Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: Розробка методу і програмних засобів реалізації вебсистеми автоматизованого аналізу та класифікації музичних композицій

Виконав: студент 4 курсу
групи 1ПІ-20б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Тимошенко В.Ю.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Ткаченко О.М.
(прізвище та ініціали)

Рецензент: к.т.н., доц. каф. ОТ Богомолов С. В.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри ПЗ О. Н. Романюк
«10» червня 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк
« 12 » березня 2024 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Тимошенку Валентину Юрійовичу

1. Тема роботи – розробка методу і програмних засобів реалізації вебсистеми автоматизованого аналізу та класифікації музичних композицій

Керівник роботи: Ткаченко Олександр Миколайович, к.т.н. техн. наук, доцент каф. ПЗ, затверджені наказом вищого навчального закладу від 11 березня 2024 р. №80.

2. Строк подання студентом роботи 3 червня 2024.

3. Вихідні дані до роботи включають в себе музичні композиції у форматі аудіофайлів, дані про їхні характеристики та метадані. Максимальна кількість аудіоданих для аналізу не обмежена. Також використовуються дані про жанри, виконавців та інші музичні властивості композицій. Результатом роботи є створення моделі класифікації на основі згорткової нейронної мережі та зв'язок з відповідними музичними категоріями.

4. Текстова частина роботи включає в себе теоретичний огляд згорткових нейронних мереж, методи пошуку та аналізу музичних даних, а також алгоритми використані для класифікації музичних композицій. Основні розділи роботи включають теорію машинного навчання, аналіз музичних характеристик, алгоритми обробки сигналів, а також алгоритми роботи зі звуковими файлами.

5. Ілюстративна частина роботи містить зображення графіків, що ілюструють роботу нейронної мережі, схеми алгоритмів пошуку та аналізу музичних даних, а також скріншоти інтерфейсу вебсистеми для автоматизованого аналізу і класифікації музичних композицій. Також включаються графіки, що демонструють результати тестування системи та її ефективність у класифікації музичних жанрів.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Ткаченко О.М., к.т.н., доц каф ПЗ	13.03.2024	03.06.2024

7. Дата видачі завдання 13 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз розвитку вебсистем класифікації	14.03.24 - 31.03.24	виконано
2	Розробка архітектури моделі CNN	01.04.24 - 18.04.24	виконано
3	Розробка алгоритмів роботи класифікації та інтерфейсу вебсистеми	19.04.24 - 06.05.24	виконано
4	Тестування вебсистеми	07.05.24 - 24.05.24	виконано
5	Оформлення матеріалів до захисту БДР	25.05.24 - 30.05.24	виконано

Студент _____ **В.Ю. Тимошенко**
(підпис) (ініціали та прізвище)

Керівник бакалаврської дипломної роботи _____ **О. М. Ткаченко**
(підпис) (ініціали та прізвище)

Анотація

УДК 378:004.8

Тимошенко В.Ю. Розробка програмних засобів реалізації вебсистеми для автоматизованого аналізу та класифікації музичних композицій: бакалаврська дипломна робота зі спеціальності 121 Інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2024. 58 с.

На укр. мові. Бібліогр. : 22 назв ; рис. : 30; табл. 3.

У бакалаврській дипломній роботі розроблено програмні засоби, алгоритми та структури даних для створення вебсистеми для автоматичного аналізу та класифікації музичних композицій з використанням моделі згорткової нейронної мережі (CNN). Обґрунтовано актуальність та доцільність розробки даної системи, а також було проведено тестування програми для виявлення та усунення недоліків. Для реалізації системи використано мову програмування Python, фреймворк TensorFlow для реалізації моделі CNN та Django для створення вебінтерфейсу. Система дозволяє автоматично аналізувати та класифікувати музичні композиції з використанням методів глибинного навчання.

Annotation

Tymoshenko V.Y. Development of software tools for the implementation of a web system for automated analysis and classification of musical compositions: bachelor's thesis in the specialty 121 Software Engineering, educational program - Software Engineering. Vinnytsia: VNTU, 2024. 58 p.

In Ukrainian. Bibliography: 22 titles; Figures: 30; Table 3.

In the bachelor's thesis, software tools, algorithms and data structures were developed to create a web-based system for automatic analysis and classification of musical compositions using a convolutional neural network (CNN) model. The relevance and feasibility of developing this system are substantiated, and the program was tested to identify and eliminate deficiencies. To implement the system, the Python programming language, the TensorFlow framework for implementing the CNN model, and Django for creating a web interface were used. The system allows to automatically analyze and classify musical compositions using deep learning method

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. У сучасному світі обробка та аналіз даних стають все важливішими, особливо з огляду на зростаючі обсяги інформації. Музична індустрія не є винятком, і з кожним роком кількість доступних аудіофайлів зростає експоненційно. Завдяки цьому, автоматичне розпізнавання жанрів музичних композицій стає дедалі важливішим завданням.

Вибір даної теми базується на потребі в розвитку систем, які можуть ефективно аналізувати та класифікувати великі обсяги аудіоданих. Розпізнавання музичних жанрів має широкі практичні застосування, включаючи музичні стрімінгові сервіси, рекомендаційні системи, сортування музичних колекцій, а також дослідження музичних тенденцій.

Створення вебсистеми для автоматичного аналізу та класифікації музичних композицій на основі згорткових нейронних мереж (CNN) відповідає потребам сучасного інформаційного суспільства. Використання CNN для розпізнавання жанрів музики дозволяє створити надійну та ефективну систему, яка може працювати з різними типами музичних жанрів та досягати високої точності класифікації.

Подальший розвиток такої системи може сприяти удосконаленню аналізу та розумінню музичних тенденцій, а також полегшити взаємодію між користувачами та музичним контентом в онлайн середовищі. Тому вибір даної теми відповідає актуальним вимогам і напрямкам розвитку сучасних технологій.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно з планом виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. *Метою бакалаврської дипломної роботи* є підвищення ймовірності правильної класифікації аудіо даних за рахунок створення вебсистеми для автоматичного аналізу та класифікації музичних композицій на основі згорткових нейронних мереж.

Для досягнення поставленої мети необхідно розв’язати такі **задачі:**

- провести аналіз існуючих методів і засобів розпізнавання музичних жанрів;
- запропонувати та розробити загальний алгоритм системи на основі CNN;
- реалізувати архітектуру CNN для розпізнавання музичних жанрів;
- використати GTZAN датасет для тренування та тестування моделі;
- розробити вебінтерфейс для взаємодії з системою;
- здійснити тестування системи згідно поставлених задач;
- підготувати систему до збереження та подальшого використання для класифікації нових аудіофайлів.

Об'єктом дослідження є процес розпізнавання жанрів музичних композицій.

Предметом дослідження є методи, алгоритми та програмні засоби розпізнавання музичних жанрів на основі згорткових нейронних мереж та їх інтеграція у вебсистему.

Методи дослідження. В процесі виконання роботи використовувалися методи обробки аудіо сигналів для створення спектрограм, алгоритми згорткових нейронних мереж для класифікації даних, методи машинного навчання для тренування та тестування моделі, а також технології веброзробки для створення користувацького інтерфейсу.

Новизна отриманих результатів.

1. **Удосконалено** архітектуру CNN, яка відрізняється від існуючих наявністю трьох згорткових шарів з функціями активації ReLU, шарів макспулінгу та нормалізації, а також повнозв'язного шару та вихідного шару з функцією активації softmax, що доволило підвищити ймовірність правильного визначення жанру композиції.

2. **Отримав подальшого розвитку** метод машинного навчання, який відрізняється від існуючих використання GTZAN Dataset для отримання характеристик аудіофайлів та збереження характеристик аудіофайлів у форматі JSON, що дозволило прискорити процес навчання нейронної мережі.

Практичне значення одержаних результатів. Розроблено вебсистему для автоматичного аналізу та класифікації музичних композицій, яка може бути використана для створення додатків та систем, що дозволяють автоматизувати процес класифікації музичних жанрів. Система забезпечує зручний інтерфейс для взаємодії з користувачами та може бути інтегрована у різні платформи для аналізу аудіо даних.

Особистий внесок здобувача. Усі наукові результати, викладені у бакалаврській дипломній роботі, отримані автором особисто.

Апробація матеріалів бакалаврської дипломної роботи. Усі наукові результати, викладені у дипломній роботі, доповідалися на Міжнародній науково-практичній інтернет-конференції "Молодь в науці: дослідження, проблеми, перспективи" (2024).

Публікації. Основні результати досліджень було опубліковано у матеріалах Міжнародної науково-практичної інтернет-конференції "Молодь в науці: дослідження, проблеми, перспективи" (2024) [1].

1 АНАЛІЗ РОЗВИТКУ ВЕБСИСТЕМ ДЛЯ АНАЛІЗУ І КЛАСИФІКАЦІЇ МУЗИЧНИХ КОМПОЗИЦІЙ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану вебсистем з використанням засобів машинного навчання

На сьогоднішній день велика кількість вебсистем використовують засоби машинного навчання для покращення своєї функціональності та ефективності. Ці системи можуть бути як великими електронними комерційними платформами, так і невеликими веб додатками.

Деякі з провідних компаній у сфері Інтернет-технологій, такі як Google, Amazon, Facebook, активно використовують засоби машинного навчання в своїх вебсистемах для покращення користувацького досвіду та забезпечення більш ефективного управління даними. Розробка програмних засобів для реалізації вебсистеми з використанням машинного навчання є важливим завданням у цьому плані, оскільки дозволяє створювати інноваційні рішення, які покращують користувацький досвід та надають конкурентні переваги бізнесу.

Вебсистеми, що використовують засоби машинного навчання, можуть використовуватися для різних цілей, включаючи автоматичну персоналізацію контенту, рекомендації для користувачів, прогнозування попиту та аналіз великих обсягів даних.

Один із типових прикладів - це системи рекомендацій, які використовують алгоритми машинного навчання для аналізу поведінки користувачів та надання персоналізованих рекомендацій щодо товарів або контенту.

Отже, розробка програмних засобів для впровадження вебсистеми з використанням машинного навчання для аналізу та класифікації музики значно покращує користувацький досвід, оскільки система може точніше аналізувати музичні переваги користувачів та забезпечувати персоналізовані рекомендації.

Застосування машинного навчання дозволяє системі вивчати вподобання користувачів, що стосуються жанрів, артистів чи специфічних музичних особливостей, таких як темп, тональність і інструментал. Крім того, система може автоматично класифікувати великі обсяги музики, роблячи її легко доступною і зрозумілою для користувачів. Це особливо важливо для нових або менш відомих виконавців, які шукають шляхи до своєї аудиторії. Завдяки машинному навчанню, музичні вебсистеми стають більш інноваційними і конкурентоспроможними, що пропонує значні переваги

1.2 Порівняльний аналіз аналогів

Можливість автоматичної класифікації музичних композицій, де користувач завантажує музику, а вебсайт чи застосунок аналізує її, є важливою та популярною функцією у сучасній музичній індустрії. Це дозволяє створювати персоналізовані музичні рекомендації для користувачів, полегшує відкриття нових виконавців і жанрів, а також сприяє аналізу музичних трендів. З розвитком технологій машинного навчання та обробки сигналів ця можливість матиме ще більші перспективи у майбутньому. До найбільш відомих рішень відносяться:

- Spotify;
- Apple Music;
- Pandora;
- Last.fm;
- Deezer.

Одним з популярних рішень для класифікації музики є Spotify (рис. 1.1), він використовує алгоритми машинного навчання для рекомендацій музики користувачам на основі їхніх звичок прослуховування [2]. Ці алгоритми аналізують музичні характеристики композицій, такі як темп, тон, ритм, та створюють персоналізовані плейлисти та рекомендації для кожного користувача.

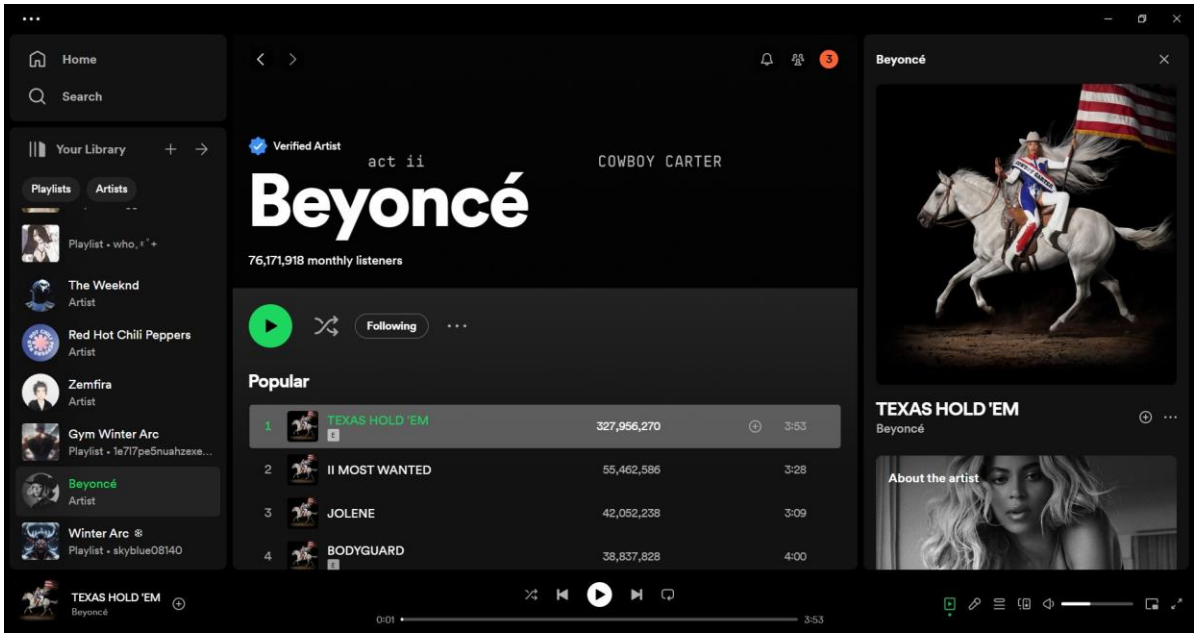


Рисунок 1.1 – Приклад роботи Spotify

Така онлайн-платформа як Pandora (рис. 1.2) використовує алгоритми машинного навчання для автоматичного створення персоналізованих радіостанцій для користувачів на основі їхніх музичних вподобань [3]. Вони аналізують аудіофайли для виявлення спільних характеристик.

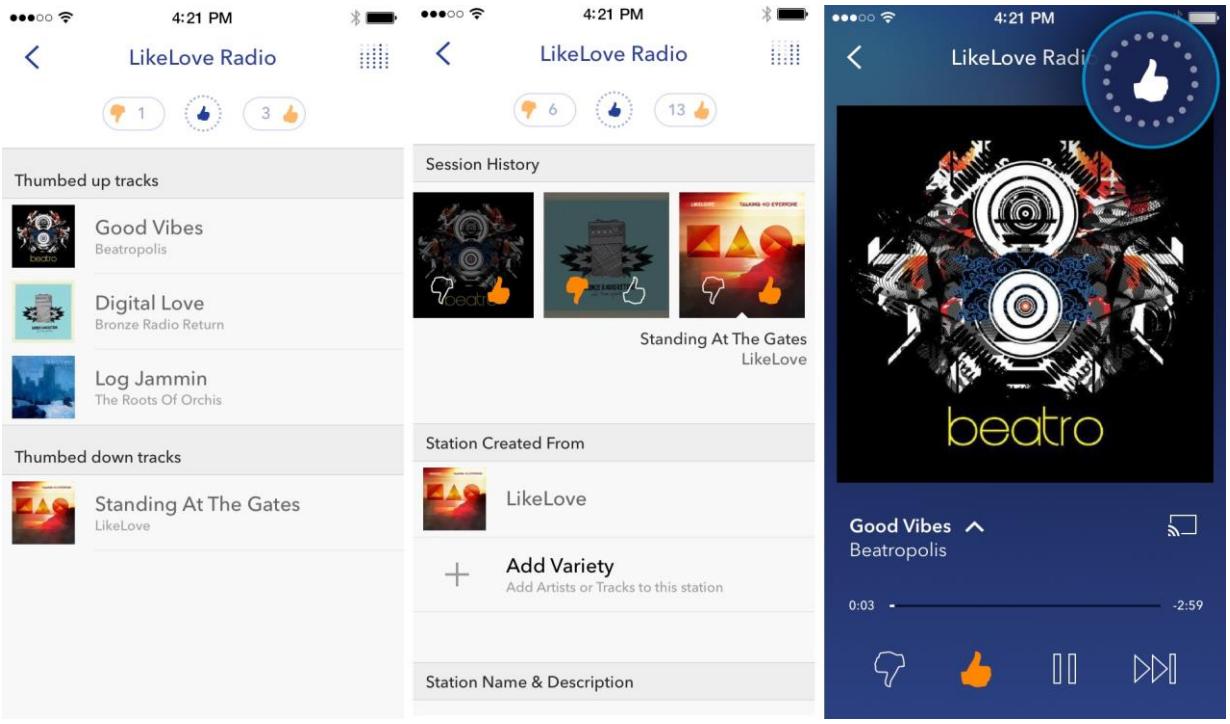


Рисунок 1.2 – Приклад роботи Pandora

Інший приклад - Apple Music (рис. 1.3) розроблена компанією Apple, яка надає доступ до мільйонів треків, альбомів та виконавців. Ця музична платформа надає функцію "Apple Music Replay", яка автоматично генерує персоналізований плейлист із найбільш часто відтворюваними треками користувача протягом року [4]. Ця функція використовує алгоритми аналізу даних для класифікації та ранжування музичних композицій.

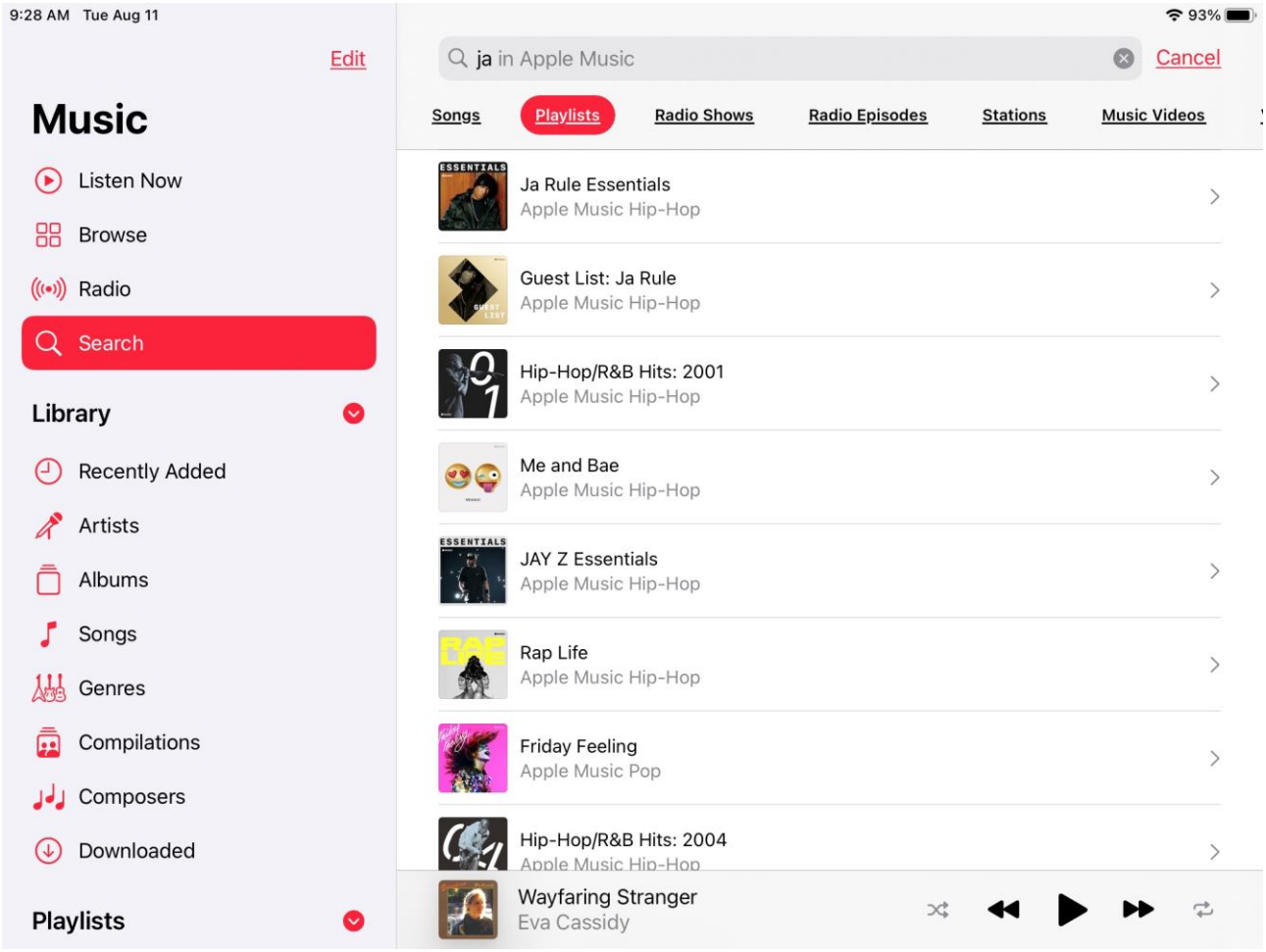


Рисунок 1.3 – Приклад роботи Apple Music

Онлайн-платформа для музичного стрімінгу та соціальна мережа Last.fm (рис. 1.4) ,яка відстежує слухання користувачів та використовує цю інформацію для рекомендацій музики [5]. Сервіс аналізує прослуховування користувачів та використовує ці дані для рекомендацій нових виконавців і треків. Він також надає інформацію про музичні події, концерти та новини зі світу музики. Хоча

це не точна класифікація музичних композицій на основі завантажень, але цей сервіс аналізує вподобання користувача та рекомендує подібні треки на основі їхнього прослуховування.

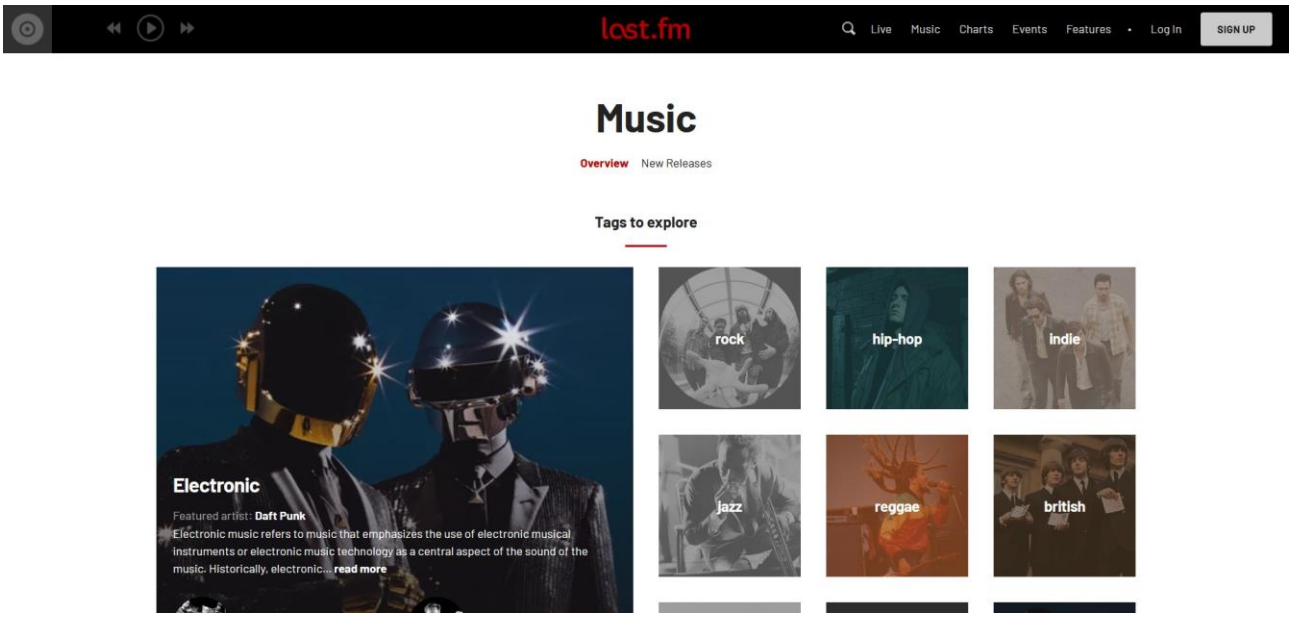


Рисунок 1.4 – Приклад роботи Last.fm

Популярний сервіс для музичного стрімінгу Deezer (рис. 1.5), який пропонує доступ до великої бібліотеки музики, підписний доступ до треків, плейлисти та персоналізовані рекомендації [6]. Він також має функції, такі як редагування текстів пісень, аудіокниги та подкасти. Deezer дозволяє користувачам слухати музику онлайн, а також завантажувати її для прослуховування офлайн.

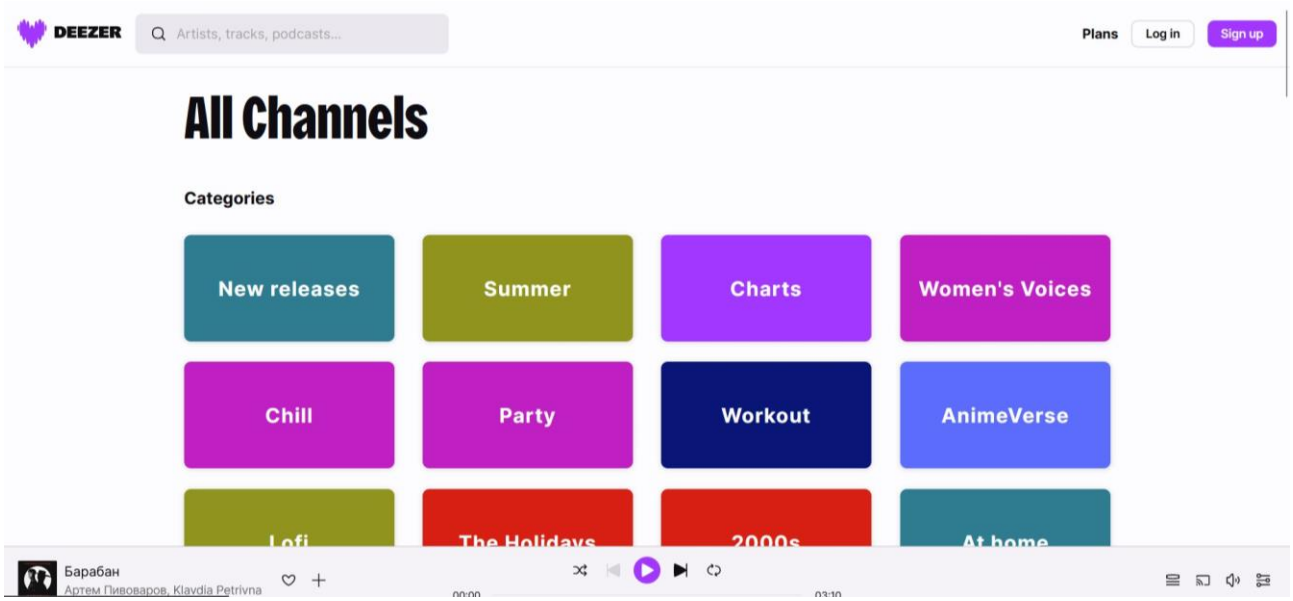


Рисунок 1.5 – Приклад роботи Deezer

Розробка програмних модулів для впровадження мобільної системи з використанням доповненої реальності вимагає досвіду в галузі AR-технологій, розробки програмного забезпечення та дизайну користувацького досвіду. Це передбачає створення внутрішньої системи, яка може розпізнавати та обробляти реальне середовище, а також безперешкодно інтегрувати віртуальні об'єкти в це середовище. Інтерфейсна система передбачає розробку користувацьких інтерфейсів, які використовують технологію доповненої реальності для створення цікавого та інтуїтивно зрозумілого досвіду.

Результати порівняння аналогів зведено в табл. 1.1.

Таблиця 1.1 – Порівняльні характеристика онлайн платформ

Критерії	Spotify	Apple Music	Pandora	Last.fm	Deezer	Власний модуль
Можливість прослуховування завантажених композицій	+	+	-	+	-	+

Підтримка різних форматів аудіо	+	+	+	-	+	+
Наявність функції керування відтворенням	+	+	+	+	-	+
Ефективне використання ресурсів	-	+	-	+	-	+
Збереження імен файлів	-	-	-	+	+	+
Загальна оцінка	60%	80%	40%	80%	40%	100%

Згідно з порівняльним аналізом у таблиці, розробка власної вебсистеми з використанням машинного навчання виявляється перспективною. Отриманий продукт зможе усунути недоліки наявних рішень і відкрити нові можливості, включаючи застосування технології доповненої реальності.

Таким чином, розробка програмного забезпечення для впровадження вебсистеми з машинним навчанням розкриває широкі перспективи для взаємодії з клієнтами, створення унікального користувацького досвіду та залучення та утримання користувачів. З розумінням потенційних переваг застосування технології доповненої реальності та вимог до розробки програмного забезпечення, компанії можуть створювати інноваційні та привабливі вебсистеми, які використовують машинне навчання для досягнення успіху.

1.3 Аналіз методів розв’язання задачі

Створення програмних засобів для втілення вебсистеми з метою аналізу та класифікації вимагає уважного розгляду найкращих стратегій вирішення цієї задачі. Існує різноманітність методів, кожен з яких має свої переваги та недоліки.

Методи машинного навчання: машинне навчання використовується для автоматичного навчання комп'ютерних систем виконувати завдання без явного програмування. Для аналізу та класифікації музичних композицій можуть бути використані наступні методи:

- Нейронні мережі: Використання глибоких нейронних мереж для аналізу аудіофайлів та класифікації їх за жанрами або іншими параметрами.
- Класифікатори: Використання алгоритмів класифікації, таких як Support Vector Machines (SVM) (рис 1.6) Машина опорних векторів (SVM) по суті знаходить найкращу лінію, яка розділяє дані у 2D. Ця лінія називається межею прийняття рішення. При наявності одномірних даних було б доцільно розділили їх за допомогою одного порогового значення. При наявності 3D-даних, результат SVM - це площина, яка розділяє два класи. Якщо дані мають більше трьох вимірів, межа рішення - це гіперплощина, яка є нічим іншим, як площиною у вищих вимірах. Навчити систему означає просто знайти лінію. Після навчення системи можна визначити, чи належить нова точка даних до синього або червоного класу, просто перевіривши, на якому боці лінії вона лежить.

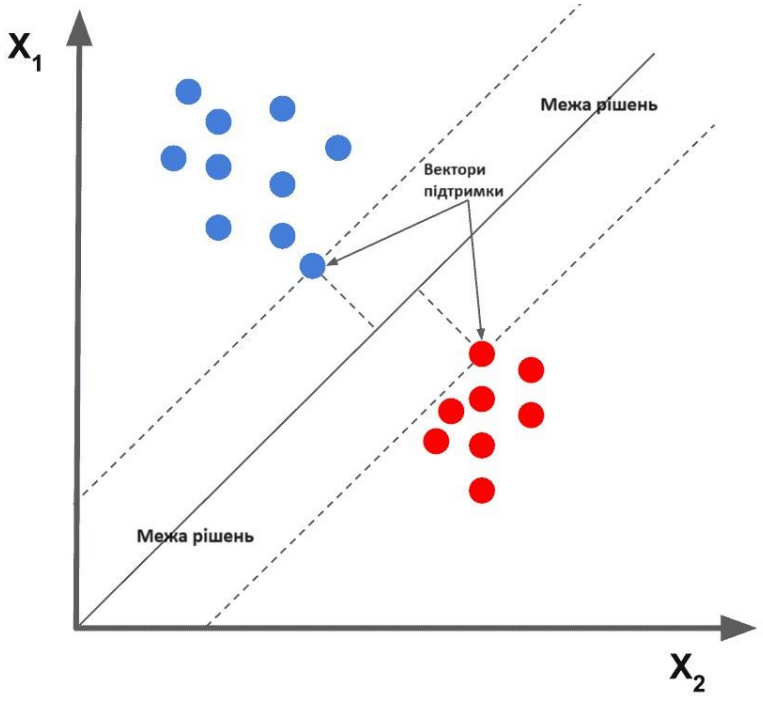


Рисунок 1.6 – Принцип роботи SVM

- Класифікатори: Random Forests. Принцип роботи Random Forests базується на створенні та об'єднанні декількох рішень дерев для прийняття рішення. Кожне дерево випадкового лісу навчається на випадковій підмножині навчальних даних та випадковій підмножині ознак. Під час класифікації нового зразка кожне дерево випадкового лісу вирішує, до якої категорії він належить. Потім, результати всіх дерев об'єднуються, зазвичай шляхом голосування, і категорія для нового зразка визначається на основі більшості голосів.
- Класифікатори: k-Nearest Neighbors (k-NN) (рис 1.7), для розподілу музичних композицій на різні категорії. У методі k-Nearest Neighbors (k-NN) категорія нового зразка визначається на основі категорій його k найближчих сусідів у просторі ознак. Під час класифікації нового зразка спочатку знаходяться k найближчих сусідів з навчального набору даних. Потім, на основі категорій цих сусідів приймається рішення щодо категорії нового зразка. Часто використовується голосування більшості серед цих сусідів, де кожен сусід має один голос, або можна використовувати ваговане голосування залежно від відстані до сусіда.

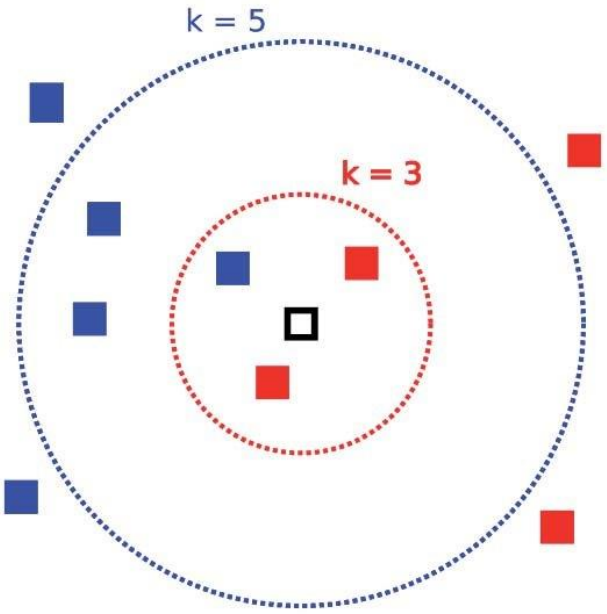


Рисунок 1.7 – Принцип роботи k-Nearest Neighbors

- Кластеризація: Групування музичних композицій на основі схожості їх характеристик без вказівки конкретних класів.

Аналіз аудіофайлів : аналіз аудіофайлів є важливим етапом у розв’язанні задачі автоматизованого аналізу та класифікації музичних композицій. Методи аналізу включають:

- Спектральний аналіз: Вимірювання частот і амплітуд звукових хвиль для визначення акустичних характеристик музичних композицій.
- Витягування ознак: Визначення важливих аудіофізичних характеристик, таких як ритм, темп, мелодія, для подальшої обробки та аналізу.

Обробка текстової інформації: текстова інформація про музичні композиції також може бути використана для їхньої класифікації. Методи обробки текстів включають:

- Аналіз метаданих: Використання інформації про виконавців, жанри, альбоми та інші метадані для допомоги в класифікації музичних композицій [7].
- Технології обробки природної мови: Використання методів обробки природної мови для аналізу тексту та коментарів до музичних композицій.

Інтеграція з базами даних: інтеграція з базами даних дозволяє зберігати та керувати музичними композиціями та їхніми характеристиками. Це включає:

- Бази даних аудіофайлів: Зберігання аудіофайлів у базі даних разом із їхніми характеристиками для подальшого доступу та обробки.
- Інтернет-джерела: Використання вебсервісів або API для отримання метаданих про музичні композиції з Інтернету для подальшого аналізу.

Розглянувши переваги та недоліки кожного методу, було прийнято рішення використовувати машинне навчання, зокрема модель згорткових нейронних мереж (CNN), для розв'язання задачі аналізу та класифікації музичних композицій.

Окрім цього, було прийнято рішення використовувати базу даних для зберігання завантажених музичних композицій. Це дозволить зручно керувати

інформацією про композиції, їхні аудіофайли та результати класифікації, забезпечуючи доступ до неї для подальшого аналізу та використання.

Також вирішено використовувати аналіз аудіофайлів для витягування ознак. Цей підхід дозволяє отримати важливі параметри з музичних композицій, які можуть бути використані для подальшої обробки та аналізу в рамках системи.

Комбінація цих методів може бути ефективною стратегією для створення програмних засобів для аналізу та класифікації музичних композицій.

1.4 Постановка задач бакалаврської дипломної роботи

Проаналізувавши переваги та недоліки існуючих вебсистем з використанням аналізу та класифікації музичних композицій було визначено наступні завдання, які необхідно виконати для вебсистеми:

- Дослідження різних методів класифікації аудіофайлів за жанрами з метою вибору найефективнішого підходу;
- Створення моделі машинного навчання для класифікації аудіофайлів на основі обраних методів;
- Розробка зручного та інтуїтивно зрозумілого вебінтерфейсу для взаємодії з користувачем;
- Створення бази даних для зберігання метаданих про завантажені аудіофайли;
- Проведення тестів для перевірки точності та надійності розробленого алгоритму класифікації.

1.5 Висновки

У першому розділі проаналізовано сучасний стан вебсистем, які використовують машинне навчання. Проведено порівняння кількох відомих платформ для аналізу та взаємодії з музикою, таких як Spotify; Apple Music; Pandora; Last.fm; Deezer.

Після оцінки було встановлено, що розглянуті платформи здобули значну популярність серед користувачів, завдяки їх зручності та доступності. Проте, під час аналізу переваг та недоліків було виявлено, що вони не повністю відповідають вимогам щодо класифікації музики. Було обґрунтовано доцільність розробки власного програмного рішення. Складено перелік завдань, необхідних для розробки власної вебсистеми з використанням машинного навчання.

2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ ВЕБСИСТЕМИ

2.1 Основні етапи розробки вебсистеми для аналізу і класифікації музичних композицій

1. Структура та аудиторія: На цьому етапі було ретельно проаналізовано аудиторію вебсистеми. Згідно з отриманими даними, система призначена для меломанів, музикантів та інших любителів музики, які шукають нову музику для прослуховування або аналізують музичні тренди. Щодо структури системи, вирішено було розробити вебдодаток з інтуїтивним і зручним інтерфейсом, який дозволить користувачам швидко та ефективно знаходити та класифікувати музичні композиції за різними категоріями.

2. Аналіз даних: На даному етапі було проведено аналіз різноманітних джерел даних про музичні композиції. Було зібрано метадані про композиції з онлайн-музичних платформ та інших джерел. Отримані дані включають назву, виконавця, жанр, рік випуску, а також характеристики звуку, такі як темп, тональність та енергія.

3. Обробка даних: Після збору даних вони пройшли через етап обробки. Це включало в себе очищення від помилкових записів, стандартизацію форматів, вилучення дублікатів та інженерію ознак. Цей процес був важливим для підготовки даних для подальшого використання в моделях класифікації.

4. Розробка моделей: На цьому етапі були розроблені моделі машинного навчання для класифікації музичних композицій. Використовуючи алгоритми, такі як CNN (рис. 2.1), були створені моделі, які класифікують композиції за різними критеріями, такими як жанр, настрій, темп тощо.

5. Інтеграція з вебінтерфейсом: Розроблені моделі були успішно інтегровані з вебінтерфейсом вебсистеми. Це дозволяє користувачам вибирати параметри для аналізу та класифікації музичних композицій через зручний та привабливий інтерфейс.

6. Тестування та налагодження: Після інтеграції система пройшла обширне тестування на функціональність, продуктивність та безпеку. Були внесені виправлення для забезпечення найвищої якості та надійності системи.

7. Випуск та підтримка: Після успішного завершення тестування система була випущена в експлуатацію. Забезпечено підтримку користувачам та вирішено всі проблеми, що можуть виникнути. Також забезпечена регулярна підтримка та оновлення системи.

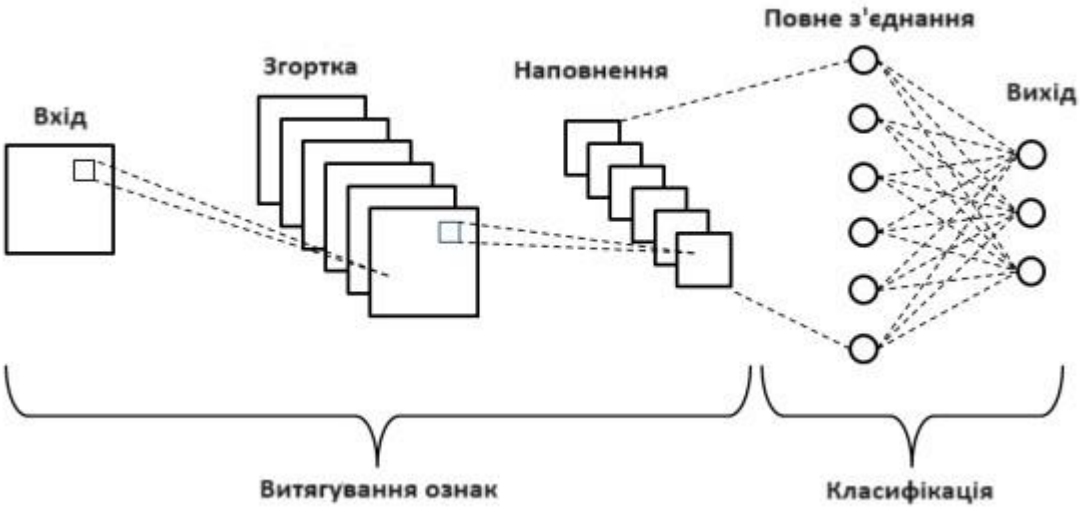


Рисунок 2.1 – Принцип роботи CNN

Створення вебсистеми для аналізу та класифікації музичних композицій виявляється доцільним з кількох причин. По-перше, у сучасному цифровому світі музика відіграє важливу роль у житті людей, що створює величезний попит на нові музичні композиції та можливість їхнього швидкого доступу. Однак, зростаюча кількість музичного контенту може призвести до затруднень у пошуку та відборі необхідних композицій. Створення вебсистеми, яка автоматизує процес аналізу та класифікації музичних композицій, дозволить користувачам ефективно знаходити музику, яка відповідає їхнім смакам та настрою.

Щодо можливості хостингу, існує безліч платформ та сервісів, які надають послуги хостингу вебдодатків. Провайдери хостингу, такі як AWS, Google Cloud Platform, Microsoft Azure, Heroku та інші, пропонують надійний та масштабований хостинг, що відповідає потребам різних проєктів.

2.2 Розробка архітектури CNN

Було вирішено використати згорткові нейронні мережі (CNN) для розпізнавання жанрів музичних композицій через їхню здатність ефективно обробляти двовимірні дані, такі як спектрограми аудіо сигналів. CNN мають відмінну здатність виявляти та інтерпретувати складні патерни в даних, що робить їх ідеальними для задач розпізнавання музичних жанрів. Процес розпізнавання жанрів музики складається з кількох основних етапів, включаючи підготовку даних, тренування моделі та класифікацію. Нижче описаний детальний процес, включаючи етапи витягування даних, їх збереження в JSON файл та подальшу обробку, з урахуванням архітектури моделі CNN.

Етапи процесу розпізнавання жанрів музичних композицій:

1. Підготовка даних:

Перший крок включає використання датасету GTZAN, який містить аудіофайли різних жанрів для тренування моделі. Датасет GTZAN включає 10 жанрів: блюз, класика, кантрі, диско, хіп-хоп, джаз, метал, поп, реггі та рок. Потім здійснюється попередня обробка аудіо, яка включає конвертацію аудіофайлів у монофонічний формат та нормалізацію гучності, а також розбиття аудіофайлів на фіксовані за тривалістю сегменти (наприклад, 30 секунд).

Наступним кроком є створення спектрограм. Це досягається шляхом перетворення аудіосигналів у спектрограми за допомогою короткочасного перетворення Фур'є (STFT). Спектрограма представляє частотний зміст аудіо у вигляді двовимірного зображення. Формула для STFT виглядає так:

$$X(m,k) = \sum_{n=0}^{N-1} x[n + mH] \cdot w[n] \cdot e^{-j2\pi kn/N} \quad (1)$$

- де x[n] - вхідний сигнал, w[n] - віконна функція, H - крок зсуву, N - довжина STFT, X(m,k) - спектрограма.

2. Тренування CNN моделі

Підготовка даних для тренування включає нормалізацію спектрограм та їх конвертацію у відповідний формат для подачі у CNN, зазвичай це тривимірні масиви, де розміри представляють висоту, ширину і кількість каналів (глибину). Архітектура моделі CNN включає кілька шарів:

- перший згортковий (conv) шар:
  - 32 фільтри розміром 3x3;
  - Функція активації ReLU;
  - Шар макспулінгу (MaxPooling) розміром 3x3 із кроком 2x2 і padding 'same';
  - Шар нормалізації (BatchNormalization).
- другий згортковий шар:
  - 32 фільтри розміром 3x3;
  - Функція активації ReLU;
  - Макспулінг розміром 3x3 із кроком 2x2 і padding 'same';
  - Шар нормалізації.
- третій згортковий шар:
  - 32 фільтри розміром 2x2;
  - Функція активації ReLU;
  - Макспулінг розміром 2x2 із кроком 2x2 і padding 'same';
  - Шар нормалізації.
- повнозв'язний шар:
  - Результат згорток сплющується у вектор (Flatten);
  - Повнозв'язний шар із 64 нейронами та функцією активації ReLU;
  - Регуляризація Dropout із ймовірністю 0.3.

- вихідний шар:
    - 10 нейронів та функція активації softmax, який видає ймовірності для кожного з 10 жанрів.

Компиляція моделі включає вибір функції втрат (loss function), наприклад, categorical cross-entropy:

$$L = -\sum_{i=1}^{N} y_i \log(\hat{y}_i) \quad (2)$$

- де N - кількість класів, уі - справжня мітка, у^і - передбачена ймовірність. Оптимізатор, Adam, використовується для оновлення ваг моделі:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \quad (3)$$

$$\vartheta_t = \beta_2 \vartheta_{t-1} + (1-\beta_2) g_t^2 \quad (4)$$

$$\widehat{m}_t = \frac{m_t}{1-\beta_1^t} \quad (5)$$

$$\hat{\vartheta}_t = \frac{\vartheta_t}{1-\beta_2^t} \quad (6)$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{\vartheta}_t + \epsilon}} \widehat{m}_t \quad (7)$$

-де gt - градієнт, η - швидкість навчання, β1,β2 - гіперпараметри, є - мале число для уникнення ділення на нуль.

Тренування моделі відбувається на тренувальних даних із використанням набору для валідації.

3. Збереження та завантаження моделі

Після тренування модель зберігається у файл (наприклад, у форматі HDF5 для Keras) для подальшого використання.

4. Витягування даних у JSON файл

Підготовка даних для збереження включає збереження метаданих і результатів класифікації в JSON файл. Це може включати інформацію про аудіофайли, спектрограми, передбачення моделі та ймовірності належності до кожного жанру. Структура JSON файлу:

{ "mapping": [], "labels": [], "mfcc": [] }

5. Класифікація нових аудіофайлів

Завантаження раніше збереженої моделі є першим кроком. Потім виконується попередня обробка нових аудіофайлів, що включає ті ж самі кроки, що і для тренувальних даних: конвертація у моно, нормалізація, розбиття на сегменти, створення спектрограм.

Далі спектрограми проганяються через модель для отримання ймовірностей належності до кожного жанру, після чого визначається жанр з найвищою ймовірністю. Результати класифікації зберігаються у JSON файл для подальшого аналізу та використання.

2.3 Аналіз вхідних даних системи

Для ефективної класифікації музичних композицій на основі їх аудіофайлів у цьому дослідженні використовується метод обробки сигналів. Дані для аналізу та класифікації були витягнуті з GTZAN датасету, який є широко використовуваним для таких цілей [8]. Візуалізація вигляду аудіофайлів у вигляді звукового вейвформату (рис 2.2) дозволяє зрозуміти їхню структуру та характеристики, такі як динаміка, тривалість, амплітуда тощо. Це може бути корисним для попереднього огляду та аналізу аудіоданих до їхньої подальшої обробки та використання у системі.

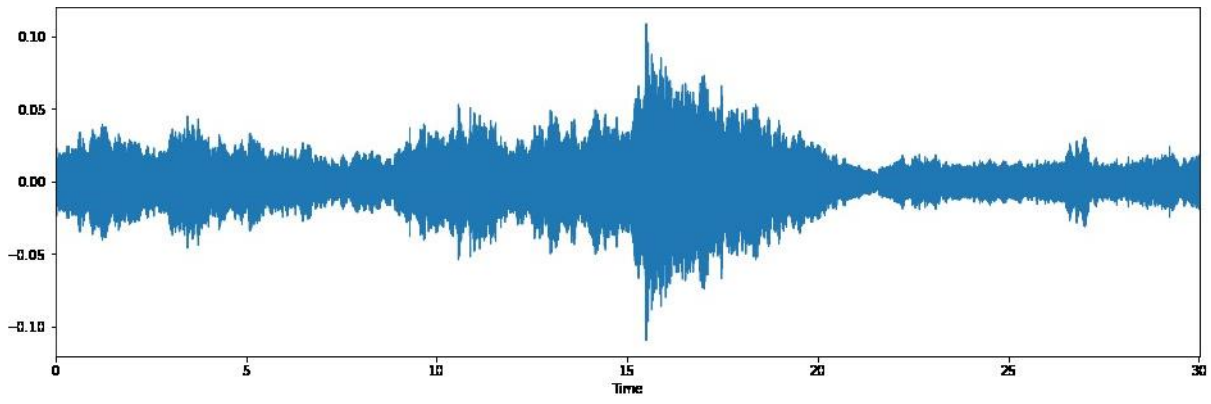


Рисунок 2.2 – Вигляд аудіофайлу у вигляді хвиль

Спектрограма відображає час на горизонтальній вісі, частоту на вертикальній вісі та інтенсивність звуку за допомогою кольору. Це дозволяє отримати візуальне уявлення про частотний склад звуку протягом часу. Застосування спектрограми у контексті аналізу вхідних даних системи дозволяє виявляти особливості аудіосигналу, такі як наявність пікових частот або структурних елементів у різних частинах композиції. Це може бути корисно для подальшого аналізу та класифікації музичних композицій з метою визначення їхнього жанру, настрою або інших характеристик.

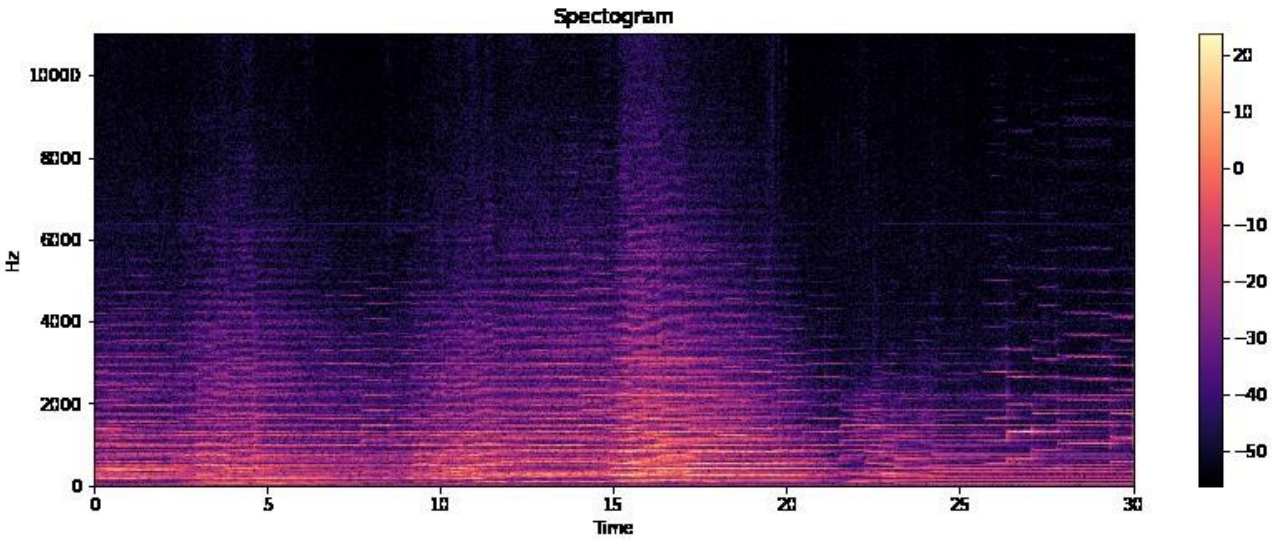


Рисунок 2.3 – Вигляд аудіофайлу у вигляді спектрограми

Для представлення аудіоданих було обрано формат Mel Frequency Cepstral Coefficients (MFCCs) (рис. 2.1) для тренування та використання моделі глибокого навчання. Кожен сегмент аудіофайлу розглядається окремо, що дозволяє отримати більш детальну інформацію про акустичні особливості звуку. Отримані MFCCs є важливими для розрізнення різних аспектів аудіосигналів, що є ключовим для правильної класифікації музичних жанрів. Таким чином, система використовує вхідні дані у вигляді MFCCs для тренування та класифікації музичних жанрів за допомогою моделі глибокого навчання, безпосередньо використовуючи їх як характеристики аудіосигналів [9].

За допомогою CNN моделі можна розпізнавати жанри музики на основі аудіофайлів у форматі MP3 або WAV. Після тренування моделі потрібно підв'язати її до вебзастосунку, який буде приймати аудіофайли вказаних форматів і використовувати натреновану модель для визначення жанру аудіофайлу.

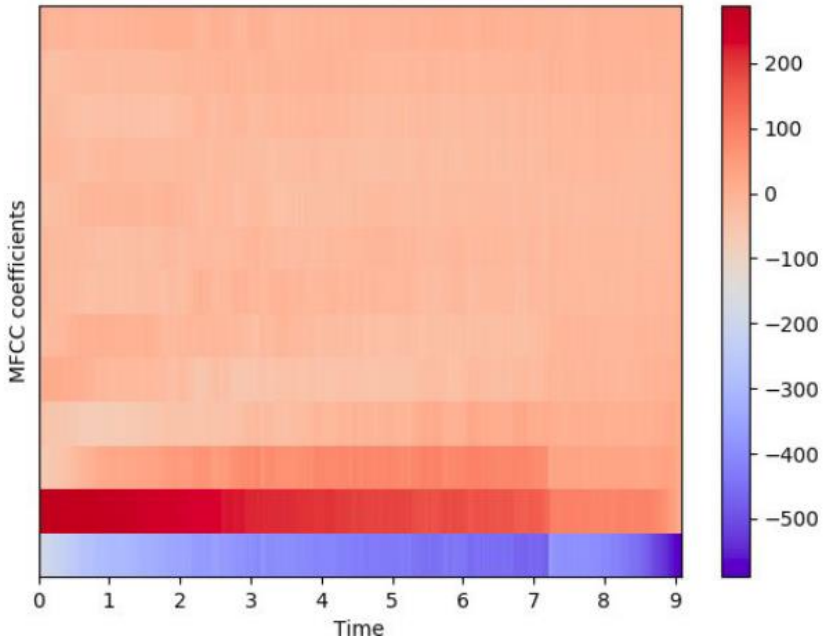


Рисунок 2.4 – Спектрограма MFFCs

Після успішного натренування CNN моделі для класифікації музичних композицій, потрібно розробити такі частини вебсистеми. Ця система може складатися з наступних програмних засобів:

Вебчастина, що дозволяє користувачам взаємодіяти з системою. Вона складається з сторінки для завантаження аудіофайлів, перегляду результатів класифікації, перегляду попередньо завантажених файлів.

Модуль обробки аудіофайлів: Модуль, який відповідає за обробку завантажених аудіофайлів форматів WAV та MP3. Він включатиме функції для конвертації аудіофайлів у внутрішній формат системи, відтворення та аналізу аудіоданих за допомогою натренованої моделі.

Серверна частина: Серверна архітектура, яка забезпечує взаємодію між вебінтерфейсом, базою даних та модулем обробки аудіофайлів. Вона відповідає за обробку запитів користувачів, зберігання та витягування даних з бази даних, а також виконання обчислень за допомогою натренованої моделі [10].

Комплексна розробка програмних засобів для системи аналізу класифікації музичних композицій передбачає ряд етапів, які виконуються послідовно з метою створення функціонального та ефективного рішення. Після кожного етапу необхідне тестування для підтвердження правильності реалізації функцій та виявлення можливих помилок.

2.4 Розробка інтерфейсу програмного додатку

Інтерфейс (або GUI) — це скорочення від "графічний інтерфейс користувача" (Graphical User Interface). Це спосіб взаємодії користувача з комп'ютерною програмою або пристроєм, який використовує графічні елементи, такі як вікна, кнопки, піктограми та меню, для візуального відображення інформації та можливостей програми. Інтерфейс користувача відображається на екрані монітора або дисплея і дозволяє користувачу взаємодіяти з програмою або пристроєм шляхом клацання мишею, торкання на сенсорному екрані, введення тексту за допомогою клавіатури тощо.

Основні складові GUI включають в себе:

1. Вікна: Візуальні області на екрані, які містять різні елементи інтерфейсу.
2. Елементи керування: Такі як кнопки, поля введення, випадаючі списки, перемикачі та інші елементи, які дозволяють користувачу взаємодіяти з програмою.
3. Меню: Списки опцій та команд, доступних для користувача.
4. Піктограми: Малюнки або символи, що представляють програми, файли або функції.

5. Текстові елементи: Текстові поля, мітки, рядки статусу та інші елементи, що містять текстову інформацію.

GUI робить взаємодію з програмами та пристроями більш інтуїтивно зрозумілою для користувача, оскільки вона дозволяє використовувати візуальні елементи для взаємодії з системою, натомість клавіатура або командний рядок. Вона широко використовується у багатьох сучасних програмах та операційних системах, що робить їх більш доступними та зручними для користувачів. Це тип інтерфейсу, який дозволяє користувачам взаємодіяти з комп'ютерними програмами через графічні елементи, такі як вікна, іконки, кнопки та меню. На відміну від текстових інтерфейсів, де користувач вводить текстові команди, GUI забезпечує більш інтуїтивно зрозумілий та зручний спосіб взаємодії.

Під час розробки інтерфейсу вебсистеми, приділялася особлива увага зрозумілості та зручності його використання для користувача. Основним кольором інтерфейсу було обрано блакитно-рожевий, оскільки він асоціюється зі спокоєм та гармонією, створюючи приємну атмосферу під час взаємодії з вебсистемою. Навігація була розроблена інтуїтивно зрозумілою, з чітко позначеними кнопками та меню, для швидкого доступу користувачів до необхідних функцій та ресурсів. Усі елементи інтерфейсу були розміщені в логічному порядку, що спрощує процес взаємодії з системою та забезпечує комфортний досвід користувача.

Головний екран інтерфейсу вебсистеми (рис. 2.1) розділений на чотири основні секції: "Home", "Classify", "Audios" та "About". Кожен розділ має свою унікальну функціональність та призначення. На головному екрані відображено невеликий GIF-файл, який ілюструє звукову доріжку, створюючи атмосферу взаємодії з музикою та аудіофайлами. Відмінність між розділами дозволяє користувачам швидко та зручно переміщатися між різними функціями системи та знаходити потрібну інформацію без зайвих зусиль.

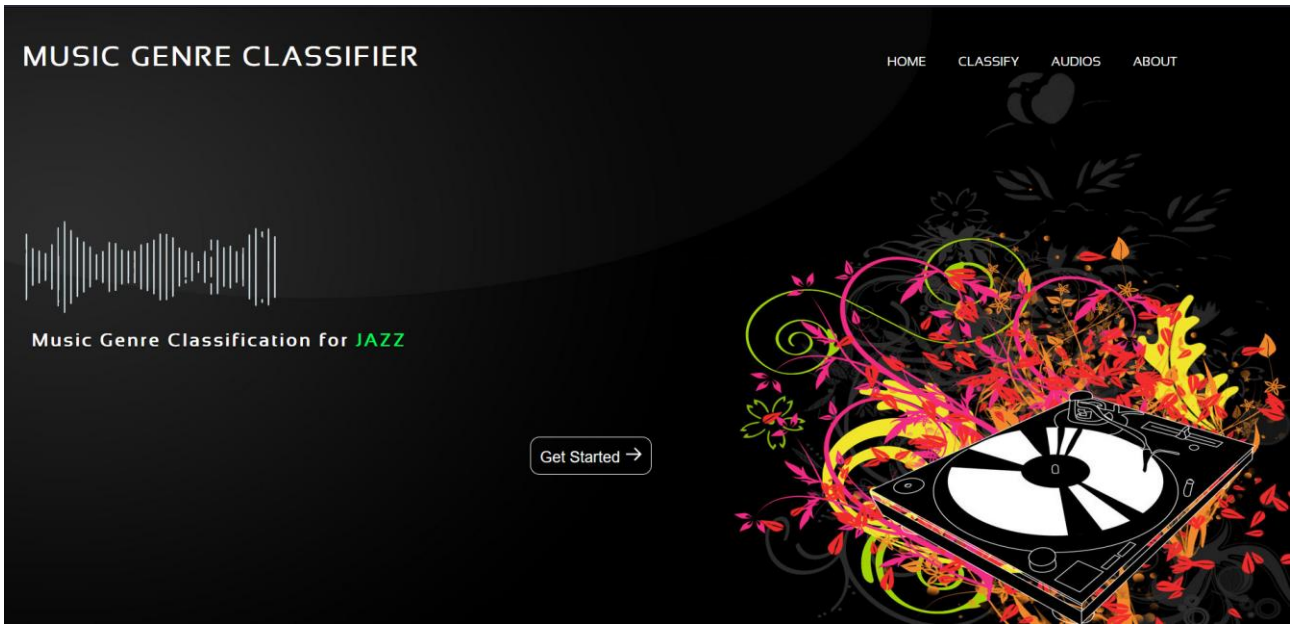


Рисунок 2.5 – Екран головної сторінки

У розділі "Classify" доступні два поля для взаємодії з музичними композиціями. Перше поле призначене для прогнозування жанру музики на основі аудіофайлу (рис. 2.2). Користувач може завантажити аудіофайл у форматі MP3 або WAV та отримати прогноз щодо жанру композиції. Друге поле призначене для прослуховування музичної композиції, що була завантажена користувачем (рис. 2.3). Воно має регулятор гучності, який дозволяє налаштувати рівень звуку, а також кнопку перемотки для переміщення по аудіодоріжці. У випадку, якщо користувач намагається завантажити аудіофайл(и) у невідповідному форматі або більше ніж один файл, система автоматично перенаправить користувача на головний екран. Організація вмісту вебсистеми (вебсайту або вебдодатку) є ключовим аспектом веброзробки, оскільки вона впливає на зручність користування, ефективність навігації та загальний досвід користувачів. Правильна організація вмісту допомагає користувачам швидко знаходити потрібну інформацію та виконує бізнес-завдання, такі як збільшення конверсії або залучення аудиторії.

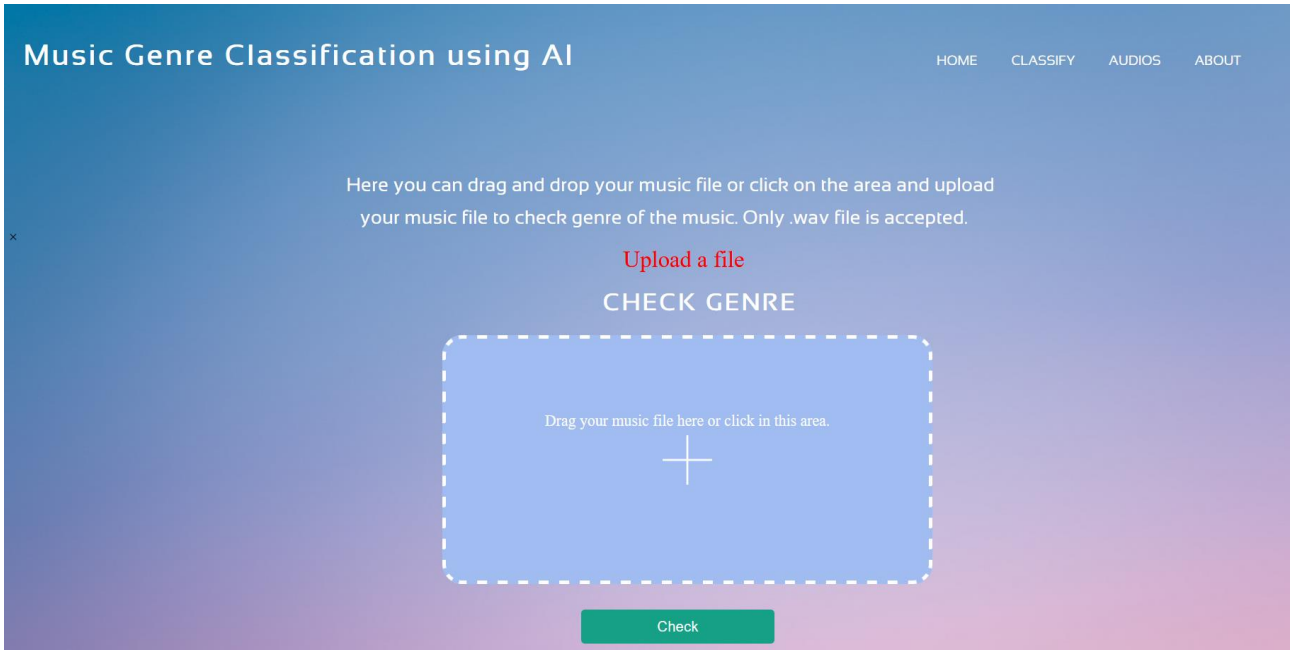


Рисунок 2.6 – Поле для класифікації

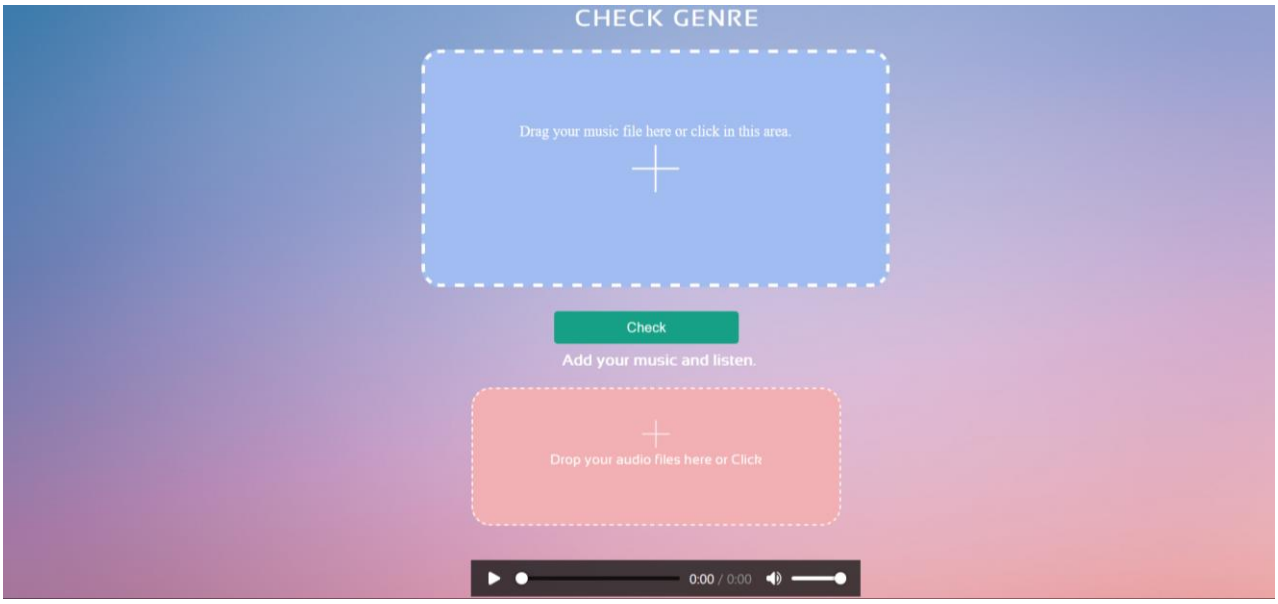


Рисунок 2.7 – Поле для відтворення файлу

У розділі "Audios" (рис. 2.4) користувач зможе переглянути інформацію про аудіофайли, які він раніше завантажував до системи. Цей розділ містить список файлів, де кожен елемент містить інформацію про окремий аудіофайл, таку як назва файлу та інші параметри.

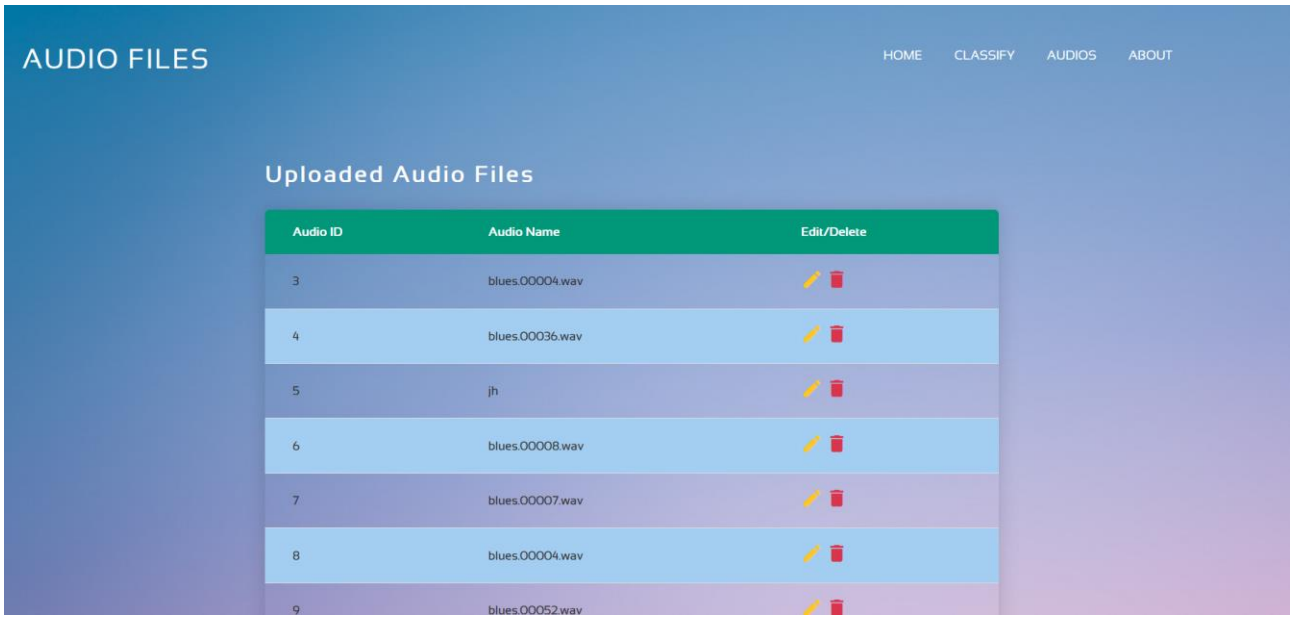


Рисунок 2.8 – Розділ "Audios"

Розділ "About" (рис. 2.5) вебсистеми надає описову інформацію про саму вебсистему, його призначення та основні можливості.

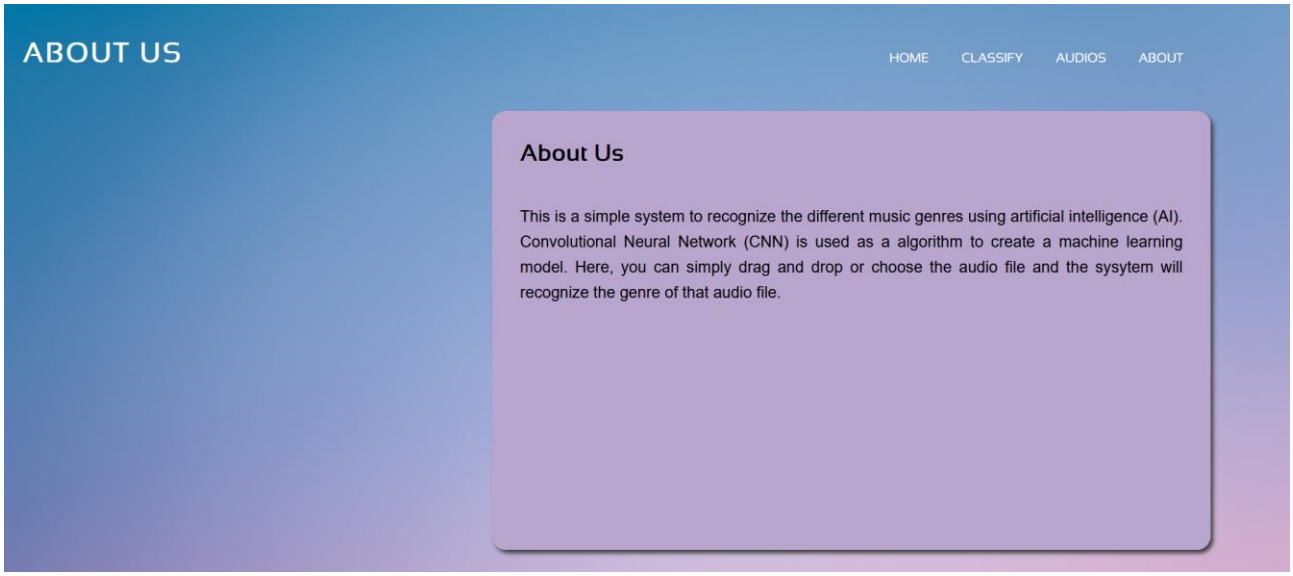


Рисунок 2.9 – Розділ "About"

Після завершення процесу класифікації музичної композиції, система виводить спливаюче вікно (рис. 2.6), яке інформує користувача про ймовірний жанр треку. Вікно відображає один з десяти можливих жанрів.

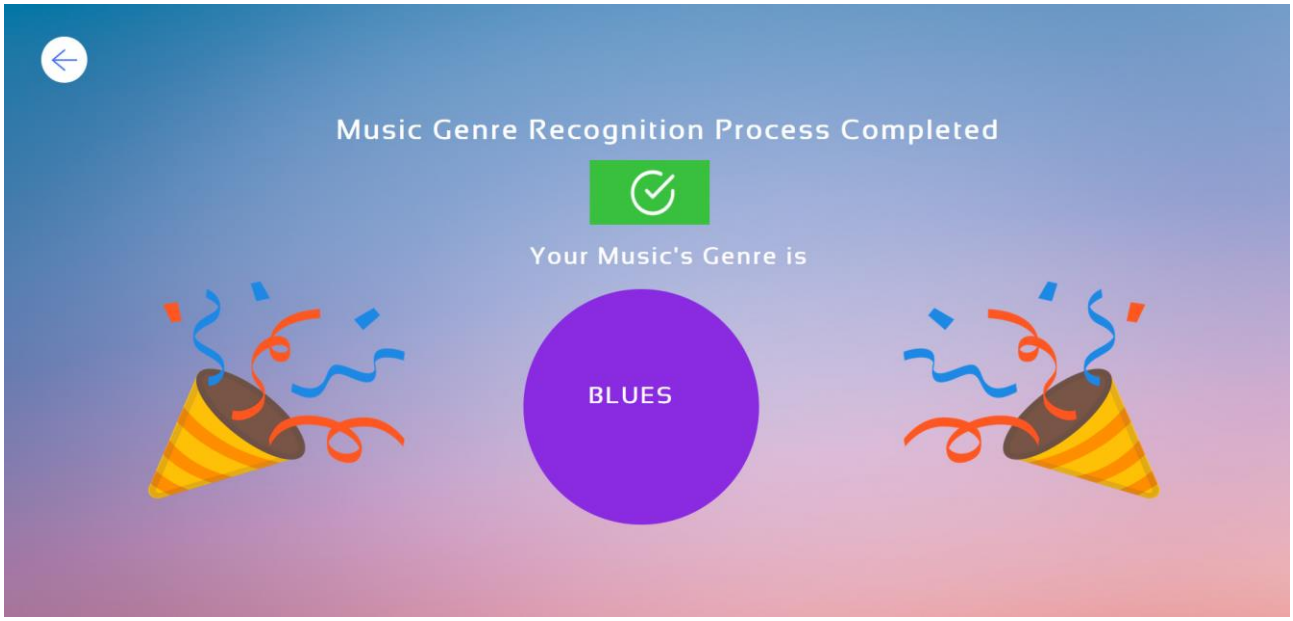


Рисунок 2.10 – Вікно з результатом класифікації

Інтерфейс вебсистеми для класифікації музичних жанрів був розроблений з використанням фреймворку Django. Django забезпечує потужну підтримку для розробки вебдодатків, дозволяючи інтеграцію з базами даних, розробку зручного для користувача інтерфейсу, і впровадження необхідних безпекових заходів [11].

2.5 Розробка програмних засобів системи

Для кращого розуміння роботи програмних модулів вебсистеми було вирішено розробити модель роботи системи на прикладі UML діаграми діяльності (рис 2.7). Така діаграма допомагає візуалізувати процеси, які відбуваються в системі, виявляти ключові компоненти та їхні взаємозв'язки [12]. Основна мета діаграми діяльності - візуалізувати послідовність дій, які виконує система або користувач для досягнення певної мети. Це може бути використано для моделювання бізнес-процесів, опису робочих потоків, визначення взаємодії між різними системами або просто для опису послідовності дій в програмі. Діаграми діяльності складаються з активностей (або дій), які представлені у

вигляді прямокутників, та стрілок, які вказують на послідовність виконання дій. Кожна активність має назву і може мати додаткові атрибути, такі як умови або вхідні та вихідні дані. Модель роботи системи та використання UML (Unified Modeling Language) відіграють ключову роль у розробці програмного забезпечення, допомагаючи зрозуміти та документувати структуру і поведінку системи.

Модель роботи системи описує, як різні компоненти системи взаємодіють між собою та з користувачами для досягнення певної функціональності. Це може включати опис бізнес-процесів, потоки даних, взаємодії користувачів з системою та інші аспекти, які показують, як система повинна працювати.

Одним із основних інструментів для моделювання систем є UML, який забезпечує набір діаграм для різних аспектів системи. UML є стандартом для візуалізації, специфікації, конструювання програмних систем.

Процес використання вебсистеми для аналізу та класифікації музичних композицій починається з перевірки інтернет-з'єднання користувача, що є критично важливим для функціонування онлайн-платформи. Після успішної валідації з'єднання, користувач входить у вебсистему. На цьому етапі система перевіряє, чи відповідає завантажений файл вимогам: один файл у дозволеному форматі і не перевищує встановленого розміру. У разі виявлення помилок файл відхиляється, а користувач отримує відповідне повідомлення із запропонованими кроками для виправлення. Якщо файл відповідає умовам, він обробляється системою, яка за допомогою алгоритмів глибокого навчання аналізує аудіосигнали, витягнуті з MFCC, для передбачення музичного жанру.

Після аналізу система відображає результати у вікні інтерфейсу, де користувач може побачити вірогідний жанр музичної композиції серед можливих десяти варіантів.

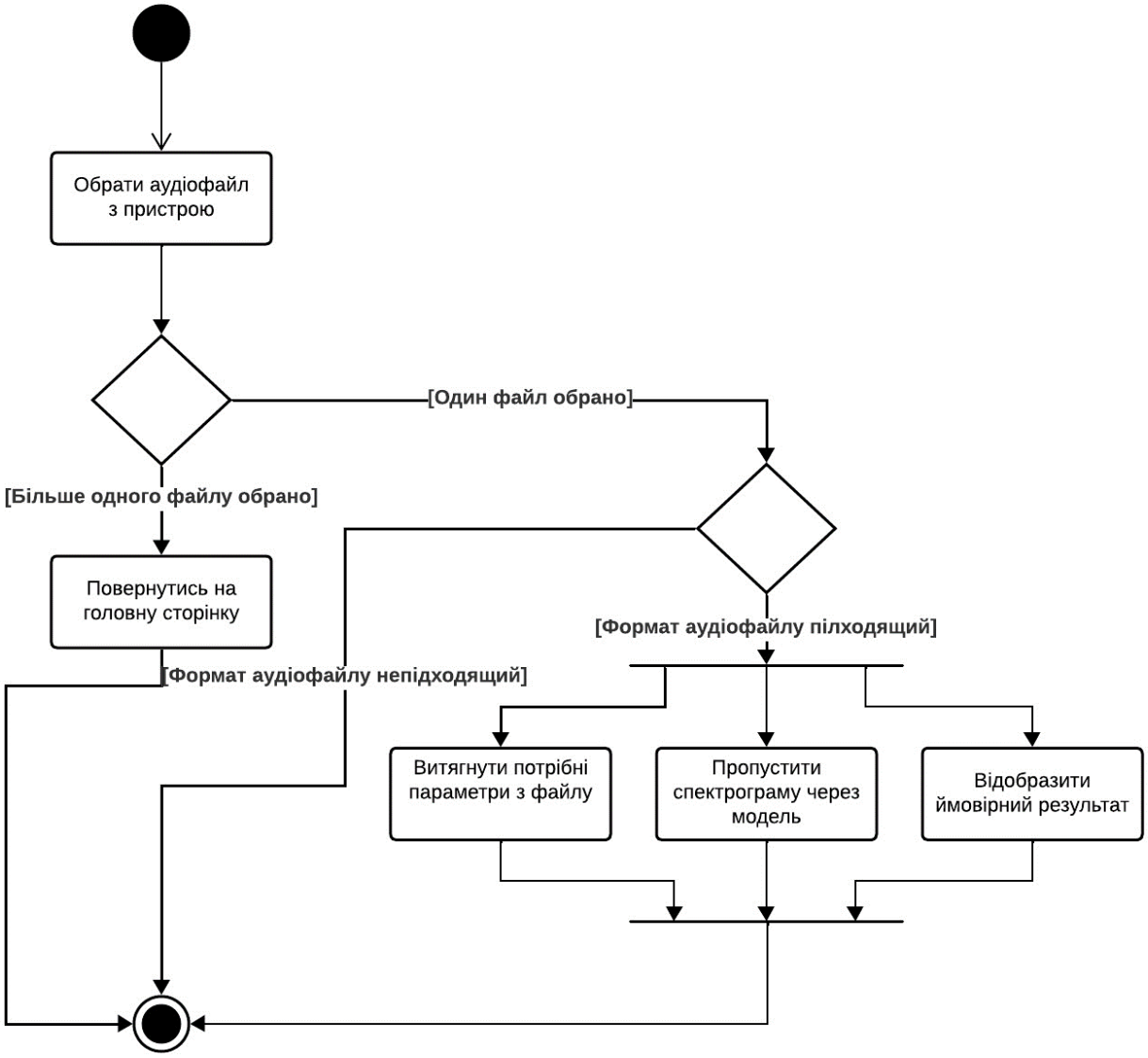


Рисунок 2.11 – Діаграма діяльності вебсистеми

2.6 Розробка моделей та алгоритмів роботи системи

Алгоритми визначають ефективність роботи програм. Добре розроблені алгоритми можуть значно зменшити час виконання програм, зменшити використання ресурсів системи та зробити програму швидкою та ефективною. : Розробка ефективних алгоритмів для захисту даних, шифрування та безпеки є критично важливою у сучасному цифровому світі. Грамотні алгоритми забезпечують захист конфіденційності, цілісності та доступності даних.Розробка

алгоритмів для системи аналізу та класифікації музичних композицій у контексті машинного навчання передбачає такі кроки:

1. Підготовка даних: Починається зі збору даних про аудіофайли музичних композицій. Ці дані можуть бути зібрані з відомих джерел або власноручно. Після цього дані обробляються для вилучення важливих аудіохарактеристик, таких як Mel Frequency Cepstral Coefficients (MFCCs), які використовуються як ознаки для подальшого аналізу.

2. Розділення даних на тренувальний та тестовий набори: Отримані дані розділяються на тренувальний та тестовий набори для оцінки ефективності моделі. Зазвичай використовується метод хресної перевірки, щоб гарантувати об'єктивність результатів.

3. Вибір моделі та параметрів: Вибирається архітектура моделі машинного навчання, наприклад, згорткова нейронна мережа (CNN), яка показала високу ефективність у задачах аудіоаналізу.

4. Навчання моделі: Тренування моделі проводиться на тренувальному наборі даних, де модель навчається розпізнавати патерни та залежності між аудіохарактеристиками та музичними жанрами.

5. Оцінка моделі та налаштування параметрів: Після завершення тренування модель оцінюється за допомогою тестового набору даних для визначення її точності та ефективності. За потреби налаштовуються параметри моделі для поліпшення її результатів.

6. Застосування моделі для класифікації: Отримавши навчену модель, її треба використовувати для класифікації нових аудіофайлів на основі їх аудіохарактеристик, таких як MFCCs. Модель здатна розпізнавати музичні жанри або інші аспекти музичних композицій з високою точністю.

Також для розробки вебсистеми з аналізу і класифікації необхідно розробити загальний алгоритм, згідно якого буде працювати система (рис. 2.8).

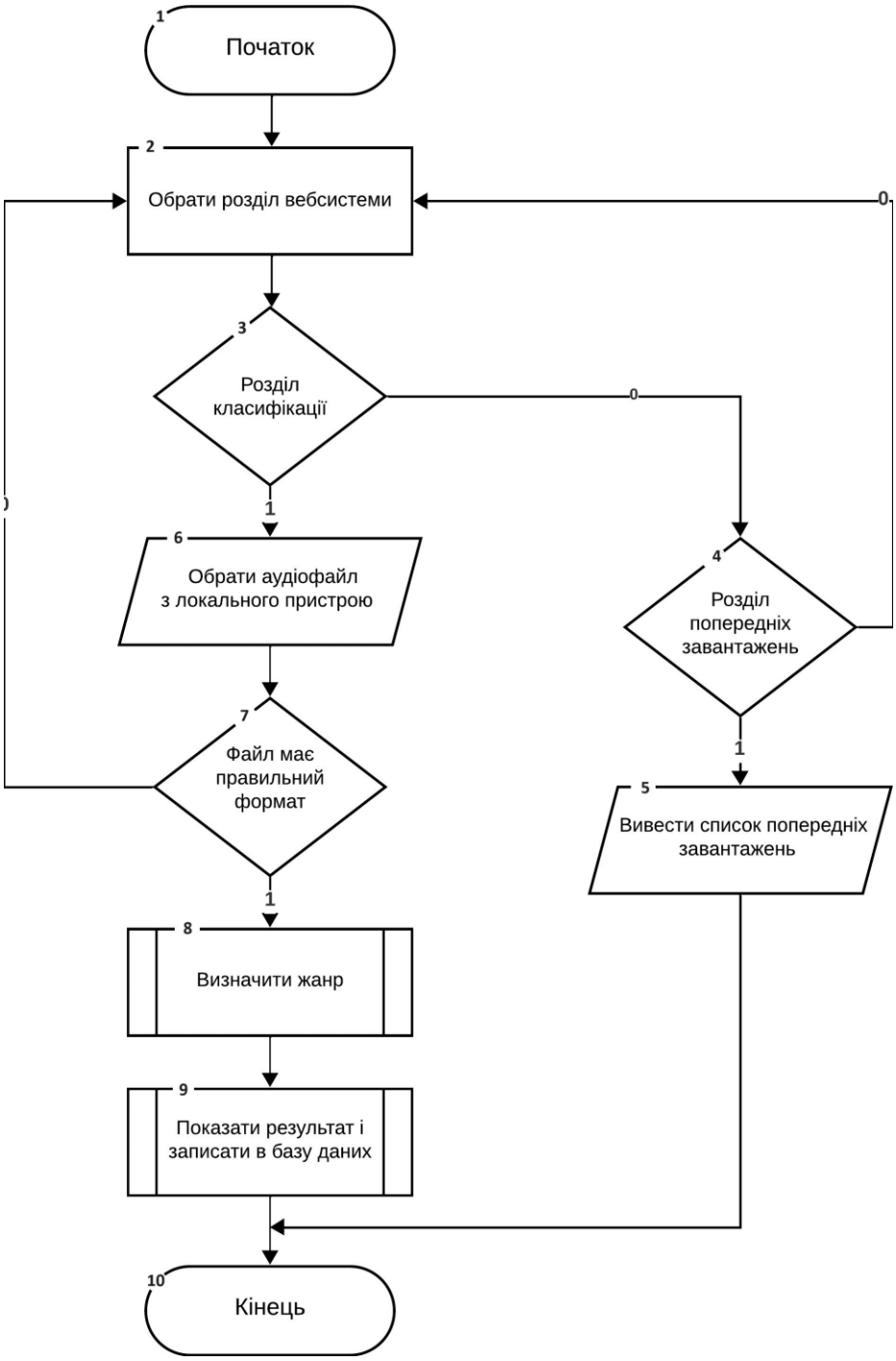


Рисунок 2.12 – Головний екран мобільної платформи

Використання UML сприяє збільшенню ясності і точності в описі вимог, проєктуванні архітектури та реалізації програмних систем. Вона є важливим інструментом для комунікації між учасниками проєкту та дозволяє покращити якість та швидкість розробки програмного забезпечення [13].

2.7 Висновки

У другому розділі було проведено аналіз вхідних даних програмного продукту та вивчено можливості покращення продуктивності вебсистеми шляхом підвищення точності класифікації. Розроблено основний алгоритм роботи мобільної платформи та алгоритм для машинного навчання. Додатково була створена модель роботи програмного продукту з використанням UML діаграм.

3 РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ ВЕБСИСТЕМИ ДЛЯ КЛАСИФІКАЦІЇ ТА АНАЛІЗУ МУЗИЧНИХ КОМПОЗИЦІЙ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення

При створенні програмних засобів для системи, яка використовує інструменти машинного навчання, критично важливо обирати технології з розсудливістю. Python - це високорівнева мова програмування, яка володіє широким спектром застосувань у різних галузях, включаючи веброзробку, аналіз даних, штучний інтелект, наукові обчислення та багато іншого. Ця мова відома своєю простотою синтаксису та читабельністю коду, що робить її дуже популярною серед програмістів.

У веброзробці, Python використовується для створення різноманітних вебдодатків та сервісів. Одним з найпопулярніших фреймворків для веброзробки на Python є Django. Django забезпечує швидку та ефективну розробку вебдодатків завдяки вбудованим функціональним можливостям, таким як адміністративний інтерфейс, система маршрутизації, аутентифікація користувачів та багато іншого. Завдяки Django, розробники можуть швидко створювати високоякісні вебдодатки з мінімальними зусиллями [14].

У контексті вебсистеми для аналізу і класифікації музичних композицій, Python може бути використаний для побудови серверної частини додатку за допомогою Django. За допомогою Django можна реалізувати вебінтерфейс для користувачів, взаємодію з базою даних, обробку запитів і відповідей, а також авторизацію та аутентифікацію користувачів. Python також може бути використаний для розробки бізнес-логіки додатку та інтеграції з бібліотеками для обробки аудіоданих та машинного навчання, такими як librosa та TensorFlow. В цілому, Python є потужним і зручним засобом для створення вебсистеми для

аналізу і класифікації музичних композицій завдяки своїм можливостям та багатофункціональності [15].

Librosa - це бібліотека для аналізу та обробки аудіоданих у Python. Вона надає різноманітні інструменти для роботи з аудіосигналами, включаючи завантаження аудіофайлів, обчислення акустичних характеристик, створення спектрограм та витягування різноманітних ознак, таких як MFCC (Mel Frequency Cepstral Coefficients). У контексті дипломної роботи, librosa може використовуватися для підготовки аудіоданих до подальшого аналізу та класифікації, включаючи витягування MFCC, які є важливими характеристиками для класифікації музичних композицій.

TensorFlow - це відкрита платформа машинного навчання розроблена компанією Google. Вона надає широкий спектр інструментів для розробки та тренування моделей глибокого навчання [16]. У контексті дипломної роботи, TensorFlow може бути використаний для побудови та тренування моделей машинного навчання, які класифікують музичні композиції за їх звуковими характеристиками. Наприклад, можна використовувати згорткові нейронні мережі (CNN) для автоматичного класифікації музичних жанрів на основі MFCC, які були витягнуті за допомогою librosa. TensorFlow надає інструменти для побудови, тренування та оцінки моделей машинного навчання, що робить його важливим компонентом у розробці вебсистеми для аналізу і класифікації музичних композицій.

Комбінація Python, Django, Librosa та TensorFlow створює потужні можливості для реалізації вебсистеми для аналізу та класифікації музичних композицій. Python, з його широким спектром бібліотек, є ідеальним вибором для розробки програмного забезпечення будь-якого рівня складності. Django надає зручні інструменти для створення високоякісних вебдодатків з широким спектром можливостей, включаючи автентифікацію користувачів і роботу з базами даних. Бібліотека Librosa дозволяє легко працювати з аудіофайлами та витягувати з них корисну інформацію, таку як Mel Frequency Cepstral Coefficients

(MFCCs), які можуть бути використані для подальшого аналізу. TensorFlow, з своїм широким спектром алгоритмів та моделей глибокого навчання, надає потужні інструменти для розробки та навчання моделей класифікації на основі аудіофункцій, отриманих за допомогою Librosa.

3.2 Вибір середовища розробки

Інтегроване середовище розробки (Integrated Development Environment або IDE) є незамінним інструментом для програмістів, що дозволяє їм зручно розробляти програмне забезпечення. На ринку існує велика кількість IDE, проте в даному випадку розглянемо Visual Studio, Code::Blocks та Qt Creator як найбільш популярні серед програмістів, які працюють з мовою програмування C++.

Visual Studio: Розроблене корпорацією Microsoft, Visual Studio (рис. 3.1) є потужним інструментом для розробки різноманітних програм, включаючи консольні та мобільні застосунки, графічні інтерфейси користувача, вебслужби та хмарні рішення. Відоме своєю розширеною функціональністю, Visual Studio підтримує не лише мову C++, а й інші мови, такі як C#, Python, Visual Basic і Javascript. Його можливості включають автозаповнення коду, рефакторинг, збирання та компіляцію програм, функціонал для тестування та налагодження коду. Visual Studio надає розробникам широкий набір інструментів для продуктивної роботи над проєктами будь-якої складності.

Переваги:

- Розширена функціональність, що дозволяє розробляти різноманітні програми.
- Підтримка не лише мови C++, а й інших популярних мов програмування.
- Автозаповнення коду та інші інструменти для підвищення продуктивності роботи.
- Інтеграція з іншими сервісами Microsoft та хмарними рішеннями.

Недоліки:

- Великий обсяг ресурсів, що може призводити до великої витрати пам'яті та часу на запуск.
- Системні вимоги можуть бути високими для менш потужних комп'ютерів.
- Для користувачів, що не працюють в екосистемі Microsoft, можуть виникати проблеми з сумісністю.

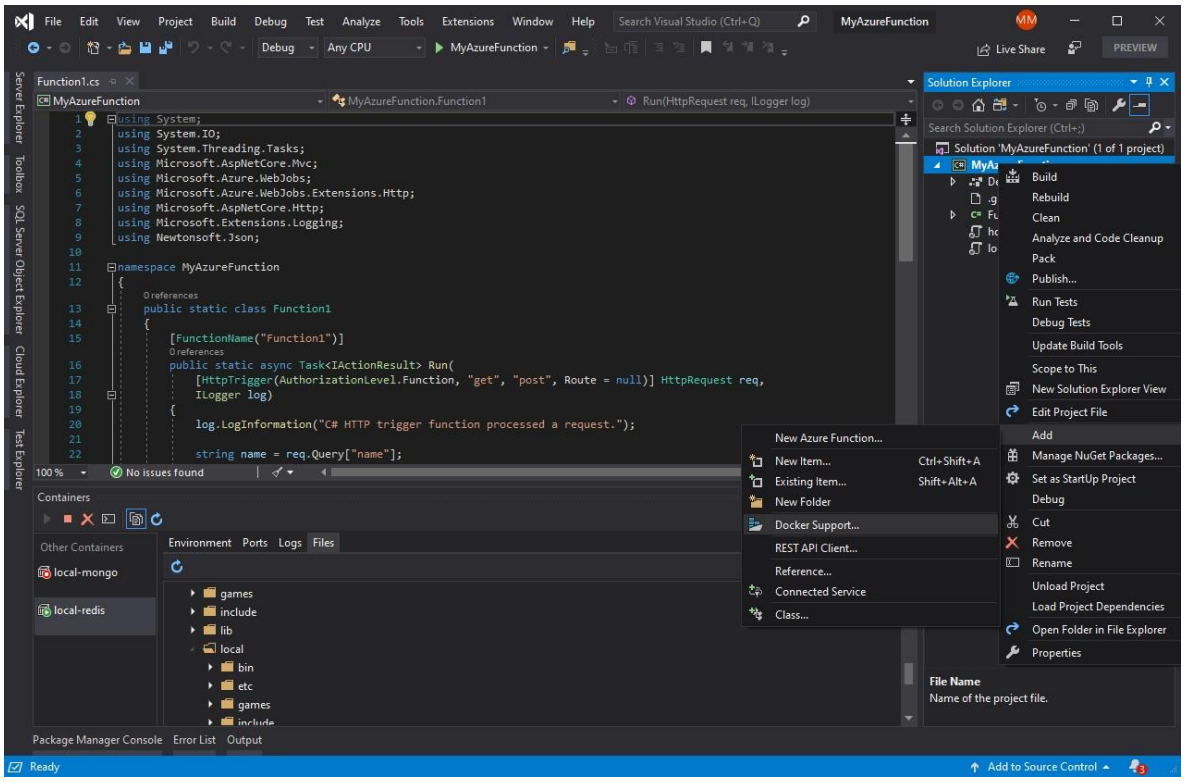


Рисунок 3.1 – Інтерфейс Visual Studio

Code::Blocks: Це безкоштовне інтегроване середовище розробки, яке особливо популярне серед програмістів, що працюють з мовою C++ (рис. 3.2). Code::Blocks пропонує широкий набір функцій, включаючи автозаповнення коду, підтримку рефакторингу, можливість відлагодження та збирання проєктів. Він має зручний інтерфейс та добре підходить як для початківців, так і для досвідчених програмістів.

Переваги:

- Безкоштовний та відкритий програмний продукт.

- Широкий набір функцій для розробки програм мовою C++.
- Зручний інтерфейс, що робить його привабливим як для початківців, так і для досвідчених програмістів.

Недоліки:

- Обмежена підтримка інших мов програмування, крім C++.
- Менша кількість інтегрованих сервісів порівняно з Visual Studio.
- Може бути менш потужним у порівнянні з іншими IDE.

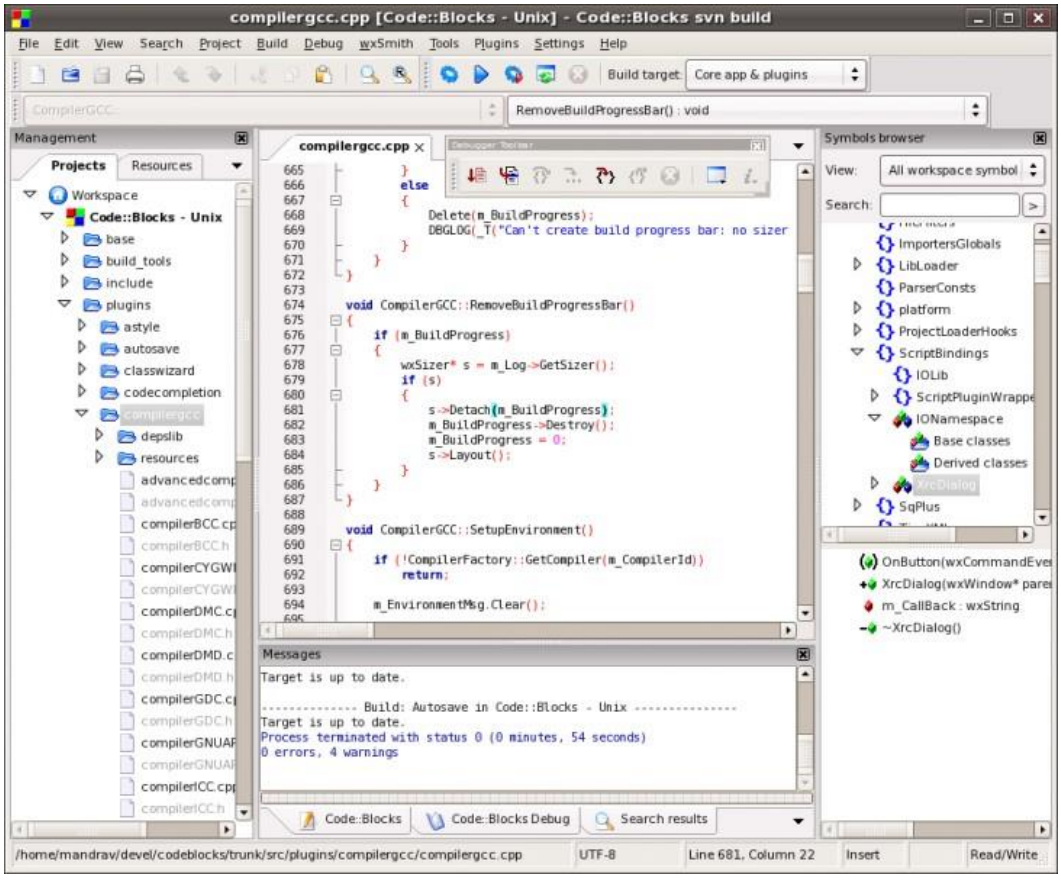


Рисунок 3.2 – Інтерфейс Code::Blocks

Qt Creator: Розроблений для роботи з фреймворком Qt, Qt Creator є потужним інструментом для розробки крос-платформенних додатків. Він має інтуїтивний інтерфейс та включає в себе всі необхідні інструменти для розробки програм, включаючи редактор коду з виділенням синтаксису, підтримку автозаповнення, налагодження та компіляцію. Qt Creator спеціально орієнтований на розробку

програм на мові C++, зокрема, для створення додатків з використанням фреймворку Qt.

Переваги:

- Орієнтований на розробку крос-платформових додатків з використанням фреймворку Qt.
- Інтуїтивний інтерфейс, що робить його зручним для використання.
- Інтегровані інструменти для розробки програм мовою C++.

Недоліки:

- Обмежена підтримка інших мов програмування, окрім C++.
- Може бути менш універсальним у порівнянні з більш загальнопризначеними IDE, як Visual Studio.
- Менша кількість розширених функцій порівняно з іншими IDE.

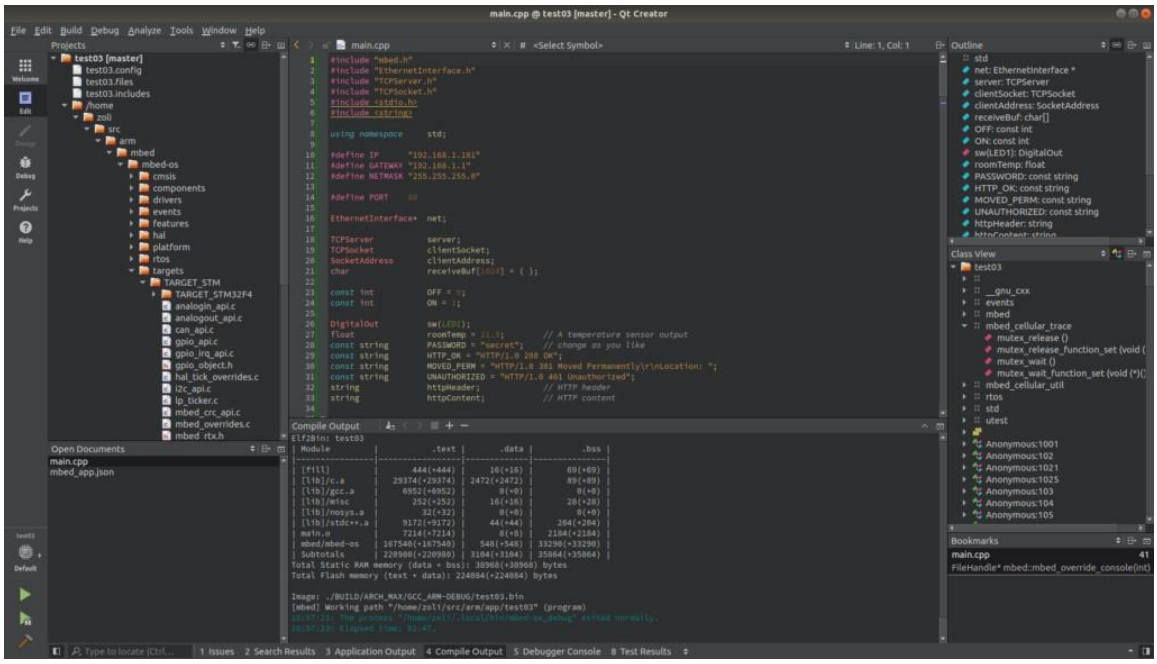


Рисунок 3.3 – Інтерфейс QtCreator

PyCharm - це інтегроване середовище розробки (IDE), призначене для програмістів, які працюють з мовою програмування Python. Це потужний

інструмент, розроблений компанією JetBrains, який надає широкий спектр функціональностей для підвищення продуктивності розробки PHP-програм.

Переваги:

- Спеціалізований IDE для роботи з мовою Python, що надає розширені можливості для розробки вебдодатків.
- Підтримка PHP та інших вебтехнологій, таких як HTML, CSS, JavaScript тощо.
- Вбудовані інструменти для рефакторингу, автоматичного завершення коду, налагодження та тестування Python-проєктів.
- Широкий вибір плагінів та можливість розширення функціональності за допомогою них.

Недоліки:

- Висока ціна ліцензії порівняно з іншими IDE.
- Системні вимоги можуть бути високими, особливо при великих проєктах.
- Менша підтримка інших мов програмування порівняно з загальнопризначеними IDE.

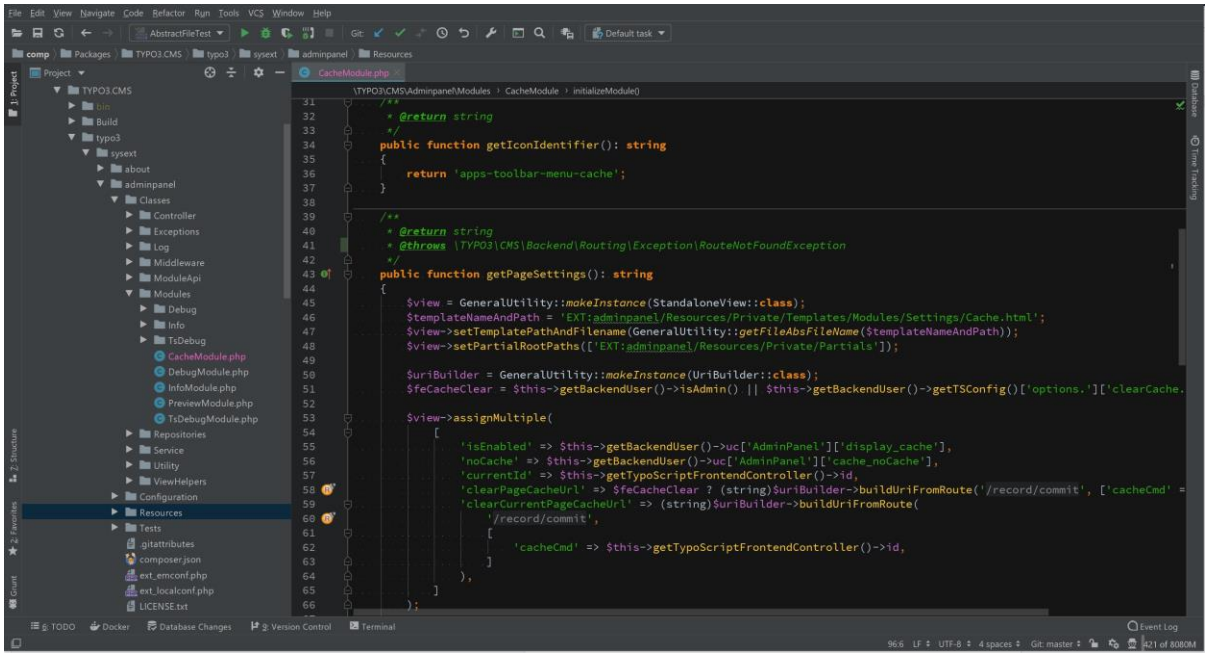


Рисунок 3.4 – Інтерфейс PyCharm

Розглянувши три популярні IDE, було сформовано стислий порівняльний аналіз наведений у таблиці 3.1.

Таблиця 3.1 – Порівняння середовищ розробки

Критерій	Visual Studio	Code::Blocks	Qt Creator	PyCharm
Сумісність з різними ОС	1	1	1	1
Можливість використання розширень	1	1	1	1
Дебагер	1	1	1	1
Фреймворки для розробки GUI	0	0	1	1
Чіткість сигналу	1	0	0	1
Сума	4	3	4	5

Після проведеного дослідження виявлено, що PHPStorm має вищий сумарний коефіцієнт в порівнянні з іншими середовищами розробки, такими як Visual Studio, QtCreator і Code::Blocks. Він перевищує їх на 20%, 20% і 40% відповідно. Однією з ключових переваг PHPStorm є наявність сучасного графічного інтерфейсу, який дозволяє зручно та ефективно працювати з PHP-кодом. Крім того, PHPStorm надає широкий спектр функціональностей, таких як підтримка мов програмування, редактор коду з автодоповненням, відлагодження коду та керування проєктом, що робить його ідеальним вибором для розробки вебдодатків.

3.3 Розробка програмного засобу моделі для класифікації музики

Розробка програмного засобу для взаємодії з моделлю включає кілька етапів, починаючи з проєктування архітектури моделі і завершуючи її інтеграцією в програмний продукт.

На етапі проєктування архітектури моделі спочатку визначаються цілі, які вона має вирішувати, і обирається тип моделі, який найбільше підходить для розв'язання конкретних завдань. Далі розробляється структура моделі, включаючи типи шарів і функції активації (рис. 3.1). CNN використовуються в аудіоаналізі через їхню здатність ефективно розпізнавати характеристики аудіосигналів, такі як частотні шаблони, ритмічні структури та інші акустичні особливості. Вони мають здатність автоматично витягувати важливі ознаки з вхідних даних і використовувати їх для класифікації музичних жанрів.

Ця модель підлягає тренуванню на великій кількості аудіоданих, після чого її можна інтегрувати в програмний засіб для роботи з музичними композиціями.

Після проєктування архітектури моделі розпочинається розробка програмного засобу. На цьому етапі реалізується код, що визначає архітектуру моделі за допомогою вибраних бібліотек. Підготовка даних також є важливою складовою цього етапу, оскільки дані обробляються та підготовлюються (рис. 3.2) для тренування моделі.

Після розробки програмного засобу відбувається тренування та налаштування моделі. Визначаються гіперпараметри, тренується модель і валідується на окремому наборі даних для тестування. Це допомагає визначити ефективність моделі та внести необхідні корективи для поліпшення результатів. Після тренування моделі необхідно провести тестування її на незалежному наборі даних для оцінки її загальної продуктивності та точності. Це допомагає впевнитися, що модель працює на реальних даних та відповідає вимогам роботи.

Нарешті, модель інтегрується в програмний продукт (рис.3.4). Вона впроваджується у програмний засіб для використання в реальних умовах, і оптимізується для забезпечення ефективної роботи. Цей процес включає в себе

оптимізацію продуктивності моделі та забезпечення швидкої та ефективної роботи програми.

```
# 1st conv layer
model.add(keras.layers.Conv2D( filters: 32, kernel_size: (3, 3), activation='relu', input_shape=input_shape))
model.add(keras.layers.MaxPooling2D( pool_size: (3, 3), strides=(2, 2), padding='same'))
model.add(keras.layers.BatchNormalization())

# 2nd conv layer
model.add(keras.layers.Conv2D( filters: 32, kernel_size: (3, 3), activation='relu'))
new *
model.add(keras.layers.MaxPooling2D( pool_size: (3, 3), strides=(2, 2), padding='same'))
model.add(keras.layers.BatchNormalization())

# 3rd conv layer
model.add(keras.layers.Conv2D( filters: 32, kernel_size: (2, 2), activation='relu'))
model.add(keras.layers.MaxPooling2D( pool_size: (2, 2), strides=(2, 2), padding='same'))
model.add(keras.layers.BatchNormalization())

# flatten output and feed it into dense layer
model.add(keras.layers.Flatten())
model.add(keras.layers.Dense( units: 64, activation='relu'))
model.add(keras.layers.Dropout(0.3))

# output layer
model.add(keras.layers.Dense( units: 10, activation='softmax'))
```

Рисунок 3.5 – Архітектура шарів моделі

```
# dictionary to store mapping, labels, and MFCCs
data = {
    "mapping": [],
    "labels": [],
    "mfcc": []
}

samples_per_segment = int(SAMPLES_PER_TRACK / num_segments)
num_mfcc_vectors_per_segment = math.ceil(samples_per_segment / hop_length)
```

Рисунок 3.6 – Вигляд словника для збереження даних аудіофайлів

```
# load data
X, y = load_data(DATA_PATH)

# create train, validation and test split
X_train, X_test, y_train, y_test = train_test_split( *arrays: X, y, test_size=test_size)
X_train, X_validation, y_train, y_validation = train_test_split( *arrays: X_train, y_train, test_size=validation_size)
```

Рисунок 3.7 – Розподіл даних на навчальний, валідаційний і тестовий набори

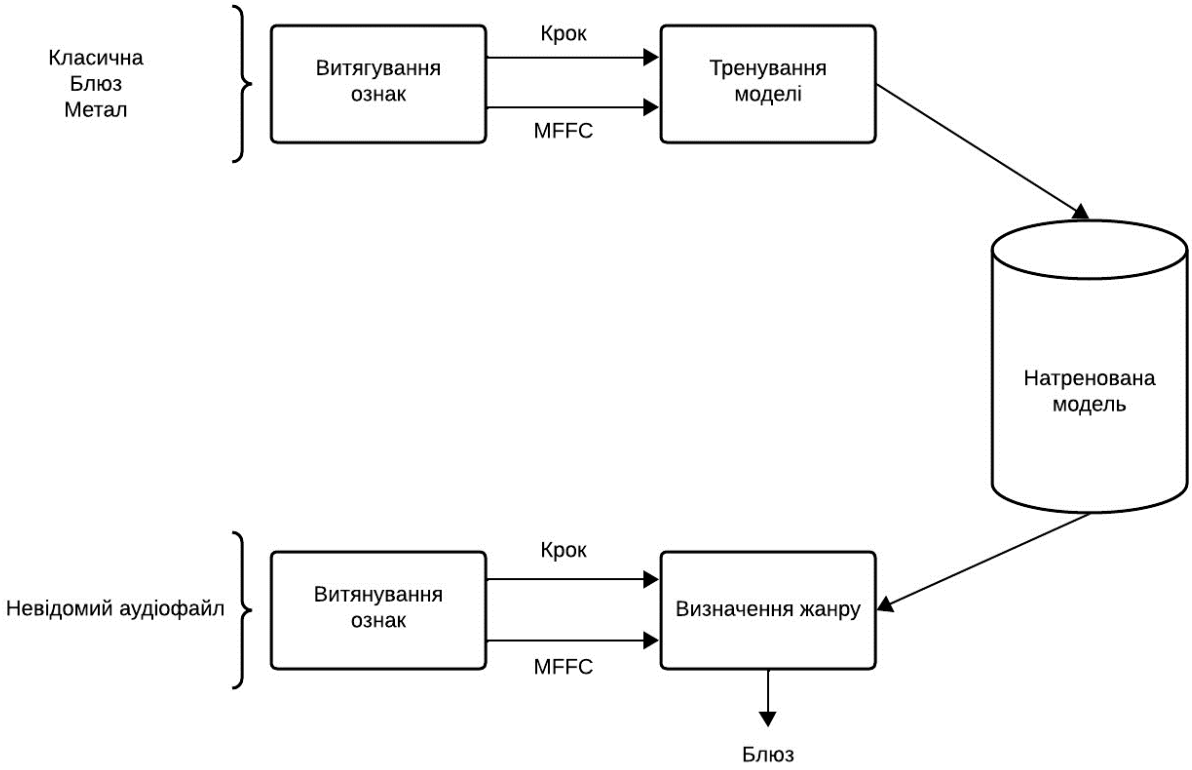


Рисунок 3.8 – Схема інтеграції моделі в вебсистему

Інтеграція натренованої моделі в вебсистему полягає у збереженні натренованої моделі, завантаженні її в вебдодаток та використанні для аналізу музичних композицій. Спочатку потрібно зберегти модель у відповідний формат, щоб можна було її завантажити. Потім вебсистема завантажує цю модель і використовує її для класифікації музичних композицій, отримуючи від неї прогнозовані мітки.

3.3 Розробка функції витягування спеціалізованих параметрів з аудіофайлу

Використання бібліотек math та librosa має велике значення у розвитку програмних засобів для обробки та аналізу аудіоданих.

Функція process_input (рис. 3.5): Ця функція використовується для попередньої обробки аудіофайлів перед подальшою обробкою та аналізом моделлю. Деякі аудіофайли можуть містити шум або інші небажані ефекти, які потрібно видалити або зменшити перед аналізом. Цей крок може включати фільтрацію або попередню обробку сигналу. Вона має наступні параметри:

- audio_file: шлях до аудіофайлу, який потрібно обробити;
- track_duration: тривалість кожного сегмента аудіофайлу (у секундах).

Константи та параметри обробки аудіофайлу: У цьому розділі коду визначаються різні константи та параметри, які використовуються для обробки аудіофайлів. Наприклад, SAMPLE_RATE - це швидкість дискретизації аудіофайлу, NUM_MFCC - кількість MFCC, які будуть видобуті з аудіофайлу, і т.д.

Обробка аудіофайлу: У цьому розділі коду виконується ряд операцій, що стосуються обробки аудіофайлів:

- аудіофайл завантажується за вказаною швидкістю дискретизації (SAMPLE_RATE);
- аудіофайл розділяється на декілька сегментів для обробки;
- для кожного сегмента обчислюються MFCC з використанням функції librosa.feature.mfcc;
- отримані MFCC повертаються як результат обробки.

```python
import librosa
import math

def process_input(audio_file, track_duration):

  SAMPLE_RATE = 22050
  NUM_MFCC = 13
  N_FTT=2048
  HOP_LENGTH=512
  TRACK_DURATION = track_duration # measured in seconds
  SAMPLES_PER_TRACK = SAMPLE_RATE * TRACK_DURATION
  NUM_SEGMENTS = 10

  samples_per_segment = int(SAMPLES_PER_TRACK / NUM_SEGMENTS)
  num_mfcc_vectors_per_segment = math.ceil(samples_per_segment / HOP_LENGTH)

  signal, sample_rate = librosa.load(audio_file, sr=SAMPLE_RATE)

  for d in range(10):

    # calculate start and finish sample for current segment
    start = samples_per_segment * d
    finish = start + samples_per_segment

    # extract mfcc
    mfcc = librosa.feature.mfcc(y=signal[start:finish], sr=sample_rate, n_mfcc=NUM_MFCC, n_fft=N_FTT, hop_length=HOP_LENGTH)
    mfcc = mfcc.T

    return mfcc
```

Рисунок 3.9 – Функція process_input

3.4 Висновки

У третьому розділі проведено обґрунтування вибору мови програмування та технологій для розробки системи аналізу та класифікації музичних композицій. Після аналізу потреб програмного продукту було вирішено використовувати мову програмування Python через її широкі можливості у сфері обробки аудіоданих та розробки вебдодатків. Для зручності розробки графічного інтерфейсу вибрано фреймворк Django, оскільки він надає ефективні інструменти для швидкого розгортання вебдодатків та взаємодії з базою даних. Щодо обробки аудіоданих та навчання моделей машинного навчання для класифікації музичних композицій, вибрано бібліотеки Librosa та TensorFlow, які забезпечують потужні інструменти для аналізу аудіосигналів та створення моделей штучного інтелекту. Було вирішено використовувати сервер баз даних SQLite, що є вбудованим рішенням у фреймворку Django.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Аналіз методів тестування програмного забезпечення

Тестування програмного забезпечення: тестування програмного забезпечення (ПЗ) є критично важливим етапом у процесі розробки програмних продуктів. Під час цього етапу проводяться різноманітні випробування з метою виявлення помилок та недоліків у програмному забезпеченні, що можуть виникнути під час його роботи. Основна мета тестування полягає не лише в виявленні помилок, а й у перевірці відповідності програмного продукту вимогам і очікуванням користувачів. Це дозволяє забезпечити якість та надійність продукту перед його випуском на ринок або використанням. Забезпечення якості та надійності продукту на етапі тестування є критично важливим для уникнення негативних наслідків, таких як втрата даних, злам безпеки або незадовільне користування програмою.

Типи тестування:

- Функціональне тестування:
  - Перевірка функціональних аспектів програмного забезпечення відповідно до вимог користувача. Це включає перевірку правильності роботи окремих функцій програми та їх взаємодії між собою.
  - Виконання функціональних сценаріїв, які відтворюють реальні ситуації використання програми, для виявлення невідповідностей у роботі програми.
  - Приклади функціонального тестування включають тестування користувацького інтерфейсу, димове тестування для перевірки базової функціональності та регресивне тестування для перевірки раніше виявлених помилок після внесення змін у код.
- Нефункціональне тестування:

- Перевірка нефункціональних аспектів програмного забезпечення, таких як продуктивність, безпека та надійність. Це включає оцінку ефективності, швидкодії та стійкості програмного продукту під час його використання.
- Використання автоматизованих тестів для оцінки продуктивності (наприклад, швидкості відкриття вебсторінок) або безпеки (наприклад, тестування на вразливості).
- Приклади нефункціонального тестування включають тестування завантаження для оцінки реакції програми на велику кількість користувачів, тестування безпеки для виявлення потенційних уразливостей та тестування сумісності для перевірки взаємодії програми з іншими системами або пристроями.

Методи тестування:

- Тестування чорної скриньки:
  - Використання тестів на основі функціональних вимог без доступу до внутрішньої структури програми. Тестування проводиться з точки зору кінцевого користувача, не враховуючи внутрішніх механізмів програми.
  - Динамічне тестування включає перевірку реакції програми на різні вхідні дані та визначення правильності її роботи.
  - Перевагами цього методу є простота в реалізації та високий рівень абстракції, однак можуть виникати складнощі у виявленні причин невдач тестів через обмежений доступ до внутрішньої структури програми.
- Тестування білої скриньки:
  - Перевірка внутрішньої структури програми та її вихідного коду з метою виявлення структурних проблем або помилок.
  - Перевагою цього методу є можливість постійного вдосконалення програмного коду, однак він може бути обмеженим у виявленні невідповідностей між функціоналом програми та її реалізацією.
- Тестування сірої скриньки:

- Комбінація підходів тестування чорної та білої скриньки, що дозволяє перевірити як функціональні вимоги, так і внутрішню структуру програми.
- Цей метод ефективний для тестування складних систем, таких як вебсервіси або розподілені системи, оскільки він дозволяє отримати високий рівень покриття тестами з різних точок зору.

2. Висновок:

- Тестування програмного забезпечення відіграє критичну роль у забезпеченні якості та надійності програмних продуктів перед їх впровадженням.

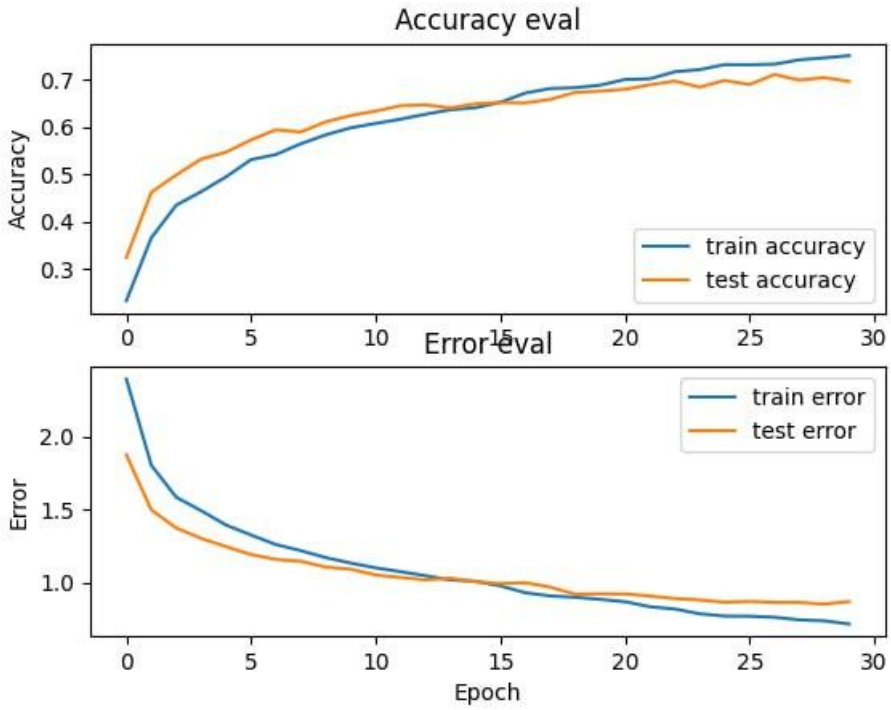
Зважаючи на сильні та слабкі показники кожного із видів тестування, було вирішено, що для тестування вебсистеми потрібно використати підхід інтеграційного ,модульного і системного тестування програми.

Отже, у цьому підрозділі було розглянуто основні види тестування для порграмних продуктів, а особливо призначення специфічних типів та області їх застосування, найпопулярніші види тестування. Використання різних методів та типів тестування дозволяє покращити якість та ефективність тестування програмного забезпечення, зменшити ризики та підвищити задоволення користувачів від використання продукту.

4.2 Аналіз результатів тренування CNN моделі

У рамках розробки вебсистеми для автоматичного аналізу та класифікації музичних композицій було розроблено та натреновано модель на основі згорткових нейронних мереж (CNN). Основна мета полягала у створенні моделі, яка здатна точно ідентифікувати та класифікувати різні музичні композиції на основі їхніх аудіо характеристик. Цей процес включав збір, підготовку даних, розробку моделі, тренування та оцінку її ефективності. Згорткові нейронні мережі були обрані через їхню здатність автоматично виявляти важливі ознаки вхідних даних, що є критично важливим для обробки складних аудіосигналів.

Модель була тренована на великому наборі музичних композицій, що включали різні жанри та стилі музики (рис. 4.1).



Риссунок 4.1 – Результат тестування моделі

Під час тренувань модель продемонструвала значний прогрес у точності тренувань (train accuracy). Спочатку точність була близькою до 0%, але поступово зросла до приблизно 75%. Це свідчило про те, що модель успішно навчилася розпізнавати та класифікувати музичні композиції зі збільшенням кількості епох тренування.

Однак точність на тестових даних (test accuracy) залишилася нижчою, досягаючи максимуму близько 65%. Це може вказувати на те, що модель ще потребує покращення для кращої узагальнюваності на нові дані. Тестова точність є важливим показником того, як добре модель може працювати з даними, які вона не бачила раніше, що є критичним для реального застосування.

Аналізуючи помилки моделі, було виявлено, що початковий тренувальний error (train error) становив 2.8 і поступово знижувався до 0.5. Це підтверджувало,

що модель успішно навчилася зменшувати помилки на тренувальних даних з кожною епохою, що свідчить про ефективність процесу навчання.

Error на тестових даних (test error) був нижчим на початку і становив 1.8, знижуючись до 0.8 під час тренувань. Ці результати можуть вказувати на те, що модель має потенціал для покращення, але її здатність узагальнювати знання на нові дані все ще обмежена. Це означає, що для досягнення кращих результатів у реальних умовах модель потребує додаткової оптимізації та тренування на більш різноманітних даних.

Підсумовуючи, результати тренувань CNN моделі для автоматичного аналізу та класифікації музичних композицій демонструють значний прогрес і потенціал. Проте, існує певний розрив між точністю на тренувальних і тестових даних, що вказує на необхідність подальших покращень. У майбутньому планується використання методів регуляризації, збільшення обсягу та різноманітності тренувальних даних, а також проведення подальшої оптимізації гіперпараметрів моделі для підвищення її ефективності та стійкості.

4.3 Тестування системи

Модульне тестування включає перевірку окремих компонентів системи, таких як алгоритми аналізу звуку та класифікатори музичних жанрів. Для цього можна використовувати автоматизовані тести, які перевіряють роботу кожного модуля окремо, вводячи в них тестові дані та перевіряючи очікувані результати.

Системне тестування зосереджується на перевірці системи в цілому. Включаючи всі функції та можливості, доступні для користувачів. Для цього тести можуть відтворювати різні сценарії використання системи, перевіряти її реакцію на різні вхідні дані та аналізувати її продуктивність [17].

Приймальне тестування зазвичай проводиться замовником або представниками клієнта з метою підтвердження відповідності вимогам та

виявлення потенційних проблем. Це важливий етап, який допомагає впевнитися в якості та готовності системи до впровадження в реальне використання [18].

Результат тестування зображено на рисунку 4.2.

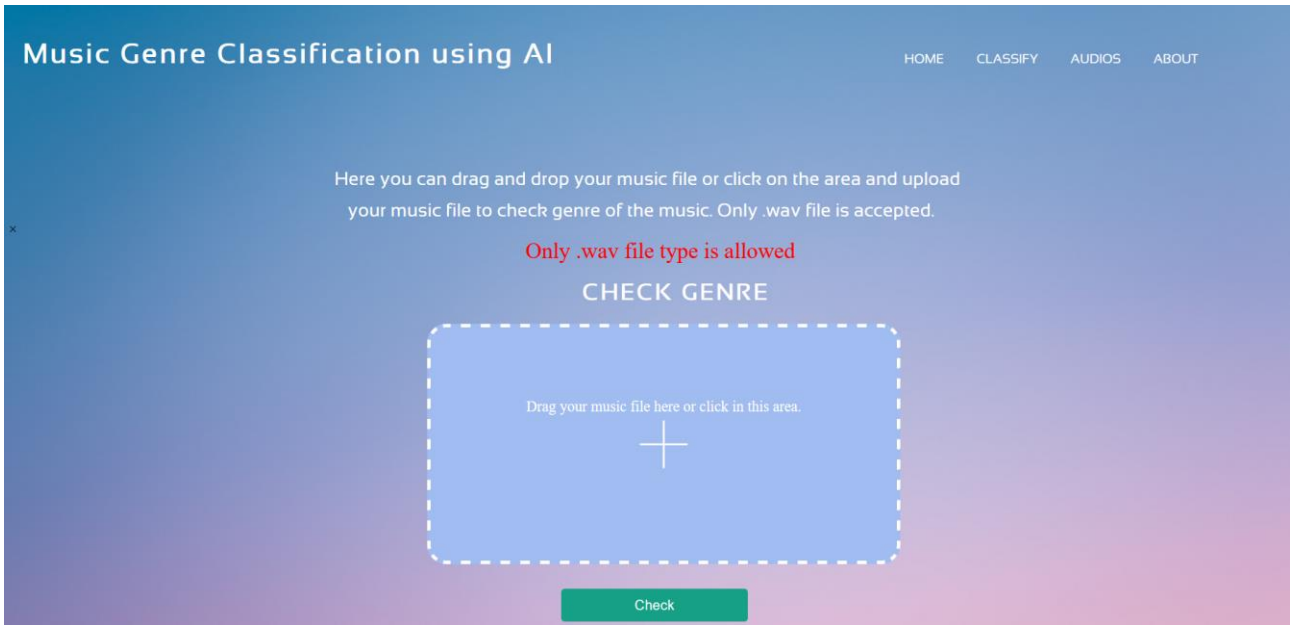


Рисунок 4.2 – Результат приймального тестування

Інтеграційне тестування (рис.4.2) вимагає перевірки взаємодії між різними компонентами системи. Наприклад, переконатися, що обробник аудіоданих правильно передає інформацію до бази даних та що інтерфейс користувача коректно відображає дані, отримані з бази даних [19].

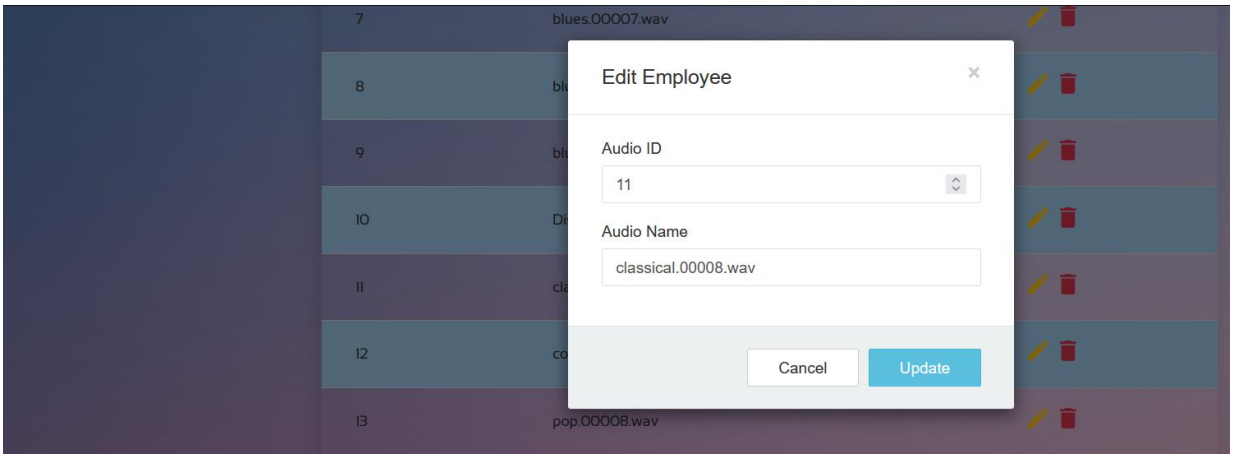


Рисунок 4.3 – Результат інтеграційнго тестування

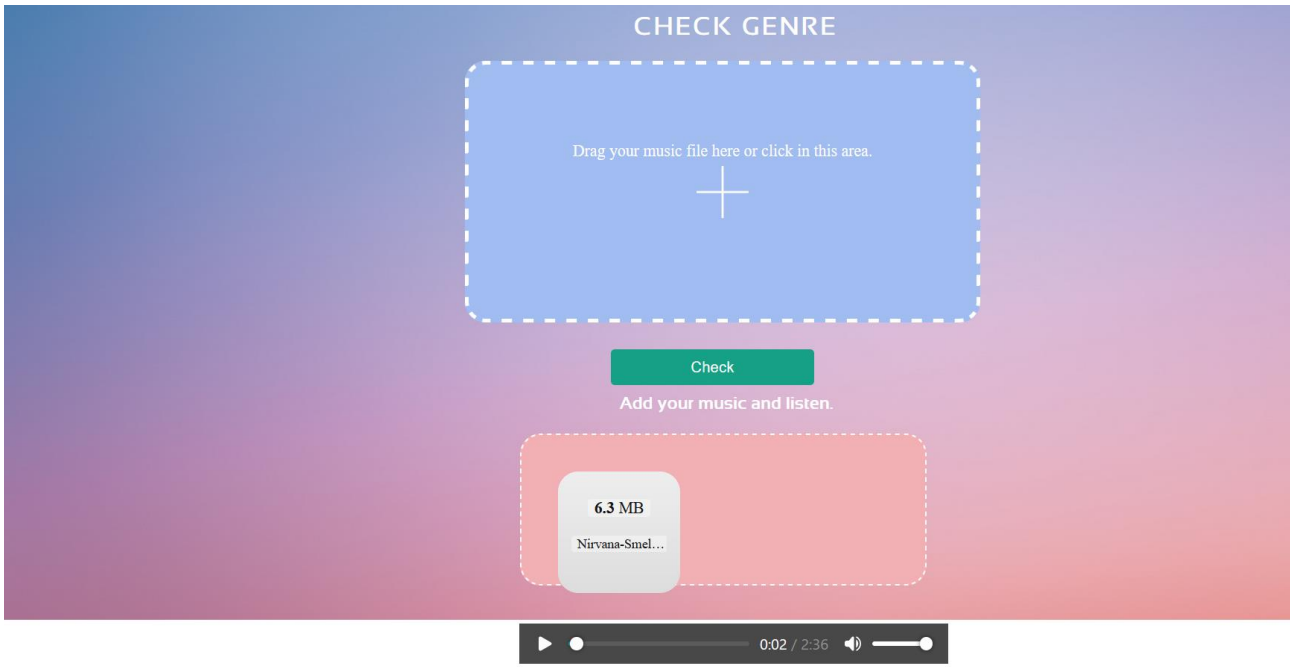


Рисунок 4.4 – Результат системного тестування

При введенні неправильних даних система автоматично перенаправляє користувача на домашню сторінку (розділ "home") (рис.4.5). Також, якщо користувач обрав більше одного файлу, система також перенаправляє його на головну сторінку для уникнення конфліктів або неправильної обробки введених даних [20].

4.4 Розробка інструкції користувача

Інструкція користувача передбачає визначення технічних вимог для запуску програмного продукту [21]. Деталі щодо мінімальної та рекомендованої конфігурації серверного комп’ютера можна знайти в таблиці 4.1.

Таблиця 4.1 – Мінімальна конфігурація:

Технічні вимоги	Мінімальні	Рекомендовані
Процесор	Двоядерний, 1.5 ГГц	Чотирьохядерний, 2.5 ГГц або вище

Технічні вимоги	Мінімальні	Рекомендовані
Оперативна пам'ять (RAM)	2 ГБ	8 ГБ або більше
Місце на диску	500 МБ	2 ГБ або більше
Операційна система	Windows 7 або новіше	Windows 10 або новіше, macOS, Linux
Браузер	Остання версія Chrome, Firefox, Safari або Edge	Остання версія Chrome або Firefox
Інтернет-з'єднання	Потрібно	Потрібно

1. Завантаження Аудіофайлів: Користувачеві доступний розділ для завантаження аудіофайлів. Вказується процедура вибору файлів на комп'ютері користувача та їхнє завантаження на сервер системи.

2. Вибір та Запуск Аналізу: Після завантаження аудіофайлів користувач може вибрати, які саме файли він бажає проаналізувати та класифікувати. Крім того, користувач має можливість вибрати параметри для аналізу, такі як методи обробки аудіоданих та класифікації музичних жанрів.

3. Перегляд та Завантаження Результатів: Після завершення аналізу користувачеві надається звіт про результати класифікації для кожного завантаженого аудіофайлу. Користувач може переглянути ці результати на екрані та вибрати, якщо необхідно, завантажити їх у файлову систему свого пристрою.

4. Взаємодія з Користувачем та Підтримка: У разі виникнення будь-яких питань або проблем користувачеві надається можливість зв'язатися з адміністраторами системи через спеціальну форму зворотного зв'язку або електронну пошту [22].

Ця інструкція призначена для забезпечення зручного та ефективного користування системою користувачами будь-якого рівня досвіду.

4.5 Висновки

У четвертому розділі було проведено функціональне тестування, спрямоване на перевірку основних функцій системи, тестування сумісності для переконання у її коректній роботі на різних платформах та тестування продуктивності для оцінки ефективності під час використання. Крім того, була розроблена інструкція користувача, яка допоможе користувачам в ефективному використанні системи для аналізу та класифікації музичних композицій.

ВИСНОВКИ

У дипломній роботі було розроблено програмні засоби реалізації вебсистеми для автоматичного аналізу і класифікації.

Для розробки було використано середовище програмної розробки VSCode. Бакалаврська дипломна робота розроблена згідно методичних вказівок.

Під час виконання дипломної роботи було проаналізовано стан проблеми на сьогоднішній день. Було розглянуто основні аналоги програмного продукту. Було визначено їхні недоліки та порівняно з власним програмним продуктом. Базуючись на порівнянні було встановлено основні задачі дипломної роботи.

Під час аналізу технологій розробки було обґрунтовано вибір мови програмування Python та бібліотек Librosa, TensorFlow. Також було розглянуто переваги використання фреймворка Django для збереження даних користувачів.

У дипломній роботі було виконано такі завдання:

- досліджено різні методи класифікації аудіофайлів за жанрами з метою вибору найефективнішого підходу;
- створено модель машинного навчання для класифікації аудіофайлів на основі обраного методу;
- розроблено зручний та інтуїтивний вебінтерфейс для взаємодії з користувачем;
- створено базу даних для зберігання метаданих про завантажені аудіофайли;
- проведено тести для перевірки точності та надійності розробленого алгоритму класифікації.

Було створено загальні схеми алгоритму функціонування вебсистеми та опис програмного продукту. Крім того, була розроблена модель системи, використовуючи UML діаграми. Тестування програми підтвердило повну працездатність продукту та відповідність поставленому технічному завданню. Також була підготовлена інструкція користувача.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Тимошенко В.Ю. Використання машинного навчання для класифікації музичних композицій. Матеріали міжнародної науково-практичної інтернет-конференції "Молодь в науці: дослідження, проблеми, перспективи" (ВНТКП ВНТУ), м.Вінниця, 20-29 травня 2024 р. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/viewFile/21317/17705.

2. Spotify: вебсайт. URL: https://hbr.org/2017/11/why-every-organization-needs-an-augmented-reality-strategydas (дата звернення: 16.03.2024).

3. Apple Music : вебсайт. URL: https:/ /www.frontiersin.org/ articles/10.3389/ fpsyg.2018.02086/full (дата звернення: 16.03.2024).

4. Pandora: вебсайт. URL: https://www.interaction-design.org/literature/article/augmented-reality-the-past-the-present-and-the-future (дата звернення: 16.03.2024).

5. Last.fm: вебсайт. URL: https://www.forbes.com/sites/ forbestechcouncil/2023/02/06/why-augmented-reality-is-one-of-the-most-pro mising-experimental-technologies-of-this-decade/?sh=4366e4fd3c85 (дата звернення: 18.03.2024).

6. Deezer : вебсайт. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2023/paper/view/17464/14582 (дата звернення: 18.03.2024).

7. Augmented Reality Development Platforms : вебсайт. URL: https://www.trustradius.com/augmented-reality-development (дата звернення: 25.03.2024).

8. What are convolutional neural networks : вебсайт. URL: https://www.ibm.com/topics/convolutional-neural-networks (дата звернення: 30.03.2024).

9. Mel-frequency cepstral coefficients (MFCCs) : вебсайт. URL: https://medium.com/@MuhyEddin/feature-extraction-is-one-of-the-most-important-steps-in-developing-any-machine-learning-or-deep (дата звернення: 05.04.2024).

10. Веброзробка : вебсайт. URL: https://softgroup.ua/veb-razrabotka (дата звернення: 08.04.2024).

11. Real Python Tutorial : вебсайт. URL: https://realpython.com/ (дата звернення: 08.04.2024).

12. Librosa : вебсайт. URL: https://pypi.org/project/librosa/ (дата звернення: 08.04.2024).

13. Python basics : вебсайт. URL: https://realpython.com/tutorials/basics/ (дата звернення: 25.04.2024).

14. ConvNet Playground: An Interactive Visualization Tool for Exploring Convolutional Neural Networks : вебсайт. URL: https://towardsdatascience.com/convnetplayground-979d441ebf82 (дата звернення: 30.04.2024).

15. The web framework for perfectionists with deadlines. : вебсайт. URL: https://www.djangoproject.com/ (дата звернення: 10.05.2024).

16. The building blocks of algorithms : вебсайт. URL: https://www.khanacademy.org/computing/ap-computer-science-principles/algorithms-101/building-algorithms/a/the-building-blocks-of-algorithms (дата звернення: 10.05.2024).

17. Від ідеї до запуску: 6 етапів розробки сайту : вебсайт. URL: https://e-client.online/blog/rozrobka-sajtiv/6-etapiv-rozrobki-sajtu (дата звернення: 02.10.2024).

18. User interface design : вебсайт. URL: https://www.geeksforgeeks.org/user-interface-ui/ (дата звернення: 10.05.2024)

19. Website testing : вебсайт. URL: https://www.browserstack.com/guide/how-to-perform-website-qa-testing (дата звернення: 02.10.2024).

20. What is software testing? : вебсайт. URL: https://www.ibm.com/topics/software-testing (дата звернення: 12.05.2024).

21. Романюк О. Н., Чорноволик Г. О., Методичні вказівки до виконання бакалаврських дипломних робіт для студентів спеціальності 121 «Інженерія програмного забезпечення»: методичні – вказівки Вінниця : ВНТУ, 2022. 51 с.

22. The different types of software testing : вебсайт. URL: https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing (дата звернення: 12.05.2024).

ДОДАТКИ

Додаток А – Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

д.т.н., проф. О. Н. Романюк

" 12 " березня 2024 р.

Технічне завдання
на бакалаврську дипломну роботу
«Розробка методу і програмних засобів реалізації вебсистеми автоматизованого аналізу та класифікації музичних композицій»
за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:

к.т.н., доц. О.М.Ткаченко

" 12 " березня 2024 р.

Виконав:

студент гр.1ПІ-20б В.Ю. Тимошенко

" 12 " березня 2024 р.

Вінниця – 2024 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка методу і програмних засобів реалізації вебсистеми автоматизованого аналізу та класифікації музичних композицій».

Галузь застосування – вебсистеми з класифікації.

2. Підстава для розробки.

Підставою для виконання бакалаврської дипломної роботи (БДР) є індивідуальне завдання на БДР та наказ № ректора по ВНТУ про закріплення тем БДР.

3 Мета та призначення розробки.

Метою роботи є створення вебсистеми для автоматичного аналізу та класифікації музичних композицій на основі моделі глибокого навчання CNN (Convolutional Neural Network). Головною метою є розробка програмного забезпечення, яке здатне ефективно аналізувати музичні дані та класифікувати їх за різними параметрами, такими як жанр, настрій або виконавець.

Призначення роботи полягає у створенні інструменту, що допомагатиме музичним дослідникам, аудіоінженерам та іншим зацікавленим особам у проведенні аналізу та класифікації музичних композицій за допомогою передових методів машинного навчання.

4 Вихідні дані для проведення БДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. What are convolutional neural networks : вебсайт. URL: https://www.ibm.com/topics/convolutional-neural-networks

2. Mel-frequency cepstral coefficients (MFCCs) : вебсайт. URL: https://medium.com/@MuhyEddin/feature-extraction-is-one-of-the-most-important-steps-in-developing-any-machine-learning-or-deep
3. ConvNet Playground: An Interactive Visualization Tool for Exploring Convolutional Neural Networks : вебсайт. URL: https://towardsdatascience.com/convnetplayground-979d441ebf82
4. The web framework for perfectionists with deadlines. : вебсайт. URL: https://www.djangoproject.com/.

5.Технічні вимоги.

Розробка алгоритмів для автоматичного аналізу та класифікації музичних композицій. Ці вимоги також охоплюють створення системи зберігання та обробки музичних даних, інтеграцію з методами машинного навчання, та розробку інтерфейсу користувача для взаємодії з системою.

6.Конструктивні вимоги.

Створення зручного та інтуїтивно зрозумілого інтерфейсу вебсистеми, здатного ефективно обробляти великі обсяги музичних даних. Документація повинна бути зрозумілою та дотримуватися стандартів представлення технічної інформації в Інтернеті.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до БДР;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи
1	Аналіз розвитку вебсистем класифікації	14.03.24 - 31.03.24
2	Розробка архітектури моделі CNN	01.04.24 - 18.04.24
3	Розробка алгоритмів роботи класифікації та інтерфейсу вебсистеми	19.04.24 - 06.05.24
4	Тестування вебсистеми	07.05.24 - 24.05.24
5	Оформлення матеріалів до захисту БДР	25.05.24 - 30.05.24

10. Порядок контролю та прийняття.

Виконання етапів бакалаврської дипломної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття бакалаврської дипломної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком

Додаток Б – Протокол перевірки роботи

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка методу і програмних засобів реалізації вебсистеми автоматизованого аналізу та класифікації музичних композицій
Тип роботи: Бакалаврська дипломна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: к.т.н., доцент кафедри ПЗ Ткаченко О.М

Оригінальність	92.8
Схожість	7.2

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**
□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.
□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Тимошенко В.Ю.

Керівник роботи _____ Ткаченко О.М.

Додаток В – Лістинг програми

```
# models.py

class Sound(models.Model):
    track_id = models.IntegerField(primary_key=True)
    title = models.CharField(max_length=100)
    file = models.FileField(upload_to='audio/')

    def __str__(self):
        return self.title

class UploadFile(models.Model):
    document = models.FileField(upload_to='documents/')

def classify_genre(data_to_classify):
    genres = {0: "classical", 1: "rock", 2: "jazz", 3: "reggae", 4: "blues", 5: "metal", 6: "hiphop", 7: "disco", 8: "country", 9: "pop"}
    # Завантаження моделі
    model = load_model('models/MusicGenreClassifier.h5')
    predictions = model.predict(data_to_classify)

    # Отримання індексу з максимальною ймовірністю
    predicted_genre_index = np.argmax(predictions, axis=1)

    return genres[int(predicted_genre_index)]

# preprocess_audio.py
# Попередня обробка аудіофайлів

import librosa
import math

def extract_features(audio_path, duration):
    SAMPLE_RATE = 22050
```

```
    MFCC_COUNT = 13
    FFT_SIZE = 2048
    HOP_SIZE = 512
    DURATION = duration  # у секундах
    SAMPLES_PER_TRACK = SAMPLE_RATE * DURATION
    SEGMENTS = 10

    samples_per_segment = int(SAMPLES_PER_TRACK / SEGMENTS)
    mfcc_vectors_per_segment = math.ceil(samples_per_segment / HOP_SIZE)

    signal, sr = librosa.load(audio_path, sr=SAMPLE_RATE)

    for segment in range(10):
        # Обчислення початку і кінця сегмента
        start = samples_per_segment * segment
        end = start + samples_per_segment

        # Витягування MFCC
        mfccs = librosa.feature.mfcc(y=signal[start:end], sr=sr, n_mfcc=MFCC_COUNT,
n_fft=FFT_SIZE, hop_length=HOP_SIZE)
        mfccs = mfccs.T

        return mfccs

# urls.py

urlpatterns = [
    path('', views.home, name='home'),
    path('analyze', views.analyze, name='analyze'),
    path('library', views.library, name='library'),
    path('info', views.info, name='info'),
    path('results', views.results, name='results'),
]
```

```
# views.py

def home(request):
    return render(request, 'home.html')

def analyze(request):
    return render(request, 'analyze.html')

def library(request):
    tracks = Sound.objects.all()
    return render(request, 'library.html', {'tracks': tracks})

def info(request):
    return render(request, 'info.html')

def results(request):
    return render(request, 'results.html')

def upload_file(request):
    if request.method == 'POST':
        uploaded_file = request.FILES.get('file')
        Sound.objects.create(title=uploaded_file.name, file=uploaded_file)
        return HttpResponse('')
    return JsonResponse({'error': 'Please upload a file'})

def edit_entry(request):
    tracks = Sound.objects.all()
    context = {'tracks': tracks}
    return redirect(render, 'library.html', context)

def update_entry(request, track_id):
    if request.method == 'POST':
        track_id = request.POST.get('track_id')
        title = request.POST.get('title')
```

```
        track = Sound(
            track_id=track_id,
            title=title,
        )
        track.save()
        return redirect('library')
    return redirect(render, 'library.html')

def delete_entry(request, track_id):
    track = Sound.objects.filter(track_id=track_id)
    track.delete()
    context = {'tracks': track}
    return redirect('library')

def upload_document(request):
    documents = UploadFile.objects.all()
    if request.method == 'POST':
        if len(request.FILES) == 0:
            messages.error(request, 'Upload a file')
            return redirect("home")

        form = DocumentForm(request.POST, request.FILES)
        if form.is_valid():
            uploaded_doc = request.FILES['document']
            if not uploaded_doc.name.endswith('.wav'):
                messages.error(request, 'Only .wav file type is allowed')
                return redirect("home")
            duration = 30
            extracted_features = extract_features(uploaded_doc, duration)
            data_to_classify = extracted_features[np.newaxis, ..., np.newaxis]
            genre = classify_genre(data_to_classify)

            context = {'genre': genre}
```

```
            return render(request, 'results.html', context)
    else:
        form = DocumentForm()

    return render(request, 'results.html', {'documents': documents, 'form': form})

def not_found(request, exception):
    return render(request, 'not_found.html')

# train_model.py
DATA_FILE = "../data/data_10.json"

def fetch_data(data_file):
    with open(data_file, "r") as file:
        data = json.load(file)
    X = np.array(data["mfcc"])
    y = np.array(data["labels"])
    return X, y

def visualize_history(history):
    fig, axes = plt.subplots(2)

    # Підплот точності
    axes[0].plot(history.history["accuracy"], label="train accuracy")
    axes[0].plot(history.history["val_accuracy"], label="test accuracy")
    axes[0].set_ylabel("Accuracy")
    axes[0].legend(loc="lower right")
    axes[0].set_title("Accuracy evaluation")

    # Підплот помилок
    axes[1].plot(history.history["loss"], label="train error")
    axes[1].plot(history.history["val_loss"], label="test error")
    axes[1].set_ylabel("Error")
    axes[1].set_xlabel("Epoch")
```

```
        axes[1].legend(loc="upper right")
        axes[1].set_title("Error evaluation")

        plt.show()

    def prepare_data_splits(test_size, validation_size):
        # Завантаження даних
        X, y = fetch_data(DATA_FILE)

        # Створення тренувальної, валідаційної та тестової вибірок
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size)
        X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=validation_size)

        # Додавання осі до наборів даних
        X_train = X_train[..., np.newaxis]
        X_val = X_val[..., np.newaxis]
        X_test = X_test[..., np.newaxis]

        return X_train, X_val, X_test, y_train, y_val, y_test

    def construct_model(input_shape):
        # Створення топології мережі
        model = keras.Sequential()

        # 1-й згортковий шар
        model.add(keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape))
        model.add(keras.layers.MaxPooling2D((3, 3), strides=(2, 2), padding='same'))
        model.add(keras.layers.BatchNormalization())

        # 2-й згортковий шар
        model.add(keras.layers.Conv2D(32, (3, 3), activation='relu'))
        model.add(keras.layers.MaxPooling2D((3, 3), strides=(2, 2), padding='same'))
        model.add(keras.layers.BatchNormalization())
```

```
    # 3-й згортковий шар
    model.add(keras.layers.Conv2D(32, (2, 2), activation='relu'))
    model.add(keras.layers.MaxPooling2D((2, 2), strides=(2, 2), padding='same'))
    model.add(keras.layers.BatchNormalization())

    # Плоский шар та dense шар
    model.add(keras.layers.Flatten())
    model.add(keras.layers.Dense(64, activation='relu'))
    model.add(keras.layers.Dropout(0.3))

    # Вихідний шар
    model.add(keras.layers.Dense(10, activation='softmax'))

    return model

def make_prediction(model, X, y):
    # Додавання виміру до вхідних даних
    X = X[np.newaxis, ...]

    # Виконання передбачення
    predictions = model.predict(X)

    # Отримання індексу з максимальною ймовірністю
    predicted_genre_index = np.argmax(predictions, axis=1)

    print("Target: {}, Predicted label: {}".format(y, predicted_genre_index))

if __name__ == "__main__":
    # Отримання тренувальної, валідаційної та тестової вибірок
    X_train, X_val, X_test, y_train, y_val, y_test = prepare_data_splits(0.25, 0.2)

    # Створення моделі
    input_shape = (X_train.shape[1], X_train.shape[2], 1)
```

```
        model = construct_model(input_shape)

        # Компіляція моделі
        optimizer = keras.optimizers.Adam(learning_rate=0.0001)
        model.compile(optimizer=optimizer,
                loss='sparse_categorical_crossentropy',
                metrics=['accuracy'])

        model.summary()
```

```
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Music Genre Classifier</title>
   {% load static %}
   <link rel = "stylesheet" href ="{%static 'css/about.css'%}">
   <script defer src="https://use.fontawesome.com/releases/v5.0.13/js/all.js" integrity="sha384-
xymdQtn1n3lH2wcu0qhcdaOpQwyoarkgLVxC/wZ5q7h9gHtxICrpcaSUfygqZGOe"
crossorigin="anonymous"></script>
   <!--Link to use the font awsesome icons-->
   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">
   <!--Link to use JQuery-->
   <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
</head>
<body>
   <!--Header-->
   <div class= "main_title">
      <h1>About Us</h1>
   </div>

   <!--Navigation Bar-->
   <div class = "wrapper">
      <nav>
         <div class="hamburger">
            <div class="line1"></div>
            <div class="line2"></div>
            <div class="line3"></div>
         </div>
         <ul class="navbar-nav">
            <li>
            <a href="{% url 'home' %}" class="active active-first">home</a>
```

```
            </li>
            <li>
            <a href="{% url 'classify' %}">classify</a>
            </li>
            <li>
            <a href="{% url 'category' %}">audios</a>
            </li>
            <li>
            <a href="{% url 'about' %}">about</a>
            </li>
        </ul>
    </nav>
</div><!--End of Navigation Bar-->

<!--Pre-loader-->
<div class="loader-wrapper">
    <div class="center">
        <div class="wave"></div>
        <div class="wave"></div>
        <div class="wave"></div>
        <div class="wave"></div>
        <div class="wave"></div>
        <div class="wave"></div>
        <div class="wave"></div>
        <div class="wave"></div>
        <div class="wave"></div>
        <div class="wave"></div>
      </div>
</div> <!--End of Pre-loader-->

<!--Container for image -->
<div class="image_holder">
    <img src = "{% static 'images/about.png' %}"/>
</div>

<!--About info holder-->
<div class = "about_container">
    <h4>About Us</h4>
    <!--About details-->
    <div class = "details">
        <p>This is a simple system to recognize the different music genres using artificial intelligence (AI).
            Convolutional Neural Network (CNN) is used as a algorithm to create a machine learning model. Here,
            you can simply drag and drop or choose the audio file and the sysytem will recognize the genre of
            that audio file.
        </p>
    </div>
    <!--Copyright holder-->
    <div class = "copy_right">
        <h5> Copyright &copy; 2021-2022 by Roshan Thapa Magar  </h5>
```

```
        </div>
        <!--Social Media Links-->
        <div class="social_media">
            <div class="rounded-social-buttons">
                <a class="social-button facebook" href="https://www.facebook.com/roshan.lungalimagar"
target="_blank"><i class="fab fa-facebook-f"></i></a>
                <a class="social-button twitter" href="https://twitter.com/RoshanT55730761" target="_blank"><i
class="fab fa-twitter"></i></a>
                <a class="social-button linkedin" href="https://www.linkedin.com/in/roshan-lungali-967441208/"
target="_blank"><i class="fab fa-linkedin"></i></a>
                <a class="social-button youtube"
href="https://www.youtube.com/channel/UCrl6B84E29vzFqQ9gRQFJIg" target="_blank"><i class="fab fa-
youtube"></i></a>
                <a class="social-button instagram" href="https://www.instagram.com/" target="_blank"><i class="fab
fa-instagram"></i></a>
            </div>
        </div>
    </div>
    <script type="text/javascript">

        //Responsive Navigation toggle//
        const hamburger = document.querySelector(".hamburger");
        const navLinks = document.querySelector(".navbar-nav");
        const links = document.querySelectorAll(".navbar-nav li");

        hamburger.addEventListener('click', ()=>{
        //Animate Links
        navLinks.classList.toggle("open");
        links.forEach(link => {
        link.classList.toggle("fade");
        });

        //Hamburger Animation
        hamburger.classList.toggle("toggle");
        });

        $(window).on("load", function() {
            $(".loader-wrapper").fadeOut(2000);
        })
    </script>
</body>
</html>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Music Genre Classifier</title>
```

```
    {% load static %}
    <link rel = "stylesheet" href ="{%static 'css/category.css'%}">
    <!--<script src = "{% static 'javascript/classify.js' %}"></script>-->
    <script defer src="https://use.fontawesome.com/releases/v5.0.13/js/all.js" integrity="sha384-
xymdQtn1n3lH2wcu0qhcdaOpQwyoarkgLVxC/wZ5q7h9gHtxICrpcaSUfygqZGOe"
crossorigin="anonymous"></script>
    <!--Link to use the font awsesome icons-->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">
    <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto|Varela+Round">
    <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body>
    <!--Header-->
    <div class= "main_title">
        <h1>Audio Files</h1>
    </div>

    <!--Navigation Bar-->
    <div class = "wrapper">
        <nav>
            <div class="hamburger">
                <div class="line1"></div>
                <div class="line2"></div>
                <div class="line3"></div>
            </div>
            <ul class="navbar-nav">
                <li>
                <a href="{% url 'home' %}" class="active active-first">home</a>
                </li>
                <li>
                <a href="{% url 'classify' %}">classify</a>
                </li>
                <li>
                <a href="{% url 'category' %}">audios</a>
                </li>
                <li>
                <a href="{% url 'about' %}">about</a>
                </li>
            </ul>
        </nav>
    </div><!--End of Navigation Bar-->

    <!--Pre-loader-->
    <div class="loader-wrapper">
```

```
    <div class="center">
      <div class="wave"></div>
      <div class="wave"></div>
      <div class="wave"></div>
      <div class="wave"></div>
      <div class="wave"></div>
      <div class="wave"></div>
      <div class="wave"></div>
      <div class="wave"></div>
      <div class="wave"></div>
      <div class="wave"></div>
    </div>
  </div> <!--End of Pre-loader-->
  <!--Main Content-->
  <div class ="audio_table_title">
    <h2>Uploaded Audio Files</h2>
  </div>
  <div class = "uploaded_audio_table">
    {% if audi %}
    <table class = "content-table">
      <thead>
        <tr>
          <th>Audio ID</th>
          <th>Audio Name</th>
          <th>Edit/Delete</th>
        </tr>
      </thead>
      <tbody>
      {% for audio in audi %}

        <tr>
          <td>{{audio.audio_id}}</td>
          <td>{{audio.name}}</td>
          <td>
            <a href="#editAudioModal-{{forloop.counter}}" class="edit" data-toggle="modal"><i
class="material-icons" data-toggle="tooltip" title="Edit" style = "color:#ffc720">&#xE254;</i></a>
            <a href="#deleteAudioModal-{{forloop.counter}}" class="delete" data-toggle="modal"><i
class="material-icons" data-toggle="tooltip" title="Delete" style = "color:#d9354f">&#xE872;</i></a>
            <!--<audio controls id="audio_player"></audio>-->
          </td>
        </tr>

      {% endfor %}
      </tbody>
    </table>

    {% else %}
      <h3>No Audio Uploaded</h3>
    {% endif %}
```

```
    </div>

    <!-- Edit Modal HTML -->
    {% for i in audi%}
    <div id="editAudioModal-{{forloop.counter}}" class="modal fade">
      <div class="modal-dialog">
        <div class="modal-content">
          <form action = "/update/{{i.audio_id}}" method = "post">
            {% csrf_token %}
            <div class="modal-header">
              <h4 class="modal-title">Edit Employee</h4>
              <button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>
            </div>
            <div class="modal-body">
              <div class="form-group">
                <label>Audio ID</label>
                <input name = "audio_id" value = "{{i.audio_id}}"type="number" class="form-control"
required>
              </div>
              <div class="form-group">
                <label>Audio Name</label>
                <input name = "name" value = "{{i.name}}"type="text" class="form-control" required>
              </div>
            </div>
            <div class="modal-footer">
              <input type="button" class="btn btn-default" data-dismiss="modal" value="Cancel">
              <input type="submit" class="btn btn-info" value="Update">
            </div>
          </form>
        </div>
      </div>
    </div>
    {% endfor %}
    <!-- Delete Modal HTML -->
    {% for i in audi%}
    <div id="deleteAudioModal-{{forloop.counter}}" class="modal fade">
      <div class="modal-dialog">
        <div class="modal-content">
          <form>
            <div class="modal-header">
              <h4 class="modal-title">Delete Audio File</h4>
              <button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>
            </div>
            <div class="modal-body">
              <p>Are you sure you want to delete these Records?</p>
              <p class="text-warning"><small>You are deleting {{i.name}}</small></p>
            </div>
            <div class="modal-footer">
              <input type="button" class="btn btn-default" data-dismiss="modal" value="Cancel">
```

```
                    <a href = "delete/{{i.audio_id}}" type="submit" class="btn btn-danger">Delete</a>
                </div>
            </form>
        </div>
    </div>
</div>
{% endfor %}

<!--script-->
<script type="text/javascript">

    //Responsive Navigation toggle//
    const hamburger = document.querySelector(".hamburger");
    const navLinks = document.querySelector(".navbar-nav");
    const links = document.querySelectorAll(".navbar-nav li");

    hamburger.addEventListener('click', ()=>{
    //Animate Links
    navLinks.classList.toggle("open");
    links.forEach(link => {
    link.classList.toggle("fade");
    });

    //Hamburger Animation
    hamburger.classList.toggle("toggle");
    });

    //Pre-loader
    $(window).on("load", function() {
        $(".loader-wrapper").fadeOut(2000);
    })
</script>
</body>
</html>
</html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Music Genre Classifier</title>
    {% load static %}
    <link rel = "stylesheet" href ="{%static 'css/classify.css'%}">
    <!--<script src="{% static 'recognize.js' %}" type="text/javascript"></script>-->
    <!--Link to use the font awsesome icons-->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">
    <!--Link to use Dropzone.js-->
    <script src="https://unpkg.com/dropzone@5/dist/min/dropzone.min.js"></script>
```

```
    <!--Link to dropzone.js css-->
    <link rel="stylesheet" href="https://unpkg.com/dropzone@5/dist/min/dropzone.min.css" type="text/css" />
    <!--Link to use JQuery-->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
</head>
<body>
    <!--Header-->
    <div class= "main_title">
        <h1>Music Genre Classification using AI</h1>
    </div>

    <!--Navigation Bar-->
    <div class = "wrapper">
        <nav>
            <div class="hamburger">
                <div class="line1"></div>
                <div class="line2"></div>
                <div class="line3"></div>
            </div>
            <ul class="navbar-nav">
                <li>
                <a href="{% url 'home' %}" class="active active-first">home</a>
                </li>
                <li>
                <a href="{% url 'classify' %}">classify</a>
                </li>
                <li>
                <a href="{% url 'category' %}">audios</a>
                </li>
                <li>
                <a href="{% url 'about' %}">about</a>
                </li>
            </ul>
        </nav>
    </div>
    <!--End of Navigation -->

    <!--Pre-loader-->
    <div class="loader-wrapper">
        <div class="center">
            <div class="wave"></div>
            <div class="wave"></div>
            <div class="wave"></div>
            <div class="wave"></div>
            <div class="wave"></div>
            <div class="wave"></div>
            <div class="wave"></div>
            <div class="wave"></div>
            <div class="wave"></div>
```

```
        <div class="wave"></div>
      </div>
  </div> <!--End of Pre-loader-->

  <!--Main Content-->
  <!--Drag and drop infomation-->
  <div class = "drag_info">
    <h5>Here you can drag and drop your music file or click on the area and upload </h5>
      <h6> your music file to check genre of the music. Only .wav file is accepted.</h6>
       {% include 'errormsg.html' %}
      <h2>CHECK GENRE</h2>
  <!--Drag and drop area-->
  <div class="row">
    <form action="{% url 'result' %}" method="POST" enctype="multipart/form-data" class="up-form" id ="up-
form">
       {% csrf_token %}
       <input type="file" name='document' multiple="" class="up-input">
       {{form.as_table}}
       <img src="{% static 'images/Plus.png' %}" alt="plus" class="plus_icon">
      <p class="up-p">Drag your music file here or click in this area.</p>
       <button type="submit" class="btn btn-sm btn-success" style="margin-left: 9rem; width:
30vh;">Check</button>
    </form>
  </div> <!-- end of row -->
</div> <!-- end of drag_info -->
  <!--Add music and listen-->
  <div class="add_music">
    <h5>Add your music and listen.</h5>
  </div>
  <form action="upload/" method = "POST" class="dropzone dz" id = "my-dropzone">
    {% csrf_token %}
    <div class="dz-message">
      <img src="{% static 'images/Plus.png' %}" alt="plus" class="plus_icon_second">
      <p>Drop your audio files here or Click</p>
    </div>

    <div class="fallback">
      <input  required name="file" type="file" id = "audio"/>
    </div>

  </form>
  <!--Audio Player-->
  <p>
    <audio controls id="audio_player"></audio>
  </p>


  <!--End of Main Content-->
  <!--Scripts-->
```

```
    <script type="text/javascript">

        //Responsive Navigation toggle//
        const hamburger = document.querySelector(".hamburger");
        const navLinks = document.querySelector(".navbar-nav");
        const links = document.querySelectorAll(".navbar-nav li");

        hamburger.addEventListener('click', ()=>{
        //Animate Links
        navLinks.classList.toggle("open");
        links.forEach(link => {
        link.classList.toggle("fade");
        });

        //Hamburger Animation
        hamburger.classList.toggle("toggle");
        });

        //Drag and drop script//
        Dropzone.autoDiscover = false;

        var myDropzone = new Dropzone("#my-dropzone",
        {url:"upload/", uploadMultile:false, maxFiles:1, maxFilesize:7, acceptedFiles:".mp3,.wav"});

        myDropzone.on("addedfile", function(file) {
            console.log("File added");
            var audio = document.getElementById("audio_player");
            audio.src = URL.createObjectURL(file);
            audio.load();
            audio.play();
        });

        $(document).ready(function(){
            $('.up-input').change(function () {
                $('.up-p').text(this.files.length + " file(s) selected");
            });
        });

        //Pre-loader fader script
        $(window).on("load", function() {
            $(".loader-wrapper").fadeOut(3000);
        })
    </script>
</body>
</html>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Music Genre Classifier</title>
    {% load static %}
    <link rel = "stylesheet" href ="{%static 'css/home.css'%}">
    <!--Link in order to use the icon from iconify-->
    <script src="https://code.iconify.design/2/2.1.0/iconify.min.js"></script>
    <!--Link to use the font awsesome icons-->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">
    <!--Link to use JQuery-->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
</head>
<body>
    <!--Header-->
    <div class= "main_title">
        <h1>Music Genre Classifier</h1>
    </div>

    <!--Navigation Bar-->
    <div class = "wrapper">
        <nav>
            <div class="hamburger">
                <div class="line1"></div>
                <div class="line2"></div>
                <div class="line3"></div>
            </div>
            <ul class="navbar-nav">
                <li>
                <a href="{% url 'home' %}" class="active active-first">home</a>
                </li>
                <li>
                <a href="{% url 'classify' %}">classify</a>
                </li>
                <li>
                <a href="{% url 'category' %}">audios</a>
                </li>
                <li>
                <a href="{% url 'about' %}">about</a>
                </li>
            </ul>
        </nav>
    </div>
    <!--End of Navigation Bar-->

        <!--Pre-loader-->
        <div class="loader-wrapper">
            <div class="center">
                <div class="wave"></div>
```

```
            <div class="wave"></div>
            <div class="wave"></div>
            <div class="wave"></div>
            <div class="wave"></div>
            <div class="wave"></div>
            <div class="wave"></div>
            <div class="wave"></div>
            <div class="wave"></div>
            <div class="wave"></div>
          </div>
    </div> <!--End of Pre-loader-->

<!--Soung wave gif image-->
<div class= "logo">
    <p>
    {% load static %}<img src= "{% static '/images/sound-wave.gif' %}"/>
  </p>
</div>

<!--Changing word conatiner-->
<div class = "change_it">
    <h2>Music Genre Classification for <span></span></h2>
</div>

<!--get started button-->
<div class="start">
    <button id="startButton" class="button">Get Started
        <span class="iconify" data-icon="zondicons:arrow-thin-right"></span>
    </button>
</div>

<!--Script for button click event listen and navigate to another page-->
<script type="text/javascript">
    document.getElementById("startButton").onclick = function () {
    location.href = "{% url 'classify' %}";
     };

    const hamburger = document.querySelector(".hamburger");
    const navLinks = document.querySelector(".navbar-nav");
    const links = document.querySelectorAll(".navbar-nav li");

    hamburger.addEventListener('click', ()=>{
    //Animate Links
    navLinks.classList.toggle("open");
    links.forEach(link => {
    link.classList.toggle("fade");
    });

    //Hamburger Animation
```

```
        hamburger.classList.toggle("toggle");
        });

        //Pre-loader fader script
        $(window).on("load", function() {
            $(".loader-wrapper").fadeOut(3000);
        })

    </script>
</body>
</html>
<!DOCTYPE html>
{%load static%}
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Music Genre Classifier</title>
    <link rel = "stylesheet" href ="{%static 'css/result.css'%}">
</head>
<body>
    <div class = "back_arrow">
        <a href = "{% url 'classify' %}">
            <img src = "{% static 'images/back key.png' %}" alt = "back_arrow">
        </a>
    </div>
    <div class ="main_heading">
        <h2>Music Genre Recognition Process Completed</h2>
    </div>
    <div class = "check_image_holder">
        <img src = "{%static 'images/check.png'%}" alt = "check">
    </div>
    <div class="row">
        <div class="column">
            <div class="your_genre">
                <h2>Your Music's Genre is </h2>
            </div> <!---your_genre--->
                <div class ="main_result">
                    <div class = "party_holder-1">
                        <img class="img-fluid" src="{% static 'images/party_popper1.png' %}" alt="alternative">
                    </div> <!---party_holder-1--->
                    <div class = "my_genre">
                        <div class="card right-pane" style="margin-top: -5rem;">
                            <div class="card-body">
                                <div class="card-icon" style="height: 38vh; width: 38vh; margin-top: 6rem; margin-left:
3rem; background-color: blueviolet; border-radius:50%" >
                                    <h3>
                                        {{ genre }}
```

```
                        </h3>
                    </div>
                </div> <!---my_genre--->
                <div class ="party_holder-2">
                    <img class="img-fluid" src="{% static 'images/party_popper2.png' %}" alt="alternative">
                </div> <!---party_holder-2--->
            </div> <!---main_result--->
        </div> <!--- column -->
    </div> <!-- end of row -->

</body>
</html>
```

Додаток Г – Ілюстративна частина

ІЛЮСТРАТИВНА ЧАСТИНА

Розробка методу і програмних засобів реалізації вебсистеми автоматизованого аналізу та класифікації музичних композицій

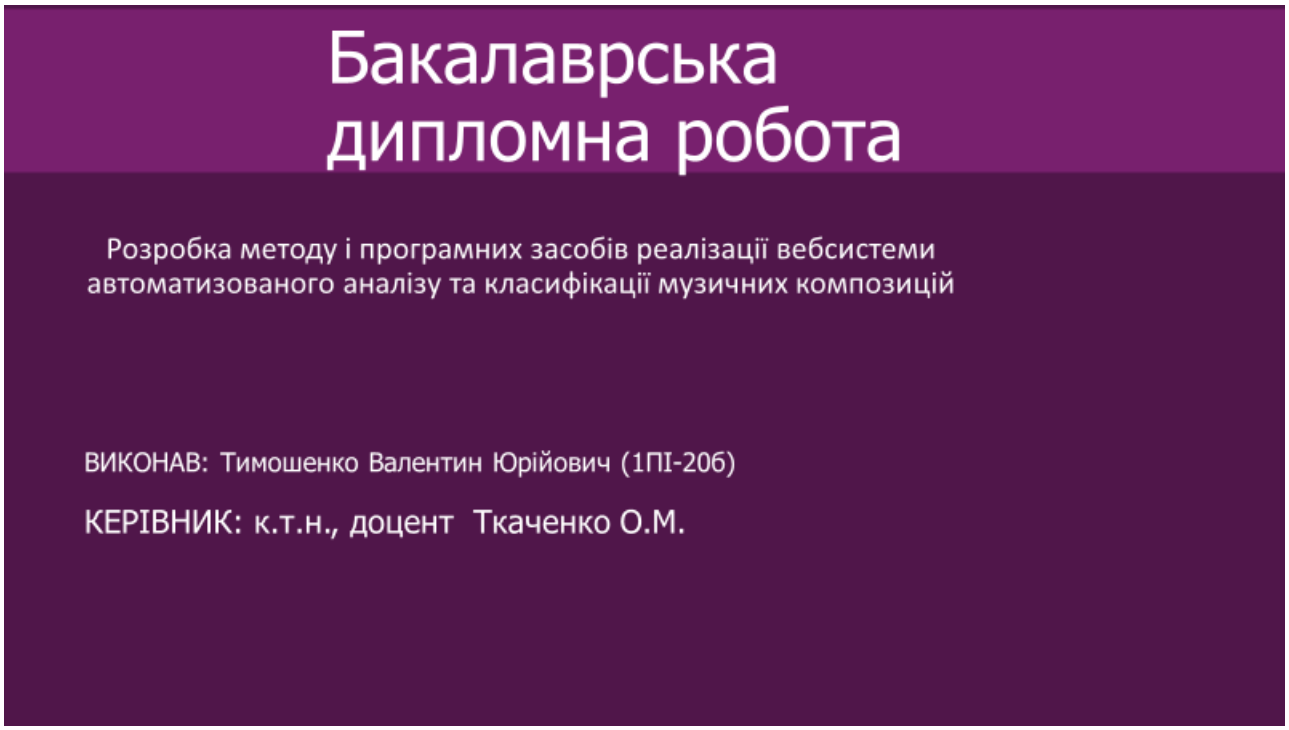


Рисунок Г.1 – Слайд презентації №1

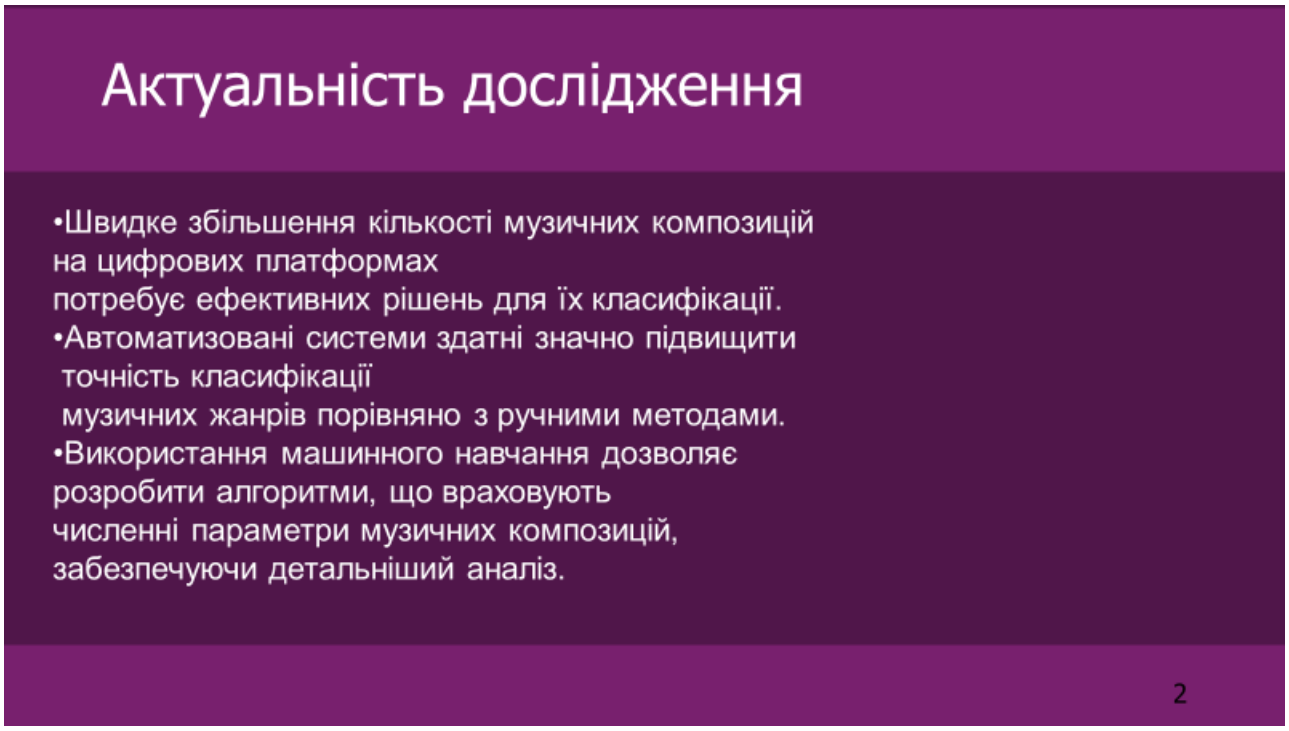


Рисунок Г.2 – Слайд презентації №2

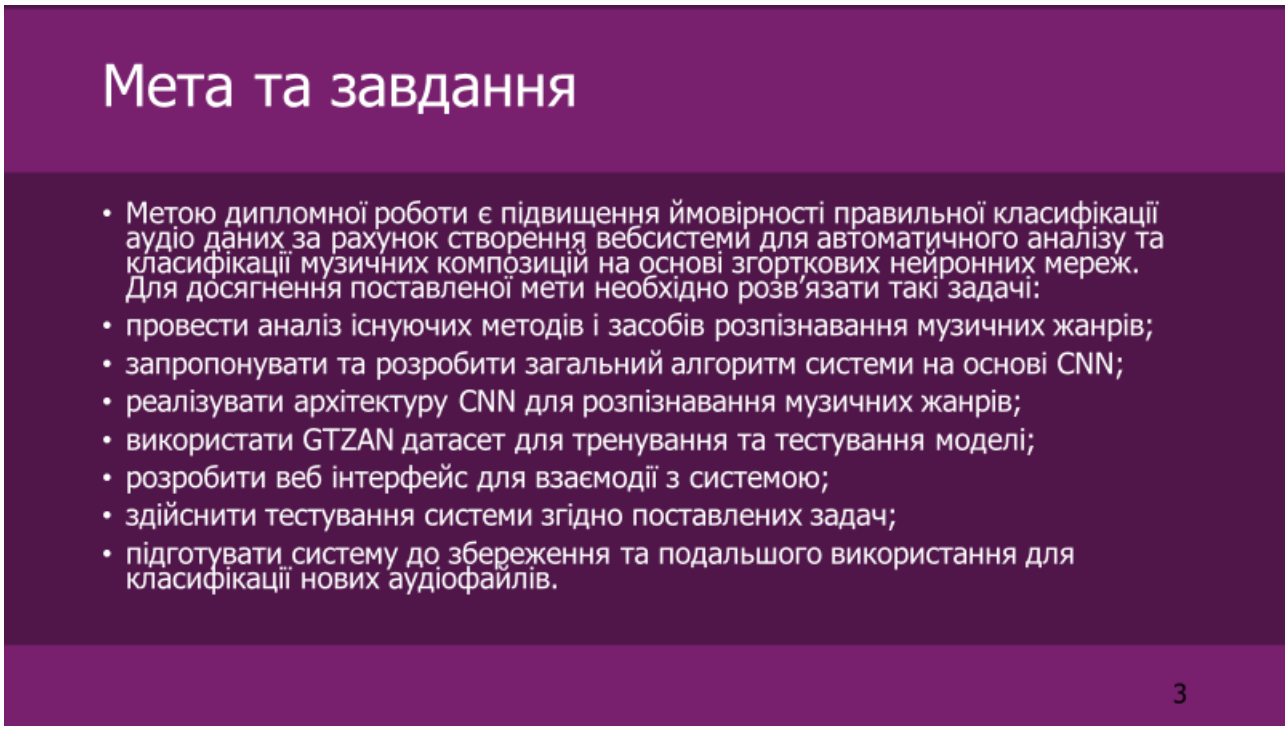


Рисунок Г.3 – Слайд презентації №3

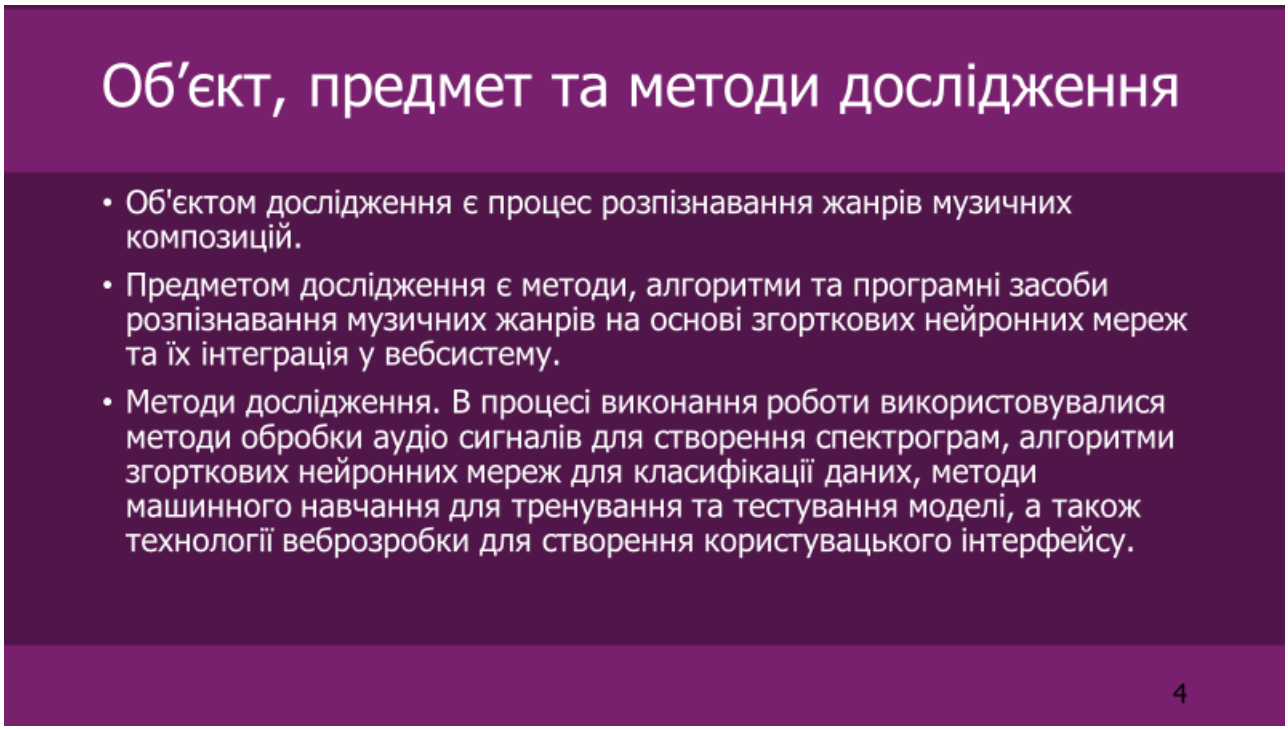


Рисунок Г.4 – Слайд презентації №4

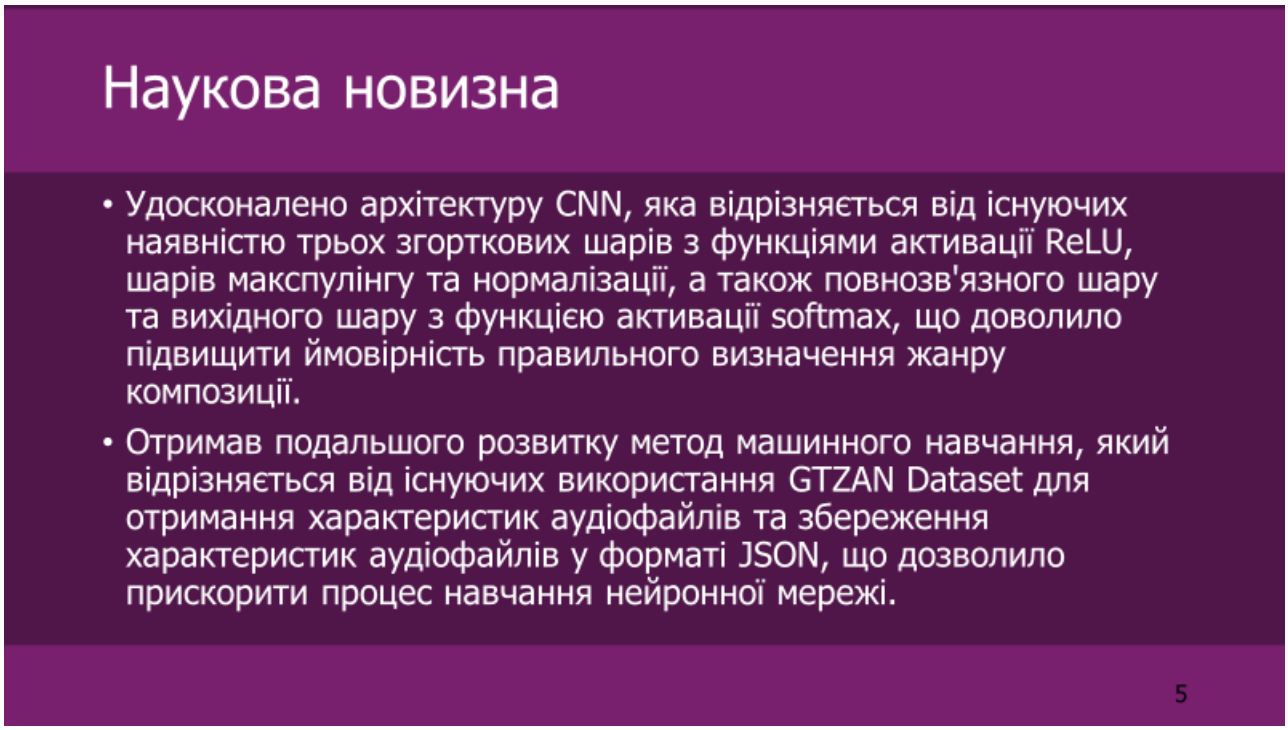


Рисунок Г.5 – Слайд презентації №5

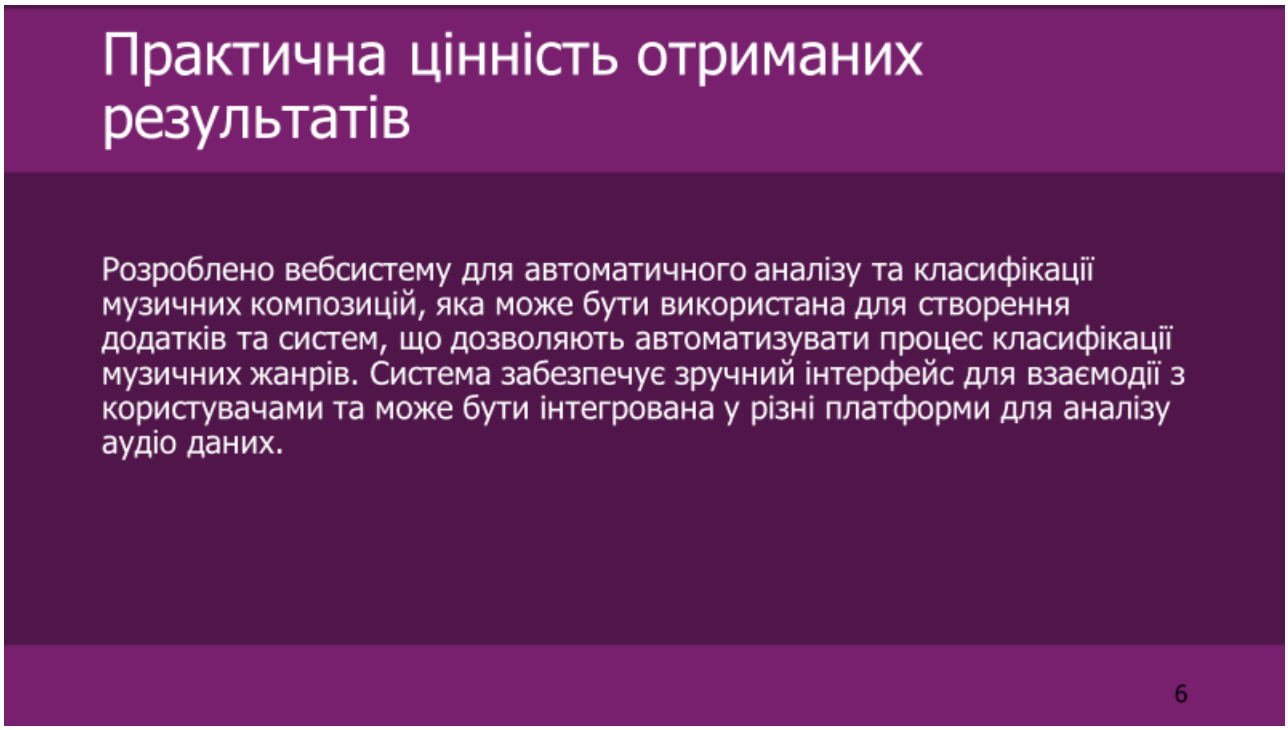


Рисунок Г.6 – Слайд презентації №6

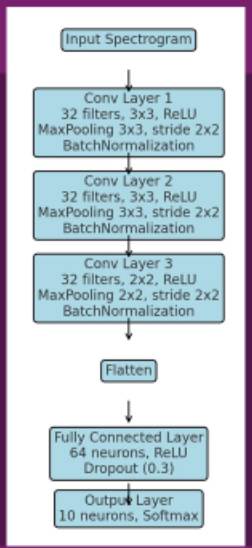
Існуючі рішення та недоліки існуючих рішень

Рішення	Недоліки
Spotify - це одна з найпопулярніших музичних платформ, що надає доступ до мільйонів треків, створення плейлистів, рекомендацій на основі прослуховувань та соціальної інтеграції.	Spotify має обмежену точність жанрової класифікації, рекомендації можуть бути не завжди точними через змішані жанри, та непрозорі алгоритми класифікації та рекомендацій.
Apple Music пропонує велику музичну бібліотеку, інтеграцію з iTunes, можливість створення плейлистів та доступ до ексклюзивного контенту.	Apple Music має обмежену кастомізацію плейлистів, непрозорі алгоритми рекомендацій та можливі проблеми з точністю визначення жанрів.
Pandora спеціалізується на радіостанціях, створених на основі музичних уподобань користувача, і використовує Music Genome Project для аналізу музики.	Pandora має обмежені можливості для безкоштовних користувачів, неповну точність жанрової класифікації та вузький вибір музики в порівнянні з іншими платформами.
Deezer пропонує велику музичну бібліотеку, функцію Flow для персоналізованих рекомендацій, інтеграцію з соціальними мережами та можливість завантаження музики для офлайн-прослуховування.	Deezer має неточну персоналізацію рекомендацій, обмежені можливості для безкоштовних користувачів та проблеми з точністю визначення жанрів.

Рисунок Г.7 – Слайд презентації №7

Вдосконалена модель CNN

Модель використовує три згорткові шари з функціями активації ReLU та макспулінгом, що забезпечує ефективне виявлення та інтерпретацію складних патернів у спектрограмах. Нормалізація на кожному етапі згортки підвищує стабільність моделі. Повнозв'язний шар з 64 нейронами та регуляризацією Dropout зменшує перенавчання, забезпечуючи більш точні передбачення. Вихідний шар з функцією активації softmax видає ймовірності для кожного з 10 жанрів, покращуючи загальну точність класифікації. Використання функції втрат categorical cross-entropy та оптимізатора Adam дозволяє ефективно налаштовувати ваги моделі, забезпечуючи швидку та точну класифікацію.

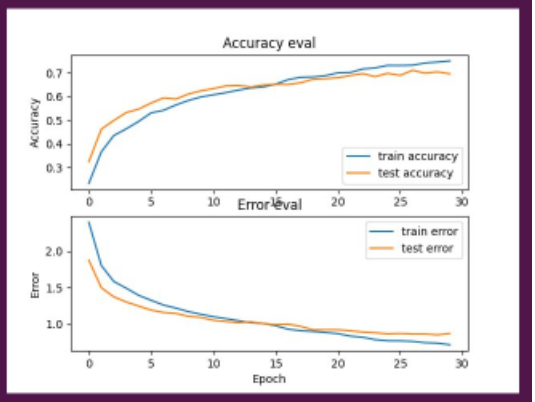


8

Рисунок Г.8 – Слайд презентації №8

Точність натренованої моделі

- Під час тренувань модель продемонструвала значний прогрес: точність тренувань поступово зросла з 0% до приблизно 75%. Точність на тестових даних досягла максимуму близько 65%, вказуючи на потребу в подальшому покращенні моделі для кращої узагальнюваності на нові дані.
- Початковий тренувальний error становив 2.8 і знизився до 0.5, що свідчить про ефективність процесу навчання. Error на тестових даних знизився з 1.8 до 0.8, вказуючи на потенціал моделі для покращення. Для досягнення кращих результатів планується використання методів регуляризації, збільшення обсягу та різноманітності тренувальних даних, а також оптимізація гіперпараметрів моделі.

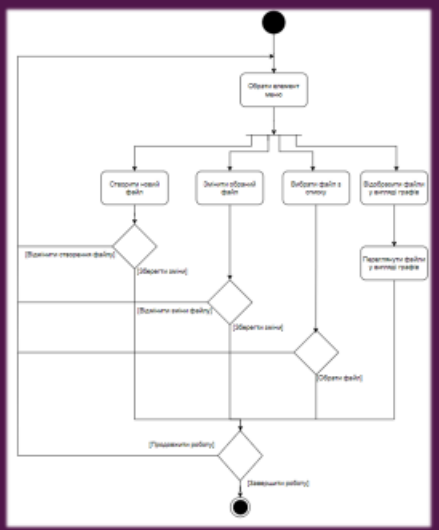


9

Рисунок Г.9 – Слайд презентації №9

Діаграма діяльності вебсистеми

Починаючи з відкриття додатку або входу до системи, діаграма показує послідовність кроків, які користувач повинен виконати для досягнення своїх цілей або виконання конкретної функції. Ці кроки можуть включати введення даних, вибір опцій, натискання кнопок та перегляд реакції системи на кожну дію користувача.



10

Рисунок Г.10 – Слайд презентації №10

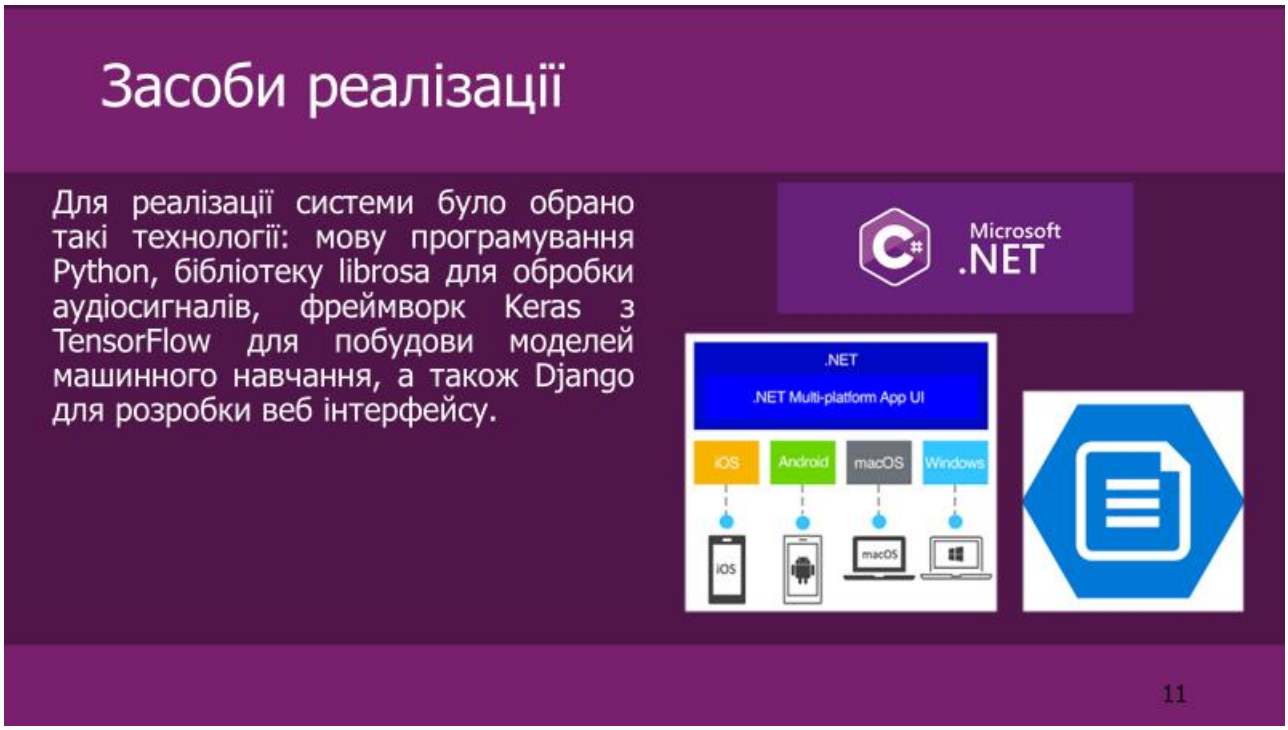


Рисунок Г.11 – Слайд презентації №11

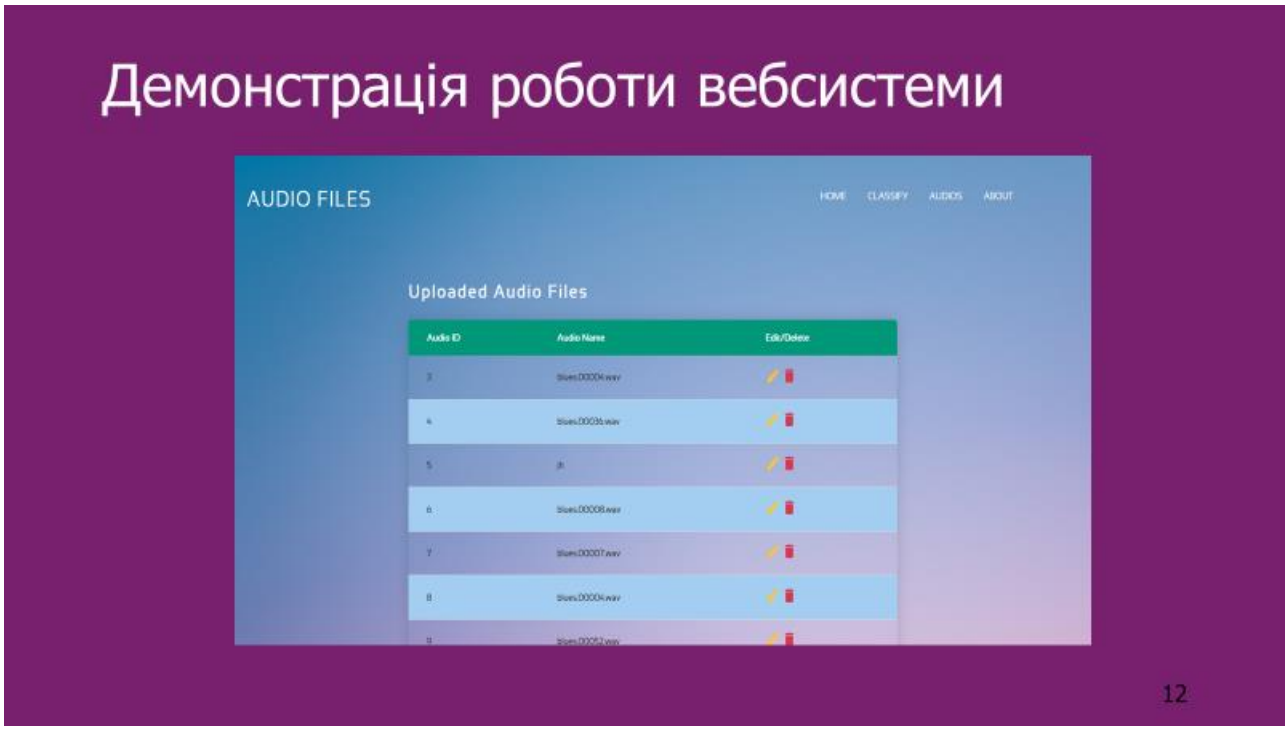


Рисунок Г.12 – Слайд презентації №12

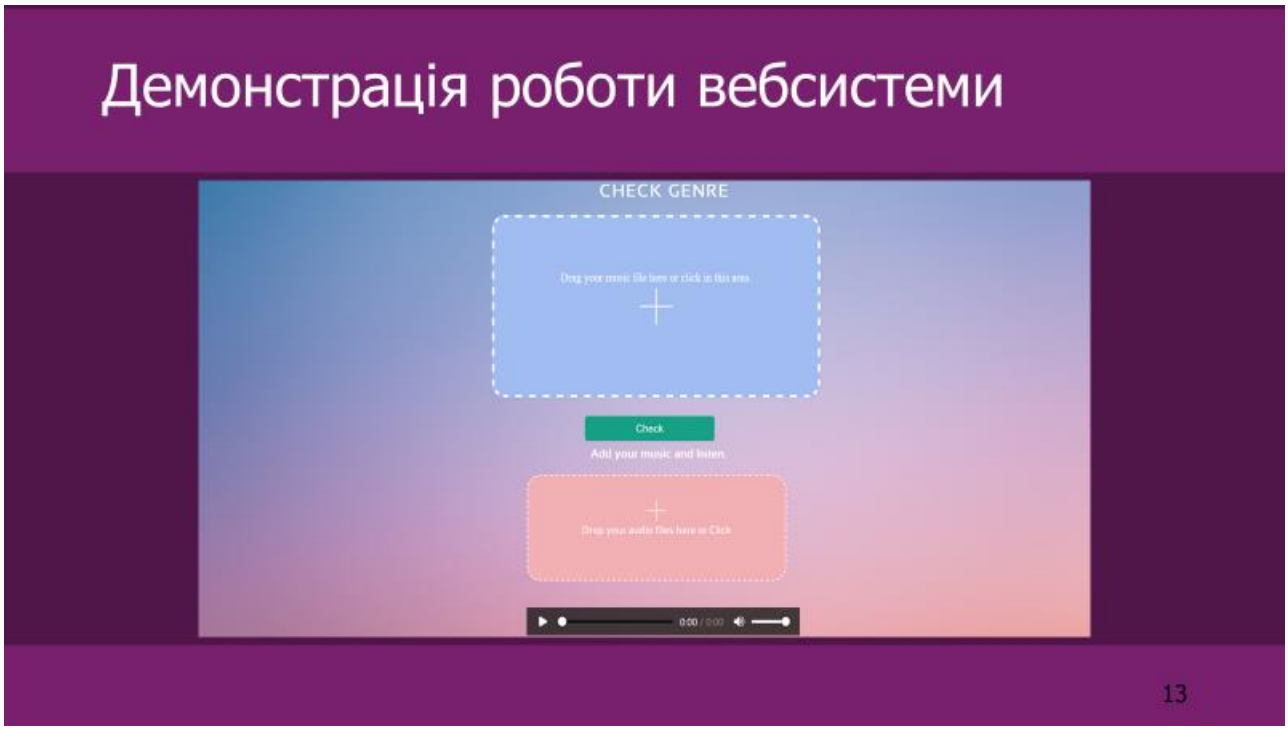


Рисунок Г.13 – Слайд презентації №13

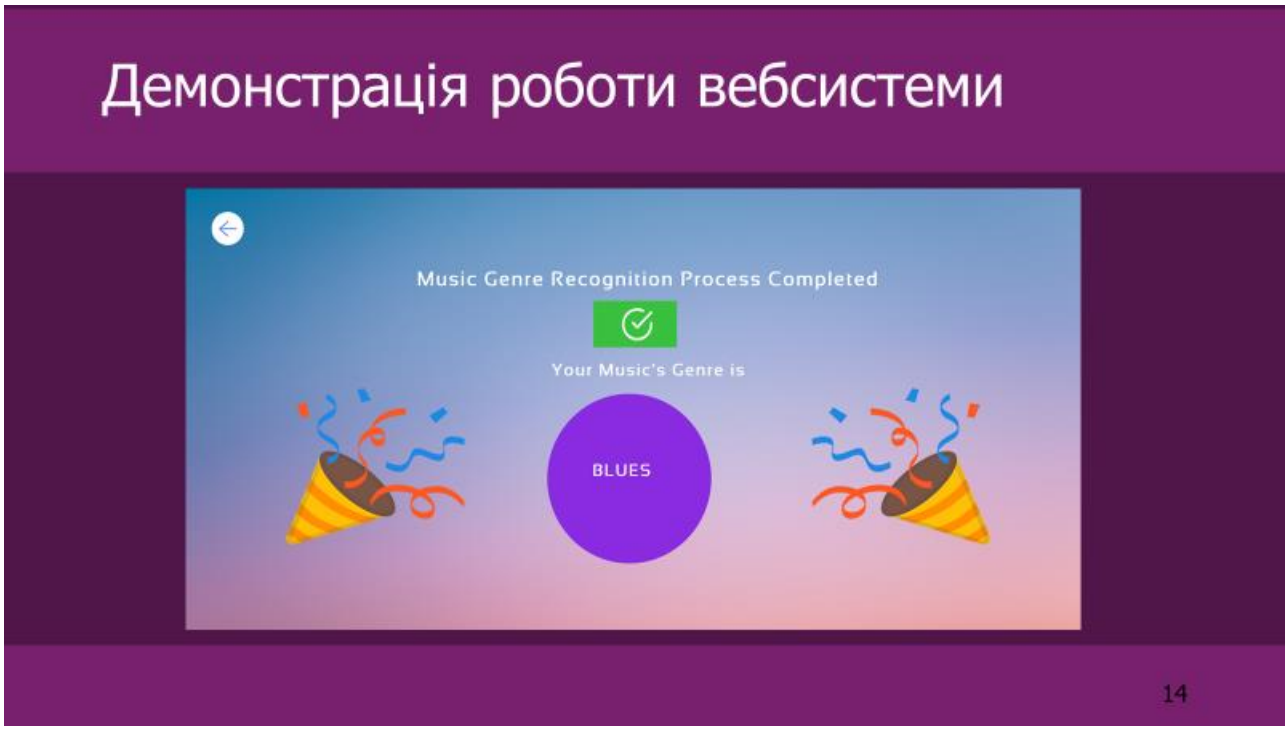


Рисунок Г.14 – Слайд презентації №14

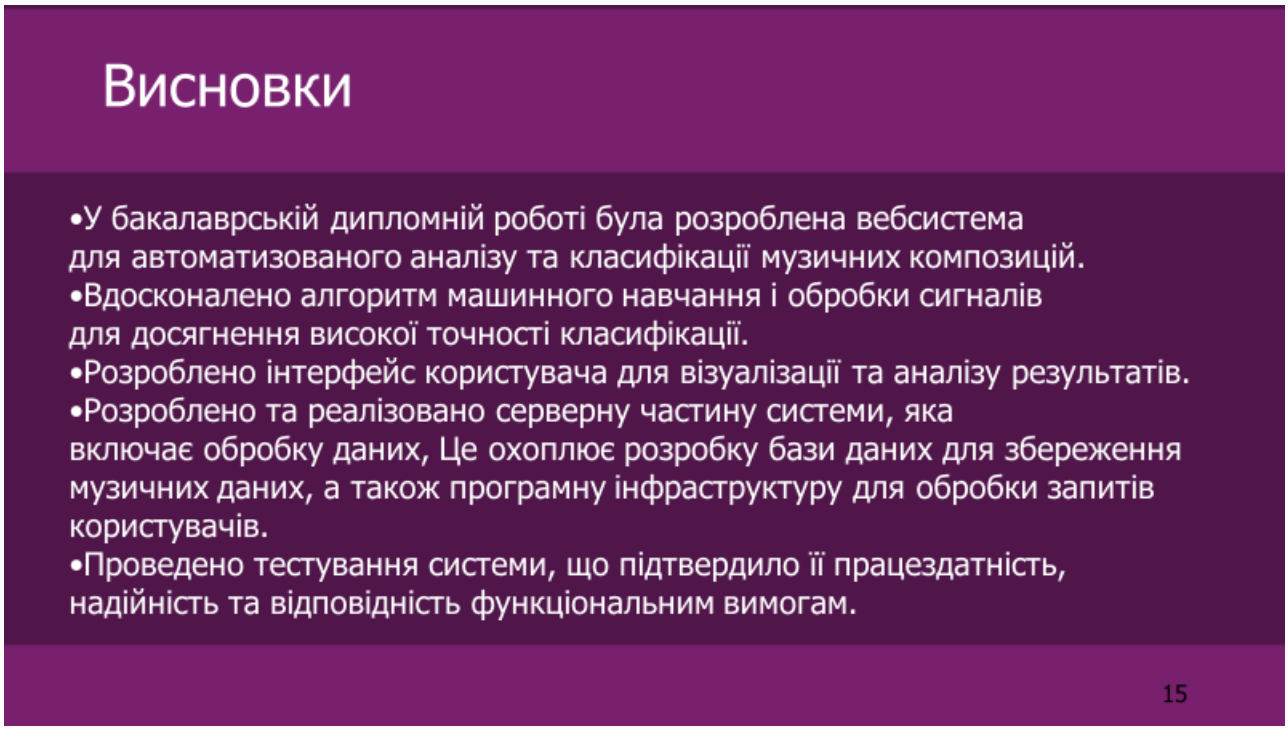


Рисунок Г.16 – Слайд презентації №16