

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

ДЛЯ ГЕНЕРАЦІЇ МУЗИЧНИХ ТВОРІВ

SOFTWARE FOR GENERATING MUSICAL WORKS

Виконав(ла): студент(ка) 4 курсу, групи КНТ-212сп

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

СИВОГРИВОВ О.Б.

(ПРИЗВИЩЕ та ініціали)

Керівник КОЦУР М.І.

(ПРИЗВИЩЕ та ініціали)

Рецензент ІЛЬЯШЕНКО М.Б.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)

Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

СИВОГРИВОВА Олександра Богдановича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення для генерації музичних творів. Software for Generating Musical Works

керівник проєкту (роботи) к.т.н., доцент, КОЦУР Михайло Ігорович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 28 ” квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 03 червня 2025 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	КОЦУР М.І., доцент		
Нормоконтроль	КАМІНСЬКА Ж.К., асистент		

7. Дата видачі завдання “ 28 ” квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Олександр СИВОГРИВОВ
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Михайло КОЦУР
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
83 с., 3 табл., 21 рис., 3 дод., 20 джерел.

РУCHARM, PYTHON, ГЕНЕРАЦІЯ МУЗИКИ, ІНФОРМАЦІЙНА СИСТЕМА, МОВА ПРОГРАМУВАННЯ, МУЗИЧНИЙ ТВІР, НЕЙРОННА МЕРЕЖА.

Об'єкт дослідження – процес розробки інформаційної системи для генерації музичних творів.

Предмет дослідження – інформаційні системи для генерації музичних творів.

Мета роботи – розробка програмного забезпечення інформаційної системи для генерації музичних творів.

Матеріали, методи та технічні засоби: мова програмування Python, середовище розробки PyCharm.

Результати. Розроблено проєктні рішення для створення програмного забезпечення інформаційної системи для генерації музичних творів. Створено програмне забезпечення інформаційної системи для генерації музичних творів. Проведено тестування розробленого програмного забезпечення інформаційної системи для генерації музичних творів.

Висновки. Мету роботи досягнуто. Розроблено програмне забезпечення інформаційної системи для генерації музичних творів за допомогою мови програмування Python та середовища розробки PyCharm.

Галузь використання – генерація музичних творів.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 83 pages, 3 tables, 21 figures, 3 appendixes, 20 sources.

PYCHARM, PYTHON, MUSIC GENERATION, INFORMATION SYSTEM, PROGRAMMING LANGUAGE, MUSICAL WORK, NEURAL NETWORK.

The object of research is the process of developing information system for generating musical works.

The subject of the research is information systems for generating musical works.

The purpose of this work is to develop the software of the information system for generating musical works.

Materials, methods and technical tools: Python programming language, PyCharm development environment.

Results. Project solutions for the creation of software for the information system for generating musical works has been designed. The software of the information system for generating musical works has been developed. The developed software of the information system for generating musical works has been tested.

Conclusions. The goal of the work has been achieved. The software tools of information system for generating musical works using the Python programming language and PyCharm development environment have been developed.

Scope of use – generation of musical works.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз предметної області	11
1.1 Інформаційні системи для генерації музичних творів	11
1.2 Аналіз інформаційних систем для генерації музичних творів	21
1.3 Висновки за розділом 1.....	31
2 Матеріали і методи	33
2.1 Вибір мови програмування.....	33
2.2 Вибір середовища розробки для створення програмного забезпечення інформаційної системи для генерації музичних творів	37
2.3 Висновки за розділом 2.....	40
3 Опис програми	41
3.1 Структура програмного забезпечення інформаційної системи для генерації музичних творів	41
3.2 Функціонування програмного забезпечення інформаційної системи для генерації музичних творів	43
3.3 Використання нейромережових моделей у програмному забезпеченні інформаційної системи для генерації музичних творів	45
3.4 Проектування інтерфейсу програмного забезпечення інформаційної системи для генерації музичних творів	51
3.5 Висновки за розділом 3.....	53
4 Експлуатація, тестування та експериментальне дослідження програми.....	54
4.1 Призначення й умови застосування програми	54
4.2 Характеристики програмного забезпечення інформаційної системи для генерації музичних творів	55
4.3 Інструкція по експлуатації програми.....	56
4.3.1 Звернення до програми.....	56
4.3.2 Вхідні й вихідні дані.....	56

4.3.3 Повідомлення.....	56
4.4 Виконання програмного забезпечення інформаційної системи для генерації музичних творів	57
4.5 Тестування програмного забезпечення інформаційної системи для генерації музичних творів	58
4.6 Висновки за розділом 4.....	58
Висновки.....	60
Перелік джерел посилань.....	63
Додаток А Технічне завдання.....	65
Додаток Б Фрагмент тексту програми	69
Додаток В Слайди презентації	77

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

LSTM – Long Short-Term Memory;

RNN – Recurrent Neural Networks;

БД – база даних;

ІС – інформаційна система;

ПЗ – програмне забезпечення;

МТ – музичний твір;

ШІ – штучний інтелект.

ВСТУП

Програмне забезпечення для генерації музичних творів стає дедалі актуальнішим завдяки стрімкому розвитку штучного інтелекту та алгоритмів машинного навчання. Сучасні нейронні мережі здатні створювати музику, що імітує певні стилі та жанри, дозволяючи автоматизувати творчий процес. Це особливо корисно для композиторів і музикантів, яким такі технології допомагають швидко генерувати ідеї та аранжування, економлячи час і ресурси [1]-[3].

Популярність інформаційних технологій, які використовують алгоритми штучного інтелекту (ШІ), постійно зростає. Програмні платформи на основі засобів ШІ роблять створення музики доступним навіть для тих, хто не має спеціальної освіти. Крім того, у сучасних цифрових аудіо-робочих станціях, все частіше інтегруються функції на основі штучного інтелекту, що розширює можливості музикантів [4], [5].

Генеративна музика також знаходить застосування у відеоіграх, віртуальних середовищах, адаптивних плейлистах та інших інтерактивних платформах, де музичний супровід може змінюватися в реальному часі відповідно до дій користувача. У комерційній сфері такі інформаційні технології дозволяють швидко створювати унікальні композиції, що значно спрощує процес створення аудіо контенту без ризику порушення авторських прав [6], [7]. Важливо також розуміти, що хоча такі технології значно розширюють можливості музичної індустрії, вони не обов'язково замінять людських композиторів, а скоріше стануть інструментом для створення нових форм мистецтва [5]-[7].

Попри значний прогрес, програмне забезпечення для генерації музичних творів має низку недоліків. Суттєвою проблемою є обмеженість варіативності та гнучкості. Багато генеративних моделей працюють за заданими параметрами, що часто призводить до одноманітності та передбачуваності музичних творів. Навіть найсучасніші інформаційні системи

на основі засобів ШІ можуть створювати композиції, які здаються механічними або надто схожими на вже існуючі твори. Крім того, оскільки генеративні алгоритми часто навчаються на великих наборах даних, що містять музику реальних композиторів, виникає ризик порушення прав інтелектуальної власності. Іншим недоліком відомих програмних засобів є використання потужних алгоритмів штучного інтелекту, що потребують великих обчислювальних потужностей, що може бути проблемою для користувачів без доступу до спеціалізованого обладнання або хмарних сервісів. Це впливає на швидкість роботи програм, особливо при генерації складних композицій з великою кількістю інструментів.

Тому актуальною є розробка інформаційної системи для генерації музичних творів. У дипломній кваліфікаційній роботі бакалавра розв'язується актуальне завдання розробки програмного забезпечення інформаційної системи для генерації музичних творів.

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та інформаційних систем для генерації музичних творів;
- здійснити проектування програмного забезпечення інформаційної системи для генерації музичних творів;
- створити програмне забезпечення інформаційної системи для генерації музичних творів;
- виконати тестування розробленого програмного забезпечення інформаційної системи для генерації музичних творів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Інформаційні системи для генерації музичних творів

Генерація музичних творів – це процес автоматичного створення музичних композицій за допомогою алгоритмів штучного інтелекту, зокрема рекурентних нейронних мереж (Recurrent Neural Networks, RNN) та їхніх вдосконалених варіацій, таких як довготривала короткочасна пам'ять (Long Short Term Memory, LSTM) та трансформери [1].

Проаналізуємо основні підходи до генерації музики. Генерація нотних послідовностей: нейромережі можуть навчатися на музичних даних у форматі MIDI, аналізуючи патерни нот, гармонійні зв'язки та ритмічні структури. Після навчання модель може передбачати наступні ноти, ґрунтуючись на попередньо зіграних фрагментах, і таким чином створювати мелодії та супровід [1].

Інший підхід полягає у використанні глибоких згорткових нейронних мереж або генеративно-змагальних мереж для синтезу звуків безпосередньо у вигляді аудіосигналів. Такі системи можуть створювати реалістичні звуки музичних інструментів або навіть голоси для вокальних партій [1].

Нейромережі можуть навчатися на музиці певних композиторів або жанрів, щоб створювати нові твори у схожому стилі. Наприклад, система може генерувати музику в стилі Моцарта, джазові імпровізації чи електронні композиції [1].

Деякі алгоритми використовують комбінацію кількох моделей для створення складніших музичних творів. Одна нейромережа може генерувати гармонію, інша – ритм, а третя – накладати ефекти чи варіювати динаміку композиції [1].

Процес генерації музики за допомогою нейронних мереж є складним і включає кілька етапів. Спочатку система навчається на великому наборі музичних даних, які можуть містити ноти, акорди, ритмічні структури та гармонійні закономірності. Це навчання дозволяє моделі розпізнавати

характерні особливості музичних стилів, розуміти взаємозв'язки між нотами та передбачати можливий розвиток композиції [1], [2].

На етапі генерації модель отримує початковий вхід, який може бути у вигляді музичної фрази, ритмічного малюнка або навіть випадкового шуму. Виходячи з цього вхідного сигналу, нейромережа формує нову мелодію, поступово додаючи ноти та акорди відповідно до навчених правил та статистичних залежностей. У деяких випадках процес включає використання алгоритмів уваги або механізмів пам'яті, що дозволяє враховувати довгострокові залежності у композиції та забезпечує більш природне звучання музики [1], [2].

Якість створеної музики значною мірою залежить від використовуваних архітектур нейромереж. Наприклад, LSTM-мережі добре підходять для аналізу та генерації послідовних даних, оскільки вони зберігають інформацію про попередні елементи музичного твору. Інший підхід передбачає використання генеративно-змагальних нейромереж, де одна модель генерує музику, а інша оцінює її якість, що сприяє покращенню кінцевого результату [1], [2].

Важливу роль у процесі генерації відіграє післяобробка. Система може виконувати корекцію помилок, уточнювати гармонію та ритм, а також адаптувати композицію до певного музичного стилю. Також можливе використання алгоритмів для додавання динаміки, нюансів виконання та варіативності у створених творах, що робить музику більш живою та виразною [1], [2].

Генерація музики за допомогою нейромереж все ще залишається експериментальною галуззю, проте вже зараз її результати знаходять застосування у створенні музики для ігор та фільмів, автоматичному компонуванні фонові музики, а також у творчій співпраці музикантів із штучним інтелектом. Розвиток цієї технології відкриває нові можливості для взаємодії людини та алгоритмів у сфері мистецтва [1], [2].

Генерація музичних творів із використанням штучного інтелекту відкриває нові можливості для створення, аналізу та обробки музичного контенту. Сучасні алгоритми дозволяють автоматизувати процес композиції, використовуючи принципи машинного навчання та глибоких нейронних мереж. Завдяки цим технологіям можливо синтезувати мелодії, гармонійні структури та навіть складні багат шарові аранжування, наближені до стилю та характеру творів, створених людиною [1], [2].

Програмне забезпечення для генерації музики базується на різноманітних методах обробки звуку та аналізу музичних послідовностей. Використання спеціалізованих моделей дозволяє не лише формувати нові композиції, а й адаптувати існуючі музичні твори, змінюючи їх структуру, ритмічні особливості або гармонійний лад. Деякі інструменти штучного інтелекту орієнтовані на імітацію певних жанрів або стилів, тоді як інші можуть створювати абсолютно нові форми музичної виразності [1], [2].

Основою роботи таких систем є поєднання алгоритмів аналізу та синтезу, що дозволяє враховувати закономірності розвитку музики, її ритмічні особливості та динамічні характеристики. Використання рекурентних нейронних мереж, таких як LSTM, дозволяє генерувати мелодії зі збереженням довготривалих музичних зв'язків, що сприяє органічності звучання. Також застосовуються трансформерні нейромережеві моделі, які забезпечують високу гнучкість та здатність до глибокого розуміння музичних структур [1], [2].

Обробка музичної інформації включає не лише її генерацію, а й автоматичне розпізнавання, аналіз та модифікацію аудіо. Деякі алгоритми можуть перетворювати вокальні партії в нотний запис або текст, що спрощує роботу над аранжуванням та редагуванням композицій. Розширені можливості технологій дозволяють використовувати штучний інтелект для автоматичного підбору супроводу, гармонізації мелодій та створення музичних ефектів, що підвищує якість кінцевого результату [1], [2].

Розвиток таких систем сприяє збагаченню музичної індустрії новими підходами до композиції, даючи змогу інтегрувати штучний інтелект у процес творчості. Інтерактивна взаємодія людини з алгоритмами створює можливості для спільного написання музичних творів, де штучний інтелект виступає в ролі помічника, здатного запропонувати унікальні музичні рішення та несподівані гармонійні варіації [1], [2].

Розвиток технологій штучного інтелекту відкриває нові можливості для створення музичних композицій. Використання програмного забезпечення дозволяє генерувати мелодії, гармонійні структури та навіть тексти пісень, враховуючи стилістичні особливості та ритмічні закономірності. Алгоритми штучного інтелекту аналізують наявні музичні зразки, визначають характерні патерни та створюють нові музичні фрагменти на їх основі [1], [2].

Для автоматизованого написання текстів пісень застосовуються спеціалізовані моделі, що здатні генерувати осмислені та зв'язні рядки відповідно до заданої тематики. Деякі алгоритми враховують ритміку й інтонацію, що дозволяє адаптувати лірику під музичний супровід. Існують системи, що синхронізують текст із мелодією, створюючи композиції, де слова та музика природно взаємодіють [1], [2].

Розроблені технології також дають змогу автоматично створювати акомпанемент, підбираючи гармонії та аранжування відповідно до заданої мелодійної лінії. Деякі системи використовують навчання на основі наявних зразків музики, щоб відтворювати характерні особливості конкретних жанрів. Такі алгоритми можуть розширювати базові мелодії, доповнюючи їх додатковими інструментальними партіями [1], [2].

Окремий напрямок розвитку пов'язаний із синтезом вокалу. Моделі глибокого навчання дозволяють імітувати спів, відтворюючи вокальні тембри з високою точністю. Використовуючи складні алгоритми обробки аудіо сигналів, штучний інтелект створює реалістичні голосові партії, що можуть інтегруватися в автоматично згенеровані музичні композиції [1], [2].

Такий підхід до створення музичних творів дає змогу значно спростити процес написання пісень та експериментувати з новими формами музичного вираження. Автоматизовані системи стають не лише інструментами для створення композицій, а й платформами для творчих експериментів, що розширюють можливості як професійних музикантів, так і аматорів [1], [2].

Розвиток технологій штучного інтелекту відкрив можливості для автоматизованого створення музичних композицій. Завдяки алгоритмам машинного навчання та нейронним мережам стає можливим генерування мелодій, гармоній і текстів пісень без прямого втручання людини. Аналізуючи великі обсяги даних, такі системи здатні навчатися стилям і особливостям музичних жанрів, що дозволяє їм створювати новий контент або адаптувати вже існуючий відповідно до заданих параметрів [1], [2].

Використання таких підходів передбачає роботу зі структурою музичних творів, включаючи послідовності нот, акордові прогресії, ритмічні патерни та ліричний зміст. Деякі інформаційні технології орієнтуються на створення музичних фрагментів, які можуть бути інтегровані у більш складні композиції, тоді як інші націлені на повноцінну генерацію твору від початку до кінця [1], [2].

Алгоритми, що застосовуються для цієї мети, можуть включати рекурентні нейронні мережі, здатні прогнозувати розвиток мелодійних ліній, або генеративно-змагальні мережі, що дозволяють створювати варіативні музичні структури. Крім того, методи глибокого навчання допомагають моделювати особливості виконання та стилю, забезпечуючи природність і виразність звучання. Завдяки цьому штучний інтелект не лише аналізує вже створені музичні твори, а й пропонує власні варіації, що можуть слугувати основою для нових композицій [1], [2].

Окремим напрямом є синтез вокалу, що дає змогу імітувати спів із заданими характеристиками, такими як тембр, висота звуку та інтонація. Це дозволяє створювати вокальні партії для пісень, зберігаючи реалістичність звучання. Крім того, системи генерації акомпанементу аналізують основну

мелодію та доповнюють її гармонічним супроводом, створюючи повноцінний музичний твір [1], [2].

Автоматизоване створення музики відкриває нові перспективи в галузі композиторської діяльності, дозволяючи експериментувати зі звуком, стилями та формами. Застосування таких технологій сприяє розширенню творчих можливостей, а також змінює підходи до музичного виробництва та споживання контенту [1], [2].

Використання штучного інтелекту у створенні музичних композицій бере початок із досліджень, спрямованих на алгоритмічний підхід до написання музики. Перші експерименти базувалися на застосуванні математичних моделей та обчислювальних методів для автоматизації процесу композиції. Ідея використання комп'ютерів для музичної творчості поступово розвивалася, а з появою цифрових технологій з'явилися можливості для інтеграції алгоритмів у роботу з аудіо інструментами [2], [3].

Значний поштовх у цьому напрямку відбувся із запровадженням цифрових протоколів, що дозволили синхронізувати електронні пристрої та музичне програмне забезпечення. Це призвело до створення перших інтелектуальних систем, здатних аналізувати музичні структури та генерувати власні варіації мелодій на основі наявних зразків. Подальші дослідження привели до розробки алгоритмів, що могли не лише наслідувати відомі музичні стилі, а й створювати оригінальні композиції [2], [3].

Розвиток штучних нейронних мереж і методів глибокого навчання значно розширив можливості музичних систем, які почали використовувати великі масиви даних для аналізу та прогнозування музичних закономірностей. Новітні моделі дозволяють автоматично підбирати гармонії, створювати ритмічні структури та навіть адаптувати музику відповідно до емоційного забарвлення чи жанрових особливостей [2], [3].

Сучасне програмне забезпечення, засноване на штучному інтелекті, пропонує композиторам та виконавцям широкий спектр інструментів для створення та обробки музики. Завдяки машинному навчанню такі системи

можуть генерувати цілісні композиції, редагувати музичний матеріал у реальному часі та надавати рекомендації щодо аранжування. Це відкриває нові перспективи у сфері музичного виробництва, поєднуючи автоматизацію з творчими процесами [2], [3].

Інтеграція штучного інтелекту в музичне мистецтво змінює традиційні підходи до створення мелодій, ритмів та гармонійних структур. Використання таких технологій дозволяє значно спростити робочий процес і водночас експериментувати з новими формами звучання. Подальший розвиток цієї галузі передбачає ще більше можливостей для взаємодії людини та штучного інтелекту в музичному середовищі [2], [3].

Використання алгоритмів та спеціалізованого програмного забезпечення значно розширило можливості створення музичних композицій. Завдяки сучасним технологіям стало можливим генерування мелодій, що відповідають заданим стилям і музичним характеристикам, без прямого втручання композитора. Процес побудований на аналізі великої кількості музичних творів, що дозволяє програмам виявляти закономірності та синтезувати нові композиції з урахуванням ритмічної структури, гармонії та інструментального оформлення [3], [4].

Застосування методів глибокого навчання та нейронних мереж дало змогу програмним засобам адаптуватися до різних жанрів, експериментувати зі звучанням та навіть враховувати емоційну складову музики. Завдяки розробленим системам можливо не лише відтворювати характерні риси класичних і сучасних стилів, а й створювати абсолютно унікальні твори, які раніше вимагали б значного часу та зусиль з боку людини [3], [4].

Велика кількість музичних платформ надає інструменти для автоматизованої композиції, дозволяючи композиторам та продюсерам інтегрувати згенеровані фрагменти у власні проєкти. Деякі сервіси пропонують можливість гнучкого налаштування параметрів звучання, тоді як інші використовують складні алгоритми для створення музики, що імітує стиль конкретних виконавців. Такі технології відкривають нові перспективи

для розвитку музичної індустрії, розширюючи межі творчості та автоматизуючи рутинні процеси [3], [4].

Розвиток технологій на основі штучного інтелекту суттєво впливає на процес створення музичних творів, змінюючи звичні уявлення про творчість. Традиційний підхід, що ґрунтувався на досвіді, натхненні та майстерності музикантів, поступово доповнюється алгоритмічними методами, які здатні генерувати гармонії, ритмічні структури та навіть тексти пісень. Використання таких технологій викликає дискусії щодо їхнього впливу на оригінальність музичних композицій, але водночас розширює межі експериментів і дає змогу шукати нові форми самовираження [3], [4].

Застосування автоматизованих рішень відкриває можливості для тих, хто не має спеціальних знань у музичній сфері, дозволяючи створювати композиції навіть без глибокого розуміння теорії. Для досвідчених митців подібні інструменти стають способом оптимізації робочого процесу, звільняючи від рутинних завдань та надаючи більше часу для творчого пошуку. Це сприяє ширшій доступності музичного мистецтва, дозволяючи будь-кому, хто володіє необхідними технічними засобами, долучитися до процесу створення мелодій та аранжувань [3], [4].

Разом із цим виникають нові виклики у сфері розподілу прав на музику, зокрема питання щодо авторства та виплати винагороди. Оскільки композиції можуть створюватися без безпосередньої участі людини, актуальними стають питання щодо справедливого розподілу роялті між розробниками алгоритмів і тими, хто використовує їх для музичної творчості. Водночас технологічний прогрес відкриває додаткові джерела доходу для митців, які можуть застосовувати штучний інтелект у своїй роботі, створювати персоналізовані алгоритми або ліцензувати власні музичні моделі [3], [4].

Зміни торкнулися й виконавської сфери, де алгоритмічні рішення можуть доповнювати живі виступи, роблячи їх більш інтерактивними. Генеровані в реальному часі звуки, автоматична імпровізація та адаптивні ефекти відкривають нові можливості. Однак паралельно виникає питання

щодо автентичності подібних виступів і збереження ролі людського фактора в мистецтві. Це створює необхідність пошуку нових підходів до взаємодії між технологіями та виконавською майстерністю, де інтелектуальні алгоритми слугуватимуть не заміною, а інструментом для розширення виразних можливостей артиста [4], [5].

Алгоритми глибокого навчання, що лежать в основі сучасних музичних генераторів, значно вдосконалили здатність штучного інтелекту створювати повноцінні композиції. Якщо раніше програмне забезпечення могло лише імітувати прості мотиви, то сучасні моделі аналізують великі масиви даних, розпізнають закономірності та на їхній основі формують гармонійно та структурно завершені музичні твори. Завдяки цьому можливості автоматизованої композиції значно розширюються, дозволяючи створювати музичні полотна, які в багатьох випадках важко відрізнити від тих, що були написані людьми [4], [5].

Розвиток технологій штучного інтелекту значно вплинув на процес створення музичних композицій. Традиційний підхід, заснований на особистому досвіді, емоціях та культурних впливах, поступово доповнюється автоматизованими методами, які дозволяють створювати складні мелодії без глибокого знання музичної теорії. Програмне забезпечення, що працює на основі алгоритмів машинного навчання, здатне аналізувати великі обсяги музичних даних, знаходити закономірності та пропонувати варіанти мелодій і гармоній, що відповідають заданому стилю [5], [6].

Технологічні інструменти для генерації музики стали доступними не лише професійним композиторам, а й початківцям, які прагнуть експериментувати. Це дозволяє значно прискорити процес створення нових треків, автоматизувати повторювані завдання та запропонувати нові творчі рішення. Водночас застосування ШІ у музиці викликає дискусії щодо авторського права, оскільки алгоритми базуються на аналізі існуючих композицій, що може поставити під сумнів оригінальність отриманих результатів [5], [6].

Використання штучного інтелекту у музичній індустрії змінює підходи до виконання та живих виступів. Технології дозволяють інтегрувати алгоритмічну імпровізацію, автоматизоване аранжування та адаптивні аудіо ефекти, створюючи нові формати взаємодії між виконавцем і слухачем. Проте це також породжує запитання щодо ролі людини у творчому процесі, адже зменшення необхідності безпосереднього виконання може вплинути на сприйняття автентичності музичного мистецтва [5], [6].

Значний вплив ШІ на музичну сферу також породжує юридичні та етичні питання. Відсутність чіткого розподілу прав на створені алгоритмами композиції може призвести до конфліктів щодо авторства та роялті. Незрозуміло, чи слід вважати власниками таких творів розробників програмного забезпечення, виконавців, які використовують ці інструменти, чи самі алгоритми не можуть мати жодних прав на результат. Окрім цього, існує ризик витіснення людських музикантів, оскільки автоматизовані системи можуть створювати музику швидше й дешевше [5], [6].

Попри всі суперечності, використання ШІ у музичній творчості відкриває нові перспективи, дозволяючи розширювати межі можливого та поєднувати традиційний підхід із сучасними технологіями. Здатність алгоритмів аналізувати та синтезувати музичні стилі дає змогу експериментувати з новими формами звучання, що може сприяти розвитку інноваційних жанрів і тенденцій [5], [6].

Розвиток штучного інтелекту дедалі більше впливає на процес створення музики, змінюючи традиційні уявлення про композицію. Технологічні рішення відкривають можливість для автоматизованого формування мелодій, ритмічних структур і гармоній, що може призвести до появи нових музичних напрямів. Водночас ці зміни викликають дискусії щодо значення людського фактору у творчому процесі, адже цифрові алгоритми поступово освоюють методи, які раніше вважалися виключно прерогативою музикантів [6], [7].

Крім безпосереднього впливу на написання музики, штучний інтелект також трансформує механізми її поширення та сприйняття. Сучасні інформаційні технології аналізують індивідуальні вподобання слухачів і пропонують контент, який найбільше відповідає їхнім смакам. Надалі такі технології здатні забезпечити ще глибшу персоналізацію музичного досвіду, спрощуючи доступ до творів і сприяючи їх ефективнішому розповсюдженню серед цільової аудиторії [6], [7].

Розширюється і сфера застосування штучного інтелекту в навчанні музиці. Інтерактивні цифрові платформи аналізують виконання, дають миттєвий зворотний зв'язок і рекомендують подальші кроки для покращення навичок. Це дозволяє оптимізувати навчальний процес, роблячи його більш доступним та ефективним, а також створює можливості для інтеграції новітніх підходів у розвиток музичних здібностей [6], [7].

Визначено, що генерація музичних творів – це процес автоматичного створення музичних композицій за допомогою алгоритмів штучного інтелекту, зокрема рекурентних нейронних мереж та їхніх вдосконалених варіацій. Проаналізовано основні підходи до генерації музики. Процес генерації музики за допомогою нейронних мереж є складним і включає кілька етапів. Спочатку система навчається на великому наборі музичних даних, які можуть містити ноти, акорди, ритмічні структури та гармонійні закономірності. Це навчання дозволяє моделі розпізнавати характерні особливості музичних стилів, розуміти взаємозв'язки між нотами та передбачати можливий розвиток композиції.

1.2 Аналіз інформаційних систем для генерації музичних творів

Проаналізуємо інформаційні системи для генерації музичних творів [8]-[14].

Програмне забезпечення Hydra (рис. 1.1) є вдосконаленою системою генерації музичних творів, що використовує алгоритми штучного інтелекту.

Ця технологія, заснована на попередній версії, отримала значні покращення та пропонує широкий набір інструментів для створення індивідуалізованої музики. Завдяки навчанню на великій базі даних, яка містить численну кількість композицій, програмне забезпечення Hydra надає можливість генерувати високоякісні музичні твори та звукові ефекти, адаптовані до потреб користувачів [8], [9], [12].

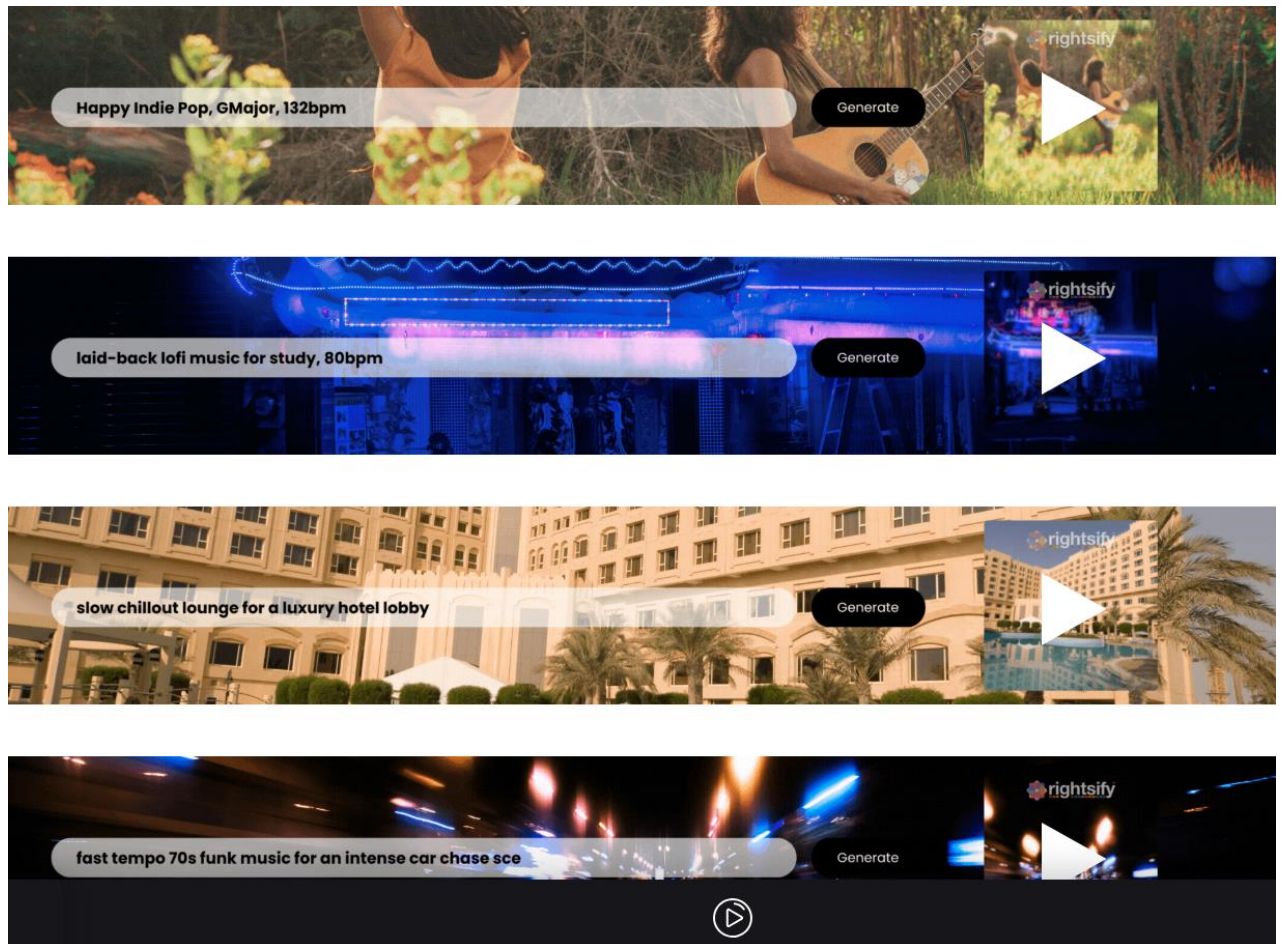


Рисунок 1.1 – Програмне забезпечення Hydra [12]

Програмне забезпечення Hydra підтримує велику кількість інструментів, що представляють різні музичні культури, а також забезпечує багатомовний інтерфейс, що робить її доступною для глобального використання. Інтеграція нових можливостей редагування дозволяє створювати унікальні аранжування та налаштовувати композиції відповідно до конкретних вимог. Завдяки цьому програмне забезпечення відкриває нові

перспективи для комерційного використання, зокрема для музичних продюсерів, маркетологів та творців контенту [8], [9], [12].

Серед функціональних можливостей програмного забезпечення Hydra можна відзначити інструмент Текст у музику, який дає змогу автоматично генерувати інструментальні композиції на основі введених текстових описів. При цьому створення вокальних партій не передбачене, що дозволяє уникнути етичних і правових суперечностей, пов'язаних із відтворенням голосу. Простий та зручний інтерфейс значно полегшує процес створення музики, усуваючи технічні бар'єри для користувачів. Доступність різних варіантів використання програмного забезпечення Hydra, включно з пробним періодом, робить цю платформу привабливою як для індивідуальних музикантів, так і для великих компаній [8], [9], [12].

Основними характеристиками програмного забезпечення Hydra є такі [8], [9], [12]:

- основне призначення: генерація інструментальної музики та звукових ефектів на основі штучного інтелекту для комерційного та творчого використання [8], [9], [12];

- база даних: навчена на понад 1 мільйоні композицій, що забезпечує різноманітність стилів та жанрів [8], [9], [12];

- функція «Текст у музику»: програмне забезпечення Hydra дозволяє перетворювати текстові описи в музичні композиції, що спрощує створення контенту [8], [9], [12];

- відсутність вокалу: програмне забезпечення Hydra не генерує вокальні партії для запобігання проблемам із правами на зміст [8], [9], [12];

- багатий вибір інструментів: програмне забезпечення Hydra містить понад 800 музичних інструментів з усього світу, що розширює можливості для створення унікального звучання [8], [9], [12];

- багатомовність: програмне забезпечення Hydra підтримує понад п'ятдесят мов, що робить його доступним для широкої міжнародної аудиторії користувачів [8], [9], [12];

- інструменти редагування: програмне забезпечення Hydra пропонує нові функції для глибокого налаштування музичних треків відповідно до потреб користувача [8], [9], [12];

- спрощене ліцензування: Hydra забезпечує легальний доступ до згенерованої музики без складних юридичних процедур [8], [9], [12];

- доступність: програмне забезпечення Hydra пропонує декілька варіантів використання, включаючи безкоштовну пробну версію, що дозволяє протестувати можливості перед покупкою [8], [9], [12].

Серед недоліків програмного забезпечення Hydra можна відзначити обмеженість у створенні вокальних партій, що може стати перепорою для тих, хто шукає повноцінні музичні композиції з вокалом. Також, незважаючи на широкий набір інструментів та гнучкість налаштувань, система може не завжди точно відтворювати творчий задум користувача, оскільки генеровані треки залежать від алгоритмів та вихідних даних. Ще одним потенційним недоліком є можливі обмеження у стилях та жанрах, оскільки штучний інтелект може краще працювати з певними музичними напрямками, тоді як інші можуть звучати менш автентично. Крім того, використання програмного забезпечення може потребувати ліцензування, що створює додаткові фінансові витрати для комерційних проєктів [8], [9], [12].

Програмне забезпечення AIVA (рис. 1.2) використовує штучний інтелект для створення музичних композицій, надаючи можливість генерувати складні гармонії та мелодії у різноманітних жанрах. Ця система розроблена для музикантів і творців контенту, які шукають допомогу в автоматизованому створенні музики, зберігаючи при цьому контроль над процесом. Алгоритми програмного забезпечення AIVA дозволяють налаштовувати численні аспекти музичних творів, включаючи темп, вибір інструментів і загальний стиль композиції. Завдяки своїм можливостям цей інструмент може генерувати кілька варіантів одного музичного твору, що дає змогу обрати найкращі фрагменти й комбінувати їх у фінальний продукт [8]-[10], [13].

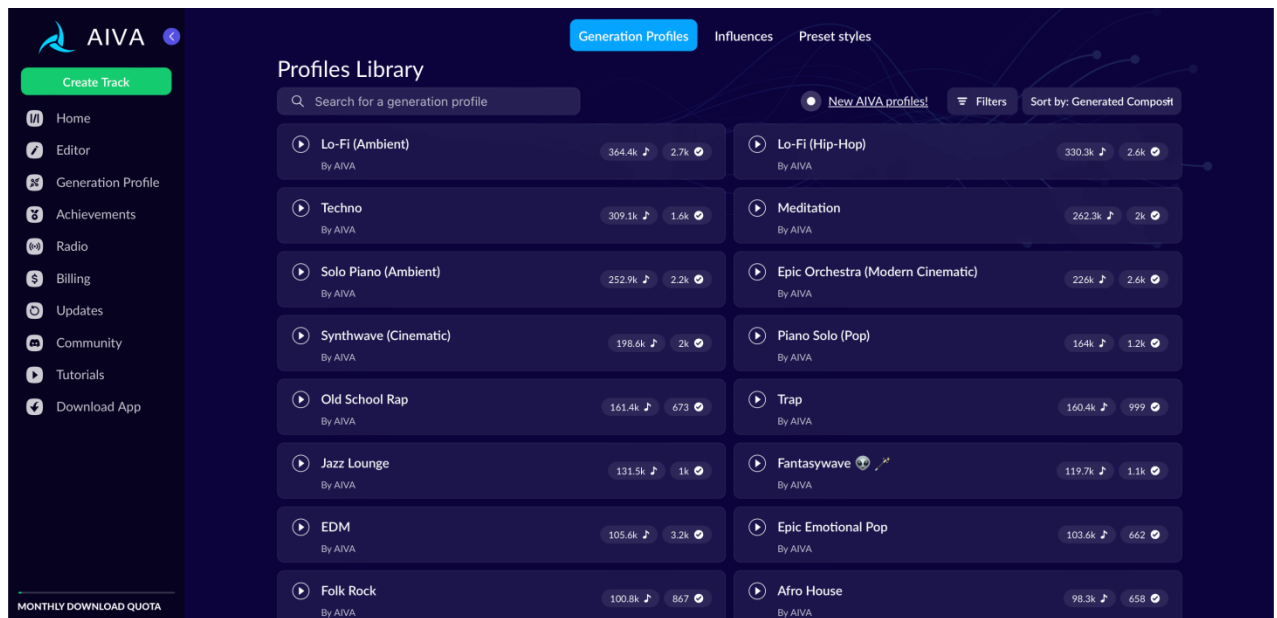


Рисунок 1.2 – Програмне забезпечення AIVA [13]

Інтерфейс програмного забезпечення AIVA орієнтований на професійне використання, тому для новачків може знадобитися певний час, щоб опанувати всі функції. Однак для досвідчених користувачів програмне забезпечення AIVA відкриває широкий спектр можливостей для творчості та експериментів [8], [9], [13].

Ця система здатна створювати музичні твори для реклами, фільмів і відеоігор, а також пропонує можливість змінювати існуючі музичні твори. Однією з головних особливостей є підтримка різних стилів, що дає змогу швидко адаптувати музику під конкретні потреби проєкту. AIVA була відзначена завдяки своїм класичним композиціям, що відтворюють характерні особливості традиційної академічної музики [8], [9], [13].

Програмне забезпечення AIVA пропонує набір попередньо встановлених стилів, що спрощує процес створення музики, а також містить функції редагування та адаптації вже наявних доріжок. Крім цього, програмне забезпечення AIVA забезпечує простий механізм ліцензування, що дозволяє використовувати згенеровану музику без юридичних складнощів. Завдяки своїм можливостям програмне забезпечення AIVA стало корисним для

композиторів, оркестрів і шанувальників класичної музики, які прагнуть отримати високоякісні оригінальні твори у витонченому стилі [8]-[10], [13].

Програмне забезпечення AIVA має такі особливості [8]-[10], [13]:

- генерація класичних композицій: програмне забезпечення спеціалізується на створенні музики в стилі класичних творів, використовуючи алгоритми штучного інтелекту для імітації гармонійної структури та стилістичних особливостей композиторів минулого [8], [9], [13];

- настроювані параметри: користувачі можуть впливати на процес створення музики, налаштовуючи темп, тональність, стиль і набір інструментів відповідно до власних потреб [8], [10], [13];

- створення варіацій: можливість генерувати кілька варіантів однієї композиції та комбінувати найкращі елементи для отримання унікального результату [8], [9], [13];

- інтеграція з музичними редакторами: підтримка експорту музичних творів у популярних форматах для подальшого редагування у професійних цифрових аудіостанціях [8], [9], [13];

- автоматизоване аранжування: алгоритми можуть самостійно адаптувати мелодії, додаючи супровідні партії та розширюючи гармонічну структуру [8], [9], [13];

- підтримка різних жанрів: хоча основний фокус зосереджено на класичній музиці, система дозволяє створювати композиції у кінематографічному, оркестровому та інших жанрах [8], [9], [13];

- інтуїтивний інтерфейс: попри професійні можливості, програмне забезпечення пропонує зручну навігацію та доступ до основних функцій навіть для користувачів без глибоких знань музичної теорії [8], [9], [13];

- ліцензування музичних творів: створена музика може використовуватися без необхідності врегулювання авторських прав, що полегшує її застосування в комерційних проектах [8], [10], [13].

Серед недоліків програмного забезпечення AIVA можна відзначити обмеженість творчої унікальності, оскільки алгоритми ШІ часто генерують

композиції, що мають схожі гармонійні та ритмічні структури. Попри широкий спектр налаштувань, індивідуальне втручання у процес створення музики все ще залишається менш гнучким порівняно з ручною композицією. Інтерфейс програми, орієнтований на професійних користувачів, може виявитися складним для початківців, що ускладнює освоєння всіх можливостей системи. Крім того, безкоштовна версія має обмежений функціонал, що звужує можливості редагування та експорту музичних творів без додаткової оплати. Ще одним недоліком є те, що AIVA зосереджена переважно на класичних і кінематографічних жанрах, тоді як для створення сучасної електронної чи популярної музики її можливості можуть бути менш ефективними [8]-[10], [13].

Програмне забезпечення Loudly (рис. 1.3) використовує штучний інтелект для автоматичної генерації музичних композицій. Основна мета програмного забезпечення Loudly полягає у наданні творцям контенту простого і швидкого інструменту для створення фонові музики, яка може використовуватися користувачами у відео, подкастах чи інших мультимедійних проєктах [8]-[11], [14].

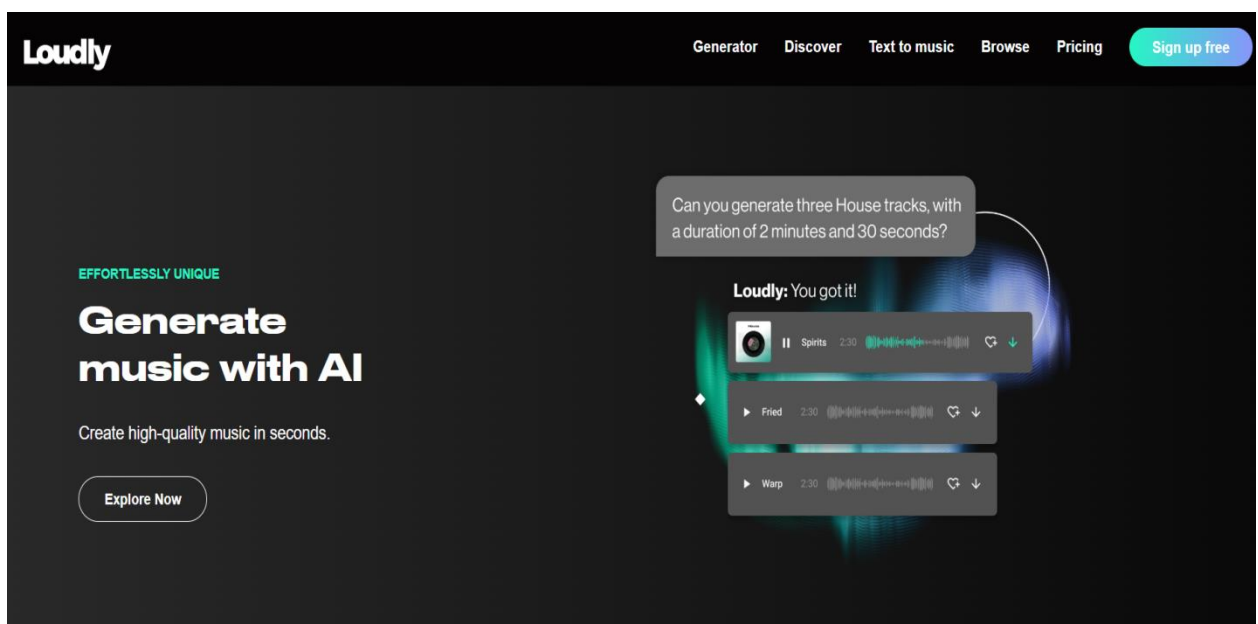


Рисунок 1.3 – Програмне забезпечення Loudly [14]

Програмне забезпечення Loudly пропонує певний рівень керування процесом створення музики. Користувач може вибирати жанр, темп, інструментальний склад і налаштовувати інтенсивність окремих елементів. Водночас гнучкість у редагуванні обмежена, оскільки глибокого налаштування аудіо, як у професійних цифрових аудіо станціях, не передбачено [8]-[11], [14].

Один із ключових аспектів Loudly є простий та інтуїтивний інтерфейс. Навігація реалізована так, що навіть користувач без досвіду у створенні музики зможе легко зорієнтуватися і отримати готовий трек. Це робить програму доступною для широкої аудиторії [8]-[11], [14].

Loudly також дозволяє експериментувати з музикою, змінювати структуру композицій, додавати нові інструменти та навіть включати власні аудіо файли. Проте можливості редагування обмежені, якщо результат не влаштовує, часто доводиться починати процес створення заново. Загалом, цей інструмент підходить для тих, хто шукає швидкий спосіб отримати базові музичні композиції без глибокого занурення в процес аранжування. Він добре виконує свою функцію для створення фонові музики, але для професійної роботи з аудіо може здатися недостатньо гнучким [8]-[11], [14].

Програмне забезпечення Loudly має такі особливості [8]-[11], [14]:

- музичний генератор на основі ІІІ: програмне забезпечення Loudly дозволяє автоматично створювати музичні треки, адаптовані до різних жанрів і настроїв [8]-[11], [14];
- зручний інтерфейс: простий та інтуїтивно зрозумілий дизайн, який забезпечує легкий доступ до основних функцій [8]-[11], [14];
- налаштування параметрів: користувач програмного забезпечення Loudly може вибирати жанр, темп, інструментальний склад та інші характеристики композиції [8]-[11], [14];
- відсутність детального редагування: програмне забезпечення Loudly не дає можливості вносити точкові зміни у згенеровані треки або додавати власні звукові ефекти [8]-[11], [14];

- залежність від Інтернету: програмне забезпечення Loudly працює лише в онлайн режимі, що обмежує його використання без підключення до мережі [8]-[11], [14];

- можливість створення реміксів: програмне забезпечення Loudly дозволяє маніпулювати окремими елементами треку, такими як вокал, барабани чи інструменти [8]-[11], [14];

- автоматизація процесу: користувач програмного забезпечення Loudly отримує готові музичні треки без необхідності володіння професійними навичками аранжування [8]-[11], [14];

- відсутність професійних функцій: немає інструментів для мікшування, накладання ефектів або корекції аудіо, що обмежує можливості досвідчених музикантів [8]-[11], [14];

- використання готових шаблонів: програмне забезпечення Loudly пропонує бібліотеку безкоштовних музичних шаблонів для швидкого створення композицій [8]-[11], [14];

- швидке генерування музики: програмне забезпечення Loudly дозволяє отримати готовий трек всього за кілька натискань миші, що робить його зручним для створення музичних творів [8]-[11], [14].

Серед недоліків програмного забезпечення Loudly можна відзначити обмежені можливості редагування музичних треків. Користувач не має змоги суттєво змінювати вже згенеровані композиції, окрім базових налаштувань жанру, темпу та інструментального складу. Якщо отриманий результат не відповідає очікуванням, доводиться починати процес створення з нуля, оскільки функції детального редагування чи внесення точкових змін відсутні. Ще одним обмеженням є необхідність постійного підключення до Інтернету. Програма не передбачає автономного режиму роботи, що може бути незручним для користувачів, які хочуть створювати музику без доступу до мережі. Також варто зазначити, що згенеровані композиції часто мають стандартне звучання без глибокої індивідуальності. Незважаючи на широкий вибір жанрів і настроїв, треки можуть здаватися схожими між собою та не

завжди відповідати унікальним творчим задумам користувача. Крім того, Loudly не пропонує інструментів для повноцінної роботи зі звуком, таких як мікшування, накладання ефектів або корекція аудіо, що є стандартними можливостями у професійному програмному забезпеченні для створення музики. Це робить його менш привабливим для досвідчених музикантів та продюсерів, які потребують більшої гнучкості у створенні та редагуванні композицій [8]-[11], [14].

Порівняльну характеристику інформаційних систем для генерації музичних творів наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння інформаційних систем для генерації музичних творів

Критерій порівняння	Hydra [12]	AIVA [13]	Loudly [14]
Можливість редагування музичних творів треків	+	+—	—
Зручність інтерфейсу	+—	+—	+
Якість створених музичних композицій	+—	+	+—
Доступність у безкоштовній версії	—	+—	+
Підходить для професійного використання	+	+—	—
Робота в офлайн режимі	+	+	—

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато інформаційних систем для генерації музичних творів. Проте існуюче програмне забезпечення для генерації музичних творів має низку недоліків. Суттєвою проблемою є обмеженість варіативності та гнучкості. Багато генеративних моделей працюють за заданими параметрами,

що часто призводить до одноманітності та передбачуваності музичних творів. Навіть найсучасніші інформаційні системи на основі засобів ШІ можуть створювати композиції, які здаються механічними або надто схожими на вже існуючі твори. Крім того, оскільки генеративні алгоритми часто навчаються на великих наборах даних, що містять музику реальних композиторів, виникає ризик порушення прав інтелектуальної власності. Іншим недоліком відомих програмних засобів є використання потужних алгоритмів штучного інтелекту, що потребують великих обчислювальних потужностей, що може бути проблемою для користувачів без доступу до спеціалізованого обладнання або хмарних сервісів. Це впливає на швидкість роботи програм, особливо при генерації складних композицій з великою кількістю інструментів. Тому актуальною є розробка програмного забезпечення інформаційної системи для генерації музичних творів.

При розробці програмного забезпечення інформаційної системи для генерації музичних творів необхідно забезпечити такі функціональні вимоги:

- можливість синтезу нейромережевої моделі для генерації музичних творів;
- можливість використання навченої моделі для генерації музичних творів;
- підтримка можливості прослуховування згенерованої музики;
- можливість зберігання синтезованої генеративної моделі;
- можливість збереження згенерованого музичного твору.

1.3 Висновки за розділом 1

Визначено, що генерація музичних творів – це процес автоматичного створення музичних композицій за допомогою алгоритмів штучного інтелекту, зокрема рекурентних нейронних мереж та їхніх вдосконалених варіацій. Проаналізовано основні підходи до генерації музики.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато інформаційних систем для генерації музичних творів. Проте існуюче програмне забезпечення для генерації музичних творів має низку недоліків. Суттєвою проблемою є обмеженість варіативності та гнучкості. Багато генеративних моделей працюють за заданими параметрами, що часто призводить до одноманітності та передбачуваності музичних творів. Навіть найсучасніші інформаційні системи на основі засобів ШІ можуть створювати композиції, які здаються механічними або надто схожими на вже існуючі твори. Крім того, оскільки генеративні алгоритми часто навчаються на великих наборах даних, що містять музику реальних композиторів, виникає ризик порушення прав інтелектуальної власності. Іншим недоліком відомих програмних засобів є використання потужних алгоритмів штучного інтелекту, що потребують великих обчислювальних потужностей, що може бути проблемою для користувачів без доступу до спеціалізованого обладнання або хмарних сервісів. Це впливає на швидкість роботи програм, особливо при генерації складних композицій з великою кількістю інструментів. Тому актуальною є розробка програмного забезпечення інформаційної системи для генерації музичних творів.

Сформульовано функціональні вимоги до програмного забезпечення інформаційної системи для генерації музичних творів.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці програмного забезпечення інформаційної системи для генерації музичних творів було використано мову програмування Python [15], [16].

Python є універсальною мовою програмування, що відзначається високим рівнем об'єктно-орієнтованої парадигми та широкими можливостями для автоматизації завдань. Гнучка структура мови забезпечує ефективне управління типами, використання динамічної пам'яті та зручність роботи з об'єктами. Python надає можливість створювати модульний код, що полегшує його багаторазове використання та розширюваність. Його стандартні бібліотеки та інтерпретатор сумісні з багатьма сучасними платформами та доступні у відкритому форматі [15], [16].

Ця мова підтримує різні програмістські стилі, включаючи процедурне, функціональне, структурне та об'єктно-орієнтоване програмування. Завдяки механізму динамічної типізації, визначення типів відбувається безпосередньо під час виконання коду. Python також забезпечує роботу з різними типами числових даних, серед яких звичайні числа, довгі цілі значення та комплексні числа. Багатий набір вбудованих бібліотек дозволяє зручно працювати з текстовими даними, зокрема підтримується Unicode [15], [16].

Стандартними структурами даних у Python є списки, кортежі, словники та множини, які доповнені можливістю використовувати колекції. Класова система підтримує концепцію множинного успадкування та мета програмування, що дає змогу створювати складні ієрархії об'єктів із розширеними можливостями. Будь-який клас у мові програмування Python може бути успадкований, включаючи базові типи, що робить архітектуру мови особливо гнучкою [15], [16].

Відмінною особливістю Python є інтерактивний режим виконання коду, який дозволяє програмістам вводити команди з клавіатури та миттєво

отримувати результат. Це спрощує процес тестування та відлагодження, особливо для новачків, а також для професійних розробників, яким потрібно швидко перевіряти роботу функцій чи виконувати складні математичні обчислення [15], [16].

Модель об'єктно-орієнтованого програмування у Python реалізована таким чином, що поєднує в собі гнучкість, елегантність та високу продуктивність. Вона суттєво відрізняється від класичних підходів у деяких інших мовах, зберігаючи водночас простоту у використанні та розширюваність, що робить Python потужним інструментом для розробки програмного забезпечення [15], [16].

Для продовження існування об'єктів у Python застосовуються спеціалізовані механізми, такі як конструктори та деструктори, а також алгоритми керування пам'яттю. Використання операторів охоплює всі стандартні випадки, за винятком окремих символів, що не можуть бути перевизначені. Контроль доступу до структур даних здійснюється за допомогою механізмів управління ресурсами та рекурсивного доступу до полів і методів. Реалізація функціональності може здійснюватися різними способами, зокрема через рядкові значення, визначення довжини об'єктів або виклик спеціальних методів, що підтримують операції запису, конкатенації та повернення результатів [15], [16].

Система мета програмування дозволяє контролювати взаємодію класів і їх комбінацій, що підсилює гнучкість під час розробки програмних компонентів. Перевірка цілісності даних у Python виконується на всіх етапах роботи програми. Існує підтримка статичних класів та методів, а також визначення області видимості для кожного елемента. Класи можуть виступати як функції або вкладатися один в одного, що дає можливість будувати складні структури [15], [16].

Однією з важливих складових Python є її стандартна бібліотека, що містить широкий набір інструментів для взаємодії з різними форматами даних та мультимедійними ресурсами. Вона включає модулі для роботи з HTTP-

запитами, форматування електронних листів, обробки XML та багатьох інших завдань. Передбачена підтримка засобів шифрування, обробки історичних даних, форматів файлів і механізмів тестування програмного коду [15], [16].

Окрім вбудованих бібліотек, існує можливість інтеграції Python із зовнішніми системами, включаючи використання API Windows через Win32, а також взаємодію з різними компонентами. У різних сферах застосування доступні бібліотеки для веброзробки, роботи з базами даних, обробки графічних зображень, аналізу текстів, реалізації чисельних методів та взаємодії з операційною системою [15], [16].

Ще однією важливою перевагою Python є його потужні засоби для машинного навчання та глибокого навчання, які можуть бути використані для генерації музики. Бібліотеки TensorFlow, PyTorch, Magenta (розроблена Google спеціально для генеративної музики) дозволяють створювати складні нейронні мережі для аналізу та генерації музичних композицій. Наприклад, модель LSTM (довготривалої короткочасної пам'яті) добре підходить для створення мелодій, оскільки вона може запам'ятовувати музичні структури та послідовності [15], [16].

Python також є платформонезалежною мовою, що дозволяє створювати програмне забезпечення, яке може працювати на різних операційних системах без значних змін у коді. Завдяки інтеграції з популярними цифровими аудіо-робочими станціями через API Python можна автоматизувати створення музики та обробку звуку [15], [16].

Для роботи з базами даних передбачений спеціальний інтерфейс, що забезпечує підтримку різних систем управління базами даних, таких як PostgreSQL, Oracle, Sybase, Firebird, Microsoft SQL Server, MySQL та sqlite. В екосистемі Python також є популярні бібліотеки, як-от SQLAlchemy, Dejavu та Django, а для створення веб-додатків широко застосовуються фреймворки Django та Pylons [15], [16].

Обґрунтування вибору мови програмування для розробки програмного забезпечення інформаційної системи для генерації музичних творів наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки програмного забезпечення інформаційної системи для генерації музичних творів

Критерій порівняння мов програмування	Мова програмування		
	Python	C++	JavaScript
Простота синтаксису	+	—	+—
Машинне навчання та нейромережі	+	+—	—
Легка інтеграція з іншими мовами	+	+—	+—
Інструменти для роботи з музикою	+	+	+—
Зручність для створення ПЗ інформаційної системи для генерації музичних творів	+	—	—
Розвинена спільнота та підтримка	+	+	+—

Отже, для реалізації програмного забезпечення інформаційної системи для генерації музичних творів обрано мову програмування Python, яка має велику кількість готових бібліотек та інструментів для роботи з музикою та обробки аудіо, а також потужні засоби для машинного навчання та глибокого навчання, які можуть бути використані для генерації музики. Крім того, Python є платформо незалежною мовою, що дозволяє створювати програмне забезпечення, яке може працювати на різних операційних системах без значних змін у коді.

2.2 Вибір середовища розробки для створення програмного забезпечення інформаційної системи для генерації музичних творів

В якості середовища розробки для створення програмного забезпечення інформаційної системи для генерації музичних творів було обрано середовище PyCharm [17], [18].

PyCharm є середовищем розробки для Python, створеним компанією JetBrains. Воно забезпечує широкий функціонал для створення, налагодження та розгортання програмного забезпечення. Однією з головних його переваг є інтелектуальне автодоповнення коду, яке аналізує написаний код у реальному часі, пропонуючи найкращі варіанти доповнення та виправлення помилок. Це значно прискорює процес розробки [17], [18].

PyCharm має вбудований відлагоджувач, який дозволяє ставити точки зупинки, відстежувати змінні та покроково виконувати код, що спрощує пошук помилок. Також середовище підтримує віртуальні середовища, такі як `venv` та `conda`, що допомагає уникнути конфліктів між бібліотеками та залежностями [17], [18].

Завдяки інтеграції з Git, GitHub, Mercurial та іншими системами контролю версій PyCharm спрощує командну розробку. Крім того, воно має широкі можливості для веб-розробки та підтримує Django, Flask, HTML, CSS і JavaScript, що дозволяє створювати повноцінні вебзатосунки на мові програмування Python [17], [18].

Для наукових розрахунків та аналізу даних середовище розробки PyCharm підтримує такі бібліотеки, як NumPy, Pandas, Matplotlib, SciPy, TensorFlow та PyTorch. Також середовище забезпечує інструменти для роботи з базами даних, такими як SQL, PostgreSQL та MySQL, що дає змогу безпосередньо взаємодіяти з ними [17], [18].

PyCharm пропонує можливість розширення функціоналу за допомогою плагінів, що дозволяє налаштувати середовище відповідно до потреб

розробника. Воно працює на Windows, macOS і Linux, що робить його доступним для використання на різних операційних системах [17], [18].

Ще однією важливою перевагою є підтримка віддаленої розробки через, що особливо корисно при роботі з великими обчисленнями або неймережами на віддалених серверах. Середовище розробки PyCharm доступно у двох версіях: безкоштовній Community Edition, яка містить основні функції для Python-розробки, та Professional Edition, яка включає розширені можливості, такі як підтримка веброзробки, інтеграція з базами даних та інші додаткові функції [17], [18].

Середовище розробки PyCharm поєднує зручність, потужність і гнучкість, що робить його ідеальним вибором для розробників Python незалежно від сфери застосування – від генерації музики до машинного навчання, веброзробки чи аналізу даних [17], [18].

Одна з головних переваг PyCharm – це потужний автодоповнювач коду, який значно спрощує роботу з бібліотеками для музичної генерації, такими як *mido* (робота з MIDI), *librosa* (аналіз аудіо), *pydub* (обробка звуку) та *Magenta* (генерація музики за допомогою машинного навчання) [17], [18].

Середовище підтримує віртуальні середовища, що дозволяє ізолювати залежності проєкту та уникати конфліктів між бібліотеками. Це особливо важливо, якщо використовується TensorFlow, PyTorch або інші бібліотеки для генерації музичних творів на основі нейронних мереж [17], [18].

Інтегрований інструмент налагодження допомагає відстежувати помилки у коді, що особливо корисно при роботі зі складними алгоритмами обробки звуку. Також середовище розробки PyCharm підтримує Git, що дозволяє легко працювати з контрольованими версіями проєкту та співпрацювати з іншими розробниками [17], [18].

Ще одна перевага – вбудована підтримка роботи з базами даних. Це може бути корисним, якщо проєкт вимагає збереження музичних композицій, параметрів генерації або історії взаємодії користувача [17], [18].

Обґрунтування вибору середовища розробки для створення програмного забезпечення інформаційної системи для генерації музичних творів наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення програмного забезпечення інформаційної системи для генерації музичних творів

Критерій порівняння середовищ розробки	Середовища розробки		
	PyCharm	Visual Studio Code	Atom
Вбудований відлагоджувач	+	+	+—
Продуктивність	+	+—	—
Інтелектуальне автодоповнення	+	+—	+—
Частота оновлень та підтримка	+	+	+—
Зручність у налаштуванні	+	+—	—

Таким чином, для створення програмного забезпечення інформаційної системи для генерації музичних творів обрано середовище розробки PyCharm, завдяки своїм потужним інструментам, інтелектуальному автодоповненню, ефективному відлагоджувачу та широкій підтримці Python-бібліотек для роботи з музикою. Крім того, PyCharm підтримує віртуальні середовища, що дозволяє ізолювати залежності проєкту та уникати конфліктів між бібліотеками, що є досить важливим, коли використовується бібліотеки з засобами штучного інтелекту для генерації музичних творів на основі нейромереж.

2.3 Висновки за розділом 2

Для реалізації програмного забезпечення інформаційної системи для генерації музичних творів обрано мову програмування Python, яка має велику кількість готових бібліотек та інструментів для роботи з музикою та обробки аудіо, а також потужні засоби для машинного навчання та глибокого навчання, які можуть бути використані для генерації музики. Крім того, Python є платформо незалежною мовою, що дозволяє створювати програмне забезпечення, яке може працювати на різних операційних системах без значних змін у коді.

Для створення програмного забезпечення інформаційної системи для генерації музичних творів обрано середовище розробки PyCharm, завдяки своїм потужним інструментам, інтелектуальному авто доповненню, ефективному відлагоджувачу та широкій підтримці Python-бібліотек для роботи з музикою. Крім того, PyCharm підтримує віртуальні середовища, що дозволяє ізолювати залежності проєкту та уникати конфліктів між бібліотеками, що є досить важливим, коли використовується бібліотеки з засобами штучного інтелекту для генерації музичних творів на основі неймереж.

3 ОПИС ПРОГРАМИ

3.1 Структура програмного забезпечення інформаційної системи для генерації музичних творів

Структуру програмного забезпечення інформаційної системи для генерації музичних творів наведено на рис. 3.1.

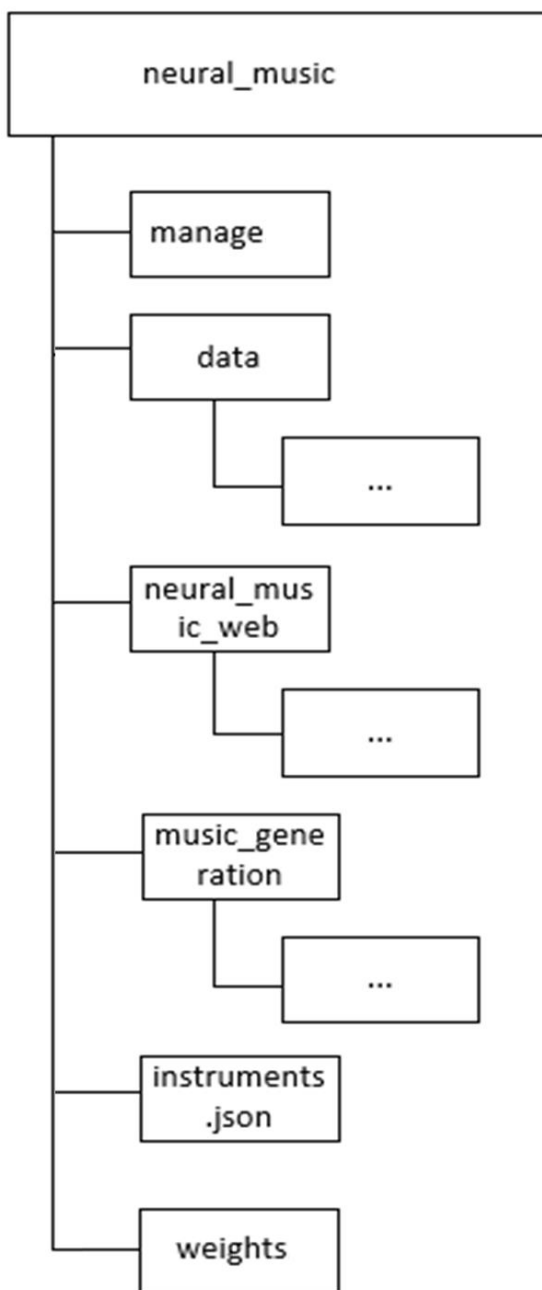


Рисунок 3.1 – Структура програмного забезпечення інформаційної системи для генерації музичних творів

Програмне забезпечення для генерації музичних творів побудоване на основі шести ключових модулів, кожен із яких виконує специфічну роль у процесі створення мелодій.

Основу функціонування забезпечує модуль керування, який відповідає за координацію всіх процесів усередині системи. Він здійснює управління запитами, контролює послідовність виконання операцій та забезпечує взаємодію між іншими частинами програми.

Збереження та обробка інформації здійснюється у спеціальному модулі, який виконує роль сховища для всіх вхідних і вихідних даних. Тут зберігаються як вхідні параметри, необхідні для генерації композицій, так і результати роботи алгоритмів, що дозволяє аналізувати та повторно використовувати створені музичні фрагменти.

Для взаємодії з користувачами передбачено вебінтерфейс, який дозволяє налаштовувати параметри генерації та отримувати результати у зручному форматі. Через цей компонент користувач може передавати необхідні дані, переглядати створені мелодії та взаємодіяти з алгоритмами на інтуїтивному рівні.

Головний процес генерації мелодій виконується спеціалізованим модулем, який базується на алгоритмах штучного інтелекту. У цьому блоці здійснюється синтез музичних творів на основі вхідних параметрів, що дає змогу створювати композиції з урахуванням різних стилів і особливостей гармонійної структури.

Щоб забезпечити підтримку різноманітних музичних інструментів, система використовує окремий конфігураційний файл, у якому зберігаються всі характеристики доступних інструментів. Цей файл містить інформацію про звучання, параметри та особливості кожного інструмента, що дозволяє програмі точно відтворювати створені композиції.

Останній компонент відповідає за збереження навчених параметрів нейронної мережі, які використовуються для створення музики. У цьому модулі зберігаються вагові коефіцієнти, що визначають особливості побудови

мелодій та гармоній. Завдяки цьому можна зберігати раніше отримані результати навчання та використовувати їх для подальшого вдосконалення генеративної моделі.

3.2 Функціонування програмного забезпечення інформаційної системи для генерації музичних творів

Функціонування програмного забезпечення інформаційної системи для генерації музичних творів подамо за допомогою схеми, зображеної на рис. 3.2. Зауважимо, що на схемі наведено процес функціонування ПЗ у режимі створення музичних творів на основі заданих користувачем даних.

Функціонування програмного забезпечення для генерації музичних творів складається з кількох основних етапів, кожен з яких забезпечує певний аспект процесу створення музики.

Спочатку відбувається введення та підготовка вхідних даних. На цьому етапі збираються музичні твори у вигляді файлів відповідних форматів.

Потім на етапі попередньої обробки файли з базовими музичними творами конвертуються у числове представлення, зручне для обробки нейронною мережею. Виконується нормалізація даних, що дозволяє зробити їх придатними для навчання моделі.

Наступним етапом є навчання нейронної мережі. Використовуючи вхідні музичні послідовності, модель аналізує структури, закономірності та стилі композицій. В процесі тренування мережа коригує свої ваги за допомогою алгоритму оптимізації, щоб покращити точність передбачень та створювати музичні фрагменти, схожі на вихідні зразки. Відзначимо, що цей етап може не виконуватися, якщо вже є наявною нейромережева модель, яку можна використовувати для створення нового музичного твору.



Рисунок 3.2 – Схема функціонування програмного забезпечення інформаційної системи для генерації музичних творів

Після успішного навчання здійснюється безпосередньо генерація музики. На основі початкової послідовності нот або випадкового стартового значення нейронмережа прогнозує наступні ноти, поступово створюючи новий музичний твір. Результат представлено у вигляді послідовності чисел, які згодом конвертуються у музичний файл відповідного формату. Далі за

необхідності генерований твір може коригуватися замовником відповідно до заданих критеріїв, можливо, вносяться зміни до ритмічного малюнку, акордової структури або тривалості нот для покращення звучання.

Завершальним кроком є візуалізація та збереження результату. Користувач отримує можливість прослухати композицію та експортувати її у необхідному форматі для подальшого використання.

3.3 Використання нейромережевих моделей у програмному забезпеченні інформаційної системи для генерації музичних творів

Розроблене програмне забезпечення для створення музичних композицій використовує нейронну мережу, побудовану за допомогою фреймворку Keras. Архітектура цієї моделі передбачає використання кількох рівнів рекурентних шарів, які обробляють вхідні музичні дані та генерують нові послідовності нот [19, 20].

Основу мережі складають кілька шарів LSTM, які здатні аналізувати послідовність вхідних нотних подій та формувати наступні елементи мелодії на основі попереднього контексту. Щоб уникнути перенавчання, між цими рівнями застосовуються шари регуляризації Dropout, які випадковим чином відключають частину нейронів у процесі тренування [19, 20].

Після рекурентних шарів іде повнозв'язний рівень Dense, що забезпечує зв'язок між всіма вхідними та вихідними нейронами. Завершує архітектуру шар активації, який визначає ймовірності для кожної можливої ноти або акорду в результаті роботи моделі [19, 20].

Кількість нейронів у кожному шарі вибирається з урахуванням складності музичних послідовностей, що генеруються. Усі рекурентні рівні працюють із вхідними послідовностями, а їхня довжина визначає, скільки попередніх нот буде враховуватись при прогнозуванні наступної [19, 20].

Щоб правильно навчити модель, використовується функція втрат категоричної крос-ентропії, оскільки генеровані музичні елементи належать до

конкретних категорій, а їх кількість перевищує два. Процес оптимізації відбувається за допомогою алгоритму RMSprop, який ефективно оновлює ваги мережі та прискорює процес навчання [19, 20].

Під час навчання мережа обробляє великий обсяг музичних фрагментів, аналізує закономірності та ритмічні структури, що дозволяє їй генерувати гармонійні композиції на основі отриманих знань [19, 20].

Рекурентні нейронні мережі є особливим видом архітектури, у якій зв'язки між елементами організовані таким чином, що дозволяють зберігати певний внутрішній стан. Завдяки цьому стає можливим обробляти послідовності даних і враховувати часовий контекст. Умовна схема подібної архітектури демонструє механізм, за яким мережа отримує вхідний сигнал та формує відповідний вихідний результат, передаючи накопичену інформацію через власну структуру [19, 20].

Основною перевагою такого підходу є можливість обробки не лише поточних значень, а й використання попередньої інформації для передбачення майбутніх даних. Це відкриває широкі можливості для застосування в задачах аналізу часових рядів, розпізнавання мовлення, автоматичного перекладу та інших областях, де важливий контекст попередніх значень [19, 20].

Одним із суттєвих викликів, з якими стикаються подібні нейромережі, є поступове згасання переданої інформації через довгий ланцюжок розрахунків. У міру збільшення відстані між актуальними і попередніми значеннями мережа починає втрачати здатність правильно інтерпретувати зв'язки між ними [19, 20].

Для вирішення цієї проблеми було запропоновано архітектуру LSTM (Long Short-Term Memory). Вона містить спеціальні механізми контролю збереження та оновлення інформації, що дає змогу зберігати важливі елементи послідовностей протягом тривалого часу. Завдяки цим особливостям LSTM стали широко використовуватися в задачах, пов'язаних із послідовною обробкою даних [19, 20].

Архітектура, що базується на короткочасній пам'яті, є варіантом рекурентних нейронних мереж, яка здатна зберігати важливу інформацію протягом тривалого часу. Для досягнення цього використовується механізм довготривалої пам'яті, який дозволяє долати обмеження стандартних рекурентних нейронних мереж [19, 20].

LSTM відіграє ключову роль у ситуаціях, коли класичні рекурентні мережі не можуть ефективно працювати через втрату значущої інформації. Часто виникає необхідність здійснювати обчислення, спираючись на великий обсяг попередніх даних, однак звичайні RNN не здатні ефективно обробляти такі «довготривалі залежності». Відсутність механізмів для вибору важливих фрагментів історії або їхнього ігнорування ускладнює навчання та масштабування таких моделей, що призводить до небажаних ефектів, таких як перенавчання або втрата контексту [19, 20].

Саме для подолання цих обмежень було розроблено LSTM. Ця архітектура покликана усунути проблему згасання градієнтів, залишаючи сам процес навчання незмінним. Використання спеціальних механізмів дозволяє LSTM справлятися з багаторівневими плануваннями, працювати з розрідженими представленнями та невизначеністю, а також забезпечувати стабільну обробку шумових даних. На відміну від традиційних підходів, де потрібне попереднє визначення масштабів, LSTM усуває цю необхідність, що робить її більш гнучкою для використання в задачах моделювання послідовностей [19, 20].

Головною відмінністю між стандартними рекурентними мережами та LSTM є наявність спеціального механізму керування пам'яттю, який реалізований через структуру замкнених комірок. Функціонування таких комірок базується на чотирьох ключових елементах, що визначають, які дані слід зберегти, а які — відкинути. Якщо у звичайних RNN інформація проходить лише через один рівень, то LSTM має складнішу організацію, включаючи три логічні елементи та додатковий рівень обробки. Використання спеціальних шлюзів дозволяє регулювати потік інформації, визначаючи, які

фрагменти даних варто передати далі, а які мають бути забуті. Оцінювання цих рішень здійснюється за шкалою від 0 до 1, де мінімальне значення означає повне ігнорування, а максимальне — повне збереження [19, 20].

У процесі введення інформації до системи враховується значний обсяг даних, що характеризують стан комірки. Спочатку ці дані проходять трансформацію за допомогою сигмоїдної функції, після чого виконуються обчислення, що залучають значення попереднього стану та поточного вхідного сигналу. Для подальшої обробки застосовується гіперболічний тангенс, що обмежує значення у діапазоні від -1 до +1, сприяючи їх нормалізації. Завершальним етапом цього процесу є використання процедур масштабування, які дозволяють отримати відкориговану інформацію для подальшої роботи [19, 20].

Формування вихідних даних відбувається через спеціальний механізм, який забезпечує обробку інформації, що надходить із певного контексту. Спочатку створюється вектор, що служить для подальшої обробки, після чого проводиться фільтрація вхідних параметрів із залученням сигмоїдної функції. Цей процес дає змогу визначити значення ключових змінних, які проходять додаткову обробку перед передачею на наступний рівень обчислень [19, 20].

Для підвищення ефективності LSTM було запропоновано низку змін, спрямованих на оптимізацію структури. Зокрема, дослідники Герс і Шмідхубер розробили модель, яка дозволяє отримувати відомості про стан системи в будь-який момент часу. У деяких модифікаціях використовуються повнозв'язні рівні з окремими механізмами прийняття рішень, що підвищує гнучкість у розподілі ресурсів. Іншим підходом є використання комірок GRU, які надають можливість ефективно визначати поточний стан та регулювати оновлення з урахуванням забутих записів [19, 20].

Попри широке поширення LSTM, архітектура GRU не здобула аналогічного рівня популярності. Запропонована Чо модель, що базується на механізмі контролю короткочасних залежностей, передбачає використання двох основних елементів: механізму оновлення, який визначає, які дані слід

зберегти, та механізму скидання, що керує процесом забування взаємопов'язаних ознак [19, 20].

На відміну від LSTM, у GRU не виконується оновлення стану комірки до моменту отримання нового блоку даних. У той час як LSTM регулює потік інформації в полі пам'яті, GRU формує прогнозовані значення, спираючись на аналіз попередніх активностей, але не здійснює автоматичного оновлення при зміні рівня активності [19, 20].

Перед використанням моделей LSTM у практичних завданнях їх необхідно попередньо навчити на відповідному наборі даних. Такі моделі мають ряд важливих переваг, які роблять їх корисними в різних сферах. У галузі обробки тексту вони дозволяють прогнозувати наступне слово в послідовності, аналізуючи контекст і закономірності в мовленні. Це може застосовуватися як на рівні символів, так і для окремих слів, речень або навіть цілих абзаців [19, 20].

При роботі із зображеннями моделі використовуються для їх аналізу та генерації текстових описів. Для цього спочатку формується набір зображень з відповідними текстовими анотаціями, а потім модель навчається знаходити відповідності між графічними характеристиками та словами. На основі отриманих знань система може будувати осмислені описи нових зображень. У сфері розпізнавання мовлення та рукописного тексту нейромережеві моделі LSTM допомагають інтерпретувати послідовності сигналів та переводити їх у текстовий формат. У генерації музики ці ж принципи застосовуються до нотних послідовностей, де мережа вчиться визначати закономірності та прогнозувати наступні звуки [19, 20].

Моделі також ефективно використовуються для автоматичного перекладу текстів між різними мовами. Як і в обробці зображень, LSTM працюють із закодованими поданнями слів, що дозволяє створювати високоточні переклади. Завдяки архітектурі "кодер-декодер" вхідні речення перетворюються у векторне представлення, а потім декодуються у відповідний текст іншою мовою [19, 20].

Попри високу ефективність у складних завданнях, використання LSTM пов'язане з певними труднощами. Оскільки обробка інформації вимагає переміщення великих обсягів даних, збільшується складність обчислень, що відбуваються всередині комірок. Зростання кількості елементів, таких як механізми забування, ускладнює обробку [19, 20].

Високі вимоги до ресурсів є ще одним викликом. Значна потреба в пам'яті та обчислювальних потужностях робить LSTM менш ефективними з апаратної точки зору. Крім того, важливо враховувати здатність цих моделей працювати із довгостроковими залежностями, що особливо актуально у сфері аналізу даних [19, 20].

LSTM використовують випадкову ініціалізацію ваг, що впливає на передбачення їх продуктивності та потребує відповідного налаштування параметрів. Через високу обчислювальну складність може виникати необхідність у методах оптимізації, таких як механізм випадання, який допомагає визначати взаємозв'язки між вагами нейронної мережі [19, 20].

LSTM-мережі широко використовуються для генерації музичних творів, оскільки вони здатні ефективно працювати з послідовними даними, такими як музичні ноти або MIDI-файли. Завдяки здатності запам'ятовувати довготривалі залежності такі нейронні мережі можуть аналізувати музичні закономірності та створювати мелодійні послідовності, що відповідають стилю певного жанру або композитора [19, 20].

Процес генерації музики за допомогою LSTM зазвичай передбачає навчання моделі на великому наборі музичних даних. Це можуть бути MIDI-файли або нотні послідовності, які містять інформацію про висоту звуку, тривалість нот, темп, динаміку та гармонійні структури. Після навчання модель отримує початковий фрагмент мелодії, на основі якого вона прогнозує наступні ноти та будує новий музичний твір [19, 20].

LSTM добре працюють з різними музичними стилями, оскільки можуть навчатися на специфічних наборах даних. Якщо модель тренується на класичних творах, вона буде генерувати мелодії, схожі на твори Бетховена чи

Моцарта, а якщо використовується джазова база – результатом буде імпровізаційний стиль із складними гармоніями [19, 20].

Однією з важливих особливостей таких моделей є їхня здатність враховувати контекст усієї мелодії, а не лише окремих нот. Це дозволяє створювати музичні структури зі складними ритмічними та гармонійними патернами [19, 20].

Попри значні досягнення, генерація музики з LSTM має певні обмеження. Модель може іноді втрачати структуру композиції або повторювати шаблони без достатньої варіативності. Крім того, якість результату сильно залежить від якості та обсягу навчальних даних. Для покращення композиційних здібностей нейромережі можуть комбінуватися з іншими методами, такими як GAN (Generative Adversarial Networks) [19, 20].

Таким чином, LSTM-мережі відкривають великі можливості у сфері музичної генерації, дозволяючи автоматично створювати оригінальні мелодії, акомпанементи або навіть складні оркестрові твори. Це може бути корисним не лише для музикантів та композиторів, а й для створення фонового аудіо супроводу в іграх, фільмах та інших мультимедійних проектах [19, 20].

3.4 Проєктування інтерфейсу програмного забезпечення інформаційної системи для генерації музичних творів

Проєктування інтерфейсу взаємодії користувача з програмним забезпеченням інформаційної системи для генерації музичних творів виконано з урахуванням вимог технічного завдання. При розробленні програмного забезпечення інформаційної системи для генерації музичних творів враховані функціональні вимоги до програми:

- можливість синтезу нейромережевої моделі для генерації музичних творів;
- можливість використання навченої моделі для генерації музичних творів;

- підтримка можливості прослуховування згенерованої музики;
- можливість зберігання синтезованої генеративної моделі;
- можливість збереження згенерованого музичного твору.

Інтерфейс сторінки з формою для генерації музики на основі музичних файлів наведено на рис. 3.3.

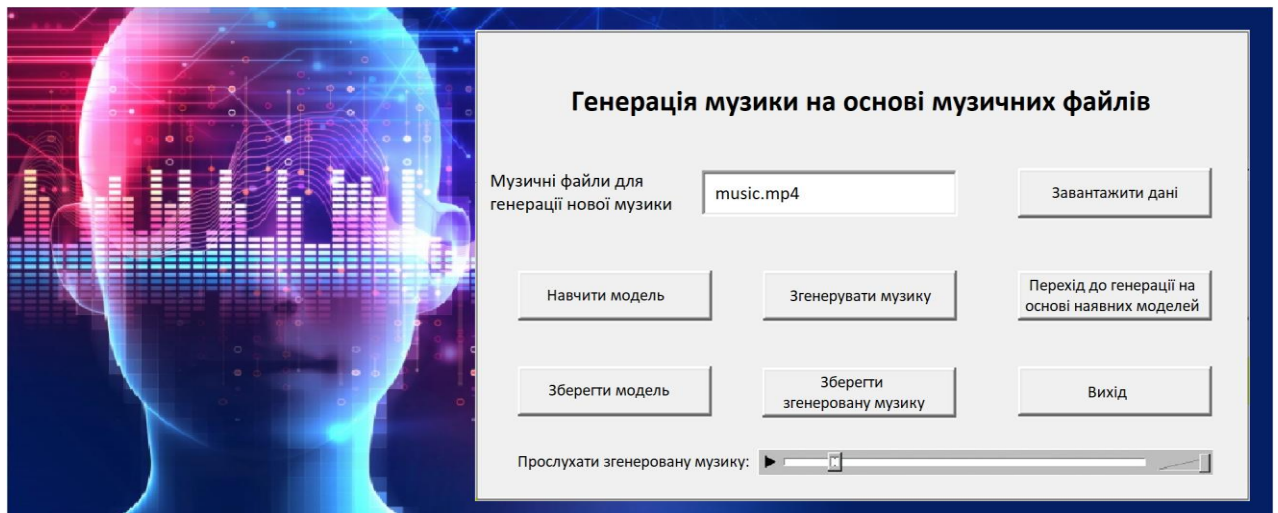


Рисунок 3.3 – Інтерфейс сторінки з формою для генерації музики на основі музичних файлів

Інтерфейс сторінки з формою для генерації музики на основі синтезованої нейромережевої моделі наведено на рис. 3.4.

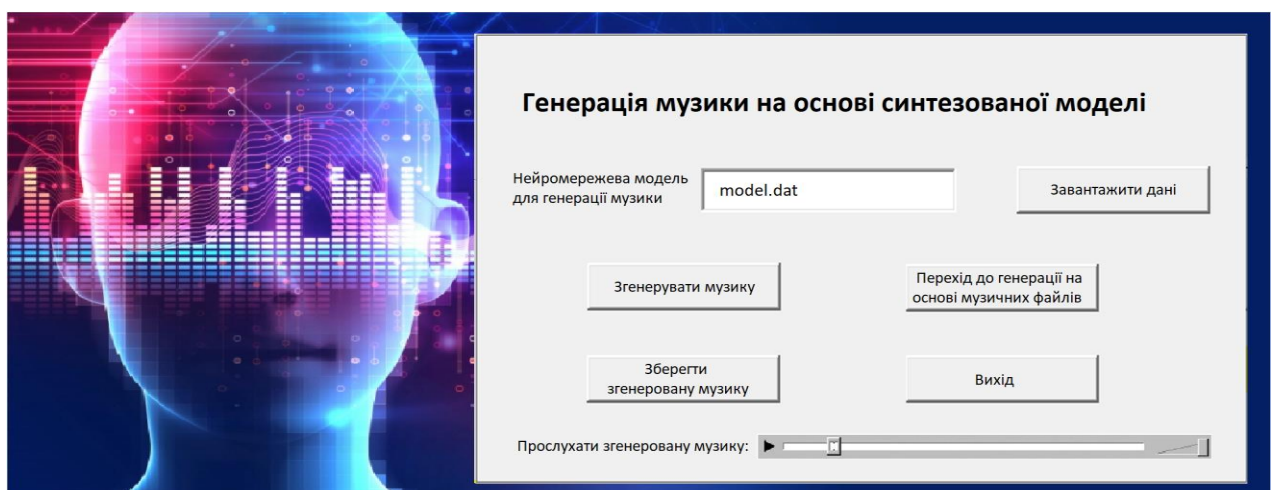


Рисунок 3.4 – Інтерфейс сторінки з формою для генерації музики на основі синтезованої нейромережевої моделі

3.5 Висновки за розділом 3

Розроблено програмне забезпечення інформаційної системи для генерації музичних творів.

Запропоновано структуру програмного забезпечення інформаційної системи для генерації музичних творів. Визначено, що програмне забезпечення для генерації музичних творів побудоване на основі шести ключових модулів, кожен із яких виконує специфічну роль у процесі створення мелодій.

Описано функціонування програмного забезпечення інформаційної системи для генерації музичних творів. Функціонування програмного забезпечення для генерації музичних творів складається з кількох основних етапів, кожен з яких забезпечує певний аспект процесу створення музики.

Описано особливості реалізації програмного забезпечення інформаційної системи для генерації музичних творів. Розроблене програмне забезпечення для створення музичних композицій використовує нейронну мережу, побудовану за допомогою фреймворку Keras.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням інформаційної системи для генерації музичних творів.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для підтримки процесу функціонування для генерації музичних творів. Програма написана на мові Python, яка має велику кількість готових бібліотек та інструментів для роботи з музикою та обробки аудіо, а також потужні засоби для машинного навчання та глибокого навчання, які можуть бути використані для генерації музики. В якості середовища розробки для створення програмного забезпечення для підтримки процесу управління проєктами обрано PyCharm, завдяки його потужним інструментам, інтелектуальному авто доповненню, ефективному відлагоджувачу та широкій підтримці Python-бібліотек для роботи з музикою. Крім того, PyCharm підтримує віртуальні середовища, що дозволяє ізолювати залежності проєкту та уникати конфліктів між бібліотеками, що є досить важливим, коли використовується бібліотеки з засобами штучного інтелекту для генерації музичних творів на основі нейромереж.

Програмне забезпечення інформаційної системи для генерації музичних творів забезпечує виконання таких функцій:

- можливість синтезу нейромережевої моделі для генерації музичних творів;
- можливість використання навченої моделі для генерації музичних творів;
- підтримка можливості прослуховування згенерованої музики;
- можливість зберігання синтезованої генеративної моделі;
- можливість збереження згенерованого музичного твору.

Для роботи програмного забезпечення інформаційної системи для генерації музичних творів потрібні такі технічні та програмні ресурси як потужне обчислювальне обладнання, зокрема процесори з високою тактовою частотою та графічні прискорювачі для швидкого оброблення нейронних

мереж, достатній обсяг оперативної пам'яті (не менше 16 ГБ) для ефективного збереження проміжних результатів, а також швидкі накопичувачі SSD для роботи з великими наборами даних. З програмного забезпечення необхідні бібліотеки для глибокого навчання, такі як TensorFlow або PyTorch, інструменти для роботи з музичними форматами. Важливим є інтегроване середовище розробки або хмарні платформи для запуску й тестування моделей.

Окрім апаратних та програмних ресурсів, знадобиться база навчальних даних, що містить аудіо файли, а також засоби для прослуховування згенерованих музичних композицій.

Функціональні характеристики розробленого програмного забезпечення інформаційної системи для генерації музичних творів наведено у технічному завданні (додаток А). Фрагмент тексту програмного забезпечення інформаційної системи для генерації музичних творів наведено у додатку Б.

4.2 Характеристики програмного забезпечення інформаційної системи для генерації музичних творів

Програмне забезпечення інформаційної системи для генерації музичних творів забезпечує автоматизований процес створення мелодій, гармоній і ритмічних структур на основі нейромережевих моделей. Воно використовує методи штучного інтелекту, машинного навчання та обробки звукових сигналів для аналізу існуючих музичних зразків і генерації нових композицій.

Програмне забезпечення інформаційної системи для генерації музичних творів містить інтерфейс, що дозволяє користувачам виконувати створення нових музичних композицій. Вбудовані алгоритми адаптуються до заданих параметрів, аналізують музичні закономірності та створюють гармонійні послідовності відповідно до заданого стилю.

Система працює на основі модульної архітектури, що дозволяє розширювати її функціональність шляхом додавання нових алгоритмів та стилів музики.

Реалізовано підтримку машинного навчання для покращення якості створюваної музики та її адаптації до вподобань користувача.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску програмного забезпечення інформаційної системи для генерації музичних творів необхідно запустити відповідний ру-файл.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення інформаційної системи для генерації музичних творів є набір музичних файлів, на основі яких буде створено новий музичний твір, нейромережева модель для генерації музичних творів.

Вихідними даними програмного забезпечення інформаційної системи для генерації музичних творів є згенерований музичний твір, синтезована нейромережева модель.

4.3.3 Повідомлення

Користувач програмного забезпечення інформаційної системи для генерації музичних творів може отримати повідомлення про початок генерації музичного твору, повідомлення про завершення генерації музичного твору та інші повідомлення.

4.4 Виконання програмного забезпечення інформаційної системи для генерації музичних творів

Після запуску програмного забезпечення інформаційної системи для генерації музичних творів відображається сторінка з формою для генерації музики (рис. 4.1). Користувачу потрібно обрати музичні файли, на основі яких буде створено нові музичні композиції. Для цього треба натиснути кнопку Завантажити дані та у діалоговому вікні обрати потрібні файли. Після цього виконується навчання нейромережевої моделі та генерація на її основі музичної композиції. Користувач має можливість зберегти синтезовану модель для подальшого використання. Також програма надає можливість прослуховування та зберігання створеної музики.

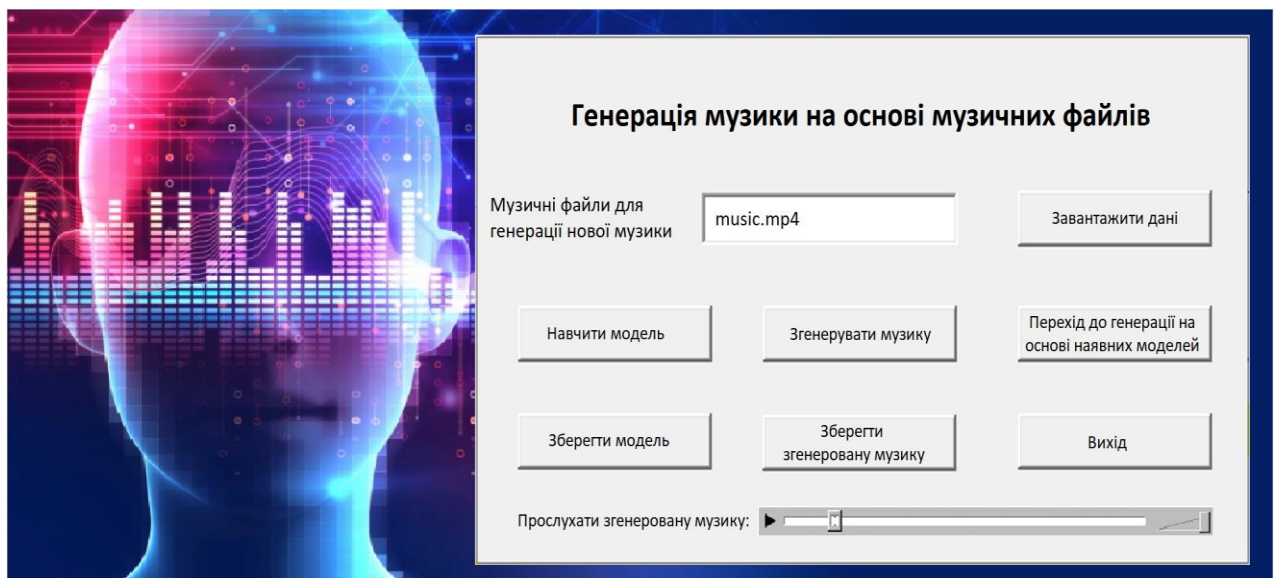


Рисунок 4.1 – Інтерфейс сторінки з формою для генерації музики на основі музичних файлів

При наявності генеративної моделі користувач може перейти до другої форми (рис. 4.2), натиснувши відповідну кнопку. Тут необхідно обрати за допомогою діалогового вікна наявну модель та згенерувати на її основі музику.

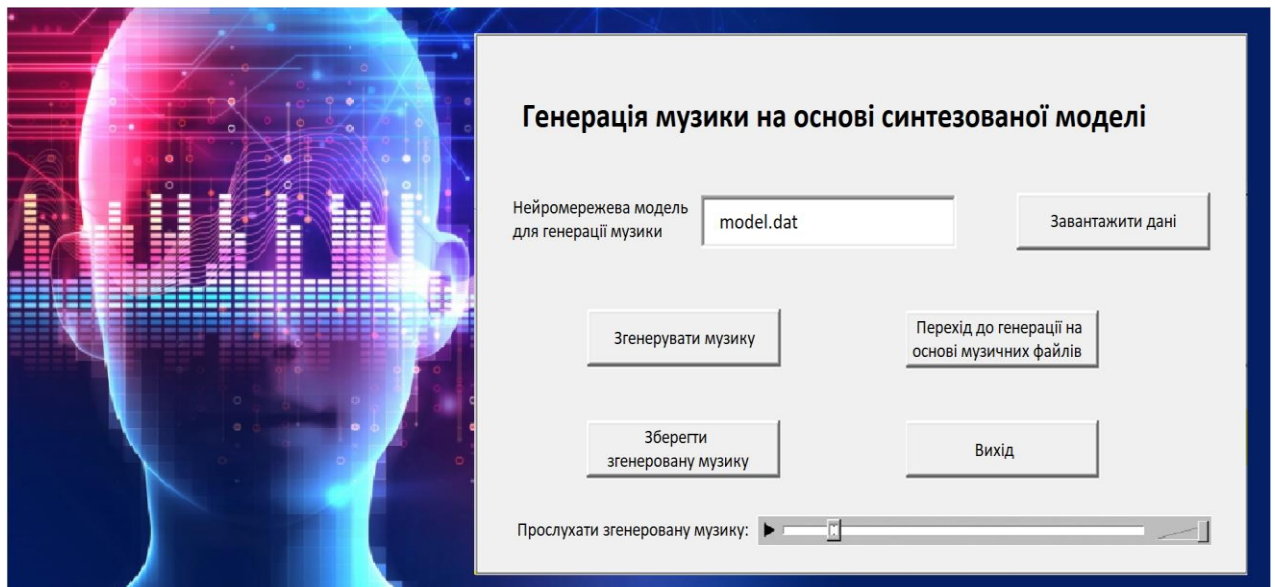


Рисунок 4.2 – Інтерфейс сторінки з формою для генерації музики на основі синтезованої нейромережевої моделі

4.5 Тестування програмного забезпечення інформаційної системи для генерації музичних творів

Проведено тестування програмного забезпечення інформаційної системи для генерації музичних творів.

Підтверджено, що програмне забезпечення інформаційної системи для генерації музичних творів працює коректно і стабільно, виконуючи всі необхідні функціональні вимоги та успішно справляючись із основною задачею підтримки процесу створення музики.

Розроблене програмне забезпечення забезпечує автоматизацію операцій інформаційної системи, пов'язаних із підтримкою процесів генерації музичних творів.

4.6 Висновки за розділом 4

Описано програмне забезпечення інформаційної системи для генерації музичних творів.

Проведено тестування створеного програмного забезпечення інформаційної системи для генерації музичних творів. Результати тестування показали, що програмне забезпечення інформаційної системи для генерації музичних творів забезпечує автоматизацію процесів, пов'язаних із створенням музичних композицій.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки інформаційної системи для генерації музичних творів.

Визначено, що генерація музичних творів – це процес автоматичного створення музичних композицій за допомогою алгоритмів штучного інтелекту, зокрема рекурентних нейронних мереж та їхніх вдосконалених варіацій. Проаналізовано основні підходи до генерації музики.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато інформаційних систем для генерації музичних творів. Проте існуюче програмне забезпечення для генерації музичних творів має низку недоліків. Суттєвою проблемою є обмеженість варіативності та гнучкості. Багато генеративних моделей працюють за заданими параметрами, що часто призводить до одноманітності та передбачуваності музичних творів. Навіть найсучасніші інформаційні системи на основі засобів ШІ можуть створювати композиції, які здаються механічними або надто схожими на вже існуючі твори. Крім того, оскільки генеративні алгоритми часто навчаються на великих наборах даних, що містять музику реальних композиторів, виникає ризик порушення прав інтелектуальної власності. Іншим недоліком відомих програмних засобів є використання потужних алгоритмів штучного інтелекту, що потребують великих обчислювальних потужностей, що може бути проблемою для користувачів без доступу до спеціалізованого обладнання або хмарних сервісів. Це впливає на швидкість роботи програм, особливо при генерації складних композицій з великою кількістю інструментів. Тому актуальною є розробка програмного забезпечення інформаційної системи для генерації музичних творів.

Сформульовано функціональні вимоги до програмного забезпечення інформаційної системи для генерації музичних творів.

Для реалізації програмного забезпечення інформаційної системи для генерації музичних творів обрано мову програмування Python, яка має велику кількість готових бібліотек та інструментів для роботи з музикою та обробки аудіо, а також потужні засоби для машинного навчання та глибокого навчання, які можуть бути використані для генерації музики. Крім того, Python є платформо незалежною мовою, що дозволяє створювати програмне забезпечення, яке може працювати на різних операційних системах без значних змін у коді.

Для створення програмного забезпечення інформаційної системи для генерації музичних творів обрано середовище розробки PyCharm, завдяки своїм потужним інструментам, інтелектуальному авто доповненню, ефективному відлагоджувачу та широкій підтримці Python-бібліотек для роботи з музикою. Крім того, PyCharm підтримує віртуальні середовища, що дозволяє ізолювати залежності проєкту та уникати конфліктів між бібліотеками, що є досить важливим, коли використовується бібліотеки з засобами штучного інтелекту для генерації музичних творів на основі нейромереж.

Розроблено програмне забезпечення інформаційної системи для генерації музичних творів.

Запропоновано структуру програмного забезпечення інформаційної системи для генерації музичних творів. Визначено, що програмне забезпечення для генерації музичних творів побудоване на основі шести ключових модулів, кожен із яких виконує специфічну роль у процесі створення мелодій.

Описано функціонування програмного забезпечення інформаційної системи для генерації музичних творів. Функціонування програмного забезпечення для генерації музичних творів складається з кількох основних етапів, кожен з яких забезпечує певний аспект процесу створення музики.

Описано особливості реалізації програмного забезпечення інформаційної системи для генерації музичних творів. Розроблене програмне

забезпечення для створення музичних композицій використовує нейронну мережу, побудовану за допомогою фреймворку Keras.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням інформаційної системи для генерації музичних творів.

Проведено тестування створеного програмного забезпечення інформаційної системи для генерації музичних творів. Результати тестування показали, що програмне забезпечення інформаційної системи для генерації музичних творів забезпечує автоматизацію процесів, пов'язаних із створенням музичних композицій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. AI for Music Generation (Overview) [Electronic resource]. – Access mode: <https://viso.ai/deep-learning/ai-music-generation/>.
2. The Rise of AI-Generated Music: What It Means for Artists [Electronic resource]. – Access mode: <https://flourishprosper.net/music-resources/the-rise-of-ai-generated-music-what-it-means-for-artists/>.
3. Music Transformer: Generating Music with Long-Term Structure [Electronic resource]. – Access mode: <https://magenta.tensorflow.org/music-transformer>.
4. The Process of Developing an AI Music Generator: A Technical Overview [Electronic resource]. – <https://medium.com/@seekmeai/the-process-of-developing-an-ai-music-generator-a-technical-overview-5a736820f259>.
5. How AI Changed the Music Industry in under a Year [Electronic resource]. – Access mode: <https://deepgram.com/learn/how-ai-changed-music>.
6. A systematic review of artificial intelligence-based music generation: Scope, applications, and future trends [Electronic resource]. – Access mode: <https://www.sciencedirect.com/science/article/pii/S0957417422013537>.
7. AI-Based Affective Music Generation Systems: A Review of Methods, and Challenges [Electronic resource]. – Access mode: <https://ar5iv.labs.arxiv.org/html/2301.06890>.
8. Best AI Music Generators [Electronic resource]. – Access mode: <https://www.unite.ai/best-ai-music-generators/>.
9. Best AI Music Generator Tools for Every Musician [Electronic resource]. – Access mode: <https://www.tech2geek.net/14-best-ai-music-generator-tools-for-every-musician-2025/>.
10. Best AI Music Generators [Electronic resource]. – Access mode: <https://usefulai.com/tools/ai-music-generators>.
11. Best AI Music Generators in 2025 [Electronic resource]. – Access mode: <https://www.cyberlink.com/blog/the-top-video-editors/3390/ai-music-generators>.

12. Hydra [Electronic resource]. – Access mode: <https://rightsify.com/hydra/>.
13. AIVA [Electronic resource]. – Access mode: <https://www.aiva.ai/>.
14. Loudly [Electronic resource]. – Access mode: <https://www.loudly.com/>.
15. Python Classes [Electronic resource]. – Access mode: <https://docs.python.org/3.13/tutorial/classes.html>.
16. Practice Python Exercises and Challenges with Solutions [Electronic resource]. – Access mode: <https://pynative.com/python-exercises-with-solutions/>.
17. Code faster with AI Assistant in PyCharm [Electronic resource]. – Access mode: <https://www.jetbrains.com/pycharm/features/ai/>.
18. PyCharm – El IDE de Python para profesionales [Electronic resource]. – Access mode: <https://elpythonista.com/pycharm>.
19. Understanding of LSTM Networks [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/understanding-of-lstm-networks/>.
20. LSTMs Explained: A Complete, Technically Accurate, Conceptual Guide with Keras [Electronic resource]. – Access mode: <https://medium.com/analytics-vidhya/lstms-explained-a-complete-technically-accurate-conceptual-guide-with-keras-2a650327e8f2>.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для підтримки процесу генерації музичних творів.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмне забезпечення інформаційної системи для генерації музичних творів», затверджене наказом Національного університету «Запорізька політехніка» № 209 від 28 квітня 2025 р.

A.2 Призначення розробки

Програмний продукт призначений для генерації музичних творів.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення інформаційної системи для генерації музичних творів забезпечує виконання таких функцій:

- можливість синтезу нейромережевої моделі для генерації музичних творів;
- можливість використання навченої моделі для генерації музичних творів;
- підтримка можливості прослуховування згенерованої музики;
- можливість зберігання синтезованої генеративної моделі;
- можливість збереження згенерованого музичного твору.

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення інформаційної системи для генерації музичних творів повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

А.3.3 Вимоги до надійності

Програмне забезпечення інформаційної системи для генерації музичних творів повинно забезпечити надійне функціонування.

А.3.4 Умови експлуатації

Для експлуатації програмного забезпечення інформаційної системи для генерації музичних творів необхідна наявність персонального комп'ютера.

А.3.5 Вимоги до складу та параметрів технічний засобів

Для роботи програмного забезпечення інформаційної системи для генерації музичних творів потрібні такі технічні та програмні ресурси як потужне обчислювальне обладнання, зокрема процесори з високою тактовою частотою та графічні прискорювачі для швидкого оброблення нейронних мереж, достатній обсяг оперативної пам'яті (не менше 16 ГБ) для ефективного збереження проміжних результатів, а також швидкі накопичувачі SSD для роботи з великими наборами даних. З програмного забезпечення необхідні бібліотеки для глибокого навчання, такі як TensorFlow або PyTorch, інструменти для роботи з музичними форматами. Важливим є інтегроване

середовище розробки або хмарні платформи для запуску й тестування моделей.

Окрім апаратних та програмних ресурсів, знадобиться база навчальних даних, що містить аудіо файли, а також засоби для прослуховування згенерованих музичних композицій.

А.3.6 Вимоги до маркування і пакування

Програма для підтримки діяльності для генерації музичних творів може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення інформаційної системи для генерації музичних творів».

А.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення інформаційної системи для генерації музичних творів є набір музичних файлів, на основі яких буде створено новий музичний твір, нейромережева модель для генерації музичних творів.

Вихідними даними програмного забезпечення інформаційної системи для генерації музичних творів є згенерований музичний твір, синтезована нейромережева модель.

ДОДАТОК Б
Фрагмент тексту програми

```

def gnrt():

    with open('dany/mscs', 'rb') as flpth:
        mscs = pickle.load(flpth)

    ptchnms = sorted(set(itnm for itnm in mscs))
    nvcb = len(set(mscs))

    ntwrkinpt, nrmlzdinpt = prepare_sequences(mscs, ptchnms,
nvcb)
    nnmdl = create_ntwrk(nrmlzdinpt, nvcb)
    prdctnotpt = gnrt_mscs(nnmdl, ntwrkinpt, ptchnms, nvcb)
    createmsc(prdctnotpt)

def prepare_sequences(mscs, ptchnms, nvcb):
    msctoint = dict((msc, nmbr) for nmbr, msc in
enumerate(ptchnms))

    poslidovnlenght = 100
    ntwrkinpt = []
    otpt = []
    for i in range(0, len(mscs) - poslidovnlenght, 1):
        poslidovnin = mscs[i:i + poslidovnlenght]
        poslidovnout = mscs[i + poslidovnlenght]
        ntwrkinpt.append([msctoint[char] for char in
poslidovnin])
        otpt.append(msctoint[poslidovnout])

    klkekseml = len(ntwrkinpt)

    nrmlzdinpt = numpy.reshape(ntwrkinpt, (klkekseml,
poslidovnlenght, 1))
    nrmlzdinpt = nrmlzdinpt / float(nvcb)

    return (ntwrkinpt, nrmlzdinpt)

```

```

def create_ntwrk(ntwrkinpt, nvcb):
    nnmdl = Sequential()
    nnmdl.add(LSTM(
        512,
        input_shape=(ntwrkinpt.shape[1], ntwrkinpt.shape[2]),
        return_sequences=True
    ))
    nnmdl.add(Dropout(0.3))
    nnmdl.add(LSTM(512, return_sequences=True))
    nnmdl.add(Dropout(0.3))
    nnmdl.add(LSTM(512))
    nnmdl.add(Dense(256))
    nnmdl.add(Dropout(0.3))
    nnmdl.add(Dense(nvcb))
    nnmdl.add(Activation('softmax'))
    nnmdl.compile(loss='categorical_crossentropy',
optimizer='rmsprop')

    nnmdl.load_weights('weights-violin.hdf5')

    return nnmdl

def gnrt_mscs(nnmdl, ntwrkinpt, ptchnms, nvcb):
    strt = numpy.random.randint(0, len(ntwrkinpt)-1)

    into_msc = dict((nmbr, msc) for nmbr, msc in
enumerate(ptchnms))

    eksemplr = ntwrkinpt[strt]
    prdctnotpt = []

    for mscindx in range(500):
        prdctninpt = numpy.reshape(eksemplr, (1, len(eksemplr),
1))

        prdctninpt = prdctninpt / float(nvcb)

        prdctn = nnmdl.predict(prdctninpt, verbose=0)

```

```

    indx = numpy.argmax(prdctn)
    rslt = into_msc[indx]
    prdctnotpt.append(rslt)

    eksemplr.append(indx)
    eksemplr = eksemplr[1:len(eksemplr)]

return prdctnotpt

def createmsc(prdctnotpt):
    ofst = 0
    otpt_mscs = []
    flinstrmnt = instrmnt.Violin()

    for eksemplr in prdctnotpt:
        if ('.' in eksemplr) or eksemplr.isdigit():
            mscs_inchrd = eksemplr.split('.')
            mscs = []
            for krntmsc in mscs_inchrd:
                nbwmsc = msc.Msc(int(krntmsc))
                nbwmsc.storedInstrmnt = flinstrmnt
                mscs.append(nbwmsc)
            nbwchrd = chrd.Chrd(mscs)
            nbwchrd.ofst = ofst
            otpt_mscs.append(nbwchrd)
            # eksemplr is a msc
        else:
            nbwmsc = msc.Msc(eksemplr)
            nbwmsc.ofst = ofst
            nbwmsc.storedInstrmnt = flinstrmnt
            otpt_mscs.append(nbwmsc)

    ofst += 0.5

    msc_strm = stream.Stream(otpt_mscs)
    msc_strm.insert(flinstrmnt)

```

```

dany = json.load(open('instrmnts.json', 'r'))

    msc_strm.write('msc', fp='testotpt_' + str(flinstrmnt) + '_'
+ str(dany[str(flinstrmnt)]) + '.mid')

dany[str(flinstrmnt)] = dany[str(flinstrmnt)] + 1

with open('instrmnts.json', 'w') as flkr:
    json.dump(dany, flkr)

urleksemppls = [
    path('', views.indx, name='indx'),
]

def indx(rqwst):
    if rqwst.method == 'POST':
        otrymandany = rqwst.POST
        dany = {}

        if otrymandany.get('inst') == 'vokal':
            app_logic.gnrt()
            dany['song'] = 'vokal_song'
            return JsonResponse(dany)

    return render(rqwst, 'indx.html')

urleksemppls = [
    path('', include('mscnrtn.urls')),
    path('admin/', admin.site.urls),
]

def train_ntwrk():
    mscs = get_mscs()

    nvcb = len(set(mscs))

    ntwrkinpt, ntwrkotpt = prepare_sequences(mscs, nvcb)

```

```

nnmdl = create_ntwrk(ntwrkinpt, nvcb)

train(nnmdl, ntwrkinpt, ntwrkotpt)

def get_mscs():
    mscs = []

    for flkr in glob.glob("msc_songs/*.mid"):
        msc = converter.parse(flkr)

        mscs_tprs = None

        prts = instrmnt.partitionByInstrmnt(msc)

        if prts: # flkr has instrmnt prts
            mscs_tprs = prts.prts[0].recurse()
        else: # flkr has mscs in a flat structure
            mscs_tprs = msc.flat.mscs

        for elmnt in mscs_tprs:
            if isinstance(elmnt, msc.Msc):
                mscs.append(str(elmnt.pitch))
            elif isinstance(elmnt, chrd.Chrd):
                mscs.append('.'.join(str(n) for n in
elmnt.normalOrder))

        with open('dany/mscs', 'wb') as flpth:
            pickle.dump(mscs, flpth)

    return mscs

def prepare_sequences(mscs, nvcb):
    poslidoynlength = 100

    ptchnms = sorted(set(itnm for itnm in mscs))

    msctoint = dict((msc, nmbr) for nmbr, msc in
enumerate(ptchnms))

```

```

ntwrkinpt = []
ntwrkotpt = []

for i in range(0, len(mscs) - poslidovnlenght, 1):
    poslidovnin = mscs[i:i + poslidovnlenght]
    poslidovnout = mscs[i + poslidovnlenght]
    ntwrkinpt.append([msctoint[char] for char in
poslidovnin])
    ntwrkotpt.append(msctoint[poslidovnout])

klkeksemp1 = len(ntwrkinpt)

ntwrkinpt = numpy.reshape(ntwrkinpt, (klkeksemp1,
poslidovnlenght, 1))
ntwrkinpt = ntwrkinpt / float(nvcb)

ntwrkotpt = np_utils.to_categorical(ntwrkotpt)

return (ntwrkinpt, ntwrkotpt)

def create_ntwrk(ntwrkinpt, nvcb):
    nnmdl = Sequential()
    nnmdl.add(LSTM(
        512,
        input_shape=(ntwrkinpt.shape[1], ntwrkinpt.shape[2]),
        return_sequences=True
    ))
    nnmdl.add(Dropout(0.3))
    nnmdl.add(LSTM(512, return_sequences=True))
    nnmdl.add(Dropout(0.3))
    nnmdl.add(LSTM(512))
    nnmdl.add(Dense(256))
    nnmdl.add(Dropout(0.3))
    nnmdl.add(Dense(nvcb))
    nnmdl.add(Activation('softmax'))
    nnmdl.compile(loss='categorical_crossentropy',
optimizer='adam')

```

```

    return nnmdl

def train(nnmdl, ntwrkinpt, ntwrkotpt):
    flpth = "weights-improvement-{epoch:02d}-{loss:.4f}-
bigger.hdf5"
    chckpnt = NnmdlChckpnt(
        flpth,
        monitor='loss',
        verbose=0,
        save_best_only=True,
        mode='min'
    )
    clbckslst = [chckpnt]

    nnmdl.fit(ntwrkinpt, ntwrkotpt, epochs=200, batch_size=64,
callbacks=clbckslst)

if __name__ == '__main__':
    train_ntwrk()

```

ДОДАТОК В
Слайди презентації

**Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів**

**Програмне забезпечення для генерації
музичних творів**

Виконав: Олександр СИВОГРИВОВ

ст. групи КНТ-212сп

Керівник: Михайло КОЦУР

к.т.н., доцент

Рисунок В.1 – Слайд 1

Об'єкт, предмет та мета роботи

- Об'єкт дослідження – процес розробки інформаційної системи для генерації музичних творів.
- Предмет дослідження – інформаційні системи для генерації музичних творів.
- Метою роботи є розробка програмного забезпечення інформаційної системи для генерації музичних творів.

2

Рисунок В.2 – Слайд 2

Задачі роботи

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та інформаційних систем для генерації музичних творів;
- здійснити проектування програмного забезпечення інформаційної системи для генерації музичних творів;
- створити програмне забезпечення інформаційної системи для генерації музичних творів;
- виконати тестування розробленого програмного забезпечення інформаційної системи для генерації музичних творів.

3

Рисунок В.3 – Слайд 3

Порівняння існуючих аналогів

Критерій порівняння	Hydra	AIVA	Loudly
Можливість редагування музичних творів треків	+	+-	-
Зручність інтерфейсу	+-	+-	+
Якість створених музичних композицій	+-	+	+-
Доступність у безкоштовній версії	-	+-	+
Підходить для професійного використання	+	+-	-
Робота в офлайн режимі	+	+	-

4

Рисунок В.4 – Слайд 4

Аналіз вимог до програми

При розробці програмного забезпечення інформаційної системи для генерації музичних творів необхідно забезпечити такі функціональні вимоги:

- можливість синтезу нейромережевої моделі для генерації музичних творів;
- можливість використання навченої моделі для генерації музичних творів;
- підтримка можливості прослуховування згенерованої музики;
- можливість зберігання синтезованої генеративної моделі;
- можливість збереження згенерованого музичного твору.

5

Рисунок В.5 – Слайд 5

Порівняння мов програмування

Критерій порівняння мов програмування	Мова програмування		
	Python	C++	JavaScript
Простота синтаксису	+	–	+-
Машинне навчання та нейромережі	+	+-	–
Легка інтеграція з іншими мовами	+	+-	+-
Інструменти для роботи з музикою	+	+	+-
Зручність для створення ПЗ інформаційної системи для генерації музичних творів	+	–	–
Розвинена спільнота та підтримка	+	+	+-

6

Рисунок В.6 – Слайд 6

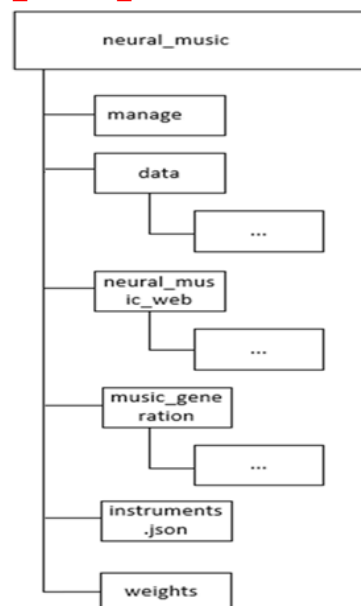
Порівняння середовищ розробки

Критерій порівняння середовищ розробки	Середовища розробки		
	PyCharm	Visual Studio Code	Atom
Вбудований відлагоджувач	+	+	+—
Продуктивність	+	+—	—
Інтелектуальне автодоповнення	+	+—	+—
Частота оновлень та підтримка	+	+	+—
Зручність у налаштуванні	+	+—	—

7

Рисунок В.7 – Слайд 7

Структура програмного забезпечення



8

Рисунок В.8 – Слайд 8

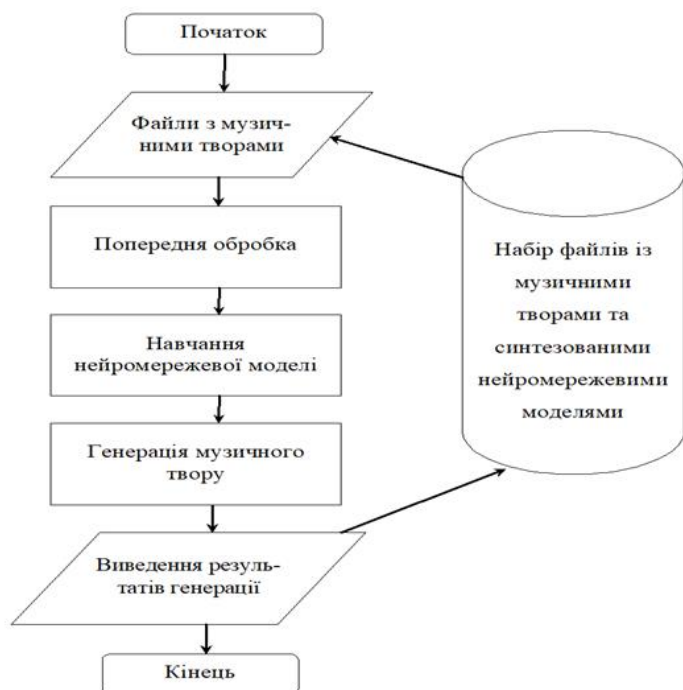
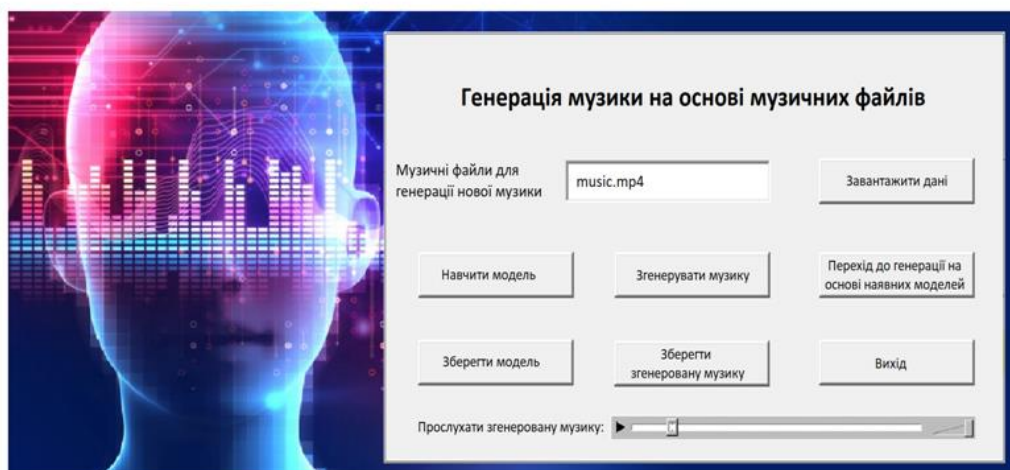


Схема функціонування

9

Рисунок В.9 – Слайд 9

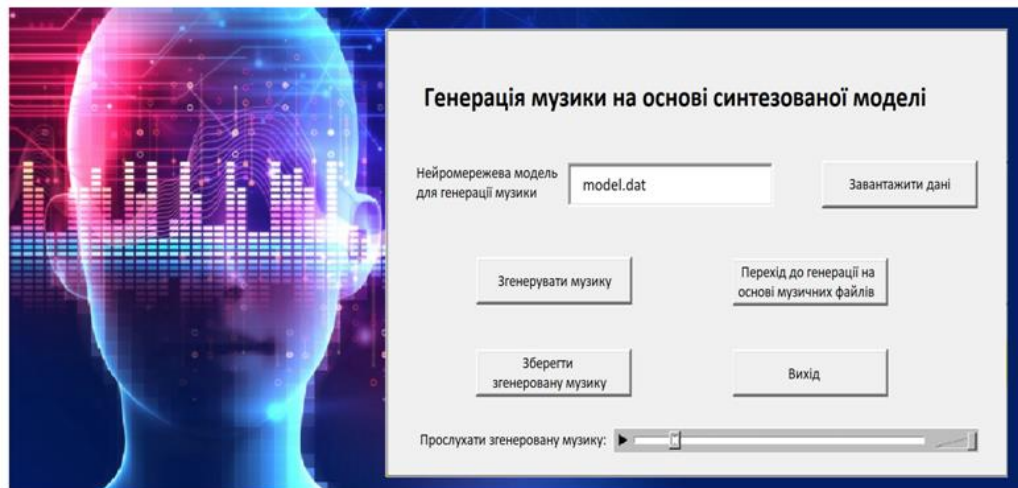
Робота з програмою. Генерації музики на основі музичних файлів



10

Рисунок В.10 – Слайд 10

Робота з програмою. Генерації музики на основі синтезованої моделі



11

Рисунок В.11 – Слайд 11

Висновки

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки інформаційної системи для генерації музичних творів.

Сформульовано функціональні вимоги до програмного забезпечення інформаційної системи для генерації музичних творів. Для реалізації програмного забезпечення інформаційної системи для генерації музичних творів обрано мову програмування Python та середовище розробки PyCharm.

Розроблено програмне забезпечення інформаційної системи для генерації музичних творів. Запропоновано структуру програмного забезпечення інформаційної системи для генерації музичних творів. Визначено, що програмне забезпечення для генерації музичних творів побудоване на основі шести ключових модулів, кожен із яких виконує специфічну роль у процесі створення мелодій.

Описано функціонування програмного забезпечення інформаційної системи для генерації музичних творів. Виконано проектування інтерфейсу взаємодії користувача з програмним забезпеченням інформаційної системи для генерації музичних творів. Виконано тестування розробленого програмного забезпечення інформаційної системи для генерації музичних творів.

Усі завдання дипломної кваліфікаційної роботи бакалавра повністю виконано.

12

Рисунок В.12 – Слайд 12