Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система рекомендацій музики на основі вподобань групи користувачів. Мобільний додаток
(тема)

Виконав:
студент 4 курсу, групи ПЗПІ-20-10

Косинський О. О.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доц. кафедри ПІ Афанасьєва І. В.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
Кафедра Програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Косинському Олексію Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Програмна система рекомендацій музики на основі вподобань групи користувачів. Мобільний додаток
Затверджена наказом по університету від 20.05. 2024р. № 471 Ст.
2. Термін подання студентом роботи до екзаменаційної комісії 12.06.2024
3. Вихідні дані до роботи Розробити мобільний додаток програмної системи рекомендацій музики на основі вподобань групи користувачів застосовуючи мовами програмування C# та платформу MAUI.
4. Перелік питань, що потрібно опрацювати в роботі
Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі, огляд існуючих рішень, вибір найбільш придатних аналогів	08.04.2024 – 11.04.2024	виконано
2	Створення специфікації ПЗ, затвердження специфікації ПЗ керівником кваліфікаційної роботи	12.04.2024 – 16.04.2024	виконано
3	Проектування ПЗ	17.04.2024 – 24.04.2024	виконано
4	Розробка ПЗ	25.04.2024 – 17.05.2024	виконано
5	Тестування ПЗ	18.05.2024 – 24.05.2024	виконано
6	Оформлення пояснювальної записки	25.05.2024 – 01.06.2024	виконано
7	Перевірка роботи на антиплагіат	02.06.2024	виконано
8	Нормоконтроль	07.06.2024	виконано
9	Оцінка роботи рецензентом, отримання відгуку від керівника кваліфікаційної роботи	07.06.2024	виконано
10	Здача роботи у електронний архів, допуск роботи до захисту завідувачем кафедри	07.06.2024	виконано
11	Участь у демо-виставці	09.06.2024	виконано
12	Захист кваліфікаційної роботи	12.06.2024	виконано

Дата видачі завдання 5 березня 2024р.

Студент (ка)_____ Косинський О. О.
(підпис)

Керівник роботи _____ доц. кафедри ПІ Афанасьєва І. В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, 64 стор., 13 рис., 11 джерел.

МОБІЛЬНИЙ ДОДАТОК, МУЗИКА, СИСТЕМА РЕКОМЕНДАЦІЙ, ШТУЧНИЙ ІНТЕЛЕКТ, C#, MAUI, XAML.

Об’єкт розробки – система програмного забезпечення для рекомендації музики на основі вподобань групи користувачів.

Мета розробки – створення мобільного додатку для рекомендації музики на основі машинного навчання.

Метод рішення – середовище розробки Visual Studio та платформа MAUI, мова програмування C# та язик розмітки сторінок XAML , фреймворк .NET 8.

У результаті спроектовано мобільний додаток для рекомендації музики, що значно спрощує процес пошуку музики для групи користувачів.

MOBILE PART, MUSIC, RECOMMENDATION SYSTEM, ARTIFICIAL INTELLIGENCE, C#, MAUI, XAML.

Development object – a software system for recommending music based on the preferences of a group of users.

Development goal – creation of the mobile part of the application for music recommendation based on machine learning.

Solution method – Visual Studio development environment and MAUI platform, C# programming language and XAML page markup language, .NET 8 framework.

As a result, a mobile application for music recommendation has been designed, significantly simplifying the music search process for a group of users.

Я, Косинський Олексій Олексійович, студент гр. ПЗПІ-20-10, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Програмна система рекомендацій музики на основі вподобань групи користувачів. Мобільний додаток», що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

MAUI – .NET Multi-platform App UI

HTTP – Hyper Text Transfer Protocol

XAML – Extensible Application Markup Language

T-SQL – Transact-SQL

JWT – JSON Web Token

NUnit – .NET Unit Testing Framework

ВСТУП

Музика є одним з популярних видів мистецтва. З часом, прослуховування музики припинило бути чимось індивідуальним і почало набирати груповий характер: люди слухають музику разом з друзями через одні навушники або включають свої улюблені композиції на вечірці через динамік.

Питання рекомендаційних систем є доволі актуальним , оскільки з появою пристроїв виведення інформації люди все частіше почали сприймати прослуховування музики не як хобі, а як ціле мистецтво, що потребує докладного підбору композицій для прослуховування. Але як знайти музику, яка вам дійсно подобається? Або, що ще складніше, як знайти музику, яка відповідає вподобанням компанії?

Метою цієї кваліфікаційної роботи є розробка та реалізація мобільної частини програмної системи рекомендації музики на основі вподобань групи користувачів.

Оскільки обсяг оброблюваних даних може бути дуже великим, найкращим рішенням в цій ситуації є використання технологій машинного навчання для прогнозування результатів на основі кількох різних факторів.

Методи розробки базуються на платформі MAUI для написання крос-платформних та мобільних додатків та мови програмування C#. База даних сервера написана на SQL за допомогою T-SQL. Зовнішній вигляд сторінок написано з використанням мови розміток XAML-.

В результаті кваліфікаційної роботи буде розроблена та впроваджена мобільна частина програмної системи рекомендації музики на основі вподобань групи користувачів. Кінцевий продукт матиме широкий спектр потенційних користувачів та буде націлений на галузь музичних сервісів та платформ. Данна система пропонуватиме метод наближення людей до музики, яка дійсно відповідає їхнім унікальним смакам або колективним вподобанням групи, поліпшуючи загальний досвід прослуховування музики.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Музика – одна з самих найстаріших форм мистецтва яка бере свій початок з моменту виникнення людства. Вона відіграє важливу роль у житті багатьох людей, супроводжуючи їх у різних ситуаціях - від святкових подій до повсякденної рутини. Музика здатна впливати на настрій, мотивацію та загальне самопочуття людини. Не дивно, що за великий час існування цієї форми мистецтва – вона продовжує бути актуальною та залучати до себе все більше людей

Рекомендаційні системи – це технології, метою яких є аналіз даних про інтереси користувачів та підбір персоналізованих рекомендацій на основі їх вподобань. Ці системи широко використовуються у різних галузях, а також працюють на багатьох платформах, в тому числі на мобільних [1]. Завдяки розвитку технологій та алгоритмів, рекомендаційні системи стали невід'ємною частиною сучасних цифрових сервісів. Вони допомагають користувачам знаходити новий контент, який відповідає їхнім вподобанням, заощаджуючи час та зусилля на пошук.

Мобільні застосунки для рекомендації музики базуються на складних алгоритмах аналізу даних та часто використовують технології машинного навчання. Вони збирають інформацію про музичні вподобання користувачів, аналізують її та використовують для рекомендації треків, які ймовірно сподобаються конкретному користувачеві [2]. Такі алгоритми враховують безліч факторів, включаючи історію прослуховувань, оцінки треків, поведінкові патерни користувачів, а також метадані музичних композицій. Це дозволяє створювати точні та персоналізовані рекомендації, які максимально відповідають інтересам слухача

У наш час, зі збільшенням кількості споживаного людиною контенту, виникає необхідність у створенні рекомендаційних систем, які могли б обробляти цей контент і рекомендувати найбільш відповідний для користувача. На жаль, не

існує універсальної системи, яка б рекомендувала контент, який безумовно сподобався б користувачеві, а також на ринку існує не дуже багато систем, які би враховували інтересі декількох користувачів. Важливим аспектом є також постійне вдосконалення алгоритмів, щоб враховувати зміни у вподобаннях користувачів та адаптуватися до нових трендів у музичній індустрії. Крім того, необхідно забезпечити високу точність та релевантність рекомендацій, щоб користувачі отримували лише ті треки, які дійсно відповідають їхнім смакам.

Отже, предметна галузь включає в себе розробку рекомендаційної системі, яка обробляє музичний контент, для вирішення проблеми пошуку музики в компанії з різними музичним інтересами. Створення такої системи потребує глибокого розуміння як індивідуальних, так і групових музичних вподобань. Важливою є також інтеграція з різними музичними платформами та забезпечення зручного та інтуїтивного користувацького інтерфейсу. Результатом стане інструмент, який дозволить групам людей насолоджуватися музикою, що відповідає їхнім спільним інтересам, створюючи приємну та гармонійну атмосферу під час прослуховування.

1.2 Виявлення та вирішення проблеми

Багато музичних плеєрів та платформ базуються на рекомендаційних системах [3]. Деякі з них використовують прості алгоритми на основі жанру та деяких атрибутів треку, а інші використовують складні алгоритми з високим рівнем точності [4]. Для того щоб виявити проблему, потрібно детально проаналізувати основних гравців що присутні на ринку рекомендаційних систем. Один з основних конкурентів у сфері персональних рекомендацій – це стрімінговий сервіс Spotify.

Головна особливість цього застосунку – одна з кращих систем персональних музичних рекомендацій на ринку. Алгоритми застосунку у 7 з 10 випадків підберуть користувачу трек, який йому сподобається. Система обирає треки на основі інтересів людей, які мають такий же плей лист, як і кінцевий користувач. Також система аналізує музику та її тексти, а також може змінювати рекомендації на ходу під час прослуховування музики, сподіваючись підібрати ідеальний трек.

Хоча Spotify має потужну систему персональних рекомендацій, але якщо мова йде про підбір рекомендацій на основі групових інтересів, можуть виникнути проблеми. Оскільки алгоритми рекомендацій та сам додаток заточені на отримання персональних пропозицій та не враховують інтереси декількох користувачів. Самі рекомендації постійно пропонуються користувачеві на головній сторінці застосунку (див.рис. 1.1).

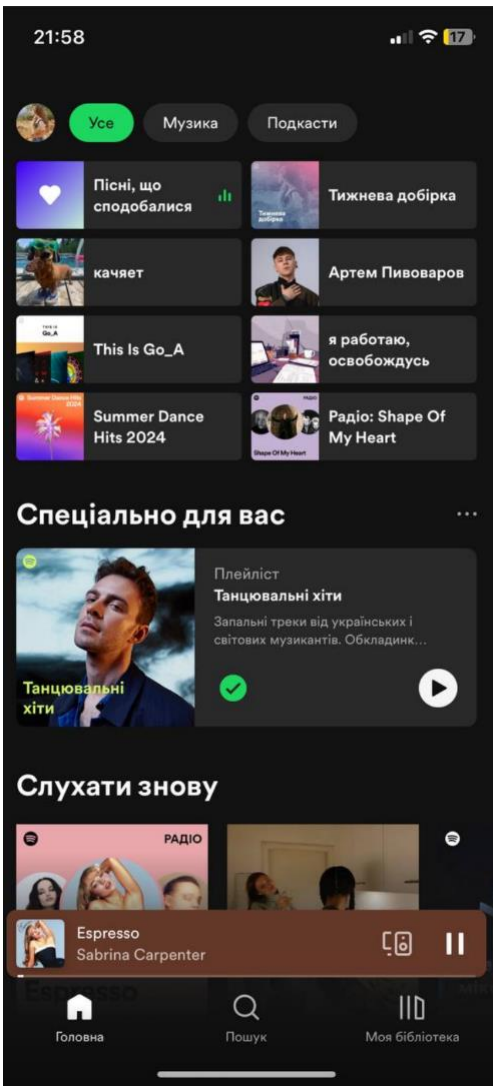


Рисунок 1.1 – Інтерфейс мобільного застосунку Spotify (рисунок виконаний самостійно)

Ще один приклад конкурентного сервісу, що вартий уваги, – це YouTube Music. Цей сервіс також надає персоналізовані рекомендації, використовуючи складні алгоритми машинного навчання. YouTube Music аналізує історію прослуховувань, вподобання користувачів та інші фактори, щоб запропонувати

нові треки та плейлисти. Водночас YouTube Music має функцію "Сімейний план", яка дозволяє декільком користувачам користуватись сервісом з одного облікового запису, але функції групових рекомендацій в ньому також обмежені. Алгоритми сервісу орієнтовані на персональні вподобання кожного користувача окремо, що робить його менш ефективним для ситуацій, коли потрібно врахувати інтереси групи людей.

Ще одним гідним конкурентом, але в іншій сфері, є сервіс – datenightmovies.com. Цей сервіс надає систему рекомендацій на основі групових інтересів, що дозволяє підбирати фільми для спільного перегляду. Хоча він не працює з музикою, його підхід до групових рекомендацій є цікавим і корисним прикладом для нашої задачі.

Цей ресурс має ряд переваг. По-перше, він пропонує великий набір даних, який постійно оновлюється новими фільмами. Це дозволяє користувачам завжди мати доступ до актуального контенту. По-друге, сервіс ефективно вирішує проблему підбору фільмів для групи людей з різними смаками, що є важливим аспектом для створення позитивного користувацького досвіду.

Оскільки розробник не має аналогічного сервісу який би працював у галузі групового підбору музики – це створює можливість для розробки нового продукту, що заповнить цю нішу. Використання методів та принципів, які застосовує datenightmovies.com для підбору фільмів, може бути адаптоване для музичних рекомендацій, створюючи унікальний інструмент для користувачів, які слухають музику в компанії. Використання позитивного досвіду datenightmovies.com може стати важливою складовою у розробці нашого продукту, забезпечуючи його конкурентоспроможність та унікальність на ринку.

До недоліків конкуренту можна віднести відсутність мобільного застосунку. Наявність окремого мобільного застосунку дозволяє користувачам отримувати швидший доступ до сервісу, використовувати його без потреби відкривати браузер, а також отримувати сповіщення та інші функції, які можуть бути важливими для користувачів. Наразі існує лише адаптована до мобільного екрану веб версія сервісу (див.рис. 1.2).

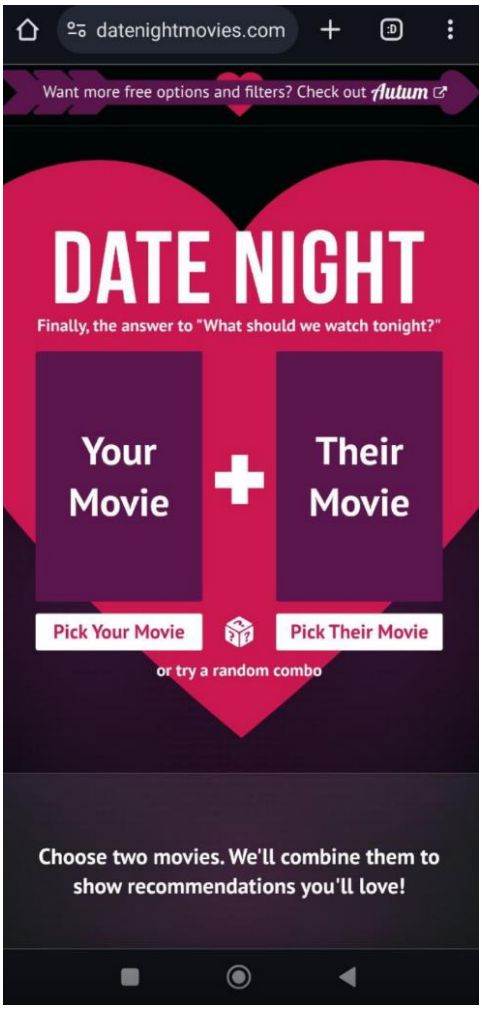


Рисунок 1.2 – Інтерфейс сервісу datenightmovies (за даними [5])

Існуючі сервіси, такі як Spotify, datenightmovies.com та YouTube Music, демонструють важливість і ефективність персоналізованих рекомендацій, проте їхні алгоритми не завжди здатні врахувати групові інтереси. Це відкриває можливості для створення нової системи, яка могла б задовольнити цю потребу. Така система повинна враховувати переваги та недоліки існуючих рішень, поєднуючи найкращі аспекти персоналізованих рекомендацій з можливістю адаптації до групових вподобань [6]. Це дозволить створити унікальний продукт, який забезпечить високий рівень задоволення користувачів та підвищить їх лояльність до сервісу.

1.3 Постановка задачі

Для вирішення актуальної проблеми людей, які хочуть знайти цікаву музику у компанії, потрібно створити мобільний додаток, який би підбирав музику на основі інтересів декількох користувачів, та надав би рекомендації стосовно треків які би змогли задовольнити музикальні інтереси усіх [7].

Створення такого застосунку в першу чергу передбачає аналіз вимог та конкурентів, під час якого збирається інформація про потреби та очікування користувачів і зацікавлених сторін а також сильні та слабі сторони основних конкурентів. Також слід приділити увагу аналізу основних користувачів застосунку. Після цього на основі цього аналізу формується технічне завдання, яке включає функціональні та нефункціональні вимоги до системи. Цей етап є критично важливим, оскільки точність та повнота зібраних вимог визначає успіх подальших фаз розробки.

Наступним кроком є проектування та прототипування системи, де визначаються архітектурні рішення, структура даних, інтерфейси та алгоритми. з загальними принципами та підходами. Важливим аспектом цієї фази є створення прототипів, що дозволяють візуалізувати основні функції системи та отримати зворотний зв'язок від користувачів.

Враховуючі поставлену проблему, можна сказати, шо розробка застосунку повинна проводитись з використанням технологій штучного інтелекту та з використанням REST підходу звертання до серверів. У проекті повинен бути не маленький дата сет для тренування моделі, цей дата сет має включати у себе різні види музики від нових виконавців, до старих., а також різні жанри та види музики. Оскільки існує безліч музикальних інтересів – у самому найкращому випадку система повинна задовольнити інтересів більшості користувачів. Також допускається використання у дата сеті реміксів та змінених версій композицій за для покращення результату рекомендацій.

Також основною вимогою при проектуванні застосунку є – питання конфіденційності та захисту даних. Збираючи інформацію про музичні вподобання,

система повинна забезпечувати її безпечне зберігання та обробку, а також дотримуватись усіх вимог щодо захисту персональних даних. Доступ до загальних даних користувачів повинен мати лише учасник системи з роллю адміністратору, а доступ до особистих даних повинен мати лише розробник системи.

При проектуванні графічного інтерфейсу застосунку повинні дотримуватися основні концепції user-friendly дизайну. Можливо сказати, що до цього також відноситься створення зручної навігації з використанням запропонованих середою розробки елементів, а також використання іконок та векторних зображень. Додаток повинен мати легкий та інтуїтивно зрозумілий інтерфейс, бути невибагливим та запускатися на більшості мобільних девайсів. Основна сторінка повинна мати як мінімум два елементи пошуку, щоб задовольнити музикальні інтереси як мінімум двох осіб. Повинен бути необхідній функціонал стосовно адміністрування та редагування особистих даних профілю, а також обов’язкова локалізація на англійську мову.

Також при розробці мобільного застосунку потрібно приділяти увагу способам взаємодії користувача і системи, наприклад користувач може взаємодіяти через сенсорні жести які доступні на мобільних девайсах. Всі ці моменти повинні бути враховані при розробці продукту

Після виконання перших двох етапів потрібно перейти до фази реалізації. Вона передбачає безпосереднє написання програмного коду відповідно до специфікацій, створених на попередніх етапах. Ця фаза також включає інтеграцію окремих компонентів системи та їх тестування. Важливо відмітити, що кінцевий продукт повинен пройти декілька циклів ретельного тестування, та мати можливість відслідковувати помилки.

Успішне створення цього застосунку дозволить вирішити проблему пошуку музики у групі людей з різними музичними інтересами. Завдяки можливості врахування інтересів декількох користувачів одночасно, розробляємий додаток зможе забезпечити унікальний користувацький досвід, сприяючи кращому взаєморозумінню в групі та підвищуючи загальне задоволення від

прослуховування музики. Такий мобільний додаток може стати незамінним помічником на усіляких вечірках або колективах, які прослуховують музику разом.

1.3.1 Цільова аудиторія продукту

Визначення цільової аудиторії є головним кроком у розробці програмного продукту, оскільки саме розуміння потреб, очікувань та поведінки користувачів дозволяє створити продукт, що буде максимально корисним та ефективним. Важливо враховувати, що точне розуміння цільової аудиторії дозволяє оптимізувати всі аспекти розробки, від дизайну інтерфейсу до функціональних можливостей, забезпечуючи тим самим прискорену розробку та високу конкурентоспроможність продукту.

Основна цільова аудиторія програмної системи рекомендацій на основі вподобань групи користувачів включає у себе всіх хто має небайдуже ставлення до прослуховування музики. В цілому дані люди поділяються на дві основні групи, ті хто слухає музику самостійно, та ті хто звик слухати музику у компанії. Саме на другу групу людей буде приходитися основа аудиторія продукту, оскільки продукт вирішує проблему знаходження цікавої музики для групи користувачів. Дослідження показують, що люди, які слухають музику в компанії, часто шукають нові способи зробити цей досвід більш приємним та взаємно задовольняючим для всіх учасників.

Важливо сказати, що серед аудиторії продукту також можуть бути люди, що полюбляють саме індивідуальне прослуховування музики, але їх відсоток буде невеликим. Незважаючи на це, продукт може бути цікавим і для них завдяки можливості відкривати нові музичні смаки та вподобання, що згодом можуть бути корисними під час різних зборів групою або просто для розширення власного музичного кругозору

До найбільш поширеного портрету користувача системи можна віднести молодь та дорослих віком від 18 до 45 років. Ця вікова група найбільш активно використовує мобільні пристрої та мобільні додатки для різних цілей Усередині цієї групи можна виділити кілька підгруп з різними потребами та вподобаннями,

зокрема студенти, молоді професіонали, сімейні люди та технологічні ентузіасти, але з одним бажанням – знаходити музику яка б подобалась не лише особисто користувачу, а також його друзям або членам родини. Кожна з цих підгруп має свої специфічні особливості та вимоги, але всі вони об'єднані прагненням до якісного музичного досвіду, що відповідає їхнім соціальним і культурним контекстам.

При аналізі цільової аудиторії мобільного продукту, важливо сказати, що цільова аудиторія характеризується високим рівнем технологічної підкованості. Більшість користувачів має значний досвід використання мобільних пристроїв, активно користується соціальними мережами, онлайн-сервісами та мобільними додатками. Вони очікують інтуїтивно зрозумілого інтерфейсу, швидкої та безперебійної роботи системи. Це означає, що продукт повинен бути не лише функціонально насиченим, але й високоякісним з точки зору користувацького досвіду. Інтуїтивний дизайн, швидкість реакції та стабільність роботи є критичними факторами для задоволення потреб цієї аудиторії

Дивлячись на високий рівень технічного досвіду та підкованості майбутніх користувачів застосунку, можливо зробити висновок що, літні люди не будуть використовувати додаток, бо використання смартфону є для них важким процесом, тому на цю категорію людей розраховувати не варто. Проте, це не виключає можливості, що з часом, зі зростанням технічної обізнаності серед старшого покоління, частина цієї аудиторії може також зацікавитися подібними технологіями, особливо якщо додаток буде адаптовано під їхні потреби і здібності.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Функціональні вимоги

Враховуючи обрану предметну галузь, можна сформувати основні функціональні вимоги до системи:

– реєстрація та аутентифікація користувачів: система повинна мати можливість реєстрації та аутентифікації користувачів з різними ролями та надавати доступ до функцій згідно з ролями користувачів системи;

– керування обліковим записом: система повинна надавати користувачу можливість виконувати основні функції стосовно редагування та оновлення даних облікового запису, а також надавати можливість за бажанням видалити його;

– адміністрування: система згідно за роллю користувача повинна надавати можливість видавати та забирати права адміністратора системи, та можливість відобразити та банити користувачів;

– робота з рекомендаціям: система повинна надавати користувачу можливість формування рекомендації на основі декількох інтересів, а також переглядати історію минулих рекомендацій;

– статистика: система повинна надавати користувачу доступ до статистики, стосовно авторизації та реєстрації , а також статистику стосовно підписки;

– оплата: система повинна надавати можливість користувачу здійснювати оплату з банківської карти.;

– локалізація: система повинна надавати змогу користувачу обирати мову: українську або англійську.

2.2 Нефункціональні вимоги

Враховуючи обрану предметну галузь, можна сформувати основні нефункціональні вимоги до системи:

– масштабованість: враховуючи зростання кількості музики кожен день а також попиту до рекомендаційних додатків, система має мати можливість справлятися зі збільшенням обсягу даних та кількістю користувачів ;

– плавність у використанні: система повинна бути добре оптимізована мати анімації та не викликати дискомфорту при використанні;

– відмовостійкість: система повинна мати можливість виконувати основний функціонал навіть за умовою виходу з ладу деяких компонентів системи;

– тестованість: система повинна надавати можливість легко протестувати будь який компонент системи та мати можливість виявити помилки при тестуванні;

– невибагливість: система повинна мати можливість добре функціонувати на не дуже сильних за технічними характеристиками девайсах, система не повинна вимагати високих технічних характеристик девайсу.

Після визначення усіх основних функціональних та нефункціональних вимог, можна перейти до проектування архітектури системи.

2.3 Припущення та залежності

Припущення 1. Продукт буде затребуваний на ринку;

Припущення 2. Збої у роботі деяких компонентів можуть привести до зупинки цілої системи;

Припущення 3. Зворотна сумісність;

Припущення 4. Через обмежені терміни проект може вийти недостатньо стабільним.

Залежність 1. Мобільний додаток буде розроблено за допомогою .NET 7 MAUI;

Залежність 2. Використання Azure SQL DB для зберігання даних на хмарі;

Залежність 3. Використання Azure Blob-сховища для зберігання графічних зображень на хмарі;

Залежність 4. Використання розмітки XAML для написання графічного інтерфейсу додатку;

Залежність 5. Залежність часу розробки цього проекту від часу, який розробники витрачають вивчення та розробку паралельних завдань

2.4 Обмеження

Обмеження 1. Мобільному додаток для роботи потребує стабільного підключення до межі інтернет. Відсутність підключення приводить до зупинки роботи застосунку;

Обмеження 2. Усі засоби розробки мають бути легітимними. Це включає в себе використання ліцензійного програмного забезпечення, дотримання ліцензійних угод та уникнення піратських або неліцензійних інструментів;

Обмеження 3. Керувати адміністраторами може лише кореневий користувач або розробник системи(root, superadmin). Це обмеження гарантує, що управління критичними аспектами системи залишатиметься під контролем обмеженого кола користувачів, що зменшує ризик несанкціонованого доступу та забезпечує високий рівень безпеки;

Обмеження 4. Доступ до адмін-панелі мають лише адміністратори;

Обмеження 5. Доступ до статистики мають лише адміністратори.

3 АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проектування ПЗ

Для відображання взаємодії між акторами та системою в рамках певного контексту була побудована UML діаграма прецедентів (див. рис. 3.1):

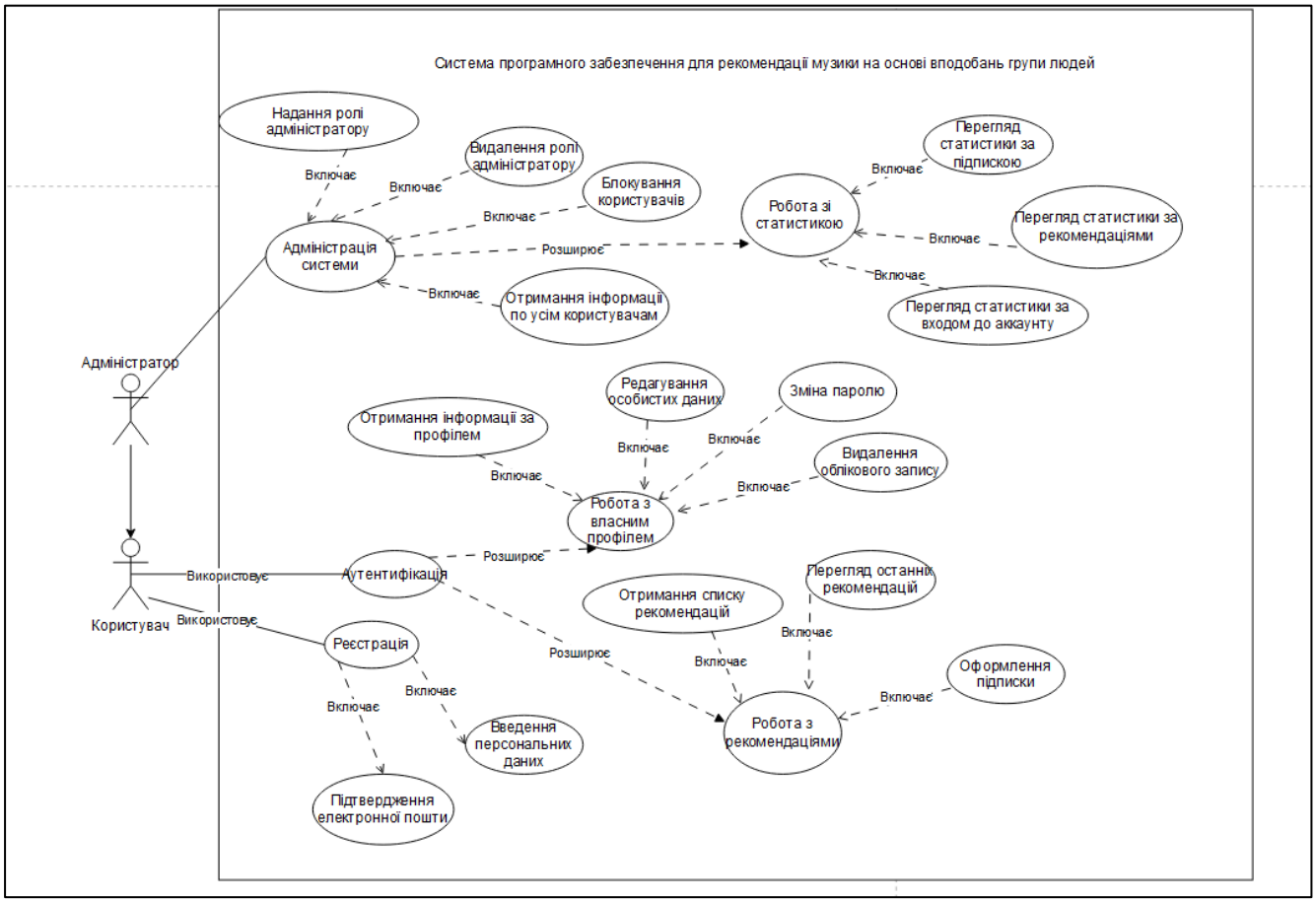


Рисунок 3.1 – UML діаграма прецедентів (Use Case Diagram) (рисунок виконаний самостійно)

Додаток має 2 типи користувачів, а саме: користувач та адміністратор системи. Загальний функціонал, який доступний обом користувачам:

1. Реєстрація акаунту;
2. Аутентифікація та авторизація;
3. Керування особистою інформацією (особистий кабінет);
4. Можливість оплати підписки;
5. Перегляд списку раніше рекомендованих пісень;

6. Доступ до генерації рекомендацій за допомогою штучного інтелекту;

7. Можливість підтвердження електронної пошти.

Для користувачів з роллю «Адміністратор» доповняють наступні функції :

1. Адміністрування;

1.1 Перегляд списків усіх користувачів;

1.2 Зміна ролей користувача;

1.3 Блокування\розблокування користувача;

2. Можливість переглянути статистику.

3.1.1 UML діаграма пакетів

Для кращого розуміння залежностей між пакетами, була створена діаграма пакетів проекту (див. рис. 3.2):

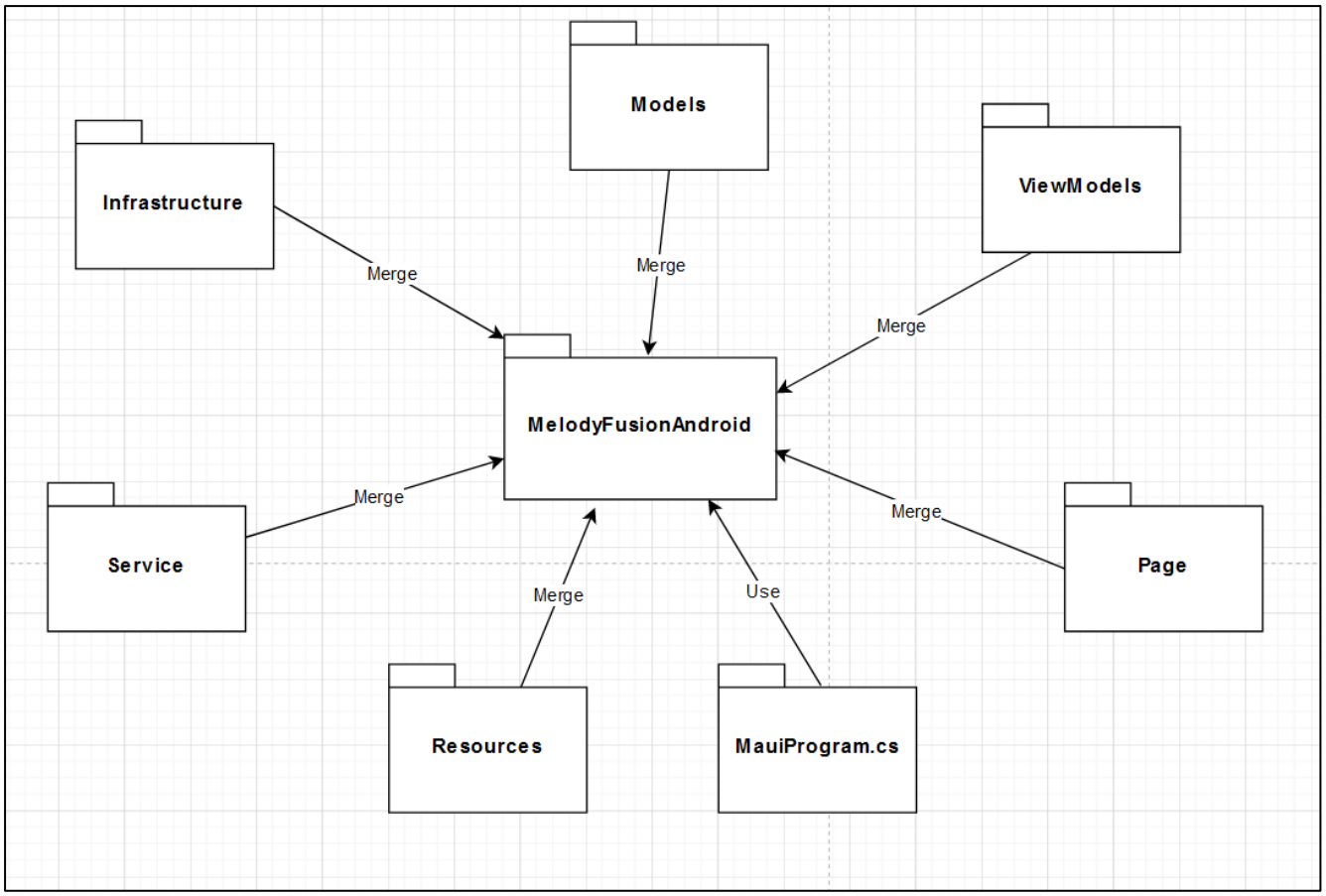


Рисунок 3.2 – UML діаграма пакетів проекту «Програмна система рекомендації музики на основі вподобань групи користувачів. Мобільний додаток» (рисунок виконаний самостійно)

Було виділено основний пакет рішення, який включає такі пакети, як :

– сторінки (Pages) включають розмітку сторінок, а також до кожної сторінки належить додатковий файл з розширенням cs для ініціалізації компонентів сторінки та виконання логіки;

– моделі (Models) містять сутності бази даних та їхні відображення, а також містять сутності для взаємодіє з сервером – усілякі класи на отримання та формування запитів;

– сервіси (Services) містять в собі класи сервісів, що виконують звернення та відправку запитів до серверу;

– інфраструктура (Infrastructure) включає компоненти, які виконують деякі індивідуальні завдання, наприклад клас локального сховища для збереження JWT токену користувача;

– моделі для відображення (ViewModels) містять сигнатурні класи деяких об’єктів для відображення певних атрибутів на сторінці, та передачі даних з функцій до розмітки;

– ресурси (Resources) містять додаткові ресурси для відображення дизайну додатку, наприклад стилі, зображення та інші;

– MauiProgram.cs є основним виконуючим файлом додатку та виконує реєстрацію усіх сервісів та задає правила поведінки застосунку.

Важливо додати, що описані вище пакеті та класи не є кінцевим виглядом директорії проекту. У ході розробки можливе створення нових пакетів та файлів.

3.1.2 Діаграмі діяльності

Для кращого розуміння процесів було створено діаграму діяльності, що графічно відображає послідовність дій, які відбуваються у системі. Ця діаграма дозволяє легко відстежувати потік робіт та приймати обґрунтовані рішення щодо оптимізації процесів. Використання діаграми допомагає виявити вузькі місця, потенційні проблеми та покращити загальну ефективність системи. (див. рис. 3.3):

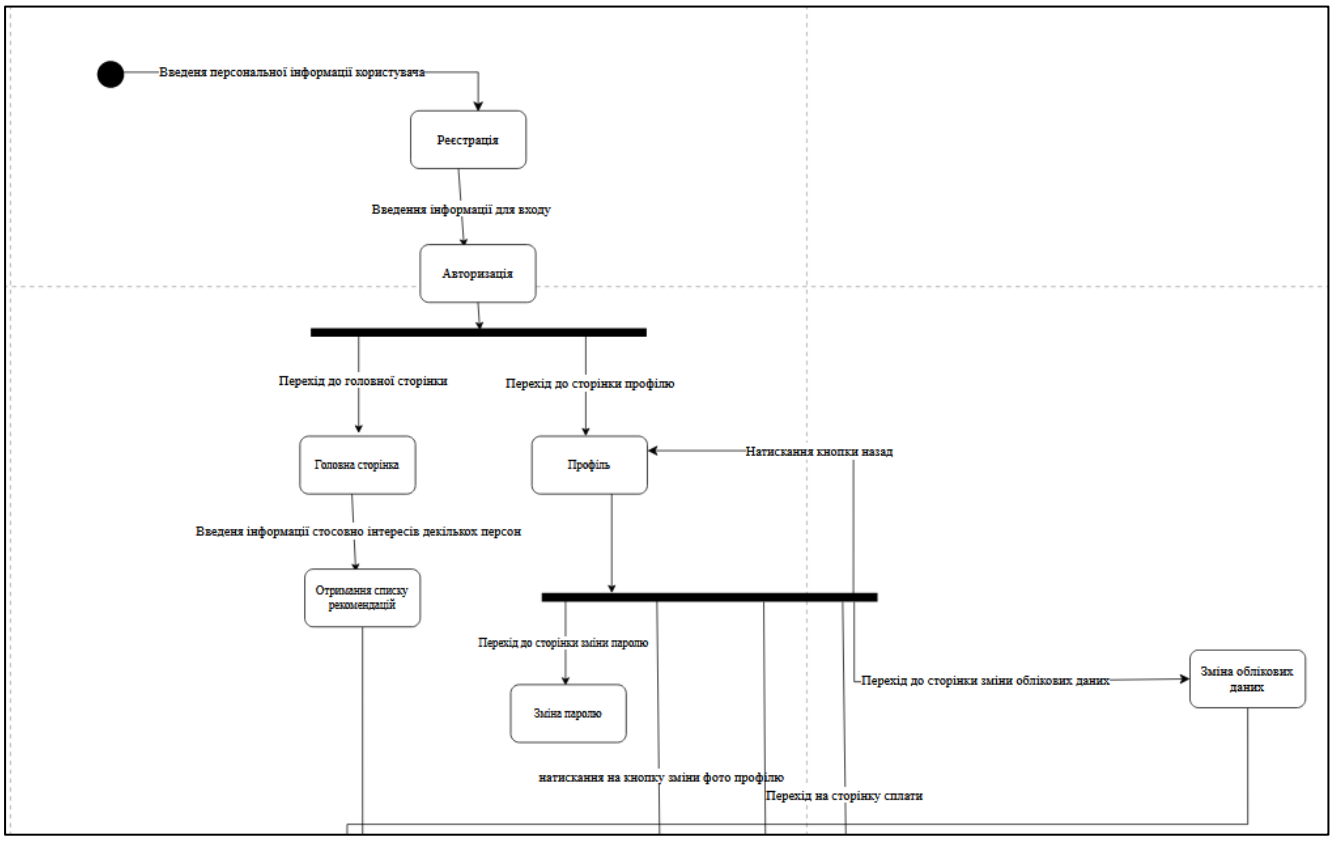


Рисунок 3.3 – UML діаграма діяльності проекту (рисунок виконаний самостійно)

Створена діаграма дозволяє візуалізувати, які кроки може виконувати користувач та як саме він може знайти деякі функції застосунку.

Діаграма починається з точки входу, що є початком взаємодії користувача з мобільним застосунком. Після введення персональних даних та реєстрації у системі користувач має можливість ввести дані на сторінці «Авторизації» та перейти на головну сторінку додатку. Сторінка «Авторизації» та «Реєстрації» – це єдині сторінки, що відображаються користувачеві, котрий не увійшов до системи під своїм обліковим записом. Авторизувавшись у системі користувач має можливість перейти до двох основних розділів застосунку: «Головна» та сторінки «Профіль користувача». На головній сторінці реалізована основна логіка застосунку, а саме отримання рекомендацій за інтересами групи людей.

На сторінці профілю користувач має можливість відкрити розділи «Зміни даних облікового запису», «Зміна паролю облікового запису» та «Підписка». Ці розділи направляють користувача на окремі сторінки для виконання обраної дії. Користувач має можливість у будь який момент повернутися на «Профіль

користувача» натиснувши кнопку з відповідною назвою. Розділ «Підписка» має у собі окрему сторінку з формою вводу банківських даних для списання коштів з карти та оформлення підписки на додаток. Розділ «Зміна аватарки облікового запису» не містить у собі нових сторінок, однак при натисканні на відповідну кнопку у користувача одразу відкривається файловий менеджер у якому він може обрати нове фото для зміни аватарки облікового запису. (див. рис. 3.4):

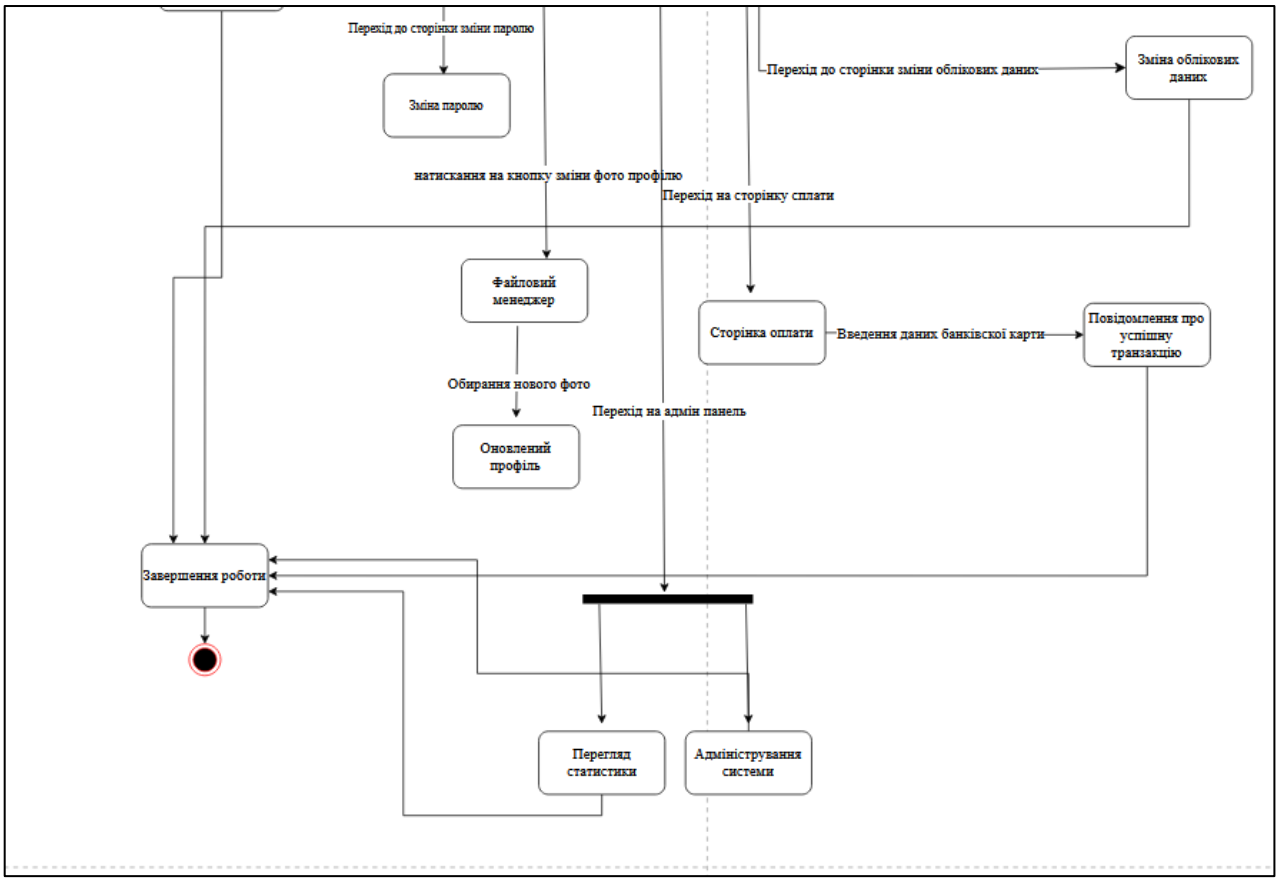


Рисунок 3.4 – Продовження UML діаграми діяльності проекту (рисунок виконаний самостійно)

Розділ «Адмін панель» відкриває окрему сторінку зі списком користувачів, у цьому розділі може виконуватися адміністрування системи, також цей розділ має підрозділ «Статистика». У цьому підрозділі адміністратор системи має можливість побудувати та переглянути графіки з відповідними статистичними даними.

Важливо додати, що розділ «Адмін панель» та «Статистика» доступні лише для адміністраторів системи, обліковий запис без ролі «Admin» не має можливості перейти на ці розділи, та також не бачить їх.

3.2 Проектування архітектури ПЗ

Програмна система для рекомендації музики на основі вподобань групи користувачів буде використовувати REST стиль та відповідно клієнт-серверний підхід до розробки застосунку[8].

У ролі серверу буде виступати BackEnd частина застосунку, що у свою чергу має 3-х поверхову архітектуру. За таким підходом, BackEnd сервер буде відповідальним за обробку запитів користувачів, а також за взаємодію з базою даних та штучним інтелектом для отримання рекомендацій музики на основі вподобань групи користувачів. Він буде включати в себе різноманітні сервіси та модулі, що забезпечують аналіз вподобань, обробку даних та генерацію персоналізованих рекомендацій. Така трьох поверхова архітектура дозволить відокремити логіку застосунку на рівні презентації, бізнес-логіки та доступу до даних, що полегшить розробку, тестування та масштабування системи.

Клієнтською частиною буде виступати сам мобільний додаток, тому що основну логіку роботи з даними буде виконувати сервер, а з мобільного застосунку будуть надходити запити. Дані у цьому підході будуть передаватися з використанням http протоколу. Клієнтська частина, буде відповідати за інтерфейс взаємодії з користувачем. Вона буде передавати запити до серверної частини застосунку та отримувати відповіді з рекомендаціями музики. Клієнтська частина також відповідатиме за відображення інформації та взаємодію з користувачем через графічний інтерфейс.

Використання HTTP протоколу для передачі даних дозволить забезпечити доступність та інтеграцію між клієнтською та серверною частинами системи, а також зробить можливим використання стандартних інструментів та бібліотек для розробки та підтримки додатку.

Клієнтська частина застосунку буде написана з використанням нової крос-платформної технології розробки мобільних додатків – MAUI[9]. Ця технологія побудована на основі Xamarin.Forms і спрямована на спрощення процесу розробки мобільних застосунків для кількох платформ, таких як Android, iOS, macOS, та

Windows, шляхом використання загального кодової бази. Серед причин вибору саме цієї платформи є висока продуктивність та широкі можливості завдяки використанню мови програмування C#, а також досвід роботи з цією мовою.

3.2.1 Опис сторонніх плагінів та розширень проекту

При розробці застосунку, для реалізації деяких бажаних функцій та методів вкрай важливо використовувати сторонні розширення. Нижче описані всі NuGet пакети які встановленні до проекту:

– Microsoft.Extensions.Http – надає зручні засоби для роботи з HTTP у застосунках на платформі .NET, забезпечуючи ефективне управління клієнтами HTTP та їх конфігурацією. У проекті використовується для створення та налаштування клієнту для подальшої відправки запитів на сервер;

– Microsoft.Maui.Controls – дозволяє розробникам створювати крос-платформні застосунки з користувацьким інтерфейсом, що працюють на різних операційних системах, з використанням зручних інструментів та шаблонів розробки. У проекті використовується для створення користувацького інтерфейсу (UI);

– Newtonsoft.Json – бібліотека для роботи з форматом JSON у застосунках, написаних на платформі .NET. Вона надає зручні і ефективні засоби для серіалізації (перетворення об'єктів .NET в JSON) і десеріалізації (перетворення JSON у об'єкти .NET) даних. У проекті використовується для роботи з JSON файлами, э необхідним пакетом у проекті;

– Google.Apis.Oauth2 – бібліотека для підключення та налаштування гугл авторизації, є необхідною для проекта;

– Microsoft.Extensions.Localization – бібліотека для створення та налаштування локалізації. У проекті використовується для перекладу користувацького інтерфейсу, є необхідним у проекті[10] ;

– LiveCharts – надає можливість для роботи з графіками. Дозволяє проводити їх створення та оновлення даних у режимі реального часу, є необхідним у проекті;

– ToastNotifications – не є необхідним додатком у проекті, але надає можливість створювати сповіщення у форматі Toast. Впливає на покращення досвіду використання застосунку, надає можливість інформувати юзера.

Важливо додати, що описані вище розширення поділяються на необхідні для коректного функціювання системі, та ні. Хоча деякі з них можуть бути видалені з проекту без зміни робото стійкості застосунку, але всі ці компоненти виконують конкретну поставлену задачу, тому повинні бути присутні у проекті у повному обсязі. Також з ходом розробки будуть доповнюватися та оновлюватися.

3.2.2 Специфікація Rest

Мобільний додаток виступає у ролі клієнту який робить запити до серверу. Після цього приймає та в залежності від відповіді виконує спеціальну логіку. Специфікація кожного з ендпоінту за яким клієнт може звернутися до серверу та отримати відповідь (див. Таблиця 3.1).

Таблиця 3.1 – Специфікація REST API (таблиця виконана самостійно)

Посилання на ендпоінт	Метод	Опис
/api/Authentication/Registration	Post	Реєстрація акаунту
/api/Authentication/Login	Post	Авторизація та аутентифікація
/api/Song	Get	Отримання існуючих пісень за назвою
/rating	Post	Відправка зворотної оцінки

Продовження таблиці 3.1

Посилання на ендпоінт	Метод	Опис
/api/Authentication/mail-confirmation	Get	Підтвердження пошти
/api/Admin/BanUser/{userId}	Patch	Блокування/Розблокування користувача
/api/Admin/AddRole	Patch	Додавання ролі
/api/Admin/DeleteRole	Patch	Видалення ролі
/api/Admin/GetUserList	Get	Отримання переліку користувачів
/api/EmailSender	Post	Відправка посилання на пошту
/api/Payment/Create	Post	Створення оплати
Song/get-recommendations	Get	Отримання рекомендації
/api/Statistic/GetUserInfo	Put	Статистика по юзерам
/api/Statistic/GetLoginInfo	Put	Статистика авторизацій/реєстрацій
/api/Statistic/GetRecommendationInfo	Get	Статистика рекомендацій
/api/User/GetUserInfo	Get	Отримання інформації про користувача
/api/User/DeleteAccount	Delete	Видалення аккаунту

Кінець таблиці 3.1

Посилання на ендпоінт	Метод	Опис
/api/User/UpdatePassword	Patch	Оновлення паролю
/api/User/UpdateUserInfo	Put	Оновлення інформації про юзера
/api/User/ChangeAvatar	Put	Оновлення аватару
/api/Song/get-last-recommendations	Get	Отримання останніх рекомендацій користувача

У мобільному застосунку звернення до кожного ендпоінту виконується у класах сервісах. Кожин сервіс має у собі екземпляр класу httpClient, в залежності від дії створюється URL-адреса ендпоінту. Це може бути запит на отримання даних, відправка нових даних, оновлення існуючих або видалення ресурсів на сервері.

Формування URL-адреси дозволяє динамічно змінювати ціль запиту та забезпечує гнучкість у роботі з різними ресурсами Потім за ситуацією формуються JSON файл для передачі у тіло запиту. Це особливо важливо при виконанні POST або PUT запитів, коли потрібно передати на сервер певні дані. JSON формат є стандартним та широко використовуваним у веб-розробці, оскільки він забезпечує зручний та зрозумілий спосіб передачі структурованих даних.

Якщо є необхідність у шапці запиту передається токен користувача та робиться звернення до серверу. Токен використовується для аутентифікації та авторизації користувача на сервері, забезпечуючи безпеку передачі даних та доступу до захищених ресурсів. Це дозволяє обмежити доступ до інформації лише авторизованим користувачам та захистити дані від несанкціонованого доступу. Після формування запиту та необхідних даних, робиться звернення до серверу.

Усі класи сервісів мають у собі TRY CATH блоки для перехвату та відслідковування помилок при виконанні звернення або при отриманні відповіді. Це дозволяє обробляти можливі помилки, такі як мережеві збої, помилки серверу або некоректні дані, що можуть бути повернені у відповіді. Обробка помилок допомагає забезпечити стабільну роботу додатку, дозволяючи вчасно реагувати на проблеми та повідомляти користувачів про можливі збої або помилки.

3.2.3 Маршрутизація всередині застосунку

На платформі MAUI існує два основних виду навігації – через вбудовану оболонку Shell та пряму маршрутизацію. В нашому випадку було обрано перший вид навігації через те, що оболонка Shell забезпечує більш гнучку маршрутизацію між сторінками, а також дозволяє створювати довільні маршрути та схеми навігації.

Окрім використання вбудованої оболонки також використовується Router-based Routing підхід, або маршрутизація на основі роутів. У спеціальному файлі AppShell.xaml.cs заздалегідь реєструються всі необхідні маршрути. Кожен маршрут асоціюється з ім'ям сторінки (вказаним за допомогою nameof()) та відповідним класом сторінки (типом). Така реєстрація дозволяє системі маршрутизації знаходити та відображати відповідні сторінки на основі їхніх імен, а також за іменем робити перехід до відповідної сторінки.

Крім того, використання Router-based Routing дає можливість заздалегідь визначити всі необхідні маршрути, що полегшує управління навігацією та спрощує розробку застосунків на платформі MAUI. Такий підхід дозволяє розробникам зосередитися на функціональності проекту, забезпечуючи при цьому зручну та ефективну навігацію для користувачів.

На додаток до цих переваг, також Shell дозволяє легко налаштовувати зручну для користувачів панель навігації та вкладки, що покращує загальний досвід використання застосунку. Завдяки цьому підходу, розробники можуть швидше та ефективніше вносити зміни до навігаційної структури без необхідності суттєвих змін у коді.

3.3 Приклади найцікавіших алгоритмів та методів

3.3.1 Передумови використання алгоритмів штучного інтелекту

З розвитком технологій штучного інтелекту в сучасному суспільстві музична індустрія стала однією з галузей, що активно використовує алгоритми штучного інтелекту для покращення процесу рекомендації музики. Передумови використання цієї технології у музичній сфері прості – розвиток цих самих алгоритмів штучного інтелекту.

Алгоритми штучного інтелекту дозволяють створювати моделі, які можуть навчатися на основі музичних уподобань користувачів і постійно покращувати свої рекомендації. Це відкриває нові можливості для автоматизованого створення плейлистів, пошуку нової музики та відкриття нових артистів, що раніше могли залишатися непоміченими.

Інтеграція алгоритмів штучного інтелекту в музичні платформи також сприяє підвищенню взаємодії користувачів з сервісами, оскільки персоналізовані рекомендації покращують користувацький досвід і збільшують задоволеність слухачів.

3.3.2 Використання у проекті

Штучний інтелект було написано на мові програмування Python з використанням бібліотеки pandas на основі контентної фільтрації даних. На даний момент штучний інтелект представлено лише у виді локального серверу, але потім планується перенесення на хмарові сервіси Azure. Саме мобільний клієнт не взаємодіє зі штучним інтелектом на пряму. Всі робочі дії з мобільного застосунку виконується лише через посередника у вигляді серверної частини. Через HTTP запит ми маємо можливість звернутися до серверу, який в свою чергу звернеться до ШІ та відправить назад до клієнту JSON файл з рекомендаційними треками які можуть сподобатись групі людей. Після цього клієнт отримає файл та відобразить музику.

Штучний інтелект містить у собі дата сет в цілому на 5000 музикальних треків. Всі вони взяті з відкритого доступу та включать в себе різні жанри та види музики. Один елемент дата сету містить у собі такі атрибути: ідентифікатор, назву, ідентифікатор виконавця, назву виконавця, жанр, рік випуску, тривалість, популярність, танцювальність, енергія, тон, гучність, мовлення, акустичність, інструментальність, живість, настрій, темп.

3.3.3 Опис алгоритмів роботи штучного інтелекту

Штучний інтелект застосовує векторну просторову модель для аналізу і надання рекомендацій стосовно треків. Ця модель дозволяє ефективно аналізувати та порівнювати різні елементи, базуючись на їхніх характеристиках. Векторна просторова модель широко використовується в системах рекомендацій на основі контенту, оскільки вона забезпечує високий рівень точності та релевантності рекомендацій.

У цій моделі кожен елемент представлений у вигляді вектора його характеристик у n-вимірному просторі. Характеристики можуть включати різні атрибути, такі як жанр, темп, настрій, виконавець та інші параметри, що описують музичний трек. Кожен з цих атрибутів відповідає певній координаті у векторному просторі, що дозволяє створити унікальний вектор для кожного треку. Ключовим аспектом векторної просторової моделі є обчислення кута між векторами для визначення схожості між ними.

У нашому випадку використовується косинусова подібність. Формула для розрахунку косинусної подібності наведено у вигляді формули 3.1:

$$similarity = \frac{A*B}{||A||*||B||}, \tag{3.1}$$

де A * B – скалярний добуток векторів A і B;

||A|| * ||B|| – їхні норми.

Для векторів $A=(a1,a2,...,an)$ і $B=(b1,b2,...,bn)$, скалярний добуток обчислюється як наведено у формулі 3.2:

$$A \cdot B = a_1 \cdot b_1 + a_2 \cdot b_2 + ... + a_n \cdot b_n, \quad (3.2)$$

де a_n – компонент вектору A;

b_n – компонент вектору B.

Норма вектора A обчислюється як наведено у формулі 3.3:

$$||A|| = \sqrt{a_1^2 + a_2^2 + ... + a_n^2}, \quad (3.3)$$

де a_n – компонент вектору A, наприклад, якщо вектор A представляє собою пісню, то a_1 може бути значенням танцювальності, a_2 - енергійності і так далі.

Косинусна схожість вимірює, наскільки два вектори спрямовані в одному напрямку, що дозволяє визначити ступінь схожості між елементами. Якщо два вектори мають малий кут між собою, це означає, що елементи, які вони представляють, мають високу схожість. Відповідно, якщо кут між векторами великий, схожість між елементами низька. Цей підхід дозволяє системі рекомендацій ефективно визначати найбільш релевантні треки для користувачів, базуючись на їхніх попередніх вподобаннях та характеристиках прослуховуваної музики.

3.4 Створення UI / UX або іншого дизайну системи

Графічний інтерфейс є важливим елементом проекту, тому що саме через інтерфейс застосунку відбувається взаємодія між користувачем та програмою. Мобільний додаток починає роботу з того, що запускається стартова сторінка авторизації. Кожна сторінка повинна, в залежності від її обов’язків, мати елементи управління. Важливо, щоб ці елементи були розташовані логічно та зручно для користувача, з мінімальною кількістю кліків для досягнення потрібної дії.

Прототипування дизайну застосунку проводилося за допомогою онлайн-сервісу Figma. Цей сервіс дозволяє створювати інтерактивні прототипи, які точно відображають вигляд і функціональність кінцевого продукту. У цьому сервісі до кожної сторінки було розроблено відповідний макет з дотримуванням всіх правил user-friendly інтерфейсу. Основні принципи прототипування можливо побачити на прикладі сторінки профілю користувача (див.рис. 3.1).

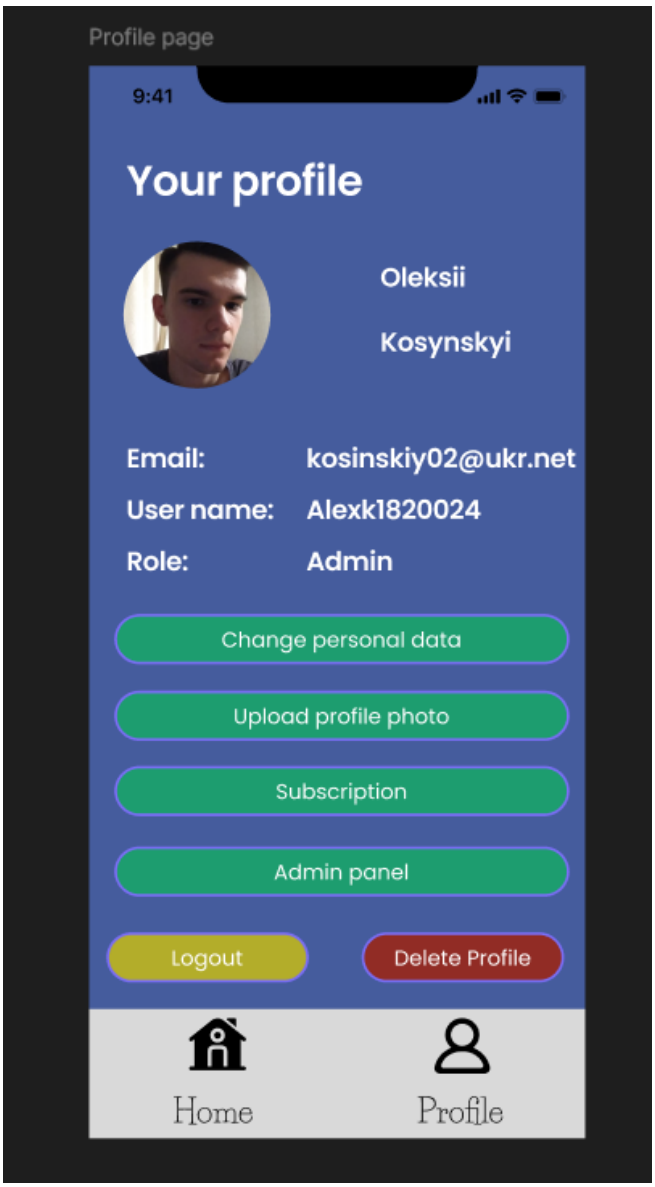


Рисунок 3.1 – Прототип сторінки профілю (рисунок виконаний самостійно)

Прототипування кожної сторінки проводилося з врахуванням стандартної роздільної здатності екрану на мобільних девайсах. При проектуванні використовувалися приємні кольори та іконки для того щоб зробити інтерфейс

застосунку інтуїтивно зрозумілим. Кнопка видалення облікового запису має агресивний червоний колір, та явно виділяється на фоні інших, це теж додає інтуїтивності дизайну. Всі сторінки застосунку виконані у єдиному кольоровому стилі.

Додатково, кожна сторінка має оптимізований дизайн, який дозволяє швидко завантажуватися навіть при низькій швидкості інтернет-з'єднання, що підвищує загальну ефективність використання застосунку. Також на деяких сторінках присутні короткі інструкції щодо використання основних функцій. Ці інструкції , хоча не є необхідністю, забезпечують користувачів необхідною інформацією для ефективного та швидкого виконання завдань. Інструкції є важливим елементом, оскільки вони допомагають новим користувачам швидко орієнтуватися в інтерфейсі та розуміти основні функції програми.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 XAML замість Blazor Hybrid

Перед початком розробки мобільного застосунку в першу чергу слід визначитися з технологіями на яких буде написано проект. При розробці застосунку на платформі MAUI перед розробником виникає питання вибору мови для написання графічного інтерфейс застосунку. На платформі MAUI зазвичай це вибір представлено між використанням Blazor Hybrid та класичної мови розміток XAML.

Blazor Hybrid — це технологія, яка дозволяє розробникам використовувати компоненти Blazor, які в свою чергу використовують компоненти написані на C# і Razor. При розробці графічного інтерфейсу на платформі MAUI мова йде конкретно про фреймворк Razor page. Це дозволяє писати код сторінки Razor Page яка містить як логіку, так і розмітку в одному файлі, що спрощує зрозуміння взаємозв'язку між ними та зменшує кількість файлів в проекті, що сприяє збереженню організації та легкості управління проектом.

Синтаксис фреймворку в основному складається з синтаксису мови розмітки HTML, за винятком інструментів для інтеграції коду C# (див.рис. 4.1).

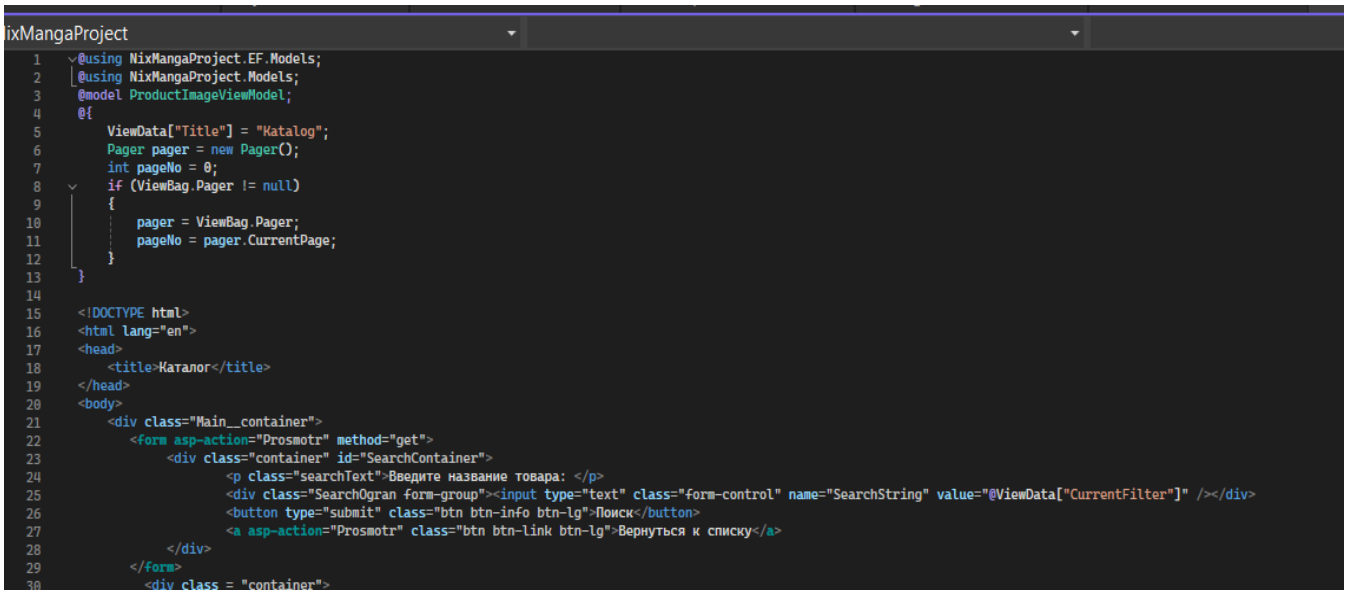


Рисунок 4.1 – Приклад синтаксису сторінки Razor Page (рисунок виконаний самостійно)

Оскільки мова розмітки HTML є однією з найбільш поширених і базових у сфері розробки графічного інтерфейсу, можна припустити, що розробка на цій мові потребує менше часу.

Незважаючи на те, що використання розмітки XAML потребує більш значного розуміння синтаксису мови та потребує більше часу на навчання. Усі сторінки застосунку написані на мові розміток XAML (див.рис. 4.2).

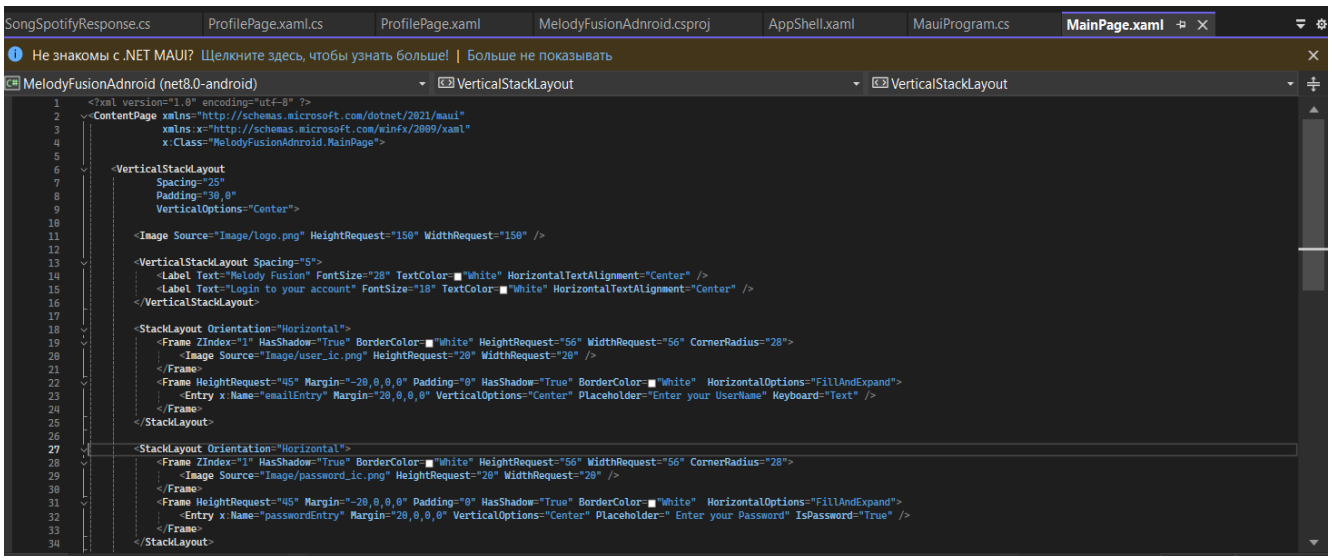


Рисунок 4.2 – Приклад синтаксису мови XAML (рисунок виконаний самостійно)

Вибір цієї мови був обумовлений великою підтримкою стилів та шаблонів, а також можливістю розділення логіки та представлення. Кожному файлу XAML відповідає файл з таким же ім'ям, але з розширенням cs, який містить логіку сторінки. Це полегшує розробку та підтримку проектів, оскільки розмітка та логіка зберігаються в окремих файлах.

Крім звичайної розмітки сторінок, до всіх сторінок застосовуються спеціальні шаблони стилів. Використання цих шаблонів сприяє єдністю в оформленні та забезпечує консистентний вигляд у всьому застосунку. Такий підхід спрощує процес розробки, підтримки та зміни дизайну, оскільки зміни потрібно вносити лише в одному місці, а вони автоматично застосовуються до всіх відповідних елементів інтерфейсу.

Одним із прикладів використання готових елементів розмітки XAML є Tabbar. Даний елемент є практично обов’язковим для всіх мобільних застосунків,

та є мабуть у кожному мобільному сервісі. Він надає розробнику можливість відобразити спеціальну навігаційну шторку у нижній частині мобільного екрану[11]. Цей елемент дає можливість користувачу швидко виконувати навігацію після дотику до іконки меню. Приклад Tabbar можна побачити нижче (див. рис. 4.3):

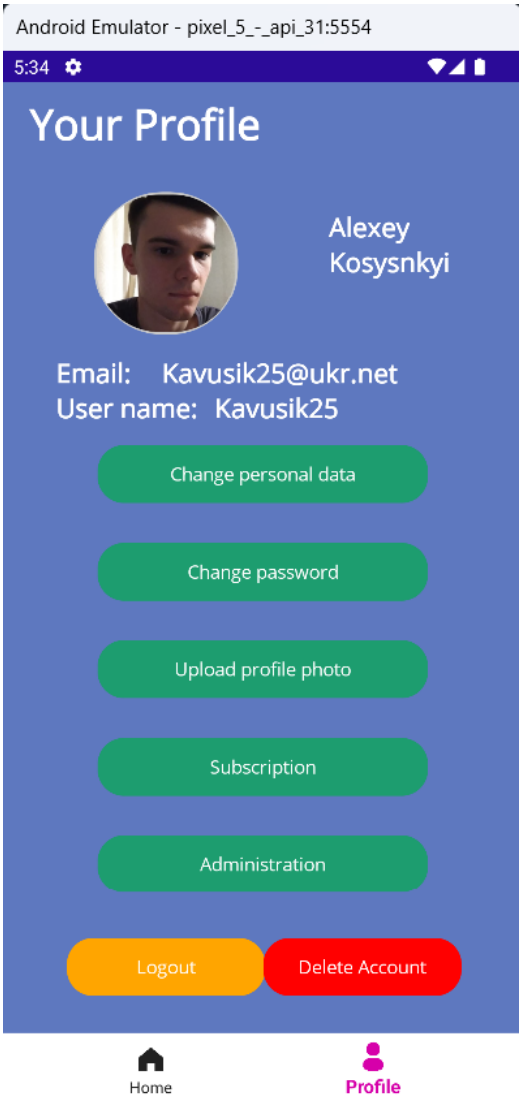


Рисунок 4.3 – Приклад використання елементу Tabbar (рисунок виконаний самостійно)

Використання цього елементу зумовлено бажанням зробити інтерфейс застосунку більш зрозумілим та легким. Саме через цей елемент виконується навігація між розділами. Також на кожний розділ встановлена окрема іконка, що візуально характеризує зміст цього розділу.

4.2 Використання local storage

Наявність можливості авторизації та реєстрації є одним із основних та базових функцій що присутні у програмному застосунку. Для цих механізмів зазвичай використовують Bearer токен.

Bearer токен являє собою рядок символів, який передається в заголовку HTTP-запиту для посвідчення та підтвердження прав доступу користувача до ресурсу або сервісу. Однією з головних переваг використання Bearer токену є спрощення процесу аутентифікації. Користувачу системи не потрібно передавати свої облікові дані при кожному запиті, що підвищує безпеку, оскільки чутлива інформація не передається багаторазово по мережі. Крім того, Bearer токени можуть мати обмежений термін дії, що додатково підвищує безпеку, дозволяючи мінімізувати ризики, пов’язані з компрометацією токенів

Використання Bearer токену для авторизації одразу підіймає питання зберігання цього токену. Для зберігання токену у мобільному застосунку використовується написаний клас «LocalStorage», представлений у просторі імен «MelodyFusionAndroid.Infrastructure», він служить для управління локальним сховищем даних у проекті. Основна мета цього класу — надання зручного інтерфейсу для збереження, вилучення та видалення токену для користувача системи.

Локальне сховище реалізоване на основі структури «Dictionary», що зберігає дані у форматі «ключ-значення». Клас конструктору ініціалізує приватне поле «_dictionary», яке є екземпляром словника. Це поле використовується для зберігання всіх даних, що додаються до локального сховища. Індексатор дозволяє отримати значення, пов'язане з певним ключем, безпосередньо через екземпляр класу, що забезпечує зручний і швидкий доступ до даних.

Методи «Add», «AddOrUpdate» і забезпечують гнучкість при роботі з даними. Перший метод додає нову пару ключ-значення до словника, а другий спочатку перевіряє наявність ключа і, якщо ключ вже існує, видаляє старе значення перед додаванням нового. Це гарантує, що у словнику завжди зберігається актуальна

інформація. Метод «Remove» використовується для видалення даних за ключем, що дозволяє ефективно керувати вмістом локального сховища, звільняючи ресурси та підтримуючи актуальність даних.

Додатково написано статичний клас «LocalStorageKeys», який визначає ключі, що використовуються для зберігання певних даних, таких як токен аутентифікації (AuthToken) і профіль користувача (Profile). Ці константи забезпечують одноманітність використання ключів у різних частинах застосунку, що зменшує ймовірність помилок, пов'язаних з друкарськими помилками або несумісністю імен ключів.

4.3 Прив’язка даних до представлень

Оскільки представлення сторінок та логіка знаходяться в окремих файлах, у розробника не має можливості передавати дані напряму у представлення, як у випадку с Razor Page, коли логіка одразу працює на сторінці представлення та одразу відображає потрібні дані. Верстка на XAML є статичною та для відображення даних моделі потребує іншого підходу.

Для цього на платформі MAUI є окремий механізм прив’язки даних моделі до представлення. Цей механізм забезпечує ефективне зв'язування елементів користувацького інтерфейсу з даними застосунку, що значно спрощує синхронізацію змін між інтерфейсом та даними. Прив'язка даних у MAUI дозволяє встановлювати зв'язок між властивостями об'єктів і елементами інтерфейсу, таким чином, зміни в даних автоматично відображаються в інтерфейсі і навпаки. У застосунку цей механізм використовується у різних випадках , один із таких — відображення облікових даних користувача у розділі профілю.

Існує декілька типів прив'язки даних, у проекті використовується лише одностороння (One-Way). Вона забезпечує передачу даних лише від джерела до приймача. При використані цього механізму зазвичай створюють окрему копію моделі яка має ті ж самі поля як і оригінальна модель. У проекті такі моделі представлені у просторі імен «MelodyFusionAndroid.ViewModels». Окремі моделі даних створюють через головну умову до прив’язки даних — коректна робота

механізму зобов’язує модель реалізовувати інтерфейс «INotifyPropertyChanged», який сповіщає систему прив'язки про те, що значення властивості змінилося.

Важливим аспектом прив'язки даних у MAUI є використання шаблонів даних «Data Templates», які визначають, як саме дані мають відображатися у інтерфейсі. Шаблони даних дозволяють створювати більш гнучкі та настроюванні інтерфейси, що враховують специфіку відображуваних даних. Ще одним корисним інструментом у MAUI є можливість використання колекційних видів «Collection Views», які забезпечують ефективне відображення та управління великими обсягами даних. Колекційні види підтримують різні режими відображення, такі як списки, сітки та каруселі, що дозволяє створювати різноманітні інтерфейси для користувачів.

У проекті «Collection Views» та «Data Templates» використовуються у парі, для того щоб відобразити список рекомендацій які пропонує ШІ та для того щоб красиво оформити відображення цих рекомендацій. Приклад фрагменту коду з використанням цих елементів можливо побачити нижче (див. рис. 4.4):

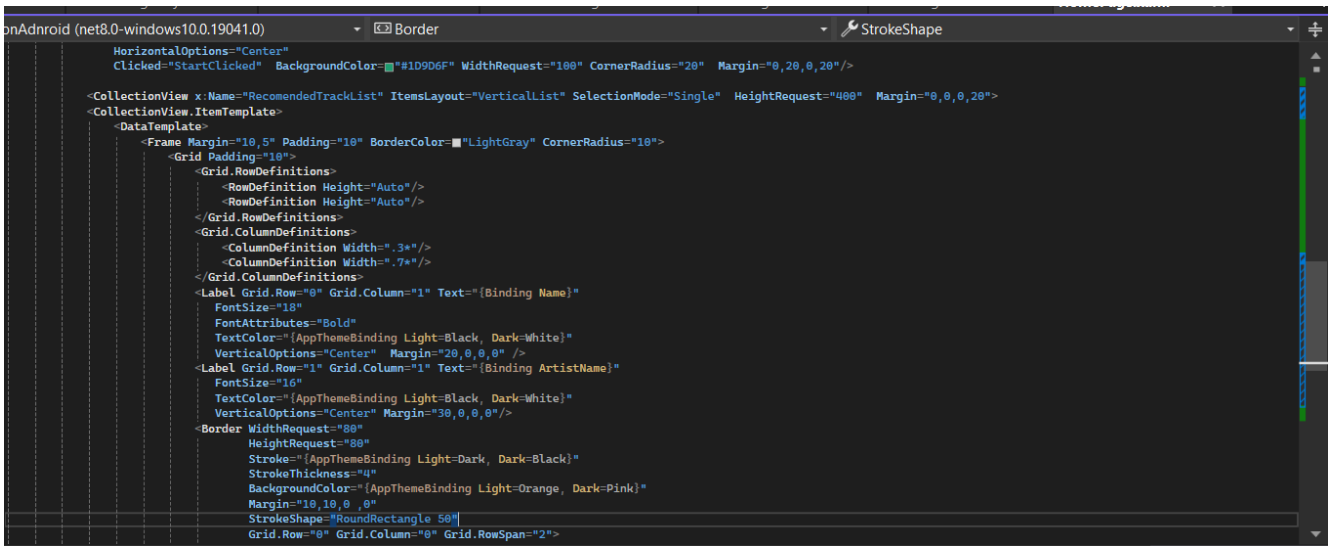


Рисунок 4.4 – Приклад використання «Collection Views» та «Data Templates» (рисунок виконаний самостійно)

При роботі з застосунком прив’язка даних до представлення зазвичай виконується при виконані функцій в результаті дій користувача, але інколи прив’язка виконується у методі «OnAppearing». Даний метод є стандартним та

викликає спрацювання логіки при кожній загрузці сторінки, це потрібно у випадках коли відображення даних повинно відбуватися незалежно від дій користувача системи.

4.4 Валідація даних

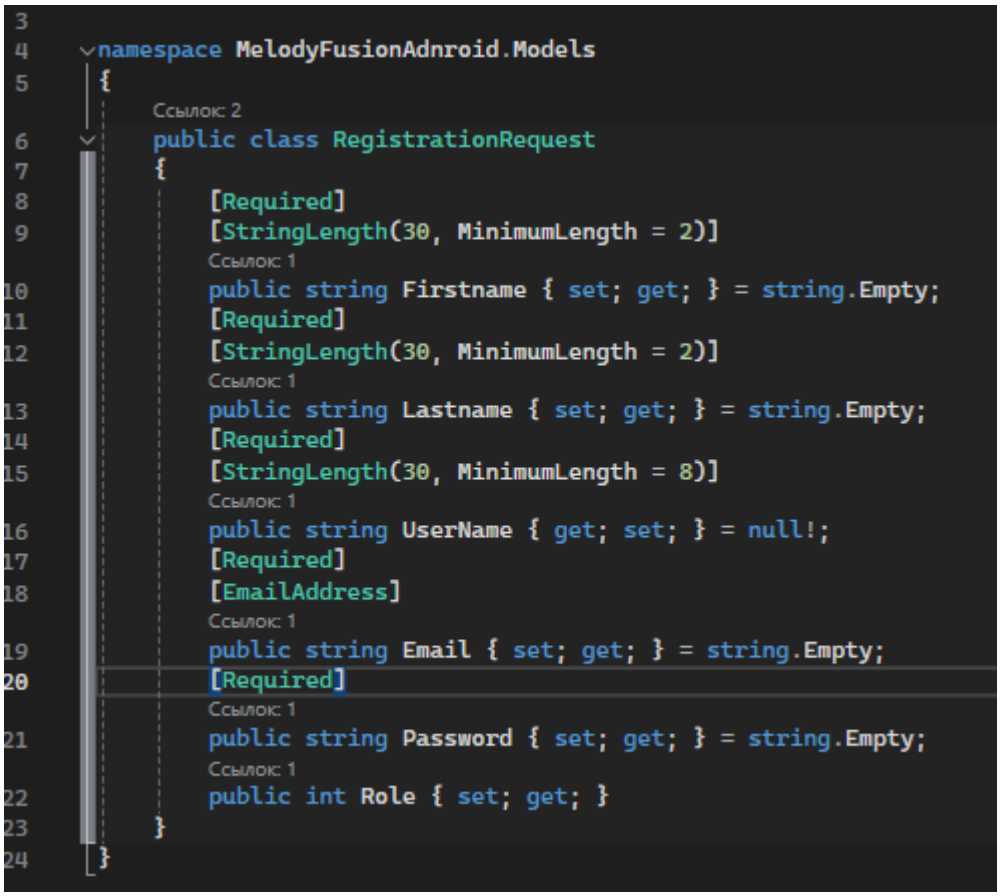
Як вже було описано, мобільний додаток виступає у ролі клієнту для звернення до серверу. Сам сервер ставить деякі умови стосовно формату даних які на нього поступають. Некоректний формат запропонованих даних викликає появу помилок та виводить із роботи додток, тому вкрай важливо дотримуватися прописаних форматів даних.

Для запобігання варіанту припинення роботи системи, вкрай важливо використовувати валідацію кінцевих даних. Валідація даних є критично важливим аспектом розробки застосунків, оскільки вона забезпечує коректність і надійність введених користувачем даних. У платформі MAUI існує кілька підходів до валідації даних, кожен з яких має свої переваги і особливості застосування.

Варто відмітити, що у розробника є можливість використання сторонніх бібліотек, що може значно спростити процес валідації, оскільки вони часто містять готові рішення для широкого спектру задач.

Однак у застосунку використовується один із основних підходів до валідації даних у MAUI, який включає використання атрибутів валідації. Цей підхід дозволяє розробникам задавати правила валідації безпосередньо у моделях даних завдяки спеціальним атрибутам даних. Такі атрибути заздалегідь прописуються у моделях. У фінальному проекті такий підхід використовується але не є основним.

Атрибути валідації, такі як «Required», «StringLength», «Range» та інші, визначають умови, які повинні бути виконані для того, щоб дані вважалися коректними. Використання цих атрибутів є простим у впровадженні та забезпечує автоматичну перевірку даних при їх введенні користувачем. Приклад використання атрибутів валідації у моделі можливо побачити нижче (див. рис. 4.5):



```
3
4   namespace MelodyFusionAdnroid.Models
5   {
        Ссылок: 2
6       public class RegistrationRequest
7       {
8           [Required]
9           [StringLength(30, MinimumLength = 2)]
            Ссылок: 1
10          public string Firstname { set; get; } = string.Empty;
11          [Required]
12          [StringLength(30, MinimumLength = 2)]
            Ссылок: 1
13          public string Lastname { set; get; } = string.Empty;
14          [Required]
15          [StringLength(30, MinimumLength = 8)]
            Ссылок: 1
16          public string UserName { get; set; } = null!;
17          [Required]
18          [EmailAddress]
            Ссылок: 1
19          public string Email { set; get; } = string.Empty;
20          [Required]
            Ссылок: 1
21          public string Password { set; get; } = string.Empty;
            Ссылок: 1
22          public int Role { set; get; }
23      }
24  }
```

Рисунок 4.5 – Приклад використання атрибутів валідації (рисунок виконаний самостійно)

Також для запобігання відправки некоректних значень, вхідні дані додатково перевіряються при виконанні деяких функцій. Наприклад при реєстрації облікового запису у функції що формує та виконує запит до серверу — виконується перевірка вхідних значень на дотримання вимог щодо створення паролю. Якщо пароль не дотримується вимог, запит не буде відправлено на сервер, а користувач отримає сповіщення про некоректний формат даних. При введені коректного формату паролю та проходження перевірки — запит буде сформовано та відправлену на сервер, а користувач отримає відповідне повідомлення, що операція пройшла успішно.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Якісне тестування проекту є однією із причин успіху застосунку на ринку. Тестування програмного забезпечення дозволяє виявити та виправити помилки ще до того як кінцевий продукт буде доступний користувачу. Це в свою чергу відіграє позитивну роль, оскільки добре розроблений та протестований мобільний додаток викликає у користувача позитивні емоції а також сприяє росту кількості нових користувачів продукту.

Процес тестування є багатогранним і включає в себе різноманітні підходи та техніки, що спрямовані на всебічну перевірку функціональності, продуктивності та безпеки програмного забезпечення. Ефективне тестування допомагає запобігти виникненню критичних помилок у фінальній версії продукту, що в свою чергу мінімізує ризики для бізнесу, пов'язані з негативними відгуками та зниженням довіри користувачів. Тестування застосунку проводилося у декілька етапів. При розробці застосунку використовувались два основні підходу до тестування програмного продукту.

5.1 Функціональне тестування

Первинне тестування застосунку буде направлено на перевірку робото спроможності усіх основних функцій та ендпоінтів серверу. Це включає перевірку базової функціональності проекту, коректності роботи API та їх взаємодії з сервером. На цьому етапі важливо впевнитися, що всі ключові компоненти працюють належним чином і відповідають специфікаціям.

Для проведення функціонального тестування всі основні та важливі функції застосунку будуть покриті Unit тестами. Для автоматизації цього процесу будуть використовуватися інструменти, такі як NUnit для Unit тестів та Postman або Swagger для тестування API. Основна мета в використанні саме Unit тестів полягає в ізолювані кожного компоненту, щоб перевірити його поведінку незалежно від інших частин системи. Це дозволяє виявити помилки на ранніх етапах розробки і переконатися, що кожна окрема частина коду працює правильно.

Unit тести, як правило, виконують виклик тестованого методу або функції з певним набором вхідних даних і перевірку результатів на відповідність очікуваним. Якщо результати тесту не відповідають очікуваним, тест вважається провaленим, і розробник отримує інформацію про те, де і в чому полягає проблема.

Тестування відбувається в ізольованому середовищі, де залежності реального світу, такі як бази даних або мережеві сервіси, замінюються мок-об'єктами або стабами, що дозволяє концентруватися на конкретному аспекті функціональності. Приклад написаних тестів можливо побачити нижче (див. рис. 5.1):

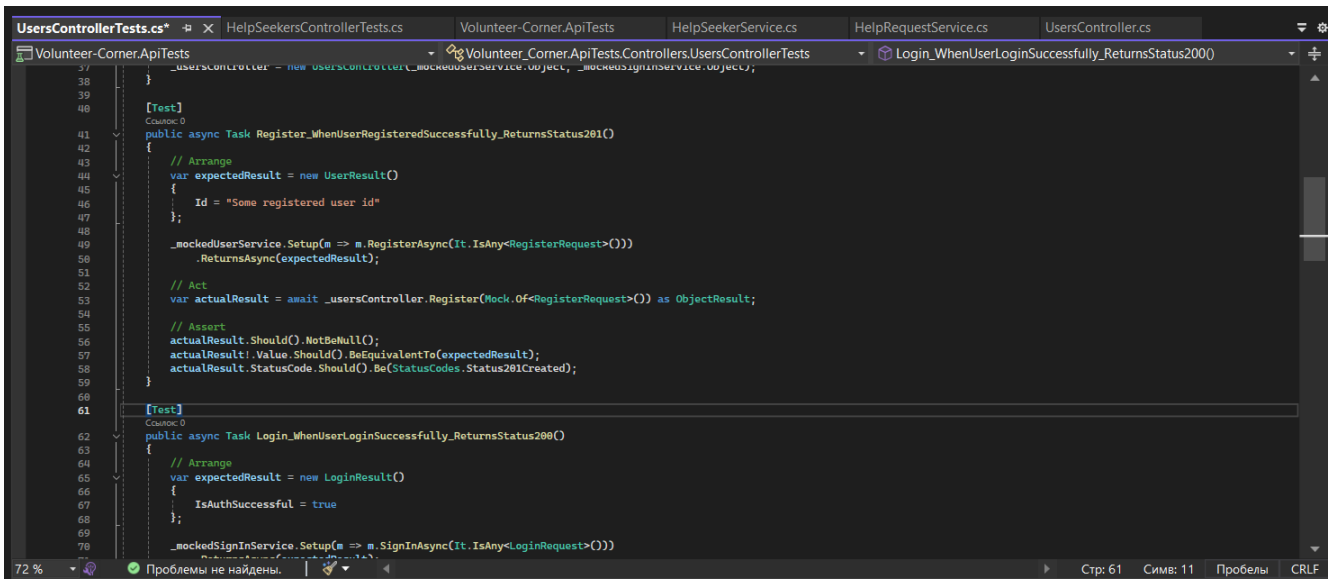


Рисунок 5.1 – Приклад тесту на перевірку функції реєстрації користувача (рисунок виконаний самостійно)

У цьому тесті спочатку встановлюються умови для тестування, також створюється та налаштовується тестовий екземпляр сервісу. Після цього виконується виклик відповідної функції сервісу та перевірка з умовами заданими розробником тесту. На цей раз перевіряється, що відповідь не є нульовим значенням, що її значення еквівалентне очікуваному результату, а також перевіряється, що статус код відповіді дорівнює 201.

5.2 Мануальне тестування

Цей вид тестування хоча і є більш трудомістким та часозатратним, але надає можливість тестувальнику швидко змінювати свій підхід у відповідь на нові вимоги або виявлені проблеми, що забезпечує динамічний процес тестування. Мануальне тестування відноситься до вторинного тестування та проводиться після завершення функціонального тестування. Незважаючи на всі недоліки є теж необхідним у проекті.

Процес мануального тестування починається з розробки тестових випадків, які описують кроки, необхідні для виконання тесту, а також очікувані результати. Тестові випадки зазвичай базуються на специфікаціях вимог до програмного забезпечення, щоб переконатися, що всі функціональні аспекти системи перевірені. Після створення тестових випадків тестувальник запускає мобільний додаток та починає їх виконувати, ретельно дотримуючись описаних кроків і фіксуючи будь-які відхилення від очікуваних результатів. При тестуванні зазвичай перевіряють коректність роботи користувацького інтерфейсу, маршрутизацію між сторінками а також роботу функцій та результат передачі даних у представлення. Мануальне тестування виходить за рамки тестування функціоналу та обмежується лише фантазією розробника та його бажаннями.

ВИСНОВКИ

Під час роботи над кваліфікаційною роботою було проаналізовано предметну галузь, визначено основу проблему, проаналізовано конкурентів та сформовано основні вимоги до кінцевого продукту. Впродовж виконання роботи було реалізовано та спроектовано архітектуру мобільного застосунку, а також визначено усі необхідні вимоги до його розробки та кінцевого результату. Зокрема, цей процес також включав у себе проектування та написанні графічного інтерфейсу застосунку та його тестування у декілька етапів. Також особливу увагу роботи було приділено проектуванню та розробці штучного інтелекту. За час роботи були здобуті практичні навички стосовно написання то супроводу мобільних застосунків, а також навички стосовно роботи з зовнішнім сервером.

Важливо сказати, що мобільний додаток має шанс на успіх, оскільки вирішує питання, котре не було напряму вирішено конкурентами та іншими додатками. Ця система може створювати персоналізовані рекомендації на основі декількох інтересах, що відповідають їхнім унікальним музичним вподобанням. Крім того, завдяки постійному вдосконаленню алгоритмів штучного інтелекту та аналізу даних, такий мобільний додаток може надавати користувачам актуальні музичні рекомендації які дійсно їм сподобаються, що підвищує задоволення від використання застосунку та забезпечує йому конкурентну перевагу на ринку музичних платформ.

У висновку можна сказати, що мобільний додаток успішно спроектований та готовий для подальшого застосування. Його розробка та проектування мали за собою мету випуску унікального та корисного застосунку. Саме тому можна впевнено сказати, що розроблений мобільний додатокМ стане гарним рішенням для осіб, які хочуть знайти музику, яка би сподобалась групі людей.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Li, X., Geng, S., Zhang, Z., & Wu, F. (2015). A survey of music recommendation systems. Journal of Intelligent Information Systems, 35(2), 333-395. doi:10.1007/s10844-014-0323-9.

2. Celma, Ò. (2010). Music recommendation and discovery in the long tail. In E. Bernardo (Ed.), Recommender Systems Handbook (pp. 569-592). Springer. doi:10.1007/978-0-387-85820-3_17.

3. Cantador, I., Brusilovsky, P., & Kuflik, T. (2011). Second Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec2011). In Proceedings of the Fifth ACM Conference on Recommender Systems (RecSys '11). doi:10.1145/2043932.2043991.

4. McNee, S. M., Riedl, J., & Konstan, J. (2006). Being accurate is not enough: How accuracy metrics have hurt recommender systems. In CHI '06: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 1097-1101). doi:10.1145/1124772.1124944.

5. Date Night Movies. https://datenightmovies.com/. URL: https://datenightmovies.com/ (date of access: 27.04.2024).

6. Pohle, J., & Höchstötter, N. (2007). Hybrid music recommendation with experts and wisdom of the crowd. In Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR '07).

7. Lamere, P., Eck, D., & Turnbull, D. (2008). Social tags and music information retrieval. In Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR '08).

8. McFee, B., & Lanckriet, G. R. G. (2012). The natural language of playlists. In Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR '12).

9. What is .NET MAUI? - .NET MAUI. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui?view=net-maui-8.0 (date of access: 14.05.2024).

10. Localization - .NET MAUI. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/dotnet/maui/fundamentals/localization?view=net-maui-8.0 (date of access: 14.05.2024).

11. .NET MAUI Shell tabs - .NET MAUI. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/dotnet/maui/fundamentals/shell/tabs?view=net-maui-8.0 (date of access: 14.05.2024).

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

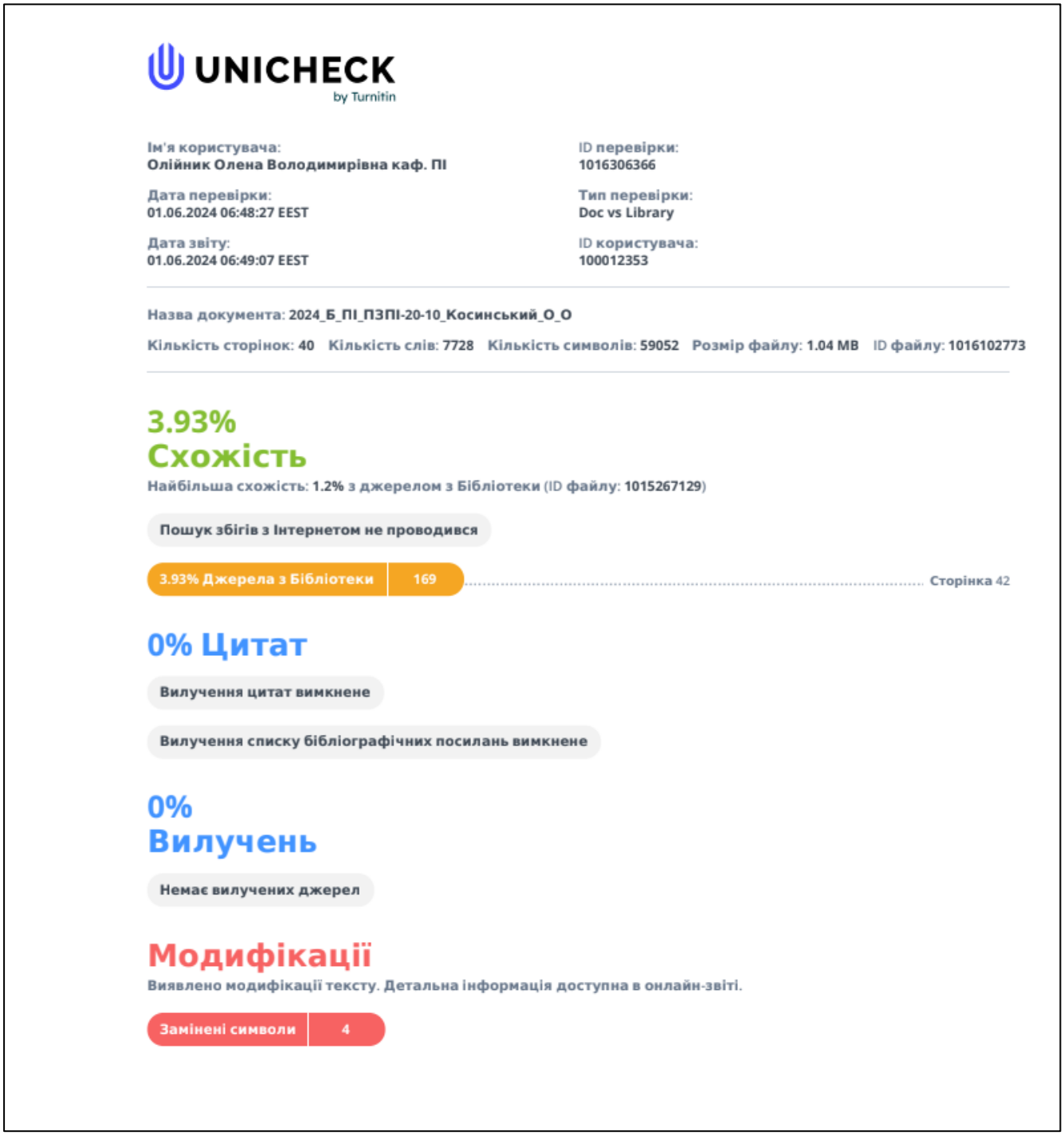


Рисунок А.1 – Результат перевірки на унікальність тексту

ДОДАТОК Б

Слайди презентації

Харківський національний університет радіоелектроніки
Кваліфікаційна робота бакалавра

Програмна система рекомендацій на основі вподобань групи користувачів. Мобільний додаток.

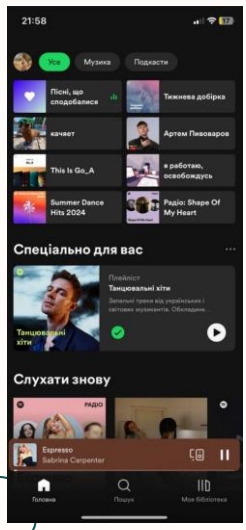
Виконав: студент 4 курсу ПЗПІ -20-10 Косинський О.О
Керівник: доц. кафедри ПІ Афанасьєва І. В.

Мета

Створення мобільного додатку, з використанням технологій штучного інтелекту, для надання можливості формування рекомендацій музики на основі вподобань декількох користувачів

Об'єкт розробки: Програмна система рекомендацій на основі вподобань групи користувачів. Мобільний додаток

Аналіз предметної області



Spotify

У наш час, зі збільшенням кількості споживаного людиною контенту, виникає необхідність у створенні рекомендаційних систем, які могли б обробляти цей контент і рекомендувати найбільш відповідний для користувача. На жаль, не існує універсальної системи, яка б рекомендувала контент, який безумовно сподобався б користувачеві, а також на ринку існує не дуже багато систем, які би враховували інтереси декількох користувачів

Date night movie

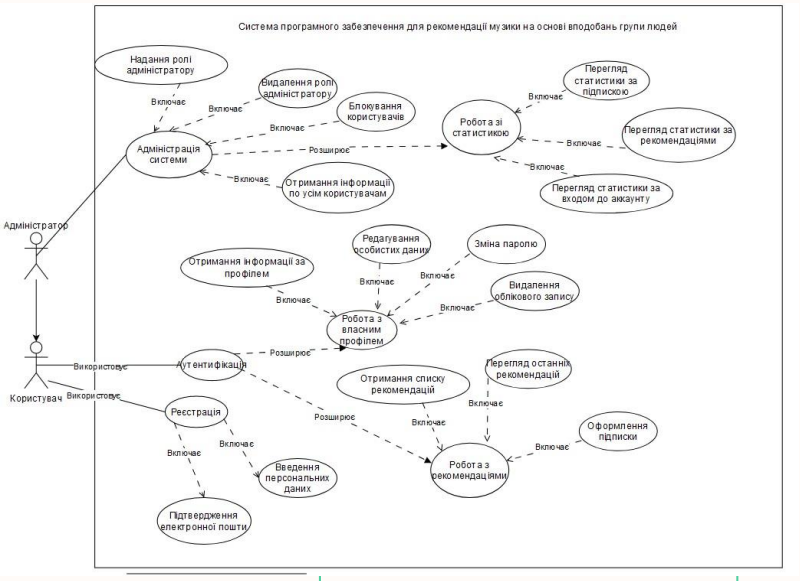
Постановка задачі

- Аналіз вимог та конкурентів
- Формування технічного завдання
- Проектування та прототипування системи
- Розробка алгоритму штучного інтелекту для підбору треків
- Створення користувацького інтерфейсу
- Розробка мобільного додатку
- Тестування мобільного додатку

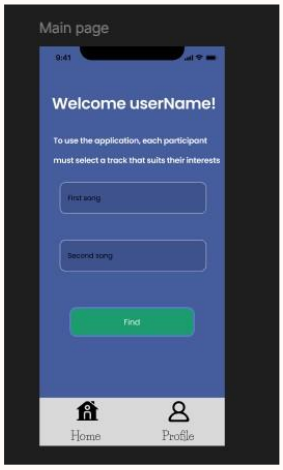
Моделювання додатку

Адміністратор

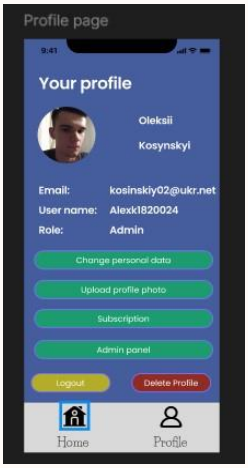
Користувач



Прототипування додатку



Прототип головної сторінки



Прототип профілю користувача

Алгоритми підбору музики

Визначення схожості між представленими треками(векторами) користувачів

$$similarity = \frac{A * B}{||A|| * ||B||},$$

де A * B – скалярний добуток векторів A і B

||A|| * ||B|| – їхні норми

Для векторів $A=(a1,a2,\ldots,an)$ і $B=(b1,b2,\ldots,bn)$, скалярний добуток обчислюється

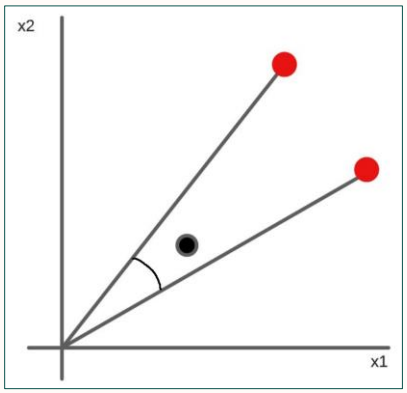
$$A \cdot B = a1 \cdot b1 + a2 \cdot b2 + \ldots + an \cdot b_n,$$

де a*n* – компонент вектору A

b_n – компонент вектору B

7

Алгоритми підбору музики



Норма вектора A обчислюється

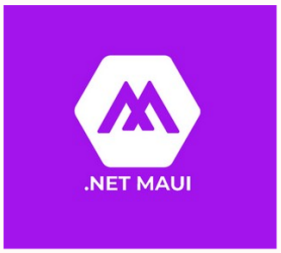
$$||A|| = \sqrt{a_1^2 + a_2^2 + \ldots + a_n^2}$$

де a_n – компонент вектору A.

З графіку видно, що якщо кут великий, то два елементи відрізняються один від одного, а якщо кут малий, то два елементи схожі один на одного.

8

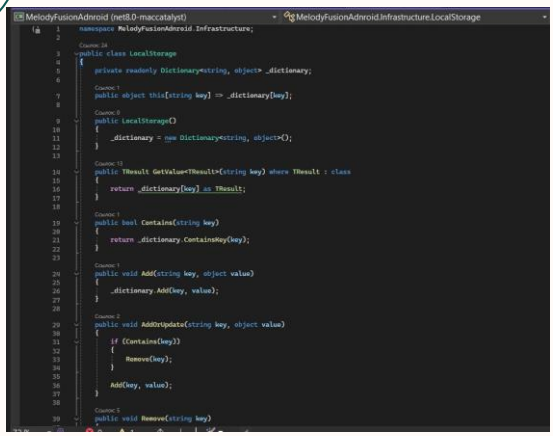
Інструменти та технології



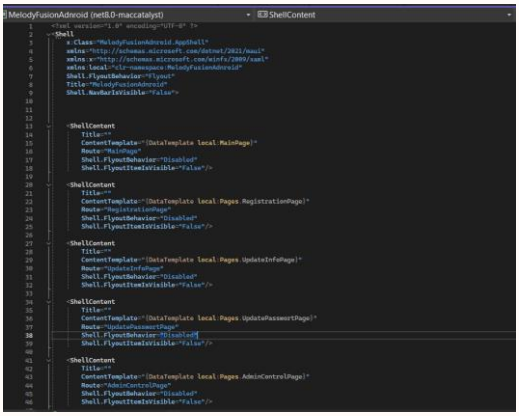
Інтеграція з зовнішніми сервісами

Braintree
A PayPal Service

Прийняті програмні рішення



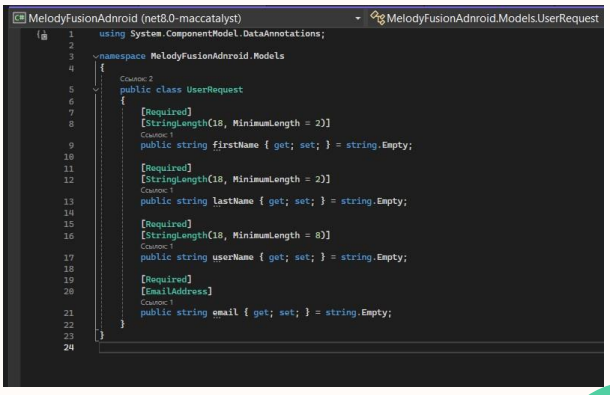
Використання LocalStorage для зберігання JWT токену користувача



Використання оболонки Shell для маршрутизації

Прийняті програмні рішення

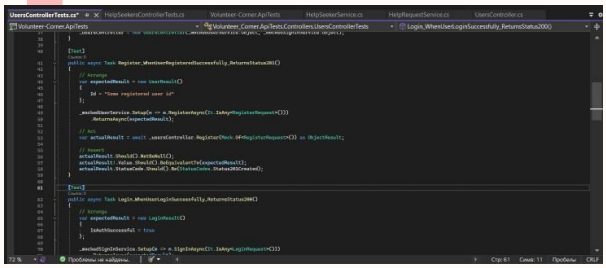
Використання htppClientFactory для звертання до серверу та відправки запитів



Використання атрибутів валідації для моделей

Тестування мобільного додатку

Функціональне:



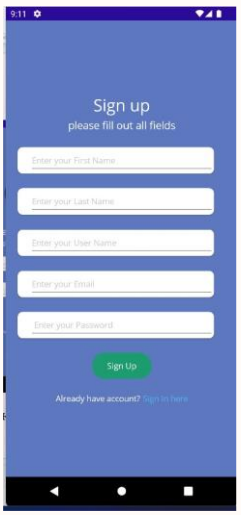
Написані Unit тести для перевірки функцій додатку

Мануальне:

- Формування тестових випадків
- Покрокове відтворення тестових випадків
- Формування та аналіз результату

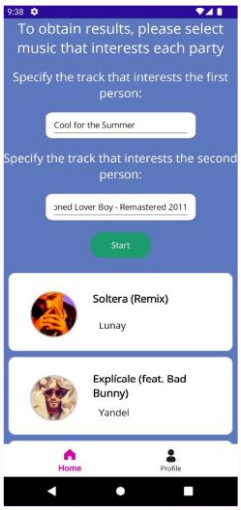
Інтерфейс додатку

Сторінка авторизації

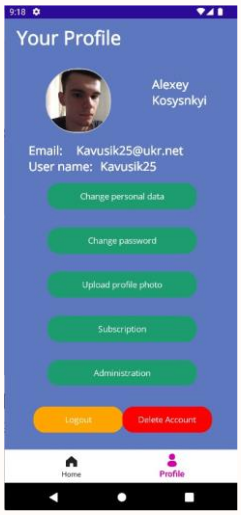


Сторінка реєстрації

Інтерфейс додатку



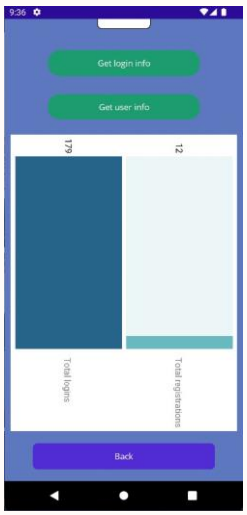
Головна сторінка



Сторінка профілю

Інтерфейс додатку

Адмін панель



Сторінка статистики

Висновки

- У ході кваліфікаційної роботи було сформоване технічне завдання та виконане прототипування системи.
- Було створено алгоритм штучного інтелекту для роботи системи, а також розроблено дизайн користувацького інтерфейсу
- За час роботи було розроблено мобільний додаток програмної системи рекомендації музики на основі вподобань групи користувачів, а також проведено все необхідне тестування написаного застосунку

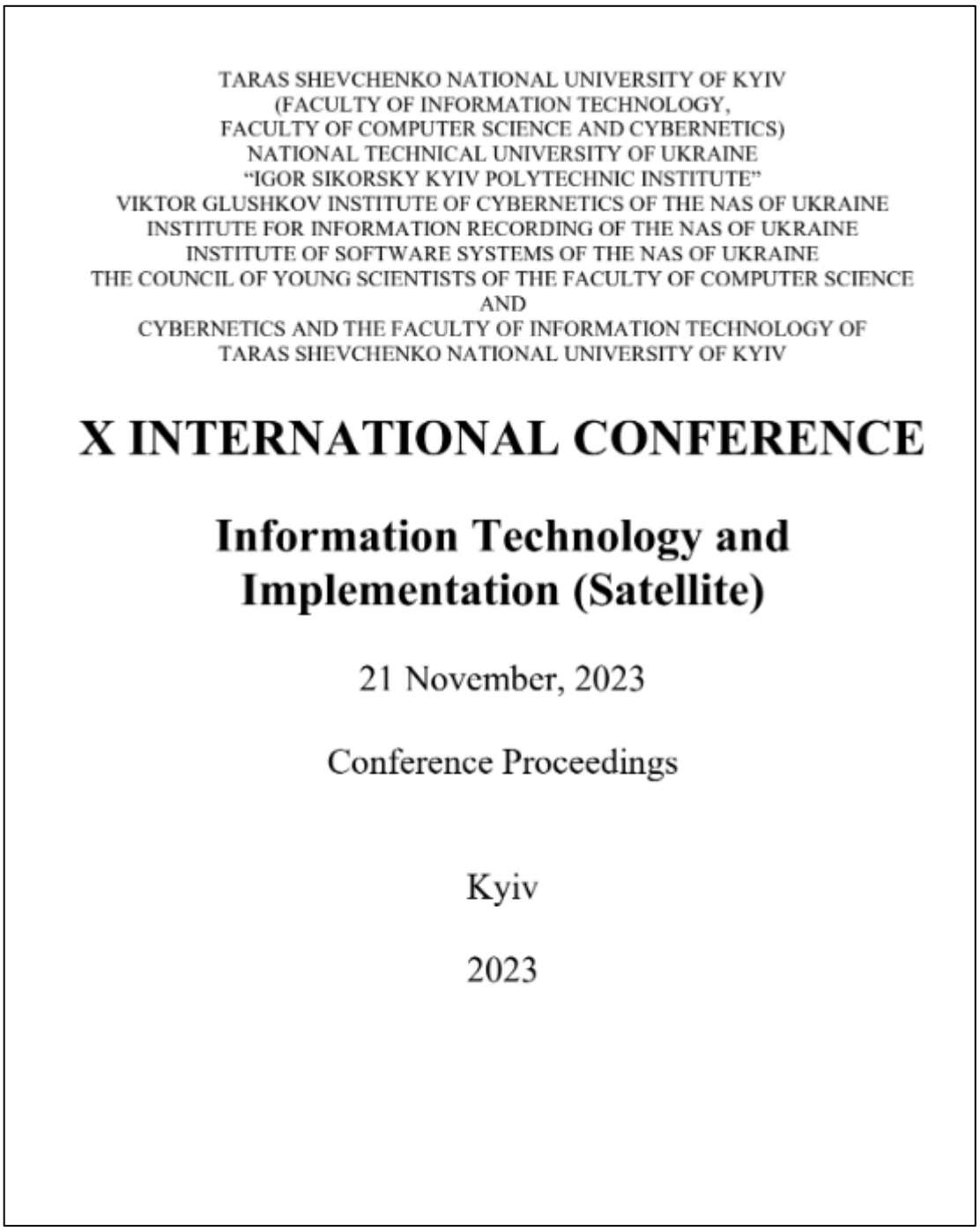
Збірка тез доповідей

Kosynskyi O., Afanasieva I. ANDROID APPLICATION FOR MUSIC RECOMMENDATION BASED ON GROUP INTERESTS. *X INTERNATIONAL CONFERENCE Information Technology and Implementation (Satellite)*, KYIV, 21 November 2023. 2023. P. 23–24.

Дякую за увагу!

ДОДАТОК В

Апробація результатів роботи



TARAS SHEVCHENKO NATIONAL UNIVERSITY OF KYIV
(FACULTY OF INFORMATION TECHNOLOGY,
FACULTY OF COMPUTER SCIENCE AND CYBERNETICS)
NATIONAL TECHNICAL UNIVERSITY OF UKRAINE
"IGOR SIKORSKY KYIV POLYTECHNIC INSTITUTE"
VIKTOR GLUSHKOV INSTITUTE OF CYBERNETICS OF THE NAS OF UKRAINE
INSTITUTE FOR INFORMATION RECORDING OF THE NAS OF UKRAINE
INSTITUTE OF SOFTWARE SYSTEMS OF THE NAS OF UKRAINE
THE COUNCIL OF YOUNG SCIENTISTS OF THE FACULTY OF COMPUTER SCIENCE
AND
CYBERNETICS AND THE FACULTY OF INFORMATION TECHNOLOGY OF
TARAS SHEVCHENKO NATIONAL UNIVERSITY OF KYIV

X INTERNATIONAL CONFERENCE

Information Technology and Implementation (Satellite)

21 November, 2023

Conference Proceedings

Kyiv

2023

Рисунок В.1 – Титульна сторінка конференції

Oleksii Kosynskyi, Student of Software Engineering Department
Iryna Afanasieva, PhD, Associate Professor of Software Engineering Department
Kharkiv National University of Radioelectronics.

ANDROID APPLICATION FOR MUSIC RECOMMENDATION BASED ON GROUP INTERESTS

Abstract

Coming from an individual experience, listening to music has become social. But how do you find the right music for a group of people? The solution is a content-based music recommendation android application. This paper raises the issue of developing an Android application that uses artificial intelligence to analyze user profiles and musical attributes in order to provide a group of people with music that suits their common interests. This helps people get closer to music that suits their tastes, improving the overall music experience.

Keywords: music, application, Android, MAUI, Artificial intelligence, Personalized Recommendations.

Music is one of the popular forms of art. Some people listen to it to relax, some to cheer up, and some just to enjoy the beautiful melodies. Over time, listening to music ceased to be something individual and began to take on a group character, people listen to music with friends through the same headphones, or play their favorite music at a party through a speaker. With the first appearance of information output devices, people increasingly began to view listening to music not as a hobby, but as an entire art that requires a detailed selection of the tracks to listen to. But how do you find the music you really like? Or, even more difficult, how to find music that matches the preferences of several people?

To solve this problem, it is necessary to develop an android application that could recommend music based on the interests of a group of people [1]. Since the volume of data processed can be very large, the best solution in this situation is to use artificial intelligence technologies to predict results based on several different factors. For the described application, a content-based filtering approach will be applied [2]. In this way, the android application will use keywords and attributes assigned to objects in the database (for example, genres of frequently listened to music) and match them with the user's profile. These profiles cover database items that users have interacted with, including items they have purchased, browsed, read, watched, or listened to, as well as their associated characteristics [4]. This approach helps establish the level of importance since not all attributes of an element have the same importance to the user. Based on weighting attributes and user interaction history, the artificial intelligence creates a unique preference model for each user. This model includes the attributes that a user is likely to prefer, or dislike based on his previous actions, taking into account their importance. User models are then compared to all items in the database, and these items are subsequently assigned a score based on their similarity to the user's profile. One of the key success factors is adaptation of the application to mobile devices. In the context of developing a music recommendation application, priority will be given to the Android platform, as this provides excellent tools for realizing high-quality music experiences.

Рисунок В.2 – Перша сторінка тез

The usage of the MAUI framework is justified not only by the ability to create cross-platform applications, but also to the fact that it provides effective interaction with Android [3]. This allows to expand audience reach by providing access to music recommendations anytime, anywhere. The MAUI platform, combined with the powerful tools of Android, will create an intuitive and attractive user interface that will significantly improve the overall music experience.

While working with the android application, users can select several tracks, which will be sent to the server in JSON format, after which all material will be processed. The artificial intelligence will select a list of music tracks that would appeal to all users. It is important to note that the android application will not be able to work as a player to listen to the received music, the user must go to specialized applications. An Internet connection is also required for the application to work.

Thus, the choice of the MAUI framework for developing is justified not only by its technical advantages, but also by its ability to effectively combine with the capabilities provided by Android to create an innovative and satisfying product. In summary, a content-based music recommendation android application offers an approach to bring people closer to music that truly suits their unique tastes or the collective preferences of a group, improving the overall music listening experience.

References:

1. Celma, Ò. (2010). Music recommendation and discovery in the long tail: Implications for Android applications. In E. Bernardo (Ed.), Recommender Systems Handbook (pp. 569-592).
2. Cantador, I., Brusilovsky, P., & Kuflik, T. (2011). Second Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec2011): Exploring Android-centric Approaches. In Proceedings of the Fifth ACM Conference on Recommender Systems (RecSys '11).
3. Microsoft Docs. (2023). .NET MAUI documentation: Building cross-platform mobile apps. Retrieved from https://docs.microsoft.com/en-us/dotnet/maui/
4. McNee, S. M., Riedl, J., & Konstan, J. (2006). Being accurate is not enough: How accuracy metrics have hurt recommender systems. In CHI '06: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 1097-1101). doi:10.1145/1124772.1124944.

Рисунок В.3 – Остання сторінка тез