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. Front-end
(тема)

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

Гладкий С. С.
(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 Гладкому Сергію Сергійовичу
(прізвище, ім’я, по батькові)

1.Тема роботи Програмна система рекомендацій музики на основі вподобань групи користувачів. Front-end

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 front-end частину програмної системи рекомендацій музики на основі вподобань групи користувачів застосовуючи мовами програмування Angular та TS.

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 _____ Гладкий С. С.
(підпис)

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

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, 66 стор., 24 рис., 1 табл., 10 джерел.

КЛІЄНТСЬКА ЧАСТИНА, МУЗИКА ДЛЯ ГРУПИ, ПІДБІР МУЗИКИ, РЕКОМЕНДАЦІЙНА СИСТЕМА, ШТУЧНИЙ ІНТЕЛЕКТ, ANGULAR

Об’єкт розробки – програмна система рекомендацій музики на основі вподобань групи користувачів.

Мета розробки – створення веб застосунку для рекомендації музики.

Метод рішення – середовище розробки WebStorm, використані мови програмування (Angular V.17.2, TypeScript, для розробки штучного інтелекту використано середовище PyCharm з мовою програмування Python V.3.12).

У процесі було спроектовано та розроблено продукт, який дозволяє зменшити час на рутинний підбір музики та ефективно прискорити вибір музики.

CLIENT SIDE, GROUP MUSIC, MUSIC SELECTION, RECOMMENDATION SYSTEM, ARTIFICIAL INTELLIGENCE, ANGULAR

Development object – a music recommendation software system based on user preferences.

The development goal is to create a web application for music recommendations.

Solution method – WebStorm development environment, programming languages used (Angular V.17.2, TypeScript). For artificial intelligence development, PyCharm environment with Python V.3.12 was utilized.

In the process, a product was designed and developed to reduce time spent on routine music selection and efficiently accelerate music choices.

Я, Гладкий Сергій Сергійович, студент гр. ПЗПІ-20-10, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Програмна система рекомендацій на основі вподобань групи користувачів. Front-end», 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

GDPR – General Data Protection Regulation

HTTP – HyperText Transfer Protocol

ICF – Item-based Collaborative Filtering

NLP – Natural Language Processing

TS – TypeScript

UCF – User-based Collaborative Filtering

WCAG – Web Content Accessibility Guidelines

ВСТУП

У сучасному цифровому світі зростає значущість персоналізованих рекомендаційних систем у різних сферах, включаючи музику. Із збільшенням обсягу музичного контенту в Інтернеті стає складніше користувачам знаходити ті пісні, які вони шукають або які їм сподобаються. Відповідно, рекомендаційні системи, що базуються на вподобаннях користувачів, набувають великого інтересу.

Цей проект зосереджений на розробці програмної системи, яка забезпечить ефективне рекомендування музики для користувачів на основі їхніх вподобань та інтересів. Основна мета полягає у створенні інтуїтивно зрозумілої та легко доступної системи, що забезпечить користувачам персоналізований підбір музичного контенту, спрощуючи процес пошуку і вибору нових аудіозаписів.

Даний проект передбачає створення Front-end частини програмної системи, що буде відповідальною за інтерактивне спілкування з користувачем, створення зручного інтерфейсу, який дозволить користувачам зручно взаємодіяти з функціоналом рекомендаційної системи.

Цей проект має практичне значення, оскільки спрямований на полегшення пошуку та вибору музичного контенту для широкого кола користувачів. Він не лише дозволить користувачам знайти нову музику, а й покращить їхнє загальне враження від використання музичних платформ.

Таким чином, розробка Front-end частини рекомендаційної системи музики є важливим кроком у покращенні користувацького досвіду та наданні персоналізованих музичних рекомендацій, що відповідають індивідуальним вподобанням кожного користувача.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Музика є потужним засобом комунікації та вираження емоцій, яка здатна об'єднувати людей різних культур та національностей. Вона виконує важливу роль у суспільстві, сприяючи обміну музичними враженнями та посиленню емоційних зв'язків між людьми. Однак уподобання до музики в різних групах користувачів можуть значно відрізнятися.

У сучасному світі музична індустрія стала важливим напрямком для бізнесу та заробітку. Компанії активно розробляють і використовують стрімінгові платформи для прослуховування музики, кожна з яких має власний алгоритм рекомендацій, що робить їх привабливішими для користувачів. Система рекомендацій є ключовою складовою сучасних музичних платформ. Вона створюється за допомогою різних алгоритмів та моделей машинного навчання, що прогнозують вподобання користувачів та надають персоналізовані пропозиції щодо музичного контенту.

Основні з них включають колаборативну фільтрацію, де User-based Collaborative Filtering (UCF) аналізує схожість між користувачами на основі їхніх вподобань і рекомендує музику, яку слухали подібні користувачі, а Item-based Collaborative Filtering (ICF) визначає схожість між музичними треками на основі вподобань користувачів і пропонує схожі треки. Також, використовується контентне фільтрування, яке рекомендує музику на основі атрибутів треків (жанр, виконавець, темп, тональність тощо) і вподобань користувача, застосовуючи методи, такі як обробка природної мови (NLP) для аналізу текстів пісень або метаданих. Гібридні методи комбінують елементи колаборативної та контентної фільтрації для покращення точності рекомендацій.

Рекомендаційні системи збирають і аналізують різноманітні дані про вподобання користувачів, що дозволяє створювати персоналізовані рекомендації. Основні джерела даних включають історію прослуховування, де зберігається

інформація про всі треки, які користувач прослухав, включаючи тривалість прослуховування та кількість повторів, лайки та дизлайки, які допомагають алгоритмам краще розуміти вподобання користувачів, шейри та плейлісти, дані про те, які треки користувачі діляться з іншими або додають до своїх плейлістів, коментарі та відгуки, які можуть бути проаналізовані за допомогою методів обробки природної мови для виявлення позитивного або негативного ставлення до певних треків або виконавців, а також взаємодії з іншими користувачами, такі як підписка на плейлісти друзів.

Прогнозуючи майбутні тенденції в розвитку рекомендаційних систем, можна виділити кілька ключових напрямків. Використання більш складних моделей штучного інтелекту, таких як глибоке навчання та нейронні мережі, допоможе покращити точність та персоналізацію рекомендацій. Розвиток інтерактивних та голосових інтерфейсів, таких як голосові асистенти, сприятиме більш природній та зручній взаємодії користувачів з музичними платформами. Використання сенсорних даних та технологій розпізнавання емоцій дозволить створювати рекомендації, які відповідають настрою або активності користувача.

Отже, предметна область даної кваліфікаційної роботи об'єднує концепції рекомендаційних систем, інформаційних технологій та музичної індустрії з метою забезпечення ефективного процесу рекомендації музики для користувачів у сучасному цифровому середовищі. Інтеграція сучасних технологій і методів штучного інтелекту дозволяє створювати більш точні і персоналізовані музичні рекомендації, покращуючи користувацький досвід і сприяючи розвитку музичної індустрії. Майбутній розвиток рекомендаційних систем обіцяє ще більшу інтеграцію з іншими аспектами цифрового життя, що відкриває нові можливості для інновацій і зростання в цій сфері.

1.2 Виявлення та вирішення проблем

На сьогоднішній день існує кілька платформ для музичних рекомендацій, проте більшість з них зосереджені на індивідуальних вподобаннях користувачів і

не забезпечують систему рекомендацій для групи користувачів. Наприклад, Spotify [1] та Apple Music використовують алгоритми рекомендацій, що аналізують історію прослуховування кожного користувача для персоналізації рекомендацій.

Однак ці системи не завжди ефективно враховують вподобання користувачів у групі, такі як друзі або родина. Існуючі платформи здебільшого використовують алгоритми UCF та ICF, які добре працюють для індивідуальних рекомендацій, але мають обмеження, коли справа доходить до врахування колективних вподобань.

TasteDive [2] є ще однією платформою, яка надає рекомендації на основі смакових уподобань користувачів, але вона також складна для використання у груповому контексті

Основною відмінністю моєї програмної системи буде зосередження на рекомендаціях, що базуються на колективних вподобаннях групи користувачів. Вона надасть можливість користувачам створювати групи і спільно формувати рекомендації музики, враховуючи інтереси всіх учасників. Така система стане важливим інструментом для спільного відкриття нової музики та зміцнення соціальних зв'язків через спільний музичний досвід.

Для реалізації цієї системи планується використовувати гібридні алгоритми, які поєднують колаборативну фільтрацію з методами агрегування вподобань усіх членів групи, що дозволить створювати більш релевантні групові рекомендації.

Крім того, система буде інтегрована з популярними музичними платформами, що забезпечить доступ до широкого спектру музичних треків. Окрім основного функціоналу, передбачається додавання можливості оцінки рекомендації всередині групи, що сприятиме більш активній взаємодії між користувачами.

На українському ринку поки що немає програмних систем, які надають персоналізовані музичні рекомендації на основі групових вподобань користувачів. Натомість, на міжнародному ринку вже існують подібні додатки, наприклад, система рекомендацій фільмів для пар «Date Night Movies». (див. рис. 1.2).

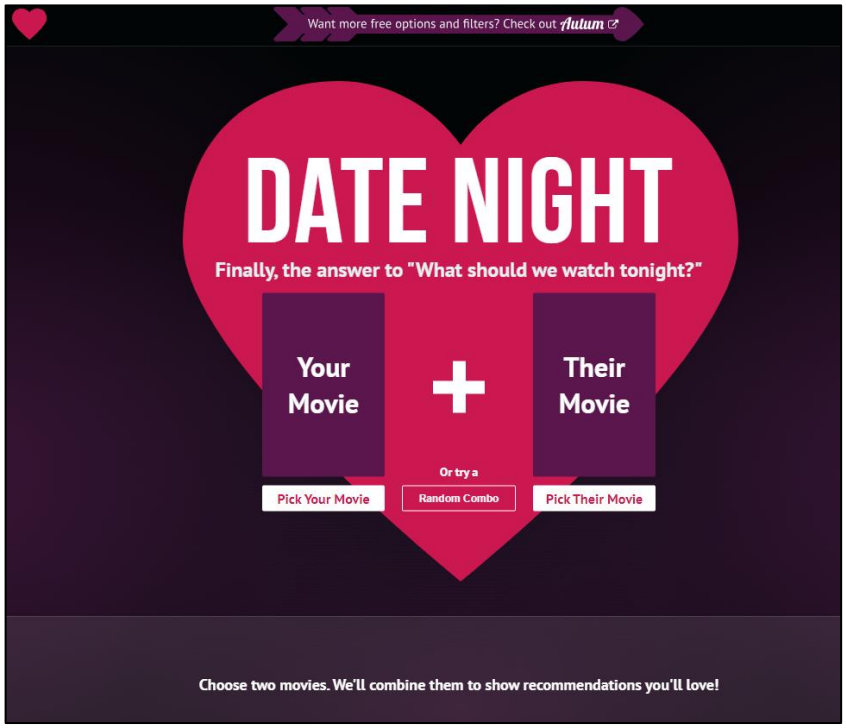


Рисунок 1.2 – Система рекомендацій фільмів для закоханих «Date Night Movies» (за даними [3])

Цей сайт відомий своєю простотою та точністю в рекомендаціях. Інтерфейс цього сайту складається лише з однієї функціональної сторінки, що робить його використання максимально простим. Користувач обирає два фільми, на основі яких підбирається третій. Планується, що наша система буде відрізнятися більш складним алгоритмом, який дозволить враховувати вподобання всіх учасників групи, що зробить її більш ефективною у контексті музичних рекомендацій.

Таким чином, ця робота відкриває нову перспективну нішу для розвитку програмного забезпечення, спрямованого на музичні рекомендації українським користувачам на основі колективних вподобань. Впровадження такої системи дозволить покращити досвід користувачів, сприяючи спільному відкриттю нової музики та зміцненню соціальних зв'язків через спільне слухання та обговорення музичних вподобань. Наприклад, група друзів зможе створювати спільні плейлісти для вечірок, враховуючи вподобання кожного учасника, що зробить прослуховування музики більш інтерактивним і соціально значущим.

1.2.1 Бізнес ризики

Бізнес ризики є невід'ємною складовою будь-якого проекту, незалежно від його розміру та галузі. Список ризиків, а також заходи для їх пом’якшення наведено в таблиці 1.1

Таблиця 1.1 – Бізнес ризики (таблиця виконана самостійно)

Ризики	Складність	Пом’якшення
Велика конкуренція на ринку	Середня	Унікальний функціонал системи; Ефективний маркетинг
Створення якісного продукту	Низька	Тестування та зворотній зв’язок від користувачів
Низький попит	Висока	Активний маркетинг та просування сервісу
Низька продуктивність сервера	Середня	Оптимізація програмного забезпечення; Апгрейд сервера
Малий бюджет	Висока	Пошук інвесторів, презентація проекту для широкої аудиторії для знаходження інвестування

Управління цими ризиками вимагає уважного планування, ефективного використання ресурсів та прийняття відповідних заходів для пом'якшення ризиків. Знання та усвідомлення цих ризиків допоможуть забезпечити стабільність та успішний розвиток проекту.

1.3 Постановка задачі

Для розробки фронтенд компонента програмної системи рекомендації музики на основі вподобань групи користувачів, буде використано фреймворк Angular з TypeScript. Для досягнення цієї мети необхідно виконати наступні завдання:

– провести дослідження та визначити вимоги користувачів, щоб зрозуміти їхні потреби та очікування. Це включатиме опитування та аналіз взаємодії з подібними сервісами;

– розробити низькорівневі макети (wireframes) основних сторінок застосунку, таких як головна сторінка, сторінка рекомендацій, профіль користувача, пошук музики тощо. Для цього будуть використані інструменти, такі як Figma або Adobe XD;

– налаштувати проект Angular за допомогою Angular CLI, створивши базову структуру додатку, включаючи модулі, компоненти, сервіси та маршрути;

– реалізувати основні компоненти інтерфейсу користувача, такі як Header, Footer, Sidebar, забезпечивши їх адаптивність для різних пристроїв (мобільних телефонів, планшетів, десктопів);

– розробити сервіси для взаємодії з бекендом, включаючи авторизацію, отримання рекомендацій, пошук музики та управління плейлистами, використовуючи HTTP-клієнт Angular для надсилання запитів;

– впровадити механізми аутентифікації та авторизації, включаючи JWT токени, забезпечивши безпечне зберігання токенів і конфіденційних даних на клієнтській стороні;

– використати Angular Material або інші бібліотеки компонентів для створення зручного та естетичного інтерфейсу, впровадити анімації та переходи для покращення взаємодії користувача з додатком;

– розробити тестові сценарії для ключових компонентів та сервісів за допомогою Jasmine та Karma, провести модульне тестування для перевірки основних функціональних можливостей;

– аналізувати продуктивність додатку за допомогою інструментів, таких як Lighthouse, та оптимізувати завантаження ресурсів, використовуючи lazy loading та інші техніки;

– створити алгоритм, що зможе чітко визначати музичні інтереси користувачів системи.

Для реалізації фронтенд компонента буде використано фреймворк Angular з TypeScript, що дозволить створити кросплатформенний додаток, який працюватиме на різних операційних системах, таких як Windows, Linux, MacOS.

1.3.1 Основні користувачі системи

Основними користувачами системи рекомендації музики на основі вподобань групи користувачів є:

– музичні ентузіасти та фанати: цінують нові музичні відкриття і люблять ділитись своїми уподобаннями з іншими. Вони використовуватимуть систему для знаходження нової музики, яка відповідає їхнім стильовим і музичним вподобанням. Крім того, вони активно стежитимуть за оновленнями в музичному світі і будуть першими, хто дізнається про нові релізи, альбоми та виконавців. Для них важливо мати можливість створювати та ділитися своїми плейлістами з іншими, а також отримувати рекомендації, що базуються на їхніх унікальних музичних вподобаннях;

– друзі та групи однодумців: будуть створювати групи з друзями або особами зі схожими музичними смаками. Вони будуть використовувати систему для спільного формування плейлистів, обговорення музики та обміну рекомендаціями. Такий формат сприятиме соціальній взаємодії та поглибленню відносин через спільні музичні інтереси. Крім того, можливість

створювати тематичні плейлісти для конкретних подій або настроїв стане важливим аспектом для цієї групи користувачів;

– організатори подій та вечірок: влаштовують заходи, можуть використовувати систему для створення спеціальних плейлистів, щоб підтримати атмосферу певної події або вечірки відповідно до музичних уподобань учасників. Вони потребують зручного інструменту для швидкого підбору музики, що відповідатиме тематиці заходу та створюватиме потрібний настрій. Також важливою є можливість аналізувати реакцію гостей на музику для подальшого вдосконалення плейлистів;

– музичні дослідники та професіонали: музиканти, композитори та музичні критики можуть використовувати систему для дослідження тенденцій, аналізу музичного ринку та взаємодії зі своєю аудиторією. Вони зможуть використовувати аналітичні інструменти для визначення популярних жанрів, аналізу вподобань слухачів та отримання зворотного зв'язку щодо своїх творів. Така інформація є цінною для подальшого розвитку їхньої творчості та розширення аудиторії;

– менеджери музичних лейблів та дистриб'ютори: можуть використовувати систему для аналізу популярності різних музичних напрямків та визначення стратегій просування артистів. Вони зможуть відслідковувати тенденції в реальному часі, аналізувати поведінку слухачів та оптимізувати маркетингові кампанії для максимального охоплення аудиторії. Також вони зможуть використовувати систему для пошуку нових талановитих виконавців, які можуть зацікавити широку аудиторію.

Кожна з цих груп користувачів має власні потреби та очікування від системи рекомендацій музики, і вона має бути спроектована таким чином, щоб задовольняти ці потреби і стимулювати активну взаємодію та співпрацю між користувачами. Наприклад, для музичних ентузіастів важливі точні та індивідуальні рекомендації, а для організаторів подій – можливість швидко адаптувати плейлісти до специфіки заходу. Таким чином, система повинна бути

гнучкою і багатофункціональною, щоб ефективно обслуговувати різні категорії користувачів.

1.3.2 Монетизація

Монетизація є важливим аспектом сучасних музичних платформ, оскільки забезпечує стабільний дохід і підтримку функціонування системи. Для рекомендаційних систем для музичних платформ, монетизація може здійснюватися через декілька ключових механізмів, які забезпечують не лише фінансову стійкість, але й покращують користувацький досвід.

Перш за все, музичні платформи часто використовують модель підписки. Вона може включати як безкоштовну, так і платну версію. Безкоштовна версія пропонує базову функціональність усім користувачам, але з певними обмеженнями, такими як реклама між треками, обмежена кількість пропусків треків або відсутність доступу до певних функцій. Платна підписка, навпаки, надає розширений функціонал, який включає відсутність реклами, необмежені пропуски треків, високу якість звуку, можливість завантаження музики для офлайн-прослуховування та доступ до ексклюзивного контенту. Однією з ключових функцій платної підписки може бути можливість перегляду детальної історії прослуховування та минулих рекомендацій, що дозволяє користувачам повторно відкривати улюблені треки та розширювати свої музичні горизонти.

Монетизація музичних платформ сприяє не лише їхньому фінансовому успіху, але й розвитку нових функцій та покращенню користувацького досвіду. Включення платних функцій, таких як перегляд історії минулих рекомендацій, дозволяє користувачам мати повний контроль над своїм музичним досвідом і максимально використовувати потенціал рекомендаційної системи. Таким чином, монетизація є ключовим елементом стратегії розвитку музичних платформ, що поєднує фінансову стійкість з високим рівнем задоволеності користувачів.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Функціональні вимоги

Основні функціональні вимоги включають:

– реєстрація облікового запису (включає в себе введення персональних даних, таких як: ім’я, прізвище, нікнейм облікового запису, пароль та поштову адресу);

– аутентифікація та авторизація;

– придбання підписки (відкриває доступ до преміум функції, а саме доступу до перегляду історії рекомендацій);

– створення запиту на отримання рекомендацій (включає введення назв двох пісень, що слугують необхідними вхідними параметрами для отримання плейлісту рекомендаційних треків);

– редагування облікового запису (включає в себе зміну персональних даних, таких як: ім’я, прізвище, нікнейм облікового запису, пароль, поштову адресу та аватар облікового запису);

– видалення облікового запису (видалення персональних даних з реєстру бази даних);

– блокування облікового запису (функція доступна лише для користувача з рівнем доступу Адміністратор та дозволяє обмежити права доступу користувачу до облікового запису);

– фільтрація та пошук облікових запитів;

– зміна ролі (Адміністратор має можливість змінити рівень доступу користувачам);

– інтеграція музичних сервісів (система повинна забезпечувати підключення до сторонніх музичних сервісів, таких як Spotify та синхронізувати дані з ним);

– відгук на рекомендацію (користувач повинен мати можливість оцінити якість рекомендації);

– генерація статистики (створення статистичних графіків для отримання інформації стосовно реєстраціє/авторизації, кількості придбаних підписок, кількості отриманих рекомендацій).

2.2 Нефункціональні вимоги

Нефункціональні вимоги визначають характеристики:
– продуктивність (система повинна швидко реагувати на запити користувачів та обробляти велику кількість запитів без затримок);
– масштабованість (система повинна працювати на тому ж рівні швидкості при зростаючій кількості користувачів та забезпечувати можливість розширення інфраструктури за потреби);
– юзабіліті (інтерфейс програми повинен бути інтуїтивно зрозумілим, легким у використанні для користувачів різної вікової категорії).

2.3 Інтерфейсні вимоги

Інтерфейсні вимоги:
– інтерфейс користувача (інтерфейс повинен бути адаптивний для різних типів пристроїв: комп’ютер, планшет та смартфон. Використання сучасного фреймворку Angular для динамічного простору);
– юзабіліті та доступність (Інтерфейс повинен відповідати принципам доступності WCAG та підтримувати мультимовність і локалізацію для забезпечення підтримки різних мов за допомогою бібліотек ngx-translate).

2.4 Безпека та конфіденційність

Вимоги до безпеки та конфіденційності:
– захист персональних даних (система повинна забезпечувати шифрування персональних даних, зберігання даних, дотримуючись вимог законодавства GDPR);

– автентифікація та авторизація (система повинна впроваджувати надійні міри автентифікації та мати можливість розмежувати права доступу).

2.5 Інтеграційні вимоги

– API інтеграції (система повинна взаємодіяти з музичним сервісом Spotify, підтримуючі стандартні протоколи обміну даними);
– сторонні бібліотеки та сервіси.

2.6 Обмеження та припущення

– обмеження 1 (система повинна бути розроблена з використанням певних технологій, Back-end: ASP.NET Web API; Front-end: Angular [4], TypeScript [5]; Mobile: MAIU);
– обмеження 2 (система повинна бути сумісною з сучасними версіями основних веб-браузерів (Chrome, Firefox, Safari, Edge));
– обмеження 3 (проект має фіксований бюджет, що може обмежувати використання платних інструментів або послуг);
– обмеження 4 (встановлені строки завершення проекту, що обмежують час, доступний для розробки та тестування).
– припущення 1 (дані, отримані від зовнішніх сервісів, будуть точними і актуальними);
– припущення 2 (користувачі матимуть базові технічні навички, достатні для використання веб-застосунку).

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ

3.1 UML проєктування ПЗ

Для початку роботи над реалізацією програмної системи було проведено детальний аналіз, де були визначені основні функції користувача веб-застосунку. Діаграма прецедентів [6] відображає взаємодію між акторами та системою в рамках певного контексту. Вона демонструє функціональність системи шляхом ідентифікації основних дій, які можуть бути виконані акторами. Два актора: авторизований та неавторизований користувач. Неавторизований користувач має можливість зареєструватись у системі, а також здійснити авторизацію, якщо вже був зареєстрований. Авторизований користувач вже має доступ до багатьох функціональних можливостей (див. рис. 3.1).

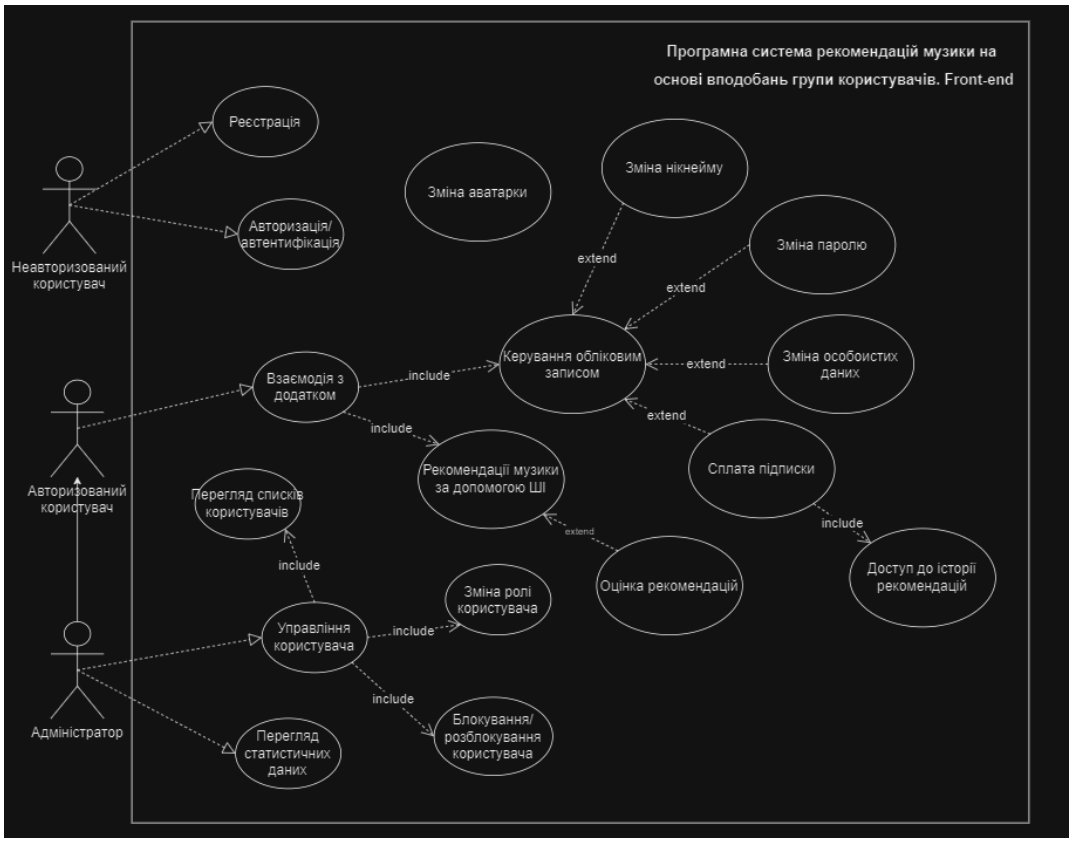


Рисунок 3.1 – Use-case діаграма веб частини(рисунок виконаний самостійно)

Незареєстрований користувач має можливість створювати та авторизуватись у власному особистому кабінеті. Після того, як користувач створив власний

аккаунт, він може отримати доступ до інших функцій, таких як взаємодія з власним профілем (включає в себе редагування та оновлення будь-якої особистої інформації; придбання підписки). Кожен зареєстрований користувач має можливість взаємодіяти зі штучним інтелектом для підбору музики. Також, слід зазначити, що за умови придбання підписки, користувач має можливість взаємодіяти з історією підбору музики.

Для кращого розуміння процесів було створено діаграму діяльності, що графічно відображає послідовність дій, які відбуваються у системі (див. рис. 3.2).

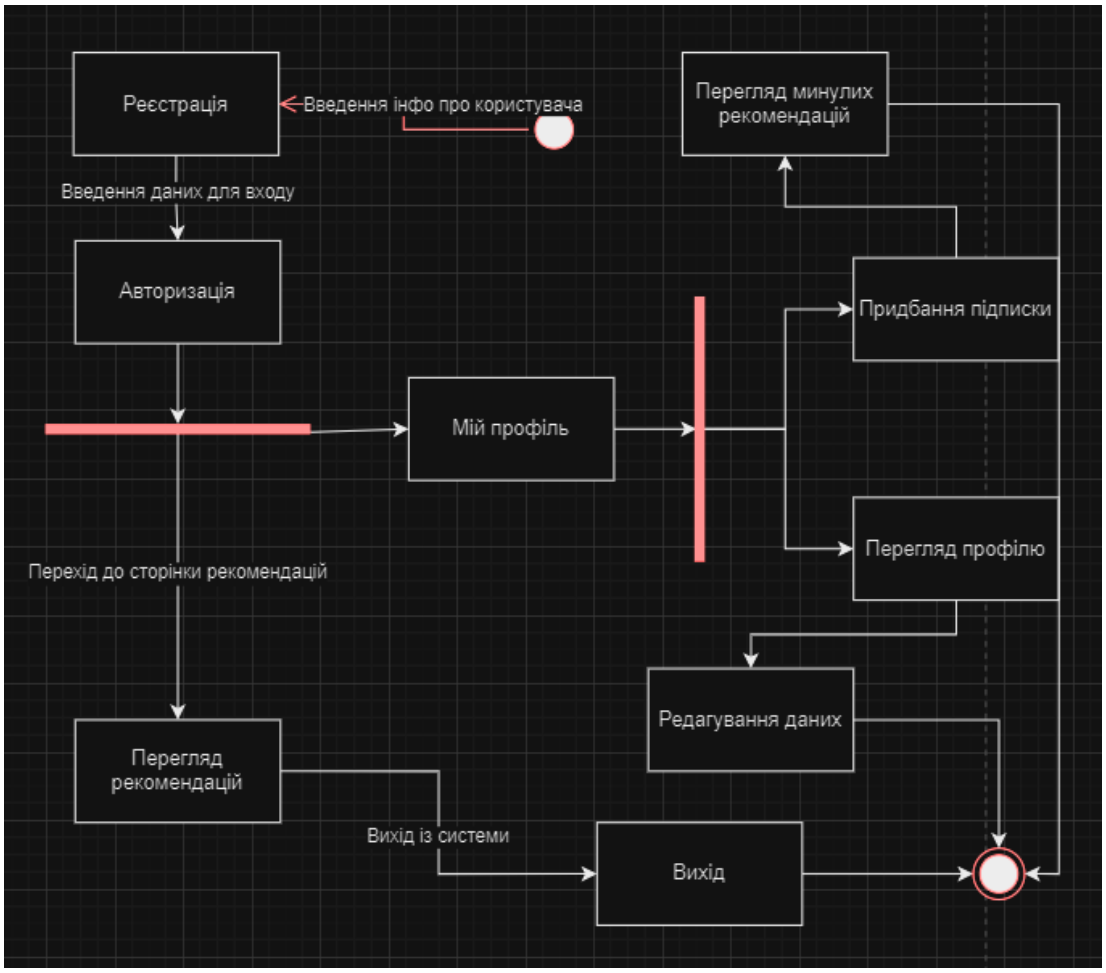


Рисунок 3.2 – Діаграма діяльності веб-застосунку (рисунок виконаний самостійно)

Створення таких діаграм дозволяє ілюструвати основні функції і взаємодію між акторами та можливостями системи. Ці діаграми використовується як основа для подальшого проектування, розробки і планування робіт над системою.

Front-end частина складається з наступних компонентів: ProfileComponent, AuthComponent, що в свою чергу складається з LoginComponent та RegistrationComponent, StatisticComponent, MusicRecommendationComponent, PaypalComponent, AdminComponent (див. рис. 3.3).

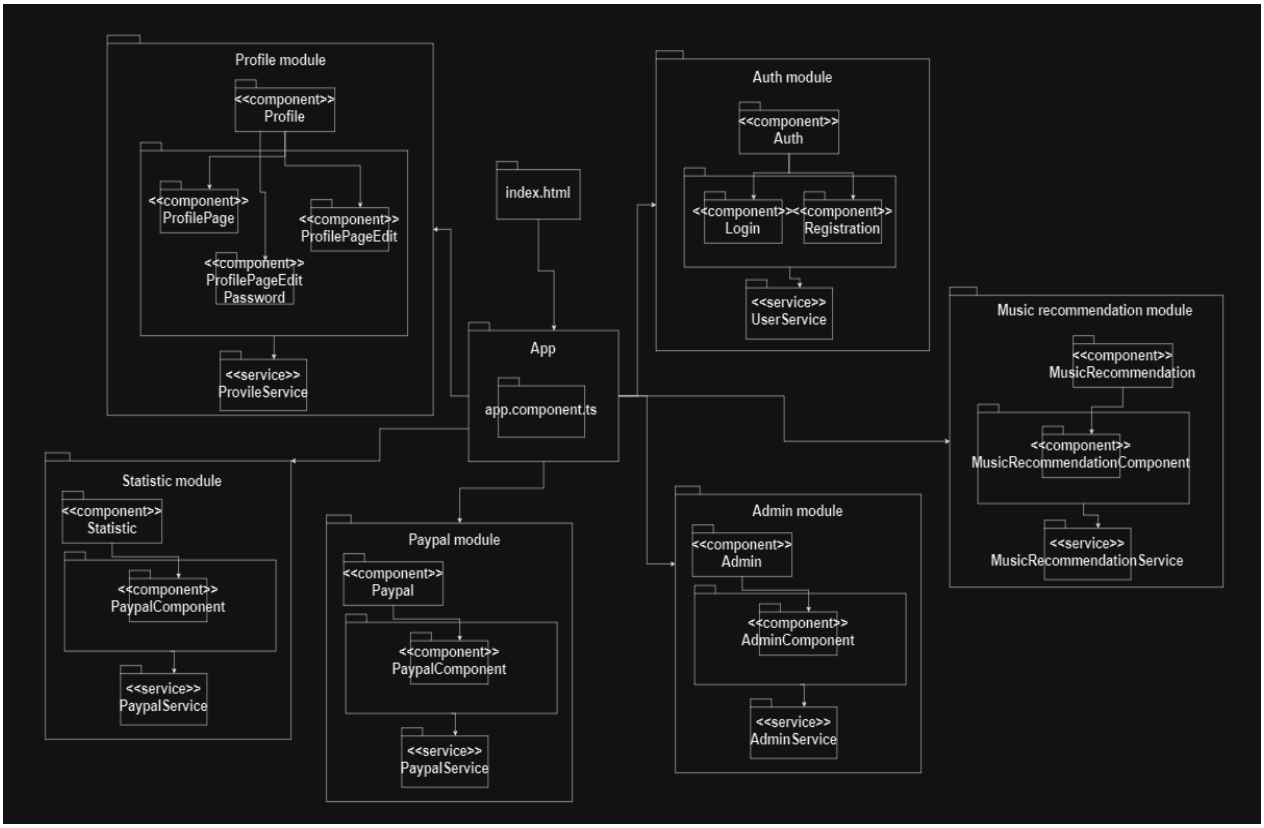


Рисунок 3.3 – UML діаграма компонентів (рисунок виконаний самостійно)

Застосунок має файл початку з назвою index.html, який розроблений за концептом single page, що відтворює в браузері відповідну сторінку в залежності від дій користувача та підставляє вміст необхідного файлу, до якого було звернення. Концепт single page дозволяє забезпечити швидку та безперервну роботу застосунку, оскільки замість перезавантаження всієї сторінки, оновлюються лише її частини, що значно підвищує зручність та швидкість користувацького досвіду.

Проаналізувавши структуру моделі програмної системи було побудовано діаграму пакетів (див. рис. 3.4).

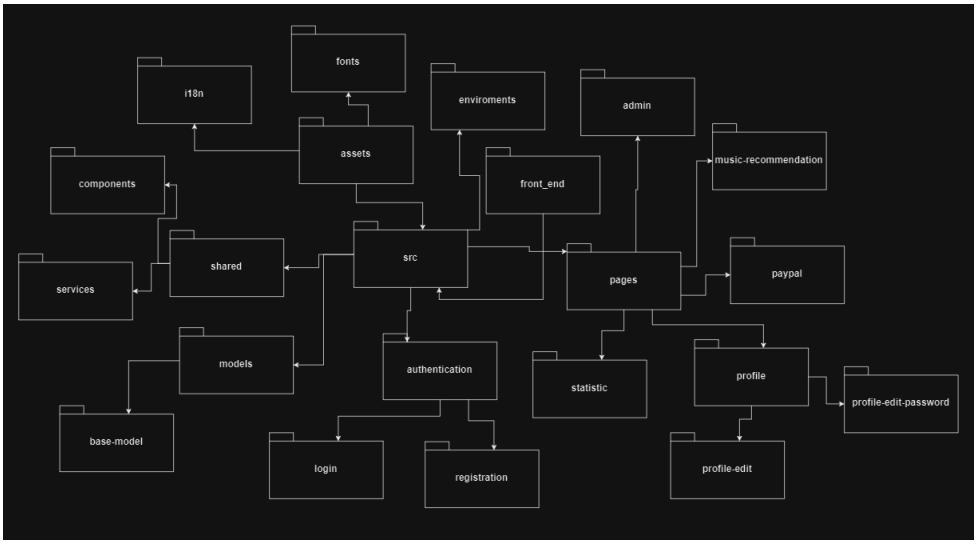


Рисунок 3.4 – UML діаграма пакетів проекту (рисунок виконаний самостійно)

Виділимо наступні пакети:

– front_end (зовнішній основний пакет, що налічує в собі бібліотеки та файли конфігурації проекту);

– src (внутрішній основний пакет, що містить усі інші пакети компоненти з усією логікою додатку та кореневий модуль проекту);

– environments (пакет, що містить файли налаштування локалізації та конфігурації додатку);

– i18n (пакет з двома файлами формату .json, що містять в собі словники з локалізаціями (українську та англійську));

– base_model (пакет, що містить інтерфейси, які загальні та використовуються для всіх компонентів);

– pages (пакет, що містить інші пакети такі як: authentication (відповідає за сторінки, пов’язані з автентифікацією користувача, включаючи логін та реєстрацію); profile: (містить компоненти для перегляду профіля та його редагування); paypal: (пакет з сторінками сплати підписки та налаштуванням платіжних даних з інтеграцією PaypalBraintree); music-recommendation: (пакет з сторінкою для рекомендаційної системи); admin: (пакет, що містить сторінку адміністратора та надає інструменти для керування додатком);

statistic (пакет, що доступний лише зі сторінки адміністратора, необхідний для будування діаграм-звітності)).

3.2 Проєктування архітектури ПЗ

Додаток буде розподілено на дві частини:

Загальна частина, яка завантажуватиметься з самого початку (бандл main.ts). Вона буде містити AppConfig з роутами верхнього рівня, а також усі основні сервіси, які плануються використовуватися у всьому додатку.

Lazy loaded features, завантажуватиметься в результаті навігації користувача до цих фіч (модулів).

Основний компонент AppComponent буде кореневим компонентом, який ініціалізується та містить основний макет. Він використовуватиме RouterOutlet для організації навігації та завантаження відповідних компонентів при зміні маршрутів.

Lazy loading реалізує компоненти, директиви та pipes, які є специфічними для цієї фічі. Наприклад, якийсь конкретний компонент, який не може бути повторно використаний іншими фічами.

– pages/admin: Компоненти та сервіси для адміністрування додатку;

– pages/music-recommendation: Компоненти для відображення рекомендацій музики;

– pages/paypal: інтеграція з платіжною системою Paypal;

– pages/profile: управління профілями користувачів;

– pages/statistic: відображення статистичних даних;

– authentication/login: компоненти для автентифікації користувачів;

– authentication/registration: компоненти для реєстрації облікових записів.

Цей підхід до архітектури дозволить ефективно управляти залежностями, забезпечувати високий рівень продуктивності та легкість підтримки додатку.

3.2.1 Стек технологій застосований для розробки

Для розробки front-end частини веб-системи буде використано наступні технології:

– Angular: потужний JavaScript фреймворк, який використовується для створення веб-додатків та односторінкових додатків (SPA). Angular забезпечує модульну структуру додатку, що спрощує розробку, тестування та підтримку коду.

– TypeScript: розширення JavaScript з підтримкою типів, що дозволяє виявляти і усувати помилки на етапі розробки. Використання TypeScript полегшує розуміння коду і забезпечує більшу надійність програмного забезпечення.

– HttpClient (вбудований у Angular): Angular надає власний сервіс HttpClient для виконання HTTP-запитів, який забезпечує зручний і простий інтерфейс для роботи з API.

– Braintree [7]: платіжна платформа, яка надає зручні інструменти для інтеграції платіжних функцій у веб-застосунок. За допомогою Braintree можна реалізувати платіжні операції в додатку з високим рівнем безпеки.

– Chart.js: бібліотека для створення інтерактивних графіків та діаграм у веб-системах. Вона дозволяє візуалізувати дані користувачів зрозумілим способом за допомогою різноманітних типів графіків.

– i18n: бібліотека для локалізації інтерфейсу веб-застосунку. Вона дозволяє підтримувати різні мови та культури користувачів, що полегшує інтернаціоналізацію додатку;

– RxJS (Reactive Extensions for JavaScript): бібліотека для роботи з асинхронними подіями за допомогою Observables, яка надає потужні засоби для композиції асинхронних і подійно-орієнтованих програм.

3.2.2 Продуктивність та оптимізація

Для забезпечення високої продуктивності та оптимізації роботи веб-системи буде впроваджено кілька стратегій та підходів:

Lazy loading, що дозволить завантажувати лише ті модулі та компоненти, які потрібні користувачеві в даний момент, що значно зменшує час початкового завантаження додатку та покращує продуктивність;

Для ефективного управління асинхронними операціями та оптимізації роботи з даними планується використовувати оператори RxJS, такі як pipe, takeUntil, tap, та catchError. Це дозволить обробляти дані більш гнучко та ефективно.

3.3 Проєктування структури зберігання даних

Для зберігання необхідних даних буде обрано кукі, що дозволить зберігати та отримувати невеликі обсяги даних на стороні клієнта. Це особливо корисно для збереження стану сесії, налаштувань користувача та інших невеликих даних, які потрібні під час взаємодії з додатком.

Для збереження фотографій буде використано Blob сховище AZURE [8], що забезпечує надійне та масштабоване зберігання великих обсягів даних (див. рис. 3.5).

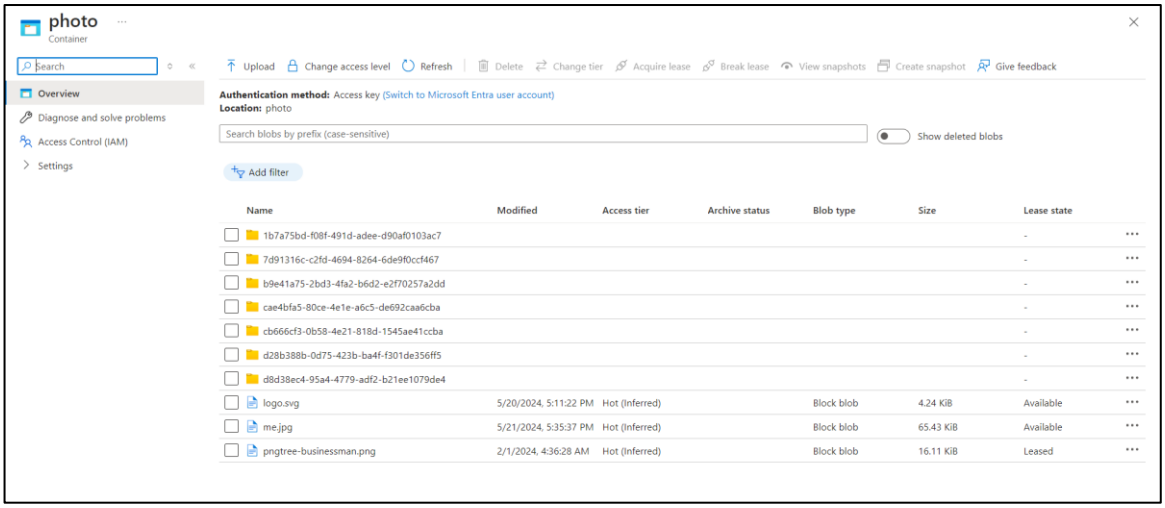


Рисунок 3.5 – Blob сховище AZURE (рисунок виконаний самостійно)

Це дозволить зменшити навантаження на сервер та покращити продуктивність додатку.

3.4 Приклади найцікавіших алгоритмів та методів

Розглянемо один із найцікавіших алгоритмів, який використовується в системах рекомендацій музики на основі вподобань користувачів. Для цього проекту було обрано контентно-орієнтовані методи, що є найбільш підходящими для поставленого завдання.

Контентно-орієнтована фільтрація базується на ідеї, що користувачам будуть подобатися елементи, які мають схожі атрибути або характеристики з тими, що вони вже оцінили позитивно. Наприклад, якщо вам сподобався комедійний фільм з певним актором, можливо, вам сподобаються й інші фільми цього ж жанру з тим самим актором. Контентно-орієнтована [9] фільтрація використовує інформацію про самі елементи, такі як їхні жанри, характеристики або описи, для формування рекомендацій для конкретного користувача. Вона не потребує жодної інформації про інших користувачів, їхні рейтинги або взаємодії.

Для обчислення рекомендацій була використана модель векторного простору, яка є популярним методом у контентно-орієнтованих системах рекомендацій. У цій моделі кожен елемент представлений як вектор його характеристик в n-вимірному просторі. Для визначення схожості між елементами обчислюється кут між векторами. Косинусна схожість є поширеною мірою для визначення близькості двох векторів у цій моделі, що в кінцевому підсумку надає оцінку їх подібності. Результат обрахування косинусної схожості 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 А;

b_n – компонент вектору В.

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 А.

На графіку показано, що чим ближче елементи за кутом, тим вища косинусна схожість (див. рис. 3.6).

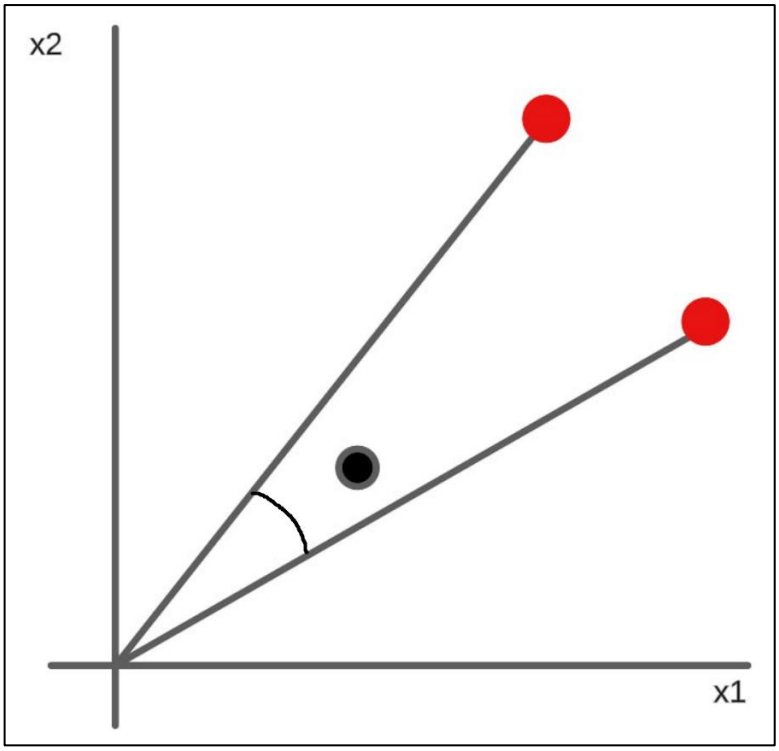


Рисунок 3.6 – Косинусна подібність (рисунок виконаний самостійно)

Це означає, що якщо кут великий, то два елементи відрізняються один від одного, а якщо кут малий, то два елементи схожі один на одного.

Кут θ між двома піснями визначає, наскільки вони схожі одна на одну. θ варіюється від 0 до 1, де 0 означає найменшу схожість, а 1 – найбільшу.

Наша функція обчислює косинусну схожість між однією піснею, що подається в алгоритм, та іншими піснями в наборі даних, потім ранжує пісні на основі косинусної схожості та генерує плейліст, що налічує п'ять найбільш схожих пісень у якості результату.

3.5 UI/UX дизайну системи

Інтерфейс веб-частини є обмеженим екранним простором, тому успішний UI/UX дизайн вимагає особливої уваги до деталей та функціональності. Веб-частина Melody Fusion надає користувачам інтуїтивно зрозумілий [10] та естетично приємний інтерфейс, що забезпечує швидкий доступ до основних функцій програмної системи.

Основні принципи UI/UX дизайну включають інтуїтивну навігацію, мінімалістичний дизайн, унікальний візуальний стиль, а також функціональність та простоту використання.

Інтуїтивна навігація є важливим аспектом планування системи навігації. Меню навігації розташоване вгорі екрану і містить основні вкладки: Home, Profile та Logout. Це дозволяє користувачам швидко переміщатися між основними функціями. Функціональні кнопки, такі як кнопки для входу та реєстрації, чітко виділені і знаходяться в центрі відповідних екранів, забезпечуючи легкий доступ.

Мінімалістичний дизайн забезпечує чіткість і простоту використання. Поля для введення логіна і пароля мають іконки для додаткової візуальної підказки, що робить процес вводу більш інтуїтивним.

Контрастні кнопки, такі як кнопка входу зеленого кольору, виділяються на основному синьому фоні, що робить їх легко помітними. Використання простих

іконок і мінімалістичних елементів забезпечує естетичний вигляд та легкість сприйняття.

Унікальний візуальний стиль досягається за рахунок використання спокійної палітри кольорів з домінуючим синім кольором, що асоціюється з музикою та спокоєм, а також контрастними кольорами для важливих елементів. Використання сучасних шрифтів, які легко читаються. Логотип Melody Fusion відображає музичну тематику та сприяє запам’ятовуванню бренду.

Функціональність та простота використання відображені у всіх аспектах дизайну. Екрани для входу (Sign In) та реєстрації (Sign Up) мають подібну структуру, що робить їх використання інтуїтивним.

Домашня сторінка включає пошукові поля для треків та відображає рекомендації у вигляді альбомних обкладинок, що полегшує користувачам знайти потрібну музику.

Прототип екрану входу (Sign In) містить поля для введення логіна і пароля, а також кнопку входу (див. рис. 3.7).

Рисунок 3.7 –Прототип сторінки логіну (рисунок виконаний самостійно)

Дизайн екрану включає синій фон, білий текст та зелену кнопку для акценту. Поля для введення логіну і паролю мають відповідні іконки, що підвищує зручність використання. Кнопка входу виділена зеленим кольором для легкої ідентифікації.

Прототип екрану реєстрації (Sign Up) подібний до екрану входу, але з кнопкою створення акаунту золотого кольору (див. рис. 3.8).

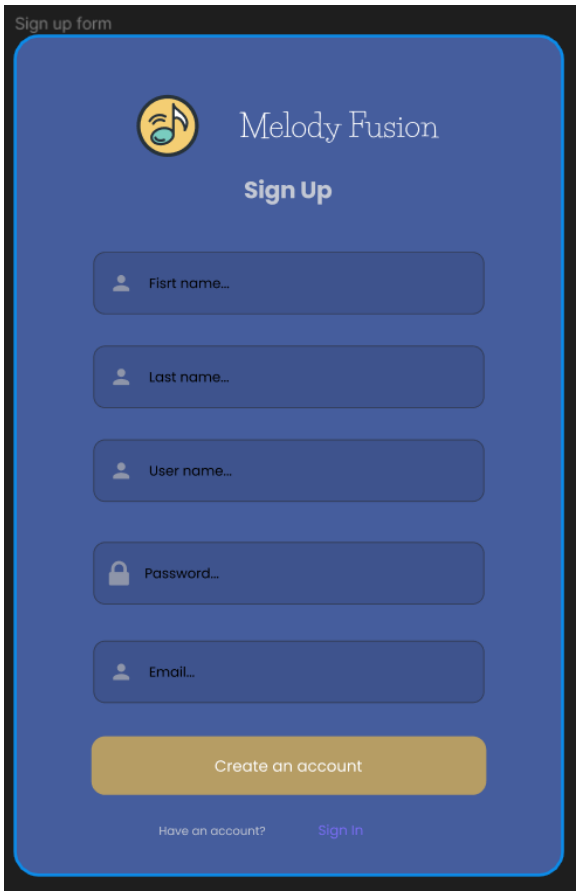


Рисунок 3.8 – Прототип сторінки реєстрації (рисунок виконаний самостійно)

Поля введення мають ті ж іконки для консистентності, що допомагає користувачам легко орієнтуватися та ідентифікувати функціонал кожного поля. Такий підхід забезпечує зручність використання та естетичну привабливість інтерфейсу, створюючи єдиний стиль та покращуючи користувацький досвід.

Прототип домашньої сторінки (Home Page) включає пошукові поля та рекомендації треків (див. рис. 3.9).

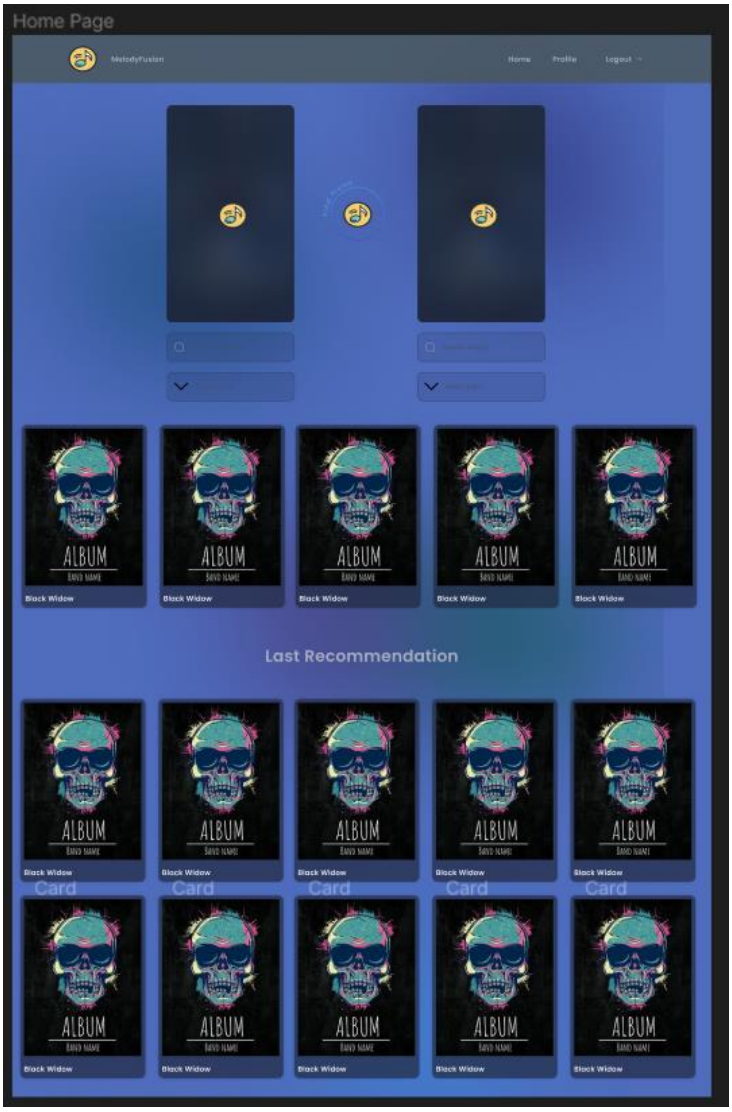


Рисунок 3.9 – Прототип сторінки підбору музики (рисунок виконаний самостійно)

Дизайн включає синій фон, білі поля для пошуку та альбомні обкладинки для рекомендацій. Пошукові поля дозволяють вводити ключові слова або точну назву треку. Після пошуку користувач отримує рекомендації треків, відображені у вигляді альбомних обкладинок. Історія рекомендацій доступна для перегляду після натискання відповідної кнопки.

Ці прототипи демонструють ключові елементи дизайну та функціональності веб-системи Melody Fusion, забезпечуючи інтуїтивно зрозумілий і візуально привабливий інтерфейс для користувачів.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Оптимізація та продуктивність

Для забезпечення високої продуктивності та оптимізації роботи веб-додатку були впроваджені кілька стратегій та підходів.

Lazy loading було реалізовано для завантаження лише тих модулів та компонентів, які потрібні користувачеві в даний момент. Це значно зменшило час початкового завантаження додатку та покращило продуктивність. Lazy loading затримує ініціалізацію об'єктів до моменту їх фактичного використання, що корисно для великих додатків з багатьма компонентами.

```
import { Routes } from '@angular/router';

export const routes: Routes = [
  {
    path: '',
    redirectTo: '/login',
    pathMatch: 'full'
  },
  {
    path: 'registration',
    loadChildren:                              ()                              =>
import('./authentication/registration/registration.module').then(m  =>
m.RegistrationModule)
  },
  {
    path: 'login',
    loadChildren:                              ()                              =>
import('./authentication/login/login.module').then(m => m.LoginModule)
  },
  {
    path: '',
    loadChildren:                              ()                              =>
import('./pages/profile/profile.module').then(m => m.ProfileModule)
  },
  {
    path: '',
    loadChildren: () => import('./pages/admin/admin.module').then(m =>
m.AdminModule)
  },
  {
    path: '',
    loadChildren:  ()  =>  import('./pages/music-recommendation/music-
recommendation.module').then(m => m.MusicRecommendationModule)
  },
  {
```

```
    path: '',
    loadChildren:                          ()                          =>
import('./pages/statistic/statistic.module').then(m                    =>
m.StatisticModule)
  }
];
```

RxJS (Reactive Extensions for JavaScript) забезпечує потужні інструменти для обробки асинхронних потоків даних. Оператори RxJS допомагають керувати потоками даних та подіями у додатку.

– метод pipe, який дозволяє ланцюжити декілька операторів RxJS разом для обробки потоку даних. Він приймає оператори як аргументи і повертає новий об'єкт Observable, що містить усі застосовані оператори;

– оператор takeUntil, який дозволяє завершити підписку на Observable, коли інший Observable видає значення. Корисний для управління підписками, особливо для уникнення витоків пам'яті;

– оператор tap, який дозволяє виконувати побічні ефекти для кожного значення у потоці, не змінюючи його;

– оператор catchError, який дозволяє обробляти помилки в Observable. Він приймає функцію, яка повертає новий Observable або значення для подальшого оброблення.

```
this.musicRecommendationService.GetRecommendationById(
      musicRecommendationRequest.FirstSongId,
      musicRecommendationRequest.SecondSongId)
      .pipe(
        takeUntil(this.unsubscribe$),
        tap((data: SongSpotifyResponseModel[]) => {
          this.toastr.success("Recommendations                received
successfully.")
          this.listRecommendedSongs = data;
        }),
        catchError(() => {
          this.toastr.warning("Both songs must be selected.")
          return of(undefined);
        })
      ).subscribe();
```

Кукі використовуються для зберігання інформації на стороні клієнта, що забезпечує швидкий доступ до цих даних без необхідності звертатися до сервера.

```
public getCultureParam(): string {
    return this.cookieService.get('culture');
  }
  public logout(){
    this.cookieService.deleteAll();
  }
  public getToken(): string {
    return this.cookieService.get('token') ?? '';
  }
  public isAuthenticated(): boolean {
    const token = this.getToken();
    return token != '';
  }
```

Azure Blob Storage надає можливість зберігання великих файлів, таких як зображення, відео та інші мультимедійні файли, з високою надійністю та масштабованістю.

```
changeAvatar(photo?: File):Observable<UserPhotoModel>{
    const formData = new FormData();
    formData.append('photo', photo ? photo : "")
    return
this.http.put<UserPhotoModel>(`${this.api}/api/UserAccount/ChangeAvat
ar`, formData);
  }
```

Метод, що надсилає запит на сторону back-end, де і відбувається запис фотокарток до blob сховища. Працює і в зворотному порядку, кожний користувач має на сховищі відповідну свою папку, де і зберігається фото. Для відображення на стороні клієнта необхідно лише URL до цього зображення.

HTTP інтерсептор покращує продуктивність і оптимізує роботу веб-додатку шляхом додавання при кожному запиті токен від сервісу UserService і додає його до заголовків запиту.

```
const token = inject(UserService).getToken();
  let modifiedReq = req;
  if (token) {
    modifiedReq = req.clone({
      setHeaders: {
        Authorization: `Bearer ${token}`
      }
    });
  }
```

Це дозволяє автоматично авторизувати користувача на сервері без необхідності вручну додавати токен до кожного запиту в різних частинах додатку.

4.2 Інтеграція з зовнішніми сервісами

Для розширення функціональності веб-застосунку була інтегрована API від Spotify. Ця інтеграція дозволяє користувачам отримувати доступ до треків та отримувати детальну інформацію про них безпосередньо через веб-затосунок.

Spotify API надає широкий спектр можливостей для доступу до музичних треків, альбомів, виконавців та плейлистів. За допомогою цього API можна здійснювати пошук, отримувати рекомендації, управляти музичними бібліотеками користувачів, а також отримувати метадані про треки.

Користувачі можуть вводити назву пісні, альбому або виконавця у пошуковий рядок. API повертає результати, що відповідають запиту. Пошук здійснюється за допомогою методу GET /v1/search, який дозволяє здійснювати пошук по треках, альбомах, виконавцях або плейлистах (див. рис. 4.1).

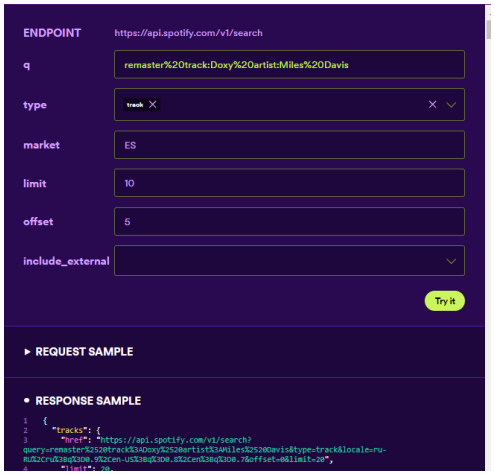


Рисунок 4.1 – Результат пошуку за назвою (за даними [1])

Для отримання детальної інформації про конкретний трек використовується метод GET /v1/tracks/{id} (див. рис. 4.2).

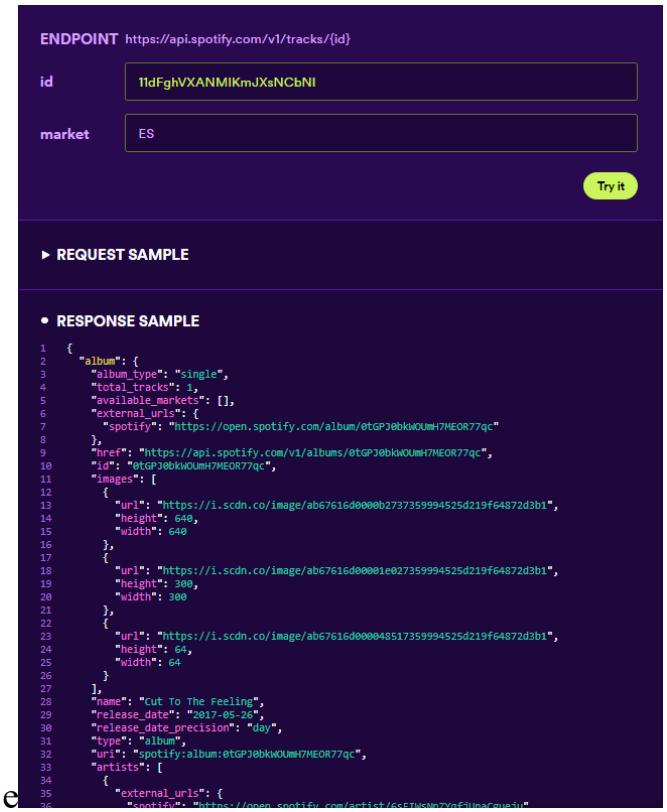


Рисунок 4.2 – Результа пошуку за ідентифікатором (за даними [1])

Цей запит повертає дані про трек, включаючи його назву, виконавця, альбом, тривалість та інші метадані.

Інтеграція Spotify API дозволяє веб-системі надавати користувачам розширений функціонал для пошуку та отримання інформації про музичні треки. Це значно підвищує зручність та привабливість додатку, надаючи доступ до величезної бібліотеки музичних творів.

Для забезпечення зручної та безпечної оплати в нашому додатку було інтегровано платіжну систему PaypalBraintree. Це дозволяє користувачам здійснювати покупки, використовуючи кредитні картки, без необхідності покидати веб-систему. Інтеграція забезпечує високу безпеку та швидкість обробки транзакцій.

Спочатку, необхідно отримати CLIENT_AUTHORIZATION, що знаходиться в налаштування середовища. Це унікальний токен, іншими словами ключ санбоксу куди і будуть поступати сплати. За допомогою методу braintree.client.create створюємо клієнтський екземпляр, використовуючи отриману авторизацію.

Наступним кроком збираємо дані, які користувач увів до відповідних полей та формуємо з них об’єкт картки для сплати. Надсилаємо об’єкт до Braintree API до відповідного ендпоінту payment_methods/credit_cards та отримуємо токенізовану версію картки для сплати, що і дозволяє створити платіжний запит використовуючи отриманий токен (див. Додаток Б). Код демонструє інтеграцію PayPal BrainTree для обробки платежів у нашому додатку, що забезпечує безпечне і швидке виконання транзакцій для користувачів.

4.3 Реалізація алгоритму рекомендації

Наша функція обчислює косинусну схожість між однією піснею, що подається в алгоритм, та іншими піснями в наборі даних, потім ранжує пісні на основі косинусної схожості та генерує плейліст, що налічує п'ять найбільш схожих пісень у якості результату.

Таким чином ми отримуємо функцію song_recommended, що очікує два вхідних параметри, що є ідентифікаторами треків (див. Додаток А). Попередньо необхідно видалити зайве поле "id_artists" з DataFrame.

Після необхідно для кожної пісні/треку у бібліотеці пісень визначити косинусну схожість із заданими піснями, що ми отримали як параметри до функції. Створюємо вектори з поля "genres" та з числових стовпців для інших пісень. Обчислюємо косинусну схожість, використовуючи спочатку текстові вектори, а після числові вектори. Беремо середнє значення обох оцінок схожості і додаємо до списку оцінок схожості. Цю інформацію ми відтворимо у новоствореному стовпці "similarity" у DataFrame. Відсортовуємо DataFrame на основі стовпця "similarity" та обираємо з відсортованого списку перші 5 варіантів рекомендації, які найбільш схожі на задані.

4.4 Демонстрація роботи веб-застосунку

При кожному запуску програмної системи, користувача зустрічає початкове вікно, яке в свою чергу слугую сторінкою логіну (див. рис. 4.3).

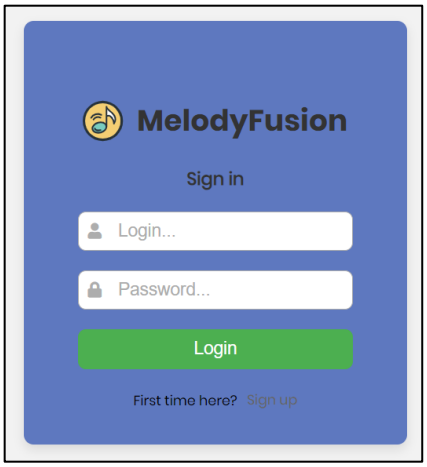


Рисунок 4.3 – Сторінка логіну (рисунок виконаний самостійно)

Надалі користувачу доступні наступні дії, якщо ж користувач вже має обліковий запив, він може ввести відповідні дані. На кожному полі вводу є валідація, яка перевіряє коректність введених даних та при успішній авторизації, користувач отримає про це повідомлення в правому нижньому куту та перейде до наступної сторінки. В іншому ж випадку, користувач не має облікового запису та має можливість створити його натиснувши «Sign up» (див. рис. 4.4).

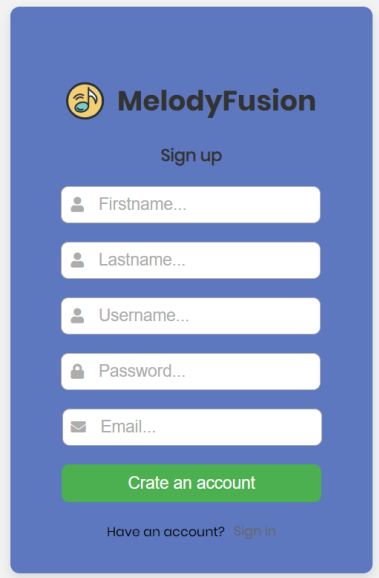


Рисунок 4.4 – Сторінка реєстрації (рисунок виконаний самостійно)

Будемо вважати, що користувач виконав необхідні дії та етап реєстрації пройдений успішно, після чого він виконує авторизацію у додатку та потрапляє до головної сторінки (див. рис. 4.5).

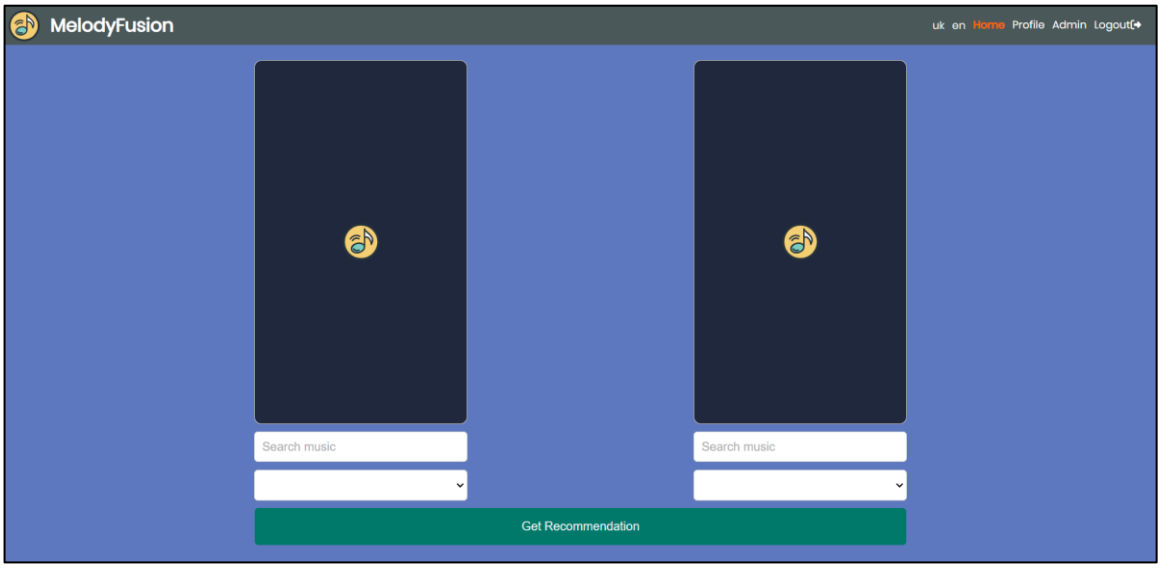


Рисунок 4.5 – Сторінка рекомендацій музики (рисунок виконаний самостійно)

З цього моменту користувачу відкривається безліч нових функцій. Звернемо увагу на навігаційну панель, так як авторизований користувач є адміністратором, то відповідно до його ролі він має доступ до адміністративної панелі (див. рис. 4.6).

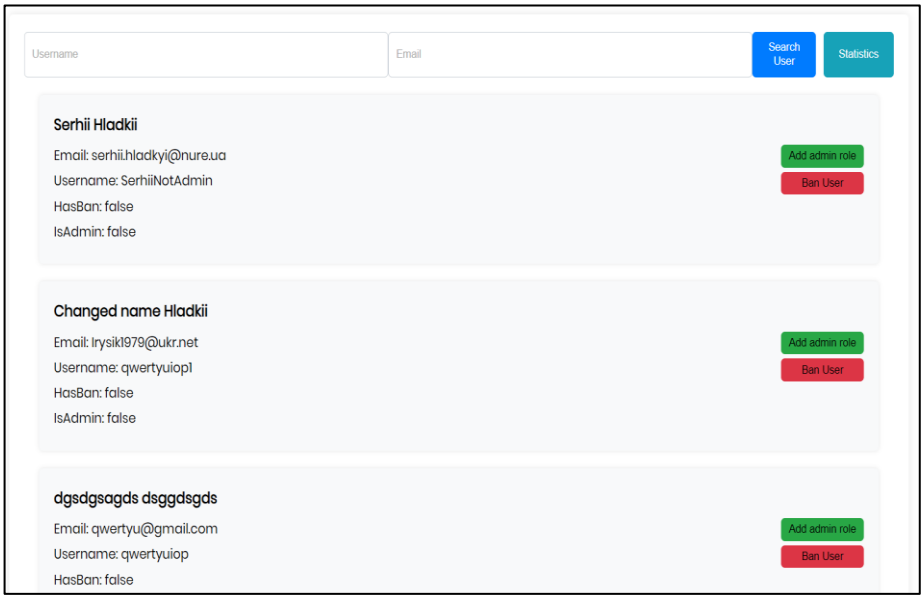


Рисунок 4.6 – Адмін панель (рисунок виконаний самостійно)

Адміністратор має можливість пошуку користувачів за наступними параметрами:

– user name;

– email address.

Також, при порушенні правил користування сайтом, адміністратор має можливість обмежити доступ облікового запису або при необхідності надати користувачу права доступу адміністратора. Для отримання статистичних даних необхідно натиснути відповідну кнопку «Statistic» (див. рис. 4.7).

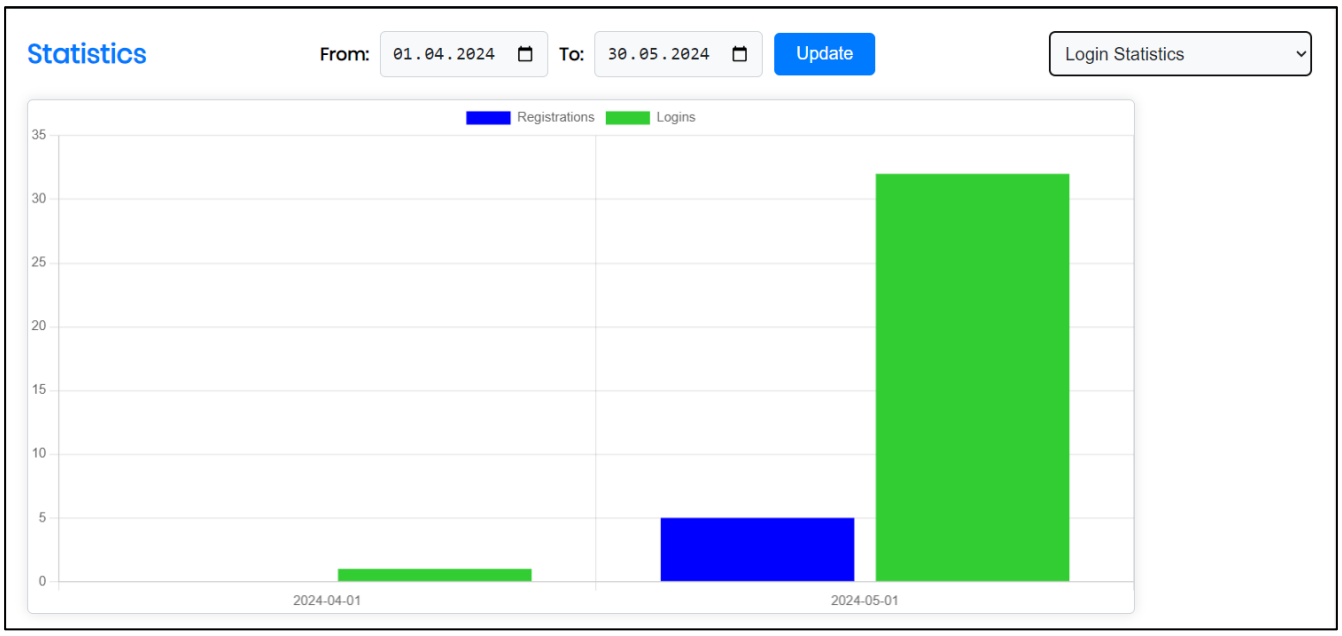


Рисунок 4.7 – Сторінка статистики (рисунок виконаний самостійно)

Відповідний блок дозволяє отримати та проаналізувати статистичні дані стосовно кількості зареєстрованих користувачів та кількості відвідувань сторінки протягом місяця, параметри статистики можна задати у відповідних полях. На сторінці є SelectBox з трьома типами статистик, що відтворюються в залежності від обраного типу:

– login statistic;

– payment statistic;

– recommendation statistic.

Повернемося до початкової сторінки та звернемо увагу на локалізацію, для зміни на іншу, необхідно натиснути на бажану після чого мова додатку змінить відповідно до обраної (див. рис. 4.8).

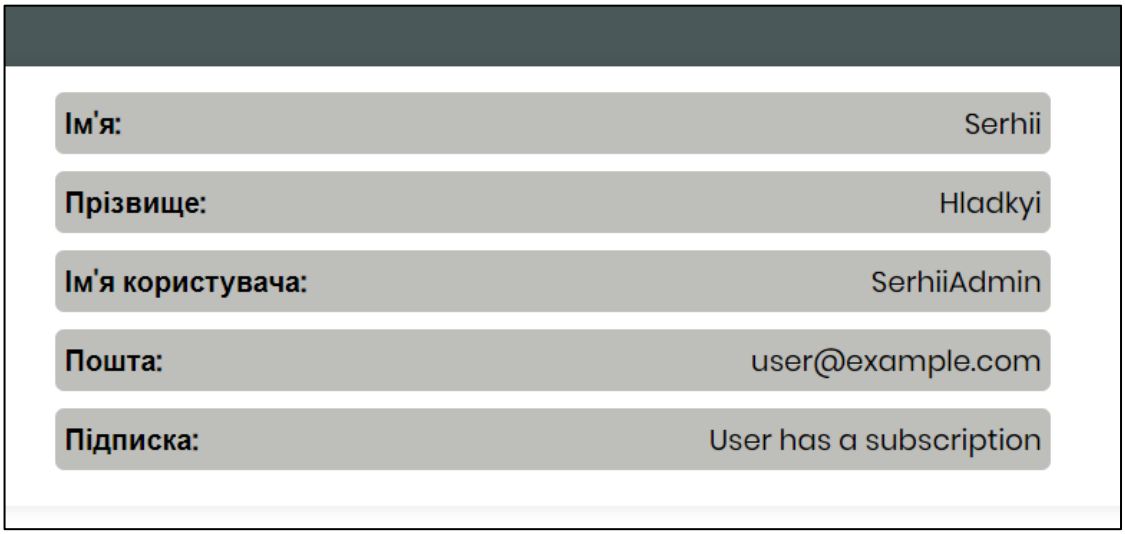


Рисунок 4.8 – Зміна мови веб-застосунку на українську (рисунок виконаний самостійно)

Знову повертаємо до основної сторінки рекомендацій музики та спробуємо заповнити поля, які передбачають символи для пошуку треку за ними. Ввели 3 символи (системне обмеження) та при співпадінні з треками, що налічує БД можемо обрати відповідний трек з selectBox (див. рис. 4.9).

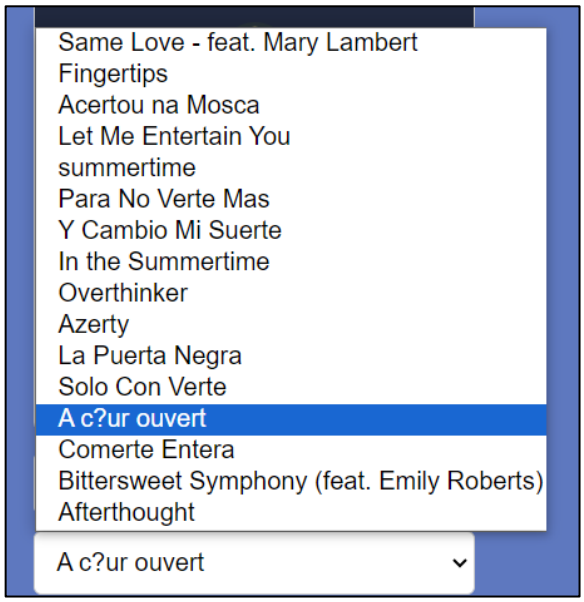


Рисунок 4.9 – SelectBox з треками (рисунок виконаний самостійно)

Так як вже було описано, що функція рекомендації треків очікує два вхідних параметра, а саме два ідентифікатори треків, то користувачу необхідно обрати бажані назви та натиснути кнопку «Get recommendation», після чого з очікуванням близько 5 секунд користувач зможе побачити на власному екрані запропонований плейліст з 5 найбільш вдалими співпадіннями (див. рис. 4.10).

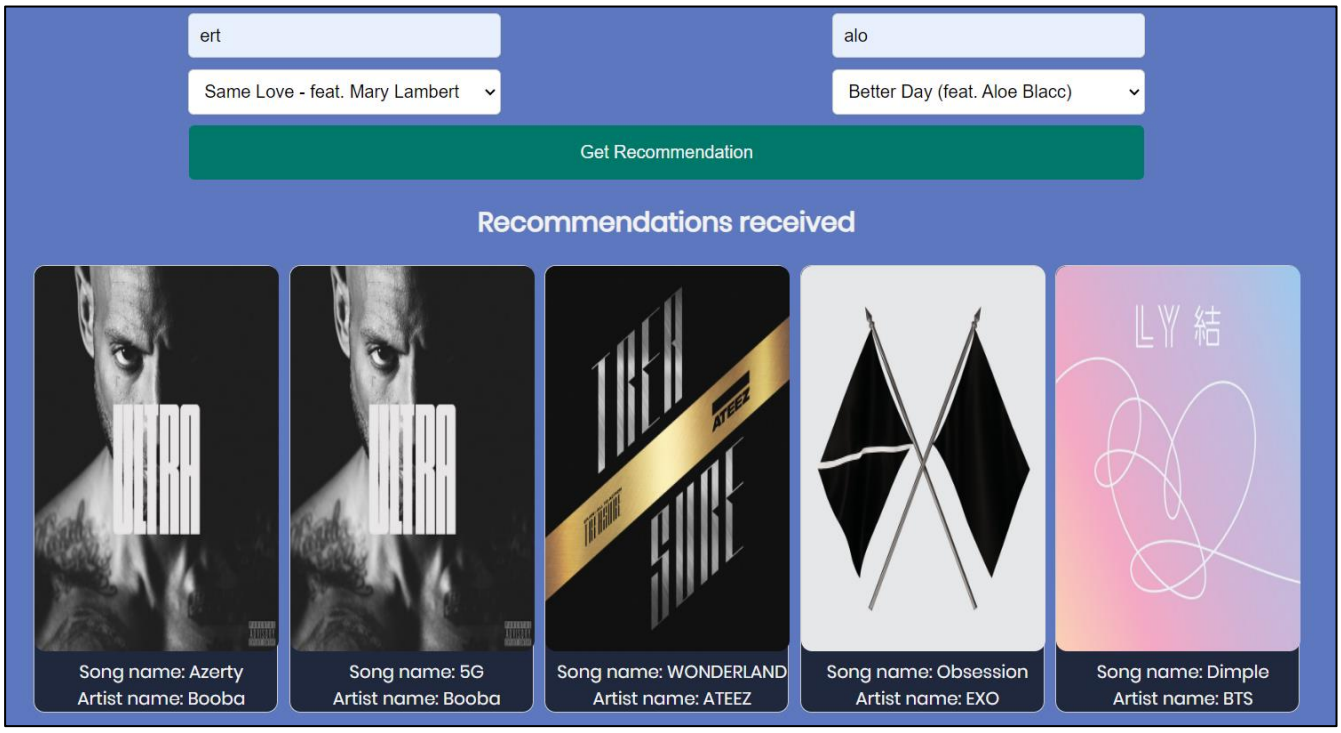


Рисунок 4.10 – Отримання рекомендації (рисунок виконаний самостійно)

Слід також зазначити, що у випадку якщо користувач не придбав підписку, то він буде отримувати спливаючі повідомлення, які інформують його про те, що підписка не придбана і для отримання історії рекомендацій, йому необхідно сплатити її через власний кабінет у блоці «Subscription». Спливаючі повідомлення будуть з'являтися у ненав'язливій формі, щоб не заважати користувачу, але водночас нагадувати про можливості, які відкриває платна підписка. Це допоможе користувачам зрозуміти цінність платної підписки та стимулюватиме їх до її придбання для покращення користувацького досвіду. В іншому випадку будемо вважати, що користувач придбав платну підписку та має можливість переглядати історію власних рекомендацій (див. рис. 4.11).

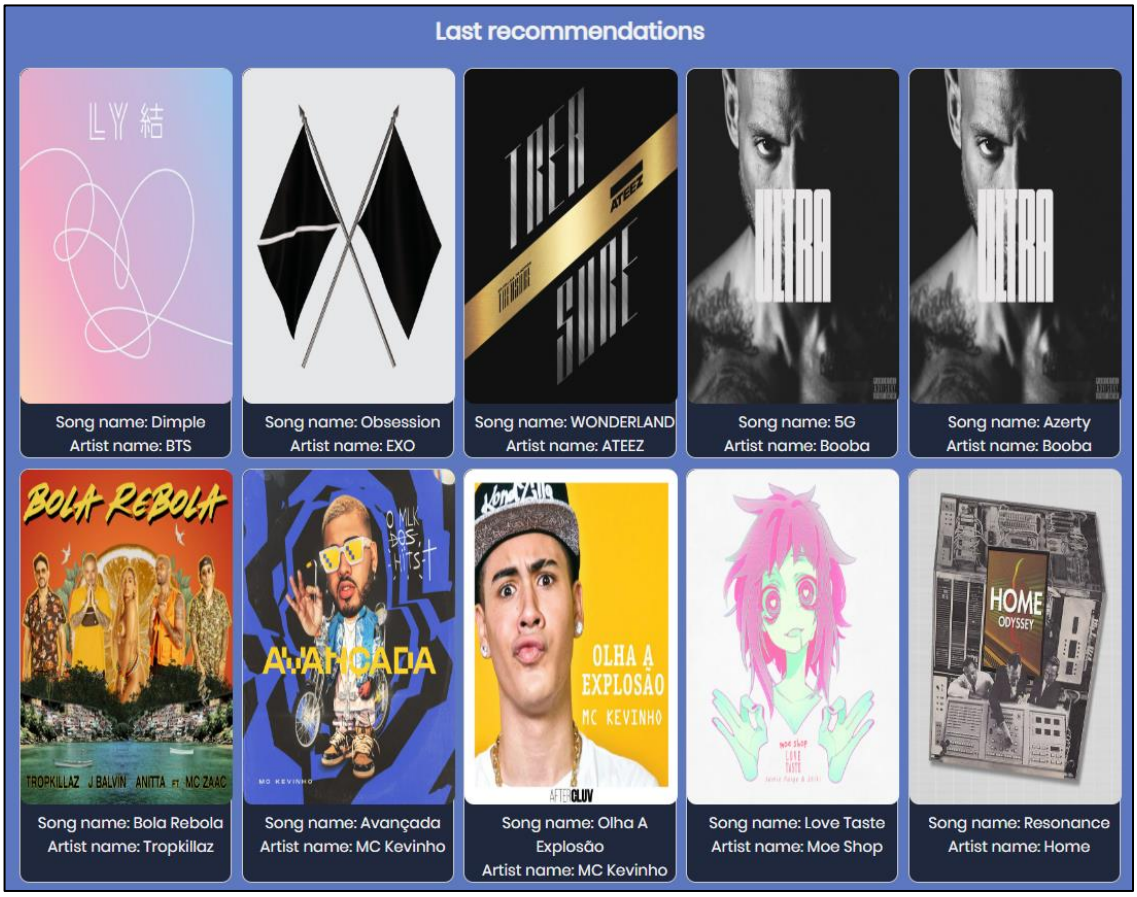


Рисунок 4.11 – Перегляд історії рекомендацій (рисунок виконаний самостійно)

При обранні у компоненті шапки сторінки пункту «Profile» або за українською локалізацією «Профіль», користувач зможе переглянути попередньо введену ним інформацію стосовно власного облікового запису, змінити необхідні параметри за бажання, а саме:

– прізвище;

– ім’я;

– нікнейм;

– пароль;

– аватарку;

– поштову адресу.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Коли ми говоримо про якісне тестування програмного забезпечення, одним із ключових інструментів є юніт тести. Юніт тести – це невеликі, ізольовані тести, які перевіряють функціональність окремих частин коду, званих модулями або юнітами. Основною метою юніт тестування є визначення, що кожен окремий юніт працює правильно. В контексті Angular, юніт тести часто пишуться з використанням фреймворку Jasmine і виконуються за допомогою тестового раннера Karma.

Karma – це інструмент для запуску тестів у реальному середовищі браузера. Він дозволяє розробникам автоматично виконувати тести при кожній зміні коду, що забезпечує негайний зворотній зв’язок про роботу програмного забезпечення. Karma підтримує безліч браузерів, що дозволяє перевіряти крос-браузерну сумісність додатку.

Одним з важливих інструментів, які пропонує Jasmine для юніт тестування, є spyOn. Цей метод використовується для стеження за викликами методів та їх поведінкою без необхідності змінювати реальний код. Він дозволяє підмінити методи об'єктів на тестові версії, що значно полегшує перевірку складних взаємодій та залежностей.

Таким чином, процес виглядає наступним чином: ми пишемо юніт тести за допомогою Jasmine, використовуємо spyOn для підміни методів, а Karma виконує ці тести у браузері. Це забезпечує всебічну перевірку коду, швидкий зворотній зв’язок та впевненість у правильності роботи програмного забезпечення.

5.1 Покриття коду тестами

Для виводу звітності та отримання графік скористуємося готовим налаштування тесту, таким як «code coverage» зі значенням true. Налаштування відбувається в файлі конфігурацій проекту angular.json (див. рис. 5.1).

```
"test": {
  "builder": "@angular-devkit/build-angular:karma",
  "options": {
    "codeCoverage": true,
    "polyfills": [
      "zone.js",
      "zone.js/testing"
    ],
```

Рисунок 5.1 – Конфігурація тестів у файлі angular.json (рисунок виконаний самостійно)

На початок створення проекту була задана планка, що код окремих модулів та компонентів буде протестований як мінімум на 80% (див. рис. 5.2).

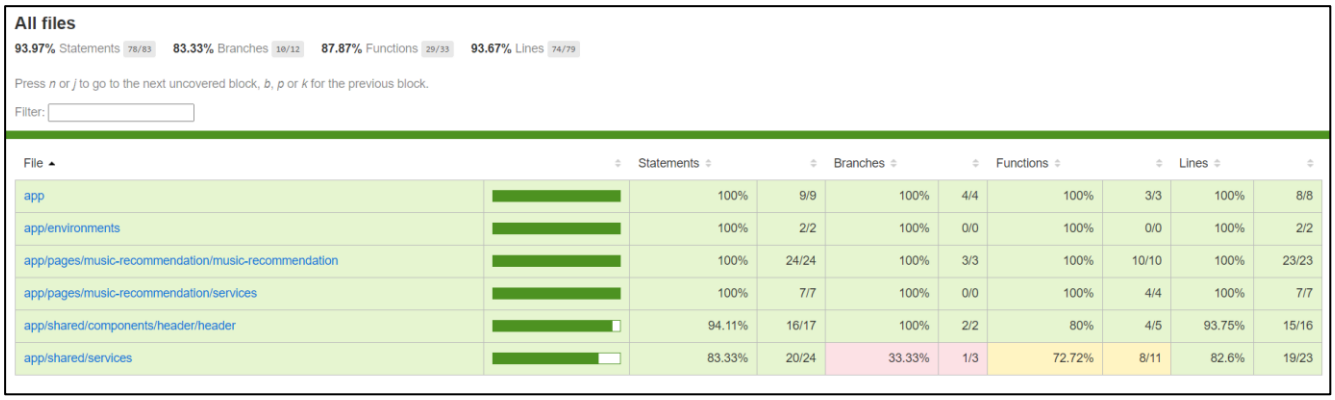


Рисунок 5.2 – Code coverage report (рисунок виконаний самостійно)

У верхній частині звіту представлено загальні показники покриття для всього проекту:

– statements: 93.97% (73/83) – покриття операторів;

– branches: 83.33% (10/12) – покриття гілок;

– functions: 87.87% (29/33) – покриття функцій;

– lines: 93.67% (74/79) – покриття рядків коду.

Більшість файлів мають дуже високий рівень покриття, близький до 100%. Це свідчить про те, що основні частини коду добре покриті тестами (див. рисю 5.3).

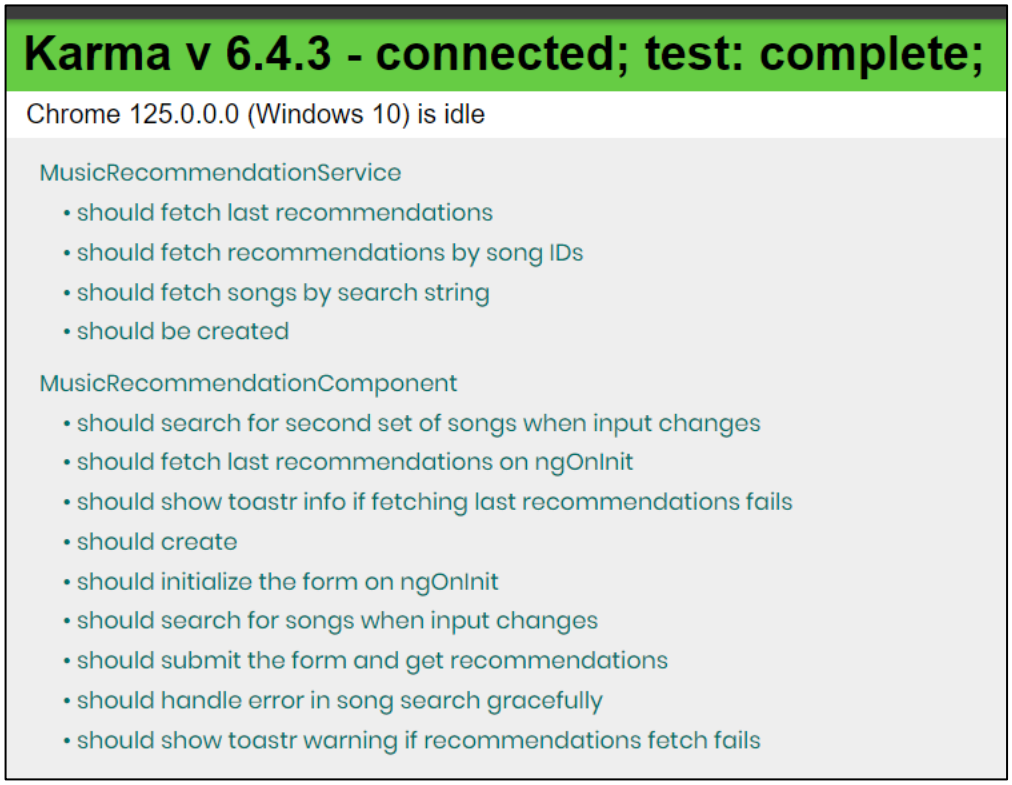


Рисунок 5.3 – Тестове середовище Carma (рисунок виконаний самостійно)

Файл, який є критичним та вважається основою функціональності проекту (app/pages/music-recommendation/music-recommendation) має 100% покриття для всіх показників.

Тестування компонентів за допомогою юніт-тестів в середовищі Karma показало високий рівень покриття коду, що свідчить про надійність та стабільність розробленого програмного забезпечення. Детальне тестування основних функціональних можливостей та обробки помилкових ситуацій дозволило бути впевненими в коректній роботі системи та її готовності до використання кінцевими користувачами.

ВИСНОВКИ

У ході кваліфікаційної роботи було розроблено фронтенд компонент програмної системи рекомендації музики на основі вподобань групи користувачів за допомогою фреймворку Angular з TypeScript. Було проведено всебічне дослідження вимог користувачів та аналіз взаємодії з подібними сервісами, що дозволило чітко зрозуміти їхні потреби та очікування.

a) розроблено низькорівневі макети (wireframes) основних сторінок застосунку:
   1) головна сторінка;
   2) сторінка рекомендацій;
   3) профіль користувача;
   4) сторінка пошуку музики;
   5) сторінка адміністратора;
   6) сторінка статистики;
   7) сторінка придбання підписки;
   8) використано інструменти Figma для створення макетів;

b) налаштовано проект Angular за допомогою Angular CLI:
   1) створено базову структуру додатку;
   2) розроблено необхідні модулі, компоненти, сервіси та маршрути;

c) реалізовано основні компоненти інтерфейсу користувача:
   1) header, footer, sidebar;
   2) забезпечено адаптивність інтерфейсу для різних пристроїв (мобільних телефонів, планшетів, десктопів);

d) розроблено сервіси для взаємодії з бекендом:
   1) авторизація користувачів;
   2) отримання рекомендацій;
   3) пошук музики;
   4) управління плейлистами;

5) використано HTTP-клієнт Angular для надсилання запитів;

e) впроваджено механізми аутентифікації та авторизації:

1) використання JWT токенів;

f) використано Angular Material та інші бібліотеки компонентів:

1) створення зручного та естетичного інтерфейсу;

2) впроваджено анімації та переходи для покращення взаємодії користувача з додатком;

g) розроблено тестові сценарії для ключових компонентів та сервісів:

1) використано Jasmine та Karma для модульного тестування;

2) проведено перевірку основних функціональних можливостей;

h) проаналізовано продуктивність додатку:

1) оптимізовано завантаження ресурсів, застосовано lazy loading та інші техніки;

i) створено алгоритм для визначення музичних інтересів користувачів:

1) реалізовано механізми збору та аналізу вподобань користувачів;

2) алгоритм інтегровано з системою рекомендацій.

Усе це забезпечило високу функціональність, безпеку та зручність використання додатку, відповідно до сучасних стандартів і потреб користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Spotify recommendations. https://www.spotify.com/ua-en/safetyandprivacy/understanding-recommendations. URL: https://www.spotify.com/ua-en/safetyandprivacy/understanding-recommendations (date of access: 30.04.2024).

2. TasteDive. https://tastedive.com/. URL: https://tastedive.com/ (date of access: 30.04.2024).

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

4. Angular. https://angular.io/docs. URL: https://angular.io/docs (date of access: 27.04.2024).

5. TypeScript. https://www.typescriptlang.org/docs/.URL: https://www.typescriptlang.org/docs/ (date of access: 27.05.2024).

6. UML diagram. https://miro.com/diagramming/what-is-a-uml-diagram/. URL: https://miro.com/diagramming/what-is-a-uml-diagram/ (date of access: 29.04.2024).

7. PayPal BrainTree. https://www.braintreepayments.com/features/paypal. URL: https://www.braintreepayments.com/features/paypal (date of access: 03.05.2024).

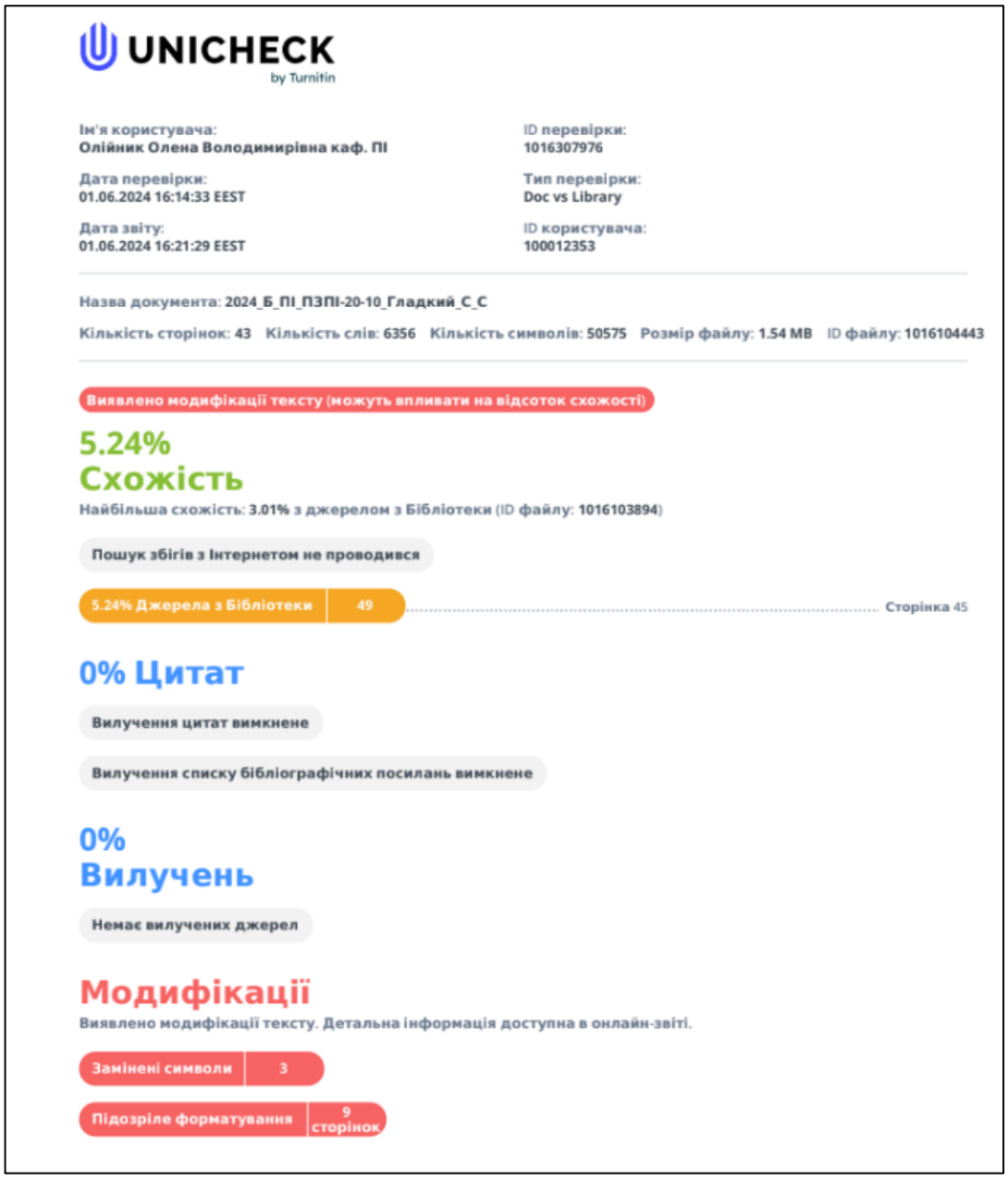
8. Perkins B. Microsoft Azure Architect Technologies and Design Complete Study Guide Exams AZ-303 and AZ-304. Wiley & Sons, Limited, John, 2021. 768 p.

9. Content-based recommendation system. https://www.researchgate.net/publication/236895069_Content-Based_Recommendation_Systems.URL: https://www.researchgate.net/publication/236895069_Content-Based_Recommendation_Systems (date of access: 06.05.2024).

10. Krug S. Don't Make Me Think, Revisited. A Common Sense Approach to Web Usability. 3rd ed. New Riders, 2014. 216 p.

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



UNICHECK
by Turnitin

Ім'я користувача:
Олійник Олена Володимирівна каф. ПІ

ID перевірки:
1016307976

Дата перевірки:
01.06.2024 16:14:33 EEST

Тип перевірки:
Doc vs Library

Дата звіту:
01.06.2024 16:21:29 EEST

ID користувача:
100012353

Назва документа: 2024_Б_ПІ_ПЗПІ-20-10_Гладкий_С_С

Кількість сторінок: 43 Кількість слів: 6356 Кількість символів: 50575 Розмір файлу: 1.54 MB ID файлу: 1016104443

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

5.24%
Схожість

Найбільша схожість: 3.01% з джерелом з Бібліотеки (ID файлу: 1016103894)

Пошук збігів з Інтернетом не проводився

5.24% Джерела з Бібліотеки 49 Сторінка 45

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 3

Підозріле форматування 9 сторінок

Рисунок А.1 – Результат перевірки на унікальність тексту

ДОДАТОК Б

Слайди презентації

Харківський національний університет радіоелектроніки
Кваліфікаційна робота бакалавра

Програмна система рекомендацій музики на основі вподобань групи користувачів. Front-end

Виконав: студент 4 курсу, ПЗПІ-20-10 Гладкий С. С.

Керівник: доц. кафедри ПІ Афанасьєва І. В.

1

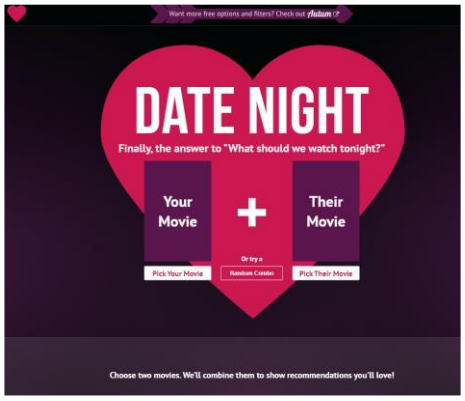
Мета кваліфікаційної роботи

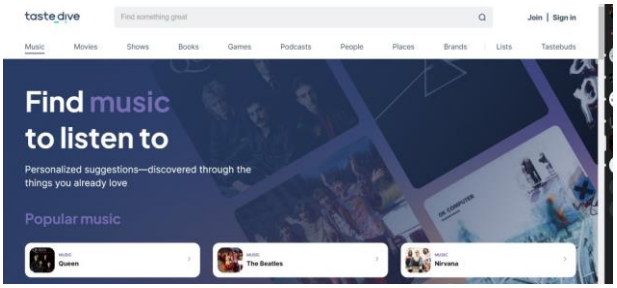
Створення веб застосунку для рекомендації музики. Автоматична генерація плейлісту рекомендованих треків.

Програмна система рекомендацій музики на основі вподобань групи користувачів.

2

Аналіз предметної галузі





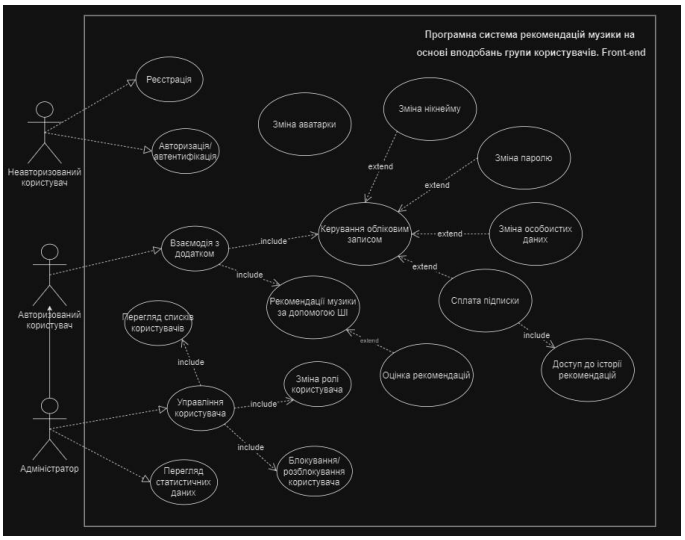
3

Постановка задачі

- розробити низькорівневі макети (wireframes) основних сторінок застосунку
- реалізувати основні компоненти інтерфейсу користувача
- розробити сервіси для взаємодії з бекендом
- впровадити механізми аутентифікації та авторизації
- розробити тестові сценарії для ключових компонентів та сервісів
- оптимізувати завантаження ресурсів
- створити алгоритм, що зможе чітко визначати музичні інтереси користувачів системи

4

Моделювання системи

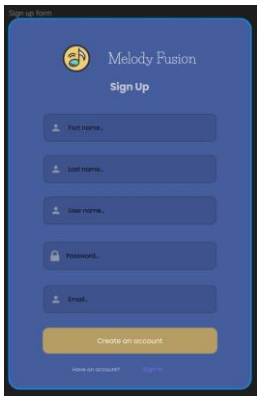


Авторизований/неавторитувачзований користувач

Адміністратор

5

Прототипування системи

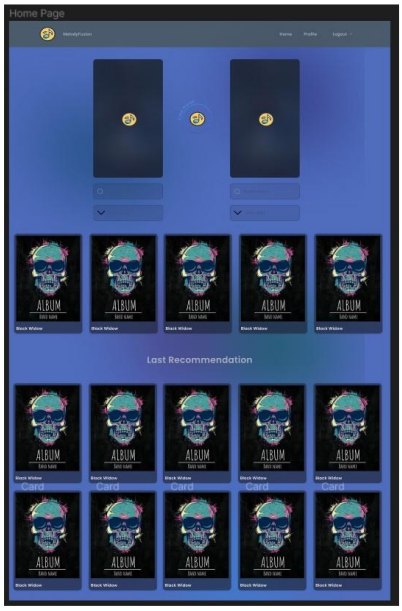


Прототип реєстрації

Прототип авторизації

6

Прототипування системи



Прототип основної сторінки

7

Алгоритм 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

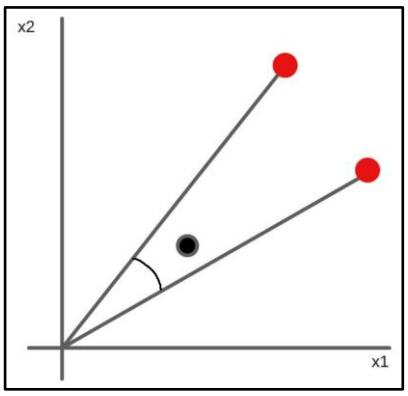
8

Алгоритм підбору музики

Норма вектора *A* обчислюється

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

де a*n* – компонент вектору A.



Косинусна подібність

Графік показує, що якщо кут великий, то два елементи відрізняються один від одного, а якщо кут малий, то два елементи схожі один на одного.

9

10

Інструменти та технології

Постановка задачі

- розробити низькорівневі макети (wireframes) основних сторінок застосунку
- реалізувати основні компоненти інтерфейсу користувача
- розробити сервіси для взаємодії з бекендом
- впровадити механізми аутентифікації та авторизації
- розробити тестові сценарії для ключових компонентів та сервісів
- оптимізувати завантаження ресурсів
- створити алгоритм, що зможе чітко визначати музичні інтереси користувачів системи

4

Прийняті програмні рішення

```
import { Routes } from '@angular/router';

export const routes: Routes = [  Show usages   ⚠ SunsetIover
  {
    path: '',
    redirectTo: '/login',
    pathMatch: 'full'
  },
  {
    path: 'registration',
    loadChildren: () => import('./authentication/registration/registration.module').then(m => m.RegistrationModule)
  },
  {
    path: 'login',
    loadChildren: () => import('./authentication/login/login.module').then(m => m.LoginModule)
  },
  {
    path: '',
    loadChildren: () => import('./pages/profile/profile.module').then(m => m.ProfileModule)
  },
  {
    path: '',
    loadChildren: () => import('./pages/admin/admin.module').then(m => m.AdminModule)
  },
  {
    path: '',
    loadChildren: () => import('./pages/music-recommendation/music-recommendation.module').then(m => m.MusicRecommend
  },
```

Lazy loading

```
this.musicRecommendationService.GetRecommendationById(
  musicRecommendationRequest.FirstSongId,
  musicRecommendationRequest.SecondSongId)
  .pipe(
    takeUntil(this.unsubscribe$),
    tap( observerOrNext: (data: SongSpotifyResponseModel[]) : void  => {
      this.toastr.success( message: "Recommendations received successfully.")
      this.listRecommendedSongs = data;
    }),
    catchError( selector: () => {
      this.toastr.warning( message: "Both songs must be selected.")
      return of( value: undefined);
    })
  ).subscribe();
```

Обробки асинхронних потоків даних

12

Прийняті програмні рішення

```
public getCultureParam(): string {  Show usages  SerhiiHladkyi
  return this.cookieService.get('culture');
}

public logout() : void {  Show usages  SerhiiHladkyi
  this.cookieService.deleteAll();
}

public getToken(): string {  Show usages  Sunset1over
  return this.cookieService.get('token') ?? '';
}

public isAuthenticated(): boolean {  Show usages  Sunset1over
  const token : string  = this.getToken();

  return token != '';
}
```

Використання кукі для зберігання інформації на стороні клієнта

```
export const authInterceptor: HttpInterceptorFn = (req : HttpRequ
  const token : string  = inject(UserService).getToken();
  let modifiedReq : HttpRequest<?>  = req;

  if (token) {
    modifiedReq = req.clone( update: {
      setHeaders: {
        Authorization: `Bearer ${token}`
      }
    });
  }
```

Додавання токену до заголовку запитів

Тестування

Karma v 6.4.3 - connected; test: complete;

Chrome 125.0.0.0 (Windows 10) is idle

UserInformationCollectorService
- should return the token from cookies
- should be created
- should return null if not authenticated
- should return user info if authenticated

MusicRecommendationService
- should fetch last recommendations
- should fetch songs by search string
- should fetch recommendations by song IDs
- should be created

UserService
- should get the culture parameter from cookies
- should delete all cookies on logout
- should register a new user
- should log in a user

AppComponent
- should use default language if getCultureParam returns a falsy value
- should have the 'angular-app' title
- should create the app
- should subscribe to cultureChange and use the culture param
- should set default language on init

MusicRecommendationComponent
- should show toastr warning if recommendations fetch fails
- should initialize the form on ngOnInit
- should show toastr info if fetching last recommendations fails
- should search for songs when input changes
- should fetch last recommendations on ngOnInit
- should submit the form and get recommendations
- should create
- should search for second set of songs when input changes
- should handle error in song search gracefully

HeaderComponent
- should reload the page
- should set culture and reload the window
- should return true if user is admin
- should return false if user is not admin
- should create

All files

93.97% Statements 78/83 **83.33%** Branches 10/12 **87.87%** Functions 29/33 **93.67%** Lines 74/79

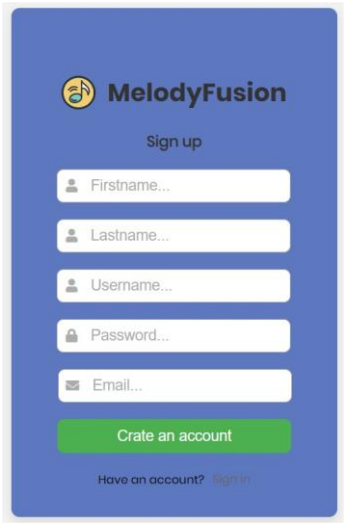
Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File		Statements		Branches		Functions		Lines	
app		100%	9/9	100%	4/4	100%	3/3	100%	8/8
app/environments		100%	2/2	100%	0/0	100%	0/0	100%	2/2
app/pages/music-recommendation/music-recommendation		100%	24/24	100%	3/3	100%	10/10	100%	23/23
app/pages/music-recommendation/services		100%	7/7	100%	0/0	100%	4/4	100%	7/7
app/shared/components/header/header		94.11%	16/17	100%	2/2	80%	4/5	93.75%	15/16
app/shared/services		83.33%	20/24	33.33%	1/3	72.72%	8/11	82.6%	19/23

Інтерфейс системи

Сторінка авторизації

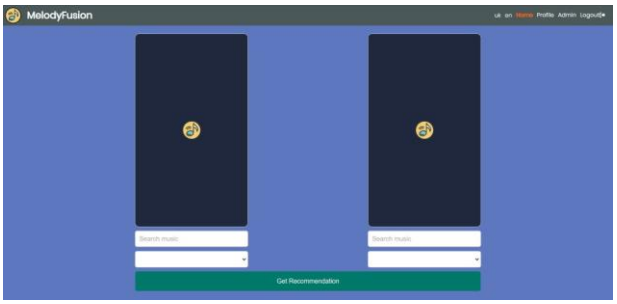


Сторінка реєстрації

15

Інтерфейс системи

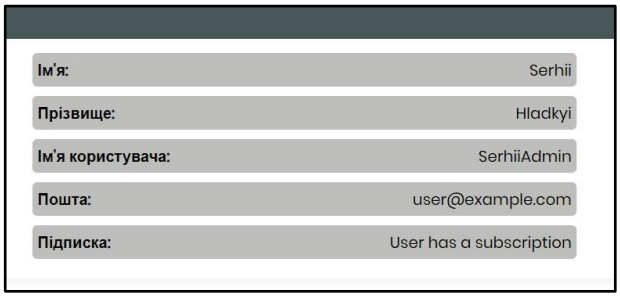
Панель управління адміністратора



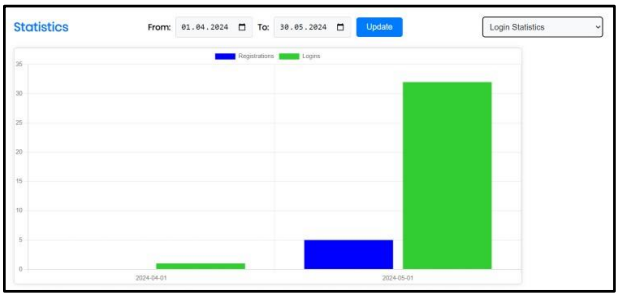
Основна сторінка

16

Інтерфейс системи

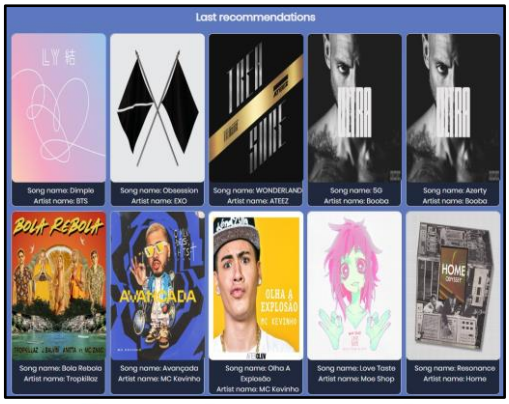


Зміна локалізації на сторінці профілю

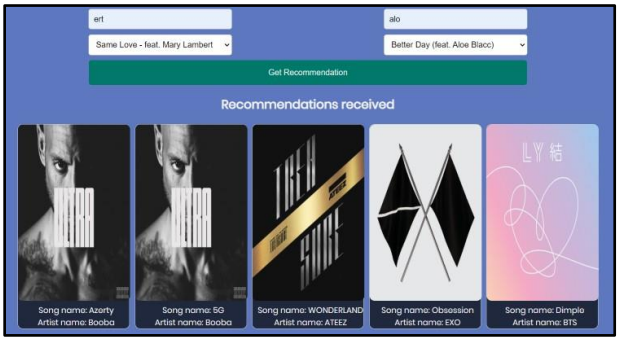


Отримання статистичних даних

Інтерфейс системи



Перегляд історії рекомендацій



Отримання рекомендацій

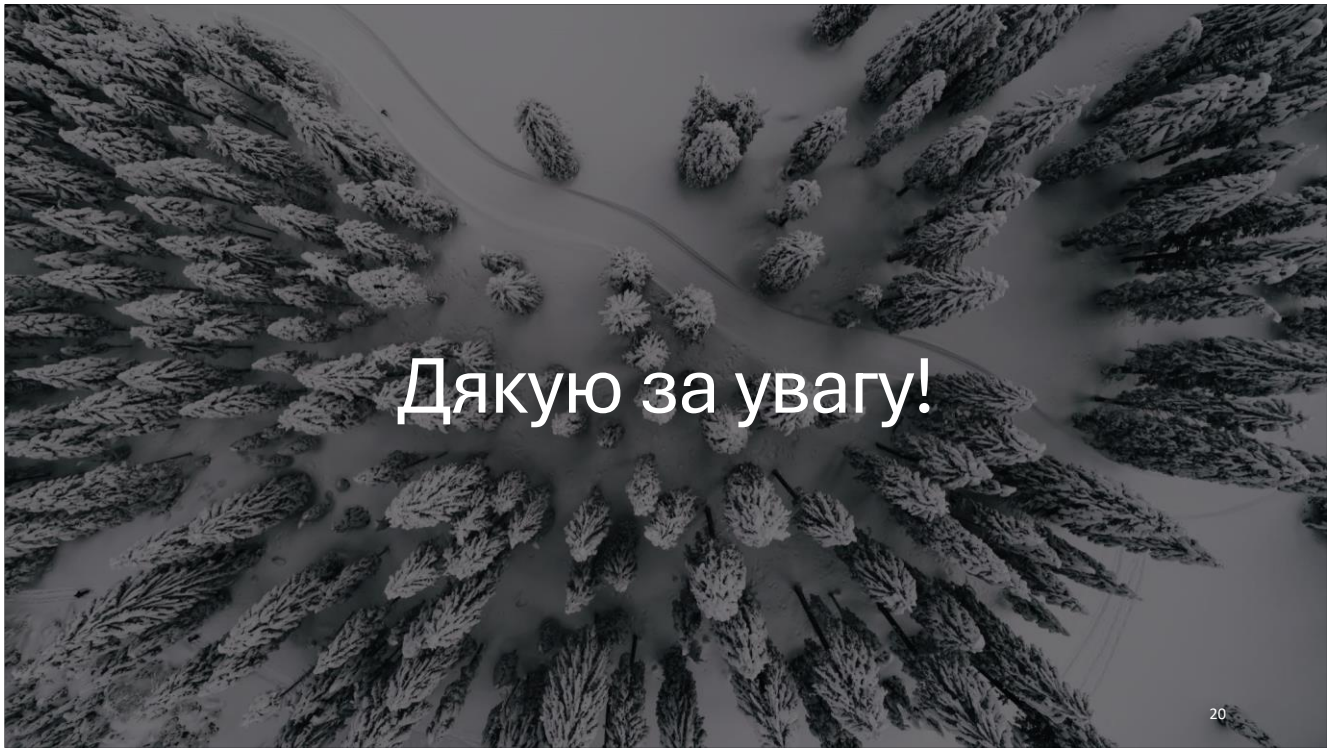
Висновки

- проведено аналіз предметної галузі
- проведено дослідження та визначено вимоги користувачів
- розроблено низькорівневі макети (wireframes) основних сторінок застосунку
- налаштовано проект Angular за допомогою Angular CLI
- реалізовано основні компоненти інтерфейсу користувача
- розроблено сервіси для взаємодії з бекендом
- впроваджено механізми аутентифікації та авторизації
- розроблено тестові сценарії для ключових компонентів та сервісів
- створено алгоритм для визначення музичних інтересів користувачів

Збірки тез доповідей:

Hladkyi S., Afanasieva I. FRONT-END PART DEVELOPMENT FOR INTELLIGENT MUSIC RECOMMENDATION SERVICE. X INTERNATIONAL CONFERENCE Information Technology and Implementation (Satellite), Kyiv, 21 November 2023. P. 248–249.

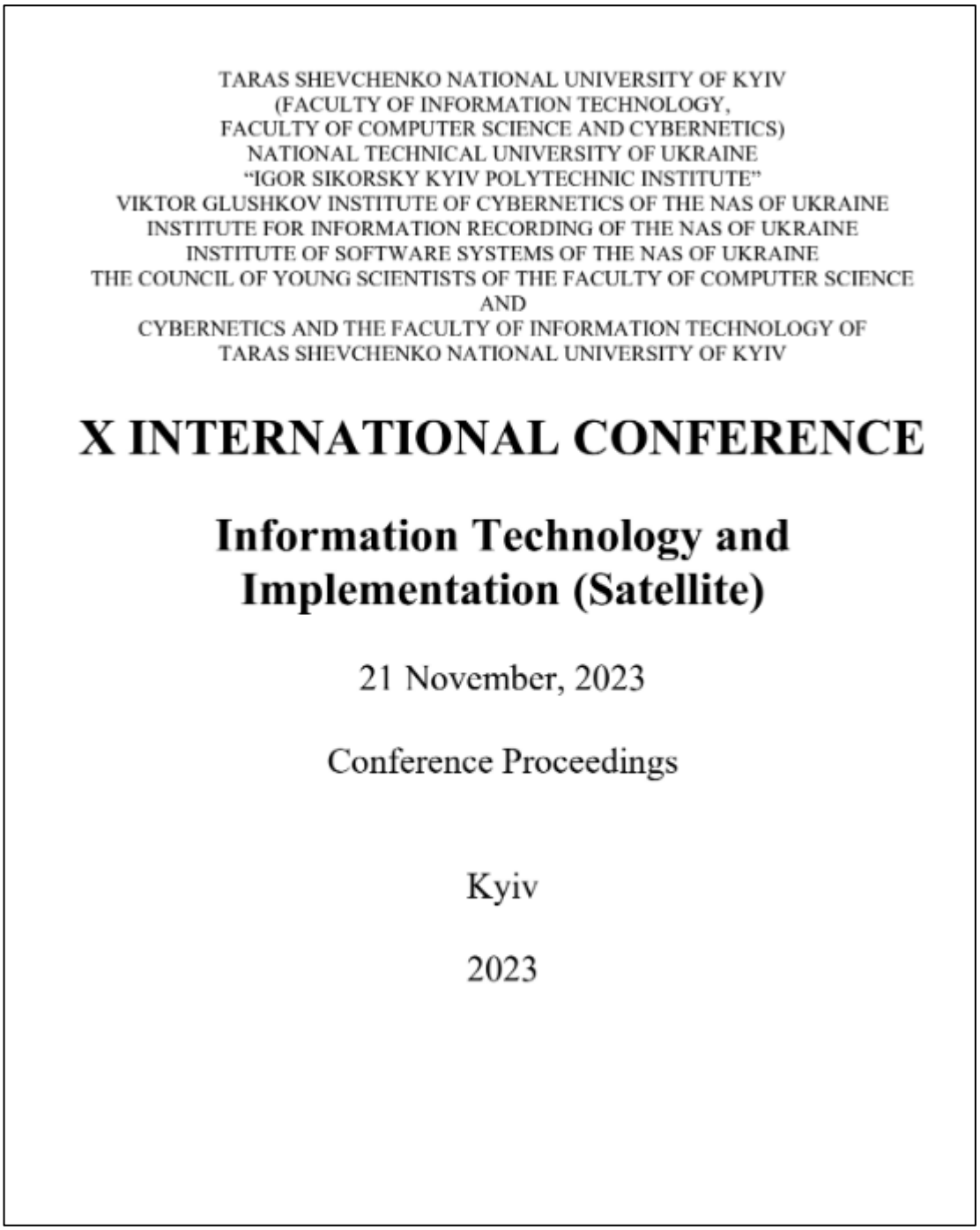
19



20

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

Serhii Hladkyi, Student of Software Engineering Department

Iryna Afanasieva, PhD, Associate Professor of Software Engineering Department
Kharkiv National University of Radioelectronics

FRONT-END PART DEVELOPMENT FOR INTELLIGENT MUSIC RECOMMENDATION SERVICE

Annotation: The research focuses on creating an innovative music recommendation service tailored for both individual and group users with diverse music preferences. Using AI, the service analyzes the characteristics of music tracks to generate recommendations. This approach greatly simplifies the music selection process, providing curated recommendations that align with individual tastes. Analyzing track characteristics and creating vector representations enable more accurate and personalized recommendations.

Keywords: Collaborative Playlist Creation, Artificial Intelligence, Personalized Recommendations, User Interface (UI) Design, Web Interface Design.

Music is an integral part of everyone's life, representing a unique element of style and individuality. However, the rapid pace at which music, especially musicians, emerges can pose a significant challenge. Obtaining a list of tracks that would satisfy all tastes can be difficult and may require the intervention of additional service.

This research emphasizes the importance of creating solutions that consider the individual preferences of each user and the ability to adapt to changing tastes.

The development of a user interface that allows interaction with created intelligent music recommendation service is an equally important stage. The frontend becomes a key aspect, ensuring comfortable and intuitively accessible pathways [6] to personalized musical preferences formed by service, making the music selection process more convenient [5] and efficient.

The frontend is the face of service, through which users will gain access to recommendations and interact with service of musical offerings. A good-looking and user-friendly frontend interface will make the music selection process even more engaging and convenient for each individual user. To develop the interface, Angular [1] technology will be used, which includes the following advantages:

– MVC architecture.
– Modularity.
– Dependency Injection.
– TypeScript [10].

In the development of the frontend, the primary goal is to make the interaction process with service intuitive, fast, and convenient. The user interface will provide easy access to personalized music recommendations, allowing users to easily find music that aligns with their tastes and preferences created using intelligent recommendation service.

The use of artificial intelligence [3] complements the above-described service, and the development of an intelligent recommendation provides an opportunity to personalize approaches to music selection, considering the unique preferences of each user as well as group preferences.

Therefore, the decision was made to create a service that, through artificial intelligence [7], enables the analysis of individual preferences [4] and significantly

Рисунок В.2 – Перша сторінка тез

streamlines this process. To avoid spending time listening to hundreds of thousands of tracks in the hope that they will be liked, users will be able to receive the same list of compositions in a more condensed format.

Therefore, the music recommendation service [8] will be in high demand due to the partial absence of features in analogs [2, 9], and the defined objective is easily achievable using AI.

References:

1. Angular. URL: https://angular.io/docs (date of access: 08.11.2023).
2. Apple Music. URL: https://music.apple.com/us/browse (date of access: 07.11.2023).
3. Bishop C. M. Pattern Recognition and Machine Learning. Springer, 2016. 758 p.
4. Briot J.-P., Hadjeres G., Pachet F.-D. Deep Learning Techniques for Music Generation. Cham : Springer International Publishing, 2020. URL: https://doi.org/10.1007/978-3-319-70163-9 (date of access: 07.11.2023).
5. Krug S. Don't Make Me Think. Was ist Web Usability?. mitp-Verlag, 2002.
6. Norman D. A. Design of Everyday Things. MIT Press, 1998. 270 p.
7. Pachet F. Content management for electronic music distribution. Communications of the ACM. 2003. Vol. 46, no. 4. P. 71–75. URL: https://doi.org/10.1145/641205.641207 (date of access: 07.11.2023).
8. Recommender Systems: An Introduction / D. Jannach et al. Cambridge University Press, 2010. 353 p.
9. Spotify. URL: https://open.spotify.com/ (date of access: 07.11.2023).
10. The starting point for learning TypeScript. URL: https://www.typescriptlang.org/docs/ (date of access: 08.11.2023).

Рисунок В.3 – Остання сторінка тез