

Міністерство освіти та науки України  
Національний університет «Львівська політехніка»  
Кафедра систем автоматизованого проектування



### **ПОЯСНЮВАЛЬНА ЗАПИСКА**

до курсової роботи з навчальної дисципліни  
«Розроблення кросплатформених додатків (Python)»  
на тему «Розробка музичної платформи з функцією музичного пошуку за голосом»

Виконав:  
ст. гр. ПП-24 Баб'юк Олександр

Перевірила:  
доцент каф. САП Стефанович Т. О.

## Зміст

Розділ 1. Проблемна ситуація, зацікавлені сторони та мета роботи.....	3
1.1. Опис проблемної ситуації .....	3
1.2. Аналіз цілей та потреб зацікавлених сторін .....	3
1.3. Формулювання мети роботи .....	4
Розділ 2. Огляд літератури за темою роботи.....	5
2.1 Music Identification Using Convolutional Neural Networks. ....	5
2.2.A Survey on Music Recommendation Systems. ....	5
2.3.Shazam: An Industrial-Strength Audio Search Algorithm .....	5
2.4.Music Recognition and Query-by-Humming .....	5
2.5.Deep Learning Techniques for Music Generation - A Survey .....	5
Розділ 3. Опис класів, методів та атрибутів .....	7
3.1. Діаграма класів.....	7
3.2. Опис класів, методів та атрибутів .....	7
Розділ 4. Опис програмної реалізації .....	11
4.1. Інструкція користувача.....	11
1. Вступ.....	11
2.Встановлення та запуск.....	11
3. Основні функції. ....	11
4.2. Програма мовою Python .....	14
Файли .py .....	14
Файли KV.....	20
4.3. Тестування програми .....	29
1)Тестування входу.....	29
2)Реєстрація.....	29
3)Головна форма .....	31
4)Playlist1.....	31
5)Прослуховування музики.....	31
Розділ 5. Обговорення результатів та висновки .....	32
5.1. Аналіз задоволення цілей та потреб.....	32
Додаток А. Самоаналіз роботи .....	34

## Розділ 1. Проблемна ситуація, зацікавлені сторони та мета роботи

### 1.1. Опис проблемної ситуації

Проблемна ситуація, яку ми розглядаємо полягає в обмеженому пошуку пісень, які ми десь чути на вулиці або в магазині, але не знаємо точної назви чи виконавця. Розглянемо цю ситуацію з точки зору об'єкта, суб'єкта, об'єктивної та суб'єктивної сторін.

**Об'єкт.** Проблемна ситуація впливає як і на кінцевих користувачів будь якої музичної платформи так і на простих людей, які не користуються технологіями. Також впливає на авторів пісень, оскільки користувачі просто не в змозі знайти пісню.

**Суб'єкт.** Проблемну ситуацію створюють засоби масової інформації, які використовують пісні, але не оголошують їх назву і виконавця.

**Об'єктивна сторона.** Проблема полягає в тому, що користувачі не можуть знайти пісню, яка їм сподобалася, через відсутність повної інформації. Це викликає незручності в пошуку і траті багато часу, можливо навіть розчарування через не знайдену пісню. Для музичних виконавців це втрата потенційних фанатів. Ситуація може виникати в різних місцях, таких як вулиці, магазини чи кафе, де грає музика, але немає можливості отримати інформацію про конкретну композицію. Також це може статися в ситуаціях, коли людина, яка почула пісню, не має можливості записати її назву або виконавця, щоб згодом знайти її.

**Суб'єктивна сторона.** Вина суб'єкта полягає в його рішенні не надавати інформацію про назву пісні або виконавця, що програється. Можливо, суб'єкт обирає таку стратегію з метою залучення уваги слухачів і підвищення інтересу до музичного контенту. Можливо, він розраховує на те, що це спонукатиме людей активніше використовувати музичні платформи для пошуку пісень.

### 1.2. Аналіз цілей та потреб зацікавлених сторін

Назва зацікавленої сторони	Опис цілей та потреб зацікавленої сторони. Роль зацікавленої сторони у проблемній ситуації
Користувачі музичних платформ	Це основна зацікавлена сторона, яка відчуває незручність через обмежені можливості пошуку пісень. Вони мають інтерес у зручному і швидкому способі знаходження музики, яка їм подобається.
Музичні платформи	Компанії, які надають послуги музичного стрімінгу, також зацікавлені в розв'язанні цієї проблеми, оскільки задоволені користувачі сприяють збільшенню аудиторії та прибутку.
Засоби масової інформації	Радіостанції, магазини та кафе, де програється музика без назв і виконавців, можуть мати власні мотиви, такі як збільшення часу уваги слухачів або привернення уваги до рекламних оголошень.
Музичні виконавці та лейбли	Автори та виконавці пісень можуть бути зацікавлені в тому, щоб їхні твори були легко доступні та знайдені слухачами, щоб збільшити свою популярність та прибуток.
Розробники музичних додатків	Ці компанії або індивідуальні розробники мають інтерес у вдосконаленні функціоналу своїх додатків, щоб привернути більше користувачів та збільшити їхню популярність.

### **1.3. Формулювання мети роботи**

Мета роботи полягає у створенні системи пошуку музичних композицій за звуковим мотивом (голосом), яка буде забезпечувати користувачам можливість ідентифікувати невідомі пісні, які вони чули у магазині, кафе або на вулиці, і не знають їхньої точної назви чи виконавця. Це сприятиме полегшенню процесу пошуку музичного контенту та збільшить задоволення користувачів від користування музичними платформами. Під час реалізації цієї мети, система буде здатна точно реагувати на звукові мотиви, використовуючи сучасні методи обробки сигналів та штучного інтелекту. Результативність системи буде вимірюватися кількістю успішно знайдених композицій за допомогою звукового пошуку в порівнянні з загальною кількістю введених користувачами мотивів. Така система матиме значення для користувачів музичних платформ, підвищивши їхнє задоволення від користування сервісом та сприяючи розвитку музичної індустрії.

## Розділ 2. Огляд літератури за темою роботи

### 2.1 Music Identification Using Convolutional Neural Networks.

Стаття присвячена застосуванню згорткових нейронних мереж для ідентифікації музичних композицій. Автори використовують алгоритми обробки сигналів та нейронні мережі для визначення пісні за її звуковими ознаками. Основні недоліки: обмежена точність ідентифікації на рівні практичного застосування, особливо при складних акустичних умовах.

**Посилання:**<https://bibliotekanauki.pl/articles/88408.pdf>

### 2.2.A Survey on Music Recommendation Systems.

У цій роботі автори проаналізували різноманітні системи рекомендацій музики. Вони досліджують методи колаборативної фільтрації, контентно-базовані методи та гібридні підходи. Основний недолік: більшість методів рекомендацій базуються на відомостях про композиції, а не на їх акустичних характеристиках.

**Посилання:**[https://www.researchgate.net/publication/334521667\\_A\\_Survey\\_of\\_Music\\_Recommendation\\_Systems\\_with\\_a\\_Proposed\\_Music\\_Recommendation\\_System](https://www.researchgate.net/publication/334521667_A_Survey_of_Music_Recommendation_Systems_with_a_Proposed_Music_Recommendation_System)

### 2.3.Shazam: An Industrial-Strength Audio Search Algorithm

Ця стаття описує алгоритм, який використовується в популярному додатку Shazam для ідентифікації музики. Вона включає в себе алгоритми обробки сигналів, хешування та шаблонний збір. Основний недолік: обмежена точність на старих або менш відомих записах.

**Посилання:**<https://github.com/leonardltk/Shazam-An-Industrial-Strength-Audio-Search-Algorithm->

### 2.4.Music Recognition and Query-by-Humming

В цій роботі автори досліджують методи визначення музичних композицій за допомогою звукових записів, створених користувачами (query-by-humming). Основний недолік: обмежена точність при використанні коротких та неточних звукових записів.

**Посилання:**[https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1895&context=etd\\_projects](https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1895&context=etd_projects)

### 2.5.Deep Learning Techniques for Music Generation - A Survey

У цій статті автори досліджують застосування глибокого навчання для генерації музики. Вони аналізують різноманітні моделі генерації музики.

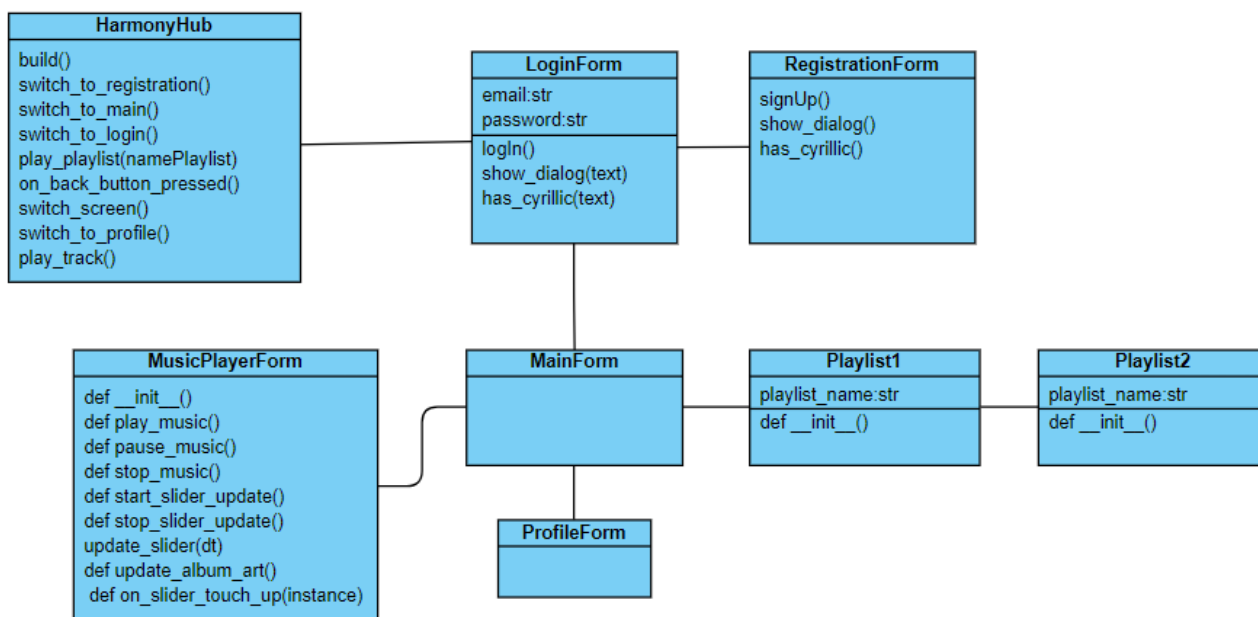
**Посилання:**<https://arxiv.org/abs/1709.01620>

### Короткий опис аналогів

№	Короткий опис аналогів	Виявлені недоліки
1.SoundHound	SoundHound пропонує функцію ідентифікації пісень за допомогою алгоритмів аудіо-пошуку. Відомий своєю здатністю розпізнавати пісні, які виконує користувач, та відповідати на питання про музику	Потребує стабільного інтернет-з'єднання для роботи, а також може не завжди точно визначити пісню, особливо якщо запис нечіткий або має шум.
2.Shazam	Shazam виявляє акустичні сліди пісні та порівнює їх з великою базою даних музичних записів.	Можливість некоректного визначення пісні у випадках, коли звуковий запис нечіткий або перекривається іншими звуками.
3.Musixmatch	Це додаток, який дозволяє користувачам шукати музичні композиції за текстом пісні. Він також надає текст пісні на додаток до інформації про виконавця та назву.	Не забезпечує можливості пошуку за звуковим мотивом, тому не підходить для визначення пісні, яку не можна знайти за словами.
4.Midomi	Цей додаток дозволяє користувачам шукати музику за допомогою співвіднесення вокалу або гуманізації музичних інструментів.	Може бути менш точним у порівнянні з іншими аналогами, особливо якщо запис зроблений в умовах великого шуму або перекривається іншими звуками.
5. АНА Music	Це розширення для браузера, яке дозволяє користувачам виявляти музику, яка грає у веб-середовищі, шляхом натискання на кнопку.	Може бути обмеженим у функціональності порівняно з іншими додатками, а також може бути не таким точним в порівнянні зі спеціалізованими аудіо-пошуковими системами.

## Розділ 3. Опис класів, методів та атрибутів

### 3.1. Діаграма класів



### 3.2. Опис класів, методів та атрибутів

Клас	Імена атрибутів та методів	Опис атрибутів та методів
HarmonyHub	build()	Створює та ініціалізує менеджер екранів (ScreenManager), додаючи до нього всі необхідні форми (екрани). Створює екземпляр ScreenManager, додає до нього екземпляри екранів (LoginForm, RegistrationForm, MainForm, Playlist1, Playlist2, ProfileForm, MusicPlayerForm) та повертає цей екземпляр для відображення.
	switch_to_registration()	Перемикає поточний екран на екран реєстрації. Встановлює значення атрибута current об'єкта root на 'registration', що змінює активний екран на екран реєстрації.
	switch_to_main()	Перемикає поточний екран на головний екран. Встановлює значення атрибута current об'єкта root на 'main', що змінює активний екран на головний екран.
	switch_to_login()	Перемикає поточний екран на екран входу. Встановлює значення атрибута current об'єкта root на 'login', що змінює активний екран на екран входу.
	play_playlist(namePlaylist)	Перемикає поточний екран на вказаний плейлист. Встановлює значення атрибута current

		об'єкта root на значення параметра namePlaylist, що змінює активний екран на відповідний плейлист
	on_back_button_pressed()	Обробляє натискання кнопки "назад", перемикаючи поточний екран на головний екран. Встановлює значення атрибута current об'єкта root на 'main', що змінює активний екран на головний екран.
	switch_screen(name)	Перемикає поточний екран на вказаний екран. Встановлює значення атрибута current об'єкта root на значення параметра name, що змінює активний екран на вказаний екран.
	switch_to_profile()	Перемикає поточний екран на екран профілю. Встановлює значення атрибута current об'єкта root на 'profile', що змінює активний екран на екран профілю.
	play_track()	Перемикає поточний екран на екран профілю. Встановлює значення атрибута current об'єкта root на 'profile', що змінює активний екран на екран профілю.
LoginForm	email:str	Атрибут для зберігання введеної електронної пошти користувача.
	password:str	Атрибут для зберігання введеного пароля користувача.
	logIn()	Виконує аутентифікацію користувача. Отримує email і пароль з текстових полів, перевіряє наявність кирилических символів у введених даних за допомогою методу has_cyrillic. Якщо є кирилическі символи, викликає show_dialog з відповідним повідомленням. Якщо ні, виконує SQL-запит для перевірки користувача. Якщо користувача знайдено, викликає switch_to_main, якщо ні – show_dialog з повідомленням про помилку. Комітить зміни в базі даних.
	show_dialog(text):	Відображає діалогове вікно з повідомленням. Створює об'єкт MDDialog з текстом, додає кнопку "OK" для закриття діалогового вікна, відкриває діалог.
	has_cyrillic(text):	Перевіряє, чи містить рядок кирилическі символи. Використовує регулярний вираз для пошуку кирилических символів у рядку, повертає True або False.
RegistrationForm	signUp()	Реєструє нового користувача. Отримує дані з текстових полів, створює новий user_id на основі кількості записів у таблиці person. Перевіряє, чи містять поля кирилическі символи за допомогою методу has_cyrillic, і показує



		діалогове вікно з відповідним повідомленням, якщо так. Якщо паролі збігаються і всі необхідні поля заповнені, вставляє нового користувача в базу даних і комітить зміни. Якщо паролі не збігаються або деякі поля порожні, показує діалогове вікно з повідомленням про помилку.
	show_dialog(text)	Відображає діалогове вікно з повідомленням. Створює об'єкт MDDialog з текстом, додає кнопку "ОК" для закриття діалогового вікна, відкриває діалог.
	has_cyrillic(text)	Перевіряє, чи містить рядок кириличні символи. Використовує регулярний вираз для пошуку кирилических символів у рядку, повертає True або False.
MusicPlayerForm	music_file:str	Шлях до файлу з музикою, який буде відтворюватися.
	album_art:str	Шлях до файлу з обкладинкою альбому.
	slider_update_event: ClockEvent	Подія для оновлення значення слайдера.
	track_length:float	Довжина музичного треку, встановлена в секундах.
	__init__()	Ініціалізує об'єкт MusicPlayerForm. Ініціалізує модуль pygame.mixer та встановлює значення за замовчуванням для атрибутів.
	play_music()	Відтворює музику з вказаного файлу. Завантажує файл музики, відтворює його, встановлює довжину треку для слайдера та оновлює обкладинку альбому.
	pause_music()	Призупиняє відтворення музики.
	stop_music()	Зупиняє відтворення музики та скидає значення слайдера і обкладинку альбому.
	start_slider_update()	Запускає оновлення значення слайдера.
	stop_slider_update()	Зупиняє оновлення значення слайдера.
	update_slider(dt)	Оновлює значення слайдера на основі поточної позиції відтворення музики.
	update_album_art()	Оновлює зображення обкладинки альбому на екрані.
	on_slider_touch_up()	Обробник події відпускання пальця зі слайдера. Відтворює музику з нової позиції слайдера.
Playlist1	playlist_name:str	Назва плейлисту (за замовчуванням порожня рядок).
	__init__()	Конструктор класу Playlist1 приймає параметр playlist_name, який може бути використаний для встановлення назви плейлисту. При створенні екземпляру класу конструктор спочатку викликає конструктор батьківського класу

		Screen за допомогою ключового слова <code>super()</code> , а потім встановлює значення атрибуту <code>playlist_name</code> на основі переданого значення або використовує значення за замовчуванням ("My Playlist").
Playlist1	<code>playlist_name:str</code>	Назва плейлисту (за замовчуванням порожня рядок).
	<code>__init__()</code>	Конструктор класу Playlist1 приймає параметр <code>playlist_name</code> , який може бути використаний для встановлення назви плейлисту. При створенні екземпляру класу конструктор спочатку викликає конструктор батьківського класу Screen за допомогою ключового слова <code>super()</code> , а потім встановлює значення атрибуту <code>playlist_name</code> на основі переданого значення або використовує значення за замовчуванням ("My Playlist").

## Розділ 4. Опис програмної реалізації

### 4.1. Інструкція користувача

#### 1. Вступ.

##### 1.1. Короткий опис програми та її мети.

Програма спрямована на розробку системи пошуку музичних композицій за звуковим мотивом (голосом), яка дозволить користувачам ідентифікувати невідомі пісні, які вони чули у магазинах, кафе або на вулиці, але не знають їхньої точної назви чи виконавця. Мета програми - полегшити процес пошуку музичного контенту та підвищити задоволення користувачів від користування музичними платформами.

##### 1.2. Уточнення цільової аудиторії, для якої призначена інструкція.

Цільова аудиторія програми включає в себе:

- Користувачів музичних платформ, які шукають спрощений та ефективний спосіб знаходження музики.
- Музичні платформи, які зацікавлені в покращенні користувацького досвіду та залученні нових аудиторій.
- Засоби масової інформації, що транслюють музику, але не надають інформацію про назви пісень або виконавців.
- Музичні виконавці та лейбли, які зацікавлені в підвищенні своєї популярності та доступності своєї музики.
- Розробники музичних додатків, що прагнуть розширити функціонал своїх програм та привернути більше користувачів.

#### 2. Встановлення та запуск.

##### 2.1. Інструкції зі встановлення програмного забезпечення.

Викачати всі файли з репозиторію <https://github.com/SashaBabiuk/HarmonyHub> і викачати всі файли собі на комп'ютер.

##### 2.2. Кроки щодо запуску програми після встановлення.

Оскільки програма повністю залежить від бази даних, то без БД не запустити програму. Я пробував заховити БД, але не знайшов безплатних і хороших ресурсів, тому на жаль перед запуском програми треба налаштувати локально БД.

Для цього потрібно:

- 1) Встановити реляційну систему керування БД MariaDB.
- 2) Створити локальну БД (скрипт для створення буде на gitHub, під назвою HarmonyHub.sql )
- 3) Налаштувати файл конфігурації з БД (harmonyHubDBConnect)

```
user = "root"
password = "vLCXoX2qkN>qw8KP,kS}E>}qUcon9V!3"
host = "127.0.0.1"
port = 3307
database = "harm"
```

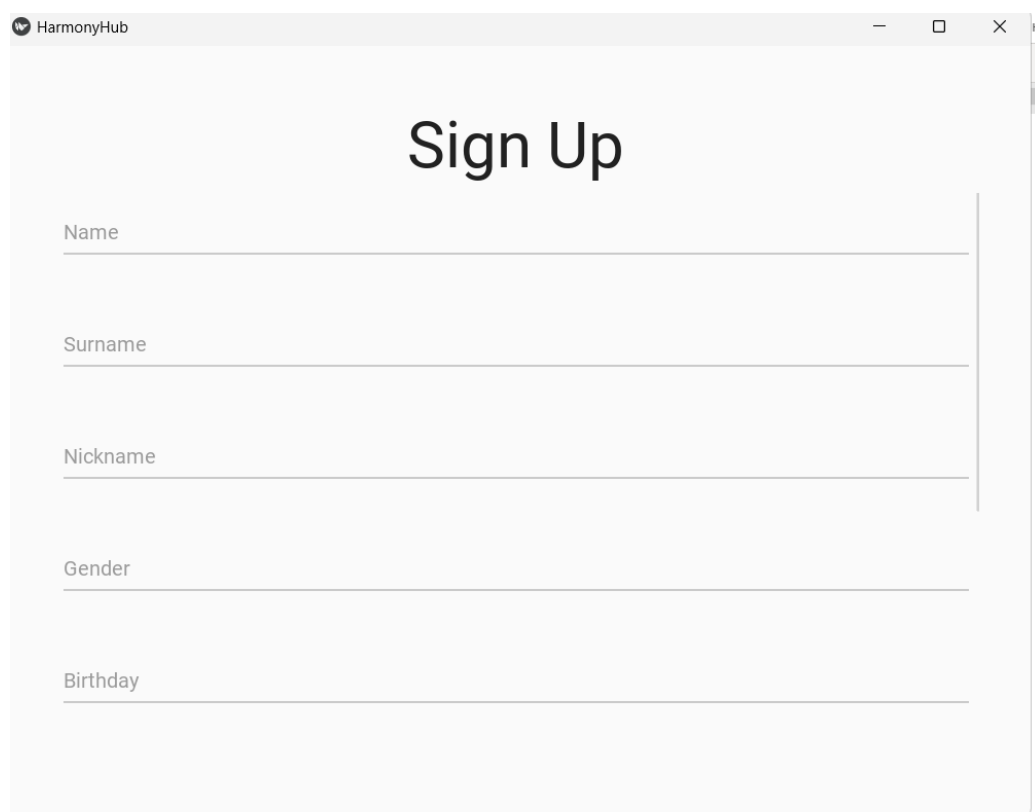
База даних була розрахована на утримання всіх звукових та графічних файлів, але через обмеження хосту я не став добавляти дані, тому, щоб протестувати програму до неї зразу йдуть готові графічна та аудіо файли (в папках images, music).

Після цього якщо все правильно налаштовано то відкриваємо файл main.cmd файл і насолоджуємося програмою

#### 3. Основні функції.

### 3.1. Детальний опис основних функцій програми.

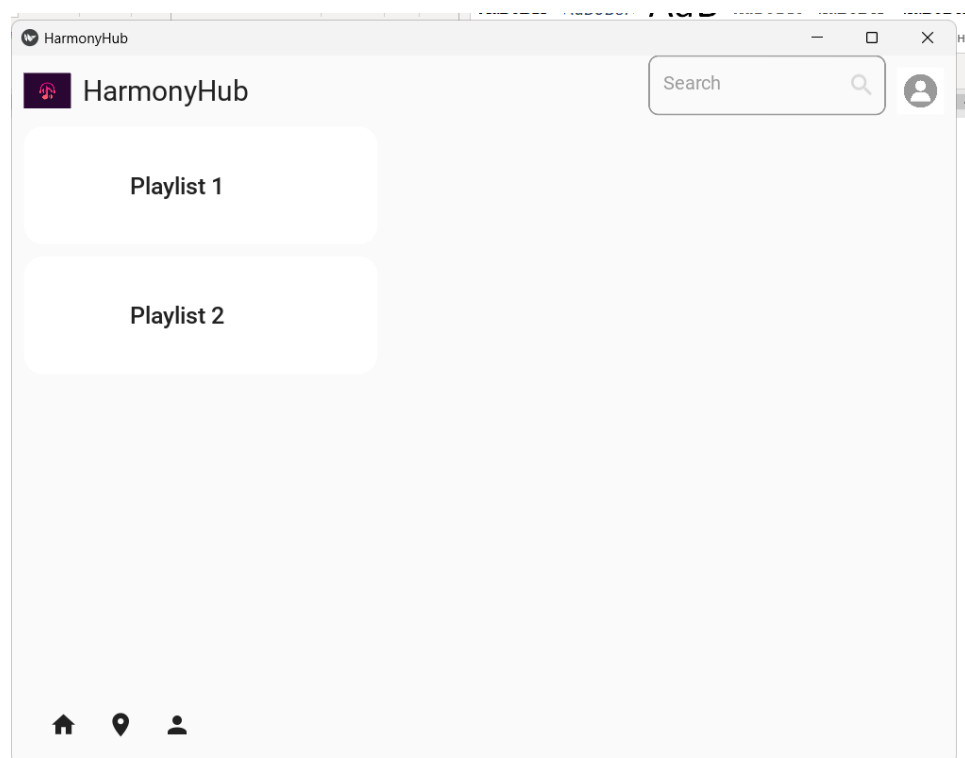
#### 1) Реєстрація

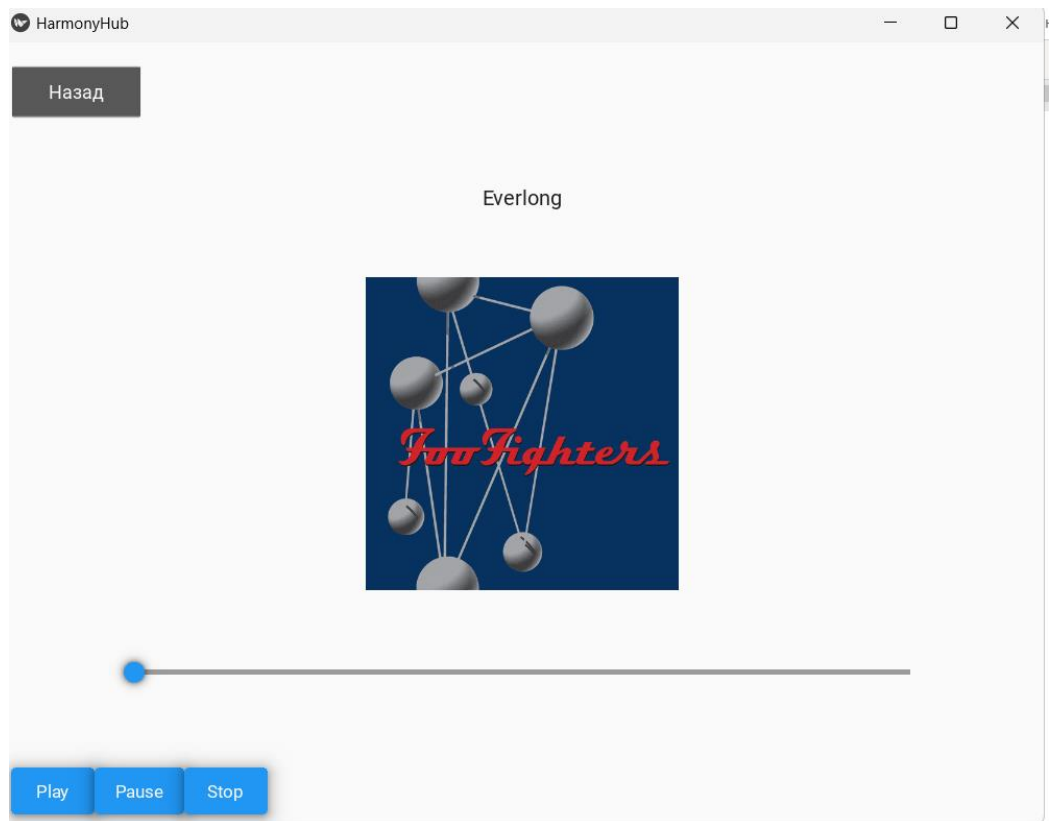
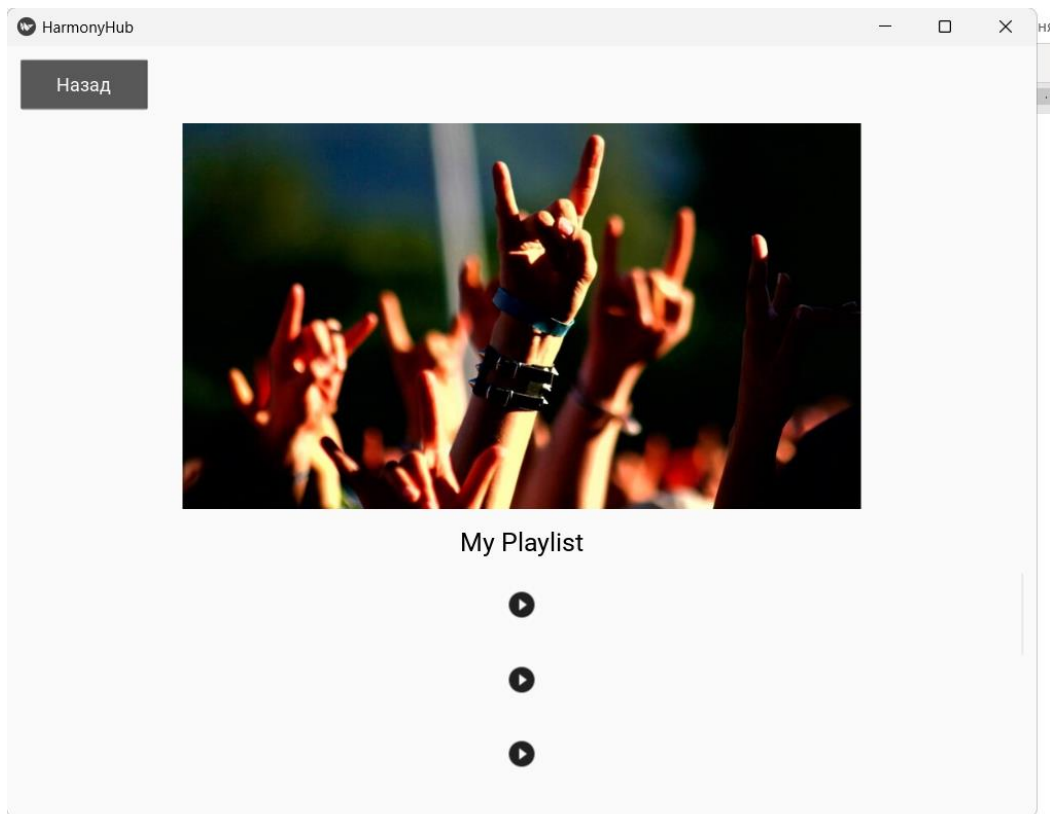


The screenshot shows a web browser window titled "HarmonyHub". The main heading is "Sign Up". Below the heading, there are five input fields arranged vertically, each with a label to its left: "Name", "Surname", "Nickname", "Gender", and "Birthday". A vertical line is positioned to the right of the "Name", "Surname", and "Nickname" fields. The browser's address bar shows a partial URL ending in "h:".

Для того, увійти в програму потрібно пройти реєстрацію після чого вже можна заходити в програму.

#### 2) Вибір плейлиста та прослуховування музики






### 3) Перегляд профілю

HarmonyHub

Profile



Username  
User123

Email  
user@example.com

Full Name  
John Doe

Bio  
This is my bio...

Save

Back

### 3.2. Покрокові інструкції з використання кожної функції.

#### 1) Реєстрація

Для реєстрації потрібно ввести свої данні і дату народження(дату обов'язково в форматі 04-10-2004) після чого придумати пароль і підтвердити його

#### 2) Вибір плейлиста

Після входу вас чекає основне меню в якому можна вибирати плейлист і пісню(для тестування доступно лише 1 плейлист і перші пісні в першому плейлисті)

Під час прослуховування пісні можна ставити її на паузу перемотувати і починати з початку

#### 3) Перегляд профілю

Тут можна переглядати і редагувати дані про себе.

## 4.2. Програма мовою Python

### Файли .py

#### 1) MusicPlayerForm.py

```
from kivy.uix.screenmanager import Screen
from kivy.clock import Clock
from kivy.lang import Builder
from kivymd.app import MDApp
import pygame

Builder.load_file('style/musicplayerform.kv')

class MusicPlayerForm(Screen):
    def __init__(self, **kwargs):
```

```

super().__init__(**kwargs)
pygame.mixer.init()
self.music_file = 'music/Foo Fighters - Everlong.mp3'
self.album_art = 'images/everlong.jpg'
self.slider_update_event = None
self.track_length = 0
def play_music(self):
    try:
        pygame.mixer.music.load(self.music_file)
        pygame.mixer.music.play()
        self.track_length = pygame.mixer.Sound(self.music_file).get_length()
        self.ids.slider.max = self.track_length
        self.start_slider_update()
        self.update_album_art()
    except pygame.error as e:
        print(f"Error playing music: {e}")

def pause_music(self):
    pygame.mixer.music.pause()
    self.stop_slider_update()

def stop_music(self):
    pygame.mixer.music.stop()
    self.stop_slider_update()
    self.ids.slider.value = 0
    self.ids.album_art.source = 'images/everlong.jpg'

def start_slider_update(self):
    if self.slider_update_event is None:
        self.slider_update_event = Clock.schedule_interval(self.update_slider,
1.0 / 60.0)

def stop_slider_update(self):
    if self.slider_update_event:
        self.slider_update_event.cancel()
        self.slider_update_event = None

def update_slider(self, dt):
    if pygame.mixer.music.get_busy():
        self.ids.slider.value = pygame.mixer.music.get_pos() / 1000
    else:
        self.stop_slider_update()
        self.ids.slider.value = 0

def update_album_art(self):
    self.ids.album_art.source = self.album_art

def on_slider_touch_up(self, instance):
    if instance is self.ids.slider:
        pygame.mixer.music.play(start=instance.value)
        self.start_slider_update()

```

## 2)playlist1.py

```

from kivymd.uix.screen import Screen
from kivy.lang import Builder

# Завантаження файлу стилів
Builder.load_file('style/playlist1.kv')

```

```
class Playlist1(Screen):
    playlist_name = "My Playlist"

    def __init__(self, playlist_name='', **kwargs):
        super(Playlist1, self).__init__(**kwargs)
        self.playlist_name = playlist_name
```

### 3)playlist2.py

```
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.screenmanager import Screen
from kivy.lang import Builder

Builder.load_file('style/playlist2.kv')

class Playlist2(Screen):
    playlist_name = "My Playlist 2"

    def __init__(self, playlist_name='', **kwargs):
        super(Playlist2, self).__init__(**kwargs)
        self.playlist_name = playlist_name
```

### 4)mainForm.py

```
from kivy.lang import Builder
from kivymd.uix.screen import Screen

# Завантаження файлу стилів
Builder.load_file('style/mainform.kv')

class MainForm(Screen):
    pass
```

### 5) profileForm.py

```
from kivy.uix.screenmanager import Screen
from kivy.lang import Builder
from loginForm import LoginForm
Builder.load_file('style/profileForm.kv')

class ProfileForm(Screen):
    pass
```

### 6)registrationForm.py

```
from kivymd.uix.dialog import MDDialog
from datetime import date
from harmonyHubDBConnect import cur, conn
from kivy.lang import Builder
from kivymd.uix.button import MDRectangleFlatButton
from kivymd.uix.screen import MDScreen
import re

Builder.load_file('style/registrationform.kv')

class RegistrationForm(MDScreen):
    def signUp(self):
        # Використання об'єкта курсора та підключення до бази даних
        cur.execute("SELECT COUNT(*) FROM person;")
        user_id = cur.fetchone()[0] + 1
        name_user = self.ids.name_user.text
```



```

        surname = self.ids.surname.text
        nickname = self.ids.nickname.text
        gender = self.ids.gender.text
        birthday = self.ids.birthday.text
        email = self.ids.email.text
        password = self.ids.password.text
        confirm_password = self.ids.confirm_password.text
        registration_date = date.today().strftime("%Y-%m-%d")

        if any(self.has_cyrillic(field) for field in
                [name_user, surname, nickname, gender, birthday, email, password,
confirm_password]):
            self.show_dialog("Please use only Latin characters in all fields.")
            return

        if password == confirm_password and all([name_user, surname, email,
password, confirm_password]):
            cur.execute("""
                INSERT INTO person(user_id, username, email, password,
registration date, type_id, settings_id, user_favorite_artist,
user_favorite_group)
                VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
                """, (user_id, nickname, email, password, registration_date,
1, user_id, 1, 1))
            conn.commit()
        else:
            self.show_dialog("Passwords do not match or some fields are empty.")

    def show_dialog(self, text):
        dialog = MDDialog(
            text=text,
            buttons=[
                MDRectangleFlatButton(
                    text="OK",
                    on_release=lambda x: dialog.dismiss()
                )
            ]
        )
        dialog.open()

    def has_cyrillic(self, text):
        return bool(re.search('[а-яА-Я]', text))

```

### 7) loginForm.py

```

from kivymd.uix.dialog import MDDialog
from harmonyHubDBConnect import cur, conn
from kivy.lang import Builder
from kivymd.app import MDApp
from kivymd.uix.button import MDRectangleFlatButton
from kivymd.uix.screen import Screen
import re

Builder.load_file('style/loginform.kv')

class LoginForm(Screen):

    email = ""
    password = ""

    def logIn(self):
        email = self.ids.email.text
        password = self.ids.password.text
        app = MDApp.get_running_app()

```

```

        if self.has_cyrillic(email) or self.has_cyrillic(password):
            self.show_dialog("Please use only Latin characters for email and
password.")
            return

        cur.execute("SELECT user_id FROM person WHERE email=? AND password=?",
(email, password,))
        result = cur.fetchone()

        if result:
            user_id = result[0]
            app.switch_to_main()
        else:
            self.show_dialog("User not found or incorrect password")

        conn.commit()

    def show_dialog(self, text):
        dialog = MDDialog(
            text=text,
            buttons=[
                MDRectangleFlatButton(
                    text="OK",
                    on_release=lambda x: dialog.dismiss()
                )
            ]
        )
        dialog.open()

    def has_cyrillic(self, text):
        return bool(re.search('[а-яА-Я]', text))

```

#### 8)harmonyHubDBConnect.py

```

import mariadb
from PIL import Image
import os
from io import BytesIO

user = "root"
password = "vLCXoX2qkN>qw8KP,kS}E>}qUcon9V!3"
host = "127.0.0.1"
port = 3307
database = "harm"

directory = 'images'
if not os.path.exists(directory):
    os.makedirs(directory)

try:
    conn = mariadb.connect(
        user=user,
        password=password,
        host=host,
        port=port,
        database=database
    )
except mariadb.Error as e:
    print(f"Error connecting to the database: {e}")

cur = conn.cursor()

def insert_image(image_data, image_type, image_size, image_ctgy, image_name):
    try:

```

```

        cur.execute("INSERT INTO image (image_type, image, image_size, image_ctgy,
image_name) VALUES (?, ?, ?, ?, ?)",
                    (image_type, mariadb.Binary(image_data), image_size,
image_ctgy, image_name))
        conn.commit()
        print("Image inserted successfully!")
    except mariadb.Error as e:
        print(f"Error inserting image into the database: {e}")

def read_image_from_database(image_name):
    try:
        cur.execute("SELECT image, image_name FROM image WHERE image_name=?",
(image_name,))
        image_data, image_name = cur.fetchone()
        image = Image.open(BytesIO(image_data))
        return image, image_name
    except mariadb.Error as e:
        print(f"Error reading image from the database: {e}")

def save_image(image, image_name):
    filename = os.path.join(directory, f"{image_name}.jpg")
    image.save(filename) # Збереження зображення у вигляді файлу
    print(f"Image saved as {filename}")

# Шлях до зображення
image_path = "images/default_photo.jpg"
image_type = "jpg"
image_size = "5"
image_ctgy = "photo"
image_name = "default_photo"

#with open(image_path, 'rb') as f:
#    image_data = f.read()

#insert_image(image_data, image_type, image_size, image_ctgy, image_name)
image, image_name = read_image_from_database(image_name)
save_image(image, image_name)

```

## 9)harmonyHub.py

```

from kivymd.app import MDApp
from kivy.uix.screenmanager import ScreenManager, Screen
from registrationForm import RegistrationForm
from loginForm import LoginForm
from mainForm import MainForm
from playlist1 import Playlist1
from playlist2 import Playlist2
from profileForm import ProfileForm
from musicPlayerForm import MusicPlayerForm

class HarmonyHub(MDApp):
    def build(self):
        sm = ScreenManager()

        sm.add_widget(LoginForm(name='login'))
        sm.add_widget(RegistrationForm(name='registration'))
        sm.add_widget(MainForm(name='main'))
        sm.add_widget(Playlist1(name='playlist1'))
        sm.add_widget(Playlist2(name='playlist2'))
        sm.add_widget(ProfileForm(name='profile'))
        sm.add_widget(MusicPlayerForm(name='music'))
        return sm

```

```

def switch_to_registration(self):
    self.root.current = 'registration'

def switch_to_main(self):
    self.root.current = 'main'

def switch_to_login(self):
    self.root.current = 'login'

def play_playlist(self, namePlaylist):
    self.root.current = namePlaylist

def on_back_button_pressed(self):
    self.root.current = 'main'

def switch_screen(self, name):
    self.root.current = name

def switch_to_profile(self):
    self.root.current = 'profile'

def play_track(self):
    self.root.current = 'music'

```

## 10) main.py

```

from harmonyHub import HarmonyHub

app = HarmonyHub()
app.run()

```

## Файли KV

### 1) playlist1.kv

```

<Playlist1>:
    BoxLayout:
        orientation: 'vertical'
        padding: dp(10)
        spacing: dp(10)

        BoxLayout:
            orientation: 'horizontal'
            size_hint_y: None
            height: dp(40)

            Button:
                text: 'Назад'
                size_hint: None, None
                size: dp(100), dp(40)
                on_press: app.on_back_button_pressed()

            Image:
                source: 'images/playlist.jpg'
                size_hint: 1, None
                height: dp(300)

            Label:
                text: root.playlist_name
                font_size: '20sp'
                size_hint: 1, None
                height: dp(30)
                color: 0, 0, 0, 1

```

```

ScrollView:
    GridLayout:
        cols: 1
        spacing: dp(10)
        size_hint_y: None
        height: self.minimum_height

        MDIconButton:
            icon: "play-circle"
            text: 'Track 1'
            size_hint: 1, None
            height: dp(30)
            on_release: app.play_track()

        MDIconButton:
            icon: "play-circle"
            text: 'Track 2'
            size_hint: 1, None
            height: dp(30)
            on_release: app.play_track(self.text)

        MDIconButton:
            icon: "play-circle"
            text: 'Track 3'
            size_hint: 1, None
            height: dp(30)
            on_release: app.play_track(self.text)

        MDIconButton:
            icon: "play-circle"
            text: 'Track 4'
            size_hint: 1, None
            height: dp(30)
            on_release: app.play_track(self.text)

        MDIconButton:
            icon: "play-circle"
            text: 'Track 5'
            size_hint: 1, None
            height: dp(30)
            on_release: app.play_track(self.text)

        MDIconButton:
            icon: "play-circle"
            text: 'Track 6'
            size_hint: 1, None
            height: dp(30)
            on_release: app.play_track(self.text)

        MDIconButton:
            icon: "play-circle"
            text: 'Track 7'
            size_hint: 1, None
            height: dp(30)
            on_release: app.play_track(self.text)

        MDIconButton:
            icon: "play-circle"
            text: 'Track 8'
            size_hint: 1, None
            height: dp(30)
            on_release: app.play_track(self.text)

```

```

MDIconButton:
    icon: "play-circle"
    text: 'Track 9'
    size_hint: 1, None
    height: dp(30)
    on_release: app.play_track(self.text)

```

## 2) playlist2.kv

```

<Playlist2>:
    BoxLayout:
        orientation: 'vertical'
        padding: dp(10)
        spacing: dp(10)

        BoxLayout:
            orientation: 'horizontal'
            size_hint_y: None
            height: dp(40)

            Button:
                text: 'Назад'
                size_hint: None, None
                size: dp(100), dp(40)
                on_press: app.on_back_button_pressed()

            Image:
                source: 'images/playlist.jpg'
                size_hint: 1, None
                height: dp(300)

            Label:
                text: root.playlist_name
                font_size: '20sp'
                size_hint: 1, None
                height: dp(30)
                color: 0, 0, 0, 1

        ScrollView:
            GridLayout:
                cols: 1
                spacing: dp(10)
                size_hint_y: None
                height: self.minimum_height

                Label:
                    text: 'Track 1'
                    size_hint: 1, None
                    height: dp(30)

                Label:
                    text: 'Track 2'
                    size_hint: 1, None
                    height: dp(30)

                Label:
                    text: 'Track 3'
                    size_hint: 1, None
                    height: dp(30)

                Label:
                    text: 'Track 4'
                    size_hint: 1, None
                    height: dp(30)

```

```

Label:
    text: 'Track 5'
    size_hint: 1, None
    height: dp(30)

Label:
    text: 'Track 6'
    size_hint: 1, None
    height: dp(30)

Label:
    text: 'Track 7'
    size_hint: 1, None
    height: dp(30)

Label:
    text: 'Track 8'
    size_hint: 1, None
    height: dp(30)

Label:
    text: 'Track 9'
    size_hint: 1, None
    height: dp(30)

```

### 3) mainForm.kv

```

<MainForm>:
    name: 'main'
    BoxLayout:
        orientation: 'vertical'

    BoxLayout:
        size_hint_y: None
        height: dp(50)
        padding: [dp(10), dp(10), dp(10), 0]

    Image:
        source: 'images/logo.jpg'
        size_hint: None, None
        size: dp(40), dp(40)
        keep_ratio: True

    Widget:
        size_hint_x: None
        width: dp(10)

    MDLabel:
        text: 'HarmonyHub'
        font_style: 'H5'
        halign: 'left'
        valign: 'middle'

    Widget:
        size_hint_x: 1

    MDTextField:
        hint_text: 'Search'
        mode: 'rectangle'
        size_hint: None, None
        size: dp(200), dp(40)
        icon_right: 'magnify'

    Widget:
        size_hint_x: None

```

```

        width: dp(10)

    Image:
        source: 'images/default_photo.jpg'
        size_hint: None, None
        size: dp(40), dp(40)
        keep_ratio: True
        radius: [20,]

# Центр з плейлістами
ScrollView:
    GridLayout:
        cols: 1
        size_hint_y: None
        height: self.minimum_height
        padding: dp(10)
        spacing: dp(10)

        MDCard:
            orientation: 'vertical'
            size_hint: None, None
            size: dp(300), dp(100)
            padding: dp(10)
            on_release: app.play_playlist('playlist1')

            BoxLayout:
                Image:
                    source: 'images/playlist1.jpg'
                    size_hint: None, None
                    size: dp(80), dp(80)
                    keep_ratio: True

                MDLabel:
                    text: 'Playlist 1'
                    font_style: 'H6'
                    halign: 'left'
                    valign: 'middle'

        MDCard:
            orientation: 'vertical'
            size_hint: None, None
            size: dp(300), dp(100)
            padding: dp(10)
            on_release: app.play_playlist('playlist2')

            BoxLayout:
                Image:
                    source: 'images/playlist2.jpg'
                    size_hint: None, None
                    size: dp(80), dp(80)
                    keep_ratio: True

                MDLabel:
                    text: 'Playlist 2'
                    font_style: 'H6'
                    halign: 'left'
                    valign: 'middle'

BoxLayout:
    orientation: 'horizontal'
    size_hint_y: None
    height: dp(56)
    padding: dp(10)

```



```

Widget:
    size_hint_x: None
    width: dp(10)

MDIconButton:
    icon: 'home'
    on_release: app.switch_screen('main')

MDIconButton:
    icon: 'map-marker'
    on_release: app.switch_screen('navigation')

MDIconButton:
    icon: 'account'
    on_release: app.switch_screen('profile')

Widget:
    size_hint_x: None
    width: dp(10)

```

#### 4) profileForm.kv

```

<ProfileForm@Screen>:
    username: "User123"
    email: "user@example.com"
    fullname: "John Doe"
    bio: "This is my bio..."
    avatar: "images/default_photo.jpg"

MDBoxLayout:
    orientation: 'vertical'
    padding: 10
    spacing: 10

MDLabel:
    text: "Profile"
    halign: 'center'
    theme_text_color: "Primary"

Image:
    source: 'images/default_photo.jpg'
    size_hint: (None, None)
    size: (150, 150)
    pos_hint: {'center_x': 0.5}
    allow_stretch: True
    keep_ratio: True

MDTextField:
    hint_text: "Username"
    text: root.username
    size_hint_x: None
    width: 300
    pos_hint: {'center_x': 0.5}

MDTextField:
    hint_text: "Email"
    text: root.email
    size_hint_x: None
    width: 300
    pos_hint: {'center_x': 0.5}

MDTextField:
    hint_text: "Full Name"
    text: root.fullname
    size_hint_x: None

```

```

        width: 300
        pos_hint: {'center_x': 0.5}

MDTextField:
    hint_text: "Bio"
    text: root.bio
    multiline: True
    size_hint_x: None
    width: 300
    pos_hint: {'center_x': 0.5}

MDRaisedButton:
    text: "Save"
    size_hint: (None, None)
    size: (150, 40)
    pos_hint: {'center_x': 0.5}

MDRaisedButton:
    text: "Back"
    size_hint: (None, None)
    size: (150, 40)
    pos_hint: {'center_x': 0.5}
    on_release: app.switch_to_main()

```

### 5) registrationform.kv

```

<RegistrationForm>:
    name: 'registration'
    name_user: name_user
    surname: surname
    nickname: nickname
    gender: gender
    birthday: birthday
    email: email
    password: password
    confirm_password: confirm_password

BoxLayout:
    orientation: 'vertical'
    padding: 50

    MDLabel:
        text: 'Sign Up'
        font_size: '50sp'
        halign: 'center'
        size_hint_y: None
        height: self.texture_size[1] + 20

    ScrollView:
        BoxLayout:
            id: form_layout
            orientation: 'vertical'
            padding: [10, 0, 10, 10]
            spacing: 30
            size_hint_y: None
            height: self.minimum_height

            MDTextField:
                id: name_user
                hint_text: 'Name'

            MDTextField:
                id: surname
                hint_text: 'Surname'

```

```

MDTextField:
    id: nickname
    hint_text: 'Nickname'

MDTextField:
    id: gender
    hint_text: 'Gender'

MDTextField:
    id: birthday
    hint_text: 'Birthday'

MDTextField:
    id: email
    hint_text: 'Email'

MDTextField:
    id: password
    hint_text: 'Password'
    password: True

MDTextField:
    id: confirm_password
    hint_text: 'Confirm Password'
    password: True

MDRaisedButton:
    text: 'Sign Up'
    on_release: root.signUp()

MDRaisedButton:
    text: 'Back to Login'
    on_release: app.switch_to_login()

```

## 6) loginform.kv

```

<LoginForm>:
    name: 'login'
    GridLayout:
        rows: 4
        padding: 50
        email: email
        password: password

    Image:
        source: 'images/music.jpg'

    MDLabel:
        text: 'Log In'
        font_size: '50sp'
        halign: 'center'

    BoxLayout:
        padding: [10, 0, 10, 10]
        spacing: 30
        orientation: 'vertical'

    MDTextField:
        id: email
        hint_text: 'Email'

    MDTextField:
        id: password
        hint_text: 'Password'
        password: True

```

```

BoxLayout:
    padding: [30, 0, 30, 30]

    MDRaisedButton:
        size_hint: 0.5, 0.5
        text: 'Log In'
        on_release: root.logIn()

    MDRaisedButton:
        size_hint: 0.5, 0.5
        text: 'Sign Up'
        on_release: app.switch_to_registration()

```

## 7)musicplayerform.kv

```

<MusicPlayerForm>:
    BoxLayout:
        orientation: 'vertical'
        padding: 10
        spacing: 10

        BoxLayout:
            size_hint_y: None
            height: dp(50)

            Button:
                text: 'Hazaа'
                size_hint: None, None
                size: dp(100), dp(40)
                on_press: app.switch_to_main()

        MDLabel:
            text: "Everlong"
            halign: 'center'
            theme_text_color: "Primary"

        Image:
            id: album_art
            source: 'images/everlong.jpg'
            size_hint: (None, None)
            size: (300, 300)
            pos_hint: {'center_x': 0.5}
            allow_stretch: True
            keep_ratio: True

        MDSlider:
            id: slider
            min: 0
            max: 100
            value: 0
            size_hint_x: 0.8
            pos_hint: {'center_x': 0.5}

        BoxLayout:
            size_hint_y: None
            height: dp(50)

            MDRaisedButton:
                text: "Play"
                on_release: root.play_music()

            MDRaisedButton:
                text: "Pause"

```

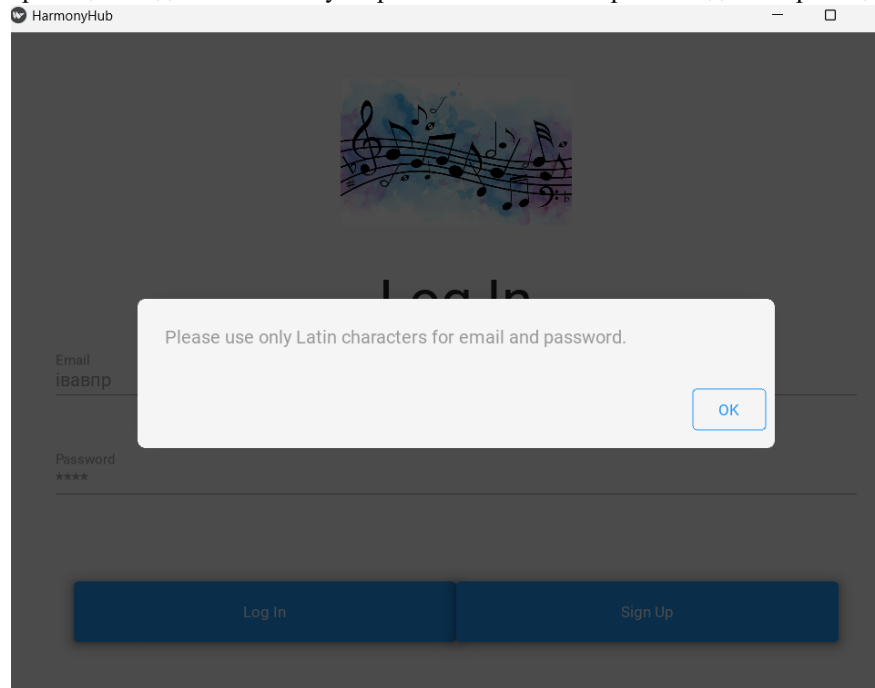
```
on_release: root.pause_music()

MDRaisedButton:
    text: "Stop"
    on_release: root.stop_music()
```

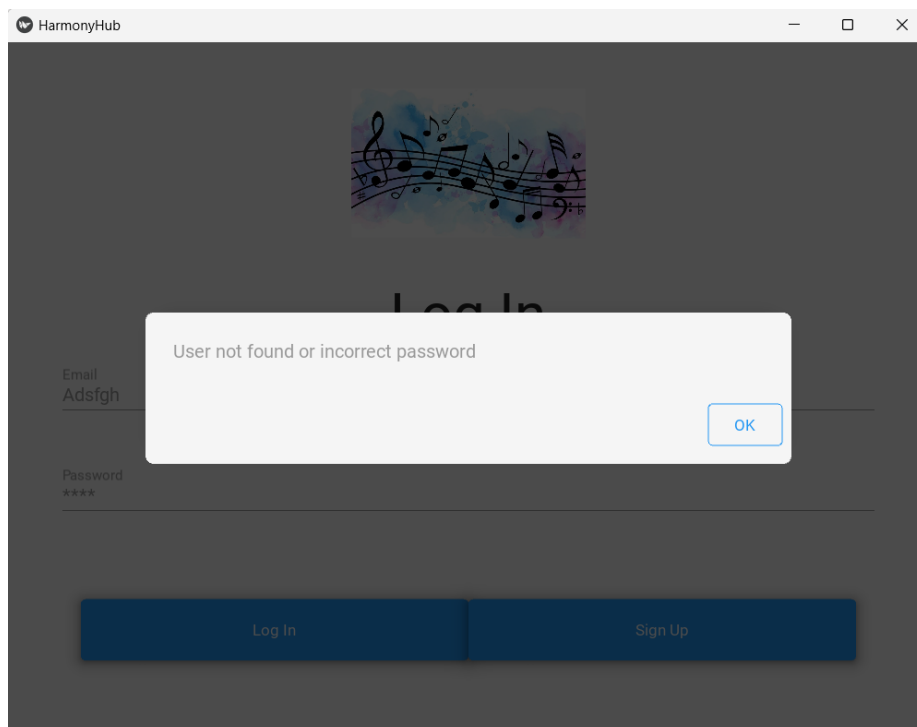
### 4.3. Тестування програми

#### 1)Тестування входу

В ході написання програми було виявлено, що дана бібліотека погано сприймає кирилицю і при спробі ввести логін користувача кирилицею видавало помилку.Вирішенням стала заборона вводити кирилицю.



При спробі ввести неіснуючого буде виведено діалогове вікно



#### 2)Ресстрація

При реєстрації, коли паролі не співпадають виводиться наступне вікно

HarmonyHub

# Sign Up

-----

Birthday  
04-10-2004

Email  
sashasbabuk82@gmail.com

Password  
\*\*\*

Confirm Password  
\*\*

Sign Up

Passwords do not match or some fields are empty.

OK

Якщо ж ввести в будь якому полі кирилицю виведеться наступне вікно

Birthday  
04-10-2004

Email  
sashasbabuk82@gmail.com

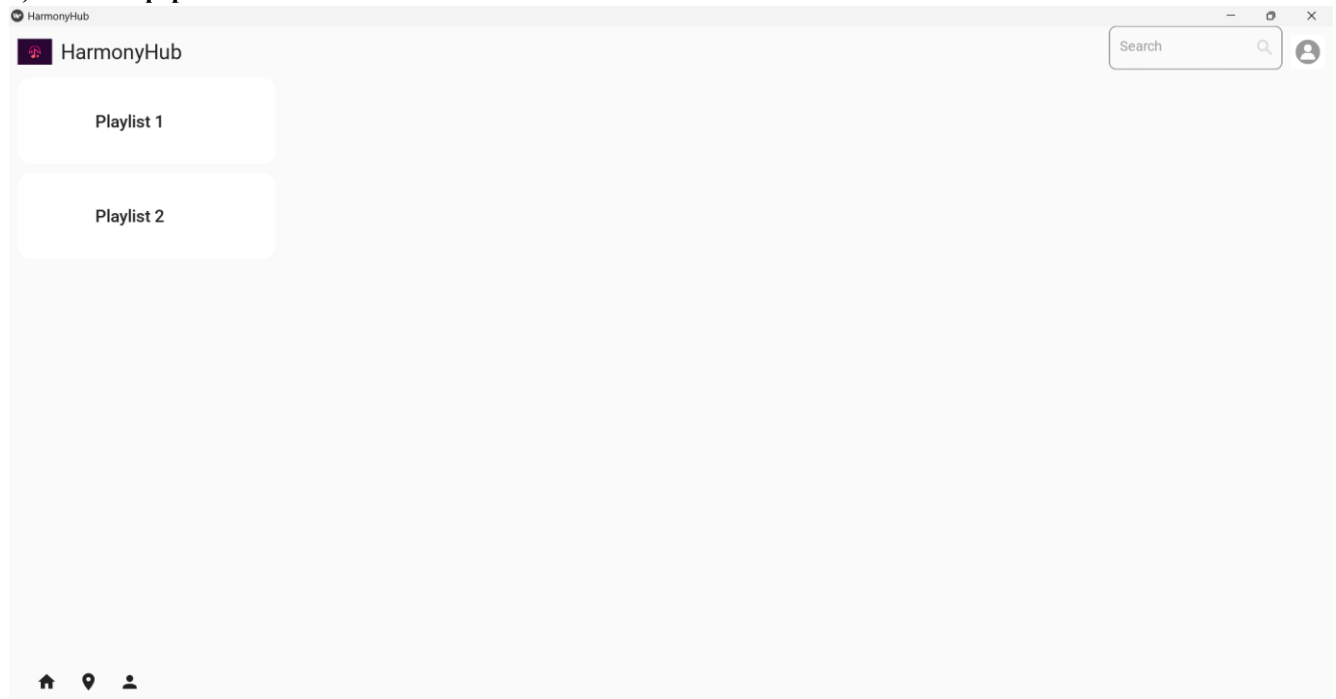
Password  
\*\*\*

Confirm Password  
\*\*\*

Please use only Latin characters in all fields.

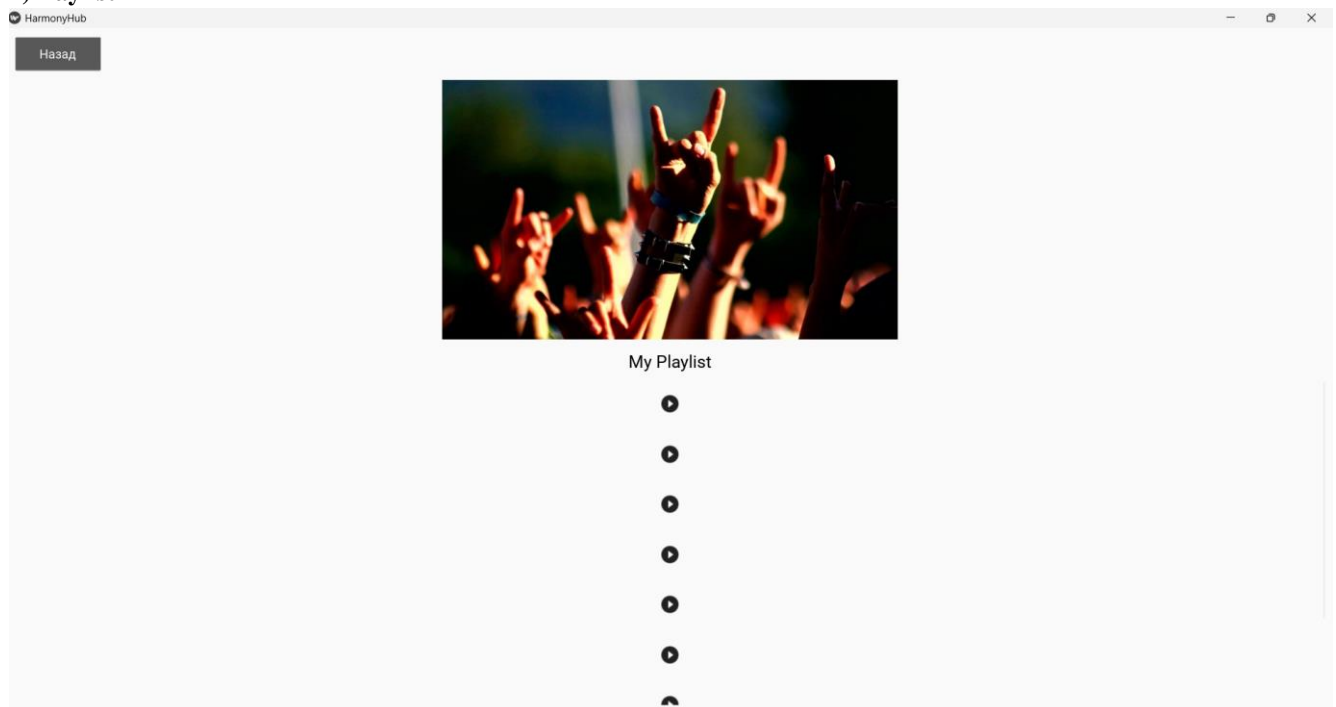
OK

### 3) Головна форма



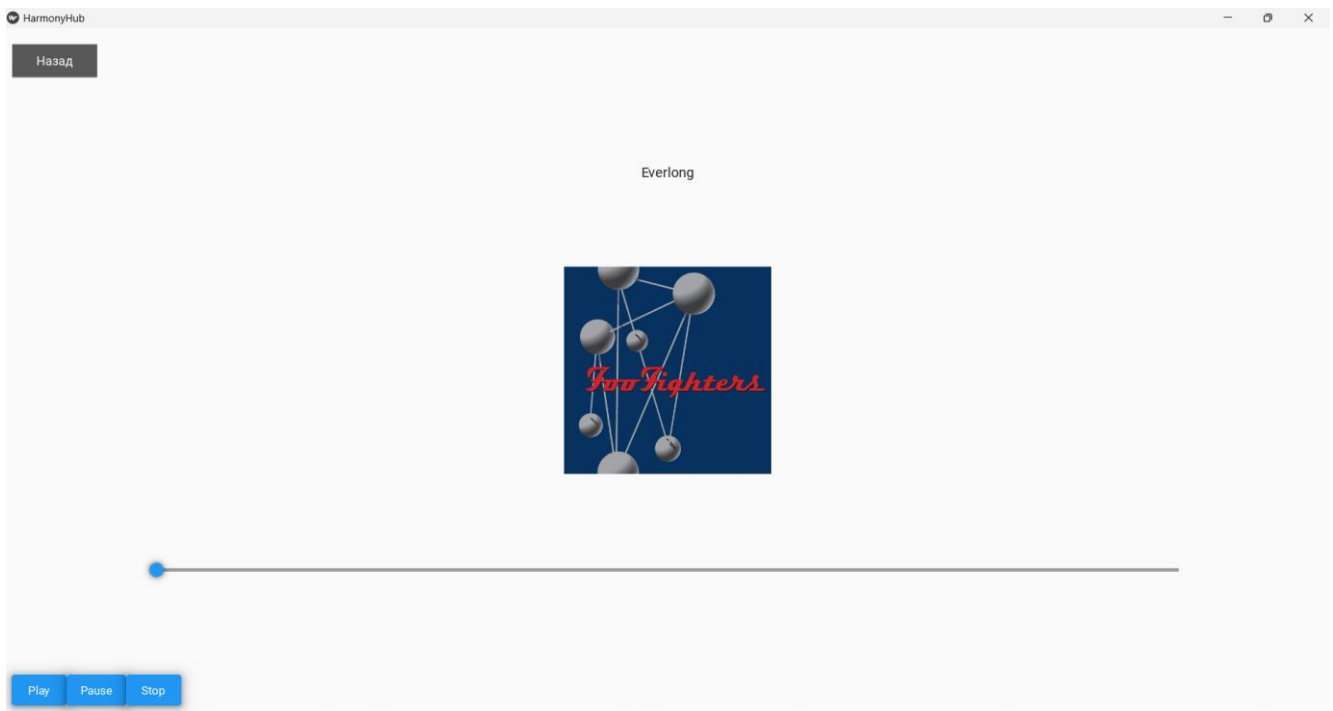
На головній формі працюють кнопки Playlist1, Playlist2, які перекидають нас на певні плейлисти. Також на нижній панелі працює кнопка, яка відкриває меню профілю.

### 4) Playlist1



На даній версії програми працює лише перший плейлист, якщо зайти в нього можна відкрити пісню для цього потрібно натиснути на першу кнопку. Кнопка назад вертає нас на головне меню.

### 5) Прослуховування музики



Прослуховуючи музику можна перемотувати її пересуваючи повзунок, також ставити на паузу і починати з початку, проте не можна регулювати гучність.

## Розділ 5. Обговорення результатів та висновки

### 5.1. Аналіз задоволення цілей та потреб

Назва зацікавленої сторони	Опис та оцінка задоволеності цілей та потреб зацікавленої сторони
Користувачі музичних платформ	Хоча система не забезпечує пошук за звуковим мотивом, вона все ж таки полегшує дозволяє прослуховувати музику. Оцінка: Помірна задоволеність
Музичні виконавці та лейбли	Виконавці та лейбли отримують вигоду від поліпшеного доступу користувачів до їх музики, навіть без функції пошуку за голосом. Оцінка: Помірна задоволеність.
Розробники музичних додатків	Розробники отримують покращену платформу для користувачів, що може залучити більше аудиторії, незважаючи на відсутність голосового пошуку. Оцінка: Помірна задоволеність.

### 5.2. SWOT-аналіз результатів роботи

Виконати SWOT-аналіз (S[trength] – сила, W[eak] – слабкість, O[pportunity] – можливість, T[hreat] – загроза). Аналіз ґрунтується на основі оцінки ступеня задоволеності зацікавлених сторін, які подано в п. 5.1.	
Зацікавлені сторони, які задіяні у проблемній ситуації БЕЗПОСЕРЕДНЬО	Зацікавлені сторони, які задіяні у проблемній ситуації ОПОСЕРЕДКОВАНО
Сильні сторони	Можливості
Прослуховування музики безкоштовно і в фоновому режимі. Повністю відкритий код. Примітивний і зручний інтерфейс для користувачів	Розширення функціоналу додатку у майбутньому шляхом додавання голосового пошуку. Можливість інтеграції з іншими музичними сервісами та платформами. Залучення нових користувачів через поліпшення існуючих функцій.



Слабкі сторони	Загрози
Відсутність функції пошуку за звуковим мотивом (голосом). Обмежені можливості пошуку музики. Недостатня інноваційність у порівнянні з конкурентами.	Втрата користувачів, які потребують функцію голосового пошуку. Зниження конкурентоспроможності порівняно з додатками, що мають пошук за голосом. Незадоволеність користувачів, що може призвести до негативних відгуків.

## Додаток А. Самоаналіз роботи

Структурний елемент пояснювальної записки	Максимальна оцінка	Самооцінка
1.1. Опис проблемної ситуації	5	5
1.2. Аналіз цілей та потреб зацікавлених сторін	5	4
1.3. Формулювання мети роботи	5	5
2. Огляд літератури за темою роботи	10	10
3.1. Діаграма класів	5	5
3.2. Опис класів, методів та атрибутів	5	5
4.1. Інструкція користувача	5	5
4.2. Програма мовою Python	5	5
4.3. Підходи до тестування	5	4
5.1. Аналіз задоволення цілей та потреб	5	4
5.2. SWOT-аналіз результатів роботи	5	4
А. Самоаналіз роботи	10	9
<b>Всього:</b>	<b>70</b>	<b>65</b>