Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
магістра
(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ ВИЗНАЧЕННЯ КАТЕГОРІЙ АУДІОПРОГРАМ
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS FOR CATEGORISING THE AUDIO PROGRAMS

Виконав: студент(ка) 2 курсу, групи КНТ-110м
Спеціальності 121 Інженерія програмного забезпечення
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

Алєєв А.Р.
(прізвище та ініціали)

Керівник Зайко Т.А.
(прізвище та ініціали)

Рецензент Гофман Є.О.
(прізвище та ініціали)

2021

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
_____С.О. Субботін
“____” _____2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Алєєва Андрія Романовича
(прізвище, ім’я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів визначення категорій аудіопрограм
Research and Software Implementation of Methods for Categorising the Audio Programs

керівник проєкту (роботи) Зайко Тетяна Анатоліївна, к.т.н., доцент ,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від “ 5 ” листопада 2021 року № 435
2. Строк подання студентом проєкту (роботи) грудень 2021 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Проєктування програмного забезпечення. 4. Основні рішення щодо реалізації компонентів системи. 5. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Зайко Т.А., доцент		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання “ 6 ” вересня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз та дослідження предметної області	2-5 тижні	Розділи 1, 2
3	Розробка архітектури програми	6 тиждень	Розділ 3
4	Розробка програми	7-9 тижні	Розділ 4
5	Тестування та експериментальне дослідження програми	10-11 тижні	Розділ 5
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	12 тиждень	Додатки
7	Захист роботи	13 тиждень	

Студент(ка) _____ Алєєв А.Р.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи) _____ Зайко Т.А.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра: 90 с., 26 рис., 4 табл., 3 дод., 21 джерело.

АУДІОПРОГРАМА, КАТЕГОРИЗАЦІЯ, ЗАДАЧА КЛАСИФІКАЦІЇ, МАШИННЕ НАВЧАННЯ, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, ALEXNET.

Об’єкт дослідження – процес категоризації аудіопрограм.

Предмет дослідження – методи визначення категорій аудіопрограм.

Мета роботи – розробити програмне забезпечення розміщення і прослуховування аудіопрограм з наданням можливості категоризації без участі авторів для підвищення зручності пошуку нових аудіопрограм.

Матеріали, методи та технічні засоби: мова програмування C#, система керування базами даних Microsoft SQL Server, згорткова нейронна мережа, архітектура AlexNet, пакет ConvNetSharp.

Результати. Розроблено програмне забезпечення розміщення і прослуховування аудіопрограм у вигляді вебзастосунку. Наукова новизна роботи полягає в методі визначення категорій аудіопрограм, який відзначається застосуванням згорткових нейронних мереж і встановленням категорій аудіопрограми на основі узагальнення категорій, співставлених з окремими випусками аудіопрограми, що дозволяє розподіляти і об’єднувати ауділрограми без участі їх користувачів.

Висновки. Розроблений вебзастосунок дозволяє користувачам прослуховувати аудіопрограми, створені іншими користувачами, розділяти їх на окремі сезони і випуски, виконувати їх пошук, переглядати найбільш рейтингові аудіопрограми або аудіопрограми за категоріями, оцінювати, співвідносити аудіопрограми з їх категоріями, підписуватися на прослуховування.

Галузь використання – розповсюдження аудіоконтенту у вигляді аудіопрограм.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 90 pages, 26 figures, 4 tables, 3 appendixes, 21 sources.

AUDIO PROGRAM, CATEGORIZATION, CLASSIFICATION PROBLEM, MACHINE LEARNING, ROLLED NEURAL NETWORK, ALEXNET.

The object of research is the process of categorization of audio programs.

The subject of research is methods for categorisinf the audio programs.

The purpose of the work is to develop software for hosting and listening to audio programs with the possibility of categorization without the participation of authors to increase the convenience of finding new audio programs.

Materials, methods and technical means: C# programming language, Microsoft SQL Server database management system, convolutional neural network, AlexNet architecture, ConvNetSharp package.

Results. Software for hosting and listening to audio programs in the form of a web application has been developed. The scientific novelty of the work is the method of categorising audio programs, which is characterized by the use of convolutional neural networks and categarization of audio programs based on the generalization of categories of individual releases of audio programs, enabling to divide and unite audio programs without the help of its authors.

Conclusions. The developed web application enables users to listen to audio programs created by other users, divide them into separate seasons and releases, search for them, view the highest rated audio programs or audio programs by category, evaluate, correlate audio programs with their categories, subscribe to audio program listening.

Area of use is distribution of audio content in the form of audio programs.

ЗМІСТ

С.

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ReLU – rectified linear unit;

БД – база даних;

ЗНМ – згорткова нейронна мережа.

ВСТУП

Загалом близько 28% європейців слухали аудіопрограми протягом останнього місяця, коли Інститут журналістики Reuters наприкінці минулого року опитував респондентів у п’ятнадцяти різних країнах [1]. Така надзвичайна популярність і активний розвиток даної сфери призводять до необхідності створення відповідних платформ для розміщення аудіопрограм.

Окрім актуальності самих програмних засобів варто окремо зазначити, що актуальною стає і потреба вмісту, який може стати основним чинником, який привертає увагу слухачів до застосунку. Але зі збільшенням обсягів вмісту важливим стає розподілення аудіопрограм таким чином, щоб потім можна було шукати ті, які зацікавлять користувача. За великої кількості даних знайти без структуризації потрібні дані неможливо. Тому щодо завантаженого авторами вмісту аудіопрограм важливо таке розподілення виконати автоматично. Тоді автор може отримати можливість визначити категорії аудіопрограм, але сама програма зможе робити це і автоматично. Тоді, якщо автор не визначив категорії, програма зможе все одно розподілити вміст, з іншого боку програма може таким чином порадити розподіл автору. Окрім того можна таким чином визначити, чи варто довіряти розподілу, виконаному автором.

Користуючись даною аргументацією і актуальністю теми, що розглядається, метою роботи було встановлено розробити програмне забезпечення розміщення і прослуховування аудіопрограм з наданням можливості категоризації без участі авторів для підвищення зручності пошуку нових аудіопрограм.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Аналіз програмних засобів розміщення і прослуховування аудіопрограм

На даний момент існує достатньо багато програм, які надають доступ до прослуховування та розміщення аудіопрограм. Це можуть бути як окремі сервіси, так і можливості на деяких вебсайтах для розміщення власних аудіопрограм авторів без можливості вільного доступу зовнішніх авторів.

Аудіопрограми науково-популярного журналу «Куншт» (рис. 1.1) можуть бути прикладом сервісів, де розміщуються аудіопрограми обмеженого кола авторів [2].

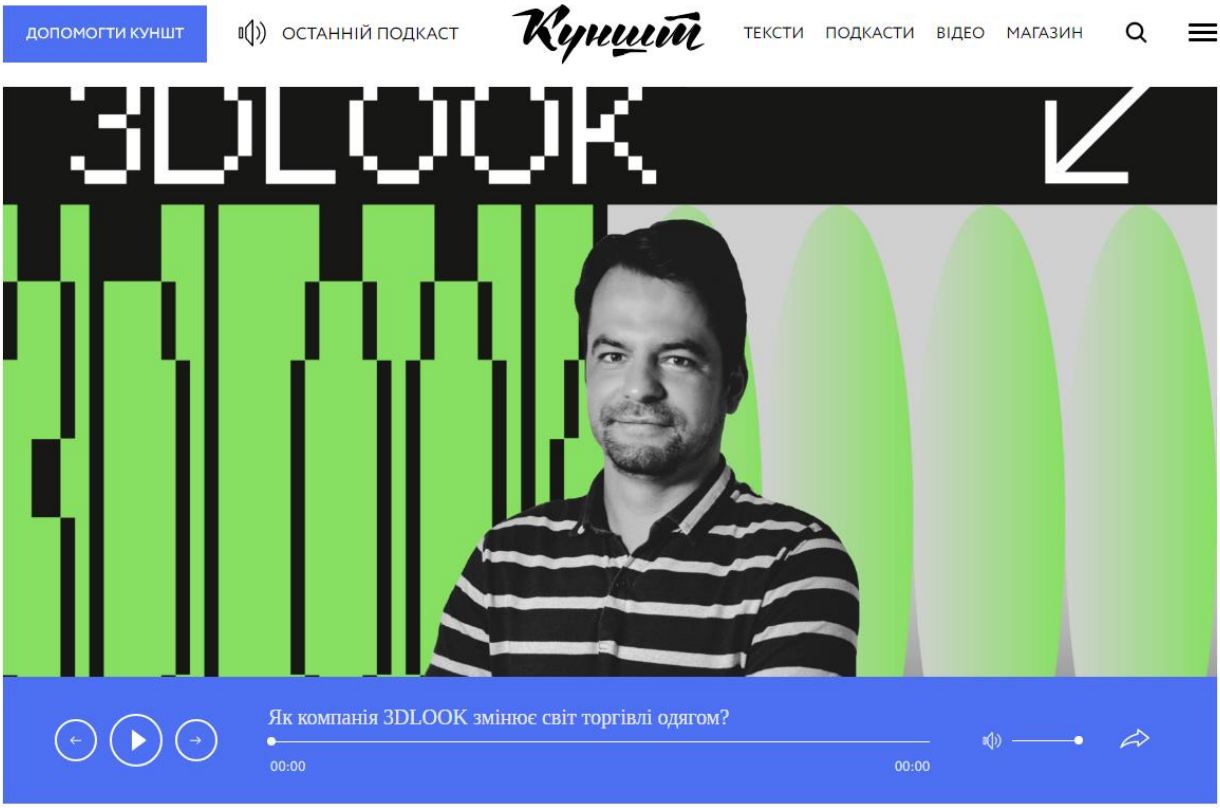


Рисунок 1.1 – Аудіопрограми журналу «Куншт»

В якості прикладу вебсайту хостингу аудіопрограм з доступом зовнішніх авторів розглянемо Podbean (рис. 1.2) [3].

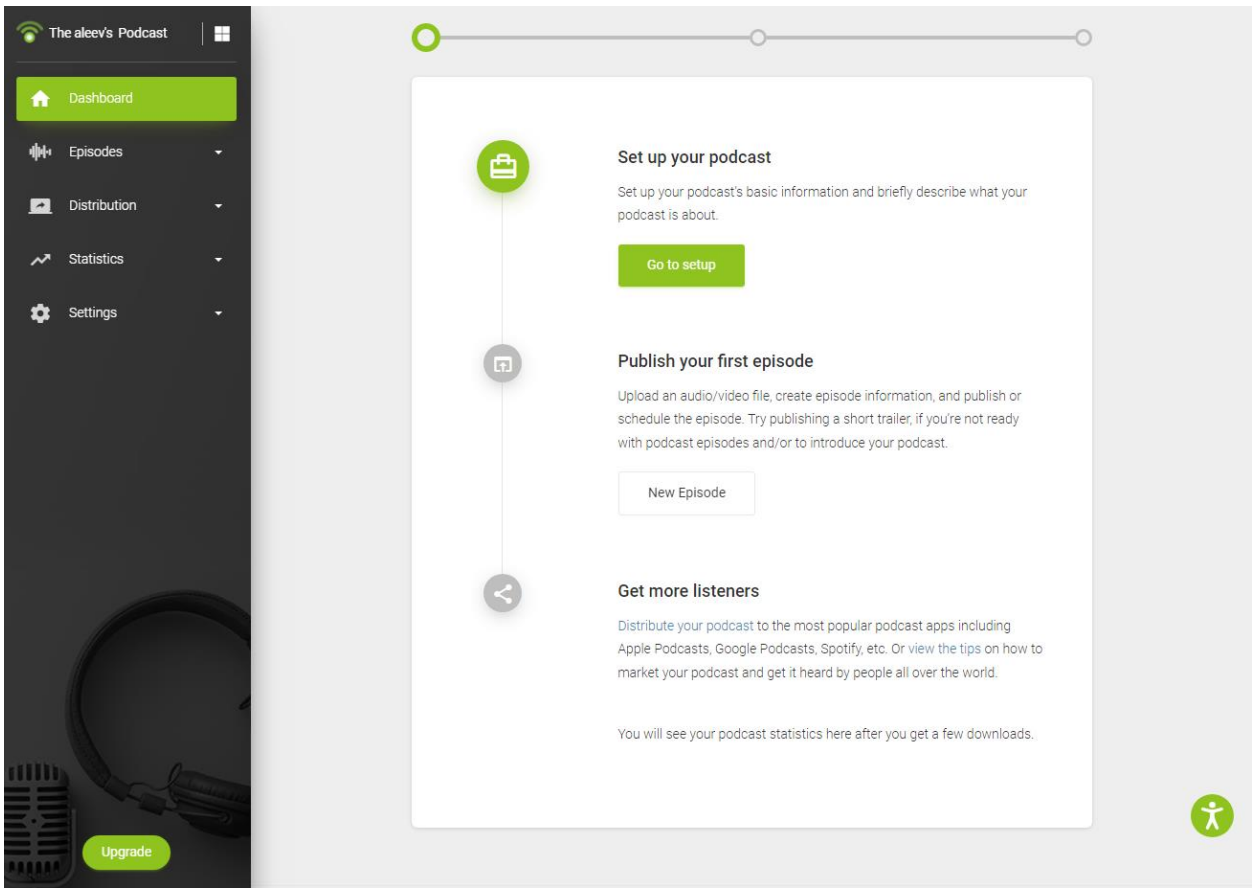


Рисунок 1.2 – Програма Podbean

Програма відзначається наступними характеристиками:

– надає чотири різні плани для користувачів, включаючи одну повністю безкоштовну версію;

– зараз у Podbean налічується понад 240 000 авторів аудіопрограм із понад 7,4 мільйонами випусків загалом, при цьому нараховано 6 мільярдів завантажень;

– основною послугою Podbean є хостинг аудіопрограм: це означає, що можна завантажувати випуски аудіопрограм на Podbean, і коли хтось захоче їх прослухати, він зможе завантажити або передати аудіодані з серверів Podbean;

– Podbean тільки розміщує вміст користувача і має право його поширювати та просувати: автор (на додаток до будь-яких співорганізаторів) автоматично володіє вмістом, єдиний спосіб, за якого Podbean міг би

володіти вмістом аудіопрограм автора, – якщо він явно передав або продав аудіопрограму сервісу;

– має наступні недоліки: безкоштовна версія обмежена 5 годинами зберігання, для запису, створення, завантаження та поширення нових подкастів необхідні деякі базові знання з програмування [4]-[5].

Transistor – ще одна програма для розміщення аудіопрограм, але з приватним платним доступом [6].

Приватні аудіопрограми підходять для тих, хто хоче, щоб їхній вміст слухали лише учасники або, можливо, навіть стягувалась плата за потік кожного вмісту, приватні аудіопрограми також широко використовуються компаніями, які хочуть надавати безпечний контент своїм співробітникам або акціонерам: Transistor ідеально підходить для корпоративних та великих брендів, хоча тарифні плани також доступні для початківців [7].

Прикладом ще однієї подібної програми є Captivate [8].

Captivate дозволяє запускати та розміщувати скільки завгодно аудіопрограм, окрім того має деякі зручні особливості інтерфейсу і функцій, що надаються:

– програвач автоматично використовує кольори обкладинки для встановлення кольорів програвача за замовчуванням

– надається пакетне завантаження аудіофайлів;

– надається необмежена кількість аудіопрограм;

– наявність приватних аудіопрограм на кожному тарифному плані [9].

Проаналізовані результати порівняння програмних аналогів, включаючи Podbean, Transistor, зведені до табл. 1.1.

Основною відмінністю програми, що розробляється, є можливість автоматичного визначення категорій аудіопрограм та надання безкоштовного вільного доступу до розміщення та прослуховування аудіопрограм. Останнє зокрема потрібне для початкового привертання уваги до програми.

Таблиця 1.1 – Порівняння програмних аналогів

Характеристика	Podbean	Transistor	Програма, що розроб-ляється
Завантаження аудіопрограм на сервер	+	+	+
Зручність для нових авторів аудіопрограм	+	+	+
Тип вмісту	Аудіопрограми та відеопрограми	Аудіопрограми	Аудіо-програми
Безкоштовний доступ	Тимчасово	–	+
Автоматичне визначення категорій аудіопрограм	–	–	+

1.2 Функціональні вимоги до програми

Розроблювана програма повинна забезпечувати виконання наступних функцій:

– відображення списку категорій аудіопрограм;

– відображення аудіопрограм за категорією;

– пошук аудіопрограм;

– створення аудіопрограми;

– редагування аудіопрограми;

– автоматичне визначення категорії аудіопрограми;

– оцінювання аудіопрограми;

– оцінювання випуску аудіопрограми;

– відображення списку підписників на аудіопрограму;

– відображення списку аудіопрограм, на які підписався користувач;

– відображення списку уподобаних випусків користувачем;

– відображення списку власних аудіопрограм;

– відображення списку аудіопрограм користувача;

– відображення списку нових аудіопрограм;

– створення рейтингу аудіопрограм;

– створення рейтингу випусків;

– створення сезонів аудіопрограми;

– підписування на аудіопрограму;

– уподобання випуску аудіопрограми;

– відображення списку з’єднаних випусків з даним;

– сортування списку аудіопрограм;

– відображення аудіопрограм за містом створення;

– відображення випусків, у яких брав участь гість;

– відображення списку сезонів аудіопрограми;

– відображення списку випусків за сезоном аудіопрограми;

– обчислення середньої оцінки аудіопрограми, сезону та випуску;

– редагування даних про сезон аудіопрограми;

– публікація випусків аудіопрограми;

– відкладена публікація випусків аудіопрограми;

– авторизація користувачів;

– реєстрація користувачів.

1.3 Висновки за розділом 1

У першому розділі було проаналізовано програмні аналоги, що включають зокрема Podbean, Transistor. Виконано порівняння аналогів з програмою, що розробляється. Визначено в якості її основних відмінностей можливість автоматичного визначення категорій аудіопрограм та надання безкоштовного вільного доступу до розміщення та прослуховування аудіопрограм.

У роботі в наступних розділах необхідно розв’язати наступні завдання:

– провести дослідження методів визначення категорій аудіопрограм;

– виконати проєктування програмного забезпечення;

– реалізувати програмне забезпечення.

2 МАТЕРІАЛИ І МЕТОДИ

Нейронні мережі є актуальним сучасним засобом вирішення багатьох існуючих проблем. Для складних задач зазвичай потрібні декілька прихованих шарів у нейронній мережі прямого зв’язку, оскільки простіша нейронна мережа може не мати змоги вивчити модель, яка відображає вхідні дані на вихідні дані в навчальних даних, при цьому наявність декількох прихованих шарів посилює проблему наявності багатьох ваг у мережі, а наявність великої кількості ваг ускладнює процес навчання, оскільки розмір простору пошуку збільшується, до того ж ця проблема ще більше ускладнюється для кольорових зображень (на відміну від зображень у градаціях сірого, кожен піксель у кольоровому зображенні представлений трьома значеннями, що представляють червоний, зелений та синій кольори), де кожен колір може бути представлений різними комбінаціями цих основних кольорів, окрім того двовимірне зображення у таких нейронних мережах представляється як одновимірний вектор у вхідному шарі, отже, будь-які просторові зв’язки в даних ігноруються [10]. Згорткові нейронні мережі (ЗНМ) підтримує просторову структуру даних і краще підходить для пошуку просторових відносин у даних зображення [10].

2.1 Згорткова нейронна мережа

Архітектура ЗНМ формується послідовністю будівельних блоків для виділення ознак, які визначають належний клас, що складається з одного або декількох шарів:

– згорткові шари – це набір згорток, при цьому декілька згорткових ядер проходять через зображення, що веде до декількох вихідних карт ознак, кожне згорткове ядро має параметри, специфічні для інформації, яка шукається в зображенні (наприклад: згорткове ядро може мати параметри для пошуку контурів на зображенні), вибір параметрів згорткового ядра

залежить від завдання, яке необхідно розв’язувати, при методах глибокого навчання ці параметри автоматично вивчаються алгоритмом із навчальних даних, при цьому функція втрат обчислює похибку між прогнозованим і цільовим значенням;

– корекційні шари, які часто називаються за функцією активації rectified linear unit (ReLU): рівень корекції або активації – це застосування нелінійної функції до карт ознак на виході згорткового шару, роблячи дані нелінійними, він полегшує виділення складних ознак, які неможливо змоделювати за допомогою лінійної комбінації алгоритму регресії;

–об’єднання шарів, яке стискає інформацію шляхом зменшення розміру проміжного зображення (часто шляхом підвибірки): як правило, шар об’єднання вставляється через регулярні інтервали між корекційним і згортковим шарами, зменшуючи розмір карт ознак, а отже, і кількість параметрів мережі, це прискорює час обчислень і зменшує ризик перенавчання, найпоширенішою операцією об’єднання є MaxPool (2x2, 2), оскільки вона максимізує вагу сильних активацій, застосовується на виході попереднього шару як фільтр згортки розміру 2 x 2 і рухається з кроком 2, на виході шару об’єднання отримується карта ознак, стиснута в 4 рази [11].

Будівельні блоки змінюють один одного, повторюючись до кінцевих шарів мережі, які виконують класифікацію зображень і обчислення похибки між прогнозом і цільовим значенням:

– повнозв’язний шар, який є персептроноподібним шаром: цей шар знаходиться в кінці мережі, це дозволяє класифікувати зображення на основі характеристик, витягнутих за допомогою послідовності блоків обробки, він повністю зв'язаний, оскільки всі входи шару з'єднані з вихідними нейронами шару, вони мають доступ до всієї вхідної інформації, кожен нейрон присвоює зображенню значення ймовірності приналежності до відповідного класу, що протилежно попереднім шарам, які відносяться до фази виділення ознак, коли нейрони незалежні один від одного і мають доступ лише до інформації рецептивного поля, яке вони обробляють;

– шар втрат є останнім шаром мережі: він обчислює похибку між прогнозом мережі та фактичним значенням, під час завдання класифікації випадкова величина є дискретною, оскільки вона може приймати лише значення 0 або 1, 1 означає, що вона належить до класу, а 0 означає, що її немає в даному класі [11]-[12].

AlexNet – це архітектура ЗНМ, здатна досягати високої точності на дуже складних наборах даних, однак видалення будь-якого з згорткових шарів різко погіршить продуктивність AlexNet, AlexNet є провідною архітектурою для будь-якого завдання виявлення об’єктів, при цьому за першими двома згортковими шарами слідує шар підвибірки (максимальне об’єднання), а далі – третій, четвертий і п'ятий шари згортки безпосередньо пов'язані один з одним, за п’ятим згортковим шаром слідує шар підвибірки (максимальне об’єднання), який потім з’єднується з повністю з’єднаними шарами [13]-[17].

Структура ЗНМ архітектури AlexNet зведена до табл. 2.1.

Таблиця 2.1 – Структура ЗНМ архітектури AlexNet

Шар	Кількість нейронів	Розмір фільтру	Функція активації
1	2	3	4
Вхідний шар	–	–	–
Згортковий шар	96	11 x 11	ReLU
Підвибірка	–	3 x 3	–
Згортковий шар	256	5 x 5	ReLU
Підвибірка	–	3 x 3	–
Згортковий шар	384	3 x 3	ReLU
Згортковий шар	384	3 x 3	ReLU
Згортковий шар	256	3 x 3	ReLU
Підвибірка	–	3 x 3	–

Продовження таблиці 2.1.

1	2	3	4
Виключення	Коефіцієнт – 0,5	–	–
Повнозв’язний шар	–	–	ReLU
Виключення	Коефіцієнт – 0,5	–	–
Повнозв’язний шар	–	–	ReLU
Повнозв’язний шар	–	–	Softmax

2.2 Метод визначення категорій аудіопрограм

ЗНМ є вдалим засобом для розпізнавання зображень, який активно використовується на практиці. Даний засіб може бути застосований і для роботи зі звуковими файлами. Для такого застосування потрібно перетворити звукові дані на графічні. Для цього створюється спектрограма, яка як раз і є графічним представленням даних. Спектрограма подає візуальне представлення спектра, що складається з частот звукового сигналу, що змінюються з часом. Вже спектрограма використовується для роботи ЗНМ.

Запропонований для визначення категорій аудіопрограм на основі створених спектрограм метод представлений наступними етапами.

На першому етапі виконується створення категоризованої вибірки фрагментів. Для цього спочатку визначається список категорій аудіопрограм. Далі визначається список фрагментів звукових файлів з визначеною довжиною L. Для кожного фрагмента створюється спектрограма, а потім визначаються категорії, до яких належить фрагмент.

На другому етапі спочатку виконується створення моделі на основі ЗНМ з архітектурою AlexNet. Далі реалізується навчання моделі за категоризованою вибіркою фрагментів, створеною на першому етапі.

Далі відбувається формування списку випусків аудіопрограми. До нього об’єднуються всі випуски даної аудіопрограми.

Після цього циклічно обробляється кожен випуск аудіопрограми зі сформованого списку. Кожен випуск розділяється на послідовність фрагментів довжини *L*. Для кожного такого фрагмента має бути сформовано спектрограму.

Для кожного виділеного фрагменту за навченою на другому етапі моделлю виконується визначення категорії, до якої даний фрагмент належить.

Після цього відбувається узагальнення виділених категорій за всіма фрагментами випуску. Таким чином має бути реалізовано голосування за категорії випуску.

Голосування може виконуватися за принципом більшості (за категорію має бути проголосовано більшістю фрагментів).

Також може використовуватися принцип переможця, коли визначається тільки одна категорія, яка отримала голоси більшості фрагментів. Якщо категорія не отримала більшість голосів, то з випуском може бути не співставлена категорія в такому випадку. Можливо також задати значення *K*. Тоді *K* категорій, які отримали найбільшу кількість голосів, визначаються асоційованими з даним випуском.

Такі дії повторюються для кожного випуску аудіопрограми. У результаті відбувається голосування за категорії аудіопрограми. Наприклад, у випадку, якщо з кожним фрагментом було співставлено декілька категорій, то зі всією аудіопрограмою співставляються категорії, які отримали більшість голосів (були співставлені з більшістю випусків). Отриманий результат і є результатом роботи всього методу.

Запропонований метод може бути узагальнений схемою роботи, представленою на рис. 2.1.

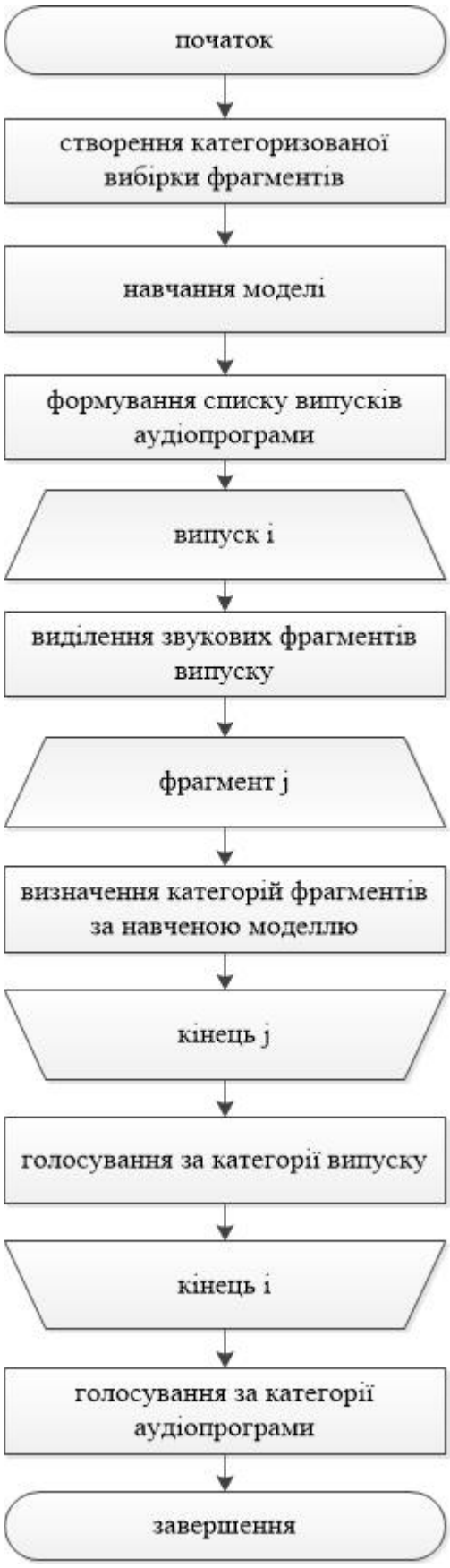


Рисунок 2.1 – Схема роботи методу визначення категорій аудіопрограм

2.3 Висновки за розділом 2

У четвертому розділі досягнуто наступних результатів:

– проаналізовано нейронні мережі, придатні для аналізу зображень;

– виділено особливості архітектури ЗНМ AlexNet;

– створено і описано метод визначення категорій аудіопрограм, який відзначається застосуванням ЗНМ і встановленням категорій аудіопрограми на основі узагальнення категорій, співставлених з окремими випусками аудіопрограми, що дозволяє розподіляти і об’єднувати аудіопрограми без участі їх користувачів.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір мови програмування

Вибір мови програмування дуже важливий для подальшого проєктування і розробки, оскільки він визначає послідовність наступних рішень. У даному проєкті важливим є те, що в результаті буде створено програмну систему, яка об’єднує в собі вебзастосунок і реалізацію машинного навчання.

Вибір мови програмування для реалізації виконувався серед мов C# і Python (табл. 3.1) [18]-[19].

Таблиця 3.1 – Порівняння мов програмування

Характеристика	C#	Python
Вебфреймворк	ASP.NET Core MVC	Django
Високорівнева мова	+	+
Швидкість роботи програм	+	+/–
Безпека	+	+/–
Робота з засобами машинного навчання	+	+
Створення великих проєктів	+	+/–

Слід зазначити, що на даний момент обидві мови підтримують веброзробку та реалізацію машинного навчання. При цьому мова програмування C# дозволяє створювати проєкти, які мають вищу швидкість роботи порівняно з мовою Python, окрім того рівень безпеки створених застосунків також вищий для мови C#. Також вона дозволяє розробляти великі проєкти, яким як раз і є програмна система, яка створюється, адже з нею в результаті може працювати велика кількість користувачів.

3.2 Моделювання роботи з програмою

Усіх користувачів можна розбити на 2 групи: авторизовані та неавторизовані. Будь-який авторизований користувач може як прослуховувати аудіопрограми інших користувачів, так і створювати власні.

Неавторизований користувач може переглядати інформацію за категоріями, аудіопрограмами, їх сезонами і випусками, прослуховувати самі випуски (рис. 3.1). При моделюванні всі сценарії були об’єднані у відповідні групи.

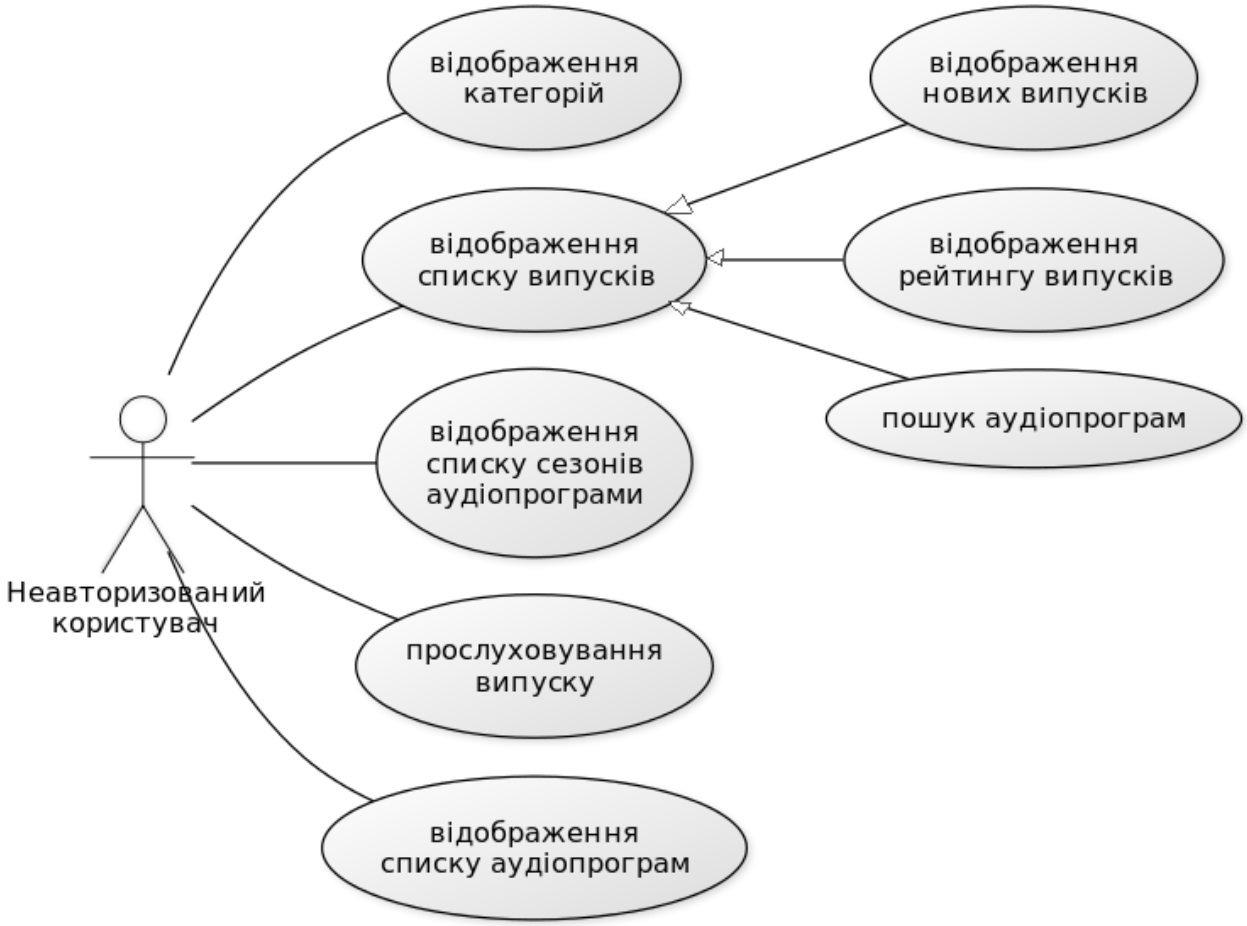


Рисунок 3.1 – Діаграма варіантів використання програми неавторизованим користувачем

Авторизований користувач може створювати власні аудіопрограми і керувати ними, в тому числі створювати і керувати сезонами, публікувати випуски цих аудіопрограм (рис. 3.2).

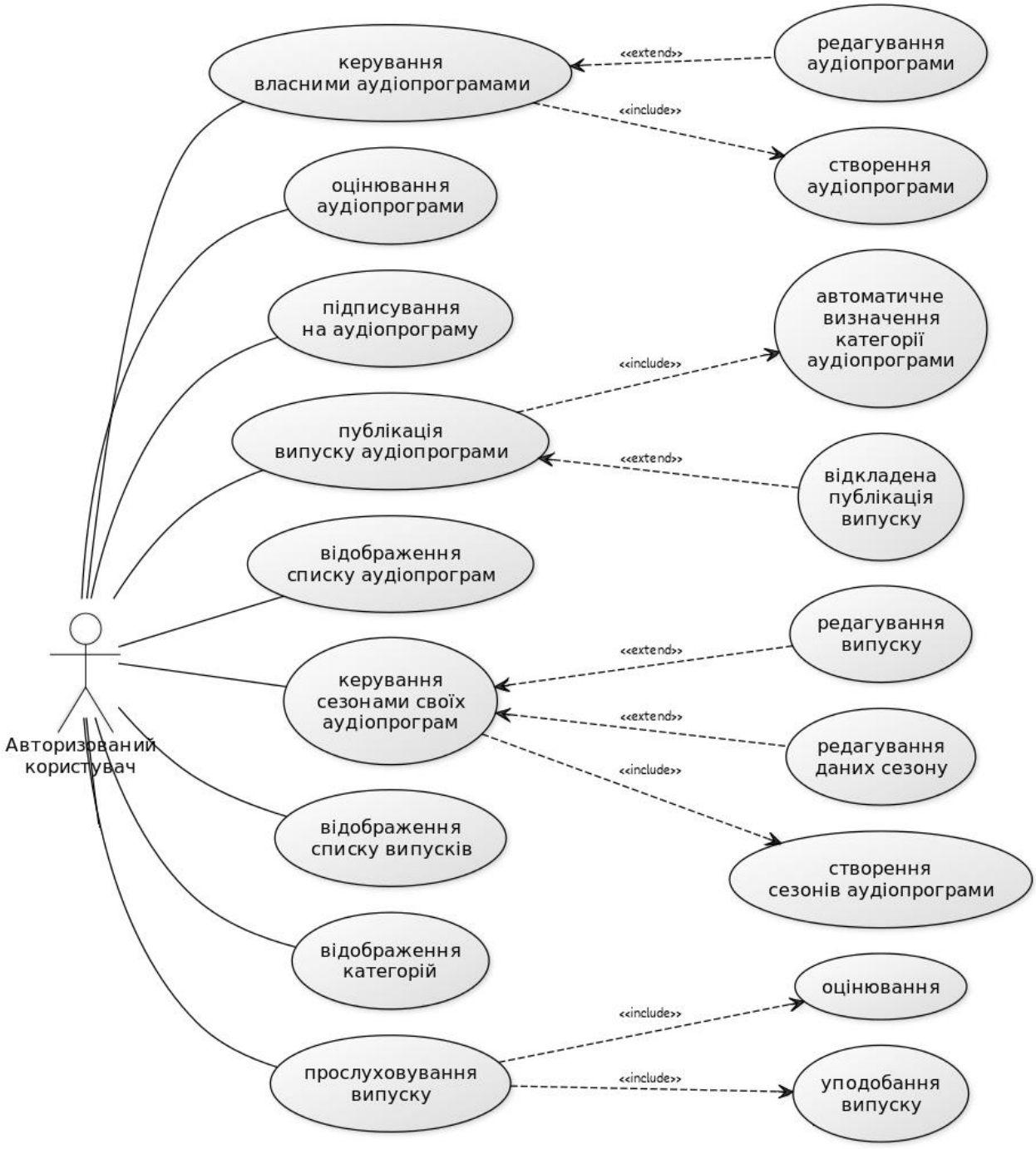


Рисунок 3.2 – Діаграма варіантів використання програми авторизованим користувачем

3.3 Проєктування бази даних

Для визначення роботи з даними необхідно визначити структуру бази даних (БД) на основі використання системи керування БД Microsoft SQL Server.

Основні сутності предметної області:

– AudioProgram – аудіопрограма;

– Season – сезон аудіопрограми, що об’єднує послідовність випусків аудіопрограми;

– Issue – випуск аудіопрограми;

– Follow – підписка на аудіопрограму;

– Account – акаунт користувача програми;

– Place – місто, де створюється аудіопрограма;

– Category – категорія аудіопрограми;

– AudioProgramScore – оцінка аудіопрограми;

– IssueScore – оцінка випуску аудіопрограми;

– Guest – гість випуску аудіопрограми, який може бути або не бути користувачем програми;

– Favor – уподобання випуску.

Таблиця AudioProgram включає наступні поля:

– AudioProgramId – номер аудіопрограми (первинний ключ таблиці);

– Name – назва аудіопрограми;

– Info – опис аудіопрограми;

– Cover – обкладинка аудіопрограми;

– Status – стан аудіопрограми (активна чи архівна);

– PubDate – дата публікації аудіопрограми вперше;

– Followers – кількість підписників на аудіопрограму;

– AccountId – автор аудіопрограми, що представлений зовнішнім ключем з таблиці Account;

– Place – місто, де створюється аудіопрограма, що представлений зовнішнім ключем з таблиці Place;

– Score – середня оцінка, виставлена користувачами аудіопрограмі.

Один користувач з відповідним акаунтом може мати декілька аудіопрограм, одна аудіопрограма може мати тільки одного автора – зв’язок один до багатьох між таблицями Account і AudioProgram.

Таблиця Season включає наступні поля:

– SeasonId – номер сезону аудіопрограми (первинний ключ);

– AudioProgramId – аудіопрограма, яку представляє даний сезон, є зовнішнім ключем таблиці AudioProgram;

– Name – назва сезону;

– Start – дата початку публікації випусків сезону;

– End – дата завершення публікації випусків сезону;

– Score – середня оцінка за випусками сезону.

Одна аудіопрограма може мати декілька сезонів, кожен сезон представляє тільки одну аудіопрограму – зв’язок один до багатьох між таблицями AudioProgram і Season.

Таблиця Issue включає наступні поля:

– IssueId – номер випуску (первинний ключ);

– SeasonId – сезон, до якого належить випуск аудіопрограми, є зовнішнім ключем таблиці Season;

– Name – назва випуску;

– Publication – дата публікації випуску;

– Info – опис випуску аудіопрограми;

– Cover – обкладинка аудіопрограми;

– Status – стан випуску аудіопрограми;

– Start – дата, з якої буде доступним для прослуховування випуск аудіопрограми;

– Favorites – кількість уподобань випуску аудіопрограми;

– Audiofile – файл аудіопрограми;

– Score – середня оцінка випуску аудіопрограми.

Один сезон аудіопрограми може мати декілька випусків, кожен випуск належить тільки одному конкретному сезону конкретної аудіопрограми – зв’язок один до багатьох між таблицями Season і Issue.

Таблиця Follow включає наступні поля:

– FollowId – номер підписки (первинний ключ);

– AudioProgramId – аудіопрограма, до якої відноситься підписка, є зовнішнім ключем таблиці AudioProgram;

– AccountId – користувач, який підписався на аудіопрограму, є зовнішнім ключем таблиці Account;

– Publication – дата підписки;

– Status – стан підписки (активний чи неактивний).

Одна програма може мати декілька підписок, кожна підписка створена на одну конкретну програму – зв’язок один до багатьох між таблицями AudioProgram і Follow. Один користувач може мати декілька підписок, кожна підписка створена одним конкретним користувачем – зв’язок один до багатьох між таблицями Account і Follow.

Таблиця Account включає наступні поля:

– AccountId – номер акаунта (первинний ключ);

– Login – логін акаунта;

– Password – пароль акаунта;

– Avatar – зображення акаунта.

Таблиця Place включає наступні поля:

– PlaceId – номер міста (первинний ключ);

– Name – назва міста;

– Country – країна, в якій знаходиться місто.

Таблиця Category включає наступні поля:

– CategoryId – номер категорії (первинний ключ);

– Name – назва категорії.

Таблиця AudioProgramCategory включає наступні поля:

– AudioProgramCategoryId – номер запису про категорію аудіопрограми (первинний ключ);

– CategoryId – категорія, що характеризує аудіопрограму, є зовнішнім ключем таблиці Category;

– AudioProgramId – аудіопрограма, якій належить категорія, є зовнішнім ключем таблиці AudioProgram;

– Source – джерело, за яким було внесено дану категорію (вручну, на основі методу або подвійне визначення);

– Voices – кількість випусків аудіопрограми, які були співставлені з цією категорією.

Одна програма може мати декілька категорій – зв’язок один до багатьох між таблицями AudioProgram і AudioProgramCategory. Одна категорія може характеризувати декілька аудіопрограм – зв’язок один до багатьох між таблицями Category і AudioProgramCategory.

Таблиця Connection визначає зв’язки між випусками і включає наступні поля:

– SrcIssueId – випуск, для якого встановлено зв’язок, є зовнішнім ключем таблиці Issue;

– IssueId – випуск, до якого направлений зв’язок (який має відображатися в апру до випуску SrcIssueId), є зовнішнім ключем таблиці Issue.

Таблиця AudioProgramScore включає наступні поля:

– AudioProgramScoreId – номер оцінювання (первинний ключ);

– AudioProgramId – аудіопрограма, якій виставлена дана оцінка, є зовнішнім ключем таблиці AudioProgram;

– AccountId – користувач, який виставив оцінку, є зовнішнім ключем таблиці Account;

– Publication – дата створення оцінки;

– Score – виставлена оцінка.

Одна аудіопрограма може мати декілька оцінок – зв’язок один до багатьох між таблицями AudioProgram і AudioProgramScore. Один користувач може мати декілька оцінок – зв’язок один до багатьох між таблицями Account і AudioProgramScore.

Таблиця IssueScore включає наступні поля:

– IssueScoreId – номер оцінювання (первинний ключ);

– AccountId – користувач, який виставив оцінку, є зовнішнім ключем таблиці Account;

– IssueId – випуск аудіопрограми, який було оцінено, є зовнішнім ключем таблиці Issue;

– Publication – дата створення оцінки;

– Score – виставлена оцінка.

Один випуск аудіопрограми може мати декілька оцінок – зв’язок один до багатьох між таблицями Issue і IssueScore. Один користувач може мати декілька оцінок – зв’язок один до багатьох між таблицями Account і IssueScore.

Таблиця Guest включає наступні поля:

– GuestId – номер гостя (первинний ключ);

– Name – ім’я гостя випуску;

– AccountId – акаунт користувача, який належить даному гостю, є зовнішнім ключем таблиці Account;

– Site – вебсайт гостя.

Один гість може мати 0 або 1 акаунт користувача – зв’язок один до одного між таблицями Account і Guest.

Таблиця IssueGuest включає наступні поля:

– IssueGuestId – номер запису про участь гостя у випуску аудіопрограми (первинний ключ);

– IssueId – випуск аудіопрограми, є зовнішнім ключем таблиці Issue;

– GuestId – гість випуску, є зовнішнім ключем таблиці Guest.

Створена схема бази даних представлена на рис. 3.3.

Рисунок 3.3 – Схема БД

Таблиця Favor включає наступні поля:

– FavorId – номер уподобання випуску (первинний ключ);

– IssueId – випуск аудіопрограми, є зовнішнім ключем таблиці Issue;

– AccountId – акаунт користувача, який уподобав випуск, є зовнішнім ключем таблиці Account;

– Publication – дата створення уподобання випуску;

– Status – стан уподобання (активний чи неактивний).

Один випуск аудіопрограми може мати декілька уподобань – зв’язок один до багатьох між таблицями Issue і Favor. Один користувач може мати декілька уподобань – зв’язок один до багатьох між таблицями Account і Favor.

Один випуск аудіопрограми може мати декілька гостей – зв’язок один до багатьох між таблицями Issue і IssueGuest. Один гість може брати участь в декількох випусках аудіопрограм – зв’язок один до багатьох між таблицями Guest і IssueGuest.

3.4 Структура програмного забезпечення

До структури програмного забезпечення входять структурні елементи, які формують загалом саму веброзробку, і підсистема визначення категорій аудіопрограм (рис. 3.4). Підсистема виконує визначення категорії, до якої належить аудіопрограма. Дана взаємодія реалізується тоді, коли користувач завантажує аудіофайл випуску і зберігає його на сервері, саме тоді відповідний контролер звертається до підсистеми з запитом при присвоєння категорій. Як тільки підсистема повертає присвоєні категорії випуску, відповідний контролер звертається до модуля контексту даних і зберігає через нього в БД отримані значення як голоси для аудіопрограми. У цей момент відбувається переобчислення голосів і переприсвоювання категорій аудіопрограми. При цьому автоматично визначені категорії тільки додаються до категорій, встановлених вручну.

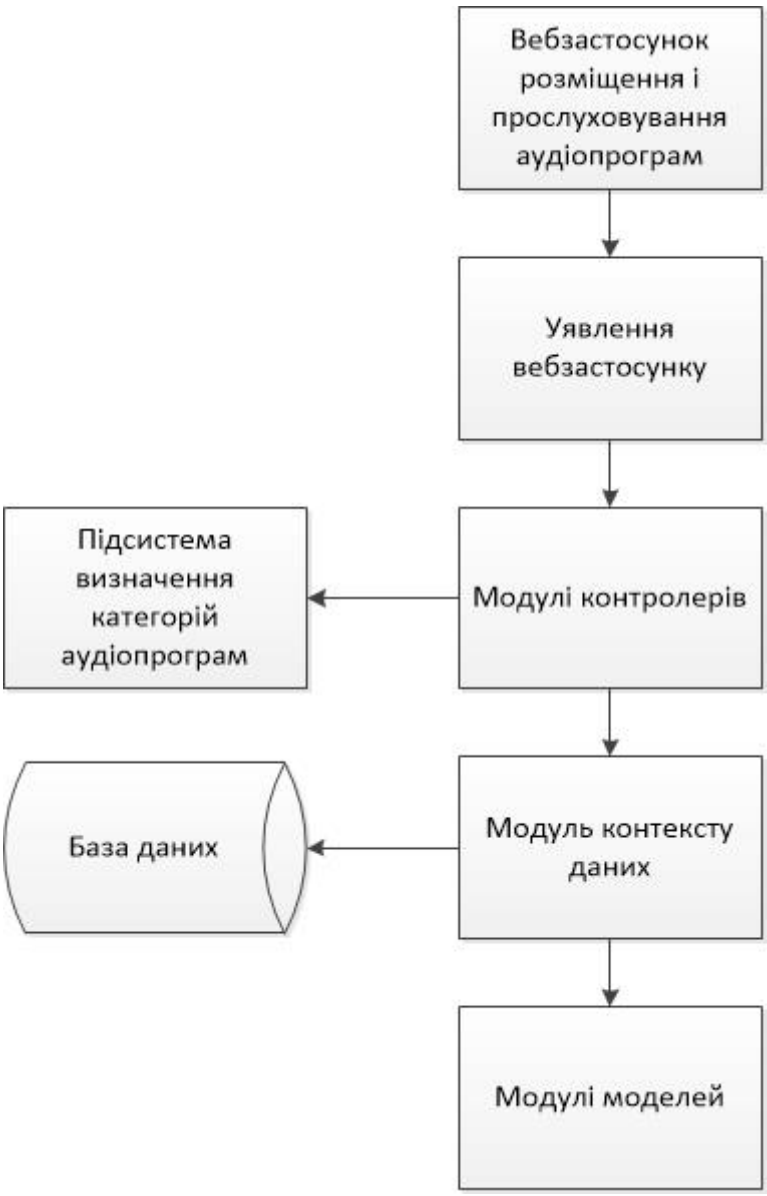


Рисунок 3.4 – Структурна схема програми

3.5 Висновки за розділом 3

Програмну систему має бути розроблено мовою програмування C#, а її структура повинна складатися з підсистеми визначення категорій аудіопрограм і безпосередньо структурних елементів вебзастосунку. Визначено основні сутності предметної області, на основі яких виконано проєктування структури БД.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

4.1 Алгоритм функціонування системи

Робота з програмою розпочинається з роботи зі списком аудіопрограм (рис. 4.1). У процесі виконання цієї роботи користувач отримує можливість обрати один з варіантів з меню.

Перший варіант доступний у тому випадку, якщо користувач авторизований. У такому випадку реалізується робота з власними аудіопрограмами.

Другий варіант реалізується для всіх користувачів, як авторизованих, так і неавторизованих. Він передбачає роботу зі списком категорій.

Третій варіант полягає в роботі зі списком аудіопрограм. Він доступний і авторизованому, і неавторизованому користувачу. Але при цьому на процес даної роботи впливає те, чи є користувач авторизованим, чи неавторизованим. Набір доступних функцій всередині різний в залежності від авторизованості користувача.

Четвертий варіант полягає в роботі зі списком випусків. Звідси також доступні різні функції в залежності від того, авторизований користувач чи ні.

П’ятий варіант може бути використаний тільки авторизованим користувачем. Він надає можливість прослуховувати список уподобаних випусків.

Шостий варіант використовується тільки авторизованими користувачами. Він передбачає роботу з акаунтом і перегляд набору дій, пов’язаних з користувачем.

Сьомий варіант роботи реалізується наступним чином: якщо користувач авторизований, то йому доступний вихід з акаунту, а якщо неавторизований, то надається можливість авторизуватися.

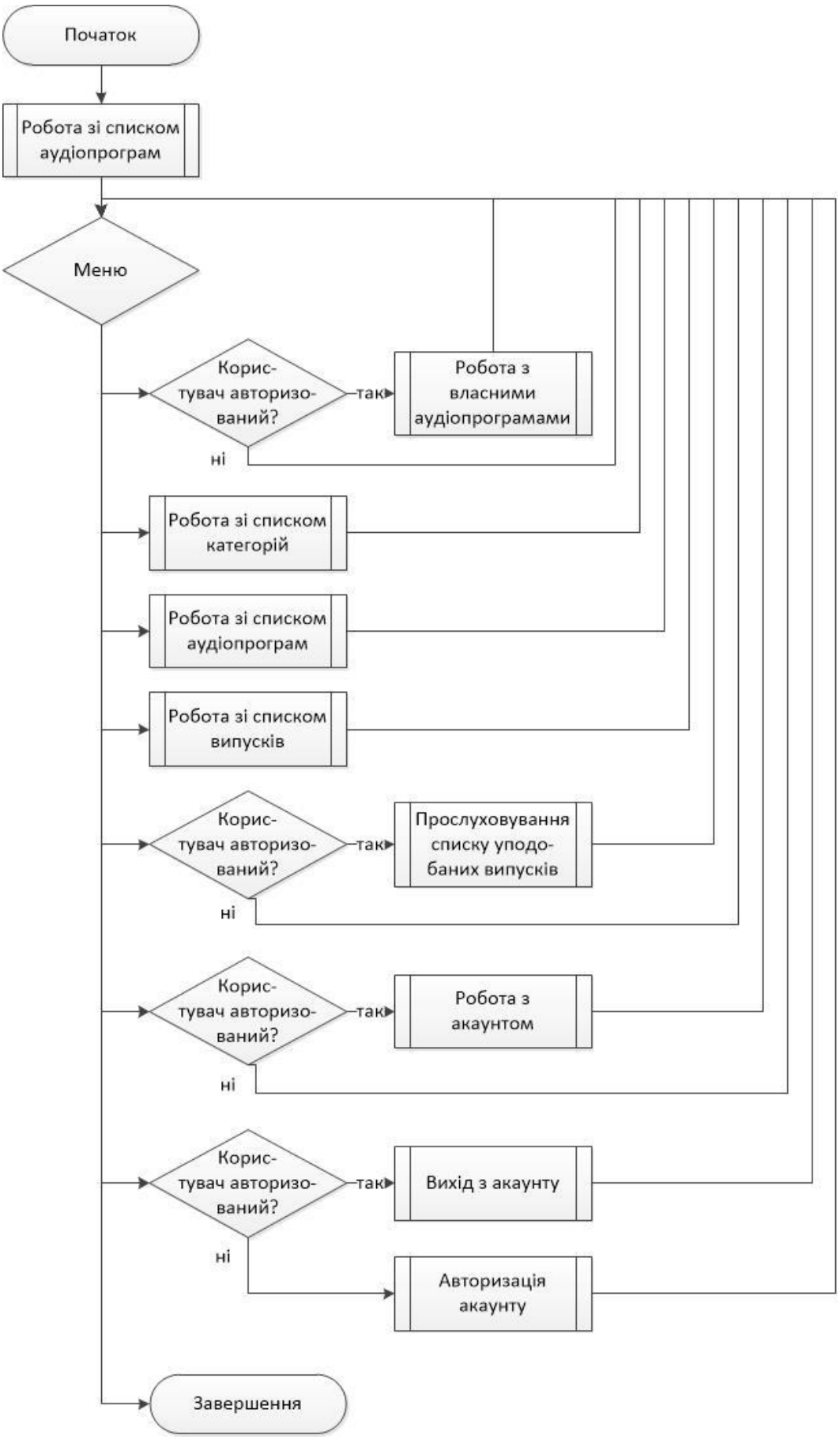


Рисунок 4.1 – Схема функціонування програми

4.2 Опис розроблених класів програми

Створені рішення дозволили реалізувати програмну систему як набір класів, які відповідають визначеній структурі програми.

Клас AudioProgramController реалізує основні засоби роботи з аудіопрограмами, надаючи доступ до них за допомогою наступних методів.

Метод AudioProgramController (APDataBaseContext c) визначає контекстом даних контексту с. За допомогою даного контексту реалізується звернення до БД.

Метод Index () надає базовий інструмент, який повертає уявлення відображення списку аудіопрограм за рейтингом виставлених їм оцінок. Метод реалізується викликом методу Rating.

Метод Rating () надає інструмент відображення списку всіх аудіопрограм, відсортованих за зменшенням середньої оцінки аудіопрограми, створюючи в результаті відповідне уявлення.

Метод ReverseRating () надає інструмент відображення списку всіх аудіопрограм, відсортованих за збільшенням середньої оцінки аудіопрограми, створюючи в результаті відповідне уявлення.

Метод NewAudioPrograms () надає інструмент відображення списку всіх аудіопрограм, відсортованих за часом публікації аудіопрограми, починаючи з останніх публікацій і створюючи в результаті відповідне уявлення.

Метод OldAudioPrograms () надає інструмент відображення списку всіх аудіопрограм, відсортованих за часом публікації аудіопрограми, починаючи з найстаріших публікацій і створюючи в результаті відповідне уявлення.

Метод MaxFollowers () надає інструмент відображення списку всіх аудіопрограм, відсортованих за зменшенням кількості підписників на аудіопрограми (починаючи з максимальної кількості), створюючи в результаті відповідне уявлення.

Метод MinFollowers () надає інструмент відображення списку всіх аудіопрограм, відсортованих за збільшенням кількості підписників на аудіопрограми (починаючи з мінімальної кількості), створюючи в результаті відповідне уявлення.

Метод AccountAudioPrograms (int account) надає інструмент відображення списку всіх аудіопрограм користувача з акаунтом з номером account, відсортованих, починаючи від найновіших створених аудіопрограм, створюючи в результаті відповідне уявлення.

Метод PlaceAudioPrograms (int place) надає інструмент відображення списку всіх аудіопрограм, які створюються в місті з номером place, відсортованих, починаючи від найновіших створених аудіопрограм, створюючи в результаті відповідне уявлення.

Метод Details (int number) повертає повну інформацію про аудіопрограму з номером number, зокрема її номер, назву, опис, обкладинку, дату публікації, кількість підписників, автора, місто створення, рядкове представлення дати публікації, оцінку та її текстове представлення, повну інформацію про акаунт автора та місто створення, а також повний список категорій, до яких віднесено аудіопрограму, повний список оцінок, виставлених аудіопрограмі, повний список підписників на аудіопрограму, повний список сезонів аудіопрограми:

```
var program = _con.AudioPrograms.Include(a => a.AudioProgramId).Include(a => a.Name).Include(a => a.Info).Include(a => a.Cover).Include(a => a.PubDate).Include(a => a.Followers).Include(a => a.AccountId).Include(a => a.PlaceId).Include(a => a.PubDateStr).Include(a => a.ScoreStr).Include(a => a.Score).Include(a => a.Account).Include(a => a.Place).Include(a => a.AudioProgramCategories).Include(a => a.AudioProgramScores).Include(a => a.Follows).Include(a => a.Seasons).FirstOrDefault(a => a.AudioProgramId == number);
```

Метод SearchAudioPrograms (string searchp) надає інструмент відображення списку аудіопрограм за результатами пошуку аудіопрограм з назвою, до якої входить searchp.

Метод ScoreAudioProgram (int number, int sc) забезпечує підтримку дій користувача з оцінювання аудіопрограми з номером number на оцінку sc, зберігаючи дану оцінку в БД. Для реалізації даного методу спочатку виконується перевірка, чи було раніше внесено оцінку даним авторизованим користувачем аудіопрограми з номером number. Якщо таку оцінку вже було зроблено раніше, то змінюється сама оцінка і дата публікації оцінки. Якщо ж така оцінка раніше не виставлялась, то вона вперше вноситься в БД. Метод без авторизації користувача працювати не може, для неавторизованого користувача доступ заборонений.

Метод CalculateScore (int number) реалізує обчислення середньої оцінки, виставленої аудіопрограмі number, вносячи отриманий розрахунок до БД. Метод без авторизації користувача працювати не може, для неавторизованого користувача доступ заборонений.

Метод CalculateFollowers (int number) реалізує обчислення кількості підписників на аудіопрограму з номером number, вносячи отриманий розрахунок до БД. Для кожного запису про підписку виконується перевірка, чи підписка є активною, чи ні. Метод без авторизації користувача працювати не може, для неавторизованого користувача доступ заборонений.

Метод CalculateAFollowers () виконує оновлення кількості підписників на всі аудіопрограми, забезпечуючи для цього виклик методу CalculateFollowers для всіх наявних аудіопрограм.

Метод CalculateScores () виконує оновлення оцінки аудіопрограми, забезпечуючи для цього виклик методу CalculateScore для всіх наявних аудіопрограм.

Останні два методи запускаються за розкладом для того, щоб періодично виконувати переобчислення.

Метод FollowAudioProgram(int number) забезпечує підтримку дій користувача з підписки на аудіопрограму з номером number, зберігаючи дану підписку в БД. Для реалізації даного методу спочатку виконується перевірка, чи раніше даний авторизований користувач підписався на аудіопрограму з номером number. Якщо така дія була, то стан підписки змінюється на неактивну. Якщо підписка була, але вона неактивна, то змінюється на активну. Якщо ж раніше підписка не оформлювалась, то вона вперше вноситься в БД. Метод без авторизації користувача працювати не може, для неавторизованого користувача доступ заборонений.

Метод Create () створює уявлення створення нової аудіопрограми, формуючи список міст, з якого потрібно буде обрати місто, де створюється аудіопрограма. Метод без авторизації користувача працювати не може, для неавторизованого користувача доступ заборонений.

Метод Create([Bind("Name, Info, Cover, PlaceId")] AudioProgram program) забезпечує створення нової аудіопрограми шляхом внесення всієї інформації про аудіопрограму в БД. Датою публікації визначається поточний день, а автором – авторизований у даний момент користувач. Метод без авторизації користувача працювати не може, для неавторизованого користувача доступ заборонений.

Метод Edit (int? number) забезпечує створення уявлення редагування аудіопрограми з номером number, формуючи список міст, з якого можна буде обрати місто, де створюється аудіопрограма. Метод без авторизації користувача працювати не може, для неавторизованого користувача доступ заборонений.

Метод Edit(int number, [Bind("AudioProgramId, Name, Info, Cover, PlaceId")] AudioProgram program) забезпечує редагування існуючої аудіопрограми шляхом змінювання інформації про аудіопрограму в БД. Метод без авторизації користувача працювати не може, для неавторизованого користувача доступ заборонений.

Клас IssueController реалізує основні засоби роботи з випусками аудіопрограмам, надаючи доступ до них за допомогою наступних методів.

Метод Details (int number) повертає повну інформацію про випуск аудіопрограми з номером number у випадку, якщо випуск знаходиться в стані опублікований, а поточна дата пізніша за дату, з якої буде доступним для прослуховування випуск аудіопрограми.

Метод Rating () надає інструмент відображення списку всіх випусків всіх аудіопрограм, відсортованих за зменшенням середньої оцінки, створюючи в результаті відповідне уявлення.

Метод ReverseRating () надає інструмент відображення списку всіх випусків всіх аудіопрограм, відсортованих за збільшенням середньої оцінки, створюючи в результаті відповідне уявлення.

Метод Rating (int number) надає інструмент відображення списку всіх випусків аудіопрограми з номером number, відсортованих за зменшенням середньої оцінки, створюючи в результаті відповідне уявлення.

Метод ReverseRating (int number) надає інструмент відображення списку всіх випусків аудіопрограми з номером number, відсортованих за збільшенням середньої оцінки, створюючи в результаті відповідне уявлення.

Метод NewIssues (int number) надає інструмент відображення списку всіх випусків аудіопрограми з номером number, відсортованих за часом публікації, починаючи з останніх випусків і створюючи в результаті відповідне уявлення.

Метод OldIssues (int number) надає інструмент відображення списку всіх випусків аудіопрограми з номером number, відсортованих за часом публікації, починаючи з найстаріших випусків і створюючи в результаті відповідне уявлення.

Метод UserFavorIssue (int number) надає інструмент відображення списку всіх випусків, уподобаних користувачем з номером акаунту number, відсортованих за часом уподобання, починаючи з останньої і створюючи в результаті відповідне уявлення.

Метод GuestIssues (int number) надає інструмент відображення списку всіх випусків, у яких приймав участь гість з номером number, відсортованих за часом публікації, починаючи з останньої і створюючи в результаті відповідне уявлення.

Метод ScoreIssue (int number, int sc) забезпечує підтримку дій користувача з оцінювання випуску аудіопрограми з номером number на оцінку sc, зберігаючи дану оцінку в БД. Для реалізації даного методу спочатку виконується перевірка, чи було раніше внесено оцінку даним авторизованим користувачем випуску аудіопрограми з номером number. Якщо таку оцінку вже було зроблено раніше, то змінюється сама оцінка і дата публікації оцінки. Якщо ж така оцінка раніше не виставлялась, то вона вперше вноситься в БД. Метод без авторизації користувача працювати не може, для неавторизованого користувача доступ заборонений.

Метод CalculateScore (int number) реалізує обчислення середньої оцінки, виставленої випуску аудіопрограми з номером number, вносячи отриманий розрахунок до БД. Метод без авторизації користувача працювати не може, для неавторизованого користувача доступ заборонений.

Метод CalculateFavor (int number) реалізує обчислення кількості уподобань випуску аудіопрограми з номером number, вносячи отримане значення в БД. Для кожного запису про уподобання виконується перевірка, чи уподобання є активним, чи ні. Метод без авторизації користувача працювати не може, для неавторизованого користувача доступ заборонений.

Метод CalculateFavors () виконує оновлення кількості уподобань всіх випусків аудіопрограми, забезпечуючи для цього виклик методу CalculateFavor для всіх наявних випусків аудіопрограм.

Метод CalculateScores () виконує оновлення оцінки випуску аудіопрограми, забезпечуючи для цього виклик методу CalculateScore для всіх наявних випусків аудіопрограм.

Останні два методи запускаються за розкладом для того, щоб періодично виконувати переобчислення.

Метод Create () створює уявлення внесення нового випуску аудіопрограми, формуючи список доступних станів випуску і список доступних сезонів аудіопрограми. Метод без авторизації користувача працювати не може, для неавторизованого користувача доступ заборонений.

Метод Create([Bind("SeasonId, Name, Start, Info, Cover, Status, Audiofile")] Issue aissue) забезпечує створення нового випуску аудіопрограми шляхом внесення всієї інформації про випуск у БД. Датою публікації визначається поточний день. Метод без авторизації користувача працювати не може, для неавторизованого користувача доступ заборонений.

Метод Edit (int? number) забезпечує створення уявлення редагування випуску аудіопрограми з номером number, формуючи список доступних станів випуску і список доступних сезонів аудіопрограми. Метод без авторизації користувача працювати не може, для неавторизованого користувача доступ заборонений.

Метод Edit(int number, [Bind("IssueId, SeasonId, Name, Start, Info, Cover, Status, Audiofile")] AudioProgram program) забезпечує редагування існуючого випуску аудіопрограми шляхом змінювання інформації про випуск у БД. Метод без авторизації користувача працювати не може, для неавторизованого користувача доступ заборонений. Доступ до кількості уподобань, оцінки випуску з методу редагування не здійснюється, ці значення обчислюються тільки програмно і не можуть змінюватися вручну.

Уявлення програмної системи побудовано на основі застосування шаблону [20].

Молелі визначають структуру даних, що використовуються для підтримки роботи за відповідними описаними вище і аналогічними контролерами.

Модель класу випуску аудіопрограми Issue організована на основі перелічення IssueStatusEnum, яке визначає набір станів випуску, до яких входять:

– стан Published («Опубліковано»);

– стан Draft («Чорнетка»);

– стан Waiting («В очікуванні»).

Клас Issue містить атрибути:

– int IssueId – номер випуску;

– string Name – назва випуску;

– DateTime Start – дата і час, з якого доступний даний випуск для прослуховування користувачам;

– DateTime Publication – дата і час публікації випуску;

– float Score – середня оцінка випуску;

– int SeasonId – сезон аудіопрограми, до якого належить даний випуск;

– string Info – опис випуску;

– string Cover – обкладинка випуску;

– IssueStatusEnum Status – стан випуску;

– int Favorites – кількість уподобань випуску;

– string Audiofile – шлях до аудіофайлу з випуском програми;

– virtual Season Season – об’єкт класу Season, який містить повні дані про сезон, до якого належить випуск;

– virtual ICollection<Favor> Favors – список уподобань випуску аудіопрограми;

– virtual ICollection<IssueGuest> IssueGuests – список гостей, що брали участь у випуску;

– virtual ICollection<IssueScore> IssueScores – список оцінок, виставлених випуску;

– virtual ICollection<Issue> Connections – список пов’язаних випусків з випуском, представленим у об’єкті;

– string ScoreStr – текстове представлення оцінки випуску, округлене до 1 знаку після коми;

– string StartStr – текстове представлення дати і часу, з якого доступний даний випуск для прослуховування користувачам;

– string PublicationStr – текстове представлення дати і часу публікації випуску.

4.3 Висновки за розділом 4

У четвертому розділі досягнуто наступних результатів:

– представлено порядок роботи програми, за результатами чого створено схему функціонування програми;

– описано принципи створення основних засобів програмної системи.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Призначення програми

Програма призначена для розміщення аудіопрограм на сервері з можливістю їх розділення на сезони та випуски. Для цього надається набір інструментів, які полегшують процес пошуку нових аудіопрограм. До таких засобів зокрема належить програмне визначення категорій аудіопрограми. Окрім того програма може використовуватися користувачами для надання доступу до їх прослуховування.

5.2 Умови виконання програми

До умов виконання програми, які апаратно мають забезпечуватися на сервері, належать:

– процесор, який має 4 ядра з тактовою частотою в 2,66 ГГц;

– оперативна пам’ять обсягом у 8 Гб;

– жорсткий диск з загальним обсягом не менше 8 Гб, призначений для серверного використання.

Окрім того має бути встановлено пакет розробника платформи .NET та систему керування БД Microsoft SQL Server.

5.3 Виконання програми

Для виконання програми потрібно звернутися до вебсервера, зазначивши його ім’я в браузері. Взаємодія з програмою реалізується таким чином. За потреби дані також можуть вноситися через БД. Така потреба може виникнути, наприклад, при необхідності додати інші категорії аудіопрограм.

Доступ до аудіопрограм реалізується всіма користувачами, тому програма не вимагає авторизації, але надає таку можливість. У випадку авторизації користувач може також завантажувати власні аудіопрограми.

Загальний набір пунктів меню для всіх користувачів реалізується наступним набором.

Пункт меню «Категорії» виводить вебсторінку відображення повного переліку всіх категорій аудіопрограм. Якщо обрати необхідну назву, то відкриється вебсторінка з аудіопрограмами, для яких було визначено обрану категорію (рис. 5.1).

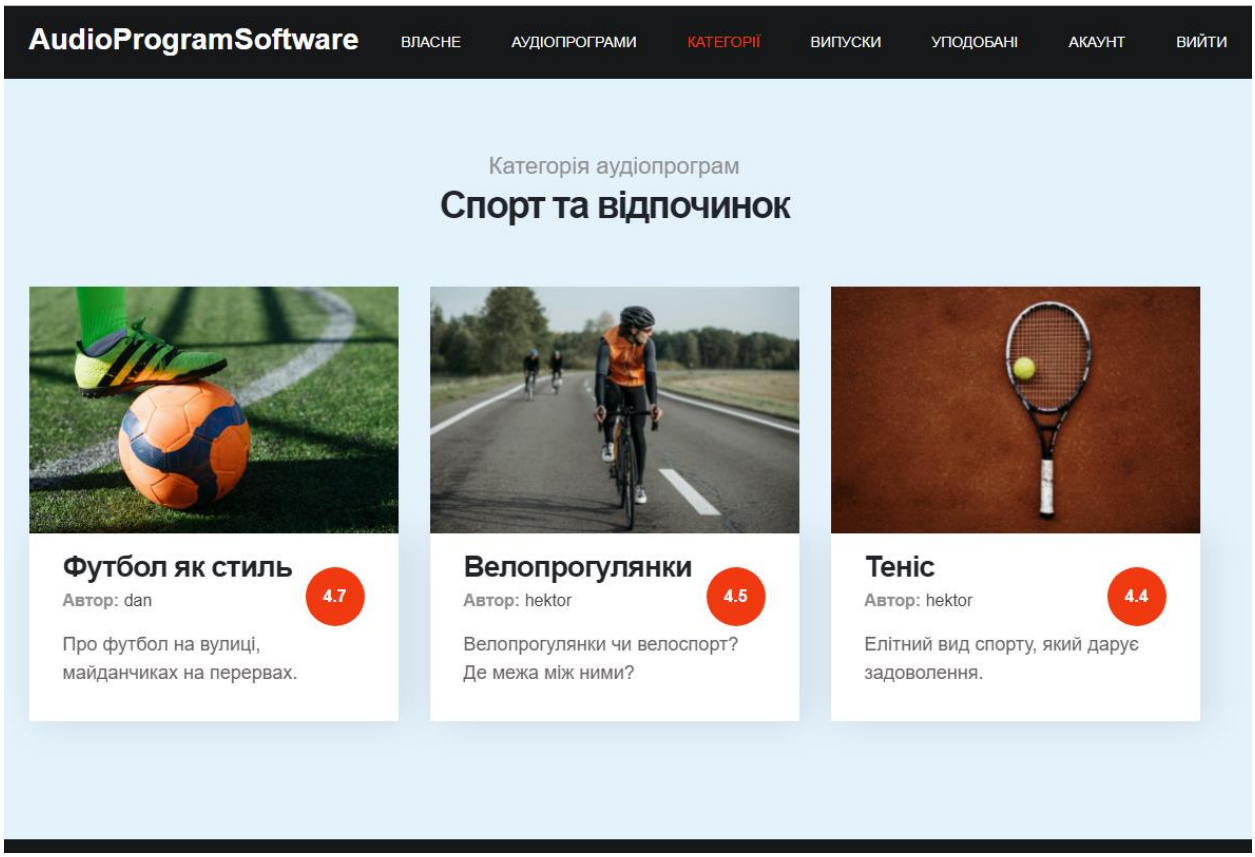


Рисунок 5.1 – Відображення аудіопрограм за категорією «Спорт та відпочинок»

У списку, який при цьому відтворюється, виводиться тільки загальна інформація про аудіопрограми. До такої загальної інформації віднесено

обкладинку, її назву, автора, короткий опис та середню оцінку. Оцінка відображається в червоному колі.

Для того щоб відтворити всі аудіопрограми, які були створені тим же самим користувачем, потрібно натиснути на логін користувача.

Якщо ж натиснути на назву аудіопрограми, то відкриється вебсторінка з цією аудіопрограмою.

Якщо обрати один з випусків аудіопрограми, то відкриється окрема вебсторінка прослуховування випуску (рис. 5.2).

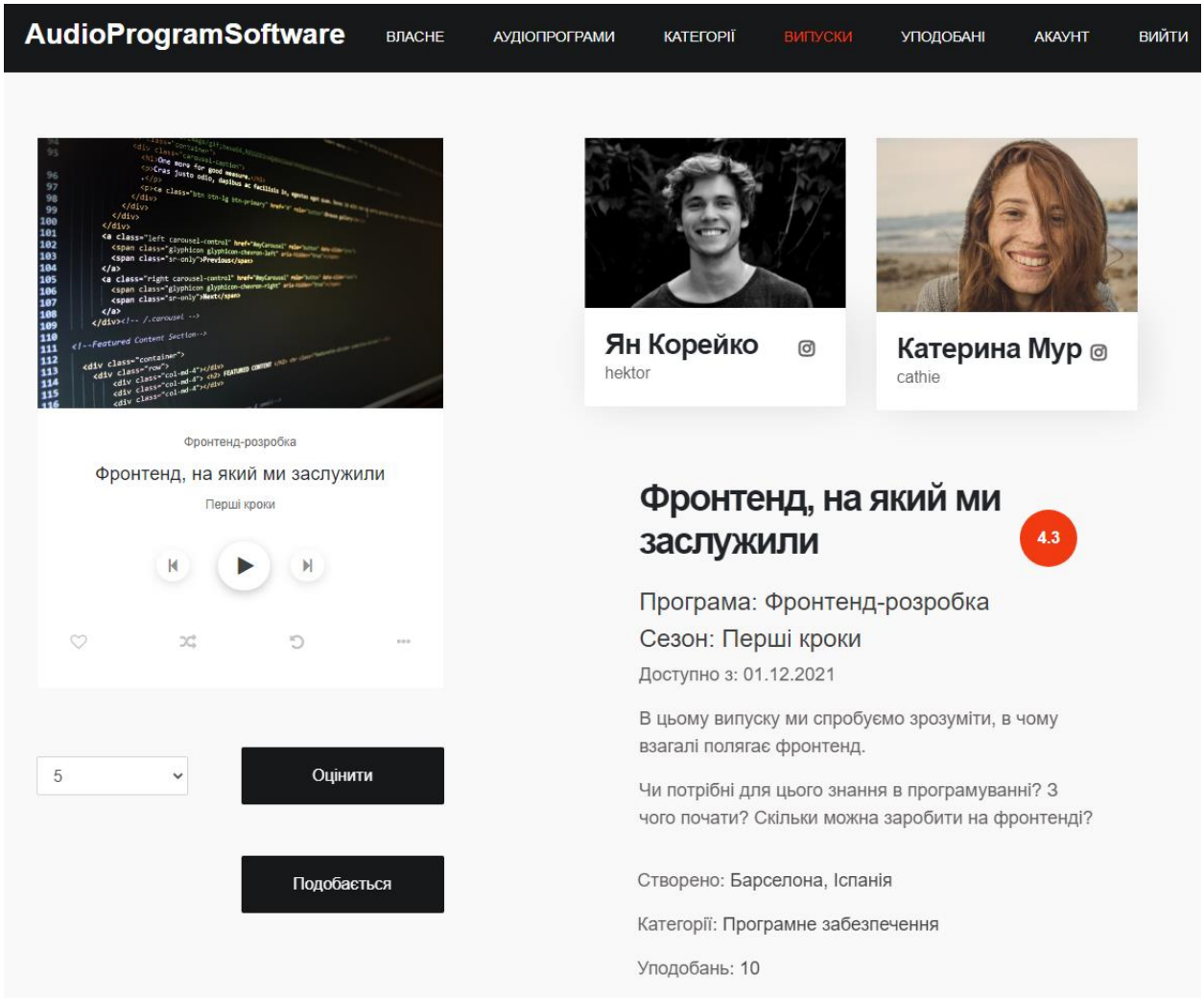


Рисунок 5.2 – Прослуховування випуску аудіопрограми «Фронтенд-розробка»

За випуском відображається програвач, який дозволяє прослухати сам випуск. Це стандартний програвач з можливістю запуску і зупинення

випуску. Також можна перемикатися між випусками. Випуски беруться з поточного сезону аудіопрограми. У програвачі зазначається назва випуску (посередині), назва аудіопрограми (вище) та назва сезону (нижче).

Нижче програвача розташовано рядок, в якому можна задати оцінку з випадаючого списку, а далі обрану оцінку можна підтвердити натисканням кнопки «Оцінити».

Для того щоб уподобати випуск, потрібно натиснути на кнопку «Подобається». Після того, як випуск уподобано, його можна відтворити у власному списку уподобаних випусків. Якщо випуск вже було уподобано, то його буде вилучено зі списку.

За випуском також зазначається перелік гостей, які брали участь у випуску разом з фотографією. За кожним гостем фотографія відображається в тому випадку, якщо гість є користувачем програми. Якщо натиснути на логін користувача, то відкриється вебсторінка зі списком аудіопрограм, створених цим користувачем. Якщо натиснути на іконку правіше, то перехід буде виконано на зовнішній вебсайт.

Нижче гостей виводиться назва випуску аудіопрограми, назва аудіопрограми, назва сезону та дата, з якої доступний для відтворення випуск. Також зазначається середня оцінка випуску. При натисканні на назву сезону, програми можна перейти до відповідної вебсторінки.

Окрім того відображається місце, де створюється аудіопрограма (для випуску підтягується з аудіопрограми). При натисканні на назву місця відкривається вебсторінка відтворення списку аудіопрограм, які були створені в цьому місці.

Також відображається список категорій, до яких належить дана аудіопрограма (для випуску окремо категорії не відображаються). При натисканні на назву категорії можна перейти до аудіопрограм, які належать цій категорії.

Також відображається кількість уподобань даного випуску. При натисканні на кількість відкривається вебсторінка з логінами і фотографіями користувачів, які уподобали випуск.

На сторінці нижче також відображається перелік випусків аудіопрограм, пов’язаних з даним.

Пункт меню «Аудіопрограми» дозволяє перейти до роботи зі списками аудіопрограм інших користувачів (не власними). Список відтворюється аналогічно способу, представленому для категорій. При цьому надаються кнопки керування даним відтворенням.

Кнопка «Рейтинг» дозволяє впорядкувати аудіопрограми за рейтингом. При цьому парна кнопка «Сортувати» дозволяє впорядкувати аудіопрограми за зменшенням рейтингу при одному натисканні і відповідно за збільшенням рейтингу при наступному натисканні.

Кнопка «Нові» дозволяє впорядкувати аудіопрограми за часом створення. При цьому парна кнопка «Сортувати» дозволяє впорядкувати аудіопрограми, починаючи з найновіших, при одному натисканні і відповідно, починаючи від найстаріших, при наступному натисканні.

Кнопка «Підписники» дозволяє впорядкувати аудіопрограми за кількістю користувачів, які підписалися на аудіопрограму. При цьому парна кнопка «Сортувати» дозволяє впорядкувати аудіопрограми за зменшенням кількості підписників при одному натисканні і відповідно за збільшенням кількості підписників при наступному натисканні.

Пункт меню «Власне» (рис. 5.3) авторизованого користувача відтворює список всіх аудіопрограм, створених користувачем. Для створення нової аудіопрограми потрібно натиснути на кнопку «Створити нову аудіопрограму». За своєю суттю дана вебсторінка є центром керування власними аудіопрограмами. Саме звідси можна переглядати, які програми вже було створено, змінювати їх склад, редагувати, а також визначати параметри публікації. Перехід на відповідні вебсторінки відбувається саме через дану.

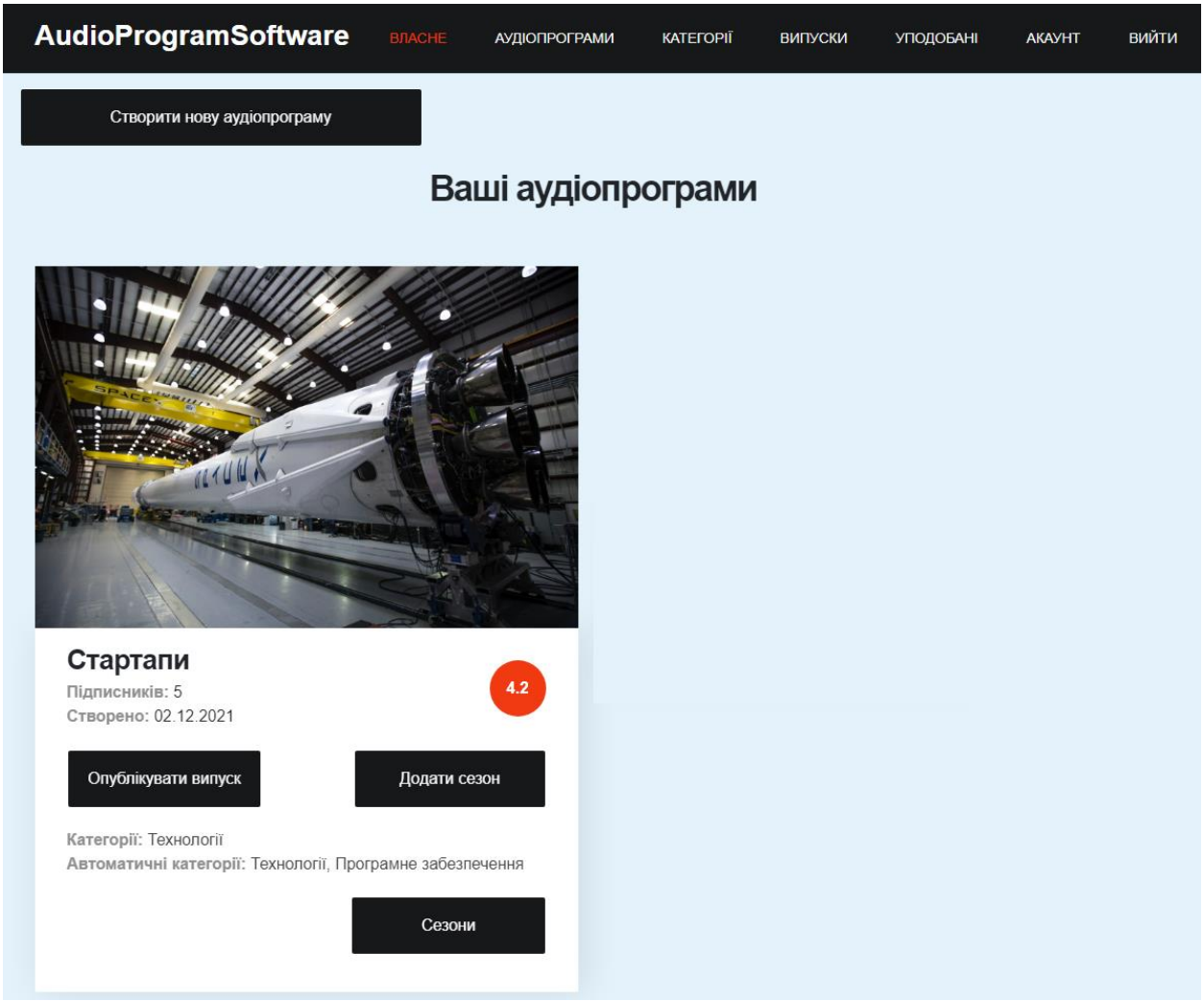


Рисунок 5.3 – Відображення власних аудіопрограм

Список створених аудіопрограм відображається по 2 на рядку. У кожній позиції виводиться основна інформація про аудіопрограму. Звідси автор може переглянути зокрема середню оцінку, виставлену аудіопрограмі.

На даній плитці відображається назва аудіопрограми, на яку необхідно натиснути, щоб перейти до вебсторінки редагування даних аудіопрограми. Також звідси доступна інформація про кількість підписників (натискання на число призводить до відкриття вебсторінки з підписниками) та дату створення.

Якщо потрібно опублікувати новий випуск, то необхідно натиснути на кнопку «Опублікувати випуск». На сторінці, що відкриється, завантажується файл випуску.

Для створення нового сезону аудіопрограми потрібно натиснути на кнопку «Додати сезон».

Нижче відображається список категорій аудіопрограми. Спочатку відображаються категорії, які були встановлені самим автором. Нижче відображаються категорії, які були встановлені програмно.

Для того щоб відкрити список всіх сезонів аудіопрограми, потрібно натиснути на кнопку «Сезони».

Пункт меню «Випуски» дозволяє перейти до роботи зі списками випусків аудіопрограм інших користувачів. При цьому надаються кнопки керування списком.

Кнопка «Рейтинг» дозволяє впорядкувати всі випуски всіх аудіопрограм за рейтингом. При цьому парна кнопка «Сортувати» дозволяє впорядкувати випуски за зменшенням рейтингу при одному натисканні і відповідно за збільшенням рейтингу при наступному натисканні.

Кнопка «Нові» дозволяє впорядкувати випуски за часом створення. При цьому парна кнопка «Сортувати» дозволяє впорядкувати випуски, починаючи з найновіших, при одному натисканні і відповідно, починаючи від найстаріших, при наступному натисканні.

Кнопка «Уподобання» дозволяє впорядкувати випуски за кількістю користувачів, які уподобали даний випуск. При цьому парна кнопка «Сортувати» дозволяє впорядкувати випуски за зменшенням кількості уподобань при одному натисканні і відповідно за збільшенням кількості уподобань при наступному натисканні.

Пункт меню «Уподобані» відтворює список усіх випусків різних аудіопрограм, які були уподобані користувачем. Список формується, починаючи з останнього уподобаного. Одразу на сторінці відтворюється програвач, який дозволяє прослуховувати випуски з даного списку. Перемикання між випусками вперед і назад відбувається з даного списку.

Пункт меню «Акаунт» окрім можливості визначення даних про користувача, включаючи зокрема фотографію, пароль, надає доступ до

виконаних дій. Звідси можна натисканням кнопки «Підписки» перейти до вебсторінки зі списком всіх аудіопрограм, на які підписався користувач. За допомогою кнопки «Оцінки аудіопрограм» перейти до списку всіх виставлених даним користувачем оцінок різним аудіопрограмам. За допомогою кнопки «Оцінки випусків» перейти до списку всіх виставлених даним користувачем оцінок різним випускам.

Пункт меню «Вийти» дозволяє залишити свій акаунт і за потреби продовжити роботу без авторизації. У такому випадку користувач не зможе оцінювати аудіопрограми, випуски, підписуватися на аудіопрограми, уподобати випуски, створювати власні аудіопрограми. Всі ці можливості на відповідних вебсторінках відтворюватися не будуть.

5.4 Експериментальне дослідження категоризації аудіопрограм

Для експериментального дослідження категоризації аудіопрограм необхідна була вибірка даних, яка визначає співвідношення між аудіопрограмами та їх жанрами. Для цього було використано рішення і вибірку [21]. Кожен екземпляр сформованої вибірки даних має наступний набір ознак:

– назва: назва аудіопрограми;

– зображення: посилання на зображення аудіопрограми;

– ідентифікатори жанрів: список ідентифікаторів жанрів, до яких віднесено аудіопрограму;

– кількість випусків: кількість випущених на даний момент даних аудіопрограм;

– тривалість випусків: список тривалості кожного випуску аудіопрограми в хвилинах;

– uniform resource locator (URL)-адреса iTunes: URL-посилання на аудіопрограму в iTunes;

– URL-адреса аудіопрограми: URL-посилання вебсайту аудіопрограми;

– опис: загальний опис аудіопрограми, написаний на її сторінці iTunes [21].

Окрім того були завантажені безпосередньо випуски аудіопрограм. Після цього випуски розділені на фрагменти і створені спектрограми для кожного такого фрагменту. З кожним фрагментом було співвіднесено відповідний жанр. На основі сформованої таким чином вибірки даних було виконано класифікацію фрагментів.

Класифікацію було виконано шляхом застосування запропонованого в роботі методу на основі моделі AlexNet та на основі звичайної ЗНМ (табл. 5.1). В якості звичайної ЗНМ розглянуто різні варіанти, що складаються максимально з 3 згорткових шарів. Найкращий отриманий результат точності класифікації представлено у відповідному рядку таблиці.

Таблиця 5.1 – Результати експериментального дослідження

Модель класифікації	Середня точність класифікації
Звичайна ЗНМ	78,4
AlexNet	83,7

ЗНМ на основі архітектури AlexNet дозволила отримати точність класифікації у 83,7 %. Оскільки результат AlexNet на 6,68 % краще за звичайну ЗНМ, то саме AlexNet краще використовувати в основі запропонованого методу.

5.5 Висновки за розділом 5

У п’ятому розділі детально описано процес роботи користувачів з програмою, визначено, яким чином виконати надані програмою функції через інтерфейс програми.

Експериментальне дослідження запропонованого методу визначення категорій аудіопрограм виконано на основі вибірки, яка характеризує фрагменти аудіопрограм та їх жанри. Найкращий результат дозволив отримати запропонований метод на основі моделі ЗНМ архітектури AlexNet.

ВИСНОВКИ

Мета роботи полягала в розробці програмного забезпечення розміщення і прослуховування аудіопрограм з наданням можливості категоризації без участі авторів для підвищення зручності пошуку нових аудіопрограм. Для досягнення цієї мети було розв’язано набір завдань роботи.

Виконано порівняння програмних аналогів, що включають зокрема Podbean, Transistor, з програмою, що розробляється. Визначено в якості її основних відмінностей можливість автоматичного визначення категорій аудіопрограм та надання безкоштовного вільного доступу до розміщення та прослуховування аудіопрограм.

Проаналізовано нейронні мережі, придатні для аналізу зображень, виділено особливості архітектури ЗНМ AlexNet. Запропоновано перетворювати звукові дані до графічної форми. Створено і описано метод визначення категорій аудіопрограм.

Представлено порядок роботи програми. Створено схему функціонування програми. Розроблено програму мовою C#. Виділено основні сутності предметної області, створено структуру БД і структуру програмної системи. Описано розроблені класи для реалізації програмної системи.

Розроблений вебзастосунок дозволяє користувачам прослуховувати аудіопрограми, створені іншими користувачами, розділяти їх на окремі сезони і випуски, виконувати їх пошук, переглядати найбільш рейтингові аудіопрограми або аудіопрограми за категоріями, оцінювати, співвідносити аудіопрограми з їх категоріями, підписуватися на прослуховування.

Таким чином, усі поставлені завдання на дипломну кваліфікаційну роботу магістра виконано.

Наукова новизна роботи полягає в методі визначення категорій аудіопрограм, який відзначається застосуванням згорткових нейронних

мереж і встановленням категорій аудіопрограми на основі узагальнення категорій, співставлених з окремими випусками аудіопрограми, що дозволяє розподіляти і об’єднувати аудіопрограми без участі їх користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Europe’s podcast evolution: “An exciting prospect” [Electronic resource]. – Access mode : https://whatsnewinpublishing.com/europes-podcast-evolution-an-exciting-prospect/.

2. Подкасти – КУНШТ [Електрон. ресурс]. – Режим доступу : https://kunsht.com.ua/podkasty/.

3. Free Podcast Hosting – Starting a Podcast in 5 Minutes | Podbean [Electronic resource]. – Access mode : https://www.podbean.com/.

4. The 5 Best Free Podcast Hosting Services in 2021 [Electronic resource]. – Access mode : https://discoverpods.com/best-free-podcast-hosting/.

5. Podbean Reviews [Electronic resource]. – Access mode : https://digital.com/best-web-hosting/podbean/.

6. Transistor – podcast hosting for creatives, brands, professionals [Electronic resource]. – Access mode : https://transistor.fm/.

7. The 12 Best Podcast Hosting Platforms in 2021 [Electronic resource]. – Access mode : https://www.omnicoreagency.com/best-podcast-hosting-platform/.

8. Unlimited podcast hosting, analytics & marketing. Captivate.fm [Electronic resource]. – Access mode : https://www.captivate.fm/.

9. Best Podcast Hosting Platforms In 2021 [Electronic resource]. – Access mode : https://podcasthosting.org/.

10. Kamali, K. Deep Learning (Part 3) – Convolutional neural networks (CNN) [Electronic resource] / K. Kamali. – Access mode : https://training.galaxyproject.org/archive/2021-08-01/topics/statistics/tutorials/CNN/tutorial.html.

11. Chane, M. Classification of medical images [Electronic resource] : understanding the convolutional neural network (CNN) / M. Chane. – Access mode : https://www.imaios.com/ru/Kompaniya/blog/Classification-of-medical-images-understanding-the-convolutional-neural-network-CNN.

12. Mandal, M. Introduction to Convolutional Neural Networks (CNN)

[Electronic resource] / M. Mandal. – Access mode : https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/.

13. Wei, J. AlexNet [Electronic resource] : The Architecture that Challenged CNNs / J. Wei. – Access mode : https://towardsdatascience.com/alexnet-the-architecture-that-challenged-cnns-e406d5297951.

14. AlexNet [Electronic resource] : The First CNN to win Image Net. – Access mode : https://www.mygreatlearning.com/blog/alexnet-the-first-cnn-to-win-image-net/.

15. Nayak, S. Understanding AlexNet [Electronic resource] / S. Nayak. – Access mode : https://learnopencv.com/understanding-alexnet/.

16. Nepal, P. AlexNet Architecture Explained [Electronic resource] / P. Nepal. – Access mode : https://medium.com/analytics-vidhya/alexnet-architecture-explained-5d19e3dca2bb.

17. Saxena, S. Introduction to The Architecture of Alexnet Electronic resource] / S. Saxena. – Access mode : https://www.analyticsvidhya.com/blog/2021/03/introduction-to-the-architecture-of-alexnet/#h2_2.

18. ASP.NET Core. Полное руководство [Электрон. ресурс]. – Режим доступа : https://metanit.com/sharp/aspnet5/.

19. ConvNetSharp [Electronic resource] : Deep Learning in C#. – Access mode : https://github.com/cbovar/ConvNetSharp.

20. Gymso Fitness Template – Free HTML Template [Electronic resource]. – Access mode : https://www.tooplate.com/view/2119-gymso-fitness.

21. Podcasts Data [Electronic resource]. – Access mode : https://github.com/odenizgiz/Podcasts-Data.

ДОДАТОК А

Технічне завдання

Вступ

Будь-який вебзастосунок, в який користувачі самі можуть завантажувати дані, потребують засобів упорядкування цих даних. Одним із таких засобів є створення категорій, до яких ці дані належать. Специфіка аудіопрограм в тому, що вони представлені аудіофайлами. Для того щоб визначити категорію аудіопрограми, потрібно проаналізувати аудіофайл. Розробка вебзастосунку з підтримкою такої можливості та створення відповідного методу вирішення цієї проблеми є важливим завданням роботи.

А.1 Підстави для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу магістра, визначене за темою «Дослідження та програмна реалізація методів визначення категорій аудіопрограм», яку затверджено наказом № 435 від 5 листопада 2021 р. за Національним університетом «Запорізька політехніка».

А.2 Призначення розробки

Програма призначена для розміщення аудіопропрограм з наданням доступу до їх прослуховування. Для цього надається набір інструментів, які полегшують процес пошуку нових аудіопрограм.

А.3 Основні вимоги до програми

А.3.1 Вимоги до функціональних характеристик

Програма має надавати можливість використовувати наступні функції:

– відображення списку категорій аудіопрограм;

– відображення аудіопрограм за категорією;

– пошук аудіопрограм;

– створення аудіопрограми;

– редагування аудіопрограми;

– автоматичне визначення категорії аудіопрограми;

– оцінювання аудіопрограми;

– оцінювання випуску аудіопрограми;

– відображення списку підписників на аудіопрограму;

– відображення списку аудіопрограм, на які підписався користувач;

– відображення списку уподобаних випусків користувачем;

– відображення списку власних аудіопрограм;

– відображення списку аудіопрограм користувача;

– створення рейтингу аудіопрограм;

– створення рейтингу випусків;

– створення сезонів аудіопрограми;

– підписування на аудіопрограму;

– уподобання випуску аудіопрограми;

– відображення списку з’єднаних випусків з даним;

– сортування списку аудіопрограм;

– відображення аудіопрограм за містом створення;

– відображення випусків, у яких брав участь гість;

– відображення списку сезонів аудіопрограми;

– відображення списку випусків за сезоном аудіопрограми;

– обчислення середньої оцінки аудіопрограми, сезону та випуску;

– редагування даних про сезон аудіопрограми;

– публікація випусків аудіопрограми;

– відкладена публікація випусків аудіопрограми;

– авторизація користувачів;

– реєстрація користувачів.

А.3.2 Вимоги до складу та параметрів технічних засобів

Для роботи програми має бути виділено окремий сервер, що має мінімально відповідати наступним вимогам:

– процесор, який має 4 ядра з тактовою частотою в 2,66 ГГц;

– оперативна пам’ять обсягом у 8 Гб;

– жорсткий диск з загальним обсягом не менше 8 Гб, призначений для серверного використання.

А.4 Порядок контролю та приймання

Календарний план є основою для контролю виконання роботи, у свою чергу приймання роботи здійснюється під час її захисту перед екзаменаційною комісією.

ДОДАТОК Б

Текст програми

Б.1 Текст файла AudioProgramController.cs

```
using AudioProgramSoftware.BaseClasses;
using AudioProgramSoftware.Models;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;

namespace AudioProgramSoftware.Controllers
{
    public class AudioProgramController: Controller
    {
        private readonly APDataBaseContext _con;

        public AudioProgramController (APDataBaseContext c)
        {
            _con = c;
        }

        public ActionResult Index()
        {
            return Rating();
        }

        public ActionResult Rating()
        {
            var programs = _con.AudioPrograms.Include(a =>
a.AudioProgramId).Include(a => a.Name).Include(a => a.Info).Include(a =>
a.Cover).Include(a => a.PubDate).Include(a => a.Followers).Include(a =>
a.AccountId).Include(a => a.PlaceId).Include(a => a.Score).Include(a =>
a.PubDateStr).Include(a => a.ScoreStr).Include(a => a.Account).Include(a
=> a.Place).OrderByDescending (a => a.Score);
            var programslist = programs.ToList();
            return View(programslist);
        }

       public ActionResult ReverseRating()
```

```
            {
                var programs = _con.AudioPrograms.Include(a =>
a.AudioProgramId).Include(a => a.Name).Include(a => a.Info).Include(a =>
a.Cover).Include(a => a.PubDate).Include(a => a.Followers).Include(a =>
a.AccountId).Include(a => a.PlaceId).Include(a => a.PubDateStr).Include(a
=> a.ScoreStr).Include(a => a.Score).Include(a => a.Account).Include(a =>
a.Place).OrderBy(a => a.Score);
                var programslist = programs.ToList();
                return View(programslist);
            }

            public ActionResult NewAudioPrograms()
            {
                var programs = _con.AudioPrograms.Include(a =>
a.AudioProgramId).Include(a => a.Name).Include(a => a.Info).Include(a =>
a.Cover).Include(a => a.PubDate).Include(a => a.Followers).Include(a =>
a.AccountId).Include(a => a.PlaceId).Include(a => a.PubDateStr).Include(a
=> a.ScoreStr).Include(a => a.Score).Include(a => a.Account).Include(a =>
a.Place).OrderByDescending (a => a.PubDate);
                var programslist = programs.ToList();
                return View(programslist);
            }

            public ActionResult OldAudioPrograms()
            {
                var programs = _con.AudioPrograms.Include(a =>
a.AudioProgramId).Include(a => a.Name).Include(a => a.Info).Include(a =>
a.Cover).Include(a => a.PubDate).Include(a => a.Followers).Include(a =>
a.AccountId).Include(a => a.PlaceId).Include(a => a.PubDateStr).Include(a
=> a.ScoreStr).Include(a => a.Score).Include(a => a.Account).Include(a =>
a.Place).OrderBy (a => a.PubDate);
                var programslist = programs.ToList();
                return View(programslist);
            }

            public ActionResult MaxFollowers()
            {
                var programs = _con.AudioPrograms.Include(a =>
a.AudioProgramId).Include(a => a.Name).Include(a => a.Info).Include(a =>
a.Cover).Include(a => a.PubDate).Include(a => a.Followers).Include(a =>
a.AccountId).Include(a => a.PlaceId).Include(a => a.PubDateStr).Include(a
=> a.ScoreStr).Include(a => a.Score).Include(a => a.Account).Include(a =>
a.Place).OrderByDescending (a => a.Followers);
```

```
                var programslist = programs.ToList();
                return View(programslist);
            }

            public ActionResult MinFollowers()
            {
                var programs = _con.AudioPrograms.Include(a =>
a.AudioProgramId).Include(a => a.Name).Include(a => a.Info).Include(a =>
a.Cover).Include(a => a.PubDate).Include(a => a.Followers).Include(a =>
a.AccountId).Include(a => a.PlaceId).Include(a => a.PubDateStr).Include(a
=> a.ScoreStr).Include(a => a.Score).Include(a => a.Account).Include(a =>
a.Place).OrderBy(a => a.Followers);
                var programslist = programs.ToList();
                return View(programslist);
            }

            public ActionResult AccountAudioPrograms(int account)
            {
                var programs = _con.AudioPrograms.Include(a =>
a.AudioProgramId).Include(a => a.Name).Include(a => a.Info).Include(a =>
a.Cover).Include(a => a.PubDate).Include(a => a.Followers).Include(a =>
a.AccountId).Include(a => a.PlaceId).Include(a => a.PubDateStr).Include(a
=> a.ScoreStr).Include(a => a.Score).Include(a => a.Account).Include(a =>
a.Place).Where(a => a.AccountId = account).OrderByDescending (a =>
a.PubDate);
                var programslist = programs.ToList();
                return View(programslist);
            }

            public ActionResult PlaceAudioPrograms(int place)
            {
                var programs = _con.AudioPrograms.Include(a =>
a.AudioProgramId).Include(a => a.Name).Include(a => a.Info).Include(a =>
a.Cover).Include(a => a.PubDate).Include(a => a.Followers).Include(a =>
a.AccountId).Include(a => a.PlaceId).Include(a => a.PubDateStr).Include(a
=> a.ScoreStr).Include(a => a.Score).Include(a => a.Account).Include(a =>
a.Place).Where(a => a.PlaceId = place).OrderByDescending (a =>
a.PubDate);
                var programslist = programs.ToList();
                return View(programslist);
            }

            public ActionResult Details(int number)
```

```
            {
                var program = _con.AudioPrograms.Include(a =>
a.AudioProgramId).Include(a => a.Name).Include(a => a.Info).Include(a =>
a.Cover).Include(a => a.PubDate).Include(a => a.Followers).Include(a =>
a.AccountId).Include(a => a.PlaceId).Include(a => a.PubDateStr).Include(a
=> a.ScoreStr).Include(a => a.Score).Include(a => a.Account).Include(a =>
a.Place).Include(a => a.AudioProgramCategories).Include(a =>
a.AudioProgramScores).Include(a => a.Follows).Include(a =>
a.Seasons).FirstOrDefault(a => a.AudioProgramId == number);
                if (program == null)
                {
                    return NotFound();
                }
                return View(program);
            }

            public ActionResult SearchAudioPrograms(string searchp)
            {
                var programs = _con.AudioPrograms.Include(a =>
a.AudioProgramId).Include(a => a.Name).Include(a => a.Info).Include(a =>
a.Cover).Include(a => a.PubDate).Include(a => a.Followers).Include(a =>
a.AccountId).Include(a => a.PlaceId).Include(a => a.PubDateStr).Include(a
=> a.ScoreStr).Include(a => a.Score).Include(a => a.Account).Include(a =>
a.Place).Where(a => a.Name.Contains(searchp)).OrderByDescending (a =>
a.Score);
                var programslist = programs.ToList();
                return View(programslist);
            }

            [Authorize]
            public ActionResult ScoreAudioProgram (int number, int sc)
            {
                var accounts = _con.Accounts;
                var acc = User.Identity.Name;
                var ac = accounts.FirstOrDefaultAsync(a => a.Login ==
acc);
                var program = _con.AudioProgramScores.FirstOrDefault (a
=> a.AccountId == ac.AccountId && a.AudioProgramId == number);
                if (program != null)
                {
                    program.Score = sc;
                    program.Publication = DateTime.Now;
                    _con.Update(program);
```

```
                    _con.SaveChanges();
                }
                AudioProgramScore aps = new AudioProgramScore();
                aps.Publication = DateTime.Now;
                aps.Score = sc;
                aps.AccountId = ac.AccountId;
                aps.AudioProgramId = number;
                _con.Add(aps);
                _con.SaveChanges();
                return RedirectToAction("Index");
            }

            [Authorize]
            public ActionResult CalculateScores ()
            {
                var programs = _con.AudioPrograms.Include(a =>
a.AudioProgramId);
                foreach (var program in programs)
                {
                    CalculateScore(program.AudioProgramId);
                }
            }

            [Authorize]
            public ActionResult CalculateScore (int number)
            {
                var program = _con.AudioPrograms.Include(a =>
a.AudioProgramId).Include(a => a.Name).Include(a => a.Info).Include(a =>
a.Cover).Include(a => a.PubDate).Include(a => a.Followers).Include(a =>
a.AccountId).Include(a => a.PlaceId).Include(a => a.PubDateStr).Include(a
=> a.ScoreStr).Include(a => a.Score).Include(a => a.Account).Include(a =>
a.Place).Include(a => a.AudioProgramCategories).Include(a =>
a.AudioProgramScores).Include(a => a.Follows).Include(a =>
a.Seasons).FirstOrDefault(a => a.AudioProgramId == number);
                if (program == null)
                {
                    return NotFound();
                }
                float sum = 0;
                int c = program.AudioProgramScores.Count();
                foreach (var ac in program.AudioProgramScores)
                {
                    sum = sum + ac.Score/c;
```

```
                }
                program.Score = sum;
                _con.Update(program);
                _con.SaveChanges();
            }

            [Authorize]
            public ActionResult CalculateAFollowers ()
            {
                var programs = _con.AudioPrograms.Include(a =>
a.AudioProgramId);
                foreach (var program in programs)
                {
                    CalculateFollowers(program.AudioProgramId);
                }
            }

            [Authorize]
            public ActionResult CalculateFollowers (int number)
            {
                var program = _con.AudioPrograms.Include(a =>
a.AudioProgramId).Include(a => a.Name).Include(a => a.Info).Include(a =>
a.Cover).Include(a => a.PubDate).Include(a => a.Followers).Include(a =>
a.AccountId).Include(a => a.PlaceId).Include(a => a.PubDateStr).Include(a
=> a.ScoreStr).Include(a => a.Score).Include(a => a.Account).Include(a =>
a.Place).Include(a => a.AudioProgramCategories).Include(a =>
a.AudioProgramScores).Include(a => a.Follows).Include(a =>
a.Seasons).FirstOrDefault(a => a.AudioProgramId == number);
                if (program == null)
                {
                    return NotFound();
                }
                int sum = 0;
                foreach (var f in program.Follows)
                {
                    if (f.Status == 1)
                    {
                        sum = sum + 1;
                    }
                }
                program.Followers = sum;
                _con.Update(program);
                _con.SaveChanges();
```

```
        }

        [Authorize]
        public ActionResult FollowAudioProgram(int number)
        {
            var accounts = _con.Accounts;
            var acc = User.Identity.Name;
            var ac = accounts.FirstOrDefaultAsync(a => a.Login ==
acc);
            var foll = _con.Follows.FirstOrDefault (a =>
a.AccountId == ac.AccountId && a.AudioProgramId == number);
            if (foll != null)
            {
                if (foll.Status == 1)
                {
                    foll.Status = 0;
                }
                else
                {
                    foll.Status = 1;
                }
                foll.Publication = DateTime.Now;
                _con.Update(foll);
                _con.SaveChanges();
            }
            Follow f = new Follow();
            f.Publication = DateTime.Now;
            f.Status = 1;
            f.AccountId = ac.AccountId;
            f.AudioProgramId = number;
            _con.Add(f);
            _con.SaveChanges();
            return RedirectToAction("Index");
        }

        [Authorize]
        public ActionResult Create()
        {
            ViewData["Places"] = new SelectList(_con.Places,
"PlaceId", "Name", "CountryStr");
            return View();
        }
```

```
            [HttpPost]
            [ValidateAntiForgeryToken]
            public ActionResult Create([Bind("Name, Info, Cover,
PlaceId")] AudioProgram program)
            {
                if (ModelState.IsValid)
                {
                    var accounts = _con.Accounts;
                    var acc = User.Identity.Name;
                    var ac = accounts.FirstOrDefaultAsync(a => a.Login
== acc);
                    program.AccountId = ac.AccountId;
                    program.PubDate = DateTime.Now;
                    program.Score = 0;
                    program.Followers = 0;
                    _con.Add(program);
                    _con.SaveChanges();
                    return RedirectToAction(nameof(Index));
                }
                return View(program);
            }

            [Authorize]
            public ActionResult Edit(int? number)
            {
                var program = _con.AudioPrograms.Find(number);
                if (program == null)
                {
                    return NotFound();
                }
                ViewData["Places"] = new SelectList(_con.Places,
"PlaceId", "Name", "CountryStr");
                return View(program);
            }

            [HttpPost]
            [ValidateAntiForgeryToken]
            public ActionResult Edit(int number, [Bind("AudioProgramId,
Name, Info, Cover, PlaceId")] AudioProgram program)
            {
                if (ModelState.IsValid)
                {
                    _con.Update(program);
```

```
                    _con.SaveChanges();
                }
                ViewData["Places"] = new SelectList(_con.Places,
"PlaceId", "Name", "CountryStr");
                return View(program);
            }
        }

    }
```

Б.2 Текст файла AudioProgram.cs

```
    using System;
    using System.Collections.Generic;
    using System.ComponentModel.DataAnnotations;

    namespace AudioProgramSoftware.Models
    {
        public class AudioProgram
        {
            [Key]
            public int AudioProgramId
            {get; set;}
            [Required]
            public string Name
            {get; set;}
            public string Info
            {get; set;}
            public string Cover
            {get; set;}
            [Required]
            [DataType(DataType.Date)]
            public DateTime PubDate
            {get; set;}
            public int Followers
            {get; set;}
            public float Score
            {get; set;}
            [Required]
            public int AccountId
            {get; set;}
            public int PlaceId
```

```
            {get; set;}
            public virtual Account Account
            {get; set;}
            public virtual Place Place
            {get; set;}
            public virtual ICollection<Season> Seasons
            {get; set;}
            public virtual ICollection<Follow> Follows
            {get; set;}
            public virtual ICollection<AudioProgramCategory>
AudioProgramCategories
            {get; set;}
            public virtual ICollection<AudioProgramScore>
AudioProgramScores
            {get; set;}
            [NotMapped]
            public string ScoreStr
            {
                get
                {
                     return Math.Round(this.Score, 1).ToString();
                }
            }
            [NotMapped]
            public string PubDateStr
            {
                get
                {
                    return this.PubDate.ToString("MMMM yyyy");
                }
            }
        }
    }
```

Б.3 Текст файла Season.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace AudioProgramSoftware.Models
```

```
{
    public class Season
    {
        [Key]
        public int SeasonId
        {get; set;}
        [Required]
        public string Name
        [Required]
        [DataType(DataType.Date)]
        public DateTime Start
        {get; set;}
        [Required]
        [DataType(DataType.Date)]
        public DateTime End
        {get; set;}
        public float Score
        {get; set;}
        [Required]
        public int AudioProgramId
        {get; set;}
        public virtual AudioProgram AudioProgram
        {get; set;}
        public virtual ICollection<Issue> Issues
        {get; set;}
        [NotMapped]
        public string ScoreStr
        {
            get
            {
                 return Math.Round(this.Score, 1).ToString();
            }
        }
        [NotMapped]
        public string StartStr
        {
            get
            {
                return this.Start.ToString("dd.MM.yyyy");
            }
        }
        [NotMapped]
        public string EndStr
```

```
        {
            get
            {
                return this.End.ToString("dd.MM.yyyy");
            }
        }
    }
}
```

Б.4 Текст файла Issue.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace AudioProgramSoftware.Models
{
    public enum IssueStatusEnum
    {
        Published,
        Draft,
        Waiting
    }

    public class Issue
    {
        [Key]
        public int IssueId
        {get; set;}
        [Required]
        public string Name
        [Required]
        [DataType(DataType.Date)]
        public DateTime Start
        {get; set;}
        [Required]
        [DataType(DataType.Date)]
        public DateTime Publication
        {get; set;}
        public float Score
        {get; set;}
```

```
[Required]
public int SeasonId
{get; set;}
public string Info
{get; set;}
public string Cover
{get; set;}
[Required]
public IssueStatusEnum Status
{get; set;}
public int Favorites
{get; set;}
public string Audiofile
{get; set;}
public virtual Season Season
{get; set;}
public virtual ICollection<Favor> Favors
{get; set;}
public virtual ICollection<IssueGuest> IssueGuests
{get; set;}
public virtual ICollection<IssueScore> IssueScores
{get; set;}
public virtual ICollection<Issue> Connections
{get; set;}
[NotMapped]
public string ScoreStr
{
    get
    {
         return Math.Round(this.Score, 1).ToString();
    }
}
[NotMapped]
public string StartStr
{
    get
    {
        return this.Start.ToString("dd.MM.yyyy");
    }
}
[NotMapped]
public string PublicationStr
{
```

```
            get
            {
                return this.Publication.ToString("dd.MM.yyyy");
            }
        }
        [NotMapped]
        public string StatusStr {
            get
            {
                switch(this.Status)
                {
                    case IssueStatusEnum.Published:
                        return "Опубліковано";
                    case IssueStatusEnum.Draft:
                        return "Чорнетка";
                    case IssueStatusEnum.Waiting:
                        return "В очікуванні";
                    default:
                        return "Не встановлено";
                }
            }
        }
    }
}
```

Б.5 Текст файла Follow.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace AudioProgramSoftware.Models
{
    public class Follow
    {
        [Key]
        public int FollowId
        {get; set;}
        [Required]
        [DataType(DataType.Date)]
        public DateTime Publication
```

```
        {get; set;}
        [Required]
        public bool Status
        {get; set;}
        [Required]
        public int AudioProgramId
        {get; set;}
        [Required]
        public int AccountId
        {get; set;}
        public virtual Account Account
        {get; set;}
        public virtual AudioProgram AudioProgram
        {get; set;}
        [NotMapped]
        public string PublicationStr
        {
            get
            {
                return this.Publication.ToString("dd.MM.yyyy");
            }
        }
    }
}
```

Б.6 Текст файла Favor.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace AudioProgramSoftware.Models
{
    public class Favor
    {
        [Key]
        public int FavorId
        {get; set;}
        [Required]
        [DataType(DataType.Date)]
        public DateTime Publication
```

```
        {get; set;}
        [Required]
        public bool Status
        {get; set;}
        [Required]
        public int IssueId
        {get; set;}
        [Required]
        public int AccountId
        {get; set;}
        public virtual Issue Issue
        {get; set;}
        public virtual Account Account
        {get; set;}
        [NotMapped]
        public string PublicationStr
        {
            get
            {
                return this.Publication.ToString("dd.MM.yyyy");
            }
        }
    }
}
```

Б.7 Текст файла IssueScore.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace AudioProgramSoftware.Models
{
    public class IssueScore
    {
        [Key]
        public int IssueScoreId
        {get; set;}
        [Required]
        public float Score
        {get; set;}
```

```
        [Required]
        public int IssueId
        {get; set;}
        [Required]
        public int AccountId
        {get; set;}
        [Required]
        [DataType(DataType.Date)]
        public DateTime Publication
        {get; set;}
        public virtual Issue Issue
        {get; set;}
        public virtual Account Account
        {get; set;}
        [NotMapped]
        public string PublicationStr
        {
            get
            {
                return this.Publication.ToString("dd.MM.yyyy");
            }
        }
    }
}
```

Б.8 Текст файла AudioProgramScore.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace AudioProgramSoftware.Models
{
    public class AudioProgramScore
    {
        [Key]
        public int AudioProgramScoreId
        {get; set;}
        [Required]
        public float Score
        {get; set;}
```

```
        [Required]
        public int AudioProgramId
        {get; set;}
        [Required]
        public int AccountId
        {get; set;}
        [Required]
        [DataType(DataType.Date)]
        public DateTime Publication
        {get; set;}
        public virtual AudioProgram AudioProgramId
        {get; set;}
        public virtual Account Account
        {get; set;}
        [NotMapped]
        public string PublicationStr
        {
            get
            {
                return this.Publication.ToString("dd.MM.yyyy");
            }
        }
    }
}
```

ДОДАТОК В

Слайди презентації

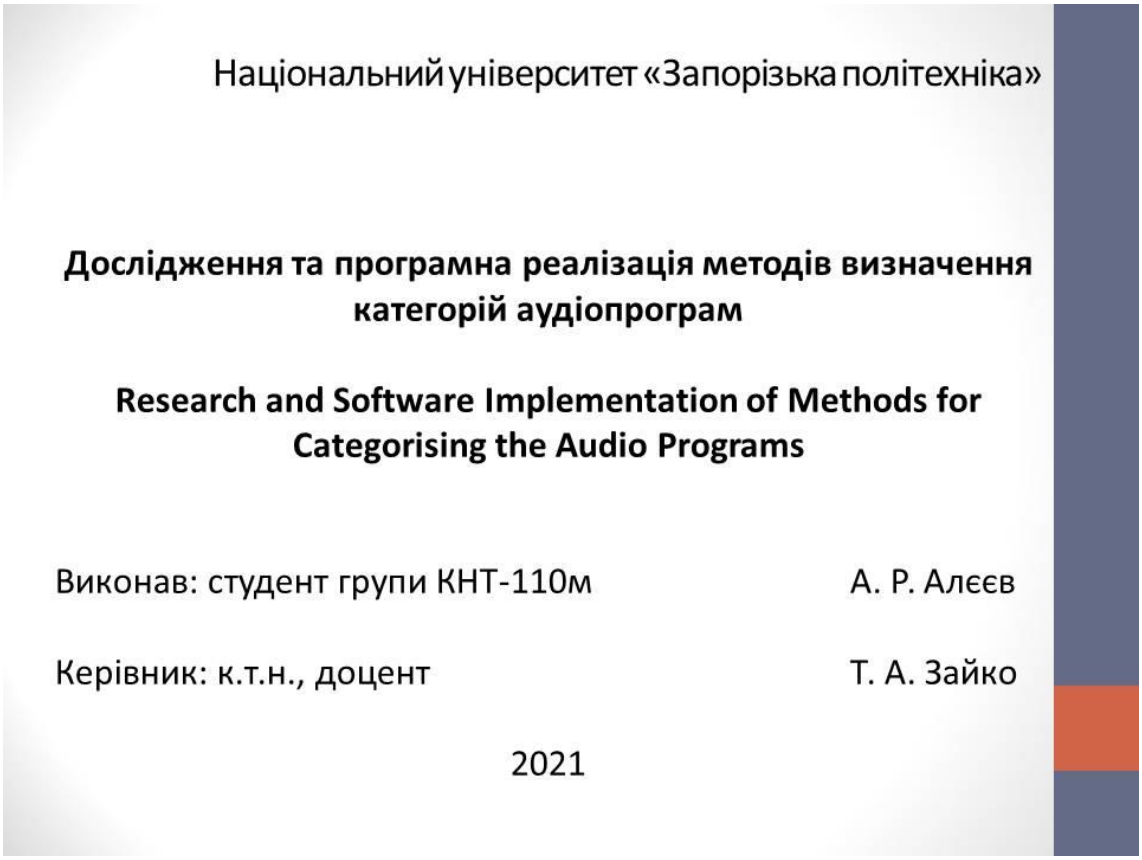


Рисунок В.1 – Слайд 1

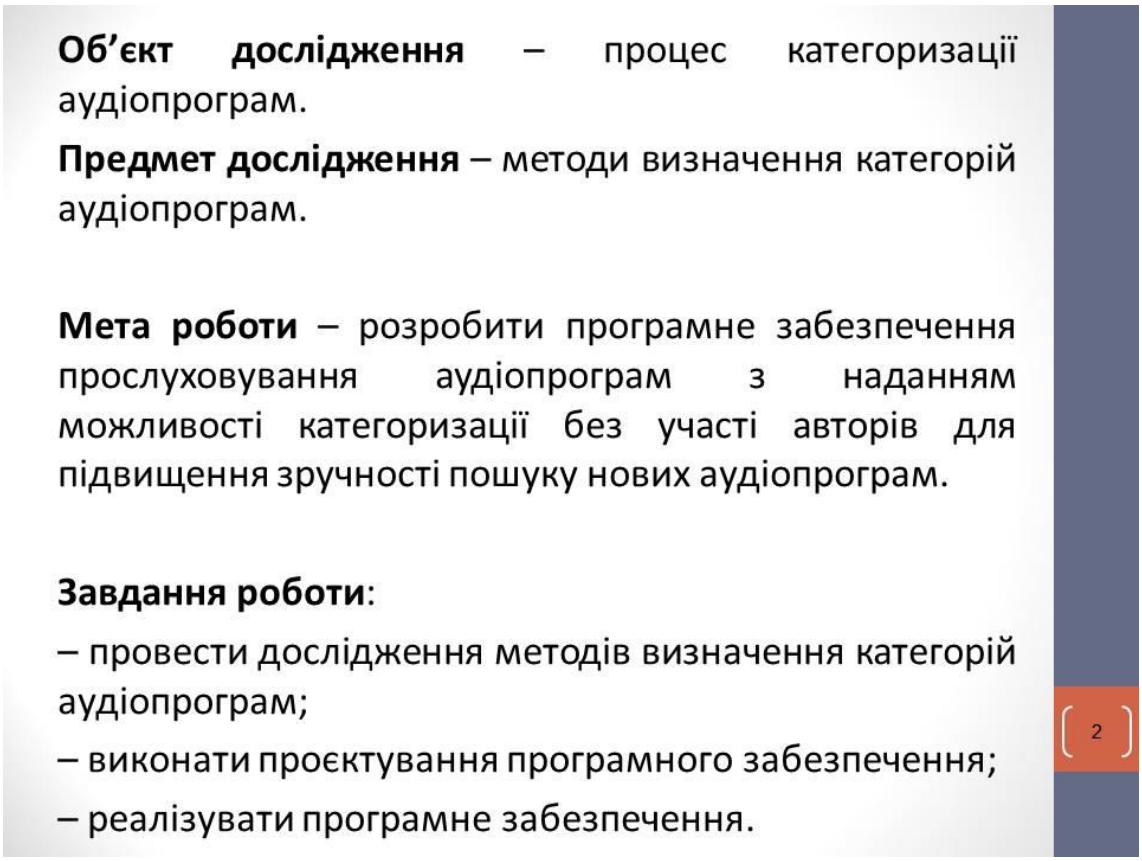


Рисунок В.2 – Слайд 2

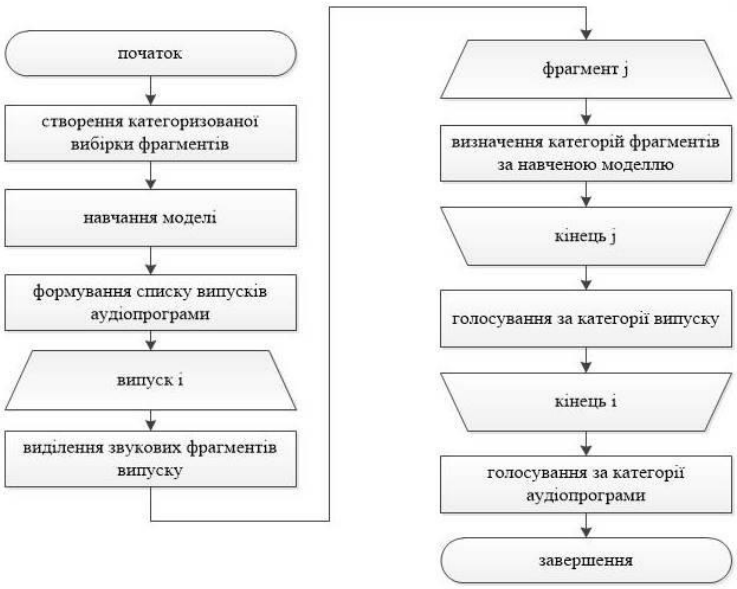
Порівняння програмних аналогів

Характеристика	Podbean	Transistor	Програма, що розробляється
Завантаження аудіопрограм на сервер	+	+	+
Зручність для нових авторів аудіопрограм	+	+	+
Тип вмісту	Аудіопрограми та відеопрограми	Аудіопрограми	Аудіопрограми
Безкоштовний доступ	Тимчасово	–	+
Автоматичне визначення категорій аудіопрограм	–	–	+

3

Рисунок В.3 – Слайд 3

Метод визначення категорій аудіопрограм



4

Рисунок В.4 – Слайд 4

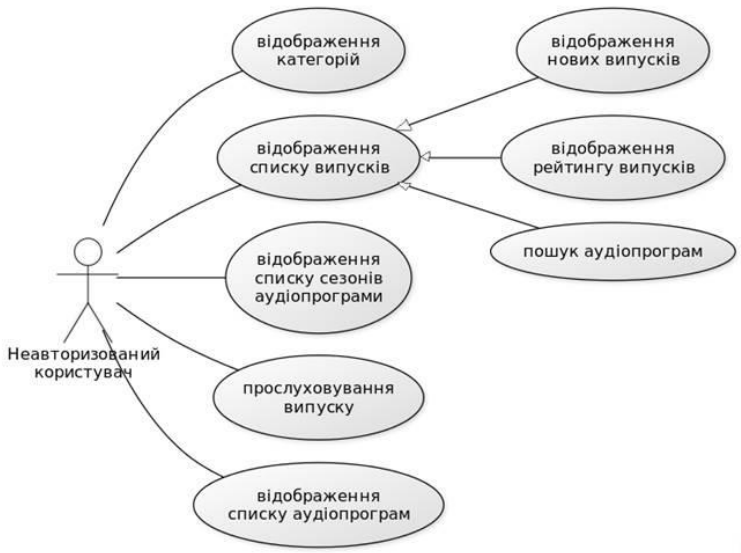
Вибір мови програмування

Характеристика	C#	Python
Вебфреймворк	ASP.NET Core MVC	Django
Високорівнева мова	+	+
Швидкість роботи програм	+	+/–
Безпека	+	+/–
Робота з засобами машинного навчання	+	+
Створення великих проєктів	+	+/–

5

Рисунок В.5 – Слайд 5

Діаграма варіантів використання програми неавторизованим користувачем



6

Рисунок В.6 – Слайд 6

Рисунок В.7 – Слайд 7

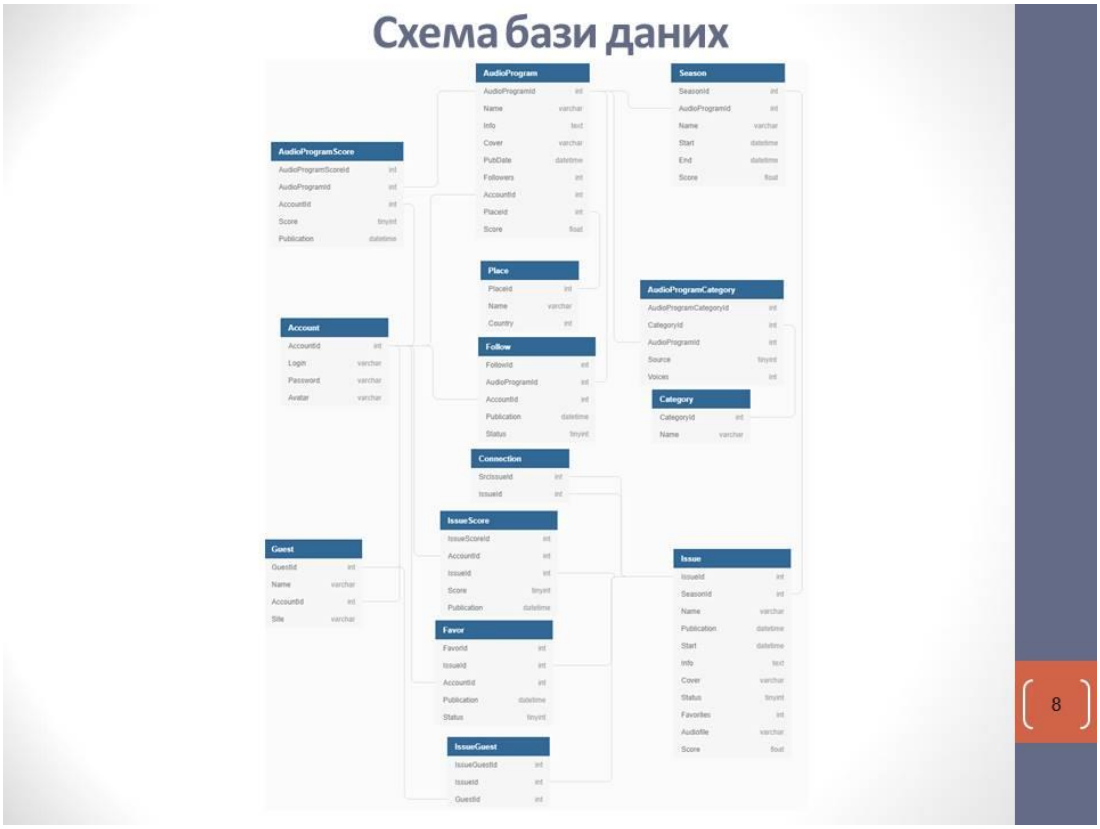


Рисунок В.8 – Слайд 8

Рисунок В.9 – Слайд 9

Рисунок В.10 – Слайд 10

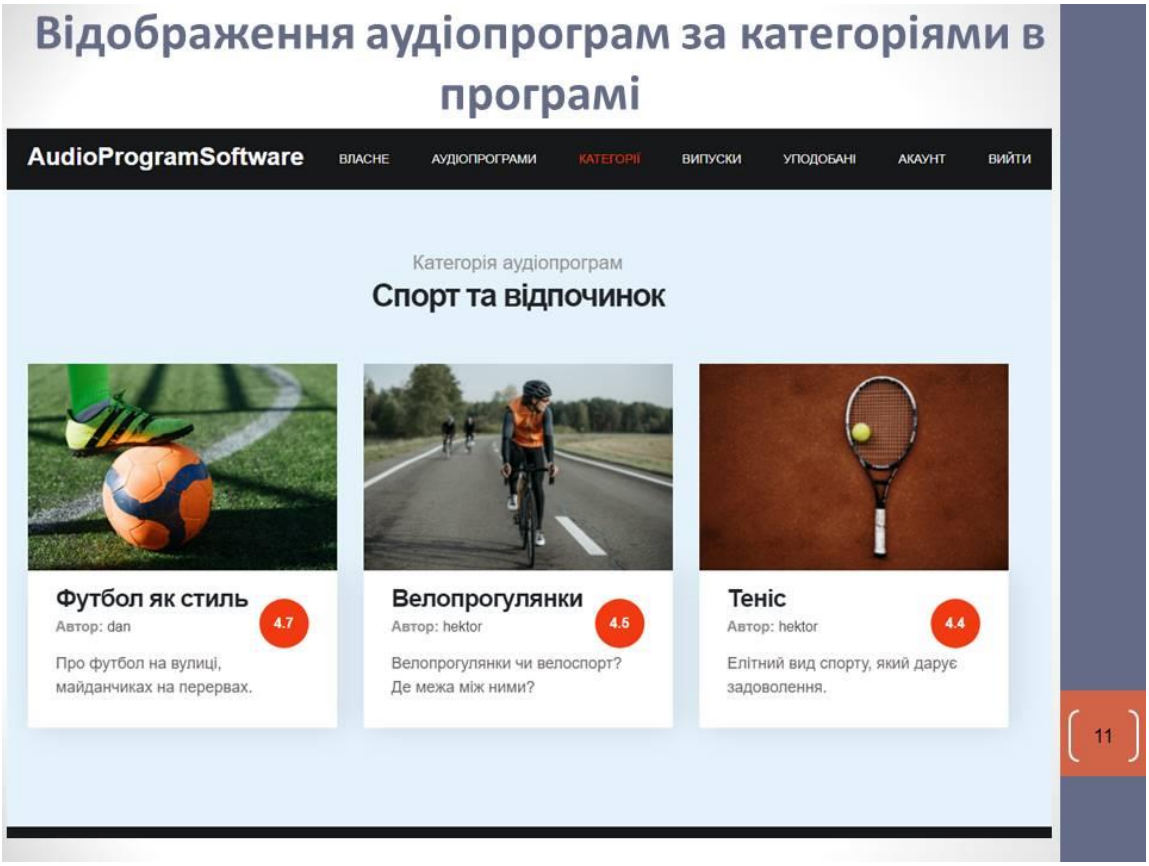


Рисунок В.11 – Слайд 11

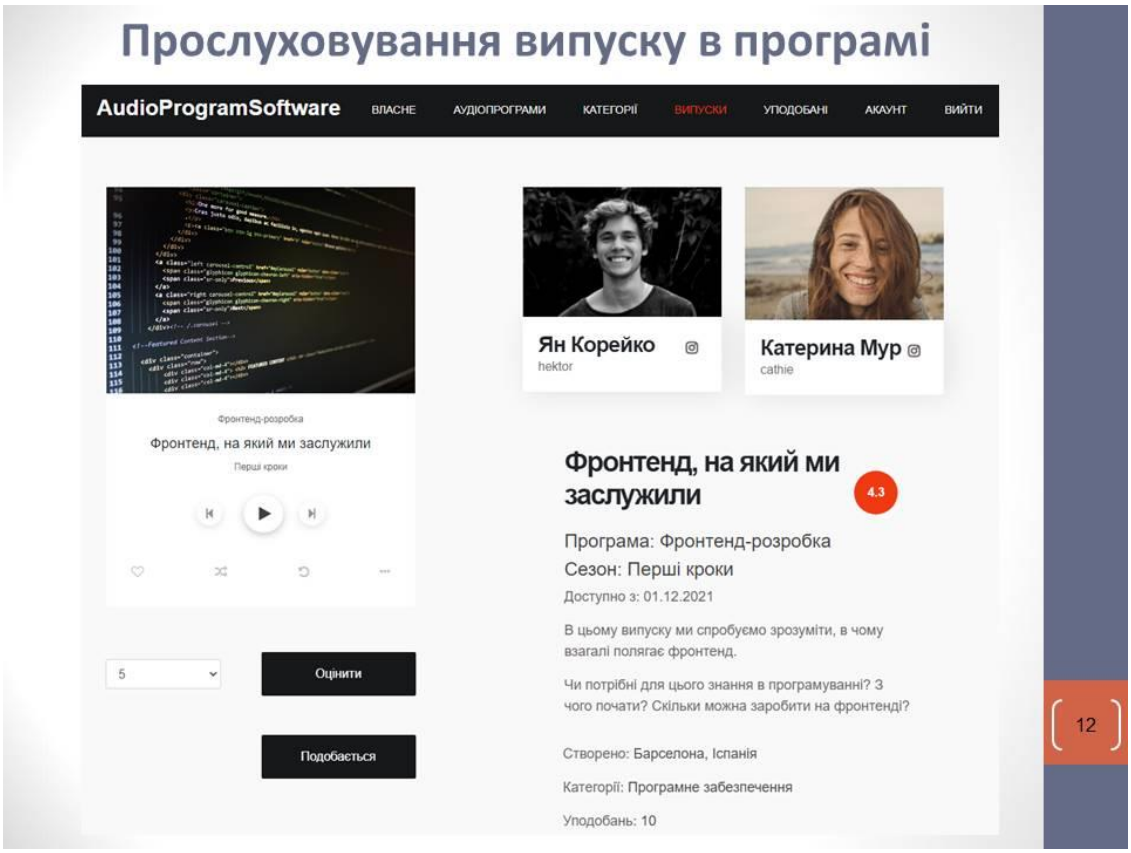


Рисунок В.12 – Слайд 12

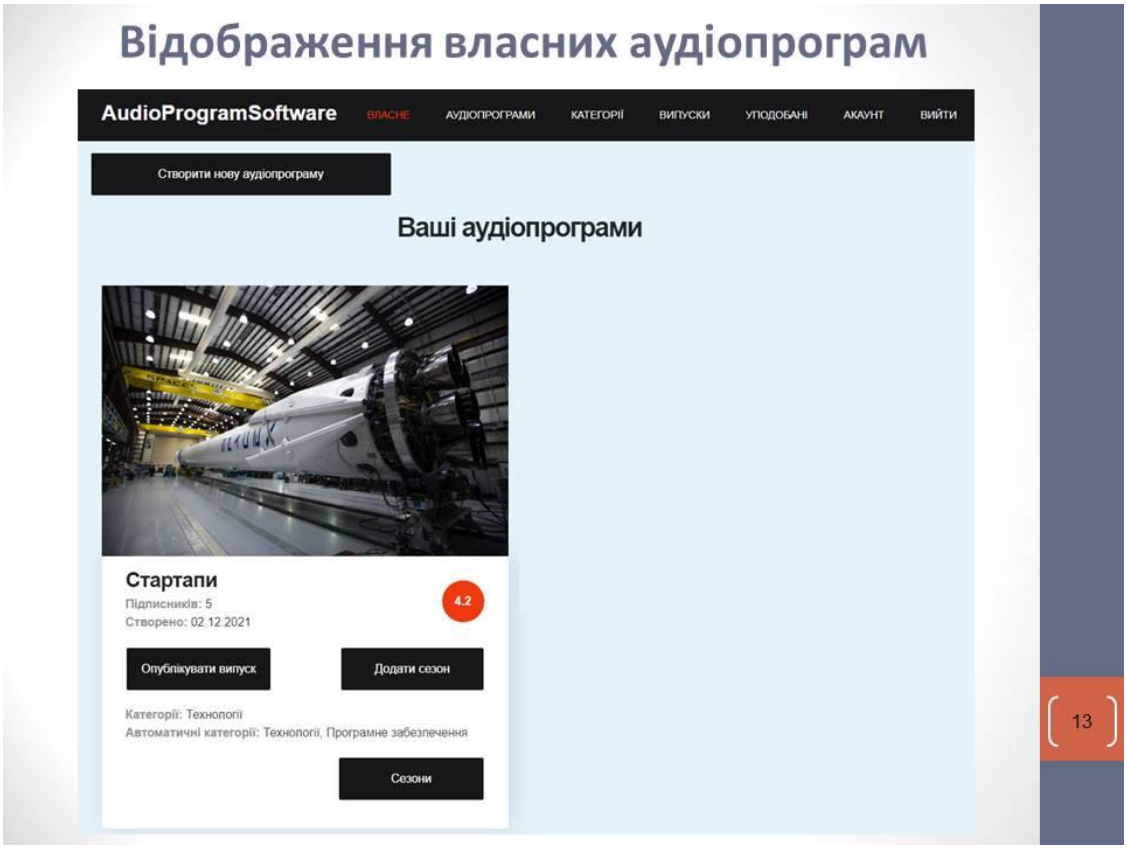


Рисунок В.13 – Слайд 13

Рисунок В.14 – Слайд 14

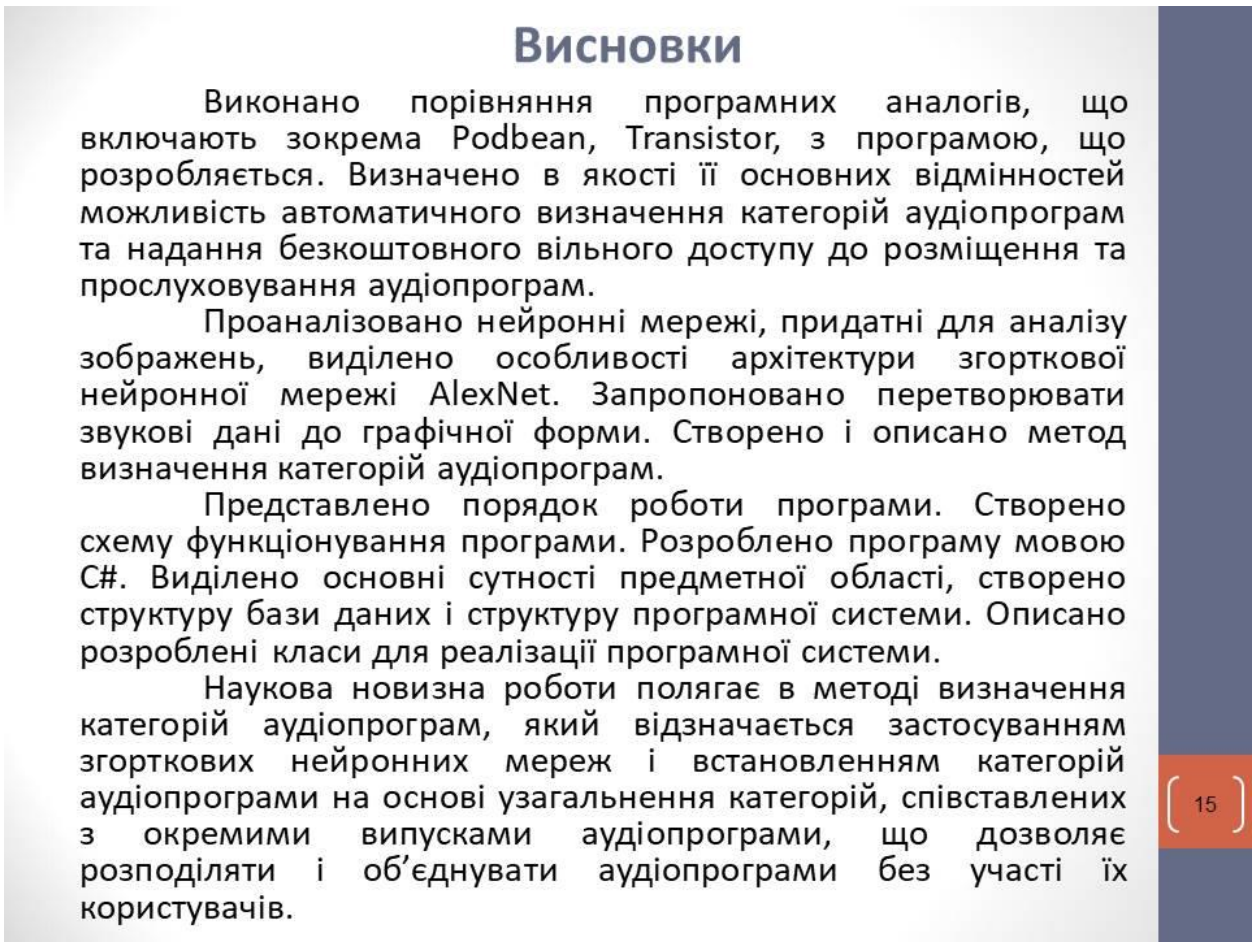


Рисунок В.15 – Слайд 15