НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»
спеціальності «121 Інженерія програмного забезпечення»

на тему: Мобільний застосунок для роботи з музичними нотами

Виконала студентка IV курсу, групи ІП-11
(шифр групи)
Фукс Вікторія Ігорівна
(прізвище, ім'я, по батькові) (підпис)

Керівник доцент, к.ф.м.н., доц., Поперешняк С. В
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант доцент, к.т.н., доц., Ліщук К. І.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент зав. каф, д.т.н., проф., Жебка В.В
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2025

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення інформаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2025 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Фукс Вікторії Ігорівні
(прізвище, ім'я, по батькові)

1. Тема проєкту Мобільний застосунок для роботи з музичними нотами
керівник проєкту Поперешняк Світлана Володимирівна, к.ф -м.н, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «23» травня 2025 р. №1705-с
2. Термін подання студентом проєкту «16» червня 2025 року
3. Вихідні дані до проєкту: технічне завдання
4. Зміст пояснювальної записки
1) Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі.
2) Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних.
3) Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів.
4) Технологічний розділ: керівництво користувача, методика випробувань програмного продукту.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань

2) Схема структурна компонентів програмного забезпечення

3) Архітектура програмного забезпечення

4) Креслення вигляду екранних форм

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «15» березня 2025 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	15.03.2025	
2	Аналіз існуючих методів розв’язання задачі	29.03.2025	
3	Постановка та формалізація задачі	18.04.2025	
4	Розробка інформаційного забезпечення	28.04.2025	
5	Алгоритмізація задачі	07.05.2025	
6	Обґрунтування вибору використаних технічних засобів	09.05.2025	
7	Розробка програмного забезпечення	16.05.2025	
8	Налагодження програми	24.05.2025	
9	Виконання графічних документів	27.05.2025	
10	Оформлення пояснювальної записки	01.06.2025	
11	Подання ДП на попередній захист	04.06.2025	
12	Подання ДП рецензенту	12.06.2025	
13	Подання ДП на основний захист	14.06.2025	

Студент _____ Вікторія ФУКС

(підпис) (ініціали, прізвище)

Керівник _____ Світлана ПОПЕРЕШНЯК

(підпис) (ініціали, прізвище)

Анотація

Пояснювальна записка дипломного проєкту складається з п’яти розділів, містить 25 таблиць, 33 рисунків та 38 джерел – загалом 69 сторінок.

Дипломний проєкт присвячений розробці мобільного застосунку для роботи з музичними нотами.

Мета роботи полягає в досягненні наступних цілей:

- конвертацію файлів із музичними нотами;
- створення відеофайлу та аудіофайлу з прочитаних нот;
- створення відео із нотами та їх звучанням;
- обробка різної швидкості програвання;
- об’єднання різних утиліт на різних етапах обробки в одному застосунку;
- інтуїтивний та доступний дизайн;
- збереження та організація творів;

На етапі аналізу були визначені необхідні функціональні та нефункціональні вимоги до програмного продукту.

У розділі «Передпроєктне обстеження предметної області» увага зосереджена на формулюванні цілей дипломного проєкту, вивченні специфіки обраної галузі, розгляді наявних аналогів, а також побудові моделей ключових процесів.

У розділі «Розроблення вимог до програмного забезпечення» подано сценарії використання розроблюваного ПЗ, визначено функціональні й нефункціональні вимоги, проведено аналіз технічних умов і економічної доцільності, а також остаточно сформульовано технічне завдання.

У розділі «Конструювання та розроблення програмного забезпечення» розглянуто архітектурні аспекти та структурні схеми, які відображають логічну організацію.

Розділ аналізу якості включає в себе опис основних сценаріїв тестування та стану системи після їх виконання.

У розділі впровадження та супроводження розглянуто процеси встановлення програмного забезпечення та його подальший супровід.

Результати роботи пройшли апробацію на Всеукраїнській науково-технічній конференції "Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і світу", науково-практичній конференції «Проблеми комп’ютерної інженерії», SoftTech-2023, Всеукраїнському конкурсі студентських робіт «Інформаційно-комунікаційні технології в освіті».

КЛЮЧОВІ СЛОВА: МОБІЛЬНИЙ ДОДАТОК, ШІ, APK, НОТИ, РОЗТРАКТОВКА, ПАРТИТУРА, МЕТРОНОМ, ОБРОБКА PDF, OMR, КОНВЕРТАЦІЯ ФАЙЛІВ, СЕРВЕР, ANDROID, ANDROID STUDIO, PYTHON.

ABSTRACT

The explanatory note of the diploma project consists of five sections, contains 25 tables, 33 figures and 38 sources – in total 69 pages.

The purpose of the diploma project is dedicated to the development of a mobile application for working with music notes.

The purpose of the work is to achieve the following goals:

- conversion of files with music notes;
- creating a video file and an audio file from the read sheet music;
- creating a video with notes and their sound;
- handle different playback speeds;
- combine different utilities at different stages of processing in one application;
- intuitive and accessible design;
- saving and organizing music;

At the analysis stage, the necessary functional and non-functional requirements for the software product were identified.

The section «Pre-design survey of the subject area» focuses on formulating the goals of the diploma project, studying the specifics of the chosen industry, reviewing existing analogues, and building models of key processes.

The section “Development of software requirements” presents scenarios for using the software being developed, identifies functional and non-functional requirements, analyzes technical specifications and economic feasibility, and finally formulates the terms of reference.

The «Software Design and Development» section discusses architectural aspects and structural diagrams that reflect the logical organization.

The «Quality Analysis» section includes a description of the main testing scenarios and the system state after their execution.

The «Implementation and Maintenance» section describes the processes of software installation and its further support.

The results of the work were tested at the All-Ukrainian Scientific and Technical Conference “Technological Horizons: Research and Application of Information Technologies for the Technological Progress of Ukraine and the World”, the Scientific and Practical Conference “Problems of Computer Engineering”, SoftTech-2023, the All-Ukrainian Student Competition “Information and Communication Technologies in Education”.

KEYWORDS: MOBILE APPLICATION, AI, APK, SHEET MUSIC, INTERPRETATION, SCORE, METRONOME, PDF PROCESSING, OMR, FILE CONVERSION, SERVER, ANDROID, ANDROID STUDIO, PYTHON.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІП-1130.045440.01.91	Технічне завдання	8	
3	А4	КПІ.ІП-1130.045440.02.81	Пояснювальна записка	69	
4	А4	КПІ.ІП-1130.045440.03.12	Текст програми	5	
5	А4	КПІ.ІП-1130.045440.04.51	Програма та методика тестування	5	
6	А4	КПІ.ІП-1130.045440.05.34	Керівництво користувача	13	
7	А3	КПІ.ІП-1130.045440.06.99	Графічний матеріал	5	

Змін.	Арк.	№ докум.	Підп.	Дата	КПІ.ІП-1132.045440.00.90		
Розроб.		Фукс В.І			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів
Перевір.		Поперешняк С.В				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІП-11	
Н.контр.		Ліщук К.І.					
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“___” _____ 2025 р.

МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ РОБОТИ З МУЗИЧНИМИ НОТАМИ

Технічне завдання

КПІ.ІП-1132.045440.01.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Світлана ПОПЕРЕШНЯК

Нормоконтроль:

_____ Катерина ЛІЩУК

Виконавець:

_____ Вікторія ФУКС

Київ – 2025

Зміст

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ РОБОТИ З МУЗИЧНИМИ НОТАМИ.

Галузь застосування:

Наведене технічне завдання поширюється на розробку мобільного застосунку для роботи з музичними нотами [Liszt], котра використовується для автоматичного розпізнавання тактів, розмірів та сторінок нот, конвертації їх у формати mp4, mp3 та зберігання у власній бібліотеці у зручному форматі. Призначена для сприяння навчанню та практиці гри, як для початківців, так і для досвідчених музикантів. Також корисний для викладачів, студентів музичних шкіл і всіх, хто цікавиться музикою.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки мобільного застосунку для роботи з музичними нотами «Liszt» є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для створення програмного забезпечення, яке забезпечить точну та швидку обробку музичних нот, а також спрощення взаємодії з музикальними інструментами, з фокусом на поліпшення зручності користування.

Метою розробки є створення інтуїтивно зрозумілого інструменту для всіх аматорів та майстрів музичного мистецтва. Зосереджуючись на покращеній візуалізації та ефективності роботи з нотами, ми створюємо додаток, який дозволяє користувачам легко взаємодіяти з музичним матеріалом та розвивати свої навички в галузі творчості. Також поліпшення роботи з нотами та спрощення гри на музичних інструментах шляхом об’єднання етапів автоматизованого розпізнавання тактів, сторінок, розмірності музичного твору, що робить можливим синхронізування нот на пристроях із метрономом.

Цей мобільний додаток розробляється для надання простого творчого інструменту усім бажаючим та загального розвитку музичних навичок у користувачів. Він також може бути корисним інструментом для викладачів та музичних педагогів у процесі навчання учнів. Застосунок покликаний зробити вивчення музики і вдосконалення своїх навичок більш доступним, захопливим та ефективним для широкого кола користувачів.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1 Користувацького інтерфейсу.

4.1.1.1 Функція додавання PDF файлу з нотами (Рисунок 4.1-4.3).

4.1.1.2 Можливість виводу статистики активності (послідовності днів) (Рисунок 4.1).

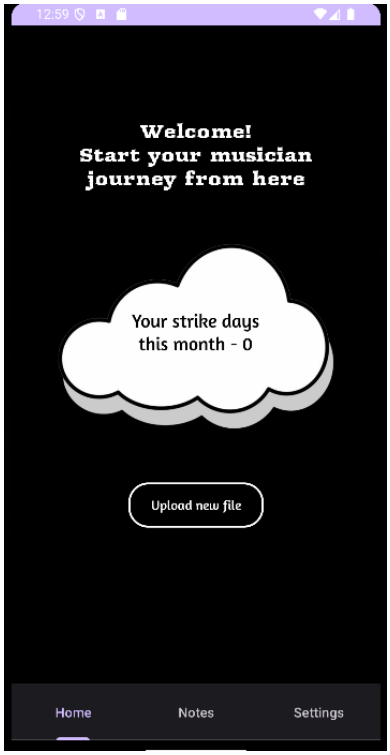


Рисунок 4.1 – Інтерфейс з кнопкою додавання PDF файлу з головної сторінки

Рисунок 4.2 – Інтерфейс з кнопкою додавання PDF файлу з нотної сторінки

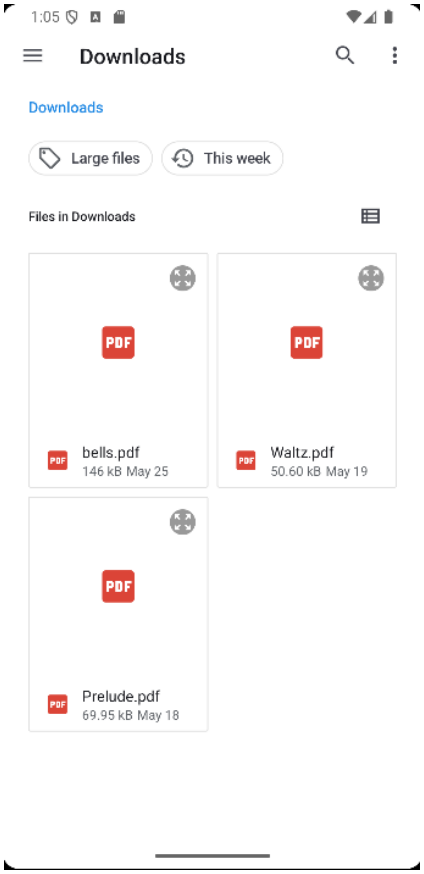


Рисунок 4.3 – Інтерфейс вибору PDF файлу для додавання

4.1.1.3 Функція зберігання PDF файлу (Рисунок 4.4).

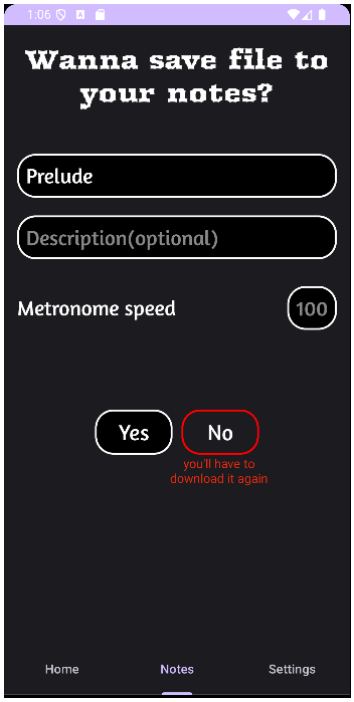


Рисунок 4.4 – Інтерфейс зберігання PDF файлу

4.1.1.4 Можливість вибору файлу для гри та редагування (Рисунок 4.5).

Рисунок 4.5 – Інтерфейс вкладки «Notes» з можливістю запустити програвання і редагування файлів

4.1.1.5 Функція пошуку по літері та сортування нот по даті, назві або опису файлу (Рисунок 4.6-4.7).

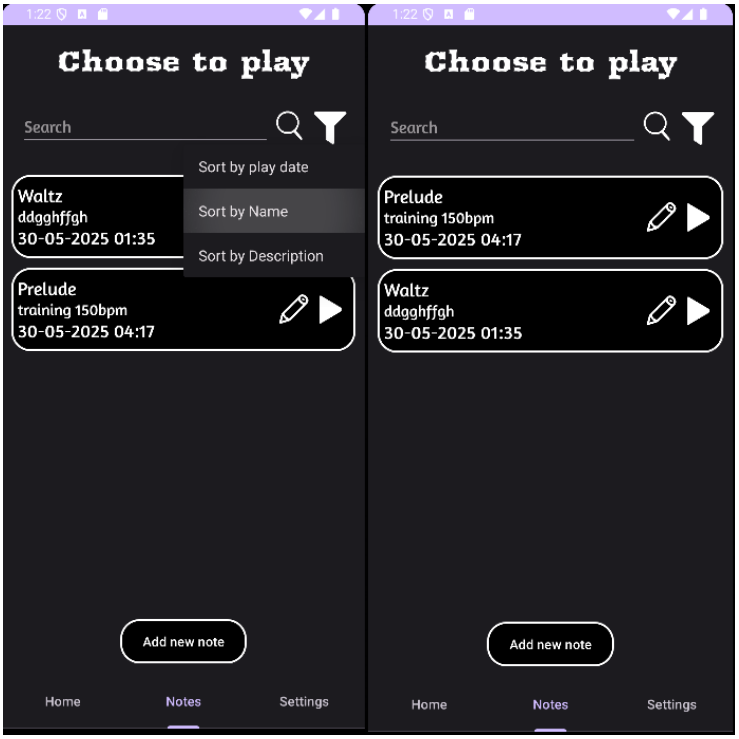


Рисунок 4.6 – Інтерфейс вкладки «Notes» з можливістю сортувати ноти за обраним критерієм

Рисунок 4.7 – Інтерфейс вкладки «Notes» з можливістю пошуку нотних записів за літерою

4.1.1.6 Можливість програвання і прослуховування файлу (Рисунок 4.8).

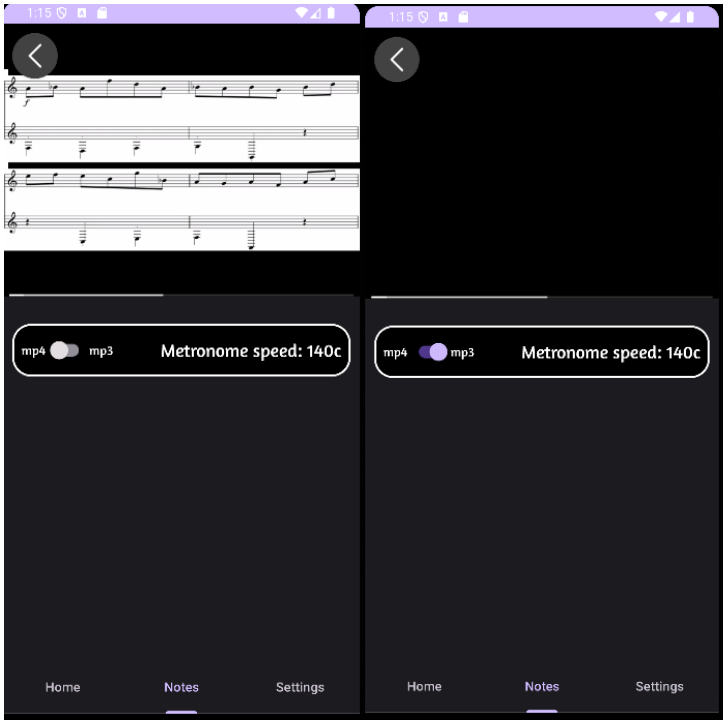


Рисунок 4.8 – Інтерфейс вкладки програвача

4.1.1.7 Функція редагування нотного запису (Рисунок 4.9).

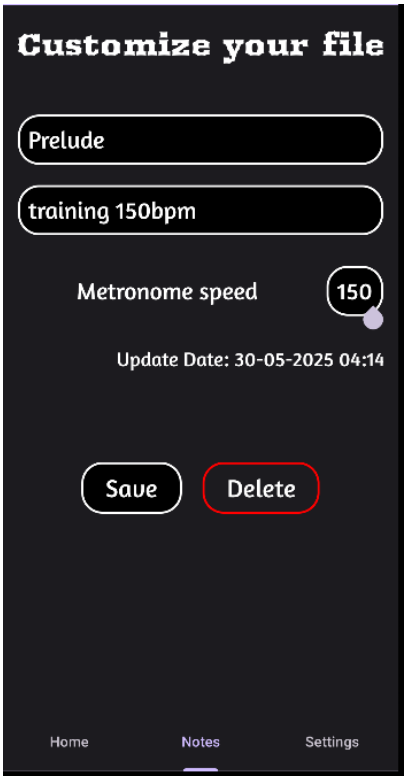


Рисунок 4.9 – Інтерфейс редагування назви, опису і швидкості метроному нотного запису з бібліотеки

4.1.1.8 Функція налаштування додатку під свої потреби – перемикання з світлої на темну тему й показ/хования статистики (Рисунок 4.10-4.12).

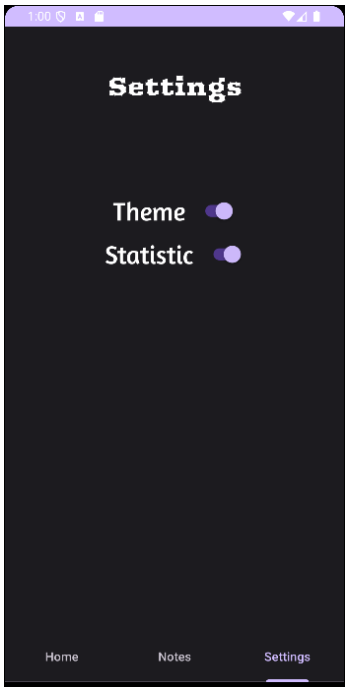


Рисунок 4.10 – Інтерфейс налаштувань додатку з включеною темною темою і статистикою

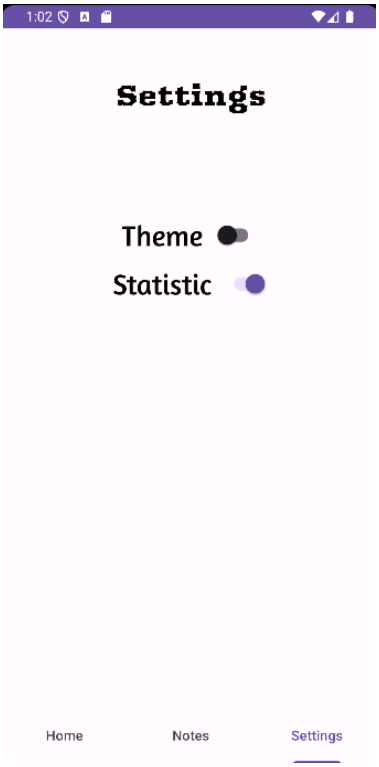


Рисунок 4.11 – Інтерфейс налаштувань додатку з включеною світлою темою і статистикою

Рисунок 4.12 – Інтерфейс налаштувань додатку з включеною світлою темою і виключеною статистикою, та вкладка «Home» без відображення статистики

4.1.2 Для користувача:

- обробка вхідних даних у форматі PDF;
- перетворення файлу у форматі PDF в оброблювані музичні формати MusicXML та MIDI;
- візуалізація нотної партитури;
- обробка синхронізації з метрономом;
- озвучення проаналізованого музичного твору;
- надання файлу із озвученням музичного твору;
- надання файлу з візуалізацією та звуком у форматі відео;

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в базі даних (якщо в ПЗ передбачена наявність БД).

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються.

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

Мінімальна конфігурація технічних засобів:

- оперативна пам'ять (RAM): 2 ГБ RAM.
- внутрішня пам'ять: 32 ГБ.
- версія Операційної Системи: Підтримка Android 6.0 (Marshmallow).

Рекомендована конфігурація технічних засобів:

- оперативна пам'ять (RAM): 6 ГБ RAM для забезпечення швидкої обробки даних та алгоритмів.
- внутрішня пам'ять: 128 ГБ внутрішньої пам'яті для зберігання додатку та записів.
- версія Операційної Системи: Підтримка Android версії вище 6.0 для забезпечення сумісності з широким спектром сучасних пристроїв.

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейства Android.

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні бути представлені в наступному форматі: PDF.

4.5.2 Вимоги до вихідних даних

Результати повинні бути представлені в наступному форматі: mp3, mp4.

4.5.3 Вимоги до мови розробки

Розробку виконати на мовах програмування Python, Java/Kotlin.

4.5.4 Вимоги до середовища розробки

Розробку виконати на платформах Java 8, Python 3.9.7.

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді програмного коду.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Згенерувати інсталяційну версію програмного забезпечення.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

– пояснювальна записка;

– технічне завдання;

– текст програми;

– програма та методика тестування;

– керівництво користувача;

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

– схема структурна варіантів використання;

– схема структурна компонентів програмного забезпечення;

– архітектура програмного забезпечення;

– креслення вигляду екранних форм;

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення рекомендованої літератури	15.03	
2.	Аналіз існуючих методів розв’язання задачі	29.03	Технічне завдання
3.	Постановка та формалізація задачі	18.04	Специфікації програмного забезпечення
4.	Розробка інформаційного забезпечення	28.04	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Алгоритмізація задачі	07.05	Тексти програмного забезпечення
6.	Обґрунтування вибору використаних технічних засобів	09.05	Тести, результати тестування
7.	Розробка програмного забезпечення	16.05	Пояснювальна записка
8.	Налагодження програми	24.05	Графічний матеріал проєкту
9.	Виконання графічних документів	27.05	Технічна документація
10.	Оформлення пояснювальної записки	01.06	

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Пояснювальна записка
до дипломного проєкту

на тему: **Мобільний застосунок для роботи з музичними нотами**

КПІ.ІП-1132.045440.02.81

Київ – 2025

Зміст

Перелік умовних позначень

IDE	–	Integrated Development Environment – інтегроване середовище розробки.
API	–	Application programming interface, прикладний програмний Інтерфейс.
ШІ	–	Штучний інтелект.
Mobile application	–	Мобільний додаток.
APK	–	"Android Package Kit". Формат архіву для розповсюдження та встановлення мобільних додатків на операційній системі Android.
БД	–	База даних.

Вступ

У сучасному світі, де ШІ визначає нові планки, до яких музична індустрія не лише адаптується, але й активно використовує для досягнення нових вершин у творчості та ефективності. За даних умов розвиток універсального інструменту, створеного для роботи з музичними нотами, представляє собою важливий крок в автоматизації завдань, пов'язаних із створенням та освоєнням музики.

Актуальність даної розробки визначається потребою в адаптації музичної індустрії до новітніх вимог шляхом застосування передових технологій. Використання ШІ у творчості відкриває нові можливості для артистів і аудиторії, роблячи цей процес більш доступним та захоплюючим.

На останніх етапах розвитку технологій вже можна спостерігати значні досягнення у взаємодії штучного інтелекту з музичною сферою. Алгоритми аналізу музичних структур, створення нових композицій та автоматизація процесів звукозапису є необхідними для еволюції цієї галузі.

Музиканти активно досліджують можливості застосування новітніх технологій в музичній індустрії. Розпізнавання музичних нот, аналіз структури композицій та написання унікальних творів – це лише деякі запити, які знаходять відгук.

Створений універсальний інструмент для роботи з музичними нотами може стати не тільки проривним рішенням для артистів, але й унікальним елементом для аналізу, організації та прослуховування музики. Його можливості знаходять застосування в різних аспектах мистецтва, забезпечуючи автоматизацію ключових базових процесів і простір для творчого розкриття музикантів.

1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка завдання дипломного проєктування

У ході детального вивчення області дослідження було визначено перелік вимог, які є важливою складовою у розробці програмної системи. Аналіз цих аспектів є ключовим етапом у формуванні характеристик та обсягу майбутнього продукту. Цей перелік охоплює як функціональні, так і нефункціональні процеси, необхідні для успішної реалізації бізнес-цілей та відповіді на потреби користувачів. Аналіз, проведений на даному етапі, закладає міцний фундамент для подальших кроків у проєктуванні та розробці програмної системи.

В умовах сьогодення нововведення у сфері музикантів найчастіше відбувається за допомогою алгоритмів розпізнавання образів та обробки аудіосигналів. Однак, незважаючи на досягнення, існують деякі прогалини.

Недоліками поточного стану речей є обмежена точність розпізнавання, особливо при роботі зі складними музичними структурами, через перетворення нотної грамоти на інтерактивну гру. Також додатки, зазвичай, спрямовані на часткове покращення ефективності робочих процесів.

У контексті дипломного проєкту вирішення цих недоліків є одним з основних завдань. Інтеграція штучного інтелекту та нейронних мереж стає одним з головних допоміжних інструментів. Було важливо розширити функціональність програми, щоб охопити якомога більше можливостей використання та надати користувачу інструмент для оперування ними в одному додатку, не шукаючи додаткових аналогів.

1.2 Аналіз предметної області

Сьогодення відзначається стрімким розвитком IT-технологій, що впливає на всі сфери життя, не оминаючи музичну творчість. У мобільних додатках активно використовують технології для калібрування інструментів,

розпізнавання нот і загалом намагаються спростити освоєння музики та посприяти розкриттю творчого потенціалу людей.

На сучасному етапі розвитку технологій вже можна спостерігати значні досягнення у використанні штучного інтелекту в музичній сфері. Важливими етапами еволюції цієї галузі є алгоритми аналізу музичних структур, створення нових композицій та автоматизація процесів звукозапису.

Сучасне положення речей є таким, що інструментів, які мають повний цикл обробки музичних файлів не так багато, тож є нагальна потреба в універсальному та зручному додатку, який би забезпечував користувача усіма програмними модулями, необхідними для розбору творів, гри та подальшого розвитку.

Новітній універсальний інструмент для роботи з музичними нотами може стати не тільки інноваційним рішенням для артистів, але й вирішальним елементом для аналізу, організації та створення творів. Такі можливості знаходять застосування в різних аспектах творчості, забезпечуючи автоматизацію ключових базових процесів і простір для творчого розкриття музик. Мобільний застосунок – це крок назустріч музикантам, вчителям, композиторам, це допомога як для тих, хто тільки починає свою кар’єру в музичному світі, так і для тих, хто вже не раз підкорював світ своєю творчістю. Дана розробка, оброблюючи нотні тексти, надає змогу людині за музичним інструментом грати, не відволікаючись на сторонні завдання, які зробить за неї цей музичний асистент.

Пропонований проєкт також може знайти застосування в області музикотерапії. Музична терапія – це область досліджень, якій останнім часом приділяють багато уваги, у зв’язку зі зростанням попиту цього напрямку. Застосунок також може бути використаний для розробки інтелектуального мультимедійного інструменту, якому можна знайти місце в сфері охорони здоров'я. У рамках цієї роботи може бути створено цифрову музичну бібліотеку, яка буде складатися з різних пісень та відповідних цілющих можливостей з точки зору музичної терапії. З погляду на різні сфери,

застосунок можна розширити системою рекомендацій для різних цілей, зокрема для навчання, розваг і здоров’я.

Додаток спрямовано на підвищення якості та універсальності мобільних застосунків для музикантів, забезпечуючи їм зручний і високоефективний інструмент для роботи з нотами. Головною метою є об’єднання усього, необхідного митцям для розвитку, в одному додатку.

1.3 Аналіз існуючих рішень

Проведемо аналіз відомих на сьогодні алгоритмічних та технічних рішень у даній області, що допоможуть у реалізації мобільного застосунку для роботи з музичними нотами. Далі будуть розглянуті готові програмні рішення, допоміжні програмні засоби та засоби розробки.

Перед початком розробки нового інформаційного технологічного проєкту, обов'язковим етапом є детальний аналіз відповідних існуючих на ринку. Цей процес визначається як необхідний для ефективного планування та успішної реалізації інноваційних ідей. Перш ніж розпочати новий етап, потрібно детально вивчити існуючі та подібні додатки, здійснюючи аналіз їхніх підходів, реалізацій, досягнень, а також помилок.

Важливість аналізу відповідних IT-проєктів визначається кількома ключовими аспектами:

- навчання на помилках:
  Вивчення існуючих проєктів дозволяє уникнути типових помилок і невдач, що ефективно економить час та ресурси організації.
- визначення технологічних трендів:
  Аналіз проєктів допомагає визначити актуальні технологічні тренди та інновації, які можуть використовуватися для покращення нового проєкту.
- визначення кращих практик:

Спостереження за успішними проєктами допомагає виявити кращі практики та стратегії, які можна впроваджувати у новому проєкті.

- оцінка споживчої потреби:
  Вивчення інших проєктів допомагає зрозуміти потреби та очікування користувачів, що є ключовим для створення продукту, який задовольнить їхні вимоги.
- уникнення дублювання робіт:
  Аналіз існуючих проєктів дозволяє уникнути дублювання робіт та ресурсів, використовуючи вже існуючі рішення та компоненти.
- підвищення шансів на успіх:
  Глибокий аналіз попереднього досвіду допомагає підготувати план дій, що збільшує ймовірність успіху нового проєкту та зменшує ризики.

Отже, дослідження подібних проєктів є необхідним етапом, спрямованим на створення оптимальних умов для успішної реалізації нового, заснованого на здобутих досвідченими гравцями рішеннях та стратегіях. Цей процес сприяє удосконаленню планування, зменшенню ризиків і підвищенню ймовірності досягнення мети.

1.3.1 Аналіз відомих програмних продуктів

Порівняльний аналіз з аналогами буде проведено у площині додатків, які напряму пов’язані з нотами та їх аналізом.

Перший аналог це Yousician[4] - популярний музичний додаток, спрямований на вдосконалення гри на різних музичних інструментах. Зокрема, додаток фокусується на грі на гітарі, фортепіано, бас-гітарі та укулеле. Він використовує інтерактивні завдання та розпізнавання звуку для покращення музичних навичок користувачів (Рисунок 1.1).

Yousician використовує технологію розпізнавання звуку для визначення точності виконання нот та акордів. Це дозволяє користувачам отримати

зворотний зв'язок та коригувати свою гру.

Додаток включає в себе метроном, який допомагає користувачам розвивати свої ритмічні навички та втілювати стабільність темпу.

Крім вивчення нот, Yousician пропонує користувачам можливість слухати та аналізувати різні музичні композиції для розвитку відчуття стилю та інтерпретації.

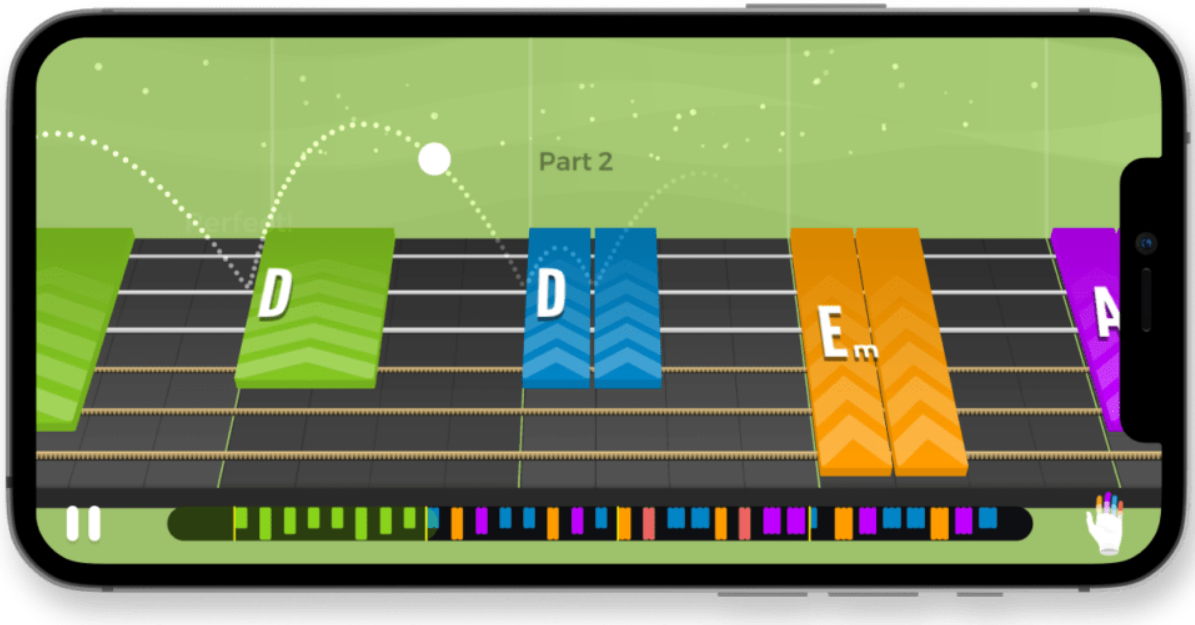


Рисунок 1.1 - Приклад завдання Yousician

Другий аналог це Musicnotes[5] - платформа, яка спеціалізується на наданні доступу до цифрових нот та нотних записів (Рисунок 1.2). Розроблена з урахуванням потреб музикантів на різних рівнях навичок та стилів.

Musicnotes надає доступ до широкого вибору електронних нот та аранжувань для різних інструментів і голосу, що дозволяє музикантам знайти та вивчати композиції в різних стилях.

Користувачі можуть користуватися функцією транспонування для зручності гри на своєму інструменті.

Musicnotes дозволяє прослуховувати та переглядати аудіо- та відео інтерпретації пісень, що сприяє кращому розумінню виконання та інтерпретації.

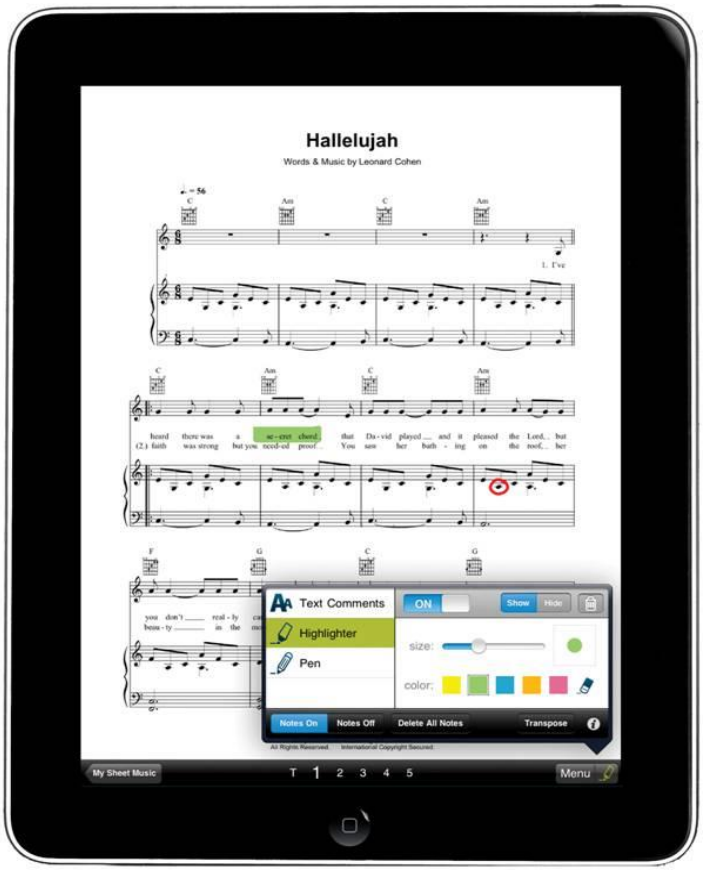


Рисунок 1.2 - Приклад нотних записів Musicnotes

Третій аналог це Simply Piano[6] - інтерактивний музичний додаток, розроблений для вивчення та вдосконалення гри на фортепіано. Розрахований на широку аудиторію, від початківців до досвідчених музикантів, Simply Piano пропонує інтуїтивно зрозумілий підхід до вивчення музики через інтерактивні уроки та ігровий процес (Рисунок 1.3).

Система розпізнавання звуку Simply Piano аналізує гру користувача та визначає точність виконання акордів.

Пропонує інтерактивні уроки, які адаптовані до індивідуального рівня навичок кожного користувача. Уроки охоплюють різні стилі музики та аспекти гри на фортепіано.

Додаток включає в себе метроном для покращення ритмічних навичок гравця.

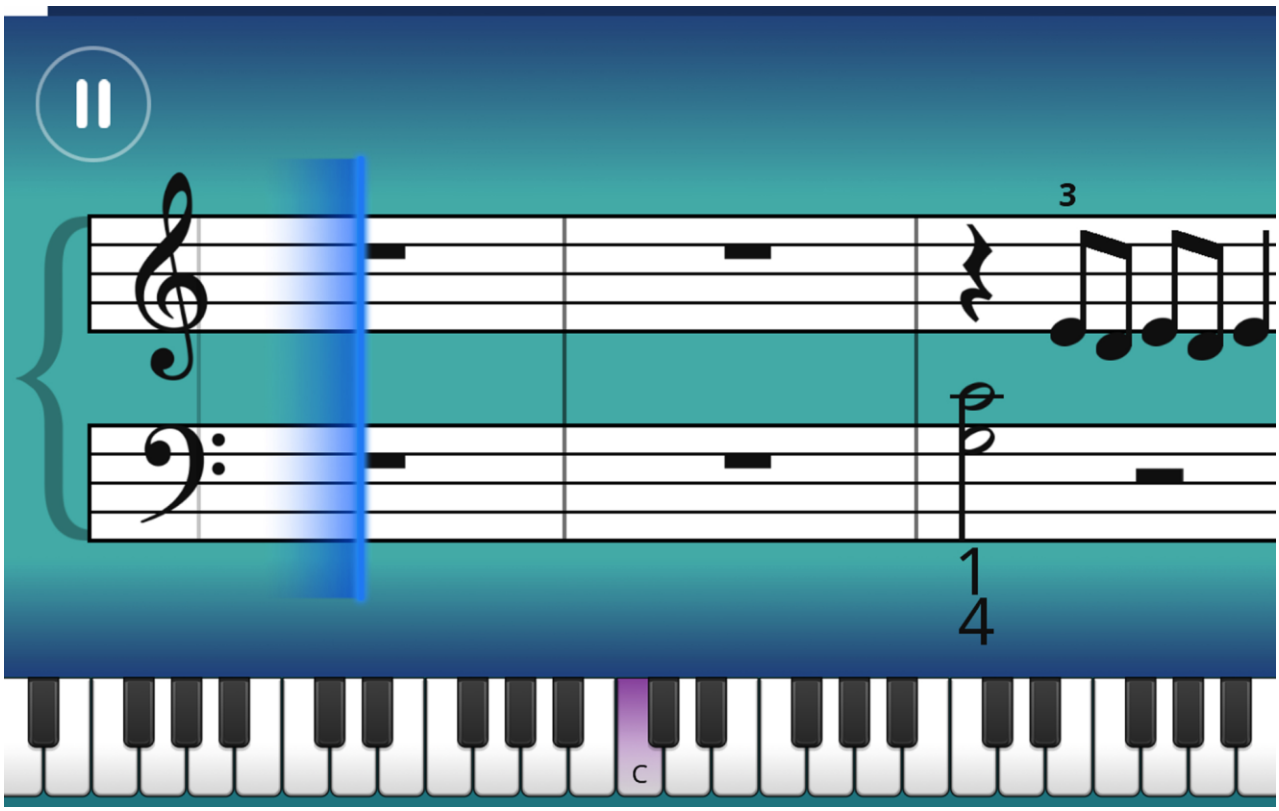


Рисунок 1.3 - Приклад інтерактивного уроку Simply Piano

Для порівняння проєкту з аналогом можна скористатись таблицею 1.1.

Таблиця 1.1 – Порівняння з аналогами

Функціонал	**Liszt**	**Yousician**	**Musicnotes**	**Simply Piano**
Завантаження власних файлів	Так	Ні	Так	Так
Наявний повний безкоштовний доступ	Так	Ні	Ні	Ні
Метроном	Так	Так	Ні	Так
Зберігання та організація файлів	Так	Ні	Так	Так
Прослуховування композицій	Так	Так	Ні	Ні

Продовження таблиці 1.1

Функціонал	**Liszt**	**Yousician**	**Musicnotes**	**Simply Piano**
Завантаження композицій для прослуховування	Так	Ні	Ні	Ні
Показ нот на екрані відповідно до налаштувань для гри	Так	Ні	Ні	Ні

Проведено аналіз властивостей різних музичних програм: Yousician, Simply Piano, Musicnotes і Liszt. Важливо відзначити, що кожна з цих програм володіє унікальними особливостями та функціоналом, але одночасно вони відрізняються за наявністю чи відсутністю певних можливостей.

Узагальнюючи отримані дані, слід відзначити, що програма "Liszt" виділяється своєю комплексною підтримкою для музикантів. Вона поєднує важливі функції розпізнавання нот, зручний метроном, систему організації файлів та можливість прослуховування композицій. Все це робить "Liszt" високоефективним інструментом для музичного навчання та творчості.

1.3.2 Аналіз відомих алгоритмічних та технічних рішень

Сучасні вимоги до програмного забезпечення для обробки музичних нот вимагають від розробників уважного вивчення існуючих алгоритмів, їхньої адаптації для використання у специфічних умовах. В цьому розділі буде виконано аналіз передових алгоритмів та технічних рішень, що забезпечують високий рівень точності та продуктивності у роботі з музичними даними. Розглянемо ключові аспекти, що формують технічний ландшафт у даній галузі.

MIDI (Musical Instrument Digital Interface)[7]:

- використовується для кодування музичної інформації, такої як ноти, в технічний формат.

- дозволяє зберігати та передавати інформацію про ноти, їхню динаміку, темп тощо.
- широко використовується у музичних програмах та додатках.

MusicXML[8]:

- визначає стандарт для обміну музичною інформацією в текстовому форматі.
- дозволяє зберігати і передавати дані про ноти, акорди, темп, але в текстовому вигляді.
- забезпечує чітку структуру для аналізу та обробки нот.

Optical Music Recognition (OMR)[9]:

- використовується для розпізнавання нот та інших символів на паперових нотних аркушах за допомогою зображення.
- вимагає використання комп'ютерного зору для аналізу зображення нотного аркуша.
- застосовується в ситуаціях, де необхідно розпізнати ноти з фізичних нотних паперів.

Оптичне розпізнавання нот (OCR)[10]:

- використовує алгоритми оптичного розпізнавання для перетворення зображення нотного аркуша у текст з нотами.
- вимагає точного зображення для ефективної роботи.
- використовуються алгоритми для розпізнавання частот та часових параметрів звучання.
- можуть враховувати динаміку та характеристики різних музичних інструментів.

FFT (Fast Fourier Transform)[11]:

- використовується для аналізу частот в аудіосигналі та визначення основних нот.
- допомагає в перетворенні аудіосигналу з нотного файлу в частотні характеристики.

Wavelet Transform[12]:

- використовується для аналізу амплітуд та частот на різних рівнях розгортання.
- забезпечує деталізовані результати аналізу аудіосигналу.

Ці алгоритмічні та технічні рішення є важливими для розробки мобільного додатку для розпізнавання нот і розтрактовки, оскільки вони визначають ефективність, точність та функціональність такого додатку. Комбінація цих методів може забезпечити високоякісний та універсальний інструмент для музикантів та творчих особистостей.

У даному проєкті основна увага приділяється реалізації оптичного розпізнавання музичних нот (OMR)[9], програмним засобам для редагування та відтворення музики, рендерінгу зображень на основі музичних файлів, а також технологіям синтезу звуку та створення відео на основі звукової доріжки та зображень. Ключовими інструментами, що розглядаються, є Audiveris[13] для OMR[9], MuseScore[14], як інструмент для редагування файлів формату MusicXML[8] та SoundFont[15] для синтезу звуку.

Audiveris[13] визначається як передовий інструмент OMR[9], який здатний перетворювати скановані партитури у цифрові формати. Це особливо корисно для музикантів і композиторів, які працюють з великими обсягами паперових нотних записів. Інтегруються складні алгоритми машинного навчання, що дозволяють з високою точністю розпізнавати музичні символи та елементи нотації. Ці алгоритми постійно вдосконалюються, враховуючи широкий спектр стилів та форматів нотних записів. За допомогою цього інструменту, значно автоматизується процес перетворення фізичних нотних аркушів у цифровий формат. Знижується потреба в ручному введенні нот і спрощується процес цифрової обробки.

Додаток здатний експортувати зчитанні дані у формати, що сумісні з багатьма музичними редакторами, включаючи MuseScore[14]. Це дозволяє легко інтегрувати Audiveris[13] у різні музичні робочі процеси. Незважаючи на складність музичних партитур, демонструється висока точність у

розпізнаванні різноманітних музичних символів, включаючи ноти, паузи, ключі, тактові риски та інші елементи нотації. Використання машинного навчання не лише підвищує поточну ефективність утиліти, але й відкриває можливості для подальшого вдосконалення на основі нових даних і зразків музики. Audiveris[13] є незамінним інструментом у сфері музичної обробки, оскільки його можливість адаптуватися до різних стилів нотації та висока точність роблять його надзвичайно цінним для широкого спектру задач і музичних застосувань.

MuseScore[14] є інтуїтивно зрозумілим та потужним інструментом для редагування нот, який використовується музикантами, композиторами та аранжувальниками в світовому масштабі. Цей програмний засіб дозволяє легко вносити зміни та візуалізувати музичні твори, що робить його ідеальним для детальної роботи над музичними композиціями після їх первісного розпізнавання, наприклад, як за допомогою Audiveris[13]. Однією з ключових переваг є його висока гнучкість у редагуванні нот, а також підтримка широкого діапазону музичних символів та нотацій, включаючи різні ключі, тактові позначення, нотні знаки. Також важливим моментом є здатність візуалізувати складні музичні структури, такі як поліфонія або комплексні ритмічні патерни.

Крім того, функціональність MuseScore[14] включає можливість експортування музики в 149 різні формати, включаючи PDF, MIDI та MusicXML[8], які безпосередньо використовуються у проєкті, що забезпечує легку інтеграцію з іншими музичними програмами та платформами. Ця гнучкість робить MuseScore[14] важливим інструментом у музичній продукції та обробці. В даній розробці MuseScore[14] використано для внесення змін у файл формату MusicXML[8].

SoundFont[15] є ключовим інструментом у сфері цифрового аудіо, оскільки він надає різноманітні та якісні звуки для синтезу музики. Це технологія, яка дозволяє музикантам та продюсерам використовувати

розширений набір звукових бібліотек, що імітують різні музичні інструменти, від традиційних до електронних.

Основна його перевага полягає у здатності точно відтворювати різні музичні інструменти, надаючи користувачам широкий спектр звуків, з якими вони можуть творити. Це означає, що музиканти можуть експериментувати з різними тембрами та стилями, не обмежуючись фізичною наявністю інструментів. Технологія SoundFont[15] також сприяє творчій свободі, оскільки вона дозволяє легко змінювати та налаштовувати вже існуючі звуки. Наприклад, користувачі можуть змінювати висоту тону, гучність, реверберацію та інші параметри, щоб досягти бажаного звучання. Це робить цей інструмент ідеальним для створення унікальних та індивідуальних звукових ландшафтів. Однією з унікальних особливостей є можливість створення власних звукових банків, що надає користувачам можливість створювати власні, індивідуальні звукові колекції. Це забезпечує необмежені можливості для експериментів та інновацій у музиці. Використання SoundFont[15] у музичному проєкті значно розширює творчі горизонти, оскільки воно дозволяє музикантам та продюсерам досліджувати нові звукові текстури та створювати більш багатогранні музичні композиції. В даній розробці SoundFont[15] використано для озвучення нот, переданих користувачем початково у PDF форматі.

При порівнянні цих інструментів, Audiveris[13] вирізняється своєю спроможністю точно розпізнавати музичні ноти та символи з паперових нотних записів. MuseScore[14] ж відрізняється зручністю в редагуванні музичних творів, а SoundFont[15] – різноманітністю звуків для синтезу. Кожен з цих інструментів відіграє ключову роль у різних стадіях розробки музичного проєкту. Беручи до уваги специфіку проєкту, використано Audiveris[13] для первинного розпізнавання нот, MuseScore[14] для подальшого редагування музики, і SoundFont[15] для фінального синтезу звуку. Ця комбінація забезпечує ефективний потік роботи від сканування нот до відтворення музики.

1.4 Аналіз та моделювання бізнес-процесів

Для опису бізнес процесу програмного забезпечення використовується UML Sequence Diagram (Рисунок 1.4).

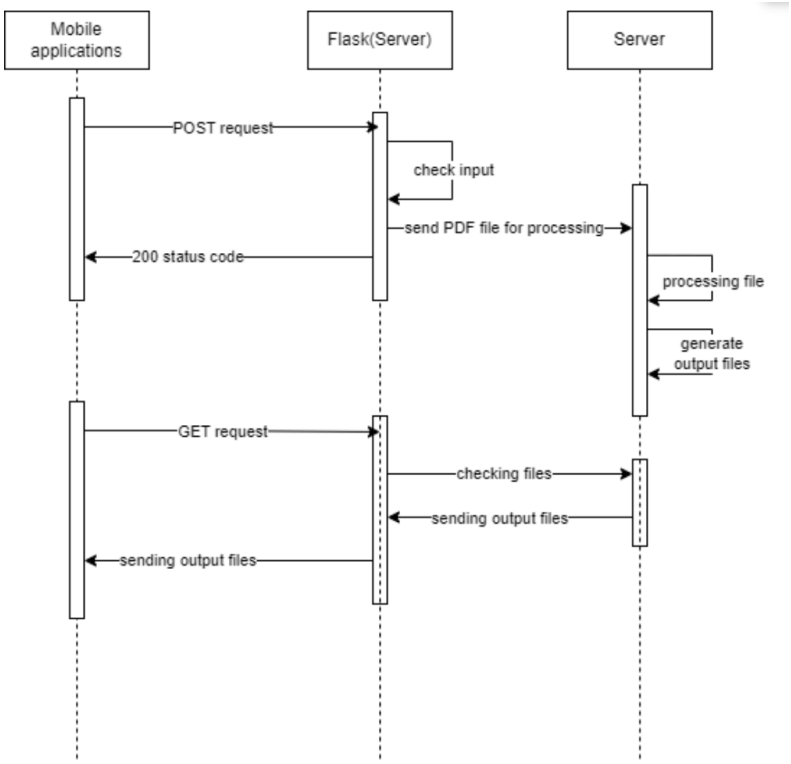


Рисунок 1.4 – Модель бізнес-процесу

Опис послідовності базового потоку комунікації сервера та мобільного додатку:

- додаток надсилає запит із файлом на обробку;
- сервер(Flask) перевіряє коректність вхідних даних;
- сервер починає обробку файлу;
- сервер надсилає підтвердження, що файл прийнято в обробку;
- мобільний додаток надсилає запит на отримання вихідних даних;
- сервер(Flask) перевіряє готовність даних;
- дані відправляються на мобільний додаток;

Висновки до розділу

У цьому розділі було проведено всебічний аналіз предметної області,

враховуючи загальні положення, технологічні досягнення, алгоритмічні та технічні рішення, а також допоміжні засоби та існуючі програмні продукти. Ретельний аналіз допоміг ідентифікувати важливі аспекти, які будуть враховані при розробці програмного забезпечення.

Визначено успішні IT-проєкти, що надало важливі висновки для власного проєкту. Докладно проаналізовано існуючі алгоритмічні та технічні рішення у сфері розпізнавання нот та обробки музичної інформації.

Вивчено допоміжні програмні засоби, а також програмні продукти, що забезпечують аналогічний функціонал. Цей аналіз визначив потребу у створенні продукту, який буде конкурентоспроможним та відповідатиме вимогам сучасного ринку.

2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Варіанти використання програмного забезпечення

Основною функцією програмного забезпечення є обробка музичних нот для подальшого відтворення їх у відповідь на задану швидкість метроному. Більше функцій можна оглянути на Рисунку 2.1.

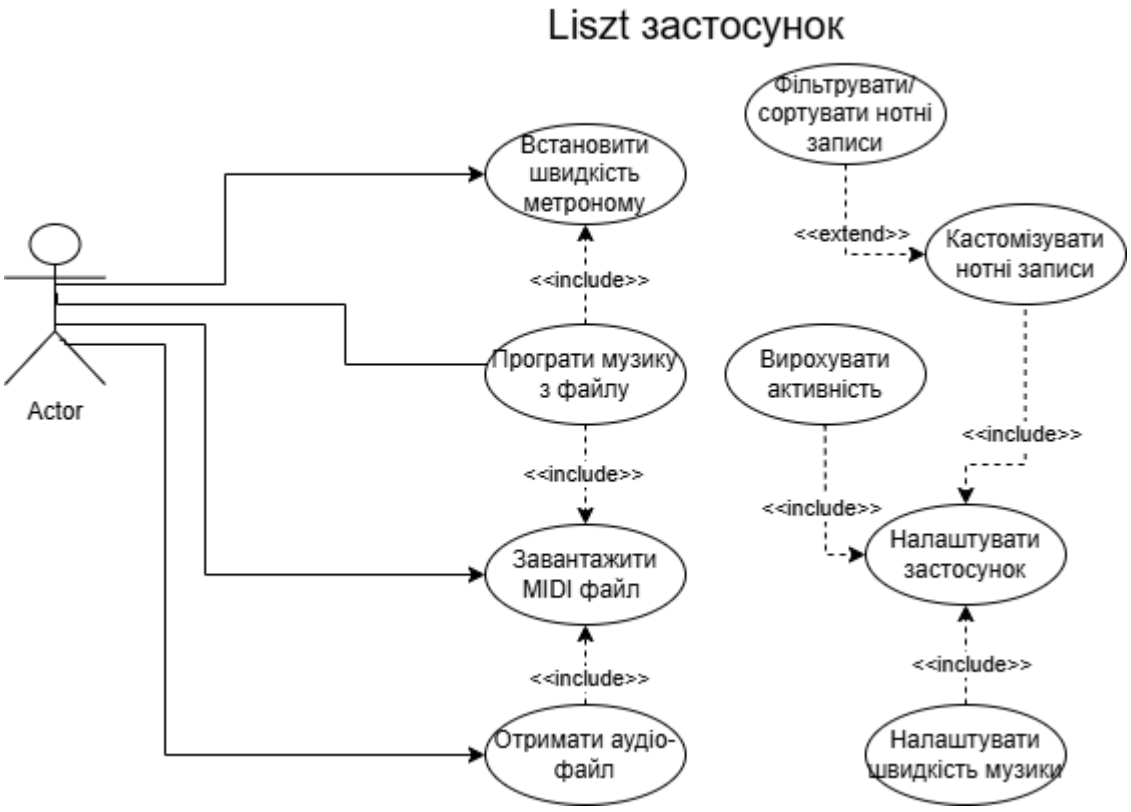


Рисунок 2.1 – Діаграма варіантів використання

В таблицях 2.1-2.7 наведено опис варіантів використання програмного забезпечення.

Таблиця 2.1 – Варіант використання UC-01

Use case name	Додати PDF-файл з нотами.
Use case ID	UC-01
Goals	Дозволити користувачам додавати PDF-файли з музичними нотами до програми.
Actors	Користувач

Продовження таблиці 2.1

Trigger	Користувач вибирає опцію «Завантажити файл».
Pre-conditions	Користувач має файли на пристрої
Flow of Events	1. Користувач вибирає PDF-файл зі свого пристрою. 2. Користувач вибирає налаштування метронома. 3. Користувач вибирає, чи хоче зберегти файл до бібліотеки. 4. Програма обробляє та додає файли mp4 та mp3. 5. Користувач відтворює ноти.
Extension	Якщо вирішите не зберігати файл, то можете взаємодіяти з ним, доки не вийдете назад. Якщо будь-яке поле введено неправильно, про це буде повідомлено.
Post-Condition	Користувач може відтворювати mp4 та mp3.

Таблиця 2.2 - Варіант використання UC-02

Use case name	Зберегти змінений PDF-файл.
Use case ID	UC-02
Goals	Дозволити користувачам зберігати змінений файл.
Actors	Користувач
Trigger	Користувач вибирає опцію «Зберегти».
Pre-conditions	PDF-файл було змінено.
Flow of Events	1. Користувач змінив файл. 2. Користувач вибирає опцію «Зберегти». 3. Додаток зберігає файл.
Extension	Помилка, якщо ім'я порожнє.
Post-Condition	Файл успішно збережено.

Таблиця 2.3 - Варіант використання UC-03

Use case name	Вибір файлу для відтворення.
Use case ID	UC-03
Goals	Дозволити користувачам вибирати PDF-файл для відтворення.
Actors	Користувач
Trigger	Користувач вибирає файл для відтворення.
Pre-conditions	PDF-файли доступні в додатку.
Flow of Events	1. Користувач вибирає PDF-файл. 2. Користувач встановлює параметри метронома, якщо необхідно. 3. Додаток відтворює ноти.
Extension	Помилка, якщо неможливо обробити файл.
Post-Condition	Файл відтворюється.

Таблиця 2.4 - Варіант використання UC-04

Use case name	Пошук/сортування нотаток
Use case ID	UC-04
Goals	Дозволити користувачам шукати та сортувати нотатки на основі назви або опису.
Actors	Користувач
Trigger	Користувач вибирає опцію «Пошук/Сортування».
Pre-conditions	Користувач має файли з нотатками в додатку.
Flow of Events	1. Користувач вибирає критерії (назва/опис). 2. Додаток відображає відсортовані/знайдені нотатки.
Extension	Пусто, якщо немає пов’язаних файлів.
Post-Condition	Нотатки успішно знайдено або відсортовано на основі критеріїв користувача.

Таблиця 2.5 - Варіант використання UC-05

Use case name	Прослуховування файлу.
Use case ID	UC-05
Goals	Прослуховування файлу.
Actors	Користувач
Trigger	Користувач вибирає файл для прослуховування.
Pre-conditions	Файл присутній у програмі.
Flow of Events	1. Користувач вибирає файл для прослуховування. 2. Програма відтворює mp3-файл
Extension	Помилка, якщо не вдається обробити mp3.
Post-Condition	Користувач успішно прослуховує вибраний файл.

Таблиця 2.6 - Варіант використання UC-06

Use case name	Статистика активності.
Use case ID	UC-06
Goals	Перегляд статистики активності.
Actors	Користувач
Trigger	Користувач переходить до розділу статистики.
Pre-conditions	Додаток запущено.
Flow of Events	1. Користувач переходить до розділу статистики, щоб переглянути дані про активність. 2. Додаток показує статистику за місяць.
Extension	Якщо статистика вимкнена, даних немає.
Post-Condition	Користувач успішно переглядає статистику активності.

Таблиця 2.7 - Варіант використання UC-07

Use case name	Налаштування програми.
Use case ID	UC-07
Goals	Налаштування параметрів програми.
Actors	Користувач
Trigger	Користувач отримує доступ до налаштувань програми.
Pre-conditions	Програма працює.
Flow of Events	1. Користувач отримує доступ до налаштувань програми. 2. Користувач налаштовує параметри програми відповідно до своїх уподобань. 3. Програма змінила налаштування.
Extension	-
Post-Condition	Налаштування програми успішно налаштовано користувачем.

2.2 Розроблення функціональних вимог

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. В таблиці 2.8 наведено загальну модель вимог, а в таблиці 2.9 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити в таблиці 2.10.

Таблиця 2.8 – Загальна модель вимог

№	Назва	ID Вимоги	Пріоритети	Ризики
1	Завантаження головної сторінки додатку	FR-1	Високий	Середній
2	Можливість завантажити файл	FR-2	Високий	Середній
3	Можливість зберегти файл у бібліотеку	FR-3	Високий	Середній

Продовження таблиці 2.8

№	Назва	ID Вимоги	Пріоритети	Ризики
4	Можливість серверу обробки клієнтських запитів	FR-4	Високий	Високий
5	Можливість встановити швидкість метроному для файлу	FR-5	Середній	Середній
6	Можливість програвати ноти у mp4 і mp3 форматах	FR-6	Високий	Високий
7	Можливість передивлятись статистику	FR-7	Середній	Середній
8	Можливість кастомізувати збережені ноти	FR-8	Середній	Середній
9	Можливість змінити тему додатку	FR-9	Низький	Низький
10	Можливість сортувати бібліотеку нот	FR-10	Середній	Середній
11	Можливість пошуку у бібліотеці нот	FR-11	Середній	Середній
12	Можливість користувача переключатись між екранними формами	FR-12	Середній	Середній
13	Можливість бачити підказки у випадку некоректних дій	FR-13	Середній	Низький

Таблиця 2.9 – Перелік функціональних вимог

Назва	Опис
FR-1	Має бути головна сторінка та можливість переходу на інші вікна
FR-2	Має бути можливість для клієнтської частини завантажувати PDF файл з нотами
FR-3	Має бути можливість для клієнтської частини створювати запити до сервера на отримання даних при завантаженні файлу
FR-4	Передбачає задоволення вимог FR-2, FR-3 та FR-6
FR-5	Користувач повинен мати можливість встановити кастомну швидкість метроному
FR-6	Користувач повинен мати можливість подивитися завантажені ноти як у відео, так і аудіо
FR-7	Користувач повинен мати можливість передивитися статистику за місяць
FR-8	Користувач повинен мати можливість змінити назву, опис нот з бібліотеки
FR-9	Користувач повинен мати можливість змінити тему додатку
FR-10	Користувач повинен мати можливість сортувати ноти за ім’ям, описом, датою
FR-11	Користувач повинен мати можливість шукати ноти за ім’ям, описом
FR-12	Користувач повинен мати можливість переключатись між екранними формами
FR-13	Користувач повинен мати можливість бачити підказки, якщо робить некоректні дії

Таблиця 2.10 – Матриця трасування вимог

	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7
FR-1	+						
FR-2	+						
FR-3	+	+	+		+		
FR-4							
FR-5		+	+				
FR-6					+		
FR-7	+						
FR-8		+					
FR-9							+
FR-10			+	+			
FR-11			+	+			
FR-12	+	+	+	+	+	+	+
FR-13	+	+	+	+	+		

2.3 Розроблення нефункціональних вимог

Реалізація має задовольняти наступні тезиси:

- додаток повинен бути простим для використання та мати приємний дизайн;
- додаток повинен бути адаптованим до різних мобільних пристроїв;
- додаток повинен бути адаптований для різних новітніх версій;
- програмне забезпечення повинне бути адаптоване для простого додавання нового функціоналу;
- інтерфейс має бути простим, інтуїтивно зрозумілим для користувача;

- інтерфейс має підтримувати шляхи поведінки при внутрішніх збоях сервісу чи помилках користувача;
- інтерфейс має відображати бібліотеку у зручному форматі (поділ на підкатегорії);
- сервер повинен бути адаптивний до внесення змін або нового функціоналу, не змінюючи при цьому спілкування з мобільним застосунком;
- сервер має унікально ідентифікувати файли, з якими він працює;
- сервер має бути доступним для зовнішніх (мобільних) пристроїв;

2.4 Аналіз системних вимог

Мінімальна конфігурація технічних засобів:

- оперативна пам'ять (RAM): 2 ГБ RAM.
- внутрішня пам'ять: 32 ГБ.
- версія Операційної Системи: Підтримка Android 6.0 (Marshmallow).

Рекомендована конфігурація технічних засобів:

- оперативна пам'ять (RAM): 6 ГБ RAM для забезпечення швидкої обробки даних та алгоритмів.
- внутрішня пам'ять: 128 ГБ внутрішньої пам'яті для зберігання додатку та записів.
- версія Операційної Системи: Підтримка Android версії вище 6.0 для забезпечення сумісності з широким спектром сучасних пристроїв.

2.5 Аналіз економічних показників програмного забезпечення

У цьому підрозділі оцінюється економічна ефективність розробки програмного забезпечення. Цей аналіз є важливою частиною будь-якого IT-

проєкту, оскільки дозволяє обґрунтувати витрати на розробку продукту, визначити необхідні ресурси, спрогнозувати терміни виконання та оцінити потенційні економічні ризики.

Для аналізу економічних показників розробки програмного забезпечення буде використано модель COCOMO (Constructive Cost Model)[16], яка є класичним емпіричним підходом для прогнозування зусиль та часу. Вона дозволить отримати кількісні показники щодо зусиль, часу та вартості розробки з урахуванням реальних умов та ресурсів.

Проєкт відноситься до типу Organic, оскільки кількість рядків < 25 тисяч, тому використовуємо відповідні значення формул:

Тип проєкту	ab	bb	cb	db
Органічний	2.4	1.05	2.5	0.38

Рисунок 2.2 – Значення типу Organic для формул.

– розрахунок зусиль:

$$Effort = ab * \frac{LOC^{bb}}{1000} = 2{,}4 * \frac{1288^{1{,}05}}{1000} \approx 4{,}42204 \ \frac{\text{людина}}{\text{місяць}}$$

, де Effort – це зусилля в людино-місяцях, LOC – кількість рядків коду, ab та bb – емпіричні коефіцієнти, які залежать від типу проєкту.

– розрахунок вартості:

$$Cost = Effort * salary(avg\ junior) = 4{,}42204 * 15000$$
$$= 66330{,}6 \ \frac{\text{людина грн}}{\text{місяць}}$$

, де $Cost$ – це вартість в людино-грн-місяцях, $salary(avg\ junior)$ – середня зарплатня молодшого спеціаліста у грн.

– розрахунок часу на розробку:

$$Time = cb * Effort^{db} = 2{,}5 * 4{,}42204^{0{,}38} = 4{,}39822 \text{ розробників}$$

, де $Time$– це час на розробку, який вимірюється в кількості людей-робітників.

Оскільки кількість рядків коду (LOC) не перевищувала 25000, проєкт було класифіковано як "Органічний".

Було отримано наступні результати:

- оцінка зусиль: приблизно 4,42 людино-місяця;
- оцінка вартості: приблизно 66330,6 грн;
- оцінка часу розробки: приблизно 4,4 особи;

Таким чином, підхід COCOMO[16] забезпечив обґрунтовану теоретичну оцінку вартості проєкту та робочого навантаження, яка може бути використана для майбутнього бюджетування, розподілу ресурсів та управління ризиками на ранніх стадіях життєвого циклу програмного забезпечення.

2.6 Постановка завдання на розробку програмного забезпечення

Шляхом детального вивчення області дослідження було визначено перелік вимог, які є важливою складовою у розробці програмної системи. Аналіз цих вимог є ключовим етапом у формуванні характеристик та обсягу майбутнього продукту. Цей перелік вимог охоплює як функціональні, так і нефункціональні аспекти, необхідні для успішної реалізації бізнес-цілей та відповіді на потреби користувачів. Аналіз, проведений на цьому етапі, закладає міцний фундамент для подальших кроків у проєктуванні та розробці програмної системи.

У підрозділі викладено функціональні можливості, які реалізовані в рамках дипломного проєкту. Нижче наведено розширений опис кожної з них.

Функція додавання PDF файлу з нотами - ця функція дозволяє користувачеві імпортувати PDF файли, які містять ноти музичних композицій.

Функція зберігання PDF файлу - після додавання PDF файлу користувач має можливість зберегти його в додатку для подальшого використання.

Функція вибору файлу для гри - користувач може обирати файли для гри зі списку збережених PDF файлів.

Функція Пошук/Сортування - у вкладці "Notes" є функціонал для пошуку та сортування нот за різними критеріями, такими як дата файлу, назва чи опис. Це дозволяє зручно організувати та знаходити необхідні композиції.

Функція прослуховування файлу - кнопка "Play" в інтерфейсі вкладки "Notes" надає можливість прослуховувати вибрану композицію перед грою, щоб користувач міг зрозуміти, як повинні звучати ноти.

Функція програвання нот та метроному - на вкладці гри та метроному реалізована функція програвання нот, яка дозволяє користувачеві грати під візуальний супровід за швидкістю метроному.

Функція кастомізації та видалення - на вкладці кастомізації файлів та видалення користувач може змінювати назву, опис та швидкість метроному збережених файлів, а також видаляти непотрібні.

Функція статистики активності - на головній сторінці інтерфейсу користувач може бачити статистику активності, що стимулює регулярність практики.

Функція налаштування додатку - користувач має можливість налаштувати додаток під свої потреби, вказавши особисті налаштування та вподобання в інтерфейсі налаштувань додатку.

Функція обробки вхідних даних у форматі PDF.

Функція перетворення файлу у форматі PDF в оброблювані музичні формати Musicxml та midi.

Функція обробки синхронізації з метрономом.

Функція надання файлу із озвученням музичного твору.

Функція надання файлу з візуалізацією та звуком у форматі відео.

Ці функції розроблені для поліпшення користувальницького досвіду та забезпечення різноманітних можливостей у освоєнні музики за допомогою додатку.

Висновки до розділу

У цьому розділі проведено детальний аналіз та опис ключових аспектів розробки системи. Описано Use Case Diagram (Діаграму Використання), яка демонструє взаємодію між актором і застосунком. Ця діаграма надає чітке розуміння функціональності та взаємодії з зовнішніми модулями.

Було визначено загальні вимоги до програмного забезпечення, включаючи як функціональні, так і нефункціональні аспекти, які будуть враховані при реалізації програмного продукту.

Зроблено аналіз економічних показників, який допоможе оцінити доцільність і порівнювати планові та фактичні показники, реагувати на відхилення та підвищувати економічну ефективність.

Завершальний етап розділу - постановка конкретних задач для подальшого розроблення програмного забезпечення. Всі вищезазначені етапи аналізу створили міцний фундамент для подальших етапів проєкту та визначили стратегію його розвитку.

За результатами розділу сформовано технічне завдання на розробку програмного забезпечення.

3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення.

Система використовує клієнт-серверну архітектуру, де сервер реалізований за допомогою веб-фреймворку Flask[17] і розташований локально на localhost. Для надання можливості доступу до сервера ззовні використовується Cloudflare[18], сервіс, який дозволяє тунелювати зовнішні запити до локального веб-сервера. Мобільний додаток виступає у ролі клієнта, взаємодіючи з сервером через HTTP-запити. Нижче описано основні компоненти:

Мобільний Додаток (Клієнт):

- відправляє запити до сервера;
- використовує API, наданий Flask-сервером, для відправки та отримання даних;
- розпаковує архів, наданий сервером та виводить на екран дані;

Flask Сервер (localhost)[17]:

- приймає та обробляє HTTP-запити від клієнта;
- основний компонент, що обробляє логіку сервера;
- обробляє PDF-файли, надіслані мобільним додатком;

Cloudflare[18]:

- тунелює запити на localhost до Flask-сервера;
- забезпечує URL-адресу, яку використовує мобільний додаток для доступу до сервера;

Описану вище архітектуру візуально показано на Рисунку 3.1.

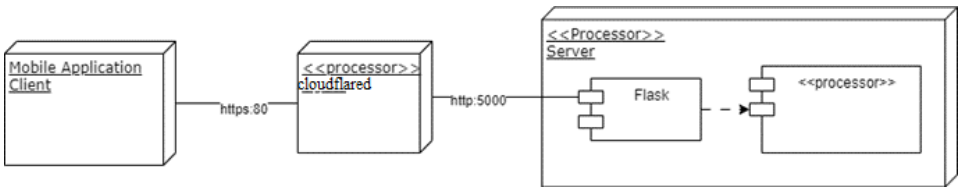


Рисунок 3.1 – Архітектура

3.2 Архітектурні рішення та обґрунтування вибору засобів розробки

У цьому розділі здійснюється огляд та аналіз допоміжних програмних засобів, включаючи бібліотеки та фреймворки, а також засоби розробки, що використовуються у даному проєкті.

Python[19] - це високорівнева, інтерпретована мова програмування, яка відома своєю читабельністю, простотою синтаксису та багатофункціональністю. Її універсальність робить її ідеальною для широкого спектра застосувань, від веб-розробки до наукових обчислень. Вагомою перевагою цієї мови програмування є також велика кількість модулів для роботи зі штучним інтелектом.

PyCharm від JetBrains[20] - це одне з найпопулярніших IDE для Python, що пропонує широкий спектр функцій для поліпшення продуктивності розробки, включаючи розширену підтримку коду, дебагінг, інтеграцію з системами контролю версій та численні плагіни.

Flask[17] - це мікрофреймворк для веб-додатків на Python, який відрізняється легкістю та гнучкістю. Він дозволяє швидко створювати надійні веб-додатки з мінімальним набором залежностей. У порівнянні з іншими веб-фреймворками, як-от Django[21], є більш гнучким, що робить його ідеальним для невеликих до середніх проєктів, яким не потрібна повна функціональність Django.

music21[22] - це потужна бібліотека Python для аналізу, маніпуляції та візуалізації музичних даних. Вона ідеально підходить для музичної теорії, аналізу музичних творів, генерації нотних зразків тощо.

partitura[23] - це ще одна бібліотека Python, спрямована на роботу з музичними нотами. Вона дозволяє ефективно читати, писати та обробляти музичні партитури, що робить її корисною для різних музичних застосувань.

music21 та partitura обидві забезпечують роботу з музичними даними, але music21 є більш всеосяжною, з більш широким спектром функцій для музичного аналізу. Partitura, з іншого боку, зосереджена на ефективній обробці

партитур, що може бути більш привабливим для специфічної роботи з нотними даними.

moviepy[24] - це бібліотека для обробки відео, яка дозволяє редагувати та об'єднувати відеокліпи, додавати аудіодоріжки, застосовувати ефекти та переходи. Вона стає в нагоді для створення музичних відеокліпів або візуалізацій. Є важливим доповненням для проєктів, які включають відеообробку, пропонуючи легкий доступ до відеомонтажу та аудіо-відео синхронізації.

androidx.core:core-ktx[25] – це допоміжна бібліотека для розширення функціоналу основних компонентів Android та надання зручного Kotlin API (KTX) для їх використання. Забезпечує підтримку сучасних функцій Android та полегшує написання коду.

androidx.appcompat:appcompat[26] - надає сумісність з новими функціями Android для старших версій платформи. Ця бібліотека дозволяє використовувати сучасний дизайн та можливості навігації навіть на більш ранніх версіях Android.

com.google.android.material:material[27] - реалізує Material Design - стиль дизайну, рекомендований Google. Забезпечує доступ до компонентів і інструментів, які надають додатку сучасний та консистентний вигляд.

androidx.constraintlayout:constraintlayout[28] - використовується для роботи з ConstraintLayout, який дозволяє ефективно розміщувати елементи інтерфейсу за допомогою обмежень, що полегшує створення сложних макетів.

Junit[29] – це бібліотека для написання тестів у Java та Kotlin. Вона дозволяє розробникам автоматизувати тестування, забезпечуючи надійність та стабільність коду.

androidx.test.espresso:espresso-core[30] – це фреймворк для написання автоматизованих тестів інтерфейсу користувача (UI) на платформі Android. Забезпечує можливість визначення та виконання тестів взаємодії з елементами UI.

androidx.media3:media3-exoplayer[31] – це бібліотека для відтворення мультимедійних контентів на Android-пристроях. ExoPlayer дозволяє налаштовувати відтворення аудіо та відео з високою гнучкістю та підтримує різноманітні формати.

androidx.media3:media3-exoplayer-dash[32] – це розширення для підтримки відтворення DASH-стрімів (Dynamic Adaptive Streaming over HTTP) в ExoPlayer, що дозволяє ефективно використовувати потік залежно від доступності мережі та пристрою.

androidx.media3:media3-ui[33] - надає компоненти інтерфейсу користувача для вбудованого відтворення медіа з ExoPlayer, полегшуючи створення користувацького інтерфейсу для медіапрогравача.

com.squareup.retrofit2:retrofit[34] - використовується для взаємодії з веб-службами та отримання даних через HTTP-запити. Забезпечує зручний та ефективний механізм виклику API та обробки відповідей.

com.squareup.retrofit2:converter-gson[35] – це розширення Retrofit для перетворення JSON-даних в об'єкти Kotlin за допомогою бібліотеки Gson.

pl.droidsonroids.gif:android-gif-drawable[36] – це бібліотека для роботи з GIF-зображеннями на платформі Android. Забезпечує можливість зручно відтворювати та управляти анімацією GIF.

Android Studio[37] – це IDE, яке користується широкою підтримкою спільноти розробників та регулярними оновленнями, що забезпечує актуальність та можливість використання новітніх технологій для розробки Android-додатків.

Є офіційним інтегрованим середовищем розробки для Android, розробленим напряму Google. Його потужність та інтеграція з Android SDK (Software Development Kit) дозволяють зручно створювати, тестувати мобільні додатки для платформи Android.

Надає доступ до інструментів, специфічних для розробки Android, таких як візуальний дизайнер інтерфейсу користувача, емулятори Android-пристроїв

для тестування та інші інструменти, які полегшують розробку на цій платформі.

Kotlin[38] - це сучасна, багатоцільова мова програмування, призначена для створення додатків на платформі Java, але вона також підтримує безліч інших цілей, включно з Android-розробкою, веб-розробкою та багато іншого.

Вирізняється високою виразністю, безпекою типів, і зручністю у використанні, що робить його привабливим вибором для широкого кола завдань і проєктів. надає більше можливостей для безпечного програмування, включаючи нульову безпеку (nullable safety) та виразніші конструкції мови. Це допомагає випереджати багато типових помилок на етапі компіляції.

3.3 Конструювання програмного забезпечення

У цьому підрозділі надається детальний огляд ключових аспектів програмного забезпечення. Описується база даних та використання утиліт, бібліотек та іншого стороннього програмного забезпечення, яке використовується у процесі розробки.

3.3.1 Опис структури бази даних

В якості системи управління базами даних використовується PostgreSQL. База даних серверу призначена для зберігання файлів, нотних записів і статистики. Опис таблиць бази даних наведено у таблицях 3.1-3.3. Модель бази даних наведена на рисунку 3.2.

Таблиця 3.1 – Опис таблиці Notes

Назва поля	Тип даних	Опис
id	int	Ідентифікація
name	varchar	Кастомна назва файлу з нотами
description	varchar	Опис файлу з нотами

Продовження Таблиці 3.1

Назва поля	Тип даних	Опис
file_id	int	FK ідентифікатор файлу з нотами
update_date	datetime	Дата оновлення запису

Таблиця 3.2 – Опис таблиці Files

Назва поля	Тип даних	Опис
id	int	Ідентифікація
name	varchar	Назва файлу в системі
path	varchar	Шлях до файлу в системі
metronome_bpm	int	Швидкість метроному
mp3_path	varchar	Шлях до mp3 файлу в системі
mp4_path	varchar	Шлях до mp4 файлу в системі

Таблиця 3.3 – Опис таблиці Statistic

Назва поля	Тип даних	Опис
id	int	Ідентифікація
strike_days	int	Кількість активних днів підряд
update_date	datetime	Дата оновлення запису

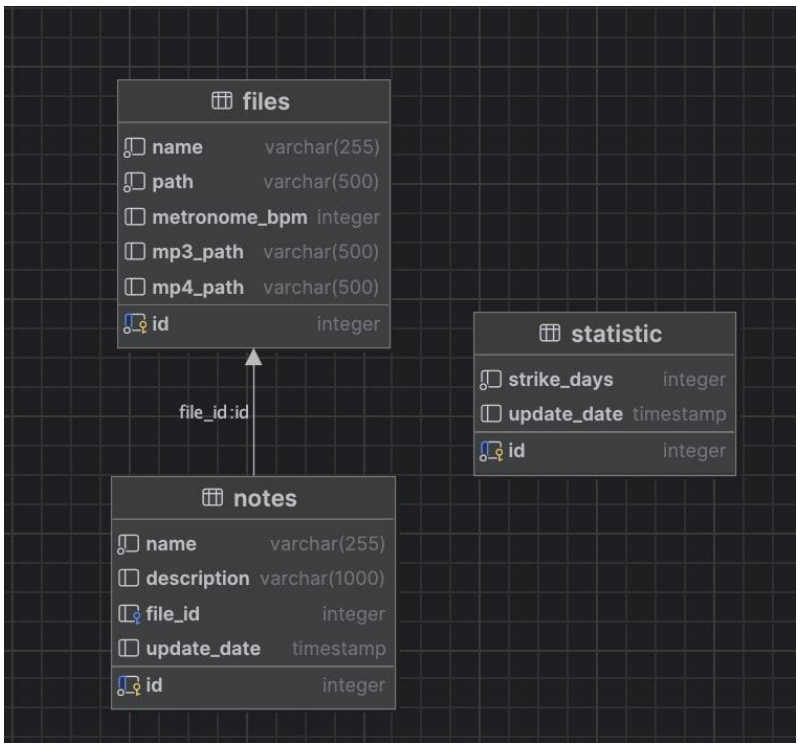


Рисунок 3.2 –Діаграма сутностей Notes, Files, Statistic

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 3.4.

Таблиця 3.4 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	Android Studio	Середовище розробки програмного забезпечення.
2	Draw.io	Застосунок для створення схеми дизайну
3	ExoPlayer (Media3)	Бібліотека для відтворення мультимедійних контентів на Android-пристроях.
4	AndroidX (core-ktx)	Допоміжна бібліотека для розширення функціоналу основних компонентів Android та надання зручного Kotlin API (KTX) для їх використання.

Продовження Таблиці 3.4

№ п/п	Назва утиліти	Опис застосування
5	Espresso	Фреймворк для написання автоматизованих тестів інтерфейсу користувача
6	Retrofit	Використовується для взаємодії з веб-службами та отримання даних через HTTP-запити. Забезпечує зручний та ефективний механізм виклику API та обробки відповідей.
7	Android GIF Drawable	Бібліотека для роботи з GIF-зображеннями на платформі Android.
8	PyCharm	Середовище розробки програмного забезпечення.
9	Audiveris	Високоякісний оптичний розпізнавач музичних символів (OMR - Optical Music Recognition), який перетворює скановані аркуші музики у формат MusicXML.
10	MuseScore	Одна з найпопулярніших програм для нотації, яка дозволяє працювати з MusicXML файлами. Вона використовується у даній системі для подальшої обробки MusicXML файлів, отриманих з Audiveris.
11	SoundFont	Технологія, яка використовується для відтворення MIDI-файлів. Вона містить записи реальних звуків інструментів, які можуть бути використані для генерації аудіофайлів з MIDI.

Тексти програмного коду наведені в окремому документі «Текст програми».

3.4 Аналіз безпеки даних

Оскільки сервер використовує файлову систему для збереження файлів, нам потрібно чітко розуміти. До яких файлів програма має доступ та які файли відправляє клієнтові. Для цього контролю передбачено створення унікального коду на основі таймстемпу, який оновлюється та привласнюється кожному 389 новому PDF-файлу, що надходить на обробку. Під усі файли для конкретного кейсу створюється директорія, файли зберігаються під даним унікальним номером.

Висновки до розділу

У цьому розділі було докладно розглянуто ключові аспекти конструювання програмного забезпечення. Описано структури даних, програмні структури та базу даних з концептуальною, логічною та фізичною моделлю, що надає чіткий огляд внутрішньої організації системи.

Детально представлені утиліти, бібліотеки та інше стороннє програмне забезпечення, яке використовується у розробці, щоб забезпечити повноту та функціональність системи. Описані алгоритми та програмні структури виявилися важливими для досягнення високої ефективності та точності обробки музичної інформації. А також зроблено аналіз безпеки даних проєкту.

Цей розділ є ключовим у розумінні та документуванні технічних деталей системи, що сприяє якісному конструюванню програмного продукту та його подальшій успішній імплементації.

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз якості ПЗ

Аналіз якості програмного забезпечення забезпечить оцінку, чи відповідає впроваджене рішення вимогам користувачів та очікуваній функціональності. Тестування допомагає виявити можливі помилки або прогалини в системі, щоб їх можна було ефективно виправити, підвищивши загальну стабільність і надійність додатку.

Метою тестування є:

- перевірити, що додаток працює коректно і відповідає визначеним функціональним вимогам;
- перевірити надійність зберігання, синхронізації та передачі даних.
- виявити помилки та недоліки для забезпечення стабільної роботи системи.

4.2 Опис процесів тестування

Тестування виконується згідно документу «Програма та методика тестування».

Важливою складовою розробки програмного забезпечення є його тестування для визначення правильності та ефективності функціоналу. У даному розділі наведено результати мануального тестування, яке було проведено для визначення якості та відповідності розробленого програмного забезпечення встановленим вимогам. Опис відповідних тестів наведено у таблицях 4.1 – 4.11.

Таблиця 4.1 – Тест 1.1 Завантаження головної сторінки додатку

Початковий стан системи	Користувач запускає додаток.
Вхідні дані	Відсутні.
Опис проведення тесту	Запускається apk файл і спостерігається головна сторінка додатку.
Очікуваний результат	Головна сторінка додатку відображена успішно.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.2 – Тест 1.2 Завантаження файлу і виведення результату

Початковий стан системи	Форма завантаження відображена.
Вхідні дані	Файл для завантаження і швидкість метроному.
Опис проведення тесту	Завантажується файл і спостерігається процес обробки.
Очікуваний результат	Файл успішно завантажений. Форма з нотами відображена і можливо продивитись/прослухати файл.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.3 – Тест 1.3 Редагування файлу

Початковий стан системи	Форма редагування файлу відображена.
Вхідні дані	Файл з бібліотеки «Notes».
Опис проведення тесту	Редагується файл і перевіряється результат.
Очікуваний результат	Файл успішно відредагований у бібліотеці.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.4 – Тест 1.4 Зміна теми додатку

Початковий стан системи	Додаток запущено. Користувач на формі «Settings».
Вхідні дані	Відсутні.
Опис проведення тесту	Переключається тема і спостерігається зміна.
Очікуваний результат	Тема змінена успішно.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.5 – Тест 1.5 Пошук нот у бібліотеці

Початковий стан системи	Додаток запущено. Користувач на формі «Notes».
Вхідні дані	Пошуковий запит.
Опис проведення тесту	Вводиться пошуковий запит і відображаються ноти відповідно до нього.
Очікуваний результат	Ноти відобразились коректно і відповідно до введеного запиту.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.6 – Тест 1.6 Сортування нот у бібліотеці

Початковий стан системи	Додаток запущено. Користувач на формі «Notes».
Вхідні дані	Параметр сортування.
Опис проведення тесту	Обирається параметр сортування і ноти сортуються відповідно до нього.
Очікуваний результат	Ноти відсортувались коректно і відповідно до обраного параметру.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.7 – Тест 1.7 Відправка POST запиту з файлом та швидкістю метроному.

Початковий стан системи	Сервер запущено.
Вхідні дані	PDF-файл та швидкість метроному.
Опис проведення тесту	У відповідні поля вводяться: поля назви файлу і параметрів, шлях до файлу, швидкість метроному.
Очікуваний результат	Запит оброблено успішно.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.8 – Тест 1.8 Відправка POST запиту без файлу, але з швидкістю метроному.

Початковий стан системи	Сервер запущено.
Вхідні дані	Швидкість метроному.
Опис проведення тесту	У відповідні поля вводяться: поля назви параметрів, швидкість метроному.
Очікуваний результат	Помилка обробки.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.9 – Тест 1.9 Відправка POST запиту з файлом, але без швидкості метроному.

Початковий стан системи	Сервер запущено.
Вхідні дані	PDF-файл.
Опис проведення тесту	У відповідні поля вводяться: поля назви файлу і шлях до файлу.
Очікуваний результат	Помилка обробки.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.10 – Тест 1.10 Відправка PUT запиту з швидкістю метроному.

Початковий стан системи	Сервер запущено.
Вхідні дані	Швидкість метроному.
Опис проведення тесту	У відповідні поля вводяться: поля назви параметрів, швидкість метроному, у шлях вводиться унікальний номер.
Очікуваний результат	Запит оброблено успішно.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.11 – Тест 1.11 Відправка GET запиту на оброблений файл.

Початковий стан системи	Сервер запущено. Файли оброблені.
Вхідні дані	Унікальний номер.
Опис проведення тесту	У шлях вводиться унікальний номер, отриманий із POST запиту.
Очікуваний результат	Успішне отримання файлів.
Фактичний результат	Збігається з очікуваним.

4.3 Опис контрольного прикладу

При запуску програмного застосунку користувачу буде відображено сторінку «Home» з кнопкою з можливістю завантажити файл (Рисунок 4.1). Користувач має змогу завантажити файл або перейти на інші сторінки.

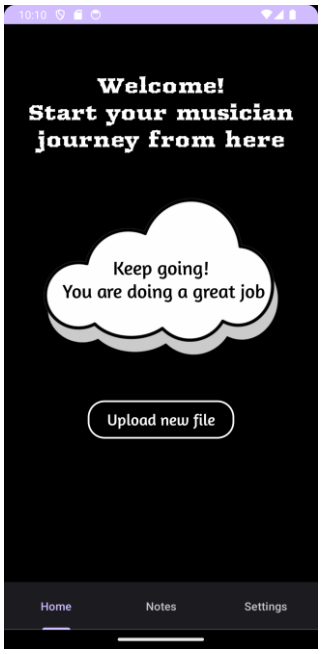


Рисунок 4.1 – Початкова сторінка додатку

На сторінці «Notes» програмного застосунку користувачу буде відображено бібліотеку з можливостями програти файл, відредагувати, сортувати і шукати по імені або опису (Рисунок 4.2). Також є можливість додати новий файл.

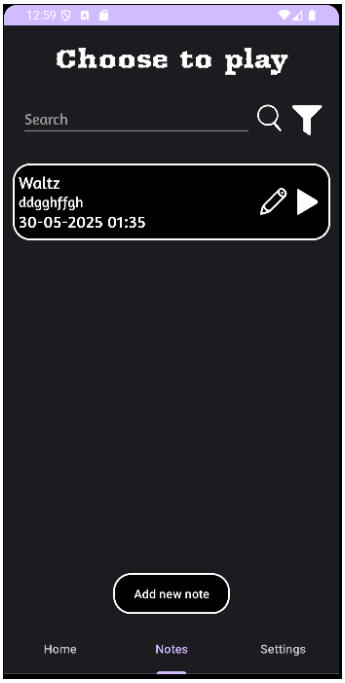


Рисунок 4.2 – Сторінка Notes

На сторінці «Settings» програмного застосунку користувачу буде відображено можливі налаштування додатку (Рисунок 4.3).

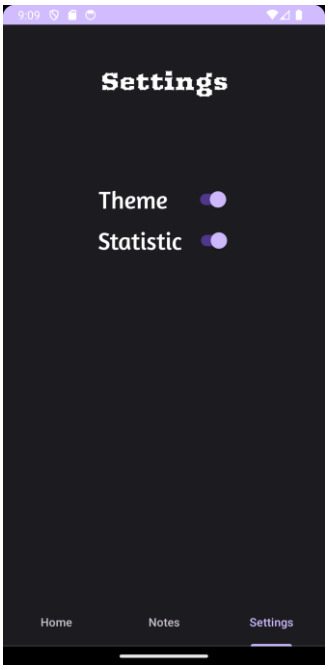


Рисунок 4.3 – Сторінка Settings

Після натискання кнопки додавання файлу, користувач може обрати файл з телефону (Рисунок 4.4). Кастомізувати його і обрати чи хоче він зберігати його в бібліотеку (Рисунок 4.5).

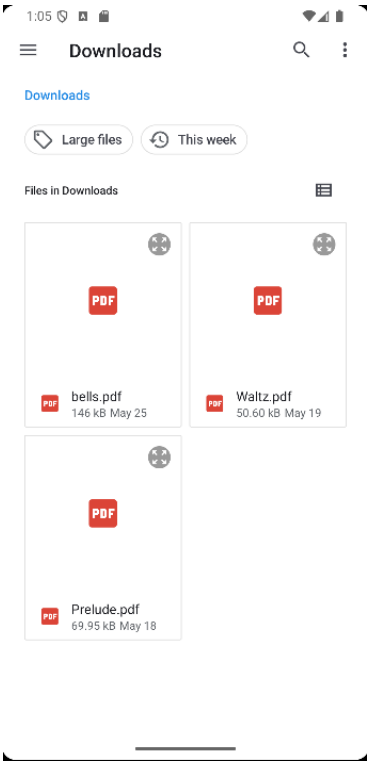


Рисунок 4.4 – Вибір файлу

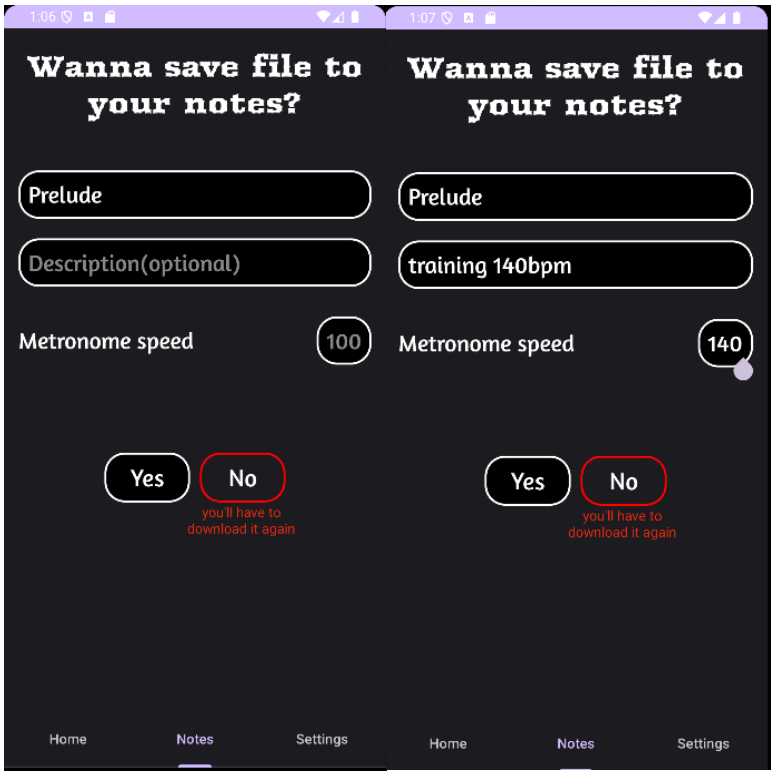


Рисунок 4.5 – Кастомізація опису і швидкості метроному

Після натискання кнопки «Yes» або «No» починається обробка файлу з описом і гіфкою на екрані (Рисунок 4.6-4.7).

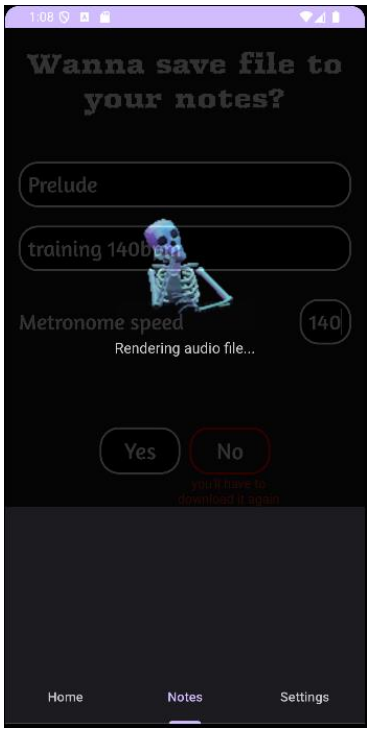


Рисунок 4.6 – Очікування обробки 1

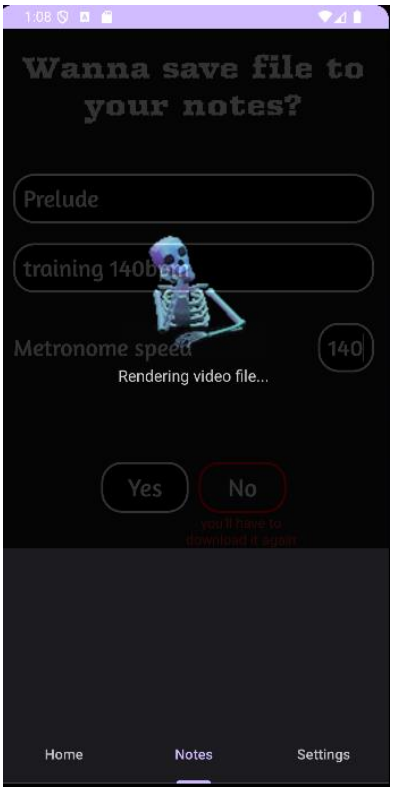


Рисунок 4.7 – Очікування обробки 2

Після обробки файлу користувач може бачити відеоряд з нотами та має можливість прослухати композицію з заданими налаштуваннями (Рисунок 4.8-4.9).

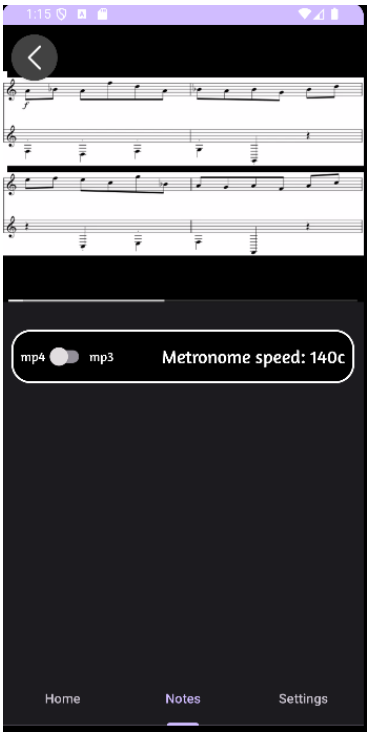


Рисунок 4.8 – Відеоряд з обробленими нотами

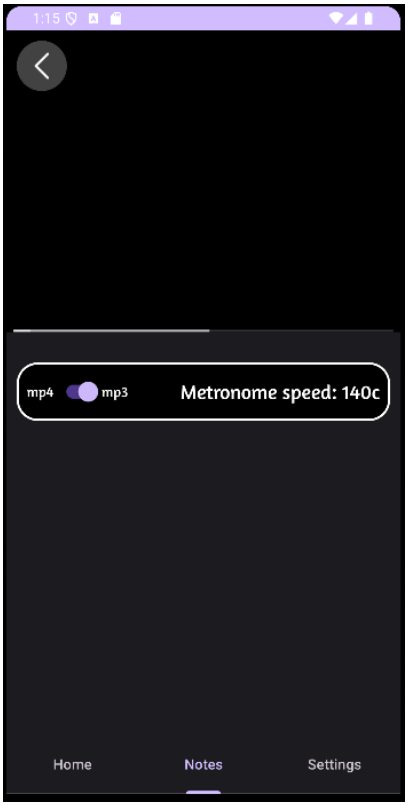


Рисунок 4.9 – Аудіо з обробленими нотами

Після того, як користувач виходить з екрана з програванням нот, то ноти відображаються у бібліотеці, за умови їх збереження (натискання кнопки «Yes») (Рисунок 4.10). Користувач може їх редагувати (Рисунок 4.11-4.12), сортувати (Рисунок 4.13-4.14) і виконувати пошук за літерою/ами (Рисунок 4.15).

Рисунок 4.10 – Список збережених нот

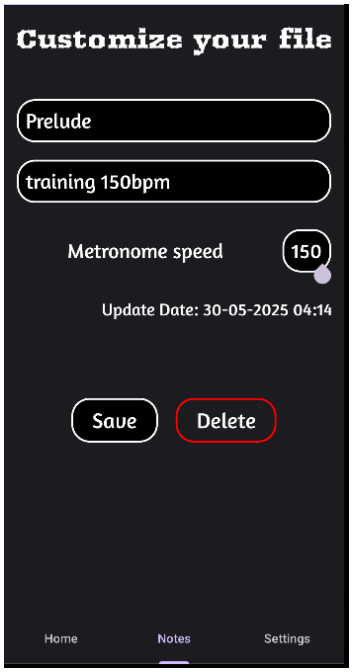


Рисунок 4.11 – Редагування збереженого нотного запису

Рисунок 4.12 – Відредагований нотний запис у бібліотеці

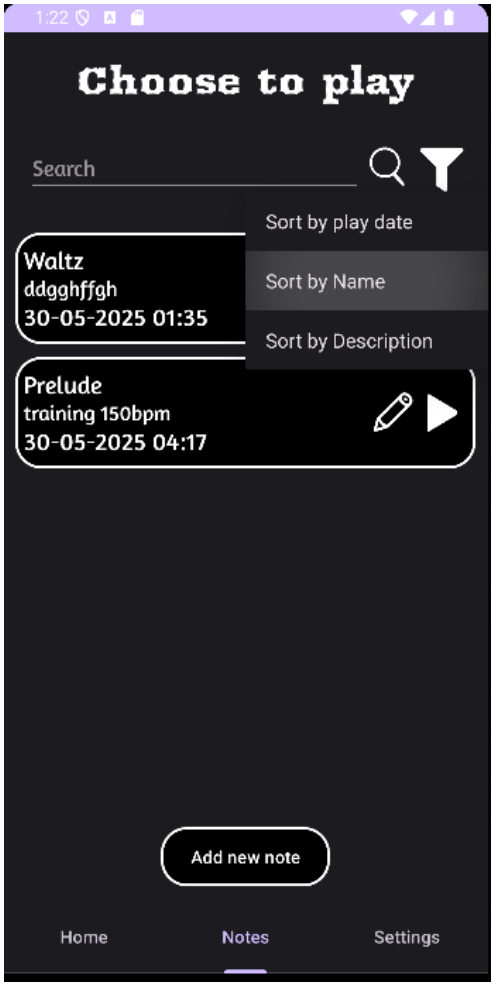


Рисунок 4.13 – Спадний список сортування

Рисунок 4.14 – Список, відсортований за іменем

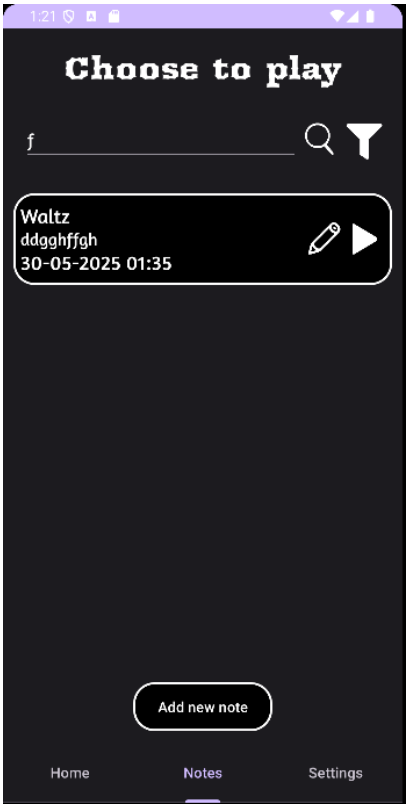


Рисунок 4.15 – Пошук по літері «f»

На сторінці «Settings» користувач може змінити тему застосунку (Рисунок 4.16-4.17).

Рисунок 4.16 – Сторінка «Settings» з включеною темною темою

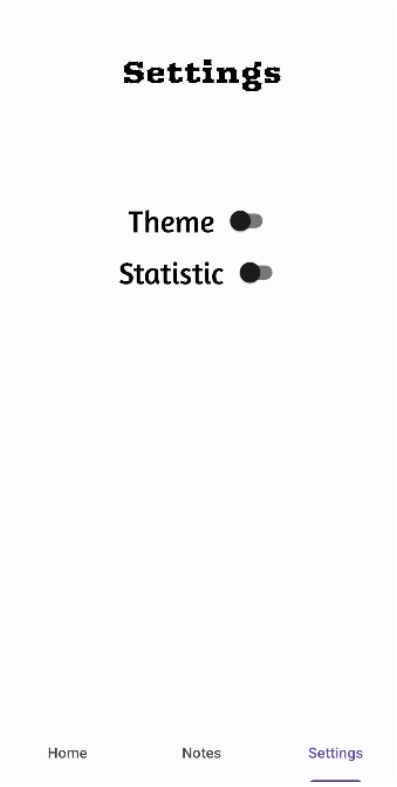


Рисунок 4.17 – Сторінка «Settings» з включеною світлою темою

На сторінці «Settings» користувач може включити відображення статистики (Рисунок 4.18-4.19).

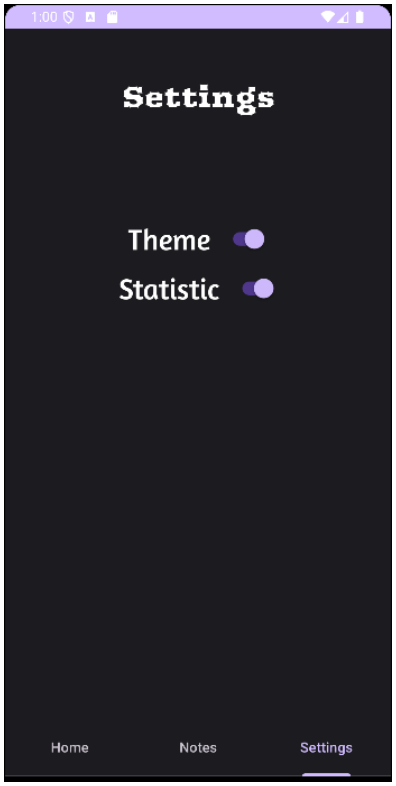


Рисунок 4.18 – Сторінка «Settings» з включеною статистикою

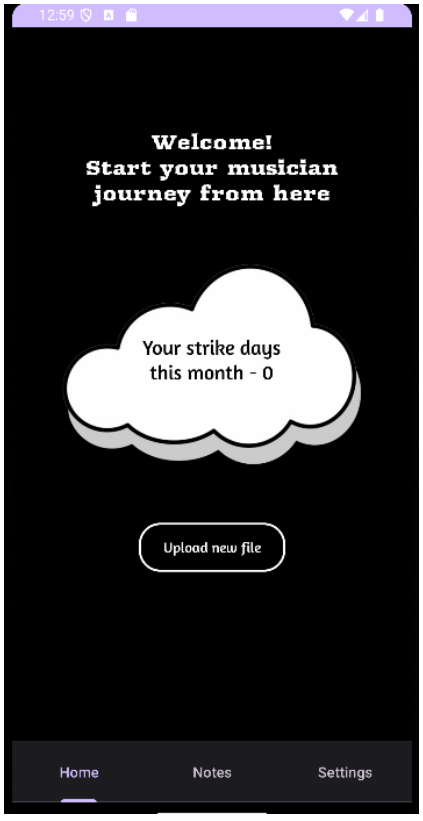


Рисунок 4.19 – Сторінка «Home» з включеною статистикою

Висновки до розділу

У цьому розділі представлено аналіз якості програмного забезпечення, спираючись на функціональну складову та стабільність у використанні. Основна мета тестів полягала в тому, щоб переконатися, що основні функції програми працюють правильно, перевірити механізми зберігання, синхронізації та передачі даних, а також виявити можливі помилки та недоліки.

Тестові кейси були розроблені для кожного основного етапу програми, охоплюючи весь життєвий цикл програми - від головної сторінки до відправки запитів на сервер і обробки результатів.

Всі тести пройшли успішно і фактичні результати відповідали очікуваним, що свідчить про коректність реалізованого функціоналу.

Крім того, в розділі детально описано контрольний приклад, що демонструє типовий сценарій використання програмного забезпечення. Цей приклад ілюструє основні функції системи, доступні користувачеві.

5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розгортання програмного забезпечення

У цьому підрозділі описується процес покрокового розгортання програмного забезпечення на локальному хості з використанням Cloudflare для забезпечення зовнішнього доступу. Розглянемо кожен етап детально.

Крок 1: Підготовка середовища.

Встановлення Python та необхідних бібліотек за допомогою pip. Встановлення утиліт Audiveris, SountFont та Musescore.

Крок 2: Локальний запуск додатку.

Запуск Flask командою flask run. Це зробить додаток доступним за адресою http://127.0.0.1:5000 (Рисунок 5.1).

Крок 3: Використання Cloudflare для зовнішнього доступу.

Завантажити Cloudflare, отримати адресу та запустити Cloudflare (Рисунок 5.2).

Крок 4: Збірка APK-файлу і запуск додатку.

Збірка APK-файлу з скомпільованого коду та необхідних ресурсів (Рисунок 5.3).

Запуск додатку на емуляторі чи реальному пристрої для перевірки його функціональності та виявлення можливих помилок (Рисунок 5.4).

Крок 5: Тестування доступу.

Надсилаємо на отриману адресу запит (Рисунок 5.5).

Додаток розгорнуто на локальному пристрої, тому що потрібна кількість утиліт не дозволяє безкоштовно захостити весь сервер на відкритих ресурсах. Cloudflare у свою чергу дає можливість відкритого доступу з інших пристроїв до локально розміщеного серверу.

Цей покроковий процес розгортання мобільного додатку доповнений ілюстраціями для кращого розуміння та ефективного використання кожного кроку у впровадженні додатку.

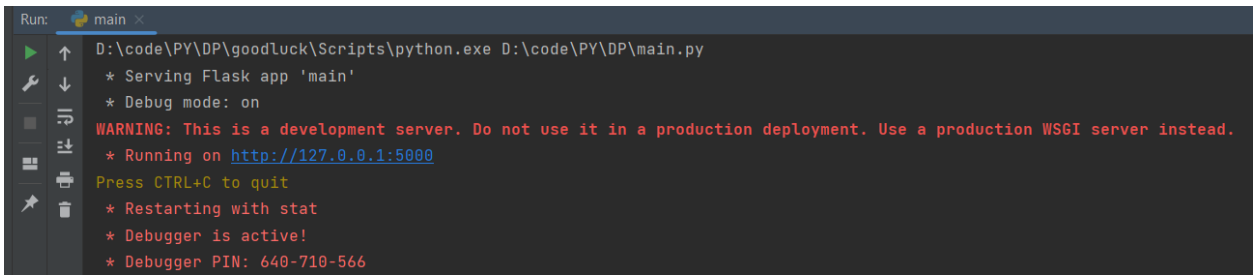


Рисунок 5.1 – Запуск Flask

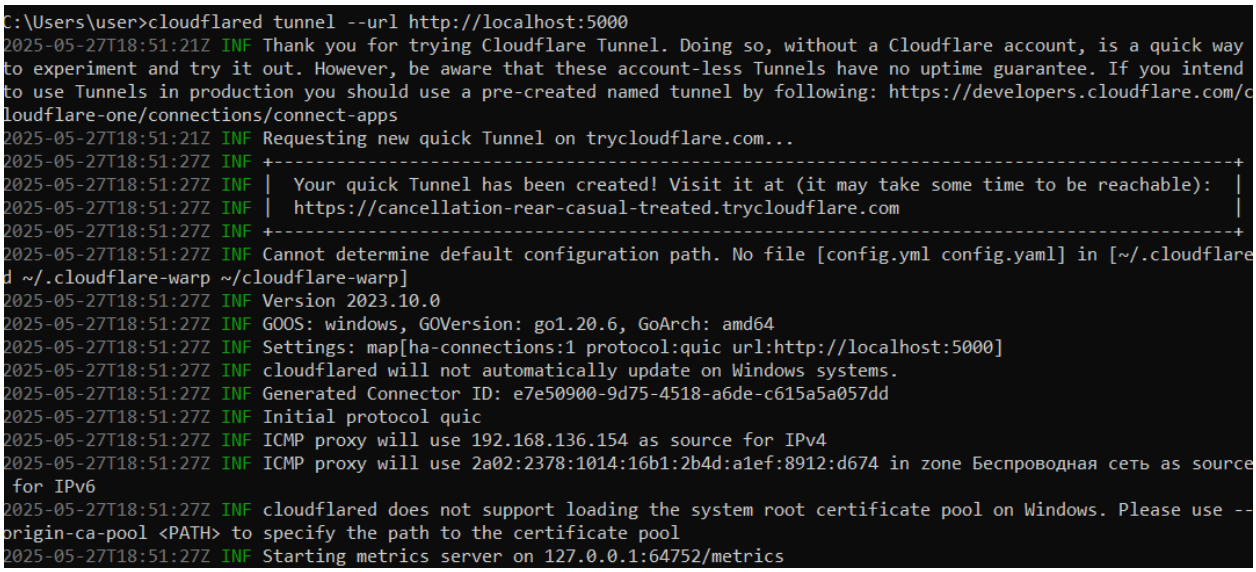


Рисунок 5.2 – Отримання адреси та запуск Cloudflare

Рисунок 5.3 – Apk, завантажене на пристрій

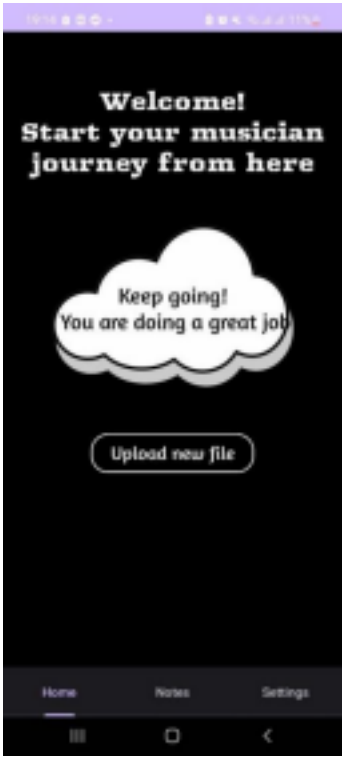


Рисунок 5.4 – Запущений додаток. Головна сторінка

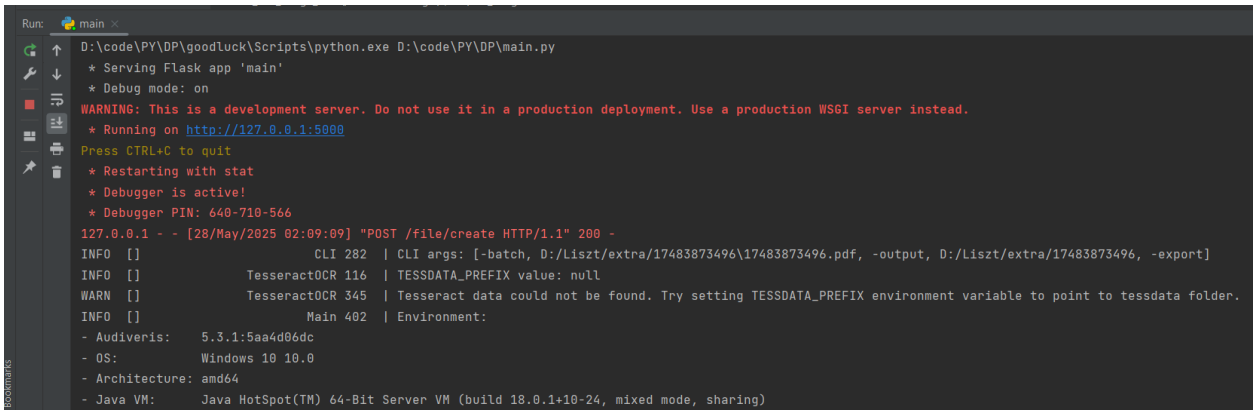


Рисунок 5.5 – POST запит на сервер з додатку

5.2 Супровід програмного забезпечення

Інструкція користувача наведена в окремому документі «ДП_ч5_КК_2025».

Користувачі повинні мати можливість отримати нову версію застосунку з кожною версією. З цією метою буде надано централізований спосіб оновлення мобільного додатку через Git-репозиторій (Рисунок 5.6).

Створено спеціальний репозиторій на платформі GitHub для зберігання стабільних версій APK. Щоразу, коли мобільний додаток оновлюється, нова версія APK-файлу додається і публікується у відповідному розділі репозиторію з вказаною версією. Таким чином, користувачі можуть швидко отримати останню версію програми без необхідності використовувати сторонні магазини додатків. Крім того, репозиторій Git містить журнал оновлень, який дозволяє перевіряти наявність оновлень і виправлень помилок.

Оновлення здійснюються шляхом завантаження нового APK-файлу зі сховища. Користувачі можуть перевстановити програму вручну, якщо це необхідно.

Перевагами такого підходу є гнучкість, прозорість і простота в обслуговуванні. Якщо виникає помилка або потрібна нова функціональність, можна швидко змінити код, створити оновлений APK-файл і опублікувати його в репозиторії. Це скорочує час реагування на запити користувачів і забезпечує безперервність роботи системи.

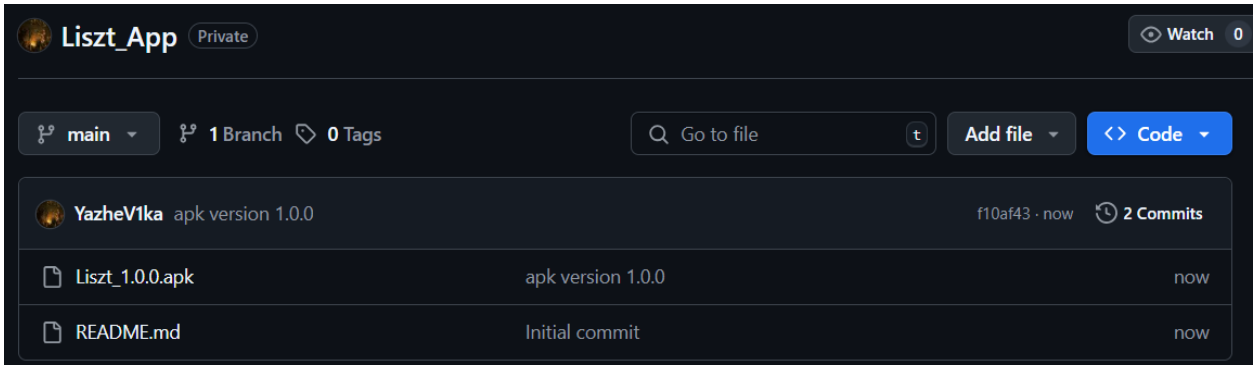


Рисунок 5.6 – Репозиторій з apk файлом

Висновки до розділу

У цьому розділі детально описано процес впровадження та супроводу розробленого програмного забезпечення. Враховуючи специфіку середовища розробки та обмеження хостингу ресурсно-вимогливих утиліт, представлено покрокове керівництво для локального розгортання додатку на стороні сервера на основі фреймворку Flask та самого мобільного додатку.

Описано використання утиліти Cloudflare для надання зовнішнього доступу до локального сервера, що дозволяє взаємодіяти з сервером з будь-якого пристрою без необхідності розгортання сервера в хмарі. Таке рішення дозволяє дотримуватися обмежень дорогого хостингу та спрощує процес тестування мобільного додатку в умовах, наближених до реального використання.

Клієнт-серверна взаємодія була перевірена за допомогою POST-запитів, що підтверджує належне функціонування обох частин програмного забезпечення.

Підтримка оновлень додатку буде проводитися з використанням відповідного Git-репозиторію, щоб користувачі мали змогу встановити оновлені версії APK-файлів. Документація для користувача описана у окремому файлі, що містить покрокові інструкції щодо встановлення та користування додатком.

ВИСНОВКИ

У результаті виконання дипломного проєкту розроблено серверну частину і мобільний додаток для розпізнавання нот і музичної інтеракції. Застосунок обробляє PDF файли з заданою швидкістю метроному у формати mp3 та mp4, що надає користувачу можливість бачити і прослуховувати завантажену нотну грамоту.

Оцінка отриманих результатів:

Отримані результати доводять конкурентоспроможність додатку, відповідність зазначеним вимогам і рівень відповідності технічним знанням у галузі розробки програмного забезпечення. Система успішно оперує даними і вміло візуалізує їх.

Аспекти значущості цього додатку є забезпечення доступності для всіх зацікавлених осіб, незалежно від рівня музичної підготовки, а також використання додатка може стати частиною музичної освіти в навчальних закладах та додаткових курсах. Він надає засіб для покращення процесу навчання музики та можливості для творчого вираження.

Впровадження сучасних технологій розпізнавання нот і візуалізації музичних елементів сприяє технологічному прогресу в області музичної розробки. Створює позитивний досвід вивчення музики, що може підтримати та зберегти зацікавлення користувачів у світі музики протягом тривалого часу.

На даному етапі існує багато напрямків розвитку даного додатку. В планах є розширення в сторону більшої кількості форматів файлів, додаткових допоміжних функцій (калібрування інструменту, тощо), додавання спільноти і відповідно можливості ділитися напрацюваннями. Стосовно самого програмного забезпечення – перш за все це прискорення операцій, покращення візуалізації і шляху взаємодії з сервером. Також важливим аспектом є подальше вдосконалення алгоритмів обробки музики для підвищення точності та ефективності.

Усі визначені завдання були успішно виконані. Після впровадження додаток пройшов тестування на мобільному пристрої та у програмах, які використовувались для взаємодії з сервером.

Виконана робота при розробці проєкту надає фундамент для подальших досліджень у галузі обробки музики.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1) Штучний інтелект та музиканти: помічник або конкурент? [Електронний ресурс] – Режим доступу до ресурсу: – https://www.neformat.com.ua/ articles /shtuchniy-intelekt.html

2) How AI is Changing the Music Industry [Електронний ресурс] – Режим доступу до ресурсу: – https://www.rollingstone.com/music/music-features/how-ai-is-changing-the-music-industry-204120/

3) AI in Music: How Artificial Intelligence is Changing the Way We Create and Listen to Music [Електронний ресурс] – Режим доступу до ресурсу: – https://syncedreview.com/2021/07/22/ai-in-music-how-artificial-intelligence-is-changing-the-way-we-create-and-listen-to-music/

4) Yousician: Unleash your inner musician with Yousician [Електронний ресурс] – Режим доступу до ресурсу: – https://yousician.com/

5) Musicnotes [Електронний ресурс] – Режим доступу до ресурсу: – https://www.musicnotes.com/?srsltid=AfmBOoqwLta985E7a-8B3-Eicb54dynVszJUfGsHzLAq6UFpo-JYSPz0

6) Simply Piano [Електронний ресурс] – Режим доступу до ресурсу: – https://hellosimply.com/simply-piano

7) MIDI (Musical Instrument Digital Interface) [Електронний ресурс] – Режим доступу до ресурсу: – https://www.techtarget.com/whatis/definition/MIDI-Musical-Instrument-Digital-Interface

8) MusicXML [Електронний ресурс] – Режим доступу до ресурсу: – https://www.musicxml.com/

9) Springer Nature: Optical music recognition [Електронний ресурс] – Режим доступу до ресурсу: – https://link.springer.com/chapter/10.1007/978-3-031-41498-5_7

10) Amazon: Optical Character Recognition [Електронний ресурс] – Режим доступу до ресурсу: – https://aws.amazon.com/what-is/ocr/

11) NTI Audio: Fast Fourier Transformation FFT - Basics [Електронний ресурс] – Режим доступу до ресурсу: – https://www.nti-audio.com/en/support/know-how/fast-fourier-transform-fft

12) ScienceDirect: Wavelet Transforms [Електронний ресурс] – Режим доступу до ресурсу: – https://www.sciencedirect.com/topics/computer-science/wavelet-transforms

13) Github: Audiveris [Електронний ресурс] – Режим доступу до ресурсу: – https://github.com/Audiveris/audiveris

14) MuseScore Studio [Електронний ресурс] – Режим доступу до ресурсу: – https://musescore.org/uk

15) Vipzone: SoundFont [Електронний ресурс] – Режим доступу до ресурсу: – https://www.vipzone-samples.com/en/what-is-a-soundfont/

16) Geekforgeeks: COCOMO Model - Software Engineering [Електронний ресурс] – Режим доступу до ресурсу: – https://www.geeksforgeeks.org/software-engineering-cocomo-model/

17) Palletsprojects: Flask [Електронний ресурс] – Режим доступу до ресурсу: – https://flask.palletsprojects.com/en/stable/

18) Ngrok [Електронний ресурс] – Режим доступу до ресурсу: – https://ngrok.com/

19) Python.org [Електронний ресурс] – Режим доступу до ресурсу: – https://www.python.org/

20) JetBrains: PyCharm [Електронний ресурс] – Режим доступу до ресурсу: – https://www.jetbrains.com/pycharm/

21) Django [Електронний ресурс] – Режим доступу до ресурсу: – https://www.djangoproject.com/

22) Music21 [Електронний ресурс] – Режим доступу до ресурсу: – https://www.music21.org/music21docs/

23) Partitura documentation [Електронний ресурс] – Режим доступу до ресурсу: – https://partitura.readthedocs.io/en/latest/introduction.html

24) Pypi: moviepy [Електронний ресурс] – Режим доступу до ресурсу: – https://pypi.org/project/moviepy/

25) Android Developers: Core [Електронний ресурс] – Режим доступу до ресурсу: – https://developer.android.com/jetpack/androidx/releases/core

26) Android Developers: Appcompat [Електронний ресурс] – Режим доступу до ресурсу: – https://developer.android.com/jetpack/androidx/releases/appcompat

27) Maven repository: Material Components For Android [Електронний ресурс] – Режим доступу до ресурсу: – https://mvnrepository.com/artifact/com.google.android.material/material

28) Android Developers: Constraintlayout [Електронний ресурс] – Режим доступу до ресурсу: – https://developer.android.com/jetpack/androidx/releases/constraintlayout

29) JUnit [Електронний ресурс] – Режим доступу до ресурсу: – https://junit.org/junit5/

30) Maven repository: AndroidX Test Library [Електронний ресурс] – Режим доступу до ресурсу: – https://mvnrepository.com/artifact/androidx.test.espresso/espresso-core

31) Android Developers: Exoplayer [Електронний ресурс] – Режим доступу до ресурсу: – https://developer.android.com/media/media3/exoplayer

32) Android Developers: Dash [Електронний ресурс] – Режим доступу до ресурсу: – https://developer.android.com/media/media3/exoplayer/dash

33) Android Developers: Media3 [Електронний ресурс] – Режим доступу до ресурсу: – https://developer.android.com/jetpack/androidx/releases/media3

34) Maven repository: Retrofit [Електронний ресурс] – Режим доступу до ресурсу: – https://mvnrepository.com/artifact/com.squareup.retrofit2/retrofit

35) Maven repository: Converter GSON [Електронний ресурс] – Режим доступу до ресурсу: – https://mvnrepository.com/artifact/com.squareup.retrofit2/converter-gson

36) Maven repository: Android GIF Drawable [Електронний ресурс] – Режим доступу до ресурсу: – https://mvnrepository.com/artifact/pl.droidsonroids.gif/android-gif-drawable

37) Android Developers: Studio [Електронний ресурс] – Режим доступу до ресурсу: – https://developer.android.com/studio

38) Kotlinlang [Електронний ресурс] – Режим доступу до ресурсу: – https://kotlinlang.org/

ДОДАТОК А ЗВІТ ПОДІБНОСТІ

StrikePlagiarism.com

Дата звіту 6/2/2025
Дата редагування 6/2/2025

Документ прийнятий

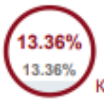
Звіт подібності

метадані

Назва організації
National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute
Заголовок
ІП-11_Фукс_ПЗ
Автор Науковий керівник / Експерт
ІП-11_ФуксПоперешняк С.В.
підрозділ
ФІОТ, К-а інформатики та програмної інженерії

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1

10	8092	61963
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	0
Інтервали	A→	0
Мікропробіли	(	18
Білі знаки	ß	0
Парафрази (SmartMarks)	a	60

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	ІП-15_Лазьов_ПЗ [illegible] 5/31/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)	88 1.09 %
2	https://openarchive.nure.ua/bitstreams/4c03742c-7ee8-4b7b-9e52-ed09030b5062/download	46 0.57 %

3	ІП-13_Радзівіло_ПЗ [redacted] 5/30/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)	36 0.44 %
4	ІП-15_Рибалка_ПЗ [redacted] 6/2/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)	34 0.42 %
5	https://ela.kpi.ua/bitstreams/36b08709-1403-4fe0-b623-3d35bebb19a9/download	31 0.38 %
6	https://ela.kpi.ua/bitstreams/36b08709-1403-4fe0-b623-3d35bebb19a9/download	24 0.30 %
7	Архітектурне рішення веб-застосунку для цифрової дистрибуції [redacted] 3/15/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	24 0.30 %
8	ІП-15_Кондрацька_ПЗ [redacted] 5/30/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)	23 0.28 %
9	ІП-15_Рибалка_ПЗ [redacted] 6/2/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)	19 0.23 %
10	Архітектурне рішення для файлового сховища [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	18 0.22 %

з домашньої бази даних (10.43 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	ІП-15_Кондрацька_ПЗ [redacted] 5/30/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)**	**132 (14) 1.63 %**
2	ІП-15_Рибалка_ПЗ [redacted] 6/2/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)**	**98 (6) 1.21 %**
3	ІП-15_Лазьов_ПЗ [redacted] 5/31/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)**	**93 (2) 1.15 %**
4	ІП-13_Радзівіло_ПЗ [redacted] 5/30/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)**	**70 (5) 0.87 %**
5	Мобільний застосунок для створення плану харчування [redacted] 3/16/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**57 (6) 0.70 %**

6	Архітектурне рішення веб-застосунку для цифрової дистрибуції [redacted] 3/15/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**52 (3) 0.64 %**
7	Мобільний застосунок "Антологія інтелектуальних ігор-головоломок" [redacted] 3/16/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**36 (5) 0.44 %**
8	Графічний конструктор SQL-запитів для взаємодії з базами даних [redacted] 3/16/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**28 (3) 0.35 %**
9	ІП-13_Лопоша_ПЗ [redacted] 5/30/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)**	**27 (2) 0.33 %**
10	ІП-15_Чорній_ПЗ [redacted] 6/2/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)**	**26 (3) 0.32 %**
11	Комп'ютера гра у жанрі Roguelike на рушії Godot [redacted] 3/16/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**24 (2) 0.30 %**
12	Архітектурне рішення для файлового сховища [redacted] 3/16/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**24 (2) 0.30 %**
13	ІП-11_Трикош_ПЗ [redacted] 5/28/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)**	**22 (2) 0.27 %**
14	ІП-15_Химич_ПЗ [redacted] 5/28/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)**	**21 (3) 0.26 %**
15	Програмне забезпечення для виявлення аномалій на МРТ знімках [redacted] 3/16/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**20 (2) 0.25 %**
16	ІП-13_Саіян_ПЗ [redacted] 5/30/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)	20 (3) 0.25 %
17	Веб-застосунок для ігрової соціальної платформи [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	14 (2) 0.17 %
18	Веб-сервіс прогнозування курсів криптовалют [redacted] 3/15/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**11 (1) 0.14 %**

19	Веб-застосунок для проведення опитування у режимі онлайн голосування [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	10 (2) 0.12 %
20	ІП-13_Шиманська_ПЗ [redacted] 6/1/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)**	**10 (1) 0.12 %**
21	ІП-12_Васильєв_ПЗ [redacted] 6/1/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)	9 (1) 0.11 %
22	Програмне забезпечення для організації та планування маршрутів та поїздок [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	9 (1) 0.11 %
23	ІП-11_Панченко_ПЗ [redacted] 5/29/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)	7 (1) 0.09 %
24	Програмний сервіс надання інформаційних послуг на основі поточної геопозиції [redacted] 3/15/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	7 (1) 0.09 %
25	ІП-15_Аджигельдієва_ПЗ [redacted] 5/30/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)	6 (1) 0.07 %
26	ІП-11_Тихонов_ПЗ [redacted] 5/31/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)	6 (1) 0.07 %
27	Сервіс сповіщення про відключення електроенергії в телеграм-боті [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	5 (1) 0.06 %

з програми обміну базами даних (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з Інтернету (2.93 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	**https://ela.kpi.ua/bitstreams/36b08709-1403-4fe0-b623-3d35bebb19a9/download**	**78 (4) 0.96 %**
2	**https://openarchive.nure.ua/bitstreams/4c03742c-7ee8-4b7b-9e52-ed09030b5062/download**	**46 (1) 0.57 %**
3	https://ela.kpi.ua/bitstreams/8c33faf5-fcf0-4e2d-b863-2ef7cd2ed818/download	39 (6) 0.48 %
4	**https://ela.kpi.ua/bitstreams/94676364-e423-4a0b-8a53-31353181de51/download**	**21 (3) 0.26 %**
5	https://ela.kpi.ua/bitstream/123456789/63871/1/Riejek_bakalavr.pdf	15 (2) 0.19 %
6	**https://ela.kpi.ua/bitstreams/772098f0-4a96-4b00-8243-06f0e106d7be/download**	**12 (1) 0.15 %**

7	**https://ela.kpi.ua/bitstreams/16089a47-ab51-4586-baae-4063fdeb7dee/download**	**10 (1) 0.12 %**
8	https://ela.kpi.ua/server/api/core/bitstreams/f463af88-756b-424a-bca4-3da434526966/content	9 (1) 0.11 %
9	https://ela.kpi.ua/server/api/core/bitstreams/a5cb95ec-11a8-40e1-9b06-d3455c267a2d/content	7 (1) 0.09 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2025 р.

МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ РОБОТИ З МУЗИЧНИМИ НОТАМИ

Текст програми

КПІ.ІП-1132.045440.03.12

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Світлана ПОПЕРЕШНЯК

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Вікторія ФУКС

Київ – 2025

Посилання на репозиторій з повним текстом програмного коду
https://github.com/YazheV1ka/MIDIConverter

Файл main.py
Реалізація функціональної задачі прийому PDF-файлу з нотами, запуску фонової обробки, оновлення темпу та завантаження готових файлів у форматах mp3 і mp4.

```
import my_funcs
from flask import Flask, request, send_file
from zipfile import ZipFile
import time
import os
import threading
import io

app = Flask(__name__)


@app.route('/file/create', methods=['POST'])
def upload_file():
    if 'metronome_speed' not in request.form:
        return 'No metronome_speed part', 400
    metronome_speed = int(request.form['metronome_speed'])
    if 'file' not in request.files:
        return 'No file part', 400
    file = request.files['file']
    if file.filename == '':
        return 'No selected file', 400
    file = request.files['file']
    unique_id = str(int(time.time() * 10))
    output_directory = "D:/Liszt/extra/" + unique_id
    if not os.path.exists(output_directory):
        os.makedirs(output_directory)
    pdf_file_path = output_directory + "/" + unique_id + ".pdf"
    file.save(pdf_file_path)
    thread =
threading.Thread(target=my_funcs.convert_pdf_to_musicxml,
args=(unique_id, metronome_speed))
    thread.start()
    return unique_id, 200


@app.route('/file/update/<unique_id>', methods=['PUT'])
def update_file(unique_id):
    if 'metronome_speed' not in request.form:
        return 'No metronome_speed part', 400
    metronome_speed = int(request.form['metronome_speed'])
    unique_id = str(unique_id)
    my_funcs.create_midi_mp34(unique_id, metronome_speed)
    return unique_id, 200
```

```
@app.route('/file/<unique_id>', methods=['GET'])
def get_file(unique_id):
    unique_id = str(unique_id)
    output_directory = "D:/Liszt/extra/" + unique_id
    mp3_file_path = output_directory + "/" + unique_id + ".mp3"
    mp4_file_path = output_directory + "/" + unique_id + ".mp4"
    if not os.path.exists(mp4_file_path):
        return 'No file yet', 404
    file_paths = [mp3_file_path, mp4_file_path]
    memory_file = io.BytesIO()
    with ZipFile(memory_file, 'w') as zipf:
        # Writing each file to the zip
        for file in file_paths:
            zipf.write(file, os.path.basename(file))
    memory_file.seek(0)
    return send_file(memory_file, mimetype='zip',
download_name="files.zip", as_attachment=True)


if __name__ == '__main__':
    app.run(debug=True)
```

Файл my_funcs.py

Реалізація функціональної задачі обробки нотного PDF файлу - розпізнавання, конвертацію у MusicXML, створення MIDI, MP3 та MP4 із синхронізованим аудіо та нотами.

```
from moviepy import ImageClip, AudioFileClip,
concatenate_videoclips
import subprocess
import os
import tempfile
from music21 import converter, stream, tempo
import partitura as pt
from pydub import AudioSegment
from PIL import Image

soundfont = 'D:/Liszt/soundfonts/ad.sf3'
audiveris_path = 'D:/Liszt/Audiveris/bin/Audiveris.bat'
musescore_path = 'D:/Liszt/MuseScore/bin/MuseScore4.exe'

ROW_SIZE = 2

FPS = 2


def convert_pdf_to_musicxml(unique_id, metronome_speed):
```

```
    output_directory = os.path.join("D:/Liszt/extra/",
unique_id)
    base = os.path.join(output_directory, unique_id)
    pdf = base + ".pdf"
    mxl = base + ".mxl"

    os.makedirs(output_directory, exist_ok=True)
    subprocess.run([audiveris_path, '-batch', pdf, '-output',
output_directory, '-export'], shell=True, check=True)
    subprocess.run([musescore_path, mxl, '-o', mxl], check=True)
    create_midi_mp34(unique_id, metronome_speed)


def create_midi_mp34(unique_id, metronome_speed):
    base = os.path.join("D:/Liszt/extra/", unique_id, unique_id)
    mxl = base + ".mxl"
    mid = base + ".mid"
    wav = base + ".wav"
    mp3 = base + ".mp3"

    score = converter.parse(mxl)
    score.insert(0, tempo.MetronomeMark(number=metronome_speed))
    score.write('midi', fp=mid)

    os.system(f'fluidsynth -ni {soundfont} {mid} -F {wav} -r
44100')
    AudioSegment.from_wav(wav).export(mp3, format='mp3')
    create_timed_mp4(unique_id, metronome_speed)


def split_score_to_measure_xmls(mxl_file, metronome_speed):
    full = converter.parse(mxl_file)
    parts = full.parts
    secs_per_beat = 60.0 / metronome_speed

    xml_paths = []
    durations = []
    measures = parts[0].getElementsByClass(stream.Measure)
    total = len(measures)


    for start in range(1, total + 1, ROW_SIZE):
        end = min(start + ROW_SIZE - 1, total)
        tmp = tempfile.NamedTemporaryFile(suffix='.xml',
delete=False)
        tmp.close()

        sc = stream.Score()
        sc.insert(0,
tempo.MetronomeMark(number=metronome_speed))
        group_dur = 0
        for p in parts:
            part_copy = stream.Part()
            for idx in range(start, end + 1):
```

```
                meas = p.measure(idx)
                if meas:
                    part_copy.append(meas)
                    if p == parts[0]:
                        group_dur +=
meas.barDuration.quarterLength * secs_per_beat
            sc.append(part_copy)

        sc.write('musicxml', fp=tmp.name)
        xml_paths.append(tmp.name)
        durations.append(group_dur)

    return xml_paths, durations


def render_and_merge_rows(xml_paths):
    png_pages = []

    row_images = []
    for xml in xml_paths:
        score = pt.load_musicxml(xml)
        png = xml.replace('.xml', '.png')
        pt.render(score, out_fn=png)
        row_images.append(png)

    for i in range(0, len(row_images), 2):
        top = row_images[i]
        bottom = row_images[i + 1] if i + 1 < len(row_images)
else None
        img_top = Image.open(top)
        if bottom:
            img_bottom = Image.open(bottom)
            width = max(img_top.width, img_bottom.width)
            height = img_top.height + img_bottom.height
            merged = Image.new('RGB', (width, height),
color=(255, 255, 255))
            merged.paste(img_top, (0, 0))
            merged.paste(img_bottom, (0, img_top.height))
        else:
            merged = img_top
        merged_path = top.replace('_row', '_page')
        merged.save(merged_path)
        png_pages.append(merged_path)
    return png_pages


def create_timed_mp4(unique_id, metronome_speed):
    base = os.path.join("D:/Liszt/extra/", unique_id, unique_id)
    mxl = base + ".mxl"
    mp3 = base + ".mp3"
    mp4 = base + ".mp4"

    xmls, durs = split_score_to_measure_xmls(mxl,
metronome_speed)
```

```
    pages = render_and_merge_rows(xmls)


    page_durs = []
    for i in range(0, len(durs), 2):
        total_d = durs[i] + (durs[i + 1] if i + 1 < len(durs)
else 0)
        page_durs.append(total_d)


    w, h = Image.open(pages[0]).size
    w += w % 2
    h += h % 2

    audio = AudioFileClip(mp3)
    clips = [ImageClip(p).with_duration(d).resized((w,
h)).with_position('center')
             for p, d in zip(pages, page_durs)]

    video = concatenate_videoclips(clips,
method='compose').with_audio(audio)
    video.write_videofile(mp4, fps=FPS, codec='libx264',
audio_codec='aac')
```

Файл FragmentEdit.kt

Реалізація функціональної задачі редагування інформації про нотний файл у, зокрема назви, опису, темпу та дати оновлення, а також дозволяє зберігати зміни або видаляти записи.

```
package com.app.midiconverter.edit

import android.content.ContentValues
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import androidx.fragment.app.Fragment
import com.app.midiconverter.R
import com.app.midiconverter.ReadDbHelper
import com.app.midiconverter.databinding.FragmentEditBinding
import com.app.quizz.home.FragmentNotes
import java.text.SimpleDateFormat
import java.util.Date
import java.util.Locale
import java.util.TimeZone

class FragmentEdit : Fragment() {
    private val binding by lazy { FragmentEditBinding.inflate(layoutInflater)
}

    override fun onCreateView(
        inflater: LayoutInflater,
```

```
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        val file_id = arguments?.getInt("file_id")
        println("file_id: $file_id")

        val dbHelper = ReadDbHelper(requireContext())
        val db = dbHelper.writableDatabase

        val cursorCourses = db.rawQuery("SELECT * FROM NOTES_BD WHERE file_Id
= '$file_id'", null)

        if (cursorCourses.moveToFirst()) {
            do {
                val name = cursorCourses.getColumnIndex("name")
                println("name: $name")
                val descr = cursorCourses.getColumnIndex("description")
                val dateIndex = cursorCourses.getColumnIndex("update_date")
                val dateMillisString = cursorCourses.getString(dateIndex)

                if (dateMillisString != null) {
                    try {
                        val dateMillis = dateMillisString.toLong()
                        val date = Date(dateMillis)

                        val formatter = SimpleDateFormat("dd-MM-yyyy HH:mm",
Locale.getDefault())
                        formatter.timeZone = TimeZone.getTimeZone("GMT+3")
                        val formattedDate = formatter.format(date)

                        binding.updateDate.text = "Update Date:
$formattedDate"
                    } catch (e: Exception) {
                        binding.updateDate.text = "Update Date: "
                        e.printStackTrace()
                    }
                }
                binding.name.setText(cursorCourses.getString(name))
                binding.description.setText(cursorCourses.getString(descr))
            } while (cursorCourses.moveToNext())
        }
        cursorCourses.close()

        val cursorCourses2 = db.rawQuery("SELECT * FROM FILES_BD WHERE id =
'$file_id'", null)
        if (cursorCourses2.moveToFirst()) {
            do {
                val bpm = cursorCourses2.getColumnIndex("metronome_bpm")

                binding.speed.setText(cursorCourses2.getInt(bpm).toString())
            } while (cursorCourses.moveToNext())
        }
        cursorCourses.close()

        binding.save.setOnClickListener {
            val newName = binding.name.text.toString()
            if (newName.isNotEmpty()) {
                val formatter = SimpleDateFormat("dd-MM-yyyy HH:mm",
```

```
Locale.getDefault())
                formatter.timeZone = TimeZone.getTimeZone("GMT+3")
                val date = Date().time

                val values = ContentValues().apply {
                    put("name", newName)
                    put("description", binding.description.text.toString())
                    put("update_date", date.toString())
                }
                val values2 = ContentValues().apply {
                    if(binding.speed.text.toString().isNullOrEmpty()) {
                        put("metronome_bpm", 100)
                    }
                    else {
                        put("metronome_bpm", binding.speed.text.toString())
                    }
                }
                db.update(
                    "NOTES_BD",
                    values,
                    "file_id = ?",
                    arrayOf(file_id.toString())
                )
                db.update(
                    "FILES_BD",
                    values2,
                    "id = ?",
                    arrayOf(file_id.toString())
                )
                println("Data with name $newName successfully updated.")
                val fragmentTransaction =
parentFragmentManager.beginTransaction()
                fragmentTransaction.replace(R.id.fragmentContainerView,
FragmentNotes())
                fragmentTransaction.commit()

            }else{
                Toast.makeText(requireContext(), "Please fill all required
fields", Toast.LENGTH_LONG).show()
            }
        }
        binding.delete.setOnClickListener {
            db.delete("NOTES_BD",
                "file_id = ?",
                arrayOf(file_id.toString())
            )
            Toast.makeText(requireContext(), "Deleted",
Toast.LENGTH_LONG).show()
            val fragmentTransaction =
parentFragmentManager.beginTransaction()
            fragmentTransaction.replace(R.id.fragmentContainerView,
FragmentNotes())
            fragmentTransaction.commit()
        }
    }
}
```

Файл DataList.kt

Реалізація визначення структури даних для відображення нотних файлів у списку.

```
package com.app.midiconverter.home

data class DataList(
    val id : Int,
    val name : String = "Note$id",
    val description : String? = null,
    val file_id : Int,
    val update_date : String? = null
)
```

Файл FragmentNotes.kt

Реалізація функціональної задачі перегляду, пошуку, сортування та додавання нотних файлів, а також забезпечення переходу до екрану завантаження нового PDF файлу.

```
package com.app.quizz.home

import android.app.Activity
import android.content.ContentResolver
import android.content.Intent
import android.net.Uri
import android.os.Bundle
import android.provider.OpenableColumns
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.PopupMenu
import androidx.activity.result.contract.ActivityResultContracts
import androidx.fragment.app.Fragment
import androidx.recyclerview.widget.LinearLayoutManager
import com.app.midiconverter.R
import com.app.midiconverter.ReadDbHelper
import com.app.midiconverter.databinding.FragmentHomeBinding
import com.app.midiconverter.home.DataList
import com.app.midiconverter.home.ListAdapter
import com.app.midiconverter.uploaded.FragmentUploaded
import java.io.File
import java.io.FileOutputStream
import java.io.IOException
import java.io.InputStream
import java.io.OutputStream


class FragmentNotes : Fragment() {

    private val binding by lazy { FragmentHomeBinding.inflate(layoutInflater)
}
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
```

```
    ): View? {
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        val dbHelper = ReadDbHelper(requireContext())
        val db = dbHelper.readableDatabase

        val cursorCourses = db.rawQuery("SELECT * FROM NOTES_BD", null)

        var listMain : ArrayList<DataList> = arrayListOf()

        if (cursorCourses.moveToFirst()) {
            do {
                val column_id = cursorCourses.getColumnIndex("id")
                val name = cursorCourses.getColumnIndex("name")
                val description = cursorCourses.getColumnIndex("description")
                val file_id = cursorCourses.getColumnIndex("file_Id")
                val update_date = cursorCourses.getColumnIndex("update_date")
                listMain.add(
                    DataList(
                        cursorCourses.getInt(column_id),
                        cursorCourses.getString(name),
                        cursorCourses.getString(description),
                        cursorCourses.getInt(file_id),
                        cursorCourses.getString(update_date)
                    )
                )
            } while (cursorCourses.moveToNext())
        }
        cursorCourses.close()

        var list: ArrayList<DataList> = arrayListOf()
        list.addAll(listMain)

        val adapter = ListAdapter(list, parentFragmentManager)

        binding.recyclerView.layoutManager =
LinearLayoutManager(requireContext())
        binding.recyclerView.adapter = adapter

        binding.searchBtn.setOnClickListener{
            val search_key = binding.search.text.toString()
            val filteredList : ArrayList<DataList> = ArrayList(
                listMain.filter{
                    it.name.lowercase().contains(search_key.lowercase()) ||
it.description?.lowercase()?.contains(search_key.lowercase()) == true
                }
            )
            val size = list.size
            list.clear()
            binding.recyclerView.adapter?.notifyItemRangeRemoved(0, size)
            list.addAll(filteredList)
            binding.recyclerView.adapter?.notifyItemRangeInserted(0,
list.size)
        }


        binding.sort.setOnClickListener{
            val popupMenu = PopupMenu(requireContext(), binding.sort)
            popupMenu.getMenuInflater().inflate(R.menu.pop_up_items,
popupMenu.getMenu())
```

```
            popupMenu.setOnMenuItemClickListener { menuItem ->
                val sortedList = if(menuItem.itemId == R.id.menu_name) {
                    ArrayList(list.sortedBy { it.name })
                } else if(menuItem.itemId == R.id.menu_description) {
                    ArrayList(list.sortedBy { it.description })
                } else {
                    ArrayList(list.sortedByDescending{it.update_date})
                }
                val size = list.size
                list.clear()
                binding.recyclerView.adapter?.notifyItemRangeRemoved(0, size)
                list.addAll(sortedList)
                binding.recyclerView.adapter?.notifyItemRangeInserted(0,
list.size)
                true
            }
            popupMenu.show()
        }

        val pickFile =

registerForActivityResult(ActivityResultContracts.StartActivityForResult()) {
result ->
                if (result.resultCode == Activity.RESULT_OK) {
                    val selectedFileUri: Uri? = result.data?.data
                    selectedFileUri?.let { uri ->
                        handleSelectedFile(uri)
                    }
                }
            }

        binding.addNew.setOnClickListener {
            val intent = Intent(Intent.ACTION_OPEN_DOCUMENT).apply {
                addCategory(Intent.CATEGORY_OPENABLE)
                type = "application/pdf"
            }
            pickFile.launch(intent)
        }
    }
    private fun handleSelectedFile(uri: Uri) {
        println("Selected File URI: $uri")
        val contentResolver: ContentResolver =
requireContext().contentResolver
        val cursor = contentResolver.query(uri, null, null, null, null)

        cursor?.use { cursor ->
            if (cursor.moveToFirst()) {
                val columnIndex =
cursor.getColumnIndex(OpenableColumns.DISPLAY_NAME)
                if (columnIndex > -1) {
                    val displayName =
cursor.getString(columnIndex).replace("'","").replace("\"", "")
                    println("display name : $displayName")

                    val inputStream: InputStream? =
contentResolver.openInputStream(uri)

                    val newFile = createNewFile(displayName)

                    copyFile(inputStream, FileOutputStream(newFile))

                    val bundle = Bundle()
                    bundle.putString("absolute_pass", newFile.absolutePath)
```

```
                    val fragment = FragmentUploaded()
                    fragment.arguments = bundle

                    val fragmentTransaction =
parentFragmentManager.beginTransaction()
                    fragmentTransaction.replace(R.id.fragmentContainerView,
fragment)
                    fragmentTransaction.addToBackStack("")
                    fragmentTransaction.commit()
                }
            }
        }
    }
    private fun createNewFile(fileName: String): File {
        val storageDir = requireContext().cacheDir
        return File.createTempFile(fileName, null, storageDir)
    }

    private fun copyFile(inputStream: InputStream?, outputStream:
OutputStream) {
        if (inputStream == null) return

        try {
            val buffer = ByteArray(1024)
            var length: Int
            while (inputStream.read(buffer).also { length = it } > 0) {
                outputStream.write(buffer, 0, length)
            }
        } catch (e: IOException) {
            Log.e("FileCopy", "Error copying file", e)
        } finally {
            try {
                inputStream.close()
                outputStream.close()
            } catch (e: IOException) {
                Log.e("FileCopy", "Error closing streams", e)
            }
        }
    }
}
```

Файл ListAdapter.kt

Реалізація функціональної задачі відображення елементів списку нот у RecyclerView та переходів до фрагментів відтворення (FragmentPlay) або редагування (FragmentEdit) нот.

```
package com.app.midiconverter.home

import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.core.os.bundleOf
import androidx.fragment.app.FragmentManager
import androidx.recyclerview.widget.RecyclerView
import com.app.midiconverter.R
import com.app.midiconverter.databinding.ItemBinding
import com.app.midiconverter.edit.FragmentEdit
import com.app.midiconverter.player.FragmentPlay
import java.text.SimpleDateFormat
```

```
import java.util.Date
import java.util.Locale
import java.util.TimeZone

class ListAdapter(private val itemList: ArrayList<DataList>, val
parentFragmentManager: FragmentManager) :
RecyclerView.Adapter<ListAdapter.MyViewHolder>(){

    class MyViewHolder(private val binding : ItemBinding) :
RecyclerView.ViewHolder(binding.root) {
        fun bind(item : DataList, parentFragmentManager: FragmentManager) {

            binding.listen.setOnClickListener {
                val bundle = bundleOf("file_id" to item.file_id)
                val fragment = FragmentPlay()
                fragment.arguments = bundle
                val fragmentTransaction =
parentFragmentManager.beginTransaction()
                fragmentTransaction.replace(R.id.fragmentContainerView,
fragment)
                fragmentTransaction.addToBackStack("")
                fragmentTransaction.commit()
            }

            binding.custom.setOnClickListener {
                val bundle = bundleOf("file_id" to item.file_id)
                println("file_id: ${item.file_id}")
                val fragment = FragmentEdit()
                fragment.arguments = bundle
                val fragmentTransaction =
parentFragmentManager.beginTransaction()
                fragmentTransaction.replace(R.id.fragmentContainerView,
fragment)
                fragmentTransaction.addToBackStack("")
                fragmentTransaction.commit()
            }
            binding.title.text = item.name
            binding.description.text = item.description

            val date = Date(item.update_date!!.toLong())
            val formatter = SimpleDateFormat("dd-MM-yyyy HH:mm",
Locale.getDefault())
            formatter.timeZone = TimeZone.getTimeZone("GMT+3")
            binding.updateDate.text = formatter.format(date)
        }
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
MyViewHolder {
        return
MyViewHolder(ItemBinding.inflate(LayoutInflater.from(parent.context), parent,
false))
    }

    override fun getItemCount(): Int {
        return itemList.size
    }

    override fun onBindViewHolder(holder: MyViewHolder, position: Int) {
        holder.bind(itemList.get(position),parentFragmentManager)
    }
}
```

Файл FragmentPlay.kt

Реалізація функціональної задачі відтворення mp3 або mp4 нотного запису за допомогою ExoPlayer, а також оновлює дату останнього відтворення.

```
package com.app.midiconverter.player

import android.content.ContentValues
import android.os.Build
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.annotation.OptIn
import androidx.annotation.RequiresApi
import androidx.fragment.app.Fragment
import androidx.media3.common.MediaItem
import androidx.media3.common.util.UnstableApi
import androidx.media3.datasource.FileDataSource
import androidx.media3.exoplayer.ExoPlayer
import androidx.media3.exoplayer.source.ProgressiveMediaSource
import com.app.midiconverter.R
import com.app.midiconverter.ReadDbHelper
import com.app.midiconverter.databinding.FragmentPlayBinding
import com.app.quizz.home.FragmentNotes
import java.text.SimpleDateFormat
import java.util.Date
import java.util.Locale
import java.util.TimeZone


class FragmentPlay : Fragment() {
    private val binding by lazy { FragmentPlayBinding.inflate(layoutInflater)
}
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        return binding.root
    }

    @RequiresApi(Build.VERSION_CODES.O)
    @OptIn(UnstableApi::class)
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        val file_id = arguments?.getInt("file_id", -1)
        var mp3 = ""
        var mp4 = ""
        val dbHelper = ReadDbHelper(requireContext())
        val db = dbHelper.writableDatabase

        binding.back.setOnClickListener{
            val fragmentTransaction =
parentFragmentManager.beginTransaction()
            fragmentTransaction.replace(R.id.fragmentContainerView,
FragmentNotes())
            fragmentTransaction.commit()
        }

        if(file_id == -1){
            val bpm = arguments?.getString("bpm")
```

```
            mp3 = arguments?.getString("mp3").toString()
            mp4 = arguments?.getString("mp4").toString()
            binding.speedTxt.text = "Metronome speed: ${bpm}c"
        }else {

            val cursorCourses2 = db.rawQuery("SELECT * FROM FILES_BD WHERE id
= '$file_id'", null)

            if (cursorCourses2.moveToFirst()) {
                do {
                    val metronome_bpm =
cursorCourses2.getColumnIndex("metronome_bpm")
                    val mp3_path = cursorCourses2.getColumnIndex("mp3_path")
                    val mp4_path = cursorCourses2.getColumnIndex("mp4_path")

                    mp3 = cursorCourses2.getString(mp3_path)
                    mp4 = cursorCourses2.getString(mp4_path)

                    binding.speedTxt.text =
                        "Metronome speed:
${cursorCourses2.getInt(metronome_bpm)}c"


                } while (cursorCourses2.moveToNext())
            }
            cursorCourses2.close()
        }

        val formatter = SimpleDateFormat("dd-MM-yyyy HH:mm",
Locale.getDefault())
        formatter.timeZone = TimeZone.getTimeZone("GMT+3")

        val formattedDateString = formatter.format(Date())

        val date: Date = formatter.parse(formattedDateString)!!

        val currentDateInMillis: Long = date.time
        val values2 = ContentValues().apply {
            put("update_date", currentDateInMillis)
        }
        db.update("NOTES_BD",
            values2,
            "file_id = ?",
            arrayOf(file_id.toString()) )

        val player = ExoPlayer.Builder(requireContext()).build()
        binding.playerView.player = player
        println("MP4 : $mp4")
        player.addMediaSource(
            ProgressiveMediaSource.Factory(FileDataSource.Factory())
                .createMediaSource(MediaItem.fromUri(mp4)))
        player.prepare()
        player.play()
        binding.switcher.setOnCheckedChangeListener { compoundButton, b ->
            player.stop()
            when (b) {
                true -> player.replaceMediaItem(0, MediaItem.fromUri(mp3))

                false -> player.replaceMediaItem(0, MediaItem.fromUri(mp4))
            }
            player.prepare()
            player.play()
        }
    }
```

```
    override fun onDestroyView() {
        binding.playerView.player?.stop()
        binding.playerView.player?.release()
        super.onDestroyView()
    }

}
```

Файл FragmentSettings.kt

Реалізація функціональної задачі зміни налаштувань застосунку, таких як тема та режим статистики.

```
package com.app.midiconverter.settings

import android.content.Context
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.appcompat.app.AppCompatDelegate
import androidx.fragment.app.Fragment
import com.app.midiconverter.databinding.FragmentSettingBinding
import androidx.core.content.edit

class FragmentSettings : Fragment() {
    private val binding by lazy {
FragmentSettingBinding.inflate(layoutInflater) }

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        val sharedPref = activity?.getSharedPreferences("settings_property",
Context.MODE_PRIVATE)

        val theme_mode = sharedPref?.getBoolean("theme_mode", false)
        val statistic_on = sharedPref?.getBoolean("statistic_mode", true)

        if(theme_mode == true){
            binding.themeSwitch.isChecked = true
        }else if(theme_mode == false){
            binding.themeSwitch.isChecked = false
        }

        if(statistic_on == true){
            binding.statisticSwitch.isChecked = true
        }else if(statistic_on == false){
            binding.statisticSwitch.isChecked = false
        }
```

```
        binding.themeSwitch.setOnCheckedChangeListener { _, isChecked ->
            sharedPref?.edit {
                putBoolean("theme_mode", isChecked)
            }

            if (isChecked) {

AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_YES)
            } else {

AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO)
            }
        }


        binding.statisticSwitch.setOnCheckedChangeListener { compoundButton,
b ->
            if(b){
                binding.statisticSwitch.isChecked = true
                sharedPref?.edit {
                    putBoolean("statistic_mode", true)
                }
            }else{
                binding.statisticSwitch.isChecked = false
                sharedPref?.edit {
                    putBoolean("statistic_mode", false)
                }
            }
        }

    }
}
```

Файл FragmentHome.kt

Реалізація функціональної задачі головної сторінки, з можливістю завантаження PDF файлу, а також ініціацією переходу до екрану завантаження з передачею шляху до файлу.

```
package com.app.midiconverter.statistic

import android.app.Activity
import android.content.ContentResolver
import android.content.Context
import android.content.Intent
import android.net.Uri
import android.os.Build
import android.os.Bundle
import android.provider.OpenableColumns
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.activity.result.contract.ActivityResultContracts
import androidx.annotation.RequiresApi
import androidx.appcompat.app.AppCompatDelegate
import androidx.fragment.app.Fragment
import com.app.midiconverter.R
import com.app.midiconverter.ReadDbHelper
```

```
import com.app.midiconverter.databinding.FragmentStatisticBinding
import com.app.midiconverter.uploaded.FragmentUploaded
import java.io.File
import java.io.FileOutputStream
import java.io.IOException
import java.io.InputStream
import java.io.OutputStream
import java.time.LocalDate
import java.time.ZoneId


class FragmentHome : Fragment() {

    private fun handleSelectedFile(uri: Uri) {
        println("Selected File URI: $uri")
        val contentResolver: ContentResolver =
requireContext().contentResolver
        val cursor = contentResolver.query(uri, null, null, null, null)

        cursor?.use { cursor ->
            if (cursor.moveToFirst()) {
                val columnIndex =
cursor.getColumnIndex(OpenableColumns.DISPLAY_NAME)
                if (columnIndex > -1) {
                    val displayName =
                        cursor.getString(columnIndex).replace("'",
"").replace("\"", "")
                    println("display name : $displayName")

                    val inputStream: InputStream? =
contentResolver.openInputStream(uri)

                    val newFile = createNewFile(displayName)

                    copyFile(inputStream, FileOutputStream(newFile))

                    val bundle = Bundle()
                    bundle.putString("absolute_pass", newFile.absolutePath)
                    val fragment = FragmentUploaded()
                    fragment.arguments = bundle

                    val fragmentTransaction =
parentFragmentManager.beginTransaction()
                    fragmentTransaction.replace(R.id.fragmentContainerView,
fragment)
                    fragmentTransaction.addToBackStack("")
                    fragmentTransaction.commit()
                }
            }
        }
    }

    private fun createNewFile(fileName: String): File {
        val storageDir = requireContext().cacheDir
        return File.createTempFile(fileName, null, storageDir)
    }

    private fun copyFile(inputStream: InputStream?, outputStream:
OutputStream) {
        if (inputStream == null) return

        try {
            val buffer = ByteArray(1024)
            var length: Int
```

```
            while (inputStream.read(buffer).also { length = it } > 0) {
                outputStream.write(buffer, 0, length)
            }
        } catch (e: IOException) {
            Log.e("FileCopy", "Error copying file", e)
        } finally {
            try {
                inputStream.close()
                outputStream.close()
            } catch (e: IOException) {
                Log.e("FileCopy", "Error closing streams", e)
            }
        }
    }

    private val binding by lazy {
FragmentStatisticBinding.inflate(layoutInflater) }
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        return binding.root
    }

    @RequiresApi(Build.VERSION_CODES.O)
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        val pickFile =

registerForActivityResult(ActivityResultContracts.StartActivityForResult()) {
result ->
                if (result.resultCode == Activity.RESULT_OK) {
                    val selectedFileUri: Uri? = result.data?.data
                    selectedFileUri?.let { uri ->
                        handleSelectedFile(uri)
                    }
                }
            }

        binding.uploadBtn.setOnClickListener {
            val intent = Intent(Intent.ACTION_GET_CONTENT).apply {
                addCategory(Intent.CATEGORY_OPENABLE)
                type = "application/pdf"
            }
            pickFile.launch(intent)
        }


        val sharedPref =
            requireContext().getSharedPreferences("settings_property",
Context.MODE_PRIVATE)

        val theme_mode = sharedPref?.getBoolean("theme_mode", false)

        if (theme_mode == true) {

AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_YES)
        } else if (theme_mode == false) {

AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO)
        }
```

```
        val last_date = sharedPref.getString("last_date", "")
        val statistic_on = sharedPref.getBoolean("statistic_mode", true)
        val current_date = LocalDate.now(ZoneId.of("GMT+3"))
        val lastMonth = sharedPref.getInt("last_month", -1)
        val currentMonth = current_date.monthValue

        var lastCheckedDate = last_date
        val dbHelper = ReadDbHelper(requireContext())
        val db = dbHelper.readableDatabase

        if (statistic_on) {
            binding.statisticTxt.visibility = View.VISIBLE
            binding.cloud.visibility = View.VISIBLE
            binding.goodJobTxt.visibility = View.INVISIBLE

            val notesCursor = db.rawQuery(
                "SELECT COUNT(*) FROM NOTES_BD WHERE update_date = ?",
                arrayOf(current_date.toString())
            )
            notesCursor.moveToFirst()
            val notesCount = notesCursor.count
            notesCursor.close()

            val editor = sharedPref.edit()

            if (lastMonth != currentMonth) {
                editor.putInt("strike", 0)
                editor.putInt("last_month", currentMonth)
                editor.apply()
            }

            val updatedStrikeDay = sharedPref.getInt("strike", 0)

            if (
                notesCount > 0 &&
                !last_date.isNullOrEmpty() &&
                last_date == current_date.minusDays(1).toString() &&
                lastCheckedDate != current_date.toString()
            ) {
                editor.putInt("strike", updatedStrikeDay + 1)
                editor.putString("last_date", current_date.toString())
                editor.apply()

                binding.statisticTxt.text = "Your strike days\nthis month -
${updatedStrikeDay + 1}"
            } else {
                binding.statisticTxt.text = "Your strike days\nthis month -
$updatedStrikeDay"
            }
        } else {
            binding.statisticTxt.visibility = View.INVISIBLE
            binding.cloud.visibility = View.VISIBLE
            binding.goodJobTxt.visibility = View.VISIBLE
        }
    }
}
```

Файл FragmentUploaded.kt

Реалізація функціональної задачі завантаження обраного PDF файлу на сервер, отримання у відповідь ZIP-архів з файлами форматів mp3 і mp4, його

розпакування і збереження (або передачі напряму до плеєру), і переходу до екрану відтворення.

```
package com.app.midiconverter.uploaded

import android.content.ContentValues
import android.content.Context
import android.os.Build
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import androidx.annotation.RequiresApi
import androidx.core.os.bundleOf
import androidx.core.view.isVisible
import androidx.fragment.app.Fragment
import androidx.lifecycle.lifecycleScope
import com.app.midiconverter.ApiService
import com.app.midiconverter.R
import com.app.midiconverter.ReadDbHelper
import com.app.midiconverter.databinding.FragmentUploadedBinding
import com.app.midiconverter.player.FragmentPlay
import com.bumptech.glide.Glide
import com.google.android.material.tabs.TabLayout
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.ExperimentalCoroutinesApi
import kotlinx.coroutines.launch
import kotlinx.coroutines.withContext
import okhttp3.MediaType
import okhttp3.MultipartBody
import okhttp3.RequestBody
import okhttp3.ResponseBody
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory
import java.io.File
import java.io.FileOutputStream
import java.net.HttpURLConnection
import java.net.URL
import java.text.SimpleDateFormat
import java.time.LocalDate
import java.time.ZoneId
import java.util.Date
import java.util.Locale
import java.util.TimeZone
import java.util.zip.ZipInputStream


class FragmentUploaded : Fragment() {
    private val binding by lazy {
FragmentUploadedBinding.inflate(layoutInflater) }
    private val BASE_URL = "https://detail-urge-foster-
ripe.trycloudflare.com"

    val apiService: ApiService by lazy {
        val retrofit = Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
```

```
            .build()

        retrofit.create(ApiService::class.java)
    }

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        return binding.root
    }

    private var file_id = -1
    private lateinit var file: File

    @RequiresApi(Build.VERSION_CODES.O)
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        val receivedData = arguments?.getString("absolute_pass")
        file = File(receivedData)
        val dbHelper = ReadDbHelper(requireContext())
        val db = dbHelper.writableDatabase
        val cursorCourses = db.rawQuery("SELECT * FROM FILES_BD WHERE name =
?", arrayOf(file.name))
        cursorCourses.close()
        val name = getFileExtension(file.name.toString())

        val multiPart = MultipartBody.Part.createFormData(
            "file",
            name,
            RequestBody.create(MediaType.parse("audio/*"), file)
        )
        println("file name: ${name}")

        binding.name.setText(name)
        binding.linearProgress.setOnClickListener { }

        binding.yes.setOnClickListener {
            if (binding.name.text.toString() != "") {
                if (binding.metronomeSpeed.text.toString().isNullOrEmpty()) {
                    addRequest(multiPart, 100)
                } else {
                    addRequest(multiPart,
Integer.parseInt(binding.metronomeSpeed.text.toString()))
                }
            } else {
                Toast.makeText(
                    requireContext(),
                    "Please fill all required fields",
                    Toast.LENGTH_LONG
                ).show()
            }
        }

        binding.no.setOnClickListener {
            if (binding.name.text.toString() != "") {
                if (binding.metronomeSpeed.text.toString().isNullOrEmpty()) {
                    addRequest(multiPart, 100, false)
                } else {
                    addRequest(
                        multiPart,
```

```
Integer.parseInt(binding.metronomeSpeed.text.toString()),
                        false
                    )

                }
            } else {
                Toast.makeText(
                    requireContext(),
                    "Please fill all required fields",
                    Toast.LENGTH_LONG
                ).show()
            }
        }
    }

    @RequiresApi(Build.VERSION_CODES.O)
    fun addStatistic() {
        val sharedPref =
            requireContext().getSharedPreferences("settings_property",
Context.MODE_PRIVATE)
        val edit = sharedPref.edit()
        edit.putString("update_date",
LocalDate.now(ZoneId.of("GMT+3")).toString())
        edit.apply()
        println("last date: ${LocalDate.now(ZoneId.of("GMT+3")).toString()}")
    }

    @OptIn(ExperimentalCoroutinesApi::class)
    fun addRequest(file: MultipartBody.Part, metronome_speed: Int, save:
Boolean = true) {

        binding.linearProgress.isVisible = true
        binding.loadingGif.isVisible = true
        binding.loadingText.isVisible = true
        binding.loadingText.text = "Rendering..."

        Glide.with(requireContext())
            .asGif()
            .load(R.drawable.rendering_gif)
            .into(binding.loadingGif)

        val call: Call<ResponseBody> = apiService.createFile(metronome_speed,
file)

        call.enqueue(object : Callback<ResponseBody> {
            override fun onResponse(call: Call<ResponseBody>, response:
Response<ResponseBody>) {
                if (response.isSuccessful) {
                    val responseData = response.body()?.string()
                    println("response $responseData")

                    val tabLayout =
requireActivity().findViewById<TabLayout>(R.id.tablayout)
                    for (i in 0 until tabLayout.tabCount) {
                        tabLayout.getTabAt(i)?.view?.isEnabled = false
                    }

                    lifecycleScope.launch(Dispatchers.Main) {
                        binding.loadingText.text = "Rendering audio file..."

                        withContext(Dispatchers.IO) {
                            Thread.sleep(40000)
                        }
```

```
                        binding.loadingText.text = "Rendering video file..."

                        withContext(Dispatchers.IO) {
                            Thread.sleep(40000)
                        }

                        if (responseData != null) {
                            processWithSuccessRequest(save, responseData)
                        }

                        binding.loadingGif.isVisible = false
                        binding.loadingText.isVisible = false
                        binding.linearProgress.isVisible = false

                        for (i in 0 until tabLayout.tabCount) {
                            tabLayout.getTabAt(i)?.view?.isEnabled = true
                        }
                    }
                } else {
                    binding.linearProgress.isVisible = false
                    Toast.makeText(requireContext(), "Cannot process file.
Choose a smaller file", Toast.LENGTH_LONG)
                        .show()
                }
            }

            override fun onFailure(call: Call<ResponseBody>, t: Throwable) {
                Toast.makeText(
                    requireContext(),
                    "Something went wrong. Please check your file" +
t.message,
                    Toast.LENGTH_LONG
                ).show()
                t.printStackTrace()
            }
        })
    }

    fun getFileExtension(fileName: String): String? {
        val indexOfMid = fileName.indexOf(".pdf")
        return if (indexOfMid != -1) {
            fileName.substring(0, indexOfMid)
        } else {
            fileName
        }
    }

    fun downloadFile(url: String, destinationDir: String): File {
        val connection = URL(url).openConnection() as HttpURLConnection
        connection.connect()

        val input = connection.inputStream
        val outputFile = File(destinationDir, "downloadedFile.zip")

        val output = FileOutputStream(outputFile)

        val buffer = ByteArray(1024)
        var bytesRead: Int

        while (input.read(buffer).also { bytesRead = it } != -1) {
            output.write(buffer, 0, bytesRead)
        }

        output.close()
```

```
        input.close()

        return outputFile
    }

    fun unzip(zipFile: File, destinationDir: String): List<File> {
        val extractedFiles = mutableListOf<File>()
        val buffer = ByteArray(1024)
        val zipInputStream = ZipInputStream(zipFile.inputStream())

        var zipEntry = zipInputStream.nextEntry
        while (zipEntry != null) {
            val newFile = File(destinationDir, zipEntry.name)
            extractedFiles.add(newFile)

            FileOutputStream(newFile).use { fos ->
                var len: Int
                while (zipInputStream.read(buffer).also { len = it } > 0) {
                    fos.write(buffer, 0, len)
                }
            }

            zipEntry = zipInputStream.nextEntry
        }

        zipInputStream.closeEntry()
        zipInputStream.close()

        return extractedFiles
    }

    private suspend fun processWithSuccessRequest(save: Boolean,
responseData: String) {
        val file2 = requireContext().cacheDir

        val zipFile = try {
            withContext(Dispatchers.IO) {
                downloadFile(
                    "https://detail-urge-foster-ripe.trycloudflare.com/file/"
+ responseData,
                    file2.absolutePath
                )
            }
        } catch (e: Exception) {
            e.printStackTrace()
            withContext(Dispatchers.Main) {
                Toast.makeText(
                    requireContext(),
                    "Cannot process file. Server shutdown or Choose smaller
file",
                    Toast.LENGTH_LONG
                ).show()
            }
            return
        }

        val extractedFiles = unzip(zipFile, file2.absolutePath)
        if (save) {
            val dbHelper = ReadDbHelper(requireContext())

            val db = dbHelper.writableDatabase
            val formatter = SimpleDateFormat("dd-MM-yyyy HH:mm",
Locale.getDefault())
            formatter.timeZone = TimeZone.getTimeZone("GMT+3")
```

```
            val formattedDateString = formatter.format(Date())

            val date: Date = formatter.parse(formattedDateString)!!
            val currentDateInMilliseconds = date.time

            val values2 = ContentValues().apply {
                put("name", binding.name.text.toString())
                put("path", file.absolutePath)
                if (binding.metronomeSpeed.text.toString().isNullOrEmpty()) {
                    put("metronome_bpm", 100)
                } else {
                    put("metronome_bpm",
binding.metronomeSpeed.text.toString())
                }
                put("mp3_path", extractedFiles[0].absolutePath)
                put("mp4_path", extractedFiles[1].absolutePath)
            }

            val newRowId2 = db?.insert("FILES_BD", null, values2)

            val cursorCourses =
                db.rawQuery("SELECT * FROM FILES_BD WHERE path =
'${file.absolutePath}'", null)

            if (cursorCourses.moveToFirst()) {
                do {
                    val column_id = cursorCourses.getColumnIndex("id")
                    file_id = cursorCourses.getInt(column_id)
                } while (cursorCourses.moveToNext())
            }
            cursorCourses.close()

            val values = ContentValues().apply {
                put("name", binding.name.text.toString())
                put("description", binding.description.text.toString())
                put("file_Id", file_id)
                put("update_date", currentDateInMilliseconds)
            }

            val newRowId = db?.insert("NOTES_BD", null, values)

            val fragment = FragmentPlay()
            fragment.arguments = bundleOf("file_id" to file_id)

            val fragmentTransaction =
parentFragmentManager.beginTransaction()
            fragmentTransaction.replace(R.id.fragmentContainerView, fragment)
            fragmentTransaction.commit()
        } else {
            val bundle = Bundle()
            bundle.putString("mp3", extractedFiles[0].absolutePath)
            bundle.putString("mp4", extractedFiles[1].absolutePath)
            if (binding.metronomeSpeed.text.toString().isNullOrEmpty()) {
                bundle.putString("bpm", "100")
            } else {
                bundle.putString("bpm",
binding.metronomeSpeed.text.toString())
            }
            parentFragmentManager.popBackStack()

            val fragment = FragmentPlay()
            fragment.arguments = bundle
            val fragmentTransaction =
```

```
parentFragmentManager.beginTransaction()
            fragmentTransaction.replace(R.id.fragmentContainerView, fragment)
            fragmentTransaction.commit()
        }
    }
}
```

Файл ApiService.kt

Реалізація функціональної задачі надсилання файлу та параметру швидкості метронома на сервер POST запитом.

```
package com.app.midiconverter

import okhttp3.MultipartBody
import okhttp3.RequestBody
import okhttp3.ResponseBody
import retrofit2.Call
import retrofit2.http.Multipart
import retrofit2.http.POST
import retrofit2.http.Part

interface ApiService {
    @POST("/file/create")
    @Multipart
    fun createFile(
        @Part("metronome_speed") metronome_speed: Int,
        @Part file: MultipartBody.Part
    ): Call<ResponseBody>
}
```

Файл MainActivity.kt

Реалізація функціональної задачі керування головною активністю з екранами, переключення між трьома вкладками: головна, нотатки та налаштування.

```
package com.app.midiconverter

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import com.app.midiconverter.databinding.ActivityMainBinding
import com.app.midiconverter.settings.FragmentSettings
import com.app.midiconverter.statistic.FragmentHome
import com.app.quizz.home.FragmentNotes
import com.google.android.material.tabs.TabLayout
import com.google.android.material.tabs.TabLayout.OnTabSelectedListener

class MainActivity : AppCompatActivity() {
    private val binding by lazy { ActivityMainBinding.inflate(layoutInflater)
}

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(binding.root)

        val fragmentHome = FragmentHome()
        val fragmentNotes = FragmentNotes()
```

```
        val fragmentSettings = FragmentSettings()

        binding.tablayout.addOnTabSelectedListener(object :
OnTabSelectedListener{
            override fun onTabSelected(tab: TabLayout.Tab?) {
                supportFragmentManager.popBackStack()
               if (tab?.position == 0) {
                    val fragmentTransaction =
supportFragmentManager.beginTransaction()
                    fragmentTransaction.replace(R.id.fragmentContainerView,
fragmentHome)
                    fragmentTransaction.commit()
                } else if(tab?.position == 1){
                    val fragmentTransaction =
supportFragmentManager.beginTransaction()
                    fragmentTransaction.replace(R.id.fragmentContainerView,
fragmentNotes)
                    fragmentTransaction.commit()
                }else if(tab?.position == 2){
                    val fragmentTransaction =
supportFragmentManager.beginTransaction()
                    fragmentTransaction.replace(R.id.fragmentContainerView,
fragmentSettings)
                    fragmentTransaction.commit()
                }
            }

            override fun onTabUnselected(tab: TabLayout.Tab?) {
            }

            override fun onTabReselected(tab: TabLayout.Tab?) {
                onTabSelected(tab)
            }

        })

    }
}
```

Файл ReadDbHelper.kt

Реалізація функціональної задачі створення та оновлення локальної бази даних з таблицями для нотаток, файлів і статистики.

```
package com.app.midiconverter

import android.content.Context
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper


class ReadDbHelper (context: Context) : SQLiteOpenHelper(context,
DATABASE_NAME, null, DATABASE_VERSION) {

    private val SQL_CREATE_ENTRIES =
        "CREATE TABLE NOTES_BD (" +
                "id INTEGER PRIMARY KEY," +
                "name TEXT," +
                "description TEXT, " +
                "file_Id INTEGER, " +
```

```
                "update_date TEXT)"

    private val SQL_CREATE_FILES =
        "CREATE TABLE FILES_BD (" +
                "id INTEGER PRIMARY KEY," +
                "name TEXT," +
                "path TEXT, " +
                "metronome_bpm INTEGER, " +
                "mp3_path TEXT, " +
                "mp4_path TEXT)"

    private val SQL_STATISTIC =
        "CREATE TABLE STATISTIC_DB (" +
                "id INTEGER PRIMARY KEY," +
                "strike_days INTEGER," +
                "update_date TEXT)"

    private val SQL_DELETE_ENTRIES = "DROP TABLE IF EXISTS NOTES_BD"

    override fun onCreate(db: SQLiteDatabase) {
        db.execSQL(SQL_CREATE_ENTRIES)
        db.execSQL(SQL_CREATE_FILES)
        db.execSQL(SQL_STATISTIC)
    }
    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion:
Int) {
        db.execSQL(SQL_DELETE_ENTRIES)
        onCreate(db)
    }
    override fun onDowngrade(db: SQLiteDatabase, oldVersion: Int, newVersion:
Int) {
        onUpgrade(db, oldVersion, newVersion)
    }
    companion object {
        const val DATABASE_VERSION = 1
        const val DATABASE_NAME = "Notes.db"
    }
}
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2025 р.

МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ РОБОТИ З МУЗИЧНИМИ НОТАМИ

Програма та методика тестування

КПІ.ІП-1132.045440.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Світлана ПОПЕРЕШНЯК

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Вікторія ФУКС

Київ – 2025

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є мобільний додаток і сервер для музичного асистента для розпізнання та обробки музичних нот у форматі PDF.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

- перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;
- перевірка збереження даних;
- знаходження проблем, помилок і недоліків з метою їх усунення;
- перевірка зручності графічного інтерфейсу;

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

- функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;
- системне тестування – перевіряється усе програмне забезпечення в цілому;
- мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення;
- тестування «чорної скриньки» – об’єктом тестування тут є функції присутні у програмі. Перевіряється коректність вихідних даних при заданих вхідних;
- тестування «білої скриньки» – об’єктом тестування тут є внутрішня поведінка програми. Перевіряється коректність побудови всіх елементів програми та правильність їхньої взаємодії один з одним;

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується мануально з метою знаходження помилок та недоліків як у функціональній частині програмного забезпечення так і в зручності користування. Для того, щоб перевірити працездатність та відмовостійкість застосунку, порядок проведення тестування буде наступним:

- тестування на відповідність функціональним вимогам;
- системне тестування;
- тестування «білої скриньки»;
- тестування «чорної скриньки»;
- тестування на виведення повідомлень про помилку, коли це необхідно;
- тестування зручності використання;
- тестування на мобільних пристроях з різною роздільною здатністю екрану;
- тестування на мобільних пристроях з різною версією операційної системи;
- тестування працездатності програми у випадку відсутності з’єднання до мережі;

Під час проведення тестування будуть використовуватись наступні допоміжні засоби:

- стороннє програмне забезпечення: Postman;

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2025 р.

МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ РОБОТИ З МУЗИЧНИМИ НОТАМИ

Керівництво користувача

КПІ.ІП-1132.045440.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Світлана ПОПЕРЕШНЯК

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Вікторія ФУКС

Київ – 2025

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

«Liszt» - це додаток для розпізнавання нот і музичної інтеракції, який спрямований на надання простого творчого інструменту усім бажаючим та загального розвитку музичних навичок у користувачів. Він також може бути корисним інструментом для викладачів та музичних педагогів у процесі навчання учнів. Застосунок покликаний зробити вивчення музики і вдосконалення своїх навичок більш доступним, захопливим та ефективним для широкого кола користувачів.

Додаток може включати в себе функції розпізнавання нот, метроному, зберігання та організації творчих напрацювань, прослуховування композицій, перегляд активності, налаштування візуалу.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Мінімальна конфігурація технічних засобів:

- оперативна пам'ять (RAM): 2 ГБ RAM.
- внутрішня пам'ять: 32 ГБ.
- версія Операційної Системи: Підтримка Android 6.0 (Marshmallow).

Рекомендована конфігурація технічних засобів:

- оперативна пам'ять (RAM): 6 ГБ RAM для забезпечення швидкої обробки даних та алгоритмів.
- внутрішня пам'ять: 128 ГБ внутрішньої пам'яті для зберігання додатку та записів.
- версія Операційної Системи: Підтримка Android версії вище 6.0 для забезпечення сумісності з широким спектром сучасних пристроїв.

2.2 Завантаження застосунку

На даний момент застосунок можна встановити власноруч, використовуючи відповідний apk-файл. Для цього спершу необхідно завантажити його на мобільний пристрій, а потім, використовуючи який-небудь інсталятор, виконати встановлення даного додатку.

2.3 Перевірка коректної роботи

По завершенню встановлення додатка на робочому столі мобільного пристрою повинна відобразитись іконка даного застосунку. У разі, якщо дана іконка не з’явилась, то встановлення відбулось не успішно. Інакше користувач має змогу запустити додаток, клацнувши на його іконку. Після натискання повинна відобразитись початкова сторінка застосунку.

3 ВИКОНАННЯ ПРОГРАМИ

При запуску програмного застосунку користувачу буде відображено головний екран «Home» (з статистикою, за умови її увімкнення в налаштуваннях) з кнопкою, яка дозволяє завантажити файл (Рисунок 3.1). Користувач має змогу завантажити файл або перейти на інші сторінки.

Рисунок 3.1 – Початкова сторінка додатку «Home»

На вкладці «Notes» програмного застосунку користувачу буде відображено бібліотеку з можливостями програти, відредагувати файли, сортувати їх і шукати по літері в імені або опису нотного запису (Рисунок 3.2). Також є можливість додавання файлу.

Рисунок 3.2 – Сторінка «Notes»

На вкладці Setting програмного застосунку користувачу буде відображено можливі налаштування додатку (Рисунок 3.3).

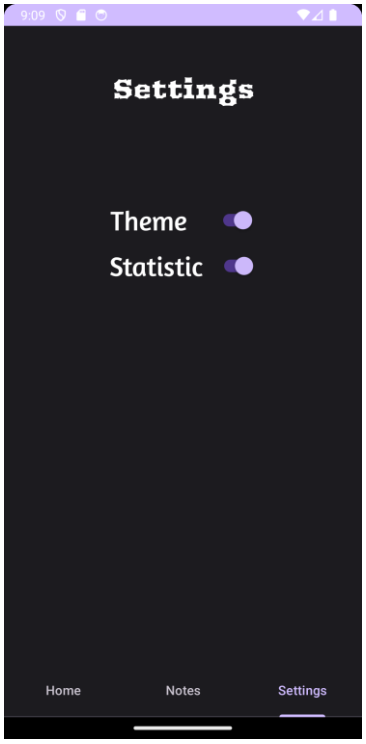


Рисунок 3.3 – Сторінка «Settings»

Після натискання кнопки додавання файлу, користувач може обрати файл з сховища пристрою (Рисунок 3.4). Кастомізувати назву, опис і швидкість метроному та обрати чи хоче він зберігати його в бібліотеку (Рисунок 3.5).

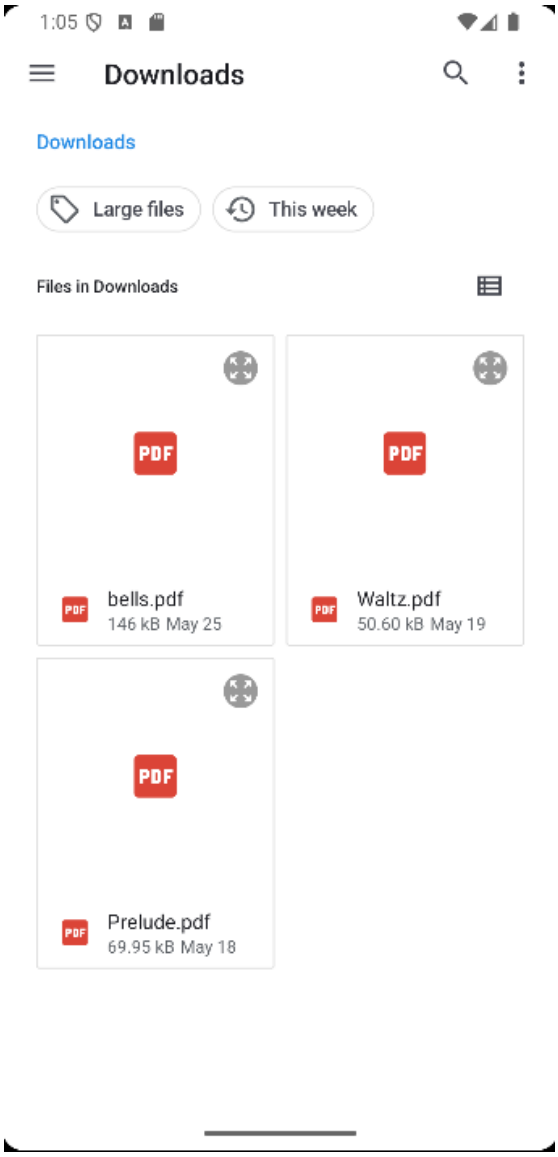


Рисунок 3.4 – Обирання файлу

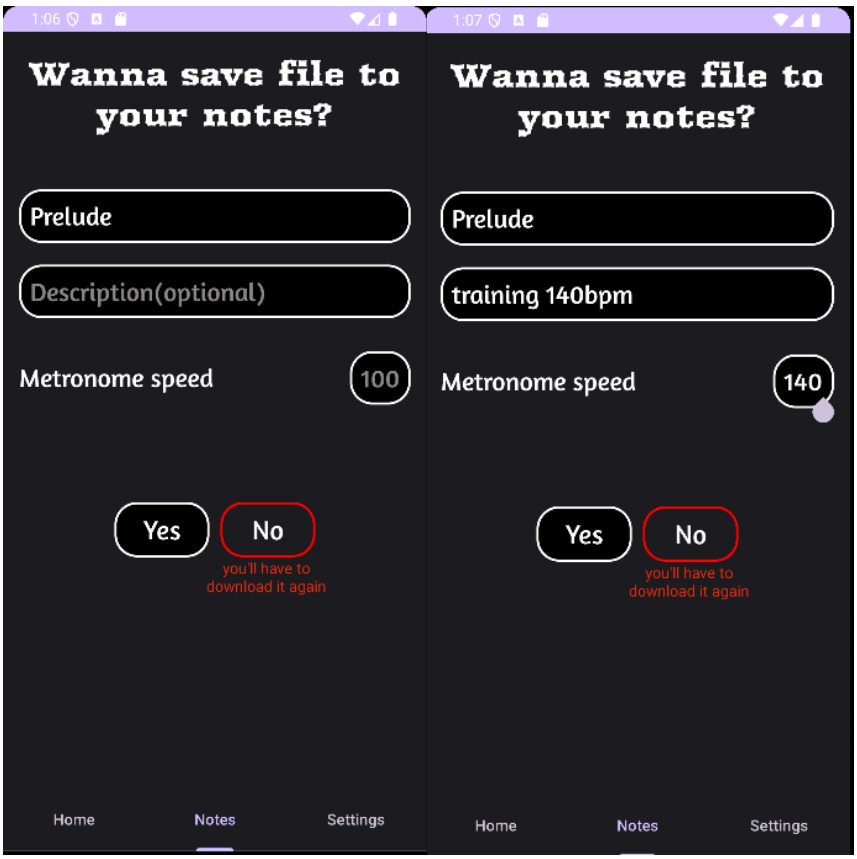


Рисунок 3.5 – Кастомізація і вибор збереження

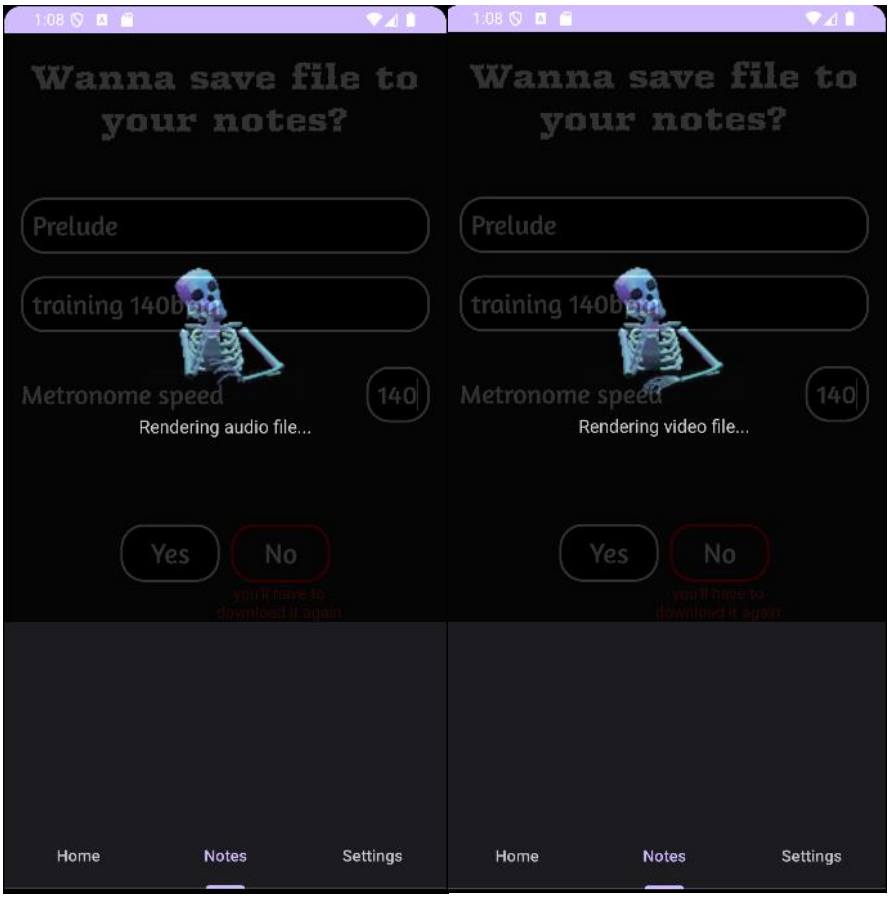


Рисунок 3.6 – Очікування обробки

Після обробки файлу користувач може бачити відео з нотами та має можливість прослухати композицію (Рисунок 3.7-3.8).

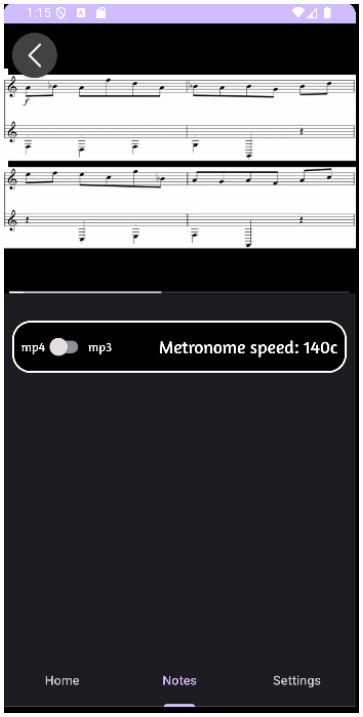


Рисунок 3.7 – Відеоряд з обробленими нотами

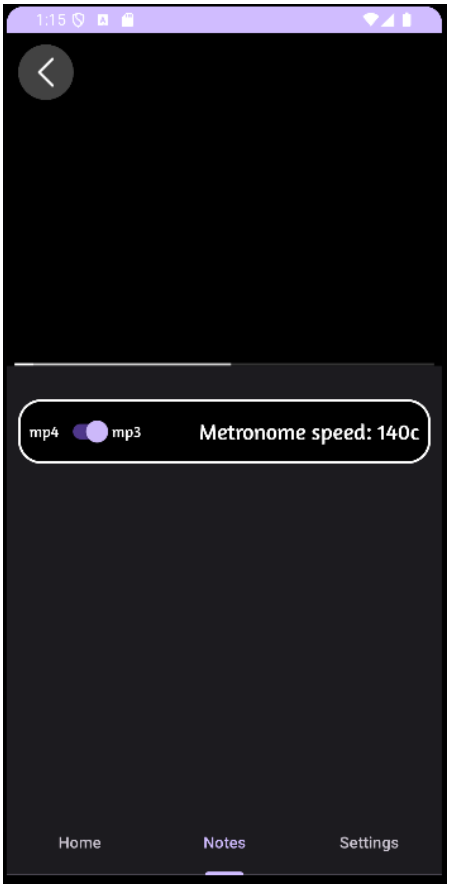


Рисунок 3.8 – Аудіоряд з обробленими нотами

Після виходу з програвача на вкладку «Notes» користувач має можливість відсортувати композиції за датою, іменем або описом чи виконати пошук за літерою в назві чи описі (Рисунок 3.9-3.10).

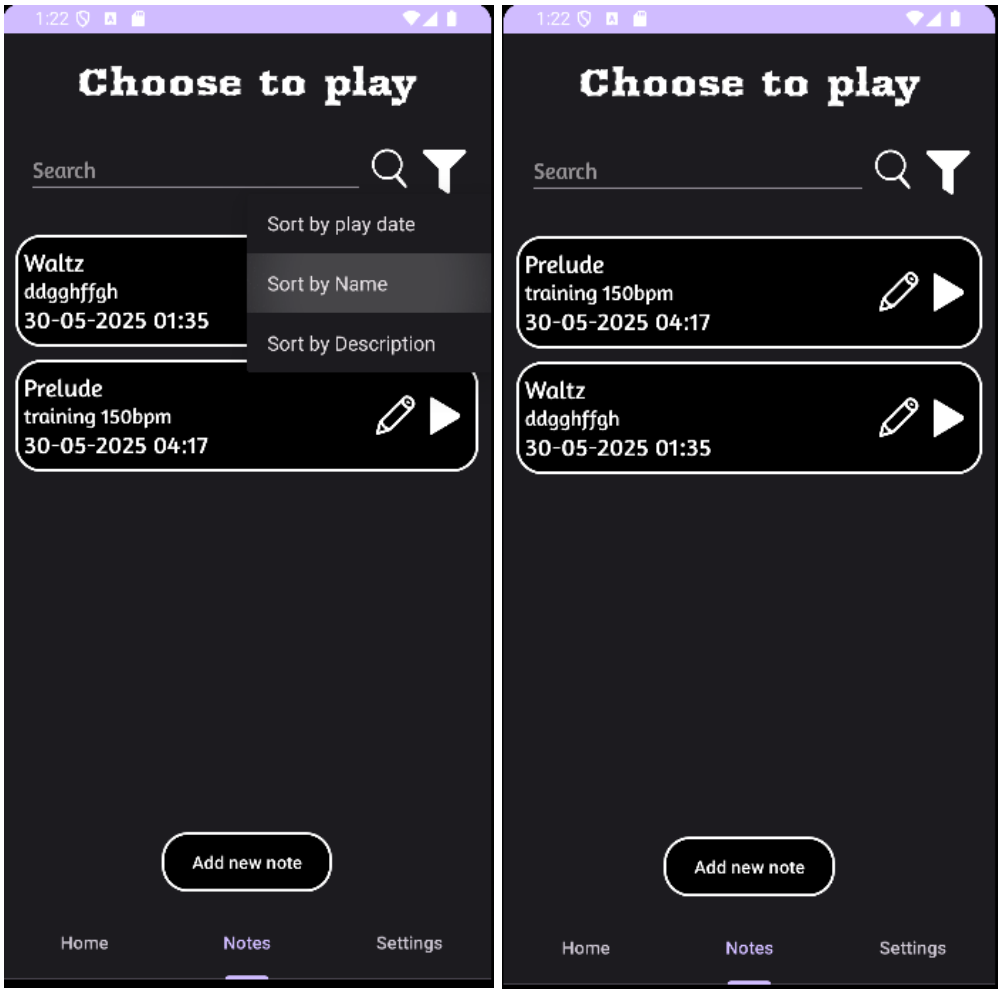


Рисунок 3.9 – Сортування композицій

Рисунок 3.10 – Пошук по літері

На сторінці «Settings» користувач має можливість змінити тему застосунку або видимість статистики (Рисунок 3.11-3.12).

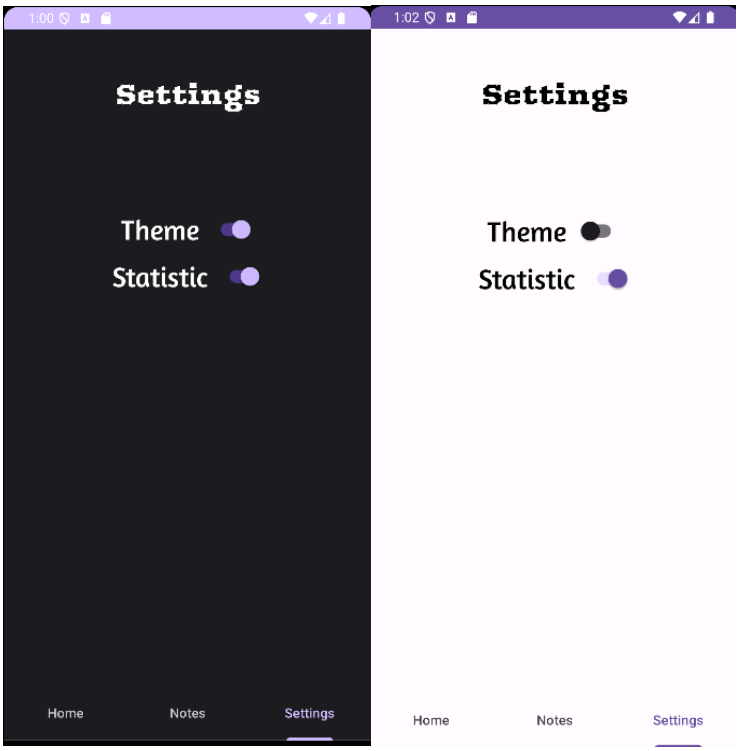


Рисунок 3.11 – Зміна на світлу тему додатку

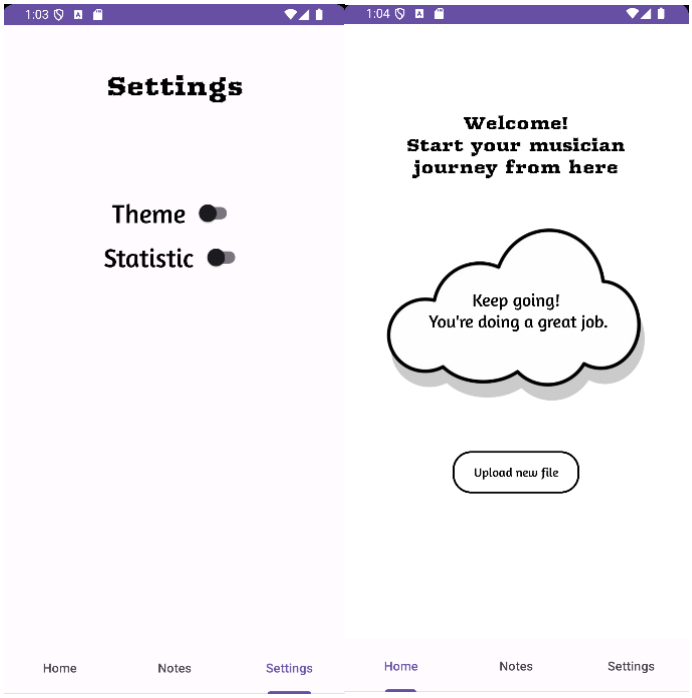


Рисунок 3.12 – Відключення статистики на головному екрані

У розділі редагування файлу користувач може змінити назву, опис та швидкість метроному (Рисунок 3.13-3.14) та видалити файл (Рисунок 3.15).

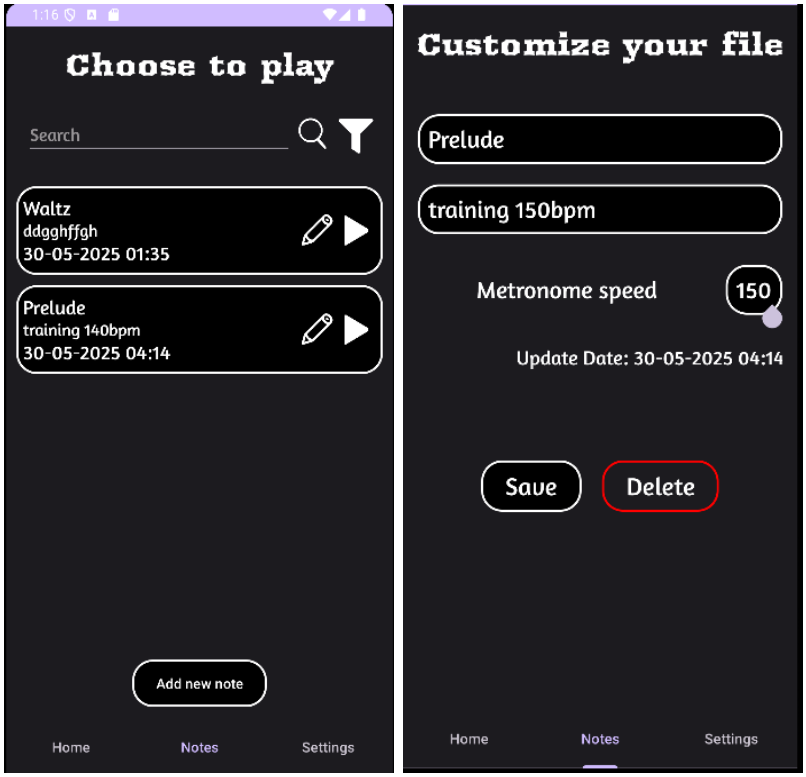


Рисунок 3.13 – Змінюємо опис та швидкість метроноум файлу

Рисунок 3.14 – Бібліотека з оновленим файлом

Рисунок 3.15 – Бібліотека з видаленим файлом «Prelude»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“___” _____ 2025 р.

МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ РОБОТИ З МУЗИЧНИМИ НОТАМИ

Графічний матеріал

КПІ.ІП-1132.045440.06.99

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Світлана ПОПЕРЕШНЯК

Нормоконтроль:

_____ Катерина ЛІЩУК

Виконавець:

_____ Вікторія ФУКС

Київ – 2025

КПІ.ІП-1132.045440.06.99.ССВ

					КПІ.ІП-1132.045440.06.99.ССВ	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна варіантів використань	Літера / Маса / Масштаб
Розробив		Фукс В.І				
Перевірив		Поперешняк С.В				Аркуш / Аркушів
Т. контр.						
Н. контр.		Ліщук К.І.			Мобільний застосунок для роботи з музичними нотами	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-11
Затвердив		Жаріков Е.В.				

КПІ.ІП-1132.045440.06.99.ССМ

					КПІ.ІП-1132.045440.06.99.ССМ			
						Літера	Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна компонентів програмного забезпечення			
Розробив		Фукс В.І				Аркуш	Аркушів	
Перевірив		Попершняк С.В						
Т. контр.								
Н. контр.		Ліщук К.І.			Мобільний застосунок для роботи з музичними нотами	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-11		
Затвердив		Жаріков Е.В.						

КПІ.ІП-1132.045440.06.99.ССМ

					КПІ.ІП-1132.045440.06.99.ССМ	
Зм.	Арк.	№ документа	Підпис	Дата	Архітектура програмного забезпечення	Літера / Маса / Масштаб
Розробив		Фукс В.І				
Перевірив		Попершняк С.В				Аркуш / Аркушів
Т. контр.						
Н. контр.		Ліщук К.І.			Мобільний застосунок для роботи з музичними нотами	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-11
Затвердив		Жаріков Е.В.				

КПІ.ІП-1132.045440.06.99.КЕ

					КПІ.ІП-1132.045440.06.99.КЕ	
Зм.	Арк.	№ документа	Підпис	Дата		
Розробив		Фукс В.І			Креслення вигляду екранних форм	Літера / Маса / Масштаб
Перевірив		Поперешняк С.В				
Т. контр.						Аркуш / Аркушів
Н. контр.		Ліщук К.І.			Мобільний застосунок для роботи з музичними нотами	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-11
Затвердив		Жаріков Е.В.				