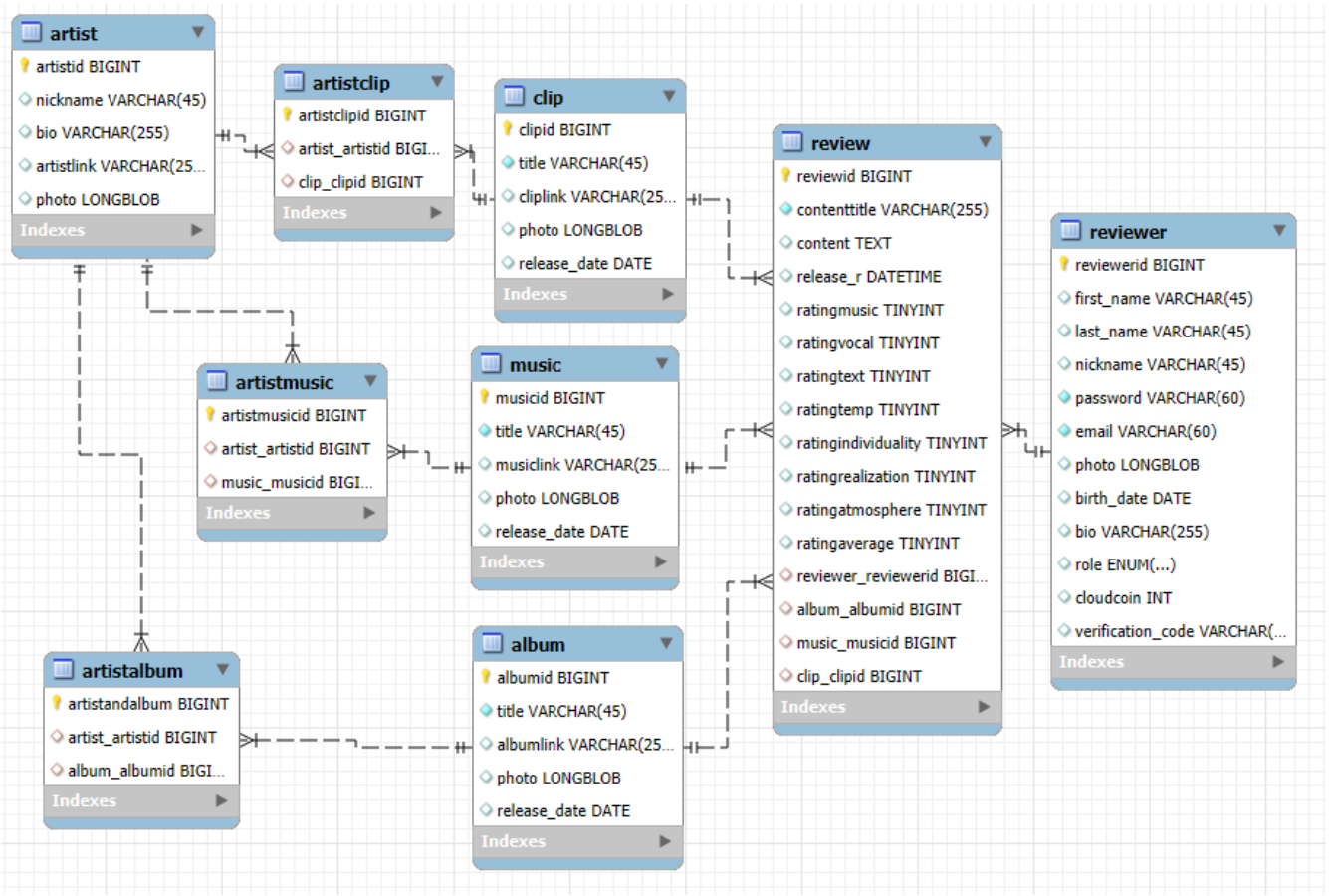


Рис.7 Діаграма даних

2.3 Створення бази даних

На основі створеної логічної моделі даних було згенеровано код для створено фізичну модель бази даних. Приклад створеної бази даних на прикладі створення користувача[8][19].

```
CREATE DATABASE CloudRate_BD;

USE CloudRate_BD;

CREATE TABLE Reviewer (
    reviewerid BIGINT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(45),
    last_name VARCHAR(45),
    nickname VARCHAR(45) UNIQUE,
    password VARCHAR(60) NOT NULL,
    email VARCHAR(60) UNIQUE NOT NULL,
    photo LONGBLOB,
    birth_date DATE,
    bio VARCHAR(255),
    role ENUM('Reviewer', 'Admin') DEFAULT 'Reviewer',
    cloudcoin INT DEFAULT 0,
    verification_code VARCHAR(6)
);
```

Приклад тестових даних для рецензантів.

```
INSERT INTO Reviewer (first_name, last_name, nickname, password, email, birth_date, bio)
VALUES
```

('Іван', 'Петров', 'ivan_r', 'pass123', 'ivan@example.com', '1995-06-15', 'Музичний критик.'),

('Ольга', 'Сидорова', 'olga_music', 'pass234', 'olga@example.com', '1998-01-20', 'Люблю електроніку.'),

('Анна', 'Коваленко', 'anna_k', 'pass345', 'anna@example.com', '2000-09-03', 'Аналізую тексти.'),

('Максим', 'Шевченко', 'max_s', 'pass456', 'max@example.com', '1992-12-12', 'Фан інді-року.'),

('Олексій', 'Бондар', 'aleksey', 'pass567', 'alex@example.com', '1993-07-25', 'Критик з досвідом.'),

('Тетяна', 'Мельник', 'tania_art', 'pass678', 'tania@example.com', '1997-11-11', 'Рецензую кліпи.'),

('Сергій', 'Іванов', 'serj_io', 'pass789', 'serg@example.com', '1994-05-05', 'Меломан.'),

('Юлія', 'Гнатюк', 'julie_h', 'pass890', 'julie@example.com', '2001-08-19', 'Люблю хіп-хоп.'),

('Дмитро', 'Мазур', 'dmazur', 'pass901', 'dmitro@example.com', '1999-04-22', 'Мій стиль — джаз.'),

('Катерина', 'Кравець', 'katya_k', 'pass012', 'katya@example.com', '1996-03-30', 'Поп музика — моє все.');

Приклад створеного запиту до бази даних рис.8, що виводить на екран інформацію про артиста, музику артиста, посилання на неї та день додавання контенту.

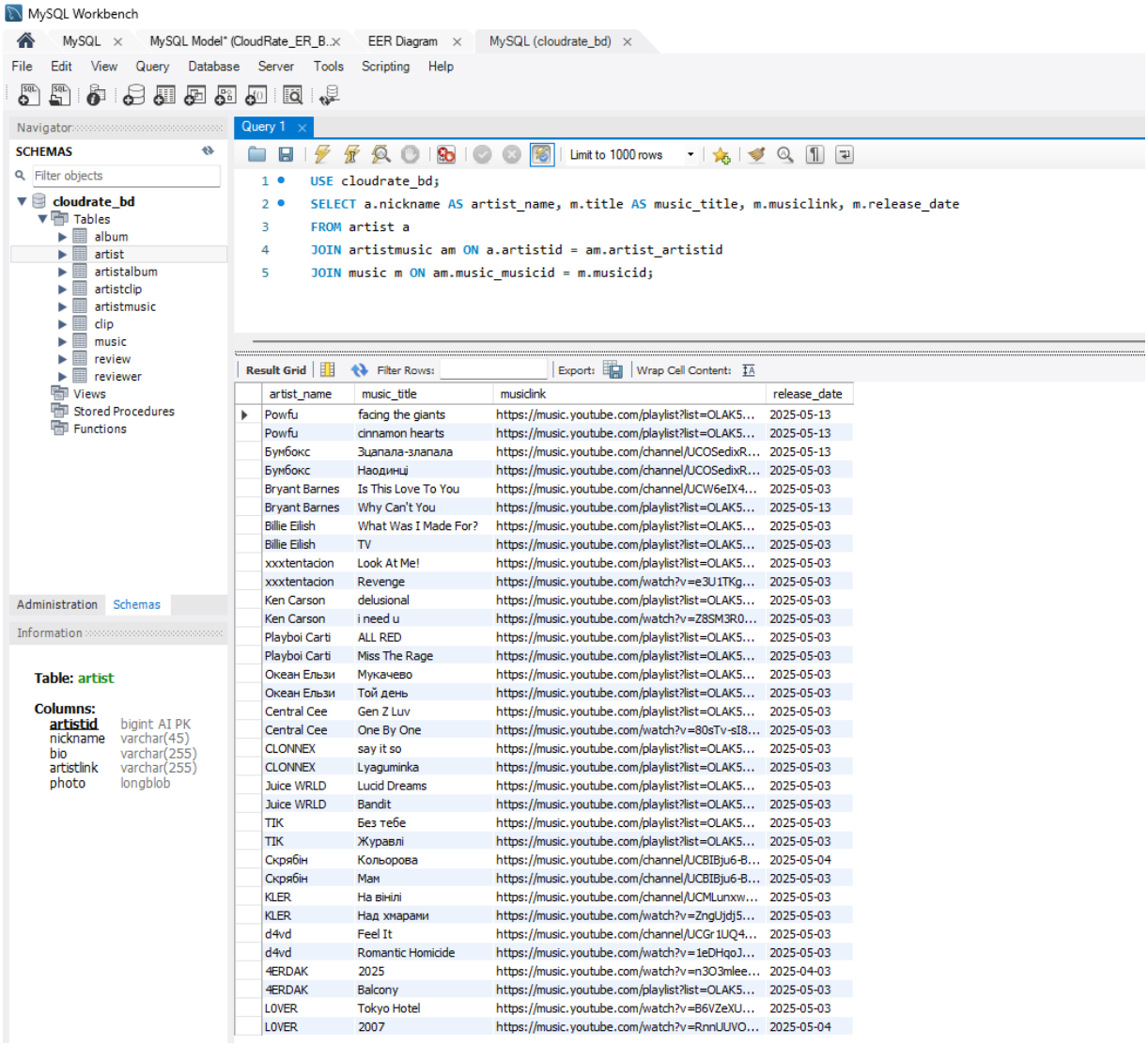


Рис.8 Запит до бази даних.

Розділ 3. СТВОРЕННЯ ПРОГРАМНОГО ПРОДУКТУ СИСТЕМИ РЕЦЕНЗАВАННЯ МУЗИКИ

3.1 Архітектура ПЗ

Архітектура програмного забезпечення — це те, що визначає структуру системи, її компоненти та взаємодію. Від правильного вибору архітектури залежить велика кількість процесів, таких як: працездатність, надійність, масштабованість, продуктивність і також зручність у повній розробці продукту[7][14].

Архітектура програмного забезпечення – це набір структур, необхідних для обґрунтування рішень, які враховують вимоги до системи[10].

Основними завданнями побудови архітектури є:

- організація компонентів програми;
- визначення способу взаємодії між модулями;
- забезпечення гнучкості та адаптивності;
- створення передумов для подальшого розвитку ПЗ.

Також архітектуру ПЗ розрізняють за такими стилями табл.4:

Стилі архітектури	Опис
Клієнт серверна	Розділяються на 2 частини клієнтська тобто інтерфейс та сервісну тобто обробка даних та логіку
Компонентна	Системна розділенна на незалежні модулі з інтерфейсом які можна використовувати багато разів
Предметно орієнтована	Система створюється навколо предметної області
Багатошарова або багаторівнева	В системі розділяються шари на 3 основі елементи інтерфейс логіка роботи та БД що забезпечує гнучкість та маштабованість
Шина повідомлень	Компоненти обмінюються повідомленнями через єдину передачу але не знають один про одного

Об’єктно-орієнтовані	Система працює за допомогою об’єктів що інкапсулють дані та методи
Сервісно-орієнтовані	Побудова ПЗ як взаємодія незалежних сервісів за допомогою стандартизованими інтерфесами.

Табл.4 Стилі архітектур ПЗ

Для своєї системи я обрав багаторівневу архітектуру програмного забезпечення з елементами клієнт-серверної моделі та використовував стиль MVVM. За дорогою такого вибору мені буде легко реалізувати всі функціональні вимоги.

Багаторівнева архітектура робить чітке розділення системи на різні компоненти, що забезпечує для кожного рівня свої завдання, що сильно полегшує розробку та підтримку системи в подальшому.

Вибраний патерн MVVM дозволяє зрозумілим чином розділяти інтерфейс користувача, не мішаючи його разом з логікою, а також, що дуже важливо, він розділяє логіку та представлення даних, що робить для програміста більш зрозумілим процес розробки.[20]

Також за допомогою багаторівневої архітектури система легко масштабується окремими елементами, що означає можливість масштабувати окремі елементи, перевіряючи роботу системи з новими навантаженнями.

Модульна система дозволяє легко додавати та змінювати функції для потрібних елементів без змін інших частин.

Також великим плюсом такої системи є безпека, що реалізована за допомогою розділення доступу та ізоляції даних.

Моя система складається з основних 3 модулів:

1. Презентаційний модуль (View)
    - Користувацький інтерфейс
    - Візуальні компоненти
    - Елементи керування
    - Стилі для приставлення
2. Модуль бізнес-логіки (ViewModel)
    - Обробка даних

- Перевірка вхідних даних
- Реалізація правил та алгоритмів
- Комунікація з модулем даних

3. Модуль даних (Model)
    - Взаємодія з базою даних
    - Зберігання та отримання даних
    - Обробка запитів до бази даних
    - Забезпечення коректності даних

Взаємодія відбувається в модулях за чітким алгоритмом:

1. Користувач працює з презентаційний модуль (View) за допомого користувацького інтерфейсу.
2. Презентаційний модуль (View) відправляє дані до модуль бізнес-логіки (ViewModel).
3. Модуль бізнес-логіки (ViewModel) обробляє передану інформацію та приймає рішення та формує запит до модуль даних (Model).
4. Модуль даних (Model) робить запит до бази даних з необхідними потребами.
5. Модуль даних (Model) отримує назад результат та відправляє її до модуль бізнес-логіки (ViewModel).
6. Модуль бізнес-логіки (ViewModel) отримує обробляє та підготовлені дані передає до презентаційний модуль (View).
7. Презентаційний модуль (View) виводить дані користувачеві на екран.

3.2 Вибір інструментів для створення додатку

Мова програмування (Programming language) – це несправжня мова, яка створена для передачі команд. Мова програмування визначається набором лексичних, синтаксичних і семантичних правил, що повністю надає вигляд для коду та дії. З створенням першої мови програмування вже пройшло багато часу, за цей час було винайдено, що найменше 2000 мов програмування, та вони що року збільшуються в своїй кількості. Деякі мови програмування можуть бути використані самостійно, а для деяких треба ще додаткові мови програмування.[15]
Також мови програмування розділяються на два основні види, такі як:

1. Мова низького рівня
    - Це такий вид мов, що дуже схожі або є машинним кодом. Вони працюють і мають повний доступ до апаратних ресурсів. До таких мов можна віднести

бінарний код (машина мова) або Асемблер. Плюсами можна назвати швидкість роботи коду, не треба використовувати інтерпретатор чи проміжний код, також повний контроль над залізом; програма працює в середині, тому може повністю керувати пам’яттю та іншими внутрішніми елементами. Мінусами ж мови низького рівня програмування є очевидно складним написання самої програми, повільна розробка та залежність від апаратної архітектури. Як правило, такі мови використовують для написання драйверів, біосів та систем з потребою в швидкості та контрольованості.

2. Мова високого рівня
   - Це такі мови, які орієнтовані на роботу людей, вони мають схожий вигляд мов, які використовують люди. Для його роботи використовують спеціальні програми, так звані компілятори чи інтерпретатори, які перетворюють написаний людиною код в код, який розуміє машина; тільки після цього команда відбувається. До мов високого рівня відносять Python, Java та багато інших. Більшість мов програмування є мовами високого рівня. З переваг мов високого рівня можна назвати зрозумілість коду, автоматичну роботу з пам’яттю, незалежність від платформ, простоту в підтримці та масштабованість коду. Мінусами ж є менший контроль над апаратними ресурсами, нижча продуктивність, залежність від середовища. Такі мови використовують майже скрізь — програми, сайти, ігри, обчислення та багато іншого.

WPF (Windows Presentation Foundation) – це технологія, яку створила компанія Microsoft ще у 2006 році разом із виходом .NET Framework 3.0 для створення графічного дизайну інтерфейсів користувачів комп’ютерних додатків на Windows. WPF повністю базується на мовах XAML для дизайну інтерфейсу та C# як основній мові програмування. Це дозволяло вже в той час створювати неймовірні інтерфейси з використанням анімації, 2D чи 3D графіки, створюючи стилі, шаблони та прив’язки даних до них. Завдяки архітектурі Model-View-ViewModel все чітко було розділено на 3 модулі. Основними перевагами я назвав би гнучкість, потужність, підтримку анімації, легкий та зручний редактор для створення інтерфейсу, але мінусом є те, що працює тільки на платформі Windows[9][18][24].

Мова програмування XAML (eXtensible Application Markup Language) – це декларативна мова розмітки, створена в 2006 році компанією Microsoft для .NET Framework 3.0 та Windows Presentation Foundation (WPF). З точки зору програмування в .NET Framework, це мова, яка полегшує розробку користувацьких інтерфейсів для додатків. Мова дає можливість легко створити користувацький інтерфейс, який буде відділений від логіки програми[17].

C# – це сучасна об'єктно-орієнтована мова програмування, створена компанією Microsoft у 2000 році як частина проєкту .NET Framework. Мова постійно оновлюється і покращується, що робить її однією з найпопулярніших мов у світі. C# має синтаксис, схожий на інші мови програмування, такі як C++ та Java. Мова повністю підтримує концепції інкапсуляції, наслідування, поліморфізму, що робить її зручною для створення масштабованих і підтримуваних програмних рішень. Однією з найважливіших особливостей C# є автоматична робота з пам'яттю за допомогою garbage collector (збирача сміття), що значно покращує та полегшує роботу програмістів і суттєво зменшує ймовірність помилок, які виникають через неправильне виділення або видалення ресурсів[2][16][31].

Завдяки тісній інтеграції .NET з C#, за допомогою цієї мови можна створювати дуже різноманітні типи застосунків — від звичайних програм для комп'ютера до веб-сайтів і навіть ігор. Якщо брати C# як частину WPF, то його використовують для реалізації бізнес-логіки, обробки подій, управління даними та зв'язування з інтерфейсом користувача, який був розроблений на мові XAML[1][25].

Для реалізації свого проекту було обрано саме таке програмне забезпечення що поєднує WPF (Windows Presentation Foundation), основну мову програмування C# та для розробки інтерфейсу користувача мову розмітки XAML[30].

Найважливішими причинами вибору такого інструментарію були:

1. Створіння гарних інтересів

- WPF є неймовірно потужною платформою для створення гарних та гнучких інтерфейсів. Для мого додатку було дуже важливо створити гарний дизайн що буде допомагати користувачам зрозуміло використовувати заложенні функціональні потреби через дизайн.

2. Ефективність та зрозумілість
   - C# є дуже зрозумілою мовою програмування з зрозумілим та легким синтаксисом що ефективно працює в середині додатку.
3. Інтеграція з операційною системою Windows.
   - Оскільки мій додаток був розрахований на десктопну частину користувачів то платформа що є най більш популярною в світі є дуже гарним вибором.
4. Архітектурне рішення MVVM
   - WPF підтримує використання шаблон MVVM (Model-View-ViewModel) який дає зручність для створення додатків з можливістю розділяти все на 3 основні частини які не заважають роботі інших а тільки покращують.

Діаграма пакетів – це тип діаграм UML, що демонструють роботу системи на високому рівні, що показує як виглядає система розбита на окремі логічні модулі та яка вони зв’язані[29].

Пакет – це логічна група об’єднаних між собою модулів, інтерфейсів чи інших елементів що відносяться до програмного забезпечення.

Такі проблеми вирішує діаграма пакетів:

- Демонстрація архітектури системи.
- Представляє залежності між різними частинами проекту.
- Покращує планування розробки та допомагає розділити роботу між командами.
- Допомагає в документуванні структуру системи.

На моїй діаграмі рис.9 демонструється архітектура додатку, що складається з 2 основних пакетами. Desktop App (клієнтська частина) та Server (серверна частина). Кожен пакет містить свої компоненти, що взаємодіють між собою та з іншими пакетами.

1. Desktop App (Клієнтська частина):

- UI (User Interface) — інтерфейс користувача,
  через який відбувається взаємодія з програмою.
- DomainLogic — бізнес-логіка клієнта, обробка даних та команд
  від користувача.
- FileHandler — робота з локальними файлами.
- NetworkManager — мережевий обмін даними із сервером.

2. Server (Серверна частина):

- Handler — приймає та обробляє запити від клієнта.
- Domain — бізнес-логіка сервера.
- Repository — доступ до бази даних, збереження та отримання інформації.
- Model — опис структури даних.
- Database — база даних, у якій зберігається інформація.

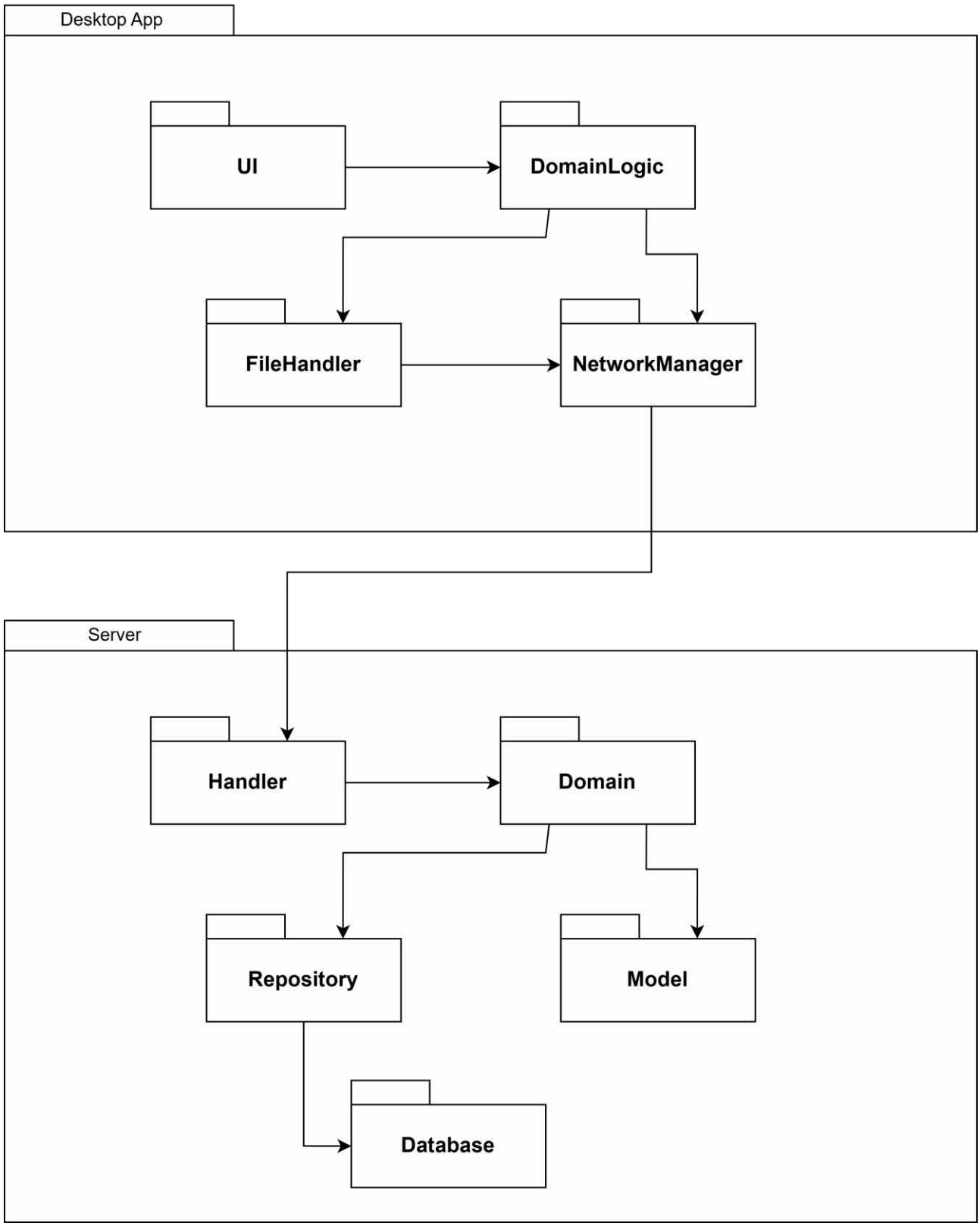


Рис.9 Діаграма пакетів

3.3 Алгоритмізація та програмування

На рис. 11 ми можемо переглянути блок-схему, що чітко демонструє процес повної роботи алгоритму дій при авторизації рис.12 рецензента (користувача) чи модератора (адміністратора).

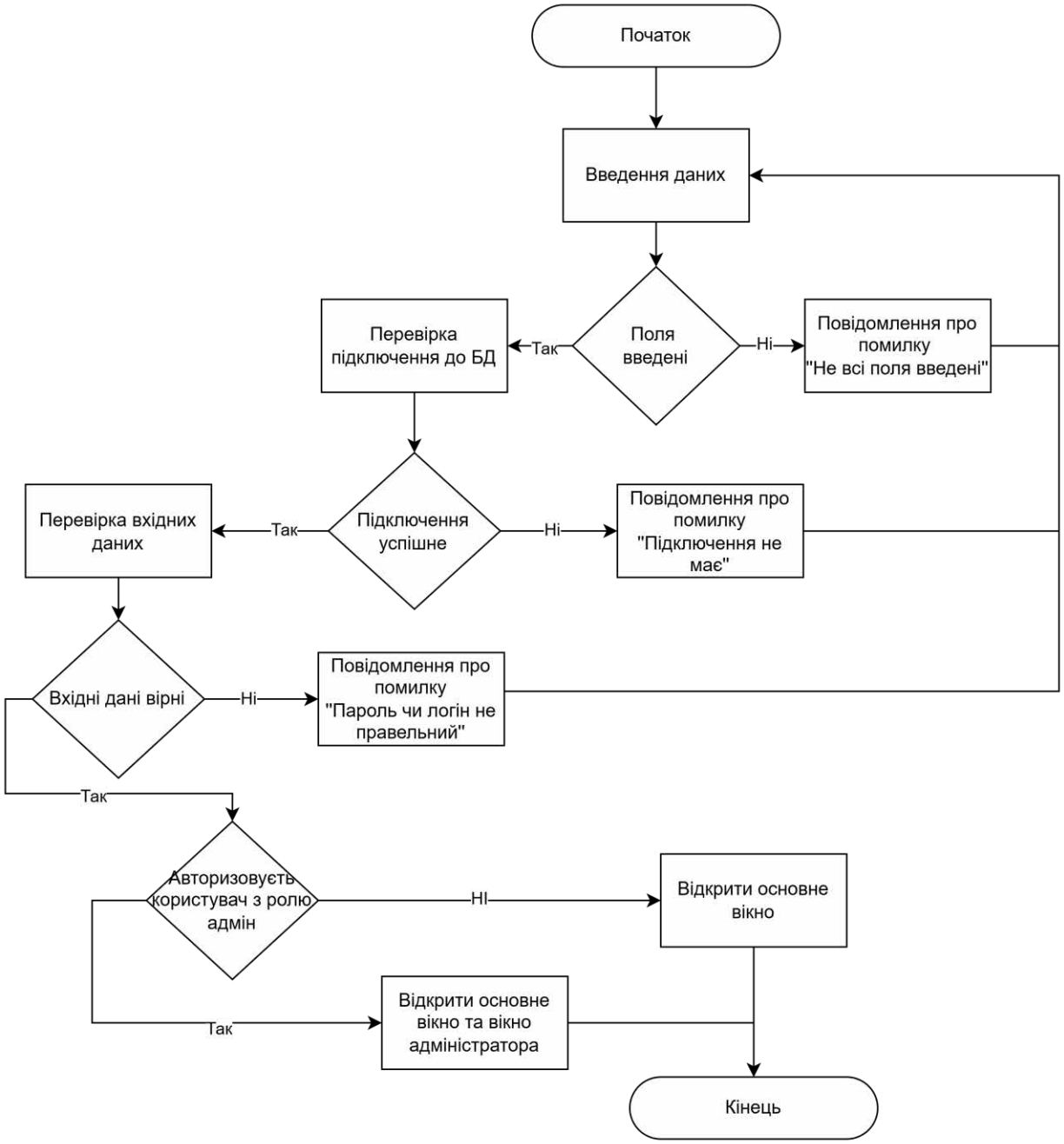


Рис.11 Блок-схему авторизації

Процес авторизації в коді демонструється. Метод LoginButton_Click реалізує авторизацію користувача, отримуючи введені дані користувача і перевіряючи, чи є

вони в базі даних. Якщо таких даних немає, виникає помилка. Якщо ж дані правильні, то за допомогою спеціального _autoSelect отримуємо інформацію про роль і ім’я користувача та перевіряємо, чи є користувач модератором. Якщо так, відкривається два меню — одне з яких є меню модератора, а якщо ні — відкривається тільки основне меню користувача рис.15 [3][26].

```
private void LoginButton_Click(object sender, RoutedEventArgs e)
{
    string login = EmailTextBox.Text;
    string password = PasswordBox.Password;

    if (string.IsNullOrWhiteSpace(login) || string.IsNullOrWhiteSpace(password))
    {
        MessageBox.Show("Будь ласка, введіть логін (email або нікнейм) та пароль", "Помилка", MessageBoxButton.OK, MessageBoxImage.Warning);
        return;
    }

    var userData = _autoSelect.AuthenticateUser(login, password);
    if (userData != null)
    {
        string role = userData["role"].ToString();
        string nickname = userData["nickname"].ToString();

        try
        {
```

```
            if (role == "Admin")
            {
                var menuWindow = new MenuWindow(nickname);
                var adminWindow = new AdminMenuWindow();
                menuWindow.Show();
                adminWindow.Show();
            }
            else
            {
                var menuWindow = new MenuWindow(nickname);
                menuWindow.Show();
            }
            Close();
            return;
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Помилка при відкритті вікна: {ex.Message}",
"Помилка", MessageBoxButton.OK, MessageBoxImage.Error);
            return;
        }
    }
```

MessageBox.Show("Невірний логін (email або нікнейм) або пароль", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

}

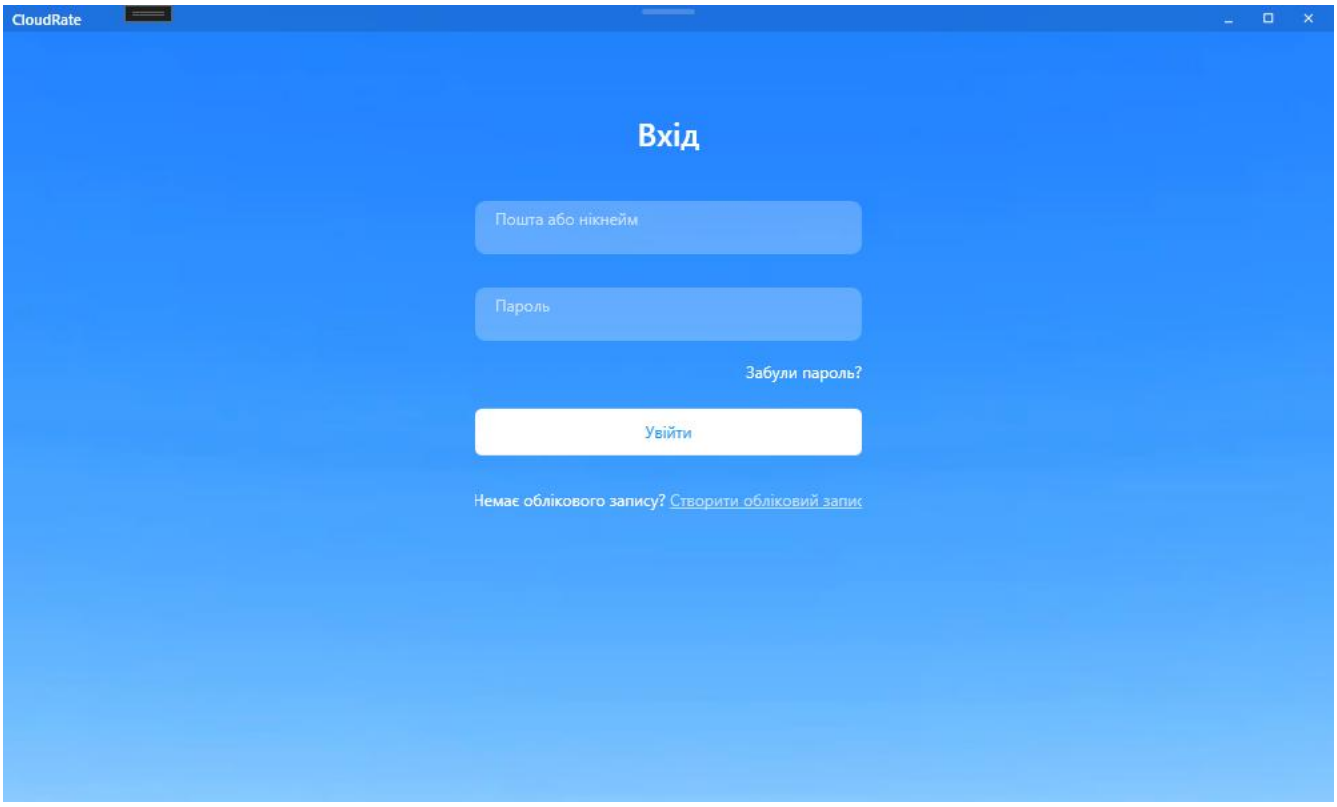


Рис.12 Вигляд вікна авторизації

Приклад використання патерну MVVM (Model-View-ViewModel) в роботі додатку[4]:

View модуль користувацького інтерфейсу на прикладі основної вкладки MenuWindow.xaml а саме компонент Track1Border. View модуль передає інформацію для отримання даних про цей трек до ViewModel.

Стиль Track1Border

<Border x:Name="Track1Border"

Background="#40FFFFFF"

CornerRadius="10"

Margin="0,0,5,0"

```
        MouseLeftButtonDown="Card_Click">
    <Grid>
      <Image x:Name="Track1Image"
          Margin="10"
          Stretch="UniformToFill"/>
      <Grid VerticalAlignment="Bottom"
          Margin="10">
        <StackPanel Margin="10,5">
          <TextBlock x:Name="Track1Title"
                Foreground="White"
                FontSize="14"
                TextWrapping="Wrap"/>

          <TextBlock x:Name="Track1Artist"
                Foreground="White"
                FontSize="12"
                Margin="0,2,0,0"/>
        </StackPanel>
      </Grid>
    </Grid>
  </Border>
```

ViewModel викликає потрібний метод для отримання даних в Model.

Запит для отримання даних

```
private void UpdateWeeklyContent()
{
    try
    {
        _weeklyTracks = _autoSelect.GetRandomTracksFromLastWeek(2);
        _weeklyAlbums = _autoSelect.GetRandomAlbumsFromLastWeek(2);
        _weeklyClips = _autoSelect.GetRandomClipsFromLastWeek(1);
        _lastUpdateDate = DateTime.Now;

        DisplayWeeklyContent();
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Помилка при оновленні контенту тижня: {ex.Message}");
    }
}
```

Model виконує SQL-запит до бази даних та отримує потрібну інформацію про трек та записує цю інформацію.

Запит до бази даних

```
public List<TrackInfo> GetRandomTracksFromLastWeek(int count)
{
    var tracks = new List<TrackInfo>();
```

```
string query = @"
    SELECT
        m.musicid,
        m.title,
        m.musiclink,
        m.photo,
        m.release_date as ReleaseDate,
        a.nickname as ArtistName
    FROM Music m
    INNER JOIN ArtistMusic am ON m.musicid = am.music_musicid
    INNER JOIN Artist a ON am.artist_artistid = a.artistid
    WHERE m.release_date >= DATE_SUB(CURDATE(), INTERVAL 7 DAY)
      AND m.release_date <= CURDATE()
    ORDER BY RAND()
    LIMIT @count";

using (var connection = new MySqlConnection(_connectionString))
{
    try
    {
        connection.Open();
```

```
                tracks = connection.Query<TrackInfo>(query, new { count }).ToList();

                Console.WriteLine($"GetRandomTracksFromLastWeek: Found {tracks.Count} tracks from last week (from {DateTime.Now.AddDays(-7):yyyy-MM-dd} to {DateTime.Now:yyyy-MM-dd})");

            }

            catch (Exception ex)

            {

                MessageBox.Show($"Помилка при отриманні треків: {ex.Message}");

            }

        }

        return tracks;

    }
```

Запис інформації

```
  public class TrackInfo

  {

      public long MusicId { get; set; }

      public string Title { get; set; }

      public string MusicLink { get; set; }

      public byte[] Photo { get; set; }

      public DateTime ReleaseDate { get; set; }

      public string ArtistName { get; set; }
```

```
}
```

ViewModel отримує інформацію від Model в списоках та передає цю інформацію для використання View.

Списки даних пісень, альбомів та кліпів.

```
private static List<TrackInfo> _weeklyTracks;


private static List<AlbumInfo> _weeklyAlbums;


private static List<ClipInfo> _weeklyClips;
```

Вивід на екран інформації про трек

```
private void DisplayWeeklyContent()
{
   try
   {
      if (_weeklyTracks != null && _weeklyTracks.Count >= 2)
      {
         var track1 = _weeklyTracks[0];
         Track1Title.Text = track1.Title;
         Track1Artist.Text = track1.ArtistName;
         if (track1.Photo != null)
         {
            var image = Track1Border.FindName("Track1Image") as Image;
            if (image != null)
            {
```

```
                    var bitmap = LoadImage(track1.Photo);
                    if (bitmap != null)
                        image.Source = bitmap;
                }
            }
        }
    }
}
```

View отримує інформацію та виводить цю інформацію про трек на екран Рис.13.

Тестування системи – це процес, що забезпечує програмісту розуміння, чи його програма працює вірно.[27]

Є 4 основних види тестування додатків такі як:

- функціональне тестування (Functional testing);
- нефункціональне тестування (Non-functional testing);
- структурне тестування (Structural testing);
- тестування змін (Change related testing).

Для свого проекту я робив структурне тестування, а сами UnitTest.

UnitTest – це розробка спеціальної програми так званих тестів що автоматично перевіряю, певну частину коду так званий юніт на певного очікуванням.

Ключовою причиною вибору UnitTest стало:

1. Швидкість запуску та знаходження помилок;
2. Легке тестування в ізоляції;
3. Автоматизації перевірок;
4. Прискорення розробці.

UnitTest був пройдений для функції авторизації та пошуку елементів рис.9. Приклад створеного тесту.

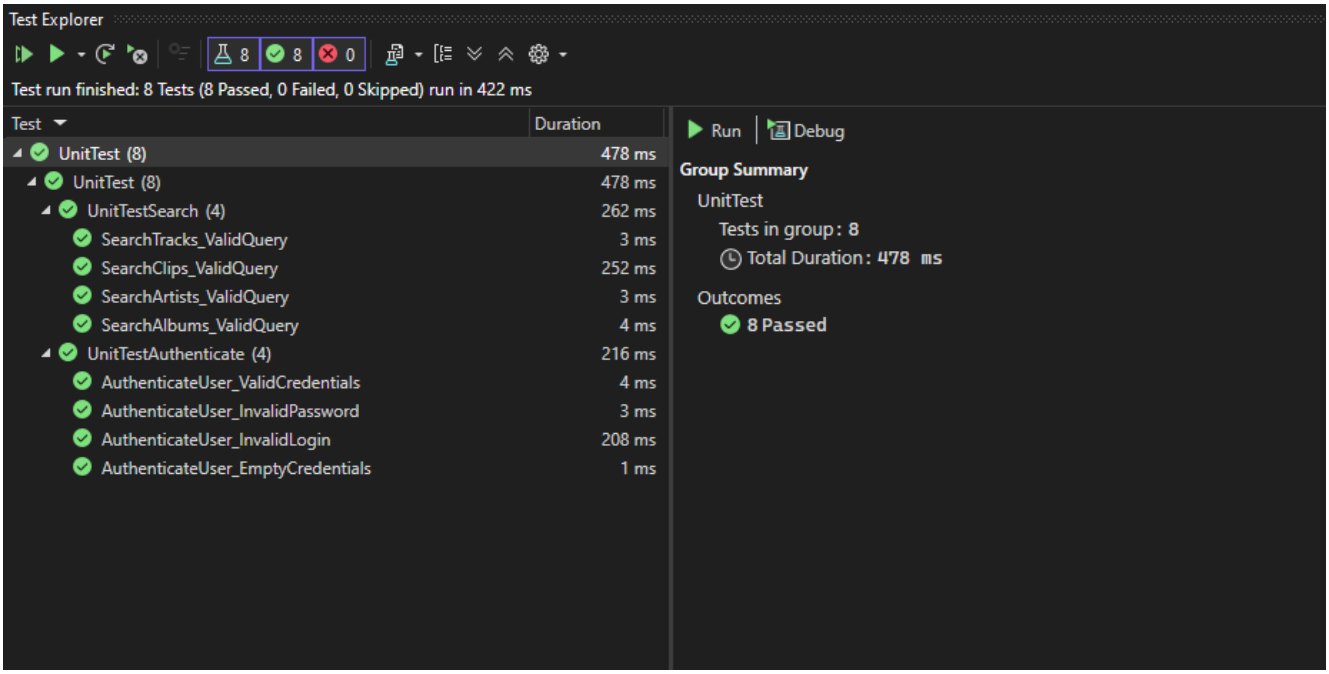


Рис. 13 Пройдені UnitTest

UnitTest пошуку альбому

```
[Fact]
public void SearchAlbums_ValidQuery()
{
    string searchQuery = "Яка є";

    var results = _autoSelect.SearchAlbums(searchQuery);

    Assert.NotNull(results);
    Assert.NotEmpty(results);
    Assert.Contains(results, album => album.Title.Contains(searchQuery,
StringComparison.OrdinalIgnoreCase));
}
```

UnitTest перевірки авторизації

```
[Fact]
public void AuthenticateUser_InvalidLogin()
{
    string login = "nonexistent@example.com";
    string password = "Bohdan";

    var result = _autoSelect.AuthenticateUser(login, password);
    Assert.Null(result);
}
```

Розділ 4. ТЕСТУВАННЯ ТА ІНТЕГРАЦІЯ ІНФОРМАЦІЇНОЇ СИСТЕМИ

4.1 Тестування інтерфейсу

Мій додаток має дві ролі рецензатор(Користувач) та модератора(Адміністратор).

Основне меню рис.15

В основному меню є 3 види виведеного контенту кліпи тижня музика тижня та альбоми тижня. Є верне меню де є основні кнопки для переходу до різних сторінок:

1. Пошук

Для пошуку використовується 2 типи такі як:

1. Ручний рис.16
    - Користувач попадає до вікна та самостійно шукає контент за допомогою додаткових елементі для вибору виду контенту.
2. Системний рис.17
    - Користувач вписує назву контенту і в системі за допомогою оператора LIKE для кожного елементу починається пошук результат пошуку групується і виводиться всі знайдені варіанти.

2. Додати рис.18

При натисканні на кнопку додати відкривається нове вікно з можливістю обрати бажаний контент для додавання. Після вибору контенту в користувача відкривається нове вікно рис.19 для якого треба заповнити всі елементи та натиснути на зберегти.

3. Популярне рис.20

В системі також є можливість переглянути найбільш популярні музичні елементи за оцінками та рецензіями, але якщо оцінки однакові пошук елемента місця працює по кількості написаних рецензії на його музичний елемент, якщо кількість написаних рецензій та бал спів падуть то твори будуть мати однакове місце в вибірці. В вибірці приймають учать всі музині елементи пісні, альбоми та кліпи, а час елементів в списку тільки 1 місяць після випуску.

4. Мій акаунт

При переході до цього елементу користувач бачить дані його акаунту та може змінити фото та інші параметри.

Основною функцією програми є рецензування та оцінка музичних елементі рис.21 де ми можемо побачити ім’я рецензованого твору та нікнейм автора також елементи для воду оцінок від 1 до 10 які будуть обраховані за формулою рис.14 та полей заголовків рецензії та тексту рецензії. Після вводу всіх потрібних полів та збереження нас переводить до сторінки з можливість прочитати рецензії рис.22 на без можливості написати ще раз рецензію для вже написаного рецензії для музичного елементу.

$$(\text{музика} + \text{вокал} + \text{текст} + \text{темп} + \text{індивідуальність} + \text{реалізація} + \text{атмосфера} + 30) / 10$$

$$(8 + 7 + 6 + 7 + 8 + 7 + 6 + 30) / 10 = (79) / 10 = 7.9$$

Рис.14 Формула обрахування оцінок

Далі буде продемонстрований інтерфейс адміністратора. На верхньому меню якого є 2 кнопки для переходу:

1. Модерація рис.23
   Основна панель для зміни та видалення контенту, рецензій чи рецензаторів.
2. Звіт рис.24
   Модератори можуть переглянути чи завантажити звіт на комп’ютер рис.25.
   Критерії звітів:

1. Популярний контент
   - Назва елементу, Тип контенту, Кількість відгуків, Середній рейтинг.
2. Активність користувачів
   - Нік рецензента, Кількість рецензій, Остання рецензія, Середній бал.
3. Статистика відгуків
   - Тип контенту, Загальна кількість рецензій, Оцінки.

4. Ріст контенту
    - Місяць, Рецензій на музику, Рецензій на альбоми, Рецензій на кліпи, Всього рецензій.
5. Рейтинг контенту
    - Назва елементу, Тип контенту, Кількість відгуків, Середні бали по кожному з категорій.

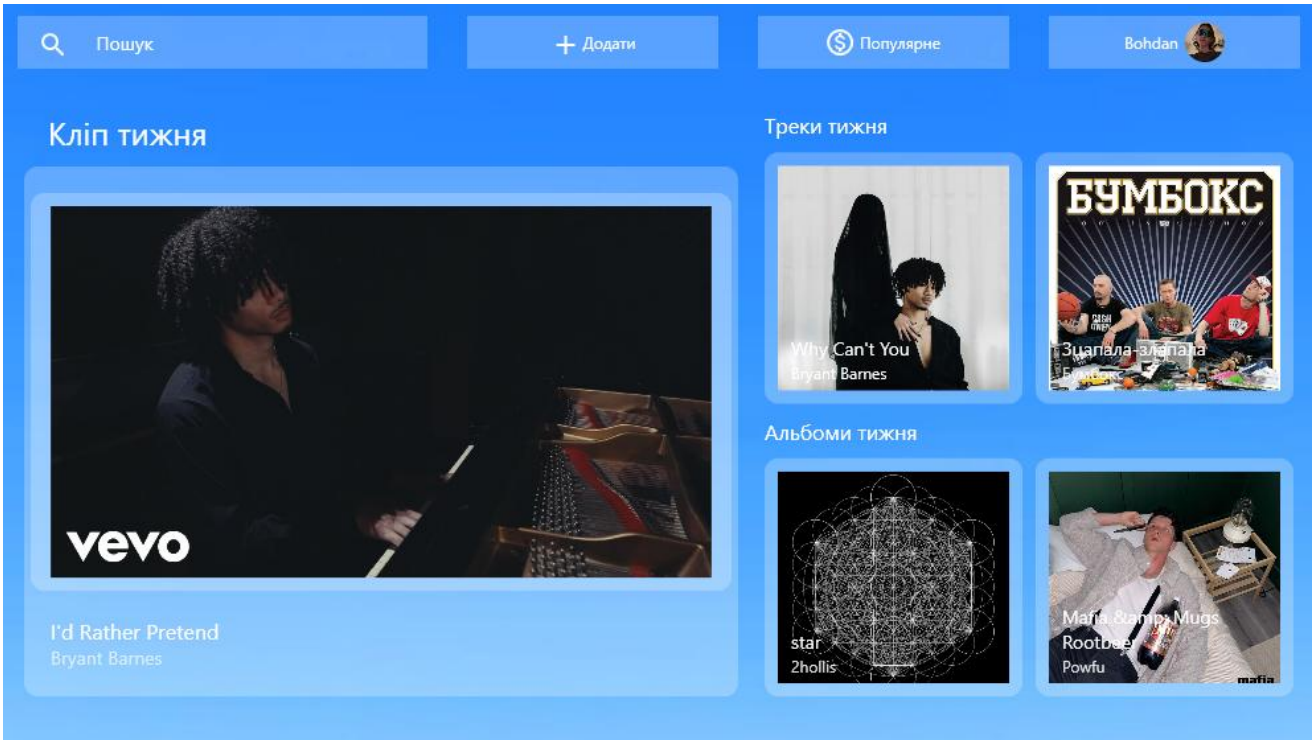


Рис.15 Основний контент з заповненими елементами

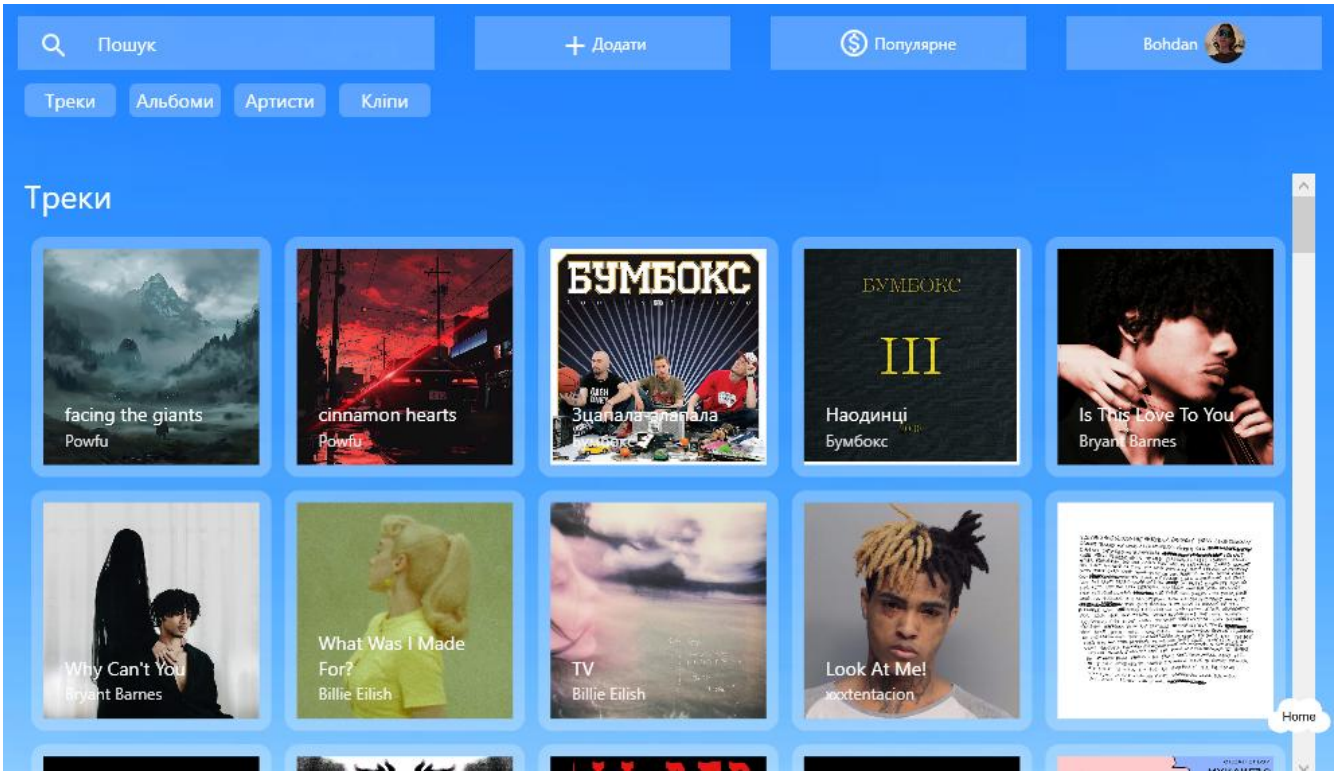


Рис.16 Ручний пошуку

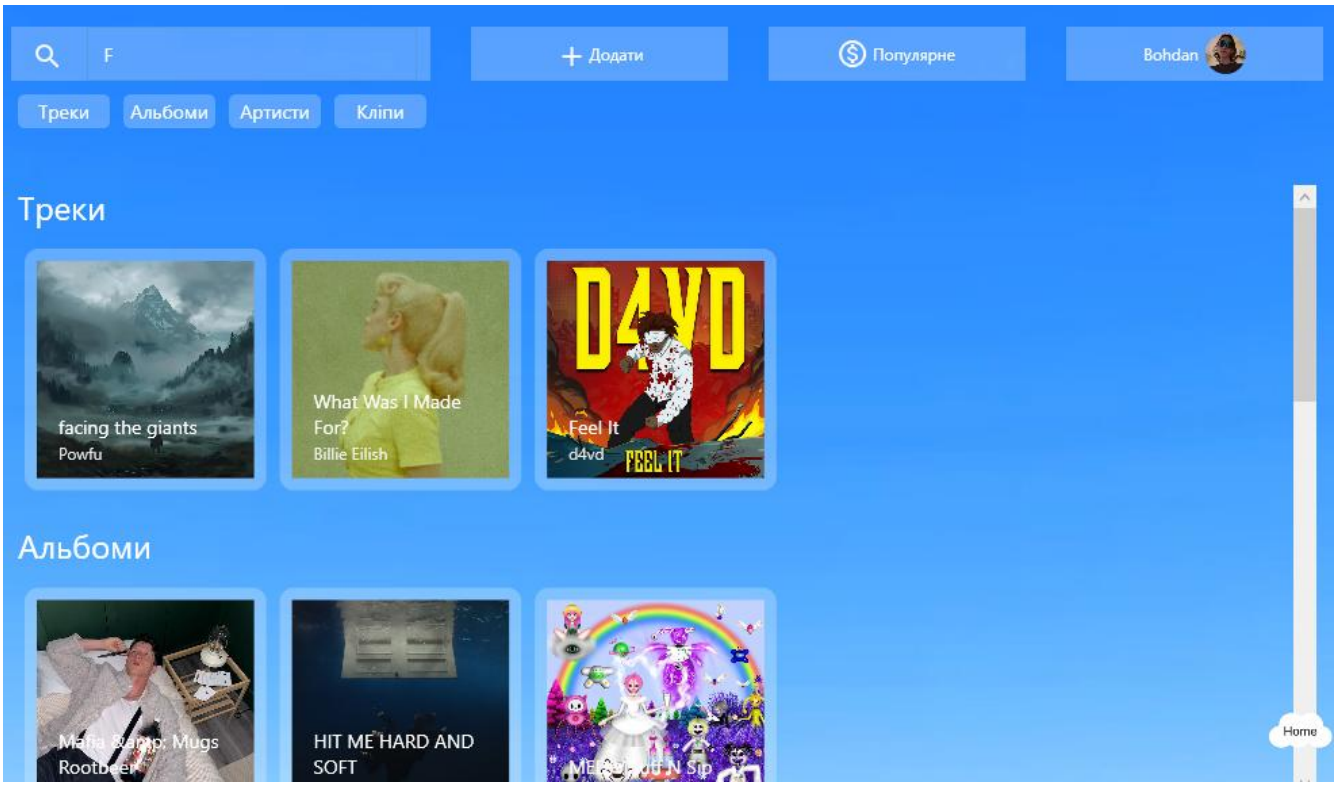


Рис.17 Системний пошуку

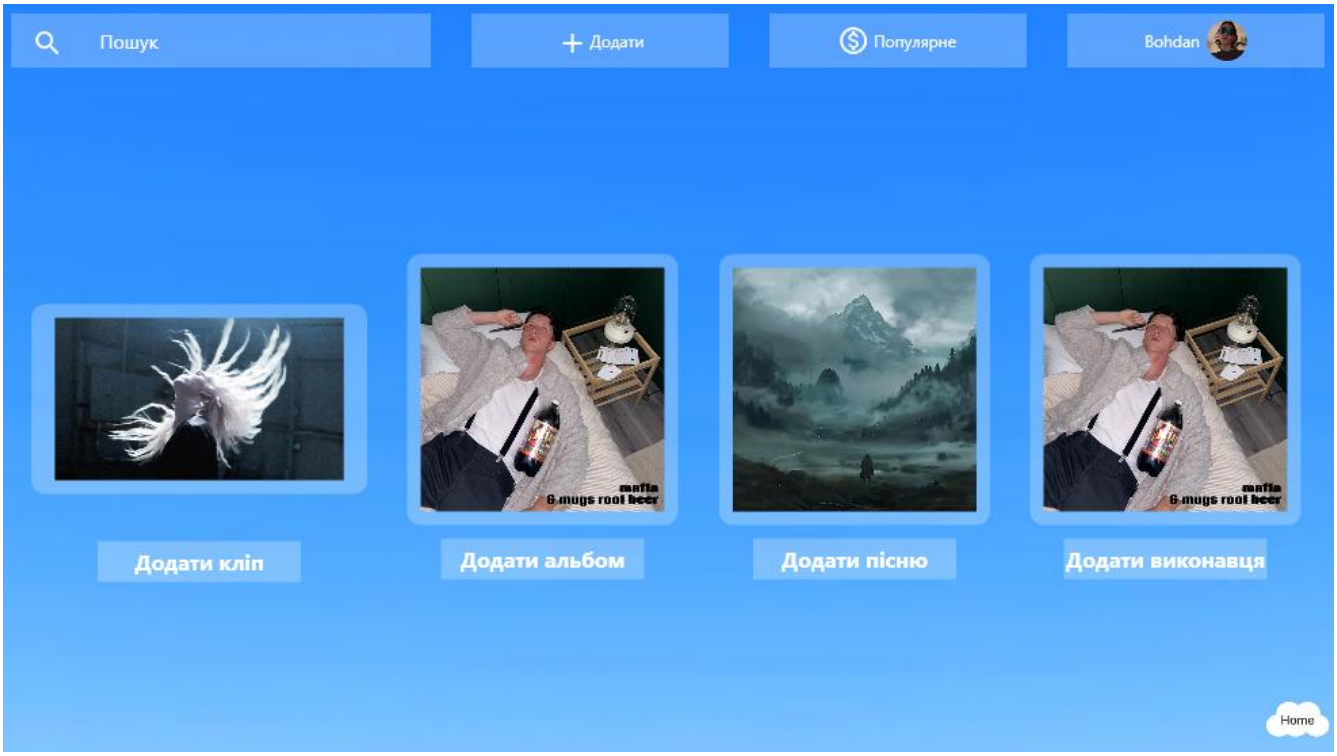


Рис.18 Вікно для вибору додавання контенту

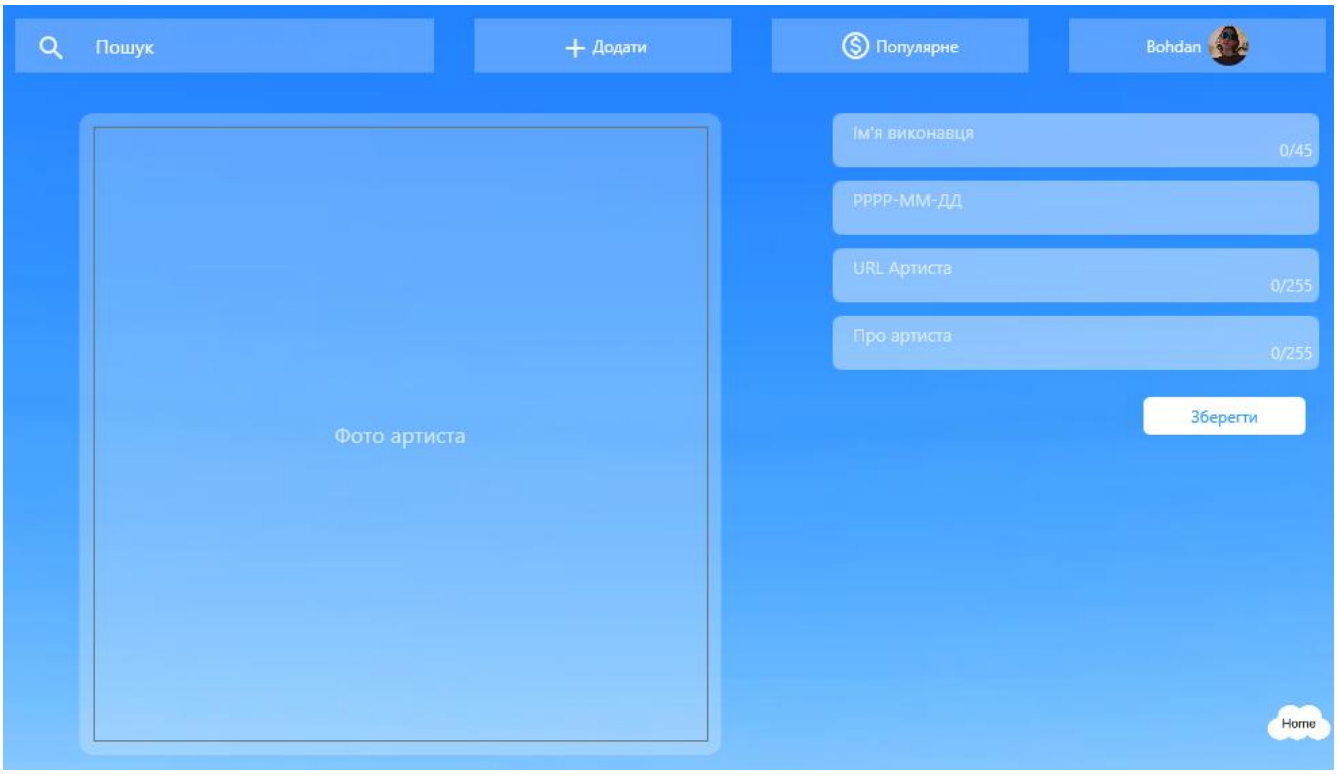


Рис.19 Вікно для додавання артиста

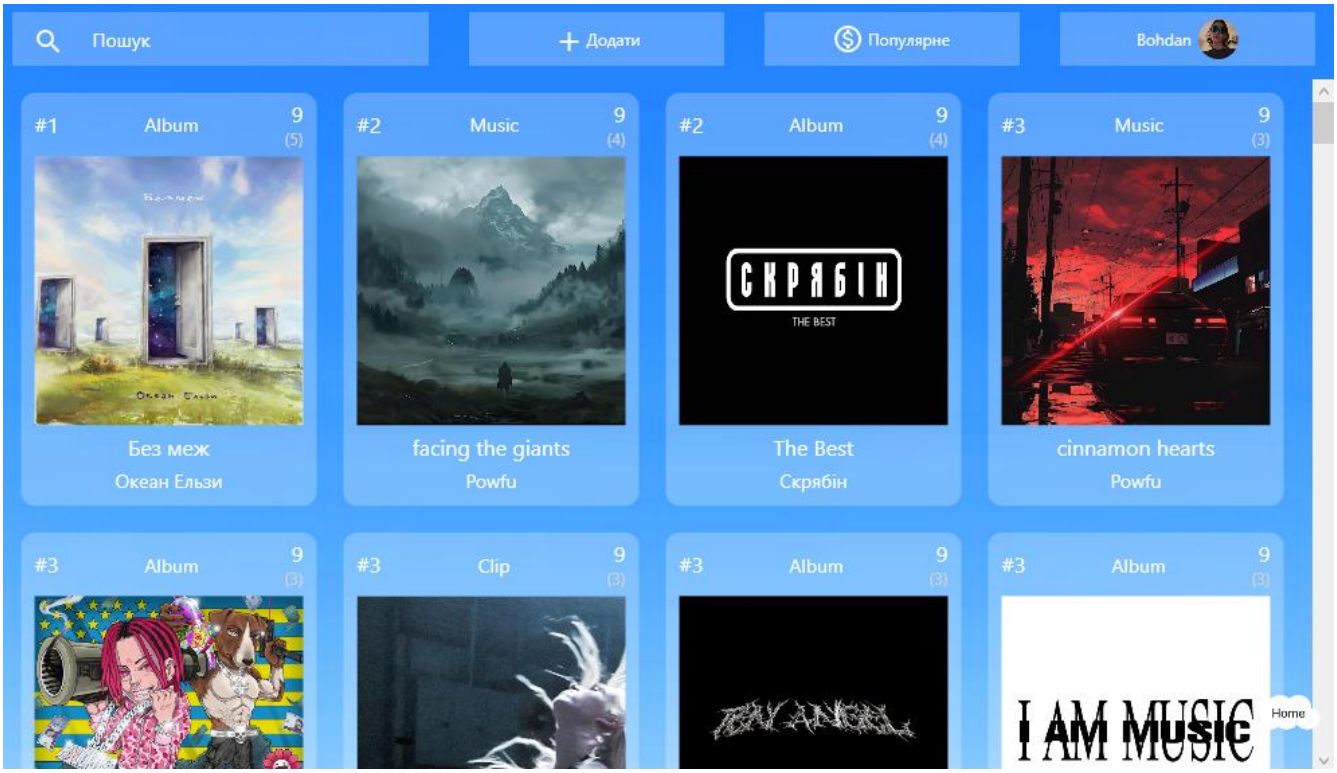


Рис.20 Найбільш популярні музичні елементи

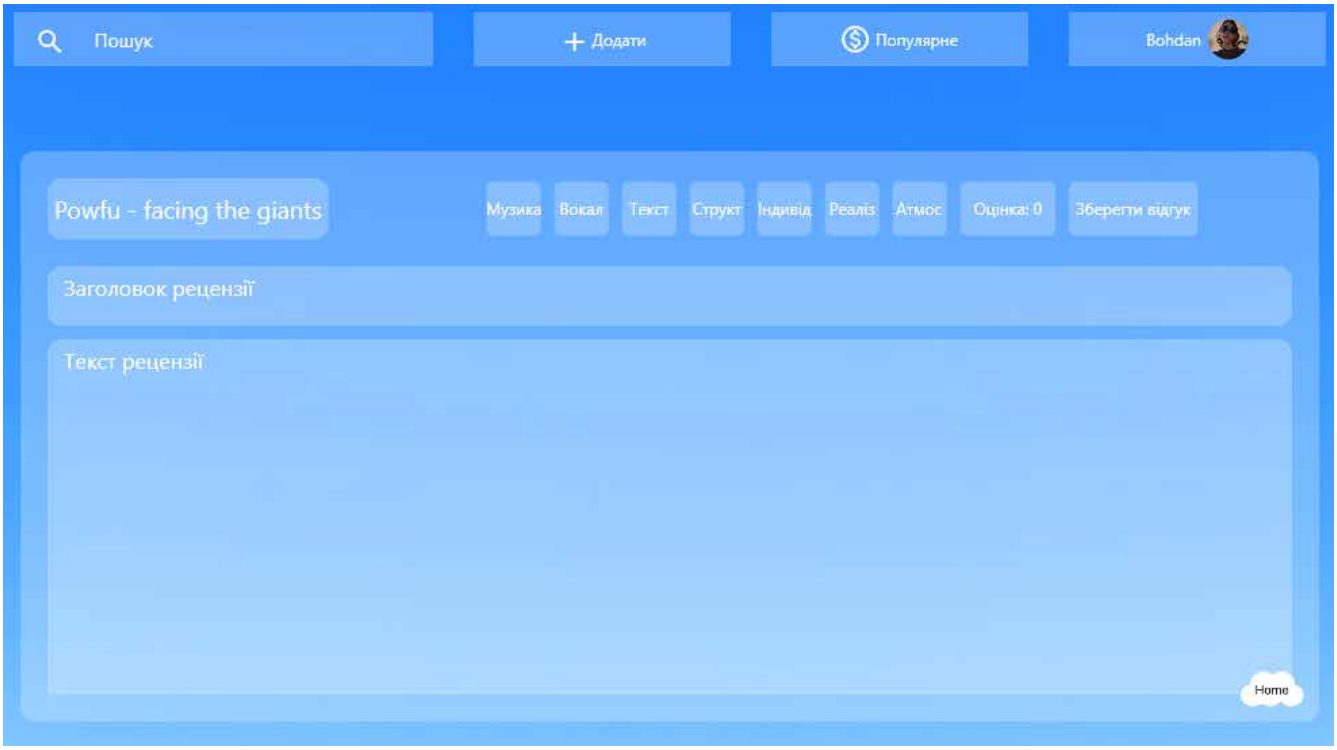


Рис.21 Сторінка для рецензування та оцінки музичного контенту

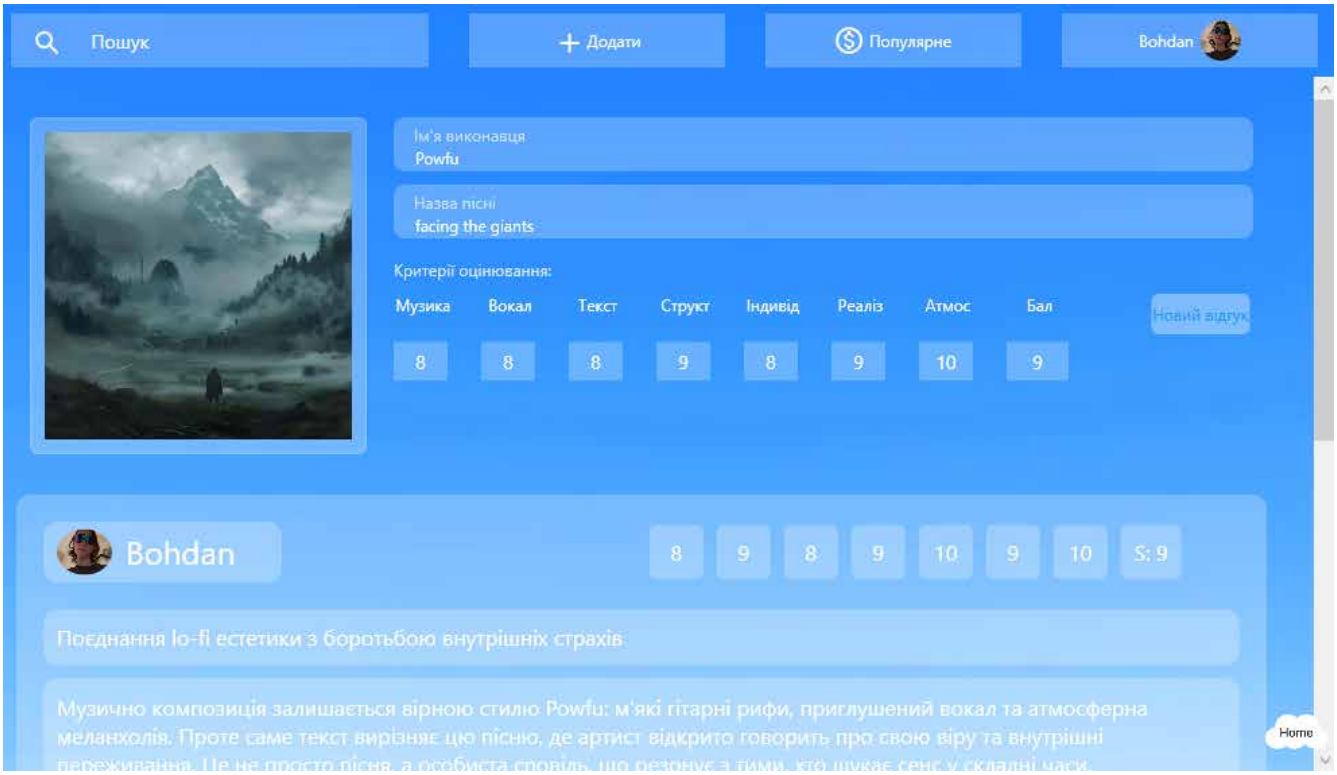
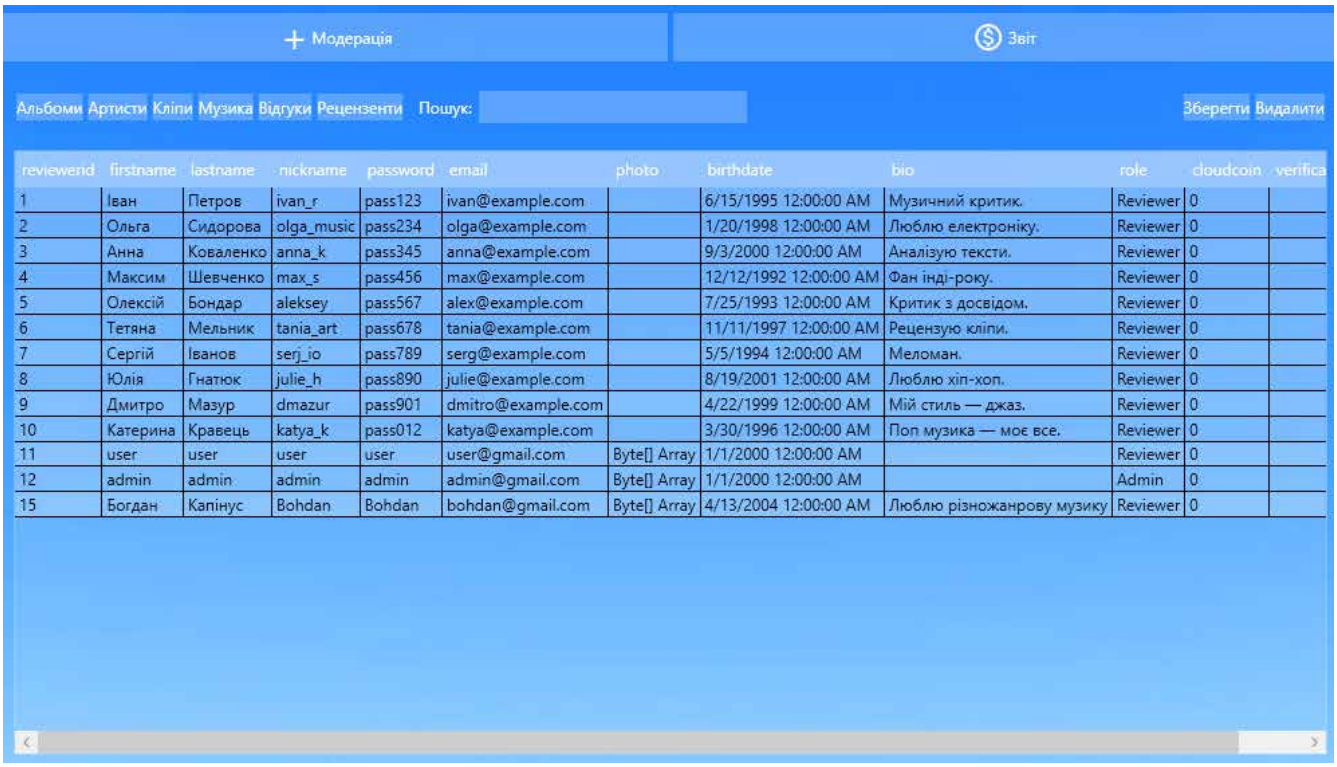


Рис.22 Сторінка для перегляду рецензій та оцінок



reviewerid	firstname	lastname	nickname	password	email	photo	birthdate	bio	role	cloudcoin	verifica
1	Іван	Петров	ivan_r	pass123	ivan@example.com		6/15/1995 12:00:00 AM	Музичний критик.	Reviewer	0	
2	Ольга	Сидорова	olga_music	pass234	olga@example.com		1/20/1998 12:00:00 AM	Люблю електроніку.	Reviewer	0	
3	Анна	Коваленко	anna_k	pass345	anna@example.com		9/3/2000 12:00:00 AM	Аналізую тексти.	Reviewer	0	
4	Максим	Шевченко	max_s	pass456	max@example.com		12/12/1992 12:00:00 AM	Фан інді-року.	Reviewer	0	
5	Олексій	Бондар	aleksey	pass567	alex@example.com		7/25/1993 12:00:00 AM	Критик з досвідом.	Reviewer	0	
6	Тетяна	Мельник	tania_art	pass678	tania@example.com		11/11/1997 12:00:00 AM	Рецензую кліпи.	Reviewer	0	
7	Сергій	Іванов	serj_io	pass789	serg@example.com		5/5/1994 12:00:00 AM	Меломан.	Reviewer	0	
8	Юлія	Гнатюк	julie_h	pass890	julie@example.com		8/19/2001 12:00:00 AM	Люблю хіп-хоп.	Reviewer	0	
9	Дмитро	Мазур	dmazur	pass901	dmitro@example.com		4/22/1999 12:00:00 AM	Мій стиль — джаз.	Reviewer	0	
10	Катерина	Кравець	katya_k	pass012	katya@example.com		3/30/1996 12:00:00 AM	Поп музика — моє все.	Reviewer	0	
11	user	user	user	user	user@gmail.com	Byte[] Array	1/1/2000 12:00:00 AM		Reviewer	0	
12	admin	admin	admin	admin	admin@gmail.com	Byte[] Array	1/1/2000 12:00:00 AM		Admin	0	
15	Богдан	Капінус	Bohdan	Bohdan	bohdan@gmail.com	Byte[] Array	4/13/2004 12:00:00 AM	Люблю різножанрову музику	Reviewer	0	

Рис.23 Меню модерації

Назва	Тип	Кількість рецензій	Музика	Вокал	Текст	Темп	Індивідуальність	Реалізац
facing the giants	Музика	4	8	8	8	8	8	8
cinnamon hearts	Музика	3	8	8	8	8	8	8
Obsessed With You (Official Video)	Кліп	2	9	9	9	8	9	9
Наодинці	Музика	2	8	8	8	7	8	8
Is This Love To You	Музика	2	9	9	9	8	9	9
DYM (Official Music Video)	Кліп	2	9	9	9	8	9	9
Y2K [Official Music Video]	Кліп	2	9	9	9	8	9	9
TV	Музика	1	9	9	8	8	9	8
Cuffed (Official Music Video)	Кліп	2	9	9	9	8	9	9
Revenge	Музика	1	9	8	9	8	9	9
Lucid Dreams (Official Music Video)	Кліп	2	9	9	9	8	9	9
i need u	Музика	1	8	8	8	7	8	8
ALL RED	Музика	1	9	8	7	8	9	9
Олені | Official Video	Кліп	2	9	9	9	8	9	9
Без тебе | Lyric Video	Кліп	2	9	9	9	8	9	9
Той день	Музика	1	8	8	8	7	8	8
Gen Z Luv	Музика	1	9	9	8	8	9	9
Старі фотографії [Official Video]	Кліп	2	9	9	9	8	9	9
Кольорова [Official Video]	Кліп	2	9	9	9	8	9	9
Lyaguminka	Музика	1	8	8	8	7	8	8
Lucid Dreams	Музика	1	9	8	8	8	9	8
На вінілі	Кліп	2	9	9	9	8	9	9
Без тебе	Музика	1	8	8	8	7	8	8
Тільки в твоїх обіймах	Кліп	2	9	9	9	8	9	9
Here With Me [Official Music Video]	Кліп	2	9	9	9	8	9	9

Рис.24 Звіт по рейтинг контенту

	A	B	C	D	E	F	G	H	I	J	K	L
1	Назва	Тип	Кількість рецензій	Музика	Вокал	Текст	Темп	Індивідуальність	Реалізація	Атмосфера	Загальна оцінка	
2	facing the giants	Музика	4	8	8	8	8	8	8	9	9	
3	cinnamon hearts	Музика	3	8	8	8	8	8	8	9	9	
4	Obsessed With You (Official Video)	Кліп	2	9	9	9	8	9	9	9	9	
5	Наодинці	Музика	2	8	8	8	7	8	8	8	9	
6	Is This Love To You	Музика	2	9	9	9	8	9	9	9	9	
7	DYM (Official Music Video)	Кліп	2	9	9	9	8	9	9	9	9	
8	Y2K [Official Music Video]	Кліп	2	9	9	9	8	9	9	9	9	
9	TV	Музика	1	9	9	8	8	9	8	9	9	
10	Cuffed (Official Music Video)	Кліп	2	9	9	9	8	9	9	9	9	
11	Revenge	Музика	1	9	8	9	8	9	9	9	9	
12	Lucid Dreams (Official Music Video)	Кліп	2	9	9	9	8	9	9	9	9	
13	i need u	Музика	1	8	8	8	7	8	8	8	9	
14	ALL RED	Музика	1	9	8	7	8	9	9	8	9	
15	Олені | Official Video	Кліп	2	9	9	9	8	9	9	9	9	
16	Без тебе | Lyric Video	Кліп	2	9	9	9	8	9	9	9	9	
17	Той день	Музика	1	8	8	8	7	8	8	8	9	
18	Gen Z Luv	Музика	1	9	9	8	8	9	9	9	9	
19	Старі фотографії [Official Video]	Кліп	2	9	9	9	8	9	9	9	9	
20	Кольорова [Official Video]	Кліп	2	9	9	9	8	9	9	9	9	
21	Lyaguminka	Музика	1	8	8	8	7	8	8	8	9	
22	Lucid Dreams	Музика	1	9	8	8	8	9	8	9	9	
23	На вінілі	Кліп	2	9	9	9	8	9	9	9	9	
24	Без тебе	Музика	1	8	8	8	7	8	8	8	9	
25	Тільки в твоїх обіймах	Кліп	2	9	9	9	8	9	9	9	9	
26	Here With Me [Official Music Video]	Кліп	2	9	9	9	8	9	9	9	9	
27	Мам	Музика	1	9	9	9	8	9	9	9	9	
28	На вінілі	Музика	1	8	8	8	7	8	8	8	9	
29	Над хмарами	Музика	1	9	9	9	8	9	9	9	9	
30	Romantic Homicide	Кліп	2	9	9	9	8	9	9	9	9	
31	Before That (live)	Кліп	2	9	9	9	8	9	9	9	9	

Рис.25 Експортовані дані звіту

4.2 Вимоги системи рецензатора

Вимоги до апаратного забезпечення:

1. Мінімальні вимоги:
   - Процесор: Intel Core i3 або AMD Ryzen 3 (1.8 GHz і вище)
   - Оперативна пам'ять: 4 GB RAM
   - Вільне місце на диску: 500 MB
   - Монітор: з роздільною здатністю 1280x720
   - Інтернет-з'єднання: стабільне підключення зі швидкістю від 1 Мбіт/с
2. Рекомендовані вимоги:
   - Процесор: Intel Core i5 або AMD Ryzen 5 (2.4 GHz і вище)
   - Оперативна пам'ять: 8 GB RAM
   - Вільне місце на диску: 1 GB
   - Монітор: з роздільною здатністю 1920x1080
   - Інтернет-з'єднання: стабільне підключення зі швидкістю від 10 Мбіт/с

Вимоги до програмного забезпечення:

1) Операційні вимоги:
   - Windows 10 або Windows 11
2) Середовище виконання:
   - .NET 8.0 Runtime
3) База даних:
   - MySql.Data Server версії 8.0
   - Драйвер MySQL.Data версії 8.3.0
4) Зовнішні бібліотеки:
   - Dapper версії 2.1.66
   - MySQL.Data версії 8.3.0
5) Мережеві вимоги:

- Стабільне інтернет-з’єднання для віддаленої роботи з базою даних

4.3 Вибір та створення інсталяційного пакету

Також було створено першу версію мого інстаційного пакету.

Інсталяційний пакет – це набір файлів та скриптів, що підготовлення для завантаження програмного продукту на комп’ютер користувача.

Види різних пакетів показано в таб.5.

Формат	Операційна система	Призначення	Особливості
.exe	Windows	Стандартний інсталятор або файл виконування	Часто використовується для портативних версій.
.msi	Windows	Стандартний інсталяційний пакет	Підтримка деінсталяції.
.appx	Windows 10,11	Пакет UWP	Вимагає цифрового підпису.
.msix	Windows 10,11	Новий універсальний формат	Поєднує .exe/.msi/.appx.
.dmg	macOS	Образ диску для інсталяції	Відкривається як диск.
.app	macOS	Пакет програми	Це папка з усіма ресурсами.
.pkg	macOS	Інсталяційний пакет	Часто використовується для драйверів.
.deb	Linux	Програмний інсталятор	Підтримує залежності
.apk	Android	Пакет додатку	Можна інсталювати через Play Market або вручну.
.ipa	iOS	Пакет додатку iOS	Потребує сертифікат.

Таб. 5 Формати інсталяційний пакет

Для своєї програми я використовував технологію створення інсталяційного пакету у форматі .appxbundle що працює за основами UWP (Universal Windows Platform). Створення локального завантаження розгортання (sideloading) рис.22, а також можна зручно публікувати програму до Microsoft Store, що забезпечить в легко доступність та більше кількість користувачів.[6]

В середині інсталяційної папки знаходяться 3 основні файли для розгортання програми на комп’ютері рис.26:

1. CloudRate_1.0.1.0_x86_Debug.appxbundle
   - Основний інсталяційний містить в собі скомпільований код програми, XAML-інтерфейс, зображення, стилі та іншу інформацію для додатку.
2. CloudRate_1.0.1.0_x86_Debug.cer
   - Сертифікат безпеки це цифровий підпис для системи Windows щоб вона могла без проблем встановлюватися.
3. Add-AppDevPackage.ps1
   - PowerShell-скрипт для автоматичного в становлення на комп’ютер робить всі потрібні перевірки та запускає для встановлення .appxbundle .

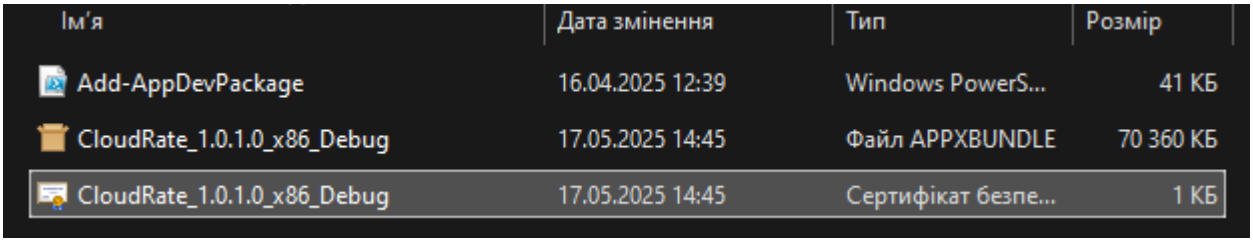


Рис.26 Інсталяційний пакет

Розділ 5. ВИСНОВКИ

В результаті виконання бакалаврської кваліфікаційної роботи було розроблено нову, повноцінно працюючу інформаційно-облікову систему рецензування музичного контенту, що виконує всі поставлені задачі для оцінювання та рецензування музичного контенту.

У процесі розробки було проведено системний аналіз предметної області, за допомогою якого було виявлено основні проблеми всіх існуючих систем, основною з яких стала відсутність єдиного комплексного підходу для оцінки та рецензації. Під час розробки було чітко сформовано набір основних критеріїв для оцінки, такі як вокал, текст, темп, індивідуальність, реалізація та атмосфера, що повністю охоплює всі потреби для моєї системи оцінювання. Мною було перевірено сервіси-конкуренти, такі як Metacritic, Pitchfork та RateYourMusic, що дало змогу зрозуміти, які слабкі сторони та проблемні місця треба виправити для уникнення схожих помилок у своїй системі.

На основі моделювання було розроблено логічну та фізичні структури бази даних у вигляді ER-діаграми, що реалізована в третій нормальній формі для надання цілісності та масштабованості інформації.

Також у межах розробки я реалізував багаторівневу архітектуру з використанням патерну MVVM (Model-View-ViewModel), що дозволив розділити логіку, інтерфейс та обробку даних для можливої в майбутньому розширеності та зручної підтримки. Весь десктопний додаток був створений за допомогою WPF та C#, а також використання реляційної бази даних MySQL Workbench. Для функціональної роботи з базами даних використовується ORM-бібліотека Dapper, яка забезпечує роботу авторизації, додавання музичного контенту, написання та перегляду рецензій, формування оцінок і рейтингів.

Ще було розроблено інсталяційний пакет, який має всі необхідні компоненти для гарного встановлення та запуску комп’ютерного додатку на операційній системі Windows. Також у процесі проектування були визначені мінімальні та рекомендовані вимоги до апаратного та програмного забезпечення, що дозволяють користувачам оцінити можливу сумісність системи з власними технічними ресурсами та забезпечити її стабільну роботу на більшості сучасних комп’ютерів.

Ототожнюючи роботу, можна стверджувати, що створена система повністю відповідає актуальним запитам цифрового музичного середовища, об’єднує

зручність користування з технологічною гнучкістю, а її впровадження сприятиме підвищенню якості аналізу музичного контенту, покращенню зворотного зв’язку між слухачами та творцями, а також відкриває нові можливості для розвитку інструментів рецензування в інших сферах мультимедійного контенту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Albahari J., Albahari B. C# 10.0 in a Nutshell: The Definitive Reference. — Sebastopol: O'Reilly Media, 2022. — 1104 p.
2. Troelsen A., Japikse P. Pro C# 8 with .NET Core: Foundational Principles and Practices in Programming. — Apress, 2020. — 1216 p.
3. Kelleher M., Udell M. Programming WPF: Building Windows UI with XAML and C#. — Microsoft Press, 2021. — 600 p.
4. Vice R. MVVM in WPF Survival Guide. — 2021. — 180 p.
5. IFPI. Глобальний звіт про музичну індустрію [Електронний ресурс] // IFPI Global Music Report. – Офіційний сайт. – Режим доступу: https://globalmusicreport.ifpi.org
6. Packaging UWP apps [Електронний ресурс] // Microsoft Learn : Офіційний сайт. – Режим доступу: https://learn.microsoft.com/en-us/windows/msix/package/packaging-uwp-apps.
7. Трофименко О.Г., Манаков С. Ю., Ларін Д. Г. Основи програмної інженерії : навч.-метод. посібник [Електронне видання] / О. Г. Трофименко, С. Ю. Манаков, Д. Г. Ларін. – Одеса : Фенікс, 2022. – 197 с.
8. Трофименко О. Г. Організація баз даних : навч. посібник / О. Г.Трофименко, Ю. В. Прокоп,. Н. І. Логінова, І. М. Копитчук. 2-ге вид. виправ. і доповн. – Одеса: «Феникс», 2019. – 244 с.
9. Ю.О. Міловідов. «Програмна технологія .NET» Навчальний посібник.– друге видання – Видавничий центр НУБіП України, 2023. – 304 с.
10. Voorhees D.P. Guide to Efficient Software Design: An MVC Approach to Concepts, Structures, and Models. — Bern: Springer, 2020. — 519 p.
11. Діаграма прецедентів [Електронний ресурс] // Вікіпедія – вільна енциклопедія. – Режим доступу: https://uk.wikipedia.org/wiki/Діаграма_прецедентів
12. Модель «сутність — зв'язок» [Електронний ресурс] // Вікіпедія – вільна енциклопедія. – Режим доступу: https://uk.wikipedia.org/wiki/Модель_«сутність_—_зв%27язок»
13. Система керування базами даних [Електронний ресурс] // Вікіпедія : вільна енциклопедія. – Режим доступу: https://uk.wikipedia.org/wiki/Система_керування_базами_даних
14. Архітектура програмного забезпечення [Електронний ресурс] // WEZOM. – 2022. – Режим доступу: https://wezom.com.ua/ua/blog/arhitektura-programmnogo-obespecheniya

15. Мова програмування [Електронний ресурс] // Вікіпедія : вільна енциклопедія. – Режим доступу: https://uk.wikipedia.org/wiki/Мова_програмування.
16. C# (мова програмування) [Електронний ресурс] // Вікіпедія : вільна енциклопедія. – Режим доступу: https://uk.wikipedia.org/wiki/C_Sharp
17. XAML [Електронний ресурс] // Вікіпедія : вільна енциклопедія. – Режим доступу: https://uk.wikipedia.org/wiki/XAML
18. Windows Presentation Foundation [Електронний ресурс] // Вікіпедія : вільна енциклопедія. – Режим доступу: https://uk.wikipedia.org/wiki/Windows_Presentation_Foundation
19. MySQL Documentation [Електронний ресурс] // MySQL: Офіційний сайт. – Режим доступу: https://dev.mysql.com/doc/
20. MVVM Architecture [Електронний ресурс] // Netguru : Офіційний сайт. – Режим доступу: https://www.netguru.com/blog/mvvm-architecture
21. Metacritic – reviews [Електронний ресурс] // Metacritic : Офіційний сайт. – Режим доступу: https://www.metacritic.com/music
22. RateYourMusic – ratings and reviews [Електронний ресурс] // RateYourMusic : Офіційний сайт. – Режим доступу: https://rateyourmusic.com/
23. Pitchfork – reviews and news [Електронний ресурс] // Pitchfork : Офіційний сайт. – Режим доступу: https://pitchfork.com/reviews/albums/
24. Windows Presentation Foundation documentation [Електронний ресурс] // Microsoft Learn : Офіційний сайт. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/
25. .NET documentation [Електронний ресурс] // Microsoft Learn. : Офіційний сайт. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/
26. C# language documentation [Електронний ресурс] // Microsoft Learn : Офіційний сайт. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/csharp/
27. Unit testing C# in .NET using dotnet test and xUnit [Електронний ресурс] // Microsoft Learn : Офіційний сайт. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/core/testing/unit-testing-csharp-with-xunit
28. Діаграма діяльності [Електронний ресурс] // Вікіпедія : вільна енциклопедія. – Режим доступу: https://uk.wikipedia.org/wiki/Діаграма_діяльності
29. Діаграма пакетів [Електронний ресурс] // Вікіпедія : вільна енциклопедія. – Режим доступу: https://uk.wikipedia.org/wiki/Діаграма_пакетів

30. Freeman A., Sanderson A. Pro WPF in .NET 6: Windows Presentation Foundation in .NET 6 — Apress, 2022. — 1136 p.
31. Schmidt K. Design Patterns Explained: A New Perspective on Object-Oriented Design. — Addison-Wesley, 2020. — 352 p.
32. Модель «сутність — зв'язок» [Електронний ресурс] // Вікіпедія – вільна енциклопедія. – Режим доступу: https://uk.wikipedia.org/wiki/Модель_«сутність_—_зв%27язок»
33. Методичні вказівки до виконання бакалаврської кваліфікаційної роботи для студентів спеціальності 122 «Комп’ютерні науки» [Електронний ресурс] / НУБіП України. – Режим доступу: https://nubip.edu.ua/sites/default/files/u423/met_bak_kn_2025n.pdf