Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

ДОСЛІДЖЕННЯ МЕТОДІВ ОПТИМІЗАЦІЇ ПРОЦЕСУ РОЗРОБКИ МУЗИЧНОГО ЗАСТОСУНКУ ЗА ДОПОМОГОЮ АНАЛІЗУ ДАНИХ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-20-2
Бобейко К.С.
(прізвище, ініціали)

Спеціальності 122 Комп’ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник проф. Безсонов О.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Бобейко Катерині Сергіївні
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів оптимізації процесу розробки музичного застосунку за допомогою аналізу даних

затверджена наказом університету від 20 травня 2024 року № 464 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 27 травня 2024 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, матеріали про розробку мобільних застосунків, результати обробки інформації за допомогою аналізу даних, теоретичні відомості про створення рекомендацій на основі існуючих даних.

4. Перелік питань, що потрібно опрацювати в роботі

1. Аналітичні збір та обробка вимог для продукту та огляд інструменатрію для розробки мобільних застосунків з аналізу розробки музичного застосунку.

2. Моделювання музичного застосунку з унікальною функцією підбору плейлисту на основі погодних умов.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Схематичні зображення бази даних, візуалізація прототипу застосунку, схематичне зображення взаємодії користувачів застосунку, діаграма активностей, діаграма классів, аналітична матриця зацікавлених сторін, SQL запити.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	08.04.2024	
2	Аналіз завдання, підбір літератури	08.04.24-15.04.24	
3	Аналіз літератури з досліджуваної проблеми	16.04.24-18.04.24	
4	Аналіз технічних засобів	19.04.24-25.04.24	
5	Розробка методу	26.04.24-14.05.24	
6	Розробка прототипу	15.05.24-23.05.24	
7	Оформлення пояснювальної записки	24.05.24-26.05.24	
8	Перевірка на плагіат	27.05.24	
9	Рецензування	28.05.24	
10	Підготовка презентації та доповіді	29.05.24-04.06.24	
11	Занесення роботи в електронний архів	05.06.24	
12	Попередній захист кваліфікаційної роботи	05.06.24	

Дата видачі завдання 8 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ _____ проф. Безсонов О.О._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 70 с., 6 табл., 31 рис., 31 джерел.

АНАЛІЗ РОЗРОБКИ МУЗИЧНОГО ЗАСТОСУНКУ, UML DIAGRAMS, ВІЗУАЛІЗАЦІЯ АНАЛІЗУ ДАНИХ ЗА ЗАПИТАМИ, POWERBI, ПРОТОТИПУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ.

Об’єктом роботи є аналіз розробки музичного застосунку.

Метою роботи є розробка застосунку з унікальною функціональністю підбору музики під реальний стан погоди для поціновувачів музики та любителів слухати музику пов’язану з навколишньою атмосферою.

У ході роботи проводиться аналітичний огляд існуючих технологій для розробки музичного застосунку, а також аналіз методик архітектурного моделювання розробки музичних застосунків. Розроблені аналітична візуалізація різних процесів розробки застосунку, визначено взаємозв’язок між базами даних для унікальної функції застосунку.

У результаті роботи здійснена візуалізація мобільної версії музичного застосунку та її головна аналітика для відстеження зміни жанру музики за допомогою даних про погоду.

MUSIC APPLICATION DEVELOPMENT ANALYSIS, UML DIAGRAMS, QUERY DATA ANALYSIS VISUALIZATION, POWERBI, MOBILE APPLICATION PROTOTYPING.

The object of the work is the analysis of the development of a music application.

The purpose of the work is to develop an application with a unique functionality of selecting music for real weather conditions for music lovers and those who like to listen to music related to the surrounding atmosphere.

In the course of the work, an analytical review of existing technologies for the development of music applications is carried out, as well as an analysis of architectural modeling techniques for the development of music applications. Analytical visualization of various application development processes was developed, the relationship between databases for a unique application function was determined.

As a result of the work, the visualization of the mobile version of the music application and its main analytics for tracking the change of the music genre using weather data were realized.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ОС – операційна система

Стейкхолдер – зацікавлена сторона

API – Application Programming Interface (інтерфейс прикладного програмування)

UML – Unified Modeling Language (уніфікована мова моделювання)

СУБД – система управління базами даних

БД – база даних

User Stories – користувацькі історії

Feature List – специфікації

Use Cases – сценарій використання

Activity Diagram – діаграма активностей

Class Diagram – діаграма класів

MVP – Minimum Viable Product (мінімально життєздатний продукт)

Post-MVP – наступна версія додатку після впровадження мінімально життєздатного продукту

«Power And Interest» матриця – матриця впливу та інтересу

RACI матриця – матриця відповідальності

Face ID – сканер 3D форми особи

ВСТУП

У сучасному світі музичні програми стали невід’ємною частиною нашого повсякденного життя, збагачуючи наші враження та створюючи неповторну атмосферу в кожний момент. З розвитком технологій і зростанням інтересу до персоналізованих сервісів, виникає потреба в інноваційних підходах до розробки музичних застосунків. У цьому контексті робота присвячена аналізу розробки музичної програми з унікальною функціональністю: підбором музичного плейлиста в реальному часі залежно від погодних умов.

Такий підхід до розробки програми не лише надає користувачеві інноваційний музичний досвід, а й прагне створити гармонію між звуковим супроводом та навколишнім середовищем. Дослідження сфокусовані на тому, щоб зрозуміти, як технології аналізу даних та інформації про погоду в реальному часі можуть бути інтегровані в процес створення персоналізованих плейлистів, що адаптуються до поточних погодних умов. Цей проєкт є інноваційним поглядом на взаємодію музичного та природного світу, що робить його актуальним та захоплюючим об’єктом дослідження в галузі інформаційних технологій [1-3].

З аналітичної точки зору дана робота пропонує інтеграцію даних про погоду в музичний застосунок з використанням методів аналізу даних. Аналіз емоційного впливу музики, поєднаний із погодними умовами, забезпечує персоналізований музичний досвід. Актуальність та новизна проєкту видно в унікальному підході до адаптації музики під зовнішні умови, що може значно підвищити привабливість програми для користувачів та надати цінні дані для подальшого розвитку персоналізованих музичних сервісів.

1 ОГЛЯД СУЧАСНИХ ПІДХОДІВ ДЛЯ РОЗРОБКИ МУЗИЧНОГО ЗАСТОСУНКУ

У сучасному світі розвиток технологій та поширення мобільних пристроїв визначають нові звичаї та підходи до споживання розваг. Однією з найвизначніших та широко використовуваних інновацій в цьому контексті є музичні застосунки, які відіграють ключову роль у впровадженні сучасних підходів до насолодження музикою.

Мобільність та зручність є однією з головних причин, чому сучасне суспільство виявило потребу у розробці музичних застосунків. Смартфони та планшети, що є практично невід'ємними частинами нашого повсякденного життя, стали основними платформами для доступу до різноманітних сервісів. Музичні застосунки забезпечують можливість слухати улюблені треки в будь-якому місці та часі, надаючи користувачам неперевершену мобільність та зручність.

Стрімінгові сервіси, що з'явилися в середині 2010-х, значно змінили спосіб, яким ми сприймаємо та отримуємо музику. Spotify, Apple Music та інші стали не лише платформами для відтворення треків, а й основними джерелами новинок та власних музичних відкриттів. Це відбулося завдяки можливості безпосереднього стрімінгу та відсутності необхідності завантажувати музику на пристрій. Такий підхід вирішив проблему обмеженого простору для зберігання та надав користувачам безперервний доступ до безлічі треків [4].

Однією з ключових переваг музичних застосунків є їхня здатність адаптуватися до вподобань користувачів. Використання технологій штучного інтелекту та машинного навчання дозволяє створювати персоналізовані плейлисти та рекомендації, враховуючи унікальні музичні смаки слухачів. Це створює неповторний музичний досвід та збагачує відносини між слухачами та їхньою музикою.

Застосунки також відіграють роль у створенні віртуальних музичних спільнот, де користувачі можуть спільно слухати, обмінюватися враженнями та взаємодіяти. Це розширює соціальні аспекти музичного досвіду та сприяє створенню спільностей, які об’єднують людей за їхньою любов’ю до музики.

Отже, розробка та використання музичних застосунків не просто задовольняють потреби користувачів у зручному способі слухання музики, але також впливають на спосіб, яким ми взаємодіємо з музичним світом. Ці застосунки не просто створюють зручність, але і трансформують сам процес сприйняття та розуміння музичного мистецтва в цифровому віці.

1.1 Опис продукту

Розроблюваний продукт – це музичний застосунок для мобільної версії. Музичний застосунок – це програмне забезпечення, створене для роботи з музикою, що надає користувачам можливість створення, редагування, відтворення або обробки звукових файлів. Такі застосунки можуть включати в себе різноманітні функції, такі як створення музики, музичний редактор, стрімінговий сервіс, аналіз звуку, ефекти та інше. Вони розробляються для різних платформ, включаючи комп’ютери, смартфони та планшети, і можуть задовольняти потреби як професійних музикантів, так і звичайних користувачів.

Мета проєкту – надати споживачам якісний продукт, який буде викликати тільки позитивні емоції. Сьогодні практично у кожного є смартфон. А завдяки музичним програмам люди можуть відволіктися та розслабитися. Є потреба створення музичного застосунку, який на сьогодні стане найкращим рішенням як для звичайних користувачів, так і для артистів. Це буде висококласна, проста у використанні програма з ширшою базою виконавців і більшою кількістю корисних опцій. Таким чином,

враховуючи всі переваги програми, застосунок розраховує взяти значну частину ринку та мати хороші переваги [5, 6].

Цей мобільний музичний застосунок буде унікальним, тому що, він дозволяє користувачам не просто слухати музику та створювати власні плейлисти, а й редагувати аудіозаписи, встановлювати ефекти, і навіть створювати власну музику чи ремікси. Він може також підтримувати функції стрімінгу для прослуховування музики онлайн, інтеграцію з соціальними мережами та Shazam, а також інші зручні можливості для задоволення потреб користувачів у мобільному музичному досвіді. Користувачі зможуть знайти артистів на будь-який смак, оскільки у застосунку буде розширена база, де будуть представлені молоді та маловідомі, але дуже талановиті та амбітні артисти та групи. Завдяки цьому застосунку користувачі зможуть розширити свій музичний кругозір. У застосунку буде розроблена ексклюзивна аналітика підбору музичного плейлісту на основі погодних даних, яких немає в жодному іншому музичному застосунку. Всі ці характеристики дають гарну можливість віднайти успіх у цьому проєкті та просунутись у сфері аналітики даних.

1.2 Аналіз підходів мобільної розробки

У сучасну цифрову епоху мобільні застосунки стали невід’ємною частиною нашого життя. Незалежно від того, чи то для спілкування, розваг чи продуктивності, мобільні програми відіграють вирішальну роль у покращенні нашого щоденного досвіду. За лаштунками кваліфіковані розробники використовують різні мови програмування, щоб втілити ці програми в життя. У цьому параграфі буде досліджено деякі з найкращих мов програмування для розробки мобільних застосунків, висвітлюючи їхні особливості, переваги та випадки використання [7].

1.2.1 Мова програмування

Нам потрібна крос-платформена підтримка для обох основних мобільних платформ (iOS та Android), тож потрібно розглянути використання наступних мов та фреймворків.

React Native фреймворк дозволяє розробляти мобільні застосунки за допомогою JavaScript. Він забезпечує можливість використовувати один код для обох платформ, що спрощує розробку та зменшує час витрачений на неї. React Native має широку спільноту та ряд сторонніх бібліотек, що полегшує інтеграцію з різними API, включаючи погодні сервіси [8].

Flutter фреймворк використовує мову програмування Dart та дозволяє створювати крос-платформені мобільні застосунки з дуже високою продуктивністю. Flutter також надає можливість розробляти один код для обох платформ, але він відрізняється від React Native своїм власним шаром інтерфейсу користувача, що надає більш однорідний вигляд на різних платформах. За допомогою Flutter можна швидко створювати відповідні анімації та інтерактивність, що може бути корисним для розробки музичного застосунку зі зручною та привабливою користувацькою інтерфейсом [9].

Java незалежний від платформи, це дозволяє один раз написати код і запускати його на різних пристроях. Широкий набір бібліотек і фреймворків, таких як Android SDK, для створення різноманітних мобільних програм. Скорочення часу та зусиль на розробку завдяки використанню цієї функціональності [10].

C# – це універсальна мова програмування, часто використовується для крос-платформенної розробки мобільних застосунків. Завдяки фреймворкам, таким як Xamarin, дозволяє розробникам писати код, який можна використовувати на різних платформах, включаючи Android та iOS. Спільне використання коду дозволяє значно скоротити час і витрати на розробку. C# пропонує надійну та розширену стандартну бібліотеку, а також функції, такі

як збирання сміття та асинхронне програмування. Потужна мова для створення продуктивних і масштабованих мобільних застосунків [11].

У світі швидко зростаючого ринку мобільних застосунків вибір правильного інструменту для розробки стає ключовим фактором для успіху. Особливо це стосується музичних застосунків, які потребують не лише функціональності, але і привабливого та інтуїтивного інтерфейсу для залучення користувачів. Далі треба розглянути та обгрутнувати вибір найкращого шляху для написання музичного застосунку на різні мобільні платформи, зосереджуючись на фреймворках React Native та Flutter.

Перше, що варто врахувати, це крос-платформеність. У відповідь на зростаючий попит на мобільні застосунки для різних платформ, розробники використовують фреймворки, які дозволяють їм писати один код, який можна використовувати на різних платформах. Обидва фреймворки, React Native та Flutter, пропонують цю можливість, забезпечуючи економію часу та зусиль на розробку.

Другий аспект – продуктивність та швидкість розробки. Flutter славиться своєю високою продуктивністю, завдяки можливості гарячого перезавантаження, що дозволяє розробникам бачити зміни в реальному часі. Це дозволяє швидше та ефективніше розробляти та тестувати застосунок, зменшуючи час, необхідний для випуску на ринок.

Останнім, але не менш важливим аспектом є інтерфейс користувача. Flutter має власний шар інтерфейсу, що надає однорідний вигляд та взаємодію на різних платформах. Це дозволяє створювати привабливий та консистентний дизайн, що є ключовим фактором для музичного застосунку, який має привернути увагу користувачів.

Отже, розглядаючи всі аспекти, Flutter виявляється найкращим вибором для написання музичного застосунку на різні мобільні платформи. Його крос-платформеність, продуктивність та можливості створення привабливого інтерфейсу користувача роблять його ідеальним інструментом для досягнення успіху в цьому конкурентному ринку [12].

1.2.2 Backend Framework

Для написання бекенду для мобільного музичного застосунку існує кілька можливих варіантів, включаючи використання різних мов програмування, фреймворків та платформ. Ось деякі з них, разом із коротким описом:

Node.js з Express або NestJS: Node.js – це середовище виконання JavaScript, що дозволяє створювати високопродуктивні та масштабовані серверні застосунки. Express – це легкий вебфреймворк для Node.js, який дозволяє швидко створювати RESTful API. NestJS – це модульний Node.js фреймворк, який забезпечує елегантний та ефективний спосіб створення серверних застосунків.

Python з Django або Flask: Python – це потужна мова програмування з багатим екосистемою. Django – це високорівневий вебфреймворк, який дозволяє швидко створювати повнофункціональні вебзастосунки. Flask – це легкий вебфреймворк для Python, який надає гнучкість та простоту в розробці вебзастосунків.

Java з Spring Boot або Micronaut: Java – це мова програмування, яка широко використовується для створення масштабованих індустріальних застосунків. Spring Boot – це фреймворк для створення вебзастосунків на Java, який пропонує широкі можливості для реалізації бекенду. Micronaut – це легкий і швидкий фреймворк для створення мікросервісів, який може бути використаний для створення бекенду мобільного застосунку.

З урахуванням використання Flutter для фронтенду, найкращим варіантом для написання бекенду може бути Node.js з Express або NestJS. Ось кілька обгрунтувань для цього вибору:

– якщо обраним фреймворком є Flutter для фронтенду, використання JavaScript або TypeScript для бекенду може спростити розробку, оскільки ці мови мають подібний синтаксис.

– Node.js має широку екосистему пакетів та бібліотек, що спрощує розробку серверної частини застосунку.

– Express або NestJS надають потужні засоби для створення RESTful API, які можуть ефективно взаємодіяти з Flutter на фронтенді.

– Node.js добре підходить для створення високопродуктивних та масштабованих серверних застосунків.

Отже, Node.js з Express або NestJS може бути оптимальним вибором для написання бекенду мобільного музичного застосунку з урахуванням використання Flutter для фронтенду [13, 14].

1.2.3 База даних та СУБД

База даних – це організована колекція даних, що описує їх характеристики та взаємозв’язки. Включає схеми, таблиці, збережені процедури та інші об’єкти. Вона організована за моделлю даних і містить засоби для обробки даних. Бази даних використовуються для зберігання різних типів інформації, від автоматизованих систем обліку до систем управління вмістом інтернет-сайтів. Вони дозволяють інтегрувати запити та забезпечують автоматичний перехід від структури бази даних до програмного середовища користувача за допомогою систем управління базами даних (СУБД).

Звісно, якщо йде мова про створення музичного застосунку то є потреба знайти необхідні для роботи музичні бібліотеки. У цю категорію потрапляють популярні програми потокової передачі музики, такі як Spotify, Pandora і Apple Music. Тут музика зберігається в серверних бібліотеках, і користувачі мають необмежений доступ до музики, але за умов власника програми [15].

Було створено аналітику для поетапної розробки мобільного застосуноку, який автоматично підбиратиме музичний супровід відповідно

до погодних умов у реальному часі. Це дозволить користувачам насолоджуватися музикою, яка відповідає їхньому настрою та атмосфері, створеній навколишнім середовищем.

Для реалізації цієї функціональності було використано дані про погоду та музику з Kaggle. На платформі Kaggle було зібрано доступні дані про погоду з різних джерел, такі як температура, вологість, швидкість вітру тощо, які будуть використовуватися для визначення погодних умов. Також було використано дані про музичні уподобання користувачів, які також можуть бути доступні на Kaggle, щоб персоналізувати вибір треків.

Щодо бази даних і системи управління базами даних (СУБД), було обрано потужну та надійну технологію – Microsoft SQL Server Management Studio 18. Ця реляційна база даних дозволить ефективно зберігати та організовувати дані про погоду та музику. Також можна використовувати NoSQL базу даних – MongoDB, якщо потрібно зберігати дані у вигляді документів або ключ-значення.

Організація бази даних була спроєктована таким чином, щоб ефективно забезпечити швидкий доступ до неї під час вибору музичних треків у реальному часі відповідно до погодних умов. Запити до бази даних були оптимізовані для швидкого виконання, з урахуванням великого обсягу даних, які можуть надходити у реальному часі з джерел погодних даних [16, 17].

1.2.4 Cloud сервіси

Хмарні музичні програми надають користувачам зручне зберігання та організацію їхньої музики, а також можливість відтворювати її з будь-якого місця у будь-який час. Основні переваги хмарних музичних сервісів полягають у зручності доступу до музики з будь-якого місця та пристрою, можливості синхронізації музичної колекції між різними пристроями, можливості створення власних плейлистів та рекомендаційних систем для

відкриття нової музики. Крім того, хмарні сервіси зазвичай надають функцію потокового відтворення, що дозволяє слухати музику без необхідності завантаження її на пристрій. Для подальшої роботи було використано AudioBox [15].

1.3 Постановка задачі

Проаналізувавши існуючі рішення в області аналізу даних та існуючі засоби для створення мобільних застосунків на Flutter, можна наголосити про актуальність проблеми та висунути ряд функціональних і нетехнічних вимог щодо розроблюваного прототипу застосунку [18].

Об’єктом роботи є аналіз розробки музичного застосунку.

Метою роботи є розробка застосунку з унікальною функціональністю підбору музики під реальний стан погоди для поціновувачів музики та любителів слухати музику пов’язану з навколишньою атмосферою.

Застосунок призначений для використання в особистих цілях користувача на мобільних пристроях. Його особливість полягає в тому, що музика, яку він пропонує, відповідає реальному стану погоди. Необхідно забезпечити можливість збору даних погодних умов у реальному часі та на основі цих даних вміти створювати плейлисти з відповідною до настрою музикою.

Для досягнення мети необхідно вирішити такі завдання щодо розробки прототипу застосунку:

– створення зручного та інтуїтивно зрозумілого інтерфейсу, який дозволить користувачам легко взаємодіяти з програмою;

– реалізація збору інформації про погоду;

– реалізація алгоритмів розрахунку музичних композицій відповідно до погодних умов;

– проведення детального аналізу вимог майбутнього продукту із залученням усіх стейкхолдерів;

– розробка прототипу музичного застосунку із можливістю залучення масштабної аудиторії слухачів;

– можливість експорту даних статистичного аналізу для подальшого вивчення або збереження.

2 МОДЕЛЮВАННЯ АРХІТЕКТУРИ ТА АНАЛІЗ ПРОЦЕСІВ РОЗРОБКИ ЗАСТОСУНКУ

Моделювання архітектури та аналіз процесів розробки застосування є фундаментальним етапом у створенні програмного забезпечення. Ці процеси дозволяють розробникам не лише визначити структуру та функціональність програми, але й забезпечити ефективність та якість у всіх аспектах розробки.

Моделювання архітектури зазвичай включає створення архітектурних діаграм, таких як UML діаграми компонентів, діаграми пакетів, діаграми розгортання тощо. Ці діаграми допомагають візуалізувати структуру програми, її компоненти та взаємозв’язки між ними. Вони дозволяють розробникам краще розуміти архітектурні потреби та вимоги проєкту, а також виявляти можливі проблеми та ризики в ранній стадії розробки.

Аналіз процесів розробки включає в себе оцінку та оптимізацію процесів, які використовуються під час розробки програмного забезпечення. Це може включати в себе використання методологій розробки програмного забезпечення, таких як Agile, Scrum, Kanban тощо, а також інструментів управління проєктами та комунікації. Аналіз процесів розробки допомагає забезпечити ефективність та продуктивність розробницької команди, а також покращує якість та швидкість поставки програмного забезпечення.

У цьому розділі було детально розглянуто ключові кроки, які допомагають розробникам зрозуміти потреби користувачів, визначити вимоги до продукту та розробити його архітектуру: аналіз стейкхолдерів, збір початкових вимог, опис ключових вимог та їх декомпозицію, а також пріоритизацію вимог. Ці етапи допомагають зрозуміти потреби користувачів, визначити функціональність продукту та розподілити завдання за пріоритетами. Без цих етапів не можливо створити якісний продукт, бо аналітика допомагає наперед розрахувати різні можливі ситуації та прорахувати ризики.

2.1 Проєктні вимоги та їх декомпозиція

Декомпозиція проєктних вимог відіграє визначну роль у впорядкуванні та ефективному керуванні процесом розробки проєктів. Цей процес передбачає розбиття складних та об’ємних вимог на менші, керовані компоненти, що сприяє кращому розумінню завдань, полегшує планування та реалізацію, а також дозволяє краще керувати ресурсами та термінами проєкту.

Ідентифікація вимог є першим та ключовим етапом у декомпозиції. Важливо уточнити всі функціональні, нефункціональні та технічні вимоги, що стосуються проєкту. Функціональні вимоги визначають, що має робити продукт чи сервіс, нефункціональні – якість, продуктивність, безпеку тощо, а технічні – технології та архітектуру. Після чого вимоги можна розбити на менші, більш конкретні елементи.

Декомпозиція на компоненти вимагає систематичного та логічного підходу. Кожен вимогу можна поділити на менші підзадачі, що спрощує їх реалізацію та контроль. Наприклад, велику функцію можна розбити на кілька менших, керованих модулів, що дозволить розробникам працювати ефективніше та зосередитися на конкретних завданнях.

Оцінка пріоритетів грає важливу роль у процесі декомпозиції. Визначення пріоритетів дозволяє визначити порядок реалізації та ресурси, які необхідно виділити на кожний компонент. Це допомагає керувати термінами та ресурсами проєкту, а також забезпечити вчасну доставку результатів.

Моніторинг та контроль – це завершальний етап у процесі декомпозиції. Після розбиття вимог на компоненти важливо постійно контролювати та відстежувати їх реалізацію. Це дозволяє вчасно виявляти будь-які зміни або відхилення від початкових вимог та вносити необхідні корективи.

Отже, декомпозиція проєктних вимог – це ключовий етап у процесі розробки, який допомагає краще розуміти, планувати та реалізовувати проєкти, забезпечуючи при цьому їх вчасне та успішне завершення [19].

2.1.1 Аналіз стейкхолдерів

Стейкхолдери – це учасники проєкту, організації або особи, які мають інтереси, права або частку в системі чи проєкті. Вони відіграють важливу роль в успіху проєкту, оскільки їхні інтереси та підтримка можуть істотно впливати на результати роботи. Вони можуть бути як внутрішніми членами команди проєкту, так і зовнішніми сторонами, включно з клієнтами, користувачами, керівництвом компанії, інвесторами, партнерами та іншими зацікавленими сторонами.

У контексті інформаційних технологій ці зацікавлені особи відіграють вирішальну роль у кожному етапі проєкту. Починаючи від визначення вимог і цілей проєкту, вони надають цінний зворотний зв’язок та інформацію про те, що потрібно розробити або поліпшити. Це може бути як технічна інформація від розробників, так і переваги кінцевих користувачів, які часто відіграють критичну роль в успіху продукту [20].

Аналіз стейкхолдерів – це систематичний процес визначення, оцінювання та управління особами або групами, які можуть бути впливовими або впливати на успіх організації, проєкту або ініціативи. Стейкхолдери – це будь-які особи, організації або групи, які мають інтереси або вплив на діяльність або результати певної діяльності.

Основна мета аналізу стейкхолдерів – зрозуміти їхні потреби, очікування, інтереси та вплив на проєкт або організацію. Це допомагає забезпечити, що їхні інтереси враховані під час прийняття рішень та планування стратегій [21-23].

Для проведення аналізу зацікавлених сторін можуть використовуватись різні техніки. Після знайомства зі стейколдерами на інтерв'ю сесії та визначення їх позицій була розроблена таблиця на основі даних усіх стейкхолдерів проєкту. Нижче продемонстрована таблиця в якій була проведена оцінка впливу і важливості кожної зацікавленої сторони та описані коментарі (табл. 2.1).

Таблиця 2.1 – Перелік зацікавлених сторін

Ім'я стейкхолдера	**Роль**	**Вплив**	**Важливість**	**Ризики та припущення**
1	**2**	**3**	**4**	**5**
Віктор Махно	Спонсор	Високий	Низький	Хоче отримати власний прибуток. Він вчасно фінансує проєкт, але не зацікавлений у проєкті.
Ганна Бойко	Замовник	Високий	Високий	Вона прагне висококласного рішення. Зацікавлена розробкою інтерфейсу.
Суспільство	Кінцевий користувач	Високий	Низький	Люди хочуть отримати зручний, багатофункціональний продукт, хочуть, щоб заряд батареї не витрачався занадто сильно.
Сергій Черемиш	Керівник проєкту	Низький	Високий	Він прагне реалізувати найкраще можливе рішення. Завжди допомагає команді, контроль процесу.

Продовження таблиці 2.1

1	2	3	4	5
Андрій Хома	Власник проєкту	Високий	Високий	Знає усе про проєкт. Має тісний контакт із замовником продукту.
IT-123 команда	Команда розробки	Низький	Високий	Можуть вносити правки та пропонувати більш новітні та ліпші рішення. Прямі
Рекламне агентство	Медіа група	Низький	Низький	Основна мета – отримати винагороду за рекламу нашого продукту. Жодної участі.
Дрейк Ньюман (власник програми-конкурента)	Прямий конкурент	Низький	Низький	Він хоче зберегти своїх користувачів. Він може збільшити своє просування та покращити свій застосунок, що може погано вплинути на наш проєкт.

Після проведеної аналітики можна зрозуміти чиї вимоги мають більший вплив на проєкт, до кого потрібно звертатись, а кого просто інформувати. Дивлячись на проставлені оцінки впливу та важливості була розроблена «Power and Interest» матриця (рис. 2.1).

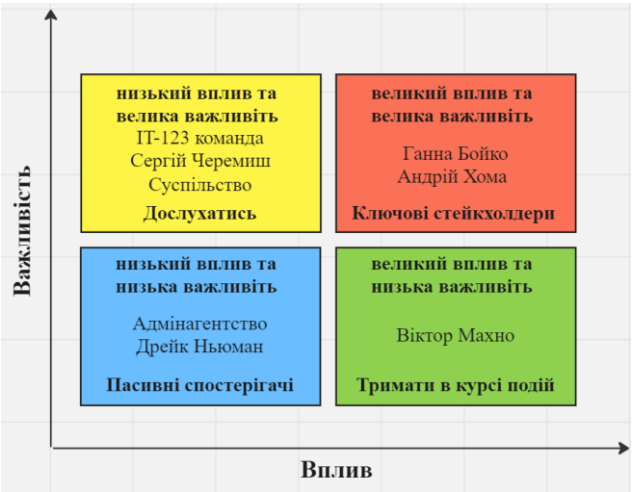

Рисунок 2.1 – «Power and Interest» матриця

Дивлячись на такий розподіл, можна зробити висновок, що в першу чергу потрібно звертатись до ключових стейкхолдерів (Ганна Бойко та Андрій Хома), а потім орієнтуватись на блок зацікавлених сторін під назвою «Дослухатись». Щоб на сплутати між собою відповідальних людей за різноманітні процеси розробки, була створена RACI матриця (табл. 2.2).

Таблиця 2.2 – RACI матриця

Роль / Діяльність	Сторона клієнта		Сторона компанії					
	Спонсор проєкту	Власник продукту	Бізнес-аналітик	Керівник проєкту	Команда розробників	Команда контролю якості	UI/UX команда дизайнерів	DevOps
Визначте бізнес-цілі	A	R	I	I				
Виявлення вимог		A	R	I				
Аналіз вимог і документація		A	R					
Планування та моніторинг проєкту	I	A	C	R	C	I	I	I
Дизайн інтерфейсу користувача		I	C	A	I	I	R	
Особливості розвитку			C	A	R	C	C	
Функції перевірки та перевірки			C	A	C	R	C	
Приймальне тестування функціональності		R	C	A	C	C	C	
Демонстрація функціональності		A	R	C	C	R		I
Розгортання випуску	I	I	C	A	I	I	I	R
Технічне обслуговування		I	I	A	I	I	I	R

Розібравшись із стейкхолдерами, було розроблено важливий артефакт для подальшої успішної комунікації між сторонами клієнта та розробників. RACI матриця – інструмент управління, який чітко визначає ролі учасників в проєкті. Де перші літери позначають наступне:

– **R**esponsible (Відповідальний за виконання);

– **A**ccountable (Відповідальний за результат);

– **C**onsulted (Підключений для консультацій);

– **I**nformed (Інформований про результати).

Отже, аналіз усіх зацікавлених сторін був проведений успішно. Виявили ключових гравців та розібрались хто за який процес відповідальний під час розробки продукту. Можемо рухатись до наступного етапу – збір вимог.

2.1.2 Початковий етап збору вимог

Початковий етап збору вимог – це період, коли проводиться систематичний аналіз і збір інформації про потреби та очікування зацікавлених сторін щодо проєкту. Основна мета цього етапу – зрозуміти, що очікується від системи чи продукту та які функції, характеристики та обмеження потрібні для задоволення цих вимог [24].

Існує купа цікавих методів виявлення вимог таких як: інтерв’ю, воркшопи, фокус-групи, проведення мозкового штурму, опитування, спостереження, аналіз документації та інше. Усі вони дієві з огляду на відповідну кількість залучених учасників. У нашому випадку орієнтир взято на ключових стейкхолдерів (Ганна Бойко та Андрій Хома), тому було обрано техніку фокус-групи – це метод дослідження, що використовується для збору думок, поглядів та переконань учасників щодо конкретної теми чи продукту. Зазвичай фокус-група складається з невеликої групи учасників, які обговорюють певний аспект, проблему чи ідею.

Після вирішення вибору підходу, була призначена зустріч для обгроворень. Під час дискусії, були отримані відповіді на вже підготовлені завчасно питання та нотувала усі озвучені вимоги – таким чином і з’явився новий артефакт (рис. 2.2-2.5) під назвою «Специфікації».

СПЕЦИФІКАЦІЇ

КОРИСТУВАЧІ

1. Типи користувачів:
   - гість;
   - користувач з обліковим записом;
   - користувач із преміум-акаунтом.
2. Гостьові користувачі можуть:
   - не мають персоналізовану аналітику;
   - мають рекламу;
   - мають дуже обмежений доступ – 20% від усієї бібліотеки пісень.
3. користувач з обліковим записом:
   - мають персоналізовану аналітику;
   - не мають реклами;
   - мають повний обсяг усієї бібліотеки пісень;
   - безкоштовне користування застосунком.
4. Критерії для преміальних клієнтів:
   - створення необмеженої кількості плейлистів;
   - у них буде досупний офлайн-режим;
   - відсутність реклами;
   - мають покращену аналітику;
   - вони підписуються за певну вартість;
   - вони зможуть підписатися на оновлення та отримувати сповіщення.

Рисунок 2.2 – Специфікації (користувачі)

ТЕХНІЧНІ ХАРАКТЕРИСТИКИ

1. Відсортований список пісень.
2. Створення геозалежних плейлистів.
3. Пошук за виконавцем, піснею, альбомом, жанром, текстом.
4. Статистична інформація про кожного виконавця та його треки.
5. По можливості текст під кожною піснею.
6. Режим відтворення пісні матиме всі основні функції.
7. Реєстрація:
    - Логін: телефон/електронна пошта
    - Пароль

Рисунок 2.3 – Специфікації (технічні характеристики)

ПЛАТФОРМИ

1. Основна платформа - смартфон:

    Операційна система - Android, IOS

УПРАВЛІННЯ ТА КОНТРОЛЬ

1. Спеціаліст з талантів несе відповідальність

для маніпулювання даними з музичним вмістом і затвердження облікового запису.

2. Адміністратор та IT команда будуть керувати та підтримувати всю систему.

КРАЇНИ ТА МОВИ

1. Країни:
    - США
    - Канада
    - Китай
    - можливо Мексика
2. Мови:
    - англійська
    - Іспанська
    - російський
    - французька
    - китайська

Рисунок 2.4 – Специфікації (платформи, управління та контроль, країни та мови)

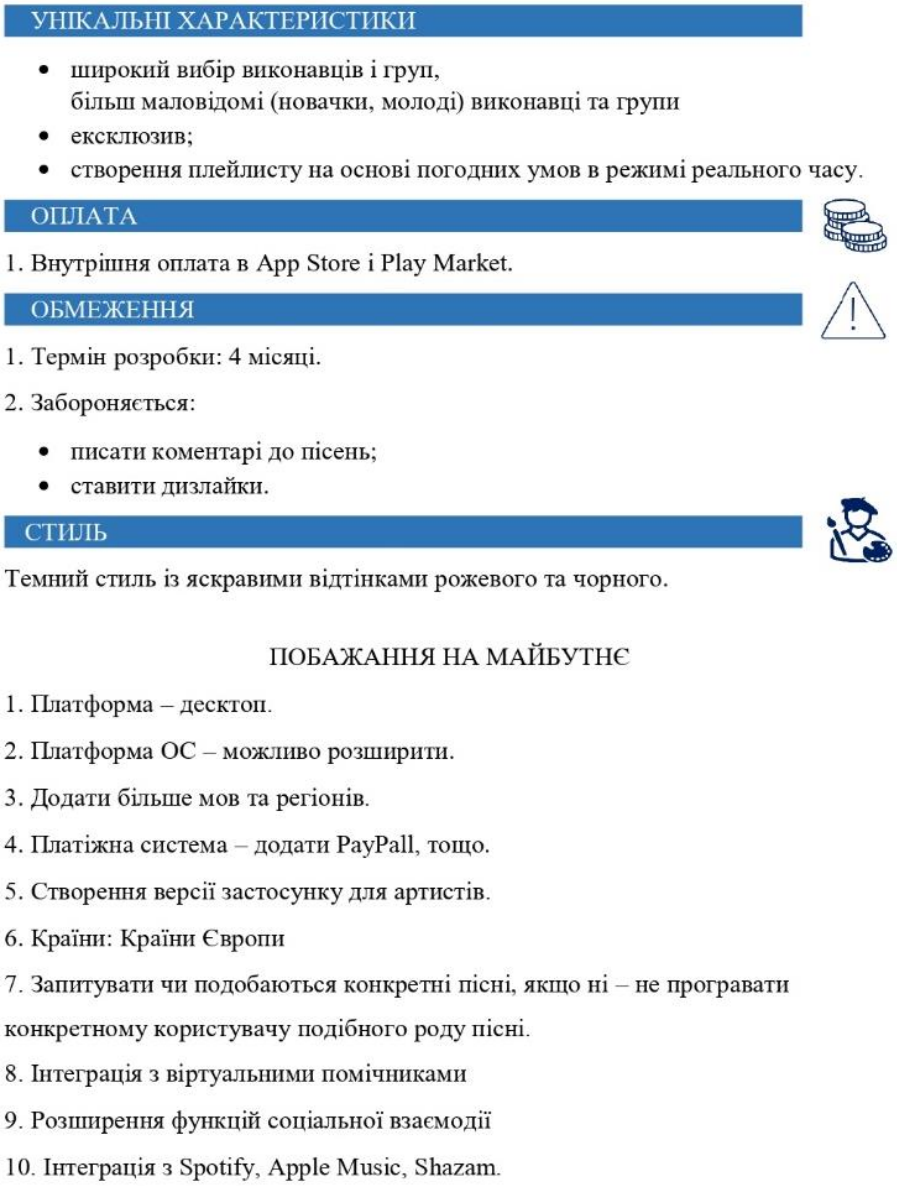

Рисунок 2.5 – Специфікації (унікальні характеристики, оплата, обмеження, стиль, побажання на майбутнє)

2.1.3 Опис ключових вимог та декомпозиція

Ключові вимоги – це основні функціональність та можливості, які повинен мати продукт. Їх було створено на основі занотованих даних у документі «Специфікації» після спілкування із зацікавленими сторонами. Це список основних завдань, які мають бути вирішені для створення успішного

продукту. Декомпозиція полягає в розбитті цих вимог на менші, більш конкретні завдання, що можуть бути розроблені окремо.

Отже на рисунках 2.6 та 2.7 представлено 10 найголовніших функцій музичного застосунку, що описані у вигляді користувацьких історій та розбиті згідно з ієрархією: зелене поле – це назва «Єпіку», червоне поле – назва «Фічі», жовте – опис користувацької історії та фіолетове – опис критеріїв прийняття.

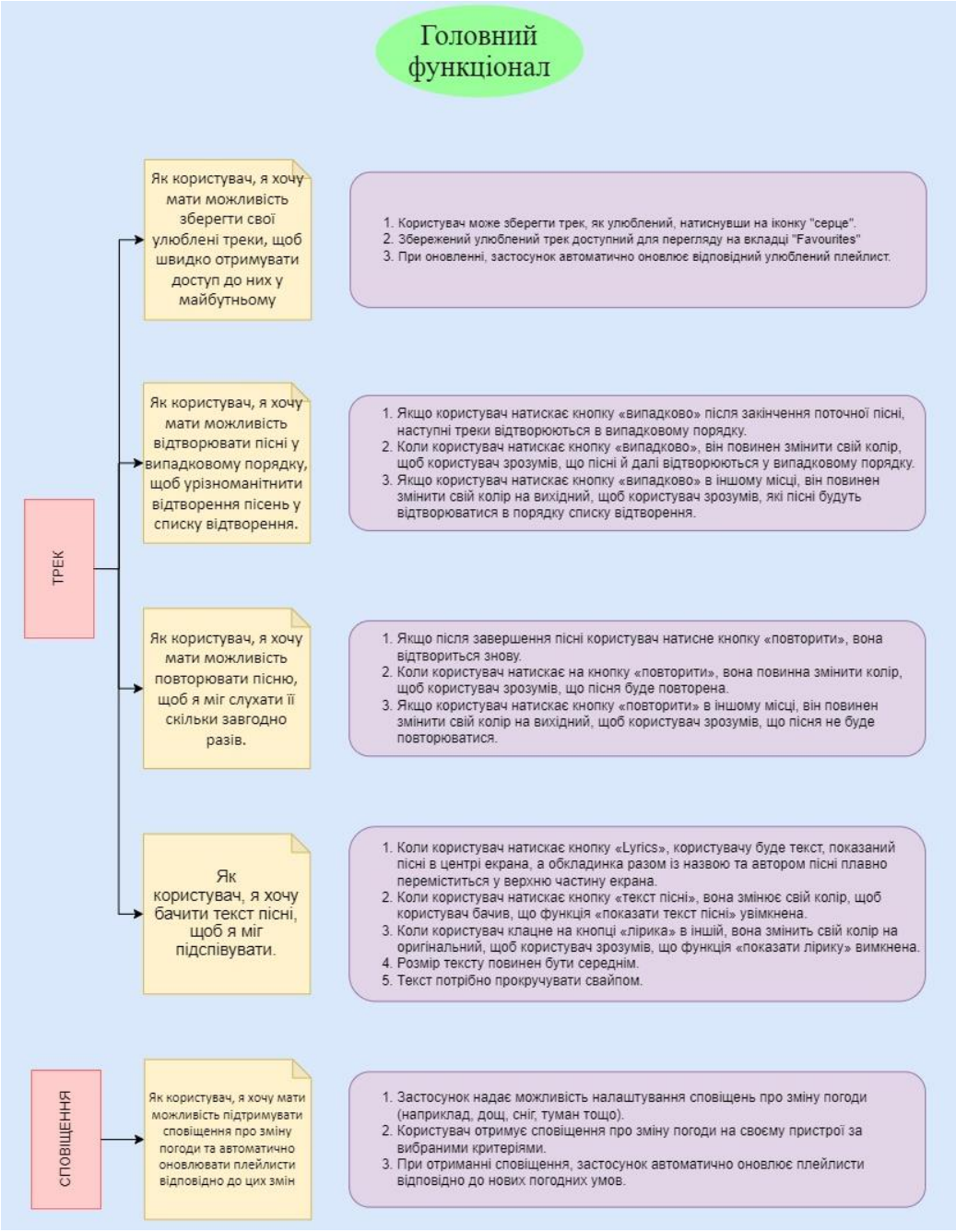

Рисунок 2.6 – Користувацькі історії (трек та сповіщення)

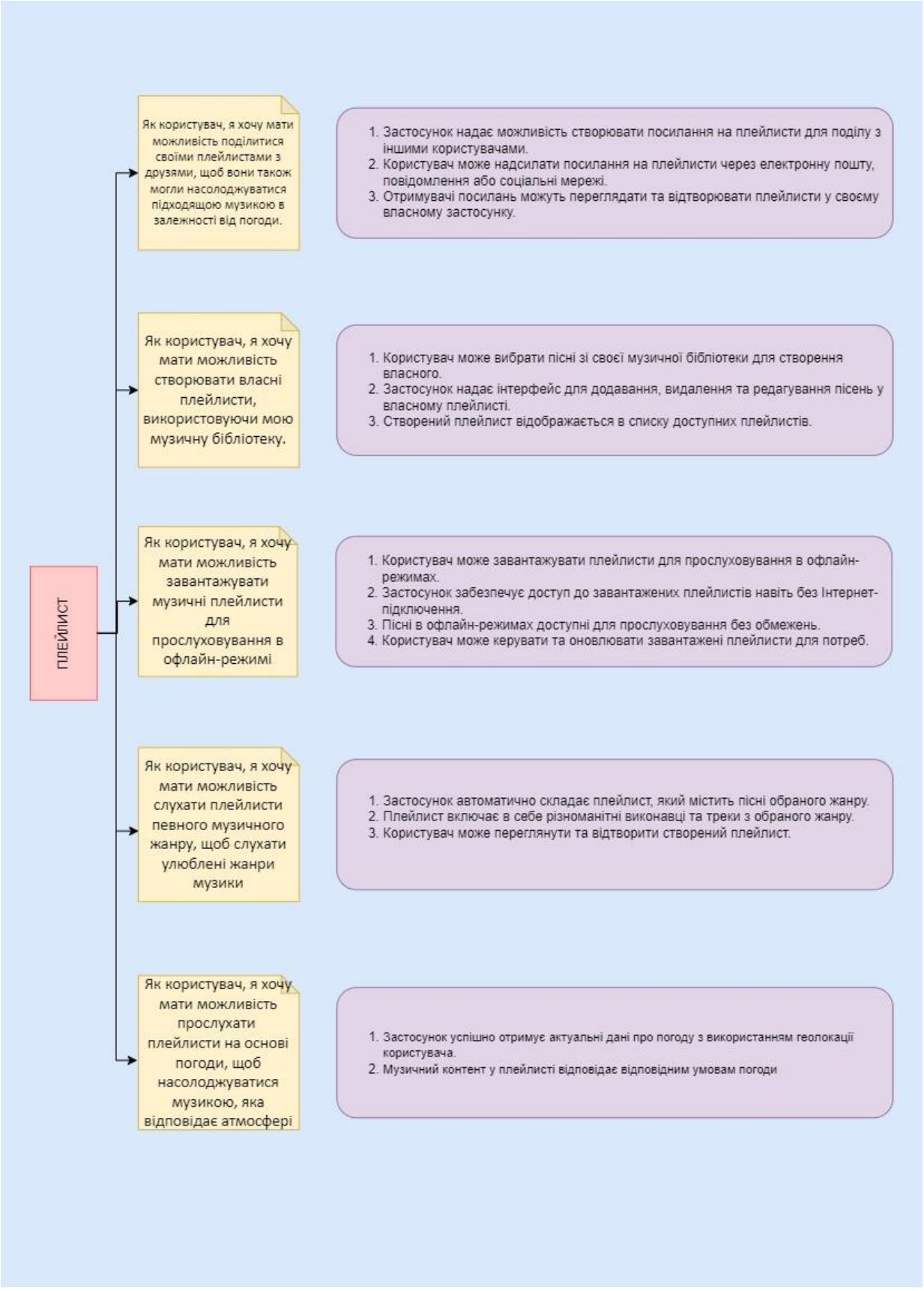

Рисунок 2.7 – Користувацькі історії (плейлист)

Для опису вимог, існує багато різних способів їх опису. Було прийнято рішення зупинилася на користувацьких історіях. Користувацькі історії – це короткі описи функціональностей продукту з точки зору користувача. Вони описують, що потрібно зробити, щоб задовольнити потреби або досягти цілей користувача. Використання користувацьких історій допомагає зосередитися на потребах користувачів та забезпечує більш зрозуміле спілкування між командою розробки.

Схема декомпозиції, яку було розроблено у Draw.io, допомагає візуалізувати структуру та ієрархію завдань, що потрібно виконати для реалізації кожної ключової вимоги. Це дає змогу краще організувати роботу з командою та розуміти взаємозв’язки між різними частинами продукту.

2.1.4 Пріоритизація

Пріоритизація – це процес визначення порядку важливості завдань або функцій для розробки на основі їхньої важливості та впливу на успіх продукту. Це допомагає команді розробки зосередитися на найбільш важливих аспектах продукту та витрачати ресурси ефективно. Цей процес є невід’ємною частиною розробки архітектури застосунку.

Існує багато різних технік пріоритизації, таких як Кано, 100$ або просто за допомогою присвоєння пріоритетів у вигляді високий, середній та низький. Вибір конкретної залежить від контексту проєкту, цілей, потреб і учасників. Однією з популярних технік є техніка ранжування [29].

Було використано метод ранжування для пріоритизації цих користувацьких історій. Варто зауважити, що можна використовувати шкалу від 1 до 10, де 1 – найнижчий пріоритет, а 10 – найвищий. Дамо кожній історії оцінку, використовуючи цю шкалу, а потім відсортуємо їх за зростанням оцінок (табл. 2.3).

Таблиця 2.3 – Пріоритизація (ранжування)

Користувацька історія	**Оцінка**
Як користувач, я хочу мати можливість завантажувати музичні плейлисти для прослуховування в офлайн-режимі.	1
Як користувач, я хочу мати можливість відтворювати пісні у випадковому порядку, щоб урізноманітнити відтворення пісень у списку відтворення.	2
Як користувач, я хочу мати можливість повторювати пісню, щоб я міг слухати її скільки завгодно разів.	3
Як користувач, я хочу бачити текст пісні, щоб я міг підспівувати.	4
Як користувач, я хочу мати можливість отримувати сповіщення про зміну погоди та автоматично оновлювати плейлисти відповідно до цих змін.	5
Як користувач, я хочу мати можливість поділитися своїми плейлистами з друзями, щоб вони також могли насолоджуватися підходящою музикою в залежності від погоди.	6
Як користувач, я хочу мати можливість зберігати свої улюблені плейлисти, щоб швидко отримувати доступ до них у майбутньому.	7
Як користувач, я хочу мати можливість створювати власні плейлисти, використовуючи мою музичну бібліотеку.	7
Як користувач, я хочу мати можливість створювати плейлисти за певним музичним жанром, щоб слухати улюблені жанри музики.	8
Як користувач, я хочу мати можливість автоматично створювати плейлисти на основі погоди, щоб насолоджуватися музикою, яка відповідає атмосфері.	9

2.2 UML діаграми

UML діаграми – це нотація, яка використовується для візуалізації, специфікації, конструювання та документування програмних систем. Вони дозволяють моделювати різні аспекти системи, такі як структура, поведінка, взаємодія та архітектура.

UML діаграми допомагають розробникам, аналітикам та іншим учасникам проєкту у візуалізації, розумінні та спілкуванні складних концепцій програмних систем. Вони служать засобом для формалізації та документування вимог, проєктування та аналізу систем, а також є важливим інструментом для спілкування між учасниками проєкту [25, 26].

2.2.1 Use case діаграма

Діаграма використання є ключовим інструментом у розробці програмного забезпечення, який допомагає визначити та візуалізувати функціональність системи з точки зору користувачів та інших систем. Цей тип UML діаграми відображає взаємодію акторів (користувачів або зовнішніх систем) з конкретними використанням (use cases) в рамках програми або системи.

Однією з основних переваг діаграми використання є її можливість сприяти уточненню вимог, розумінню функціональних потреб користувачів та узгодженню сподівань між усіма зацікавленими сторонами проєкту. Шляхом визначення та візуалізації конкретних використань, ця діаграма допомагає розробникам чітко уявити, які функції повинна виконувати програма або система.

Крім того, діаграма використання дозволяє визначити границі системи, показуючи, які зовнішні системи або користувачі взаємодіють з системою, а також які внутрішні використання входять до складу системи. Це допомагає

розробникам розуміти контекст, в якому функціонує система, та ефективно планувати її розвиток [27].

Одним із важливих використань діаграми використання є комунікація з різними зацікавленими сторонами проєкту. Вона дозволяє розробникам та зацікавленим сторонам взаємодіяти, виразно демонструючи їхні потреби та очікування.

Отже, на рисунку 2.8 зображено діаграму використання.

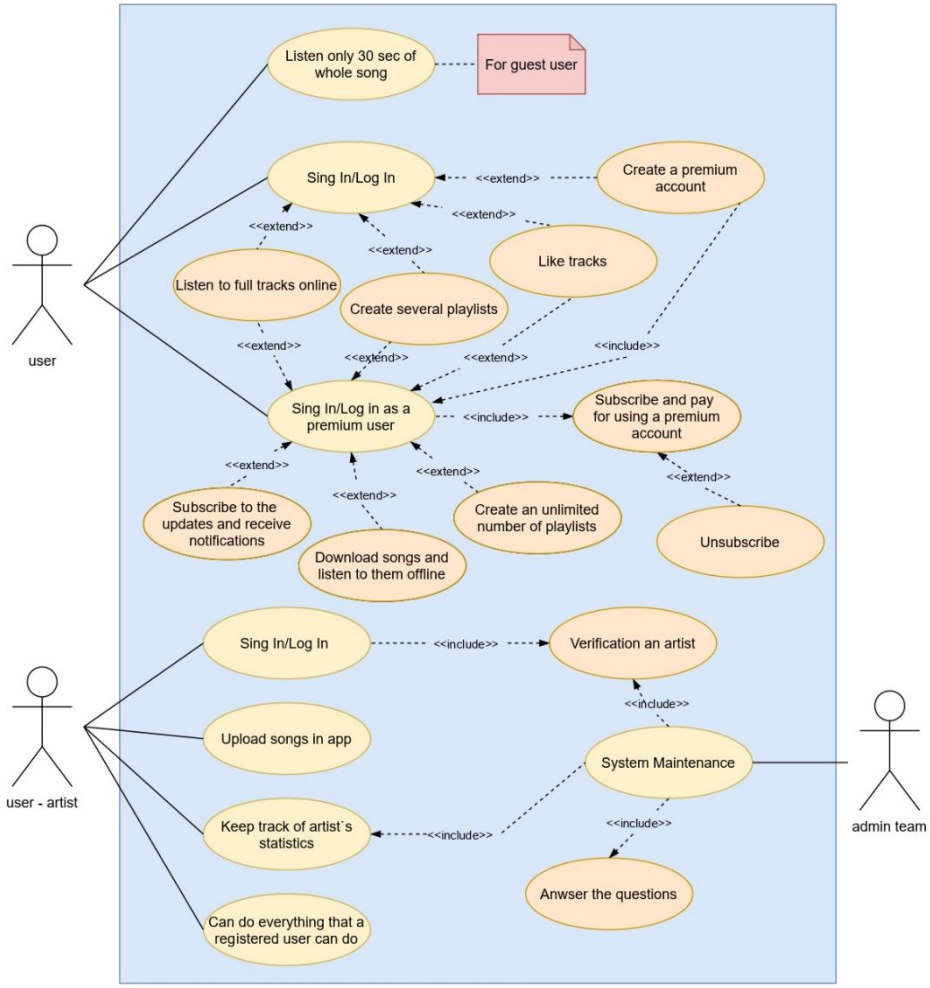

Рисунок 2.8 – Use case діаграма

На ній можна побачити такі ролі як:

– User: користувач застосунку який ділиться на зареєстрованого та гостя.

– User-artist: користувач, який є артистом та має свій професійний профіль у застосунку, для просування своєї музичної творчості.

– Admin team: команда IT-спеціалістів, яка займається загальною підтримкою застосунку, верифікацією артистів та вирішенням різноманітних користувацьких питань.

У діаграмі представлені усі головні та найважливіші кроки взаємодії акторів із музичним застосунком.

Загалом, діаграма використання є важливим інструментом у розробці програмного забезпечення, який сприяє уточненню вимог, плануванню подальшого розвитку системи та полегшенню спілкування між всіма учасниками проєкту.

2.2.2 Activity діаграма

Діаграма діяльності є одним із найважливіших інструментів у процесі розробки програмного забезпечення. Цей тип UML діаграм дозволяє візуалізувати послідовність дій та процесів, що відбуваються під час виконання певної діяльності. Використання Activity діаграм допомагає команді розробників та зацікавленим сторонам краще розуміти, моделювати та керувати різними процесами у розробці програмного забезпечення [28].

По-перше, Activity діаграми широко використовуються для моделювання бізнес-процесів. Вони надають зручний спосіб візуалізації послідовності дій та рішень у бізнес-сфері, що допомагає учасникам проєкту краще розуміти робочі процеси та їхні взаємозв’язки.

По-друге, Activity діаграми використовуються для моделювання алгоритмів. Вони дозволяють розробникам графічно відобразити послідовність операцій або кроків, необхідних для виконання конкретної функціональності програми.

Крім того, Activity діаграми допомагають у визначенні процесів розробки програмного забезпечення. Вони можуть бути використані для моделювання та візуалізації різних етапів розробки, таких як управління вимогами, проєктування архітектури, тестування та інші.

У представленій діаграмі діяльності, на рисунку 2.9, був описаний покроково процес прослуховування улюбленого музичного треку для зареєстрованого користувача.

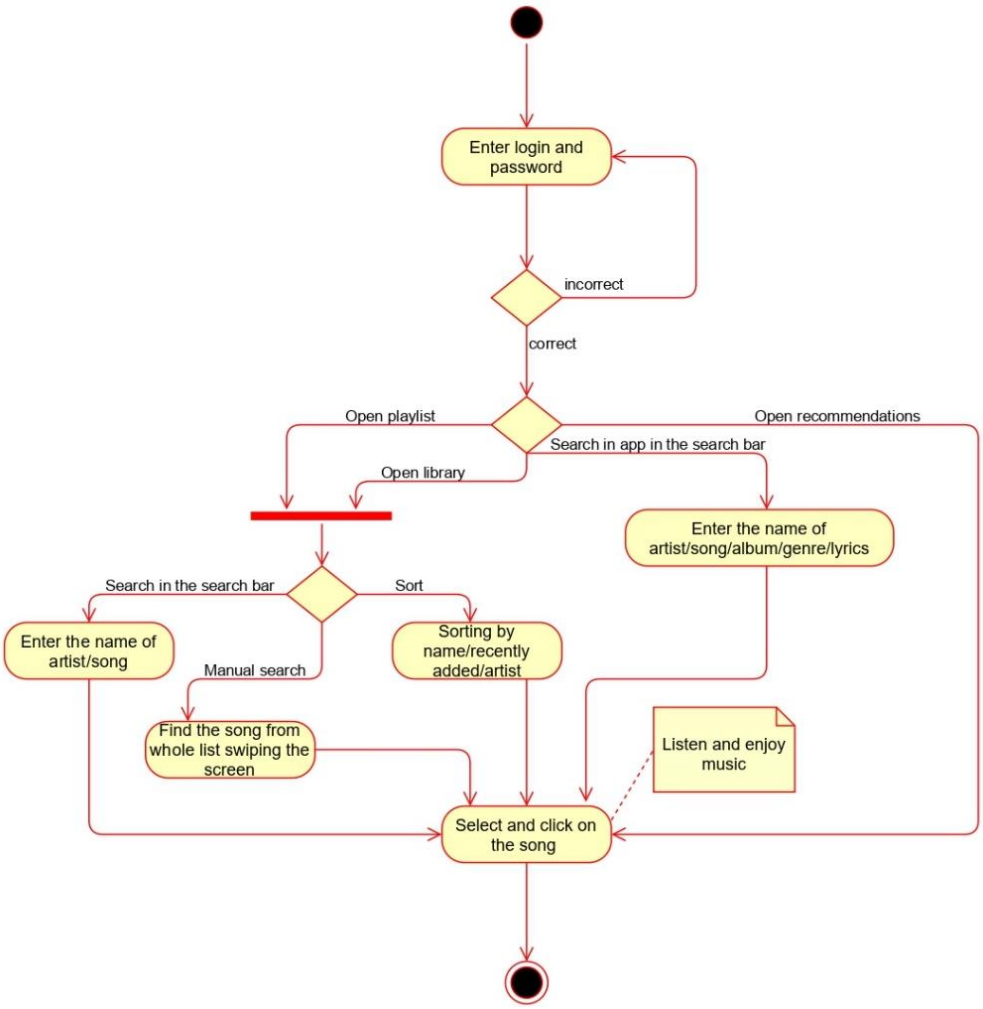

Рисунок 2.9 – Activity діаграма: Прослухати улюблений трек

Загалом, діаграми діяльності є потужним інструментом, який сприяє розумінню, моделюванню та керуванню процесами у розробці програмного забезпечення. Їх використання сприяє покращенню ефективності розробки та забезпечує краще управління проєктом.

2.2.3 Діаграма класів

У світі програмної розробки UML діаграма класів є важливим інструментом для моделювання структури програмного забезпечення. Вона дозволяє розробникам візуалізувати класи, їх властивості та зв’язки між ними.

Головна мета UML діаграми класів – це представлення абстракцій програмного забезпечення у вигляді класів та їхніх взаємозв’язків. Кожен клас представляє собою шаблон для об’єктів з певними характеристиками та методами. Зв’язки між класами вказують на взаємодію між ними, включаючи асоціації, композиції, агрегації та спадковість.

Використання UML діаграми класів включає наступні кроки та переваги:

Моделювання системи: Діаграма класів допомагає розробникам розуміти структуру системи шляхом візуалізації класів та їхніх взаємозв’язків. Це дає загальне уявлення про архітектуру програми.

Аналіз та проєктування: Розробники можуть використовувати діаграми класів для аналізу існуючої системи або для проєктування нової. Вони допомагають ідентифікувати класи, методи та взаємозв’язки, необхідні для виконання функцій системи.

Документування коду: UML діаграми класів можуть служити як документація для програмного забезпечення, де вони надають графічний огляд структури системи для інших розробників або для майбутніх редакцій.

Комунікація між учасниками проєкту: Використання діаграм класів спрощує комунікацію між членами команди розробки. Вони можуть використовувати діаграми для пояснення структури системи та взаємозв’язків між класами.

На рисунку 2.10 представлена діаграма класів проєкту музичного застосунку.

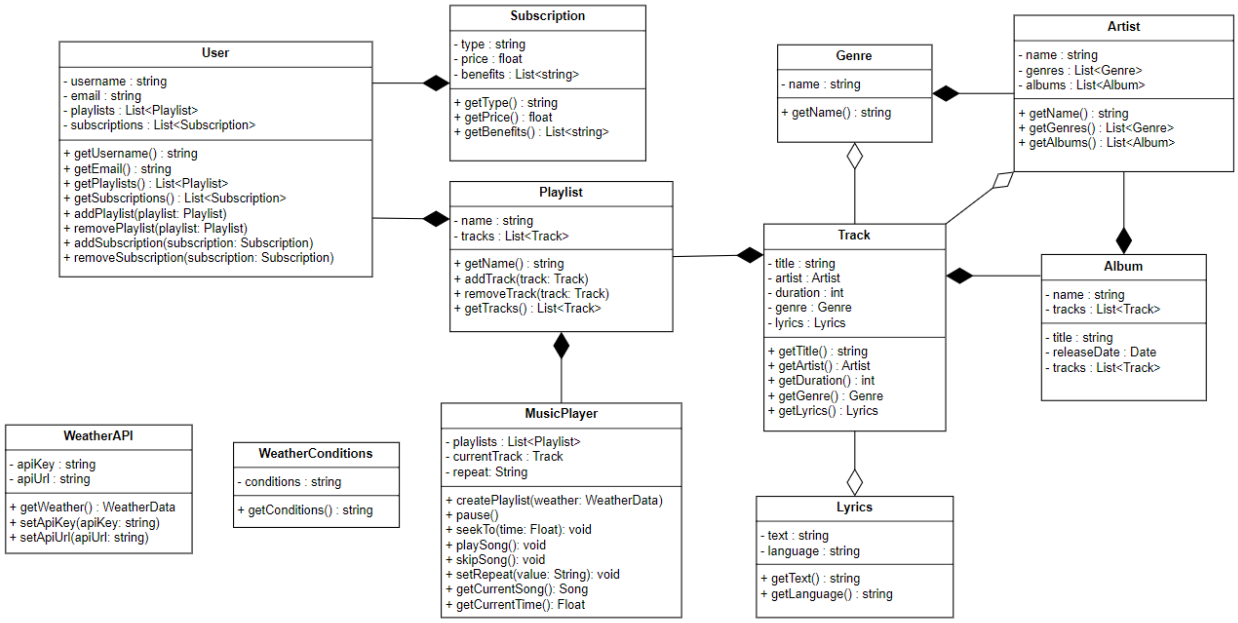

Рисунок 2.10 – UML Class Diagram

Узагальнюючи, UML діаграма класів – це потужний інструмент для моделювання та аналізу програмного забезпечення. Вона допомагає розробникам краще розуміти структуру системи, планувати розробку та підтримку програми, а також спрощує спілкування між учасниками проєкту.

2.3 Розробка ключової аналітики

Для створення плейлиста на основі погодних умов, використовуючи створені таблиці, необхідно виконати такі дії:

Крок 1. Отримати поточну геопозицію користувача та надіслати її на сервер.

Крок 2. Сервер робить запит у API погоди та отримує відповідь з даними про погоду, включаючи параметр «Weather Condition».

Крок 3. Використовуючи отримане значення «Weather Condition», алгоритм зіставляє отриману погодну умову з відповідними жанрами музики.

Крок 4. Через SQL запит створюється плейлист на основі певних жанрів музики та беручи в рахунок поточне значення «Weather Condition».

Для реалізації цього процесу необхідно використовувати SQL-запити для перевірки існування та додавання даних у відповідні таблиці бази даних, а також для вибірки та зв’язування даних для формування плейлиста.

2.3.1 Аналіз та створення зв’язку

Для того щоб створити ключовий зв’язок за допомогою якого буде створюватись плейлист на основі погодних умов, необхідно зрозуміти зв’язок між погодою та музикою. Плейлист містить у собі треки пов’язані одною темою. Трек має свої характеристики, серед яких є жанр, що може об’єднувати велику кількість треків за своїм настроєм. Щодо погодних умов, то залежно від стану погоди можна провести аналогію з супутнім настроєм. Отже, настрій це головний критерій, що відчувається у жанрі пісні та погодних умовах. Залишилось пов’язати між собою погодній умови з супутніми за настроєм жанрами музики.

Було обрано 20 ключових назв погодних умов та зіставила з 50 різноманітними жанрами музики на основі подібного настрою. Результати аналітики представлені у таблиці 2.4.

Таблиця 2.4 – Співставлення погоди до музики

№	Погодні умови	Жанр музики
1	Sunny	Pop, Reggae, Ska, Bossa Nova
2	Cloudy	Ambient, Shoegaze, Post-rock
3	Rainy	Blues, Jazz, Indie, Folk
4	Stormy	Metal, Hardcore, Grime, Dubstep
5	Snowy	Classical, Folk, Ambient, Post-rock
6	Foggy	Ambient, Jazz, Chillhop
7	Windy	Folk, Celtic, Ambient, Shoegaze

Продовження таблиці 2.4

№	Погодні умови	Жанр музики
8	Overcast	8Indie, Alternative, Post-rock, Emo
9	Hazy	Shoegaze, Psychedelic, Ambient
10	Humid	Reggae, Samba, Bossa Nova, Jazz
11	Dry	Desert Rock, Blues, Folk, Indie
12	Wet	Blues, Jazz, Reggae, Folk
13	Thunderstorms	Metal, Hardcore, Dubstep, Grime
14	Blustery	Folk, Celtic, Ambient, Post-rock
15	Showers	Jazz, Blues, Folk, Indie
16	Drizzle	Jazz, Blues, Ambient, Post-rock
17	Sleet	Classical, Ambient, Experimental
18	Freezing Rain	Classical, Ambient, Experimental
19	Misty	Ambient, Jazz, Blues, Folk
20	Tornadoes	Metal, Hardcore, Grime, Dubstep

2.3.2 Створення таблиць до бази даних

Насамперед, була створена база даних (проєкт) до якої було поступово додано таблиці та прописано між ними зв’язки.

SQL-запити CREATE TABLE використовуються для створення нових таблиць у базі даних. Коли ми створюємо таблицю, ми визначаємо її структуру, включаючи назви стовпців та їх типи даних. Це корисно для організації та збереження даних у базі даних. Кожна таблиця може мати свої власні стовпці та обмеження, такі як унікальність значень або зовнішні ключі для забезпечення цілісності даних [30].

Опис створення архітектури бази даних у вигляді SQL-запитів, що відображає структуру та зв’язки між датасетами для розробки музичного плейлисту на основі погоди та, безпосередньо, таблиць за допомогою SQL-запитів представлений на рисунках 2.11-2.18.

```
CREATE TABLE WeatherConditions (
    id INT PRIMARY KEY IDENTITY,
    title VARCHAR(255)
);
```

Рисунок 2.11 – SQL код для створення WeatherConditions (погодні умови)

```
CREATE TABLE Genre (
id INT PRIMARY KEY IDENTITY,
name VARCHAR(100)
);
```

Рисунок 2.12 – SQL код для створення Genre (жанри музики)

```
CREATE TABLE WeatherConditionstoGenre (
    genre_id INT,
    conditions_id INT,
    PRIMARY KEY (genre_id, conditions_id),
    FOREIGN KEY (genre_id) REFERENCES Genre(id),
    FOREIGN KEY (conditions_id) REFERENCES
      WeatherConditions(id)
    );
```

Рисунок 2.13 – SQL код для створення WeatherConditionstoGenre (таблиця для встановлення зв’язку між погодними умовами та жанрами музики)

```
CREATE TABLE Artist (
    id INT PRIMARY KEY IDENTITY,
    name VARCHAR(255),
    genre_id INT,
    FOREIGN KEY (genre_id) REFERENCES Genre(id)
);
```

Рисунок 2.14 – SQL код для створення Artist (виконавці)

```
CREATE TABLE Lyrics (
    id INT PRIMARY KEY IDENTITY,
    text TEXT,
    language VARCHAR(50)
);
```

Рисунок 2.15 – SQL код для створення Lyrics (тексту пісень)

```
CREATE TABLE Track (
    id INT PRIMARY KEY IDENTITY,
    title VARCHAR(255),
    artist_id INT,
    duration INT,
    genre_id INT,
    lyrics_id INT,
    FOREIGN KEY (artist_id) REFERENCES Artist(id),
    FOREIGN KEY (genre_id) REFERENCES Genre(id),
    FOREIGN KEY (lyrics_id) REFERENCES Lyrics(id)
);
```

Рисунок 2.16 – SQL код для створення Track (музичний трек)

```
CREATE TABLE Playlist (
    id INT PRIMARY KEY IDENTITY,
    name VARCHAR(255));

CREATE TABLE Track_Playlist (
    track_id INT,
    playlist_id INT,
    PRIMARY KEY (track_id, playlist_id),
    FOREIGN KEY (track_id) REFERENCES Track(id),
    FOREIGN KEY (playlist_id) REFERENCES Playlist(id));
```

Рисунок 2.17 – SQL код для створення Playlist та Track_Playlist

```
CREATE TABLE Album (
    id INT PRIMARY KEY IDENTITY,
    title VARCHAR(255),
    releaseDate DATE,
    artist_id INT,
    FOREIGN KEY (artist_id) REFERENCES Artist(id)
);
```

Рисунок 2.18 – SQL код для створення Album

Для заповнення таблиць даними було написано SQL запити з використанням INSERT INTO. На рисунках 2.19 та 2.20 представлені приклади заповнення даними таблиці Genre та WeatherConditions відповідно.

Таким самим чином були заповнені й інші таблиці. Дані для заповнення були взяті з вебсервісу Kaggle.

Рисунок 2.19 – SQL код для заповнення Genre

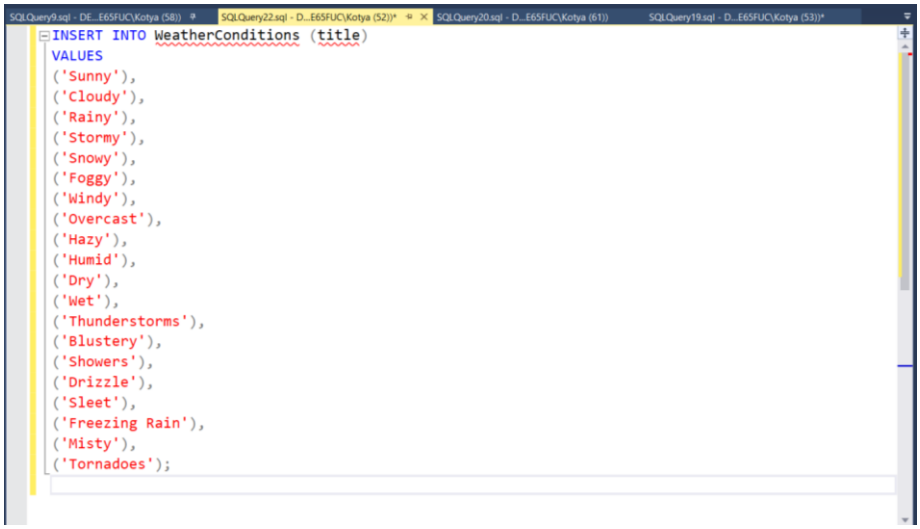

Рисунок 2.20 – SQL код для заповнення WeatherConditions

2.3.3 Діаграма бази даних

Створена база даних моделює музичний сервіс, де зберігаються інформація про жанри музики, виконавців, тексти пісень, треки, альбоми,

плейлисти та умови погоди, які впливають на вибір музики. На рисунку 2.21 представлена діаграма бази даних з провідними зв'язками між таблицями по зовнішнім ключам.

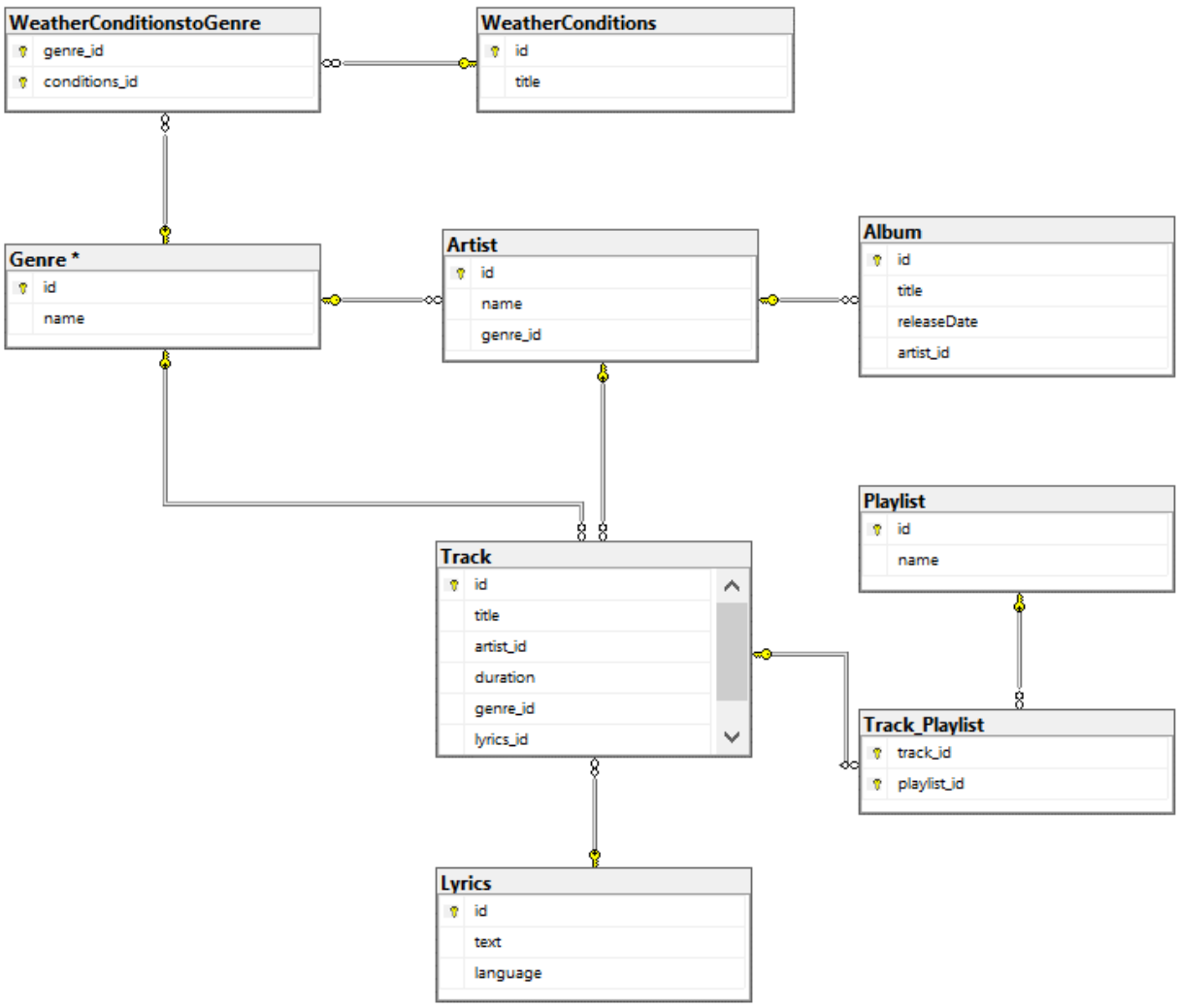

Рисунок 2.21 – Діаграма бази даних

Ця база даних дозволить зберігати і керувати різноманітною інформацією про музику, включаючи дані про виконавців, тексти пісень, треки та альбоми, а також дозволить створювати та управляти плейлистами. Крім того, за допомогою таблиці WeatherConditionstoGenre можна визначити відповідність між жанрами музики та певними погодними умовами і створити плейлисти за на основі поточної погоди по сумісному настрою та атмосфері.

2.3.4 Плейлист «Weather»

Для того щоб створити унікальний плейлист, необхідно написати SQL запит, що буде створювати його беручи поточні дані стану погоди. Ці дані передаються у таблицю WeatherConditions до поля title, та оновлюються кожну годину. На дану хвилину WeatherConditions.title тримає у собі значення «Sunny». Отже, на рисунку 2.22 можна побачити універсальний код для витягування даних з таблиць та створення актуального плейлисту «Weather» з відповідною за настроєм підбіркою треків.

```
DECLARE @WeatherConditionTitle VARCHAR(255) = 'sunny';

-- Insert the weather condition if not exists
IF NOT EXISTS (SELECT 1 FROM WeatherConditions WHERE title = @WeatherConditionTitle)
BEGIN
    INSERT INTO WeatherConditions (title) VALUES (@WeatherConditionTitle);
END

-- Get the ID of the weather condition
DECLARE @WeatherConditionID INT;
SELECT @WeatherConditionID = id FROM WeatherConditions WHERE title = @WeatherConditionTitle;

-- Create a playlist for the weather condition if not exists
DECLARE @PlaylistName VARCHAR(255) = 'weather playlist';

IF NOT EXISTS (SELECT 1 FROM Playlist WHERE name = @PlaylistName)
BEGIN
    INSERT INTO Playlist (name) VALUES (@PlaylistName);
END

-- Get the ID of the playlist
DECLARE @PlaylistID INT;
SELECT @PlaylistID = id FROM Playlist WHERE name = @PlaylistName;

-- Add tracks to the playlist based on genre and weather condition
INSERT INTO Track_Playlist (track_id, playlist_id)
SELECT DISTINCT t.id, @PlaylistID
FROM Track t
INNER JOIN Artist a ON t.artist_id = a.id
INNER JOIN Genre g ON t.genre_id = g.id
INNER JOIN WeatherConditionstoGenre wtg ON g.id = wtg.genre_id
INNER JOIN WeatherConditions wc ON wtg.conditions_id = wc.id
WHERE wc.id = @WeatherConditionID
AND g.name IN ('Pop', 'Reggae', 'Ska', 'Bossa Nova');
```

Рисунок 2.22 – SQL код для створення плейлісту «Weather»

Крок 1. Перевіряє, чи існує умова погоди «Sunny» в таблиці WeatherConditions. Якщо ні, він вставляє його.

Крок 2. Отримує ідентифікатор погодних умов «Sunny».

Крок 3. Перевіряє, чи існує список відтворення «Weather playlist». Якщо ні, він вставляє його.

Крок 4. Отримує ідентифікатор списку відтворення «Weather playlist».

Крок 5. Вставляє треки в таблицю Track_Playlist на основі вказаних жанрів («Pop», «Reggae», «Ska», «Bossa Nova») і погодних умов «Sunny».

3 ПРОТОТИПУВАННЯ ЗАСТОСУНКУ

3.1 Створення прототипів екранів для застосунку

Прототипування мобільних застосунків є надзвичайно важливим для визначення структури, функціональності та подальшої взаємодії користувача із застосунком. Мета прототипування полягає в тому, щоб візуалізувати ідеї та ідентифікувати можливі проблеми на ранніх стадіях розробки.

Прототипування може бути як низькорівневим, із використанням простих ескізів, так і високорівневим, із застосуванням інструментів, що дозволяють створити інтерактивний досвід, схожий на кінцевий продукт. Вибір інструментів для прототипування залежить від конкретних потреб проєкту, бюджету та команди. Серед популярних інструментів можна згадати Figma, Sketch, Adobe XD та InVision, які пропонують широкий спектр можливостей для створення та тестування прототипів.

У цьому розділі розглядаються основні компоненти, які складають прототипи застосунку «Melody», такі як головний екран, екрани входу, особистий профіль, екран налаштувань, та інше. Кожен з цих елементів потребує окремого розгляду, щоб забезпечити узгоджений користувацький досвід та відповідність принципам UX/UI. Прототипи також можуть включати різні сценарії використання, які відображають типові шляхи взаємодії користувача із застосунком, щоб переконатися в зручності та інтуїтивності інтерфейсу.

3.1.1 Екран авторизації/автентифікації

Екран входу для застосунку «Melody» починається з великого логотипа та назви застосунку, що розташовані вгорі, привертаючи увагу користувачів.

Для авторизації використовується звичайний класичний підхід, здійснений на багатьох стрімінгових платформах – вхід здійснюється за допомогою електронної адреси (або номеру телефону) та пароля користувача.

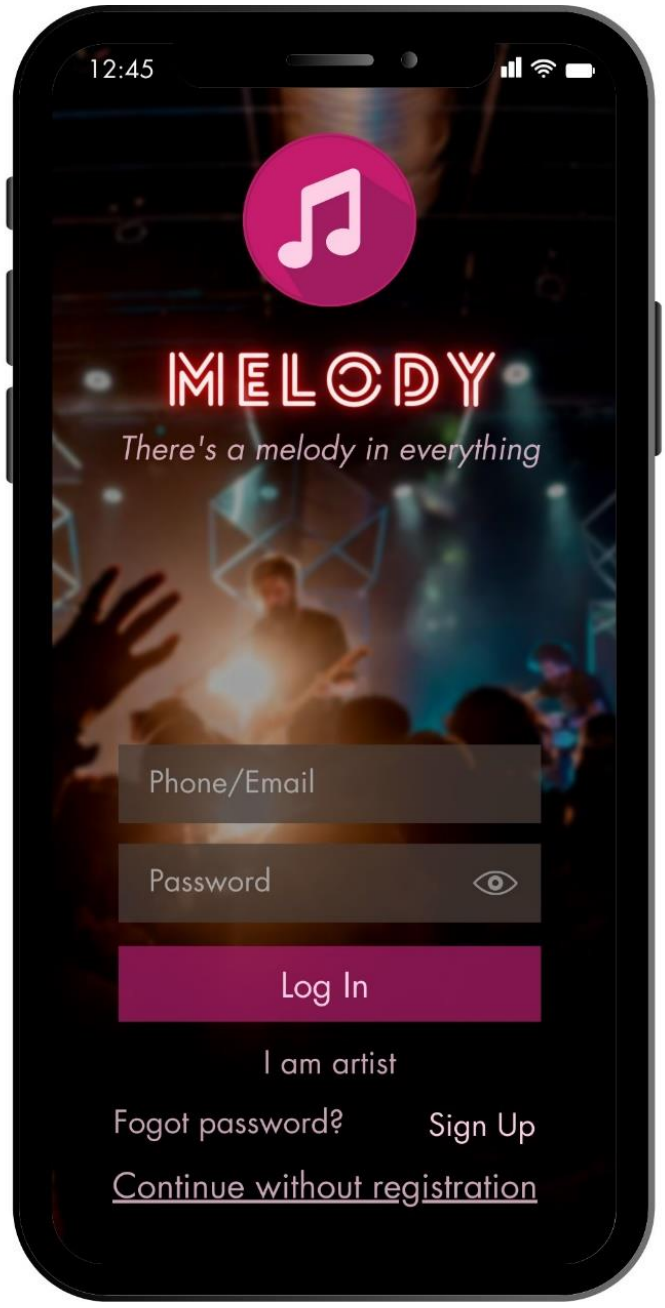

Рисунок 3.1 – Екран авторизації/автентифікації

З рисунку 3.1 видно пропоновані варіанти входу:

– звичайні користувачі можуть ввести свої дані, і, натиснувши кнопку «Log In», розпочати процес швидкої автентифікації;

– для авторів передбачено окремий вхід з аналогічними полями, але за окремою кнопкою «I am artist». Наразі ця кнопка не відпрацьовується, адже цей функціонал розрахований на перспективу у наступних релізах (post-mvp);

– для нових користувачів передбачена кнопка «Sign Up», що відкриває екран реєстрації для створення нового облікового запису;

– якщо користувачі хочуть випробувати застосунок без реєстрації, вони можуть натиснути кнопку «Continue without registration», що дозволить їм мати обмежений доступ до функцій застосунку, таких як перегляд погоди, створення базових плейлистів і прослуховування деяких треків. Але для збереження персональних налаштувань і плейлистів їм все ж таки доведеться авторизуватися.

Передбачено також наявність посилання «Forgot password?» для відновлення доступу для застосунку.

Для зручності користувачів застосунок підтримує біометричну автентифікацію для швидкого входу (вхід за сканером відбитку пальцю та/або за допомогою вбудованої в смартфони функцією Face ID).

3.1.2 Особистий профіль користувача

Особистий профіль користувача в застосунку «Melody» слугує центральним місцем, де користувачі можуть переглядати та керувати своїми персональними даними, налаштуваннями та вподобаннями. Крім цього, профіль автоматично визначає регіон користувача за допомогою геолокації, щоб забезпечити персоналізований досвід на основі місцезнаходження та надалі створити добірки відповідно до погодних умов.

Таким чином, особистий профіль дозволяє користувачам налаштовувати застосунок під свої потреби та уподобання, забезпечуючи персоналізований досвід використання. Екран особистого профілю користувача зображено на рисунку 3.2.

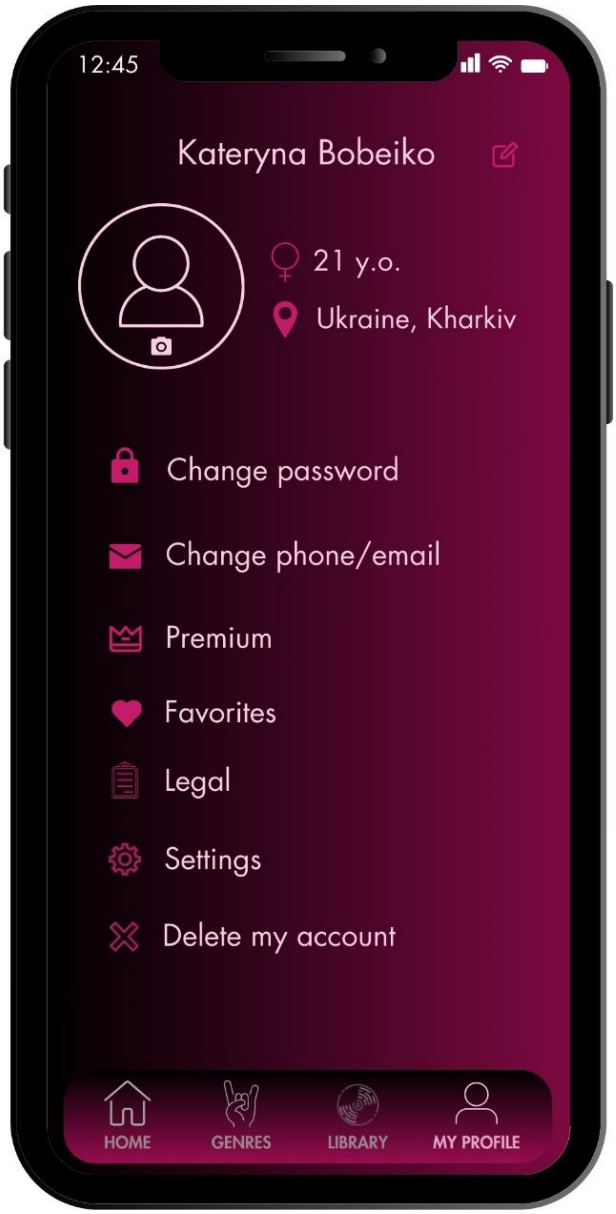

Рисунок 3.2 – Екран особистого профілю користувача

На цьому екрані користувачеві доступні такі функції та кнопки:

– редагування загальної інформації: ім’я та прізвище користувача, вік, стать, фото профілю (опціонально);

– автоматичне визначення регіону користувача через геолокацію (для відображення місцевої погоди та пропозицій музики);

– зміна контактних даних та паролю;

– перехід на Premium версію застосунку;

– налаштування музичних вподобань, перегляд змісту та редагування папки «Favorites»;

– загальні налаштування профілю, такі як вибір мови застосунку, налаштування сповіщень та повідомлень, інше;

– видалення та призупинення власного акаунту.

3.1.3 Музичний плеєр

Ця сторінка – основа будь-якого музичного застосунку, це саме той інтерфейс, де користувачі можуть керувати відтворенням музики та взаємодіяти з треками. Вона має бути інтуїтивно зрозумілою, з дизайном, який дозволяє користувачам легко виконувати основні операції, такі як відтворення, призупинення, пропуск, додавання в плейлист і багато іншого.

У центрі екрана знаходиться обкладинка альбому або зображення треку, що наразі відтворюється. Це велике зображення є основним візуальним елементом сторінки, навколо якого розташовані інші елементи управління. Під ним вказано назву треку та ім’я виконавця, а також панель управління, що складається з наступних елементів:

– кнопка відтворення/призупинення треку;

– перемикачі на наступний та попередній трек;

– кнопка для активації/деактивації режиму «Випадкове відтворення»;

– кнопка для повторного відтворення треку або плейлиста;

– додавання треку до папки з улюбленими треками;

– повзунок – шкала прогресу, яка показує поточне місцезнаходження в треку з можливістю перетягувати повзунок для переміщення в межах треку або переходу до певного моменту пісні;

– кнопка для виходу в плейлист;

– кнопка для виходу з програвача на попередньо відкриту користувачем сторінку (зазвичай знаходиться в лівій верхній частині екрану програвача);

– інтерактивний блок з текстом пісні.

Всі ці функції були додані до плеєра застосунку «Melody», його екран можна побачити на рисунку 3.3.

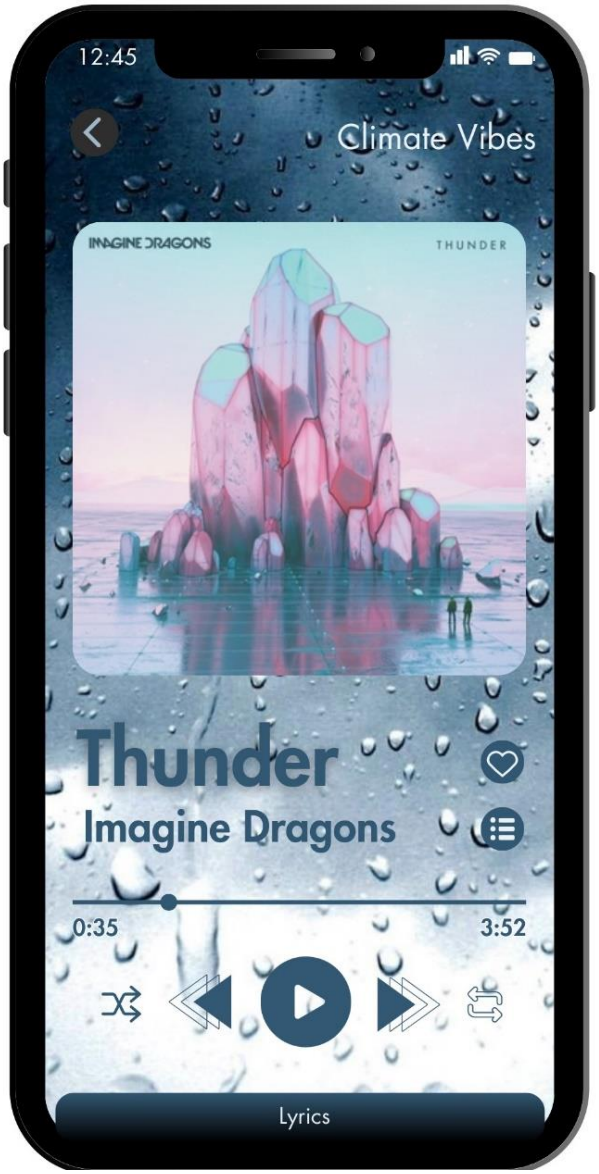

Рисунок 3.3 – Екран музичного плеєру

Інтерактивні елементи програвача оптимізовані для зручного використання на мобільних пристроях, з урахуванням сенсорного керування та великих екранів. Таким чином, сторінка музичного програвача в «Melody» пропонує зручний і зрозумілий інтерфейс, який дозволяє користувачам насолоджуватися музикою та легко керувати відтворенням треків.

3.1.4 Головний екран застосунку

Головний екран є пунктом, де користувачі можуть отримати доступ до різноманітного музичного контенту та функцій. Екран має кілька секцій, які забезпечують швидкий доступ до рекомендацій та пошуку музики (рис. 3.4).

Рисунок 3.4 – Головний екран застосунку

Рекомендовані добірки можна розділити за наступними категоріями:

– ряд секцій, які містять добірки музики, рекомендовані на основі музичних уподобань користувача, улюблених жанрів та виконавців;

– добірки можуть бути тематичними (наприклад, «Нова Поп-музика», «Релакс для вечора») або засновані на активності користувача («Треки, які ви слухали вчора»);

– секція, яка генерує плейлист на основі поточних погодних умов у регіоні користувача. Музика в цій секції змінюється згідно з погодою, створюючи відповідний настрій (наприклад, енергійні пісні для сонячних днів або спокійні композиції для дощової погоди).

Поле для пошуку знаходиться у верхній частині екрану та дозволяє шукати пісні за назвою, виконавцями або навіть частиною тексту. Під час введення тексту відображаються підказки з пропозиціями, а після натискання кнопки «Пошук» відображаються результати, які можна сортувати за різними критеріями (назва, виконавець, жанр).

3.1.4 Бібліотека користувача

Бібліотека в застосунку «Melody» є місцем, де користувачі можуть організовувати, зберігати та отримувати доступ до різноманітного музичного контенту. Цей екран складається з кількох розділів, які надають користувачам можливість керувати своїми плейлистами, переглядати історію відтворень, переглядати підписки, а також зберігати вподобані та завантажені треки. Зверху цього екрана є посилання на добірку музики згідно до погодних умов, що автоматично оновлюється щоразу, як змінюється погода в регіоні користувача. У цьому блоку з погодним плейлистом під назвою «Climate Vibes» можна побачити коли останній раз було оновлено погодні дані. Також на ньому присутній віджет з отриманими даними, що демонстує реальний стан погоди. Загальний вигляд цього екрану можна побачити нижче на рисунку 3.5.

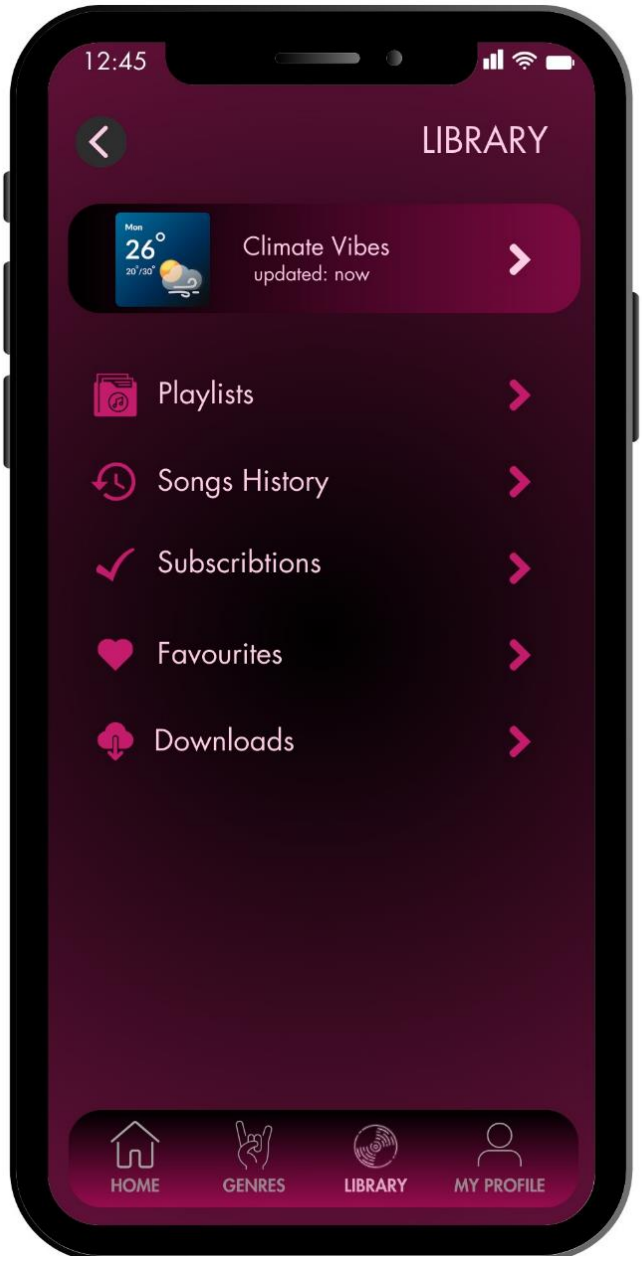

Рисунок 3.5 – Бібліотека користувача

Розглянемо детальніше деякі основні розділи бібліотеки.

Розділ «Playlists» (рис. 3.6) відображає всі плейлисти, які користувач колись створив або зберіг до своєї медіатеки та дає користувачам можливість організовувати та відтворювати власні добірки музичних треків. До власних плейлистів користувачі можуть додавати треки з різних частин застосунку, включаючи рекомендовану музику, вподобані треки, історію відтворень та пошук. Окрім створення та редагування, застосунком передбачено також видалення плейлистів за вимогою користувача.

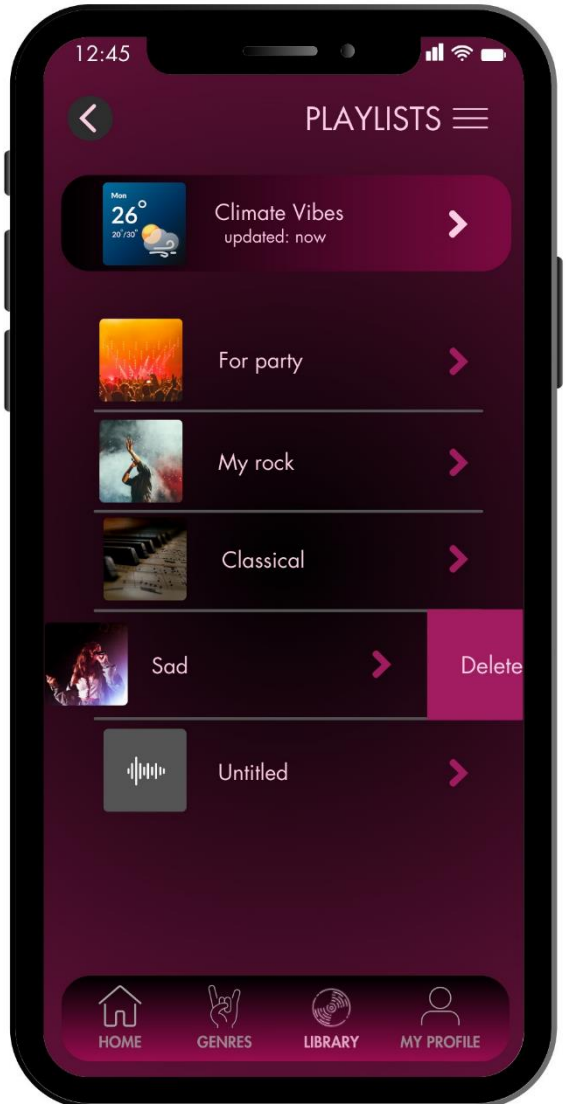

Рисунок 3.6 – Плейлисти

У розділі «Songs History» реалізована можливість перегляду користувачем списку треків, прослуханих їм раніше. Ця функція дуже зручна та корисна, особливо для випадків, коли користувач випадково втратив поточний плейлист або забув назву треку. Це забезпечує додатковий рівень захисту від втрати музичного контенту.

На сторінці «Subscriptions» користувачі можуть керувати підписками на улюблених виконавців, а також мають можливість переглядати свої архівовані підписки (ті, які користувач в якийсь момент часу призупинив) та будь-коли відновлювати їх за бажанням. Вигляд екрану «Subscriptions» зображено на рисунку 3.7.

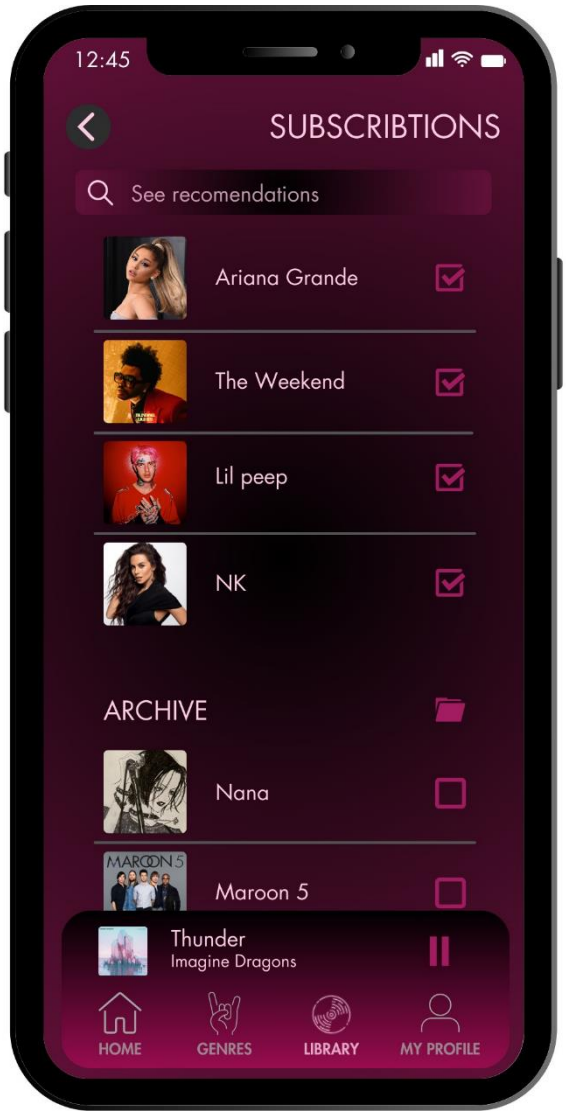

Рисунок 3.7 – Підписки на виконавців

У першій частині екрану розташовані поточні підписки користувача на виконавців. Тут можна переглядати нові релізи, альбоми, сінгли та інші оновлення від виконавців, які вони публікують. Користувач може легко відписатися від будь-якого виконавця зі списку поточних підписок. Для цього потрібно лише прибрати відмітку біля імені виконавця. Відписані виконавці не видаляються повністю, а переміщуються в розділ архівованих підписок, що дозволяє в будь-який момент відновити підписку.

У другій половині екрану знаходиться архів виконавців. Тут можна переглядати їхню активність, як і у поточних підписках, але без автоматичного отримання сповіщень про нові релізи. Користувач може

відновити підписку на будь-якого виконавця з цього розділу, повернувши його до поточних підписок. Така функція дозволяє користувачам експериментувати з підписками без страху втратити доступ до музики або артистів, які їм раніше подобалися.

Розділ «Favourites» призначений для збереження швидкого доступу та зручного відтворення треків, які користувач позначив як улюблені або вподобані. Додавати аудіо-доріжки до цього розділу можна з будь-якого місця в застосунку. Ось деякі корисні функції, що можуть бути реалізовані в цьому розділі:

– одним натисканням кнопки «Відтворити все» можна прослухати всі вподобані треки або відтворювати їх у випадковому порядку;

– доступна функція створення плейлістів на основі вподобаних треків або додавання їх до існуючих плейлістів;

– підтримується сортування за різними критеріями, такими як назва, виконавець, дата додавання або тривалість;

– доступна функція створення плейлістів на основі вподобаних треків або додавання їх до існуючих плейлістів;

– можливість пошуку за назвою треку, ім’ям виконавця або назвою альбому, що дозволяє швидко знаходити потрібний контент серед вподобаних;

– підтримується сортування за різними критеріями, такими як назва, виконавець, дата додавання або тривалість;

– передбачена функція завантаження вподобаних треків на пристрій для офлайн-програвання;

– видалення треків, якщо вони більше не відповідають музичним смакам, або якщо потрібно звільнити місце;

– спільний доступ до вподобаних треків: користувачі можуть ділитися улюбленими піснями з друзями чи родиною, відправляючи посилання на конкретний трек або добірку через соціальні мережі чи месенджери.

В розділі «Downloads» зберігаються треки, що були попередньо завантажені користувачем для офлайн-прослуховування.

На сторінці «Climat Vibes» можна знайти автоматично сформований застосунком плейліст на основі поточних погодних умов у вказаному користувачем регіоні (рис. 3.8).

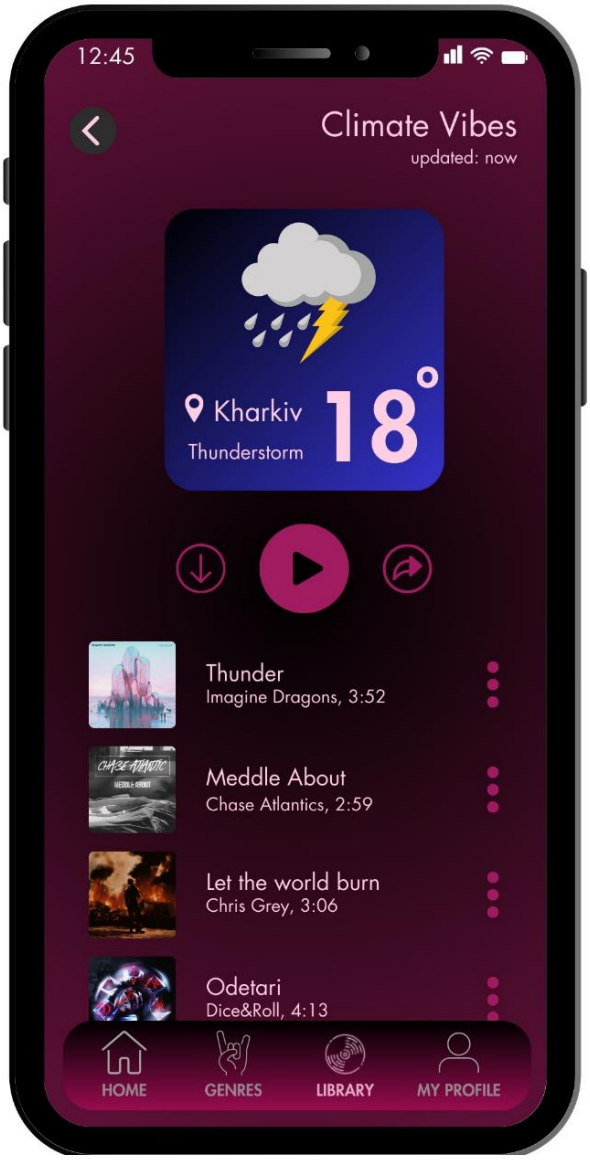

Рисунок 3.8 – Плейлист, заснований на погодних умовах

Цей розділ має кілька ключових компонентів і функцій, які забезпечують персоналізований музичний досвід, відповідний атмосфері за вікном. По-перше, на сторінці відображається інформація про поточну

погоду, включаючи температуру, погодний стан (сонячно, хмарно, дощ, сніг, тощо), іконки, які візуально передають атмосферу, і, можливо, навіть короткий прогноз на найближчі години або дні. Ця інформація знаходиться у верхній частині сторінки, допомагаючи користувачам орієнтуватися у контексті.

Нижче знаходиться плейлист, який автоматично створюється відповідно до поточної погоди. Він містить треки, підібрані для різних погодних умов, з урахуванням темпоритму, настрою та жанрів. Наприклад, для сонячних днів це можуть бути енергійні та яскраві пісні, для дощової погоди – спокійніші та мелодійні треки, а для сніжної – зимові або святкові мелодії. Функції взаємодії з цим плейлистом повністю збігаються з тими, що пропонуються користувачеві в «Favourites».

Сторінка також має опцію оновлення, щоб отримувати свіжі плейлисти, коли змінюється погода або коли користувач бажає нових рекомендацій. Це забезпечує постійний потік свіжої музики, яка відповідає поточним атмосферним умовам, і робить сторінку «Climat Vibes» динамічною та цікавою для користувачів.

3.2 Алгоритм підбору треків відповідно до погодних умов

Алгоритм підбору пісень згідно з кліматом має враховувати кілька основних параметрів, зокрема, поточні погодні умови, час доби, сезон і музичні вподобання користувачів.

В рамках цього застосунку для формування плейлиста «Climat Vibes», алгоритм складатиметься з 7 основних кроків та дозволяє поєднувати погодні умови з музичними вподобаннями користувачів, забезпечуючи унікальний і персоналізований досвід. Більш наглядно з алгоритмом підбору треків можна ознайомитись на рисунку 3.9.

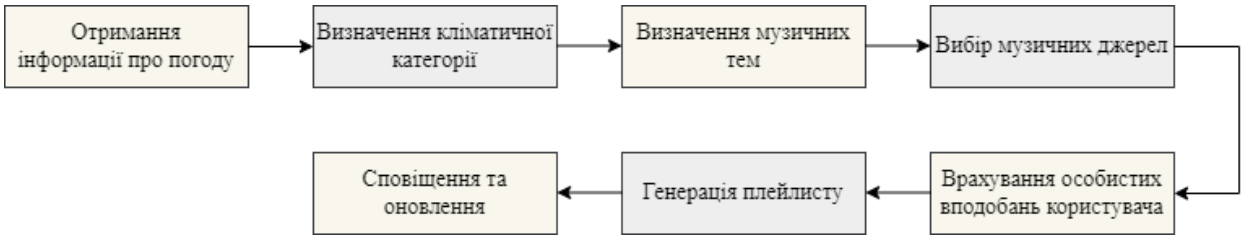

Рисунок 3.9 – Алгоритм підбору треків відповідно до погодних умов

Крок 1. Отримання інформації про погоду. Використовується API сервісу прогнозу погоди (наприклад, з вбудованих в смартфон сервісів Apple Weather або Google Weather) для отримання інформації про поточні погодні умови в регіоні користувача. Дані включають температуру, хмарність, вологість, швидкість вітру, погодний стан (сонячно, дощ, сніг, тощо), а також час сходу та заходу сонця.

Крок 2. На основі отриманих даних про погоду алгоритм визначає кліматичну категорію.

Крок 3. Визначення музичних тем. Для кожної кліматичної категорії визначаються основні музичні теми. Наприклад, для сонячної погоди – енергійна та оптимістична музика, для дощової – спокійніша та мелодійніша, а для снігу – атмосферна або святкова. Ці музичні теми можуть бути пов’язані з певними жанрами, темпом і настроєм пісень.

Крок 4. Вибір музичних джерел. Алгоритм використовує музичну базу даних застосунку для отримання доступу до великої кількості пісень та плейлистів. Пісні відбираються на основі категорій, музичних тем та жанрів, які відповідають кожній кліматичній категорії.

Крок 5. Врахування особистих вподобань кожного користувача. Для персоналізації пропозиції алгоритм враховує вподобані треки користувача, його історію відтворень та підписки, що допомагає створити плейлисти, які не тільки відповідають клімату, але й резонують із музичним смаком слухача.

Крок 6. Генерація плейлисту. Алгоритм на основі кліматичної категорії, музичних тем і вподобань користувача генерує плейлист. Плейлист

є динамічним і оновлюється в режимі реального часу, коли змінюються погодні умови.

Крок 7. Сповіщення та оновлення. Застосунок надсилає сповіщення користувачам про нові плейлисти, коли змінюється погода або коли будуть доступні нові пісні, що відповідають поточному клімату.

3.3 Технічні вимоги до реалізації застосунку

Наведена нижче таблиця 3.1 відображає основні компоненти застосунку «Melody», а також технології та інструменти, які будуть використані для їх розробки.

Таблиця 3.1 – Перелік інструментів та технологій

Компонент	**Технологія**	**Інструменти**
Front-end	Flutter	Dart, Android Studio, Visual Studio Code, Xcode
Back-end	Node.js (Express або NestJS)	Node.js, npm/yarn, Swagger/OpenAPI
База даних	MS SQL	Microsoft SQL Server Management Studio 18
Хмарні сервіси	AudioBox	AudioBox API
Погодні служби	OpenWeatherMap або WeatherStack	API ключі, Axios (бібліотека HTTP)
Авторизація/ автентифікація	JWT	jsonwebtoken (JWT), bcrypt (хешування паролів)
CD/CI	GitHub Actions або GitLab CI/CD	GitHub/GitLab, Docker, Docker Hub, Jenkins
Хостинг	AWS, GCP, Azure, або Heroku	Heroku CLI, AWS CLI, Google Cloud CLI, Azure CLI

Зазвичай, вибір технічних засобів для реалізації застосунку залежить здебільшого від типу пристроїв та платформ, на який розрахована пропозиція. «Melody» буде реалізований на основі крос-платформенного фреймворка Flutter, що дозволяє одночасно розробляти для обох основних мобільних платформ: Android та iOS. Такий підхід забезпечує високу ефективність розробки, зменшує витрати часу та ресурсів на створення та підтримку застосунку на різних платформах. В основі застосунку лежатиме архітектура з розділенням фронтенду та бекенду, що гарантує гнучкість, масштабованість і легкість підтримки.

3.4 Перспективи подальшої роботи

У розвитку музичних застосунків надзвичайно важливо постійно вдосконалюватися та розширювати функціонал для задоволення потреб користувачів. Інтеграція з іншими популярними платформами – ключовий аспект успіху будь-якого музичного застосунку. В рамках подальшого розвитку застосунок «Melody» може розширювати свої можливості, за рахунок інтеграції з музичними сервісами. Крім цього, розглядаються інші напрямки розвитку, такі як підвищення соціальної взаємодії, інтеграція з віртуальними помічниками та інші технології, що підвищують цінність для користувачів. Також, у наступних версіях заплановано реалізувати іншу сторону додатка як для артиста, що буде мати можливість створення професійного профілю для публікації своїх музичних творінь. Нижче наведено таблицю 3.2, яка ілюструє основні напрямки розвитку застосунку, їхній опис та можливі вигоди, що виникають від цих інтеграцій. Це допоможе застосунку залишатися актуальним, привабливим та конкурентоспроможним на ринку музичних застосунків.

Таблиця 3.2 – Перспективи подальшої роботи

Перспективи подальшої роботи	**Опис**	**Можливі вигоди від імплементації**
1	**2**	**3**
Інтеграція зі Spotify	Колаборація із Spotify дозволить користувачам відтворювати пісні з цієї платформи безпосередньо в застосунку та синхронізувати з ним вподобані треки та створені плейлісти.	Користувачі отримають доступ до великої бібліотеки музики, зможуть створювати плейлісти та використовувати функції застосунку з існуючим акаунтом Spotify. Це підвищить цінність застосунку для тих, хто вже користується Spotify.
Інтеграція із Apple Music	Аналогічно до Spotify, користувачі зможуть використовувати синхронізаційні функції.	Це також розширить аудиторію застосунку, дозволивши користувачам Apple Music використовувати його можливості.
Інтеграція з Shazam	Додавання підтримки Shazam дозволить користувачам швидко ідентифікувати пісні та додавати їх до плейлістів або вподобаних треків.	Інтеграція з Shazam надасть користувачам зручний спосіб знайти нову музику та розширити свої плейлісти, тим самим підвищивши прослуховування в застосунку.
Розширення функцій соціальної взаємодії	Взаємодіяти із застосунком та музикою в соціальних мережах.	Це сприятиме збільшенню залученості користувачів і підвищить активність використання застосунком. Соціальні функції можуть привабити нових користувачів і зміцнити спільноту.
Інтеграція з віртуальними помічниками	Застосунок може бути інтегрований з віртуальними помічниками, такими як Siri, Google Assistant або Alexa, для голосового керування музикою.	Інтеграція з віртуальними помічниками додасть зручності у використанні застосунку, щоб користувачі керували музикою голосом. Це може залучити більше користувачів, які надають перевагу hands-free.

Продовження таблиці 3.2

1	2	3
Створення версії застосунку для артистів	Можливість входу до застосунку в якості артиста на професійний профіль для публікацій своєї творчості та перегляду аналітики.	Таке поліпшення призведе до залучення великої кількості маловідомих артистів та їх професійному просуненню.

ВИСНОВКИ

У рамках кваліфікаційної роботи було розглянуто процес розробки музичного застосунку «Melody» та був реалізований його прототип з унікальною можливістю створення плейлисту на основі погоди завдяки використанню поточної геолокації користувача та розроблено ключову аналітику для ефективного використання даних щодо погодних умов та музичних вподобань користувачів.

Було проведено дослідження та вирішено низку завдань, включаючи створення зручного інтерфейсу, збір інформації, розробку алгоритмів для підбору музичних композицій відповідно до погодних умов, а також проведення глибинного аналізу вимог майбутнього продукту залученням усіх зацікавлених сторін.

Важливою частиною роботи була розробка прототипу, який базується на отриманій аналітиці та вимогах до музичного застосунку з можливістю приваблення широкої аудиторії користувачів. Цей прототип має забезпечити користувачам можливість отримувати персоналізовані рекомендації музики в залежності від поточних погодних умов.

У висновку можна сказати, що ця робота дозволила показати ефективність використання аналітики даних для розробки персоналізованих музичних застосунків та підкреслити важливість зв’язку між погодою, настроєм та музикою для задоволення потреб користувачів.

Результати роботи апробовано у вигляді тез доповідей під час Міжнародної конференції «Trends in the development of quality training of future specialists» [31].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Gorokhovatskyi, V., Tvoroshenko, I., Kobylin, O., & Vlasenko, N. (2023). Search for visual objects by request in the form of a cluster representation for the structural image description. *Advances in Electrical and Electronic Engineering*, 21(1), 19-27.

2. Tvoroshenko, I., Gorokhovatskyi, V., Kobylin, O., & Tvoroshenko, A. (2023). Application of deep learning methods for recognizing and classifying culinary dishes in images.

3. Yakovleva, O., Kovtunenko, A., Liubchenko, V., Honcharenko, V., & Kobylin, O. (2023). Face Detection for Video Surveillance-based Security System. *In COLINS (3)* (pp. 69-86).

4. Bourreau, Marc, and Germain Gaudin. «Streaming platform and strategic recommendation bias.» *Journal of Economics & Management Strategy* 31.1 (2022): 25-47.

5. Tvoroshenko, I., Pomazan, V., Gorokhovatskyi, V., & Kobylin, O. (2023). Application of video data classification models using convolutional neural networks.

6. Lyashenko, V., Kobylin, O., & Selevko, O. (2020). Wavelet analysis and contrast modification in the study of cell structures images.

7. Kuzminska, O., Mazorchuk, M., Morze, N., & Kobylin, O. (2020). Digital learning environment of ukrainian universities: The main components to influence the competence of students and teachers. In *Information and Communication Technologies in Education, Research, and Industrial Applications: 15th International Conference, ICTERI 2019, Kherson, Ukraine, June 12–15, 2019, Revised Selected Papers 15* (pp. 210-230). Springer International Publishing.

8. Narayn, H. (2022). Just React!. *Learn React the right way. E-kirja. Viitattu*.

9. Windmill, E. (2020). *Flutter in action*. Simon and Schuster.

10. Singh, N., Chouhan, S.S., & Verma, K. (2021, October). Object oriented programming: Concepts, limitations and application trends. In *2021 5th International Conference on Information Systems and Computer Networks (ISCON)* (pp. 1-4). IEEE.

11. Liberty, J., Juarez, R., & Montaquila, M. (2023). . *NET MAUI for C# Developers: Build cross-platform mobile and desktop applications*. Packt Publishing.

12. Kobylin, O., & Lyashenko, V. (2020). Time series clustering based on the k-means algorithm.

13. Sabo, M. (2020). NestJS (Doctoral dissertation, Josip Juraj Strossmayer University of Osijek. Department of Mathematics. Chair of Applied Mathematics. Computer Science Research Group).

14. Dubnitskiy, V., Kobylin, A., Kobylin, O., Kushneruk, Y., & Khodyrev, A. (2023). Обчислення значень функції Харрінгтона (функції бажаності) при інтервальному визначенні її аргументів. *Advanced Information Systems*, 7(1), 71-81.

15. Bryan, N.J., Herrera, J., Oh, J., & Wang, G. (2010, June). MoMu: A Mobile Music Toolkit. In *NIME* (pp. 174-177).

16. Dyadun, S., Yakovlev, S., & Kobylin, O. (2022). Mathematical Modeling of Steady Flow Distribution in Water Supply Networks with Pumping Stations and Regulating Capacitances. In *ProfIT AI* (pp. 78-83).

17. Dyadun, S., Bodyanskiy, Y.V., Korobchynskyi, K., & Kobylin, O. (2022). Methods For Increasing The Reliability Of The Functioning Of Pipeline Systems. In *ProfIT AI* (pp. 104-109).

18. Dubnitskiy, V., Kobylin, A., Kobylin, O., Kushneruk, Y., & Sheviakov, I. (2022). Обчислення значень функцій комплексної змінної 3 інтервальним аргументом, визначеним в гіперболічній формі. *Advanced Information Systems*, 6(3), 83-91.

19. Декомпозуй це: розкладаємо великі та складні задачі на компоненти. URL: https://www.globallogic.com/ua/insights/blogs/decomposition-and-estimati on- techniques/ (дата звернення 26.03.2024).

20. Хто такі стейкхолдери. URL: https://foxminded.ua/stejkkholdery/ (дата звернення 18.04.2024).

21. Стейкхолдери: що таке і чому важливі. URL: https://smartik.Kiev. Ua/ stejkkholdery-shcho-take-i-chomu-vazhlyvi/ (дата звернення 19.04.2024).

22. Стейкхолдери проєкту: хто такі та чому важливо налагодити з ними комунікацію. URL: https://wizeclub.education/blog/stejkholderi-proyektu-hto-taki -chomu-vazhlivo-nalagoditi-z-nimi-komunikatsiyu/ (дата звернення 19.04.2024).

23. Oleg, K., & Anastasiia, L. (2021). Research of superpixel image segmentation method. Editorial board, 566.

24. Lyashenko, V., Kobylin, O., Ryazantsev, O., Ryazantsev, I., Barbaruk, V., & Zhychenko, Y. (2020). General Ideology of Analysis Digital Medical Images in RGB Format.

25. Berardi, D., Calvanese, D., & De Giacomo, G. (2005). Reasoning on UML class diagrams. *Artificial intelligence*, 168(1-2), 70-118.

26. Kobylin, O., & Lysenko, A. (2021). Research of superpixel image segmentation method.

27. Gemino, A., & Parker, D. (2009). Use case diagrams in support of use case modeling: Deriving understanding from the picture. *Journal of Database Management (JDM)*, 20(1), 1-24.

28. Daradkeh, Y.I., Gorokhovatskyi, V., Tvoroshenko, I., & Zeghid, M. (2022). Tools for fast metric data search in structural methods for image classification. *IEEE Access, 10*, 124738-124746.

29. Ibrahim Daradkeh, Y., Gorokhovatskyi, V., Tvoroshenko, I., & Al-Dhaifallah, M. (2022). Classification of Images Based on a System of Hierarchical Features. *Computers, Materials & Continua*, 72(1).

30. Daradkeh, Y.I., Gorokhovatskyi, V., Tvoroshenko, I., Gadetska, S., & Al-Dhaifallah, M. (2023). Statistical data analysis models for determining the relevance of structural image descriptions. *IEEE Access*, *11*, 126938-126949.

31. Бобейко, К.С., Кобилін, О.А. (2024). Дослідження методів оптимізації процесу розробки музичного застосунку за допомогою аналізу даних. Proceedings of the XX International Scientific and Practical Conference. Oslo, Norway. 2024. Pp. 351-353